COLING 2018

The 27th International Conference on Computational Linguistics

Proceedings of the Conference

August 20-26, 2018
Santa Fe, New Mexico, USA

ISBN 978-1-948087-50-6

Preface: General Chair

On behalf of the International Committee on Computational Linguistics, I am thrilled to welcome you all to the 27th International Conference on Computational Linguistics (COLING 2018) here in Santa Fe, New-Mexico. Many of you are too young to remember the last COLING that took place in North America: it was actually the Montreal 1998 joint COLING-ACL conference. And indeed, most of you are too young to remember the last COLING in the U.S.A. which (would you believe?) was held as far back as 1984, also as a joint COLING-ACL conference. Moreover, the only other COLING that took place in the U.S.A. was the very first one, chaired by David G. Hays back in 1965. It gave rise to the oldest publication you can currently read in the ACL anthology. Whatever the reasons for the rarity COLING in the USA, we are very proud to bring you the present one and our team has worked very hard to make it a memorable event.

Over the last 18 months, a tremendous amount of work has been carried out by the COLING 2018 organizing committee. Please join me in thanking Sergei Nirenburg (Local Organizer), Emily Bender and Leon Derczynski (Program Chairs), Tim Baldwin, Yoav Goldberg and Jin Jiang (Workshop Chairs), Donia Scott, Pascale Fung and Marilyn Walker (Tutorial Chairs), Dongyan Zhao (Demonstration Chair), Xiaodan Zhu and Zhiyuan Liu (Publications Chairs), Yuji Matsumoto and Hiroshi Noji (Publicity Chairs), Qian Chen and Christine Tang (Webmasters) and last but not least, Leo Wanner, Soo-Min Pantel, Satoshi Sekine and Le Sun (Sponsorship Chairs).

I hope that you will enjoy the special atmosphere and the spectacular scenery offered by Santa Fe, the convenience of our meeting site as well as the social events and excursions carefully arranged by our Local Organizing Committee.

Most importantly, I hope that you will love the content of our conference. I trust that the COLING difference will be unmistakable. While paying due tribute to the hottest methods and trends of the times, we will also offer you a more diversified intellectual menu. Our Program Chairs deserve highly special thanks for their truly remarkable work in setting up a program that comprises some 331 papers which, in addition to the ubiquitous type of scientific contribution these days ("NLP engineering experiment"), accommodates a variety of other worthwhile kinds of contributions, each with appropriate evaluation criteria. This emphasis on diversity is also visible in their choice of our four outstanding invited speakers: Fabiola Henri, James Pustejovsky, Min-Yen Kan and Hannah Rohde. Throughout their program development work, Emily and Leon have made sure to engage our community on all aspects of their decision-making through a lively blog. And as if this wasn't enough, they also found the energy to setup a highly successful mentoring program for authors. Our scientific program also includes 35 system demonstrations, seven enticing tutorials, and twelve different workshops.

I would like to thanks our generous COLING 2018 sponsors:

Amazon Alexa, Baidu, Disney Research, the Linguist List, Lenovo, Brandeis University, the University of Washington and the University of Colorado. And last but not least, I offer my sincere thanks to all those who have submitted papers and demos to COLING 2018 as well as to all those who have served as area chairs, reviewers or helped our organizing committee in any other way.

I wish you all a pleasant and fruitful conference.

Pierre Isabelle (National Research Council, Canada)
COLING 2018 General Chair

Preface: Program Chairs

COLING is the oldest Computational Linguistics conference, from its first instance fifty-three years ago in 1965, held in New York. This year marks its third visit to the USA, after the second in 1984 at Stanford. COLING remains the strongest conference in the field that is beyond the ACL, instead organised by the ICCL. The ICCL and COLING hold ideals of being a conference for all, originally to bring together both NATO and Soviet scientists without an overseeing US-based organization. These are values that we have constantly held in mind with our construction of COLING 2018, in Santa Fe, and we are honored to have had so much support in doing so.

As we began the process of constructing the program for COLING 2018, we started by identifying our goals. These were set on our PC blog in a post in August 2017. They are: (1) to create a program of high quality papers which represent diverse approaches to and applications of computational linguistics, written and presented by researchers from throughout our international community; (2) to facilitate thoughtful reviewing which is both informative to ACs (and to us as PC co-chairs), and helpful to authors; and (3) to ensure that the results published at COLING 2018 are as reproducible as possible.

To give a bit more detail on the first goal, by diverse approaches/applications, we mean that we aimed to attract (in the tradition of COLING):

- papers which develop linguistic insight as well as papers which deepen our understanding of how machine learning can be applied to NLP—and papers that do both!
- research on a broad variety of languages and genres
- many different types of research contributions (application papers, resource papers, methodology papers, position papers, reproduction papers. . .)

We had the challenge and the privilege of taking on this role at a time when our field is growing tremendously quickly. We seized the opportunity to advance the way our conferences work by trying new things and improving the experience from all sides. To this end, while ever-mindful of author, reviewer and chair workload, we took specific actions.

Paper types

We reasoned that one cause of a lack of diversity of type of papers is that the typical review form in our field is primarily designed for the dominant type (which we dubbed 'NLP engineering experiment paper'), and that this makes it harder for other paper types to receive positive reviews. We decided to address this by identifying a small range of paper types and creating specialized review forms for each one. We developed an initial set, then solicited community feedback via a PC blog post (of 17 August 2017), then refined the types based on that input. A second function of the paper types was to explicitly elicit submissions of non-standard format. The paper types defined are (in order of number of submissions we received in each type): NLP engineering experiment paper, computer assisted linguistic analysis paper, resource paper, reproduction paper, position paper, and survey paper. Though the NLP engineering experiment paper type remains dominant, we are pleased at how the process has led to the diverse range of papers appearing in this volume. Submission and acceptance numbers for each type are given in Table 1.

Type	Submitted	Withdrawn	Accepted	Acceptance rate
NLPEE	657	85	217	37.94%
CALA	163	28	45	33.33%
Resource	106	7	32	32.32%
Reproduction	35	0	17	48.57%
Position	31	6	8	32.00%
Survey	25	3	12	54.55%
Overall	1017	129	331	37.27%

Table 1: Submission and acceptance, by paper type

Mentoring

One of our strategies for achieving goal (1) given above was to create a writing mentoring program, which took place before the reviewing stage. This optional program focused on helping those who perhaps aren't used to publishing in the field of computational linguistics, are early in their careers, and so on. We see mentoring as a tool that makes a conference accessible for a broader range of high-quality ideas. In other words, this isn't about pushing borderline papers into acceptance but rather alleviating presentational problems with papers that, in their underlying research quality, easily make the high required standard.

The response from mentors was very enthusiastic—over 100 people volunteered! We see this as a good indication of the long-term viability of mentoring programs. There were fewer papers submitted for mentoring than available mentors, but still a strong number: 56. We asked authors seeking mentoring to submit an abstract 4 weeks ahead of the deadline and a draft for feedback 3 weeks ahead. We asked mentors to bid on abstracts once they were received and to provide feedback on papers within one week (leaving 2 weeks for mentees to revise their submissions based on that feedback). We also provided a structured mentoring form. Of the 56 papers receiving mentoring, 47 were ultimately submitted to the conference, 3 subsequently withdrawn, and 14 accepted. We see value in the mentoring of all of these papers, both those that ultimate ended up in this volume and those that did not. The authors received valuable feedback from their mentors which can inform their future work, both on the mentored papers specifically and more generally.

Real double-blind

We know that non-blind peer review favors work by well-recognised authors or from well-recognised institutions (Tomkins et al. 2017, Laband and Piette 2014, Peters and Ceci 1982). This has two unfavorable impacts: first, substandard work may be presented as more worthy of reader and audience time than it is; secondly, when presentation slots are limited, substandard work will displace excellent work from beyond well-recognised institutions or authors. We took steps at every point of COLING 2018's review process to reduce or remove this bias.

Firstly, unlike the majority of conferences in the field, we hid author identities from area chairs, who are those making the primary recommendation about paper acceptance. We also hid reviewer identities from each other, thinking that reviewers may be inadvertently swayed by comments from a well-known reviewer, or afraid to contradict e.g. a potential future employer.[1] We also took pains to hide author identities from ourselves as PC co-chairs in making our final decisions. In addition, we hid authors

[1] We did set things up so that reviewer identities were revealed to co-reviewers at the end of the process, and announced this to reviewers. The goal here was to promote civility in reviews.

from the best paper committee; this involved withholding the "Accepted Papers" list until the best paper awards had been nominated and fixed. Finally, we followed ACL policy on preprints, blocking from consideration non-blind papers that had been published within the month before the deadline. We made one mistake in this process, where we asked authors to contact their ACs about belated author responses; this leaked author identity in a handful of cases (fewer than ten submissions).

To avoid the second unfavorable impact, we were lucky enough to have a venue of size such that we did not need to limit the number of accepted papers. Poster and oral presentation are of identical standard and receive the same treatment in the proceedings, which helped with this space limit. Having sufficient room meant that the acceptance of a substandard paper based on its author's name or institution did not lead to another, better paper being displaced. Additionally, we did not impose a maximum acceptance rate. Fixed acceptance rates under non-blind review lead to good borderline papers being lost in favor of weaker papers from well-recognized origins. Fixed acceptance rates also risk that some excellent work will be missed (Church 2005).

Reviewing quality

It is important to maintain consistency, good standards, and to be responsive. To achieve this, we had at least two people involved at every step in the program selection process. The PC chairs were co-chairs; each area had two chairs, who were both in the same coarse-grained geographic region for all but one area; all papers were reviewed by many people; and then to ensure this continued, there were Special Circumstances ACs who filled in AC gaps, and the General Chair would stand in when a PC co-chair couldn't.

We know that reviewing in computer science can be a little harsher than in general (Meyer 2011). Reviewers are not as kind to authors, or to each other. Malicious or unconstructive reviews hurt quality. So, we asked ACs to specifically look out for this kind of behavior and to ask reviewers to edit their reviews for tone where necessary, and we discounted malicious reviews when making acceptance decisions. Reviews tend to be of better tone and more constructive when signed by the author, so we gave the option for reviewers to add their names to their reviews. This couldn't be compulsory: this might risk reviewers disagreeing by name with e.g. their later bosses, which is too much to ask of the generally young CL reviewing pool.

Review quality also suffers when area chairs' jobs are too difficult. This can be addressed by reducing area chair load. To enable flexible load management, we provided information as early as possible, helped ACs move unrelated papers out of their areas, gave rapid responses during the critical period (being nine hours apart enabled an almost 24h availability), and critically, kept area size limited. This meant that there was, for example, no one area for machine translation—this would have been unmanageable and have had unacceptably low reviewing quality. Instead, papers with facets that placed them in very large areas were placed based on their other features. For example, papers on MT in low-resourced languages might be put together; those on MT for dialogue would also be grouped, but in a separate area. This way, load was managed and the risk of unreliable (and so unfair) review reduced.

Broader recognition

In keeping with building a diverse program that brings together excellent work in all aspects of our field, we wanted to make sure that our awards also recognized the many kinds of excellence to be found in work on computational linguistics. To this end, we enumerated 10 awards, of which 9 were ultimately given. We received 44 nominations for best papers over these categories, and conferred best paper awards in the categories as follows:

- Best error analysis: 'SGM: Sequence Generation Model for Multi-label Classification', by Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu and Houfeng Wang.
- Best evaluation: 'SGM: Sequence Generation Model for Multi-label Classification', by Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu and Houfeng Wang.
- Best linguistic analysis: 'Distinguishing affixoid formations from compounds', by Josef Ruppenhofer, Michael Wiegand, Rebecca Wilm and Katja Markert
- Best NLP engineering experiment: 'Authorless Topic Models: Biasing Models Away from Known Structure', by Laure Thompson and David Mimno
- Best position paper: 'Arguments and Adjuncts in Universal Dependencies', by Adam Przepiórkowski and Agnieszka Patejuk
- Best reproduction paper: 'Neural Network Models for Paraphrase Identification, Semantic Textual Similarity, Natural Language Inference, and Question Answering', by Wuwei Lan and Wei Xu
- Best resource paper: 'AnlamVer: Semantic Model Evaluation Dataset for Turkish—Word Similarity and Relatedness', by Gökhan Ercan and Olcay Taner Yıldız
- Best survey paper: 'A Survey on Open Information Extraction', by Christina Niklaus, Matthias Cetto, André Freitas and Siegfried Handschuh
- Most reproducible: 'Design Challenges and Misconceptions in Neural Sequence Labeling', by Jie Yang, Shuailong Liang and Yue Zhang

The lack of an award in the "Best challenge" category (for a paper that sets a new challenge) may reflect one of two things. On the one hand, reviewers and ACs may not have been on the look-out for this kind of excellence. On the other, there may not have been any papers with this particular strength at COLING 2018.

We also used the best paper awards to further our goal of promoting reproducibility. Specifically, we required that, in order to be eligible for one of the above awards, any code/resources associated with the research reported be publicly available by camera ready time. This encouraged all authors (not just those of nominated papers) to be sure their code/resources were available.

A team effort

In total, we relied upon and COLING was excellently supported by 76 area chairs, 1029 reviewers, and 11 best paper committee members, all of whom we thank profusely for an excellent iteration of the International Conference on Computational Linguistics.

Emily M. Bender (University of Washington)
Leon Derczynski (IT University of Copenhagen)
COLING 2018 Program Committee Co-chairs

References

Kenneth Church. 2005. Reviewing the reviewers. *Computational Linguistics*, 31(4):575–578.

David N. Laband and Michael J. Piette. 1994. Does the "blindness" of peer review influence manuscript selection efficiency? *Southern Economic Journal*, 60(4):896–906.

Bertrand Meyer. 2011. The nastiness problem in computer science. CACM blog; https://cacm.acm.org/blogs/blog-cacm/123611-the-nastiness-problem-in-computer-science/fulltext.

Eric Price. 2014. The NIPS experiment. Online blog; http://blog.mrtz.org/2014/12/15/the-nips-experiment.html.

Andrew Tomkins, Min Zhang, and William D. Heavlin. 2017. Reviewer bias in single-versus double-blind peer review. *PNAS*, 114(48):12708–12713.

Peters, Douglas P. and Ceci, Stephen J. 1982. Peer-review practices of psychological journals: The fate of published articles, submitted again. *Behavioral and Brain Sciences*, 5(2):187-–195.

Organizers

General Chair

Pierre Isabelle, National Research Council, Canada

Local Arrangements Chair

Sergei Nirenburg, Rensselaer Polytechnic Institute, USA

Program Chairs

Emily M. Bender, University of Washington, USA
Leon Derczynski, IT University of Copenhagen, Denmark

Sponsorships Chairs

Leo Wanner, Pompeu Fabra University, Spain
Satoshi Sekine, New York University, USA
Sun Le, Chinese Academy of Sciences, China
Soo-Min Pantel, Amazon, USA

Tutorials Chairs

Donia Scott, University of Sussex, UK
Pascale Fung, Hong Kong University of Science and Technology, Hong Kong
Marilyn Walker, University of California Santa Cruz, USA

Workshops Chairs

Timothy Baldwin, The University of Melbourne, Australia
Yoav Goldberg, Bar Ilan University, Israel
Jing Jiang, Singapore Management University, Singapore

Demos Chair

Dongyan Zhao, Peking University, China

Proceedings Chairs

Xiaodan Zhu, Queen's University, Canada
Zhiyuan Liu, Tsinghua University, China

Webmaster

Qian Chen, University of Science and Technology of China, China
Christine Tang, Rensselaer Polytechnic Institute, USA

Publicity Chairs

Yuji Matsumoto, Nara Institute of Science and Technology, Japan
Hiroshi Noji, National Institute of Advanced Industrial Science and Technology, Japan

Area Chairs

Afra Alishahi, Tilburg University, Netherlands
Alexandre Rademaker, IBM Brazil Research Lab, Brazil
Alexis Palmer, University of North Texas, USA
Aline Villavicencio, University of Essex, UK
Alvin Grissom II, Ursinus College, USA
Anders Søgaard, University of Copenhagen, Denmark
Andreas Vlachos, University of Sheffield, UK
Andrew Caines, University of Cambridge, UK
Ann Clifton, Amazon, USA
Anna Rumshisky, University of Massachusetts Lowell, USA
Antske Fokkens, Vrije Universiteit Amsterdam, Netherlands
Arash Eshghi, Heriot-Watt University, UK
Asad Sayeed, University of Gothenburg, Sweden
Aurélie Herbelot, University of Trento, Italy
Avirup Sil, IBM, USA
Barry Devereux, Queen's University Belfast, UK
Chaitanya Shivade, IBM, USA
Dan Garrette, Google, USA
Daniel Lassiter, Stanford University, USA
David Schlangen, Bielefeld University, Germany
Deyi Xiong, Soochow University, China
Di Jiang, Chinese Academy of Social Science, China
Eric Nichols, Honda Research Institute Japan, Japan
Francis Bond, Nanyang Technological University, Singapore
Francis Ferraro, University of Maryland, Baltimore County, USA
Georgiana Dinu, Amazon AWS, USA
Gerard de Melo, Rutgers University, USA
Gina-Anne Levow, University of Washington, USA
Harry Bunt, Tilburg University, Netherlands
Hatem Haddad, Universite Libre de Bruxelles, Belgium
Isabelle Augenstein, University of Copenhagen, Denmark
Jiajun Zhang, Chinese Academy of Sciences, China
Jose Camacho-Collados, Cardiff University, UK
Karin Verspoor, The University of Melbourne, Australia
Kevin Duh, Johns Hopkins University, USA
Klinton Bicknell, Northwestern University, USA
Lilja Øvrelid, University of Oslo, Norway
Maja Popović, Humboldt University of Berlin, Germany
Manuel Montes-y-Gómez, National Institute of Astrophysics, Optics and Electronics, Mexico
Marcos Zampieri, University of Wolverhampton, UK
Marie-Catherine de Marneffe, The Ohio State University, USA
Meliha Yetisgen, University of Washington, USA
Michael Tjalve, Microsoft, USA
Miguel Ballesteros, IBM, USA
Mike Tian-Jian Jiang, DG Lab, Japan
Mohammad Taher Pilehvar, University of Cambridge, UK
Na-Rae Han, University of Pittsburgh, USA
Naomi Feldman, University of Maryland, USA
Natalie Schluter, IT University of Copenhagen, Denmark

Nathan Schneider, Georgetown University, USA
Nikola Ljubešić, Jožef Stefan Institute, Slovenia
Nurit Melnik, The Open University of Israel, Israel
Qin Lu, Hong Kong Polytechnic University, Hong Kong
Roman Klinger, University of Stuttgart, Germany
Sadid A. Hasan, Philips Research North America, USA
Sanja Štajner, University of Mannheim, Germany
Sara Tonelli, Fondazione Bruno Kessler, Italy
Sarvnaz Karimi, CSIRO, Australia
Steven Bethard, University of Arizona, USA
Sujian Li, Peking University, China
Sunayana Sitaram, Microsoft Research India, India
Tal Linzen, Johns Hopkins University, USA
Valia Kordoni, Humboldt University of Berlin, Germany
Vivek Kulkarni, University of California Santa Barbara, USA
Viviane Moreira, Federal University of Rio Grande do Sul, Brazil
Wei Xu, The Ohio State University, USA
Wenjie Li, The Hong Kong Polytechnic University, Hong Kong
Xiang Ren, University of Southern California, USA
Xiaodan Zhu, Queen's University, Canada
Yang Feng, Chinese Academy of Sciences, China
Yonatan Bisk, University of Washington, USA
Yue Zhang, Singapore University of Technology and Design, Singapore
Yun-Nung Chen, National Taiwan University, Taiwan
Zachary Chase Lipton, Carnegie Mellon University, USA
Zhiyuan Liu, Tsinghua University, China
Željko Agić, IT University of Copenhagen, Denmark

Best-Paper Committee

Steven Bethard, University of Arizona, USA
Kevin Duh, Johns Hopkins University, USA
Eva Hajicova, Charles University, Czech
Aurelie Herbelot, University of Trento, Italy
Qin Lu, Hong Kong Polytechnic University, Hong Kong
Diana Maynard, University of Sheffield, UK
Asad Sayeed, University of Gothenburg, Sweden
Donia Scott, University of Sussex, UK
Aline Villavicencio, University of Essex, UK
Andreas Vlachos, University of Sheffield, UK
Lilja Øvrelid, University of Oslo, Norway

Outstanding Reviewers

Aaron Steven White, Adam Tsakalidis, Akihiro Tamura, Alberto Barrón-Cedeño, Alexander Erdmann, Alexander Kuhnle, Alice Lai, Anne Cocos, Anne-Lyse Minard, Annemarie Friedrich, Antoine Bosselut, Antonio Toral, Ari Holtzman, Arman Cohan, Arne Köhn, Artuur Leeuwenberg, Barbara Plank, Bing Liu, Çağrı Çöltekin, Carlos Gómez-Rodríguez, Carlos Ramisch, Chiraag Lala, Chloé Braud, Chloé Braud, Christoph Cerisara, Christoph Tillmann, Christos Christodoulopoulos, Colin Cherry, Cory Shain, Daisuke Kawahara, Dan Simonson, Daniel Cer, Daniel Fried, Devamanyu Hazarika, Edward Greffenstette, Elena Kochkina, Emmanuele Chersoni, Erin Bennett, Erin Olson, Farah Benamara, Feifei Zhai, Filip Ilievski, Gabriel Stanovsky, Gaël Dias, Gianluca Lebani, Guy Emerson, Hamed Hassanzadeh, Hamed Khanpour, Helen O'Horan, Henning

Wachsmuth, Hongyin Luo, Igor Malioutov, Ines Rehbein, Ingrid Falk, Ivan Vulic, Jan Buys, Jeff Good, Jena D. Hwang, Jennifer Chu-Carroll, Jianfei Yu, Jiawei Wu, Jie Yang, Jing Jiang, Johannes Welbl, Josiah Wang, Kai Zhao, Karen Fort, Karl Pichotta, Kasia Hitczenko, Keelan Evanini, Kevin Cohen, Kevin Small, Kris Cao, Lifeng Jin, Linfeng Song, Manfred Stede, Manuel Ciosici, Marine Carpuat, Markus Saers, Martin Reynaert, Marty van Schijntel, Matthias Huck, Maxwell Forbes, Md Arafat Sultan, Menno van Zaanen, Mereno Coco, Michael Elhadad, Michael Wiegand, Mikhail Kozhevnikov, Min-Yen Kan, Miryam de Lhoneux, Mohit Bansal, Morgan Sonderegger, Na-Rae Han, Naoaki Okazaki, Nitish Gupta, Olivier Ferret, Ondřej Dušek, Orphee De Clercq, Ozan Arkan Can, Parisa Korsjamshidi, Patrick Littell, Penny Labropoulou, Peter Jansen, Philippe Thomas, Pranava Swaroop Madhyastha, Raquel Fernandez, Richard Evans, Rik Koncel-Kedziorski, Robert Frank, Robert Ostling, Ron Artstein, Roser Morante, Sebastian Schuster, Shachar Mirkin, Shervin Malmasi, Simon Keizer, Stella Frank, Stephen Wan, Stergios Chatzikyriakidis, Sujan Pereira, Sujay Kumar Jauhar, Sumeet Agarwal, Takuya Matsuzaki, Tanel Alumäe, Taraka Rama, Tatjana Scheffler, Tim Dozat, Tracy Holloway King, Valerio Basile, Vincent Ng, Wei-Ning Hsu, Weiwei Yang, Xu Sun, Yang Liu, Yogarshi Vyas, Yonatan Belinkov, Yong Cheng, Yoshihiko Hayashi, Yung-Chun Chang, Yuval Pinter, and Zhe Zhao

Reviewers

Aaditya Prakash, Aaron Steven White, A Ramanathan, A. Seza Doğruöz, Abidalrahman Moh'd, Adam Grycner, Adam Jardine, Adam Meyers, Adam Pease, Adam Przepiórkowski, Adam Tsakalidis, Adam Wyner, Adhiguna Kuncoro, Aditya Joshi, Adrià de Gispert, Ahmed Abdelali, AiTi Aw, Akihiro Tamura, Alane Suhr, Albert Gatt, Alberto Barrón-Cedeño, Aldo Gangemi, Alessandro Lenci, Alessandro Mazzei, Alessandro Moschitti, Alessandro Raganato, Alessio Palmero Aprosio, Alexander Clark, Alexander Erdmann, Alexander Kuhnle, Alexander Panchenko, Alexander Rudnicky, Alexandra Balahur, Alexandra I. Cristea, Alexandre Rademaker, Alexey Romanov, Alexis Nasr, Alfio Gliozzo, Alice Lai, Alisa Zhila, Alistair Willis, Allyson Ettinger, Alona Fyshe, Alvin Grissom II, Amal Zouaq, Amalia Todirascu, Amba Kulkarni, Amir Hazem, Amitava Das, Amittai Axelrod, Amrita Saha, Amrith Krishna, Amy Isard, Amália Mendes, Anastassia Loukina, Anders Søgaard, Andrea E. Martin, Andreas Eisele, Andreas Maletti, Andreea Godea, Andrew Caines, Andrew Lamont, Anette Frank, Animesh Mukherjee, Anirban Laha, Anja Belz, Anna Rogers, Anna Lisa Gentile, Anne Cocos, Anne-Laure Ligozat, Anne-Lyse Minard, Annemarie Friedrich, Antal van den Bosch, Anthony Nguyen, Antoine Bosselut, Antoine Doucet, Antonio Jimeno Yepes, Antonio Pareja-Lora, Antonio Toral, Antonios Anastasopoulos, Antske Fokkens, Antti Arppe, António Branco, Anupam Guha, Aoife Cahill, Arantza Diaz de Ilarraza, Archna Bhatia, Ari Holtzman, Ari Rappoport, Arkaitz Zubiaga, Arman Cohan, Arne Köhn, Arpit Mittal, Artuur Leeuwenberg, Ashutosh Modi, Ashwini Vaidya, Ashwinkumar Ganesan, Asif Ekbal, Asma Ben Abacha, Atsushi Fujita, Awais Athar, Axel Ngonga, Baobao Chang, Baoli LI, Barbara Plank, Beata Megyesi, Beatrice Daille, Behrang QasemiZadeh, Ben Hachey, Benoit Crabbé, Berlin Chen, Bernd Bohnet, Berthold Crysmann, Bhuwan Dhingra, Biao Zhang, Bing Liu, Bingning Wang, Bishan Yang, Björn Gambäck, Bofang Li, Bolette Pedersen, Boliang Zhang, Bonan Min, Bonaventura Coppola, Boyi Xie, Braja Gopal Patra, Brian Plüss, Brian Riordan, Brian Roark, Bryan Wilkinson, Caitlin Richter, Caleb Chen Cao, Camilo Thorne, Canasai Kruengkrai, Cao Liu, Carla Parra Escartín, Carlo Strapparava, Carlos Alzate, Carlos Gallo, Carlos Gómez-Rodríguez, Carlos Ramisch, Carolina Scarton, Caroline Brun, Casey Kennington, Chara Tsoukala, Charles Welch, Chen Chen, Chenchen Ding, Chengqing Zong, Chengyao Chen, Chiraag Lala, Chloé Braud, Chris Brockett, Christian Hardmeier, Christian RETORE, Christian Wartena, Christina Bergmann, Christine Howes, Christoph Tillmann, Christophe Cerisara, Christophe Gravier, Christopher Bryant, Christopher Clark, Christopher Potts, Christos Christodoulopoulos, Claire Bonial, Claire Gardent, Clara Vania, Claudio Delli Bovi, Claudio Gutierrez, Colin Cherry, Collin Baker, Constantin Orasan, Coral Hughto, Cory Shain, Costanza Navarretta, Cristina Bosco, Cristina Marco, Cyril Goutte, Dag Haug, Daisuke Kawahara, Damir Cavar, Dan Flickinger, Dan Goldwasser, Dan Liu,

Dan Simonson, Daniel Bauer, Daniel Beck, Daniel Cer, Daniel Fried, Daniel Hershcovich, Daniel Khashabi, Daniel Paiva, Daniel Preoţiuc-Pietro, Danilo Croce, Danish Contractor, Dasha Bogdanova, David Inman, David Kauchak, David Mimno, David Traum, David Vilar, David M. Howcroft, David R. Mortensen, Deepak Venugopal, Dehong Ma, Denis Newman-Griffis, Denis Paperno, Desmond Elliott, Devamanyu Hazarika, Diarmuid Ó Séaghdha, Diego Frassinelli, Dimitri Kartsaklis, Dina Wonsever, Dipankar Das, Dipti Sharma, Dirk Hovy, Djamé Seddah, Dmitriy Dligach, Donghong Ji, Dorothee Beermann, Doug Downey, Duy Tin Vo, E Umamaheswari Vasanthakumar, Eduard Hovy, Eduardo Blanco, Edward Stabler, Efstathios Stamatatos, Egoitz Laparra, Ehsan Zare Borzeshi, Ehud Reiter, Eiji ARAMAKI, Ekaterina Kochmar, Ekaterina Lapshinova-Koltunski, Ekaterina Vylomova, Elaheh ShafieiBavani, Eleftherios Avramidis, Elena Cabrio, Elena Kochkina, Elena Volodina, Elisabetta Jezek, Elke Teich, Els Lefever, Emmanuel Morin, Emmanuele Chersoni, Eric De La Clergerie, Eric Nichols, Erik Velldal, Eriks Sneiders, Erin Bennett, Erin Olson, Esau Villatoro-Tello, Eugenio Martínez-Cámara, Eva Hajicova, Ewan Dunbar, Ezer Rasin, Fabio Massimo Zanzotto, Fabiola Henri, Fandong Meng, Farah Benamara, Fatiha Sadat, Federico Fancellu, Federico Nanni, Fei Liu, Fei Xia, Feifei Zhai, Felix Engelmann, Felix Gervits, Feng Ji, Filip Ginter, Filip Ilievski, Fintan Costello, Florian Boudin, Florin Bulgarov, Forough Poursabzi-Sangdeh, Francesca Bonin, Francesco Ronzano, Francis Mollica, Francis Tyers, Francisco Casacuberta, Francisco Rangel, Franco M. Luque, François Lareau, François Portet, François Yvon, Frederic Bechet, Frédéric Blain, Gabriel Stanovsky, Gabriella Lapesa, Gaël Dias, Gaël Lejeune, George Foster, George Gkotsis, George Aaron Broadwell, Georgios Petasis, Gerasimos Lampouras, Gerhard Jäger, Gerhard Weikum, Gerlof Bouma, German Rigau, Gholamreza Haffari, Gianluca Lebani, Gianni Barlacchi, Giorgio Magri, Giorgio Orsi, Giovanni Da San Martino, Goran Glavaš, Goran Nenadic, Gosse Bouma, Gourab Kundu, Grace Hui Yang, Graeme Hirst, Gregory Finley, Gregory Grefenstette, Gregory Kobele, Gregory Mills, Grzegorz Chrupała, Guenter Neumann, Guillaume Wisniewski, Guillem Collell, Guoping Huang, Guy Emerson, Guy Lapalme, Gülşen Eryiğit, Hai Zhao, Hamed Hassanzadeh, Hamed Khanpour, Hamidreza Ghader, Hao Cheng, Hao Zhang, Haris Papageorgiou, Harry Bunt, Hayate Iso, Heather Pon-Barry, Helen O'Horan, Helen Yannakoudakis, Helena Caseli, Helmut Horacek, Helmut Schmid, Hen-Hsen Huang, Hendrik Buschmeier, Heng Yu, Henning Wachsmuth, Heriberto Cuayahuitl, Hila Gonen, Hiram Calvo, Hiroshi Noji, Hoang Cuong, Holger Schwenk, Hongmin Wang, Hongyin Luo, Hsin-Hsi Chen, Hsin-Min Wang, Hua Wu, Hugo Gonçalo Oliveira, Hugo Jair Escalante, Hwee Tou Ng, Hyokun Yun, Iacer Calixto, Igor Boguslavsky, Igor Malioutov, Igor Shalyminov, Ildikó Pilán, Ines Rehbein, Ingrid Falk, Ingrid Zukerman, Ioana Hulpus, Irina Temnikova, Iris Hendrickx, Ivan Sanchez, Ivan Vulić, Ivan Vladimir Meza Ruiz, Iz Beltagy, Jad Kabbara, Jae Sung Lee, Jaime Lorenzo-Trueba, James Reid, James H. Martin, Jamie Macbeth, Jan Buys, Jan Hajic, Jan Odijk, Jan Šnajder, Jana Diesner, Jane Chandlee, Jason Naradowsky, Jason Riesa, Javid Ebrahimi, Jeff Good, Jeff Mitchell, Jelke Bloem, Jen-Tzung Chien, Jena D. Hwang, Jeniya Tabassum, Jennifer Chu-Carroll, Jennifer Williams, Jeremy Barnes, Jesse Dodge, Jesse Thomason, Jesús González-Rubio, Jey Han Lau, Ji Xin, Jiaming Xu, Jianfei Yu, Jiang Guo, Jiangming Liu, Jiawei Wu, Jie Yang, Jill Burstein, Jim Blevins, Jin-Dong Kim, Jing Jiang, Jinho D. Choi, Joachim Bingel, Joakim Nivre, Jodi Schneider, Joel Tetreault, Joern Wuebker, Johan Bos, Johann Petrak, Johanna Monti, Johannes Bjerva, Johannes Welbl, John Chen, John Hale, John Wieting, John Philip McCrae, Jojo Sze-Meng Wong, Jonathan Brennan, Jongbok Kim, Joonsuk Park, Jordi Atserias Batalla, Jorge Gracia, Josef Ruppenhofer, Joseph Le Roux, Josiah Wang, Joyce Chai, Juan Soler, Judith Eckle-Kohler, Juergen Trouvain, Julian Hough, Julian Michael, Julian Richardson, Julie Weeds, Jun Araki, Jun Zhao, Junqiu Wei, Junwei Bao, Junyi Jessy Li, Jörg Tiedemann, Jürgen Wedekind, Kai Zhao, Kalika Bali, Kalina Bontcheva, Kalliopi Zervanou, Kallirroi Georgila, Karin Verspoor, Karl Pichotta, Karl Stratos, Karën Fort, Kasia Hitczenko, Katerina Frantzi, Katharina Kann, Kathy Lee, Katrin Erk, Katsuhito Sudoh, Kazunori Komatani, Keelan Evanini, Keh-Yih Su, Keisuke Sakaguchi, Keith VanderLinden, Kellie Webster, Kemal Oflazer, Kenji Sagae, Kenneth Church, Kenneth Heafield, Kentaro Inui, Kentaro Torisawa, Kevin Cohen, Kevin Duh, Kevin Scannell,

Kevin Seppi, Kevin Small, Khalid Al Khatib, Kilian Evang, Kim Anh Nguyen, Kiril Simov, Kirk Roberts, Koji Mineshima, Koji Murakami, Koldo Gojenola, Kris Cao, Kristen Howell, Krister Lindén, Kristina Striegnitz, Kuang-hua Chen, Kumiko Tanaka-Ishii, Kun Xu, L. Alfonso Urena Lopez, Lan Du, Lane Schwartz, Larry Moss, Laurel Perkins, Laurent Besacier, Laurette Pretorius, Lawrence Cavedon, Leanne Rolston, Lei He, Lei Zhang, Leila Kosseim, Lemao Liu, Lena Jäger, Leon Bergen, Leonel Figueiredo de Alencar, Li Dong, Liang-Chih Yu, Lianhui Qin, Libby Barak, Lifeng Jin, Lili Mou, Liling Tan, Lin Yang, Linfeng Song, Lionel Nicolas, Long Duong, Long Zhou, Lonneke van der Plas, Lori Levin, Luciana Benotti, Lucy Vanderwende, Lucy H. Lin, Luheng He, Luis Espinosa Anke, Lun-Wei Ku, Maciej Piasecki, Maja Miličević, Makoto Miwa, Malvina Nissim, Mamoru Komachi, Manex Agirrezabal, Manfred Sailer, Manfred Stede, Manikandan R, Mans Hulden, Manuel Ciosici, Maofu Liu, Marcel Bollmann, Marcelo Luis Errecalde, Marco Basaldella, Marco Idiart, Marco Turchi, Mareike Hartmann, Marek Rei, Maria Barrett, Maria Liakata, Maria Nadejde, Mariana Neves, Marianna Apidianaki, Mariano Felice, Marie Candito, Marie-Catherine de Marneffe, Marie-Francine Moens, Marina Sokolova, Marine Carpuat, Marion Weller-Di Marco, Mark Carman, Mark Dras, Mark Finlayson, Mark Fishel, Mark Greenwood, Mark Hasegawa-Johnson, Mark Last, Mark Sammons, Mark Stevenson, Mark-Jan Nederhof, Markus Egg, Markus Freitag, Markus Saers, Markus Zopf, Marta Recasens, Marta Tatu, Marta R. Costa-jussà, Marten Postma, Marten van Schijndel, Martha Palmer, Martin Benjamin, Martin Potthast, Martin Reynaert, Martin Riedl, Marzieh Saeidi, Masaaki Nagata, Masoud Jalili Sabet, Massimo Poesio, Mathieu Roche, Matko Bošnjak, Matt Gardner, Matteo Negri, Matthew Purver, Matthias Gallé, Matthias Hartung, Matthias Huck, Matthieu Constant, Maud Ehrmann, Mauro Cettolo, Max Kisselew, Maxwell Forbes, Maya Ramanath, Mayank Singh, Md Arafat Sultan, Md Shad Akhtar, Meishan Zhang, Meliha Yetisgen, Memduh Gokirmak, Menno van Zaanen, Meriem Beloucif, Michael Becker, Michael Denkowski, Michael Elhadad, Michael Maxwell, Michael Roth, Michael Wiegand, Michael Zock, Michael Wayne Goodman, Michael Yoshitaka Erlewine, Michal Lukasik, Michel Gagnon, Michel Simard, Mihaela Vela, Mike Tian-Jian Jiang, Mikhail Kozhevnikov, Milan Dojchinovski, Milica Gasic, Miloslav Konopik, Miloš Stanojević, Min-Yen Kan, Min-Yuh Day, Mingbo Ma, Mingxuan Wang, Minjoon Seo, Minlie Huang, Mireia Farrús, Miriam R L Petruck, Miryam de Lhoneux, Mo Yu, Mohamed Yahya, Mohammad Ebrahimi, Mohammad Shahed Sorower, Mohammad Taher Pilehvar, Mohit Bansal, Mohsen Mesgar, Monica Dominguez, Monojit Choudhury, Montserrat Marimon, Moreno Coco, Morgan Sonderegger, Muntsa Padró, Na-Rae Han, Nadir Durrani, Nancy Ide, Nanyun Peng, Naoaki Okazaki, Naoki Yoshinaga, Naomi Tachikawa Shapiro, Naoya Inoue, Nasredine Semmar, Nazneen Fatema Rajani, Ni Lao, Nicoletta Calzolari, Nikhil Londhe, Nikola Ljubešić, Nikolaos Aletras, Nils Reiter, Nina Dethlefs, Nina Zhou, Niranjan Balasubramanian, Nitish Gupta, Niyati Chhaya, Nizar Habash, Octavia-Maria Şulea, Oier Lopez de Lacalle, Oladimeji Farri, Olga Uryupina, Olga Zamaraeva, Oliver Hellwig, Olivier Ferret, Omer Levy, Omri Abend, Ondřej Dušek, Or Biran, Oren Melamud, Orphee De Clercq, Oscar Täckström, Ozan Arkan Can, Pablo Gamallo, Pamela Shapiro, Paolo Rosso, Paramita Mirza, Parisa Kordjamshidi, Pascual Martínez-Gómez, Pasquale Minervini, Patrick Ehlen, Patrick Littell, Patrick Paroubek, Paul Buitelaar, Paul Piwek, Paul Reisert, Pavel Braslavski, Pavel Logacev, Pawan Goyal, Pei-Hao Su, Penny Labropoulou, Peter Dirix, Peter Jansen, Peter Ljunglöf, Petya Osenova, Philipp Cimiano, Philipp Koehn, Philippe Muller, Philippe Thomas, Phivos Mylonas, Phoebe Mulcaire, Phong Le, Piek Vossen, Pierre Lison, Pierre Nugues, Pierre Zweigenbaum, Ping Jian, Pontus Stenetorp, Pranava Swaroop Madhyastha, Prerana Singhal, Preslav Nakov, Prokopis Prokopidis, Pushpak Bhattacharyya, Qi Zhang, Qian Chen, Qian Yang, Qin Lu, Qinlan Shen, Quan Liu, Qun Liu, Quynh Ngoc Thi Do, Rabih Zbib, Rachael Tatman, Rachele Sprugnoli, Radu Florian, Rafael E. Banchs, Raffaella Bernardi, Raghav Goyal, Rami Al-Rfou, Ramy Eskander, Ramón Astudillo, Raphael Rubino, Raquel Fernández, Raquel Martínez, Renfen Hu, Richard Eckart de Castilho, Richard Evans, Richard Futrell, Richard Tzong-Han Tsai, Richong Zhang, Rik Koncel-Kedziorski, Rob Malouf, Rob Voigt, Robert Daland, Robert Frank, Robert Östling, Roberto Basili, Robin Cooper, Rodrigo Wilkens, Roland Kuhn, Roland

Roller, Ron Artstein, Rong Xiang, Roser Morante, Roxana Girju, Roy Bar-Haim, Rui Chaves, Rui Wang, Ruifeng Xu, Ruihong Huang, Ruiji Fu, Ruobing Xie, Ruochen Xu, Ryan Cotterell, Ryan Georgi, Ryu Iida, Ryuichiro Higashinaka, Sabine Bergler, Sadao Kurohashi, Saif Mohammad, Salah Ait-Mokhtar, Salvatore Romeo, Sameer Singh, Samuel Bowman, Sanda Harabagiu, Sander Wubben, Sandesh Swamy, Sandra Kübler, Sang Phan, Sanghoun Song, Sanja Štajner, Santanu Pal, Sapna Negi, Sara Stymne, Sara Tonelli, Sascha Rothe, Satoshi Sekine, Sean MacAvaney, Sebastian Krause, Sebastian Nordhoff, Sebastian Schuster, Seong-Bae Park, Sergio Jimenez, Sergiu Nisioi, Shachar Mirkin, Shalom Lappin, Shammur Absar Chowdhury, Shaolei Wang, Shaonan Wang, Shashi Narayan, Sheila Castilho, Shervin Malmasi, Shih-Hung Wu, Shizhu He, Sho Hoshino, Shoaib Jameel, Shuhan Wang, Shujian Huang, Shuohang Wang, Siddharth Patwardhan, Silvio Cordeiro, Simon Baker, Simon Keizer, Simon Kocbek, Simon Mille, Simone Paolo Ponzetto, Sina Zarrieß, Siqi Bao, Siva Reddy, Sorcha Gilroy, Soujanya Poria, Stanislau Semeniuta, Stasinos Konstantopoulos, Stefania Degaetano-Ortlieb, Stefano Menini, Steffen Eger, Stella Frank, Stephen McGregor, Stephen Wan, Stergios Chatzikyriakidis, Stergos Afantenos, Steven Moran, Suchet Chachra, Sudha Rao, Sudip Kumar Naskar, Sujan Perera, Sujay Kumar Jauhar, Sumeet Agarwal, Sunayana Sitaram, Suncong Zheng, Sunghwan Mac Kim, Susanne Burger, Suzan Verberne, Svetla Koeva, Svetlana Stoyanchev, Svitlana Volkova, Swabha Swayamdipta, Tadashi Nomoto, Taesung Lee, Takenobu Tokunaga, Takuya Matsuzaki, Tanel Alumäe, Tanja Samardzic, Taraka Rama, Tatjana Scheffler, Taylor Mahler, Ted Pedersen, Tejaswini Deoskar, Teresa Lynn, Tetsuji Nakagawa, Thamar Solorio, Thiago Pardo, Thien Huu Nguyen, Thierry Declerck, Thierry Hamon, Thomas Francois, Thomas Schatz, Thomas Wasow, Tim Hunter, Tim Van de Cruys, Timo Baumann, Timothy Baldwin, Timothy Dozat, Timothy Miller, Ting Liu, Ting Qian, Tobias Horsmann, Tommaso Pasini, Tomáš Brychcín, Tomáš Hercig, Torsten Zesch, Tracy Holloway King, Tristan Miller, Tsung-Hsien Wen, Tyler Baldwin, Valeria dePaiva, Valerio Basile, Vasileios Lampos, Veno Pachovski, Verena Lyding, Verónica Pérez-Rosas, Victoria Yaneva, Victoria Zayats, Viet-An Nguyen, Vikramjit Mitra, Vincent Claveau, Vincent Ng, Vincent Vandeghinste, Vinodkumar Prabhakaran, Virach Sornlertlamvanich, Vishrawas Gopalakrishnan, Vivek Datla, Vivek Kulkarni, Vivek Srikumar, Vivek Kumar Rangarajan Sridhar, Viviane Moreira, Volkan Cirik, Vuk Batanović, Wei Li, Wei Lu, Wei Song, Wei-Fan Chen, Wei-Ning Hsu, Weinan Zhang, Weiwei Sun, Weiwei Yang, Wen-wai Yim, Wencan Luo, Wenduan Xu, Wenjie Li, Wenliang Chen, Wenpeng Yin, William Schuler, William Yang Wang, Wolfgang Maier, Woodley Packard, Wuwei Lan, Xavier Tannier, Xi Victoria Lin, Xian-Ling Mao, Xianpei Han, Xiao Ling, Xiao Sun, Xiaochang Peng, Xiaodong Liu, Xiaofeng He, Xiaoman Pan, Ximing Li, Xing Wang, Xingyi Song, Xiujun Li, Xu Sun, Xuanjing Huang, Yaakov HaCohen-Kerner, Yadollah Yaghoobzadeh, Yael Netzer, Yan Shao, Yan Song, Yang Liu, Yangfeng Ji, Yankai Lin, Yannis Korkontzelos, Yannis Papanikolaou, Yanqing He, Yanran Li, Yansong Feng, Ye Tian, Yi Luan, Yimai Fang, Yiming Cui, Yiping Song, Yogarshi Vyas, Yonatan Belinkov, Yong Cheng, Yoong Keok Lee, Yoshihiko Hayashi, Yoshinori Sagisaka, Young-Suk Lee, Yu Wu, Yu Zhang, Yuan Cao, Yuan Ling, Yuanfeng Song, Yubo Chen, Yufan Guo, Yufang Hou, Yugo Murawaki, Yuhang Guo, Yuhao Zhang, Yuichiroh Matsubayashi, Yulan He, Yulia Tsvetkov, Yunfei Long, Yung-Chun Chang, Yusuke Oda, Yuta Tsuboi, Yutaka Sasaki, Yuval Pinter, Yvette Graham, Yvonne Adesam, Zdenka Uresova, Zeerak Waseem, Zhao Yan, Zhe Zhao, Zhengxian Gong, Zhiguo Wang, Zhiting Hu, Zhiyang Su, Zhiyang Teng, Zhiyuan Chen, Zhongqing Wang, Zhongyu Wei, Zhunchen Luo, Ziqiang Cao, Zongyang Ma, Çağrı Çöltekin, and Özlem Çetinoğlu

Outstanding Paper Mentors

Kevin Cohen, Carla Parra Escartín, David Mimno, Emily Morgan, Irina Temnikova, and Jennifer Williams

Paper Mentors

Aditya Joshi, Ahmed Abdelali, Alexandra Balahur, Alexandre Rademaker, Arkaitz Zubiaga, Arpit

Mittal, Carla Parra Escartín, Christian Retore, Christina Bergmann, Christos Christodoulopoulos, Constantin Orasan, Daniel Cer, David Chiang, David Mimno, Devamanyu Hazarika, Donia Scott, Ekaterina Vylomova, Emily Morgan, Francis Tyers, Guy Emerson, Helen O'Horan, Heng Yu, Hiram Calvo, Irina Temnikova, Ivan Vladimir Meza Ruiz, Jen-Tzung Chien, Jennifer Williams, Junwei Zhou, Kenneth Church, Kevin Cohen, Lei Zhang, Lili Mou, Linfeng Song, Luis Espinosa Anke, Marek Rei, Mariana Neves, Milan Dojchinovski, Mireia Farrús, Octavia-Maria Şulea, Parisa Kordjamshidi, Prerana Singhal, Ramón Astudillo, Raphael Rubino, Rik Koncel-Kedziorski, Rui Wang, Shervin Malmasi, Siddharth Patwardhan, Taraka Rama, Tracy Holloway King, Victoria Yaneva, Xiaofeng He, Yuan Ling, and Zhongqing Wang

Invited Speakers

Fabiola Henri, University of Kentucky, USA
Hannah Rohde, University of Edinburgh, UK
James Pustejovsky, Brandeis University, USA
Min-Yen Kan, National University of Singapore, Singapore

Table of Contents

Conference Program

Tuesday 21st August

9:00–9:30 Opening ceremony

9:30–10:30 Invited talk: James Pustejovsky

10:30–11:00 Refreshment break

11:00–12:20 Session 1-1-a: Co-reference

A New Approach to Animacy Detection
Labiba Jahan, Geeticka Chauhan and Mark Finlayson

Zero Pronoun Resolution with Attention-based Neural Network
Qingyu Yin, Yu Zhang, Weinan Zhang, Ting Liu and William Yang Wang

They Exist! Introducing Plural Mentions to Coreference Resolution and Entity Linking
Ethan Zhou and Jinho D. Choi

Triad-based Neural Network for Coreference Resolution
Yuanliang Meng and Anna Rumshisky

Tuesday 21st August (continued)

11:00–12:20 Session 1-1-b: Low-resource languages

Unsupervised Morphology Learning with Statistical Paradigms
Hongzhi Xu, Mitchell Marcus, Charles Yang and Lyle Ungar

Challenges of language technologies for the indigenous languages of the Americas
Manuel Mager, Ximena Gutierrez-Vasques, Gerardo Sierra and Ivan Meza-Ruiz

Low-resource Cross-lingual Event Type Detection via Distant Supervision with Minimal Effort
Aldrian Obaja Muis, Naoki Otani, Nidhi Vyas, Ruochen Xu, Yiming Yang, Teruko Mitamura and Eduard Hovy

Neural Transition-based String Transduction for Limited-Resource Setting in Morphology
Peter Makarov and Simon Clematide

11:00–12:20 Session 1-1-c: Parsing

Distance-Free Modeling of Multi-Predicate Interactions in End-to-End Japanese Predicate-Argument Structure Analysis
Yuichiroh Matsubayashi and Kentaro Inui

Sprucing up the trees – Error detection in treebanks
Ines Rehbein and Josef Ruppenhofer

Two Local Models for Neural Constituent Parsing
Zhiyang Teng and Yue Zhang

RNN Simulations of Grammaticality Judgments on Long-distance Dependencies
Shammur Absar Chowdhury and Roberto Zamparelli

Tuesday 21st August (continued)

11:00–12:20 Session 1-1-posters: Application, extraction and knowledge

How Predictable is Your State? Leveraging Lexical and Contextual Information for Predicting Legislative Floor Action at the State Level
Vladimir Eidelman, Anastassia Kornilova and Daniel Argyle

Learning to Search in Long Documents Using Document Structure
Mor Geva and Jonathan Berant

Incorporating Image Matching Into Knowledge Acquisition for Event-Oriented Relation Recognition
Yu Hong, Yang Xu, Huibin Ruan, Bowei Zou, Jianmin Yao and Guodong Zhou

Representation Learning of Entities and Documents from Knowledge Base Descriptions
Ikuya Yamada, Hiroyuki Shindo and Yoshiyasu Takefuji

Simple Neologism Based Domain Independent Models to Predict Year of Authorship
Vivek Kulkarni, Yingtao Tian, Parth Dandiwala and Steve Skiena

Neural Math Word Problem Solver with Reinforcement Learning
Danqing Huang, Jing Liu, Chin-Yew Lin and Jian Yin

Personalizing Lexical Simplification
John Lee and Chak Yan Yeung

From Text to Lexicon: Bridging the Gap between Word Embeddings and Lexical Resources
Ilia Kuznetsov and Iryna Gurevych

Lexi: A tool for adaptive, personalized text simplification
Joachim Bingel, Gustavo Paetzold and Anders Søgaard

Identifying Emergent Research Trends by Key Authors and Phrases
Shenhao Jiang, Animesh Prasad, Min-Yen Kan and Kazunari Sugiyama

Embedding WordNet Knowledge for Textual Entailment
Yunshi Lan and Jing Jiang

Tuesday 21st August (continued)

Attributed and Predictive Entity Embedding for Fine-Grained Entity Typing in Knowledge Bases
Hailong Jin, Lei Hou, Juanzi Li and Tiansi Dong

Joint Learning from Labeled and Unlabeled Data for Information Retrieval
Bo Li, Ping Cheng and Le Jia

Modeling the Readability of German Targeting Adults and Children: An empirically broad analysis and its cross-corpus validation
Zarah Weiß and Detmar Meurers

Automatic Assessment of Conceptual Text Complexity Using Knowledge Graphs
Sanja Štajner and Ioana Hulpus

Par4Sim – Adaptive Paraphrasing for Text Simplification
Seid Muhie Yimam and Chris Biemann

Topic or Style? Exploring the Most Useful Features for Authorship Attribution
Yunita Sari, Mark Stevenson and Andreas Vlachos

A Deep Dive into Word Sense Disambiguation with LSTM
Minh Le, Marten Postma, Jacopo Urbani and Piek Vossen

Enriching Word Embeddings with Domain Knowledge for Readability Assessment
Zhiwei Jiang, Qing Gu, Yafeng Yin and Daoxu Chen

WikiRef: Wikilinks as a route to recommending appropriate references for scientific Wikipedia pages
Abhik Jana, Pranjal Kanojiya, Pawan Goyal and Animesh Mukherjee

Authorship Identification for Literary Book Recommendations
Haifa Alharthi, Diana Inkpen and Stan Szpakowicz

A Nontrivial Sentence Corpus for the Task of Sentence Readability Assessment in Portuguese
Sidney Evaldo Leal, Magali Sanches Duran and Sandra Maria Aluísio

Adopting the Word-Pair-Dependency-Triplets with Individual Comparison for Natural Language Inference
Qianlong Du, Chengqing Zong and Keh-Yih Su

Cooperative Denoising for Distantly Supervised Relation Extraction
Kai Lei, Daoyuan Chen, Yaliang Li, Nan Du, Min Yang, Wei Fan and Ying Shen

Adversarial Feature Adaptation for Cross-lingual Relation Classification
Bowei Zou, Zengzhuang Xu, Yu Hong and Guodong Zhou

One-shot Learning for Question-Answering in Gaokao History Challenge
Zhuosheng Zhang and Hai Zhao

Dynamic Multi-Level Multi-Task Learning for Sentence Simplification
Han Guo, Ramakanth Pasunuru and Mohit Bansal

Interpretation of Implicit Conditions in Database Search Dialogues
Shunya Fukunaga, Hitoshi Nishikawa, Takenobu Tokunaga, Hikaru Yokono and Tetsuro Takahashi

Few-Shot Charge Prediction with Discriminative Legal Attributes
Zikun Hu, Xiang Li, Cunchao Tu, Zhiyuan Liu and Maosong Sun

Can Taxonomy Help? Improving Semantic Question Matching using Question Taxonomy
Deepak Gupta, Rajkumar Pujari, Asif Ekbal, Pushpak Bhattacharyya, Anutosh Maitra, Tom Jain and Shubhashis Sengupta

Natural Language Interface for Databases Using a Dual-Encoder Model
Ionel Alexandru Hosu, Radu Cristian Alexandru Iacob, Florin Brad, Stefan Ruseti and Traian Rebedea

Tuesday 21st August (continued)

12:20–13:50 Lunch

13:50–15:50 Session 1-2-a: Discourse relations

Employing Text Matching Network to Recognise Nuclearity in Chinese Discourse
Sheng Xu, Peifeng Li, Guodong Zhou and Qiaoming Zhu

Joint Modeling of Structure Identification and Nuclearity Recognition in Macro Chinese Discourse Treebank
Xiaomin Chu, Feng Jiang, Yi Zhou, Guodong Zhou and Qiaoming Zhu

Implicit Discourse Relation Recognition using Neural Tensor Network with Interactive Attention and Sparse Learning
Fengyu Guo, Ruifang He, Di Jin, Jianwu Dang, Longbiao Wang and Xiangang Li

Transition-based Neural RST Parsing with Implicit Syntax Features
Nan Yu, Meishan Zhang and Guohong Fu

Deep Enhanced Representation for Implicit Discourse Relation Recognition
Hongxiao Bai and Hai Zhao

A Knowledge-Augmented Neural Network Model for Implicit Discourse Relation Classification
Yudai Kishimoto, Yugo Murawaki and Sadao Kurohashi

Tuesday 21st August (continued)

13:50–15:50 Session 1-2-b: Machine translation

Modeling Coherence for Neural Machine Translation with Dynamic and Topic Caches
Shaohui Kuang, Deyi Xiong, Weihua Luo and Guodong Zhou

Fusing Recency into Neural Machine Translation with an Inter-Sentence Gate Model
Shaohui Kuang and Deyi Xiong

Improving Neural Machine Translation by Incorporating Hierarchical Subword Features
Makoto Morishita, Jun Suzuki and Masaaki Nagata

Design Challenges in Named Entity Transliteration
Yuval Merhav and Stephen Ash

A Comparison of Transformer and Recurrent Neural Networks on Multilingual Neural Machine Translation
Surafel Melaku Lakew, Mauro Cettolo and Marcello Federico

On Adversarial Examples for Character-Level Neural Machine Translation
Javid Ebrahimi, Daniel Lowd and Dejing Dou

13:50–15:50 Session 1-2-c: Named entities

Systematic Study of Long Tail Phenomena in Entity Linking
Filip Ilievski, Piek Vossen and Stefan Schlobach

Neural Collective Entity Linking
Yixin Cao, Lei Hou, Juanzi Li and Zhiyuan Liu

Exploiting Structure in Representation of Named Entities using Active Learning
Nikita Bhutani, Kun Qian, Yunyao Li, H. V. Jagadish, Mauricio Hernandez and Mitesh Vasa

A Practical Incremental Learning Framework For Sparse Entity Extraction
Hussein Al-Olimat, Steven Gustafson, Jason Mackay, Krishnaprasad Thirunarayan and Amit Sheth

Tuesday 21st August (continued)

An Empirical Study on Fine-Grained Named Entity Recognition
Khai Mai, Thai-Hoang Pham, Minh Trung Nguyen, Nguyen Tuan Duc, Danushka Bollegala, Ryohei Sasano and Satoshi Sekine

Does Higher Order LSTM Have Better Accuracy for Segmenting and Labeling Sequence Data?
Yi Zhang, Xu Sun, Shuming Ma, Yang Yang and Xuancheng Ren

13:50–15:50 Session 1-2-posters: Sentiment, NLG, understanding

Ant Colony System for Multi-Document Summarization
Asma Al-Saleh and Mohamed El Bachir Menai

Multi-task dialog act and sentiment recognition on Mastodon
Christophe Cerisara, Somayeh Jafaritazehjani, Adedayo Oluokun and Hoa T. Le

RuSentiment: An Enriched Sentiment Analysis Dataset for Social Media in Russian
Anna Rogers, Alexey Romanov, Anna Rumshisky, Svitlana Volkova, Mikhail Gronas and Alex Gribov

Self-Normalization Properties of Language Modeling
Jacob Goldberger and Oren Melamud

A Position-aware Bidirectional Attention Network for Aspect-level Sentiment Analysis
Shuqin Gu, Lipeng Zhang, Yuexian Hou and Yin Song

Dynamic Feature Selection with Attention in Incremental Parsing
Ryosuke Kohita, Hiroshi Noji and Yuji Matsumoto

Vocabulary Tailored Summary Generation
Kundan Krishna, Aniket Murhekar, Saumitra Sharma and Balaji Vasan Srinivasan

Reading Comprehension with Graph-based Temporal-Casual Reasoning
Yawei Sun, Gong Cheng and Yuzhong Qu

Projecting Embeddings for Domain Adaption: Joint Modeling of Sentiment Analysis in Diverse Domains
Jeremy Barnes, Roman Klinger and Sabine Schulte im Walde

Tuesday 21st August (continued)

Cross-lingual Argumentation Mining: Machine Translation (and a bit of Projection) is All You Need!
Steffen Eger, Johannes Daxenberger, Christian Stab and Iryna Gurevych

HL-EncDec: A Hybrid-Level Encoder-Decoder for Neural Response Generation
Sixing Wu, Dawei Zhang, Ying Li, Xing Xie and Zhonghai Wu

Multi-Perspective Context Aggregation for Semi-supervised Cloze-style Reading Comprehension
Liang Wang, Sujian Li, Wei Zhao, Kewei Shen, Meng Sun, Ruoyu Jia and Jingming Liu

A Lexicon-Based Supervised Attention Model for Neural Sentiment Analysis
Yicheng Zou, Tao Gui, Qi Zhang and Xuanjing Huang

Open-Domain Event Detection using Distant Supervision
Jun Araki and Teruko Mitamura

Semi-Supervised Lexicon Learning for Wide-Coverage Semantic Parsing
Bo Chen, Bo An, Le Sun and Xianpei Han

Summarization Evaluation in the Absence of Human Model Summaries Using the Compositionality of Word Embeddings
Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond Wong and Fang Chen

A review of Spanish corpora annotated with negation
Salud María Jiménez-Zafra, Roser Morante, Maite Martin and L. Alfonso Urena Lopez

Document-level Multi-aspect Sentiment Classification by Jointly Modeling Users, Aspects, and Overall Ratings
Junjie Li, Haitong Yang and Chengqing Zong

Leveraging Meta-Embeddings for Bilingual Lexicon Extraction from Specialized Comparable Corpora
Amir Hazem and Emmanuel Morin

Learning Emotion-enriched Word Representations
Ameeta Agrawal, Aijun An and Manos Papagelis

Evaluating the text quality, human likeness and tailoring component of PASS: A Dutch data-to-text system for soccer
Chris van der Lee, Bart Verduijn, Emiel Krahmer and Sander Wubben

Tuesday 21st August (continued)

Answerable or Not: Devising a Dataset for Extending Machine Reading Comprehension
Mao Nakanishi, Tetsunori Kobayashi and Yoshihiko Hayashi

Style Obfuscation by Invariance
Chris Emmery, Enrique Manjavacas Arevalo and Grzegorz Chrupała

Encoding Sentiment Information into Word Vectors for Sentiment Analysis
Zhe Ye, Fang Li and Timothy Baldwin

Multi-Task Neural Models for Translating Between Styles Within and Across Languages
Xing Niu, Sudha Rao and Marine Carpuat

Towards a Language for Natural Language Treebank Transductions
Carlos A. Prolo

Generating Reasonable and Diversified Story Ending Using Sequence to Sequence Model with Adversarial Training
Zhongyang Li, Xiao Ding and Ting Liu

Point Precisely: Towards Ensuring the Precision of Data in Generated Texts Using Delayed Copy Mechanism
Liunian Li and Xiaojun Wan

Enhancing General Sentiment Lexicons for Domain-Specific Use
Tim Kreutz and Walter Daelemans

An Operation Network for Abstractive Sentence Compression
Naitong Yu, Jie Zhang, Minlie Huang and Xiaoyan Zhu

Enhanced Aspect Level Sentiment Classification with Auxiliary Memory
Peisong Zhu and Tieyun Qian

Tuesday 21st August (continued)

15:50–16:20 Refreshment break

16:20–18:00 Session 1-3-a: Aspect-based sentiment, Ethics

Author Profiling for Abuse Detection
Pushkar Mishra, Marco Del Tredici, Helen Yannakoudakis and Ekaterina Shutova

Automated Scoring: Beyond Natural Language Processing
Nitin Madnani and Aoife Cahill

Aspect and Sentiment Aware Abstractive Review Summarization
Min Yang, Qiang Qu, Ying Shen, Qiao Liu, Wei Zhao and Jia Zhu

Effective Attention Modeling for Aspect-Level Sentiment Classification
Ruidan He, Wee Sun Lee, Hwee Tou Ng and Daniel Dahlmeier

Bringing replication and reproduction together with generalisability in NLP: Three reproduction studies for Target Dependent Sentiment Analysis
Andrew Moore and Paul Rayson

16:20–18:00 Session 1-3-b: Relation extraction, Summarization

Multilevel Heuristics for Rationale-Based Entity Relation Classification in Sentences
Shiou Tian Hsu, Mandar Chaudhary and Nagiza Samatova

Adversarial Multi-lingual Neural Relation Extraction
Xiaozhi Wang, Xu Han, Yankai Lin, Zhiyuan Liu and Maosong Sun

Neural Relation Classification with Text Descriptions
Feiliang Ren, Di Zhou, Zhihui Liu, Yongcheng Li, Rongsheng Zhao, Yongkang Liu and Xiaobo Liang

Abstract Meaning Representation for Multi-Document Summarization
Kexin Liao, Logan Lebanoff and Fei Liu

Abstractive Unsupervised Multi-Document Summarization using Paraphrastic Sentence Fusion
Mir Tafseer Nayeem, Tanvir Ahmed Fuad and Yllias Chali

16:20–18:00 Session 1-3-c: Dialogue systems

Adversarial Domain Adaptation for Variational Neural Language Generation in Dialogue Systems
Van-Khanh Tran and Le-Minh Nguyen

Ask No More: Deciding when to guess in referential visual dialogue
Ravi Shekhar, Tim Baumgärtner, Aashish Venkatesh, Elia Bruni, Raffaella Bernardi and Raquel Fernández

Sequence-to-Sequence Data Augmentation for Dialogue Language Understanding
Yutai Hou, Yijia Liu, Wanxiang Che and Ting Liu

Dialogue-act-driven Conversation Model : An Experimental Study
Harshit Kumar, Arvind Agarwal and Sachindra Joshi

Structured Dialogue Policy with Graph Neural Networks
Lu Chen, Bowen Tan, Sishan Long and Kai Yu

16:20–18:00 Session 1-3-posters: Translation, Variation

JTAV: Jointly Learning Social Media Content Representation by Fusing Textual, Acoustic, and Visual Features
Hongru Liang, Haozheng Wang, Jun Wang, Shaodi You, Zhe Sun, Jin-Mao Wei and Zhenglu Yang

MEMD: A Diversity-Promoting Learning Framework for Short-Text Conversation
Meng Zou, Xihan Li, Haokun Liu and Zhihong Deng

Refining Source Representations with Relation Networks for Neural Machine Translation
Wen Zhang, Jiawei Hu, Yang Feng and Qun Liu

A Survey of Domain Adaptation for Neural Machine Translation
Chenhui Chu and Rui Wang

An Evaluation of Neural Machine Translation Models on Historical Spelling Normalization
Gongbo Tang, Fabienne Cap, Eva Pettersson and Joakim Nivre

Fine-Grained Arabic Dialect Identification
Mohammad Salameh and Houda Bouamor

Who Feels What and Why? Annotation of a Literature Corpus with Semantic Roles of Emotions
Evgeny Kim and Roman Klinger

Local String Transduction as Sequence Labeling
Joana Ribeiro, Shashi Narayan, Shay B. Cohen and Xavier Carreras

Deep Neural Networks at the Service of Multilingual Parallel Sentence Extraction
Ahmad Aghaebrahimian

Diachronic word embeddings and semantic shifts: a survey
Andrey Kutuzov, Lilja Øvrelid, Terrence Szymanski and Erik Velldal

Interaction-Aware Topic Model for Microblog Conversations through Network Embedding and User Attention
Ruifang He, Xuefei Zhang, Di Jin, Longbiao Wang, Jianwu Dang and Xiangang Li

Cross-media User Profiling with Joint Textual and Social User Embedding
Jingjing Wang, Shoushan Li, Mingqi Jiang, Hanqian Wu and Guodong Zhou

Incorporating Syntactic Uncertainty in Neural Machine Translation with a Forest-to-Sequence Model
Poorya Zaremoodi and Gholamreza Haffari

Ensure the Correctness of the Summary: Incorporate Entailment Knowledge into Abstractive Sentence Summarization
Haoran Li, Junnan Zhu, Jiajun Zhang and Chengqing Zong

Extracting Parallel Sentences with Bidirectional Recurrent Neural Networks to Improve Machine Translation
Francis Grégoire and Philippe Langlais

Fast and Accurate Reordering with ITG Transition RNN
Hao Zhang, Axel Ng and Richard Sproat

Tuesday 21st August (continued)

Neural Machine Translation with Decoding History Enhanced Attention
Mingxuan Wang, Jun Xie, Zhixing Tan, Jinsong Su, Deyi Xiong and Chao Bian

Transfer Learning for a Letter-Ngrams to Word Decoder in the Context of Historical Handwriting Recognition with Scarce Resources
Adeline Granet, Emmanuel Morin, Harold Mouchère, Solen Quiniou and Christian Viard-Gaudin

SMHD: a Large-Scale Resource for Exploring Online Language Usage for Multiple Mental Health Conditions
Arman Cohan, Bart Desmet, Andrew Yates, Luca Soldaini, Sean MacAvaney and Nazli Goharian

Crowdsourcing a Large Corpus of Clickbait on Twitter
Martin Potthast, Tim Gollub, Kristof Komlossy, Sebastian Schuster, Matti Wiegmann, Erika Patricia Garces Fernandez, Matthias Hagen and Benno Stein

Cross-lingual Knowledge Projection Using Machine Translation and Target-side Knowledge Base Completion
Naoki Otani, Hirokazu Kiyomaru, Daisuke Kawahara and Sadao Kurohashi

Assessing Quality Estimation Models for Sentence-Level Prediction
Hoang Cuong and Jia Xu

User-Level Race and Ethnicity Predictors from Twitter Text
Daniel Preoţiuc-Pietro and Lyle Ungar

Multi-Source Multi-Class Fake News Detection
Hamid Karimi, Proteek Roy, Sari Saba-Sadiya and Jiliang Tang

Killing Four Birds with Two Stones: Multi-Task Learning for Non-Literal Language Detection
Erik-Lân Do Dinh, Steffen Eger and Iryna Gurevych

Twitter corpus of Resource-Scarce Languages for Sentiment Analysis and Multilingual Emoji Prediction
Nurendra Choudhary, Rajat Singh, Vijjini Anvesh Rao and Manish Shrivastava

Wednesday 22nd August

09:00–10:00 Invited talk: Fabiola Henri

10:00–10:30 Refreshment break

10:30–11:50 Session 2-1-a: Language change, Historical linguistics

Towards identifying the optimal datasize for lexically-based Bayesian inference of linguistic phylogenies
Taraka Rama and Søren Wichmann

The Road to Success: Assessing the Fate of Linguistic Innovations in Online Communities
Marco Del Tredici and Raquel Fernández

Ab Initio: Automatic Latin Proto-word Reconstruction
Alina Maria Ciobanu and Liviu P. Dinu

A Computational Model for the Linguistic Notion of Morphological Paradigm
Miikka Silfverberg, Ling Liu and Mans Hulden

10:30–11:50 Session 2-1-b: Embedding creation

Relation Induction in Word Embeddings Revisited
Zied Bouraoui, Shoaib Jameel and Steven Schockaert

Contextual String Embeddings for Sequence Labeling
Alan Akbik, Duncan Blythe and Roland Vollgraf

Learning Word Meta-Embeddings by Autoencoding
Danushka Bollegala and Cong Bao

GenSense: A Generalized Sense Retrofitting Model
Yang-Yin Lee, Ting-Yu Yen, Hen-Hsen Huang, Yow-Ting Shiue and Hsin-Hsi Chen

Wednesday 22nd August (continued)

10:30–11:50 Session 2-1-c: ML methods

Variational Attention for Sequence-to-Sequence Models
Hareesh Bahuleyan, Lili Mou, Olga Vechtomova and Pascal Poupart

A New Concept of Deep Reinforcement Learning based Augmented General Tagging System
Yu Wang, Abhishek Patel and Hongxia Jin

Learning from Measurements in Crowdsourcing Models: Inferring Ground Truth from Diverse Annotation Types
Paul Felt, Eric Ringger, Jordan Boyd-Graber and Kevin Seppi

Reproducing and Regularizing the SCRN Model
Olzhas Kabdolov, Zhenisbek Assylbekov and Rustem Takhanov

12:00 Lunch and excursion

Thursday 23rd August

09:00–10:00 Invited talk: Hannah Rohde

10:00–10:30 Refreshment break

Thursday 23rd August (continued)

10:30–12:10 Session 3-1-a: Generation

Structure-Infused Copy Mechanisms for Abstractive Summarization
Kaiqiang Song, Lin Zhao and Fei Liu

Measuring the Diversity of Automatic Image Descriptions
Emiel van Miltenburg, Desmond Elliott and Piek Vossen

Extractive Headline Generation Based on Learning to Rank for Community Question Answering
Tatsuru Higurashi, Hayato Kobayashi, Takeshi Masuyama and Kazuma Murao

A Multi-Attention based Neural Network with External Knowledge for Story Ending Predicting Task
Qian Li, Ziwei Li, Jin-Mao Wei, Yanhui Gu, Adam Jatowt and Zhenglu Yang

A Reinforcement Learning Framework for Natural Question Generation using Bi-discriminators
Zhihao Fan, Zhongyu Wei, Siyuan Wang, Yang Liu and Xuanjing Huang

10:30–12:10 Session 3-1-b: Embedding creation

Embedding Words as Distributions with a Bayesian Skip-gram Model
Arthur Bražinskas, Serhii Havrylov and Ivan Titov

Assessing Composition in Sentence Vector Representations
Allyson Ettinger, Ahmed Elgohary, Colin Phillips and Philip Resnik

Subword-augmented Embedding for Cloze Reading Comprehension
Zhuosheng Zhang, Yafang Huang and Hai Zhao

Enhancing Sentence Embedding with Generalized Pooling
Qian Chen, Zhen-Hua Ling and Xiaodan Zhu

Treat us like the sequences we are: Prepositional Paraphrasing of Noun Compounds using LSTM
Girishkumar Ponkiya, Kevin Patel, Pushpak Bhattacharyya and Girish Palshikar

Thursday 23rd August (continued)

10:30–12:10 Session 3-1-c: Humor, rumor, sarcasm and spam

CASCADE: Contextual Sarcasm Detection in Online Discussion Forums
Devamanyu Hazarika, Soujanya Poria, Sruthi Gorantla, Erik Cambria, Roger Zimmermann and Rada Mihalcea

Recognizing Humour using Word Associations and Humour Anchor Extraction
Andrew Cattle and Xiaojuan Ma

A Retrospective Analysis of the Fake News Challenge Stance-Detection Task
Andreas Hanselowski, Avinesh PVS, Benjamin Schiller, Felix Caspelherr, Debanjan Chaudhuri, Christian M. Meyer and Iryna Gurevych

Exploiting Syntactic Structures for Humor Recognition
Lizhen Liu, Donghai Zhang and Wei Song

An Attribute Enhanced Domain Adaptive Model for Cold-Start Spam Review Detection
Zhenni You, Tieyun Qian and Bing Liu

10:30–12:10 Session 3-1-posters: Entities, QA and classification

Robust Lexical Features for Improved Neural Network Named-Entity Recognition
Abbas Ghaddar and Phillippe Langlais

A Pseudo Label based Dataless Naive Bayes Algorithm for Text Classification with Seed Words
Ximing Li and Bo Yang

Visual Question Answering Dataset for Bilingual Image Understanding: A Study of Cross-Lingual Transfer Using Attention Maps
Nobuyuki Shimizu, Na Rong and Takashi Miyazaki

Style Detection for Free Verse Poetry from Text and Speech
Timo Baumann, Hussein Hussein and Burkhard Meyer-Sickendiek

A Neural Question Answering Model Based on Semi-Structured Tables
Hao Wang, Xiaodong Zhang, Shuming Ma, Xu Sun, Houfeng Wang and Mengxiang Wang

Thursday 23rd August (continued)

LCQMC:A Large-scale Chinese Question Matching Corpus
Xin Liu, Qingcai Chen, Chong Deng, Huajun Zeng, Jing Chen, Dongfang Li and Buzhou Tang

Genre Identification and the Compositional Effect of Genre in Literature
Joseph Worsham and Jugal Kalita

Transfer Learning for Entity Recognition of Novel Classes
Juan Diego Rodriguez, Adam Caldwell and Alexander Liu

Location Name Extraction from Targeted Text Streams using Gazetteer-based Statistical Language Models
Hussein Al-Olimat, Krishnaprasad Thirunarayan, Valerie Shalin and Amit Sheth

The APVA-TURBO Approach To Question Answering in Knowledge Base
Yue Wang, Richong Zhang, Cheng Xu and Yongyi Mao

An Interpretable Reasoning Network for Multi-Relation Question Answering
Mantong Zhou, Minlie Huang and Xiaoyan Zhu

Task-oriented Word Embedding for Text Classification
Qian Liu, Heyan Huang, Yang Gao, Xiaochi Wei, Yuxin Tian and Luyang Liu

Adaptive Learning of Local Semantic and Global Structure Representations for Text Classification
Jianyu Zhao, Zhiqiang Zhan, Qichuan Yang, Yang Zhang, Changjian Hu, Zhensheng Li, Liuxin Zhang and Zhiqiang He

Lyrics Segmentation: Textual Macrostructure Detection using Convolutions
Michael Fell, Yaroslav Nechaev, Elena Cabrio and Fabien Gandon

Learning What to Share: Leaky Multi-Task Network for Text Classification
Liqiang Xiao, Honglun Zhang, Wenqing Chen, Yongkun Wang and Yaohui Jin

Towards an argumentative content search engine using weak supervision
Ran Levy, Ben Bogin, Shai Gretz, Ranit Aharonov and Noam Slonim

Improving Named Entity Recognition by Jointly Learning to Disambiguate Morphological Tags
Onur Gungor, Suzan Uskudarli and Tunga Gungor

Thursday 23rd August (continued)

Farewell Freebase: Migrating the SimpleQuestions Dataset to DBpedia
Michael Azmy, Peng Shi, Jimmy Lin and Ihab Ilyas

An Analysis of Annotated Corpora for Emotion Classification in Text
Laura Ana Maria Bostan and Roman Klinger

Investigating the Working of Text Classifiers
Devendra Sachan, Manzil Zaheer and Ruslan Salakhutdinov

A Review on Deep Learning Techniques Applied to Answer Selection
Tuan Manh Lai, Trung Bui and Sheng Li

A Survey on Recent Advances in Named Entity Recognition from Deep Learning models
Vikas Yadav and Steven Bethard

Distantly Supervised NER with Partial Annotation Learning and Reinforcement Learning
Yaosheng Yang, Wenliang Chen, Zhenghua Li, Zhengqiu He and Min Zhang

Joint Neural Entity Disambiguation with Output Space Search
Hamed Shahbazi, Xiaoli Fern, Reza Ghaeini, Chao Ma, Rasha Mohammad Obeidat and Prasad Tadepalli

Learning to Progressively Recognize New Named Entities with Sequence to Sequence Models
Lingzhen Chen and Alessandro Moschitti

Responding E-commerce Product Questions via Exploiting QA Collections and Reviews
Qian Yu, Wai Lam and Zihao Wang

Thursday 23rd August (continued)

12:10–13:40 Lunch

13:40–15:20 Session 3-2-a: Sentiment

Aff2Vec: Affect–Enriched Distributional Word Representations
Sopan Khosla, Niyati Chhaya and Kushal Chawla

Aspect-based summarization of pros and cons in unstructured product reviews
Florian Kunneman, Sander Wubben, Antal van den Bosch and Emiel Krahmer

Learning Sentiment Composition from Sentiment Lexicons
Orith Toledo-Ronen, Roy Bar-Haim, Alon Halfon, Charles Jochim, Amir Menczel, Ranit Aharonov and Noam Slonim

Representations and Architectures in Neural Sentiment Analysis for Morphologically Rich Languages: A Case Study from Modern Hebrew
Adam Amram, Anat Ben-David and Reut Tsarfaty

Scoring and Classifying Implicit Positive Interpretations: A Challenge of Class Imbalance
Chantal van Son, Roser Morante, Lora Aroyo and Piek Vossen

13:40–15:20 Session 3-2-b: IE

Exploratory Neural Relation Classification for Domain Knowledge Acquisition
Yan Fan, Chengyu Wang and Xiaofeng He

Who is Killed by Police: Introducing Supervised Attention for Hierarchical LSTMs
Minh Nguyen and Thien Nguyen

Open Information Extraction from Conjunctive Sentences
Swarnadeep Saha and Mausam -

Graphene: Semantically-Linked Propositions in Open Information Extraction
Matthias Cetto, Christina Niklaus, André Freitas and Siegfried Handschuh

Thursday 23rd August (continued)

An Exploration of Three Lightly-supervised Representation Learning Approaches for Named Entity Classification
Ajay Nagesh and Mihai Surdeanu

13:40–15:20 Session 3-2-c: Multimodal processing, ASR, NLI

Multimodal Grounding for Language Processing
Lisa Beinborn, Teresa Botschen and Iryna Gurevych

Stress Test Evaluation for Natural Language Inference
Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose and Graham Neubig

Grounded Textual Entailment
Hoa Vu, Claudio Greco, Aliia Erofeeva, Somayeh Jafaritazehjan, Guido Linders, Marc Tanti, Alberto Testoni, Raffaella Bernardi and Albert Gatt

Recurrent One-Hop Predictions for Reasoning over Knowledge Graphs
Wenpeng Yin, Yadollah Yaghoobzadeh and Hinrich Schütze

Hybrid Attention based Multimodal Network for Spoken Language Classification
Yue Gu, Kangning Yang, Shiyu Fu, Shuhong Chen, Xinyu Li and Ivan Marsic

13:40–15:20 Session 3-2-posters: Distributional information

Exploring the Influence of Spelling Errors on Lexical Variation Measures
Ryo Nagata, Taisei Sato and Hiroya Takamura

Stance Detection with Hierarchical Attention Network
Qingying Sun, Zhongqing Wang, Qiaoming Zhu and Guodong Zhou

Correcting Chinese Word Usage Errors for Learning Chinese as a Second Language
Yow-Ting Shiue, Hen-Hsen Huang and Hsin-Hsi Chen

Retrofitting Distributional Embeddings to Knowledge Graphs with Functional Relations
Ben Lengerich, Andrew Maas and Christopher Potts

Context-Sensitive Generation of Open-Domain Conversational Responses
Weinan Zhang, Yiming Cui, Yifa Wang, Qingfu Zhu, Lingzhi Li, Lianqiang Zhou and Ting Liu

A LSTM Approach with Sub-Word Embeddings for Mongolian Phrase Break Prediction
Rui Liu, Feilong Bao, Guanglai Gao, Hui Zhang and Yonghe Wang

Synonymy in Bilingual Context: The CzEngClass Lexicon
Zdenka Uresova, Eva Fucikova, Eva Hajicova and Jan Hajic

Convolutional Neural Network for Universal Sentence Embeddings
Xiaoqi Jiao, Fang Wang and Dan Feng

Rich Character-Level Information for Korean Morphological Analysis and Part-of-Speech Tagging
Andrew Matteson, Chanhee Lee, Youngbum Kim and Heuiseok Lim

Why does PairDiff work? - A Mathematical Analysis of Bilinear Relational Compositional Operators for Analogy Detection
Huda Hakami, Kohei Hayashi and Danushka Bollegala

Real-time Change Point Detection using On-line Topic Models
Yunli Wang and Cyril Goutte

Automatically Creating a Lexicon of Verbal Polarity Shifters: Mono- and Cross-lingual Methods for German
Marc Schulder, Michael Wiegand and Josef Ruppenhofer

Part-of-Speech Tagging on an Endangered Language: a Parallel Griko-Italian Resource
Antonios Anastasopoulos, Marika Lekakou, Josep Quer, Eleni Zimianiti, Justin DeBenedetto and David Chiang

One vs. Many QA Matching with both Word-level and Sentence-level Attention Network
Lu Wang, Shoushan Li, Changlong Sun, Luo Si, Xiaozhong Liu, Min Zhang and Guodong Zhou

Learning to Generate Word Representations using Subword Information
Yeachan Kim, Kang-Min Kim, Ji-Min Lee and SangKeun Lee

Urdu Word Segmentation using Conditional Random Fields (CRFs)
Haris Bin Zia, Agha Ali Raza and Awais Athar

Thursday 23rd August (continued)

ReSyf: a French lexicon with ranked synonyms
Mokhtar Boumedyen BILLAMI, Thomas François and Nuria Gala

If you've seen some, you've seen them all: Identifying variants of multiword expressions
Caroline Pasquer, Agata Savary, Carlos Ramisch and Jean-Yves Antoine

Learning Multilingual Topics from Incomparable Corpora
Shudong Hao and Michael J. Paul

Using Word Embeddings for Unsupervised Acronym Disambiguation
Jean Charbonnier and Christian Wartena

Indigenous language technologies in Canada: Assessment, challenges, and successes
Patrick Littell, Anna Kazantseva, Roland Kuhn, Aidan Pine, Antti Arppe, Christopher Cox and Marie-Odile Junker

Pluralizing Nouns across Agglutinating Bantu Languages
Joan Byamugisha, C. Maria Keet and Brian DeRenzi

Automatically Extracting Qualia Relations for the Rich Event Ontology
Ghazaleh Kazeminejad, Claire Bonial, Susan Windisch Brown and Martha Palmer

SeVeN: Augmenting Word Embeddings with Unsupervised Relation Vectors
Luis Espinosa Anke and Steven Schockaert

Evaluation of Unsupervised Compositional Representations
Hanan Aldarmaki and Mona Diab

Using Formulaic Expressions in Writing Assistance Systems
Kenichi Iwatsuki and Akiko Aizawa

What's in Your Embedding, And How It Predicts Task Performance
Anna Rogers, Shashwath Hosur Ananthakrishna and Anna Rumshisky

Word Sense Disambiguation Based on Word Similarity Calculation Using Word Vector Representation from a Knowledge-based Graph
Dongsuk O, Sunjae Kwon, Kyungsun Kim and Youngjoong Ko

Thursday 23rd August (continued)

Learning Semantic Sentence Embeddings using Sequential Pair-wise Discriminator
Badri Narayana Patro, Vinod Kumar Kurmi, Sandeep Kumar and Vinay Namboodiri

A Reassessment of Reference-Based Grammatical Error Correction Metrics
Shamil Chollampatt and Hwee Tou Ng

Information Aggregation via Dynamic Routing for Sequence Encoding
Jingjing Gong, Xipeng Qiu, Shaojing Wang and Xuanjing Huang

A Full End-to-End Semantic Role Labeler, Syntactic-agnostic Over Syntactic-aware?
Jiaxun Cai, Shexia He, Zuchao Li and Hai Zhao

15:20–15:50 Refreshment break

15:50–17:30 Session 3-3-a: Applications

Authorship Attribution By Consensus Among Multiple Features
Jagadeesh Patchala and Raj Bhatnagar

Modeling with Recurrent Neural Networks for Open Vocabulary Slots
Jun-Seong Kim, Junghoe Kim, SeungUn Park, Kwangyong Lee and Yoonju Lee

Challenges and Opportunities of Applying Natural Language Processing in Business Process Management
Han Van der Aa, Josep Carmona, Henrik Leopold, Jan Mendling and Lluís Padró

Novelty Goes Deep. A Deep Neural Solution To Document Level Novelty Detection
Tirthankar Ghosal, Vignesh Edithal, Asif Ekbal, Pushpak Bhattacharyya, George Tsatsaronis and Srinivasa Satya Sameer Kumar Chivukula

What represents "style" in authorship attribution?
Kalaivani Sundararajan and Damon Woodard

15:50–17:30 Session 3-3-b: Distributional semantics

Learning Target-Specific Representations of Financial News Documents For Cumulative Abnormal Return Prediction
Junwen Duan, Yue Zhang, Xiao Ding, Ching-Yun Chang and Ting Liu

Model-Free Context-Aware Word Composition
Bo An, Xianpei Han and Le Sun

Learning Features from Co-occurrences: A Theoretical Analysis
Yanpeng Li

Towards a unified framework for bilingual terminology extraction of single-word and multi-word terms
Jingshu Liu, Emmanuel Morin and Peña Saldarriaga

Neural Activation Semantic Models: Computational lexical semantic models of localized neural activations
Nikos Athanasiou, Elias Iosif and Alexandros Potamianos

15:50–17:30 Session 3-3-c: Emotion

Folksonomication: Predicting Tags for Movies from Plot Synopses using Emotion Flow Encoded Neural Network
Sudipta Kar, Suraj Maharjan and Thamar Solorio

Emotion Representation Mapping for Automatic Lexicon Construction (Mostly) Performs on Human Level
Sven Buechel and Udo Hahn

Emotion Detection and Classification in a Multigenre Corpus with Joint Multi-Task Deep Learning
Shabnam Tafreshi and Mona Diab

How emotional are you? Neural Architectures for Emotion Intensity Prediction in Microblogs
Devang Kulshreshtha, Pranav Goel and Anil Kumar Singh

Expressively vulgar: The socio-dynamics of vulgarity and its effects on sentiment analysis in social media
Isabel Cachola, Eric Holgate, Daniel Preoţiuc-Pietro and Junyi Jessy Li

15:50–17:30 Session 3-3-posters: ML, parsing, MT

Clausal Modifiers in the Grammar Matrix
Kristen Howell and Olga Zamaraeva

Sliced Recurrent Neural Networks
Zeping Yu and Gongshen Liu

Multi-Task Learning for Sequence Tagging: An Empirical Study
Soravit Changpinyo, Hexiang Hu and Fei Sha

Using J-K-fold Cross Validation To Reduce Variance When Tuning NLP Models
Henry Moss, David Leslie and Paul Rayson

Incremental Natural Language Processing: Challenges, Strategies, and Evaluation
Arne Köhn

Gold Standard Annotations for Preposition and Verb Sense with Semantic Role Labels in Adult-Child Interactions
Lori Moon, Christos Christodoulopoulos, Fisher Cynthia, Sandra Franco and Dan Roth

Multi-layer Representation Fusion for Neural Machine Translation
Qiang Wang, Fuxue Li, Tong Xiao, Yanyang Li, Yinqiao Li and Jingbo Zhu

Toward Better Loanword Identification in Uyghur Using Cross-lingual Word Embeddings
Chenggang Mi, Yating Yang, Lei Wang, Xi Zhou and Tonghai Jiang

Adaptive Weighting for Neural Machine Translation
Yachao Li, Junhui Li and Min Zhang

Generic refinement of expressive grammar formalisms with an application to discontinuous constituent parsing
Kilian Gebhardt

Double Path Networks for Sequence to Sequence Learning
Kaitao Song, Xu Tan, Di He, Jianfeng Lu, Tao Qin and Tie-Yan Liu

Thursday 23rd August (continued)

An Empirical Investigation of Error Types in Vietnamese Parsing
Quy Nguyen, Yusuke Miyao, Hiroshi Noji and Nhung Nguyen

Learning with Noise-Contrastive Estimation: Easing training by learning to scale
Matthieu Labeau and Alexandre Allauzen

Parallel Corpora for bi-lingual English-Ethiopian Languages Statistical Machine Translation
Solomon Teferra Abate, Michael Melese, Martha Yifiru Tachbelie, Million Meshesha, Solomon Atinafu, Wondwossen Mulugeta, Yaregal Assibie, Hafte Abera, Binyam Ephrem, Tewodros Abebe, Wondimagegnhue Tsegaye, Amanuel Lemma, Tsegaye Andargie and Seifedin Shifaw

Multilingual Neural Machine Translation with Task-Specific Attention
Graeme Blackwood, Miguel Ballesteros and Todd Ward

Combining Information-Weighted Sequence Alignment and Sound Correspondence Models for Improved Cognate Detection
Johannes Dellert

Tailoring Neural Architectures for Translating from Morphologically Rich Languages
Peyman Passban, Andy Way and Qun Liu

deepQuest: A Framework for Neural-based Quality Estimation
Julia Ive, Frédéric Blain and Lucia Specia

Butterfly Effects in Frame Semantic Parsing: impact of data processing on model ranking
Alexandre Kabbach, Corentin Ribeyre and Aurélie Herbelot

Sensitivity to Input Order: Evaluation of an Incremental and Memory-Limited Bayesian Cross-Situational Word Learning Model
Sepideh Sadeghi and Matthias Scheutz

Sentence Weighting for Neural Machine Translation Domain Adaptation
Shiqi Zhang and Deyi Xiong

Quantifying training challenges of dependency parsers
Lauriane Aufrant, Guillaume Wisniewski and François Yvon

Seq2seq Dependency Parsing
Zuchao Li, Jiaxun Cai, Shexia He and Hai Zhao

Thursday 23rd August (continued)

Revisiting the Hierarchical Multiscale LSTM
Ákos Kádár, Marc-Alexandre Côté, Grzegorz Chrupała and Afra Alishahi

Character-Level Feature Extraction with Densely Connected Networks
Chanhee Lee, Young-Bum Kim, Dongyub Lee and Heuiseok Lim

Neural Machine Translation Incorporating Named Entity
Arata Ugawa, Akihiro Tamura, Takashi Ninomiya, Hiroya Takamura and Manabu Okumura

Semantic Parsing for Technical Support Questions
Abhirut Gupta, Anupama Ray, Gargi Dasgupta, Gautam Singh, Pooja Aggarwal and Prateeti Mohapatra

Deconvolution-Based Global Decoding for Neural Machine Translation
Junyang Lin, Xu Sun, Xuancheng Ren, Shuming Ma, Jinsong Su and Qi Su

Friday 24th August

09:00–10:00 Invited talk: Min-Yen Kan

10:00–10:30 Refreshment break

10:30–12:30 Session 4-1-a: Question answering

Pattern-revising Enhanced Simple Question Answering over Knowledge Bases
Yanchao Hao, Hao Liu, Shizhu He, Kang Liu and Jun Zhao

Integrating Question Classification and Deep Learning for improved Answer Selection
Harish Tayyar Madabushi, Mark Lee and John Barnden

Knowledge as A Bridge: Improving Cross-domain Answer Selection with External Knowledge
Yang Deng, Ying Shen, Min Yang, Yaliang Li, Nan Du, Wei Fan and Kai Lei

Modeling Semantics with Gated Graph Neural Networks for Knowledge Base Question Answering
Daniil Sorokin and Iryna Gurevych

Friday 24th August (continued)

Rethinking the Agreement in Human Evaluation Tasks
Jacopo Amidei, Paul Piwek and Alistair Willis

Dependent Gated Reading for Cloze-Style Question Answering
Reza Ghaeini, Xiaoli Fern, Hamed Shahbazi and Prasad Tadepalli

10:30–12:30 Session 4-1-b: Rumor

Automated Fact Checking: Task Formulations, Methods and Future Directions
James Thorne and Andreas Vlachos

Can Rumour Stance Alone Predict Veracity?
Sebastian Dungs, Ahmet Aker, Norbert Fuhr and Kalina Bontcheva

Attending Sentences to detect Satirical Fake News
Sohan De Sarkar, Fan Yang and Arjun Mukherjee

Predicting Stances from Social Media Posts using Factorization Machines
Akira Sasaki, Kazuaki Hanawa, Naoaki Okazaki and Kentaro Inui

Automatic Detection of Fake News
Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre and Rada Mihalcea

All-in-one: Multi-task Learning for Rumour Verification
Elena Kochkina, Maria Liakata and Arkaitz Zubiaga

Friday 24th August (continued)

10:30–12:30 Session 4-1-c: Second language, Biomedical

Open Information Extraction on Scientific Text: An Evaluation
Paul Groth, Mike Lauruhn, Antony Scerri and Ron Daniel, Jr.

Simple Algorithms For Sentiment Analysis On Sentiment Rich, Data Poor Domains.
Prathusha K Sarma and William Sethares

Word-Level Loss Extensions for Neural Temporal Relation Classification
Artuur Leeuwenberg and Marie-Francine Moens

Personalized Text Retrieval for Learners of Chinese as a Foreign Language
Chak Yan Yeung and John Lee

Punctuation as Native Language Interference
Ilia Markov, Vivi Nastase and Carlo Strapparava

Investigating Productive and Receptive Knowledge: A Profile for Second Language Learning
Leonardo Zilio, Rodrigo Wilkens and Cédrick Fairon

10:30–12:30 Session 4-1-posters: Dialog, discourse, argumentation and applications

iParaphrasing: Extracting Visually Grounded Paraphrases via an Image
Chenhui Chu, Mayu Otani and Yuta Nakashima

MCDTB: A Macro-level Chinese Discourse TreeBank
Feng Jiang, Sheng Xu, Xiaomin Chu, Peifeng Li, Qiaoming Zhu and Guodong Zhou

Corpus-based Content Construction
Balaji Vasan Srinivasan, Pranav Maneriker, Kundan Krishna and Natwar Modani

Bridging resolution: Task definition, corpus resources and rule-based experiments
Ina Roesiger, Arndt Riester and Jonas Kuhn

Friday 24th August (continued)

Semi-Supervised Disfluency Detection
Feng Wang, Wei Chen, Zhen Yang, Qianqian Dong, Shuang Xu and Bo Xu

ISO-Standard Domain-Independent Dialogue Act Tagging for Conversational Agents
Stefano Mezza, Alessandra Cervone, Evgeny Stepanov, Giuliano Tortoreto and Giuseppe Riccardi

Arrows are the Verbs of Diagrams
Malihe Alikhani and Matthew Stone

Improving Feature Extraction for Pathology Reports with Precise Negation Scope Detection
Olga Zamaraeva, Kristen Howell and Adam Rhine

Bridge Video and Text with Cascade Syntactic Structure
Guolong Wang, Zheng Qin, Kaiping Xu, Kai Huang and Shuxiong Ye

Multi-task and Multi-lingual Joint Learning of Neural Lexical Utterance Classification based on Partially-shared Modeling
Ryo Masumura, Tomohiro Tanaka, Ryuichiro Higashinaka, Hirokazu Masataki and Yushi Aono

Source Critical Reinforcement Learning for Transferring Spoken Language Understanding to a New Language
He Bai, Yu Zhou, Jiajun Zhang, Liang Zhao, Mei-Yuh Hwang and Chengqing Zong

A Prospective-Performance Network to Alleviate Myopia in Beam Search for Response Generation
Zongsheng Wang, Yunzhi Bai, Bowen Wu, Zhen Xu, Zhuoran Wang and Baoxun Wang

Adaptive Multi-Task Transfer Learning for Chinese Word Segmentation in Medical Text
Junjie Xing, Kenny Zhu and Shaodian Zhang

Addressee and Response Selection for Multilingual Conversation
Motoki Sato, Hiroki Ouchi and Yuta Tsuboi

Graph Based Decoding for Event Sequencing and Coreference Resolution
Zhengzhong Liu, Teruko Mitamura and Eduard Hovy

DIDEC: The Dutch Image Description and Eye-tracking Corpus
Emiel van Miltenburg, Ákos Kádár, Ruud Koolen and Emiel Krahmer

Friday 24th August (continued)

Narrative Schema Stability in News Text
Dan Simonson and Anthony Davis

NIPS Conversational Intelligence Challenge 2017 Winner System: Skill-based Conversational Agent with Supervised Dialog Manager
Idris Yusupov and Yurii Kuratov

AMR Beyond the Sentence: the Multi-sentence AMR corpus
Tim O'Gorman, Michael Regan, Kira Griffitt, Ulf Hermjakob, Kevin Knight and Martha Palmer

Incorporating Argument-Level Interactions for Persuasion Comments Evaluation using Co-attention Model
Lu Ji, Zhongyu Wei, Xiangkun Hu, Yang Liu, Qi Zhang and Xuanjing Huang

Learning Visually-Grounded Semantics from Contrastive Adversarial Samples
Haoyue Shi, Jiayuan Mao, Tete Xiao, Yuning Jiang and Jian Sun

Structured Representation Learning for Online Debate Stance Prediction
Chang Li, Aldo Porco and Dan Goldwasser

Modeling Multi-turn Conversation with Deep Utterance Aggregation
Zhuosheng Zhang, Jiangtong Li, Pengfei Zhu, Hai Zhao and Gongshen Liu

Argumentation Synthesis following Rhetorical Strategies
Henning Wachsmuth, Manfred Stede, Roxanne El Baff, Khalid Al Khatib, Maria Skeppstedt and Benno Stein

A Dataset for Building Code-Mixed Goal Oriented Conversation Systems
Suman Banerjee, Nikita Moghe, Siddhartha Arora and Mitesh M. Khapra

Sequence-to-Sequence Learning for Task-oriented Dialogue with Dialogue State Representation
Haoyang Wen, Yijia Liu, Wanxiang Che, Libo Qin and Ting Liu

Incorporating Deep Visual Features into Multiobjective based Multi-view Search Results Clustering
Sayantan Mitra, Mohammed Hasanuzzaman, Sriparna Saha and Andy Way

Integrating Tree Structures and Graph Structures with Neural Networks to Classify Discussion Discourse Acts
Yasuhide Miura, Ryuji Kano, Motoki Taniguchi, Tomoki Taniguchi, Shotaro Misawa and Tomoko Ohkuma

Friday 24th August (continued)

12:30–14:00 Lunch

14:00–15:20 Session 4-2-bp: Best papers opening

AnlamVer: Semantic Model Evaluation Dataset for Turkish - Word Similarity and Relatedness
Gökhan Ercan and Olcay Taner Yıldız

Arguments and Adjuncts in Universal Dependencies
Adam Przepiórkowski and Agnieszka Patejuk

Distinguishing affixoid formations from compounds
Josef Ruppenhofer, Michael Wiegand, Rebecca Wilm and Katja Markert

A Survey on Open Information Extraction
Christina Niklaus, Matthias Cetto, André Freitas and Siegfried Handschuh

15:20–15:50 Refreshment break

15:50–17:10 Session 4-3-bp: Best papers closing

Design Challenges and Misconceptions in Neural Sequence Labeling
Jie Yang, Shuailong Liang and Yue Zhang

Neural Network Models for Paraphrase Identification, Semantic Textual Similarity, Natural Language Inference, and Question Answering
Wuwei Lan and Wei Xu

Authorless Topic Models: Biasing Models Away from Known Structure
Laure Thompson and David Mimno

SGM: Sequence Generation Model for Multi-label Classification
Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu and Houfeng Wang

Friday 24th August (continued)

17:10–17:25 Closing ceremony

A New Approach to Animacy Detection

Labiba Jahan[1], **Geeticka Chauhan**[2], **Mark A. Finlayson**[1]
[1]School of Computing and Information Sciences
Florida International University, Miami, FL 33199
[2]Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology, Cambridge, MA
{ljaha002, markaf}@fiu.edu, geeticka@mit.edu

Abstract

Animacy is a necessary property for a referent to be an agent, and thus animacy detection is useful for a variety of natural language processing tasks, including word sense disambiguation, co-reference resolution, semantic role labeling, and others. Prior work treated animacy as a word-level property, and has developed statistical classifiers to classify words as either animate or inanimate. We discuss why this approach to the problem is ill-posed, and present a new approach based on classifying the animacy of co-reference chains. We show that simple voting approaches to inferring the animacy of a chain from its constituent words perform relatively poorly, and then present a hybrid system merging supervised machine learning (ML) and a small number of hand-built rules to compute the animacy of referring expressions and co-reference chains. This method achieves state of the art performance. The supervised ML component leverages features such as word embeddings over referring expressions, parts of speech, and grammatical and semantic roles. The rules take into consideration parts of speech and the hypernymy structure encoded in WordNet. The system achieves an F_1 of 0.88 for classifying the animacy of referring expressions, which is comparable to state of the art results for classifying the animacy of words, and achieves an F_1 of 0.75 for classifying the animacy of coreference chains themselves. We release our training and test dataset, which includes 142 texts (all narratives) comprising 156,154 words, 34,698 referring expressions, and 10,941 co-reference chains. We test the method on a subset of the OntoNotes dataset, showing using manual sampling that animacy classification is 90%$\pm$2% accurate for coreference chains, and 92%$\pm$1% for referring expressions. The data also contains 46 folktales, which present an interesting challenge because they often involve characters who are members of traditionally inanimate classes (e.g., stoves that walk, trees that talk). We show that our system is able to detect the animacy of these unusual referents with an F_1 of 0.95.

1 Introduction

Animacy is the characteristic of being able to independently carry out actions (e.g., movement, communication, etc.). For example, a person or a bird is animate because they move or communicate under their own power. On the other hand, a chair or a book is inanimate because they do not perform any kind of independent action.

Animacy is a useful semantic property for different NLP systems, including word sense disambiguation (WSD), semantic role labeling (SRL), coreference resolution, among many others. Animacy can be used to distinguish different senses and thus help a WSD systems assign senses to different words. As an example, animacy has been applied in grouping senses from WordNet (Palmer et al., 2004; Palmer et al., 2007). Animacy can also be used directly in a WSD system to decide thematic assignment, which is useful for assigning senses: for example, Carlson and Tanenhaus (1988) used the presence of an animate subject in a sentence to determine if a the verb is transitive, which is a useful for thematic role assignment. Another task where animacy can play an important role is semantic role labeling (SRL). Agentive

Proceedings of the 27th International Conference on Computational Linguistics, pages 1–12
Santa Fe, New Mexico, USA, August 20-26, 2018.

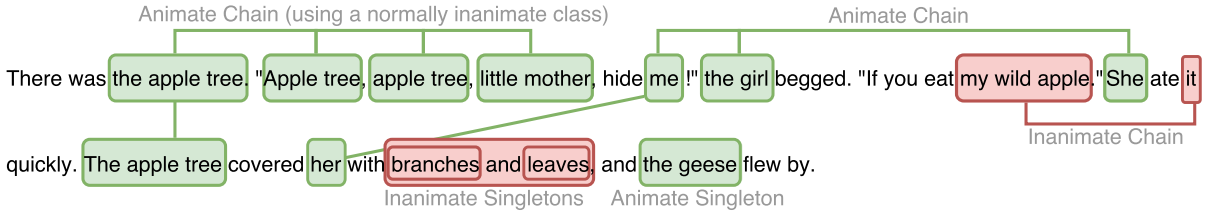


Figure 1: Example text containing animate and inanimate coreference chains. Colored boxes represent referring expressions, while links between them signify coreference. Animate chains are green, while inanimate chains are red. The text is drawn from Story #113 *The Magic Swan Geese* (Guterman, 1975, p. 350) and has been slightly modified for clarity. The figure is adapted from (Jahan et al., 2017).

or semantic subject roles must often be filled by animate entities, whereas goal, theme, patient, instrument and location roles are often filled by inanimate entities (Kittila et al., 2011). In some works (Connor et al., 2013; Kittilä, 2006, for example), animacy is used as a feature that helps to identify agents, and Ferreira (1994) showed how knowing the animacy of roles allows one to better identify the passive voice. In many coreference resolution systems (Raghunathan et al., 2010; Iida et al., 2003; Cardie and Wagstaf, 1999, for example) animacy is used as an semantic feature to determine co-referents of an expression.

In addition to these broad uses of animacy, our own research group is particularly interested in detecting animacy with a view toward identifying characters in stories. Most definitions of narrative acknowledge the central role of character, for example: "a representation of a possible world . . . at whose centre there are one or several protagonists of an *anthropomorphic* nature . . . who (mostly) perform goal-directed actions . . . " *(emphasis ours)* (Fludernik, 2009, p. 6). If we are to achieve the long-term goal of automatic story understanding, it is critical that we be able to automatically identify a story's characters, distinguishing them from non-character entities. All characters are necessarily animate—although not all animate things are necessarily characters—and so detecting animacy will immediately narrow the set of possibilities for character detection.

Prior work treated animacy as a word-level phenomenon, marking animacy as an independent feature on individual words (Orăsan and Evans, 2007; Bowman and Chopra, 2012; Karsdorp et al., 2015). But word-level animacy is not always sufficient to identify an animate or an inanimate object. For example, *horse* is normally animate, but a *dead horse* is obviously inanimate. On the other hand, *tree* is an inanimate word but a *talking tree* is definitely an animate thing. So, assigning animacy at the word level confuses the issue and makes it more difficult classify these type of complex cases.

Furthermore, referents are expressed in texts as coreference chains comprised of referring expressions, and so conceiving of animacy as a word-level phenomenon requires an additional method for computing chain animacy from word animacy. One way to do this is to combine word-level animacy markings—say, using majority vote—into referring expressions animacy and then coreference chains. As it turns out, this does not work all that well and we used this method as our baseline. Alternatively, we can attempt to compute animacy directly on the referring expressions and then use majority vote of referring-expression-level animacy to compute animacy of coreference chains, the approach we pursue here.

Although detecting animacy might seem to be straightforward, it presents a number of subtleties. For example, some theorists have proposed closed lists of linguistic expressions that should be automatically considered animate entities, such as titles, animals, or personal pronouns (Quirk et al., 1985; Yamamoto, 1999). However, texts, especially stories about unreal worlds, can arbitrarily introduce characters that would not be animate in real life, for example, walking stoves or talking trees. Figure 1 shows an example sentence from a Russian fairytale which contains three animate chains, one of which is a tree that talks: trees would not be normally be considered animate according to canonical lists of animate entities. Therefore some context sensitivity in detection is needed.

In our work, we compute animacy directly on referring expressions, and transfer those markings up to the coreference chain level, to get a direct classification of the animacy of the whole chain. We present a hybrid system combining statistical machine learning (ML) and hand-built rules for classifying the

Text Types	# Texts	# Tokens	# Referring Expressions	# Coref. Chains
Russian Folktales	46	109,120	20,391	4,950
Islamist Extremist Texts	32	26,557	8,041	3,684
Islamic Hadiths	64	20,477	6,266	2,307

Table 1: Counts of various text types in the corpus.

animacy of referring expression, and also present a voting model to identify the animacy of coreference chains based on the animacy of the chain's constituent referring expressions. The paper proceeds as follows. First we discuss our data sources and annotation procedures (§2). Next we discuss the experimental setup including the ML features, rules, and classification models (§3), and then describe our results (§4). We analyze the error patterns of the system and mention potential future work (§5), and also discuss work that is related to this study (§6). We finish with a discussion of the contributions of the paper (§7).

2 Data

We started this project seeking to use existing data annotated for animacy, as there have been a number of studies on animacy detection already (as we discuss in §6). However, no prior data in English was readily available to use; the best performing prior work on word-level animacy was done on a corpus of 74 stories comprising 74,504 words in Dutch (Karsdorp et al., 2015). Orăsan and Evans (2007) did their work in English but their data was not available. Therefore we sought other data (specifically stories, because of our interest in story understanding), and our annotated data was a corpus comprising a variety of Russian folktales, Islamist Extremist stories, and Islamic Hadiths that are freely available and assembled for other work, and had been annotated for referring expressions and coreference chains (Finlayson, 2017; Finlayson et al., 2014). The composition of the corpus is shown in Table 1.

The corpus contains 46 Russian folktales, originally collected in Russian in the late 1800's but translated into English in the mid-twentieth century (Finlayson, 2017). The other portion (the N2 corpus) contains 96 stories of relevance to Islamist Extremists (Finlayson et al., 2014). All but 31 of the texts in the corpus already contained gold-standard annotations for token and sentence boundaries, parts of speech, referring expressions, and coreference chains (as well as other layers of annotation. We processed these 31 un-annotated texts using the Stanford CoreNLP suite (Manning et al., 2014), automatically generating tokens, sentences, parts of speech, referring expressions, and coreference chains.

We annotated the whole corpus for animacy of coreference chains, and the first fifteen stories for animacy at the word level. We propagated the animacy annotations from the chains to their constituent referring expressions to generate animacy annotations at that level. Because we had to automatically compute referring expression and coreference chains on 31 of the texts, and the CoreNLP coreference resolution is somewhat noisy, we hand-corrected the chains. We did this hand-correction using the Story Workbench annotation tool (Finlayson, 2008; Finlayson, 2011) that allows for the manipulation and correction of referring expression and coreference chains.

The annotation of the animacy of coreference chains and referring expressions for the first fifteen stories was performed by the first two co-authors. Disagreements were discussed and corrected to generate a gold-standard annotation. Agreement for the coreference-level was 0.99 F_1 and 0.99 Cohen's kappa coefficient (κ), which represents near-perfect overall agreement (Landis and Koch, 1977). The annotation of the rest of the stories was performed by only the first author.

We also annotated first fifteen Russian tales for word-level animacy so that we could test via reimplementation the existing best performing word animacy model (Karsdorp et al., 2015). This annotation was done under the following guidelines. First, all nouns that would refer to animate entities in real life, such humans or animals, as discussed in (Quirk et al., 1985, pp. 314 & 345) were marked animate. We marked gendered pronouns as animate, e.g., *he*, *she*, *his*, *hers*, etc. We also marked adjectives suggesting animacy as animate, e.g., *alive*, *vital*, *kindlier*, etc., whereas adjectives implying inanimacy, such as

	Total entity	Animate entity	Inanimate entity	Unique Animate	Unique Inanimate
Token (15 stories)	23,291	3,896	19,395	291	2,221
Referring Expression (142 stories)	34,698	22,052	12,646	1,104	2,249
Coreference-chain (142 stories)	10,941	3,832	7,109	-	-

Table 2: Total number of animate and inanimate tokens, referring expressions, and coreference chains, with breakdowns of number of unique items in each class.

Referring Expression	**Class**	**Explanation**
Muslims, the dragon, Abu Bakr	Animate	Normally animate entities
walking stove, talking tree	Animate	Normally inanimate but are animate in context
"those who do not know what it is"	Inanimate	Discourse acts, when marked as referents
the mosque, this world, every house	Inanimate	Normally inanimate objects
dead horse	Inanimate	Normally animate but are inanimate in context
her eyes, his hands , horse tail	Inanimate	Inanimate parts of animate entities
Word		
princess, dragon, Abdullah	Animate	Nouns denoting animate entities
he, she, his, her	Animate	Personal pronouns referring to animate objects
kind [prophet], stronger [dragon]	Animate	Adjectives that suggest animacy
Morning, Evening, [talking] stove	Animate	Usually inanimate but are animate in context
Kiev, world, mosque	Inanimate	Nouns denoting inanimate entities
it, that, this	Inanimate	Personal pronouns referring to inanimate objects

Table 3: Examples of annotation of coreference- and word-level animacy. At the word level, only an adjectives suggesting animacy or nouns referring to an animate object are marked animate. Everything else (including verbs, adverbs, determiners, and so forth) are marked inanimate.

dead in the noun phrase *dead horse*, were marked inanimate. Second, we marked as animate any words directly referring to entities that acted animately in a story, regardless of the default inanimacy of the words. For example, we marked *stove* animate in the case of a walking stove, or *tree* animate in the case of a talking tree. This also covered proper names that might normally be marked as inanimate because of their ostensible class, such as those underlined in the next example:

> *All of them were born in one night—the eldest in the evening, the second at midnight, and the youngest in the early dawn, and therefore they were called Evening, Midnight, and Dawn. (Guterman, 1975, Tale #140, p. 458)*

The word-level annotation was done by the first two co-authors. Disagreements were discussed and corrected to generate a gold-standard annotation. We annotated every word in the corpus for animacy directly (marking each word as either animate or not). Agreement was 0.97 F_1 and 0.97 Cohen's kappa coefficient (κ), which represents near-perfect overall agreement (Landis and Koch, 1977).

A summary of the counts of animate and inanimate words, referring expressions, and coreference chains is given in Table 2. Examples of animate and inanimate words are given in Table 3. The data is included in the supplementary materials archive for the paper, which is publicly available[1].

3 Approach

Our hybrid system comprises two parts: a rule-based classifier that can mark the animacy of roughly 50% of the referring expressions, followed by a statistical classifier trained on the annotated data that can be

[1] https://dspace.mit.edu/handle/1721.1/116172

applied to the remaining referring expressions. Once all referring expressions are marked for animacy, the animacy of a coreference chain is inferred from the animacy of its constituent referring expression.

3.1 Rules

We implemented five rules that considered semantic subjects parsed from the semantic role labeler associated with the Story Workbench annotation tool (Finlayson, 2008; Finlayson, 2011), the named entities computed using the classic API of Stanford dependency parse (Manning et al., 2014, v3.7.0), and knowledge from WordNet (Fellbaum, 1998). These rules were inspired by existing rule-based animacy systems. We also considered the last word of a referring expression in most of the rules because it helps to mark quotes as inanimate, as well as to detect the regular animate and inanimate referring expression.

1. If the last word of a referring expression is a gendered personal, reflexive, or possessive pronoun (i.e., excluding *it*, *its*, *itself*, etc.), we marked it animate.
2. If the last word of a referring expression is the semantic subject to a verb, we marked it animate.
3. If a referring expression contains a proper noun we marked it animate. We excluded anything tagged as *location*, *organization*, or *money*, as determined by the Stanford CoreNLP NER system.
4. If the last word of a referring expression is a descendant of *living_being* in WordNet, we marked it animate.
5. If the last word of a referring expression is a descendant of *entity* WordNet, we marked it inanimate.

3.2 Features

We explored seven different binary and vector features to train the statistical classification model, some of which are drawn from prior work.

1. **Word Embeddings (WE)**: We computed pre-trained word embeddings in 300 dimensions for all the words in the stories using the skip-gram architecture algorithm (Mikolov et al., 2013). We used the DeepLearning4J library (Deeplearning4j Development Team, 2017), and configured the built-in skip-gram model with a minimum word frequency of 3, layer width (dimensions) of 300, a window size of 5, and trained for 10 iterations. We explored a few different combinations of these parameters, but found that these settings produced the best results. This is a vector feature drawn from (Karsdorp et al., 2015), and is primarily relevant to classifying word-level animacy. We ran this model on each word of our data and used the output vector as a feature.

2. **Word Embeddings on Referring Expressions (WER)**: We calculated pre-trained word embeddings in 450 dimensions for just the words within the referring expressions, again using the skip-gram approach as above, except with a minimum word frequency of 1. Again, this is a vector feature. 450 dimensions worked better for this feature (rather than 300), which we discovered after doing a small amount of parameter exploration. We ran this model on each referring expression of our data used the output vector as a feature.

3. **Composite Word Embedding (CWE)**: We computed a composite pre-trained word embedding for the neighborhood of each word, adding together the word embedding vectors for three words before and three words after the target word (excluding the target). This is also a vector feature, and is again partially drawn from (Karsdorp et al., 2015). The idea of this feature is that it estimates the similarities of the context among all animate words (or all inanimate words) as well as the dissimilarities of animate from inanimate, and vice versa.

4. **Parts of Speech (POS)**: By analogy with the other embeddings, we computed an embedding over part of speech tags in 300 dimensions, with the same settings as in feature #1 (WE). This feature models the tendency of nouns, pronouns, and adjectives to refer to animate entities.

5. **Noun (N)**: We checked whether a given referring expression contained a noun and encoded this as a boolean feature because we observed that in the first 15 stories 43% of nouns are animate. Thus this feature explicitly captures the tendency of nouns to refer to animate entities. We used dependency parses generated by the classic API of Stanford dependency parser (Manning et al., 2014, v3.7.0).

6. **Grammatical Subject (GS)**: Animate references tend to appear as the grammatical subjects of verbs (Ovrelid, 2005). We used dependency parses generated by the classic API of Stanford dependency parser (Manning et al., 2014, v3.7.0) to check if the last word of a given referring expression was used as a grammatical subject relative to any verb in the sentence, and encoded this as a boolean feature.

7. **Semantic Subject (SS)**: We also computed whether or not a referring expression appeared as a semantic subject to a verb. We used the semantic role labeler associated with the Story Workbench annotation tool (Finlayson, 2008; Finlayson, 2011) to compute semantic roles for all the verbs in the stories. We then checked whether the last word of a given referring expression contained an ARG0 for a verb (an exact match was not required), and encoded this as a boolean feature.

3.3 Classification Models

We implemented our classification models using SVM (Chang and Lin, 2011), with a Radial Basis Function Kernel. The features used to train the different models are shown in Table 4. We trained each model using cross validation, and report macroaverages across the performance on test folds.

We have three models for animacy: referring expressions, coreference chains, and words. For our referring expression animacy model, we implemented two approaches. The first is a ML-only approach, in which we explored different combinations of features: word embedding over referring expressions (WER), noun (N), grammatical subject (GS), and semantic subject (SS). We configured the SVM with $\gamma = 1, C = 0.5$ and $p = 1$. We measured the performance of the classifier using 10-fold cross validation. The second approach is the hybrid system where we we first applied the rules, then applied the ML classifier for referring expressions not covered by the rules. In our prior work we only implemented the first approach (Jahan et al., 2017) on a small data set.

For the coreference chain animacy model, we implemented a majority voting approach for combining the results of the referring expression animacy model to obtain a coreference animacy prediction. In the case of ties, the chain was marked inanimate.

To compare with prior work, we also implemented a word animacy model, adapting an existing system with the best performance (Karsdorp et al., 2015). That model used features based on word N-grams, parts of speech, and word embeddings. Similarly, we implemented our classifier using word embeddings over words (WE), combined word embeddings (CWE), and parts of speech (POS). The SVM was configured with $\gamma = 5, C = 5000$ and $p = 1$, and we measured the performance with 20-fold cross validation. This model performed very close to the prior state of the art with our small data set of 15 stories.

4 Results & Discussion

We calculated two baselines for referring expression animacy. The first baseline is to choose the majority class (animate). The second baseline combines word-level animacy predictions generated by our word animacy model via a majority vote; we measured the upper bound for this over the 15 texts for which we have gold-standard word animacy annotations.

We evaluated our models by measuring accuracy, precision, recall, F_1, and Cohen's kappa (κ) compared to the gold-standard annotations. Table 4 shows the results for both classes. Our word animacy model achieved an F_1 of 0.98, whereas the prior state of the art achieved F_1 of 0.99 for marking inanimacy. On the other hand, for marking animacy our model achieved F_1 of 0.90 where the state of the art achieved F_1 of 0.93. For referring expression animacy we varied the features to determine the optimal set. We obtained the best result (F_1 of 0.84) using different combinations of three features: noun (N), grammatical subject (GS) and semantic subject (SS). Our hybrid model for referring expression animacy performed better (F_1 of 0.88) than the statistical model (F_1 of 0.84). The rule-based model achieved 0.88 F_1 when we applied the rules first, and marked any remaining referring expressions as majority class. Our rule based model performed similarly to the hybrid model, but the hybrid model is more consistent.

For the coreference animacy model, we implemented the majority vote approach to detect animacy of coreference chain using the best output of referring expression model. Majority vote resulted in an overall F_1 of 0.75, which substantially outperforms the result from our prior work of 0.61 F_1. Around 3% of coreference chains resulted in a tied vote, and these were marked as inanimate (the majority class).

Model	Feature Set	Acc.	Inanimate				Animate			
			κ	Prec.	Rec.	$\mathbf{F_1}$	κ	Prec.	Rec.	$\mathbf{F_1}$
Word	(Karsdorp et al., 2015)	-	-	0.98	0.99	0.99	-	0.94	0.91	0.93
	WE, CWE, POS	96%	0.87	0.98	0.98	0.98	0.87	0.91	0.88	0.90
Ref. Expr.	Baseline MFC	61%	0.0	0.0	0.0	0.0	0.0	0.61	1.0	0.76
	Baseline Maj. Vot.	75%	0.53	0.59	0.99	0.74	0.53	0.99	0.62	0.76
	Hybrid on Stanford	80%	0.61	0.89	0.73	0.79	0.61	0.74	0.90	0.81
	WER, N, GS, SS	76%	0.47	0.80	0.51	0.62	0.47	0.76	0.92	0.83
	N, GS	78%	0.51	0.83	0.54	0.65	0.51	0.77	0.93	0.84
	N, SS	79%	0.53	0.80	0.60	0.68	0.53	0.78	0.91	0.84
	N, GS, SS	79%	0.53	0.81	0.59	0.68	0.53	0.78	0.91	0.84
	Prior best result	86%	0.70	0.83	0.77	0.80	0.68	0.87	0.91	0.90
	Rule Based model	82%	0.60	0.89	0.60	0.72	0.60	0.81	0.96	0.88
	Hybrid model	83%	0.62	0.84	0.67	0.74	0.62	0.83	0.93	0.88
	Random Sampling	92%*	0.85	0.87	0.93	0.91	**0.85**	**0.96**	**0.92**	**0.94**
Coref.	Prior best result	79%	0.48	0.93	0.80	0.86	0.48	0.50	0.76	0.61
	Maj. vote on Stanford	79%	0.54	0.91	0.78	0.84	0.54	0.60	0.82	0.69
	Maj. vote (current)	82%	0.61	0.87	0.84	0.86	0.61	0.73	0.77	0.75
	Random Sampling	90%†	0.80	0.86	0.98	0.92	**0.80**	**0.97**	**0.81**	**0.88**

Table 4: Result of different Animacy Models (Bolded according to when our F_1 measure is higher). MFC stands for "Most Frequent Class", and the other abbreviations stand for features as indicated in the text. *Estimated $\pm 2\%$ with 95% confidence. †Estimated $\pm 1\%$ with 95% confidence.

We also evaluated our model using direct sampling (Saunders et al., 2009). We ran our hybrid model over 200 news articles from the OntoNotes (Hovy et al., 2006) data set containing 46,088 referring expressions and 7,836 coreference chains. We randomly sampled 558 coreference chains and checked their animacy markings by hand, resulting in a estimated accuracy of 90% $\pm 2\%$ at a 95% confidence level, as well as estimated precision, recall, and F_1 listed in Table 4. Those coreference chains contained 3,543 referring expressions, which allowed us to estimate the accuracy of the referring expression model at 92% $\pm 1\%$ at a 95% confidence level.

The data contains 46 folktales, which have 142 mentions of 12 characters who are members of traditionally inanimate classes (e.g., stoves that walk, trees that talk). We manually identified those 12 characters and evaluated our model's performance on them. Our system is able to detect the animacy of these unusual referents with an F_1 of 0.95. Conversely, there was only one mention of a normally animate class that was inanimate in context ("dead horse"), and this was correctly marked by the system.

5 Error Analysis & Future Work

A detailed error analysis of the results revealed at least two major problems for the hybrid model that we will focus on in future work: short chains, quotes, and exceptions to the rules.

Determining the animacy of short coreference chains was challenging for our system: approximately 11% of short chains are incorrectly marked. As the length of a chain tends toward a single referring expression, the coreference classifier should converge to the referring expression classifier performance. However, for chains between two and four referring expressions long, the majority voting approach seems to fall short. We suspect this is because many referring expressions are themselves quite short, and can contain false alarms: e.g., our system classifies "his hands" as animate because of the animate word "his" in the expression. We believe another approach to solving this problem is to generate new rules in our hybrid model so that it can handle these type of special cases.

Second, many quotes are full of animate words, e.g., "the fate of the tsar's daughter to go to the dragon" is a phrase that is itself a referring expression in one story, and should be inanimate according

to our animacy annotation rule. However, the classifier marks the quote as animate because it finds three animate words: *tsar*, *daughter*, and *dragon*. In our data, approximately 2.5% of quotes that are referring expressions are incorrectly marked, and handling this likely will require rule-based processing.

Finally, a common error type was exceptions to the rules. In the hybrid system we combined together a large number of similar referring expressions under one rule so that we can handle them under a similar animacy class. But there are always exceptions for every rule: for example, we define "it" as inanimate but of course sometimes "it" can refer to an animate object. For the most part these individual instances will be out-voted by animate referring expressions in long chains, so it is a relatively small problem. One approach to solving this would to implement the idea of Orăsan and Evans (2001; 2007) to use supervised machine learning to mark unseen WordNet senses by their animacy rather using specific rules.

6 Related Work

We divide the related work into two sections: first animacy detection in English, followed by animacy detection in other languages. The work reported here is in English (thus the related work of the first section), but the material covered in non-English second section makes clear both that our approach had not attempt before in any language, and also that no language-specific features have been used in any prior work. There have been both rule-based and machine learning methods to classify the animacy of words, but to the best of our knowledge, no one has combined both techniques, and no one has tackled animacy classification at the referring expression or coreference level.

6.1 Animacy Detection in English

Evans and Orăsan (2000) performed animacy classification to improve anaphora resolution using a rule-based method to identify animate WordNet hypernym branches. In later work they used supervised machine learning to mark unseen WordNet senses for their animacy (Orăsan and Evans, 2001; Orăsan and Evans, 2007). The rule-based method uses the unique beginners in WordNet for classification of sense animacy using a statistical chi-squared method, while the machine learning method uses k-nearest neighbors in a multi-step procedure, along with careful feature engineering, to determine noun animacy. They achieved an F_1 of 0.94 for animacy, and also performed an extrinsic evaluation using the MARS anaphora resolution system and a word sense disambiguation algorithm. Similarly, Moore et al. (2013) combined a majority vote model using rule-based methods, features from WordNet, and a SVM to achieve an accuracy of 89% for majority voting and 95% for SVM (no F_1 score was reported).

Bowman and Chopra (2012) used a maximum entropy classifier to predict multiple classes for noun phrases as *human*, *vehicle*, *time*, *animal*, etc., with an overall accuracy of 85%. A binary animacy classification could be derived from each of these classes, with a performance of 94% accuracy.

Additionally, there are others that have used pure rule-based and pattern matching methods. Ji and Lin (2009) generate n-grams and performed pattern matching using the Google n-gram corpus to label gender and animacy properties for words for to assist in person mention detection. With these gender and animacy markings, they applied a confidence estimation which is compared against the test document using fuzzy matching. The highest F_1 they achieved for animacy was 0.67, with an F_1 of 0.46 for gender.

Declerck et al. (2012) used an ontology-based method to detect characters in folktales. Their ontology consists of family relations as well as elements of folktales such as supernatural entities. After looking at the heads of noun phrases and comparing them with labels in the ontology, they added the noun phrase to the ontology as a potential character if a match was found. Then, they applied inference rules to the candidate characters in order to find two strings in the text that refer to the same character. They discarded strings that are related to a potential character only once and are not involved in an action. They obtained an accuracy of 79%, a precision of 0.88, a recall 0.73, and an F_1 of 0.80.

Wiseman et al. (2015) used a mention-ranking approach for coreference resolution, using animacy as a feature, derived from the Stanford Coreference System (Lee et al., 2013). The Stanford Coreference System set animacy attributes using a static list for pronouns, named entity labels, and a dictionary.

Finally, a marginally related rule-based system was implemented by Goh et al. (2012) using verbs and WordNet in order to determine the protagonists in fairy tales (where protagonists must of necessity be

animate). They used the Stanford parser's phrase structure trees to obtain the subjects and objects of the verbs and used the dependency structure to obtain the head noun of compound phrases. Additionally, they used WordNet's *derivationally_related* relation to find verb associated with a particular nominal action. They achieved a precision of 0.69, a recall of 0.75, and an F_1 of 0.67.

6.2 Animacy Detection in Other Languages

Nøklestad (2009) implemented animacy detection for Norwegian nouns, using this along with Named Entity Recognition to improve the performance of anaphora resolution. They explored various pattern matching methods, using web data to extract lists of animate nouns as well as to check the animacy of a particular noun. For example, if a noun co-referred frequently with *han* (he) or *hun* (she), then it was characterized as animate. This method achieved an accuracy of 93%. The main problem here, from our point of view, is that using data from the web makes the problem too general: you only measure the typicality of animacy, not the animacy of an item in context. In the case of folktales, we have unusual animate entities (e.g., talking stoves) that will on the whole be seen by the web as inanimate.

Bloem and Bouma (2013) developed an automatic animacy classifier for Dutch nouns by dividing them into *Human*, *Nonhuman* and *Inanimate* classes. They use the k-nearest neighbor algorithm with distributional lexical features—e.g., how frequently the noun occurs as a subject of the verb "to think" in a corpus—to decide whether the noun was predominantly animate. Prediction of the *Human* category achieved 87% accuracy, and the large inanimate class was predicted correctly 98% of the time. But, again, this work focuses on individual noun phrases, not coreference chains, and is concerned with the default animacy of the expression, not its animacy in context.

Another implementation of word-level animacy for Dutch was performed by Karsdorp et al. (2015) on folktale texts. Because this work was the highest performing word-level system, many of our features were inspired by their approach. They used lexical features (word forms and lemmas), syntactic features (dependency parses to check which word is a subject or an object), part of speech tags, and semantic features (word embedding using a skip-gram model to vectorize each word). They implemented a Maximum Entropy Classifier to classify words according to their animacy and obtained a good result of 0.93 F_1 for the animate class, by just using the words, parts of speech, and embedding features.

Baker and Brew (2010) performed animacy classification on a multilingual dataset containing English and Japanese. They used Bayesian logistic regression with morphological features, WordNet semantic categories, and frequency counts of verb-argument relations. They obtained 95% classification accuracy. In sum, all the prior work has been for word-level animacy (usually nouns, sometimes noun phrases). In contrast, we focus on characterizing the animacy of referring expressions and coreference chains.

7 Contributions

This paper makes four major contributions. First, we have redefined the problem of animacy classification as one of marking animacy on coreference chains, in contrast to all prior work that seeks to mark the animacy at the world level. Second, we have presented a hybrid system merging an SVM classifier and hand-built rules to predict the animacy of referring expressions directly, achieving performance of 0.90 F_1, which is comparable to the state of the art for word-level animacy detection. Third, we used a majority voting approach to obtain the animacy of coreference chains. The overall performance of this approach is substantially improved in comparison with our prior work. Our error analysis further suggests several potentially profitable ways forward to improving the performance. Finally, we provide 15 texts annotated for word-level animacy and 142 texts annotated for coreference chain animacy, as well as the code reproducing the results, in the supplementary materials archive[2].

Acknowledgements

This work was partially supported by Office of Naval Research (ONR) grant number N00014-17-1-2983. We would also like to thank Deya Banisakher, Joshua Eisenberg and W. Victor H. Yarlott from the FIU Cognac Lab for their discussions and assistance.

[2] https://dspace.mit.edu/handle/1721.1/116172

References

Kirk Baker and Chris Brew. 2010. Multilingual animacy classification by sparse logistic regression. *Information Concerning OSDL OHIO STATE DISSERTATIONS IN LINGUISTICS*, page 52.

Jelke Bloem and Gosse Bouma. 2013. Automatic animacy classification for Dutch. *Computational Linguistics in the Netherlands Journal (CLIN)*, 3:82–102.

Samuel R Bowman and Harshit Chopra. 2012. Automatic animacy classification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT): Student Research Workshop*, pages 7–10. Montreal, Canada.

Claire Cardie and Kiri Wagstaf. 1999. Noun phrase coreference as clustering. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Greg N Carlson and Michael K Tanenhaus. 1988. Thematic roles and language comprehension. *Syntax and semantics*, 21:263–288.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Michael Connor, Cynthia Fisher, and Dan Roth. 2013. Starting from scratch in semantic role labeling: Early indirect supervision. In *Cognitive aspects of computational language acquisition*, pages 257–296. Springer.

Thierry Declerck, Nikolina Koleva, and Hans-Ulrich Krieger. 2012. Ontology-based incremental annotation of characters in folktales. In *Proceedings of the 6th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 30–34. Association for Computational Linguistics.

Deeplearning4j Development Team. 2017. Deeplearning4j: Open-source distributed deep learning for the JVM, Apache Software Foundation License 2.0. `http://deeplearning4j.org`. Accessed: 2017-04-08.

Richard Evans and Constantin Orăsan. 2000. Improving anaphora resolution by identifying animate entities in texts. In *Proceedings of the Discourse Anaphora and Reference Resolution Conference (DAARC2000)*, pages 154–162. Lancaster, England.

Christiane Fellbaum, editor. 1998. *WordNet An Electronic Lexical Database*. MIT Press, Cambridge, MA.

Fernanda Ferreira. 1994. Choice of passive voice is affected by verb type and animacy. *Journal of Memory and Language*, 33(6):715–736.

Mark A Finlayson, Jeffry R Halverson, and Steven R Corman. 2014. The n2 corpus: A semantically annotated collection of islamist extremist stories. In *LREC*, pages 896–902.

Mark Alan Finlayson. 2008. Collecting semantics in the wild: The story workbench. In *Proceedings of the AAAI Fall Symposium on Naturally Inspired Artificial Intelligence*, pages 46–53. Arlington, VA.

Mark A Finlayson. 2011. The Story Workbench: An extensible semi-automatic text annotation tool. In *Proceedings of the 4th Workshop on Intelligent Narrative Technologies (INT4)*, pages 21–24. Stanford, CA.

Mark A. Finlayson. 2017. ProppLearner: Deeply Annotating a Corpus of Russian Folktales to Enable the Machine Learning of a Russian Formalist Theory. *Digital Scholarship in the Humanities*, 32(2):284–300.

Monika Fludernik. 2009. *An Introduction to Narratology*. Routledge, New York.

Hui-Ngo Goh, Lay-Ki Soon, and Su-Cheng Haw. 2012. Automatic identification of protagonist in fairy tales using verb. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 395–406. Springer.

Norbert Guterman. 1975. *Russian Fairy Tales*. Pantheon Books, New York.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60. Association for Computational Linguistics.

Ryu Iida, Kentaro Inui, Hiroya Takamura, and Yuji Matsumoto. 2003. Incorporating contextual cues in trainable models for coreference resolution. In *Proceedings of the EACL Workshop on The Computational Treatment of Anaphora*, pages 23–30. Citeseer.

Labiba Jahan, Geeticka Chauhan, and Mark Finlayson. 2017. Building on word animacy to determine coreference chain animacy in cultural narratives. In *The AIIDE-17 Workshop on Intelligent Narrative Technologies WS-17-20*.

Heng Ji and Dekang Lin. 2009. Gender and animacy knowledge discovery from web-scale n-grams for unsupervised person mention detection. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation, Volume 1*, volume 1.

Folgert Karsdorp, M Meulen, Theo Meder, and APJ van den Bosch. 2015. Animacy detection in stories. In *Proceedings of the 6th Workshop on Computational Models of Narrative (CMN'15)*, pages 82–97. Atlanta, GA.

Seppo Kittila, Katja Vasti, and Jussi Ylikoski. 2011. *Introduction to Case, animacy and semantic roles*. John Benjamins Publishing Company.

Seppo Kittilä. 2006. Object-, animacy-and role-based strategies: A typology of object marking. *Studies in Language. International Journal sponsored by the Foundation Foundations of Language*, 30(1):1–32.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL): System Demonstrations*, pages 55–60. Baltimore, MD.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. http://arxiv.org/abs/1301.3781.

Joshua Moore, Christopher JC Burges, Erin Renshaw, and Wen-tau Yih. 2013. Animacy detection with voting models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 55–60.

Anders Nøklestad. 2009. *A Machine Learning Approach to Anaphora Resolution Including Named Entity Recognition, PP Attachment Disambiguation, and Animacy Detection*. Ph.D. thesis, University of Oslo, Oslo, Norway, May.

Constantin Orăsan and Richard Evans. 2001. Learning to identify animate references. In *Proceedings of the 2001 Workshop on Computational Natural Language Learning (CoNLL)*, page Article No. 16. Toulouse, France.

Constantin Orăsan and Richard J Evans. 2007. NP animacy identification for anaphora resolution. *Journal of Artificial Intelligence Research*, 29(1):79–103.

Lilja Ovrelid. 2005. Animacy classification based on morphosyntactic corpus frequencies: some experiments with Norwegian nouns. In *Proceedings of the Workshop on Exploring Syntactically Annotated Corpora*, pages 24–34. Birmingham, England.

Martha Palmer, Olga Babko-Malaya, and Hoa Trang Dang. 2004. Different sense granularities for different applications. In *Proceedings of the 2nd International Workshop on Scalable Natural Language Understanding (ScaNaLU 2004) at HLT-NAACL 2004*.

Martha Palmer, Hoa Trang Dang, and Christiane Fellbaum. 2007. Making fine-grained and coarse-grained sense distinctions, both manually and automatically. *Natural Language Engineering*, 13(2):137–163.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, and Jan Svartvik. 1985. *A Comprehensive Grammar of the English Language*. Longman, London.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 492–501. Association for Computational Linguistics.

Mark Saunders, Philip Lewis, and Adrian Thornhill. 2009. *Research methods for business students*. Prentice Hall,UK.

Sam Joshua Wiseman, Alexander Matthew Rush, Stuart Merrill Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing,*. Association for Computational Linguistics.

Mutsumi Yamamoto. 1999. *Animacy and reference: A cognitive approach to corpus linguistics*. John Benjamins Publishing, Amsterdam.

Zero Pronoun Resolution with Attention-based Neural Network

Qingyu Yin[♯], Yu Zhang[♯]*, Weinan Zhang[♯], Ting Liu[♯], William Yang Wang[♭]
[♯]Harbin Institute of Technology, China
[♭]University of California, Santa Barbara, USA
{qyyin, yzhang, wnzhang, tliu}@ir.hit.edu.cn
william@cs.ucsb.edu

Abstract

Recent neural network methods for zero pronoun resolution explore multiple models for generating representation vectors for zero pronouns and their candidate antecedents. Typically, contextual information is utilized to encode the zero pronouns since they are simply gaps that contain no actual content. To better utilize contexts of the zero pronouns, we here introduce the self-attention mechanism for encoding zero pronouns. With the help of the multiple hops of attention, our model is able to focus on some informative parts of the associated texts and therefore produces an efficient way of encoding the zero pronouns. In addition, an attention-based recurrent neural network is proposed for encoding candidate antecedents by their contents. Experiment results are encouraging: our proposed attention-based model gains the best performance on the Chinese portion of the OntoNotes corpus, substantially surpasses existing Chinese zero pronoun resolution baseline systems.

Title and Abstract in Chinese

基于注意力神经网络模型的零指代消解研究

传统的关于零指代的方法提出了多种关于先行语和零代词的表示模型。这些模型中，研究者们用零代词的上下文信息来帮助表示缺省的信息。为了更好的帮助建模零代词，我们提出了一种基于注意力机制的神经网络模型，通过注意力模型来获取更有表示性信息的上下文信息。实验结果表明：我们的方法能够有效提升效果，并在中文OntoNotes 5.0数据集上取得了最好的结果，超越了现有的基准系统。

1 Introduction

In natural languages, expressions that can be deduced contextually by people are frequently omitted in texts. This is special the case in pro-dropped languages, such as Chinese, where a kind of anaphoric expression is frequently eliminated. A zero pronoun is a gap in the sentence that is found when a phonetically null form is used to refer to a real-world entity (Chen and Ng, 2016). We here show a case of zero pronouns from the OntoNotes-5.0 dataset.

> 这 次 地震 ϕ_1 有 一些 房屋 塌 的 , 这 里面 如果 有 建房 的 质量 问题 ， ϕ_2 是 要 追究 责任 的 。
>
> In this earthquck ϕ_1 some rooms collapsed, if there exsit some room quality issues, ϕ_2 will need to call to account.

We use ϕ to represent the zero pronouns in this example. Among these zero anaphoras, we can assign the mention "政府/the government" that appears in leading text, to be the antecedent of ϕ_2 while there are no such mentions for ϕ_1. Hence, ϕ_2 is an anaphoric zero pronoun, and ϕ_1 is the un-anaphoric case.

*Corresponding author.

Proceedings of the 27th International Conference on Computational Linguistics, pages 13–23
Santa Fe, New Mexico, USA, August 20-26, 2018.

With the fact that zero pronouns are gaps that have no text, it is almost impracticable to representant the zero pronouns by themselves. This issue has received increasingly attention. In recent time, deep neural network methods for Chinese zero pronoun resolution (Chen and Ng, 2016; Yin et al., 2017a; Yin et al., 2017b) have been proposed and are intended to encode zero pronouns into the vector-space semantic by additional elements. Chen and Ng (2016) propose a neural network model that learns to encode the anaphoric zero pronoun by using the leading word and governing verb, which leads to the insufficient text issue. To better use associated text information for expressing zero pronouns, Yin et al. (2017a) present the ZP-centered-LSTM architecture that learns to encode zero pronouns by their text words. However, it could bring with some defects: their model regards all the words in the sentence equally, thus fails in capturing informative parts of the sentence. As described in (Chen and Ng, 2016), the clause information of a zero anaphora is very important in explaining these gaps, it is a natural way of modeling zero pronoun by focusing on some important texts. For instance, in sentence "two trains in ten weeks ago ϕ crashed in their way to the southern German countryside", "crashed in their way" is an important clue for the zero pronoun but "several weeks ago" is not. Under this consideration, a systematic solution that can encode zero pronouns by focusing on informative parts of associate texts is the better choice. Besides, another important issue for the task of zero pronoun resolution is modeling candidates. Recent researches use context and content words to encode candidates (Yin et al., 2017a; Yin et al., 2017b). Typically, these words are modeled in a sequential way by recurrent neural networks. We argue that some of the words in noun phrases contains more important information than others, a need that leads to the usage of attention mechanism.

To alleviate the above-mentioned issues, in this paper, we propose a novel attention-based neural network model to deal with the task. Following existing neural network work for Chinese zero pronoun resolution (Chen and Ng, 2016; Yin et al., 2017a; Yin et al., 2017b), we focus on anaphoric zero pronoun resolution task, introducing a pair-wise model to resolve anaphoric zero pronouns. For some natural language processing tasks (Mnih et al., 2014; Tang et al., 2016), people investigate to apply attention mechanism on top of the convolutional neural network or recurrent neural network to introduce an extra source of information to guide the modeling of useful information. However, since zero pronouns are simple gaps that have no such kind of extra information, the above-mentioned attention mechanism can rarely be directly practiced for modeling these gaps. Inspired by (Lin et al., 2017), we here investigate the usage of a self-attentive mechanism for encoding the zero anaphoras. With the help of self-attentive mechanism, our model is able to effectively focus on informative texts of the zero pronouns and therefore captures essential information on encoding the zero pronouns. In addition, on purpose of modeling informative texts of mentions (noun phrases), we propose an attention-based recurrent neural network to build the mention encoder. With the help of representative vector of zero anaphoras, our model is able to effectively focus on important parts of mentions and therefore brings an efficient way of expressing candidates at the semantic level. Empirically, we show that our method has brought performance gains in baselines, achieving great performance on the widely used OntoNotes-5.0 dataset. Our contributions are three-fold:

- By utilizing the self-attention mechanism, our model is able to focus on informative parts of associate texts when modeling zero pronouns, leading to an effective way of capturing useful information for interpreting the zero pronouns;

- Our model is capable of modeling candidate antecedents by their informative words with the help of zero pronouns, which in return brings a better way of explaining candidate antecedents;

- We show that our model substantially surpasses all baseline systems, gains state-of-the-art performance on the benchmark dataset.

In Section 2, we will discuss related work on zero pronoun resolution. Next, we will intorduce our attention-based neural network model in Section 3. In Section 4, empirical evaluation results are shown. And finally, we conclude in Section 5.

2 Related Work

In this section, we give a brief summary of early efforts for zero pronoun resolution both for Chinese and other languages.

2.1 Zero Pronoun Resolution for Chinese

Converse (2006) is the first rule-based study that integrate Hobbs-algorithm into the resolution of zero pronoun in the Chinese Treebank (Xue et al., 2005). After that, a variety of learning-based methods have been investigated. Zhao and Ng (2007) use the learning-based model to locate and resolve zero anaphoras. They investigate a serious of features and apply the decision-tree algorithm to train the classifier. To better capture the syntactic-level information, Kong and Zhou (2010) introduce the context sensitive tree-kernel unified framework for zero anaphor resolution. On the base of Zhao and Ng (2007), Chen and Ng (2013) further investigate their model, introducing two extensions to the resolver, namely, novel features and zero pronoun links. However, these work deeply rely on annotation dataset. To alleviate this issue, Chen and Ng (2014) present the first unsupervised model that first convert zero anaphoras into ten pre-defined pronouns and then apply a ranking-based pronoun resolution model to select antecedent mentions. Chen and Ng (2015) build a discourse-aware model that can jointly locate and resolve zero anaphoras.

More recently, with the advance of neural network techniques, deep-learning-based methods are introduced and have been demonstrated to be effective for this task. Chen and Ng (2016) first introduce a feed-forward neural network framework, where zero anaphoras are encoded by its previous word and headword. However, their model overlooks context of a zero anaphora, which inevitably misses some valuable information. Naturally, some works try to alleviate this issue by investigating information from associate texts. Yin et al. (2017a) introduce a novel memory-based network neural network model that learns to encode zero anaphoras by its texts and antecedent mentions. They take advantage of multi-hops architecture, producing abstract information from external-memories as hints for explaining zero anaphoras. Yin et al. (2017b) focus on encoding global-information for candidates, where a hierarchical candidate encoder is introduced that learns to model the candidates. Liu et al. (2017) investigate the issue of generating pseudo training-data for the task of zero anaphora resolution. They use a novelty two-step training strategy that helps to overcome the diversity between the generated pseudo training-data and the real one. Even though these above-mentioned methods can reveal the semantic of zero anaphoras by its context, they regard all the words equally, overlooking the diversity of different words. In this paper, we focus on exploring an effective way of modeling zero pronoun by using the associated texts. More specifically, we integrate a novel self-attentive mechanism, which provides our model an ability to focus on multi-aspects text, benefiting the encodings of zero anaphoras. In addition, by employing an attention-based technique for modeling candidates, our model learns to encode more informative parts of the mentions. All these bring advantages to the resolution of zero pronouns.

2.2 Zero Pronoun Resolution for other Languages

There has been a variety of work on zero pronoun resolution for other languages besides Chinese, such as Korean and Japanese. These methods could be categorized as rule-based and learning-based. Ferrández and Peral (2000) investigate a rule-based method that can encode preferences for candidates for resolving zero anaphoras in Spanish. In recent time, learning-based methods (Han, 2006; Iida and Poesio, 2011; Isozaki and Hirao, 2003; Iida et al., 2006; Iida et al., 2007; Sasano and Kurohashi, 2011; Iida and Poesio, 2011; Iida et al., 2015; Iida et al., 2016) have been well studied. Iida et al. (2016) present a novel CNN-based deep neural network model for intrasentential subjective zero anaphora resolution in Japanese. As clues, they use both the surface-word and the dependency tree-structure of a sentence. Their model gains higher precision, which is needed for real-world natural language processing (NLP) applications.

3 Methodology

We introduce an attention-based neural network model for anaphoric zero pronoun resolution. Compared to the prior studies that have the underutilized context of zero pronouns, we investigate an attention

mechanism that helps to effectively capture useful information from associate texts. We here present the methodology in details, which include the preliminary, the architecture of proposed attention-based neural network and training objective of the model.

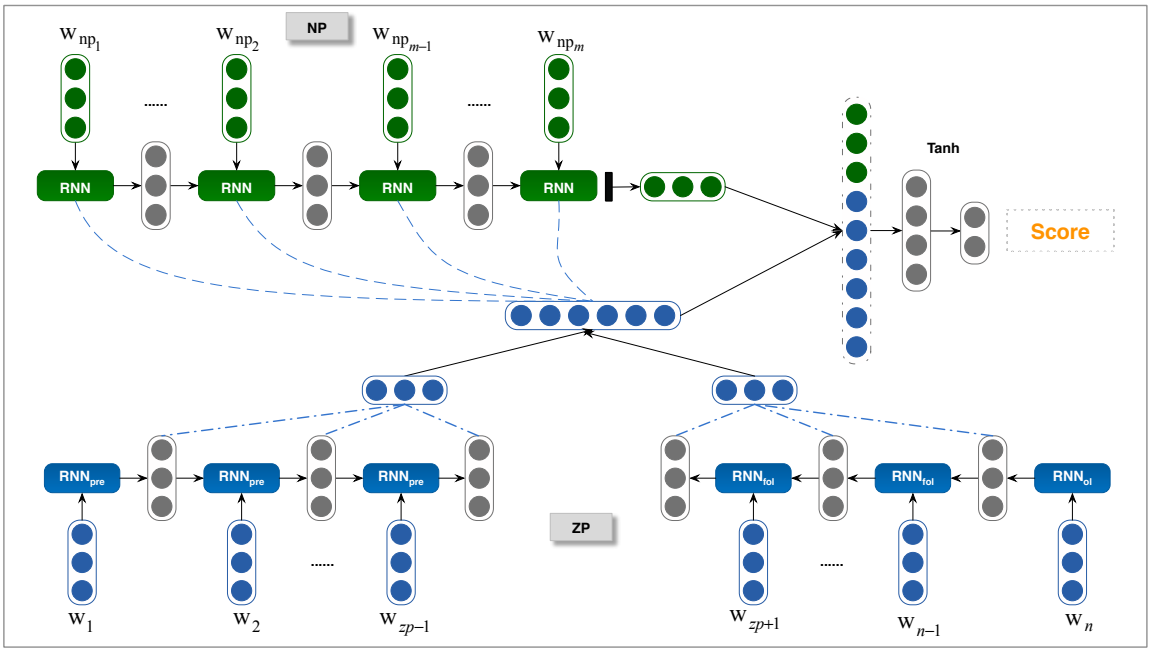


Figure 1: Framework of the proposed attention-based neural network for zero pronoun resolution. The w_i for zero pronoun part means the i-th word in the associated sentence, w_{zp-i} is the i-th word before the zero pronoun and w_{zp+i} is the i-th word behind the zero pronoun. w_{np_i} is the i-th word of the noun phrase. After generating the representative vector of zero pronoun and candidate mention, we generate its resolution score by going through two $tanh$ layers.

3.1 Preliminary

In the first place, we select its candidate mentions from the associated texts. More specifically, we choose the noun phrases that appear within two sentences from the zero anaphora to be the candidates. In addition, we choose the strategy used in prior approaches (Chen and Ng, 2016; Yin et al., 2017a) for Chinese zero pronoun resolution, reserving those mentions that are max noun phrases or modifier noun phrases as candidates. By doing so, we can recall most (about 98%) of the antecedents with a small loss. Then, what we desire the resolver to do is to accurately recognize antecedents for zp from its candidates list $NP = \{np_1, np_2, ..., np_n\}$.

Our model is basically a pair-wise model (Zhao and Ng, 2007; Chen and Ng, 2016; Yin et al., 2017b). For each candidate mention of zp, we classify it into two classifications, namely, "corefer" that means the candidate is the antecedent of the zero pronoun; or otherwise, "un-corefer". We build the classifier by applying the attention-based techniques. More specifically, an attention-based neural network model is utilized that generates the coreference score of each zero pronoun-candidate antecedent pair.

3.2 Attention-based Neural Network Model

In this part, we introduce our attention-based neural network for anaphoric zero pronoun resolution in detail. The architecture of our model is shown in Figure 1. For an anaphoric zero pronoun zp, we convert the input instances into real-valued vectors by using its context words. With the help of the self-attentive mechanism, our model learns to represent zero pronouns by focusing on different parts of contexts. In this way, we generate the representative vector of zp as v_{zp}. In addition, when dealing with candidate mentions, our model learns to encode the candidate mentions by using its informative content words. We here manipulate the v_{zp} as an external vector to attend to the informative parts of its candidate mentions.

By doing so, our model captures the important information of each candidate according to the zp. After that, we generate the representative vectors of candidates as $\{v_{np_1}, v_{np_2}, ..., v_{np_n}\}$. Lastly, we feed the representative vectors of the candidate and zero pronoun to a two-layer neural network, generating the resolution score for each zero pronoun-candidate antecedent pair. After that, we obtain the resolution probability for each candidate. The candidate mention with the biggest probability is regarded as the final result. Basically, our model involves three modules, namely, a zero pronoun encoder; a candidate antecedent encoder; and a feed-forward neural network that learns to score each candidate antecedent.

3.2.1 Modeling Zero Pronoun

Inspired by Lin et al. (2017), we investigate the self-attention mechanism when modeling zero anaphoras. In practice, we use two recurrent neural networks (RNNs) to encode the preceding and following texts of the zero anaphora (Yin et al., 2017a). The last hidden vectors of these two RNNs are concatenated as the vector-space semantic of the zero pronoun. On top of the employed recurrent neural network architecture, we apply the proposed attention technique that helps the model to capture the informative parts of associated texts.

Especially, our self-attention mechanism provides the attention-weight vectors for hidden states of the employed RNN architectures. We then dot these attention-weight vectors with counterpart hidden states and use the weighted summation vector as the representative vector for the zero pronoun. For a zero pronoun, we map its associated text words as sequential embeddings.

$$Context_{precedding} = (w_1, w_2, ..., w_{zp-1}) \tag{1}$$

$$Context_{following} = (w_{zp+1}, w_{zp+2}, ..., w_n) \tag{2}$$

where w_i is a d dimensional embedding for the ith word in the sentence. After that, we generate the representative vector of the preceding and the following text by using two separate RNNs:

$$h_t^{pre} = RNN_{pre}(w_t, h_{t-1}^{pre}) \tag{3}$$

$$h_t^{fol} = RNN_{fol}(w_t, h_{t-1}^{fol}) \tag{4}$$

where RNN_{pre} and RNN_{fol} are two employed RNNs that model the precedding and following context of the zero pronoun independently. After that, we get the hiddent vector for each word, which has the dimension of u. We represent all the hidden states of RNN_{pre} and RNN_{fol} as $H_{pre} \in \mathbb{R}^{n_{pre} \times u}$ and $H_{fol} \in \mathbb{R}^{n_{flo} \times u}$, seperately:

$$H_{pre} = \{h_1^{pre}, h_2^{pre}, ..., h_{n_{pre}}^{pre}\} \tag{5}$$

$$H_{fol} = \{h_1^{fol}, h_2^{fol}, ..., h_{n_{fol}}^{fol}\} \tag{6}$$

We then apply the self-attention mechanism, which computes linear-combinations of the hidden vectors in H_{pre} and H_{fol}. The attention mechanism takes H_{pre} (or H_{fol}) as the inputs and produces a matrix of attention-weight A_{pre} (A_{fol}):

$$A_{pre} = softmax(W_2^{pre} tanh(W_1^{pre} H_{pre}^T)) \tag{7}$$

$$A_{fol} = softmax(W_2^{fol} tanh(W_1^{fol} H_{fol}^T)) \tag{8}$$

where W_1 is a weight matrix with a shape of d_a-by-u and W_2 is in shape of r-by-d_a; r represents the number of hops of attention we choose. The $softmax()$ is performed along the second dimension of its input. In this way, the attention matrix A could be seen as a multi-hope attention matrix. Comparing with the single-attention matrix, such a mechanism enables our model to focus on different parts of the contexts, bringing a more efficient way of modeling sentence-level information for the zero pronoun at the semantic level.

We get the r weighted sums by multiplying the attention matrix A and hidden states H, regarding the resulting matrix as the representative vector of the zero pronoun's preceding and following texts:

$$M_{pre} = A_{pre} H_{pre} \tag{9}$$

$$M_{fol} = A_{fol} H_{fol} \tag{10}$$

Subsequently, we obtain the representative vectors of the associated text of the zero pronoun by averaging the row vectors in each representative matrix (namely, M_{pre} and M_{fol}). After that, these two vectors are concatenated as the representative vector of the zero pronoun. Our experiments show that the attention-based model leads to a considerable improvement from the non-attentive model, indicating that the self-attention mechanism can help to better encode the zero pronoun, focusing on informative parts of the associated texts.

3.2.2 Modeling Candidate Antecedent

We here build the candidate antecedent encoder by using an RNN architecture, whose input is comprised by the words in the candidate antecedent. In an effort to better align the more informative parts of phrases to the anaphoric zero pronoun, we here integrate an attention technique into our model. In this work, we use a gating-function as our attention mechanism. Especially, given the representative vector of the anaphoric zero pronoun $v^{(zp)}$, the output vector and input embedding vector of RNN in the candidate mention part at time t, $h_t^{(np)}$ and e_t, the attention mechanism computes a gate as: $attention_t = att(e_t, h_t^{(np)}, v^{(zp)})$, where att is defined as:

$$s_t = tanh(W^{(att)} \cdot [e_t; h_t^{(np)}; v^{(zp)}] + b^{(att)}) \tag{11}$$

$$attention_t = \frac{exp(s_t)}{\sum_{t'=1}^{m} exp(s_{t'})} \tag{12}$$

where $W^{(att)}$ and $b^{(att)}$ are parameters to be learned, m is the number of words in the mention. After that, we regard the averaged attention-hidden vector as the vector-space semantic of the candidate antecedent, which takes considerations of a hierarchy of historical semantic:

$$\tilde{v}_{np} = \sum_{i=1}^{m} h_i^{(np)} * attention_i \tag{13}$$

3.2.3 Calculating Resolution Scores

After generating the representive vector of zero pronoun, v_{zp} and vectors of its candidates $\{\tilde{v}_{np1}, \tilde{v}_{np2}, ..., \tilde{v}_{npn}\}$, we calcualte the resolution score for each zero pronoun-candidate antecedent by using a two-layers feed-forwd nerual network. Taking v_{zp} and its i-th candidate mention v_{np_i} as inputs, our model calcuate the resolution score by going through two $tanh$ layers:

$$s_j = tanh(W_i^{(s)} \cdot s_{j-1} + b_j^{(s)}) \tag{14}$$

where $s_0 = [v(zp); v(np_i); v_i^{(fe)}]$, $W^{(s)}$ and $b^{(s)}$ are the parameters of this feed-forward neural network. In addition, to better capture the syntactics, position and other relations between an anaphoric zero pronoun and its candidates, we encode hand crafted features ($v^{(fe)}$) as inputs to our neural network model. We utilize the features from exsiting work on zero anaphora resolution (Chen and Ng, 2013; Chen and Ng, 2016), map them into vectors to estimate the resolution score for the zero pronoun-candidate mention pair as:

$$score_i = W^{(sco)} \cdot s_{-1} + b^{(sco)} \tag{15}$$

where $score_i$ denotes the probability of the i-th candidate mention (np_i) being predicted to be the antecedent, and s_{-1} is the output vector of the second hidden layer. After that, we obtain the resolution scores for all the candidates $\{score_1, score_2, ..., score_n\}$. The candidate mention with the biggest score is eventually selected to be the antecedent of the anaphoric zero pronoun.

3.3 Training Objective

Same as Yin et al. (2017b), we train our model by minimizing the cross entropy error of coreference classification. The training objective is defined as:

$$loss = -\sum_{t \in T} \sum_{np \in NP} \delta(zp, np) \log(P(zp, np)) \tag{16}$$

where T represents all training instances, NP is the candidate-set of the anaphoric zero pronoun zp; $\delta(zp, np)$ represents the coreference of zp and its candidate mention np: if they are coreference, $\delta(zp, np) = 1$ or otherwise, $\delta(zp, np) = 0$.

4 Experiments

4.1 Experiment Setup

4.1.1 Evaluation Metrics

Same as early work on Chinese zero pronoun resolution (Zhao and Ng, 2007; Chen and Ng, 2016; Yin et al., 2017a; Yin et al., 2017b), we manipulate to evaluate the quality of our model by Recall, Precision and F-score (denoted as F). More specifically, recall and precision are defined as:

$$Recall_{Res} = \frac{\#\ Res\ Hit}{\#\ AZP\ in\ Key} \tag{17}$$

$$Precision_{Res} = \frac{\#\ Res\ Hit}{\#\ AZP\ in\ Predictions} \tag{18}$$

where a "Res Hit" means that the anaphoric zero pronoun is successfully identified and successfully resolved to a candidate mention that is in the same coreference chain as in the golden answer key annotated in the dataset.

4.1.2 Experiment Settings

Same to existing work on Chinese zero pronoun resolution (Chen and Ng, 2016; Yin et al., 2017a; Yin et al., 2017b), we run experiments on the Chinese part of the OntoNotes-5.0 dataset[1] used in the Conll-2012 task. Because zero pronoun coreferences are only annotated in the training and development set, we thus train our model on the training dataset and evaluate the model on the development dataset. Table 1 is the statistics of our dataset.

	Documents	Sentences	Words	Anaphoric Zero Pronouns
Training	1,391	36,487	756K	12,111
Test	172	6,083	110K	1,713

Table 1: Statistics on the training and test dataset.

We use the recent zero pronoun resolution systems for Chinese as our baselines, namely, a learning-based model (Zhao and Ng, 2007); an unsupervised method (Chen and Ng, 2015); and others are deep-learning-based methods (Chen and Ng, 2016; Liu et al., 2017; Yin et al., 2017a; Yin et al., 2017b). As we are focusing on the anaphoric zero pronoun resolution, we run experiments by directly employing golden parse tree and golden anaphoric zero pronouns that are annotated in the dataset. In addition, documents in the datasets are from 6 sources: **BN** (Broadcast News), **NW** (Newswire), **BC** (Broadcast Conversation), **WB** (Web Blog), **TC** (Telephone Conversation) and **MZ** (Magazine). We report the overall results on the complete test dataset and also the result from the different source of the dataset.

[1] `http://catalog.ldc.upenn.edu/LDC2013T19`

	NW (84)	MZ (162)	WB (284)	BN (390)	BC (510)	TC (283)	**Overall**
Zhao and Ng (2007)	40.5	28.4	40.1	43.1	44.7	42.8	41.5
Chen and Ng (2015)	46.4	39.0	51.8	53.8	49.4	52.7	50.2
Chen and Ng (2016)	48.8	41.5	56.3	55.4	50.8	53.1	52.2
Yin et al. (2017b)	50.0	45.0	55.9	53.3	55.3	54.4	53.6
Yin et al. (2017a)	48.8	46.3	59.8	58.4	53.2	**54.8**	54.9
Liu et al. (2017)	59.2	51.3	60.5	53.9	55.5	52.9	55.3
Our model	**64.3**	**52.5**	**62.0**	**58.5**	**57.6**	53.2	**57.3**

Table 2: Experiment results on the test dataset, including the results on the overall dataset and different sources of the dataset. The first six columns show the results on the different source of documents and the last is the overall result. The strongest F-score in each row is in **bold**. The parenthesized number beside a source's name is the number of anaphoric zero pronouns in that source.

4.2 Hyperparameter

To tune the hyperparameters of our model, 20% of the training dataset are reserved as a held out development set. Such a strategy is also utilized in the baseline systems (Chen and Ng, 2016; Yin et al., 2017a). We randomly initialize the parameters and minimize the loss-function by Adagrad (Duchi et al., 2011) with learning-rate 0.003. The input embedding vector dimension is 100, the dimension of hidden layer of RNNs (namely, the u) is 256 and d_a is 128. Besides, we fix the dimensions of two hidden layer of the feed-forward neural network to 256 and 512. We add the dropout (Hinton et al., 2012) with a probability of 50% on the output of each layer. The code for this work is released in `https://github.com/qyyin/AttentionZP.git`.

4.3 Experiment Results

We report the experiment results (F-score) of our model and the baselines in Table 2. The number of hops of attention is fixed to be 2 ($r = 2$), where we gain the best result. We report the overall results on the complete test dataset and also the results for each source of documents. As we can observe that our model gains 57.3% in overall F-score, which significantly beats the best baseline system (Liu et al., 2017) by 2.0%. In addition, we run experiments on different sources of test corpus, as shown in the first six columns Table 2. The parenthesized number beside a source's name represents the number of anaphoric zero pronouns in that source. We can observe that our model improves performance significantly in 5 of 6 sources of the dataset. More specifically, our model beats the best baseline (Liu et al., 2017) on all documents in F-score: by 5.1% (source NW), 1.2% (source MZ), 1.5% (source WB), 4.6% (source BN), 2.1% (source BC) and 0.3% (source TC). The main reason why our model gains worse performance on source "TC" lie in the short length of text in this source, which makes our model hard to learn to capture useful information for expressing the zero pronouns. Besides, there are full of numerous verbose words such as "呃/Er", "哟/Yo", which brings difficulties for our model to accurately encode a zero pronoun by focusing on its informative contexts. More efforts could be performed in order to encode the associated text of a zero pronoun in a more efficient way, modeling word-sequences beyond the sentence boundary, for instance.

We show in Figure 2 the learning curve of our model on the development dataset. As we can observe, after the first epoch, the F-score on the test dataset is about 46%, and it gradually grows to 52% after about 30 iterations when performance starts to plateau. It is well accepted that modeling useful parts of associated text play an important role in encoding the zero pronouns. By applying the self-attentive mechanism, our model learns to focus on important parts of the contexts, revealing multi-aspect sentence-level information in a more efficient way than those without attention. On the other side, by assigning different attention to the words in a candidate mention with respect to the information of the counterpart zero pronoun, our model learns to encode candidate mentions in a more natural way. Hence, both of them bring benefit to selecting accurate antecedents, leading to better performance.

In addition, having multi-aspect sentence-level information is expected to afford more abundant in-

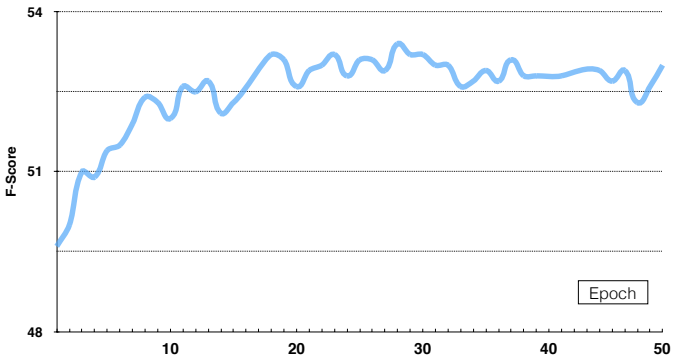


Figure 2: Learning curve of our model on the development dataset.

formation about the encoded zero pronoun, we thus evaluate how the improvement can be brought by tuning r in the self-attentive mechanism. We vary r from 2 to 6, as is shown in Figure3. We can observe that the best performance is reached when $r = 2$. The results are not confusing because we are tried to focus on the informative part for zero pronouns, and we cut the sentence into two separate parts that are zero pronoun-centric, when $r = 2$ means to attend on totally 4 parts of the sentence, thus our model performances well in this situation.

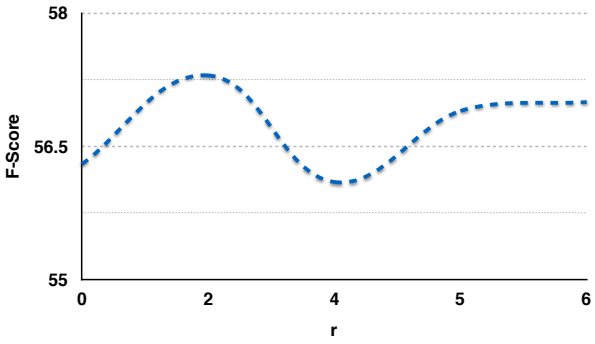


Figure 3: Effect of tuning r for encoding zero pronouns.

For illustrating the effect of the proposed attention mechanism for zero pronoun and candidate mentions, we also run an experiment without applying the attention mechanism for both zero pronouns and candidates, which is $r = 0$ in Figure 3. As we can observe that the performance drops by removing the attention mechanism. For the model without the attentive mechanism, the performance drops by 0.9% in F-score compared with that of the full model. Because that the proposed self-attentive mechanism for zero pronoun brings our model an ability to access multi-aspect sentence-level information, removing such an architecture unsurprising influences the results significantly. With an inspiration that not all the words are equally important for explaining the mention, the performance of removing the attention for candidate mentions is reasonable weak. All these show the benefit of our attention-based model.

Lastly, we give a case study to illustrate the power of our self-attentive mechanism, as is shown in Figure 4. From the figure, we can tell that the model successfully focuses on informative parts of the texts. Though there are redundancies between different hops of attention, our model can capture useful information for explaining the zero pronoun (denoted as "*pro*" in the picture). Our model learns to focus on the informative words such as "数字证书/digital certificate", which is essential for expressing the zero pronoun.

Figure 4: Heat maps of our attention-based model. In this case, we show the detailed attention weight taken by the attention matrix taken by the attention matrix ($r = 2$). Darker color means higher weight.

5 Conclusion

We proposed a novel attention-based neural network model for Chinese zero pronoun resolution. Using recent advances in attention mechanism, we developed a self-attentive architecture for modeling zero pronouns, which enables our model to focus on parts of the associated texts. In addition, we also investigated an attention-based candidate antecedent encoder that learns to model important parts of the noun phrases with respect to the representative vector of zero anaphoras. Our experiments demonstrated that our method significantly surpasses the state-of-the-art on a benchmark dataset for anaphoric zero pronoun resolution. Future work will evaluate our model on other natural language processing problems, such as anaphora resolution for Chinese and English. We also plan to investigate training the anaphora-specific embedding that could better reveal the descriptive attribute for the zero anaphoras.

Acknowledgments

This work has been supported by the Major State Basic Research Development 973 Program of China (No.2014CB340503), National Natural Science Foundation of China (No.61472105 and No.61502120). We are grateful to Xuxiang, Xiaocheng Feng and anonymous reviewers for their useful feedback. By the meaning by Harbin Institute of Technology, the contact author of this work is Yu Zhang.

References

Chen Chen and Vincent Ng. 2013. Chinese zero pronoun resolution: Some recent advances. In *EMNLP*, pages 1360–1365.

Chen Chen and Vincent Ng. 2014. Chinese zero pronoun resolution: An unsupervised approach combining ranking and integer linear programming. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Chen Chen and Vincent Ng. 2015. Chinese zero pronoun resolution: A joint unsupervised discourse-aware model rivaling state-of-the-art resolvers. In *Proceedings of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, page 320.

Chen Chen and Vincent Ng. 2016. Chinese zero pronoun resolution with deep neural networks. In *Proceedings of the 54rd Annual Meeting of the ACL*.

Susan P Converse. 2006. Pronominal anaphora resolution in chinese.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Antonio Ferrández and Jesús Peral. 2000. A computational approach to zero-pronouns in spanish. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 166–172. Association for Computational Linguistics.

Na-Rae Han. 2006. *Korean zero pronouns: analysis and resolution*. Ph.D. thesis, Citeseer.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Ryu Iida and Massimo Poesio. 2011. A cross-lingual ilp solution to zero anaphora resolution. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 804–813. Association for Computational Linguistics.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2006. Exploiting syntactic patterns as clues in zero-anaphora resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 625–632. Association for Computational Linguistics.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2007. Zero-anaphora resolution by learning rich syntactic pattern features. *ACM Transactions on Asian Language Information Processing (TALIP)*, 6(4):1.

Ryu Iida, Kentaro Torisawa, Chikara Hashimoto, Jong-Hoon Oh, and Julien Kloetzer. 2015. Intra-sentential zero anaphora resolution using subject sharing recognition. *Proceedings of EMNLP'15*, pages 2179–2189.

Ryu Iida, Kentaro Torisawa, Jong-Hoon Oh, Canasai Kruengkrai, and Julien Kloetzer. 2016. Intra-sentential subject zero anaphora resolution using multi-column convolutional neural network. In *Proceedings of EMNLP*.

Hideki Isozaki and Tsutomu Hirao. 2003. Japanese zero pronoun resolution based on ranking rules and machine learning. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 184–191. Association for Computational Linguistics.

Fang Kong and Guodong Zhou. 2010. A tree kernel-based unified framework for chinese zero anaphora resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 882–891. Association for Computational Linguistics.

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

Ting Liu, Yiming Cui, Qingyu Yin, Shijin Wang, Weinan Zhang, and Guoping Hu. 2017. Generating and exploiting large-scale pseudo training data for zero pronoun resolution. In *ACL*.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212.

Ryohei Sasano and Sadao Kurohashi. 2011. A discriminative approach to japanese zero anaphora resolution with large-scale lexicalized case frames. In *IJCNLP*, pages 758–766.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 214–224.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(2):207–238.

Qingyu Yin, Yu Zhang, Weinan Zhang, and Ting Liu. 2017a. Chinese zero pronoun resolution with deep memory network. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1309–1318.

Qingyu Yin, Yu Zhang, Weinan Zhang, and Ting Liu. 2017b. A deep neural network for chinese zero pronoun resolution. In *IJCAI*.

Shanheng Zhao and Hwee Tou Ng. 2007. Identification and resolution of chinese zero pronouns: A machine learning approach. In *EMNLP-CoNLL*, volume 2007, pages 541–550.

They Exist! Introducing Plural Mentions to Coreference Resolution and Entity Linking

Ethan Zhou
Computer Science
Emory University
Atlanta, GA 30322
ethan.zhou@emory.edu

Jinho D. Choi
Computer Science
Emory University
Atlanta, GA 30322
jinho.choi@emory.edu

Abstract

This paper analyzes arguably the most challenging yet under-explored aspect of resolution tasks such as coreference resolution and entity linking, that is the resolution of plural mentions. Unlike singular mentions each of which represents one entity, plural mentions stand for multiple entities. To tackle this aspect, we take the character identification corpus from the SemEval 2018 shared task that consists of entity annotation for singular mentions, and expand it by adding annotation for plural mentions. We then introduce a novel coreference resolution algorithm that selectively creates clusters to handle both singular and plural mentions, and also a deep learning-based entity linking model that jointly handles both types of mentions through multi-task learning. Adjusted evaluation metrics are proposed for these tasks as well to handle the uniqueness of plural mentions. Our experiments show that the new coreference resolution and entity linking models significantly outperform traditional models designed only for singular mentions. To the best of our knowledge, this is the first time that plural mentions are thoroughly analyzed for these two resolution tasks.

1 Introduction

Resolution tasks such as coreference resolution and entity linking are challenging because they require a holistic view of a document (or across multiple documents) to find correct entities. Although many models have been proposed for these tasks (Clark and Manning, 2016; Francis-Landau et al., 2016; Wiseman et al., 2016; Gupta et al., 2017; Lee et al., 2017), most of them are focused on singular mentions such that they are insufficient for resolving the other type of mentions, plural, although the amount of plural mentions is not negligible in practice.[1] Table 1 illustrates how mentions are annotated for coreference resolution by the CoNLL'12 shared task (Pradhan et al., 2012) and our proposed work. In the CoNLL'12 annotation, the plural mention *They*$_8$ is grouped with the noun phrase [*Mary*$_1$ and *John*$_2$]$_3$; however, the other plural mention *We*$_7$ becomes a singleton because there is no noun phrase representing such an entity. Since CoNLL'12 limits each plural mention to be linked to a single noun phrase, it loses connections to individual entities that exist within the document but not grouped as a noun phrase.

Document	[*Mary*$_1$ and *John*$_2$]$_3$ came to see *me*$_4$ yesterday. *She*$_5$ looked happy, and so did *he*$_6$. *We*$_7$ had a great time together. *They*$_8$ left around noon.
CoNLL'12	{*Mary*$_1$, *She*$_5$}, {*John*$_2$, *he*$_6$}, {[*Mary*$_1$ and *John*$_2$]$_3$, *They*$_8$}, {*me*$_4$}, {*We*$_7$}
Our Work	{*Mary*$_1$, *She*$_5$, *We*$_7$, *They*$_8$}, {*John*$_2$, *he*$_6$, *We*$_7$, *They*$_8$}, {*me*$_4$, *We*$_7$}

Table 1: Snippets of how mentions are annotated by the CoNLL'12 shared task and our work.

In our work, the plural mentions *We*$_7$ and *They*$_8$ are linked to multiple entities that those mentions refer to. This allows higher-level NLP tasks such as question answering or machine translation to reason more explicitly about those entities while adding another level of challenges to the resolution tasks. In this paper, we first present the annotation scheme for resolving plural mentions that is used to expand the corpus

[1] A singular mention is a noun phrase that refers to exactly one entity while a plural mention is one that refers to multiple entities.

Proceedings of the 27th International Conference on Computational Linguistics, pages 24–34
Santa Fe, New Mexico, USA, August 20-26, 2018.

provided by the Character Mining project (Section 3). We then introduce a novel algorithm for coreference resolution that selectively creates clusters for singular and plural mentions, as well as evaluation metrics to handle plural mentions for coreference resolution (Section 4). We also present a new deep learning-based entity linking model that jointly identifies both singular and plural mentions (Section 5). All models are evaluated on our dataset (Section 6); the experiments reveal significant improvement from our new models compared to the previous state-of-the-art models dedicated for singular mentions. As far as we can tell, this is the first time that such annotation for plural mentions is provided in a large enough scale that deep learning models can be trained on, at the same time, machine learning models are developed to achieve promising results for the resolution of plural mentions.

2 Related Work

Chen and Choi (2016) were the first to introduce the task of character identification and provided a new corpus based on TV show transcripts. Given a dialogue transcribed in text where all mentions are detected, character identification aims to find the entity for each personal mention, who may or may not be active in the dialogue. Unlike most other entity linking tasks focusing on Wikification, this task is challenging because it is dialogue-based where the entities are general characters in the show. This corpus was later expanded by Chen et al. (2017) who added annotation for the ambiguous entity types. In this work, we expanded the corpus further by doubling the size of the annotation and adding new annotation for plurals.

The character identification corpus can be used for both coreference resolution and entity linking tasks. Our approach to coreference resolution was partially motivated by the previous works, Clark and Manning (2016) and Durrett et al. (2013), who tackled the general cases of coreference resolution including plurals; however, since their approaches were based on the annotation provided by CoNLL'12, they did not handle plural mentions to our satisfaction (Table 1). Jain et al. (2004) presented a rule-based system for resolving plural mentions, which was limited to unambiguous plural types. Our work is distinguished because we handle both ambiguous and unambiguous types of plural mentions, which makes it more challenging. Chen et al. (2017) presented an entity linking model that identified the real entity of each singular mention, which we adapted to develop a new multi-task learning model that jointly handles singulars and plurals.

3 Corpus

3.1 Annotation

The Character Mining project provides transcripts from the TV show *Friends* for all ten seasons in JSON.[2] A subset of the first two seasons of this show was annotated for the task of character identification by Chen et al. (2017), who made it publicly available through the International Workshop on Semantic Evaluation (SemEval 2018).[3] Given this annotation, we expanded the corpus as follows:

1. We realized that about 20% of the first two seasons were not covered by the previous annotation. Following the annotation guidelines suggested by Chen et al. (2017), we completed the annotation for the first two seasons and further annotated two more seasons. As a result, the first four seasons are completely annotated for character identification in our corpus.

2. There were quite a few mismatches among the speaker and the entity labels in the previous annotation. For instance, while mentions were annotated by the entity's full name such as `Monica_Geller`, some utterances were paired with speaker labels represented by only the first name, `Monica`, which could cause confusions for machine learning models. We manually went through the entire annotation and made sure the speaker and the entity labels were coherent across all seasons.

3. The previous annotation consisted of only singular mentions such that each mention was guaranteed to be linked to exactly one entity. We annotated plural mentions for the first four seasons through crowdsourcing. Unlike singular mentions that were automatically recognized by the heuristic-based

[2]Character Mining: `https://github.com/emorynlp/character-mining`
[3]SemEval 2018 Task 4: `https://competitions.codalab.org/competitions/17310`

mention detector (Chen and Choi, 2016), plural mentions in our corpus were manually detected by the crowd workers who were also asked to link each plural mention to a set of its referent entities.

The annotation guidelines used for singular mentions are adapted to annotate plural mentions as well such that the only difference in annotation between these two types of mentions is the number of entities to which the mentions refer. Formally, each mention m is annotated with a set of entities E, where each element in E belongs to one of the following four groups:

1. Known entities: include all the primary and secondary characters recurring in the show.
2. `GENERIC`: indicates actual characters in the show whose identities are unknown across the show: e.g., That *waitress* is really cute, I am going to ask *her* out.
3. `GENERAL`: indicates mentions referring to a general case rather than a specific entity: e.g., The ideal *guy* you look for doesn't exist.
4. `OTHER`: indicates actual characters in the show whose identities are unknown in this dialogue but revealed in some other dialogue.

The `COLLECTIVE` type, used to distinguish the plural usage of the pronoun *you* in the previous annotation, is discarded in our annotation because each *you* is now annotated with a set of entities such that the plural usage can be deterministically distinguished by the size of its entity set.

Speaker	**Utterance**
Jack	And I_1 read about these *women*$_2$ trying it all, and I_3 thank God '*Our*$_4$ *Harmonica*$_5$' doesn't have this problem.
Monica	So, *Ross*$_6$, what's going on with *you*$_7$ two? Any stories? No little anecdotes to share with *mom*$_8$ and *dad*$_9$?
Ross	Okay, I_{10} just got this from the *guy*$_{11}$ next to *me*$_{12}$. *He*$_{13}$ was selling a whole bunch of stuff.

$\{I_1, I_3, Our_4, dad_9\} \rightarrow$ `Jack` $\{Our_4, mom_8\} \rightarrow$ `Judy`, $\{Harmonica_5\} \rightarrow$ `Monica`, $\{Ross_6, you_7, I_{10}, me_{12}\} \rightarrow$ `Ross`, $\{women_2\} \rightarrow$ `GENERAL`, $\{you_7\} \rightarrow$ `OTHER`, $\{guy_{11}, He_{13}\} \rightarrow$ `MAN_1`

Table 2: An example of entity annotation in our corpus, where *Our*$_4$ and *you*$_7$ are the plural mentions.

Table 2 shows examples of all types of entities for both singular and plural mentions. The mention *women*$_2$ does not refer to any specific character so it is identified as `GENERAL`. Both the mentions *guy*$_{11}$ and *He*$_{13}$ refer to a specific person whose identity is never revealed so it is annotated with the generic type, `MAN_1`. There are two plural mentions, *Our*$_4$ and *you*$_7$, which are handled differently. All entities of *Our*$_4$ can be identified from the context of this dialogue so it is annotated with the known entities `Jack` and `Judy`. However, only one of *you*$_7$ can be identified in this context so it is annotated with the known entity `Ross` and also `OTHER`, implying that it refers to some other entity that can be identified in a separate dialogue. This method is used to distinguish non-immediately identifiable entities from the generic case of `MAN_1` whose identity is unknown across the entire show.

3.2 Analytics

Table 3 shows the statistics of our corpus. Compared to the previous annotation including 18,608 mentions, our corpus is comprised of 47,367 annotated mentions, which is 2.5 times larger. Plural mentions together compose about 9% of the entire dataset, which is significant enough to make a difference in resolution. Each cluster contains about 6 mentions on average when each scene is treated as an independent dialogue.

All mentions were double-annotated by crowd workers. From this double-annotation, Cohen's kappa score of 56.88% was achieved for plural mentions, which was about 20% lower than the one achieved for singular mentions (Chen and Choi, 2016). The lower inter-annotator agreement was expected due to the high complexity of this task. A subset of the disagreed annotation was manually adjudicated by experts, from which we found that taking the union of the entity sets annotated by two workers would effectively give the correct set of entities for each of those disagreed plural mentions. Thus, a vast amount of plural mentions were pseudo-adjudicated by taking their unions of double-annotation.

Season	General				Mention			Entity	
	Episode	Scene	Utterance	Speaker	Singular	Plural	Total	Cluster	Type
1	24	326	5,968	107	10,313	1,147	11,460	2,162	270
2	24	293	5,747	107	10,521	1,156	11,677	1,934	285
3	25	348	6,495	108	11,458	907	12,365	1,925	230
4	24	334	6,318	100	10,726	1,139	11,865	1,881	175
Total	97	1,301	24,528	331	43,018	4,349	47,367	7,902	781

Table 3: The overall statistics of our corpus. All columns show raw counts except that the speaker column and the type column in the entity section give the set counts of all speakers and entities, respectively.

Table 4 shows the distributions of entity types. The primary characters compose about 67% of all mentions whereas the ambiguous types together compose about 8.6%, which implies that the majority of mentions can be linked to known entities. Notice that the total count of `GENERAL` increases by 554 from Seasons 1-2 to 3-4, whereas the total count of `OTHER` decreases by 654 for those seasons; these two ambiguous entity types are easily confused because they do not refer to any specific entity within the dialogue. Considering that annotation tasks for the first two seasons were mostly conducted by Chen et al. (2017) whereas the next two seasons were conducted by us, it is possible that our crowdsourcing instructions were more biased towards `GENERAL` than `OTHER`, which we will analyze in the future.

Season	Known Entities		Ambiguous Entities			Total
	Primary	Secondary	`GENERIC`	`GENERAL`	`OTHER`	
1	9,247	3,616	214	641	463	14,181
2	9,591	3,704	184	598	455	14,532
3	9,491	3,512	200	896	136	14,235
4	9,807	3,181	112	897	128	14,125
Total	38,136	14,013	710	3,032	1,182	57,073

Table 4: The distributions of entity types. Each column shows the number of mentions annotated with the corresponding entity type. Note that the total number of mentions here is different from the one in Table 3 (57,073 vs. 47,367) because each plural mention is counted more than once in this table.

4 Coreference Resolution

The presence of plural mentions brings up several challenges for coreference resolution. First, the search scope becomes broader. For each mention m_j, a typical coreference resolution system would find another mention m_i that is referent to m_j, and assigns m_j to the cluster C_i that m_i belongs to if it exists; otherwise, creates a new cluster and assigns both m_i and m_j to that cluster.[4] As soon as m_j is assigned, the search can stop for m_j. This strategy works for singular mentions but fails with plural mentions because they can be assigned to more than one cluster. Second, the referent relations are no longer transitive. Let $m_i \leftarrow m_j$, $m_i \rightarrow m_j$, $m_i \leftrightarrows m_j$ stand for referent relations such that m_j is referent to m_i, m_i is referent to m_j, m_i is coreferent to m_j, respectively. Then, $m_i \leftrightarrows m_j$ and $m_j \leftrightarrows m_k$ would imply $m_i \leftrightarrows m_k$ for singular mentions, but this transitivity fails with plural mentions when m_j belongs to two different clusters $C_i = \{m_i, m_j\}$ and $C_k = \{m_j, m_k\}$ such that m_i and m_k have no referent relation. Third, some of the popular evaluation metrics for coreference resolution such as B^3 (Bagga and Baldwin, 1998) are not necessarily designed for plural mentions such that they need to be revisited.

Section 4.1 introduces our new coreference resolution algorithm that selectively creates clusters with respect to different mention types. This algorithm ensures singular mentions representing different entities get assigned to separate clusters. For example, let m_p be a plural mention and m_i be a singular mention such that $m_p \rightarrow m_i$. When the referent relation is found, the cluster C_i is created and both m_p and m_i are assigned to C_i. Let m_j be another singular mention such that $m_p \rightarrow m_j$. Now, the algorithm must decide

[4]The term 'cluster' indicates a group of mentions that refer to the same entity within a document such that each cluster represents a distinct entity although a cluster in one document can represent the same entity as another cluster in a different document.

whether to assign m_j to C_i or create another cluster C_j for m_j. If $m_i \leftrightarrows m_j$, m_j should be assigned to C_i; otherwise to C_j. Our algorithm allows a model to learn this decision during training so that the clusters can be created accordingly during decoding. Section 4.3 describes how existing evaluation metrics can be adjusted to evaluate both singular and plural mentions for coreference resolution, which is the first time that these metrics are adapted for plural mentions linked to multiple entities.

4.1 Algorithm

For each mention m_j, our algorithm compares it against all of the preceding mentions m_i to determine whether or not they are referent, where i and j are the ordered indices such that $0 < i < j$. Additionally, two more mentions, m_g and m_o, are compared to m_j that represent the `GENERAL` and the `OTHER` types, respectively (Section 3.1). For each mention pair (m_i, m_j), the algorithm assigns one of the following three labels for multi-classification:

1. `N`: m_i is not referent to m_j.
2. `L`: m_j gets assigned to the cluster that m_i belongs to. If m_i does not yet belong to any cluster, a new cluster C_i is created and both m_i and m_j are assigned to C_i.
3. `R`: m_i gets assigned to the cluster that m_j belongs to. If m_j does not yet belong to any cluster, a new cluster C_j is created and both m_i and m_j are assigned to C_j.

During training, labels are determined by consulting the oracle. `L` is labeled if m_i is a singular mention. `R` is labeled if m_i is plural and m_j is singular. `N` is labeled for all the other cases. Notice that this algorithm does not allow any plural mention to be directly linked to another plural mention; in other words, it does not create any cluster consisting of only plural mentions. Plural mentions can still be indirectly linked through clusters created for singular mentions. The creation of clusters comprising only plural mentions would not help identifying the known entities of those mentions, which defeats the purpose of character identification. It is possible to link plural mentions directly by using the `GENERIC` type (Section 3.1), which is not adapted to annotate entities for plural mentions in the current annotation scheme.

$[m_i] \rightarrow \{$**N, L, R**$\}$	m_j	**Clusters**
[G, O] → N	1	$\emptyset_g, \emptyset_o$
[O, 1] → N, [G] → L	2	$\{2\}_g, \emptyset_o$
[G, O, 2] → N, [1] → L	3	$\{2\}_g, \emptyset_o, \{1,3\}_1$
[G, O, 2] → N, [1, 3] → L	4	$\{2\}_g, \emptyset_o, \{1,3,4\}_1$
[G, O, 1..4] → N	5	$\{2\}_g, \emptyset_o, \{1,3,4\}_1$
[G, O, 1..5] → N	6	$\{2\}_g, \emptyset_o, \{1,3,4\}_1$
[G, 1..5] → N, [O, 6] → L	7	$\{2\}_g, \{7\}_o, \{1,3,4\}_1, \{6,7\}_6$
[G, O, 1..3, 5..7] → N, [4] → R	8	$\{2\}_g, \{7\}_o, \{1,3,4\}_1, \{6,7\}_6, \{4,8\}_8$
[G, O, 2, 5..8] → N, [1, 3, 4] → L	9	$\{2\}_g, \{7\}_o, \{1,3,4,9\}_1, \{6,7\}_6, \{4,8\}_8$
[G, O, 1..5, 8, 9] → N, [6] → L	10	$\{2\}_g, \{7\}_o, \{1,3,4,9\}_1, \{6,7,10\}_6, \{4,8\}_8$
[G, O, 1..10] → N	11	$\{2\}_g, \{7\}_o, \{1,3,4,9\}_1, \{6,7,10\}_6, \{4,8\}_8$
[G, O, 1..5, 8, 9, 11] → N, [6, 10] → L	12	$\{2\}_g, \{7\}_o, \{1,3,4,9\}_1, \{6,7,10,12\}_6, \{4,8\}_8$
[G, O, 1..10, 12] → N, [11] → L	13	$\{2\}_g, \{7\}_o, \{1,3,4,9\}_1, \{6,7,10,12\}_6, \{4,8\}_8, \{11,13\}_{11}$
Singleton Processing		$\{2\}_2, \{7\}_7, \{1,3,4,9\}_1, \{6,7,10,12\}_6, \{4,8\}_8, \{11,13\}_{11}, \{5\}_5$

Table 5: A demonstration of our algorithm using the example in Table 2. The m_j column indicates the index of m_j that the algorithm is currently processing. The first column shows the labels generated for all mention pairs (m_i, m_j), where the indices of m_i are indicated inside the square brackets (e.g, [O, 1] stands for m_o and m_1) and the labels are indicated next to the right arrows (e.g., → L). The clusters column shows the list of entity sets created by taking the labeling information from the first column.

Table 5 depicts how this algorithm finds the referent relations for all the mentions in Table 2. Note that the special mentions m_g and m_o are considered singular and placed prior to any other mention here. The algorithm labels `L` for (m_g, m_2), which makes *women*$_2 \in C_g$ representing `GENERAL`. For m_4, it labels `L`

for m_1 and m_2, which makes *Our*$_4$ $\in C_1$ representing JACK; although *Our*$_4$ is a plural mention, it gets assigned to only one cluster at the moment since the other entity has yet been revealed. For m_7, it labels L for both m_o and m_6, which makes *you*$_7$ $\in C_o$ representing OTHER and $\in C_6$ representing Ross. For (m_4, m_8), it labels R because m_4 is plural and m_8 is singular, which creates a new cluster C_8 and assigns both *Our*$_4$ and *mom*$_8$ to C_8. Once all mention pairs are compared, the algorithm collects mentions that are not assigned to any cluster, and makes them singletons such that *Harmonica*$_5$ becomes the singleton C_5. Furthermore, every mention that belongs to either C_g or C_o gets turned into a singleton such that C_2 and C_7 are created at the end. This is because mentions assigned to those ambiguous types are not referent to one another, if they were, they would have been assigned to GENERIC instead.

4.2 Learning Model

Our learning model uses a modified version of the Agglomerative Convolutional Neural Networks (ACNN) introduced by Chen et al. (2017). This architecture incorporates multiple sets of features and learns the most optimized feature combination at each convolution layer. It also allows the model to dynamically accumulate the most salient features for eventual inclusion in the mention and mention pair embeddings. ACNN takes a mention pair (m_i, m_j), performs multiple convolutions to extract features from different groups (CONV$_1$), combines the extracted features among groups using more convolutions (CONV$_2$), and generates mention embeddings $r_s(m_i)$ and $r_s(m_j)$. These mention embeddings are then concatenated with discrete features $\phi_d(m)$ and combined through convolutions to generate a mention-pair embedding $r_p(m_i, m_j)$. The mention-pair embedding together with pairwise features $\phi_p(m_i, m_j)$ are used to make a binary classification for m_i and m_j being referent or not.

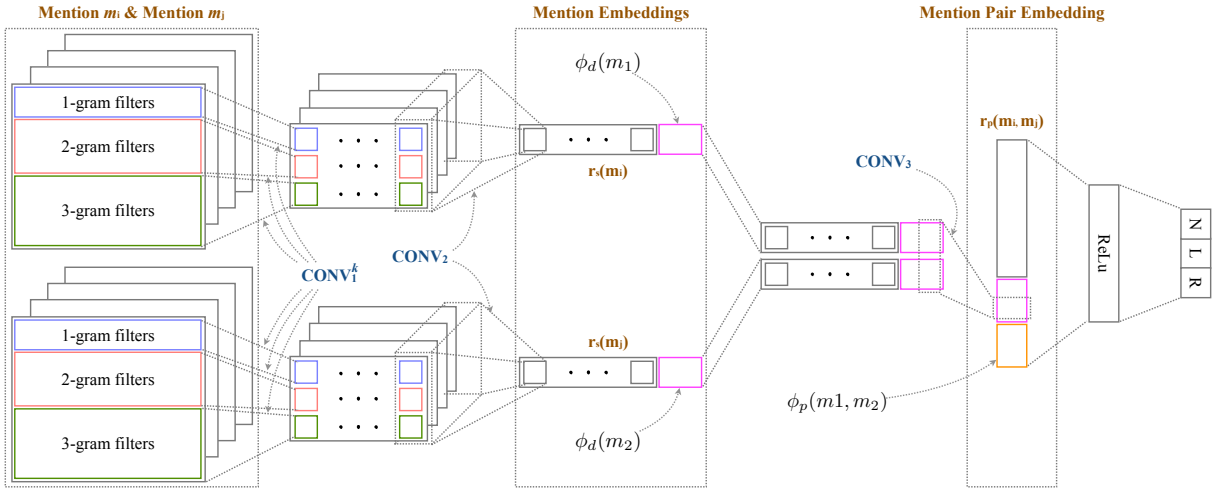


Figure 1: The overview of our coreference resolution model using the multi-class ACNN.

To be adapted to our coreference resolution algorithm in Section 4.1, ACNN is modified at the output layer to include three labels, N, L, and R, such that it is optimized for multi-class instead of binary classification. The modified ACNN, called the multi-class ACNN, generates mention embeddings, $r_s(m_i)$ and $r_s(m_j)$, as well as mention pair embeddings, $r_p(m_i, m_j)$, which are used to create cluster embeddings and fed as input to our entity linking model in Section 5.1.

4.3 Evaluation Metrics

Three metrics proposed by the CoNLL'12 shared task (Pradhan et al., 2012), B^3, CEAF$_{\phi_4}$, and BLANC, are used to evaluate our coreference resolution models. B^3 (Bagga and Baldwin, 1998) is a mention-based metric that measures precision (P) and recall (R) as follows (D: a set of documents, N: the total number of mentions in D, $C_m^{s/o}$: the cluster from the system (s) or the oracle (o) that the mention m belongs to):

$$P = \frac{1}{N} \sum_{d \in D} \sum_{m \in d} \frac{|C_m^s \cap C_m^o|}{|C_m^s|} \qquad R = \frac{1}{N} \sum_{d \in D} \sum_{m \in d} \frac{|C_m^s \cap C_m^o|}{|C_m^o|}$$

In our case, each mention can be assigned to more than one cluster; thus, C^*_m is replaced by the union of all clusters that the mention m belongs to, which enables this metric to evaluate plural mentions.

CEAF$_{\phi_4}$ (Luo, 2005) is an entity-based metric that first creates a similarity matrix $M \in \mathbb{R}^{|S|\times|O|}$ where S and O are the sets of clusters produced by the system and the oracle, respectively. It then measures the similarity between every pair of clusters $(C^s, C^o) \in S \times O$ where $s \in [1, |S|]$ and $o \in [1, |O|]$ such that:

$$M_{s,o} = \frac{2 \times |C^s \cap C^o|}{|C^s| + |C^o|}$$

Given this similarity matrix, the Hungarian algorithm is used to find the list $\mathcal{H}$ that contains similarity scores from the most similar matching pairs of clusters $(C^s, C^o) \in S \times O$ such that $|\mathcal{H}| = \min(|S|, |O|)$. Finally, the overall similarity between S and O is measured as $\Phi = \sum_{\phi\in\mathcal{H}} \phi$, and precision and recall are measured as $P = \Phi/|S|$ and $R = \Phi/|O|$, respectively. Since CEAF$_{\phi_4}$ is entity-based, the metric can be used to evaluate plural mentions without any modification. The potential pitfall is that certain clusters may include a greater number of plural mentions than singular mentions, in which case, distinct clusters with similar sets of plural mentions may yield a high similarity score. However, plural mentions make up less than 10% of the dataset, so we are not concerned about these plural-majority clusters, since most if not all clusters would be dominated by singular mentions.

BLANC (Recasens and Hovy, 2011) is a link-based metric. Let L_s and L_o be the sets of links generated by the system (s) and the oracle (o), respectively. Let G be the set of all possible links between every pair of mentions whether or not they are referent. This metric first creates a confusion matrix $B \in \mathbb{R}^{2\times2}$ such that $B_{0,0} = |L_s \cap L_o|$, $B_{0,1} = |L_o - L_s|$, $B_{1,0} = |L_s - L_o|$, and $B_{1,1} = |(G - L_s) \cap (G - L_o)|$. It then measures precision and recall for referent links (P_c and R_c) and also for non-referent links (P_n and R_n):

$$P_c = \frac{B[0,0]}{B[0,0] + B[1,0]} \quad R_c = \frac{B[0,0]}{B[0,0] + B[0,1]} \quad P_n = \frac{B[1,1]}{B[1,1] + B[0,1]} \quad R_n = \frac{B[1,1]}{B[1,1] + B[1,0]}$$

$$F1_c = \frac{2 \times P_c \cdot R_c}{P_c + R_c} \qquad F1_n = \frac{2 \times P_n \cdot R_n}{P_n + R_n}$$

Finally, precision, recall, and F$_1$-score are measured as $P = {}^{P_c+P_n}/_2$, $R = {}^{r_c+r_n}/_2$, and $F_1 = {}^{F1_c+F1_n}/_2$. Note that we decide to replace MUC (Vilain et al., 1995), another popular metric used by the CoNLL'12 shared task, with BLANC because both are link-based and BLANC takes singletons into consideration, which consume a large portion of our dataset (over 20%), whereas MUC does not so that BLANC is more appropriate for our case. It is worth mentioning that a separate confusion matrix B_d is constructed for each document d such that $B = \sum_{d\in D} B_d$ where B_d is based on links only in d. This prevents potential inflation of $B[1, 1]$, which could become huge if it were to be measured across the entire dataset. BLANC can also be used to evaluate plural mentions without any modification because each link is treated independently regardless of its mention type in this metric.

5 Entity Linking

5.1 Multi-Task Learning

The task of character identification requires each mention to be identified by the names of actual characters (e.g., `Monica`, `Ross` in Table 2). Figure 2 gives the overview of our entity linking model, which adapts the underlying architecture from the entity linking model proposed by Chen et al. (2017) and generalizes it to jointly handle singular and plural mentions. It assumes the output from ACNN in Section 4.2 such that for each mention m_i, the embedding of that mention and the set of clusters $\{C_1, \ldots, C_k\}$ that m_i belongs to are taken. For each cluster C_a, ACNN gives the list of mention pair embeddings $m^{C_a}_{i,j}$, where $m_i, m_j \in C_a$. Similarly to the previous model, the cluster embedding and the cluster pair embedding are created. Unlike the previous model, our model creates multiple cluster and cluster pair embeddings when m is assigned to more than one cluster during coreference resolution so that the average vectors of those embeddings are generated, which get concatenated with the mention embedding of m_i and passed onto the fully-connected layers for prediction.

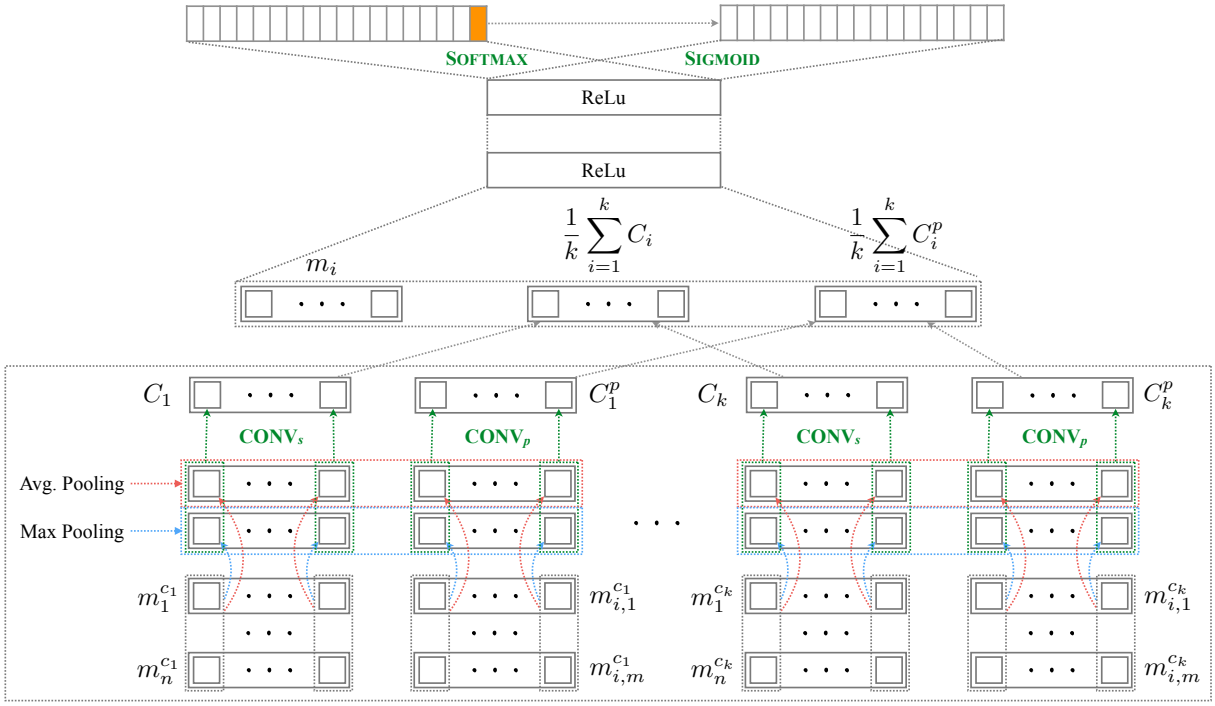


Figure 2: The overview of our entity linking model using multi-task learning.

The final `ReLu` layer is fed to two output layers optimized by softmax and sigmoid functions, respectively. The dimension of the output layer from softmax is $|E| + 1$ where E is the set of all entities such that each cell represents an entity and the extra cell gives an indication of m being plural. When this extra cell is predicted, the output layer from sigmoid is used, whose dimension is $|E|$, to predict multiple entities for m. Since the sigmoid function optimizes each cell to be between 0 and 1, any entity whose score is greater than 0.5 is taken. These two output layers are optimized jointly, treating the resolution of singular and plural mentions as multi-task learning.

5.2 Evaluation Metrics

Two metrics are used to evaluate the entity linking models. One is the micro-average F1 score whose precision (P) and recall (R) are measured as follows (D: a set of documents, $E_m^{s/o}$: the set of entities found for m by the system (s) or the oracle (o)):

$$P = \frac{\sum_{d \in D} \sum_{m \in d} |E_m^s \cap E_m^o|}{\sum_{d \in D} \sum_{m \in d} |E_m^s|} \qquad R = \frac{\sum_{d \in D} \sum_{m \in d} |E_m^s \cap E_m^o|}{\sum_{d \in D} \sum_{m \in d} |E_m^o|}$$

The micro-average F1 tends to weigh more on frequently occurring entities so it is useful if you need to know the raw prediction power of your model. The other is macro-average F1 score that measures the micro-average F1 for each entity e, say F_1^e, and takes the average, that is $1/|E| \sum_{e \in E} F_1^e$ where E is the set of all entities. The macro-average F1 treats all entities evenly so it is useful if you need to optimize your model to make correct predictions for as many entities as possible.

6 Experiments

6.1 Configuration

Experiments are conducted on two tasks, coreference resolution and entity linking. For both tasks, models from Chen et al. (2017) are used to establish strong baseline (CZC). Since they take only singular mentions, a pseudo-singular dataset is created where exactly one entity is chosen for each plural mention based on the closest matching previous speaker or if there is none, chosen randomly. Thus, the models trained on this pseudo-singular dataset always predicts one entity per mention. These models are compared to our

models described in Sections 4 and 5 (Ours). Additionally, CZC models are evaluated on the singular-only dataset (S-only) where all plural mentions are filtered out, which should give an intuition of how much impact the addition of plural mentions has on the predictions for singular mentions. All results reported from these experiments are averages of three randomly initialized trials. The corpus in Section 3 is split into training, development, and evaluation sets, where all models are tuned on the development set and the best models are tested on the evaluation set. Episodes 1–19, 20–21, and the rest from each season are used to generate the training, development, and evaluation sets, respectively.

6.2 Coreference Resolution

Table 6 shows that our coreference model is capable of learning to handle plural mentions effectively while significantly outperforms the CZC model. The CZC model is trained on the pseudo-singular dataset but evaluated on the full dataset by the metrics adjusted for plurals (Section 4.3) such that it is penalized for not predicting multiple entities for plural mentions. Both the B^3 and BLANC metrics show a similar trend that the CZC and our models achieve higher precision and recall, respectively, whereas our model dominates both precision and recall for the $CEAF_{\phi_4}$ metric. The remarkable gap in performance between these two models signals that our model finds referents for plural mentions well without compromising its ability to find referents for singular mentions. The S-only model gives comparable performance as the one reported by Chen et al. (2017), ensuring that our implementation of the CZC model is robust.

	B^3			**$CEAF_{\phi_4}$**			**BLANC**		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
CZC	**84.5±0.6**	60.7±0.2	70.6±0.3	49.0±0.8	63.7±0.3	55.4±0.6	**81.2±1.0**	73.3±0.4	75.9±0.5
Ours	83.8±1.5	**67.0±2.7**	**74.4±1.1**	**52.1±1.2**	**68.0±0.6**	**59.0±0.5**	80.4±0.8	**76.5±1.2**	**78.0±0.6**
S-only	84.3±1.2	71.9±1.4	77.6±1.0	54.5±1.3	71.8±1.0	62.0±0.6	84.3±1.6	80.4±1.1	82.1±1.3

Table 6: Coreference resolution results on the evaluation set (±: standard deviation).

6.3 Entity Linking

Tables 7 and 8 show the micro and macro average scores achieved by all models. For the micro average, the trend is clear across all types of mentions such that the CZC and our models achieve higher precision and recall, respectively. The precision gap for micro average is quite small, signaling that there is no significant loss of ability in entity resolution for singular mentions in our model. For the macro average, our model completely dominates except for the precision of plural mentions, which implies that our model is more generalizable across different entities regardless of their frequency rates in the training set. The recall of micro-average for plural mentions shows relatively high standard deviations for our model. We expect that running more trials of experiments potentially mitigates this variance, which we will explore. It is expected for the micro average scores to be higher than the macro average scores because the micro average favors frequently appearing entities such that it is possible to achieve high micro average scores without handling infrequent entities well, whereas that is not the case for the macro average.

	Singular			**Plural**			**All**		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
CZC	**72.8±0.5**	72.8±0.5	72.8±0.5	**60.8±2.4**	19.7±0.8	29.8±1.2	**71.8±0.4**	61.4±0.4	66.2±0.4
Our	72.7±0.3	**72.9±0.4**	**72.8±0.4**	59.9±1.7	**32.2±4.8**	**41.7±4.1**	71.1±0.4	**64.2±1.3**	**67.4±0.8**
S-only	73.7±0.6	73.7±0.6	73.7±0.6						

Table 7: Micro-average scores for entity linking on the evaluation set (±: standard deviation).

Table 9 shows the micro average F1 score for each entity. The top-15 frequently appearing characters are considered to be known entities, whereas all the other secondary characters are considered `OTHER`, which composes about 26.8%. Our model dominates all the main characters (the first six entities) and `OTHER`, together of which gives about 90% of the entire annotation. Given that these results are achieved by using automatically generated clusters from our coreference resolution models, they are encouraging.

	Singular			Plural			All		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
CZC	72.9±5.0	55.5±1.0	59.4±2.3	**37.9±1.0**	10.5±0.3	14.0±0.3	71.1±4.6	46.2±1.1	53.2±1.9
Our	**75.8±1.4**	**56.9±1.1**	**61.8±1.1**	34.8±5.0	**15.8±1.7**	**20.5±1.6**	**74.2±1.4**	**48.8±1.5**	**55.5±0.8**
S-only	73.3±2.5	55.4±1.6	59.6±2.3						

Table 8: Macro-average scores for entity linking on the evaluation set (±: standard deviation).

	Ro	**Ra**	**Ch**	**Mo**	**Jo**	**Ph**	**Em**	**Ri**	**Ca**	**Be**	**Pe**	**Ju**	**Ba**	**Ja**	**Ka**	**OT**	**GN**
CZC	69.2	77.5	69.0	71.3	71.5	79.0	**63.4**	76.4	**31.3**	**41.8**	**56.4**	09.3	**49.2**	11.8	24.7	58.2	**45.1**
Our	**71.9**	**78.4**	**71.5**	**72.2**	**72.3**	**79.7**	61.5	**82.0**	29.6	**41.8**	54.8	**12.8**	45.0	**18.2**	**47.3**	**59.2**	**45.1**
S-only	78.3	86.5	78.8	81.7	78.3	88.8	69.2	83.9	40.3	39.3	59.2	16.1	39.8	24.8	35.2	64.0	49.7
%	12.65	11.58	11.16	9.71	9.33	8.61	0.98	0.96	0.71	0.64	0.57	0.44	0.34	0.28	0.26	26.79	5.01

Table 9: Entity linking results on evaluation set per character.Ro: Ross, Ra: Rachel, Ch: Chandler, Mo: Monica, Jo: Joey, Ph: Phoebe, Em: Emily, Ri: Richard, Ca: Carol, Be: Ben, Pe: Peter, Ju: Judy, Ba: Barry, Ja: Jack, Ka: Kate, OT: `OTHER`; GN: `GENERAL`.

7 Conclusion

In this paper, we explore a new paradigm for handling plural mentions in two resolution tasks, coreference resolution and entity linking, on multiparty dialogue. We address this challenge by showing the inadequacy of traditional approaches in handling plural mentions, and present an innovative approach to overcome the shortcomings of existing methods for these tasks at hand. For resource creation, we expand upon the Character Identification corpus and augment it with the manual annotation of plural mentions (Section 3). For linguistic analysis, we propose a novel transition-based algorithm and evaluation metrics to process different types of mentions for coreference resolution (Section 4). For NLP engineering, we introduce a neural-based entity linking model using multi-task learning that comprehensively handles plural mentions (Section 5). The results of our models demonstrate significant improvements on these tasks, implying the feasibility of our approach to handle plural mentions (Section 6).

To the best of our knowledge, this paper provides the first extensive framework for resolving referents for plural mentions, which is a critical problem in any resolution task. Further work includes improving the quality of the dataset as well as expansion of its size, and addressing the issue of extracting global and external features for complete coreference and entity resolution for both singular and plural mentions. All resources including the annotated corpus and source codes are publicly available through the Character Identification project: `https://github.com/emorynlp/character-identification`.[5]

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for Scoring Coreference Chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, pages 563–566.

Henry Yu-Hsin Chen and Jinho D. Choi. 2016. Character Identification on Multiparty Conversation: Identifying Mentions of Characters in TV Shows. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL'16, pages 90–100.

Henry Yu-Hsin Chen, Ethan Zhou, and Jinho D. Choi. 2017. Robust Coreference Resolution and Entity Linking on Dialogues: Character Identification on TV Show Transcripts. In *Proceedings of the 21st Conference on Computational Natural Language Learning*, CoNLL'17, pages 216–225, Vancouver, Canada.

Kevin Clark and Christopher D. Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, EMNLP'16, pages 2256–2262.

[5]The Character Mining project provides a superset of the corpus presented in this paper for several other tasks: `https://github.com/emorynlp/character-mining`.

Greg Durrett, David Hall, and Dan Klein. 2013. Decentralized entity-level modeling for coreference resolution. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 114–124, Sofia, Bulgaria, August. Association for Computational Linguistics.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. Capturing semantic similarity for entity linking with convolutional neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1256–1261, San Diego, California, June. Association for Computational Linguistics.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2681–2690, Copenhagen, Denmark, September. Association for Computational Linguistics.

Prateek Jain, Manav Ratan Mital, Sumit Kumar, Amitabha Mukerjee, and Achla M. Raina. 2004. Anaphora resolution in multi-person dialogues. In Michael Strube and Candy Sidner, editors, *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue*, pages 47–50, Cambridge, Massachusetts, USA, April 30 - May 1. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark, September. Association for Computational Linguistics.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes. In *Proceedings of the Sixteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL'12, pages 1–40.

Marta Recasens and Eduard H. Hovy. 2011. BLANC: Implementing the Rand index for coreference evaluation. *Natural Language Engineering*, 17(4):485–510.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A Model-theoretic Coreference Scoring Scheme. In *Proceedings of the 6th Conference on Message Understanding*, MUC6 '95, pages 45–52, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 994–1004, San Diego, California, June. Association for Computational Linguistics.

Triad-based Neural Network for Coreference Resolution

Yuanliang Meng
Text Machine Lab for NLP
Department of Computer Science
University of Massachusetts Lowell
ymeng@cs.uml.edu

Anna Rumshisky
Text Machine Lab for NLP
Department of Computer Science
University of Massachusetts Lowell
arum@cs.uml.edu

Abstract

We propose a triad-based neural network system that generates affinity scores between entity mentions for coreference resolution. The system simultaneously accepts three mentions as input, taking mutual dependency and logical constraints of all three mentions into account, and thus makes more accurate predictions than the traditional pairwise approach. Depending on system choices, the affinity scores can be further used in clustering or mention ranking. Our experiments show that a standard hierarchical clustering using the scores produces state-of-art results with MUC and B^3 metrics on the English portion of CoNLL 2012 Shared Task. The model does not rely on many handcrafted features and is easy to train and use. The triads can also be easily extended to polyads of higher orders. To our knowledge, this is the first neural network system to model mutual dependency of more than two members at mention level.

1 Introduction

Entity coreference resolution aims to identify mentions that refer to the same entity. A mention is a piece of text, usually a noun, a pronoun, or a nominal phrase. Resolving coreference often requires understanding the full context, and sometimes also world knowledge not provided in the text. Generally speaking, three types of models have been used for coreference resolution: pairwise models, mention ranking models, and entity-mention models. The first two are more common in literature, and the third one is somewhat less studied.

Pairwise models a.k.a. mention pair models build a binary classifier over pairs of mentions (Soon et al., 2001; McCallum and Wellner, 2003). If all the pairs are classified correctly, then all coreferent mentions are identified. The mention ranking models do not rely on the full pairwise classification, but rather compare each mention to its possible antecedents in order to determine whether the mention might refer to an existing antecedent or starts a new coreference chain (Durrett and Klein, 2013; Wiseman et al., 2016; Clark and Manning, 2016). The entity-mention models try constructing representations of discourse entities, and associating different mentions with the entity representations (Luo et al., 2004).

However, none of these model types consider more than two mentions together at the low level. By low level here, we mean the processing of input mention features, as opposed to processing of constructed representations. Pairwise models and mention ranking models make low-level decisions on mention pairs only. Some further processing may be applied to reconcile global scope conflicts, but this process no longer relies directly on mention features.

This paper proposes a neural network model which works on triads of mentions directly. Each time, the system takes three mentions as input, and decisions on their coreference relations are made while taking into account all mutual dependencies. Inferences drawn from three mentions, if correctly modeled, should be more reliable than those from two mentions, simply because entities in a text tend to have multiple mutual dependencies. Firstly, coreference relation is transitive, and transitivity can be revealed only by 3 or more participants. Secondly, mutual dependencies are not just at the level of transitivity,

Proceedings of the 27th International Conference on Computational Linguistics, pages 35–43
Santa Fe, New Mexico, USA, August 20-26, 2018.

but can occur among lexical items, syntactic structures, or discourse information. Modeling dependency at these lower levels can therefore be helpful for coreference resolution. We believe it is also a closer approximation of humans' cognitive process. When we read text, we often look in two or more places (including not only mentions, but also their context) to decide what a pronoun might refers to. Therefore it is reasonable to account for it at an early stage of system design.

We show that the decisions made by the triad model are more accurate than those made by the dyad model. Such decisions can be further used in mention ranking, or simply followed up by clustering or graph partitioning as in the canonical mention pair models. The triad system can be easily extended to higher order polyads, if necessary. In this paper, we only consider triads, and dyads (pairs) are used for comparison. We use the English portion of CoNLL 2012 Shared Task dataset for training and evaluation. Our experiments show that a standard hierarchical clustering algorithm using the triad model output achieves state-of-art performance under several evaluation measures.[1]

2 Related Work

Before the neural network models became popular in coreference resolution tasks, graphical models had often been used to capture dependencies. McCallum and Wellner (2003) described a system which draws pairwise inferences but also accounts for transitivity constraints. Essentially, their model can be summarized as in equation

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_x} \exp\left(\sum_{i,j,l} \lambda_l f_l(x_i, x_j, y_{ij}) + \sum_{i,j,k,l'} \lambda_{l'} f_{l'}(y_{ij}, y_{jk}, y_{ik})\right) \quad (1)$$

The first term describes the potential function of a mention pair x_i, x_j as well as their label y_{ij}. For instance, $y_{ij} = 1$ if x_i and x_j belong to the same entity. The second term adds constraints on the labels to assure logical consistency. A particular assignment of values to x_i, x_j does not only affect the potential function involving these two nodes, but also other potential functions involving one of them. This makes the variables (mentions, in this case) dependent on each other. Exact algorithms to solve such problems are NP-hard, and some approximation techniques are often applied.

Our proposed model can be viewed as constructing potential functions over three variables x_i, x_j and x_k. However, we do not look to optimize the product of all the potential functions. Instead, we train a neural network model to assign labels to all edges within a triad locally. Note that the label y_{ij} for a given mention pair x_i, x_j may have different optimal values when different x_k's are used to construct a triad. The final assignment is determined by computing the average of y_{ij}'s. Moreover, our input features are mainly series of word embeddings and part of speech (POS) embeddings, encoding rich context information. The conventional algorithms used in graphical models cannot deal with such high dimensional features.

Graphical neural network (GNN) models have recently been used to process graphs (Duvenaud et al., 2015; Santoro et al., 2017; Kipf et al., 2018). For example, the graph convolutional networks (GCN) (Kipf and Welling, 2016; Defferrard et al., 2016) take graphs as input. Each node of the graph contains features, and a matrix represents their mutual relations. The features and the relation matrix are both used as input. Some filter layers, often shared by all nodes, process the features. The output of the filters and the relation matrix are further processed by other layers. The final output is new representations of nodes, which can be labels.

Our model shares some characteristics with GCNs. The triad input can be viewed as a basic graph: triangle, and each node is a mention. The features we used (word embeddings, POS embeddings, speaker identity, mention distance) are all associated with a node or a pair of nodes (edges). The three nodes share recurrent neural network layers. Because the output of such layers are used together, higher layers in the system have access to information from all the nodes. The output is a 3 dimensional binary vector, which can be considered a graphical representation too. However, our goal here is to find pairwise relations, and the triangle graphs are employed only to model (partial) mutual dependency among three mentions.

[1] Our source code is freely available here: `https://github.com/text-machine-lab/entity-coref`

In contrast, CGNs are capable of generating new complex representations for the nodes and they rely on the structure of the input graph, both of which are not applicable in our case.

3 System

The system consists of two major parts: the triad-based neural network model to compute mutual distances and a model to perform clustering. These two stages are not clearly divided, since defining mutual distances affects the clustering strategy.

3.1 Input Features

Our input is triads of entity mentions. The triads have mutual (joint) features and individual features as input. Speaker identity and mention distance are mutual features. The files in the CoNLL 2012 dataset are largely transcripts of broadcast news and conversations, which typically involve several speakers. We use a binary feature to indicate whether two mentions are from utterances of the same speaker (1) or not (0). Mention distance indicates how far apart two mentions are in the text. If they are next to each other, the distance is 1; if there is another mention in between, the distance is 2, and so on. We only count the mentions in between, regardless the number of the words or sentences in between.

Individual features are word tokens and POS tokens for each entity mention. The word tokens include the mentions themselves, as well as their 5 preceding tokens and 5 succeeding tokens. We also design two special tokens to mark the beginning and end of each mention. Similarly, the POS tokens include the POS tags of the mentions, as well as the POS tags of 5 preceding and 5 succeeding tokens. Two other special tags are used to mark the beginning and end of the mentions for POS tokens too.

Each word token is represented by a 300-dimensional vector. We use glove.840B.300d word vectors[2] to initialize them, and they are updated in the training process. Each POS token is represented by a one-hot vector, and updated during training too. This enables the model to learn the similarities between different POS tags (such as NNPS and NNS, for example). Table 1 gives a summary of input features.

Feature	Description
Word tokens	word embeddings of the mentions, and of 5 words before and after
POS tokens	part-of-speech tag embeddings
Speaker identity	whether two mentions are from the same speaker
Mention distance	how many other mentions are between them

Table 1: Input features

3.2 Triad Neural Network

Word embeddings are fed into a bidirectional LSTM layer, which generates a representation for each mention. The three members of the triad share the same LSTM layer. Similarly, POS embeddings are fed into a shared bidirectional LSTM layer.

$$h_i^{word} = \text{Word-LSTM}(X_i^{word}) \tag{2}$$

$$h_i^{pos} = \text{POS-LSTM}(X_i^{pos}) \tag{3}$$

where $i = 0, 1, 2$ is the index of the three mentions, X_i^{word} is the sequence of word embeddings used to represent mention i, and X_i^{pos} is the corresponding sequence of POS embeddings. Word-LSTM and POS-LSTM are both bidirectional, and shared by all input mentions. For each pair in the triad, the LSTM outputs for the two entities are concatenated with their joint features:

$$h_{ij} = f(X_{ij}^{speaker}, X_{ij}^{distance}, h_i^{word}, h_j^{word}, h_i^{pos}, h_j^{pos}) \tag{4}$$

[2]https://nlp.stanford.edu/projects/glove/

where $X_{ij}^{speaker}$ is a binary speaker identity feature for the mentions i and j, $X_{ij}^{distance}$ is the positive integer feature tracking the distance between them, and f represents several fully connected layer(s), shared by the three pairs. Our implementation uses two layers, with dropout between them.

While h_{ij} represents the relation between i and j, the other triad member needs taken into account as well. We achieve this by constructing a shared context h_{ijk}:

$$h_{ijk} = g(h_{ij} \bigoplus h_{jk} \bigoplus h_{ki}) \tag{5}$$

where g is another fully connected layer. Operator $\bigoplus$ means elementwise vector summation. Now, we can have a decoder layer d_{ij} for each of the pairwise relations.

$$d_{ij} = f_d(h_{ij}, h_{ijk}) \tag{6}$$

Function f_d is another fully connected layer. The three decoders work together to generate a 3D binary vector, as in equation 7. Each element represents whether the mention pair refers to the same entity (0 or 1).

$$\mathbf{y} = \text{sigmoid}(W(d_{ij}, d_{jk}, d_{ki}) + b) \tag{7}$$

where W and b are the weights and bias to be trained. The output $\mathbf{y}$ is a 1×3 vector. As we can see, the three decoders do not make decisions independently, but rather, "consult" with each other, as in equation 7. Each decoder also uses the shared context h_{ijk} at a lower level, as seen in equation 6.

3.3 Triads Generation

For n mentions in the text, there can be $\binom{N}{3}$ triads. In most cases, we have dozens of mentions in an article, which is not an issue. However, some long articles have hundreds of mentions, so generating all triads is unpractical and unnecessary. For instance, for 500 mentions, the total number of triads would be 20,708,500!

During the training process, we use only the mentions within the distance of 15 or less. In other words, we consider the pairs with 14 or fewer mentions between them. For testing, we consider the mentions with distances up to 40. However, this does not mean the long-distance coreference can never be detected. Often, coreferent mentions in-between the distant ones may serve as bridges, and our clustering algorithm is able to put them together. That being said, it is also true that long-distance mention pairs are less likely to corefer than those in closer proximity. Training with triads that include very distant pairs could also have the harmful effect of introducing too many negative samples.

3.4 Dyad Baseline System

To demonstrate that triads have advantages over a strictly pairwise approach, we also build a neural network model which takes mention pairs as input, and make binary decisions on the pairs only. The input features are the same as in the triad model, and the architecture can be considered a reduced triad system. Now there is no context information shared by three entities. The pair representation is directly connected to the output layer.

4 Entity Clustering

After the likelihood of pairwise coreference between all mentions has been determined, we use a clustering algorithm to group them. At the end of this process, each entity is represented by a mention cluster.

For every triad a, b and c, the system will produce three real values between [0,1] to represent the "probability" of a coreference link. We will refer to them as affinity scores. The higher the score, the more likely a pair of mentions refers to the same entity. The affinity score over a pair is computed as the average of their scores in all triads, as shown in equation 8.

$$\Phi(a, b) = \frac{1}{N} \sum_{c \in W(a,b)} \Phi(a, b; c) \tag{8}$$

Here, N is the total number of triads containing (a, b), $\Phi(a, b)$ is the affinity score of a and b, and $\Phi(a, b; c)$ is the affinity score of a and b when another mention c is in the triad. $W(a, b)$ represents the set of mentions within the distance window of a or b. We have experimented with other methods besides averaging, including taking the maximum, or the average of several top candidates. We found that the average produces better results.

The mutual distance between a and b is defined as the reciprocal of the affinity score, except we set the maximum value to be 10. Since the maximum value of $\Phi(a, b)$ is 1, the minimum value of $d(a, b)$ is also 1 according to equation 9. In principle, we would like distance metrics to have 0 as the minimum, which can be achieved by subtracting 1. However, for the purpose of clustering, it is not necessary.

$$d(a, b) = min\{\frac{1}{\Phi(a, b)}, 10\} \tag{9}$$

Recall that our system does not consider mention pairs too far apart in the text. For evaluation, the maximum distance for consideration is 40 (i.e. they may have up to 39 other mentions in between). We set the mutual distance between out-of-window mentions as 10, the maximal distance. As mentioned before, this does not mean they can never be clustered together. The result depends on the choice of linkage, and whether there is any coreferent entities in-between.

We use the hierarchical clustering function provided by SciPy library to build the sets of coreferent entities. Other than the customized distance metric, we used the default settings, opting for the $distance$, rather than $inconsistent$ cutoff criterion. When the clusters are built hierarchically, those with distances lower than a threshold are joined. Presumably, the threshold t should be lower than 2, which corresponds to affinity score higher than 0.5. The higher the threshold, the fewer clusters will be produced at the end. Preliminary experiments showed that the results are not affected very much when t is between 1.5$\sim$2.0. We used the $t = 1.7$, which corresponds to affinity score of 0.59.

The choice of linkage has a major impact on the results. We found the $single$ linkage produces the best results. Intuitively, it also makes sense. In a text, coreference among a group of mentions can only be recognized via one or several bridging mentions. In this case, the $single$ linkage can best represent the relation, while the $average$ and other linkages tend to overestimate the distances between clusters.

5 Experiments

For all the experiments, hyperparameters were tuned with the development set only. We use Adam optimizer with binary cross-entropy loss. The learning rate is initially set as 10^{-3}, then 5×10^{-4} after 100 sub-epochs, and 10^{-4} after 100 sub-epochs. We use the term "sub-epoch" to refer to training on 50 files, rather than the whole training set. The training set is relatively big, so we implemented a data generator with multiple subprocesses with a shared output queue. There are 1940 training files in total, so roughly all training files can be consumed in 40 sub-epochs, although smaller files may be used more frequently due to the nature of multiprocessing. The training completed in 300 sub-epochs. We use input dropout ratio 0.5 for word embeddings and POS embeddings. The last layer of each pair representations has dropout ratio 0.3. The bidirectional LSTMs use the average output of two directions.

For the baseline dyad model, the settings are similar. We also use 300 sub-epochs, but here it refers to training on 100 files, not 50.

5.1 Results of Triad Model

Table 2 shows the results of our triad system, compared to other systems in literature. All results are on the CoNLL 2012 English test data. As we can see, all the recent system have pretty much the same average F1 scores over the three metrics. Likely, they all capture the relatively easy cases, and the difficult ones remain to be tackled. Compared to other systems, ours has a very different distribution of scores across the three metrics, which suggests our results are quite different. Our system performs by far the best with the MUC evaluation metric, and is also the best with B^3 metric, measured with F1 score. However, the performance is quite low from the $CEAF_{\phi 4}$ metric. To understand the discrepancies, we need to analyze not only the nature of our system but also the nature of the metrics.

	MUC			B^3			$CEAF_{\phi4}$			
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Avg. F1
Wiseman et al. (2016)	77.49	69.75	73.42	66.83	56.95	61.50	62.14	53.85	57.70	64.21
Clark & Manning (2016)	79.91	69.30	74.23	71.01	56.53	62.95	63.84	54.33	58.70	65.29
Heuristic Loss	79.63	70.25	74.65	69.21	57.87	63.03	63.62	53.97	58.40	65.36
REINFORCE	80.08	69.61	74.48	70.70	56.96	63.09	63.59	54.46	58.67	65.41
Reward Rescaling	79.19	70.44	74.56	69.93	57.99	63.40	63.46	55.52	59.23	65.73
Triad	**84.93**	**77.26**	**80.92**	60.35	**71.77**	**65.65**	44.43	**59.20**	50.76	**65.78**

Table 2: Results of coreference resolution systems on the ConLL 2012 English test data. Our model (Triad) was trained on triads with the maximum distance of 15, making predictions on the triads with the maximum distance of 40. All other results are copied from Clark and Manning (2016).

MUC (Vilain et al., 1995) is a link-based metric. Mentions in the same entity/cluster are considered "linked". MUC penalizes the missing links and incorrect links, each with the same weight. It has been noted that MUC prefers over-merging entities (Luo, 2005) over under-merging. Incorrectly merging two entities is penalized less than incorrectly splitting an entity. Since we chose the $single$ linkage in our clustering algorithm, it is likely to over-merge sometimes.

B^3 (Bagga and Baldwin, 1998) is a mention-based metric. The evaluation score depends on the fraction of the correct mentions included in the response entities (i.e. entities created by the system). If a system does not make any decision and leaves every mention as singletons (i.e. no coreference at all), it will get a perfect precision score. Luo (2005) indicates that the B^3 precision score prefers no decision. On the other hand, the recall prefers over-merging entities. We have a high B^3 recall and relatively low precision, which reflects our results in MUC.

We manually spot-checked our results and the over-merging suggested by our MUC and B^3 scores appears to be true. Figure 1 provides an example to illustrate this. The top subfigure is the true clusters of entities, and the bottom shows the predictions. As we can see, the mentions are mostly grouped correctly, but there is a very large cluster (red) on the right, which basically merges two big ground truth clusters (light blue and yellow).

$CEAF_{\phi4}$ (Luo, 2005) assumes each key entity should only be mapped to **one** response entity, and vice versa. It aligns the key entities (clusters) with the response entities in the best way, and compute scores from that alignment. However, a major disadvantage of this approach is that the **unaligned responses are totally ignored**, even if they are legitimate clusters. Moosavi and Strube (2016) used an example to illustrate the problem. From a text, system $cr1$ identifies two entities $\{the\ American\ administration, it_1, it_2, it_3\}$ and $\{they_1, they_2, them, their\}$. However, it misses the fact that the two are actually the same entity. As we know, relatively few entities can be referred by both it and $they$, as $administration$ can. Another system $cr2$ only identifies the fist cluster, and misses all of the second cluster. Our intuition is $cr1$ does a better job, because it resolves much more coreference relations. However, $CEAF_{\phi4}$ will score $cr2$ higher. This is what happens to our system. There are 4217 true coreference entities from the test set. However, our system generated 5619 entities, or 33% more. Note our $CEAF_{\phi4}$ recall score is also 33% higher than $CEAF_{\phi4}$ precision score, which is related to their definitions. As a result, our system identifies a lot of small subsets of entities, but may miss some crucial links in between. MUC and B^3 will give partial credits to it, but $CEAF_{\phi4}$ completely ignores the efforts and only picks one subset for the best alignment.

In order to understand the distribution of entity/cluster sizes, we collected all the entities from test data and from our system response. Figure 2 shows the distribution of entities with respect to their size. As we can see, our system generates some very large entities, quite a few bigger than 100. This is evidence of over-merging. On the other hand, the system also generates many small entities, as shown by the left-most green bar. Note the y-axis there is in log scale, so the difference is not proportionally visualized. Figure 1 also shows that there are 13 true entities in the file, but the system finds 17. On the one hand, it merges two major entities; but on the other hand, it breaks down some small entities.

Different evaluation metrics help to diagnose different problems, and every system, as well as every

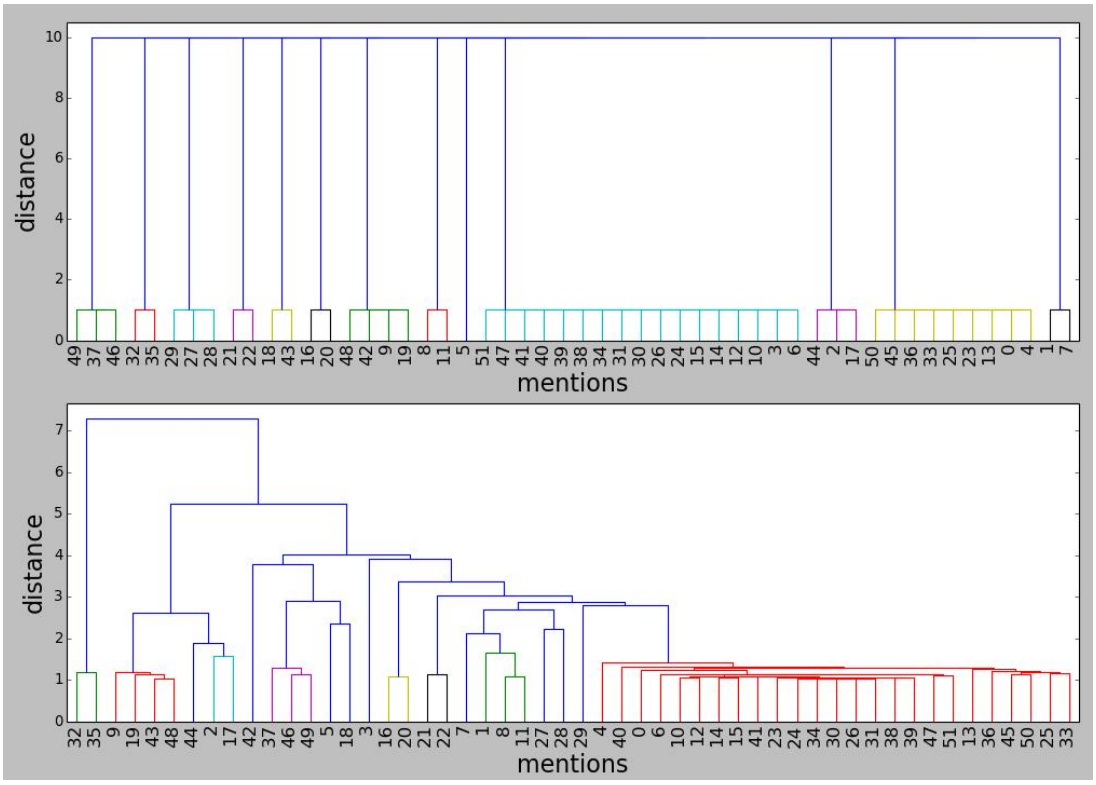


Figure 1: Clustering results from the test file voa_0049. Top subfigure is the true clusters. Bottom is the predicted results. Mentions with the same color are in the same cluster. The cutoff threshold is 1.7.

metric, would have its strength and weakness. In practice, what is needed depends on the purpose. CoNLL 2012 Shared Task uses the average of the three metrics to rank systems. Even though our system shows the best performance, the average here may not be very meaningful, given that our system has a very different distribution of scores from the others.

5.2 Results of Dyads Model

The results of the dyad system are shown in Table 3. As we can see, the triad system has a big advantage over the dyad system. Although we use maximum mention distance 15 for training, for prediction, it is beneficial to allow larger distances in the triad model. However, even in prediction, it is difficult to increase the distance beyond a certain point, since the number of triads increases very fast when the allowed distance becomes larger. Below, we show the results of both dyad and triad systems using the distance of up to 15 and up to 40 at test time.

	MUC			**B^3**			**$CEAF_{\phi4}$**			
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Avg. F1
Dyad, max distance 15	76.83	78.16	77.49	46.03	70.59	55.73	44.43	41.58	42.96	58.73
Dyad, max distance 40	77.80	**82.84**	80.24	37.65	**77.86**	50.76	**47.99**	36.49	41.46	57.48
Triad, max distance 15	84.48	75.26	79.60	**62.36**	68.30	65.20	42.78	**59.98**	49.94	64.91
Triad, max distance 40	**84.93**	77.26	**80.92**	60.35	71.77	**65.65**	44.43	59.20	**50.76**	**65.78**

Table 3: Results of the dyad model compared to the triad model. Two maximum mention distances are tested: 15 and 40. They are all trained with maximum distance 15.

Generally, we found that dyad-based system generates more entities (less coreference) than the triad system. For test data, we had 11,299 entities from the dyad-based model, but only 5,619 from the triad-based model.

Triad model can also support additional restrictions. For example, we can require at least one pair in a

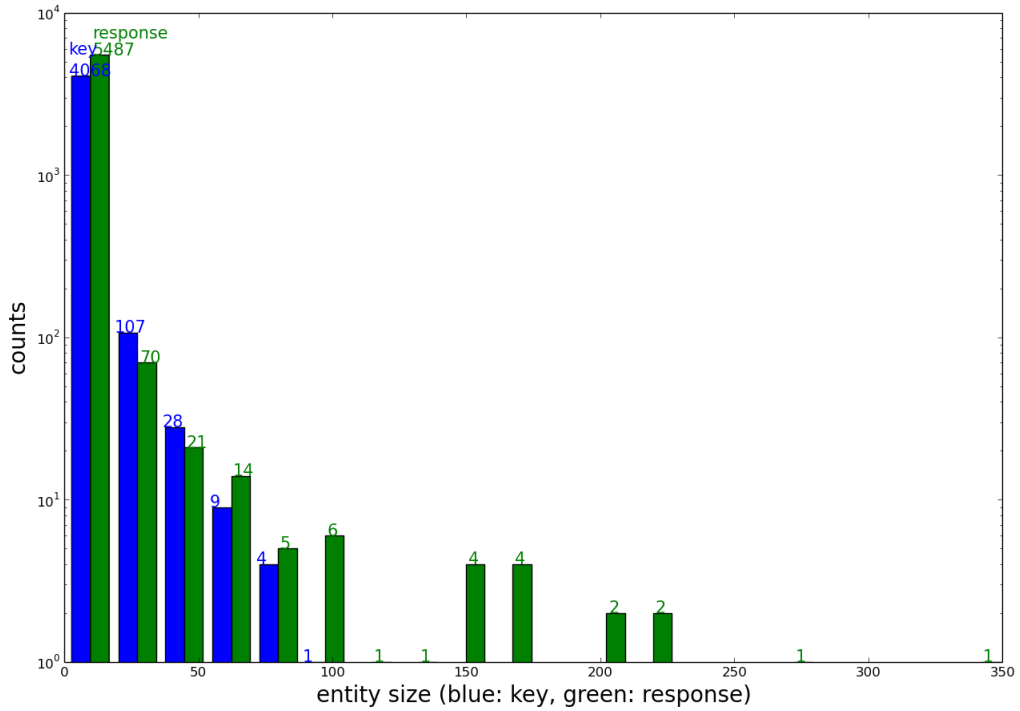


Figure 2: Entity sizes (number of mentions). Horizontal axis is the size of entities. Blue/left bars are the true counts from test set. Green/right bars are the counts from system response. Note the vertical axis is drawn in log scale. Compared to the truth, our system produces more extreme cases i.e. entities with very small number of mentions, or very big number of mentions.

triad to have a smaller distance. The point of allowing longer distances between mentions is to identify coreferent mentions that are far apart in text. However, it is typically fairly rare to have mentions that are far away refer to the same entity. We do not have to allow all sides of a triangle to be big, and imposing this restriction may improve the overall quality of the response entities.

Note that this system can be easily extended from triads to tetrads (union of four mentions) and higher polyads. Sometimes we may want to look at two more other places to determine whether a coreference relation is present. Ideally, the larger the polyad, the better we can capture mutual dependencies. However, since the number of polyads grows fast with the polyad order, the computation may quickly become intractable for larger texts.

6 Conclusion

We developed a triad-based neural network model that assigns affinity scores to mention pairs. A standard clustering algorithm using the resulting scores produces state-of-art performance on MUC and B^3 metrics. Our system appears to behave quite differently from others, judging by its performance on different metrics. A dyad-based baseline model has substantially lower performance, suggesting that using triads plays an important role. Note that approaches other than clustering, such as the mention ranking models, can easily be used with our output as well, and we expect some of them would work better than the simple agglomerative clustering.

Mutual dependencies among multiple mentions are important in coreference resolution tasks, but it is often ignored. Our triad-based model addresses this gap. This model can be additionally constrained to improve performance, and if necessary, easily extended from triads to polyads with higher order.

Acknowledgments

This project is funded in part by an NSF CAREER award to Anna Rumshisky (IIS-1652742).

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566.

Kevin Clark and Christopher D. Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. *CoRR*, abs/1609.08667.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *CoRR*, abs/1606.09375.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, October. Association for Computational Linguistics.

David K. Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. *CoRR*, abs/1509.09292.

Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907.

Thomas N. Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard S. Zemel. 2018. Neural relational inference for interacting systems. *CoRR*, abs/1802.04687.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 25–32, Stroudsburg, PA, USA. Association for Computational Linguistics.

Andrew McCallum and Ben Wellner. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, pages 79–86, Acapulco, Mexico, August.

Nafise Sadat Moosavi and Michael Strube. 2016. Which coreference evaluation metric do you trust? A proposal for a link-based entity aware metric. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy P. Lillicrap. 2017. A simple neural network module for relational reasoning. *CoRR*, abs/1706.01427.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Comput. Linguist.*, 27(4):521–544, December.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Conference on Message Understanding*, MUC6 '95, pages 45–52, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In *HLT-NAACL*, pages 994–1004. The Association for Computational Linguistics.

Unsupervised Morphology Learning with Statistical Paradigms

Hongzhi Xu[1], **Mitch Marcus**[1], **Charles Yang**[2], **Lyle Ungar**[1]
[1] Computer and Information Science, University of Pennsylvania
[2] Linguistics Department, University of Pennsylvania
Philadelphia, PA 19104, USA
[1] {xh, mitch, ungar}@cis.upenn.edu
[2] charles.yang@ling.upenn.edu

Abstract

This paper describes an unsupervised model for morphological segmentation that exploits the notion of *paradigms*, which are sets of morphological categories (e.g., suffixes) that can be applied to a homogeneous set of words (e.g., nouns or verbs). Our algorithm identifies statistically reliable paradigms from the morphological segmentation result of a probabilistic model, and chooses reliable suffixes from them. The new suffixes can be fed back iteratively to improve the accuracy of the probabilistic model. Finally, the unreliable paradigms are subjected to pruning to eliminate unreliable morphological relations between words. The paradigm-based algorithm significantly improves segmentation accuracy. Our method achieves state-of-the-art results on experiments using the Morpho-Challenge data, including English, Turkish, and Finnish. [1]

1 Introduction

Morphological learning aims to automatically uncover constitutive units of words. It is an especially important task for many NLP applications such as language generation, information retrieval etc. (Sproat, 1992). Morphology analyzing is non-trivial especially for morphologically rich languages such as Turkish where the word formation process is extremely productive and can create in principle tens of billions of word forms. The identification of morphological relations between words provides a basis for uncovering their syntactic and semantic relations, which in turn can be exploited by downstream NLP applications.

Most unsupervised models of morphological segmentation (Virpioja et al., 2013; Goldwater and Johnson, 2004; Creutz and Lagus, 2005; Creutz and Lagus, 2007; Lignos, 2010; Poon et al., 2009; Snyder and Barzilay, 2008) treat words as concatenation of morphemes. In some models, the dependencies between morphemes (e.g., the English suffix *-es* often follows a verbal stem with *y* changed to *i*, as in *carries*) are recognized (Narasimhan et al., 2015), making use of transformations akin to rewrite rules (Goldwater and Johnson, 2004; Lignos et al., 2010). In all these approaches, the dependency between morphemes is generally local, and the overall distribution of the underlying paradigms implied by the segmentation result is not explored.

In this paper, we propose to exploit the notion of the *paradigm*, a global property of morphological systems, for the task of unsupervised morphological segmentation (Parkes et al., 1998; Goldsmith, 2001; Chan, 2006). The idea of using paradigm to describe the morphological structure of a language can be traced back to a long time ago, and has been widely adopted in modern linguistic studies, starting from Ferdinand de Saussure. A paradigm refers to a set of morphological categories such as suffixes that can be can be applied to a homogeneous class of words. For instance, the paradigm (*NULL, -er, -est, -ly*) is defined over English adjectives (e.g., *high, higher, highest, highly*), the paradigm (*NULL, -ing, -ed, -s, -er*) is defined over English verbs (e.g, *walk, walking, walked, walks, walker*), etc. In essence, a paradigm establishes an equivalence class for word formation such that a word realized in one of the categories in a paradigm can be expected to appear in all the categories in the paradigm.

[1] Code is available here: https://github.com/xuhongzhi/ParaMA

Proceedings of the 27th International Conference on Computational Linguistics, pages 44–54
Santa Fe, New Mexico, USA, August 20-26, 2018.

The advantages of using paradigms in morphological learning are manifold. On the one hand, paradigms provide a principled strategy for tackling the data sparsity problem. Not all morphologically possible forms of a word will be attested (Chan, 2006) and in a morphologically rich language such as Turkish, only a small fraction will be attested even in very large corpora. Paradigms can extend the attested morphological forms from few but high frequency words to low frequency words, likely the majority, for which there is little data. On the other hand, high quality paradigms may prove effective at detecting spurious morphological relations between words that have plagued many previous models. For instance, it is not uncommon for unsupervised morphological segmentation models to produce segmentations such as *with-in*, *with-out*, and *with-er*, where *with* is an attested word and *-in*, *-out*, and especially *-er*, are highly plausible suffixes (or more generally, morphemes). From the perspective of the paradigm, a global property defined over all words that take the suffix set (*-in, -er, -out*), it is clear that such a paradigm is very poorly supported—in fact by only one stem, namely, *with*, rather than a substantial set. This suffix set, then, is very unlikely to be a true paradigm and will be discarded, thereby eliminating segmentation errors such as *with-er*.

In this paper, we show that high quality morphological paradigms can be automatically constructed, resulting in considerable improvement in unsupervised morphological segmentation accuracy. Section 2 provides a review of previous and related work. Section 3 describes the general framework of our approach. Section 4 describes how to use linguistically-motivated language-independent heuristics to generate candidate segmentations with transformation rules for each word. Section 5 describes a probabilistic model of morphological learning that provides an initial segmentation including the identification of potential suffixes. Section 6 lays out the details of constructing morphological paradigms and a pruning process that eliminates spurious morphological relations. Section 7 reports the results of our experiments on Morpho-Challenge data including English, Turkish, and Finnish in comparison with previous models. Section 8 concludes with a discussion of future research.

2 Related Work

The Morpho-Challenge, held from 2005 to 2010, led to many successful morphology learning models. The Morfessor baseline system (Creutz and Lagus, 2002; Virpioja et al., 2013) provides a framework that maximizes the likelihood of the observation under the MDL principle. Creutz and Lagus (2005; 2007) extend the model with the maximum a posteriori (MAP) on both observed data and the model. Semi-supervised models have shown to be effective on morphological segmentation (Kohonen et al., 2010; Spiegler et al., 2010). In this paper, we focus on unsupervised learning of language morphologies, based on the consideration that constructing annotating data is expensive, especially for low-resource languages.

Narasimhan et al. (2015) adopt a log-linear model with semantic similarity measures obtained from word embedding to identify morphologically related word pairs (Schone and Jurafsky, 2001) and achieve impressive segmentation results on the Morpho-Challenge data. Such semantically based model, however, requires a large corpus to train reliable word embeddings, which renders the method unsuitable for low-resource languages.

The idea of paradigms has been explored in previous studies (Parkes et al., 1998; Goldsmith, 2001; Dreyer and Eisner, 2011; Ahlberg et al., 2014). Parkes et al. (1998) propose a model that learns neat inflectional paradigms only for English verbs from a corpus. Goldsmith (2001; 2006) uses heuristic rules with the MDL principle to greedily search morphological patterns (signatures). But the performance of rule-based search methods is crucially determined by the heuristic rules, and transformation rules are difficult to incorporate. Dreyer and Eisner (2011) proposed a log-linear model to identify paradigms. However, their method requires a number of seed paradigms for training. In morphologically rich languages such as Turkish, where one paradigm can be extremely large, this method requires considerable human annotation effort. Ahlberg et al. (2014) use a semi-supervised approach to learn abstract paradigms from a given inflection table. However, the task is different from what we discuss here, which somehow discovers inflection tables as an intermediate step.

In the paper, instead of constructing paradigms as a goal, we select statistically reliable paradigms from

the initial segmentation generated from a simple probabilistic model, and then use the reliable paradigms for pruning the unreliable ones, which we refer to paradigm pruning. The advantage of the proposed model is that it mathematically maximizes the likelihood of the observed data through the probabilistic model as well as maintains the global morphological structure in terms of paradigms. As will be demonstrated later, our method produces state-of-the-art results for morphological segmentation. It also provides a promising approach to unsupervised morphological learning for low-resource languages for which there is no sufficient quantity of data to enable embedding methods.

3 Our Method

We now formally describe our model. We write $w = (r, s, t)$ for a word w that consists of a root r, a suffix s, and a transformation rule t which captures stem changes in morphological processes. A morphologically simple word is treated as taking an empty suffix *NULL* without transformation rules. For example, the word *realizing* can be analyzed as deleting the last letter *e* from the word *realize* and adding suffix *-ing*, i.e. (*realize, -ing, DEL-e*). To deal with words with multiple suffixes is trivial. If $w = (r, s, t)$ and $r = (r', s', t')$, then $w = ((r', s', t'), s, t)$. Here, we call r the *immediate root* of w. If the word has itself as immediate root, i.e. taking a *NULL* suffix, it is called *atomic*. If r' is atomic, it is called the *final* root of w, otherwise, it is called an *intermediate* root of w. For example, the word *realizing* can be represented as ((*real, -ize, NULL*), *-ing*, *DEL-e*), where (*real, -ize, NULL*) represents the word *realize*. So, the final root of *realizing* is *real*, and the word *realize* is an intermediate root. Finally, the task of morphological segmentation for a word is to recursively find immediate root until its final root is found.

3.1 Modeling Transformation Rules

We model three stem changes, called transformation rules, namely *deletion*, *substitution*, and *duplication*, similarly to (Narasimhan et al., 2015). These three transformation rules were mainly designed to capture stem changes that are involved in suffixation. All transformation rules are represented with the specific characters involved in changes. The definitions of the three transformation rules are as follows.

1. **Deletion** (DEL) of the end letter of the root. For example, the word *using* can be analyzed as (*use, -ing, DEL-e*).

2. **Substitution** (SUB) of the end letter of the root with another. For example, the word *carries* can be analyzed as (*carry, -es, SUB-y+i*)

3. **Duplication** (DUP) of the end letter of the root. For example, the word *stopped* can be analyzed as (*stop, -ed, DUP+p*).

We note, however, that certain morphological phenomena do not readily yield to the transformation-based approach here. Infixation and templatic morphology are obvious examples. Even agglutinative systems, which at first glance appear suitable for transformation rules that operate at word edges, may still prove problematic when more global morphological processes are at play. For instance, the Turkish suffixes *-lar* and *-ler* will fall under two distinct transformational rules but are in fact one morpheme that is realized differently due to vowel harmony. This problem does not pose insurmountable problems for the purpose of morphological segmentation since both *-lar* and *-ler* are relatively frequent and can be identified as genuine (and distinct) suffixes, but clearly a more robust representation of morphological processes will be necessary to account for the full range of languages. We leave this problem for future research.

3.2 Morphological Segmentation Framework

Our method is schematically described in Algorithm 1. It has several major components. The GETPRIOR function sets the prior of the model parameters by assigning each candidate segmentation (r, s, t) of a word w equal probability. The function GENSEG generates candidate segmentations, (r, s, t), for each word w. A probabilistic model is then used to compute the probability of each candidate, i.e. $P(r, s, t)$

based on the parameters estimated by the function ESTIMATE. Then the segmentation with the maximum probability is chosen. The final segmentation (e.g. words with multiple suffixes) can be constructed recursively as described in the beginning of this section.

After that, the function PARADIGMS reorganizes the segmented words into paradigms. The function RELIABLE then selects a set of statistically reliable paradigms. The function ESTIMATE estimates the model parameters based on the segmentation result derived from reliable paradigms. Then the new parameters are used by the probabilistic model to get better segmentation result. The procedure iterates for several times. Here, we let the algorithm iterate twice as we find it sufficient to produce high quality segmentations. The function PRUNE prunes the unreliable paradigms. The final result is generated based on the reliable paradigms and the pruned ones with function SEGMENTATION. The following sections describe each component in details.

Algorithm 1 The main procedure

```
1: procedure MAIN(WordList D)
2:     {P(r, s, t)} ← GETPRIOR(D)
3:     while iter < maxIter do
4:         morph ← {}
5:         for all w in D do
6:             segs ← GENSEG(w)
7:             seg ← arg max_{(r,s,t)∈segs} P(r, s, t)
8:             morph ← morph + (w, seg)
9:         pdgs ← PARADIGMS(morph)
10:        pdgs_reliable, pdgs_unreliable ← RELIABLE(pdgs)
11:        {P(r, s, t)} ← ESTIMATE(pdgs_reliable)
12:    pdgs_pruned ← PRUNE(pdgs_unreliable)
13:    return SEGMENTATION(pdgs_pruned + pdgs_reliable)
```

4 Generating Candidate Segmentations

4.1 Selecting Candidate Suffixes

To obtain a working set of suffixes, we first adopt a simple method from previous studies: given a word pair (w_1, w_2), if $w_2 = w_1 + s$, then s is a candidate suffix (Keshava and Pitler, 2006; Dasgupta and Ng, 2007). By comparing all possible word pairs, we can generate a set of candidate suffixes with their counted frequencies. The more frequent a candidate is, the more likely it is to be a real suffix. In our system, we only keep candidate suffixes that are at most six character long and appear at least three times in the word list. If applied naively, this method produces many short, spurious suffixes that are frequently occurring substrings in words, e.g. (*for, fore*), (*are, area*), (*not, note*), (*she, shed*) etc. The problem can be overcome by imposing a minimum length on words that are subject to candidate suffix generation. In practice, we find that a minimum word length of four characters works well, which partially reflects the prosodic constraints on minimal words from the linguistic literature (McCarthy and Prince, 1999).

In addition, if a candidate suffix can be taken by words of various lengths, it is more likely to be a real one; if a candidate suffix can only be taken by short words, it is likely to be a false one. To further utilize this information, we use the following equation to calculate the confidence value (conf) of a candidate suffix.

$$conf(s) = \log(1 + |W_s|) \times \frac{1}{|W_s|} \times \sum_{w \in W_s} len(w) \quad (1)$$

where W_s is the set of words that can take the candidate suffix s and form a new word, and $len(w)$ is the length of word w. Finally, we can select the top N candidates.

4.2 Generating Candidate Segmentations with Transformation Rules

The procedure for generating a candidate segmentation of target word w begins by stripping a possible suffix s from the word. If the remaining part r is a valid word, then (r, s, *NULL*) is a possible candidate. If r is not a word, but there exists a word r'=r+c and $c \neq s$, then (r', s, *DEL-c*) is a candidate. If r is in

the form r'+c, and r' is a valid word also ending with c, then (r', s, *DUP-c*) is a candidate. If r is in the form r'+c, r' is not a word, but there exist another word w'=r'+c' and $c' \neq c$, then (w', s, *SUB-c+c'*) is a candidate. If no possible suffix s could be found, then add (w, *NULL*, *NULL*) as a candidate.

We apply the transformation rule types in the following ordering: (*NULL, duplication*, *deletion*, *substitution*). A candidate with a transformation rule is generated only if no candidate could be found with a previous type. The ordering reflects the linguistic approaches to morphology where suffixation applies to the stem changes, and the changes take place at morpheme boundaries before affecting the rest of the words on either side (Halle and Marantz, 1993). The linguistic reality of these processes, once more widespread in English, is now only latently reflected in modern English orthography but can be transparently observed in other languages.

5 A Probabilistic Model for Morphological Segmentation

We evaluate the conditional probability of a segmentation (r, s, t) given a word w. Since each triple (r, s, t) is uniquely associated with a single word w, denoted as $(r, s, t) = w$, for all (r, s, t), $P(r, s, t|w) = 0$ if $(r, s, t) \neq w$. Otherwise, we use the following formula to calculate this probability.

$$P(r, s, t|w) = \frac{P(r, s, t)}{\sum_{(r', s', t')=w} P(r', s', t')} \tag{2}$$

To compute $P(r, s, t)$, we assume that r is independent of s, and that t depends on both r and s. Taking into account that the transformation rules are not word form specific, but rather follow some constraints based on the phonological structures of the word and the suffix. We assume that the transformation rule t is dependent on feature extracted from r and s, denoted by $f(r, s)$. Thus, $P(r, s, t)$ can be decomposed as follows.

$$P(r, s, t) = P(r) \times P(s) \times P(t|f(r, s)) \tag{3}$$

In our implementation, we assume that t depends on the last character of the root r ($end(r)$) and the initial character of the suffix s ($init(s)$), i.e. $f(r, s) = end(r)$-$init(s)$. For example, the probability of the segmentation (*carry, -es, SUB-y-i*) for the word *carries* can be calculated by multiplying P(*carry*), P(*-es*), and P(*SUB-y+i|f*), where *f* is *y-e*. Again, this is an approximation of the morpho-phonological properties of language but one which nevertheless proves effective for morphological segmentation.

Finally, the segmentation of a word w can be predicted by choosing the segmentation (r, s, t) that maximizes $P(r, s, t)$ as follows.

$$seg(w) = \underset{(r,s,t)=w}{\operatorname{argmax}} P(r, s, t) \tag{4}$$

5.1 Parameter Estimation

To estimate the parameters $P(r)$, $P(s)$, $P(t|f)$, we initially assume that each candidate segmentation of a word has equal probability because the unsupervised model has no access to gold data. Each (r, s, t) of all possible segmentations seg of a word w then obtains $1/|seg|$ weight. After that, the probabilities $P(r)$, $P(s)$, and $P(t|f)$ can be easily computed based on the frequencies of r, s, f, and (t, f). This first estimation of the parameters $P(r)$, $P(s)$, $P(t|f)$ is the prior returned by the function GETPRIOR in Algorithm 1.

Consequently, the probability of a segmentation (r, s, t) of a word w can be computed using Formula 3, and select the segmentation with the maximum probability. Here, EM can also be used for estimating parameters, but we found that this simple method works very well. After the first round, the parameters can be re-estimated by only using the predicted segmentation of each word, with the function ESTIMATE in Algorithm 1.

As discussed above, the reliability of a candidate suffix is also related to the length of words that can take the suffix. So, another way of estimating $P(s)$ is to use the confidence value calculated with Equation 1. We will show that the method gives better results.

English		Turkish		Finnish	
Suffix Set	**Sup**	**Suffix Set**	**Sup**	**Suffix Set**	**Sup**
(-ed, -ing, -s)	772	(-ki, -n)	1560	(-ssa, -ta)	2465
(-ed, -ing)	331	(-nı, na)	207	(-en, -ta)	1132
(-ed, -er, -ing, -s)	219	(-nı, -na, -nda)	201	(-la, -le, -ta)	808
(-ly, -ness)	208	(-ne, -ni)	199	(-n, -ssa)	693
(-ed, -ing, -ion, -s)	154	(-ı, -a)	165	(-ssä, -tä)	677
(-ic, -s)	125	(-de, -e, -i, -in)	126	(-sen, -set, -sia, -ta)	462
(-ly, -s)	109	(-nde, -ne, -ni, -nin)	82	(-en, -ssa, -ta)	328
(-ed, -ing, -ment, -s)	63	(-dir, -ki, -n)	81	(-sen, -set, -siä, -tä)	177
(-ism, -s)	52	(-ğı, -tir)	81	(-a, -ksi, -la, -le, -ssa, -ta)	160
(-ed, -es, -ing)	52	(-de, -i)	79	(-aan, -ni)	156

Table 1: Examples of paradigm suffix sets and their supports of English, Turkish, and Finnish.

6 Paradigm Construction and Pruning

In this section, we discuss how to perform a post-pruning with a paradigm-based algorithm to exclude noisy segmentations. We define a paradigm formally as follows.

- **Paradigm** $p = S \times R$ is a Cartesian product of a set of suffix $S = \{s_i\}$ and a set of roots $R = \{r_i\}$, such that for any suffix $s \in S$ and root $r \in R$, r can take s to form a valid word w by applying a transformation rule t [2].

With this definition, the larger the cardinality $|R|$ is, the more reliable a paradigm p is. On the other hand, it is not always true that the larger $|S|$ is, the more reliable a paradigm is. An extreme case occurs when there is only one root in the paradigm, i.e. $|R| = 1$. For example, the root *the* forms a paradigm with 42 possible suffixes and all of them are false. We therefore define the support of a paradigm as follows.

- **Support** (SUP) of the a paradigm $p = S \times R$ is $|R|$, the cardinality of the root set R.

6.1 Constructing Paradigms

After we get the segmentation (r, s, t) for each word w, the paradigms can be easily obtained by grouping together the words that share the same immediate root r, regardless of the transformation rules that are involved. For example, the words *reporting*, *reported*, and *reports* are segmented as (*report, -ed, NULL*), (*report, -ing, NULL*), and (*report, s, NULL*) respectively, and the words *baked*, *baking*, *bakes* are segmented as (*bake, -ed, DEL-e*), (*bake, -ing, DEL-e*), and (*bake, -s, NULL*). Then we can construct a paradigm based on these two words as {*-ed, -ing, -s*} × {*report, bake*}.

6.2 Paradigm Pruning

We crucially assume that even though the segmentation result given by the initial model is not highly accurate, the distribution of paradigms constructed will provide clear evidence of whether they are reliable. Table 1 shows some examples of paradigms with more than one suffix, and these are indeed consistent with the morphological structures of the languages.

The paradigms with more than one suffix and with support value larger than 1 are selected as the reliable ones, the same strategy used by Goldsmith (2001) for filtering when constructing candidate paradigms. The method has two consequences. First, we exclude a large proportion of suffixes that only appear in unreliable paradigms so that the frequency of suffixes be estimated based on the reliable paradigms. Second, we can use the reliable paradigms as references for pruning the unreliable ones.

The basic idea of pruning unreliable paradigms with reliable ones is to take the intersection of the suffix set of a paradigm to be pruned and that of any of the reliable ones and to choose the one that achieves the best score. The score of a set of suffixes is calculated according to the following equation.

[2]Transformation rules (or stem changes) are not considered a part of the paradigms, because they are usually not directly driven by morphological processes but rather some phonological rules or others

	English	Turkish	Finnish
Training (MC:10)	878,036	617,298	2,928,030
Test (MC:05-10)	2,218	2,534	2,495

Table 2: Data Set. MC:10 is the Morpho-Challenge 2010 and MC:05-10 is the combined data of Morpho-Challenge 2005-2010.

$$score(S) = \sum_{s \in S} conf(s) \tag{5}$$

For instance, suppose we have an unreliable paradigm {*-ed, -ing, -s, -se*} × {*appear*} [3], which only has support value 1, and there are two reliable paradigms with suffix sets (*-ed, -ing, -s*) and (*-ed, -ing*) respectively. Then, the pruning algorithm calculates the intersection of the unreliable one and each of the reliable ones, resulting in (*-ed, -ing, -s*) and (*-ed, -ing*). According to Equation 5, the former one will be kept since it has a higher score. Thus, the original paradigm is pruned to {*-ed, -ing, -s*} × {*appear*}, and consequently, the false morphological relation between *appease* and *appear* (by *-se*) is filtered out.

7 Experiments

7.1 Experiment Setting

7.1.1 Data

We ran experiments on a combined version of the Morpho-Challenge 2005-2010 data sets including English, Turkish, and Finnish, the same setup as Narasimhan et al. (2015). In our experiments, testing words are included in the training set since the method is unsupervised. A statistical description of the data is shown in Table 2.

As in previous work on unsupervised morphological learning, we use a frequency-based filtering method to reduce noise in the data. This is necessary because the word list provided by Morpho-Challenge 2010 is generated by lower-casing all running tokens in the corpora including abbreviations and proper nouns. Many of these are three characters words. We use the following conditions to select reliable roots: 1) $freq \geq 2000$ if $len(word) \leq 3$; 2) $freq \geq 200$ if $len(word) \leq 4$; 3) $freq \geq 20$ if $len(word) \leq 5$; 4) $freq \geq 3$ else. The motivation is that short words are expected to be more frequent; rare short words are likely to be abbreviations or noise.

7.1.2 Compounding

We also add a compound inference module, which simply splits a word w before generating candidate segmentations if it is composed of w_1 and w_2. If there is more than one possible segmentation, then we choose the one with maximum length of w_1. The candidate segmentations are then generated for w_1 and w_2 separately. The final segmentation result is obtained by combining the segmentation results of w_1 and w_2.

7.1.3 Evaluation

Following (Narasimhan et al., 2015), we measure the performance of our model with segmentation points, i.e. the boundaries between morphemes in words. The precision, recall and F1 values on the identification of segmentation points are reported.

7.2 Experiment Results

7.2.1 Ablation test

We first test five different variations of our model. The first one is the baseline system (Base) which only uses the probabilistic Model without transformation rules. The second one (+Trans) is the baseline plus transformation rules. The third one (+Comp) is the second one plus the compounding inference module. The fourth one (+Prune) is the third one plus the paradigm pruning algorithm. The fifth one (+Conf) is

[3]The word *appear* takes suffix *-se* to form *appease* through a transformation rule *DEL-r*.

	English			Turkish			Finnish		
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**
Base	0.483	0.686	0.567	0.616	0.621	0.619	0.521	0.245	0.333
+Trans	0.531	0.807	0.641	0.589	0.728	0.651	0.393	0.338	0.363
+Comp	0.511	0.861	0.641	0.582	0.728	0.647	0.389	0.606	0.474
+Prune	0.814	0.783	**0.798**	0.651	0.514	0.574	0.688	0.436	0.534
+Conf	0.810	0.787	**0.798**	0.600	0.746	**0.665**	0.712	0.481	0.574

Table 3: Experimental result of our model.

	English			Turkish			Finnish		
Method	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**
Morf-Base	0.740	0.623	0.677	0.827	0.362	0.504	0.839	0.357	0.501
Morf-Cat	0.673	0.587	0.627	0.522	0.607	0.561	0.782	0.452	0.573
LogLinear-C	0.555	0.792	0.653	0.516	0.652	0.576	0.483	0.650	0.554
LogLinear-Full	0.807	0.722	0.762	0.743	0.520	0.612	0.428	0.496	0.460
Our model	0.810	0.787	**0.798**	0.600	0.746	**0.665**	0.824	0.452	**0.584**

Table 4: Comparison of our model with others. The numbers for Finnish are obtained by running the systems by ourselves. The other numbers are from (Narasimhan et al., 2015).

the same as the fourth one except that the estimation of $P(s)$ is based on the confidence value calculated through Equation 1. In order to achieve the best performance, if a feature harms the performance, it will be removed in the next round.

The results are shown in Table 3. Firstly, we can see that incorporating of transformation rules improves the performance for all the three languages with 7.6%, 3.2%, and 3.0% improvements of F1 measure respectively. After adding the compounding analysis module, the performance is significantly improved on Finnish, with 11.1% improvement of F1 measure. The paradigm pruning algorithm significantly improves the performance on English and Finnish, with 15.7% and 6.0% of F1 measure, and improves precision of the model on Turkish, although the overall performance decreases. Finally, by using confidence based estimation of $P(s)$, the performance is improved further on all the three languages, achieving the best result for English and Turkish. For Finnish, the best result is actually achieved without transformation rules, namely 0.824, 0.452, and 0.584 in Precision, Recall, and F1 respectively.

7.2.2 Comparison with other models

We compare our model with three systems including the Morfessor Baseline system (Morf-Base) (Virpioja et al., 2013), Morfessor CatMAP (Morf-Cat), and the Log-linear model with full features (LogLinear-Full) and the model without semantic similarity (LogLinear-C) in (Narasimhan et al., 2015). Besides English and Turkish as used in (Narasimhan et al., 2015), we also add Finnish for experiments. For training word embeddings which will be in LogLinear-Full model, we use a corpus created in the DARPA LORELEI [4] project, which contains about 101 million tokens.

The result is shown in Table 4. The numbers for English and Turkish are from (Narasimhan et al., 2015). We can see that our model achieves the best performance in all the three languages. The LogLinear-Full model is the second best model on English and Turkish. However, it is worth noting that that model is based on semantic similarity features which requires training word vectors on independent corpora. Our model, on the other hand, only uses a list of words. The word frequencies are only used to filter noise. Our model is significantly better than the (Narasimhan et al., 2015) model without semantic embedding (i.e. LogLinear-C), with 20.1% relative improvement of F1 on English and 12.8% relative improvement on Turkish. For Finnish, the Morfessor CatMAP model has similar result as ours. The semantic similarity information harms the LogLinear-Full model. We think that this is due to the data sparseness problem as Finnish is a highly synthetic language, which then requires a larger corpus for training effective word embeddings.

[4] https://www.darpa.mil/program/low-resource-languages-for-emergent-incidents

7.3 Error Analysis and Discussion

Morphology learning systems in general suffer from two major problems, namely over-segmentation and under-segmentation. Over-segmentation is usually caused by spurious roots, either intermediate or final, such as the over-segmented words caused by the short frequent words, e.g. *the, with*, etc. Under-segmentation is usually caused by unseen intermediate roots or unidentified real suffixes. In morphologically rich languages like Turkish and Finnish where a root can take multiple suffixes, this problem is more serious.

Firstly, we can see from Table 3 that the use of transformation rules significantly reduces the under-segmentation problem in all the three languages as indicated by the increased recall rates. For English, the transformation rules also reduce the over-segmentation problem. This is due to fact that the transformation rules can well capture the morphology of English and thus significantly increase the true positive segmentations. On the other hand, the transformation rules increase the over-segmentation problem for Turkish and Finnish as indicated by the decreased precisions. For Finnish, the best performance of the system is achieved without transformation rules. That is due to the fact that Finnish morphology involves a large number of vowel changes, e.g. lengthening and shortening at non-boundary positions, which cannot be captured by the current transformation rules. However, the transformation rules we use introduce a large number of false stem changes which then causes the over-segmentation problem increased. We will address this problem in our future research.

Secondly, the compounding module can further improve the recall rates and thus decrease the under-segmentation problem for English and Finnish. This also reflects the compounding nature of the languages and also the distribution of the test set.

Finally, the pruning algorithm significantly reduces over-segmentations for all three languages as indicated by the increased precision values. However, the under-segmentation problem is also increased. This is due to the identification of incomplete paradigms which is then caused by data sparseness problem. For Turkish, the problem is even more serious. Pruning with incomplete paradigms will falsely exclude real suffixes from an unreliable paradigm. This problem can be potentially addressed by merging proper paradigms to identify the maximal suffix sets (complete paradigms). This will be in our future research.

8 Conclusion and Future Work

In this paper, we propose an unsupervised model of morphology learning which outperforms the state-of-the-art systems, using only orthographic information from a word list. Our contribution also lies in providing a new method of using automatically learned paradigms to fine tune the morphological segmentation results produced by a simple probabilistic model. This method is effective in eliminating spurious segmentations and improving the segmentation accuracy. Finally, we also use the word length information to select good candidate suffixes and estimate the suffix probabilities, which can further improve the performance of the model. In addition, combining our model and semantics based systems can potentially yield better result since they use different kinds of information and complement each other.

We believe that our approach of using paradigms provides a foundation for dealing with other morphological types such as prefixes, infixes, reduplication etc. In detail, the notation of the suffix variable s can be generalized to f, a morphological function that takes a root as input and produces the derived form. Correspondingly, the definition of paradigms could be easily revised as a set of morphological functions that can take a set of words and generate their derived forms. We will work on extending our system to process other types of morphologies in our future research.

Acknowledgements

We thank the rest of the University of Pennsylvanias LORELEI research team for the helpful discussions. We also thank the anonymous reviewers who have given valuable and constructive comments as well as insightful suggestions for improving our system. This research was funded by the DARPA LORELEI program under Agreement No. HR0011-15-2-0023.

References

Malin Ahlberg, Mans Hulden, and Markus Forsberg. 2014. Semi-supervised learning of morphological paradigms and lexicons. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 569–578.

Erwin Chan. 2006. Learning probabilistic paradigms for morphology in a latent class model. In *Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology and Morphology*, pages 69–78. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6*, pages 21–30. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2005. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Helsinki University of Technology.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(3):1–34.

Sajib Dasgupta and Vincent Ng. 2007. High-performance, language-independent morphological segmentation. In *HLT-NAACL*, pages 155–163.

Markus Dreyer and Jason Eisner. 2011. Discovering morphological paradigms from plain text using a dirichlet process mixture model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 616–627. Association for Computational Linguistics.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational linguistics*, 27(2):153–198.

John Goldsmith. 2006. An algorithm for the unsupervised learning of morphology. *Natural Language Engineering*, 12(4):353–371.

Sharon Goldwater and Mark Johnson. 2004. Priors in bayesian learning of phonological rules. In *Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology: Current Themes in Computational Phonology and Morphology*, pages 35–42. Association for Computational Linguistics.

Morris Halle and Alec Marantz. 1993. Distributed morphology and the pieces of inflection. In Kenneth Hale and Samuel Jay Keyser, editors, *The view from Building 20: Essays in linguistics in honor of Sylvain Bromberger*, pages 111–176. MIT Press, Cambridge, MA.

Samarth Keshava and Emily Pitler. 2006. A simpler, intuitive approach to morpheme induction. In *Proceedings of 2nd Pascal Challenges Workshop*, pages 31–35.

Oskar Kohonen, Sami Virpioja, and Krista Lagus. 2010. Semi-supervised learning of concatenative morphology. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 78–86. Association for Computational Linguistics.

Constantine Lignos, Erwin Chan, Mitchell P. Marcus, and Charles Yang. 2010. A rule-based acquisition model adapted for morphological analysis. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments*, pages 658–665. Springer.

Constantine Lignos. 2010. Learning from unseen data. In *Proceedings of the Morpho Challenge 2010 Workshop*, pages 35–38.

John J McCarthy and Alan Prince. 1999. Faithfulness and identity in prosodic morphology. *The prosody-morphology interface*, 79:218–309.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2015. An unsupervised method for uncovering morphological chains. *Transactions of the Association for Computational Linguistics*, 3:157–167.

Cornelia Parkes, Alexander M. Malek, and Mitchell P. Marcus. 1998. Towards unsupervised extraction of verb paradigms from large corpora. In *Proceedings of the Sixth Workshop on Very Large Corpora (COLING-ACL)*.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 209–217. Association for Computational Linguistics.

Patrick Schone and Daniel Jurafsky. 2001. Knowledge-free induction of inflectional morphologies. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–9.

Benjamin Snyder and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of the 46th Annual Meeting on Association for Computational Linguistics*, pages 737–745.

Sebastian Spiegler, Bruno Golénia, and Peter A. Flach. 2010. Word decomposition with the promodes algorithm family bootstrapped on a small labelled dataset. In *Proceedings of the Morpho Challenge 2010 Workshop*, pages 49–52.

Richard William Sproat. 1992. *Morphology and computation*. MIT press.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, Mikko Kurimo, et al. 2013. Morfessor 2.0: Python implementation and extensions for morfessor baseline.

Challenges of language technologies for the indigenous languages of the Americas

Manuel Mager
Instituto de Investigaciones en Matemáticas
Aplicadas y en Sistemas
Universidad Nacional Autónoma de México
mmager@turing.iimas.unam.mx

Ximena Gutierrez-Vasques
GIL IINGEN
Universidad Nacional
Autónoma de México
xim@unam.mx

Gerardo Sierra
GIL IINGEN
Universidad Nacional
Autónoma de México
gsierram@iingen.unam.mx

Ivan Meza
Instituto de Investigaciones en Matemáticas
Aplicadas y en Sistemas
Universidad Nacional Autónoma de México
ivanvladimir@turing.iimas.unam.mx

Abstract

Indigenous languages of the American continent are highly diverse. However, they have received little attention from the technological perspective. In this paper, we review the research, the digital resources and the available NLP systems that focus on these languages. We present the main challenges and research questions that arise when distant languages and low-resource scenarios are faced. We would like to encourage NLP research in linguistically rich and diverse areas like the Americas.

Title and Abstract in Nahuatl

Masehualtlahtoltecnologias ipan Americatlalli

In nepapan Americatlalli imacehualtlahtol, inin tlahtolli ahmo quinpiah miac tlahtoltecnologías ("tecnologías del lenguaje"). Ipan inin amatl, tictemoah nochin macehualtlahtoltin intequiuh, nochin recursos digitales ihuan nochin tlahtoltecnologías in ye mochiuhqueh. Cequintin problemas monextiah ihcuac tlahtolli quinpiah tepitzin recursos kenin amoxtli, niman, ohuic quinchihuaz tecnología ihuan ohuic quinchihuaz macehualtlahtolmatiliztli. Cenca importante in ocachi ticchihuilizqueh tlahtoltecnologías macehualtlahtolli, niman tipalehuilizqueh ahmo mopolozqueh inin tlahtolli.

1 Introduction

The American continent is linguistically diverse, it comprises many indigenous languages that are nowadays spoken from North to South America. There is a wide range of linguistic families and they exhibit linguistic phenomena that are different from the most common languages usually studied in Natural Language Processing (NLP). There are approximately 28 million[1] people who self identify as members of an indigenous group (Wagner, 2016) and they speak around 900[2] native or indigenous languages. This represents an important cultural and linguistic richness. This richness was captured by the following quote from McQuown (1955) "in one small portion of the area, in Mexico just north of the Isthmus of Tehuantepec, one finds a diversity of linguistic type hard to match on an entire continent in the Old World". In spite of this, few language technologies have been developed for these languages, moreover, many of the indigenous languages spoken in the Americas face a risk of language extinction.

The aim of this work is to explore the research in the NLP field for the indigenous languages spoken in the American continent and to encourage research for these languages. We stress the need of developing language resources and NLP tools for these languages and we point out some of the challenges that arise

[1] Each country has its own methodology and criteria to estimate the amount of speakers. This is the sum of all estimations.

[2] This number varies depending on the classification criteria used on different studies.

Proceedings of the 27th International Conference on Computational Linguistics, pages 55–69
Santa Fe, New Mexico, USA, August 20-26, 2018.

when working on this field. Since indigenous languages are digitally scarce, developing technologies can have a positive social impact for the communities which depend on these languages. The great diversity of these languages posses interesting scientific challenges, e.g., adapting well established approaches, creation of new methods, collecting new data. Tackling these challenges contributes to building more general computational models of language, and to get a deeper insight into human language understanding. Moreover, many statistical NLP methods seek to achieve language independence, however they often lack of linguistic knowledge or they do not cover the broad diversity of languages (Bender, 2011). In this sense, it is important to acknowledge the characteristics of the indigenous languages of the Americas as a way of complementing the current NLP methods.

Contributions. To sum up, we made the following contributions: (i) we gave a brief introduction into the diverse language space of the indigenous languages of the American continent; (ii) we provide an overview of the existing digital corpora and language sytems that have been developed for some of the languages spoken in the Americas; (iii) and we discusses the advances, used methodologies, challenges and open questions for the most researched NLP tasks in these languages.

In order to maintain the information of this paper updated of the computer readable resources, developed systems and scientific research we made a public available list[3] with the latest advances for the Indigenous Languages of the Americas.

1.1 Languages overview

Linguistic typology is a field that studies the different languages of the world and tries to establish the relationships among them. This is not an easy task since there is still limited knowledge about many languages, specially in highly diverse regions. However, according to several linguistic atlas[4], there are around 140 linguistic families in the world, from these linguistic families almost 40% are native to the Americas. Nowadays, approximately 900 different indigenous languages are spoken of this region, making this continent a linguistically diverse territory.

Americas native languages can exhibit very different linguistic phenomena, these typological features are important to be taken into account when developing language technologies (O'Horan et al., 2016). It would be hard to provide a general description of all the languages spoken in the Americas, however, we would like to highlight some of the linguistic features that usually represent a challenge when doing NLP.

Many languages in North America tend to have a high degree of morphological synthesis, i.e., many morphemes per word (Mithun, 2017). For instance, languages that belong to the Eskimo-Aleut family (native to Canada, Alaska, Greenland and Siberia) are highly polysynthetic suffixing languages. Languages from other linguistic families spoken in North America also show specific degrees of agglutination, polysynthesis, and they have morphemes that express a wide range of functions and nuances of space or direction (Mithun, 2001). Languages with this type of phenomena usually have compact word constructions that are equivalent to whole sentences in other languages like English.

Another linguistic phenomena that is found in many languages spoken in the Americas is the tone, i.e., languages where the pitch is important to distinguish one word from another (the tone can express lexical meaning and grammatical function). Some linguistics families like Oto-manguean (spoken in Mexico) have languages with many types of tones. The orthography of these languages need to mark the wide range of tones, however, many indigenous languages face a lack of orthographic normalization. This can be specially problematic when trying to do NLP and processing text documents.

In general, the native languages of the Americas have a limited digital text production, in some cases they may have a strong oral tradition but not a written one. Due to social and political reasons, alphabetization and education programs are not always available for native speakers. Details about the languages families for which we were able to identify digital and technological resources, are given in Appendix A

[3]Updated list of resources for Indigenous languages of the Americas: `https://github.com/pywirrarika/naki`

[4]Based on Ethnologue (Simons and Fennig, 2017), Glottolog(Nordhoff et al., 2013), WALS (Dryer and Haspelmath, 2013).

2 Corpus and digital resources

Most of the current state of the art methods in NLP are data-driven approaches that require vast amounts of corpora in order to achieve good performance. Widely popular machine learning methods and vector space representations, e.g., neural networks and word embeddings, often rely in big monolingual corpora.

Annotated and unannotated corpora are required for several NLP tasks. For instance, parallel corpora are essential for building statistical machine translation (MT) systems, while morphological annotated data is essential for POS (Part-of-Speech) taggers and morphological analyzers, just to name a few.

In the case of MT, the most common sources for gathering large amounts of parallel data include specific domain texts such as parliamentary proceedings, religious texts, and software manuals that are translated into several languages. Additionally, the World Wide Web represents a good and typical source for finding large-size and balanced parallel and monolingual text (Resnik and Smith, 2003). However, many of the world languages do not have readily available digital corpora. Indigenous languages of the Americas do not have a web presence or text production comparable to richer resourced languages and it is difficult to find websites that offer their content the native languages.

We explored the resources that are digitally available for some of the native languages spoken in the Americas. Regarding parallel corpora, the bible is a common source that contains translations to many of these languages, although it is not always straightforward to extract the content in a digital format. On the other hand, there are some projects that offer parallel content through a web search interface, e.g., Axolotl (Spanish-Nahuatl parallel corpus) that was mainly gathered from non-digital sources (books from several domains), the documents have dialectal, diachronic and orthographic variation (Gutierrez-Vasques et al., 2016). Nahuatl is a Uto-Aztecan language spoken in Mexico (approx. 1.5M speakers) that lack of an orthographic normalization. In fact, this is the case for many languages spoken the Americas: large dialectal variation, and missing standardization.

Also for the same language (Nahuatl), a comprehensive digital dictionary has been collected with information from five previous dictionaries (Thouvenot, 2005), these dictionaries date from 16th to 21st Century (de Olmos et al., 2002; Walters et al., 2002). The query interface of this resource is available online[5].

Another language that belongs to the Uto-Aztecan family is Wixarika or Huichol (approx. 45K speakers). For this language, there is a parallel corpus (Wixarika-Spanish) that gathers translations of classic Hans Christian Andersen's literature (Mager et al., 2018). In this case, the translations belong to one specific dialect (Nayarit). This resource can be fully downloaded from the web [6].

The Shipibo-konibo language (approx. $26,000$ speakers) belongs to the Panoan language family, it is spoken in the Amazon region of Peru and Brazil. Several types of digital resources are available for this language. A parallel corpus between Spanish and this language was constructed using educational and religious documents (Galarreta et al., 2017). Moreover, Shipibo-konibo language has a POS tagged corpus, a set of words and its lemmas and an online-dictionary that has been recently released by Pereira-Noriega et al. (2017).

Also spoken in South America, the Guarani language (approx. 8M speakers) belongs to the Tupi-Guarani family. Abdelali et al. (2006) developed software for collecting Guarani resources from speakers, from this gathering they were able to construct a parallel corpus (Spanish-English-Guarani) and a monolingual corpus. There is also a digital Guarani dictionary (Ramírez and Wolf, 1996) available online[7].

Quechua is one of the most spoken language families on the continent (approx. 9M speakers) but there is scarceness of corpora and language technologies. Espichán-Linares and Oncevay-Marcos (2017) released monolingual corpora in 16 Peruvian languages that belong to different linguistic families (including Quechua).

Regarding speech resources, Guarani has a spoken corpus comprised by $1,000$ phrases from 110 different speakers, it was collected through a web interface (Maldonado et al., 2016), however, this dataset

[5] www.gdn.unam.mx

[6] https://github.com/pywirrarika/wixarikacorpora

[7] https://www.uni-mainz.de/cgi-bin/guarani2/dictionary.pl

has not been publicly released. The Chatino language (approx. 45K speakers) is an Oto-Manguean language spoken in southern Mexico, recently a language documentation and revitalization project has been developed. They use automatic speech recognition and forced alignment tools to time align transcriptions. Parts of this resource are freely available (Cavar et al., 2016).

There are some other types of datasets that are useful for developing language technologies, e.g., morphological annotated data. The CoNLL-SIGMORPHON 2017 Shared Task (Cotterell et al., 2017) released a large morphological database with inflection information of 52 languages, including Haida (7,040 words), Navajo (12,000 words), and Quechua (12,000 words), all of these are indigenous languages spoken in the Americas. In the same way, there is a Oto-manguean inflectional class database which contains over 13,000 verbal entries from twenty Oto-Manguean languages spoken in Mexico, along with information about each verb's inflectional class membership (Palancar and Feist, 2015). For morphological segmentation a data set of four Uto-Aztecan languages[8] were used and released by Kann et al. (2018) (4,468 segmented words from the Mexicanero, Nahutal, Wixarika and Yorem Nokki languages). The UQAILAUT Project contains roots, lexicalized words, infixes, noun and verb endings for the Inuktitut language[9] (Farley, 2012). Plain Cree language, spoken in North America (Algic language family) has also a lexicon databases (16,452 words) collected by Walters et al. (2002) and Wolfart and Pardo (1973).

To improve the data recollection, Dunham et al. (2014) developed tools for annotation of text and audio for Blackfoot, Gitksan, Okanagan, Tlingit, Plains Cree, Coeur d'Alene and Kwak'wala. With these tool they gather 19,187 word forms, 324 texts and 18.8 GB of audio.

A collection of datasets have been developed for the Mapudungun or Mapuche language spoken mainly in Chile (Araucanian language family) (Huenchullan, 2000; Monson et al., 2004). An audio dataset with 170 hours of spoken Mapudungun, that covers three dialects (120 hours of Nguluche, 30 hours of Lafkenche and 20 hours of Pewenche) has been released. This resource contains a word list with the 70,000 most frequent full form words (stem plus inflections) to support a spelling checker for Mapudungun. Also a bilingual Mapudungun-Spanish lexicon was included, containing sample entries 1,600.

Annotated data corresponding to higher linguistic levels is harder to find. For instance, almost no treebanks have been developed for the indigenous languages of the Americas. To our knowledge, the only available dataset is a parallel aligned treebank between Quechua and Spanish (Rios et al., 2008) with 2,000 annotated sentences.

It is important to mention that many of the languages spoken in the Americas have a Wikipedia's set of articles available[10]. This is useful for building monolingual and comparable corpora. Furthermore, Wikipedia can be a helpful resource to predict Part-of-Speech (POS) tags for low resource languages and other tasks (Hoenen, 2016). In any case, one common limitation of the digital resources for these languages is the lack of orthographic standardization and difficulties for processing certain types of characters. Table 1 summarizes the above-mentioned resources.

3 Morphological segmentation and analyses

Morphology has been studied in NLP field focusing mainly on the following tasks: lemmatization, stemming, segmentation, analysis and inflection/reinfection. These tasks serve to other higher level tasks such as machine translation. On this regard, there have been several studies which have been applied to the Americas languages.

In NLP, lemmatization and stemming methods are used to reduce the morphological variation by converting words forms to a standard form, i.e., a lemma or a stem. However, most of these technologies are focused in a reduced set of languages. For languages like English there are plenty of resources, however, this is not the case for all the languages. Specially for languages in the Americas with rich

[8] Available at `http://turing.iimas.unam.mx/wix/mexseg`

[9] `http://www.inuktitutcomputing.ca/DataBase/info.php`

[10] The available languages in wikipedia can be consulted at: `https://es.wikipedia.org/wiki/Portal:Lenguas_indgenas_de_Amrica`. Until the publication of this article there are only entries in: Nahuatl, Navajo, Guarani, Aymara, Klaalisut, Esquimal, Inukitut, Cherokee, and Cree.

Type of resource	Languages	Size	Reference
Parallel	Nahuatl-Spanish	18K sentences	Gutierrez-Vasques et al. (2016)
Parallel	Wikarika-Spanish	8K sentences	Mager et al. (2018)
Parallel	Shipibo konibo - Spanish	11.8K sentences	Galarreta et al. (2017)
Parallel	Spanish-English-Guarani	250K sentences	Abdelali et al. (2006)
Parallel	1259 languages		Mayer and Cysouw (2014)
POS Tagged	Shipibo konibo	217 sentences	Pereira-Noriega et al. (2017)
Lemmatized words	Shipibo konibo	3.5K words	Pereira-Noriega et al. (2017)
Dictionary	Shipibo konibo - Spanish	3.5K words	Pereira-Noriega et al. (2017)
Dictionary	Nahuatl		Palancar and Feist (2015)
Dictionary	Guarani		(Ramírez and Wolf, 1996)
Speech	Guarani	1K phrases	Maldonado et al. (2016)
Speech	Chatino	10 hours with Transcription	Cavar et al. (2016)
Speech	Blackfoot, Nata, Gitksan, Okanagan, Tlingit, Plains Cree, Ktunaxa, Coeur d'Alene, Kwak'wala	19.8 GB	Dunham et al. (2014)
Speech	Mapudungun	170 hours	Huenchullan (2000) and Monson et al. (2004)
Morphological Inflection	Quechua, Navajo, Haida	31K words	Cotterell et al. (2017)
Morphological Inflection	20 Oto-Manguean languages	13K verbs	Palancar and Feist (2015)
Morphological Segmentation	Uto-Aztecan languages (Mexicanero, Nahutal, Wixarika, Yorem Nokki)	4.4K words	Kann et al. (2018)
Morphological segmentation	Inuktitut	2K roots, 1.8K affixes	Farley (2012)
Monolingual	16 Peruvian languages	Unknown	Espichán-Linares and Oncevay-Marcos (2017)
Monolingual	Plain Cree	16K words	Walters et al. (2002) and (Wolfart and Pardo, 1973)
Treebank	Quechua	2K sentences	Rios et al. (2008)

Table 1: Digital available resources of American Indigenous Languages for NLP

morphological phenomena, and not always suffixal, where it is not enough to remove inflectional endings in order to obtain a stem.

Morphological segmentation is the task of splitting a word into the surface forms of its smallest meaning-bearing units, its *morphemes*. On the other hand, Morphological analysis not only focuses in the segmentation of words, but also in assigning tags to each part of the word. There are several approaches to do these tasks, i.e., rule-based, semi-supervised and unsupervised (Goldsmith, 2001; Creutz and Lagus, 2002; Kohonen et al., 2010). Some examples of rule-based methods applied to the Americas languages are the Finite State approaches to model the morphology of a language: plains Cree (Arppe et al., 2017; Harrigan et al., 2017; Wolfart and Pardo, 1973; Snoek et al., 2014), East Cree (Arppe et al., 2017) , for the East Odawa dialect of Ojibwe (Bowers et al., 2017), for Mohawk (Iroquoian language family) (Assini, 2014), for the Bribri (Chibchense language family) (Solórzano, 2017) using the FOMA tool (Hulden, 2009), Quechua (Vilca et al., 2012; Monson et al., 2006), Mapudungun (Monson et al., 2006), and the Argentinian branch of Quechua and Toba (Porta, 2010). More recently a new hybrid ap-

proach of finite-state transducer (FST) with statistical inference is part of the *Basic Language Technology Toolkit for Quechua* (Rios, 2016).

For Uto-Aztecan languages, there exists a computational tool called "*chachalaca*" that performs morphological analysis (Thouvenot, 2011) of Nahuatl. This is a rule-based software focused on Classical Nahuatl, it is able to generate more than one morphological analysis candidate per word. It is based on grammars that describe most of the 16th-century-word constructions. Additionally, Mager et al. (2018) propose a morphological segmentation tool for the Wixarika language, with a supervised approach, using previous given morphological tables and a probabilistic model to infer the inherent morphological rules.

Regarding to unsupervised methods, neural methods have been used to tackle the rich morphology of the languages of the continent. Micher (2017) propose a Segmental Recurrent Neural Network (RNN) for segmenting and tagging Inuktitut. Kann et al. (2018) used a set of extensions to the Encoder-Decoder RNN architecture with Gated Recurrent Units (GRU) for automatically segmenting four Uto-Aztecan languages (Mexicanero, Nahuatl, Wixarika and Yorem Nokki). Semisupervised segmentation approaches like Morfessor have also been successfully applied to Nahuatl (Gutierrez-Vasques, 2017). For the Uto-Aztecan language Tarahumara and the Mayan language Chuj, there are works that try to automatically discover affixes through unsupervised approaches (Medina-Urrea, 2007; Medina-Urrea, 2008; Medina Urrea and García, 2006).

Lately, there has been interest in the reinflection task, i.e., generating an inflected form for a given target tag and lemma. The CoNLL-SIGMORPHON Shared Task (Cotterell et al., 2016; Cotterell et al., 2017) released a dataset for reinflection of 52 languages, including 3 Native American languages. The systems that got the best performance (Kann and Schütze, 2016; Kann and Schütze, 2017; Makarov et al., 2017).

In some cases it is difficult to disambiguate between homonym morphs. To deal with this problem, Rios et al. (2008) used Conditional Random Fields (CRFs) (Lafferty et al., 2001).

Most of the methods that we found that deal with morphology are based on FST. However, the indigenous languages of the continent are far too diverse, it would be expensive to build such analyzers with expert knowledge for each language, besides the fact that the analyzers need to be constantly updated to cope with language change (Creutz and Lagus, 2005).

4 Machine Translation

Machine Translation is a natural task for indigenous languages, since it might provide a communication window with more popular languages. The development of MT systems for indigenous languages have follow the trends in the field, from rule-based, to statistical and neural based approaches.

Rule-Based Machine Translation (RBMT) approaches are sometimes suitable for low resource languages since they do not require aligned parallel corpora. However. In recent years, research on data-driven approaches has increased, with the aim to overcome the scarcity of data using different methods. In any case, translation of low-resource languages represents an interesting and active research problem in the NLP field.

In the case of native American languages, there have been some efforts in building rule-based systems. The Apertium system (Tyers et al., 2009; Forcada et al., 2011) is a big help for this approach, and at least two languages has translation teams working with it. This is the case of Quechua (Eastern Apurimac Quechua and Cusco Quechua)-Spanish (Cavero and Madariaga, 2007) and Spanish-Wayuunaiki (Spoken in Venezuela and Colombia) (Fernández et al., 2013). Other RBMT systems were created for Aymara-Spanish (spoken in Peru) (Coler and Homola, 2014) Wayuunaki-Spanish, Quechua-Spanish and Mapuche-Spanish (Monson et al., 2006). For the latter a web available translator[11](González Hernández, 2016) is available. We found that there are mobile apps for translating indigenous languages, this is the case of Zapotec-Spanish language pair (spoken in Mexico from the Oto-Manguean family)[12].

All of this RBMT systems have a set of shortcomings. The majority is not able to translate complex constructions, specially when the languages are distant from each other, which increases the complexity

[11] `http://142.4.219.173/wt/`

[12] `https://play.google.com/store/apps/details?id=com.SimplesoftMx.Didxazapp&hl=es`

of the machine translation rules. One way to overcome this is to include linguistic information, e.g., morphology, syntax. However, this kind of knowledge or linguistic tools is not always available, especially for low-resource languages. Experiments using the Example Based Machine Translation (EBMT) methodology are not common, we only found the work of Llitjós et al. (2005) and Monson et al. (2006) for the Mapuche-Spanish pair.

Statistical Machine Translation (SMT) Systems are data-driven since they use vast amounts of parallel corpora to model the translations between sentences or subunits. Their performance is highly dependent on the number of training data; they represent a challenge when low resource conditions are faced. In the case of the native languages of the Americas, they tend to be morphologically rich and this must be taken into account to improve the translation and reduce the data sparseness. An example of this is the Wixarika-Spanish SMT system that aligns Wixarika morphemes with Spanish words or tokens (Mager Hois et al., 2016; Mager Hois, 2017)[13]. A similar case can be found for the Nahuatl-Spanish pair. Uto-Aztecan languages can be highly agglutinative with the polysynthetic tendency, Gutierrez-Vasques (2015) extracts bilingual correspondences from a parallel corpus, by aligning the Nahuatl non-grammatical morphs to Spanish words. Another example was collected for the pair Mixteco-Spanish (Santiago, 2017). The same trend can be observed in SMT for Shipibo-konibo (Galarreta et al., 2017).

Regarding commercial systems, Microsoft has targeted some languages spoken in Mexico, Mayan and Otomi (Queretaro variant)[14]. SMT was also applied for the Guarani, it translates to Spanish, but also to English, French, Italian, German and Portuguese[15].

Recently, there has been an increasing interest in Neural Machine Translation (NMT) models, which are also statistical based, but they use neural networks architectures that are feed with very big amounts of parallel corpora. Mager and Meza (2018) showed that in such low-resource scenarios, translating from Mexicanero, Nahuatl, Purepecha, Wixarika and Yorem-Nokki to Spanish, SMT systems achieve better performance than NMT. Even though these architectures are not suitable for low-resource settings, there have been some recent efforts to adapt them. Soriano (2018) experimented with the Mexican Purepecha (an isolated language with about 140 thousand speakers) using the OpenNMT toolkit (Klein et al.,). Tiedemann (2018) took the massive bible corpus (Mayer and Cysouw, 2014) and trained a multilingual NMT model to improve overall translation performance. Experiments included Oto-manguean, Quechua and Mayan families. Moreover, empirical results (Vania and Lopez, 2017) show that problem of data sparsity of rich morphological languages can be handled with subword models: the usage of character level NMT improve performance over token level translation and unsupervised morphological segmentation (Creutz and Lagus, 2002). But their experiment also conclude that a canonical segmentation enhances character level translation.

In order to alleviate the lack of resources automatic data recollection has been proposed, this has been tried for Guarani language (Abdelali et al., 2006). Moreover, it would be very useful to have big repositories of translated texts. One alternative is to create parallel corpora between many languages using manual translations (controlled elicitation) as described for the Mapudungun (Mapuche) language (Probst et al., 2001).

In any case, SMT and NMT systems should be adapted to deal with the scarcity, of the sparseness of word forms and the rich morphology of languages. Although there are works that try to deal with morphology (Virpioja et al., 2007; Popovic and Ney, 2004; Costa-jussà and Fonollosa, 2016; Sennrich et al., 2016; Dalvi et al., 2017), they are rarely applied to Native American languages.

5 Other studies and tools

5.1 Multilinguality and Code-Switching

Most native speakers of indigenous languages are at least bilingual, they have to communicate using the primary or official language of their own country, i.e., Spanish, Portuguese, French or English. Only few

[13] Available at `http://turing.iimas.unam.mx/wix`

[14] `https://www.microsoft.com/en-us/translator/languages.aspx`

[15] `http://www.iguarani.com/?p=traductor`

communities remain completely monolingual in their native language, moreover, modern migrations and the use of social networks contribute to bilingualism and code-switching.

Code-switching occurs when a speaker alternates between two or more languages in a conversation. This adds a challenge when doing NLP for this kind of data. Code-switching is not a new phenomenon, it can be found in historical documents, Garrette and Alpert-Abrams (2016) proposes an unsupervised approach of paired encoding (words and characters) to improve language modeling (Latin, Spanish and Nahuatl) in an Optical character recognition (OCR) task. King and Abney (2013) applies weakly supervised methods for labeling the language of each word in documents that can have many mixed languages. The targeted languages are 30, including Chippewa, Nahuatl, and Ojibwa.

Being able to automatically detect Code-switching could be useful when doing NLP for minority languages, for instance to use the web as a source for a corpus.

From the quantitative linguistics perspective, parallel corpora of an outstanding number of languages have been extracted from the Bible and used to perform typological studies in many languages, included native languages of the Americas. For instance, exploring the tense behavior (Asgari and Schütze, 2017), contrasting the morphological complexity in many languages (Bentz et al., 2016; Kettunen, 2014) just to mention some.

5.2 Language Tools

For some rich resource languages, there are already available NLP tools that deal with several phenomena and linguistics levels of processing. However, for low resource languages, there is still much work to do. In this section we summarize some of the works related with POS Tagging, OCR, Parsing, Spell Checking, Language Identification and other tasks, that we have found for the languages of the Americas.

Speech synthesis and recognition has made some progress for the Raramuri language (Urrea et al., 2009), using a unit selection approach based on function words, suffix sequences and diphones of the language. For speech recognition, Maldonado et al. (2016) applied on Guarani a Hidden Markov Model (HMM) with the CMU Sphinx toolkit (Lamere et al., 2003). Coto-Solano and Solórzano (2016) proposes an automatic aligner of text and voice for the indigenous language Bribri of Costa Rica.

Part-of-Speech (POS) tagging assigns a category from a given symbol set to each token in an input string. It is used as a prepossessing step that serves as input for other tasks or for higher level NLP task. POS Tagging was also incorporated into the Peruvian Shipibo-konibo NLP toolkit (Pereira-Noriega et al., 2017).

Spell checking is not a trivial task for highly agglutinative and polysynthetic languages, that can't rely on a token based evaluation, and need sub-word level models. We found only two tools that handle this issue: Monson et al. (2006) build a dictionary based tool for Quechua and Alva and Oncevay (2017) for Shipibo-Konibo used rule-based analysis and dictionaries.

Another field that is crucial to increase the amount of digitized data for other tasks is **Optic Character Recognizing** (OCR). Maxwell and Bills (2017) studied the challenges regarding the digitalization of Tzeltal-Spanish, Muinane-Spanish, Cubeo-Spanish dictionaries. Garrette and Alpert-Abrams (2016) developed an unsupervised transcription model for dealing with orthographic variation in digitized historical documents, some of them were written in Nahuatl.

In the context of indigenous **Language Identification** (LID), Espichán-Linares and Oncevay-Marcos (2017) studied LID on 17 languages from the Arawak, Aru, Jíbaro, Pano and Quechua linguistic families. The proposed models were flexible enough to handle the lack of orthographic standardization of the language.

Another important NLP task is **parsing**. The research in this area is also weak, however, the Quechua-Spanish Treebank helped to perform some experiments in this topic (Bresnan et al., 2015; Rios, 2016). For other languages parsing experiments were performed on Ayamara (Homola, 2011) using Lexical-Functional Grammar (LFG).

6 Discussion

Study of the languages of the Americas has increased in the recent years, in both the linguistic and the language technology fields. Many factors have contributed to this, such as speakers self-awareness about the importance of their languages and digital inclusion. Figure 1 shows that many of the papers that we reviewed, were published from year 2000 to present day, with a notable increase in the activity in the last five years. The most studied NLP tasks are Machine Translation and Morphology, however, from 2013 upon now, other tasks, e.g., POS-tagging, parsing, speech, spell correction, also received attention.

Despite the fact that we found NLP contributions for around 35 languages, this is still a small number if we take into account the big diversity and number of languages that exist in the continent. Table 2 showed that some linguistic families have concentrated the attention, but even for these languages the developed technology is not enough. We noticed that North American languages are the most studied, despite some of them don't have a big number of speakers compared to other indigenous languages, e.g., Navajo, Haida, Cree, Chippewa, Ojibwa, Blackfoot, Nata, Gitksan, Okanagan, Tlingit, Plains Cree, Ktunaxa, Coeur d'Alene, Kwak'wala, and Inuktitut. Uto-Aztecan language family that includes languages like Raramuri, Nahuatl, Wixarika, Yorem Nokki, Mexicanero (spoken mainly in Mexico) have also received attention from the NLP community. Regarding to South America, the most spoken native languages, Quechua, Mapuche, Guarani and Ayamara, have several resources available. Surprisingly, languages with less speakers such as Shipibo-konibo, Arawak, Aru, Jíbaro, Pano and Wayuunaki have been also studied.

The diversity in the linguistic phenomena of these languages makes developing language technologies a challenging task. In recent years, NLP and Machine Learning fields have paid attention to low resource settings, organizing workshops and special tracks to tackle this issue. Indigenous languages could be greatly benefit from this kind of research in the future. In particular, American languages with rich morphology, e.g., agglutinative and polysyntehthic, seems to benefit from approaches that take into account the morphology and sub-word modeling.

We also noticed that some NLP tasks that are considered almost solved for languages like English, need to be adapted or started from scratch when applied to the languages of the American continent. Moreover, fields like machine translation could enable in the future the creation of multilingual technologies for all of the languages in the world that face a similar situation, this could have a great impact in these communities.

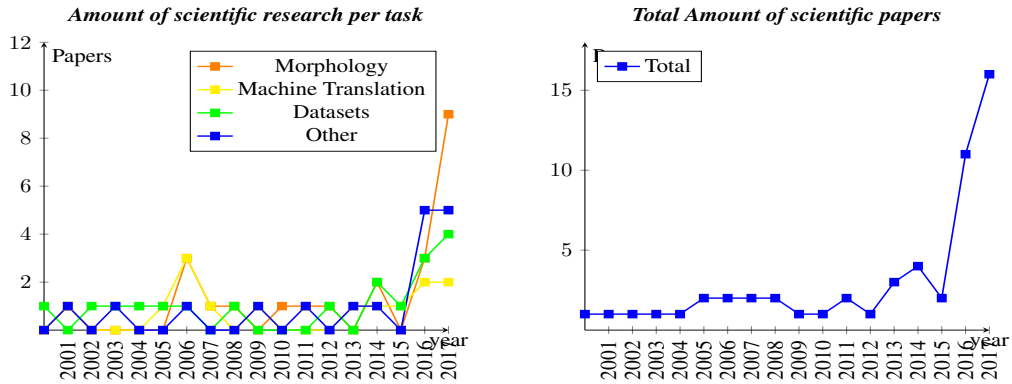


Figure 1: NLP papers and digital resources that contain any indigenous language of the Americas (between 2000 and 2017)

7 Conclusions

In this work, we presented a review of NLP research focused on Indigenous Languages of the Americas. We showed which languages have available digital resources and their related tools. Research has focused in tasks like morphology and machine translation, however, there is still a lot of work to be done since these languages exhibit a wide range of linguistic phenomena while resources are scarce.

Through this work, we discussed some of the challenges that must be taken into account, e.g., small datasets, high dialectal variation, rich morphology, lack of orthographic normalization, scarcity of linguistic preprocessing tools.

NLP research for these languages can broad the understanding of human language structures and help to build more general computational models. Moreover, the development of language technologies can have a positive social impact for the speakers of the indigenous languages, helping to maintain the living cultural heritage that each language represents.

Acknowledgements

This work was supported by the Mexican Council of Science and Technology (CONACYT), fund 2016-01-2225. We will also thank the reviewers for their valuable comments and to Katharina Kann for her comments and support.

References

Ahmed Abdelali, James Cowie, Steve Helmreich, Wanying Jin, Maria Pilar Milagros, Bill Ogden, Hamid Mansouri Rad, and Ron Zacharski. 2006. Guarani: a case study in resource development for quick ramp-up mt. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas,"Visions for the Future of Machine Translation*, pages 1–9.

Carlo Alva and Arturo Oncevay. 2017. Spell-checking based on syllabification and character-level graphs for a peruvian agglutinative language. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 109–116.

Antti Arppe, Marie-Odile Junker, and Delasie Torkornoo. 2017. Converting a comprehensive lexical database into a computational model: The case of East Cree verb inflection. *ComputEL-2*, page 52.

Ehsaneddin Asgari and Hinrich Schütze. 2017. Past, present, future: A computational investigation of the typology of tense in 1000 languages. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 113–124.

Alicia Alexandra Assini. 2014. *Natural language processing and the Mohawk language: creating a finite state morphological parser of Mohawk formal nouns*. Scholars' Press.

Emily M Bender. 2011. On achieving and evaluating language-independence in nlp. *Linguistic Issues in Language Technology*, 6(3):1–26.

Christian Bentz, Tatyana Ruzsics, Alexander Koplenig, and Tanja Samardzic. 2016. A comparison between morphological complexity measures: typological data vs. language corpora. In *Proceedings of the Workshop on Computational Linguistics for Linguistic Complexity (CL4LC)*, pages 142–153.

Dustin Bowers, Antti Arppe, Jordan Lachler, Sjur Moshagen, and Trond Trosterud. 2017. A morphological parser for Odawa. *ComputEL-2*, page 1.

Joan Bresnan, Ash Asudeh, Ida Toivonen, and Stephen Wechsler. 2015. *Lexical-functional syntax*, volume 16. John Wiley & Sons.

Malgorzata Cavar, Damir Cavar, and Hilaria Cruz. 2016. Endangered language documentation: Bootstrapping a Chatino speech corpus, forced aligner, asr. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*, Paris, France, may. European Language Resources Association (ELRA).

Indhira Castro Cavero and Jaime Farfán Madariaga. 2007. Traductor morfológico del castellano y quechua. *Revista I+i*, 1(1).

Matt Coler and Petr Homola. 2014. Rule-based machine translation for aymara. *Endangered Languages and New Technologies*, page 67.

Marta R Costa-jussà and José AR Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 357–361.

Rolando Coto-Solano and Sofía Flores Solórzano. 2016. Alineación forzada sin entrenamiento para la anotación automática de corpus orales de las lenguas indígenas de costa rica. *Káñina*, 40(4):175–199.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Gėraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, et al. 2017. CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection in 52 Languages. *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 1–30.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Workshop on Morphological and Phonological Learning*.

Mathias Creutz and Krista Lagus. 2005. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Helsinki University of Technology.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, and Stephan Vogel. 2017. Understanding and improving morphological learning in the neural machine translation decoder. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 142–151.

Andrés de Olmos, Ascensión H de León-Portilla, and Miguel León Portilla. 2002. *Arte de la lengua mexicana*, volume 9. UNAM.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Joel Dunham, Gina Cook, and Joshua Horner. 2014. Lingsync & the online linguistic database: New models for the collection and management of data for language communities, linguists and language learners. In *Proceedings of the 2014 Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 24–33.

Alexandra Espichán-Linares and Arturo Oncevay-Marcos. 2017. A low-resourced peruvian language identification model. In *Proceedings of the SIMBig 2017 Track on Applied Natural Language Processing, ANLP 2017*.

Benoit Farley. 2012. The Uqailaut project. *URL http://www. inuktitutcomputing. ca*.

Dayana Iguarán Fernández, Ornela Quintero Gamboa, Jose Molina Atencia, and Oscar Elías Herrera Bedoya. 2013. Design and implementation of an "web api" for the automatic translation colombia's language pairs: Spanish-Wayuunaiki case. In *Communications and Computing (COLCOM), 2013 IEEE Colombian Conference on*, pages 1–9. IEEE.

Mikel L Forcada, Mireia Ginestí-Rosell, Jacob Nordfalk, Jim O'Regan, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Felipe Sánchez-Martínez, Gema Ramírez-Sánchez, and Francis M Tyers. 2011. Apertium: a free/open-source platform for rule-based machine translation. *Machine translation*, 25(2):127–144.

Ana Paula Galarreta, Andrés Melgar, and Arturo Oncevay. 2017. Corpus creation and initial SMT experiments between Spanish and Shipibo-konibo. In *Proceedings of RANLP*.

Dan Garrette and Hannah Alpert-Abrams. 2016. An unsupervised model of orthographic variation for historical document transcription. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 467–472.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

José González Hernández. 2016. *Herramienta de traduccin automática de wayuunaiki a español. Caso de estudio: sintagmas nominales y verbales simples*. Ph.D. thesis, Universidad de Zulia.

Ximena Gutierrez-Vasques, Gerardo Sierra, and Hernandez Isaac. 2016. Axolotl: a web accessible parallel corpus for Spanish-Nahuatl. In *International Conference on Language Resources and Evaluation (LREC)*.

Ximena Gutierrez-Vasques. 2015. Bilingual lexicon extraction for a distant language pair using a small parallel corpus. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 154–160.

Ximena Gutierrez-Vasques. 2017. Exploring bilingual lexicon extraction for Spanish-Nahuatl. In *ACL Workshop in Women and Underrepresenting Minorities in Natural Language Processing*.

Atticus G Harrigan, Katherine Schmirler, Antti Arppe, Lene Antonsen, Trond Trosterud, and Arok Wolvengrey. 2017. Learning from the computational modelling of Plains Cree verbs. *Morphology*, 27(4):565–598.

Armin Hoenen. 2016. Wikipedia titles as noun tag predictors. In *International Conference on Language Resources and Evaluation (LREC)*.

Petr Homola. 2011. Parsing a polysynthetic language. In *RANLP*, pages 562–567.

Carolina Huenchullan. 2000. Data collection and language technologies for Mapudungun. In *International Conference on Language Resources and Evaluation (LREC)*.

Mans Hulden. 2009. Foma: a finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session*, pages 29–32. Association for Computational Linguistics.

Katharina Kann and Hinrich Schütze. 2016. MED: The LMU system for the SIGMORPHON 2016 shared task on morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 62–70.

Katharina Kann and Hinrich Schütze. 2017. The LMU system for the CoNLL-SIGMORPHON 2017 shared task on universal morphological reinflection. *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 40–48.

Katharina Kann, Manuel Mager, Ivan Meza, and Hinrich Shütze. 2018. Fortification of neural morphological segmentation models for polysynthetic minimal-resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 47–57.

Kimmo Kettunen. 2014. Can type-token ratio be used to show morphological complexity of languages? *Journal of Quantitative Linguistics*, 21(3):223–245.

Ben King and Steven Abney. 2013. Labeling the languages of words in mixed-language documents using weakly supervised methods. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1110–1119.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Oskar Kohonen, Sami Virpioja, and Krista Lagus. 2010. Semi-supervised learning of concatenative morphology. In *ACL Special Interest Group on Computational Morphology and Phonology*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Paul Lamere, Philip Kwok, Evandro Gouvea, Bhiksha Raj, Rita Singh, William Walker, Manfred Warmuth, and Peter Wolf. 2003. The CMU SPHINX-4 speech recognition system. In *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2003), Hong Kong*, volume 1, pages 2–5.

Ariadna Font Llitjós, Roberto Aranovich, and Lori Levin. 2005. Building machine translation systems for indigenous languages. In *Second Conference on the Indigenous Languages of Latin America (CILLA II), Texas, USA*.

Manuel Mager and Ivan Meza. 2018. Hacia la traducción automática de las lenguas indígenas de méxico. In *Proceedings of the 2018 Digital Humanities Conference*. The Association of Digital Humanities Organizations.

Manuel Mager, Diónico Carrillo, and Ivan Meza. 2018. Probabilistic finite-state morphological segmenter for the Wixarika (Huichol) language. *Journal of Intelligent & Fuzzy Systems*, 34(5):3081–3087.

Jesus Manuel Mager Hois, Carlos Barron Romero, and Ivan Vladimir Meza Ruíz. 2016. Traductor estadístico wixarika - español usando descomposición morfológica. *COMTEL*, (6).

Jesus Manuel Mager Hois. 2017. Traductor híbrido wixárika - español con escasos recursos bilingües. Master's thesis, Universidad Autónoma Metropolitana.

Peter Makarov, Tatiana Ruzsics, and Simon Clematide. 2017. Align and copy: Uzh at sigmorphon 2017 shared task for morphological reinflection. *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 49–57.

Diego Manuel Maldonado, Rodrigo Villalba Barrientos, and Diego P Pinto-Roa. 2016. Eñe'e: Sistema de reconocimiento automático del habla en guaraní. In *Simposio Argentino de Inteligencia Artificial (ASAI 2016)-JAIIO 45 (Tres de Febrero, 2016)*.

Michael Maxwell and Aric Bills. 2017. Endangered data for endangered languages: Digitizing print dictionaries. *ComputEL-2*, page 85.

Thomas Mayer and Michael Cysouw. 2014. Creating a massively parallel bible corpus. *Oceania*, 135(273):40.

Norman A McQuown. 1955. The indigenous languages of latin america. *American Anthropologist*, 57(3):501–570.

A Medina Urrea and M Alvarado García. 2006. Un experimento de reconocimiento automático de la derivación léxica en el ralámuli. *La lengua y la antropología para un conocimiento global del hombre*.

Alfonso Medina-Urrea. 2007. Affix discovery by means of corpora: Experiments for Spanish, Czech, Ralámuli and Chuj. In *Aspects of Automatic Text Analysis*, pages 277–299. Springer.

Alfonso Medina-Urrea. 2008. Affix discovery based on entropy and economy measurements. *Texas Linguistics Society*, pages 99–112.

Jeffrey C Micher. 2017. Improving coverage of an inuktitut morphological analyzer using a segmental recurrent neural network. *ComputEL-2*, page 101.

Marianne Mithun. 2001. *The languages of native North America*. Cambridge University Press.

Marianne Mithun. 2017. Polysynthesis in north america. In *The Oxford Handbook of Polysynthesis*.

Christian Monson, Lori Levin, Rodolfo M Vega, Ralf D Brown, Ariadna Font-Llitjos, Alon Lavie, Jaime G Carbonell, Eliseo Cañulef, and Rosendo Huisca. 2004. Data collection and analysis of Mapudungun morphology for spelling correction. *Computer Science Department*, page 300.

Christian Monson, Ariadna Font Llitjós, Roberto Aranovich, Lori Levin, Ralf Brown, Eric Peterson, Jaime Carbonell, and Alon Lavie. 2006. Building NLP systems for two resource-scarce indigenous languages: mapudungun and quechua. *Strategies for developing machine translation for minority languages*, page 15.

Sebastian Nordhoff, Harald Hammarström, Robert Forkel, and Martin Haspelmath. 2013. *Glottolog 2.0*. Max Planck Institute for Evolutionary Anthropology.

Helen O'Horan, Yevgeni Berzak, Ivan Vulić, Roi Reichart, and Anna Korhonen. 2016. Survey on the use of typological information in natural language processing. *arXiv preprint arXiv:1610.03349*.

E.L. Palancar and T. Feist. 2015. *Oto-Manguean Inflectional Class Database*. University of Surrey.

José Pereira-Noriega, Rodolfo Mercado-Gonzales, Andrés Melgar, Marco Sobrevilla-Cabezudo, and Arturo Oncevay-Marcos. 2017. Ship-lemmatagger: Building an NLP toolkit for a peruvian native language. In *International Conference on Text, Speech, and Dialogue*, pages 473–481. Springer.

Maja Popovic and Hermann Ney. 2004. Towards the use of word stems and suffixes for statistical machine translation. In *International Conference on Language Resources and Evaluation (LREC)*.

Andrés Osvaldo Porta. 2010. The use of formal language models in the typology of the morphology of amerindian languages. In *Proceedings of the ACL 2010 Student Research Workshop*, pages 109–113. Association for Computational Linguistics.

Katharina Probst, Ralf D Brown, Jaime G Carbonell, Alon Lavie, Lori Levin, and Erik Peterson. 2001. Design and implementation of controlled elicitation for machine translation of low-density languages. *Computer Science Department*.

Gilbert Ramírez and Lustig Wolf. 1996. Interactive guarani dictionary. `https://www.uni-mainz.de/cgi-bin/guarani2/dictionary.pl`. Accessed: 2018-03-16.

Annette Rios, Anne Göhring, and Martin Volk. 2008. A Quechua-Spanish parallel treebank. *LOT Occasional Series*, 12:53–64.

Annette Rios. 2016. A basic language technology toolkit for quechua.

H. Santiago. 2017. Método para la alieación automática de textos entre los idiomas mixteco y español. Master's thesis, Centro Nacional de Investigación y Desarrollo Tecnológico.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1715–1725.

Gary F Simons and Charles D Fennig. 2017. Ethnologue: Languages of the world. *SIL, Dallas, Texas*.

Conor Snoek, Dorothy Thunder, Kaidi Loo, Antti Arppe, Jordan Lachler, Sjur Moshagen, and Trond Trosterud. 2014. Modeling the noun morphology of Plains Cree. In *Proceedings of the 2014 Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 34–42.

Sofía Flores Solórzano. 2017. Desarrollo de un analizador automático de estados finitos para la lengua bribri. `http://morphology.bribri.net/`.

Miguel Soriano. 2018. Traductor Automático español –purépecha mediante OpenNMT. Master's thesis, Universidad de Guadalajara, Mexico.

Marc Thouvenot. 2005. Gran diccionario náhuatl. `http://www.gdn.unam.mx/`.

Marc Thouvenot. 2011. Chachalaca en cen, juntamente. In *Compendio Enciclopedico del Nahuatl, DVD*. INAH.

Jörg Tiedemann. 2018. Emerging language spaces learned from massively multilingual corpora. *arXiv preprint arXiv:1802.00273*.

Francis M Tyers, Mikel L Forcada, and Gema Ramırez-Sánchez. 2009. The Apertium machine translation platform: Five years on. In *Proc. of the First Intl. Workshop on Free/Open-Source Rule-Based Machine Translation*, pages 3–10.

Alfonso Medina Urrea, José Abel Herrera Camacho, and Maribel Alvarado Garcıa. 2009. Towards the speech synthesis of raramuri: A unit selection approach based on unsupervised extraction of suffix sequences. *Advances in Computational Linguistics*, page 243.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2016–2027, Vancouver, Canada, July. Association for Computational Linguistics.

Calderon Vilca, Hugo David, Cárdenas Mariñó, Flor Cagniy, and Edwin Fredy Mamani Calderon. 2012. Analizador morfológico de la lengua quechua basado en software libre helsinkifinite-statetransducer (hfst). *COMTEL*.

Sami Virpioja, Jaakko J Väyrynen, Mathias Creutz, and Markus Sadeniemi. 2007. Morphology-aware statistical machine translation based on morphs induced in an unsupervised manner. *Machine Translation Summit XI*, 2007:491–498.

Claudio Wagner. 2016. Las lenguas indígenas de américa (lenguas amerindias). *Revista Documentos Lingüísticos y Literarios UACh*, (17):30–37.

JCW Walters, M Mirten, P Hernández, E Pérez, and CH Upton. 2002. Diccionario náhuatl de los municipios de mecayapan y tatahuicapan de juárez. *Veracruz. 2ª edición electrónica (cuerpo), Instituto Lingüístico de Verano, AC www. sil. org/mexico/nahuatl/istmo/G020a-DiccNahIst-NAU. htm*.

H Christoph Wolfart and Francis Pardo. 1973. Computer-assisted linguistic analysis. *University of Manitoba Anthropology Papers*, (6).

Appendix A. Language Families

Table 2 summarizes the language families for which we were able to identify some digital and technological resources during this research. We distinguish among only two geographical macroareas: North America (it includes Central America) and South America (Dryer and Haspelmath, 2013).

L. Family	Macroarea	Papers	L. Family	Macroarea	Papers
Uto-Aztecan	North A.	16	Mayan	North A.	4
Oto-Manguean	North A.	3	Arawakan	South A.	3
Haida	North A.	4	Ayamaran	South A.	2
Na-Dene	North A.	5	Aru	South A.	1
Eskimo-Aleut	North A.	2	Jibaro	South A.	1
Algic	North A.	8	Bora–Witóto	South A.	1
Tsimshianic	North A.	1	Tucanoan	South A.	1
Penutian	North A.	1	Araucanian	South A.	7
Salishan	North A.	2	Panoan	South A.	4
Wakashan	North A.	1	Tupian	South A.	5
Iroquoian	North A.	1	Quechuan	South A.	15
Chibchan	North A.	2	Guaicuruan	South A.	1

Table 2: Language families (L. Family) for which some technology was found, and the number of NLP/Computer Linguistic papers referring to each (one paper can reference more than one languages). North A. stands for North America and South A. for South America.

Low-resource Cross-lingual Event Type Detection in Documents via Distant Supervision with Minimal Effort

Aldrian Obaja Muis Naoki Otani Nidhi Vyas Ruochen Xu
Yiming Yang Teruko Mitamura Eduard Hovy
Language Technology Institute
Carnegie Mellon University
Pittsburgh, PA
{amuis,notani,nkvyas,ruochenx,yiming,teruko,ehovy}@andrew.cmu.edu

Abstract

The use of machine learning for NLP generally requires resources for training. Tasks performed in a low-resource language usually rely on labeled data in another, typically resource-rich, language. However, there might not be enough labeled data even in a resource-rich language such as English. In such cases, one approach is to use a hand-crafted approach that utilizes only a small bilingual dictionary with minimal manual verification to create distantly supervised data. Another is to explore typical machine learning techniques, for example adversarial training of bilingual word representations. We find that in event-type detection task—the task to classify [parts of] documents into a fixed set of labels—they give about the same performance. We explore ways in which the two methods can be complementary and also see how to best utilize a limited budget for manual annotation to maximize performance gain.

1 Introduction

For most languages of the world, few or no language processing tools or resources exist (Baumann and Pierrehumbert, 2014). This hinders efforts to apply certain language technologies enjoyed by languages like English, in which much current research is done.

To perform natural language processing tasks in resource-poor languages, one way to overcome data scarcity is to tap on resources from another resource-rich language. Assuming that there are already good resources and tools to solve the same tasks in the more resource-rich language (henceforth, *auxiliary language*), the only remaining challenge is to transfer the learning process into the resource-poor language (henceforth, *target language*) and adapt it to the specifics of that language. One way to do this is to build a shared word representation across the two languages and train an NL engine on this shared representation, perhaps using an adversarial domain adaptation approach to handle the domain (language) shift (Chen et al., 2017). Usually, these approaches assume the availability of large labeled data in the auxiliary language, on the order of hundred thousands to millions.

However, for some more complex or specialized tasks, there might not be enough available training data even in a resource-rich language such as English. A case in point is the event-type classification task over the publicly available datasets, such as ACE 2005[1] and TAC KBP [2] datasets, which usually contain only a few hundred to a few thousand documents. The situation frame (SF) detection task is one example of event-type classification task, where the objective is to extract from each document one or more *situation frames* with their corresponding arguments. A situation frame (SF) is either an *issue* being described in the articles, such as civil unrest or terrorism, or a *need* situation such as the need for water or medical aid. In our task there are 11 situation frame types, each associated with a set of arguments, namely the location, status, relief, and urgency. For example, an article titled "Millions of people are at the risk of starvation due to the food shortage in South Sudan", with content describing the details and the cause of the food shortage, including a mention of difficulty accessing certain regions, can be classified as describing a *food need* and an *infrastructure need*.

[1] https://catalog.ldc.upenn.edu/ldc2006t06

[2] https://tac.nist.gov/2016/KBP/

Proceedings of the 27th International Conference on Computational Linguistics, pages 70–82
Santa Fe, New Mexico, USA, August 20-26, 2018.

As described below, we have tried two approaches: (1) a simple keyword-matching system utilizing only a small bilingual dictionary and minimal manual verification and (2) a sophisticated neural adversarial network that learns bilingual word representations for cross-lingual transfer. We find that the methods have similar performance. We therefore explore ways in which few keyword-based models can create additional, distantly supervised data to improve the performance of a neural cross-lingual event type detection system. Our contributions are: (1) an evaluation of a state-of-the-art method in a different task showing its similar result against a simple baseline, (2) ways to improve performance of such models, and (3) an analysis of the result, with insights to practitioners as to where to focus the available, yet limited, budget for manual annotation work.

This paper is organized as follows: we first describe the related work on cross-lingual NLP tasks in low-resource settings, specifically how the available resources are used. Based on previous work, we then apply our proposed training data augmentation methods and run experiments to show the effectiveness of our methods. We then analyze the results, and follow with a few suggestions on how to best utilize the available annotation effort for maximum gain.

2 Related Work

Keyword-based Models A keyword-based heuristic model is a simple yet effective approach to extract specific information such as events (Keane et al., 2015), because keywords often indicate a strong presence of important information contained in documents (Marujo et al., 2015). Such methods have been used in different tasks like text categorization (Özgür et al., 2005) and information retrieval (Marujo et al., 2013) to extract the required information. Keyword heuristics have also been used to overcome language and domain barriers using bilingual dictionaries (Szarvas, 2008; Tran et al., 2013). However, a weak bilingual dictionary could result in low coverage with this method. Hence, to overcome the limiting bilingual dictionary people employ bootstrapping methods to improve the coverage (Knopp, 2011; Ebrahimi et al., 2016).

Cross-Lingual Text Classification Cross-lingual event type detection is closely related to cross-lingual text classification (CLTC), which aims to classify text in a target language using training data from an auxiliary language (Bel et al., 2003).

To bridge the language gap, early approaches of CLTC relied on a comprehensive bilingual dictionary to translate documents between languages (Bel et al., 2003; Shi et al., 2010; Mihalcea et al., 2007). However, in resource-poor languages, bilingual dictionaries may be small and sparse. Therefore, the performance of direct word translation will be unsatisfactory. Some researchers utilized the bilingual dictionary to translate the models instead (Xu et al., 2016; Littell et al., 2017).

Another line of work focuses on the use of automatic machine translation as an oracle. The various learning algorithms treated the translations as a second view of document and facilitate cross-lingual learning with co-training (Wan, 2009), majority learning (Amini et al., 2009), matrix completion (Xiao and Guo, 2013) and multi-view co-regularization (Guo and Xiao, 2012a).

Instead of word-level or sentence-level translation, various other approaches seek some cross-lingual mapping between document representation (Littman et al., 1998; Vinokourov et al., 2002; Platt et al., 2010; Jagarlamudi et al., 2011; De Smet et al., 2011; Guo and Xiao, 2012b; Zhou et al., 2015; Zhou et al., 2016a; Zhou et al., 2016b; Chen et al., 2017) or label distribution (Xu and Yang, 2017).

Bilingual Word Embedding The most recent method for sharing document representation between languages is bilingual word embedding (Mikolov et al., 2013a; Faruqui and Dyer, 2014; Luong et al., 2015). The goal is to learn a shared embedding space between words in two languages. With the shared embedding, we are able to project all documents into a shared space. The model trained in one language can, therefore, be used in the other language.

3 Models

To see how well recent state-of-the-art methods for CLTC work in our task, we implemented a convolutional neural classifier. We compare this against a simple keyword-based method.

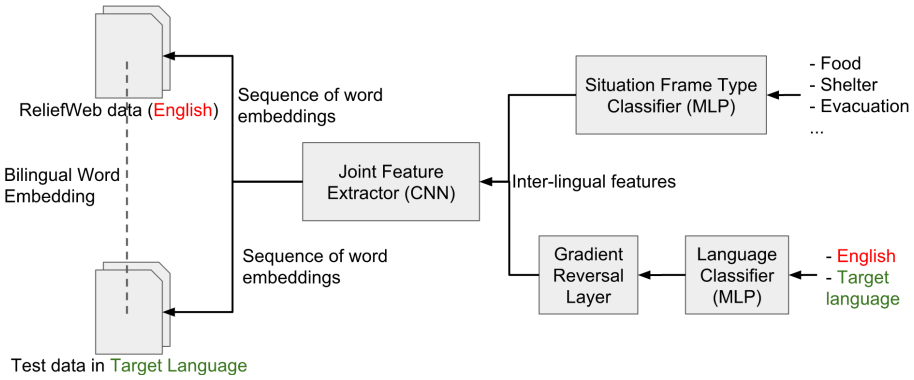


Figure 1: Architecture of the neural classifier with adversarial domain (language) adaptation by Ganin and Lempitsky (2015). Arrows show the flow of gradient.

3.1 Adversarial Convolutional Network

The first step is to train a bilingual word embedding as a shared feature representation space between the two languages. We trained our bilingual word embedding for English and the incident language using the method proposed in XlingualEmb (Duong et al., 2016). This method is a cross-lingual extension from word2vec model (Mikolov et al., 2013b) to bilingual text using two large monolingual corpora and a bilingual dictionary.

Based on the shared representation, we then used a convolutional neural network (CNN) (Kim, 2014) to perform the classification. There are two main advantages of choosing a deep neural classifier over a shallow one. First, CNN outperforms shallow models like SVM or Logistic Regression in various text classification benchmark datasets (Kim, 2014; Lai et al., 2015; Johnson and Zhang, 2015; Xu and Yang, 2017). Second, CNN takes dense word vector representations as input, allowing one to incorporate the state-of-the-art bilingual word embedding methods into the pipeline.

The CNN model takes a sequence of word embeddings as input and applies 1-D convolutional operation on the input to extract semantic features. The features are then passed through a fully-connected layer before reaching the final soft-max layer. The model is trained in English using the ReliefWeb dataset (Littell et al., 2017, Sec 2.3), which is annotated at sentence level with disaster relief needs and emergency situations. Thanks to the bilingual word embedding, which maps the words from the two languages to the same distributional semantic space, the model trained in English can be applied to documents in the target language.

Ideally, if the bilingual word embedding captures the ground-truth mapping between two languages, a classifier learned from English training documents should generalize well on the target language. However, in practice, we can observe obvious domain gaps between documents in different languages when we represent them with bilingual word embeddings. In order to close the domain (language) gap between training and testing, we adapt our learned model in English to the target language with similar adversarial training techniques used in (Xu and Yang, 2017; Chen et al., 2017). In order to alleviate the domain mismatch, we are essentially looking for a feature extractor that only captures the semantics of the event types but not the difference in language usage between English and the target language. In other words, we want the features captured by CNN to be informative for the event type classification and to be language-invariant at the same time. To achieve this goal, we include an auxiliary classifier that takes the features extracted by CNN and predicts the language that the input belongs to. During training, we update our parameter to simultaneously minimize the loss of the event type classification and maximize the loss of language classification through Gradient Reversal Layer (Ganin and Lempitsky, 2015).

3.2 Keyword-based Model

As mentioned previously, a keyword-based model is a simple, quick, yet effective approach to perform text classification without much training data. In our case, we do this in two steps: (1) build a list of keywords for each SF type in English, then (2) translate the keywords into the target language automatically

	Instances	Distribution of Situation Frames (%)												Visualization
		terror	violence	regime	food	water	med	infra	shelter	evac	utils	search	none	
Eng-Orig (Ⓞ)	82,096	2.4	1.8	3.9	14.0	33.0	8.2	6.0	9.2	3.7	4.3	8.6	4.9	
Eng-KW (Ⓔ)	1,356,425	17.5	29.2	6.0	11.3	12.6	9.9	2.7	4.0	3.6	1.6	1.5	0.3	
Tigrinya														
Tgt-Boot (Ⓣ)	98	12.2	33.7	10.2	4.1	6.1	20.4	0.0	11.2	1.0	0.0	1.0	0.0	
Tgt-Ann (Ⓐ)	1,012	6.7	14.3	17.6	2.0	0.8	6.6	2.2	3.3	2.4	0.9	2.7	40.6	
Test Data	2,991*	3.2	6.2	0.7	2.5	0.9	3.2	0.4	0.3	0.8	0.3	0.8	80.7	
Oromo														
Tgt-Boot (Ⓣ)	92	3.3	13.0	25.0	21.7	8.7	10.9	1.1	6.5	3.3	1.1	5.4	0.0	
Tgt-Ann (Ⓐ)	652	3.2	4.0	3.5	0.8	0.2	0.6	0.2	0.6	0.0	0.0	0.5	86.5	
Test Data	2,810*	0.6	11.4	2.3	1.4	0.5	2.1	1.7	1.2	0.7	0.5	1.5	76.2	

Table 1: Statistics of the various sources of training data. Eng-Orig and Eng-KW refer to training data in English described in Littell et al. (2017, Sec 2.3) and from our English keyword model's output on ReliefWeb corpus, respectively, while Tgt-Boot and Tgt-Ann refer to training data in the target language obtained from bootstrapped keyword-spotting and from annotation, respectively. The "none" class signifies negative examples in the data. The last column shows a visualization of the SF types, excluding "none". Note: for Test Data, the instances refer to documents, while for the rest, instances refer to sentences.

using a bilingual dictionary. We also asked native speakers of the target language to refine the translation, especially for domain-specific keywords which are not adequately captured by the bilingual dictionary.[3]

Building English keywords is again a two-step process. First, we use the ReliefWeb dataset to generate a list of 100 candidate keywords for each class by taking the top-100 words with the highest TF.IDF scores. Similar to the keyword generation method described by Littell et al. (2017), we manually refined the keyword list by pruning based on world knowledge. For each candidate keyword, we added 30 most similar words using the English word2vec model trained on the Google News corpus[4]. We retained only the words that have cosine-similarity greater than 70%. For each candidate keyword in this extended list, we computed a label affinity score with each SF class label (e.g., *water*, *evacuation*) using cosine-similarity between their word2vec embeddings. Candidate keywords with similarity above a certain threshold th_1 were retained and used as keywords for the corresponding classes[5].

4 Method: Training Data Augmentation

Chen et al. (2017) assumed the auxiliary language contains a large amount of labeled data for the task, about 700k Yelp reviews. For our case, the original training data, which was built semi-automatically by Littell et al. (2017, Sec 2.3), contains only about 80k sentences (Table 1, first row). To improve the performance of the neural model, therefore, we propose to utilize the keyword-based system to automatically augment the original training data. We also explore using additional training data obtained via manual annotation for comparison.

Figure 2 is a summary of the various training data sources we compare in this paper: the original training data (Ⓞ), keyword-spotting in the auxiliary language (Ⓔ), keyword-spotting with bootstrapping in the target language (Ⓣ), and annotated data in the target language (Ⓐ). The additional training data from keyword-spotting in English (Ⓔ) can be directly obtained by using the keyword list in English that we used for the keyword model (Section 3.2) to label a larger ReliefWeb corpus. We describe the other two ways (Ⓣ and Ⓐ) to obtain additional training data in the following sections.

[3]We showed the native speakers translation pairs obtained through the bilingual dictionary, and asked them to verify the translation as acceptable, or to supply a better translation.

[4]https://code.google.com/archive/p/word2vec/

[5]Threshold $th_1 = 0.9$ was determined by a grid search on a held-out English dataset.

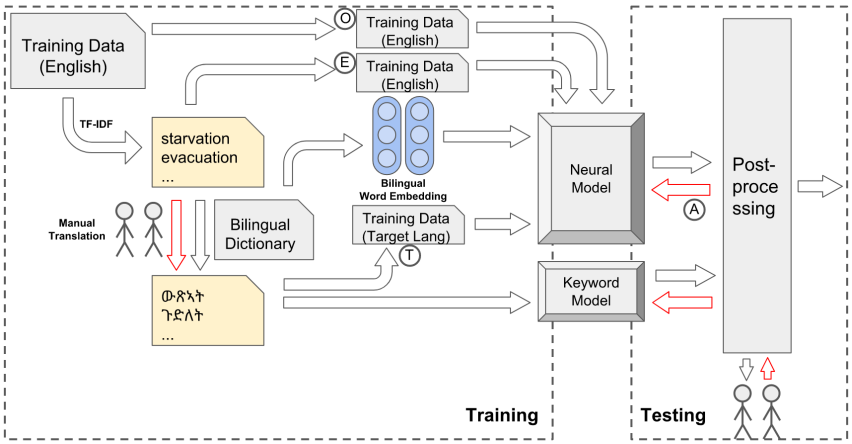


Figure 2: A summary of the various training data sources that we compare in this paper (Ⓞ, Ⓔ, Ⓣ, Ⓐ).

4.1 Bootstrapping Language-specific Keywords

We note that using simple keyword matching can result in low coverage due to missing keywords in the bilingual dictionary or word-variations in the target language. To overcome this, we developed an iterative bootstrapping algorithm that takes into account the newly labeled documents from keyword-spotting and generates additional language-specific keywords in a two-step process (Ⓣ in Figure 2).

Clustering: In the first step, we collected labeled documents from each class, and generated clusters for them ($D = \{D_{c_1},..,D_{c_m}\}$, where D_{c_p} is the cluster of the class c_p). For each cluster D_{c_p} and non-keyword w_i in it, we then computed the label affinity score $S_p(w_i)$, defined as follows:

$$S_p(w_i) = tfidf(w_i) + \frac{\sum_{w_j \in W_p} cos(w_i, w_j)}{|W_p|}$$

where W_p was the set of keywords present in D_{c_p}. We then appended the words which exceed a certain threshold th_2 to the keywords list of class c_p.

The rationale for this step is that the keywords that were missed in the initial keyword list (due to an incomplete bilingual dictionary, the keywords being language-specific, or incident-specific) may appear more frequently in the document cluster, and the second term in $S_p(w_i)$ will capture this.

Labeling: With the updated set of keywords for each class, we relabeled the documents to obtain a new set of labeled documents and again executed the clustering step to get more keywords. We can repeat this two-step process n times until we have the desired coverage or until this process no longer gives useful new keywords. In our experiments, we found that $n = 10$ generally gives good coverage. To generate the training data, we ran this procedure on the test set and took the top-100 most confident predictions.[6]

4.2 Annotation in Target Language

When we have the budget and annotators to do so, we can also annotate documents in the target language with class labels of interest. Given the limited budget and the rarity of documents with SFs (14-18% in our dataset), however, one question remains: how to best pick the documents to be annotated to maximize the gain from the additional training data? Seeing that the number of documents with at least one positive class is much less common compared to the number of documents without any positive class (see Table 1), simply taking a randomly sampled document from the unlabeled documents will likely give a document with no class, which is less useful compared to document with at least one positive class. Thus, we opt for a simpler method of asking annotators to make a binary decision on a

[6] As explained below, we used sentences as our training data, by taking the sentences which contain the keywords found.

	Tigrinya	Oromo
#Documents	2,991 (100%)	2,810 (100%)
– single sentence	2,508 (83.9%)	2,432 (86.6%)
– with 0 SFs	2,565 (85.7%)	2,307 (82.1%)
– with 1 SF	295 (9.9%)	361 (12.9%)
– with 2 SFs	95 (3.2%)	99 (3.5%)
– with 3 SFs	26 (0.9%)	26 (0.9%)
#Sentences	9,412	11,905
#SFs	612	721

Table 2: Data statistics for Situation Frame (SF) type extraction task in Tigrinya and Oromo dataset.

subset of our model's output on a separate dataset, different from the test set. We obtained 653 annotated sentences in Tigrinya this way (and 652 in Oromo). In addition to the native speakers, we also had non-speaker linguists annotate another separate (359) sentences in Tigrinya, assisted by grapheme-to-phoneme conversion, morphological glossing, and machine translation (MT) output.[7] This results in 1,012 sentences annotated in Tigrinya (Ⓐ in Figure 2 and Table 1). Overall, we spent less than 12 man-hours with native speakers of the target language to do the keyword translation and the annotation, with the larger amount of time spent on keyword translation.

5 Experiments

5.1 Dataset

For the purpose of the experiments and analysis, we used the dataset from the LoReHLT 2017[8] shared task, which consists of news articles in two Ethiopian languages: Tigrinya and Oromo.[9] The statistics of the dataset is shown in Table 2.

The available resources that we used for this experiment consist of:

1. Monolingual articles in Tigrinya and Oromo in various genres (news, discussion, social media).
2. Bilingual word dictionary (English-Tigrinya and English-Oromo).
3. A few hours of interaction with volunteers who are native speakers of Tigrinya or Oromo.
4. English documents about disaster recovery from ReliefWeb[10] and CrisisNet[11] annotated semi-automatically with disaster type and theme (Littell et al., 2017, Sec 2.3).[12]

5.2 Setup

We summarize more details about the experiment setup.

Sentence-level prediction: Although the model we used can be applied to produce document-level predictions directly, working at sentence-level provided more training data for the model and made it easier to train. Doing so also enabled some insight on which sentences contain the information about the document-level predictions.

Document-level aggregation: We then aggregate sentence-level predictions to a document-level prediction by assigning to each SF type the maximum confidence score of that type across all sentences in the document. Based on these scores, we calculate the mean confidence score μ_{c_p} of each SF type c_p. We then took the top-k ($k = 3$ in our experiments) SF types as our document-level prediction and filter out the predicted SF types which have confidence scores below μ_{c_p}. In the absence

[7] The MT model was also trained in a low-resource setting, with BLEU score around 12 for Tigrinya.

[8] https://www.nist.gov/itl/iad/mig/lorehlt-evaluations#lorehlt17

[9] For Oromo, the original dataset includes one annotator (out of 4) which annotated most of the documents with a single class. We did not consider this outlier annotator in our experiments.

[10] https://reliefweb.int/

[11] http://http://crisis.net/

[12] Also available at http://dx.doi.org/10.7910/DVN/TGOPRU

	Tigrinya			Oromo		
	P	R	F	P	R	F
KW	48.72	55.63	51.95	14.83	24.40	18.45
KW (bootstrap)	45.90	60.71	52.28	13.35	**44.56**	20.55
NN (Ⓞ)	50.30	58.38	54.04	9.09	18.24	12.13
NN (Ⓞ+Ⓔ)	**56.53**	65.86	60.84	13.65	22.10	16.87
NN (Ⓞ+Ⓐ)	53.32	67.67	59.64	14.82	23.79	18.27
NN (Ⓞ+Ⓔ+Ⓐ)	55.40	65.69	60.11	25.76	28.62	**27.12**
NN (Ⓞ+Ⓣ)	48.01	65.42	56.46	17.45	14.76	16.00
NN (Ⓞ+Ⓔ+Ⓣ+Ⓐ)	55.39	**70.25**	**61.94**	**32.80**	14.07	19.70

Table 3: Performance of the neural model (NN) with various sources of training data. Ⓞ is the original training data, Ⓔ is the additional training data in English from keyword-spotting, Ⓣ is the additional training data in target language from bootstrapping, and Ⓐ is the additional training data in target language from annotation. The result on keyword model (KW) is also shown for comparison.

of labeled data in the target language to be used as development set, this is one method that we can use without much tuning. In later sections we show how different document-level aggregation procedures may affect the performance.

Metric: We followed the metric defined in LoReHLT 2017 guidelines[13], which is *occurrence-weighted scores*, defined as follows:

$$P_{occ} = \frac{\sum \alpha_{tp}}{\sum \alpha_{tp} + \sum \alpha_{fp}}, \qquad R_{occ} = \frac{\sum \alpha_{tp}}{\sum \alpha_{tp} + \sum \alpha_{fn}}, \qquad F_{occ} = \frac{2 \cdot P_{occ} \cdot R_{occ}}{P_{occ} + R_{occ}}$$

where $\sum \alpha_{tp}$ is the number of true positives, weighted with the number of annotators that agree with it. $\sum \alpha_{fp}$ and $\sum \alpha_{fn}$ are similarly defined for false positives and false negatives. False negatives always have weight 1. For brevity, we drop the occ subscript when referring to these scores.

Model hyperparameters: In our neural CNN model, we used filter lengths of $\{3, 4, 5\}$ and 300 filters for each length. We also applied dropout on the extracted feature by CNN at a rate of 0.2. The model was optimized in mini-batches of size 128 by Adam (Kingma and Ba, 2014) optimizer at the learning rate of 0.001. The optimization was terminated after 30 epochs or a convergence criteria was satisfied on the held-out training data.

5.3 Results

Table 3 shows the results in Tigrinya and Oromo with the varying training data described in Section 4.

First, the keyword model (KW) gave results comparable to the neural model (NN Ⓞ), even outperforming it in Oromo. This suggests that in a low-resource setting, a keyword-based model can be used as a way to quickly get a working classifier, without the hassle of training a machine learning classifier or getting a large additional training data.

Next, the additional training data did help to significantly improve the performance of the baseline neural model in both languages. The large additional English data (+Ⓔ) provided a large boost both in Tigrinya (+6.8) and Oromo (+4.7). Interestingly, with only about 900 examples in the target language, the additional annotation in the target language (+Ⓐ) gave about the same improvements in F_1-score in Tigrinya, and even 1.4 points higher in Oromo compared to the large additional training data in English. Recall that the annotation was done on a subset of the neural model's output (trained on Ⓞ). This suggests that when an annotation budget is available, using that to verify the output of a model is a good investment.

It is interesting to note that each source of additional training data improved a different aspect of the model. The additional training data in English (Ⓞ+Ⓔ) seemed to improve precision more, while the

[13] https://goo.gl/FwRCwj

additional training data in target language from annotation seemed to improve recall more (Ⓞ+Ⓐ), and combining both (Ⓞ+Ⓔ+Ⓐ) provided the best of both worlds, especially in Oromo.

When we included the training data in target language from the keyword model with bootstrapping together with all other training data (Ⓞ+Ⓔ+Ⓣ+Ⓐ), it further improved the result in Tigrinya, but not in Oromo, although when it was used alone (Ⓞ+Ⓣ) it still gave some improvements. This could be due to the lower quality of the keyword system in Oromo. Recall that it was created by taking the top-100 most confident predictions of the keyword model. This set of predictions gave 75.9% precision in Tigrinya and 47.1% precision in Oromo. This lower quality of Oromo bootstrapping method can also be seen in the diverging SF Type distribution, as can be seen in Table 1.

The best overall improvement was more pronounced in Oromo (+14.99 points in F_1 for Ⓞ+Ⓔ+Ⓐ) than in Tigrinya (+7.90 points in F_1 for Ⓞ+Ⓔ+Ⓣ+Ⓐ). This could be related to the fact that the baseline score was much lower in Oromo than in Tigrinya to begin with.

For completeness, we also compare the results of the keyword model (KW) in target language without and with bootstrapping in the first two rows of Table 3. As anticipated, the bootstrapping process increased recall significantly, almost doubling the recall in Oromo. Although the precision was slightly reduced, it still resulted in an overall improvement in F_1-score for both languages.

In summary, there are four main observations:

1. The Neural model (NN Ⓞ) gave similar performance to the keyword (KW) model in Tigrinya and lower performance in Oromo, although the keyword model was a much simpler system.
2. With large additional training data in English (+Ⓔ), we obtained large improvements both in Tigrinya (+6.8) and Oromo (+4.7).
3. With only small additional annotations in target language (+Ⓐ) we obtained similar performance to using large English training data in Tigrinya, and even better in Oromo.
4. Getting additional training data in the target language through the keyword model can help if the quality of the keyword model is good enough.

6 Discussion

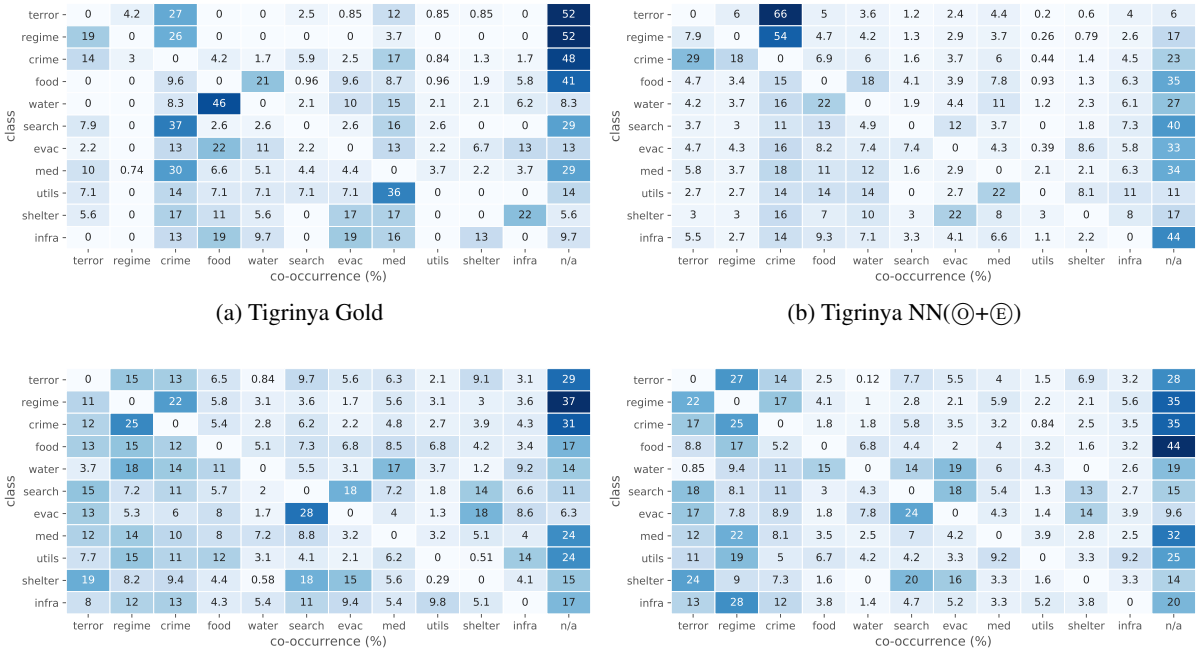
We hypothesize that the focused improvements on precision when using additional training data in English (Ⓔ) could be attributed to the similarity of the SF distribution to the original training data, since both are in English. This causes the model to be more confident in its prediction, at the expense of diverging away from the true distribution of SF types in the target language. In contrast, the annotated dataset in the target language (Ⓐ) has similar distribution to the true distribution, making the model able to rank correct SF types higher. We can see this from the SF type distribution shown in Table 1 by comparing the visualization at the last column.

Another explanation for the higher recall when adding annotated data in the target language is word coverage. The other two additional sources of data rely on keywords, and although bootstrapping helps improve coverage, the annotated data in the target language will cover more subtle correlations between word forms and class labels.

To analyze the differences between the various source of additional training data, we plot the co-occurrence of classes on the Tigrinya dataset in Figure 3. Each row describes the percentage of a particular SF type co-occurring with other SF types in the same document (recall that each document might be labeled with multiple SF types), including none, in which the SF type is the single label for that document. The numbers in a row sums to unity.

As can be seen, predictions of the NN system trained on the additional English data (Figure 3b) and target language data (Figure 3c) have different co-occurrence patterns. The additional English data apparently allowed the NN to find a strong correlation between the *crime violence* class and the *terrorism* and *regime change* classes, which is consistent with our intuition. On the other hand, the NN fine-tuned on the Tigrinya annotations apparently found the *terrorism* and *regime change* classes tend to occur alone.

There is also an interesting phenomenon that arises from the correlation between keywords and class labels. We found that the SF type *terrorism* is associated with the keyword "መንእሰይ" which means



(a) Tigrinya Gold

(b) Tigrinya NN(Ⓞ+Ⓔ)

(c) Tigrinya NN(Ⓞ+Ⓐ)

(d) Tigrinya NN(Ⓞ+Ⓔ+Ⓐ)

Figure 3: Co-occurrence of classes.

	Method	Tigrinya P	Tigrinya R	Tigrinya F	Oromo P	Oromo R	Oromo F
NN (Ⓞ+Ⓔ+Ⓐ)	top-1	68.77	52.45	59.51	36.03	19.93	25.66
NN (Ⓞ+Ⓔ+Ⓐ)	top-2	61.05	62.94	61.98	27.15	24.76	25.90
NN (Ⓞ+Ⓔ+Ⓐ)	top-3	55.40	65.69	60.11	25.76	28.62	27.12
NN (Ⓞ+Ⓔ+Ⓐ)	mean	42.11	71.54	53.01	17.74	36.71	23.92
NN (Ⓞ+Ⓔ+Ⓐ)	Tuned on ref.	80.00	55.03	65.21	36.43	44.08	39.89

Table 4: Impact of various aggregation strategy to the performance.

"youth" or "juvenile", as in the example sentence (English translation, the word recognized as keyword in the original language underlined) "According to the information, the Eritrean girls killed in the incident hide near the tyre of a car and was hot by the Sudanese soldiers." Examining the various examples in the dataset, we found that the violence inherent in terrorism is often depicted with youths as the victims. This could be related to the tendency of news outlets to focus on the suffering experienced by young people to make more emotional appeal.

6.1 Impact of Document-level Aggregation Strategy

In Section 5.2 we showed one heuristic to do document-level aggregation. One might wonder whether one can do better in the classification performance by using another aggregation strategy, such as filtering out classes with confidence scores under certain threshold, or using different k when taking top-k classes. In this section, we explore the impact of different aggregation strategies on the performance under different conditions. Assuming the more realistic case of having no development set to prefer one strategy over another, we can use the top-k strategy like we did in our experiments, or set a fixed threshold on the confidence score based on the average confidence score of each type across all documents in the test set. We also show the result when we set the threshold based on the reference annotation, to show how well the result can be in the case that we have a development set to find the best threshold. The result is shown in Table 4.

The significant gap of performance between the one tuned on the reference annotation and the rest suggests that if additional training data can be obtained in the target language, independently from the model's predictions, we should allocate a portion of them to be used for validation, since there are still large room for improvements just from tuning the thresholds (3-4% in Tigrinya and over 12% in Oromo). Note that in our experiments, since the annotation was done on the output of our neural model, we cannot use them as validation set, as it is biased towards the output of our model. So there is a trade-off between ease of annotation process and the amount of data that can be used as validation set.

7 Conclusion and Future Work

In this paper we tackled the problem of event type detection and classification in low-resource setting. We found that a neural model with adversarial training compared about the same as a simple keyword-based model using a small bilingual dictionary. Given that the problem lies with the limited amount of training data available, we proposed and compared methods to increase the amount of training data: to get significant gain in performance one can either use a very large additional semi-automatically labeled dataset in a resource-rich language, or annotate a small amount of documents in the target resource-poor language. We also showed how investing in a development set for tuning might also be a good strategy when there is a limited budget for annotation, after allocating some of them for keyword translation and additional training data.

One possible direction for future work is to address the mismatch of distribution of the classes between the additional training data and the actual test data as we see in Oromo. One way to mitigate the mismatch would be to make the classifier itself less prone to overfitting. In Section 6.1 we have shown how the document-level aggregation strategy may significantly affect the final result. Thus, exploring ways to effectively select the thresholds might be worthwhile. We could also incorporate the correlation between classes as evident from Figure 3, which has proven useful in multi-label classification (Zhang and Zhang, 2010).

Acknowledgments

We acknowledge NIST for coordinating the SF type evaluation and providing the test data. NIST serves to coordinate the evaluations in order to support research and to help advance the state-of-the-art. NIST evaluations are not viewed as a competition, and such results reported by NIST are not to be construed, or represented, as endorsements of any participants system, or as official findings on the part of NIST or the U.S. Government. We thank Lori Levin for the inputs for an earlier version of this paper. This project was sponsored by the Defense Advanced Research Projects Agency (DARPA) Information Innovation Office (I2O), program: Low Resource Languages for Emergent Incidents (LORELEI), issued by DARPA/I2O under Contract No. HR0011-15-C-0114.

References

Massih Amini, Nicolas Usunier, and Cyril Goutte. 2009. Learning from multiple partially observed views-an application to multilingual text categorization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 28–36, Vancouver, British Columbia, Canada, December.

Peter Baumann and Janet Pierrehumbert. 2014. Using resource-rich languages to improve morphological analysis of under-resourced languages. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*, pages 3355–3359, Reykjavik, Iceland, May.

Nuria Bel, Cornelis H A Koster, and Marta Villegas. 2003. Cross-lingual text categorization. *The 7th European Conference on Research and Advanced Technology for Digital Libraries*, 2769:126–139.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2017. Adversarial deep averaging networks for cross-lingual sentiment classification. *ArXiv e-prints*.

Wim De Smet, Jie Tang, and Marie-Francine Moens. 2011. Knowledge transfer across multilingual corpora via latent topics. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 549–560, Shenzhen, China, May. Springer.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2016. Learning crosslingual word embeddings without bilingual corpora. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1285–1295, Austin, Texas, USA, November. Association for Computational Linguistics (ACL).

Javid Ebrahimi, Dejing Dou, and Daniel Lowd. 2016. Weakly supervised tweet stance classification by relational bootstrapping. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1012–1017, Texas, USA, November. Association for Computational Linguistics (ACL).

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 462–471, Gothenburg, Sweden, April. Association for Computational Linguistics (ACL).

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

Yuhong Guo and Min Xiao. 2012a. Cross language text classification via subspace co-regularized multi-view learning. *arXiv preprint arXiv:1206.6481*.

Yuhong Guo and Min Xiao. 2012b. Transductive representation learning for cross-lingual text classification. In *2012 IEEE 12th International Conference on Data Mining (ICDM)*, pages 888–893. Institute of Electrical and Electronics Engineers (IEEE).

Jagadeesh Jagarlamudi, Raghavendra Udupa, Hal Daumé III, and Abhijit Bhole. 2011. Improving bilingual projections via sparse covariance matrices. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 930–940, Edinburgh, Scotland, UK. Association for Computational Linguistics (ACL).

Rie Johnson and Tong Zhang. 2015. Semi-supervised convolutional neural networks for text categorization via region embedding. In *Advances in Neural Information Processing Systems (NIPS)*, pages 919–927, Montréal, Canada, December.

Nathan Keane, Connie Yee, and Liang Zhou. 2015. Using topic modeling and similarity thresholds to detect events. In *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 34–42, Denver, Colorado, USA, June. Association for Computational Linguistics (ACL).

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Stroudsburg, PA, USA. Association for Computational Linguistics (ACL).

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Johannes Knopp. 2011. Extending a multilingual lexical resource by bootstrapping named entity classification using wikipedia's category system. In *Proceedings of the Fifth International Workshop On Cross Lingual Information Access*, pages 35–43. Asian Federation of Natural Language Processing (AFNLP).

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, pages 2267–2273, Austin, Texas, USA, January. Association for the Advancement of Artificial Intelligence (AAAI).

Patrick Littell, Tian Tian, Ruochen Xu, Zaid Sheikh, David Mortensen, Lori Levin, Francis Tyers, Hiroaki Hayashi, Graham Horwood, Steve Sloto, Emily Tagtow, Alan Black, Yiming Yang, Teruko Mitamura, and Eduard Hovy. 2017. The ARIEL-CMU situation frame detection pipeline for LoReHLT16: a model translation approach. *Machine Translation*, pages 1–22, October.

Michael L Littman, Susan T Dumais, and Thomas K Landauer. 1998. Automatic cross-language information retrieval using latent semantic indexing. In *Cross-Language Information Retrieval*, pages 51–62. Springer.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159, Berlin, Germany, August. Association of Computational Linguistics (ACL).

Luís Marujo, Miguel Bugalho, João Paulo da Silva Neto, Anatole Gershman, and Jaime Carbonell. 2013. Hourly traffic prediction of news stories. *arXiv preprint arXiv:1306.4608*.

Luis Marujo, Wang Ling, Isabel Trancoso, Chris Dyer, Alan W Black, Anatole Gershman, David Martins de Matos, João Neto, and Jaime Carbonell. 2015. Automatic keyword extraction on twitter. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 637–643, Beijing, China, July. Association for Computational Linguistics (ACL).

Rada Mihalcea, Carmen Banea, and Janyce Wiebe. 2007. Learning multilingual subjective language via cross-lingual projections. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 976–983, Czech Republic, June. Association of Computational Linguistics (ACL).

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119, Lake Tahoe, Nevada, United States, December.

Arzucan Özgür, Levent Özgür, and Tunga Güngör. 2005. Text categorization with class-based and corpus-based keyword selection. In *Proceedings of the 20th International Symposium on Computer and Information Sciences (ISCIS)*, pages 606–615, Istanbul, Turke, October. Springer.

John C Platt, Kristina Toutanova, and Wen-tau Yih. 2010. Translingual document representations from discriminative projections. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 251–261, Cambridge, Massachusetts, USA, October. Association for Computational Linguistics (ACL).

Lei Shi, Rada Mihalcea, and Mingjun Tian. 2010. Cross language text classification by model translation and semi-supervised learning. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1057–1067, Cambridge, Massachusetts, USA, October. Association for Computational Linguistics (ACL).

György Szarvas. 2008. Hedge classification in biomedical texts with a weakly supervised selection of keywords. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 281–289, Columbus, Ohio, June. Association for Computational Linguistics (ACL).

Dang Tran, Cuong Chu, Son Pham, and Minh Nguyen. 2013. Learning based approaches for vietnamese question classification using keywords extraction from the web. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing (IJCNLP)*, pages 740–746, Nagoya, Japan, October. Asian Federation of Natural Language Processing (AFNLP).

Alexei Vinokourov, Nello Cristianini, and John S Shawe-Taylor. 2002. Inferring a semantic representation of text via cross-language correlation analysis. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1473–1480, Vancouver, British Columbia, Canada, December.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pages 235–243, Singapore, August. Association for Computational Linguistics (ACL).

Min Xiao and Yuhong Guo. 2013. A novel two-step method for cross language representation learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1–9, Lake Tahoe, Nevada, United States, December.

Ruochen Xu and Yiming Yang. 2017. Cross-lingual distillation for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1415–1425, Vancouver, British Columbia, Canada, July. Association for Computational Linguistics (ACL).

Ruochen Xu, Yiming Yang, Hanxiao Liu, and Andrew Hsi. 2016. Cross-lingual text classification via model translation with limited dictionaries. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM)*, pages 95–104, Indianapolis, Indiana, USA, October. Association for Computing Machinery (ACM).

Min-Ling Zhang and Kun Zhang. 2010. Multi-label Learning by Exploiting Label Dependency. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 999–1008, Washington DC, USA, July. Association for Computing Machinery (ACM).

Huiwei Zhou, Long Chen, Fulin Shi, and Degen Huang. 2015. Learning bilingual sentiment word embeddings for cross-language sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 430–440, Beijing, China, July. Association for Computational Linguistics (ACL).

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016a. Attention-based LSTM network for cross-lingual sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 247–256, Austin, Texas, USA., November. Association for Computational Linguistics (ACL).

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016b. Cross-lingual sentiment classification with bilingual document representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1403–1412, Berlin, Germany, August. Association for Computational Linguistics (ACL).

Neural Transition-based String Transduction for Limited-Resource Setting in Morphology

Peter Makarov **Simon Clematide**
Institute of Computational Linguistics
University of Zurich, Switzerland
makarov@cl.uzh.ch simon.clematide@cl.uzh.ch

Abstract

We present a neural transition-based model that uses a simple set of edit actions (copy, delete, insert) for morphological transduction tasks such as inflection generation, lemmatization, and reinflection. In a large-scale evaluation on four datasets and dozens of languages, our approach consistently outperforms state-of-the-art systems on low and medium training-set sizes and is competitive in the high-resource setting. Learning to apply a generic copy action enables our approach to generalize quickly from a few data points. We successfully leverage minimum risk training to compensate for the weaknesses of MLE parameter learning and neutralize the negative effects of training a pipeline with a separate character aligner.

1 Introduction

Morphological string transduction involves mapping one word form into another, possibly given a feature specification for the mapping, and comprises such inflectional morphology tasks as reinflection and lemmatization (Figure 1), and related problems such as normalization of historical texts. Traditionally, this task has been solved with weighted finite state transducers (Mohri, 2004; Eisner, 2002, WFST). Recently, it has been approached with neural sequence-to-sequence (seq2seq) methods (Faruqui et al., 2016; Kann and Schütze, 2016), inspired by the advances in neural machine translation (Sutskever et al., 2014; Bahdanau et al., 2014). Albeit offering a general solution to a special case of the string-to-string mapping problem, seq2seq models are highly data-intensive. The long tradition of modeling for morphology offers insights into the specifics of the task, suggesting models that would exploit inductive biases and thereby attain lower sample complexity. Recent works in seq2seq morphology model full input string context and unbounded dependencies in the output, but also propose conditioning generation on the context-enriched representation of one input character at a time (Aharoni and Goldberg, 2017; Yu et al., 2016). This and constraining character alignment to be monotonic bring this line of work close to traditional WFST approaches, which monotonically modify a string by performing local changes.

Figure 1: Morphological inflection generation in German (left). Lemmatization in Irish (right).

Having as our starting point the hard monotonic attention model of Aharoni and Goldberg (2017, HA), our goal is to improve seq2seq morphological processing by explicitly modeling local string edits commonly studied in traditional approaches. Our contributions are as follows:

- First, we explain HA as a neural transition-based system over edit actions. Alternative models are then available, differing in the choice of edit actions. We argue that extending HA with the COPY

Proceedings of the 27th International Conference on Computational Linguistics, pages 83–93
Santa Fe, New Mexico, USA, August 20-26, 2018.

edit action is crucial and supported by the nature of the problem, accounting for large performance gains especially in the low-resource setting.

- Second, trained with the original MLE procedure, HA relies on gold action sequences computed by a separate character aligner. As a result, the overall approach is a pipeline. We propose enabling exploration at training (e.g. via expected risk minimization (MRT) or reinforcement learning-style training), thereby allowing the model to prefer alternative actions that also lead to the correct output sequence and neutralizing negative effects of the pipelined architecture. Additionally, this approach benefits from directly optimizing a sequence-level performance metric.

- Third, we conduct extensive experiments on the morphological inflection generation, reinflection and lemmatization tasks, showing that our approaches come near to or improve on the state-of-the-art results. We make our code and model predictions publicly available.[1]

2 Model Description

In our approach, we seek the most probable sequence of edit actions for a given input string and an optional feature specification for the transduction. Unlike traditional WFST approaches to this problem, we abandon the explicit modeling of all possible edit sequences via latent alignments in favor of a greedy, representationally rich RNN-powered transition-based architecture. When training with the MLE criterion following Aharoni and Goldberg (2017), our overall set-up is a pipeline of a character aligner followed by a greedy neural string transducer. Character alignments generated by the aligner are mapped to gold action sequences, whose conditional likelihood the neural transducer then learns to maximize. Under training with exploration, the neural transducer no longer relies on gold action sequences. Instead, the parameters are adjusted to directly maximize the model's accuracy of producing training-set output sequences.

Let Σ_x, Σ_y, and Σ_a be alphabets of input characters, output characters, and edit actions, respectively. Let $\mathbf{x} = x_1, \ldots, x_n$, $x_i \in \Sigma_x$ denote an input sequence, $\mathbf{y} = y_1, \ldots, y_p$, $y_j \in \Sigma_y$ an output sequence, and $\mathbf{a} = a_1, \ldots, a_m$, $a_t \in \Sigma_a$ an action sequence. Let $\{f_h\}_{h=1}^{H}$ be the set of morpho-syntactic features.

seq2seq state-transition system We build a greedy transition-based string transducer that uses a seq2seq neural network to model arbitrary dependencies in the input sequence, the unbounded action history, and the non-deterministic choice of the next action. The system operates a buffer filled with RNN-encoded input characters, and a decoder RNN, which implements a push-only stack. The configuration of the system is given by the decoder state. Transitions are scored based on the output of the decoder, which takes as input the encoded character from the top of the buffer. Here, we elaborate on the model architecture.

We encode input sequence $\mathbf{x}$ with a bidirectional LSTM (Graves and Schmidhuber, 2005)

$$\mathbf{h}_1, \ldots, \mathbf{h}_n = \text{BiLSTM}(E(x_1), \ldots, E(x_n)), \tag{1}$$

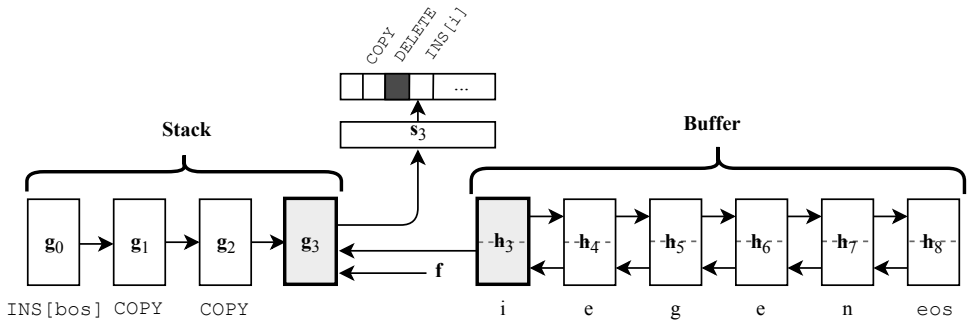
where E returns the embedding for x_i. Vector $\mathbf{h}_i$ is thus the representation of x_i in the context of the entire sequence $\mathbf{x}$. We push $\mathbf{h}_1, \ldots, \mathbf{h}_n$ in reversed order into the buffer. Transduction begins with the full buffer and the empty decoder state.

The decoder LSTM keeps track of the past actions and—through conditioning at each step on $\mathbf{h}_i$—knows of character x_i at the top of the buffer and the full contents of the buffer. From the latest state of the decoder $\mathbf{c}_{t-1}$, we compute the configuration of the system:

$$\mathbf{s}_t = \text{LSTM}(\mathbf{c}_{t-1},\ [A(a_{t-1})\ ;\ \mathbf{h}_i\ ;\ \mathbf{f}]), \tag{2}$$

where the input is the concatenation of (i) the embedding of the previous action (given by A), (ii) $\mathbf{h}_i$ from the top of the buffer indicating the current position in $\mathbf{x}$, and—optionally—(iii) feature vector $\mathbf{f}$,

[1] https://github.com/ZurichNLP/coling2018-neural-transition-based-morphology



	Transition a_t	Output $\mathbf{y}$	Stack	Buffer
0	INSERT(BOS)	[]	[INSERT(BOS)]	[f, l, i, e, g, e, n, EOS]
1	COPY	[f]	[COPY, INSERT(BOS)]	[l, i, e, g, e, n, EOS]
2	COPY	[f, l]	[COPY, COPY, . . .]	[i, e, g, e, n, EOS]
3	DELETE	[f, l]	[DELETE,COPY, . . .]	[e, g, e, n, EOS]
4	DELETE	[f, l]	[DELETE, DELETE, . . .]	[g, e, n, EOS]
5	INSERT(o)	[f, l, o]	[INSERT(o), DELETE, . . .]	[g, e, n, EOS]
6	COPY	[f, l, o, g]	[COPY, INSERT(o), . . .]	[e, n, EOS]
7	DELETE	[f, l, o, g]	[DELETE, COPY, . . .]	[n, EOS]
8	DELETE	[f, l, o, g]	[DELETE, DELETE, . . .]	[EOS]
9	INSERT(EOS)	[f, l, o, g]		

Figure 2: Transduction of "fliegen" to "flog". (Above) Visualization of the system as it chooses $a_3 =$ DELETE. (Below) Full transition sequence. Action a_0 is always fixed to INSERT(BOS).

which is the concatenation of the embedded morpho-syntactic features $\phi \subseteq \{f_h\}_{h=1}^{H}$ associated with this transduction: $\mathbf{f} = [F(f_1)\ ;\ \cdots\ ;\ F(f_H)]$ and $F(f_h) = F(0)$ if $f_h \notin \phi$.

To compute probabilities of transitions $\mathbf{a}_t$, we feed $\mathbf{s}_t$ through a softmax classifier:

$$P(a_t = k \mid \mathbf{a}_{<t}, \mathbf{x}, \Theta) = \text{softmax}_k(\mathbf{W} \cdot \mathbf{s}_t + \mathbf{b}) \tag{3}$$

Model parameters Θ include softmax classifier parameters $\mathbf{W}$ and $\mathbf{b}$, the embedding parameters, and the parameters of the encoder and decoder.

Edit actions Traditional transducers edit input sequence $\mathbf{x}$ into output sequence $\mathbf{y}$ by a sequence of single-character edit actions from the following set (Cotterell et al., 2014):

- DELETE: Read x_i and write nothing.
- INSERT(c) for $c \in \Sigma_y$: Write c and read nothing.
- SUBST(c) for $c \in \Sigma_y$: Read x_i and write c.
- COPY: Read x_i and write x_i.

Let INSERTS$_y$ be the set of all insertions with respect to Σ_y. We consider the following two action alphabets: $\Sigma_a^{HA} = \text{INSERTS}_y \cup \{\text{DELETE}\}$ and $\Sigma_a^{CA} = \Sigma_a^{HA} \cup \{\text{COPY}\}$.

Alphabet Σ_a^{HA} is from Aharoni and Goldberg (2017) and includes only the INSERT and DELETE actions. Both substitution and copying of c are expressed as an INSERT(c) followed by a DELETE.

Alphabet Σ_a^{CA} adds a designated COPY action to Σ_a^{HA}. Thus, copying x_i to the output sequence can be executed by one single action. This results in shorter and simpler action sequences dominated by COPY actions, following the observation that inflectional changes are typically small and most of $\mathbf{x}$ is preserved in $\mathbf{y}$.[2]

Action execution Operationally, reading x_i corresponds to popping its representation $\mathbf{h}_i$ from the top of the buffer. The transducer terminates when the buffer is empty and the latest action a_t is INSERT(EOS), where EOS is the end-of-sequence character. If we constrain the number of successive insertions to at most q, the transducer runs in $O(n)$ time, where n is the length of input $\mathbf{x}$.[3]

[2]We also experimented with extending Σ_a^{HA} and Σ_a^{CA} with actions for character substitutions. The resulting models perform similarly to models without substitutions, and so we do not report them here.

[3]In practice, we simply cap the number of actions at 150.

f	l	i	e	g	e	n
|	|	|	|	|	|	|
f	l	o	g	ϵ	ϵ	ϵ

f	l	i	e	g	e	n
|	|	|	|	|	|	|
f	l	o	ϵ	g	ϵ	ϵ

Figure 3: Longest Common Substring (LCS, left) and Chinese Restaurant Process (CRP, right) character alignments for the same $\mathbf{x}$ and $\mathbf{y}$. Input sequence $\mathbf{x}$ is at the top, output sequence $\mathbf{y}$ at the bottom. A CRP aligner recovers this alignment given sufficient training data and number of iterations.

MLE training The model is trained to maximize the conditional log-likelihood of the data $D = \{(\mathbf{x}^{(l)}, \mathbf{a}^{(l)})\}_{l=1}^{N}$, which is an everywhere differentiable function of parameters Θ:

$$\mathcal{L}(D, \Theta) = \sum_{l=1}^{N} \sum_{t=1}^{m} \log P(a_t^{(l)} \mid \mathbf{a}_{<t}^{(l)}, \mathbf{x}^{(l)}, \Theta) \tag{4}$$

The gold action sequences $\mathbf{a}^{(l)}$ are computed by a deterministic algorithm from some character alignment: $\mathbf{a}^{(l)} = C_{\Sigma_a}^{Align}(\mathbf{x}^{(l)}, \mathbf{y}^{(l)})$. Figure 3 illustrates different character alignment algorithms that we use in our experiments. A simple procedure for the generation of gold actions from alphabet Σ_a^{CA} would call the following subroutine d on each pair of character alignment $(b_1, c_1), \ldots, (b_r, c_r)$ between input $\mathbf{x}$ and output $\mathbf{y}$, where $b_k \in \Sigma_x \cup \{\epsilon\}$ and $c_k \in \Sigma_y \cup \{\epsilon\}$ but not $b_k = c_k = \epsilon$:

$$d(b, c) = \begin{cases} \text{COPY}, & \text{if } b = c, \\ \text{DELETE}, & \text{if } c = \epsilon, \\ \text{INSERT(c)}, & \text{if } b = \epsilon, \\ \text{DELETE, INSERT(c)} & \text{otherwise} \quad \% \textit{ substitution} \end{cases}$$

Applying this procedure to e.g. the CRP alignment from Figure 3, we obtain the following gold action sequence: COPY, COPY, DELETE, INSERT(o), DELETE, COPY, DELETE, DELETE.

Learning with exploration Training with MLE comes with a number of limitations. First, the model is not exposed to its own errors at training time: It makes predictions conditioned on gold-action histories, which is at odds with test time when the model has to condition on predicted actions. Second, MLE training increases the model's per-action likelihood, although at test time, the model's performance is assessed with sequence-level accuracy or edit distance. Both constitute well-known MLE training biases—the exposure bias and the loss-evaluation mismatch (Wiseman and Rush, 2016). Finally, we would like the model to be less dependent on the gold actions generated by the aligner, which is uninformed of the downstream task, and that at training, the model can choose alternative action sequences leading to correct predictions, if that helps it generalize.

To address all these issues at once, we train the model by minimizing the expected risk (Och, 2003; Smith and Eisner, 2006) of the actual training data $T = \{(\mathbf{x}^{(l)}, \mathbf{y}^{(l)})\}_{l=1}^{N}$:

$$\mathcal{R}(T, \Theta) = \sum_{l=1}^{N} \mathbb{E}_{\mathbf{a}|\mathbf{x}^{(l)};\Theta} \left[\Delta(\mathbf{y}, \mathbf{y}^{(l)}) \right], \tag{5}$$

where $\mathbf{y}$ is computed from $\mathbf{a}$ and $\mathbf{x}$, and the risk is given by a combination of normalized Levenshtein distance (NLD) and accuracy:

$$\Delta(\mathbf{y}, \mathbf{y}^{(l)}) = \text{NLD}(\mathbf{y}, \mathbf{y}^{(l)}) - \mathbb{1}\{\mathbf{y} = \mathbf{y}^{(l)}\} \tag{6}$$

Thus, an action sequence $\mathbf{a}$ attains the lowest risk of -1 if its corresponding output sequence $\mathbf{y}$ is identical to $\mathbf{y}^{(l)}$ of the training sample and the highest risk of $+1$ if the number of edits from $\mathbf{y}$ to $\mathbf{y}^{(l)}$ equals the maximum of their lengths.

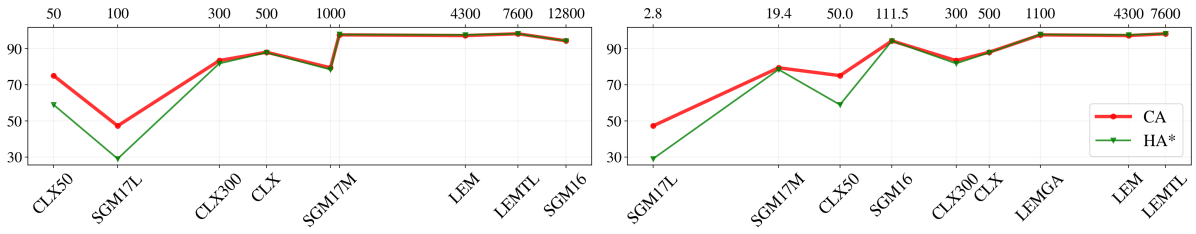


Figure 4: Accuracy as a function of dataset size (left) and the ratio of dataset size to the number of unique transformations (right) for selected experiments. A log scale is used for the X axis. CLX50/CLX300=average scores on CELEX with 50/300 samples (Figure 5), SGM16 =SIGMORPHON2016, SGM17L/SGM17M=SIGMORPHON2017-low/medium, LEM=average scores on lemmatization, LEMGA/LEMTL=lemmatization Irish/Tagalog.

Following Shen et al. (2016), we approximate the expectation under the posterior distribution $P(\mathbf{a} \mid \mathbf{x}^{(l)}; \Theta)$ with ancestral sampling from the model and re-normalize the sampled probability scores to get a new distribution Q:

$$\mathcal{R}(D, \Theta) \approx \sum_{l=1}^{N} \sum_{\mathbf{a} \in S(\mathbf{x}^{(l)})} Q(\mathbf{a} \mid \mathbf{x}^{(l)}; \Theta, \alpha)\ \Delta(\mathbf{y}, \mathbf{y}^{(l)}) \tag{7}$$

$$Q(\mathbf{a} \mid \mathbf{x}^{(l)}; \Theta, \alpha) = \frac{P(\mathbf{a} \mid \mathbf{x}^{(l)}; \Theta)^{\alpha}}{\sum_{\mathbf{a}' \in S(\mathbf{x}^{(l)})} P(\mathbf{a}' \mid \mathbf{x}^{(l)}; \Theta)^{\alpha}} \tag{8}$$

Here, $S(\mathbf{x}^{(l)})$ denotes the set of samples from $P(\mathbf{a} \mid \mathbf{x}^{(l)}; \Theta)$ and $\alpha \in \mathbb{R}$ is a hyper-parameter that controls for the peakedness of the new distribution Q.

3 Experiments

In the following experiments, we evaluate the performance of our model with an explicit copy action (referred to as CA) and show how it further improves with exploration training (-MRT).

Unless stated otherwise, our MLE models are trained on gold actions computed using Mans Hulden's Chinese Restaurant Process string-pair aligner (indicated as CRP)[4] and decoded with beam search. On some problems, we find a simple strategy, which heuristically maximizes the number of COPY actions, to work surprisingly well: The Longest Common Substring aligner (LCS) first aligns the longest common substring of $\mathbf{x}$ and $\mathbf{y}$ and then pads both strings to the same length.

MRT models are initialized with the corresponding MLE models and decoded with beam search. We found the best value of $\alpha = 1$ from $\{1, 0.1, 0.05\}$ on the CELEX-ALL task (§ 3.1) and used that for all other datasets as well.

We use the same embedding parameters for characters and insertion actions (i.e. $A(\text{INSERT}(b)) = E(b)$) to match closely the set-up of Aharoni and Goldberg (2017). In all our systems, the dimension of the character and action embeddings is 100, LSTM hidden layers are of size 200, and all LSTMs are single-layer. We use LSTMs with peephole connections and coupled input and forget gates (Greff et al., 2016). We optimize with ADADELTA (Zeiler, 2012) and update parameters at a single training sample (=batch size 1) during MLE training. For MRT, we build sets $S(\mathbf{x}^{(l)})$ by drawing twenty samples per training example. We mini-batch using these sets as batches. We include the gold action sequence (generated for MLE training) into the batch. We implement our models using DyNet (Neubig et al., 2017).

All experiments report exact accuracies. They are mean accuracies over single runs with different initializations, unless the model is an ensemble (marked with an -E suffix). The ensembles are built with majority voting over differently initialized runs of the same model.

[4] https://github.com/ryancotterell/sigmorphon2016/blob/master/src/baseline/align.c

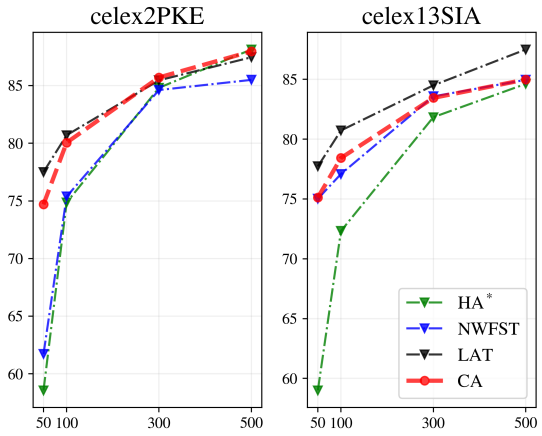


Figure 5: Learning curves on the CELEX dataset.

Model	13SIA	2PIE	2PKE	rP	Avg.
CELEX-BY-TASK					
LAT	**87.5**	93.4	87.4	84.9	88.3
NWFST	85.1	94.4	85.5	83.0	87.0
HA*	84.6	93.9	**88.1**	85.1	87.9
CA	85.0	94.5	88.0	84.9	88.1
HA*-MRT	84.8	94.0	88.1	85.2	88.0
CA-MRT	85.6	**94.6**	88.0	**85.3**	**88.4**
CELEX-ALL (ensembles)					
MED	83.9	95.0	87.6	84.0	87.2
HA	85.8	**95.1**	**89.5**	87.2	89.5
HA*-E	85.3	94.8	88.9	87.4	89.1
CA-E	85.8	94.9	88.8	86.7	89.1
HA*-MRT-E	85.8	95.0	89.2	**87.7**	89.4
CA-MRT-E	**86.7**	94.9	89.3	87.1	**89.5**

Table 1: Results on the CELEX dataset.

We evaluate our approaches on four standard morphological datasets and compare to the following published systems: (HA) the ensemble of five MLE models over Σ_a^{HA} of Aharoni and Goldberg (2017) as well as our re-implementation of a single model marked as HA*; (MED) the ensemble of five soft-attentional models of Kann and Schütze (2016) and an alternative implementation of the soft-attention approach, SOFT, by Aharoni and Goldberg (2017); (NWFST) the neural WFST model of Rastogi et al. (2016) and (LAT) the non-neural WFST with latent variables of Dreyer et al. (2008).

3.1 Morphological reinflection

The task is to map an inflected form $\mathbf{x}$ into another form $\mathbf{y}$ of that word given a feature specification ϕ for this transformation.

CELEX This dataset of German verbal morphology transformations was compiled by Dreyer et al. (2008) from the CELEX database (Baayen et al., 1993). It comprises four transformations (13SIA↦13SKE, 2PIE↦13PKE, 2PKE↦z, rP↦pA),[5] featuring such morphological phenomena as circumfixation, infixation, and irregular stem changes. The data are split into five folds, each with 500 training samples per transformation. We conduct two types of evaluation on these data. In the original experiment, which we call CELEX-BY-TASK, models are trained on each transformation separately, and scores are averaged over the folds. In the second experiment, CELEX-ALL, five single models are trained on all the 2,000 samples of one fold and then ensembled. Again, scores are averaged over the folds. As part of CELEX-BY-TASK, we additionally evaluate how our models perform on even fewer—50, 100, and 300—training samples on two tasks, 2PKE and 13SIA. CELEX could be considered a relatively simple dataset as the ratio of the number of training samples to the number of unique transformations is high, even though the overall training-data size is modest. On the other hand, most CELEX tasks require learning complex lexical properties such as the distinction between strong and weak verbs or prefix types.

3.2 Morphological inflection generation

Given a feature specification ϕ and a base form $\mathbf{x}$, the task is to generate the corresponding inflected form $\mathbf{y}$.

Sigmorphon 2017 The low (100 training samples) and medium (1,000 training samples) settings of the SIGMORPHON 2017 shared task data (Cotterell et al., 2017) feature fifty-two languages. The datasets contain extremely diverse language material and morphological transformations. Unlike CELEX, input $\mathbf{x}$ is always a dictionary form, however morphological changes are unrestricted. The low setting constitutes a very hard learning problem, with the ratio of training samples to unique transformations being 2.8 on

[5]Glossary: 13SIA=1st/3rd person singular indicative past; 13SKE=1st/3rd person singular subjunctive present; 2PIE=2nd person plural indicative present; 13PKE=1st/3rd plural subjunctive present; 2PKE=2nd person plural subjunctive present; z="zu" infinitive; rP=plural imperative; pA=past participle.

Model (averages)		**low**	**medium**	**Model** (ensembles)		**low**	**medium**
baseline		37.9	64.7			37.9	64.7
HA*	lcs	29.1	78.5	HA*-E	lcs	31.5	80.2
CA	lcs	47.3	79.5	CA-E	lcs	48.8	81.0
HA*	crp	23.9	75.4	HA*-E	crp	26.1	77.8
CA	crp	42.5	78.9	CA-E	crp	44.0	80.6
HA*-MRT	lcs	30.2	79.6	HA*-MRT-E	lcs	33.1	81.5
CA-MRT	lcs	**48.1**	80.3	CA-MRT-E	lcs	49.9	81.9
HA*-MRT	crp	25.3	78.1	HA*-MRT-E	crp	28.1	80.5
CA-MRT	crp	43.6	**81.1**	CA-MRT-E	crp	45.7	**82.9**
				HACM-E7		46.8	81.8
				HAEM-E7		48.5	80.3
				HA[EC]M-E15		**50.6**	82.8

Table 2: Results on the SIGMORPHON 2017 dataset.

average (SD $= 2.9$). In the medium setting, the mean number of unique transformations rises to 19.8 (SD $= 29.3$), with a minimum of 1.4 observed for Basque and a maximum of 200 for English.

For this dataset, we also show the results for the official baseline, a ruled-based system that is particularly strong in the low setting,[6] and the best systems of the shared task (Makarov et al., 2017).

Sigmorphon 2016 The SIGMORPHON 2016 shared task dataset (Cotterell et al., 2016) is the largest dataset. It comprises ten languages with about 12,800 training examples on average. The number of samples per transformation varies from 6 for Maltese to 198 for Hungarian, being 112 samples per transformation on average (SD $= 51.3$). In both SIGMORPHONS, we train five single models for each language.

3.3 Lemmatization

Given an inflected word form $\mathbf{x}$ (without any feature specification), the task is to predict the correct dictionary form $\mathbf{y}$. Following Dreyer (2011) and Rastogi et al. (2016), we evaluate our approach on a subset of the dataset by Wicentowski (2002). The data, split into ten folds, comprise four languages, with per-fold training sizes ranging on average from 1,100 for Irish to 7,635 for Tagalog. For each language, we train a separate model for each fold and then average the scores over the folds.

4 Results and Discussion

Generally, comparing the performance of CA and HA (or HA*), we observe that CA achieves great performance gains on small-sized problems while matching HA in the higher-resource setting (Figure 4).

4.1 Morphological reinflection

CA is a very competitive model on both CELEX-BY-TASK and CELEX-ALL, and adding exploration (CA-MRT) results in the strongest performance in both evaluations (Table 1). In contrast to HA*, in very low settings (Figure 5), CA performs not much worse than the only non-neural model, LAT. HA* and NWFST need around 300 training examples to start catching up, and the extremely low-resource conditions (50, 100) on 13SIA are especially troublesome for HA*. On CELEX-ALL, even with more training data, soft-attentional ensemble MED is typically much weaker, including tasks with infixation (2PKE) and circumfixation (rP).

Advancing further on most CELEX tasks is difficult due to morphological irregularities. As an example, examining the predictions of CA-MRT on one fold of the rP task reveals that the system largely fails to predict strong-verb participles (71% of the errors), conjugating 67% of them as if they were regular.

[6] https://github.com/sigmorphon/conll2017/tree/master/baseline

Model	RU	DE	ES	KA	FI	TR	HU	NV	AR	MT	Avg.
HA*	**91.32**	95.91	98.63	97.69	94.75	96.99	98.44	**90.57**	**93.93**	85.28	94.35
CA	90.81	**95.97**	**98.75**	**97.97**	**95.59**	**97.11**	**98.64**	89.74	93.59	**85.77**	**94.39**
ensembles											
MED	91.46	95.80	98.84	98.50	95.47	98.93	96.80	91.48	**99.30**	**88.99**	95.56
SOFT	92.18	96.51	98.88	**98.88**	**96.99**	**99.37**	97.01	**95.41**	**99.30**	88.86	**96.34**
HA	**92.21**	**96.58**	**98.92**	98.12	95.91	97.99	96.25	93.01	98.77	88.32	95.61
HA*-E	91.95	96.28	98.85	97.90	95.78	97.55	98.77	92.14	95.08	87.82	95.21
CA-E	91.87	96.36	98.84	98.35	96.50	97.74	**98.90**	92.14	94.63	87.66	95.30

Table 3: Results on the SIGMORPHON 2016 dataset: ru=Russian, de=German, es=Spanish, ka=Georgian, fi=Finnish, tr=Turkish, hu=Hungarian, nv=Navaho, ar=Arabic, mt=Maltese.

4.2 Morphological inflection generation

Table 2 summarizes the results on the SIGMORPHON 2017 dataset. In the low setting, CA easily beats the baseline system, whereas HA* fails to do so. Our simple majority-vote ensemble CA-MRT-E over five models comes very close to the complex 15-strong ensemble HA[EC]M-E15 of Makarov et al. (2017), the best system of the shared task. Under a paired permutation test, the latter system is statistically significantly better ($p < 0.05$) on only twenty one languages.

In the medium setting, CA maintains the advantage, although the performance gap from HA* is much smaller. CA-MRT-E even outperforms the shared task's best system, although the gain is statistically significant for only ten languages. In both settings, MRT consistently improves the performance of both the HA* and CA models.

In the high-resource scenario of SIGMORPHON 2016 (Table 3), HA* and CA attain virtually identical results, occasionally outperforming the soft-attentional ensembles. Unlike HA, we use the same set of hyper-parameters (the dimension of embeddings, the number of hidden LSTM layers, etc.) for all of our experiments, which might explain that both our reimplementation HA* and CA perform less strongly here. Due to computational restrictions, we could not apply MRT to this dataset.

4.3 Lemmatization

On the lemmatization task (Table 4), CA strongly outperforms WFST models LAT and NWFST on average. Yet, the HA* reimplementation consistently delivers the best results on every language. The error analysis for English in Rastogi et al. (2016) mentions the tendency of their system, NWFST, to simply copy the inflected word over, which accounts for 25% of English-language errors. Given that CA also has a dedicated copy action, one might suspect that the inferior performance of CA compared to HA* for English and Basque would be due to excessive copying. An inspection of the incorrectly predicted lemmas reveals that both systems produce virtually the same number of copy errors. The difference in error counts is actually due to cases where the system modifies the inflected word form. For English, errors typically occur in strong verbs and verbs with graphemic alternations, as e.g. "oozing" gets incorrectly lemmatized as "ooz". The scores of over 97% on every language and the kind of unsolved cases, likely requiring external resources, suggest that this task should be considered solved.

As a final remark, we note that with the datasets at hand, performance attribution is often hampered by the lack of explicit characterization of morphological phenomena or lexical properties at the example level (we have derived some of these meta-data for the CELEX rP task). Given the difficulties interpreting neural models, computational morphology could arguably profit from challenge sets that have recently been gaining popularity in machine translation (Sennrich, 2017; Avramidis et al., 2018).

5 Related Work

Traditional models for morphological string transduction are discriminatively trained WFSTs (Cotterell et al., 2014; Dreyer et al., 2008; Eisner, 2002). The transducer defines eligible edit sequences for $\mathbf{x}$ (each implying a different monotonic character alignment), and its weights are expressed in terms of hand-crafted features. Rastogi et al. (2016) employ RNNs to parametrize the weights of a globally normalized

Model		basque	english	irish	tagalog	Avg.
Size		4.7K	3.9K	1.1K	7.6K	4.3K
LAT		93.6	96.9	**97.9**	88.6	94.2
NWFST		91.5	94.5	**97.9**	97.4	95.3
HA*	lcs	**97.0**	97.5	**97.9**	**98.3**	**97.7**
CA	lcs	96.3	96.9	97.7	**98.3**	97.3
HA*	crp	96.2	**97.7**	97.3	97.9	97.3
CA	crp	96.1	96.7	96.8	97.6	96.8

Table 4: Results on the lemmatization dataset.

WFST, thereby conditioning on global context. The powerful approach of Dreyer et al. (2008) adds latent variables to a globally normalized log-linear WFST to learn task-specific properties: a word's paradigm class and approximate morphological segmentation.

Enabling soft character alignment via a deterministic function of inputs (Kann and Schütze, 2016) has proven crucial to the success of seq2seq models first proposed for this task in Faruqui et al. (2016). In line with the traditional simplification of the task, other neural-network approaches treat hard monotone character alignment as a latent variable that the model marginalizes out using dynamic programming, while enabling unbounded dependencies in the output and permitting online generation (Yu et al., 2016; Graves, 2012). An appealing alternative to latent alignment is to learn from supervised alignment, an idea explored to train soft-attention models (Mi et al., 2016). For hard-attention models (Aharoni and Goldberg, 2017), training with an observed alignment is particularly simple as it results in learning from a single gold action sequence.

A state-transition system is an elegant, linear-time model for morphological string transduction, in which eligible monotonic edit sequences are implied by the semantics of the actions. As demonstrated on other tasks (Dyer et al., 2015; Andor et al., 2016), when provided with global context via RNNs, the model overcomes the limitations of a locally normalized conditional distribution, while retaining computational efficiency.

Using a single designated copy action in not new in morphological string transduction, e.g. the SIGMORPHON 2016 feature-based state-transition baseline uses COPY$[n]$, where n is the number of characters to copy. Biasing towards copy edits is crucial to the performance of the model of Rastogi et al. (2016). An alternative to the copy action is to introduce a binary latent variable that signals whether y_i is copied from x_i or generated (Gu et al., 2016; Gulcehre et al., 2016; See et al., 2017). Extending models with alignment variables with such a copying mechanism is simple as the the choice of which x_i has to be copied need not be modeled (Makarov et al., 2017): The copy variable points to the x_i that y_j is aligned with. This alternative requires learning additional model parameters, which could explain its somewhat worse performance on smaller-sized problems.

Minimum risk training (Smith and Eisner, 2006; Och, 2003) is one simple solution enabling exploration and addressing the loss-evaluation mismatch. The approach of Shen et al. (2016) closely relates to classical policy gradient methods in reinforcement learning (Edunov et al., 2018). A number of alternative methods have recently been proposed to address the MLE training biases in the context of seq2seq models (Andor et al., 2016; Wiseman and Rush, 2016; Ranzato et al., 2016; Rennie et al., 2017).

6 Conclusion

In a large-scale evaluation on different morphological tasks and languages, we show that a neural transition-based system over edit actions consistently outperforms state-of-the-art systems on morphological string transduction tasks in low- and medium-resource settings and is competitive on large training sets. Crucially, adding a designated action to copy the input character over to the output string helps the transition model generalize quickly from very few data points. Using a training procedure that enables exploration of the action space (e.g. minimum risk training) consistently improves the performance of our models as they are exposed to action sequences other than those proposed by the character aligner underlying the static oracle in the MLE training procedure.

Acknowledgements

We would like to thank Tatyana Ruzsics, Tanja Samardžić, Mathias Müller, Roee Aharoni, Pushpendre Rastogi, and the anonymous reviewers. Peter Makarov has been supported by European Research Council Grant No. 338875.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *ACL*.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *ACL*.

Eleftherios Avramidis, Vivien Macketanz, Arle Lommel, and Hans Uszkoreit. 2018. Fine-grained evaluation of quality estimation for machine translation based on a linguistically-motivated test suite. In *Proceedings of the 1st Workshop on Translation Quality Estimation and Automatic Post-Editing*. AMTA.

RH Baayen, R Piepenbrock, and H Van Rijn. 1993. The CELEX lexical database. LDC.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2014. Stochastic contextual edit distance and probabilistic FSTs. In *ACL*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 Shared Task—Morphological Reinflection. In *Proceedings of the 2016 Meeting of SIGMORPHON*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. The CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*. ACL.

Markus Dreyer, Jason R Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *ACL*.

Markus Dreyer. 2011. *A non-parametric model for the discovery of inflectional paradigms from plain text using graphical models over strings*. Ph.D. thesis, The Johns Hopkins University.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *ACL*.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Classical structured prediction losses for sequence to sequence learning. In *NAACL-HLT*.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *ACL*.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *NAACL-HLT*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5).

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

K. Greff, R. K. Srivastava, J. Koutnk, B. R. Steunebrink, and J. Schmidhuber. 2016. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99).

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *CoRR*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *ACL*.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *ACL*.

Peter Makarov, Tatiana Ruzsics, and Simon Clematide. 2017. Align and copy: UZH at SIGMORPHON 2017 shared task for morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*. ACL.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. In *EMNLP*.

Mehryar Mohri. 2004. Weighted finite-state transducer algorithms. an overview. In *Formal Languages and Applications*, volume 148 of *Studies in Fuzziness and Soft Computing*. Springer Berlin Heidelberg.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR*.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *NAACL-HLT*.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *CVPR 2017*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *ACL*.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing mt quality with contrastive translation pairs. In *EACL*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *ACL*.

David A Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *COLING/ACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Richard Wicentowski. 2002. *Modeling and learning multilingual inflectional morphology in a minimally supervised framework*. Ph.D. thesis, Johns Hopkins University.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*.

Lei Yu, Jan Buys, and Phil Blunsom. 2016. Online segment to segment neural transduction. In *EMNLP*.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. arXiv:1212.5701.

Distance-Free Modeling of Multi-Predicate Interactions in End-to-End Japanese Predicate-Argument Structure Analysis

Yuichiroh Matsubayashi[♠◇] and **Kentaro Inui**[♠◇]
[♠]Graduate School of Information Sciences, Tohoku University
[◇]RIKEN Center for Advanced Intelligence Project
{y-matsu, inui}@ecei.tohoku.ac.jp

Abstract

Capturing interactions among multiple predicate-argument structures (PASs) is a crucial issue in the task of analyzing PAS in Japanese. In this paper, we propose new Japanese PAS analysis models that integrate the label prediction information of arguments in multiple PASs by extending the input and last layers of a standard deep bidirectional recurrent neural network (bi-RNN) model. In these models, using the mechanisms of pooling and attention, we aim to directly capture the potential interactions among multiple PASs, without being disturbed by the word order and distance. Our experiments show that the proposed models improve the prediction accuracy specifically for cases where the predicate and argument are in an indirect dependency relation and achieve a new state of the art in the overall F_1 on a standard benchmark corpus.

1 Introduction

A predicate-argument structure (PAS) is a structure that represents the relationships between a predicate and its arguments. Identifying PASs in Japanese text is a long-standing challenge chiefly due to the abundance of omitted (elliptical) arguments. In the example in Figure 1, the dative relation between *answer* and *reporters* is not explicitly indicated by the syntactic structure of the sentence. We regard such arguments as elliptical and call those argument slots *Zero* cases. 25% of the obligatory arguments in Japanese newspaper articles are reported to be elliptical.[1] The accuracy of identifying the fillers of such Zero cases remains only around 50% in terms of F_1 even if the task is restricted to the identification of intra-sentential predicate-argument relations (Matsubayashi and Inui, 2017).

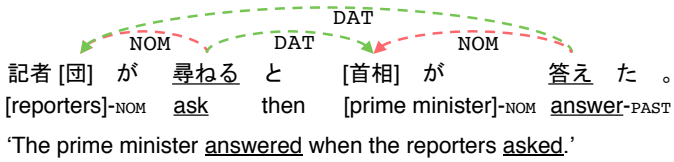
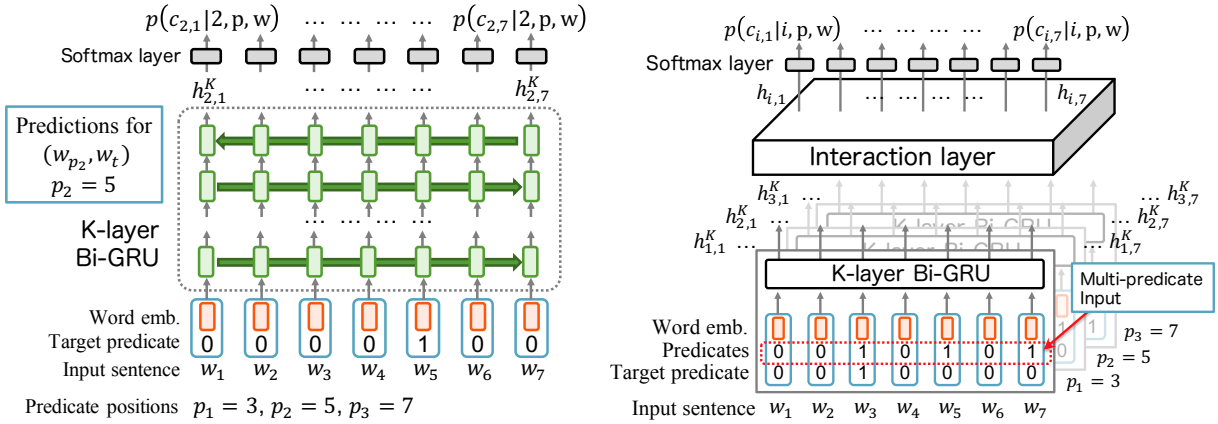


Figure 1: Example of PAS analysis. The dashed lines represent the predicate-argument relations. "[reporters]-NOM ask then" constitutes a subordinate clause and "[prime minister]-NOM answer-PAST" constitutes a matrix clause.

One promising approach for addressing this problem is to model argument sharing across multiple predicates (Iida et al., 2015; Ouchi et al., 2015; Ouchi et al., 2017). In Figure 1, for example, one can find very limited syntactic clues for predicting the long-distance dative relation between *answer* and *reporters*. However, the relation must be easy to identify for human readers who know that *the person who asks a question is likely to be answered*; namely, the nominative argument of *ask* is likely to be shared with the dative argument of *answer*. Capturing such inter-predicative dependencies has, therefore, been considered crucial of Japanese PAS analysis.

[1]Statistics from the NAIST Text Corpus 1.5. (Iida et al., 2017)

Proceedings of the 27th International Conference on Computational Linguistics, pages 94–106
Santa Fe, New Mexico, USA, August 20-26, 2018.



(a) Base model (b) Proposed models

Figure 2: Network structures of the base and proposed models.

With this goal in mind, Iida et al. (2015) constructed a *subject-shared predicate network* with an accurate recognizer of subject-sharing relations and deterministically propagated the predicted subjects to the other predicates in the graph. However, this method is applied only to subject sharing, so it cannot take into account the relationships among multiple argument labels.

More recently, as an end-to-end model considering multi-predicate dependencies, Ouchi et al. (2017) used Grid RNN to incorporate intermediate representations of the prediction for one predicate generated by an RNN layer into the inputs of the RNN layer for another predicate. However, in this model, since the information of multiple predicates also propagates through the RNNs, the integration of the prediction information is influenced by word order and distance, which is not necessarily important for aspects of syntactic and semantic relations. Consequently, there might be information loss caused by the surface distances of words, as previous work had pointed out for RNN language models (Linzen et al., 2016).

In this study, we propose new Japanese PAS analysis models that integrate the prediction information of arguments in multiple predicates. We extend a standard end-to-end style deep bi-RNN model (Figure 2a) and introduce components that consider the multiple predicate interactions into both the input and last layers (Figures 2b and 3). In contrast to Grid RNN, our extended models stack the extra layers using pooling and attention mechanisms on top of a deep bi-RNN so that they can directly associate the label prediction information for a target (predicate, word) pair with the predictions for words strongly related to the target pair. Through experiments, we show that the proposed models improve argument prediction accuracy, especially for the *Zero* cases, and achieve a new state-of-the-art performance in the overall F_1 on a standard benchmark corpus.

2 Task

In this paper, we employ a task definition based on the NAIST Text Corpus (NTC) (Iida et al., 2010; Iida et al., 2017), a commonly used benchmark corpus annotated with nominative (NOM), accusative (ACC), and dative (DAT) arguments for predicates. Given a tokenized sentence $\mathrm{w} = w_1, ..., w_n$ and its predicate positions $\mathrm{p} = p_1, ..., p_q$, our task is to identify at most one head of the filler tokens for each argument slot of each predicate. In this study, we follow the setting of Iida et al. (2015), Ouchi et al. (2017), and Matsubayashi and Inui (2017), and focus only on analyzing arguments in a target sentence. In addition, we exclude argument instances that are in the same *bunsetsu*, a base phrase unit in Japanese, as the target predicate, following Ouchi et al. (2017), which we will compare with the results in experiments.

The semantic labels used in NTC may seem to be rather syntactic as they are named nominative, accusative, etc. However, this annotation task markedly differs from shallow syntactic parsing and is, in fact, more like a semantic role labeling (SRL) task including implicit argument prediction. First, the semantic labels in NTC generalize case alteration caused by voice alteration and thus represent semantic roles analogous to ARG0, ARG1, etc. in the PropBank-style annotation (Palmer et al., 2005). Second,

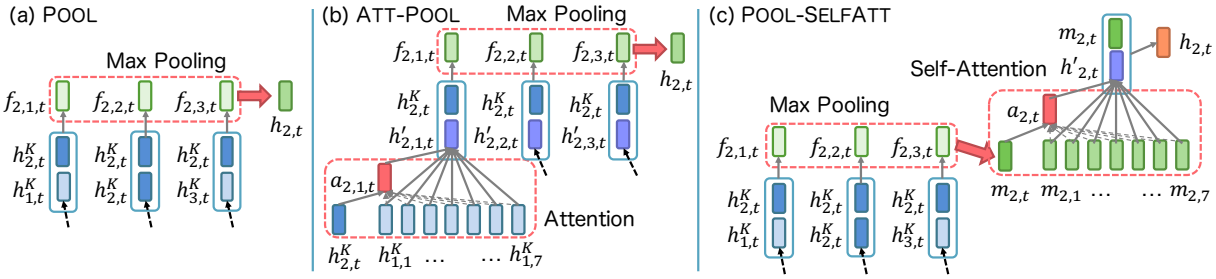


Figure 3: Three variants of interaction layers.

in the corpus, when an argument is omitted (i.e., zero-anaphora), the antecedent is identified with an appropriate semantic role, which is a prominent problem in Japanese semantic analysis and is the primary target of this study.

3 Base Model

Our proposed models extend end-to-end style SRL systems using deep bi-RNN (Zhou and Xu, 2015; He et al., 2017; Ouchi et al., 2017) to combine mechanisms that consider multiple predicate interactions. Figure 2a shows the network of our base model. Formally, given a word sequence $\mathrm{w} = w_1, ..., w_n$ and a target predicate position p_i in p, the model outputs a label probability for each word position: $p(c_{i,1}|i, \mathrm{p}, \mathrm{w}), ..., p(c_{i,n}|i, \mathrm{p}, \mathrm{w})$. Here, $c_{i,t} \in \{\texttt{NOM}, \texttt{ACC}, \texttt{DAT}, \texttt{NONE}\}$ represents the argument label of the word w_t for the target predicate w_{p_i}.

The input layer creates a vector $h^0_{i,t} \in \mathbb{R}^{d_w+1}$ for each pair of a predicate w_{p_i} and a word w_t by concatenating a word embedding $e(w_t) \in \mathbb{R}^{d_w}$ and a binary value representing the target predicate position in a method similar to that of He et al. (2017). The obtained vectors are then input into the deep bi-RNN, where the directions of the layers alternate (Zhou and Xu, 2015):

$$h^1_{i,t} = r^1(h^0_{i,t}, h^1_{i,t-1}), \quad h^k_{i,t} = \begin{cases} h^{k-1}_{i,t} + r^k(h^{k-1}_{i,t}, h^k_{i,t-1}) & (k \text{ is odd}) \\ h^{k-1}_{i,t} + r^k(h^{k-1}_{i,t}, h^k_{i,t+1}) & (k \text{ is even}) \end{cases} (k \geq 2). \tag{1}$$

Here, $h^k_{i,t} \in \mathbb{R}^{d_r}$ is the output of the k-th RNN layer for a pair (w_{p_i}, w_t), and r^k is a function representing the k-th RNN layer. We employ gated recurrent units (GRUs) (Cho et al., 2014) for the RNNs. In addition, we use the residual connections (He et al., 2016) following Ouchi et al. (2017). Then, a four-dimensional vector representing a probability $p(c_{i,t}|i, \mathrm{p}, \mathrm{w})$ is obtained by applying a softmax layer to each output of the last RNN layer $h^K_{i,t}$. For each argument label c of each predicate, we eventually select a word with the maximum probability that exceeds an output threshold θ_c.

4 Proposed Models

Our base model independently predicts the arguments of each predicate. In order to capture dependencies between the arguments of multiple predicates, we apply two extensions to our base model: a *multi-predicate input layer* and three variants of *interaction layers* on top of the deep bi-RNNs. Figures 2b and 3 show the network structures of the extended models.

In contrast to the Grid RNN model of Ouchi et al. (2017), where the information of multiple predicates propagates through the RNNs, our interaction layers use pooling and attention mechanisms to directly associate the label prediction information for a target (predicate, word) pair with that for words strongly related to the target pair, without being disturbed by word order and distance.

4.1 Interaction Layers

Pooling (POOL) Argument sharing across multiple predicates can be captured with both syntactic and semantic clues. At the syntactic level, we want to capture tendencies that, for example, the subject of the predicate of a matrix clause is likely to fill argument slots of other predicates in the same sentence.

At the semantic level, we want to model semantic dependencies between neighboring events such as *the person who asks a question is likely to be answered*, as in Figure 1. Our proposal is to capture both types of clues by incorporating a max pooling layer on top of the base model.

Specifically, as illustrated in Figure 3a, for each word w_t, we integrate the intermediate representation of label prediction for each predicate $h^K_{i,t}$ by applying max pooling to the vectors that represent pairs of prediction information for two predicates $h^K_{i,t}$ and $h^K_{j,t}$ (including the case $i = j$):

$$h_{i,t} = \mathrm{maxpool}_j(f_{i,j,t}), \quad \text{where} \quad f_{i,j,t} = \mathrm{ReLU}(W_f[h^K_{i,t}, h^K_{j,t}] + b_f). \tag{2}$$

In this equation, $\mathrm{maxpool}_j(f_{i,j,t})$ is an operation to extract the maximum value of each dimension in $\{f_{i,1,t}, ..., f_{i,q,t}\}$. The newly obtained vector $h_{i,t}$ for w_{p_i} and w_t is input into the softmax layer in the same manner as in the base model.

Attention-then-Pooling (ATT-POOL) Besides the argument sharing across multiple predicates, we would also like to capture dependencies between different arguments of a single predicate (and potentially, arguments of multiple predicates). For example, syntactically, two distinct argument slots of a single predicate are unlikely to share the same filler. Semantically, the subject of a predicate *take* is likely to be a person when its object is *a bread*, but is likely to be a company if the object is *a new employee*.

To capture such dependencies, we integrate the intermediate label prediction $h^K_{j,t'}$ of $w_{t'}$ for an arbitrary predicate w_{p_j} (including the case $i = j$) into the prediction of w_t for a target predicate w_{p_i}. In the integration, we aim to weigh the prediction information for $(w_{p_j}, w_{t'})$ based on its relatedness to the target pair (w_{p_i}, w_t) using the attention mechanism (Bahdanau et al., 2015). As in Figure 3b, we calculate a weight $a_{i,j,t}(t') \in \mathbb{R}$ for each of $h^K_{j,1}, ..., h^K_{j,n}$ on the basis of the prediction $h^K_{i,t}$ for the target pair and we obtain a weighted sum of $h^K_{j,t'}$ as a summary of the argument information of w_{p_j}, which is expected to be useful for the label prediction of (w_{p_i}, w_t):

$$h'_{i,j,t} = \textstyle\sum_{t'} a_{i,j,t}(t') \cdot h^K_{j,t'}, \quad \text{where} \quad a_{i,j,t}(t') = \frac{\exp(W_a g_{i,j,t,t'} + b_a)}{\sum_{t''} \exp(W_a g_{i,j,t,t''} + b_a)}, \tag{3}$$

$$g_{i,j,t,t'} = \tanh(W_g[h^K_{i,t}, h^K_{j,t'}] + b_g). \tag{4}$$

The obtained $h'_{i,j,t}$ are concatenated with the prediction for the target pair $h^K_{i,t}$ and linearly transformed with the ReLU activation. Max pooling is then applied to these vectors to combine the predictions for multiple predicates.

$$h_{i,t} = \mathrm{maxpool}_j(f_{i,j,t}), \quad \text{where} \quad f_{i,j,t} = \mathrm{ReLU}(W_f[h^K_{i,t}, h'_{i,j,t}] + b_f) \tag{5}$$

Pooling-then-Self-Attention (POOL-SELFATT) The ATT-POOL model involves a high computational cost because it must compute nq^2 different attentions regarding the number of words n and the number of predicates q in a sentence. Therefore, as illustrated in Figure 3c, in this model, we first apply the max pooling that we applied in the POOL model to reduce the sequences for which attentions must be computed by integrating the label predictions of w_t for all the other predicates in advance.

$$m_{i,t} = \mathrm{maxpool}_j(f_{i,j,t}), \quad \text{where} \quad f_{i,j,t} = \mathrm{ReLU}(W_f[h^K_{i,t}, h^K_{j,t}] + b_f) \tag{6}$$

Then, we combine the information in the obtained sequence $m_{i,1}, ..., m_{i,n}$ in a similar manner as in the ATT-POOL model using the attention mechanism, but this time, with self-attention, that is, computing the weights of the elements in the sequence based on the relatedness to the element inside the sequence.

$$h_{i,t} = \mathrm{ReLU}(W_h[m_{i,t}, h'_{i,t}] + b_h) \tag{7}$$

$$h'_{i,t} = \textstyle\sum_{t'} a_{i,t}(t') \cdot m_{i,t'}, \quad \text{where} \quad a_{i,t}(t') = \frac{\exp(W_a g_{i,t,t'} + b_a)}{\sum_{t''} \exp(W_a g_{i,t,t''} + b_a)} \tag{8}$$

$$g_{i,t,t'} = \tanh(W_g[m_{i,t}, m_{i,t'}] + b_g) \tag{9}$$

Consequently, the number of attentions that must be computed is reduced to nq.

Self-Attention (SELFATT) To conduct ablation tests to assess the impact of the proposed extensions, we also implemented a model only with self-attention. This model explicitly considers the relationships between arguments of a single predicate, but not arguments across multiple predicates.

$$h_{i,t} = \mathrm{ReLU}(W_h[h^K_{i,t}, h'_{i,t}] + b_h) \tag{10}$$

$$h'_{i,t} = \textstyle\sum_{t'} a_{i,t}(t') \cdot h^K_{i,t'}, \quad \text{where} \quad a_{i,t}(t') = \frac{\exp(W_a g_{i,t,t'} + b_a)}{\sum_{t''} \exp(W_a g_{i,t,t''} + b_a)} \tag{11}$$

$$g_{i,t,t'} = \tanh(W_g[h^K_{i,t}, h^K_{i,t'}] + b_g) \tag{12}$$

4.2 Multi-Predicate Input Layer (MP)

In addition, we add a simple but effective extension to the input layer. As He et al. (2016) reported, the information of the target predicate w_{p_i} propagates to the intermediate prediction $h^K_{i,t}$ of the candidate argument w_t through the deep bi-RNN by just adding a binary value representing the predicate position. Inspired by this finding, as shown in Figure 2b, in the input layer, we add another binary value that represents all the predicate positions to $h^0_{i,t}$, aiming to propagate multiple predicate information.

5 Experiments

We evaluated the impacts of our extensions and compared their performances to those of previous studies. Our main hypothesis is that the pooling and attention mechanisms are both useful for capturing different types of argument interactions as we explained in Section 4 and do work complementarily of each other to improve the prediction accuracy, especially for arguments in a long-distance dependency.

5.1 Settings

5.2 Dataset and Implementation Details

The experiments were performed on NTC 1.5. We divided the corpus into the commonly used divisions of training, development, and test sets (Taira et al., 2008), each of which includes 24,283, 4,833, and 9,284 sentences, respectively. NTC represents each argument of a predicate by indicating a coreference cluster in a text. For each given predicate-argument slot, we count a system's output as correct if the output token is included in the coreference cluster corresponding to the slot fillers. The evaluation is performed on the basis of the precision, recall, and F_1 score.

The hyperparameters were selected to obtain a maximum F_1 on the development set. The details of the hyperparameter selection and preprocessing are described in the supplemental material. In the following experiments, we train each model 10 times with the same training data and hyperparameters and then show the average scores.

5.3 Grid RNN Baseline (GRID)

In order to strictly compare the impact of our extensions to the method used for integrating multiple pieces of predicate information in the state-of-the-art end-to-end model, in addition to our base model, we replicated the method of Ouchi et al. (2017) by modifying Equations (1) of our base model as follows:

$$h^1_{i,t} = r^1([h^0_{i,t}, h^1_{i-1,t}], h^1_{i,t-1}), \quad h^k_{i,t} = h^{k-1}_{i,t} + \begin{cases} r^k([h^{k-1}_{i,t}, h^k_{i-1,t}], h^k_{i,t-1}) & (k \text{ is odd}) \\ r^k([h^{k-1}_{i,t}, h^k_{i+1,t}], h^k_{i,t+1}) & (k \text{ is even}) \end{cases} (k \geq 2), \tag{13}$$

if $1 \leq i \leq q$; otherwise, $h^k_{i,t} = \mathbf{0}$. The performance of this replicated model may not be strictly the same as that reported in Ouchi et al. (2017) due to discrepancies in the embeddings of inputs, hyperparameters (a training batch size, a hidden unit size, etc.), and training strategy (an optimizing algorithm, a regularization method, an early stopping method, etc.). The predicate positions $\mathrm{p} = p_1, ..., p_q$ are arranged in ascending order.

		All				F_1 at different dependency distances					
	Model	F_1 (%)	SD	Prec.	Rec.	*Dep*	*Zero*	2	3	4	≥ 5
Baseline Models	BASE ($d_r = 32, K = 8$)	81.22	±0.19	84.30	78.37	88.39	49.12	55.73	47.1	39	29
	GRID ($d_r = 32, K = 8$)	81.06	±0.31	84.33	78.04	88.17	48.73	55.26	47.5	39	28
	BASE	83.39	±0.13	85.85	81.07	89.90	54.37	61.09	53.8	44	31
	GRID	82.94	±0.17	85.38	80.63	89.51	53.57	60.28	52.4	44	32
Proposed Models	SELFATT	83.56	±0.22	85.91	81.34	90.06	54.84	61.36	54.3	**45**	32
	POOL	83.56	±0.16	86.05	81.21	90.00	54.81	61.54	54.3	**45**	31
	ATT-POOL	83.48	±0.24	85.97	81.12	89.98	54.57	61.19	54.0	44	32
	POOL-SELFATT	83.76	±0.17	86.11	81.54	90.17	55.19	62.10	54.0	**45**	32
	MP	83.67	±0.22	86.08	81.39	90.10	54.80	61.67	53.8	44	32
	MP-SELFATT	83.79	±0.22	86.11	**81.60**	90.22	55.26	61.88	54.3	**45**	**33**
	MP-POOL-SELFATT	**83.94**	±0.12	**86.58**	81.46	**90.26**	**55.55**	**62.44**	**54.7**	**45**	32
Previous SOTAs	Ouchi et al. (2017)	81.42				88.17	47.12				
	M&I 2017	83.50	±0.17	86.00	81.15	89.89	51.79	60.17	49.4	38	23
Ensemble Models	MP-POOL-SELFATT (10 models)	**85.34**		**87.90**	**82.93**	**91.26**	**58.07**	**64.89**	**57.5**	**47**	**33**
	M&I 2017 (5 models)	84.07		86.09	82.15	90.24	53.66	61.94	51.8	40	24

Table 1: F_1 scores on the NTC 1.5 test set. *Dep* and *Zero* denote instances where the dependency distance between the predicate and argument is one and more than one, respectively. M&I 2017 is the model of Matsubayashi and Inui (2017).

Model A \ Model B	F_1 (%)	SD	BASE	ATT-POOL	SELFATT	POOL	MP	POOL-SELFATT	MP-SELFATT
BASE	83.39	±0.13							
ATT-POOL	83.48	±0.24	0.18						
SELFATT	83.56	±0.22	**0.03**	0.22					
POOL	83.56	±0.16	**0.014**	0.21	0.53				
MP	83.67	±0.22	**0.003**	**0.048**	0.16	0.12			
POOL-SELFATT	83.76	±0.17	**4.3E-5**	**0.004**	**0.023**	**0.0084**	0.16		
MP-SELFATT	83.79	±0.22	**1.0E-4**	**0.0046**	**0.021**	**0.0096**	0.13	0.39	
MP-POOL-SELFATT	83.94	±0.12	**5.4E-6**	**5.4E-6**	**2.2E-4**	**2.7E-5**	**0.0013**	**0.013**	**0.046**

Table 2: *p-values* in one-sided permutation test using 10 overall F_1 scores for each model. The bold values indicate that an average F_1 score of model A outperforms that of model B at the 5% significance level.

5.4 Results

Impact of Extensions

The first two sets of rows in Table 1 compare the impact of each component of our extension. The effects of incorporating the interaction layer can be seen in the comparisons of the BASE model with the SELFATT, POOL, ATT-POOL, and POOL-SELFATT models. Among the four proposed extensions, POOL-SELFATT, an integration of POOL and SELFATT, achieved the best performance (83.76 in F_1), gaining 0.37 points in overall F_1 from BASE. Also, the significance tests in Table 2 show that the POOL and SELFATT models significantly outperform the BASE model, and the POOL-SELFATT model makes a further significant gain from the POOL and SELFATT models. This indicates that POOL and SELFATT work complementarily with each other, and combining them makes a further improvement from each individual extension. Recall that SELFATT is designed to capture long-distance dependencies over a single predicate-argument structure, whereas POOL is expected to capture argument sharing across multiple predicates. These results provide empirical support to the hypotheses behind our design of the interaction layer.

The MP model, where the input layer is extended to represent the positions of all the predicates in a sentence, significantly outperforms the BASE model by 0.28 points in overall F_1. This result suggests the importance of position information regarding the neighboring predicates in identifying the arguments of a given predicate. Furthermore, the MP-POOL-SELFATT model, which is a combination of MP and POOL-SELFATT, resulted in a further 0.27-point improvement and consequently achieved the best overall F_1 of 83.94 as a single model.

Following Matsubayashi and Inui (2017), we also assess F_1 values at different dependency distances. The results are shown in the right half of Table 1. From the table, we can see that MP-POOL-SELFATT

Model	ALL	Dep				Zero			
		ALL	NOM	ACC	DAT	ALL	NOM	ACC	DAT
MP-POOL-SELFATT	**83.94**	**90.26**	90.88	94.99	**67.57**	**55.55**	**57.99**	**48.9**	**23**
Ouchi et al. (2015)	79.23	86.07	88.13	92.74	38.39	44.09	48.11	24.4	4.8
Ouchi et al. (2017)	81.42	88.17	88.75	93.68	64.38	47.12	50.65	32.4	7.5
M&I 2017	83.50	89.89	**91.19**	**95.18**	61.90	51.79	54.69	41.8	17
MP-POOL-SELFATT (ens.)	**85.34**	**91.26**	**91.84**	**95.57**	**70.8**	**58.07**	**60.21**	**52.5**	**26**
M&I 2017 (ens. of 5)	84.07	90.24	91.59	95.29	62.61	53.66	56.47	44.7	16

Modified NTC 1.5 (Iida et al., 2016)

Model	Zero NOM
Ouchi et al. (2015)	**57.3**
Iida et al. (2015)	41.1
Iida et al. (2016)	52.5

(Note) Results on a dataset different from our experiments

Table 3: F_1 scores of each argument label on the NTC 1.5 test set.

improves F_1 from BASE by 0.9–1.4 points consistently across all the distance categories other than *Dep*.

Comparison to Related Work

The third set of rows in Table 1 shows the reported performance of related studies. Grid RNN of Ouchi et al. (2017) is a state-of-the-art end-to-end model, designed to capture interactions among multiple predicate-argument relations. A comparison between their model and the proposed models was somewhat tricky because our replication of Grid RNN did not reproduce the reported performance on the same dataset (see the row of GRID in Table 1). Unlike the results reported in Ouchi et al. (2017), the GRID model in our experiment did not clearly outperform the model without the grid architecture, i.e., the Base model. We first suspected that this might have resulted from the difference in dimensionality d_r of RNN hidden states: $d_r = 32$ in Ouchi et al. (2017), whereas $d_r = 256$ in our experiments. Specifically, we speculated that the base model with a low dimensionality left a larger margin for improvement and incorporating the Grid architecture derived positive effects. We thus trained our GRID model with Ouchi et al. (2017)'s settings ($d_r = 32$ and $K = 8$) and the best performing hyperparameters; however, we were not able to reproduce the reported gain from Grid RNN (see the row of "GRID ($d_r = 32$, $K = 8$)" in Table 1).[2] This might be an indication of the difficulty in capturing multi-predicate interactions by threading deep bi-RNNs with RNNs, as we discussed in Section 1.

Another previous state-of-the-art model was proposed by Matsubayashi and Inui (2017) (M&I 2017). This model extends a feedforward NN with dependency path embeddings and other new features to capture long-distance dependencies in a single PAS. The row "M&I 2017" in Table 1 shows the reported performance of their model.[3] The performance of M&I 2017 is comparable with the performance of our SELFATT model. This result provides another piece of empirical evidence that the self-attention mechanism has a comparably positive effect in incorporating dependency path information for capturing long-distance dependencies in a single PAS.

Overall, the proposed methods of using the pooling and attention mechanisms for capturing interactions across predicates and arguments gained considerable improvement and achieved state-of-the-art accuracy, significantly outperforming the previous state-of-the-art models. The last set of rows in Table 1 shows the results of the ensemble models. A model that predicts arguments with the average score of the 10 MP-POOL-ATT models further improves the overall F_1 by 1.4 points from that of a single model, achieving state-of-the-art accuracy for NTC 1.5.

Table 3 shows the F_1 score for each case label. In a comparison of the single models, although our MP-POOL-ATT model slightly degrades the scores of NOM and ACC on the *Dep* cases compared to the state-of-the-art model (M&I 2017), it greatly improves the scores for DAT and the *Zero* cases. Regarding the ensemble models, MP-POOL-ATT improves the scores for all cases.

Iida et al. (2015) and Iida et al. (2016) report Japanese subject anaphora resolution systems, designed to predict only *Zero* NOM arguments. It is not straightforward to directly compare their results with ours due to the differences in the experimental settings. However, our best performing model outperforms the

[2]We discussed this negative result, including the implementation details, with one of the authors of Ouchi et al. (2017). However, we could not find a plausible reason for the results.

[3]For the purpose of a strict comparison with Ouchi et al. (2017), we re-evaluate the model of Matsubayashi and Inui (2017) by excluding the instances for which the argument is in the same *bunsetsu* phrase as the predicate; this is the same setting as that in Ouchi et al. (2017). We have reported the new results in Tables 1 and 3.

(1) 背筋 を [伸ばし $_{\text{PRED}}$] 少 [考 $_{\text{NOM\_FALSE}}$] の 後 、 応じる [谷川 $_{\text{NOM\_GOLD}}$] 。
spine ACC stretch little thinking of after (nominal) , respond Tanigawa .
'[Tanigawa $_{\text{NOM\_GOLD}}$], responding after [stretching $_{\text{PRED}}$] his spine and [thinking $_{\text{NOM\_FALSE}}$] briefly.'

(2) 大学 教授 [ら $_{\text{NOM\_GOLD}}$] が 地下 に 潜って 、 ファクス で 連絡 を 取り 合い 、 地方 の 組織 の 活動 を [支えて $_{\text{PRED}}$] いる 。 [*NONE*$_{\text{NOM\_FALSE}}$]
university professor PLURAL NOM underground DAT dive , fax by contacting ACC take each-other , district of organization of activities ACC support PROGRESSIVE
'The university [professors $_{\text{NOM\_GOLD}}$] went into hiding and are [supporting $_{\text{PRED}}$] the activities of the local organizations, contacting each other by fax. [NONE $_{\text{NOM\_FALSE}}$]'

(3) 中央 [省庁 $_{\text{NOM\_FALSE}}$] が [職員 $_{\text{NOM\_GOLD}}$] に 対し 「 夜 の 接待 は [受ける $_{\text{PRED}}$] な 」 と 通達 すれば 済む こと だ 。
central ministries NOM staff DAT against " night in entertainment TOP accept NEG$_{\text{IMPERATIVE}}$ " as notification VERBALIZER$_{\text{CONDITIONAL}}$ finish NOMINALIZER COPULA.
'It is sufficient enough if the central [ministries $_{\text{NOM\_FALSE}}$] tell the [staff$_{\text{NOM_GOLD}}$] "Do not [accept $_{\text{PRED}}$] a business dinner."'

(4) 十三 日 午後 に は 盆栽 作家 、 木村 正彦 [氏 $_{\text{NOM\_GOLD}}$] に よる 実技 の デモンストレーション が [行わ $_{\text{PRED}}$] れる 。 [*NONE*$_{\text{NOM\_FALSE}}$]
13 day afternoon DAT TOP bonsai artist , Kimura Masahiko Mr. {by- -by} techniques of demonstration NOM perform PASSIVE .
'On the afternoon of 13th, a practical demonstration by the bonsai artist [Mr. $_{\text{NOM\_GOLD}}$] Masahiko Kimura will be [performed $_{\text{PRED}}$]. [NONE $_{\text{NOM\_FALSE}}$]'

Figure 4: Examples of prediction errors. In Example (1), only SELFATT failed to predict the answer. In Example (2), only MP-POOL-SELFATT correctly predicted the answer. In Examples (3) and (4), none of the systems predict the answers correctly.

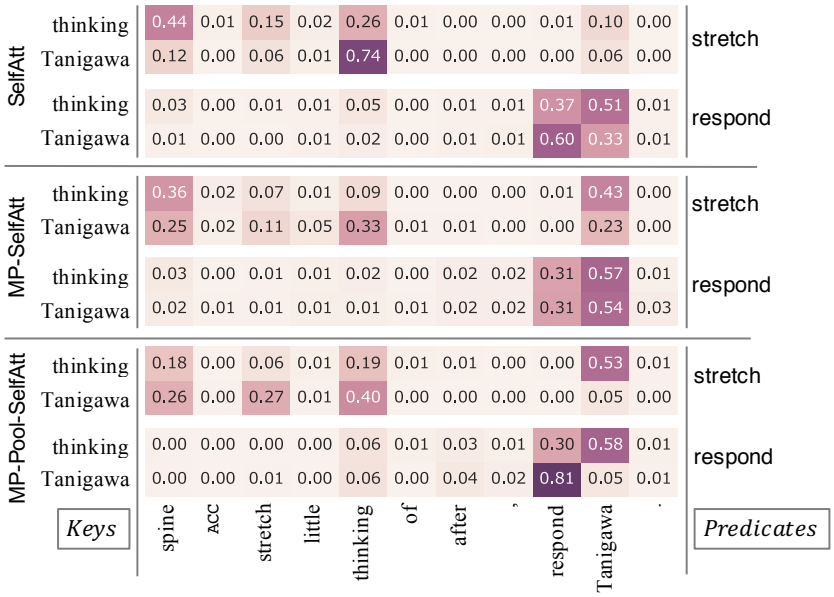


Figure 5: Attention weights of proposed models for Example (1).

model of Ouchi et al. (2015), which is then reported to outperform both Iida et al. (2015) and Iida et al. (2016) in their experimental settings.

5.5 Detailed Analysis

To analyze the behavior of our proposed models in detail, we show some prediction examples of the SELFATT, MP-SELFATT, and MP-POOL-ATT models in the development set with the weights in the attention layers in Figures 4-7.

In Figure 4, Examples (1) and (2) are the instances for which only SELFATT failed to predict the answer and for which only MP-POOL-SELFATT correctly predicted the answer, respectively. For these examples, the weights in the attention layers behave similarly. Figure 5 shows the weights for Example (1). In this sentence, the correct NOM of *stretch*, *Tanigawa*, is also NOM of *respond*, which is relatively easy to predict. SELFATT, which is designed to capture dependencies over a single predicate-argument structure, failed to predict NOM of *stretch* most likely because the answer *Tanigawa* is distant from the target predicate with its limited syntactic clues. Conversely, MP-POOL-SELFATT and MP-SELFATT successfully predicted the answer by taking the answer token *Tanigawa* into account when computing the

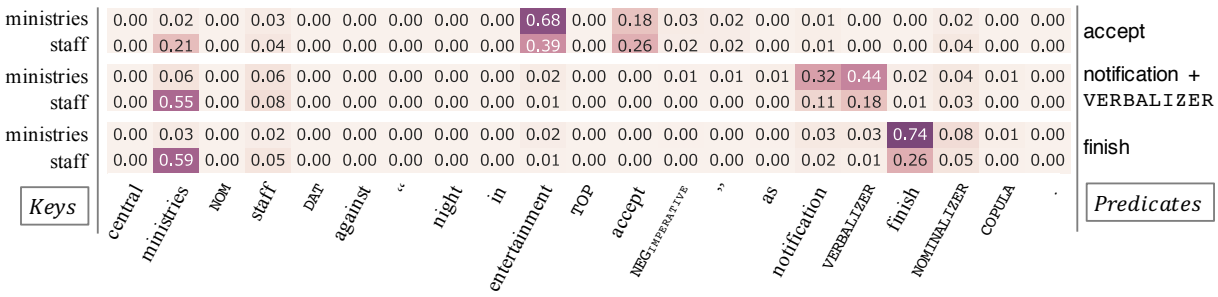


Figure 6: Attention weights of MP-POOL-SELFATT for Example (3).

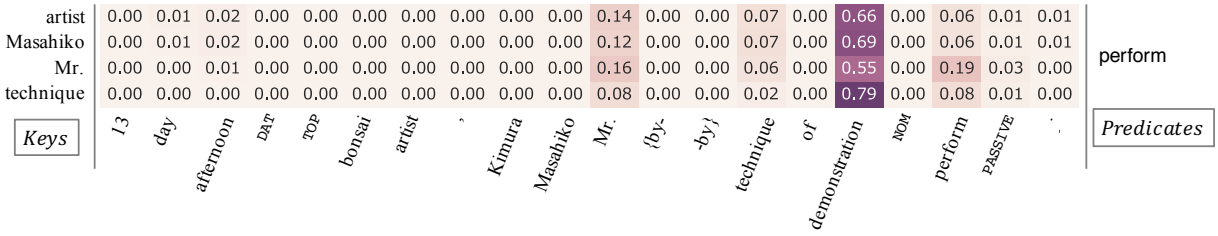


Figure 7: Attention weights of MP-POOL-SELFATT for Example (4).

score of the counter candidate *thinking*. MP-SELFATT, the model that incorporates the other predicate positions into SELFATT, significantly increases the weight for the answer token. MP-POOL-SELFATT, which explicitly integrates the predictions for the other predicates, further increases the weight for the answer token. This example demonstrates that the proposed extensions successfully predict a correct argument by considering the relation to the argument in another predicate where the syntactic relation between the predicate and argument is much clearer and thus the argument relation is relatively easy to predict. Due to space limitations, we cannot show the weights for Example (2), but the same also holds for that example. MP-POOL-SELFATT focuses on professors, which is the "easy-to-predict" NOM argument of dive, when the model computes the scores of this token for *take* and, consequently, *support*. SELFATT and MP-SELFATT assign smaller weights to that token for *take* and even smaller weights for *support*, which is far from the answer token.

Examples (3) and (4) are the instances where all the three models failed to predict the answers. Figure 6 illustrates the attention weights in MP-POOL-SELFATT for Example (3). To solve this example, the model is expected to understand that NOM of *accept* should be the same as the persons who received the order from the *ministries*. However, MP-POOL-SELFATT could not acquire this kind of dialog-level knowledge and pays little attention to the correct argument *staff* when the model computes the score of the wrong answer *ministries* for NOM of *accept*.

In Example (4), NOM of the nominal predicate *demonstration* can be a clue for predicting NOM of *perform*. However, the models currently do not predict the arguments of nominal predicates and therefore cannot capture the relationships between these two sufficiently (Figure 7). This example suggests one of our future directions: the joint prediction of verbal and nominal predicates.

6 Related Work

End-to-End Models in SRL End-to-end approaches to SRL have been widely explored recently, and many state-of-the-art results have been achieved (Zhou and Xu, 2015; He et al., 2017; Marcheggiani and Titov, 2017; Tan et al., 2018). Following these advanced models, we adopted a stacked bi-RNN as our base model.

Methods for Dealing with Long-Distance Dependencies in End-to-End Models In SRL studies, Marcheggiani and Titov (2017) proposed a variant of deep bi-RNN models that connects the intermediate representations of the predictions for the words in syntactic dependency relations on top of the deep RNN.

Very recently, aiming to directly connect the related words, Tan et al. (2018) stacked self-attention layers, each of which followed a feedforward layer, in a manner similar to the method of Vaswani et al. (2017), which was originally applied to an encoder-decoder model.

Self-attention has been successfully applied to several NLP tasks, including textual entailment, sentiment analysis, summarization, machine translation, and language understanding (Paulus et al., 2017; Shen et al., 2018; Lin et al., 2017; Vaswani et al., 2017). Techniques using pooling have been applied to merge intermediate expressions in predictions in the tasks where related tokens are often at long distance such as coreference resolution and machine reading (Clark and Manning, 2016; Kobayashi et al., 2016). One major contribution of this study is its novel idea of using these techniques for capturing long-distance dependencies for modeling interactions among multiple predicate-argument relations.

Approaches to Capturing Multi-Predicate Interactions For Japanese, Ouchi et al. (2015) jointly identified arguments of multiple predicates by modeling argument interactions with a bipartite graph. Iida et al. (2015) constructed a *subject-shared predicate network* and deterministically propagated the predicted subjects to other predicates. Shibata et al. (2016) adapted a NN framework to Ouchi et al. (2015)'s model using a feedforward network. For an end-to-end neural model, Ouchi et al. (2017) used a Grid RNN to capture multiple predicate interactions. Through experiments, we demonstrated that our proposed models outperformed these models in terms of the overall F_1 on a standard benchmark corpus.[4]

To the best of our knowledge, there are few previous studies related to SRL considering multiple predicate interactions for languages other than Japanese. Yang and Zong (2014) performed a discriminative reranking in the role classification of shared arguments. Lei et al. (2015) proposed an SRL model based on the dimensionality reduction on a tensor representation to capture meaningful interactions between the argument, predicate, corresponding features, and role label. It is not straightforward to compare these methods with our models; however, it is an intriguing future issue to consider how well the techniques devised for Japanese PAS analysis work for other languages.

Other Approaches to Argument Omission In order to perform robust prediction for arguments with fewer syntactic clues, several previous studies have explored various types of selectional preference scores that consider the semantic relations between a predicate and its arguments (Iida et al., 2007; Imamura et al., 2009; Komachi et al., 2010; Sasano and Kurohashi, 2011; Shibata et al., 2016). This direction of research is orthogonal to our approach, suggesting that the models could be further improved by being combined with these extra features.

7 Conclusion

In this study, we have proposed new Japanese PAS analysis models that integrate prediction information of arguments in multiple predicates. We extended the end-to-end style model using a deep bi-RNN and introduced the components that consider the multiple predicate interactions into the input and last layers. As a result, we achieved a new state-of-the-art accuracy on the standard benchmark data.

Our detailed analysis showed that the proposed models successfully predict the correct arguments by using the information of the *"easy-to-predict"* arguments in other predicates. In addition, the error analysis suggests that jointly predicting the arguments of verbal and nominal predicates may further improve the performance. An intriguing issue we plan to address next is how to extend the proposed interaction layer to cross-sentential interactions of PASs.

Acknowledgements

We are grateful to the anonymous reviewers for their useful comments and suggestions. We thank Hiroki Ouchi for his help in checking our re-implementation. We also thank Shun Kiyono and Kento Watanabe for valuable discussions. This work was partially supported by JSPS KAKENHI Grant Numbers 15H01702 and 15K16045.

[4] Shibata et al. (2016) evaluated the model on a different dataset and hence, it is difficult to compare the results directly.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*, pages 1–15.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*, pages 1724–1734.

Kevin Clark and Christopher D. Manning. 2016. Improving Coreference Resolution by Learning Entity-Level Distributed Representations. In *ACL*, pages 643–653.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778.

Luheng He, Kenton Lee, Mike Lewis, Luke Zettlemoyer, and Paul G Allen. 2017. Deep Semantic Role Labeling: What Works and What's Next. In *ACL*, pages 473–483.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2007. Zero-anaphora Resolution by Learning Rich Syntactic Pattern Features. *ACM Transactions on Asian Language Information Processing*, 6(4):1:1–1:22.

Ryu Iida, Mamoru Komachi, Naoya Inoue, Kentaro Inui, and Yuji Matsumoto. 2010. Annotating Predicate-Argument Relations and Anaphoric Relations: Findings from the Building of the NAIST Text Corpus. *Natural Language Processing*, 17(2):25–50.

Ryu Iida, Kentaro Torisawa, Chikara Hashimoto, Jong-Hoon Oh, and Julien Kloetzer. 2015. Intra-sentential Zero Anaphora Resolution using Subject Sharing Recognition. In *EMNLP*, pages 2179–2189.

Ryu Iida, Kentaro Torisawa, Jong-Hoon Oh, Canasai Kruengkrai, and Julien Kloetzer. 2016. Intra-Sentential Subject Zero Anaphora Resolution using Multi-Column Convolutional Neural Network. In *EMNLP*, pages 1244–1254.

Ryu Iida, Mamoru Komachi, Naoya Inoue, Kentaro Inui, and Yuji Matsumoto. 2017. NAIST Text Corpus: Annotating Predicate-Argument and Coreference Relations in Japanese. In *Handbook of Linguistic Annotation*, pages 1177–1196. Springer.

Kenji Imamura, Kuniko Saito, and Tomoko Izumi. 2009. Discriminative Approach to Predicate-Argument Structure Analysis with Zero-Anaphora Resolution. In *ACL-IJCNLP*, pages 85–88.

Sosuke Kobayashi, Ran Tian, Naoaki Okazaki, and Kentaro Inui. 2016. Dynamic Entity Representation with Max-pooling Improves Machine Reading. In *NAACL-HLT*, pages 850–855.

Mamoru Komachi, Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2010. Argument Structure Analysis of Event-nouns Using Lexico-syntactic Patterns of Noun Phrases. *Journal of Natural Language Processing*, 17(1):141–159.

Tao Lei, Yuan Zhang, Lluis Marquez, Alessandro Moschitti, and Regina Barzilay. 2015. High-Order Low-Rank Tensors for Semantic Role Labeling. In *NAACL-HLT*, pages 1149–1159.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A Structured Self-attentive Sentence Embedding. In *ICLR*, pages 1–15.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. *Transactions of the Association for Computational Linguistics*, 4(1):521–535.

Diego Marcheggiani and Ivan Titov. 2017. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling. In *EMNLP*, pages 1507–1516.

Yuichiroh Matsubayashi and Kentaro Inui. 2017. Revisiting the Design Issues of Local Models for Japanese Predicate-Argument Structure Analysis. In *IJCNLP*, pages 128–133.

Hiroki Ouchi, Hiroyuki Shindo, Kevin Duh, and Yuji Matsumoto. 2015. Joint Case Argument Identification for Japanese Predicate Argument Structure Analysis. In *ACL-IJCNLP*, pages 961–970.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2017. Neural Modeling of Multi-Predicate Interactions for Japanese Predicate Argument Structure Analysis. In *ACL*, pages 1591–1600.

M. Palmer, P. Kingsbury, and D. Gildea. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A Deep Reinforced Model for Abstractive Summarization. In *ICLR*, pages 1–12.

Ryohei Sasano and Sadao Kurohashi. 2011. A Discriminative Approach to Japanese Zero Anaphora Resolution with Large-scale Lexicalized Case Frames. In *IJCNLP*, pages 758–766.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. DiSAN: Directional Self-Attention Network for RNN/CNN-Free Language Understanding. In *AAAI-18*, pages 5446–5455.

Tomohide Shibata, Daisuke Kawahara, and Sadao Kurohashi. 2016. Neural Network-Based Model for Japanese Predicate Argument Structure Analysis. In *ACL*, pages 1235–1244.

Hirotoshi Taira, Sanae Fujita, and Masaaki Nagata. 2008. A Japanese Predicate Argument Structure Analysis using Decision Lists. In *EMNLP*, pages 523–532.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep Semantic Role Labeling with Self-Attention. In *AAAI-18*, pages 4929–4936.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *NIPS*, pages 5998–6008.

Haitong Yang and Chengqing Zong. 2014. Multi-Predicate Semantic Role Labeling. In *EMNLP*, pages 363–373.

Jie Zhou and Wei Xu. 2015. End-to-end Learning of Semantic Role Labeling Using Recurrent Neural Networks. In *ACL*, pages 1127–1137.

Appendix A: Implementation Details

Hyperparameters The hyperparameters were selected to obtain a maximum F_1 on the development set. The dimension of the word embeddings d_w was set to 256. The dimension of the hidden state of the GRUs d_r was set to 256 from $\{128, 256, 512\}$ and the number of the GRU layers was set to 10 from $\{6, 8, 10, 12\}$. The dropout rate of the GRUs was set to 0.1 from $\{0.0, 0.1, 0.2\}$. The dimensions of the outputs of the nonlinear transformations f, g and $h_{i,t}$ were set to 1024 from $\{512, 768, 1024\}$. We set the batch size of the training data as the number of predicates in each sentence. We employed the negative log likelihood as the training loss and an Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e-08$. During the training, we halved the learning rate when the F_1 score on the development set did not improve after four epochs, and restarted training with the parameters that obtained the maximum F_1 score. We repeated this process and terminated the training when the new learning rate was less than 1/16 of the initial value. The initial learning rate of each model was selected from $\{0.00002, 0.00005, 0.0001, 0.0002, 0.0005\}$. The output threshold for each label $\theta_c \in [0.0, 1.0]$ was searched in increments of 0.01 to maximize the F_1 score in the training data.

Preprocessing As initial word embeddings, we used vectors obtained via the same procedure as the one proposed by Matsubayashi and Inui (2017) using Japanese Wikipedia articles. These vectors were fine-tuned in the training. Following their approach, we used part of speech (PoS) vectors for words that were not contained in the lexicon of the Wikipedia articles. We used the CaboCha parser v0.68 [5] with the JUMAN dictionary for word segmentation and PoS tagging of NTC.

[5] https://taku910.github.io/cabocha/

Sprucing up the trees – Error detection in treebanks

Ines Rehbein
Leibniz ScienceCampus
Heidelberg/Mannheim
rehbein@cl.uni-heidelberg.de

Josef Ruppenhofer
Leibniz ScienceCampus
Heidelberg/Mannheim
ruppenhofer@ids.mannheim.de

Abstract

We present a method for detecting annotation errors in manually *and* automatically annotated dependency parse trees, based on ensemble parsing in combination with Bayesian inference, guided by active learning. We evaluate our method in different scenarios: (i) for error detection in dependency treebanks and (ii) for improving parsing accuracy on in- and out-of-domain data.

1 Introduction

Structural syntactic information is an important ingredient for many NLP applications. In recent years, the Universal Dependencies (UD) framework (Nivre et al., 2016) has become increasingly popular as a source for syntactically annotated training data, mainly for two reasons. First, the UD project provides a unified framework for multilingual applications and second, its annotation scheme encodes semantic information in a more transparent way, yielding better support for semantic applications.

The project already provides treebanks for over 60 languages, some of them, however, rather small. To obtain high parsing accuracies, a sufficient amount of training data is needed, preferably from different genres and domains. Since treebanking is a notoriously time-consuming task, most of the UD treebanks have been automatically converted from other frameworks and thus include some noise introduced in the conversion, in addition to the usual annotation noise that stems from inconsistencies in the human annotations. Due to limited funding, a full manual quality check is often not feasible. Therefore, methods that are able to detect errors in automatically predicted parse trees are of great value for data clean-up and might also be of use for domain adaptation settings and when dealing with low-resource languages.

We present such a method, aimed at the detection of parser errors in manually *and* automatically predicted parse trees. We evaluate our approach in two different scenarios, i) in an active learning (AL) setup where we try to detect and manually correct errors in existing treebanks, and ii) in a domain adaptation setting where we automatically improve parsing performance without manual correction. We make an implementation of our approach publically available.[1]

2 Model

Our error detection model is an adaptation and substantial extension of MACE-AL (Rehbein and Ruppenhofer, 2017) which combines a generative, unsupervised method for estimating annotator reliability (MACE) (Hovy et al., 2013) with active learning for the task of error detection in *automatically* annotated data. MACE-AL can be applied to any classification task and has been tested in two applications, POS tagging and Named Entity Recognition (NER) (Rehbein and Ruppenhofer, 2017). The model is, however, not applicable to tasks with structured output such as trees or graphs. In contrast to POS tagging or NER where we try to detect annotation errors in the predicted labels for individual tokens, when looking for errors in parse trees we have to deal with directed, labelled relations between nodes, and changing the relation or label between two nodes in the tree usually requires the adjustment of other attachment and labelling decisions in the same tree.

[1] MACE-AL-TREE can be downloaded from http://www.cl.uni-heidelberg.de/research/downloads/

Proceedings of the 27th International Conference on Computational Linguistics, pages 107–118
Santa Fe, New Mexico, USA, August 20-26, 2018.

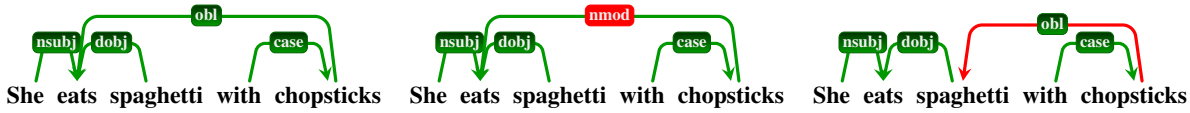


Figure 1: Correct tree (left), tree with label error (center) and tree with attachment error (right).

A high-precision method for error detection in automatically predicted trees, however, would be of tremendous use for many treebanking projects. Below, we first describe the approach of Hovy et al. (2013) and Rehbein and Ruppenhofer (2017) for unstructured data (§2.1) and then our extension of the model for the purpose of error detection in tree structures (§2.2).

2.1 MACE-AL: Error detection in unstructured data

MACE-AL combines variational inference (VI) with active learning (AL) to model the reliability of *automatically* predicted annotations. As input, it takes the output of a committee of classifiers (e.g. the output of N POS taggers) and uses *Bayesian inference* to learn which taggers' predictions are more trustworthy than others. The method is unsupervised, meaning that the distribution of the true labels $Y = \{y_1, y_2, ..., y_n\}$ is unknown. Instead, the model tries to approximate the posterior distribution over the set of unobserved random variables Y by a variational distribution $Q(Y)$, based on the observed data X (the predictions made by the classifier committee).

MACE-AL is based on MACE (Hovy et al., 2013), a tool for estimating annotator reliability in a crowdsourcing setting. MACE implements a simple graphical model where the observed annotations A are generated as follows. The (unobserved) "true" label T_i is sampled from a uniform prior, based on the assumption that the annotators always try to predict the correct label and thus the majority of the annotations should, more often than not, be correct. To model each annotator's behaviour, a binary variable S_{ij} (also unobserved) is drawn from a Bernoulli distribution that describes whether annotator j is trying to predict the correct label for instance i or whether s/he is just spamming (a behaviour not uncommon in crowdsourcing settings). If S_{ij} is 0, the "true" label T_i is used to generate the annotation A_{ij}. If S_{ij} is 1, the predicted label A_{ij} for instance i comes from a multinomial distribution with parameter vector ξ_j.

Rehbein and Ruppenhofer (2017) showed that MACE, while highly successful when applied to human annotations from crowdsourcing, is not able to outperform the majority baseline when the predictions have been generated *automatically* by a classifier committee. MACE-AL, however, addresses this shortcoming by combining variational inference with human feedback from active learning. Active learning is an approach to minimise human annotation effort by only selecting instances for labelling that provide useful information to the classifier. It has been shown that annotating only a small set of carefully selected instances not only reduces manual annotation effort but can yield the same accuracy as when training on a larger set of randomly selected instances (Settles, 2009).

The extended model, guided by human feedback, is now able to pick up on the signal and significantly outperforms the majority baseline as well as a query by committee strategy for error detection where the next instance for manual inspection is selected based on the entropy in the classifiers' predictions.

2.2 MACE-AL-TREE – An extension to structured data

To be able to apply the model to tree structures, we need to solve the following two problems. First, we need to track two different types of errors, namely *labelling* errors and *attachment* errors, and encode them in a meaningful way so that the variational model is able to learn from the data. Second, we need to collect the local predictions from the variational model and translate them back into a tree structure.

Error encoding We need to track *labelling* errors where a directed arc has been assigned the wrong label (Figure 1 (center)), and *attachment* errors where the relation between two nodes has been predicted incorrectly (Figure 1 (right)). We do this by training two separate variational models, one for the labelling decisions and the other one for the attachment relations between nodes.[2]

[2]Combining both into one model would blow up the number of possible labels and thus make the model unnecessarily hard to learn, and would also take too much time for training to be used in an active learning scenario.

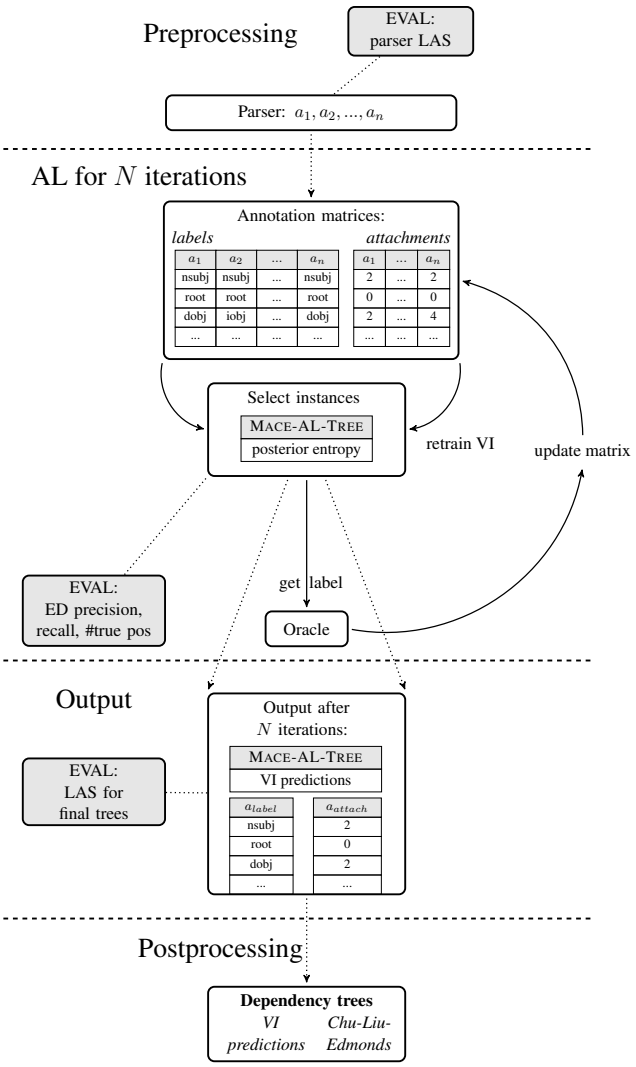


Figure 2: Error detection model with VI and AL (Error Detection (ED) precision: no. of True Positives / total no. of instances selected for error correction; True Positives: no. of instances selected for correction that are true errors; Recall: correctly identified errors / total no. of errors in the corpus).

Let's assume we have a dependency tree forest where each tree consists of a set of nodes V, one for each word in the sentence, and of a set of edge labels R that hold between the nodes. Let $L = l_1, l_2, ..., l_m$ be the set of all possible edge labels. $A = a_1, ..., a_n$ is the set of annotators, in our case the committee of dependency parsers used for obtaining the predicted parse trees. We then train two variational models where the first model aims at learning the reliability of each annotator with regard to the edge label predictions, and the second model learns the reliability of the attachment decisions for each of the parsers. The input to the *labelling* model for MACE-AL-TREE is a matrix with all labels predicted by the annotators (parsers) A_{1-n}, and the MACE-AL-TREE *attachment* model takes a matrix with the head information for each token, extracted from the parser output of the different annotators A_{1-n}.

After preprocessing the data to obtain the parser output trees, we transform the predicted trees as described above and train the variational inference model of MACE-AL-TREE separately on the label and attachment encodings (Figure 2). After training, each model outputs the posterior entropies for its respective decisions about the preprocessed input. We then alternate in extracting the label and attachment decisions with the highest posterior entropy, according to the variational model. The intuition behind this is simple: The labels or attachments with the highest entropy are most likely incorrect.

We either present the selected instances to the human annotator and request the correct annotation or, in a simulation, we extract the correct annotation from the gold tree. In the first setting, the human annotator is shown the sentence with the token in question highlighted in blue while and all the nodes that are predicted as potential heads of the token by any of the parsers are highlighted in red. The annotator

then enters the correct label *and* head for this token. The information is used to update both the label and attachment matrices by randomly selecting one of the annotators and replacing the predictions of that annotator for the instance in question with the new prediction.[3] Note that the update does not necessarily result in an increase in accuracy (or decrease in the number of errors) because the prediction may already have been correct to start with. After updating the annotation matrices, we start over and retrain the variational model on the updated annotations. To save time, we alternate in retraining the models so that each model is retrained after every other iteration. We can repeat this process as many times as we like.

Generating trees Once we are done with error correction, we want to output the trees. However, what we get from the variational model are local decisions for individual labels and edges, and it is not straight-forward to generate connected trees based on these decisions. We test two different methods for generating the output trees. In the first setting, MACE-AL-TREE, we simply use the final predictions of the variational inference model to select the most trustworthy labels and edges and combine those in a final tree. Please note that this does not necessarily result in a fully connected tree, a problem that also affects the output of many local, greedy parsers. Our second approach uses the Chu-Liu-Edmonds (CLE) algorithm (Chu and Liu, 1965; Edmonds, 1967) to select the highest scoring, well-formed tree from a weight matrix, where the weights are based on the votes from the parser committee, weighted by the competence estimates learned by the variational inference model (MACE-AL-TREE-CLE).

Overall workflow Figure 2 illustrates the workflow of our error detection model. During preprocessing, we collect parse predictions from a committee of N dependency parsers. Based on these predictions, we create the two input matrices, one for dependency labels and one for attachments, that we use to train the two variational inference models. Based on the posterior entropy from the variational inference model, we select the next instances to be annotated during active learning. After the instances have been corrected, we alternately retrain the variational inference model: in even iterations we retrain the labeling model and in odd runs the attachment model. This saves time over retraining both models in each iteration. We do, however, update both matrices in each iteration, based on the attachment and label annotations we get from the human annotator (the oracle).

Once we are done with active learning, we collect the annotations for the labels and the attachments and generate the output trees, either based on the predictions from the variational inference model (MACE-AL-TREE) or on the output from the Chu-Liu-Edmonds algorithm (MACE-AL-TREE-CLE).

A key advantage of our model is that, while making use of the feedback from active learning, we do not have to retrain the parsers after each iteration, which would be infeasible due to the time requirements. Instead, we only train the parsers once, for offline preprocessing, before active learning starts.

3 Experiments

In our experiments, we use five different dependency parsers for preprocessing to predict the parse trees. We employ the neural BiLSTM parser (BIST$_{Graph}$) of Kiperwasser and Goldberg (2016) and the RBG parser (Lei et al., 2014), both graph-based, as well as the transition-based IMSTrans parser (Björkelund and Nivre, 2015) and the slightly outdated Malt parser (Nivre et al., 2006).[4] As the fifth model, we use our in-house implementation of the head-selection parser of Zhang et al. (2017) which also employs bidirectional LSTMs (Hochreiter and Schmidhuber, 1997) for learning the feature representations. We chose the parsers so that they cover a range of different parsing algorithms and approaches, as it has been shown before that the parsers' predictions, due to the inductive biases of the algorithms involved, can complement each other (McDonald and Nivre, 2007). All parsers use default settings as reported in the literature. For the BIST$_{Graph}$ parser, we selected the model with the highest LAS on the development set after 10 training iterations.[5] We run the Malt parser with the settings *arc-eager, liblinear* without any further optimisation. For all experiments, we remove the language-specific extended labels and only train the parsers on the universal dependency relations.

[3] Other update strategies such as always updating the predictions of the annotator with the lowest performance, or updating the predictions of all annotators did not yield the same increase in results.

[4] It has been shown that the most important success criterion for ensemble parsing is the diversity of the base parsers rather than model complexity or a high accuracy for the individual parsers (Surdeanu and Manning, 2010).

[5] We started with 30 iterations but noticed no further increase in LAS after the first 10 iterations.

genre	# trees	dev	test	train	# unlabelled sent.
answers	3,488	1,000	2,488	13,134	27,274
email	4,900	1,000	3,900	11,722	1,194,173
newsgroups	2,391	1,000	1,391	14,231	1,000,000
rewiews	3,813	1,000	2,813	12,809	1,965,350
weblogs	2,030	1,000	1,030	14,592	524,834

Table 1: Distribution in the UD-En web treebank (training set does not include data from the respective genre but contains all the remaining treebank sentences *excluding* the ones for this particular genre).

3.1 Error detection with active learning for manually annotated trees

The first experiment focusses on error detection in the German UD treebank which includes newswire, reviews and text from Wikipedia. Our goal is to test our error detection method in a realistic setup where we want to improve the quality of an existing dataset with minimal manual effort.

We run the task as an active learning experiment with a real human annotator in the loop who provides the correct labels for the selected error candidates. The annotator is a trained linguist with experience in linguistic annotation. The annotation task, however, is not easy due to the non-canonicity of the web data which includes many ungrammatical structures and does not conform to standard spelling conventions. Therefore, we asked a second trained linguist to adjudicate the disagreements between the original treebank annotations and the modifications carried out by the first annotator, using MaltEval (Nilsson and Nivre, 2008) to visualise the trees.

For preprocessing, we split the training part of the treebank into two sections and trained the five parsers on each section. Then we used the parsing models to predict the trees for the test set. Splitting the training data has the advantage that we now have 10 different versions of the test set (5 parsers x 2 training sets) that we can use to extract the input matrices for the active learning approach. A systematic and principled evaluation of the impact of data size and the number of annotators for the error detection approach is still outstanding.[6] Here we start with a set of 7 annotators (parsing models) for preprocessing as this setup yielded good results for POS tagging and NER (Rehbein and Ruppenhofer, 2017).[7]

Then we run 200 iterations of active learning where in each iteration the human judge had to assign the correct dependency label and head for the next token selected for annotation. After 200 iterations, we generated the parse trees for the corrected instances using the CLE algorithm weighted by the competence scores from the VI model (see §2.2) and compared them to the original treebank annotations.

The evaluation shows that out of the 200 instances from the test set inspected by the first annotator, the annotator did not agree with the original treebank annotation in 96 cases. Our second annotator agreed with the first judge on 71 of the 96 instances. For 15 instances, the annotator prefered the original annotations and for 8 instances, the second judge would have chosen a label or attachment different from the original treebank and from the first judge. This corresponds to an error detection precision of 35.5% (71/200) which should be considered as a *lower bound* as the instances where the second judge did neither agree with the first annotator nor with the original annotations might also be errors.

3.2 Error detection with AL on out-of-domain data

After validating our approach on the task of identifying errors in manually annotated trees, we now present an AL simulation study where we try to detect errors in *automatically* predicted parse trees. For this, we use the data from the English web treebank (Linguistic Data Consortium release LDC2012T13), originally annotated with constituency trees (Bies, 2012) and automatically converted to universal dependencies. The treebank includes data from five different web genres (reviews, weblogs, answers, emails, newsgroups) and provides us with a good test bed for error detection in a domain adaptation scenario. The simulation study also allows us to systematically evaluate different parameters in a controlled setting, without the need to employ human annotators.

[6] Preliminary experiments on English showed the same results with a smaller number of annotators for some datasets while for others a higher number of annotators was crucial to obtain a high error detection precision.

[7] To obtain a diverse parser committee, we selected the BIST and Dense models trained on the first section of the treebank, the Malt parser (arc-eager, lib-linear) trained on the second section, and both models for the RBG and IMSTrans parsers.

line no.		**answer LAS**	**ED prec**	**email LAS**	**ED prec**	**newsgroup LAS**	**ED prec**	**reviews LAS**	**ED prec**	**weblog LAS**	**ED prec**
	parser					*LAS for individual parsers*					
1	BIST_{Graph}	**85.03**		82.27		82.86		**86.73**		84.73	
2	Dense	84.57		**82.81**		82.43		86.41		84.41	
3	IMSTrans	84.99		82.21		81.94		86.19		84.75	
4	Malt	81.38		78.61		79.18		83.24		79.23	
5	RBG	85.01		82.23		**83.53**		86.41		**84.93**	
	# iterations					*MACE-AL-TREE*					
6	0	86.82	–	84.07	–	**85.72**	–	87.28	–	**86.98**	–
7	100	87.04	85.0	84.26	89.0	86.07	92.0	87.40	49.0	87.33	78.0
8	200	87.27	86.0	84.45	86.5	86.34	80.5	87.48	42.0	87.65	75.5
9	300	87.46	83.0	84.61	83.3	86.64	78.7	87.45	24.3	87.96	73.7
10	400	87.64	80.5	84.77	80.7	86.91	76.7	87.56	29.0	88.27	72.5
11	500	87.84	79.8	84.96	81.8	87.14	73.4	87.68	33.8	88.56	71.2
12	600	88.04	79.0	85.11	80.2	87.31	68.3	87.84	39.2	88.84	69.7
13	700	88.20	76.7	85.28	79.7	87.60	69.1	88.02	44.0	89.13	69.3
14	800	88.37	75.5	85.45	79.7	87.90	70.1	88.17	46.6	89.46	69.7
15	900	88.51	73.4	85.63	79.7	88.13	68.9	88.36	49.7	89.71	67.3
16	1000	88.69	73.1	85.79	79.2	88.41	69.2	88.54	52.7	89.97	67.3
	# iterations					*MACE-AL-TREE-CLE*					
17	0	**86.91**	–	**84.10**	–	85.67	–	87.47	–	86.87	–
18	500	87.86	74.2	84.93	76.8	87.08	72.4	88.19	59.6	88.49	72.8
19	1000	88.67	68.8	85.75	76.4	88.33	68.4	88.74	53.0	89.93	68.8
	# iterations					*Majority vote baseline*					
20	0	86.72	–	84.06	–	85.52	–	**88.05**	–	86.83	–
21	500	87.07	27.0	84.38	29.8	86.01	25.0	88.47	35.0	87.49	30.0
22	1000	87.35	24.4	84.59	24.2	86.44	23.7	88.91	35.8	88.01	26.7

Table 2: Performance for individual parsers and parse combinations based on the predictions of the VI model with AL (MACE-AL-TREE) and with CLE (MACE-AL-TREE-CLE) for generating the trees (simulation study; Labelled attachment score (LAS) and error detection (ED) precision)

We preprocess the data as follows. First, we separate the data according to the five web genres and put all the trees for each genre into a genre-specific test set, pretending that no annotated trees for this genre are available for training. Instead, we use all remaining genres as training data. We further split the available test data for each genre into a development set, using the first 1,000 trees of each genre, and a test set including all the remaining trees (Table 1).

Table 2 (lines 1-5) shows the results for the individual baseline parsers on the different web genres. Please note that we did not optimise the parsers on the data and thus the comparison should be taken with a grain of salt. However, it is interesting to see that there is not one best parser but that the different parsers (excluding the 'vintage' Malt system) all yield results in the same range and the best performing parser varies depending on the dataset.[8]

Then we run an AL simulation for 1,000 iterations, where in each iteration one instance is selected according to our error detection model and the correct label and attachment decision is retrieved from the gold standard. Table 2 (lines 6-16) shows the parsing accuracies after increasing iterations of error correction (from 0-1000 where 0 is the output of MACE-AL-TREE with 0 iterations of active learning).

We first compare the results for MACE-AL-TREE with the ones for the best individual parser and observe an increase in LAS between 0.6% (for reviews) and more than 2% (for newsgroups and weblogs). This increase was obtained *without* any manual error correction. Using MACE-AL-TREE in this particular setup can be seen as a form of ensemble parsing where we use the predictions of the variational inference model to select the edges and labels of the parser output trees to be included in the final tree.

Now we want to evaluate the precision of the error detection method after increasing iterations of error correction via active learning. We calculate *error detection (ED) precision* as the number of error candidates that are real errors, divided by the number of instances selected for error detection.

Table 2 (lines 6-16) shows the LAS and ED precision for the different web genres. In the beginning, ED precision is quite high (between 78-92%) for nearly all genres, with the exception of the reviews.

[8]The results are not directly comparable to the ones from the SANCL shared task where the parsers were trained on the PTB while we use the data from the other web genres as training data.

line no.	*# iterations*	**ED prec**	**LAS**	*line no.*	*# iterations*	**ED prec**	**LAS**
1	0	–	88.8				
2	100	74.0	89.7	7	600	57.8	92.8
3	200	66.0	90.3	8	700	55.3	93.3
4	300	64.3	91.1	9	800	54.4	93.9
5	400	62.7	91.7	10	900	52.7	94.3
6	500	60.0	92.3	11	1000	50.1	94.6

Table 3: Results for error detection on data from the German TiGer treebank for increasing iterations of AL; trees generated with MACE-AL-TREE-CLE (simulation study; ED prec: error detection precision).

Here we start with an ED precision of 49% after 100 iterations. While for the other genres it becomes slightly harder with increasing numbers of iterations to detect new errors and ED precision slowly decreases, for the reviews the picture is rather mixed. After inspecting 1,000 instances we have a higher precision than in the beginning. We can only suspect that this might be an artefact of either idiosyncracies in the data or inconsistencies in the gold annotations.

While precision for error detection is high for all genres, the increase in LAS seems to vary across individual genres. This is because the number of tokens in the test sets also varies. The inspected 1,000 tokens correspond to 2,6% of the answers data, 2.2% for emails, 3,9% for newsgroups, 2,4% for reviews and 4.4% for the weblogs data. This means that we examined between 2.2 to 4.4% of the tokens in the data which resulted in an increase in labelled accuracy (LAS) in the range of 2 to 5%, which seems like a very good trade-off between time investment and reward in terms of data quality.[9]

Next, we compare the two different methods for generating the final parse trees. The first methods simply uses the predictions of the variational inference model (MACE-AL-TREE) and combines the highest-ranked labels and attachments into a tree. Our second method uses the Chu-Liu-Edmonds algorithm to generate well-formed trees (MACE-AL-TREE-CLE) that also allow for non-projectivity. Interestingly, as shown in Table 2 (lines 17-19), we notice only very small differences between the two methods and it is hard to say which method is superior as the differences in results might also be an effect of different initialisations of the variational inference model.[10] This observation is in line with the results from Zhang et al. (2017) who compared the output of their greedy local bi-LSTM parser with the same trees that had undergone additional postprocessing with the CLE algorithm. The authors also report only insignificant improvements for postprocessing German and Czech parser output.

Finally, we want to evaluate the importance of the generative unsupervised model as a component in our error detection model. Surdeanu and Manning (2010) emphasize that the most important success criterion for ensemble parsing is not the complexity of the model but the diversity of the baseline parsers. Therefore it is conceivable that we could get similar results using a simpler model, based merely on the predictions of the parsers *without* the variational inference model. To test this, we run 1,000 iterations of AL but select the next instance based on the entropy in the predictions of the baseline parsers instead of using the posterior entropy from the VI model (Table 2, lines 20-22) and generate the final trees running the CLE algorithm on the unweighted votes of the parser ensemble.

Table 2 shows that for 4 out of 5 genres we get substantially higher results using MACE-AL-TREE. After 1,000 iterations our model shows an increase in LAS that is between 1-2% higher than the one for the simpler model (lines 16, 19 and 22). The only outlier is the reviews subcorpus which also seemed to behave differently from the other genres with regard to error detection precision.[11]

[9] Improvements in LAS over the best individual parser after correcting 1,000 instances: answers 3.6%, emails 2.9%, newsgroup 4.8%, reviews 2.0%, weblogs 5.0%.

[10] As we use the model in an active learning setup where we need to keep the time requirements for training as small as possible, we only initialise the model once. The original model of Hovy et al. (2013) (MACE) allows for a higher number of restarts with random initialisation and then selects the best model.

[11] The lower error detection precision for the reviews in combination with higher LAS (line 22: ED: 35.8%, LAS: 88.91% after 1,000 iterations) is due to the fact that the initial predictions for the reviews based on the majority vote of the baseline parsers yielded a higher accuracy, which means that we start with a smaller amount of errors in the data. As ED precision is based on the overall number of errors in the set, we obtain a lower precision even though the model was able to detect more errors than MACE-AL-TREE.

core args		*coord & modifiers*		*clause structure*		*other*	
SB	42	CJ	28	RC	12	CM, CC,	
OA	23	CD	2	PAR	12	DM,	
PD	11	APP	8	OC	10	NK,	28
DA	8	MO	58	RE	6	NMC	
AG	2	MNR	23	RS	2	PNC	
PG	2	OP	11	ROOT	11	SVP	

head type	#	*head type*	#
modifiers	128	clauses	43
punctuation	77	core args	43
coordination	39	apposition	10
noun kernel	19	other	26

Table 4: Distribution of labelling errors (left table) and attachment errors (right table) in the German TiGer data found in 1000 iterations of AL error correction.

3.3 Error detection with AL on in-domain data

We showed that our approach yields high error detection precision in out-of-domain scenarios. We now want to test whether our method is also suited for in-domain settings where we start with a higher parser output quality, which means that errors in the data might be harder to find.

Again, we run a simulation study, but this time on German newspaper text from the TiGer treebank (Brants et al., 2002). We use the version from the SPRML 2014 shared task (Seddah et al., 2014) with 40,472 sentences in the training set and 5000 sentences for testing. We train five parsers[12] on the training portion of the data and predict parse trees for the automatically tagged test file provided by the shared task organisers. Parsing accuracies for the individual parsers on our testset with automatically predicted POS and morphological tags are in the range of 84.2-90.3% UAS and of 81.1-87.6% LAS.

We then run an AL simulation on the first 500 sentences in the testset, with 1000 iterations of active learning. In each iteration we select one instance and replace its head and label with the gold information. Again, it is not guaranteed that each modification will improve results as we might select instances which, according to the VI model, are already predicted correctly. Table 3 shows results for increasing iterations of error correction with AL, and error detection (ED) precision and LAS.

We can observe the same trend on the German in-domain data as we saw for the English out-of-domain data. LAS for the generated output trees based on the predictions of the parser ensemble with CLE (Table 3, line 1), weighted by the competence score from the VI model, is significantly higher with 88.9% than the score for the best individual parser (87.6% LAS). Looking at increasing numbers of AL iterations (lines 2-11), we also see a high error detection (ED) precision, starting with 70% and slowly decreasing as it gets harder to find new errors. After 1000 iterations, we still have an error detection precision of $> 50\%$, meaning that every second instance that we inspect is a real error. In terms of accuracy, running 1000 iterations of error correction on the TiGer subcorpus translates into an increase in LAS from 88.8% to 94.6% while keeping the number of instances for manual correction reasonably low.

3.4 Error analysis

Having established that our model is able to detect errors in manually and automatically annotated parse trees on in-domain and out-of-domain data, we would now like to learn more about the types of errors our model is able to detect. We thus evaluate the output trees generated from the German newspaper corpus (Section 3.3) *before* and *after* running 1000 iterations of AL for error correction.

Overall, our error correction resulted in 385 changed attachment decisions and in 298 modified dependency labels. In 167 cases, both label and attachment have been changed. Looking only at the label errors, we can see that the errors are distributed over 25 different dependency labels. The most frequent ones concern labels for core arguments (subject (SB), direct and indirect object (OA/DA), predicate (PD) and genitive attributes (AG/PG)) that are, due to the semi-free German word order and ambiguity arising through case syncretism, one of the major sources for parsing errors.

Other frequent labelling errors include the distinction between verb and noun attachment for PPs (modifier (MO)/noun modifier to the right (MNR)). The distinction between modifying (MO) and obligatory PPs (OP) is another frequent error that the model was able to detect. Our model also finds errors concerning coordination (CJ, CD) and clause structure, such as relative clauses (RC), parenthesis (PAR), clausal objects (OC), repeated element (RE) and reported speech (RS).

[12]For German, we used the IMSTrans parser, the head-selection parser and the BIST$_{Graph}$ parser that we also used in the previous set of experiments. In addition, we also use the transition-based biLSTM parser of (Kiperwasser and Goldberg, 2016) (BIST$_{Trans}$) and the RBG parser (Lei et al., 2014).

# unlabelled sentences	**LAS answer**	**LAS email**	**LAS newsgroup**	**LAS reviews**	**LAS weblog**
Results without self-training (best individual parser)					
	85.03	82.81	83.53	86.73	84.93
Baseline 1: self-training (best individual parser)					
5,000	85.11	83.40	83.23	87.19	85.14
10,000	85.33	83.25	84.06	87.06	85.32
15,000	85.43	83.37	83.62	87.38	84.82
20,000	85.26	83.25	82.52	87.01	85.27
25,000	–	83.46	82.56	87.50	85.79
Baseline 2: self-training (agreement betw. 2 parsers)					
5,000	85.36	82.58	83.38	87.33	85.80
10,000	–	82.65	82.56	87.28	86.22
15,000	–	82.53	83.34	87.31	85.57
20,000	–	83.16	83.64	87.62	85.26
25,000	–	83.09	82.82	87.74	85.04
MACE-AL-TREE *(with self-training)*					
5,000	85.77	83.07	83.93	87.39	86.27
10,000	85.89	83.71	84.25	87.82	86.28
15,000	85.74	83.85	83.83	87.54	86.08
20,000	86.09	83.51	84.41	87.93	86.26
25,000	–	**84.11**	85.09	**88.04**	86.16
MACE-AL-TREE *(without self-training)*					
	86.82	84.07	**85.72**	87.28	**86.98**

Table 5: Baselines and results for self-training on the different web genres.

Looking at the attachment errors that have been detected by our model (Table 4, right), we find that the most frequent head types that have been re-attached are modifiers (MO, MNR). This not only includes PPs but also adverbal and adjectival modification. The next frequent error type are incorrectly attached punctuations. This, however, is not an error type we are concerned with and it might make sense to exclude punctuation from the error correction.[13] More interesting attachment errors include coordination (CJ, CD), parenthesis (PAR), apposition (APP) and again the core arguments (OA, AG, DA, SB, PD).

This shows that our model is not biased toward a small set of specific error types but is able to detect frequent parsing errors that are well-known from the literature (Kummerfeld et al., 2012).

3.5 Domain adaptation with self-training

In the final set of experiments we want to test whether we can use our method to automatically correct parser output for self-training, to improve parsing accuracy for out-of-domain data. As before, we use the English web treebank, split into different genres as described above. Again, we pretend that no annotated training data for the different genres are available. This time, however, we make use of the supplementary raw text data for the five web genres (Table 1) that were provided for the SANCL shared task (Petrov and McDonald, 2012). The data is segmented into sentences and pre-tokenised. For preprocessing, we use UDPipe (Straka and Straková, 2017) with the pretrained models from the CoNLL-2017 shared task.[14] We also remove unlabelled sentences with a length > 60 tokens.[15]

We parse the additional data using the following settings. As our first baseline, we use the parser that achieved the best LAS on this particular genre for preprocessing. For our second baseline, we apply the best two parsers for each genre to parse the raw text and then select only those trees for self-training where the two parsers agree (excluding punctuation). In the last setting, we use MACE-AL-TREE (but without active learning) to generate the parse trees for each bin of automatically parsed text, based on the predictions of the parser ensemble.

We then use the output of each setting and combine the bins with the original training data for each genre, resulting in new training sets of increasing sizes. Then we train the BIST$_{Graph}$ parser on the new data and select the best model, based on the results on the development set after 10 iterations of training.

[13] In the original constituency-based TiGer treebank, all punctuation marks are attached to a virtual root node and have been reattached during the conversion to dependencies. This results in many arbitrary attachment decisions which makes it hard for the parser to learn these attachments.

[14] The models are available from `https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1990`.

[15] For the answers where we only have 27,274 raw sentences, this results in only 4 samples after removing long sentences, and only one sample for the agreement-based self-training (baseline 2).

Table 5 reports results for the three different settings. Looking at the self-training results where we retrain the BIST$_{Graph}$ parser on the predictions of the best individual parser, we can see mostly small improvements but sometimes also a decrease in LAS. The results for baseline 2 are also a bit mixed. When adding only 5,000 additional sentences, we obtain slightly higher results as compared to baseline 1. When adding more data, however, it shows that this improvement is not stable and sometimes results are lower than for the first baseline. A possible explanation is that the agreement-based selection strategy favors shorter sentences (as those are the ones where the two parsers tend to agree more often), and that we thus only add less complex structures that are not very informative for the parser. In addition, we introduce a bias in the training set that also might have a negative effect on results.

The third setting, however, clearly outperforms our baselines and shows a significant increase in parsing accuracy. Nonetheless, when comparing the results for MACE-AL-TREE *with* self-training to the results *without* self-training we see that our self-training approach fails to beat the results obtained on the original data on three web genres. We only observe improved results on the emails and on the reviews. These two genres are also the ones that have the largest amount of unknown words which might explain why self-training is more promising for emails and reviews than for the other three genres, as Reichart and Rappoport (2007) have shown that the number of unknown words can be a good indicator for the potential benefit from self-training.

4 Related work

Most studies on error detection in treebanks have focussed on finding errors in manual annotations (Dickinson and Meurers, 2003; Ule and Simov, 2004; van Noord, 2004; Dickinson and Meurers, 2005; Agrawal et al., 2013). Dickinson and Meurers (2003; 2005) proposed the use of *variation n-grams* to detect inconsistencies in manually annotated constituency treebanks and Boyd et al. (2008) extend this line of work to dependency trees. This approach, however, only works for manually annotated treebanks while for automatic parses where the errors are consistent and systematic, looking for variation in the predictions will not be very helpful. Volokh and Neumann (2011) address the latter problem and use two different parsing models to reproduce the annotations in the *training data*. They consider an instance to be erroneous if the two parsers agree in their predictions for the attachment of the token but disagree with the gold standard. The authors report a high precision for automatic error correction. However, their method only addresses attachment errors but does not handle label errors.

Work on predicting *automatic* parser errors includes Dickinson and Smith (2011) who develop a grammar-based method for error detection. They use ngrams extracted from automatic parses and weigh them by comparing them to the rules in a small gold grammar. Follow-up work (Dickinson and Smith, 2017) builds on this approach and uses parse simulations to extend the training grammar.

Relevant for our work are also studies on parse accuracy prediction. Ravi et al. (2008) predict parser performance for data from new domains. Our error detection method can be applied to automatic parses in a domain adaptation setting and can not only predict the accuracy of the parses but also improve the accuracy of the parser output. Our work can be seen as a combination of semi-supervised and ensemble-based methods and is thus similar in spirit to Søgaard and Rishøj (2010) who use tri-training in combination with self-training for dependency parsing and report an increase in labelled attachment score (LAS) for different languages in the range of 1.4 to 2.6%. Their self-training experiments, however, were based on a much larger amount of additional data (100,000 sentences vs. up to 25,000 sentences in our self-training experiments). It would be interesting to see whether more unlabelled data can further improve results, as suggested by the results for MACE-AL-TREE with self-training.

5 Conclusions

We presented a method for error detection in treebanks, based on the combination of (i) ensemble parsing, (ii) an unsupervised generative inference model, and (iii) human guidance from active learning. We validated our model on the task of error detection in manual and automatic annotations using active learning and showed that we can identify parse errors with precisions in the range of 35% to over 90%. Our approach can also be used to improve the quality of automatic annotations on out-of-domain data, without any human feedback, where we obtained gains in accuracy of up to 2% and more.

References

Bhasha Agrawal, Rahul Agarwal, Samar Husain, and Dipti M. Sharma. 2013. An automatic approach to treebank error detection using a dependency parser. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part I*, CICLing'13, pages 294–303.

Ann et al. Bies. 2012. English Web Treebank LDC2012T13.

Anders Björkelund and Joakim Nivre. 2015. Non-deterministic oracles for unrestricted non-projective transition-based dependency parsing. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 76–86, Bilbao, Spain, July. Association for Computational Linguistics.

Adriane Boyd, Markus Dickinson, and Detmar Meurers. 2008. On detecting errors in dependency treebanks. *Research on Language and Computation*, 6:113–137.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, TLT'02, pages 24–41.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On shortest arborescence of a directed graph. *Scientia Sinica*, 14(10):1396–1400.

Markus Dickinson and Detmar Meurers. 2003. Detecting inconsistencies in treebanks. In *The Second Workshop on Treebanks and Linguistic Theories*, TLT.

Markus Dickinson and W. Detmar Meurers. 2005. Prune Diseased Branches to Get Healthy Trees! How to Find Erroneous Local Trees in a Treebank and Why It Matters. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories*, TLT 2005, pages 41–52, Barcelona, Spain.

Markus Dickinson and Amber Smith. 2011. Detecting dependency parse errors with minimal resources. In *Proceedings of the 12th International Conference on Parsing Technologies*, IWPT, pages 241–252.

Markus Dickinson and Amber Smith. 2017. Simulating dependencies to improve parse error detection. In *Proceedings of the 15th International Workshop on Treebanks and Linguistic Theories*, TLT15, pages 76–88.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(4):233–240.

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

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT'13, pages 1120–1130, Atlanta, Georgia, USA.

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

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the Wall Street Corral: An empirical investigation of error types in parser output. In *The 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1048–1059.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi S. Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1381–1391.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL'2007, pages 122–131.

Jens Nilsson and Joakim Nivre. 2008. MaltEval: An evaluation and visualization tool for dependency parsing. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*, LREC'08, Marrakech, Morocco, may. European Language Resources Association (ELRA).

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Linguistic Resources and Evaluation*, LREC'2006, pages 2216–2219.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL).

Sujith Ravi, Kevin Knight, and Radu Soricut. 2008. Automatic prediction of parser accuracy. In *Empirical Methods in Natural Language Processing*, EMNLP'08, pages 887–1575.

Ines Rehbein and Josef Ruppenhofer. 2017. Detecting annotation noise in automatically labelled data. In *Proceedings of the 55th annual meeting of the Association for Computational Linguistics*, ACL'17, Vancouver, Canada.

Roi Reichart and Ari Rappoport. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *The 45th Annual Meeting of the Association of Computational Linguistics*, pages 616–623.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the SPMRL 2014 shared task on parsing morphologically-rich languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 103–109, Dublin, Ireland, August. Dublin City University.

Burr Settles. 2009. Active learning literature survey. Computer sciences technical report, University of Wisconsin–Madison.

Anders Søgaard and Christian Rishøj. 2010. Semi-supervised dependency parsing using generalized tri-training. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1065–1073.

Milan Straka and Jana Straková. 2017. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Proceedings of the North American Chapter of the Association for Computational Linguistics Conference (NAACL-2010)*, Los Angeles, CA, June.

Tylman Ule and Kiril Simov. 2004. Unexpected productions may well be errors. In *The Fourth International Conference on Language Resources and Evaluation*, LREC'04.

Gertjan van Noord. 2004. Error mining for wide-coverage grammar engineering. In *The 42nd Annual Meeting of the Association for Computational Linguistics*, pages 446–453.

Alexander Volokh and Günter Neumann. 2011. Automatic detection and correction of errors in dependency treebanks. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 346–350.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, EACL'17, pages 665–676.

Two Local Models for Neural Constituent Parsing

Zhiyang Teng and **Yue Zhang**
Singapore University of Technology and Design
zhiyang_teng@mymail.sutd.edu.sg
yue_zhang@sutd.edu.sg

Non-local features have been exploited by syntactic parsers for capturing dependencies between sub output structures. Such features have been a key to the success of state-of-the-art statistical parsers. With the rise of deep learning, however, it has been shown that local output decisions can give highly competitive accuracies, thanks to the power of dense neural input representations that embody global syntactic information. We investigate two conceptually simple local neural models for constituent parsing, which make local decisions to constituent spans and CFG rules, respectively. Consistent with previous findings along the line, our best model gives highly competitive results, achieving the labeled bracketing F1 scores of 92.4% on PTB and 87.3% on CTB 5.1.

1 Introduction

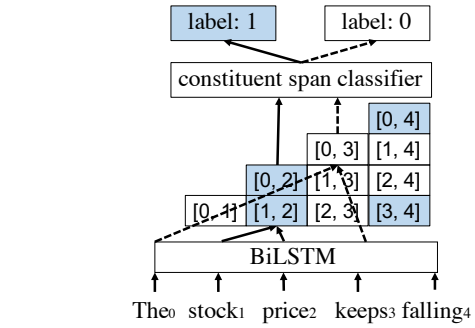
Non-local features have been shown crucial for statistical parsing (Huang, 2008a; Zhang and Nivre, 2011). For dependency parsing, High-order dynamic programs (Koo and Collins, 2010), integer linear programming (Martins et al., 2010) and dual decomposition (Koo et al., 2010) techniques have been exploited by graph-based parser to integrate non-local features. Transition-based parsers (Nivre, 2003; Nivre, 2008; Zhang and Nivre, 2011; Bohnet, 2010; Huang et al., 2012) are also known for leveraging non-local features for achieving high accuracies. For most state-of-the-art statistical parsers, a global training objective over the entire parse tree has been defined to avoid label bias (Lafferty et al., 2001).

For neural parsing, on the other hand, local models have been shown to give highly competitive accuracies (Cross and Huang, 2016b; Stern et al., 2017) as compared to those that employ long-range features (Watanabe and Sumita, 2015; Zhou et al., 2015; Andor et al., 2016; Durrett and Klein, 2015). Highly local features have been used in recent state-of-the-art models (Stern et al., 2017; Dozat and Manning, 2016; Shi et al., 2017). In particular, Dozat and Manning (2016) show that a locally trained arc-factored model can give the best reported accuracies on dependency parsing. The surprising result has been largely attributed to the representation power of long short-term memory (LSTM) encoders (Kiperwasser and Goldberg, 2016).

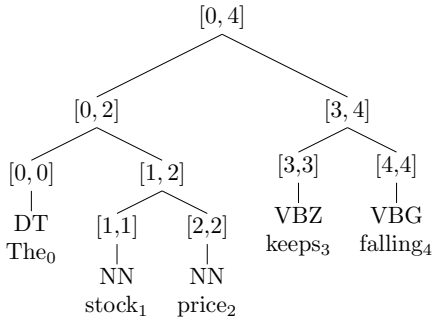
An interesting research question is to what extent the encoding power can be leveraged for constituent parsing. We investigate the problem by building a chart-based model that is local to unlabeled constituent spans (Abney, 1991) and CFG-rules, which have been explored by early PCFG models (Collins, 2003; Klein and Manning, 2003). In particular, our models first predict unlabeled CFG trees leveraging bi-affine modelling (Dozat and Manning, 2016). Then, constituent labels are assigned on unlabeled trees by using a tree-LSTM to encode the syntactic structure, and a LSTM decoder for yielding label sequences on each node, which can include unary rules.

Experiments show that our conceptually simple models give highly competitive performances compared with the state-of-the-art. Our best models give labeled bracketing F1 scores of 92.4% on PTB and 87.3% on CTB 5.1 test sets, without reranking, ensembling and external parses. We release our code at

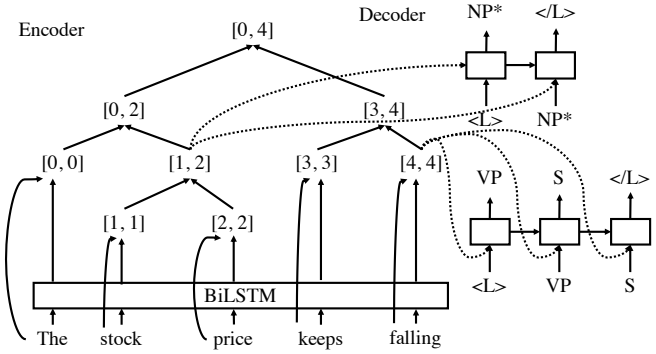
Proceedings of the 27th International Conference on Computational Linguistics, pages 119–132
Santa Fe, New Mexico, USA, August 20-26, 2018.



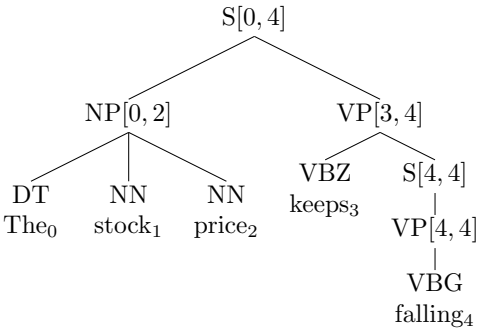
(a) BiLSTM based constituent span classifier.



(b) unlabeled binarized parse tree.



(c) Label generator for two example spans. **NP*** is an intermediate constituent label.



(d) Final example parse tree.

Figure 1: An example workflow of our parsers for the sentence "The stock price keeps falling". We annotate every non-terminal span with its covered span range. Figure 1a shows constituent span classifiers making 0/1 decisions for all possible spans. Based on the local classification probabilities, we obtain an unlabeled binarized parse tree (Figure 1b) using binary CKY parsing algorithms. We then hierarchically generate labels for each span (Figure 1c) using encoder-decoder models. Figure 1d shows the final output parse tree after debinarization.

`https://github.com/zeeeyang/two-local-neural-conparsers`.

2 Model

Our models consist of an unlabeled binarized tree parser and a label generator. Figure 1 shows a running example of our parsing model. The unlabeled parser (Figure 1a, 1b) learns an unlabeled parse tree using simple BiLSTM encoders (Hochreiter and Schmidhuber, 1997). The label generator (Figure 1c, 1d) predicts constituent labels for each span in the unlabeled tree using tree-LSTM models.

In particular, we design two different classification models for unlabeled parsing: the **span model** (Section 2.1) and the **rule model** (Section 2.2). The span model identifies the probability of an arbitrary span being a constituent span. For example, the span $[1, 2]$ in Figure 1a belongs to the correct parse tree (Figure 1d). Ideally, our model assigns a high probability to this span. In contrast, the span $[0, 3]$ is not a valid constituent span and our model labels it with 0. Different from the span model, the rule model considers the probability $P([i, j] \rightarrow [i, k][k + 1, j]|S)$ for the production rule that the span $[i, j]$ is composed by two children spans $[i, k]$ and $[k + 1, j]$, where $i \leq k < j$. For example, in Figure 1a, the rule model assigns high probability to the rule $[0, 2] \rightarrow [0, 0][1, 2]$ instead of the rule $[0, 2] \rightarrow [0, 1][2, 2]$. Given the local probabilities, we use CKY algorithm to find the unlabeled binarized parses.

The label generator encodes a binarized unlabeled tree and to predict constituent labels for every span. The encoder is a binary tree-LSTM (Tai et al., 2015; Zhu et al., 2015), which recursively composes the representation vectors for tree nodes bottom-up. Based on the representation vector of a constituent span, a LSTM decoder (Cho et al., 2014; Sutskever et al., 2014) generates chains of constituent labels, which

can represent unary rules. For example, the decoder outputs "VP →S→ </L>" for the span [4, 4] and "NP→ </L>" for the span [0,2] in Figure 1c where </L> is a stopping symbol.

2.1 Span Model

Given an unlabeled binarized tree T_{ub} for the sentence S, $S = w_0, w_1 \dots w_{n-1}$, the span model trains a neural network model $P(Y_{[i,j]}|S, \Theta)$ to distinguish constituent spans from non-constituent spans, where $0 \leq i \leq n-2$, $1 \leq j < n$, $i < j$. $Y_{[i,j]} = 1$ indicates the span [i, j] is a constituent span ($[i, j] \in T_{ub}$), and $Y_{[i,j]} = 0$ for otherwise, Θ are model parameters. We do not model spans with length 1 since the span $[i, i]$ always belongs to T_{ub}.

Network Structure. Figure 2a shows the neural network structures for the binary classification model. In the bottom, a bidirectional LSTM layer encodes the input sentence to extract non-local features. In particular, we append a starting symbol <s> and an ending symbol </s> to the left-to-right LSTM and the right-to-left LSTM, respectively. We denote the output hidden vectors of the left-to-right LSTM and the right-to-left LSTM for $w_0, w_1, \dots, w_{n-1}$ is $\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_n$ and $\mathbf{r}_0, \mathbf{r}_1, \dots, \mathbf{r}_{n-1}$, respectively. We obtain the representation vector $\mathbf{v}[i, j]$ of the span $[i, j]$ by simply concatenating the bidirectional output vectors at the input word i and the input word j,

$$\mathbf{v}[i, j] = [\mathbf{f}_{i+1}; \mathbf{r}_i; \mathbf{f}_{j+1}; \mathbf{r}_j]. \tag{1}$$

$\mathbf{v}[i, j]$ is then passed through a nonlinear transformation layer and the probability distribution $P(Y_{[i,j]}|S, \Theta)$ is given by

$$\mathbf{o}[i, j] = \tanh(\mathbf{W}_o\mathbf{v}[i, j] + \mathbf{b}_o), \;\; \mathbf{u}[i, j] = \mathbf{W}_u\mathbf{o}[i, j] + \mathbf{b}_u, \;\; P(Y_{[i,j]}|S, \Theta) = \text{softmax}(\mathbf{u}[i, j]), \tag{2}$$

where $\mathbf{W}_o, \mathbf{b}_o, \mathbf{W}_u$ and $\mathbf{b}_u$ are model parameters.

Input Representation. Words and part-of-speech (POS) tags are integrated to obtain the input representation vectors. Given a word w, its corresponding characters $c_0, \dots, c_{|w|-1}$ and POS tag t, first, we obtain the word embedding $\mathbf{E}^w_{word}$, character embeddings $\mathbf{E}^{c_0}_{char}, \dots, \mathbf{E}^{c_{|w|-1}}_{char}$, and POS tag embedding $\mathbf{E}^t_{pos}$ using lookup operations. Then a bidirectional LSTM is used to extract character-level features. Suppose that the last output vectors of the left-to-right and right-to-left LSTMs are $\mathbf{h}^f_{char}$ and $\mathbf{h}^r_{char}$, respectively. The final input vector $\mathbf{x}_{input}$ is given by

$$\mathbf{x}_{char} = \tanh(\mathbf{W}^l_{char}\mathbf{h}^l_{char} + \mathbf{W}^r_{char}\mathbf{h}^r_{char} + \mathbf{b}_{char}), \;\; \mathbf{x}_{input} = [\mathbf{E}^w_{word} + \mathbf{x}_{char}; \mathbf{E}^t_{pos}], \tag{3}$$

where $\mathbf{W}^l_{char}$, $\mathbf{W}^r_{char}$ and $\mathbf{b}_{char}$ are model parameters.

Training objective. The training objective is to maximize the probabilities of $P(Y_{[i,j]} = 1|S, \Theta)$ for spans $[i, j] \in T_{ub}$ and minimize the probabilities of $P(Y_{[i,j]} = 1|S, \Theta)$ for spans $[i, j] \notin T_{ub}$ at the same time. Formally, the training loss for binary span classification $\mathcal{L}_{\text{binary}}$ is given by

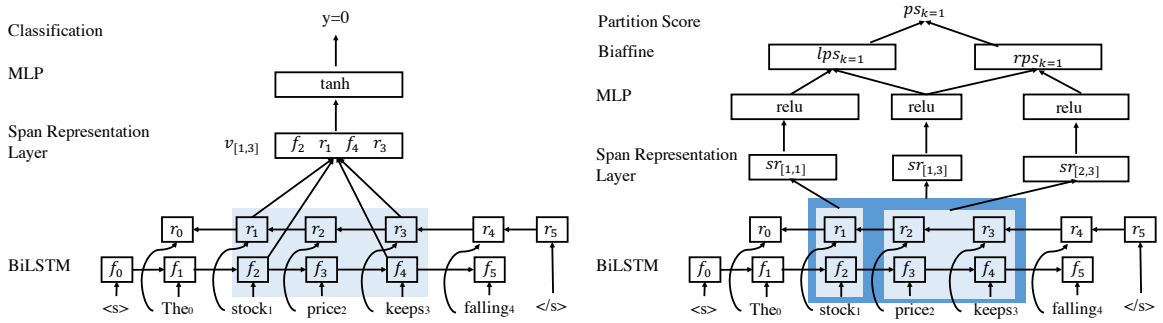
$$\mathcal{L}_{\text{binary}} = -\sum_{[i,j]\in T_{ub}} \log P(Y_{[i,j]} = 1|S, \Theta) - \sum_{[i,j]\notin T_{ub}} \log P(Y_{[i,j]} = 0|S, \Theta), \\ (0 \leq i \leq n-2, 1 \leq j < n, i < j) \tag{4}$$

For a sentence with length n, there are $\frac{n(n-1)}{2}$ terms in total in Eq 4.

Neural CKY algorithm. The unlabeled production probability for the rule $r : [i, j] \rightarrow [i, k][k + 1, j]$ given by the binary classification model is,

$$P(r|S, \Theta) = P(Y_{[i,k]} = 1|S, \Theta)P(Y_{[k+1,j]} = 1|S, \Theta).$$

During decoding, we find the optimal parse tree T^*_{ub} using the CKY algorithm. Note that our CKY algorithm is different from the standard CKY algorithm mainly in that there is no explicit phrase rule probabilities being involved. Hence our model can be regarded as a zero-order constituent tree model, which is the most *local*. All structural relations in a constituent tree must be implicitly captured by the BiLSTM encoder over the sentence alone.



(a) Span model. (b) Rule model.

Figure 2: Neural network structures for span and rule models using BiLSTM encoders.

Multi-class Span Classification Model. The previous model preforms binary classifications to identify constituent spans. In this way, the classification model only captures the existence of constituent labels but does not leverage constituent label type information. In order to incorporate the syntactic label information into the span model, we use a multi-class classification model $P(Y_{[i,j]} = c|S, \Theta)$ to describe the probability that c is a constituent label for span $[i, j]$. The network structure is the same as the binary span classification model except the last layer. For the last layer, given $\mathbf{o}_{[i,j]}$ in Eq 2, $P(Y_{[i,j]} = c|S, \Theta)$ is calculated by,

$$\mathbf{m}[i, j] = \mathbf{W}_m \mathbf{o}[i, j] + \mathbf{b}_m, \ P(Y_{[i,j]} = c|S, \Theta) = \text{softmax}(\mathbf{m}[i, j])_{[c]}. \tag{5}$$

Here $\mathbf{W}_m, \mathbf{b}_m, \mathbf{W}_m$ and $\mathbf{b}_m$ are model parameters. The subscript $[c]$ is to pick the probability for the label c. The training loss is,

$$\mathcal{L}_{\text{multi}} = -\sum_{[i,j]\in T_{ub}} \sum_{c\in[i,j], c\neq</L>} \log P(Y_{[i,j]} = c|S, \Theta) - \sum_{[i,j]\notin T_{ub}} \log P(Y_{[i,j]} = </L>|S, \Theta), \tag{6}$$
$$(0 \leq i \leq n-2, 1 \leq j < n, i < j)$$

Note that there is an additional sum inside the first term in Eq 6, which is different from the first term in Eq 4. This is to say that we treat all constituent labels equally of a unary chain. For example, suppose there is a unary chain S→VP in span [4,4]. For this span, we hypothesize that both labels are plausible answers and pay equal attentions to VP and S during training. For the second term in Eq 6, we maximize the probability of the ending label for non-constituent spans.

For decoding, we transform the multi-class probability distribution into a binary probability distribution by using,

$$P(Y_{[i,j]} = 1|S, \Theta) = \sum_{c, c\neq</L>} P(Y_{[i,j]} = c|S, \Theta), \ \ P(Y_{[i,j]} = 0|S, \Theta) = P(Y_{[i,j]} = </L>|S, \Theta)$$

In this way, the probability of a span being a constituent span takes all possible syntactic labels into considerations.

2.2 Rule Model

The rule model directly calculates the probabilities of all possible splitting points k $(i \leq k < j)$ for the span $[i, j]$. Suppose the partition score of splitting point k is ps_k. The unlabeled production probability for the rule $r : [i, j] \rightarrow [i, k][k + 1, j]$ is given by a softmax distribution,

$$P([i, j] \rightarrow [i, k][k + 1, j]|S, \Theta) = \frac{\exp(ps_k)}{\sum_{k'=i}^{j-1} \exp(ps_{k'})}.$$

The training objective is to minimize the log probability loss of all unlabeled production rules.

$$\mathcal{L}_{\text{rule}} = -\sum_{r \in T_{ub}} \log P(r : [i,j] \rightarrow [i,k][k+1,j] | S, \Theta)$$

The decoding algorithm is the standard CKY algorithm, which we omit here. The rule model can be regarded as a first-order constituent model, with the probability of each phrase rule being modeled. However, unlike structured learning algorithms (Finkel et al., 2008; Carreras et al., 2008), which use a global score for each tree, our model learns each production rule probability individually. Such local learning has traditionally been found subjective to label bias (Lafferty et al., 2001). Our model relies on input representations solely for resolving this issue.

Span Representation. Figure 2b shows one possible network architecture for the rule model by taking the partition point $k = 1$ for the span $[1, 3]$ as an example. The BiLSTM encoder layer in the bottom is the same as that of the previous span classification model. We obtain the span representation vectors using difference vectors (Wang and Chang, 2016; Cross and Huang, 2016b). Formally, the span representation vector $\mathbf{sr}[i,j]$ is given by,

$$\begin{aligned} \mathbf{s}[i,j] &= [\mathbf{f}_{j+1} - \mathbf{f}_i; \mathbf{r}_i - \mathbf{r}_{j+1}], \\ \mathbf{sr}[i,j] &= [\mathbf{s}[0,i-1]; \mathbf{s}[i,j]; \mathbf{s}[j+1,n-1]]. \end{aligned} \tag{7}$$

We first combine the difference vectors $(\mathbf{f}_{j+1} - \mathbf{f}_i)$ and $(\mathbf{r}_i - \mathbf{r}_{j+1})$ to obtain a simple span representation vector $\mathbf{s}[i,j]$. In order to take more contextual information such as $\mathbf{f}_p$ where $p > j+1$ and $\mathbf{r}_q$ where $q < i$, we concatenate $\mathbf{s}[0,i-1]$, $\mathbf{s}[i,j]$, and $\mathbf{s}[j+1,n-1]$ to produce the final span representation vector $\mathbf{sr}[i,j]$. We then transform $\mathbf{sr}[i,j]$ to an output vector $\mathbf{r}[i,j]$ using an activation function ϕ,

$$\mathbf{r}[i,j] = \phi(\mathbf{W}_r^M \mathbf{sr}[i,j] + \mathbf{b}_r^M), \tag{8}$$

where $\mathbf{W}_r^M$ and $\mathbf{b}_r^M$ and model parameters, and M is a parameter set index. We use separate parameters for the nonlinear transforming layer. $M \in \{P, L, R\}$ are for the parent span $[i,j]$, the left child span $[i,k]$ and the right child span $[k+1,j]$, respectively.

After obtaining the span representation vectors, we use these vectors to calculate the partition score ps_k. In particular, we investigate two scoring methods.

Linear Model. In the linear model, the partition score is calculated by a linear affine transformation. For the splitting point k,

$$ps_k = \mathbf{w}_{ll,k}^T \mathbf{r}[i,k] + \mathbf{w}_{lr,k}^T \mathbf{r}[k+1,j] + b_{ll,k}$$

where $\mathbf{w}_{ll,k}^T$ and $\mathbf{w}_{ll,k}^T$ are two vectors, and $b_{ll,k}$ is a size 1 parameter.

Biaffine model. Since the possible splitting points for spans are varied with the length of span, we also try a biaffine scoring model (as shown in Figure 2b), which is good at handling variable-sized classification problems (Dozat and Manning, 2016; Ma and Hovy, 2017). The biaffine model produces the score lps_k between the parent span $[i,j]$ and the left child span $[i,k]$ using a biaffine scorer

$$lps_k = (\mathbf{r}[i,j] \oplus 1)^T \mathbf{W}_{pl} (\mathbf{r}[i,k] \oplus 1) \tag{9}$$

where $\mathbf{W}_{pl}$ is model parameters. Similarly, we calculate the score rps_k between the parent span $[i,j]$ and the right child span $[k+1,j]$ using $\mathbf{W}_{pr}$ and $\mathbf{b}_{pr}$ as parameters. The overall partition score ps_k is therefore given by

$$ps_k = lps_k + rps_k.$$

2.3 Label Generator

Lexicalized Tree-LSTM Encoder. Shown in Figure 1c, we use lexicalized tree LSTM (Teng and Zhang, 2016) for encoding, which shows good representation abilities for unlabeled trees. The encoder first propagates lexical information from two children spans to their parent using a lexical gate, then it produces the representation vectors of the parent span by composing the vectors of children spans using a

binarized tree-LSTM (Tai et al., 2015; Zhu et al., 2015). Formally, the lexical vector $\mathbf{tx}[i,j]$ for the span $[i,j]$ with the partition point at k is defined by:

$$\mathbf{i}^{lex}_{[i,j]} = \sigma(\mathbf{W}^{lex}_{l}\mathbf{tx}[i,k] + \mathbf{W}^{lex}_{r}\mathbf{tx}[k+1,j] + \mathbf{W}^{lex}_{lh}\mathbf{h}_{[i,k]} + \mathbf{W}^{lex}_{rh}\mathbf{h}_{[k+1,j]} + \mathbf{b}_{lex})$$
$$\mathbf{tx}[i,j] = \mathbf{i}^{lex}_{[i,j]} \odot \mathbf{tx}[i,k] + (\mathbf{1.0} - \mathbf{i}^{lex}_{[i,j]}) \odot \mathbf{tx}[k+1,j],$$

where $\mathbf{W}^{lex}_{l}$, $\mathbf{W}^{lex}_{r}$ and $\mathbf{b}_{lex}$ are model parameters, $\odot$ is element-wise multiplication and σ is the logistic function. The lexical vector $\mathbf{tx}[i,i]$ for the leaf node i is the concatenate of the output vectors of the BiLSTM encoder and the input representation $\mathbf{x}_{\textbf{input}}[i]$ (Eq 3), as shown in Figure 1c.

The output state vector $\mathbf{h}[i,j]$ of the span $[i,j]$ given by a binary tree LSTM encoder is,

$$\mathbf{i}_p = \sigma(\mathbf{W}_1\mathbf{tx}_p + \mathbf{W}_2\mathbf{h}_l + \mathbf{W}_3\mathbf{c}_l + \mathbf{W}_4\mathbf{h}_r + \mathbf{W}_5\mathbf{c}_r + \mathbf{b}_1),$$
$$\mathbf{f}^l_p = \sigma(\mathbf{W}_6\mathbf{tx}_p + \mathbf{W}_7\mathbf{h}_l + \mathbf{W}_8\mathbf{c}_l + \mathbf{W}_9\mathbf{h}_r + \mathbf{W}_{10}\mathbf{c}_r + \mathbf{b}_2),$$
$$\mathbf{f}^r_p = \sigma(\mathbf{W}_{11}\mathbf{tx}_p + \mathbf{W}_{12}\mathbf{h}_r + \mathbf{W}_{13}\mathbf{c}_l + \mathbf{W}_{14}\mathbf{h}_r + \mathbf{W}_{15}\mathbf{c}_r + \mathbf{b}_3),$$
$$\mathbf{g}_p = \tanh(\mathbf{W}_{16}\mathbf{tx}_p + \mathbf{W}_{17}\mathbf{h}_l + \mathbf{W}_{18}\mathbf{h}_r + \mathbf{b}_4),$$
$$\mathbf{c}_p = \mathbf{f}^l_p \odot \mathbf{c}_l + \mathbf{f}^r_p \odot \mathbf{c}_r + \mathbf{i}_p \odot \mathbf{g}_p,$$
$$\mathbf{o}_p = \sigma(\mathbf{W}_{19}\mathbf{tx}_p + \mathbf{W}_{20}\mathbf{h}_p + \mathbf{W}_{21}\mathbf{h}_r + \mathbf{W}_{22}\mathbf{c}_p + \mathbf{b}_5), \;\; \mathbf{h}_p = \mathbf{o}_p \odot \tanh(\mathbf{c}_p).$$

Here the subscripts p, l and r denote $[i,j]$, $[i,k]$ and $[k+1,j]$, respectively.

Label Decoder. Suppose that the constituent label chain for the span $[i,j]$ is $(\mathrm{YL}^0_{[i,j]}, \mathrm{YL}^1_{[i,j]}, \ldots, \mathrm{YL}^m_{[i,j]})$. The decoder for the span $[i,j]$ learns a conditional language model depending on the output vector $\mathbf{h}[i,j]$ from the tree LSTM encoder. Formally, the probability distribution of generating the label at time step z is given by,

$$P(\mathrm{YL}^z_{[i,j]}|T_{ub}, \mathrm{YL}^{z<m}_{[i,j]}) = \mathrm{softmax}\Big(g(\mathbf{h}[i,j], \mathbf{E}_{label}(\mathrm{YL}^{z-1}_{[i,j]}), \mathbf{d}_{z-1})\Big),$$

where $\mathrm{YL}^{z<m}_{[i,j]}$ is the decoding prefix, $\mathbf{d}_{z-1}$ is the state vector of the decoder LSTM and $\mathbf{E}_{label}(\mathrm{YL}^{z-1}_{[i,j]})$ is the embedding of the previous output label.

The training objective is to minimize the negative log-likelihood of the label generation distribution,

$$\mathcal{L}_{\text{label}}[i,j] = -\sum_{z=0}^{m} \log P(\mathrm{YL}^z_{[i,j]}|T_{ub}, \mathrm{YL}^{z<m}_{[i,j]}),$$
$$\mathcal{L}_{\text{label}} = \sum_{[i,j]\in T_{ub}} \mathcal{L}_{\text{label}}[i,j].$$

2.4 Joint training

In conclusion, each model contains an unlabeled structure predictor and a label generator. The latter is the same for all models. All the span models perform binary classification. The difference is that BinarySpan doesn't consider label information for unlabeled tree prediction. While MultiSpan guides unlabeled tree prediction with such information, simulating binary classifications. The unlabeled parser and the label generator share parts of the network components, such as word embeddings, char embeddings, POS embeddings and the BiLSTM encoding layer. We jointly train the unlabeled parser and the label generator for each model by minimizing the overall loss

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{parser}} + \mathcal{L}_{\text{label}} + \frac{\lambda}{2}||\Theta||^2,$$

where λ is a regularization hyper-parameter. We set $\mathcal{L}_{\text{parser}} = \mathcal{L}_{\text{binary}}$ or $\mathcal{L}_{\text{parser}} = \mathcal{L}_{\text{multi}}$ and $\mathcal{L}_{\text{parser}} = \mathcal{L}_{\text{rule}}$ when using the binary span classification model, the multi-class model and the rule model, respectively.

hyper-parameter	value	hyper-parameter	value
Word embeddings	English: 100 Chinese: 80	Word LSTM layers	2
Word LSTM hidden units	200	Character embeddings	20
Character LSTM layers	1	Character LSTM hidden units	25
Tree-LSTM hidden units	200	POS tag embeddings	32
Constituent label embeddings	32	Label LSTM layers	1
Label LSTM hidden units	200	Last output layer hidden units	128
Maximum training epochs	50	Dropout	English: 0.5, Chinese 0.3
Trainer	SGD	Initial learning rate	0.1
Per-epoch decay	0.05	ϕ	ELU (Clevert et al., 2015)

Table 1: Hyper-parameters for training.

3 Experiments

3.1 Experimental Settings

Data. We perform experiments for both English and Chinese. Following standard conventions, our English data are obtained from the Wall Street Journal (WSJ) of the Penn Treebank (PTB) (Marcus et al., 1993). Sections 02-21, section 22 and section 23 are used for training, development and test sets, respectively. Our Chinese data are the version 5.1 of the Penn Chinese Treebank (CTB) (Xue et al., 2005). The training set consists of articles 001-270 and 440-1151, the development set contains articles 301-325 and the test set includes articles 271-300. We use automatically reassigned POS tags in the same way as Cross and Huang (2016b) for English and Dyer et al. (2016) for Chinese.

We use ZPar (Zhang and Clark, 2011)[1] to binarize both English and Chinese data with the head rules of Collins (2003). The head directions of the binarization results are ignored during training. The types of English and Chinese constituent span labels after binarization are 52 and 56, respectively. The maximum number of greedy decoding steps for generating consecutive constituent labels is limited to 4 for both English and Chinese. We evaluate parsing performance in terms of both unlabeled bracketing metrics and labeled bracketing metrics including unlabeled F1 (UF)[2], labeled precision (LP), labeled recall (LR) and labeled bracketing F1 (LF) after debinarization using EVALB[3].

Unknown words. For English, we combine the methods of Dyer et al. (2016) and Cross and Huang (2016b) to handle unknown words. In particular, we first map all words (not just singleton words) in the training corpus into unknown word classes using the same rule as Dyer et al. (2016). During each training epoch, every word w in the training corpus is stochastically mapped into its corresponding unknown word class unk_w with probability $P(w \rightarrow unk_w) = \frac{\gamma}{\gamma + \#w}$, where $\#w$ is the frequency count and γ is a control parameter. Intuitively, the more times a word appears, the less opportunity it will be mapped into its unknown word type. There are 54 unknown word types for English. Following Cross and Huang (2016b), $\gamma = 0.8375$. For Chinese, we simply use one unknown word type to dynamically replace singletons words with a probability of 0.5.

Hyper-parameters. Table 1 shows all hyper-parameters. These values are tuned using the corresponding development sets. We optimize our models with stochastic gradient descent (SGD). The initial learning rate is 0.1. Our model are initialized with pretrained word embeddings both for English and Chinese. The pretrained word embeddings are the same as those used in Dyer et al. (2016). The other parameters are initialized according to the default settings of DyNet (Neubig et al., 2017). We apply dropout (Srivastava et al., 2014) to the inputs of every LSTM layer, including the word LSTM layers, the character LSTM layers, the tree-structured LSTM layers and the constituent label LSTM layers. For Chinese, we find that 0.3 is a good choice for the dropout probability. The number of training epochs is decided by the evaluation performances on development set. In particular, we perform evaluations on development set for every 10,000 examples. The training procedure stops when the results of next 20 evaluations do not become better than the previous best record.

[1] https://github.com/SUTDNLP/ZPar

[2] For UF, we exclude the sentence span [0,n-1] and all spans with length 1.

[3] http://nlp.cs.nyu.edu/evalb

Model	SpanVec	LP	LR	LF
BinarySpan	$\mathbf{v}[i,j]$	**92.16**	**92.19**	**92.17**
	$\mathbf{sr}[i,j]$	91.90	91.70	91.80
BiaffineRule	$\mathbf{v}[i,j]$	91.79	91.67	91.73
	$\mathbf{sr}[i,j]$	**92.49**	**92.23**	**92.36**

Table 2: Span representation methods.

Model	English			Chinese		
	LP	LR	LF	LP	LR	LF
BinarySpan	92.16	92.19	92.17	91.31	90.48	90.89
MultiSpan	92.47	**92.41**	**92.44**	**91.69**	90.91	**91.30**
LinearRule	92.03	92.03	92.03	91.03	89.19	90.10
BiaffineRule	**92.49**	92.23	92.36	91.31	**91.28**	91.29

Table 3: Main development results.

3.2 Development Results

We study the two span representation methods, namely the simple concatenating representation $\mathbf{v}[i,j]$ (Eq 1) and the combining of three difference vectors $\mathbf{sr}[i,j]$ (Eq 7), and the two representative models, i.e, the binary span classification model (**BinarySpan**) and the biaffine rule model (**BiaffineRule**). We investigate appropriate representations for different models on the English dev dataset. Table 2 shows the effects of different span representation methods, where $\mathbf{v}[i,j]$ is better for `BinarySpan` and $\mathbf{sr}[i,j]$ is better for `BiaffineRule`. When using $\mathbf{sr}[i,j]$ for `BinarySpan`, the performance drops greatly ($92.17 \rightarrow 91.80$). Similar observations can be found when replacing $\mathbf{sr}[i,j]$ with $\mathbf{v}[i,j]$ for `BiaffineRule`. Therefore, we use $\mathbf{v}[i,j]$ for the span models and $\mathbf{sr}[i,j]$ for the rule models in latter experiments.

Table 3 shows the main results on the English and Chinese dev sets. For English, `BinarySpan` acheives 92.17 LF score. The multi-class span classifier (**MultiSpan**) is much better than `BinarySpan` due to the awareness of label information. Similar phenomenon can be observed on the Chinese dataset. We also test the linear rule (**LinearRule**) methods. For English, `LinearRule` obtains 92.03 LF score, which is much worse than `BiaffineRule`. In general, the performances of `BiaffineRule` and `MultiSpan` are quite close both for English and Chinese.

For `MultiSpan`, both the first stage (unlabeled tree prediction) and the second stage (label generation) exploit constituent types. We design three development experiments to answer what the accuracy would be like of the predicted labels of the first stage were directly used in the second stage. The first one doesn't include the label probabilities of the first stage for the second stage. For the second experiment, we directly use the model output from the first setting for decoding, summing up the label classification probabilities of the first stage and the label generation probabilities of the second stage in order to make label decisions. For the third setting, we do the sum-up of label probabilities for the second stage both during training and decoding. These settings give LF scores of 92.44, 92.49 and 92.44, respectively, which are very similar. We choose the first one due to its simplicity.

3.3 Main Results

English. Table 4 summarizes the performances of various constituent parsers on PTB test set. `BinarySpan` achieves 92.1 LF score, outperforming the neural CKY parsing models (Durrett and Klein, 2015) and the top-down neural parser (Stern et al., 2017). `MultiSpan` and `BiaffineRule` obtain similar performances. Both are better than `BianrySpan`. `MultiSpan` obtains 92.4 LF score, which is very close to the state-of-the-art result when no external parses are included. An interesting observation is that the model of Stern et al. (2017) show higher LP score than our models (93.2 v.s 92.6), while our model gives better LR scores (90.4 v.s. 93.2). This potentially suggests that the global constraints such as structured label loss used in (Stern et al., 2017) helps make careful decisions. Our local models are likely to gain a better balance between bold guesses and accurate scoring of constituent spans. Table 7 shows the unlabeled parsing accuracies on PTB test set. `MultiSpan` performs the best, showing 92.50 UF score. When the unlabeled parser is 100% correct, `BiaffineRule` are better than the other two, producing an oracle LF score of 97.12%, which shows the robustness of our label generator. The decoding speeds of `BinarySpan` and `MutliSpan` are similar, reaching about 21 sentences per second. `BiaffineRule` is much slower than the span models.

Chinese. Table 5 shows the parsing performance on CTB 5.1 test set. Under the same settings, all the three models outperform the state-of-the-art neural model (Dyer et al., 2016; Liu and Zhang, 2017a).

Parser	LR	LP	LF	Parser	LR	LP	LF
Zhu et al. (2013) (S)	91.1	91.5	91.3	Charniak (2000)	89.5	89.9	89.5
McClosky et al. (2006) (S)	92.1	92.5	92.3	Collins (2003)	88.1	88.3	88.2
Choe and Charniak (2016) (S,R,E)			93.8	Sagae and Lavie (2006)	87.8	88.1	87.9
Durrett and Klein (2015) (S)			91.1	Petrov and Klein (2007)	90.1	90.2	90.1
Vinyals et al. (2015) (S, E)			92.8	Carreras et al. (2008)	90.7	91.4	91.1
Charniak and Johnson (2005) (S, R)	91.2	91.8	91.5	Zhu et al. (2013)	90.2	90.7	90.4
Huang (2008b) (R)			91.7	Watanabe and Sumita (2015)			90.7
Huang and Harper (2009) (ST)	91.1	91.6	91.3	Fernández-González and Martins (2015)	89.9	90.4	90.2
Huang et al. (2010) (ST)	92.7	92.2	92.5	Cross and Huang (2016b)	90.5	92.1	91.3
Shindo et al. (2012) (E)			92.4	Kuncoro et al. (2017)			91.2
Socher et al. (2013) (R)			90.4	Liu and Zhang (2017b)	91.3	92.1	91.7
Dyer et al. (2016) (R)			93.3	Stern et al. (2017) top-down	90.4	93.2	91.8
Kuncoro et al. (2017) (R)			93.6	**BinarySpan**	91.9	92.2	92.1
Liu and Zhang (2017a) (R)			94.2	**MultiSpan**	92.2	92.5	**92.4**
Fried et al. (2017) (ES)			**94.7**	**BiaffineRule**	92.0	92.6	92.3

Table 4: Results on the PTB test set. *S* denotes parsers using auto parsed trees. *E*, *R* and *ST* denote ensembling, reranking and self-training systems, respectively.

Parser	LR	LP	LF	Parser	LR	LP	LF
Charniak and Johnson (2005) (R)	80.8	83.8	82.3	Petrov and Klein (2007)	81.9	84.8	83.3
Zhu et al. (2013) (S)	84.4	86.8	85.6	Zhang and Clark (2009)	78.6	78.0	78.3
Wang et al. (2015) (S)			86.6	Watanabe and Sumita (2015)			84.3
Huang and Harper (2009) (ST)			85.2	Dyer et al. (2016)			84.6
Dyer et al. (2016) (R)			86.9	**BinarySpan**	85.9	87.1	86.5
Liu and Zhang (2017b)	85.2	85.9	85.5	**MultiSpan**	86.6	88.0	**87.3**
Liu and Zhang (2017a)			86.1	**BiaffineRule**	87.1	87.5	**87.3**

Table 5: Results on the Chinese Treebank 5.1 test set.

Compared with the in-order transition-based parser, our best model improves the labeled F1 score by 1.2 (86.1 $\rightarrow$ 87.3). In addition, `MultiSpan` and `BiaffineRule` achieve better performance than the reranking system using recurrent neural network grammars (Dyer et al., 2016) and methods that do joint POS tagging and parsing (Wang and Xue, 2014; Wang et al., 2015).

4 Analysis

Constituent label. Table 6 shows the LF scores for eight major constituent labels on PTB test set. `BinarySpan` consistently underperforms to the other two models. The error distribution of `MultiSpan` and `BiaffineRule` are different. For constituent labels including SBAR, WHNP and QP, `BiaffineRule` is the winner. This is likely because the partition point distribution of these labels are less trivial than other labels. For NP, PP, ADVP and ADJP, `MultiSpan` obtains better scores than `BiaffineRule`, showing the importance of the explicit type information for correctly identifying these labels. In addition, the three models give similar performances of VP and S, indicating that simple local classifiers might be sufficient enough for these two labels.

LF v.s. Length. Figure 3 and Figure 4 show the LF score distributions against sentence length and span length on the PTB test set, respectively. We also include the output of the previous state-of-the-art top-down neural parser (Stern et al., 2017) and the reranking results of transition-based neural generative parser (RNNG) (Dyer et al., 2016), which represents models that can access more global information. For sentence length, the overall trends of the five models are similar. The LF score decreases as the length increases, but there is no salient difference in the downing rate (also true for span length $\leq$6), demonstrating our local models can alleviate the label bias problem. `BiaffineRule` outperforms the other three models (except RNNG) when the sentence length less than 30 or the span length less than 4. This suggests that when the length is short, the rule model can easily recognize the partition point. When the sentence length greater than 30 or the span length greater than 10, `MultiSpan` becomes the best option (except RNNG), showing that for long spans, the constituent label information are useful.

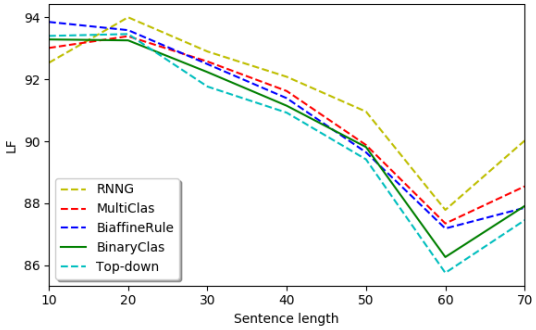


Figure 3: Sentence length v.s LF scores.

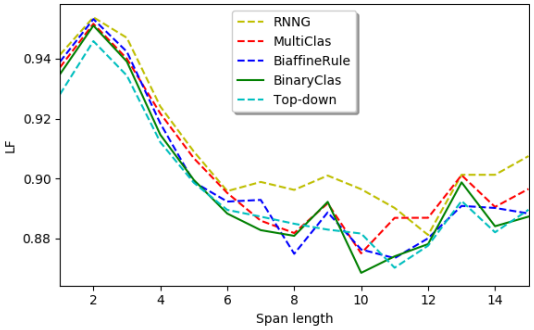


Figure 4: Span length v.s LF scores.

Model	NP	VP	S	PP	SBAR	ADVP	ADJP	WHNP	QP
BinarySpan	93.35	93.26	92.55	89.58	88.59	85.85	76.86	95.87	89.57
MultiSpan	**93.61**	93.41	92.76	**89.96**	89.16	**86.39**	**78.21**	95.98	89.51
BiaffineRule	93.53	**93.46**	**92.78**	89.30	**89.56**	85.89	77.47	**96.66**	**90.31**

Table 6: LF scores for major constituent labels.

Model	UF	LF	Speed(sents/s)
BinarySpan	92.16	96.79	22.12
MultiSpan	92.50	97.03	21.55
BiaffineRule	92.22	97.12	6.00

Table 7: UF, oralce LF and speed.

5 Related Work

Globally trained discriminative models have given highly competitive accuracies on graph-based constituent parsing. The key is to explicitly consider connections between output substructures in order to avoid label bias. State-of-the-art statistical methods use a single model to score a feature representation for all phrase-structure rules in a parse tree (Taskar et al., 2004; Finkel et al., 2008; Carreras et al., 2008). More sophisticated features that span over more than one rule have been used for reranking (Huang, 2008b). Durrett and Klein (2015) used neural networks to augment manual indicator features for CRF parsing. Structured learning has been used for transition-based constituent parsing also (Sagae and Lavie, 2005; Zhang and Clark, 2009; Zhang and Clark, 2011; Zhu et al., 2013), and neural network models have been used to substitute indicator features for transition-based parsing (Watanabe and Sumita, 2015; Dyer et al., 2016; Goldberg et al., 2014; Kiperwasser and Goldberg, 2016; Cross and Huang, 2016a; Coavoux and Crabbé, 2016; Shi et al., 2017).

Compared to the above methods on constituent parsing, our method does not use global structured learning, but instead learns local constituent patterns, relying on a bi-directional LSTM encoder for capturing non-local structural relations in the input. Our work is inspired by the biaffine dependency parser of Dozat and Manning (2016). Similar to our work, Stern et al. (2017) show that a model that bi-partitions spans locally can give high accuracies under a highly-supervised setting. Compared to their model, we build direct local span classification and CFG rule classification models instead of using span labeling and splitting features to learn a margin-based objective. Our results are better although our models are simple. In addition, they collapse unary chains as fixed patterns while we handle them with an encoder-decoder model.

6 Conclusion

We investigated two locally trained span-level constituent parsers using BiLSTM encoders, demonstrating empirically the strength of the local models on learning syntactic structures. On standard evaluation, our models give the best results among existing neural constituent parsers without external parses.

Acknowledgement

Yue Zhang is the corresponding author. We thank all the anonymous reviews for their thoughtful comments and suggestions.

References

Steven P. Abney. 1991. Parsing by chunks. In *Principle-based parsing*, pages 257–278. Springer.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany, August. Association for Computational Linguistics.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97, Beijing, China, August. Coling 2010 Organizing Committee.

Xavier Carreras, Michael Collins, and Terry Koo. 2008. Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 9–16. Association for Computational Linguistics.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Association for Computational Linguistics.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas, November. Association for Computational Linguistics.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289.

Maximin Coavoux and Benoit Crabbé. 2016. Neural greedy constituent parsing with dynamic oracles. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 172–182, Berlin, Germany, August. Association for Computational Linguistics.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637.

James Cross and Liang Huang. 2016a. Incremental parsing with minimal features using bi-directional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 32–37, Berlin, Germany, August. Association for Computational Linguistics.

James Cross and Liang Huang. 2016b. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Austin, Texas, November. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 302–312, Beijing, China, July. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California, June. Association for Computational Linguistics.

Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1523–1533, Beijing, China, July. Association for Computational Linguistics.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proceedings of ACL-08: HLT*, pages 959–967, Columbus, Ohio, June. Association for Computational Linguistics.

Daniel Fried, Mitchell Stern, and Dan Klein. 2017. Improving neural parsing by disentangling model combination and reranking effects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 161–166, Vancouver, Canada, July. Association for Computational Linguistics.

Yoav Goldberg, Francesco Sartorio, and Giorgio Satta. 2014. A tabular method for dynamic oracles in transition-based parsing. *Transactions of the association for Computational Linguistics*, 2:119–130.

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

Zhongqiang Huang and Mary Harper. 2009. Self-training PCFG grammars with latent annotations across languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 832–841, Singapore, August. Association for Computational Linguistics.

Zhongqiang Huang, Mary Harper, and Slav Petrov. 2010. Self-training with products of latent variable grammars. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 12–22, Cambridge, MA, October. Association for Computational Linguistics.

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

Liang Huang. 2008a. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Columbus, Ohio, June. Association for Computational Linguistics.

Liang Huang. 2008b. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Columbus, Ohio, June. Association for Computational Linguistics.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *CoRR*, abs/1603.04351.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan, July. Association for Computational Linguistics.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11, Uppsala, Sweden, July. Association for Computational Linguistics.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298, Cambridge, MA, October. Association for Computational Linguistics.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1249–1258, Valencia, Spain, April. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, Fernando Pereira, et al. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289.

Jiangming Liu and Yue Zhang. 2017a. In-order transition-based constituent parsing. *Transactions of the Association for Computational Linguistics*, 5:413–424.

Jiangming Liu and Yue Zhang. 2017b. Shift-reduce constituent parsing with neural lookahead features. *Transactions of the Association for Computational Linguistics*, 5:45–58.

Xuezhe Ma and Eduard Hovy. 2017. Neural probabilistic model for non-projective mst parsing. *arXiv preprint arXiv:1701.00874*.

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

Andre Martins, Noah Smith, Eric Xing, Pedro Aguiar, and Mario Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 34–44, Cambridge, MA, October. Association for Computational Linguistics.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 337–344, Sydney, Australia, July. Association for Computational Linguistics.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT*, pages 149–160.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Comput. Linguist.*, 34(4):513–553, December.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April. Association for Computational Linguistics.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 125–132. Association for Computational Linguistics.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 129–132, New York City, USA, June. Association for Computational Linguistics.

Tianze Shi, Liang Huang, and Lillian Lee. 2017. Fast(er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 12–23, Copenhagen, Denmark, September. Association for Computational Linguistics.

Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. 2012. Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–448, Jeju Island, Korea, July. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, Sofia, Bulgaria, August. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada, July. Association for Computational Linguistics.

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

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1556–1566, Beijing, China, July. Association for Computational Linguistics.

Ben Taskar, Dan Klein, Mike Collins, Daphne Koller, and Christopher Manning. 2004. Max-margin parsing. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 1–8, Barcelona, Spain, July. Association for Computational Linguistics.

Zhiyang Teng and Yue Zhang. 2016. Bidirectional tree-structured LSTM with head lexicalization. *CoRR*, abs/1611.06788.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2306–2315, Berlin, Germany, August. Association for Computational Linguistics.

Zhiguo Wang and Nianwen Xue. 2014. Joint pos tagging and transition-based constituent parsing in chinese with non-local features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 733–742, Baltimore, Maryland, June. Association for Computational Linguistics.

Zhiguo Wang, Haitao Mi, and Nianwen Xue. 2015. Feature optimization for constituent parsing via neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1138–1147, Beijing, China, July. Association for Computational Linguistics.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based neural constituent parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1169–1179, Beijing, China, July. Association for Computational Linguistics.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(02):207–238.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 162–171, Paris, France, October. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational linguistics*, 37(1):105–151.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA, June. Association for Computational Linguistics.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1213–1222, Beijing, China, July. Association for Computational Linguistics.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 434–443, Sofia, Bulgaria, August. Association for Computational Linguistics.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1604–1612, Lille, France, July.

RNN Simulations of Grammaticality Judgments on Long-distance Dependencies

Shammur Absar Chowdhury and Roberto Zamparelli
CIMeC: Center for Mind/Brain Sciences
University of Trento
{shammur.chowdhury, roberto.zamparelli}@unitn.it

Abstract

The paper explores the ability of LSTM networks trained on a language modeling task to detect linguistic structures which are ungrammatical due to extraction violations (extra arguments and subject-relative clause island violations), and considers its implications for the debate on language innatism. The results show that the current RNN model can correctly classify (un)grammatical sentences, in certain conditions, but it is sensitive to linguistic processing factors and probably ultimately unable to induce a more abstract notion of grammaticality, at least in the domain we tested.

Title and Abstract in Italian

RNN Simulazioni di giudizi di grammaticità sulle dipendenze a distanza

L'articolo studia la capacità delle reti neurali LSTM addestrate su un compito di modellazione linguistica di rilevare strutture linguistiche che sono agrammaticali a causa di violazioni nella estrazione di argomenti (dovute alla presenza di argomenti di troppo, o alla presenza di isole del soggetto e delle frasi relative), esplorando le implicazioni per il dibattito sull'innatismo linguistico. I risultati mostrano che l'attuale modello RNN può classificare correttamente frasi grammaticali, in certe condizioni, ma è eccessivamente sensibile a fattori di elaborazione linguistica e probabilmente non in grado di indurre una nozione più astratta di grammaticalità, almeno nel dominio da noi testato.

1 Introduction

Native speaker intuitions about the meaning and grammaticality of linguistic expressions have been the key methodology in theoretical linguistics since at least Chomsky (1957), and are widely seen as a crucial window on the internalized linguistic competence which theoretical linguistics aims to study. Despite lively discussions on the limits of the methodology (see Cowart, 1997; Sprouse and Almeida, 2012; Sprouse *et al.*, 2013, 2016), the availability of very large corpora has not replaced in linguistics the need for judgments on artificially constructed cases whenever theoretical points hinge on the status of very rare or complex constructions, concern languages or dialects for which large corpora do not exist, or involve semantic intuitions that have no easily detectable correlates in corpora (e.g. semantic scope alternations).

Due to its scarce practical applications, modeling grammaticality judgments with computers has never been a typical NLP task, but it can be an important testbed for theories of language processing and grammatical competence. In particular, judgments on syntactic well-formedness require a sensitivity to long-distance structural cues which is a crucial aspect of language competence (Everaert *et al.*, 2015). Elman's (1991) pioneering work on the application of simple recurrent neural network (RNN, Elman 1990) to linguistic sequences showed that such networks could indeed learn some linguistic structures, but had a tendency to forget important linguistic features (e.g. the presence of a Wh-element) as new

Proceedings of the 27th International Conference on Computational Linguistics, pages 133–144
Santa Fe, New Mexico, USA, August 20-26, 2018.

material arrived. However, more recent network architectures, like Long-Short Term Memory Network (LSTMs, Hochreiter and Schmidhuber 1997) or Gated Recurrent Networks (GRU, Chung *et al.* 2014) incorporate memory-management systems which yield much better performances. The seminal work of Linzen *et al.* (2016) and other recent papers (Bernardy and Lappin, 2017; Gulordava *et al.*, 2018) have shown these networks to be capable, in certain conditions, to approximate human levels in a subject-verb number agreement task, even across intervening nouns and verb (e.g. *The boy**s** [that the girl has seen] **are/*is**...*).

These advances raise the question whether similar models, trained on corpora of naturally occurring sentences, could come to approximate the full range of human grammaticality judgments, including judgments on structures which, unlike agreement, are virtually non-existent in the training input. The ability to do so would have implications for the debate on language innatism (Christiansen and Kirby, 2003; Sprouse *et al.*, 2013): a generalist (R)NN has no language-specific learning skills, no innate linguistic parameters to set (Chomsky and Lasnik, 1993; Baker, 2001); if such a device could manage to replicate fine-grained human intuitions inducing them from the raw training input this would be evidence that exposure to language structures (albeit in an amount orders of magnitude larger than the one a child receives, and without a connection to the non-linguistic context of utterance) should in principle be sufficient to derive a syntactic competence, against the innatist hypothesis. Suppose on the other hand that NNs could approximate human intuitions on some linguistic phenomena but not on others, despite similar statistical distributions in the training input: this would now count as strong evidence that the 'unlearnable' phenomena tap on aspects of the grammar faculty that have limited representations in normal language samples, and are good candidates for being innate.

This paper is a step in the direction of this research program. We trained two types of RNNs (GRU and LSTM) on a large corpus of English (Wikipedia and parts of UKWAC, Ferraresi *et al.* 2008), then tested their performances or a range of artificially constructed language structures, contrasting grammatical and ungrammatical examples. Unlike Lau *et al.* (2017), who use LSTM to prove that they can learn the *graded* nature of human grammaticality, we only look at long distance dependencies (the relation between a dislocated element like a Wh-nominal or the head of a relative clause and its gap). As a preliminary task (Task A), we check whether the network is sensitive to the difference in processing complexity between subject and object relatives, a much-studied domain in the psycholinguistic literature; next, we turn to two cases of ungrammaticality, one due to a violation of the principle of Full Interpretation (Chomsky, 1986a, p.98-99) (Task B), one to extraction out of "strong" syntactic islands (Ross, 1982), specifically, subject and relative clause islands (Task C). Violations of these types have been regarded as sharp both in the theoretical (Szabolcsi and den Dikken, 1999) and the experimental literature (Sorace and Keller, 2005; Cowart, 1997; Sprouse *et al.*, 2013).

Our preliminary conclusions are that while the models are able to catch a surprising range of subtle differences in Task A and to correctly classify grammatical and ungrammatical cases in Task B, their performances are highly sensitive to processing factors (esp. sentence length, sentence complexity), a fact which becomes particularly evident in Task C. Two possible conclusions can be drawn: either the NN has troubles disentangling processing and grammaticality, or the most obvious ways for detecting this distinction are not effective for this target.

In the following sections, we first present a brief overview of the methodology used in this study (Section 2), followed by the a detailed task description in Section 3. The results and observation of the network behavior is presented in Section 4; we conclude the study and discuss future directions in Section 5.

2 Methodology

The use of RNN for the evaluation of grammaticality can already be found in Tomida and Utsumi (2013) (a reply to a non-NN learning model in Pearl and Sprouse 2013). However, their RNN (a Jordan RNN, Jordan 1997) worked on abstract data (it was trained on preassigned constituent labels and had to generate other label sequences), a choice that in our opinion requires too many underlying assumptions. RNN trained on raw textual input are first found in Linzen *et al.* (2016), who achieve a very high score (<1%

error rate) in the subject-verb number agreement task. However, they specifically train an LSMN classifier on this task alone. This is fine as a demonstration that the input contains enough network-accessible information to make the correct choice, but it puts the network at an unfair advantage over humans, since giving explicit grammaticality judgment is a rather marginal and non-natural linguistic task (though we make use of *implicit* judgments when we decide if someone is a native speaker, we choose the best wording in a text, etc.). Moreover, it is a task which arguably plays no role in language acquisition. While there are of course immense quantitative and qualitative differences between the language learning process in humans and NNs, we believe that a comparison between their final states can still make sense, under two conditions: (i) that the comparison is not directly between humans and NN judgments, but rather between task-dependent judgment differences, i.e. minimal task-pairs which are very similar for the machine but very different for humans, or vice versa; (ii) that the NN has **not** been specifically trained on this tasks. When Linzen et al. (2016) trained their LSTM on a general task (language modeling: predicting the following word, arguably a more natural task, see van Berkum 2010) and tried to use the resulting network for the judgment task, the error rate increased to around 7% error rate. More recently, a group including Linzen himself (Gulordava *et al.*, 2018) has shown that, with better tuning, a different LSTM trained on a language modeling task is in fact capable of performances comparable to those of the classifier in Linzen *et al.* (2016), even in the absence of any semantic information (a potential confound in Linzen *et al.* 2016).

Assessing grammaticality judgments in a task which is not binary like number agreement raises important methodological questions. We want the network to discriminate between (1a) and (1b). The acceptable Wh-question in (1a) might end with a question mark right at the gap (the object position of *catch*), but also continue with an adverbial, or even (at the coast of decreasing acceptability) a nominal containing a gap (*the tail of*). The ungrammatical case (1b) ends with a normal verb argument (*the tail*).

(1) a. Which mouse did the cat catch {? / last night ? / the tail of ? }
b. *Which mouse did the cat catch the tail?

The question is which NN measure best corresponds to the speaker's perception of ungrammaticality in (1b), keeping into account that even in the theoretical and psycholinguistic literature there are no established metrics to measure 'degrees of ungrammaticality' (see Cowart 1997; Sorace and Keller 2005 for discussion).

2.1 Evaluation Measures

We explored various possibilities, from *global* measures like *perplexity* (PPL) and non-normalized sentence cross-entropy loss (CEL), to *local* measures like the normalized log probability of a full stop $LogP_n(FS)$ or question mark $LogP_n(QM)$ after the current word.

Cross-entropy loss (CEL) measures the distance between the output distribution predicted by the model and one-hot vector representing the target word (t_i). The loss for the test sentence can therefore be described as an approximation of the equation $CEL = \sum_{i=1}^{n} \ln P(t_i)$, where $P(t_i)$ is the probability given by the model to the i^{th} target word in the sentence of length n. In our analysis, we mostly use the averaged CEL (ACEL) while comparing each particular set of cases. Note that (A)CEL is not normalized by sentence length. Perplexity measures how many equally probable words can follow a point in the text; as the sentence grows longer and more information accumulates, the options for the following word decrease. Perplexity is calculated by $PPL = e^{-\frac{TCEL}{N}}$, where the total cross-entropy loss ($TCEL = \sum CEL$) is computed for the sub-dataset and the total number of words, N, in the corresponding dataset. Both measures are based on the intuition that an ungrammatical sentence should 'confuse' the NN more than a corresponding grammatical one, and that this confusion will translate in a decreased ability to make correct predictions.

As for the local measure, normalized log probability of the full stop and question mark is calculated as $LogP_n(QM/FS) = log(\frac{p_m(\Im)}{p_u(\Im)})$ where $p_m(\Im)$ is the probability of the symbol, $\Im$, at a given position given by the model; $p_u(\Im)$ is the unigram probability of the $\Im$.

As it turns out, neither PPL or CEL are perfect ways to evaluate a language model for ungrammatical-

ity, since they do not locate it at a specific point in the sentences. Yet, this could also be an advantage, since global measures like PPL/CEL can potentially catch parsing problems that arise earlier than expected, and record a perturbation in the NN's later predictions as it recovers from an ungrammatical point earlier in the text. Therefore, these methods seem appropriate for a first exploration of this task.

On the other hand, the advantage of the local measure used in this study (P(FS/QM), i.e. the confidence that the sentence is about to end) is that it can give precise information about the point at which ungrammaticality is detected (e.g. after *catch* in (1)), or can be used to track the NN expectations through time, as we do in Figure 1. The disadvantage is that, as example (1a) shows, a sentence can always continue in unexpected ways. Moreover, in some cases of ungrammaticality the presence of individual words might not be a significant predictor.

2.2 RNN Architecture

In order to simulate an "unbiased learner" that tries to model human grammatical intuitions, inducing them from the raw training input, we designed a RNN-based language model. A language model is often defined as a model learning the conditional probability over words, given a historical context. The model includes a vocabulary V of size $|V|$; when learning a training sentence, each word is represented by a one-hot encoding vector, $x \in \mathbb{R}^{|V|}$, with a corresponding index in V.

For this study we used two successful variant of simple RNN – long-short term memory (LSTM) and gated recurrent unit (GRU) models. GRU is basically an LSTM without the output gate; the content of the memory cell is copied into the network at each time steps. We trained the models varying the number of hidden units (u=$\{$100, 1500$\}$). For our datasets we observed that the un-tuned LSTM (u=100) performed slightly better than the GRU architecture with same parameters. Though it is possible that with better tuning the GRU could outperform LSTM, for the purpose of this study we present data from just the LSTM model tuned only for number of hidden units $u = 1500$, $layer = 2$ and embedding dimension, $e = 256$. To avoid over-fitting the model on the training data we also applied dropout — a regularization technique — in different layers. For training the model, we used a PyTorch RNN implementation with an SGD optimizer. We have not tuned the models for different dropout or learning rate parameters, and we used a fixed batch size of 80.

To train the RNN model we used an English corpus extracted from Wikipedia plus a small subset of UKWaC ($\approx$ 0.32% of the training data), crawled from *.uk* domain. For the Wikipedia data, we downloaded the WikiText-103 raw data containing the original case, punctuation and numbers. We then tokenized both datasets, removing urls, email addresses, emoticons and text enclosed in any form of brackets ({.},(.), [.]). We replaced rare words (tokens with frequency $\leq$10) with *<unk>* token along with its signatures (e.g. *-ed, -ing, -ly* etc.) to represent every possible out-of-vocabulary (OOV) words. We also replaced numbers (exponential, comma separated etc) with a *<NUM>* tag. After these preprocessing steps, our training data consisted of $\approx 136M$ words, with a vocabulary of size $|V| = 0.1M$. It is to be noted that the style of the texts selected to train the language model is mostly encyclopedic, due to the prevalence of Wikipedia over other web corpus data. This is of course not representative of the typical registers of English language, but it does give us a good proportion of complex embedded sentences, which match some of the deeply embedded constructions present in the task set. Recall, moreover, that the measures in this study are always relative, i.e. they contrast minimally different inputs within the same language model. Thus, any imbalance in the training data should not be expected to make a large difference in the global results.

3 Task Description

In order to filter out the effect of words which could affect the network performance in ways orthogonal to the structures at issue, we opted to increase the number of sentences to evaluate by building them as *sentence schemata*, (e.g. (2) for the subject relative Task A), which were expanded automatically to generate all the possible ways of picking one of the expressions within {}. Note that some of the variable were experimental conditions (e.g. the presence of *that* or *who* in (2)), others were added just to increase variety at the level of the content lexicon, so as to minimize possible effects of collocations or sentences

the network might have encountered in the training phase. The results we present are averaged across all the sentences that express the same experimental condition.

(2) { The / A / Every / Some } { student / man / professor / driver } { that / who } had { seen / spoken with / interviewed / mentioned / approached / met / lived with }{ her / Mary / the woman } gave a brief speech.

The phenomena[1] we tested are the following:

A. SUBJECT VS. OBJECT RELATIVE CLAUSES: Much psycholinguistic literature since Gibson (1998); Gordon *et al.* (2001) has shown that object relatives such as (3a) are harder to parse than subject relatives (3b) (in terms of time, accuracy, etc.), and that the presence of certain material intervening between the head (*boy*) and the gap (indicate with an underscore) can affect reaction times (Gordon *et al.*, 2002, 2004). While this is not a contrast in grammaticality (both structures are clearly acceptable for adults, Adani *et al.* 2010), we designed this preliminary task to check if the network was sensitive to the position of the gap and to the type of intervening subject: a pronoun, a proper name or a full noun phrase. According to Rizzi's Featural Relativized Minimality (Friedmann *et al.*, 2009; Adani *et al.*, 2010), an intervener with many grammatical features in common with the moved element can make the extraction harder, or even ungrammatical (see esp. Villata *et al.* 2016 for the case of 'weak islands').

(3) a. The boy that Mary has invited __ *object extraction*
b. The boy that __ invited Mary *subject extraction*

A second variable was the relative pronoun, which could be *that*, *who* or null (in object relatives: *The girl Anna saw*). This test set comprised 1680 expanded sentences.

B. WH EXTRACTIONS: A second test set involves cases of Wh-extraction where the gap position could be empty (4a) or filled by an overt element (4b) (a personal pronoun, an indefinite pronoun, a demonstrative NP).

(4) a. Which candidate/issue should the students discuss ?
b. *Which candidate/issue should the students discuss {him / it / something else / this candidate / this issue}

(4b) is a very strong semantic/syntactic violation, since either the Wh or the final nominal cannot be connected to the verb, violating the principle of Full Interpretation (Chomsky, 1986b). However, certain uses of pronouns are standardly treated as bound variables in formal semantics (and there are languages, e.g. Hebrew or Welsh, which use so-called *resumptive* pronouns in place of gaps at least in relative clauses), so we expected that filling the gap with pronouns might be better than filling it with full noun phrases. If the NN is able to carry over semantic information from the Wh-phrase to the gap position, we also expected that gap-fillers that match in animacy (*which candidate . . . him/this candidate*) should be better than non matching cases (*which issue . . . him/this candidate*). The sentences were generated at 0, 1 or 2 levels of embedding (e.g. *Who did John claim [that the professor assumed [that the students should discuss (it)]]?* is level 2), to see if distance makes the network 'forget' the Wh or its features.

All the Wh cases above were contrasted with corresponding affirmative sentences. In this case, however, the gap is the ungrammatical case (5a), while the gap fillers we see in (5b) all yield grammatical sentences. The pre-gap verbs, *discuss, mention, describe, write about, worry about, address, promote, consider* were chosen not to easily allow intransitive counterparts.

(5) a. *The student should consider .

[1]The expanded test sets for each task can be found in
https://github.com/LiCo-TREiL/Computational-Ungrammaticality.

b. The student should consider {him / it / something else / this candidate / this issue}

This allows us to directly compare the interrogative and affirmative case. What makes this task particularly interesting is the fact that, locally, the beginning and the end of each sentence is perfectly grammatical; they become strongly ungrammatical only when seen together, possibly at a distance which is rarely (if ever) attested in corpora. When fully expanded, this test set contains 72720 sentences.

C. SUBJECT AND RELATIVE ISLAND VIOLATIONS: While the previous test measures the ability of the network to carry information from the Wh phrase across the whole sentence, the last test set pitches them against the phenomenon of *syntactic islands* (Ross, 1967). Descriptively, a *subject island* blocks a dependency whose gap is *inside* (hence the tag 'subextraction') a nominal subject (6a), as opposed to a nominal object (6b); a *relative clause island* bars gaps inside relatives (7). Note that (7b) combines the two types of islands, and should be worse if the effect of multiple violations is cumulative.

(6) a. *Who did [a classmate of __] ruin John ?
b. ?Who did John see of [a classmate of __]?

(7) a. *Which girl did John see [the person that dated __]?
b. **Which girl did [the person that dated __] see John ?

Despite decades of research, there is no established functional explanations why islands exist (though see Szabolcsi and Zwarts 1990; Sprouse *et al.* 2012 for some approaches). Thus, they represent a good starting point to verify the limits of NN performance. In this case, we provided Y/N question and affirmative counterparts for each of the Wh-interrogatives, as a point of reference.

4 Results and Discussion

Task A. Subject vs. Object Relative Clauses: Tables 1-2 contains the results of the Subject *vs.* Object Relative Task. As it can be observed, the network is better (both in terms of PPL and CEL) at dealing with subject than object relatives, thus indicating a sensitivity for the position of the gap which corresponds to the preference found in humans, and especially children (Adani *et al.*, 2010). Moreover, the network loss (Avg. CEL) improves (i.e. it is lower) when the other nominal inside the RC is a pronoun (recall that in *object* relatives the non-gapped nominal intervenes between the Wh-element and the gap): the pronoun improves by ≈ 2.83 over (proper name) PN and ≈ 2.94 over the (noun phrase) NP in subject RC, by ≈ 5.55 and ≈ 5.93 respectively in object relative; there is no significant difference in loss between PNs and full NPs. The fact that the pronoun effect is larger in subject position is particularly noteworthy. It could be due to the greater frequency of pronouns in subject than object position reported in Gordon *et al.* (2001), but this tendency is also in line with the Featural Relativized Minimality approach of Friedmann *et al.* (2009); Villata *et al.* (2016): in object RC the RC-internal nominal intervenes between the relative head and its gap; the more features the head and the nominal have in common, the more the connection between head and gap is disrupted; pronouns have fewer features in common with the relative head (*boy* in (3)) than other nominals (e.g. no +N(oun)), so they interfere less.

With respect to relative pronoun type, Table 2 shows a significant preference for *that* over *who* in Subject RC, and a preference for null relative pronouns over overt ones in object RC (i.e. *the boy Mary saw* > *the boy that/who Mary saw*, ">" = easier for the NN), in terms of ACEL. This seems in line with corpus frequencies measured on UKWAC01, where **Det NP** *that* is about 4 times more frequent than **Det NP** *who*.

Cases	**PPL**	**ACEL** ($\pm std$)	**#**
Subj-relatives	84.52	55.99 ($\pm$3.60)	672
Obj-relatives	105.59	57.25 ($\pm$4.23)	1008

Table 1: Task A Results: Subject vs. Object RC. # represents the number of instances in that sub-data

Nom-Rel	Subj-R	Obj-R	Rel-Pronoun	Subj-R	Obj-R
	ACEL	*ACEL*		*ACEL*	*ACEL*
pronoun	54.07	53.42	*that*	57.7	57.4
proper name	56.9	58.97	*who*	54.28	58.25
full Noun Phrase	57.01	59.35	*no Rel Pron.*	–	56.09

Table 2: Task A Results: Nom-Rel represents Nominal inside Relative clause and Rel-Pronoun represents Relative Pronouns.

Task B. Wh extractions and FI violations: To analyze the Task B dataset we initially divided it into four cases: [WH. . . GAP], [AFF(ERMATIVE). . . NOGAP] (both grammatical) and [WH. . . NOGAP], [AFF. . . GAP] (both ungrammatical). We studied the overall network perplexity (PPL) and the ACEL loss for the sentence given by the NN. We observed that the PPL is 106.10 overall for the grammatical sets, 151.57 for the ungrammatical ones. When we keep Wh and affirmative cases apart, [AFF. . . NOGAP] have PPL = 67.72 (calculated over 11640 instances), which is as expected lower than [AFF. . . GAP], PPL = 76.63 (calculated over 2940 instances). However, the perplexity given by grammatical [WH...GAP] (163.16, with 11616 instances) is higher than that of the ungrammatical [WH...NOGAP]-sentences (PPL = 156.42, 46560 instances).

Options to Select from	CEL	Comments
O1: What should Mary discuss? *Gramm.*	**29.14**	
O2: What should Mary discuss it?	32.79	
O3: What should Mary discuss him?	34.32	CEL(O1) <CEL(O2-6)
O4: What should Mary discuss something else?	39.63	Correct if O1 is chosen
O5: What should Mary discuss this topic?	40.22	
O6: What should Mary discuss this candidate?	42.37	
O1: The professor has said that Mary should consider. *UnGramm.*	**48.40**	
O2: The professor has said that Mary should consider it.	49.15	CEL(O1) <CEL(O2-5)
O3: The professor has said that Mary should consider something else.	55.53	Correct if any of O2-O5
O4: The professor has said that Mary should consider this topic.	56.87	is chosen
O5: The professor has said that Mary should consider this candidate.	58.40	

Table 3: Example of Gramm. vs UnGramm. classification task by RNN using ACEL as a measure. Lower values are better.

To explore this further, we designed a simple classification task where each sentence was presented with 5 or 6 possible alternatives in the gap position, as shown in Table 3. The NN's choice was correct if the CEL for the correct option was the lowest. The experiment included a total of 14520 instances containing Wh and affirmative sentences and had an accuracy of 91.45%, indicating that the RNN network is largely able to pick the correct option for Wh (99.1% out of 11616 instances) and for Aff (60.7% out of 2904 instances). The very different margins of the two effects, which probably account for the WH/AFF PPL difference above, are actually not unexpected, as it is probably easier to accommodate the existence of an intransitive version of a transitive verb than to explain away an extra argument.

Since all the sentences in Task B could potentially end at the (filled) gap, we decided to investigate the effect of sentence type on the network's perception that the sentence is about to end. Figure 1 tracks the NN expectation that the following word is going to be a full stop (P(FS)), or a question mark (P(QM)). The results are intriguing: in general, an end-of-sentence (EOS) is least expected after an auxiliary or modal; in affirmatives "." is unlikely after the final verb, very likely after the object. Interestingly, the possibility of a question mark (i.e. a question marked by intonation alone) is always present in affirmatives. Wh-sentences are dominated by the expectation of a gap (whose proxy is "?"), peaking at the first verb and decreasing slowly. The unexpected arrival of the object seems to convince the NN that the sentence is after all not a direct interrogative ("." higher than "?").

Turning to an analysis of the impact of each filled overt element in the Wh-extraction dataset, Table 3 shows that when the gap is filled by a personal pronoun (Pro, e.g. *it, him*), the overall PPL, of such set, is much better than when the gap filled by an indefinite pronoun (IndPro, *something else*) or a demonstrative NP (a 10 point difference). A similar pattern obtains with ACEL: Pro $>$ IndPro $>$ DemNP. The ability of the NN to track semantic information (specifically, animacy) from the Wh to the gap (i.e. *which candidate . . . this candidate* and *which issue . . . this topic*) was also confirmed: the global PPL of the cases with a match in animacy is lower ($PPL = 220.05$) than the unmatched cases ($PPL = 223.89$),

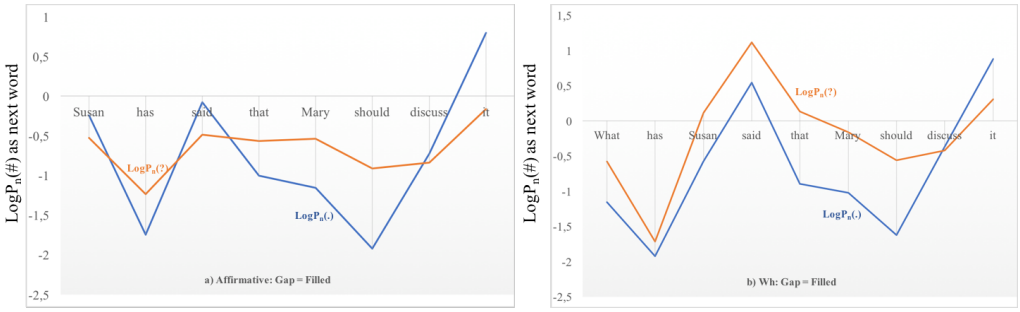


Figure 1: The EOS expectations for different types of sentences. Log Probability that the next word is a "?" (red) or a "." (blue), according to the LSTM model. The sentence is an example for the whole category.

but by only 3 points; once again, a similar pattern with a minor difference of 0.31 obtains using average CEL.

In Figure 2, we presents the effect of different levels of embedding between the wh-element and the gap position in cases where the gap is filled (Ungrammatical cases) or empty (Grammatical). Our findings suggests that in all our scenarios (grammatical and ungrammatical, Wh and affirmatives) an increase in the level of embedding increases the average CEL very significantly. A similar pattern is observed with affirmative sentences.

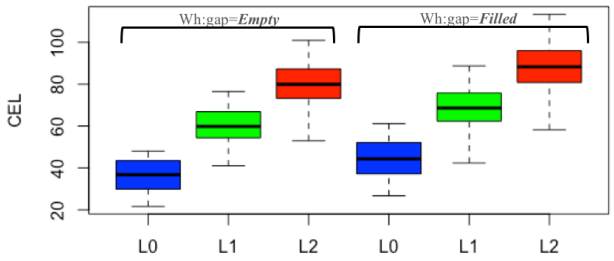


Figure 2: Variation of CEL for different level of embedding; Wh cases with gap empty or filled.

Task C. Subject and Relative Island Violations: So far we have seen that the RNN model was able to distinguish grammatical from ungrammatical pairs with some success, but also to capture a number of interesting effect from the psycholinguistic literature: in Task A, the preference for subject relatives and the effect of intervening pronouns vs. full DPs (see Table 2); in Task B, the almost acceptability of Wh-resumptive pronouns vs. indefinites vs. the fully ungrammatical demonstrative fillers (Table 3), the matching animacy, and the shifting preferences for full stops or question marks (Figure 1). However, in many of these cases the margin of success was small with respective to others; for instance, only 3 ACEL points divide *What should Mary discuss* from *What should Mary discuss it*, but the ACEL distance between e.g. *What should Mary discuss* and *What has she said that Mary should discuss* is over 20 ACEL points.

We now turn to see how the model performs on the typically syntactic Task C. The overall results for subject (S) and object (O) position are presented in Table 4. Since many of the gaps are non final we do not employ the local P(FS/QM) measure and only present the two global measures. At a first glance, the network seems to have been able to capture the facts with exactly the right progression: in the ACEL measure object subextractions are best, followed at an 8 ACEL point distance by subextractions from subjects, subject of passives, relatives in object position and relatives in subject position. The PPL facts are harder to fit, especially for the passive (NSp: 95.64). It should be noted, however, that NSp could be interpreted as a *parasitic gap* construction (Ross, 1967; Engdahl, 1985), which is normally judged fairly acceptable (*Who was a portrait of __ painted by __ ?*), plus a spurious *Tim* filling the second gap, as in Task B.

Has our model learned syntactic islands from Wikipedia? It would appear so, down to the fact that the

Cases		PPL	ACEL ($\pm std$)	#
OBJ NP extraction (NO):	*Who has Tim seen a portrait of __?*	95.87	43.50 ($\pm$8.32)	480
SUBJ NP extraction (NS):	*Who has a portrait of __ scared Tim?*	159.78	48.37 ($\pm$7.30)	480
SUBJ NP extr. from pass. (NSp):	*Who was a portrait of __ painted by Tim?*	95.64	49.75 ($\pm$5.95)	480
OBJ RC extraction (RO):	*Who has Tim seen a portrait that showed __ ?*	131.75	51.64 ($\pm$8.76)	768
SUBJ RC extraction (RS):	*Who has a portrait that showed __ scared Tim?*	181.87	56.11 ($\pm$6.90)	960

Table 4: Overall results of Task C: Subject and Relative Clause (RC) island violations, where overall PPL represents the perplexity on the sub-dataset, ACEL represents the average cross-entropy loss given by the network and # represents the number of instances in that sub-data set. The losses differ from each other significantly, p-value < 0.05.

Cases	Wh-interrogatives	Y/N interrogatives	Affirmatives	Ratio PPL	Ratio CEL
	ACEL	ACEL	ACEL	(Wh-/YN), (Wh-/Aff)	(Who/YN), (Who/Aff)
OBJ pos. extraction (NO)	**51.26**	**45.81**	**36.4**	(1.57, 2.78)	(1.12, 1.42)
SUBJ. pos. extraction (NS)	54.32	51.05	41.73	(1.27, 2.03)	(1.06, 1.30)
SUBJ. Pos. extr. + Passive (NSp)	53.51	51.92	45.08	(1.07, 2.01)	(1.03, 1.19)
OBJ. RC (RO)	59.11	54.35	43.95	(1.43, 2.46)	(1.09, 1.34)
SUBJ. RC (RS)	60.76	58.54	50.57	(1.12, 1.47)	(1.04, 1.20)

Table 5: Results of Task C: Subject and Relative Island Violations, divided in to *wh- interrogatives*; *Y/N interrogatives* and *affirmatives*. ACEL represents the average cross-entropy loss given by the network and #, the number of instances in that sub-data set. For ACEL the lower the value the better. The losses we obtain differ from each other significantly, p-value < 0.05.

last case, RS extraction, is the worst because, arguably, it is the sum of two distinct islands — relative and subject. However, a look at the performance of the model on a set of parallel cases shows that the interpretation of these facts should probably be quite different. Recall that for each Wh case, our Task C test set contains the corresponding Y/N and affirmative sentences (8):

(8) a. Who did John see the person that dated __?
b. Did John see the person that dated Mary?
c. John saw the person that dated Mary.

Table 5 shows the ACEL scores for all three types, as well as the ratio between the scores given to the sentence types. One remarkable aspect is that the Wh cases have a much higher (i.e. worse) score than the corresponding affirmative; even more remarkable, however, is the fact that Y/N questions are also very far from assertions, much closer in fact to their Wh counterparts. But Y/N questions have no Wh gap, hence no island effects. Equally remarkably, Y/N-questions and assertions follow a progression which is almost identical to the one of Wh-cases: NO>NS>NSp>RO>RS. This is quite visible from the ratios, which remain very stable under ACEL, and taper down slightly in PPL.

This data shows that the increased perplexity with Wh cases has nothing to do with island effects, or we would not find it in Yes/No questions and assertions. Our hypothesis is that it is rather the cumulative effect of increasing syntactic complexity, plus position. Suppose that an NP such as *a classmate of John* is more complex than *John*, *a classmate* and possibly *John's classmate*, thus potentially more ambiguous. Suppose further that relative clauses are even more complex/ambiguous. Ambiguity leads to uncertainty, so by increasing it we increase perplexity as well, yielding the difference between NO,NS>RO,RS.

This does not yet explain why NO>NS and RO>RS (recall that they are made of the same words, just in different orders). This, we hypothesize, is the effect of position. A complex structure at the beginning of a sentence (subject position) can be more damaging than one at the end (object position), probably since it can lead the intermediate network units into a "wrong" state of activation, which will be fed back to the RNN as the next word enters, generating additional perplexity. To test this hypothesis, we placed other types of complex nominals (NP conjunctions, NPs containing adjectival and PP modifiers in both subject ((9a), (9c)) and object position (e.g. (9b), (9d)) in Y/N-interrogatives and in the corresponding affirmative sentences. We observed that the PPL/ACEL for sentences with complex subjects was 154.79/73.01, vs. 99.70/66.64 with complex objects. The pattern was similar in Y/N-interrogatives

and affirmatives, and across the two different types of complex nominals.

(9) a. Did [the publisher, the journalist and their families] meet Mary?
b. Did Mary meet [the man, the woman and their families]?
c. Did [a well built, blond-haired man with a large, heavy backpack] meet Mary ?
d. Did Mary meet [a well built, blond-haired man with a large, heavy backpack]?

In addition, all things being equal, a Wh might generate additional perplexity, since it leads the model to expect a gap at multiple points (cf. Figure 1b). Putting these factors together, it is not strange that the RS case (relative complexity + subject position + Wh perplexity) might come out with the highest score.

5 Conclusions and future work

Our work builds on Linzen *et al.* (2016); Gulordava *et al.* (2018); Bernardy and Lappin (2017), but with a difference goal: the possibility for an RNN to learn to classify sentences by grammaticality, focusing on two specific long-distance effects: the presence of an extra argument (Task B) and the island effects in extraction (Task C). The results showed, first of all, that our NN model was good at capturing known effects in the processing of relatives (subject/object preferences, effect of interveners), and good at spotting the selective need for a final argument (Task B). The fact that the ungrammatical cases were graded, with pronouns next best after gaps (see Table 3) shows that the network wasn't just using the simple rule "If it starts with Wh*, pick the version with fewer words, if not, don't". On the other hand, the overwhelming effect of processing factors like the level of embedding (Figure 2), and the fact that the apparent success of the NN in the island task in not based on the island extraction effect itself cast doubts on the idea that the NN is using an abstract dimension of 'grammaticality'.

It could be tempting to take this as a cue that even human ungrammaticality should be reduced to processing (see Villata *et al.* 2016 for discussion in the domain of *weak islands*, which we did not test here), but there are reasons to believe that, while processing might play a role, it cannot be the whole story. As is well-known to anyone who has practiced a musical instrument, pronounced tongue-twisters or read centrally-embedded sentences, processing difficulties improve a lot with practice. However, multiple repetitions of *Who did John see Mary?* by humans are not likely to make it better.

The study raises several methodological issues, first and foremost in the choice of measures. Perplexity seems to be an obvious first choice for a language model, but even when it is normalized by the number of words in the sentence it is sensitive to contrasting effects: longer sentences are more predictable, but recursively embedded sentences are less. Moreover, complex structures increase perplexity more when they are at the beginning of a sentence than at the end (see our discussion of Task C results). In future work we plan to explore variations of these measures with different properties, and use *bidirectional* LSTMs to mitigate the latter effect. Local prediction tracking as in Figure 1, on the other hand, seems to be a promising, intuitive tool to see what the NN is "thinking" throughout a parse, but it is not always applicable. This is related to a final point. Grammaticality is typically judged relative to a (grammatical) point of comparison. Thus, *Who did a portrait that showed scared Tim?* is truly awful if judged as the RC+Subj extraction from *The portrait that showed **who** scared Bill?*, but what if the NN takes it as a variant of a structure such as *[(the person) who has a dress that (really) showed] scared Tim*? Humans can be given minimal pairs to make the difference clear. Learning how to do this with NN remains for future work. Only after these issues have been resolved and a performance plateau has been reached we will be in a position to go back to the original question: are (R)NN feasible models of innate-grammar-free language learners? Which abstract properties can they learn from the input?

References

Adani, F., van der Lely, H. K., Forgiarini, M., and Guasti, M. T. (2010). Grammatical feature dissimilarities make relative clauses easier: A comprehension study with italian children. *Lingua*, **120**(9), 2148 – 2166.

Baker, M. C. (2001). *The atoms of language: the mind's hidden rules of grammar.* Oxford University Press, Oxford.

Bernardy, J.-P. and Lappin, S. (2017). Using deep neural networks to learn syntactic agreement. *LiLT*, **15**(2).

Chomsky, N. (1957). *Syntactic Structures*. Mouton, The Hague.

Chomsky, N. (1986a). *Barriers*. Linguistic Inquiry Monograph 13. MIT Press, Cambridge, Mass.

Chomsky, N. (1986b). *Knowledge of Language: Its Nature, Origins and Use*. Praeger, New York.

Chomsky, N. and Lasnik, H. (1993). The theory of principles and parameters. In J. Jacobs and al., editors, *Syntax: An International Handbook of Contemporary Research*, volume 1, pages 506–569. Walter de Gruyter.

Christiansen, M. H. and Kirby, S. (2003). Language evolution: consensus and controversies. *TRENDS in Cognitive Sciences*, **7**(7), 300–307.

Chung, J., Cho, C. G. K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

Cowart, W. (1997). *Experimental syntax: Applying objective methods to sentence judgments*. Sage Publications, Thousand Oaks.

Elman, J. (1990). Finding structure in time. *Cognitive Science*, **14**, 179–211.

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, **7**(2–3), 195–225.

Engdahl, E. (1985). Parasitic gaps, resumptive pronouns, and subject extractions. *Linguistics*, pages 3–44.

Everaert, M., Huybregts, M., Chomsky, N., Berwick, R., and Bolhuis, J. (2015). Structures, not strings: Linguistics as part of the cognitive sciences. *Trends in Cognitive Sciences*, **19**(12), 729–743.

Ferraresi, A., Zanchetta, E., Baroni, M., and Bernardini, S. (2008). Introducing and evaluating ukWaC, a very large web-derived corpus of English. In *Proceedings of the 4th Web as Corpus Workshop WAC4*, pages 47–54.

Friedmann, N., Belletti, A., and Rizzi, L. (2009). Relativized relatives: types of intervention in the acquisition of a-bar dependencies. *Lingua*, **119**(1), 67–88.

Gibson, E. (1998). Linguistic complexity: locality of syntactic dependencies. *Cognition*, **68**, 1–76.

Gordon, P., Randall, H., and Marcus, J. (2001). Memory interference during language processing. *J. Exp. Psychol. Learn. Mem. Cogn.*, **27**, 1411–1423.

Gordon, P., Hendrick, R., and Johnson, M. (2004). Effects of noun phrase type on sentence complexity. *Journal of Memory and Language*, **51**(1), 97–114.

Gordon, P. C., Hendrick, R., and Levine, W. H. (2002). Memory load interference in syntactic processing. *Psychological Science*, **13**, 425–430.

Gulordava, K., Bojanowski, P., Grave, E., Linzen, T., and Baroni, M. (2018). Colorless green recurrent networks dream hierarchically. In *Proceedings of NAACL*.

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

Jordan, j. M. (1997). Serial order: A parallel distributed processing approach. *Advances in psychology*, **495**, 121–471.

Lau, J. H., Clark, A., and Lappin, S. (2017). Grammaticality, acceptability, and probability: A probabilistic view of linguistic knowledge. *Cognitive Science*, **41**(5), 1202–1241.

Linzen, T., Dupoux, E., and Goldberg, Y. (2016). Assessing the ability of LSTMs to learn syntax-sensitive dependencies. In *Transactions of the Association for Computational Linguistics*, volume 4, pages 521–535.

Pearl, L. and Sprouse, J. (2013). Syntactic islands and learning biases: Combining experimental syntax and computational modeling to investigate the language acquisition problem. *Language Acquisition*, **68**, 20–23.

Ross, J. R. (1967). *Constraints on Variables in Syntax*. Indiana University Linguistics Club, Bloomington.

Ross, J. R. (1982). Pronoun deleting processes in German. Paper presented at the Annual Winter Meeting of the LSA, San Diego.

Sorace, A. and Keller, F. (2005). Gradience in linguistic data. *Lingua*, **115**, 1497–1524.

Sprouse, J. and Almeida, D. (2012). Assessing the reliability of textbook data in syntax: Adger's "core syntax". *Journal of Linguistics*, **48**(3), 609–652.

Sprouse, J., Wagers, M., and Phillips, C. (2012). A test of the relation between working-memory capacity and syntactic island effects. *Language*, **88**(1), 82–123. Linguistic Society of America.

Sprouse, J., Schütze, C. T., and Almeida, D. (2013). A comparison of informal and formal acceptability judgments using a random sample from linguistic inquiry 2001-2010. *Lingua*, **134**, 219–248. DOI: 10.1016/j.lingua.2013.07.002.

Sprouse, J., Caponigro, I., Greco, C., and Cecchetto, C. (2016). Experimental syntax and the variation of island effects in english and italian. *Natural Language and Linguistic Theory*, **34**, 307–344.

Szabolcsi, A. and den Dikken, M. (1999). Islands. *GLOT Internationaal*, **4**(6), 3–8.

Szabolcsi, A. and Zwarts, F. (1990). Semantic properties of composed functions and the distribution of wh-phrases. In M. Stokhof and L. Torenvliet, editors, *Proceedings of the Seventh Amsterdam Colloquium*, pages 529–555. Institute for Language, Logic and Information, Amsterdam.

Tomida, Y. and Utsumi, A. (2013). A connectionist model for acquisition of syntactic islands. *Procedia - Social and Behavioral Sciences*, **97**, 90–97.

van Berkum, J. J. A. (2010). The brain is a prediction machine that cares about good and bad – any implications for neuropragmatics? *Italian Journal of Linguistics*, **22**.

Villata, S., Rizzi, L., and Franck, J. (2016). Intervention effects and relativized minimality: New experimental evidence from graded judgments. *Lingua*, **179**, 76–96.

How Predictable is Your State? Leveraging Lexical and Contextual Information for Predicting Legislative Floor Action at the State Level

Vlad Eidelman **Anastassia Kornilova** **Daniel Argyle**
FiscalNote Inc.
Washington DC
{vlad,anastassia,daniel}@fiscalnote.com

Abstract

Modeling U.S. Congressional legislation and roll-call votes has received significant attention in previous literature. However, while legislators across 50 state governments and D.C. propose over 100,000 bills each year, and on average enact over 30% of them, state level analysis has received relatively less attention due in part to the difficulty in obtaining the necessary data. Since each state legislature is guided by their own procedures, politics and issues, however, it is difficult to qualitatively asses the factors that affect the likelihood of a legislative initiative succeeding. Herein, we present several methods for modeling the likelihood of a bill receiving floor action across all 50 states and D.C. We utilize the lexical content of over 1 million bills, along with contextual legislature and legislator derived features to build our predictive models, allowing a comparison of the factors that are important to the lawmaking process. Furthermore, we show that these signals hold complementary predictive power, together achieving an average improvement in accuracy of 18% over state specific baselines.

1 Introduction

Federal institutions in the U.S., like Congress and the Supreme Court, play a significant role in lawmaking, and in many observable ways define our legal system. Thus, as data and computational resources have become more readily available, political scientists have increasingly been adopting quantitative methods focused on understanding these entities and the role they play in our society (Katz et al., 2017; Poole and Rosenthal, 2007; Slapin and Proksch, 2008; Lauderdale and Clark, 2014).

Although many issues are legislated and regulated primarily at the federal level, state governments have significant power over certain areas. An increasing number of important issues are being decided at the state or local levels, especially in emerging industries and technologies, such as the gig economy and autonomous vehicles (Hedge, 1998). Moreover, there are 535 members of Congress who introduce over 10,000 pieces of legislation a session,[1] of which less than 5% is enacted. Similar dynamics exist at the state level, except on a much broader scale. There are over 7,000 state legislators, in aggregate introducing over 100,000 pieces of legislation, with over 30% being enacted. In order to be enacted, every bill must pass through one or more legislative committees and be considered on the chamber floor, a process we refer to as receiving floor action. This process is one of the most pivotal steps during lawmaking (Rosenthal, 1974; Hamm, 1980; Francis, 1989; Rakoff and Sarner, 1975), as on average, only 41% of bills receive floor action, with most legislation languishing in the committees.

Legislative policymaking decisions are extremely complex, and are influenced by a myriad of factors, ranging from the content of the legislation, to legislators' personal characteristics, such as profession, religion, and party and ideological affiliations, to their constituents' demographics, to governor agendas, to interest group activities, and to world events (Canfield-Davis et al., 2010; Hicks and Smith, 2009; Talbert and Potoski, 2002).

Despite this complexity, in this paper we present an approach to better understand state lawmaking dynamics and the legislative process by focusing on the task of predicting the likelihood that legislation

[1] A session is the period of time a legislative body is actively enacting legislation, usually one to two years.

Proceedings of the 27th International Conference on Computational Linguistics, pages 145–160
Santa Fe, New Mexico, USA, August 20-26, 2018.

will reach the floor in each state. As there are many dimensions underlying the content of the legislation, such as the policy area and ideology of the sponsor (Linder et al., 2018), that may affect the likelihood of floor action, in addition to text we examine several established contextual legislature and legislator derived features. To the best of our knowledge, this is the first work quantitatively modeling the floor action process across all 50 states and using the text of legislation alongside traditional contextual information.

2 Related Work

Much of the work analyzing the federal legislature is aimed at understanding legislator preferences through the use of voting patterns. One of the most popular techniques in political science is the application of spatial, or ideal point, models built from voting records (Poole and Rosenthal, 1985; Poole and Rosenthal, 2007), that is often used to represent unidimensional or multidimensional ideological stances (Clinton et al., 2004). However, there is also an increasing literature examining broader legislative dynamics, such as measuring legislative effectiveness (Harbridge, 2016), evaluating the impact of legislation on stock prices using legislators' constituents (Cohen et al., 2012), creating cosponsorship networks (Fowler, 2006), and examining the role of lobbying (Bertrand et al., 2018; Matthew et al., 2013),

In recent years a variety of primary and secondary textual legal data, such as legislation, floor debates, and committee transcripts, has become increasingly available, enabling the NLP community to create richer multidimensional ideal point estimation (Gerrish and Blei, 2011; Nguyen et al., 2015; Kornilova et al., 2018), and examine ideology detection from political speech (Iyyer et al., 2014), voting prediction from debates (Thomas et al., 2006), committee referral (Yano et al., 2012), and enactment (Nay, 2016).

While there is also an increasing amount of state legislative research, states have received significantly less attention (Hamm et al., 2014). One major reason for this is that quantitative methods require data, and the availability of data for Congress far exceeds that of states. In fact, Yano et al. (2012) noted "When we consider a larger goal of understanding legislative behavior across many legislative bodies (e.g., states in the U.S., other nations, or international bodies), the challenge of creating and maintaining such reliable, clean, and complete databases seems insurmountable." Thus, while there has been scholarship quantifying the role of committees, it has been limited in scope, to a few sessions or states, or reliant on survey data (Francis, 1989; Rakoff and Sarner, 1975; Rosenthal, 1974; Hamm, 1980). More recently, as different kinds of state data has become more accessible, it has enabled studying the affect of interest groups on legislative activity (Gray and Lowery, 1995), the application of spatial models (Shor et al., 2010; Shor and McCarty, 2011), and comparisons of textual similarity (Linder et al., 2018).

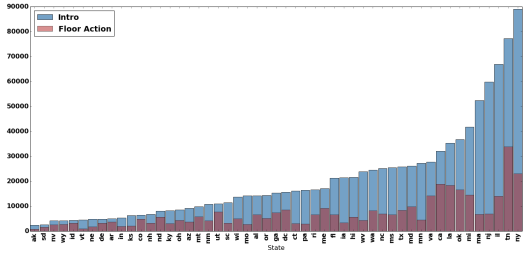
The contribution of this work is to continue building a broader understanding of state legislative dynamics by evaluating how predictable state lawmaking is, and what factors influence that process. We create a novel task, predicting the likelihood of legislation receiving floor action, and utilize a corpus of over 1 million bills to build computational models of all 50 states and D.C. We present several baseline models utilizing various features, and show that combining the legislative and legislator contextual information with the text content of bills consistently provides the best predictions. Our analysis considers various factors and their respective importance in the predictive models across the states, showing that although there are some consistent patterns, there are many variations and differences in what affects the likelihood in each state.

3 Data

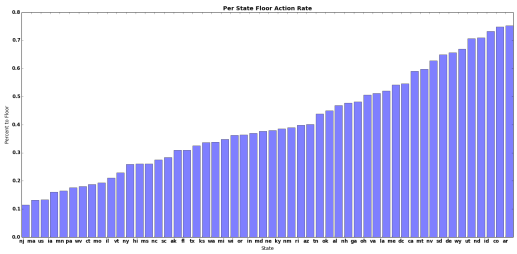
There is state-to-state variation in the legislative procedure of how a bill becomes law, but the path is largely similar. Legislation is introduced by one or more members of the legislature in their respective chamber,[2] and assigned to one or more standing subject committees.[3] Committees are made up of a subset of members of their respective chamber, and are chaired by the majority party. Once in committee,

[2] All legislatures are bicameral, with either a House or Assembly as the lower chamber, and the Senate as the upper chamber, except D.C. and Nebraska, which are unicameral.

[3] Depending on the state, other groups can introduce legislation, including legislative committees, legislative delegations, the governor, or non-elected individuals. For the purpose of this work we focus on legislator sponsored legislation.



(a) Number of bills introduced and receiving floor action for each state.



(b) Percent of bills reaching floor per state.

Figure 1: Dataset characteristics.

legislation is subject to debate and amendment only by the committee members, with the successful outcome being a favorable referral, or a recommendation, to be considered by the full chamber on the floor.

The primary data we use to model floor action was scraped directly from each state legislatures' website. For each state, we downloaded legislation, committee, and legislator pages for all sessions that were publicly accessible. Legislation pages were automatically parsed to determine legislative contextual metadata, which includes bill text versions, sponsors, committee assignments, and the timeline of actions. Legislator pages were parsed to obtain sponsor contextual metadata, which includes party affiliation, committee assignments, and committee roles.

As states demarcate legislative status in the timeline of actions differently, we automatically map and normalize all textual descriptions of legislative actions to a finite set of statuses.[4] These statuses are used to determine whether a piece of legislation survived committee and received a floor action, or consideration on the floor. All bills having a status of passed in their introductory chamber, or having had a recorded floor vote are treated as positive examples, while any status prior to passed is considered failed, including legislation that was reported out of committee but not considered on the floor.

Finally, since each state follows their own conventions with regard to classifying the type of legislation, we normalize all legislation across states to two types: resolutions and bills.[5]

Figure 1a shows the total number of bills introduced and receiving floor action for each state. In total, our dataset consists of 1.3 million pieces of state legislation, broken into 1 million bills, with 360k receiving floor action, at an average rate across states of 41%, and 275k resolutions, with 210k receiving floor action. On average, we have 10 legislative sessions of data per state.[6] As bills represent substantive legislation, with a much lower floor action rate, while resolutions are much more likely to receive floor action, for the rest of this paper we focus on bills only, and refer to bills and legislation interchangeably. We include 15 sessions of U.S federal legislation in our data for comparative purposes, with 23k of 172k bills receiving floor action.

Figure 1b presents the percent of bills receiving floor action. It is interesting to note the difference in difficulty for legislation to receive floor action in different states. For example, in New Jersey and Massachusetts, fewer than 15% of bills reach the floor, whereas 75% do in Colorado and Arkansas.[7]

4 Methods

4.1 Models

In order to not only be able to predict, but also examine the importance of features to our prediction, we chose three relatively interpretable models for our modeling framework. Formally, let our training

[4]The normalized statuses include introduced, assigned to committee, reported from committee, and passed.

[5]Resolutions are pieces of legislation of type appointment, resolution, joint resolution, concurrent resolution, joint memorial, memorial, proclamation, nomination. Bills are those of type bill, amendment, urgency, appropriation, tax levy, or constitutional amendment.

[6]Full data statistics are given in Table 7 in Appendix A.

[7]Our average across states, chambers, and sessions is in line with previous single state and session findings; in examining five states Rosenthal (1974) found between 34% and 73% of legislation did not survive committee.

data $(\boldsymbol{X}, \boldsymbol{Y})$ consist of n pairs $(\boldsymbol{x}_i, y_i)_{i=1}^n$ where, each $\boldsymbol{x}_i$ is a bill and y_i a binary indicator of whether $\boldsymbol{x}_i$ received floor action. Let $\mathbf{f}(\boldsymbol{x}_i)$ be a feature vector representation of $\boldsymbol{x}_i$, and $\mathbf{w}$ the parameter vector indicating the weight of each feature learned by the model.

The first two models are linear classifiers, where the prediction of floor action, $\hat{y}_i$, is given by $\texttt{sign}(\mathbf{w}^\top \mathbf{f}(\boldsymbol{x}))$. The first is a regularized conditional log-linear model $p_{\mathbf{w}}(y \mid \boldsymbol{x})$:

$$p_{\mathbf{w}}(y \mid \boldsymbol{x}) = \frac{\exp\left\{\mathbf{w}^\top \mathbf{f}(\boldsymbol{x})\right\}}{Z(\boldsymbol{x})} \tag{1}$$

where $Z(\boldsymbol{x})$ is the partition function given by $\sum_y \exp\left\{\mathbf{w}^\top \mathbf{f}(\boldsymbol{x})\right\}$. The model optimizes $\mathbf{w}$ according to

$$\min_{\mathbf{w}} \sum_{i}^{n} -\log p_{\mathbf{w}}(y_i \mid \boldsymbol{x}_i) + \lambda ||\mathbf{w}|| \tag{2}$$

The second model is NBSVM (Wang and Manning, 2012), an interpolation between multinomial Naive Bayes and a support vector machine, which optimizes $\mathbf{w}$ according to:

$$\min_{\mathbf{w}} C \sum_{i}^{n} \max(0, 1 - y_i(\mathbf{w}^\top(\mathbf{f}(\boldsymbol{x}_i) \circ \mathbf{r})))^2 + ||\mathbf{w}||^2 \tag{3}$$

where $\mathbf{r}$ is the log-count ratio of features occurring in positive and negative examples. The third model is non-linear, in the form of a tree-based gradient boosted machine (Friedman, 2000), which optimizes $\mathbf{w}$ according to:

$$\min_{\mathbf{w}} \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(\mathbf{t}_k) \tag{4}$$

where K is the number of trees, l is the loss function, typically binomial deviance, and $\hat{y}_i$ is given by $\sum_{k=1}^{K} \mathbf{t}_k(\boldsymbol{x}_i)$ where $\mathbf{t}_k$ is a tree.

We use the `scikit-learn` (Pedregosa et al., 2011) implementation for the log-linear and gradient boosted models, and implemented NBSVM based on the interpolated version in Wang and Manning (2012).

As hyperparameters, such as learning rate and regularization, have a significant impact on model performance, we use Bayesian hyperparameter optimization (Bergstra et al., 2011) to select the optimal hyperparameters for each model on a held-out development set. We used the tree-structured Parzen Estimator (TPE) algorithm implemented in `hyperopt` for our sequential model-based optimization (Bergstra et al., 2013). After individually optimizing hyperparameters and training each of the three base models, we use their outputs to train a meta-ensemble model, a regularized conditional log-linear model, forming a linear combination over their predictions (Breiman, 1996).

As the lawmaking process in each state, and even within each chamber, is different, we divide the problem space by state and chamber, building separate models for each subset. Specifically, we consider each of these as separate problems: upper chamber bills and lower chamber bills. Thus, we have 2 predictions per state, and each prediction is comprised of 4 model outputs, three from the base models, and one from the meta-ensemble, resulting in 400 models.[8]

4.2 Features

As there are many dimensions underlying bills that may affect the likelihood of floor action, we compute and utilize several established contextual legislature and legislator derived features. Previous literature has proposed various factors that may affect legislation, including the content of bills,[9] number of and

[8] There are only upper chamber bills in D.C. and Nebraska, resulting in 49 states x 2 prediction types + 2 states x 1 prediction type) x 4 models = 400.

[9] In most previous literature the content is determined via a manual analysis of each bill to establish the scope of impact, the complexity, or the incremental nature.

Feature Type	Description
Sponsor	primary and cosponsor(s) identity, primary and cosponsors(s) party affiliation, number of primary and sponsors, number of Republicans, number of Democrats, sponsors bicameral, sponsors bipartisan, sponsor in majority/minority, majority party Republican or Democrat
Committee	identity of assigned committee(s), number of committee assignments, number of sponsors members of the committee, sponsor same party as committee chairman, sponsor role on the committee, referral rate of committee(s)
Bill	chamber, bill type, session, introductory date, companion bill(s) existence, companion(s) current status.

Table 1: Contextual feature types and descriptions.

identity of sponsors, extra-legislative forms of support, timing of introduction, leadership's position, seniority, identity of chairperson of the committee, identity of one's own party, and membership of the dominant faction (Hamm, 1980; Rakoff and Sarner, 1975; Harbridge, 2016; Yano et al., 2012).

In order to quantitatively evaluate these factors and establish a strong baseline from which to measure the affect of text, we include the contextual features shown in Table 1. These indicator features derived from the sponsors, committees, and bills are meant to capture many of the major factors that are proposed in the literature.[10] To strengthen the representation of legislators in our model beyond the basic features described above, we compute several measures of legislator effectiveness. The effectiveness score is calculated from the sponsoring and cosponsoring activity of each legislator, and meant to represent where they stand in relation to other legislators in successfully passing legislation.[11]

Similar to Harbridge (2016), the score we compute for each legislator is a combination of several partial scores, computed for each important stage of the legislative process. Each legislator gets a score for how many bills they sponsored, getting those bills out of committee, getting them to the floor, passing their own chamber, passing the legislature, and getting enacted. The score for each stage is further broken down by how many of those pieces of legislation were substantive, i.e. bills, attempting a meaningful legal change, versus non-substantive, i.e. resolutions. This results in 12 factors for each individual. To compute a score for each legislator's relative performance at each stage to the other members in the chamber, we create a weighted combination of that legislator's bills and resolutions, where bills get more weight, and compute the ratio based on the weighted contribution of the other members in the chamber. All the stage scores are then combined into a second weighted combination, where each successive stage in the process gets more weight, to get the final score. Finally, the scores are normalized to 0-10. In addition to using the effectiveness scores directly as features, we further compute and discretize several statistics derived from them, including ranks, percentiles, and deviations from the mean thereof.

To further enrich the bill representation beyond contextual information, we utilize the textual content of the bills. The legislation in our collection is comprised of long documents, with an average of 11 thousand words, often containing significant amounts of procedural language and pieces of extant statutes. As this can create additional challenges in identifying the salient points, for this work we chose to focus on a condensed amount of text, specifically the state provided title and description, that average 17 and 18 words, respectively.[12] Both are preprocessed by lowercasing and stemming. We compute the tf-idf weighting for n-grams of size (1,3) on the training data for each prediction task, and select the top 10k

[10]Each count based feature, such as number of sponsors, also spawns a number of discretized features, including ranks, percentiles, and deviations from the mean thereof. We automatically compute companion bills using a cosine-based lexical similarity.

[11]This is not a holistic representation of being an effective legislator, as someone may consider themselves effective by not passing anything, or preventing others from doing so. Members may also be highly influential and their support is needed behind the scenes but their names do not appear on the legislation. We can only account for recorded activity. Despite the limitations, we argue this is a fair, if incomplete, assessment of how well the legislator advances their agenda.

[12]Although this is a coarse approximation of the bill content, we believe it should capture the substantive aspects of the bill. Full details of the length of documents in each state are given in Table 7 in Appendix A.

Condition	Feature Set
`combined`	sponsor, committee, bill, text
`no_txt`	sponsor, committee, bill
`no_txt_spon`	committee, bill
`just_txt`	text
`just_spon`	sponsor

Table 2: The five feature settings with contextual and lexical features.

n-grams from the title and description separately.

While we would like to study the predictability of reaching the floor upon first introduction, bills often change after introduction and are updated with additional information. Thus, we limit our features to those available at the time of first introduction.

5 Results

In order to clarify the impact that each set of features described in Section 4.2 has on predictive performance, we create five different subsets of features described in Table 2, and train models on each one of them separately.

The first condition, `combined` contains all the contextual and text content features. The second condition, `no_txt`, removes text content, allowing us to study the importance of all contextual features, and by comparing `combined` to `no_txt` we can evaluate if text has any complementary information to contextual features. The third condition, `no_txt_spon` further removes sponsor features, essentially allowing us to study the importance of committee information. By comparing `no_txt` to `no_txt_spon` we can evaluate what sponsors contribute. The fourth and fifth conditions use only sponsor and only text features, respectively, to study the importance of each individually.

All models for a given condition are built from the same training data and feature space. We measure and report several performance metrics of our models using 10-fold cross validation. The baseline model represents guessing the majority class; for some states this means all fail, for others it is all receive floor action, based on the state specific rate.

Although accuracy is informative with respect to how many correct binary decisions the model made, as noted in Bradley (1997) for imbalanced problems such as this, where one class dominates, the baseline accuracy can be very high. As a supplement, it is useful to measure a probabilistic loss, where there is a cost associated with how correct the decision was. Thus, we move beyond pure predictive performance and consider the actual probability distributions created by our models under different conditions. The log-linear and gradient boosted models are probabilistic, while NBSVM is not, thus we train a probability transformation on top of NBSVM using Platts Scaling to obtain probability estimates.

In additional to accuracy, we measure model performance on log-loss and AUROC (area under the receiver operating characteristic curve) (Bradley, 1997). Log-loss, LL is defined as:

$$LL = -\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(y_i = \hat{y_i}) \log(p_i) + (1 - \mathbb{1}(y_i = \hat{y_i})) \log(1 - p_i) \tag{5}$$

where $\mathbb{1}(y_i = \hat{y_i})$ is an binary indicator function equaling 1 if the model prediction $\hat{y_i}$ was correct, and 0 otherwise. LL equals zero for a perfect classifier, and increases with worse probability estimates. Specifically, LL penalizes models more the more confident they are in an incorrect classification.

AUROC allows us to measure the relationship between a model's true positive (TP), how many floor action bills were correctly predicted as floor action, and false positive rate (FP), how many failed bills were predicted as floor action. It is defined by:

$$AUROC = \sum_{i=1}^{N} p(TP)\Delta p(FP) + \frac{1}{2}(\Delta p(TP)\Delta p(FP)) \tag{6}$$

Feature Set	Accuracy		Log-Loss		AUROC	
	Average	Std Dev	Average	Std Dev	Average	Std Dev
baseline	0.68	0.1	0.6	0.09	0.5	0
just_txt	0.732	0.09	0.53	0.14	0.7	0.14
just_spon	0.759	0.102	0.48	0.16	0.74	0.15
no_txt_spon	0.81	0.113	0.39	0.18	0.8	0.18
no_txt	0.846	0.098	0.32	0.18	0.82	0.21
combined	**0.859**	0.093	**0.31**	0.17	**0.85**	0.21

Table 3: Average and standard deviation across states on accuracy, log-loss, AUROC for bills on each feature set.

By considering the TP and FP at different values, we can construct an ROC curve. The area under that curve, AUROC, can be interpreted as the probability that the model will rank a uniformly selected positive instance (floor action) higher than a uniformly selected negative instance (failure), or in other words, the average rank of a positive example. A random model will have a AUROC of 0.5, and a 45-degree diagonal curve, while a perfect model will have an AUROC of 1, and be vertical, then horizontal.

Table 3 shows the average accuracy, LL, and AUROC with standard deviations for each of the five conditions. The `just_txt` model achieves an accuracy of 73%, outperforming the baseline by 5%, and notably, shows that there is a predictive signal even within the limited amount of text available in the title and descriptions.

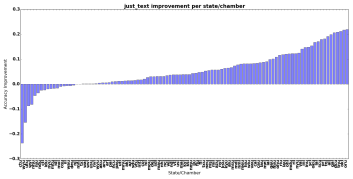
To examine where text content is most and least predictive on its own, we disentangle the average performance of the `just_txt` model in Figure 2a, showing the per state and chamber pair change from baseline. The states that improve the most over baseline, with 15% improvement or more using only textual features are Oregon, Oklahoma, Tennessee, D.C., South Carolina, Louisiana (lower), Georgia (lower), and Alabama (lower). On the other hand, text is least predictive in Connecticut, Wyoming, Idaho, New Jersey, Utah (upper), New Hampshire (upper), North Dakota (upper), all underperforming the baseline.

The relatively small improvement over baseline of `just_txt` provides insight into the lawmaking process, raising the possibility that other contextual factors, outside the subject matter of the legislation, such as who the sponsors are and what committee the bill is assigned to, are often more important than the subject of the legislation.

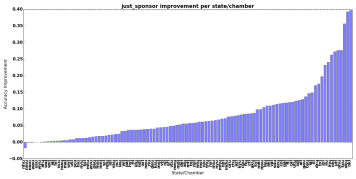
The `just_spon` model achieves an average accuracy of 76%, slightly outperforming `just_txt` with an improvement over baseline of 8%. This further indicates that knowing sponsor related information, without reference to the subject of the legislation, is itself highly predictive. In fact, Figure 2b shows that except for New Hampshire (upper), almost all states achieve gains using sponsor only information, with Oklahoma, Texas, and Ohio achieving gains of 30% or more. The committee information in `no_txt_spon`, which includes the sponsor committee positions, is even more predictive than sponsor and text only, and the addition of sponsors in `no_txt` improves performance by 3.5%.

Including text in the `combined` model further improves performance by 1.3% over `no_txt`, and 18% over the majority class baseline, showing the complementary effects of contextual and lexical information, as this model consistently outperforms all others. Figure 2c shows the per state and chamber pair `baseline` and `combined` model performance. The AUROC performance follows a very similar trajectory.

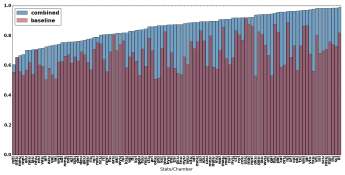
On LL, the model performance follows a similar path, with all models showing improvement in probability estimates from the baseline. LL almost doubles from the `combined` model's 0.31 to 0.6 on `baseline`. This reinforces that the `combined` model makes very confident correct predictions. Including text in the `combined` improves performance slightly over `no_txt`, while having just sponsors or just text decreases the LL to around 0.5.



(a) Change from baseline with text only features.



(b) Change from baseline with sponsor only features.



(c) Prediction accuracy on bills with combined model.

Figure 2: Performance improvements for text-only (2a) and sponsor-only models (2b), and combined performance (2c)

6 Analysis

All contextual and lexical features considered above are available upon the introduction of a bill, or shortly thereafter,[13] thus the evaluation above indicates how well floor action can be predicted from the day of introduction. However, after the bill is introduced, subsequent legislative actions indicate further contextual information about the legislative process. As it is reasonable to assume these actions carry relevant predictive information, we further examine subsequent events in the legislative process in the `combined+act` feature set. We include a binary feature for the occurrence of amendment introduction and outcomes, votes, committee referral outcomes and readings up to the point of floor action. By comparing `combined+act` to `combined` we can examine how important different events in the legislative procedure are to predicting floor action.

Table 4: Average accuracy, log-loss, AUROC for bills using legislative events post introduction.

Feature Set	Accuracy		Log-Loss		AUROC	
	Average	SD	Average	SD	Average	SD
`combined`	0.859	0.093	0.31	0.17	0.85	0.21
`combined+act`	**0.94**	0.059	**0.16**	0.12	**0.97**	0.04

Table 4 shows the results of the `combined+act` feature set. Accuracy improves to 0.94, while LL drops by half to 0.16, confirming that legislative events occurring up to the point point of floor action carry significant complementary information to other contextual factors and are highly indicative of floor action. While `combined+act` confirms the predictive power of procedural factors outside the legislative text, sponsor, and committee assignment, the `combined` model is arguably the most important result, as it indicates how well we can predict on features that are available upon introduction.

Beyond the predictions, we are interested in identifying the different features that contribute to legislative success across the states. As there are both a large number of models, and features in each model, in order to understand the relative predictive importance of contextual and legislature specific dynamics, we choose several previously proposed factors deemed to be important for floor action, and compare the rank and weight they received in each model.

We first examine the median rank and weight given to the following features in the `just_spon` condition across all states: bipartisan, sponsor in minority, sponsor in majority, and the number of sponsors. While many of these contextual features are highly ranked, there are many variations and differences across states. The top half of Table 5 shows the top ten states for which each feature was ranked among the top 20. For example, the bipartisan feature is ranked in the top 5 in Missouri, Virginia, Maine, and Mississippi, accounting for up to 6% of the explanatory power. As a comparison, in South Dakota, Hawaii, Minnesota, Wisconsin, and Pennsylvania, bipartisanship ranked lower than 200. Whether the sponsor is in the minority is important in the U.S. Congress, where it is ranked 6^{th}, along with Delaware, Tennessee, and West Virginia. Being in the majority accounts for 10% in Kentucky, and 7% in Wisconsin. This aligns with previous literature, as Wisconsin is known to have a strong party system (Hamm,

[13] Some states do not indicate committee assignment immediately, for those we include the first assignment after introduction.

Feature	Median	Top	Top States	Bottom States
Bipartisan	64	11	mo,va,me,ms,nc,sc,ak,de,wa,us	sd,mn,wi,pa,ut,ne,id,fl,dc,ar
in Min	24	20	de,us,wv,tn,ia,wi,al,nd,md,mi	ca, il,hi,tx,nj,ut,ne,fl,dc,ar
in Maj	23	22	wi,mn,tn,ky,nh,nc,co,il,al,oh	ak,tx,ma,va,ne,nj,me,ut,fl,ar
Num Spon	28	20	co, ut,il,vt,in,ia,or,sd,oh,us	pa, az,wa,wy,nm,nv,va,ms,mn,ar
Ranking Mbr	24	15	ne,vt,ar,ky,us,ga,ok,me,ny,or	ks,mo,mt,oh,pa,ri,tn,ut,va,wy
No Cmte Mbr	17	23	ar,me,ne,il,sd,nc,nv,nm,de,ky	hi,id,ia,ks,mt,oh,pa,tn,ut,wy
Members	6	33	de,ct,me,sd,nc,nv,ok,ny,ga,il	hi,id,ia,ks,mt,oh,pa,tn,ut,wy

Table 5: Median ranking of across states for bipartisan, sponsor in minority, sponsor in majority, and number of sponsor features for sponsor only model, and having a ranking majority of the committee as a sponsor, not being a committee member as a sponsor, and being a member of the committee as a sponsor in the committee model. The top column indicates how many states have that feature ranked within the top 20 weighted features. The top states lists the ten states where each feature was ranked the highest and was one of the first 20 features. The bottom rows lists the ten states where each feature was ranked the worst.

1980), and indeed we find sponsor in majority and in minority features to be ranked 1^{st} and 9^{th}, respectively, while in Texas, which has a much weaker party system, those features are ranked among the lowest of all states.[14]

Similar ranking is presented for committee features in the bottom half of Table 5 in the `no_txt_spon` condition. The committee features play a similarly predictive role, with the sponsors holding membership positions on the committee accounting for over 10% of explanatory power in Delaware, Connecticut, Maine, and South Dakota.

To examine the difference in probability assigned by the models under different conditions, we chose a representative example where neither contextual nor lexical features dominate, as shown in Figure 2, and show the boxplots for Pennsylvania's lower chamber in Figure 3. Each subfigure shows the probability of floor action for legislation that received floor action (pass) and did not (failed). In all cases, the median of the probabilities on legislation that received floor action is higher than the median of the probabilities on legislation that failed. The `combined` models median and mean predictions on bills receiving action are above 90%, and it has the largest difference between the two cases. The `no_txt` model has a similar mean, but the probabilities become more distributed on both pass and fail. Removing sponsors significantly affects the distribution, and shifts the mean lower to 70%. `just_spon` and `just_text` both drop the mean to around 40%.

In addition, we show the calibration curves and distribution of predictions for the same settings in Figure 4 and ROC curves in Figure 5 in Appendix A. All models are well calibrated, closely following the diagonal line. The `combined` model is very confident in its predictions, resembling a bimodal distribution, placing most predictions close to either 0 or 1 probability. `just_spon` has the most distributed probability estimates, while `just_txt` moves the lower part of the distribution slightly forward. The `combined` model is quite accurate, with each subsequent model moving the ROC curve to the right, and thus allowing more false positives to reach the same true positive rate.[15]

Finally, we examine language ranked most and least predictive on the `just_txt` condition for New Mexico, Pennsylvania, and New York in Table 6.[16] Previous literature has proposed several theories on how content affects legislative passage, including that the more redistributive a policy is perceived, the higher in controversy, or the greater in scope, the lower the passage likelihood (Rakoff and Sarner, 1975; Hamm, 1980). While each state has a unique set of issues that are likely to be taken to the floor, and conversely, to be left in committee, there is also evident overlap. In the top phrases, several states contain

[14]Full rankings and weights are presented in Table 8 in Appendix A.

[15]For comparison to a state with a higher rate of floor action, we include analogous figures for California's lower chamber in Appendix A.

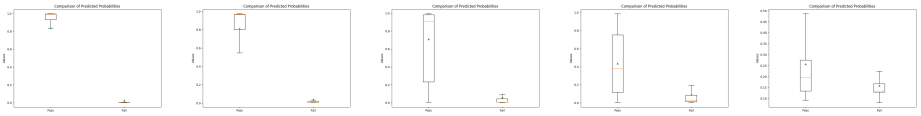
[16]Addition states are presented in Table 9 in Appendix A.

State	Top Phrases	Bottom Phrases
New Mexico (upper)	day, campus, recognit, month, defin, alcohol, date, recipi, procur, cours, registr plate, revis,	tax credit chang, enmu, residenti, lobbi, statewid, or, abort, safeti, date for, test for, primari care, analysi,
New Mexico (lower)	day, studi, length, citi, of nm, fingerprint, geotherm, fund project, dog, definit, loan for, month,	of game fish, peac, senior citizen, math scienc, transfer of, state fair, self, bachelor, develop tax credit, nmhu, wolf, equip tax,
Pennsylvania (upper)	provid for alloc, creation of board, manufactur or, an appropri to, of applic and, medic examin, fiscal offic, for request for, corpor power, within the general, for the offic,	an act amend, as the tax, known the, act provid, known the tax, wage, act prohibit, citizen, of pennsylvania further, tax, youth, requir the depart,
Pennsylvania (lower)	or the, contract further, and for special, memori highwai, within the general, in game, first class township, whistleblow, emerg telephon, offens of sexual, for promulg,	act amend titl, an act amend, act provid, known, act prohibit, amend the, pennsylvania, an act provid, code of, act establish, an act relat, the constitut,
New York (upper)	fiscal year relat, memori highway, year relat to, implement the health, for retroact real, portion of state, the public protect, implement the public, inc to appli, budget author, program in relat, which are necessari	languag assist, direct the superintend, the develop of, author shall, subsidi, automobil insur, such elect, limit profit, disabl act, polici base, polici to provid
New York (lower)	care insur, applic for real, physic educ, fire district elect, establish credit, to file an, abolit or, hous program, the suspens of, are necessari to, the membership of, relat to hous	appropri, fuel and, numer, school ground, vehicular, incom tax for, prohibit public, tag, senat and assembl, on school, on school, class feloni

Table 6: Top and bottom ranked phrases for New Mexico, Pennsylvania, and New York.

budgetary issues, expressed with fiscal and appropriation language, as most states have to pass budgetary measures. We also see commendation and procedural language, which is often less contentious. In the bottom phrases, several states have tax related language, and several education related topics.

While outside the scope of this work, in future work we hope to explore the differing language identified by the model to help identify important questions about the policymaking process in each state, and allow comparison within states of what successful legislation contains, and across states, of how different issues take shape. In addition, as we only included a limited amount of text, we would like to explore how to incorporate the full body text of legislation effectively.



(a) `combined` (b) `no_txt` (c) `no_txt_spon` (d) `just_spon` (e) `just_txt`

Figure 3: Box plot distributions of predicted probabilities for legislation in Pennsylvania lower chamber. The box extends from the lower to upper quartile values of the predictions, with a line at the median, and triangle at the mean.

7 Conclusion

In this paper we explored the state legislative process by introducing the task of predicting floor action across all 50 states and D.C. We presented several baseline models and showed that combining contextual information about the legislators and legislatures with bill text consistently provides the best predictions, achieving an accuracy of 86% on which legislation will reach the floor upon first introduction. We further analyzed various factors and their respective importance in the predictive models across the states, gaining a broader understanding of state legislative dynamics. While the factors that influence legislative floor action success are diverse and understandably inconsistent among states, by examining them we can empirically help elucidate the similarities and differences of the policymaking processes.

References

James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2546–2554. Curran Associates, Inc.

James Bergstra, Dan Yamins, and David D. Cox. 2013. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in Science Conference*.

Marianne Bertrand, Matilde Bombardini, Raymond Fisman, and Francesco Trebbi. 2018. Tax-exempt lobbying: Corporate philanthropy as a tool for political influence. Nber working papers.

Andrew P. Bradley. 1997. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recogn.*, 30(7):1145–1159, July.

Leo Breiman. 1996. Stacked regressions. *Machine Learning*, 24(1):49–64, Jul.

Kathy Canfield-Davis, Sachin Jain, Don Wattam, Jerry McMurtry, and Mike Johnson. 2010. Factors of influence on legislative decision making: A descriptive study. *Journal of Legal, Ethical and Regulatory Issues*, 13(2).

Joshua Clinton, Simon Jackman, and Douglas Rivers. 2004. The statistical analysis of roll call data. *American Political Science Review*, 98(02):355–370.

Lauren Cohen, Karl B. Diether, and Christopher Malloy. 2012. Legislating Stock Prices. NBER Working Papers 18291, August.

James H. Fowler. 2006. Connecting the congress: A study of cosponsorship networks. *Political Analysis*, 14(4):456487.

Wayne L. Francis. 1989. *The Legislative Committee Game: A Compartative Analysis of Fifty States*. Columbus: Ohio State University Press.

Jerome H. Friedman. 2000. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232.

Sean Gerrish and David M. Blei. 2011. Predicting legislative roll calls from text. In *Proceedigns of ICML*.

Virginia Gray and David Lowery. 1995. Interest representation and democratic gridlock. *Legislative Studies Quarterly*, 20(4):531–552.

Keith Hamm, Ronald Hedlund, and Nancy Miller. 2014. State legislatures. In Donald Haider-Markel, editor, *The Oxford Handbook of State and Local Government*, chapter 13. Oxford University Press.

Keith E. Hamm. 1980. U. s. state legislative committee decisions: Similar results in different settings. *Legislative Studies Quarterly*, 5(1):31–54.

Laurel M Harbridge. 2016. Legislative effectiveness in the united states congress: The lawmakers. by craig volden and alan e. wiseman. *The Journal of Politics*, 78(1).

David Hedge. 1998. *Governance And The Changing American States*. New York: Routledge.

William Hicks and Daniel Smith. 2009. Do parties matter? explaining legislative productivity in the american states. In *The State of the Parties: 2008 and Beyond Conference*.

Mohit Iyyer, Peter Enns, Jordan L. Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Proceedings of ACL*.

Daniel Martin Katz, Michael J. Bommarito, and Josh Blackman. 2017. A general approach for predicting the behavior of the supreme court of the united states. *PLoS ONE 12.4*.

Anastassia Kornilova, Daniel Argyle, and Vlad Eidelman. 2018. Party matters: Enhancing legislative embeddings with author attributes for vote prediction. In *Proceedings of ACL*.

Benjamin E. Lauderdale and Tom S. Clark. 2014. Scaling politically meaningful dimensions using texts and votes. *American Journal of Political Science*, 58(3):754–771.

Fridolin Linder, Bruce A. Desmarais, Matthew Burgess, and Eugenia Giraudy. 2018. Text as policy: Measuring policy similarity through bill text reuse. *SSRN*.

Hill Matthew, Kelly Wayne, Lockhart Brandon, and Ness Robert. 2013. Determinants and effects of corporate lobbying. *Financial Management*, 42(4):931–957.

John J. Nay. 2016. Predicting and understanding law-making with machine learning. *CoRR*, abs/1607.02109.

Viet-An Nguyen, Jordan L. Boyd-Graber, Philip Resnik, and Kristina Miler. 2015. Tea party in the house: A hierarchical ideal point topic model and its application to republican legislators in the 112th congress. In *Proceedings of ACL*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, November.

Keith T Poole and Howard Rosenthal. 1985. A spatial model for legislative roll call analysis. *American Journal of Political Science*, pages 357–384.

Keith T Poole and Howard L Rosenthal. 2007. *Ideology and Congress*. New Brunswick, NJ: Transaction Publishers.

Stuart H. Rakoff and Ronald Sarner. 1975. Bill history analysis: A probability model of the state legislative process. *Polity*, 7(3):402–414.

Alan Rosenthal. 1974. *Legislative Performance in the States: Explorations of Committee Behavior*. New York: Free Press.

Boris Shor and Nolan McCarty. 2011. The ideological mapping of american legislatures. *American Political Science Review*, 105(03):530–551.

Boris Shor, Christopher Berry, and Nolan McCarty. 2010. A bridge to somewhere: Mapping state and congressional ideology on a cross-institutional common space. *Legislative Studies Quarterly*, 35(3):417–448.

Jonathan B Slapin and Sven-Oliver Proksch. 2008. A scaling model for estimating time-series party positions from texts. *American Journal of Political Science*, 52(3):705–722.

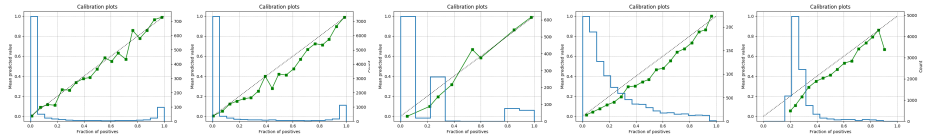
Jeffery C. Talbert and Matthew Potoski. 2002. Setting the legislative agenda: The dimensional structure of bill cosponsoring and floor voting. *The Journal of Politics*, 64(3).

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *Proceedings of EMNLP*.

Sida I. Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL (2)*, pages 90–94. The Association for Computer Linguistics.

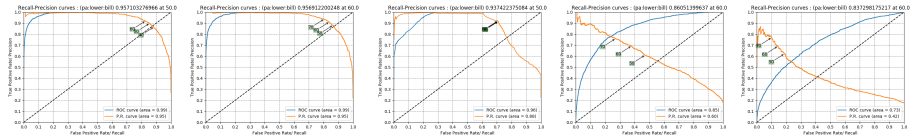
Tae Yano, Noah A. Smith, and John D. Wilkerson. 2012. Textual predictors of bill survival in congressional committees. In *Proceedings of NAACL*.

A Appendix



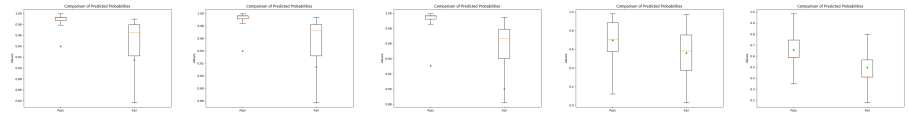
(a) `combined` (b) `no_txt` (c) `no_txt_spon` (d) `just_spon` (e) `just_txt`

Figure 4: Calibration plots for Pennsylvania lower chamber.



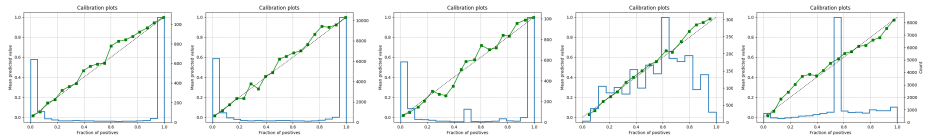
(a) `combined` (b) `no_txt` (c) `no_txt_spon` (d) `just_spon` (e) `just_txt`

Figure 5: ROC curves and AUC for Pennsylvania lower chamber. Green pointers indicate probability thresholds on the Recall-Precision curve, and the title includes accuracy at the best performing threshold.



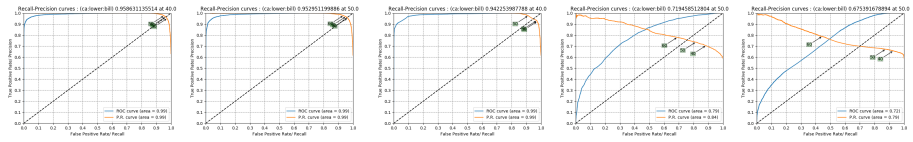
(a) `combined` (b) `no_txt` (c) `no_txt_spon` (d) `just_spon` (e) `just_txt`

Figure 6: Distributions of predicted probabilities for legislation in California lower chamber that received floor action (pass) and did not (fail) in a box plot. The box extends from the lower to upper quartile values of the predictions, with a line at the median, and triangle at the mean.



(a) `combined` (b) `no_txt` (c) `no_txt_spon` (d) `just_spon` (e) `just_txt`

Figure 7: Calibration plots for California lower chamber.



(a) `combined` (b) `no_txt` (c) `no_txt_spon` (d) `just_spon` (e) `just_txt`

Figure 8: ROC curves and AUC for California lower chamber. Green pointers indicate probability thresholds on the Recall-Precision curve, and the title includes accuracy at the best performing threshold.

State	Floor Action	Introduced	Rate	Sessions	Title	Desc	Body
al	6697	14327	0.467	16	21	-	11280
ak	781	2527	0.309	4	34	-	12054
az	3719	9308	0.4	24	4	-	13284
ar	3809	5076	0.75	8	17	-	7297
ca	18978	32143	0.59	17	5	186	16631
co	4808	6428	0.748	11	5	20	10671
ct	3044	16236	0.187	8	12	19	4748
de	3185	4858	0.656	6	19	89	9327
dc	8515	15593	0.546	9	11	-	7083
fl	6592	21298	0.31	15	6	29	16471
ga	7416	15379	0.482	13	8	43	5654
hi	5630	21615	0.26	5	9	28	6910
id	3259	4446	0.733	8	40	-	10458
il	14106	66926	0.211	10	3	46	7022
in	1958	5291	0.37	4	5	55	16918
ia	3434	21457	0.16	7	28	-	12740
ks	2123	6324	0.336	6	12	-	22294
ky	3149	8185	0.385	20	51	52	14218
la	18346	35277	0.52	32	20	-	7454
me	9268	17095	0.542	8	14	-	5583
md	9857	26125	0.377	14	10	59	8107
ma	6862	52467	0.131	7	11	26	14535
mi	14520	41730	0.348	11	27	-	10929
mn	4494	27240	0.165	10	11	-	10251
ms	6621	25450	0.26	21	12	77	14475
mo	2736	14143	0.193	8	18	-	16486
mt	5910	9905	0.597	8	7	-	13758
ne	1837	4829	0.38	6	12	-	11775
nv	2614	4163	0.628	7	14	12	21165
nh	3243	6793	0.477	6	12	-	3269
nj	6900	59861	0.115	8	16	-	12883
nm	4253	10909	0.39	8	5	-	11639
ny	23071	89072	0.259	4	16	23	8913
nc	6922	25152	0.275	10	4	-	7039
nd	5735	8089	0.709	9	42	3	5847
oh	4356	8605	0.506	9	8	-	33216
ok	16579	36827	0.45	10	13	-	10333
or	5240	14404	0.364	12	9	54	14500
pa	2887	16414	0.176	5	32	-	5466
ri	6596	16584	0.398	5	31	-	8838
sc	3269	11532	0.283	6	77	7	6255
sd	1647	2539	0.649	9	16	-	6437
tn	33936	77331	0.439	12	30	-	3256
tx	8371	25771	0.325	9	16	-	5145
ut	7816	11072	0.706	23	7	-	24623
vt	1035	4520	0.229	4	14	-	6046
va	14215	27813	0.511	24	9	37	8310
wa	8317	24578	0.338	6	10	-	14735
wv	4308	23917	0.18	12	12	10	11501
wi	4982	13761	0.362	14	40	2	12090
wy	2825	4223	0.669	11	4	38	7032
us	22973	172921	0.133	15	14	178	14043

Table 7: Data statistics for number of bills introduced and receiving floor action for each state. Word counts are given for title, description, and bill body.

State	Bipartisan		in Min		in Maj		Num Spon		AP Eff		BP Eff	
	Rank	Weight	Rank	Weight	Rank	Weight	Rank	Weight	Rank	Weight	Rank	Weight
al	57	0.003	12	0.012	5	0.017	54	0.003	17	0.015	95	0.003
ak	9	0.029	22	0.009	57	0.002	62	0.006	3	0.059	3	0.046
az	133	0.003	46	0.004	28	0.008	69	0.006	89	0.003	11	0.012
ar	-	-	-	-	-	-	-	-	0	0.326	1	0.236
ca	43	0.006	108	0.003	29	0.007	13	0.01	70	0.003	3	0.027
co	49	0.006	19	0.014	3	0.025	0	0.137	94	0.002	135	0.0
ct	128	0.001	75	0.003	26	0.008	58	0.004	210	0.001	-	-
de	13	0.013	3	0.023	36	0.013	22	0.009	79	0.003	115	0.002
dc	-	-	-	-	16	0.015	33	0.011	-	-	-	-
fl	-	-	-	-	-	-	27	0.012	1	0.072	0	0.137
ga	80	0.003	23	0.01	9	0.015	32	0.006	303	0.0	291	0.0
hi	224	0.0	120	0.001	26	0.006	12	0.009	136	0.001	0	0.321
id	-	-	24	0.001	28	0.001	24	0.002	14	0.018	-	-
il	159	0.001	109	0.015	3	0.037	2	0.015	179	0.0	24	0.006
in	43	0.005	101	0.002	50	0.009	2	0.044	42	0.007	48	0.004
ia	98	0.001	8	0.026	14	0.015	3	0.086	192	0.0	-	-
ks	112	0.0	69	0.001	32	0.007	16	0.016	39	0.009	61	0.005
ky	35	0.004	38	0.006	2	0.103	67	0.004	-	-	-	-
la	32	0.004	30	0.005	24	0.006	14	0.053	23	0.005	35	0.003
me	5	0.055	21	0.012	-	-	21	0.013	40	0.001	-	-
md	36	0.006	14	0.011	22	0.008	33	0.006	4	0.049	7	0.029
ma	151	0.004	69	0.003	66	0.003	16	0.009	35	0.009	3	0.045
mi	80	0.002	14	0.01	10	0.033	28	0.006	98	0.002	138	0.001
mn	231	0.0	60	0.003	1	0.046	283	0.0	156	0.003	140	0.003
ms	5	0.029	30	0.01	32	0.018	107	0.003	58	0.005	32	0.007
mo	3	0.049	19	0.008	17	0.019	49	0.003	90	0.001	118	0.001
mt	123	0.003	18	0.022	11	0.019	16	0.017	-	-	-	-
ne	-	-	-	-	275	0.0	19	0.01	0	0.057	3	0.037
nv	20	0.01	33	0.006	35	0.006	101	0.002	2	0.072	6	0.044
nh	21	0.012	57	0.011	2	0.089	42	0.004	44	0.005	27	0.009
nj	66	0.004	352	0.0	363	0.0	14	0.01	397	0.0	340	0.0
nm	225	0.0	60	0.006	34	0.009	91	0.002	69	0.005	101	0.004
ny	167	0.002	20	0.012	20	0.014	36	0.004	47	0.007	49	0.006
nc	8	0.017	18	0.026	3	0.058	30	0.007	37	0.005	43	0.003
nd	121	0.002	13	0.016	14	0.02	12	0.014	-	-	-	-
oh	207	0.001	18	0.01	5	0.033	10	0.026	29	0.019	21	0.019
ok	116	0.0	82	0.0	21	0.001	66	0.46	134	0.0	174	0.0
or	32	0.012	34	0.006	27	0.007	4	0.025	40	0.007	2	0.029
pa	333	0.001	18	0.014	20	0.013	68	0.004	16	0.01	36	0.01
ri	126	0.002	16	0.022	24	0.008	42	0.006	-	-	-	-
sc	8	0.019	19	0.01	14	0.014	37	0.007	-	-	-	-
sd	227	0.0	46	0.006	27	0.0	7	0.042	-	-	-	-
tn	62	0.004	7	0.016	2	0.023	17	0.012	96	0.003	68	0.006
tx	147	0.003	212	0.001	61	0.003	22	0.014	112	0.001	108	0.002
ut	-	-	-	-	-	-	1	0.032	-	-	19	0.003
vt	46	0.004	53	0.003	44	0.004	2	0.084	20	0.019	50	0.097
va	4	0.048	59	0.003	139	0.001	101	0.002	136	0.001	222	0.001
wa	16	0.014	20	0.011	24	0.011	73	0.003	51	0.004	111	0.002
wv	35	0.004	7	0.018	6	0.042	34	0.003	2	0.099	7	0.021
wi	232	0.002	9	0.024	1	0.072	47	0.007	5	0.022	1	0.103
wy	162	0.0	49	0.004	48	0.004	85	0.002	-	-	-	-
us	20	0.016	6	0.029	6	0.025	10	0.01	16	0.02	16	0.013

Table 8: Feature ranks and weight for bipartisan, sponsor in minority, sponsor in majority, number of sponsors, average primary sponsor effectiveness and best primary sponsor effectiveness features in the gradient boosted model with just sponsor features across all states. Dash indicates feature was not ranked within the top 400.

State	Top Phrases	Bottom Phrases
New Jersey (upper)	for farmland preserv, preserv trust, green acr fund, acquisit and, mmvv million from, vehicl from, budget for, fund for state, in feder fund, unemploy, for state acquisit, infrastructur trust	retir benefit for, of educ for, school board member, clarifi law, contract and, tax reimburs program, appropri mmvv for, to develop and, tax rate, credit under corpor, certain vehicl
New Jersey (lower)	environment infrastructur, to dissemin, farm to, dmva to, concern certain, and dhs, unsolicit, atm, contract law, link to, manufactur rebat, limit liabil	polit, import, all school, for water, facil to be, respons for, grant program for, relat crime, state administ, from tax, chair, to all
Maryland (upper)	financ the construct, festiv licens, issu the licens, grante provid and, to effect, advisori commiss, an evalu of, that provis of, financ statement, board licens, to borrow, defer	not to, phase, be use as, use as, facil locat in, to own, law petit, or expenditur, trust establish, expend match, and expend match
Maryland (lower)	charl counti alcohol, improv or, to financ the, termin provis relat, counti sale, sanction, alter, counti special tax, montgomeri counti alcohol, report requir repeal, length, licens mc	grant to the, creation state debt, educ fund, state debt baltimor, establish the amount, elimin, disclos to, propos amend, incom tax rate, purpos relat, crimin gang, deced die after
California (upper)	ab, revolv fund, household, intent that, these provis until, restitut, counsel, employe, onli if ab, properti, if ab, would incorpor addit	legisl, cost of, veterinari, enact legisl, to the, law, regul econom, governor, incom tax deduct, hour, motor vehicl recreat, decis
California (lower)	add articl, to amend repeal, bill would incorpor, to add and, budget act of, urgenc statut, make nonsubstant, and make, as bill provid, relat the budget, and of the, ab	would make nonsubstant, enact legisl, make technic nonsubstant, would make technic, unspecifi, code to add, baccalaur degre, salari, fraud prevent, flexibl, of the state, would
Florida (upper)	ogsr, abrog provis relat, grant trust fund, govern act, person inform, to supplement, employ contribut to, legisl audit committe, jac, maintain by the, insur regul, financi inform	senat relat to, senat relat, to, ssb, elder, school, municip that, and legislatur by, that law enforc, provid minimum, admiss to, local law enforc
Florida (lower)	etc, certain propos, re creat, repeal under, to qualifi, boundari, program revis requir, environment permit, counti hospit district, alcohol beverag licens, except under, ranch	hous relat, day, renew energi, provid for alloc, make recommend, for employ of, of damag, from particip, week, catastroph, dhsmv to develop, employ from
Delaware (upper)	uniform, would increas the, amend chapter volum, person convict, relat the delawar, dealer, child support, bureau, violenc, associ, charter chang, for fiscal year	rent, state languag, the content, act regul, certain licens, give local, assembl from, delawar code establish, for citizen, reimburs, propos constitut amend, salari
Delaware (lower)	of the th, tax refund, thi act also, amend of the, this section of, the titl, the act to, of member of, electron transmiss, for in the, and the date, parent guardian	predatori, hour per, relat state employe, unfair practic, communic, open meet, equal the, to the construct, the construct, medicaid, state agenc, relat to prevail

Table 9: Top and bottom ranked phrases for New Jersey, Maryland, California, and Florida.

Learning to Search in Long Documents Using Document Structure

Mor Geva
Tel Aviv University
morgeva@mail.tau.ac.il

Jonathan Berant
Tel Aviv University
joberant@cs.tau.ac.il

Abstract

Reading comprehension models are based on recurrent neural networks that sequentially process the document tokens. As interest turns to answering more complex questions over longer documents, sequential reading of large portions of text becomes a substantial bottleneck. Inspired by how humans use document structure, we propose a novel framework for reading comprehension. We represent documents as trees, and model an agent that learns to interleave quick navigation through the document tree with more expensive answer extraction. To encourage exploration of the document tree, we propose a new algorithm, based on Deep Q-Network (DQN), which strategically samples tree nodes at training time. Empirically we find our algorithm improves question answering performance compared to DQN and a strong information-retrieval (IR) baseline, and that ensembling our model with the IR baseline results in further gains in performance.

1 Introduction

Reading comprehension (RC), the task of reading documents and answering questions about their content, has attracted immense attention recently. While early work focused on simple questions and short paragraphs (Hermann et al., 2015; Rajpurkar et al., 2016; Trischler et al., 2017; Onishi et al., 2016), current work is shifting towards more complex questions that require reasoning over long documents (Joshi et al., 2017; Hewlett et al., 2016; Welbl et al., 2017; Kočisky et al., 2017).

Long documents pose a challenge for current RC models, as they are dominated by recurrent neural networks (RNNs) (Chen et al., 2016; Kadlec et al., 2016; Xiong et al., 2017). RNNs process documents token-by-token, and thus using them for long documents is prohibitive. A common solution is to retrieve part of the document with an IR approach (Chen et al., 2017; Clark and Gardner, 2017) or a cheap model (Watanabe et al., 2017), and run an RNN over the retrieved excerpts. However, as documents become longer and questions become complex, two problems emerge: (a) retrieving all the necessary text with a one-shot IR method when performing complex reasoning becomes harder, and thus thousands of tokens are retrieved (Clark and Gardner, 2017). (b) Running even a cheap model over the document in its entirety becomes expensive (Choi et al., 2017).

Humans, in lieu of a mental inverted index, use document structure to guide their search for answers. E.g., the answer to *"What high school did Leonard Cohen go to?"* is likely to appear in *"Early life"*, while the answer to *"How hot is it in Melbourne in July?"* is likely to appear in *"Climate"*. In this work we investigate whether we can train a model to navigate through a document using its structure and find the answer, while reading only a small portion of the entire document.

We represent documents as trees and train an agent that navigates through the document tree until returning a final answer. Figure 1 illustrates this process. Our agent reads the question *"The vacation destinations of Pattaya and Phuket are in which country?"* and starts navigation at the title of the document. After reading a paragraph, and skipping *"History"*, it drills down to *"Geography"* until finally halting at a paragraph that specifies the answer (*"Thailand"*). The agent observes at each step only a glimpse of the local text to determine its next action, which can be movement to a tree node, answering

Proceedings of the 27th International Conference on Computational Linguistics, pages 161–176
Santa Fe, New Mexico, USA, August 20-26, 2018.

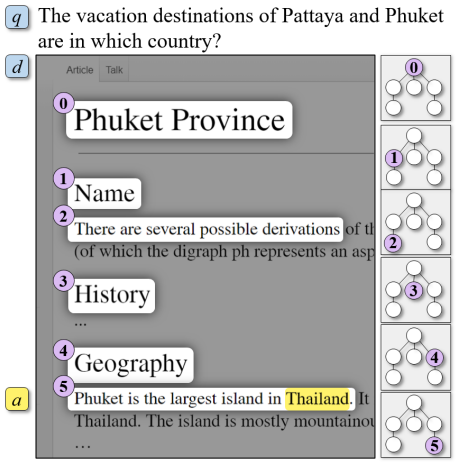


Figure 1: An overview of our framework: an agent answers the question q for a document d by starting at the title, performing navigation actions until reaching the relevant paragraph, extracting the answer a, and then stopping.

the question with a more expensive RC model, or terminating navigation. Thus, the agent consumes only a small fraction of the entire document.

Our training data is question-document-answer triplets, without gold navigation paths, and thus we train our model with the Deep Q-Network (DQN) algorithm. Because the dataset is biased towards answers appearing at the beginning of the document, the algorithm tends to stop early and does not explore the document well. To overcome this challenge, we propose DocQN: a variant of DQN for tree navigation that improves exploration by sampling nodes from multiple parts of the tree.

We evaluate the ability of our agent to navigate to paragraphs containing the answer on a variant of TriviaQA (Joshi et al., 2017) and find that: (a) DocQN navigates better than DQN in documents both quantitatively and qualitatively. (b) While DocQN observes only 6% of the document tokens, it outperforms an IR method in end-to-end QA performance. (c) An ensemble of DocQN and IR substantially improves both navigation and end-to-end QA performance over the ensemble components.

To summarize, in this paper we ask: can an agent use document structure and learn to find answers for complex questions in long documents? We propose a new model and training algorithm that overcomes an inherent bias in the data, answering the aforementioned question in the affirmative. Our code and dataset are available at `https://github.com/mega002/DocQN`.

2 Problem Overview

We work in the traditional RC setup, where we are given question-document-answer triplets $\{(q_i, d_i, a_i)\}_{i=1}^{N}$ as a training set, and aim to learn a function that finds the answer for an unseen question-document pair. Unlike prior work, we assume documents are trees, where every tree node u corresponds to a structural element and is labeled with text $l(u)$. Specifically, the root is labeled with the document title, sections and subsections are labeled by their title, and paragraphs and sentences are labeled by the text they contain. In addition, we order all non-sentence tree nodes by a pre-order traversal (which corresponds to the linear order of text in the document), and denote the index of a node u by $n(u)$. For sentence nodes, $n(u)$ is the index of their parent (a paragraph).

Figure 1 shows an example tree, where for each node we show the relevant structural element and index (sentence nodes are not shown in the figure).

With this document representation, answering questions can be viewed as a Markov Decision Process (MDP), where in each state the agent is located in a particular tree node, actions allow movement through the document tree, answering the question with a RC model, or stopping, and a reward is based on whether the agent locates a node that contains the answer.

Our agent interleaves actions that *navigate* in the document, with an action that runs an RC model in a certain document position and *extracts an answer*. Thus, the agent can decide to continue navigation after extracting a certain answer. This is strictly more expressive than existing approaches that combine

		TRIVIAQA	TRIVIAQA-NOP
Train	Questions	61,888	57,220
	Documents	110,647	99,315
Development	Questions	7,993	7,336
	Documents	14,229	12,706
Test	Questions	7,701	6,507
	Documents	13,661	10,481

Table 1: Data statistics for TRIVIAQA vs. TRIVIAQA-NOP.

Average number of tokens	5590.7
Average number of tree nodes	332.2
Average number of high-level sections	6.6

Table 2: Data statistics for the evidence documents of TRIVIAQA-NOP.

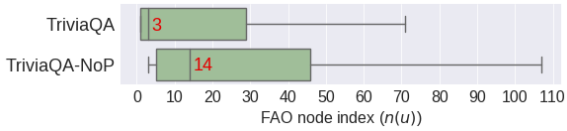


Figure 2: FAO node index distribution of TRIVIAQA and TRIVIAQA-NOP (median values in red).

	TRIVIAQA	TRIVIAQA-NOP
Questions	86.7%	83.8%
Question-document pairs	69.7%	66.9%

Table 3: Portion of answerable samples in a random subset of the training set, which contains 278 questions and 475 question-document pairs.

IR with RC, where some text is retrieved exactly once before applying an RC model. As RC shifts to reasoning over complex questions, navigating and reading multiple parts of the document will become necessary. We show in Section 5 this approach indeed improves QA performance.

3 Data

To test our framework, we capitalize on the recently-released TRIVIAQA dataset, which contains question-answer pairs, along with a small set of documents that (in almost all cases) contain the answer. TRIVIAQA is suitable for our purposes as it is a large scale dataset, where questions are relatively complex and documents are fairly long. The dataset includes only raw text, and thus for every evidence document, we built a tree representation by extracting the html metadata from the corresponding Wikipedia page, and constructing the document structure from it.

Because our goal is to investigate whether a model can learn to search through a document, it is important that a non-negligible fraction of the questions require navigation through the document. However, in Wikipedia each document starts with a preface that summarizes the document, and thus often contains the answer. Consequently, a model that ignores the question and document and always stops in the first paragraph is likely to obtain good performance. Figure 2 shows the distribution of the first answer occurrence (FAO) in a document in TRIVIAQA over the node indices $n(u)$ (x-axis), where all tree nodes, except sentences are considered (see Section 2). We find that in most question-document pairs the FAO is in the first few paragraphs, and that in 60% of the cases it is in the preface section.

To alleviate this heavy bias, we derive a new dataset, termed TRIVIAQA-NOP, where we remove the preface section from every document. After removing the preface, 2,144 out of 77,582 (2.8%) questions and 6,124 out of 138,537 (4.4%) question-document pairs are left without an answer and are removed. To further reduce the number of cases where an answer can not be inferred from a document, we drop question-document pairs where: (a) the answer appears only in titles; (b) the answer is a single-character; (c) the FAO node index is > 700 (in most cases the answer is an item in a list). Finally, the dataset includes 91.6% of the questions and 88.4% of the question-document pairs from the original dataset. We provide full statistics on the dataset in Tables 1 and 2.

To verify that questions remain answerable in TRIVIAQA-NOP after removing the preface, we perform manual analysis on a random sample of 278 questions and 475 documents from the training set (Table 3). We find that the portion of answerable questions and question-document pairs remains high and is reduced by less than 3% in comparison to TRIVIAQA . This demonstrates that indeed, for most questions and documents, the context necessary for answering the question also appears in the document body and not only in the preface.

Figure 2 shows the FAO node index distribution in TRIVIAQA-NOP. We observe, compared to TRIVIAQA, that the first occurrence of an answer is much more spread out across the document, and that the median increases from 3 to 14, which will require more navigation from the agent. However, even in TRIVIAQA-NOP answers tend to appear at the beginning of the document, because document content is usually organized by importance. This bias results in an exploration challenge for our training algorithm,

	Example
o	*"Phuket Province Name"*
ϕ_n^1: `height`	2
ϕ_n^2: `depth`	1
ϕ_n^3: `h_dist_start`	0
ϕ_n^4: `h_dist_end`	2
ϕ_n^5: `parent.h_dist_start`	0
ϕ_n^6: `parent.h_dist_end`	0
ϕ_n^7: `navigation step`	1

Table 4: An example observation o and list of navigation features ϕ_n for node 1 in Figure 1. The features `height` and `depth` correspond to distance from the farthest leaf and root, respectively (`height` is 2 since sentence nodes are omitted from the figure). `h_dist_start` and `h_dist_end` measure the horizontal distance from the first and last child of the node's parent. `navigation step` is a counter for the number of performed steps.

which we will address in Section 4.

4 Method

In this section, we describe a model for the agent and a training algorithm based on DQN (Mnih et al., 2015). Specifically, we introduce a tree-sampling strategy, which addresses the exploration challenge stemming from the bias towards answers early in the document.

4.1 Framework

We represent the MDP as a tuple $(\mathcal{S}, \mathcal{A}, R, T)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $R(s, a)$ is a reward function, and $T(s, a)$ is a deterministic transition function. Our model implements an action-value function $Q(s, a)$, which takes a state $s \in \mathcal{S}$ and returns a value for every action $a \in \mathcal{A}$. This function defines a policy $\pi(s) = \arg\max_{a \in \mathcal{A}} Q(s, a)$. We now describe the state space, actions and reward function.

States Given a tree node u, a state is a tuple $s^u = (q, o, z, \phi_n, \phi_z)$, where q is the question, o is an observation, z is an answer prediction, and ϕ_n, ϕ_z are navigation and answer prediction features. An observation $o = (o_1, \ldots, o_{|o|})$ is a sequence of tokens produced by recursively concatenating the first k tokens of text in the label $l(u)$ to the observation of u's parent. An answer prediction $z = (z_1, \ldots, z_{|z|})$ is the sequence of tokens that were extracted by an RC model, if an RC model was already run on $l(u)$ (and a null token otherwise). The answer prediction features $\phi_z = (z_e, z_l, z_n)$ provide information on the distribution over answer spans provided by the RC model, which reflects its confidence: z_e is the entropy of the distribution, z_l is the logit value for z, and z_n is the number of tokens in $l(u)$. Navigation features ϕ_n provide information on the relative location of u in the document. An example observation and full list of navigation features are given in Table 4.

Note that the state s does not depend on the history of visited tree nodes.[1] While incorporating history could be beneficial, a memory-less model enables us to explore tree-sampling strategies, which is important for training (Section 4.2).

Actions We define the following set of actions $\mathcal{A}$. Let u be a node with an ordered list of children (v_1, v_2, v_3), and w be a child of v_2. We define five movement actions (Figure 3), where DOWN moves from u to its first child v_1, RIGHT moves from v_2 to v_3, and LEFT moves from v_2 to v_1. Because moving upwards reaches a node we already visited, we define UPR, which moves from w to v_3, and UPL, which moves from w to v_1. If an illegal action is chosen (e.g., LEFT from v_1), then the agent stays in its current position. The action ANSWER returns an answer (and a distribution over spans) by running a RC model on $l(u)$, unless u is a sentence, in which case it is run on the paragraph containing u. After ANSWER, the agent can resume navigation. The action STOP also returns the answer given the current node u, but also terminates navigation.

Reward Our goal is to develop an agent that can navigate in the document, and thus we define the reward based on whether the agent stops in a node that contains the gold answer text (this is noisy,

[1] except for `navigation step`, which can be approximated by the shortest path from the root to any node.

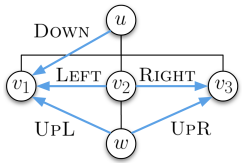


Figure 3: Movement actions in our environment.

because the answer might be there sporadically). While a simple reward would be an indicator for whether the agent stopped at a correct node, such a reward would not capture the proximity of the agent to the answer. Moreover, we would like to consider the overall document length, rewarding successful navigations in long documents. Therefore, we define the following reward:

$$r(s,a) = \begin{cases} 2 & a = \text{STOP}, |n(u) - n(u^*)| = 0 \\ 1 - \frac{|n(u)-n(u^*)|}{\max_u n(u)} & a = \text{STOP}, |n(u) - n(u^*)| > 0 \\ -0.06 & a = \text{ANSWER} \\ -0.02 & a \notin \{\text{ANSWER}, \text{STOP}\} \end{cases}$$

where u is the node where the agent is located and u^* is the closest tree node that contains the answer ($n(u)$ is the node index as defined above). Thus, when stopping, the reward is proportional to the distance to the closest answer location given the document length. An additional reward is given if navigation is successful, and a penalty is given for any other action, to encourage shorter trajectories. We further penalize the ANSWER action to discourage frequent usage of the RC model.

4.2 DOCQN: DQN with Tree Sampling

Training the navigation model is based on DQN (Mnih et al., 2015). In DQN, at every step an agent at state s_t selects an action a_t using ϵ-greedy policy, given the current action-value function $Q_\theta(s_t, a_t)$ parameterized by θ. The agent observes a reward r_t and a state $s_{t+1} = T(s_t, a_t)$ and adds a transition (s_t, a_t, r_t, s_{t+1}) to a replay memory buffer $\mathcal{D}$ that holds a large number of recent transitions. The parameters are then optimized so that the action-value function matches better the observed reward. This is done by sampling a batch of random transitions from $\mathcal{D}$ and minimizing the regression loss

$$(r_t + \gamma \max_{a'} Q_{\hat{\theta}}(s_{t+1}, a') - Q_\theta(s_t, a_t))^2, \tag{1}$$

where $\hat{\theta}$ are the parameters of a *target network*, which is a periodic copy of θ that is not optimized, and γ is a discount factor.

We also add some of the recent enhancements to DQN (Hessel et al., 2017), which have proved to be useful in our setup. Specifically we implement *Double Q-Learning* (van Hasselt et al., 2016), *Prioritized Experienced Replay* (Schaul et al., 2015), and *Dueling Networks* (Wang et al., 2016).

Reducing bias with DOCQN The DQN algorithm contains episodes, where in each episode the agent is placed at an initial state s_0, from which it starts taking actions. In our setup, this state corresponds to the root of the document tree. Because the data has a bias towards answers appearing at the beginning of the document, the agent learns that stopping early improves the reward and is stuck at a local minimum, where it ceases exploration. Examining Figures 2 and 8, we observe that DQN learns to stop very early in the document compared to the FAO node index distribution of TRIVIAQA-NOP, amplifying the bias in the data.

To address this issue, we suggest DOCQN, a variant of DQN aimed at increasing exploration when navigating in a document structure. DOCQN capitalized on two properties. First, DQN is an off-policy algorithm that trains from transitions (s_t, a_t, r_t, s_{t+1}). Second, our model is memory-less, and thus we can sample any node u and compute the corresponding state s^u. Therefore, we modify DQN, and instead of initializing every episode with s_0 and performing a sequence of actions, we sample states from distributions that explore the document better. By exploring transitions from across the document, the model learns from more distant parts of the document.

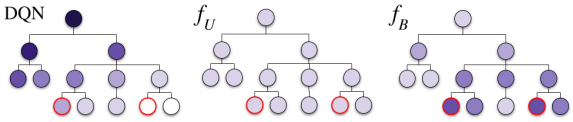


Figure 4: Illustration of the different state distributions for a specific documents tree. Nodes with darker color have higher sampling probability, and nodes marked in red are paragraph nodes containing the answer.

Algorithm 1: **DocQN**

1: Let f be a distribution over tree nodes
2: Initialize replay memory $\mathcal{D}$ and parameters $\theta, \hat{\theta}$
3: **for** episode $= 1, M$ **do**
4: **if** random() $< \varepsilon_s$ **then**
5: **for** $i = 1 \dots K$ **do**
6: sample node $u \sim f$ and generate s^u
7: sample action a from s^u (ϵ-greedily).
8: $r \leftarrow R(s^u, a), s' \leftarrow T(s^u, a)$
9: store (s^u, a, r, s') in $\mathcal{D}$
10: **else**
11: Initialize start state s_0
12: **for** $t = 0 \dots T-1$ **do**
13: sample action a from s_t (ϵ-greedily).
14: $r \leftarrow R(s_t, a), s_{t+1} \leftarrow T(s_t, a)$
15: store (s_t, a, r, s_{t+1}) in $\mathcal{D}$
16: UpdateParams$(\theta, \hat{\theta}, \mathcal{D})$ (Equation 1)

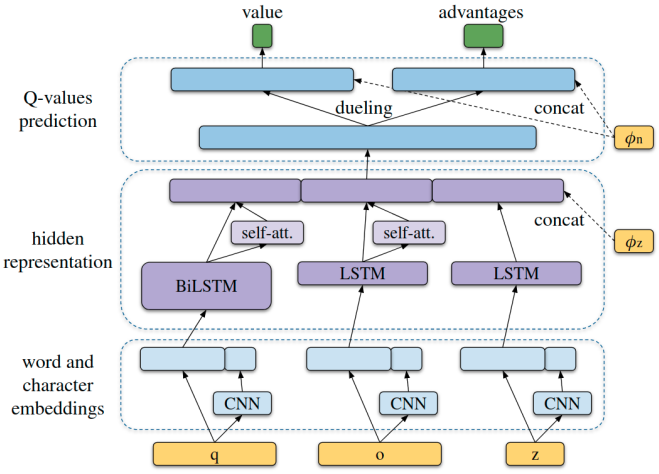


Figure 5: High-level overview of the network architecture.

Algorithm 1 provides the details of DocQN. The algorithm initializes the replay memory buffer, the model and target network parameters, and chooses a distribution f over tree nodes. At the beginning of every episode, with probability ϵ_s (line 4),[2] we sample K state transitions using f and store them in $\mathcal{D}$, and with probability $(1 - \epsilon_s)$ (line 10), we start at the initial state s_0 and sample a trajectory as in DQN. Parameters are updated as in DQN by sampling transitions from the replay buffer and minimizing Equation 1. If f samples nodes from various locations in the document, the algorithm will explore better and will not getting stuck at its beginning. We consider the following instantiations of f:

1. f_U: *Uniform sampling* over nodes, except we discourage sampling sentences by uniformly sampling with probability 0.2 a leaf (sentence), and probability 0.8 an inner node.
2. f_B: *Backward sampling*: We uniformly sample a paragraph node p that contains the gold answer, then uniformly a number of actions $B \in \{1, 2, 3\}$, and perform B random movement actions from p to output a node. This results in a node that is "close" to the answer a, and can be viewed as similar to bi-directional search or backward search (Lao et al., 2015).

Figure 4 shows the distribution over tree nodes for a specific document tree where the answer appears in two paragraph nodes (ignoring sentence nodes). We see that sequential sampling (as in DQN) puts most of the probability mass close to the root, uniform sampling is uniform (across paragraphs, as sentence nodes are omitted), and backward sampling is concentrated close to answer nodes. This illustrates how different distributions result in different exploration strategies.

Network Architecture We briefly describe the neural architecture (Figure 5) and provide full details in the supplementary material. As explained in Section 4.1, the input to the network is the state s, which comprises the question tokens q, observation tokens o, answer prediction tokens z and features ϕ_n, ϕ_z. The question, observation and answer prediction tokens are encoded with pre-trained word embeddings and trained character embeddings, where character embeddings are followed by a convolutional layer with max pooling, yielding a single vector per token. Each token is then represented by concatenating the word embedding with the character embedding.

Question tokens, observation tokens, and answer to are then fed into a BiLSTM and LSTM (Hochreiter and Schmidhuber, 1997) respectively and the LSTM outputs are compressed to a single vector through

[2] ϵ_s is annealed from 1 to 0.5 during training.

self attention (Cheng et al., 2016), resulting in one vector for q and one for o. Answer prediction tokens are fed into an LSTM, where the last hidden state is concatenated to the features ϕ_z, thus creating a third vector for the answer prediction.

We concatenate these three vectors, and pass them through a one layer feed-forward network that then branches to two networks according to the Dueling DQN architecture (Wang et al., 2016). In each branch we also concatenate the navigation features ϕ_n. One branch predicts the value of the state $V_\theta(s) \in \mathbb{R}$, and the other branch predicts the advantage of every action $A_\theta(s,a) \in \mathbb{R}$ for every possible action a. The output of the network is $Q_\theta(s,a) = V_\theta(s) + (A_\theta(s,a) - \frac{1}{\mathcal{A}} \sum_{a'} A_\theta(s,a'))$ as in Wang et al. (2016).

5 Experimental Evaluation

Our experimental evaluation aims to answer the following questions: (a) Can document structure be used to learn to navigate to an answer in a document? (b) How does DOCQN compare to DQN? (c) How does DOCQN compare to IR methods that observe the entire document?

5.1 Experimental Setup

We evaluate on TRIVIAQA-NOP. Because our focus is on the navigation ability of the agent, we train a single RC model and fix it in all experiments. Specifically, we download RASOR (Lee et al., 2017; Salant and Berant, 2018),[3] and exactly follow the procedure described by the authors of TRIVIAQA for training a RC model, i.e., we train RASOR on the first 400 tokens of each document in TRIVIAQA-NOP. As a sanity check for our RC model, we also train and evaluate RASOR on the original TRIVIAQA dataset. Indeed, RASOR obtains 48.6% EM and 53.4% F1, which is substantially higher than the baseline reported by Joshi et al. (2017).

Evaluation We use two metrics: First, we measure *navigation accuracy*, i.e., for a question-document pair, whether a method returns text containing a gold answer (if the agent stops at a sentence node we evaluate the encompassing paragraph). Because questions in TRIVIAQA-NOP often have more than one evidence document, we also measure *aggregated navigation accuracy*, where we give credit if the agent navigated correctly in any of the documents. This gives performance assuming an oracle that always chooses the best document for the question. Because the test set in TRIVIAQA is hidden, we evaluated navigation accuracy on the development set only.

In addition, we measure end-to-end QA performance with the official Exact Match (EM) and F1 metrics. The EM metric measures the percentage of predictions that match exactly any of the answer aliases, and the F1 metric measures the average overlap between the prediction and answer. To aggregate evidence from multiple documents we follow Joshi et al. (2017) and define the score of an answer to be the sum of probabilities for that answer across all documents.

Models We compare the following models:

- **DOCQN**: This is our main model, where we use the sampling distribution $f_{U+B} = \frac{1}{2} f_U + \frac{1}{2} f_B$.
- **DQN**: DQN algorithm without state sampling.
- {**DOCQN**|**DQN**}-**COUPLED**: A less expressive version of the model, where the actions ANSWER and STOP are coupled, i.e., the agent can use the RC model exactly once and then stops. This is similar to a setup where retrieval is performed once and not interleaved with navigation.
- **RANDOMWALK** An agent that selects an action at each step uniformly at random.
- **RANDOMPARA** An agent that randomly selects a non-sentence tree node.
- **DOC-TF-IDF**: An IR baseline, where a paragraph is selected based on its tf-idf score. We implement the tf-idf scheme of DOCQA (Clark and Gardner, 2017), which is a high-performing system on TRIVIAQA. Here, idf is computed for each paragraph in the context of the current document, and the paragraph with highest cosine similarity to the question is chosen. Note that DOC-TF-IDF processes the entire document (up to tens of thousands of tokens), while DOCQN processes only a small fraction.
- **TF-IDF**: Vanilla tf-idf, where the idf score is computed from all documents in TRIVIAQA-NOP.

[3] https://github.com/shimisalant/RaSoR

- **Ensemble**: We ensemble DocQN f_{U+B} with Doc-Tf-Idf in two ways: (1) for finding the final answer, we aggregate scores as described above, that is, we sum the probabilities from both models over all documents and choose the answer with highest score. (2) For navigation, we simply tune on the development set a threshold l, where we take the prediction of DocQN f_{U+B} if it stopped at a node with index $\leq l$ and use Doc-Tf-Idf otherwise.
- **ReadTop**: Following Joshi et al. (2017), a model that runs the RC model on the first 800 tokens of the document. Note that running the RC model on the text retrieved by DocQN involves consuming far fewer tokens, namely, RaSoR consumes only 160 tokens on average.

We report the value of all hyper-parameters in the supplementary material (all fixed without tuning).

5.2 Results

q: According to the rhyme, what is Wednesday's child? ***a:*** ['full of woe']

Step	Node	Observation Tokens	Answer	Action
0	0	['Wednesday']		DOWN
1	1	['Wednesday', 'Etymology']		DOWN
2	2	['Wednesday', 'Etymology', 'See', 'Names', 'of', 'the', 'days', 'of', 'the', 'week', 'for', 'more', 'on', 'naming', 'conventions', '.']		UPR
3	25	['Wednesday', 'Religious', 'observances']		DOWN
4	26	['Wednesday', 'Religious', 'observances', 'The', 'Creation', 'narrative', 'in', 'the', 'Hebrew', 'Bible', 'places', 'the', 'creation', 'of', 'the', 'Sun', 'and', 'Moon', 'on', '``', 'the', 'fourth', 'day']		ANS
5	26	['Wednesday', 'Religious', 'observances', 'The', 'Creation', 'narrative', 'in', 'the', 'Hebrew', 'Bible', 'places', 'the', 'creation', 'of', 'the', 'Sun', 'and', 'Moon', 'on', '``', 'the', 'fourth', 'day']	Sun and Moon	UPR
6	37	['Wednesday', 'Cultural', 'usage']		DOWN
7	38	['Wednesday', 'Cultural', 'usage', 'In', 'Hindu', 'mythology', ',', '', 'is', 'the', 'god', 'of', 'Mercury', '(', 'planet', ')', ',', 'mid-week', 'Wednesday', ',', 'and', 'of', 'Merchants']		ANS
8	38	['Wednesday', 'Cultural', 'usage', 'In', 'Hindu', 'mythology', ',', '', 'is', 'the', 'god', 'of', 'Mercury', '(', 'planet', ')', ',', 'mid-week', 'Wednesday', ',', 'and', 'of', 'Merchants']	Budha	RIGHT
9	39	['Wednesday', 'Cultural', 'usage', 'In', 'the', 'folk', 'rhyme', '``', 'Wednesday', '\'s', 'child', 'is', 'full', 'of', 'woe', '\'\'', ',', 'reciting', 'the', 'days', 'of', 'the', 'week']		ANS
10	39	['Wednesday', 'Cultural', 'usage', 'In', 'the', 'folk', 'rhyme', '``', 'Wednesday', '\'s', 'child', 'is', 'full', 'of', 'woe', '\'\'', ',', 'reciting', 'the', 'days', 'of', 'the', 'week']	full of woe	STOP

Figure 6: Navigation example of DocQN.

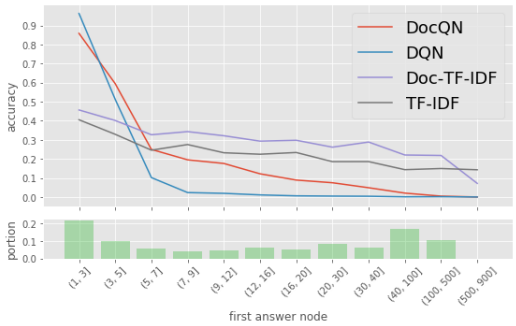


Figure 7: Model performance (top) and portion of samples in the development set (bottom) as functions of the node index of the FAO.

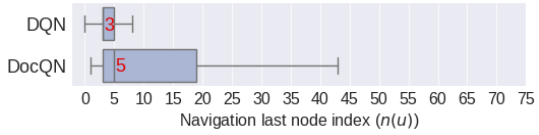


Figure 8: Distribution of node index at navigation stopping point (median value in red) for DQN and DocQN.

Tables 5 and 6 show the results of our experiments. Focusing on navigation accuracy, we see that randomly walking or choosing a paragraph yields low performance. Vanilla Tf-Idf performs considerably better than the random baselines, but is outperformed by all other models. Comparing DQN to DocQN, we see that DocQN outperforms DQN. Allowing DQN and DocQN to run the RC model during navigation improves their performance, but the accuracy gain is substantially higher for DocQN (2.3% accuracy and 3.1% aggregated accuracy) than for DQN (0.4% accuracy and 0.7% aggregated accuracy). Doc-Tf-Idf, which has access to the entire document outperforms DocQN, which consumes on average 6% of the entire document. Nonetheless, the two models obtain the same aggregated accuracy. This good performance of Doc-Tf-Idf shows that in TriviaQA-NoP answer paragraphs share a lot of lexical material with the question. Importantly, an ensemble of Doc-Tf-Idf with DocQN substantially improves the overall performance, reaching accuracy of 35% and aggregated accuracy of 48%. This is

	Dev.	Dev. Agg.
RANDOMWALK	1.8	3.1
RANDOMPARA	13.4	20.9
DQN-COUPLED	26.8	38.8
DQN	27.2	39.5
DOCQN-COUPLED	28.1	40.5
DOCQN	30.4	43.6
TF-IDF	25.3	35.4
DOC-TF-IDF	32.4	43.5
Ensemble (threshold, $l = 5$)	**35.0**	**48.0**

Table 5: Navigation accuracy for all models on the development set of TRIVIAQA-NOP.

	Development		Test	
	EM	F1	EM	F1
DQN	21.7	26.5	19.1	24.1
DOCQN	23.6	27.9	21.0	25.5
DOC-TF-IDF	21.4	27.1	18.2	23.5
Ensemble (threshold, $l = 5$)	26.8	32.0	24.2	29.4
Ensemble (answer)	**28.4**	**33.4**	**25.4**	**30.5**
READTOP	**32.5**	**36.7**	**28.1**	**32.4**

Table 6: End-to-end QA performance of all models on the development and test sets of TRIVIAQA-NOP. For the ensemble model, we choose the best threshold of $l = 5$.

	DQN	DOCQN
Path length	avg. 7.7, range [1,36]	avg. 15.2, range [3,100]
Answer predictions	avg. 2.8	avg. 4.1
Tokens consumed	3.4%	6.2%
Stopping node	0.5% title, 2.2% headline, 89.0% paragraph, 8.3% sentence	0% title, 0.4% headline, 75.1% paragraph, 24.5% sentence

Table 7: Comparing navigation properties of DQN and DOCQN on the development set. The average and range of navigation path length in steps, relative amount of consumed tokens, and distribution of stopping node types.

in sharp contrast to an ensemble with DQN, where for any value of l, DOC-TF-IDF performs better on its own.

Examining end-to-end performance, DOCQN again outperforms DQN, but DOCQN is now better than DOC-TF-IDF. This suggests that when DOC-TF-IDF selects a paragraph with the answer, it is often difficult to extract with the RC model. Again, the ensembles leads to a dramatic increase in performance, showing that DOC-TF-IDF and DOCQN are complementary. However, READTOP, which consumes the first 800 tokens of each document compared to ~ 160 for DOCQN, substantially outperforms all navigation models. This shows that the bias for answers at the beginning of documents is still strong in TRIVIAQA-NOP.

To further elucidate the differences between navigation models, Figure 7 shows navigation accuracy of different models and the proportion of samples for different node indices of the FAO. We see that DQN outperforms DOCQN when the answer is at the top of the document, but DOCQN dominates DQN when the answer is further down, showing that DOCQN learns to find answers deeper in the document. DOC-TF-IDF has a more balanced navigation accuracy across the document, which explains why an ensemble of DOC-TF-IDF with DOCQN works, as they are complementary to one another.

Analysis Figure 6 shows a navigation example, which includes the navigation step, node index, observation o, and the action a taken. For the observation we also highlight the attention distribution from the self-attention component. In this figure the question is about culture, and we see the agent going into multiple sections, reading them, running the RC model and continuing forward, until finally stopping at a paragraph that contains the answer. We provide many more examples in the supplementary material.

Table 7 highlights some differences between DQN and DOCQN. DOCQN has longer trajectories, and stops at sentence nodes more frequently, suggesting it reads the document at a finer granularity. Additionally, it leverages the RC model to collect more information and confidence during navigation, by choosing the action ANSWER more frequently. Both models consume less than 7% of the entire document. Figure 8 illustrates the navigation stopping point, and shows that DOCQN navigates deeper into the document.

6 Related Work

Handling the challenges of reasoning over multiple long documents is gaining fast momentum recently (Shen et al., 2017). As mentioned, some approaches use IR for reducing the amount of processed text (Chen et al., 2017; Clark and Gardner, 2017), while others use cheap or parallelizable models to handle long documents (Hewlett et al., 2017; Swayamdipta et al., 2018; Wang et al., 2018a). Searching for

answers while using a trained RC model as a black-box was also implemented recently in Wang et al. (2018b), for open-domain questions and multiple short evidence texts from the Web. Another thrust has focused on skimming text in a sequential manner (Yu et al., 2017), or designing recurrent architectures that can consume text quickly (Bradbury et al., 2017; Seo et al., 2018; Campos et al., 2018; Yu et al., 2018). However, to the best of our knowledge no work has previously applied these methods to long documents such as Wikipedia pages.

In this work we use TRIVIAQA-NOP for evaluation of our navigation based approach and comparison to an IR baseline. While there are various aspects to consider in such evaluation setup, our choice of data was derived mainly by the requirements for long and structured context. Recently, several new datasets such as WIKIHOP and NARRATIVEQA were published. These datasets try to focus on the tendency of RC models to match local context patterns, and are designed for multi-step reasoning. (Welbl et al., 2017; Wadhwa et al., 2018; Kočisky et al., 2017).

Our work is also related to several papers which model an agent that navigates in an environment to find objects in an image (Ba et al., 2015), relations in a knowledge-base (Das et al., 2018), or documents on the web (Nogueira and Cho, 2016).

7 Conclusions

We investigate whether document structure can be leveraged to train an agent that finds answers to questions in long documents while reading only a small fraction of it. We show that an agent that reads 6% of the document can improve QA performance compared to an IR method that utilizes the entire document, and that ensembling the two substantially improves performance. We also present DOCQN, an algorithm that promotes better exploration of the document, and show it outperforms DQN qualitatively and quantitatively.

Our approach represents a conceptual departure from previous methods for reading long documents, as it interleaves searching for an answer in the document with extracting the answer from a particular paragraph, which we show improves both navigation and QA performance. We expect that as RC models tackle longer documents that require reasoning and reading text that is spread in multiple parts of the document, models that can efficiently navigate and collect evidence will become more and more crucial. Our agent provides a first step in this important research direction.

Acknowledgments

We thank Eunsol Choi from Washington University for helpful discussions and comments on the paper. This research was partially supported by The Israel Science Foundation grant 942/16. This work was completed in partial fulfillment for the Ph.D degree of the first author.

References

J. Ba, V. Mnih, and K. Kavukcuoglu. 2015. Multiple object recognition with visual attention. In *International Conference on Learning Representations (ICLR)*.

J. Bradbury, S. Merity, C. Xiong, and R. Socher. 2017. Quasi-recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

V. Campos, B. Jou, X. Giro i Nieto, J. Torres, and S. Chang. 2018. Skip RNN: Learning to skip state updates in recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

D. Chen, J. Bolton, and C. D. Manning. 2016. A thorough examination of the CNN / Daily Mail reading comprehension task. In *Association for Computational Linguistics (ACL)*.

D. Chen, A. Fisch, J. Weston, and A. Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)*.

J. Cheng, L. Dong, and M. Lapata. 2016. Long short-term memory-networks for machine reading. In *Empirical Methods in Natural Language Processing (EMNLP)*.

E. Choi, D. Hewlett, A. Lacoste, I. Polosukhin, J. Uszkoreit, and J. Berant. 2017. Coarse-to-fine question answering for long documents. In *Association for Computational Linguistics (ACL)*.

C. Clark and M. Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*.

R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations (ICLR)*.

K. M. Hermann, T. Koisk, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

M. Hessel, J. Modayil, H. V. Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. 2017. Rainbow: Combining improvements in deep reinforcement learning. *arXiv preprint arXiv:1710.02298*.

D. Hewlett, A. Lacoste, L. Jones, I. Polosukhin, A. Fandrianto, J. Han, M. Kelcey, and D. Berthelot. 2016. Wikireading: A novel large-scale language understanding task over Wikipedia. In *Association for Computational Linguistics (ACL)*.

D. Hewlett, L. Jones, A. Lacoste, et al. 2017. Accurate supervised and semi-supervised machine reading for long documents. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2011–2020.

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

M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Association for Computational Linguistics (ACL)*.

R. Kadlec, M. Schmid, O. Bajgar, and J. Kleindienst. 2016. Text understanding with the attention sum reader network. In *Association for Computational Linguistics (ACL)*.

T. Kočisky, J. Schwarz, P. Blunsom, C. Dyer, K. M. Hermann, G. Melis, and E. Grefenstette. 2017. The NarrativeQA reading comprehension challenge. *arXiv preprint arXiv:1712.07040*.

N. Lao, E. Minkov, and W. Cohen. 2015. Learning relational features with backward random walks. In *Association for Computational Linguistics (ACL)*.

K. Lee, S. Salant, T. Kwiatkowski, A. Parikh, D. Das, and J. Berant. 2017. Learning recurrent span representations for extractive question answering. *arXiv*.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

R. Nogueira and K. Cho. 2016. End-to-end goal-driven web navigation. In *Advances in Neural Information Processing Systems (NIPS)*.

T. Onishi, H. Wang, M. Bansal, K. Gimpel, and D. McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. In *Empirical Methods in Natural Language Processing (EMNLP)*.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

S. Salant and J. Berant. 2018. Contextualized word representations for reading comprehension. In *North American Association for Computational Linguistics (NAACL)*.

T. Schaul, J. Quan, I. Antonoglou, and D. Silver. 2015. Prioritized experience replay. In *International Conference on Learning Representations (ICLR)*.

M. Seo, S. Min, A. Farhadi, and H. Hajishirzi. 2018. Neural speed reading via skim-RNN. In *International Conference on Learning Representations (ICLR)*.

Y. Shen, P. Huang, J. Gao, and W. Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *International Conference on Knowledge Discovery and Data Mining (KDD)*.

S. Swayamdipta, A. P. Parikh, and T. Kwiatkowski. 2018. Multi-mention learning for reading comprehension with neural cascades. In *International Conference on Learning Representations (ICLR)*.

A. Trischler, T. Wang, X. Yuan, J. Harris, A. Sordoni, P. Bachman, and K. Suleman. 2017. NewsQA: A machine comprehension dataset. In *Workshop on Representation Learning for NLP*.

H. van Hasselt, A. Guez, and D. Silver. 2016. Deep reinforcement learning with double Q-learning. In *Association for the Advancement of Artificial Intelligence (AAAI)*, volume 16, pages 2094–2100.

S. Wadhwa, V. Embar, M. Grabmair, and E. Nyberg. 2018. Towards inference-oriented reading comprehension: Parallelqa. In *Workshop on New Forms of Generalization in Deep Learning and Natural Language Processing at NAACL*.

Z. Wang, T. Schaul, M. Hessel, H. V. Hasselt, M. Lanctot, and N. D. Freitas. 2016. Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*.

S. Wang, M. Yu, X. Guo, Z. Wang, T. Klinger, W. Zhang, S. Chang, G. Tesauro, B. Zhou, and J. Jiang. 2018a. R3: Reinforced ranker-reader for open-domain question answering. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

S. Wang, M. Yu, J. Jiang, W. Zhang, X. Guo, S. Chang, Z. Wang, T. Klinger, G. Tesauro, and M. Campbell. 2018b. Evidence aggregation for answer re-ranking in open-domain question answering. In *International Conference on Learning Representations*.

Y. Watanabe, B. Dhingra, and R. Salakhutdinov. 2017. Question answering from unstructured text by retrieval and comprehension. *arXiv preprint arXiv:1703.08885*.

J. Welbl, P. Stenetorp, and S. Riedel. 2017. Constructing datasets for multi-hop reading comprehension across documents. *arXiv preprint arXiv:1710.06481*.

C. Xiong, V. Zhong, and R. Socher. 2017. Dynamic coattention networks for question answering. In *International Conference on Learning Representations (ICLR)*.

A. W. Yu, H. Lee, and Q. V. Le. 2017. Learning to skim text. In *Association for Computational Linguistics (ACL)*.

K. Yu, Y. Liu, A. G. Schwing, and J. Peng. 2018. Fast and accurate text classification: Skimming, rereading and early stopping. In *International Conference on Learning Representations (ICLR)*.

A Supplementary Material

A.1 Network Architecture Details

Here, we elaborate on the network architecture, which was briefly described in the paper. Given an input state $s = (q, o, z, \phi_n, \phi_z)$, we denote by $q = (q_1, ..., q_n)$, $o = (o_1, ..., o_m)$ and $z = (z_1, ..., z_r)$ the question tokens, observation tokens and answer prediction tokens, respectively.

Word-level embedding For every $i = 1, ..., n$, every $j = 1, ..., m$ and every $k = 1, ..., r$, we create an embedding of the input tokens q_i, o_j, z_k in two steps. First, we embed every token with a pre-trained GloVe word embedding matrix W_w:

$$e^w_{q_i} = W_w \cdot q_i \quad ; \quad e^w_{o_j} = W_w \cdot o_j \quad ; \quad e^w_{z_k} = W_w \cdot z_k$$

Next, we apply character-level embeddings with a learned embedding matrix W_c. The embedded characters are then summarized with a max-pooling convolutional neural network:

$$\begin{aligned} e^c_{q_i} &= \mathrm{CNN}(\{W_c \cdot q_{ix}\}^{|q_i|}_{x=1}) \\ e^c_{o_j} &= \mathrm{CNN}(\{W_c \cdot o_{jx}\}^{|o_j|}_{x=1}) \\ e^c_{z_k} &= \mathrm{CNN}(\{W_c \cdot o_{kx}\}^{|z_k|}_{x=1}) \end{aligned}$$

Concatenation of the two components yields the final word-level embeddings:

$$e_{q_i} = [e^w_{q_i} \,;\, e^c_{q_i}] \quad ; \quad e_{o_j} = [e^w_{o_j} \,;\, e^c_{o_j}] \quad ; \quad e_{z_k} = [e^w_{z_k} \,;\, e^c_{z_k}]$$

Question sequence encoding The question is encoded with a BiLSTM, where for every timestamp i, the forward and backward outputs are concatenated:

$$\begin{aligned} \{u_{q_1}, ..., u_{q_n}\} &= \mathrm{BiLSTM}(e_{q_1}, ..., e_{q_n}) \\ u_{q_i} &:= [\overrightarrow{\mathrm{LSTM}}(\overrightarrow{h}_{q_{i-1}}, e_{q_i}) \,;\, \overleftarrow{\mathrm{LSTM}}(\overleftarrow{h}_{q_{i+1}}, e_{q_i})] \end{aligned}$$

The outputs are then summarized with a self-attention:

$$h_q = \sum_{i=1}^{n} \alpha_i u_{q_i}$$

where the coefficients $\alpha_1, ..., \alpha_n$ are obtained by feeding the outputs to a two-layer feed-forward network, and normalizing the output logits with softmax:

$$a_i = \mathrm{FFNN}(u_{q_i}) \quad ; \quad \alpha_i = \frac{e^{a_i}}{\sum_{k=1}^{n} e^{a_k}}$$

Observation sequence encoding The observation sequence encoding h_o is obtained in an analogous manner to h_q, except that we use a LSTM rather than a BiLSTM.

Answer prediction encoding The answer prediction encoding h_z is obtained by running the answer prediction tokens through a LSTM, and concatenating the last hidden state with the input feature vector ϕ_z:

$$\begin{aligned} \{u_{z_1}, ..., u_{z_r}\} &= \mathrm{LSTM}(e_{z_1}, ..., e_{z_r}) \\ u_{z_k} &:= \overrightarrow{\mathrm{LSTM}}(\overrightarrow{h}_{z_{k-1}}, e_{z_k}) \\ h_z &= [u_{z_r} \,;\, \phi_z] \end{aligned}$$

	Parameter	Value
Network	Maximum node observation length	20 tokens
	Maximum observation length	120 tokens
	Word embedding dimension	300
	Character embedding dimension	20
	Convolution filter size	5
	BiLSTM and LSTM hidden dimension	300
	First feed-forward layer dimension	512
	Second feed-forward layer dimension	256
	Dropout rate	0.2
Training	RMSprop learning rate	0.0001
	Batch size	64
	Target network period	10K steps
	Initial memory size	50K transitions
	Maximal memory size	300K transitions
	Action sampling ϵ	$1.0 \rightarrow 0.1$
	Discount factor γ	0.996
	Prioritization usage α	0.6
	Prioritization importance sampling β	$0.4 \rightarrow 1.0$
Sampling	State sampling ϵ_s	$1.0 \rightarrow 0.5$
	Annealing steps for ϵ_s	1.2M steps
	Sampling repetitions K	5
	Maximum navigation length T (train)	30 steps
	Maximum navigation length T (test)	100 steps
	Interpolation coefficient for f_{U+B}	0.5

Table 8: DOCQN hyper-parameters

State representation The final state representation is formed by concatenating the encoded question h_q with the encoded observation h_o. Concretely:

$$h_s = [h_q \ ; \ h_o \ ; \ h_z]$$

The input node features ϕ_o are concatenated to an upper layer, as we describe next , to increase their weights in the final predictions. We have found that incorporation of these features in this way accelerates the navigation learning process.

Q-values prediction We implement a Dueling DQN architecture, where the final Q-values are composed of a state value $V(s)$ prediction and advantage predictions $A(s, a)$ for every possible action a. Denoting by FFNN a single-layer feed-forward neural network, predictions are derived as follows:

$$v_0 = \text{FFNN}(h_s)$$
$$v_1^V = \text{FFNN}(v_0) \ ; \ v_1^A = \text{FFNN}(v_0)$$
$$v_2^V = \text{FFNN}([v_1^V \ ; \ \phi_n]) \ ; \ v_2^A = \text{FFNN}([v_1^A \ ; \ \phi_n])$$

Where $v_2^V \in \mathbb{R}$, $v_2^A \in \mathbb{R}^{|\mathcal{A}|}$, and ϕ are the navigation features. The final Q-values prediction is obtained by averaging:

$$v^Q = v_2^V + \left(v_2^A - \frac{\sum_{i=1}^{|\mathcal{A}|} v_{2i}^A}{|\mathcal{A}|}\right)$$

A.2 Hyper Parameters

Table 8 summarizes the hyper-parameters used for building and training the DOCQN models.

A.3 Navigation Examples

Figures 9,10,11,12,13 show sample navigations, performed by the DOCQN model. In each example, q, a, n denote the question, answer aliases and answer node numbers. The observation tokens are highlighted according to the self-attention weights, given by the model. By choosing the action ANS, the agent executes the RC model to obtain an answer prediction, which is part of the observation in the next step.

q: How many legs does a ladybird have?
a: ['six', '6']
n: [2, 12, 14, 22, 46, 51, 60, 72, 74, 102, 150, 159, 163, 200]

Step	Node	Observation Tokens	Answer	Action
0	0	['Insect']		DOWN
1	1	['Insect', 'Etymology']		DOWN
2	2	['Insect', 'Etymology', 'The', 'word', '``', 'insect', '\\', 'comes', 'from', 'the', 'Latin', 'word', '\', ',', 'meaning', '``', 'with', 'a', 'notched', 'or', 'divided', 'body']		UPR
3	5	['Insect', 'Phylogeny', 'and', 'evolution']		DOWN
4	6	['Insect', 'Phylogeny', 'and', 'evolution', 'The', 'evolutionary', 'relationship', 'of', 'insects', 'to', 'other', 'animal', 'groups', 'remains', 'unclear', '.']		UPR
5	45	['Insect', 'Diversity']		DOWN
6	46	['Insect', 'Diversity', 'Though', 'the', 'true', 'dimensions', 'of', 'species', 'diversity', 'remain', 'uncertain', ',', 'estimates', 'range', 'from', '2.6', '--', '7.8', 'million', 'species', 'with', 'a']		UPR
7	49	['Insect', 'Morphology', 'and', 'physiology']		DOWN
8	50	['Insect', 'Morphology', 'and', 'physiology', 'External']		DOWN
9	51	['Insect', 'Morphology', 'and', 'physiology', 'External', 'Insects', 'have', 'segmented', 'bodies', 'supported', 'by', 'exoskeletons', ',', 'the', 'hard', 'outer', 'covering', 'made', 'mostly', 'of', 'chitin', '.']		ANS
10	51	['Insect', 'Morphology', 'and', 'physiology', 'External', 'Insects', 'have', 'segmented', 'bodies', 'supported', 'by', 'exoskeletons', ',', 'the', 'hard', 'outer', 'covering', 'made', 'mostly', 'of', 'chitin', '.']	six	STOP

Figure 9: Navigation example 1. Although the document is more general than the question subject and the answer appears multiple times across the document, the agent finds the correct context for answering the question.

q: Justine Thornton is the fiancee of which politician?
a: ['ed milliband', 'red ed', 'tom baldwin journalist', 'ed miliband', 'edward miliband']
n: [21]

Step	Node	Observation Tokens	Answer	Action
0	0	['Justine', 'Thornton']		DOWN
1	1	['Justine', 'Thornton', 'Early', 'life', 'and', 'education']		DOWN
2	2	['Justine', 'Thornton', 'Early', 'life', 'and', 'education', 'Thornton', 'was', 'born', 'in', 'Manchester', 'to', '', 'professionals', ',', 'Margaret', 'and', 'Dr', 'Stewart', 'Thornton', '.']		ANS
3	2	['Justine', 'Thornton', 'Early', 'life', 'and', 'education', 'Thornton', 'was', 'born', 'in', 'Manchester', 'to', '', 'professionals', ',', 'Margaret', 'and', 'Dr', 'Stewart', 'Thornton', '.']	Dr Stewart	UPR
4	10	['Justine', 'Thornton', 'Career']		DOWN
5	11	['Justine', 'Thornton', 'Career', 'Thornton', 'practises', 'in', 'Environmental', 'Law', ',', 'now', 'at', '39', 'Essex', 'Street', '.']		ANS
6	11	['Justine', 'Thornton', 'Career', 'Thornton', 'practises', 'in', 'Environmental', 'Law', ',', 'now', 'at', '39', 'Essex', 'Street', '.']	Environmental Law	UPR
7	20	['Justine', 'Thornton', 'Personal', 'life']		DOWN
8	21	['Justine', 'Thornton', 'Personal', 'life', 'Thornton', '\'s', 'husband', 'is', 'former', 'Labour', 'Party', 'Leader', 'Ed', 'Miliband', 'MP', '.']		STOP

Figure 10: Navigation example 2. The agent explores several sections before reaching the most relevant one.

q: What was the French sounding winner of the 2011 Epsom Derby?
a: ['pour moi']
n: [15, 40]

Step	Node	Observation Tokens	Answer	Action
0	0	['2011', 'Epsom', 'Derby']		DOWN
1	1	['2011', 'Epsom', 'Derby', 'Race', 'details']		DOWN
2	2	['2011', 'Epsom', 'Derby', 'Race', 'details', '*', 'Sponsor', ':', '']		UPR
3	9	['2011', 'Epsom', 'Derby', 'Full', 'result']		DOWN
4	10	['2011', 'Epsom', 'Derby', 'Full', 'result', '*', 'The', 'distances', 'between', 'the', 'horses', 'are', 'shown', 'in', 'lengths', 'or', 'shorter', ';', '']		UPR
5	14	['2011', 'Epsom', 'Derby', 'Winner', 'details']		DOWN
6	15	['2011', 'Epsom', 'Derby', 'Winner', 'details', 'Further', 'details', 'of', 'the', 'winner', ',', 'Pour', 'Moi', '.']		STOP

Figure 11: Navigation example 3. The agent quickly navigates to the answer, without running the RC model.

q: What nationality was the pointillist artist MONDRIAN?
a: ['autochtoon', 'people of holland', 'etnic dutch', 'dutch person', 'dutch', 'dutch people', 'dutch ethnic group', 'ethnic dutch', 'dutch ethnic group and nation', 'netherlandians', 'autochtones', 'dutchmen', 'autochtonen', 'netherlanders', 'netherlander']
n: [3, 9, 53]

Step	Node	Observation Tokens	Answer	Action
0	0	['Piet', 'Mondrian']		DOWN
1	1	['Piet', 'Mondrian', 'The', 'Netherlands', '(', '1872', '--', '1911', ')']		DOWN
2	2	['Piet', 'Mondrian', 'The', 'Netherlands', '(', '1872', '--', '1911', ')', 'Mondrian', 'was', 'born', 'in', '', 'in', 'the', 'Netherlands', ',', 'the', 'second', 'of', 'his', 'parents', '\', 'children', '.']		ANS
3	2	['Piet', 'Mondrian', 'The', 'Netherlands', '(', '1872', '--', '1911', ')', 'Mondrian', 'was', 'born', 'in', '', 'in', 'the', 'Netherlands', ',', 'the', 'second', 'of', 'his', 'parents', '\', 'children', '.']	Netherlands	RIGHT
4	3	['Piet', 'Mondrian', 'The', 'Netherlands', '(', '1872', '--', '1911', ')', 'After', 'a', 'strictly', 'Protestant', 'upbringing', ',', 'in', '1892', ',', 'Mondrian', 'entered', 'the', 'Academy', 'for', 'Fine', 'Art', 'in', 'Amsterdam', '.']		ANS
5	3	['Piet', 'Mondrian', 'The', 'Netherlands', '(', '1872', '--', '1911', ')', 'After', 'a', 'strictly', 'Protestant', 'upbringing', ',', 'in', '1892', ',', 'Mondrian', 'entered', 'the', 'Academy', 'for', 'Fine', 'Art', 'in', 'Amsterdam', '.']	Dutch	STOP

Figure 12: Navigation example 4. The agent leverages the RC model and decides to stop after observing enough information to answer.

q: Who created Rumpole of the Bailey?
a: ['john mortimer qc', 'john mortimer', 'john clifford mortimer', 'sir john mortimer']
n: [31, 46, 56, 144, 154, 168, 178, 188, 198, 208, 245, 258, 274, 282, 290, 306, 341, 345, 466, 475]

Step	Node	Observation Tokens	Answer	Action
0	0	['Rumpole', 'of', 'the', 'Bailey']		DOWN
1	1	['Rumpole', 'of', 'the', 'Bailey', 'Horace', 'Rumpole']		RIGHT
2	44	['Rumpole', 'of', 'the', 'Bailey', 'Production']		RIGHT
3	127	['Rumpole', 'of', 'the', 'Bailey', 'Television', 'episodes']		DOWN
4	128	['Rumpole', 'of', 'the', 'Bailey', 'Television', 'episodes', 'There', 'were', 'a', 'grand', 'total', 'of', '44', 'episodes', '.']		RIGHT
5	129	['Rumpole', 'of', 'the', 'Bailey', 'Television', 'episodes', 'All', 'listed', 'dates', 'indicate', 'first', 'UK', 'transmission', 'date']		RIGHT
6	130	['Rumpole', 'of', 'the', 'Bailey', 'Television', 'episodes', '', 'Film', 'for', 'BBC', 'TV', '\'s', 'Play', 'for', 'Today', 'Series', '(', '1975', ')']		RIGHT
7	131	['Rumpole', 'of', 'the', 'Bailey', 'Television', 'episodes', '*', '``', 'Rumpole', 'of', 'the', 'Bailey', '\'\'', '(', '16', 'December', '1975', ')', '[', 'Set', 'in', '1974', ']']		RIGHT
8	132	['Rumpole', 'of', 'the', 'Bailey', 'Television', 'episodes', '**', '(', '', '``', 'Rumpole', 'and', 'the', 'Confession', 'of', 'Guilt', '\'\'', 'for', 'radio', 'adaptation', 'in', '1980', 'and', 'for', 'DVD', 'release']		RIGHT
9	133	['Rumpole', 'of', 'the', 'Bailey', 'Television', 'episodes', '*', 'This', 'was', 'a', 'stand-alone', 'production', 'in', '1975', 'for', 'BBC', 'TV', '\'s', 'anthology', 'series', '``', 'Play', 'for', 'Today', '\'\'', '.']		RIGHT
10	134	['Rumpole', 'of', 'the', 'Bailey', 'Television', 'episodes', 'TV', 'Season', 'One', '(', '1978', ')']		UPL
11	44	['Rumpole', 'of', 'the', 'Bailey', 'Production']		DOWN
12	45	['Rumpole', 'of', 'the', 'Bailey', 'Production', 'Origins']		DOWN
13	46	['Rumpole', 'of', 'the', 'Bailey', 'Production', 'Origins', 'The', 'origins', 'of', 'Rumpole', 'of', 'the', 'Bailey', 'lie', 'in', '``', '', 'Took', 'Place', '\'\'', ',', 'a', 'one-off', 'filmed', 'television', 'play']		ANS
14	46	['Rumpole', 'of', 'the', 'Bailey', 'Production', 'Origins', 'The', 'origins', 'of', 'Rumpole', 'of', 'the', 'Bailey', 'lie', 'in', '``', '', 'Took', 'Place', '\'\'', ',', 'a', 'one-off', 'filmed', 'television', 'play']	John Mortimer	STOP

Figure 13: Navigation example 5. After reading through the "Television episodes" section, the agent goes back to the "Production" section to find the answer.

Incorporating Image Matching Into Knowledge Acquisition for Event-Oriented Relation Recognition

Yu Hong Yang Xu Huibin Ruan Bowei Zhou Jianmin Yao Guodong Zhou*
School of Computer Science and Technology, Soochow University
No.1 Shizi ST, Suzhou, China, 215006
{tianxianer, andreaxu41, hbr416, zoubowei}@gmail.com;
{jyao, gdzhou}@suda.edu.cn

Abstract

Event relation recognition is a challenging language processing task. It is required to determine the relation class of a pair of query events, such as causality, under the condition that there isn't any reliable clue for use. We follow the traditional statistical approach in this paper, speculating the relation class of the target events based on the relation-class distributions on the similar events. There is minimal supervision used during the speculation process. In particular, we incorporate image processing into the acquisition of similar event instances, including the utilization of images for visually representing event scenes, and the use of the neural network based image matching for approximate calculation between events. We test our method on the ACE-R2 corpus and compare it with the fully-supervised neural network models. Experimental results show that we achieve a comparable performance to CNN while slightly better than LSTM.

1 Introduction

Event relation recognition aims to predict the relationship between the query events. Here, an event is defined as a text span (sentence or clause) which describes the occurrence of a genuine event, such as "*The 2nd industrial revolution*". An event-oriented relation recogntion system is required to assign a relation type tag to a pair of query events, such as those defined in Hong et al (2016)'s natural event relation scheme, including causality, temporality, conditionality, etc. Listed below are a pair of related query events, along with the relation need to be predicted:

(1) **Event Instance 1** – *The 2nd industrial revolution* <Cause>
Event Instance 2 – *The killer fog that blanketed London* <Result>
Relation – `Causality` <Need to be predicted>

Recognizing event relations in an automatic way is a challenging task. It is because the query events are selected from different paragraphs in a document or even different documents, so that there is lack of explicit clue and shared context can be used for semantic relation analysis.

Statistics based inference is one of the promising solutions. It performs in a straight-forward manner: i) seeking for the similar events to the query events, ii) surveying the probability distribution of different types of relations over the similar events, and iii) assigning the most widely distributed relation to the query events. In order to fully implement the inference process, we need to address two crucial issues. One is to collect a large set of pairwise event instances whose relations are either previously known or explicitly signaled, such as the ones in (2). The other is to develop effective similarity measurement approaches, so as to retrieve the event instances similar to the query events.

(2) **Query Event 1** – *China's industrial development in the 21st century* <Cause>
Query Event 2 – *Heavy smog alerts issued for Beijing and other cities* <Result>
Relation – `Causality` <Previously Known>

* Corresponding author

Proceedings of the 27th International Conference on Computational Linguistics, pages 177–189
Santa Fe, New Mexico, USA, August 20-26, 2018.

With regard to the first issue, we employ a connective based approach to collect explicitly related events. Current connective-based explicit relation recognition techniques (Pitler and Nenkova, 2009; Wu et al., 2017) have been proven effective in determining the explicit relations, with no less than 93% accuracy, for the sequential text structures connected by a connective. For example, a pair of clauses connected by the connective "*because*" can be determined to have a `causal` relation with a high level of confidence. This allows us to acquire numerous explicitly-related events from texts using a few carefully-selected connectives and simple patterns. Accordingly, we build a large-scale Textual Event Relation Bank (TERB) to support the relation inference.

We focus on the second issue in this paper, measuring the similarity between events. Recently, neural network has been successfully used in event-oriented semantic encoding to some extent (Nguyen and Grishman, 2016; Ghaeini et al., 2016; Feng et al., 2016; Peng et al., 2016; Liu et al., 2017; Chen et al., 2017), yielding substantial performance gains in the related information extraction tasks, such as event extraction (Doddington et al., 2004; Ahn, 2006) and nugget detection (Ellis et al., 2015). Semantic encoding enables the generation of a high-dimensional distributed representation for characterizing an event. So that it prompts semantic learning, computing and understanding at a very deep level. Undoubtedly, this can facilitate the acquisition of the "semantically-similar but pragmatically-different" event instances for the query events, such as those in (2).

Figure 1: Similar visual scenes

However, in our recent research on utilizing semantic similarity calculation, we fail to pass through a bottleneck that, for some query events, there doesn't exist any semantically-similar event instance in a finite dataset (e.g., TERB). It causes that the performance of the state-of-the-art approaches is actually far from what it should be. In order to overcome the problem, we propose to use visual scenes of events for similarity calculation. It is motivated by the fact that some semantically-dissimilar event instances may possess similar visual scenes with the query events. Figure 1 exhibits the scenes of the query events in (1) and that of the event instances in 3), where the visual scenes are similar (compared between the left and the right images), although the textual descriptions of the events are semantically different.

(3) **Event Instance 3** – *Pollution from steel mills blows over residential buildings* <Cause>
Event Instance 4 – *Mask wearing is in fashion* <Result>
Relation – `Causality` <Previously known>

In our experiments, images are taken as the visual scenes. On the basis, we introduce image captions into cross-media semantic matching, with the purpose of mining possible visual scenes by similarity measurement between query events and image captions. In addition, the Convolutional Neural Network (CNN) based image representation is utilized in visual scene matching. Over the ACE-R2, we compare our model with two fully-supervised discourse relation classification models, including Qin et al. (2016)'s CNN and Chen et al. (2016)'s Long-Short Term Memory (LSTM) based Recurrent Neural Network. Experimental results show that our minimally-supervised model slightly outperforms LSTM and obtain comparable performance with CNN.

In the rest of the paper, we overview the related work in section 2. And then we present the methodological framework in section 3, the caption based cross-media semantic matching in section 4 and image matching in section 5. Section 6 will give the TERB establishment method. In section 7, we evaluate the proposed method and analyze the experimental results. We conclude the paper in section 8.

2 Related Work

2.1 Causal and Temporal Event Relation Identification

So far, the study of event relation parsing mainly concentrates on identifying two kinds of relationships, causality and temporality respectively. The early work on causality identification can be traced back to

the use of lexical-syntactic patterns (Girju et al., 2002; Girju, 2003). Soon thereafter, Chang and Choi (2004) revise Girju et al. (2002)'s patterns using lexical pairs (LP) and cue phrases (e.g., *due to*). Besides they generalize the model for binary relation classification using the Bayes theorem. Abe et al. (2008) further use co-occurrence probability as the novel feature. Recently, scholars have made a great effort to model fine-grained causal relations (Inui et al., 2005), exploit the features for classification (Blanco et al., 2008), refine the patterns by syntactic and discourse parsing (Ittoo and Bouma, 2011; Do et al., 2011) and predict relations by semantic network (Radinsky et al., 2012).

Mani et al. (2006) and Lapata and Lascarides (2006) presented the first study on the temporal relation parsing. Both focus on the machine learning approaches. In the past decade, the SemEval (Verhagen et al., 2007; Pustejovsky and Verhagen, 2009) has promoted a great deal of experimental study, including on the grammatical, syntactic, semantic and ordering features (Bethard and Martin, 2007; Hepple et al., 2007; Puşcaşu, 2007; Bethard, 2013; Mirza and Minard, 2015; Caselli et al., 2015; Hashimoto et al., 2015). Meanwhile, temporal relation modeling has been implemented in different ways, such as sequence labeling, Markov logic networks and hybrid systems (Cheng et al., 2007; Min et al., 2007; Yoshikawa et al., 2009; UzZaman and Allen, 2010; Llorens et al., 2010; Velupillai et al., 2015).

2.2 Multi-class Discourse Relation Classification

In April 2006 (Prasad et al., 2007), the Penn Discourse Tree Bank (PDTB) was released as a corpus of discourse-level arguments as well as annotations of explicit and implicit relations. The PDTB relation scheme consists of 4 main relation classes and 16 sub-classes. Since it is released together with the corpus, a great deal of research has focused on the methodologies for multi-class relation classification.

Motivated by Marcu and Echihabi (2002)'s work, in the earlier study, both feature engineering and sophisticated machine learning dominated the field in a large region (Pitler and Nenkova, 2009; Lin et al., 2009; Louis et al., 2010; Park and Cardie, 2012; Rutherford and Xue, 2014).

Recently, it becomes increasingly popular to use neural networks for learning to classify discourse relations (Zhang et al., 2015; Qin et al., 2016; Chen et al., 2016; Qin et al., 2017; Liu and Li, 2016). Typically, Zhang et al (2015) propose a Shallow Convolutional Neural Network (SCNN) model, which is used by Ponti and Korhonen (2017) to identify contingent relation. Qin et al (2016) utilize CNN with a collaborative gated neural network (CGNN) for recognizing implicit relations. Chen et al (2016) develop a Gated Relevance Network (GRN) based on Bidirectional Long-Short Term Memory (Bi-LSTM) framework, combining bilinear model and single layer network for relevance measurement between arguments. Besides, the current work improves the classification performance of implicit relations by expanding the training data with connectives (Rutherford and Xue, 2015; Braud and Denis, 2016; Wu et al., 2017) or combining multiple corpora via multi-task learning (Liu et al., 2016).

3 Relation Inference Engine

In our statistical inference process, TERB is an indispensable knowledge base and needed to be build first. But in this section, we suppose that TERB has been established successfully, and focus on presenting the inference approach. TERB building will be treated as a separate work and presented in section 6.

3.1 Framework and Terminology

The input of the relation inference engine is a pair of query events, while the output is a relation type tag. The inference engine is constituted of three components, including Cross-Media Semantic Matching (CMSM), image matching and inference model (see the framework in Figure 2). In order to facilitate understanding of the following discussion, we first define the terminologies used as below.

Caption is the textual annotation of an image.

Mention is the textual description of an event.

Visual Scene is an image that best describes how an event happens as well as the surroundings. There is no regard to any concrete element or fact (such as who the participants are).

CMSM verifies whether an event mention has the consistent meaning with a scene, or in other word, the mention evokes the perception of the scene. In our method, CMSM plays an important role because

it helps transform the textual representation (i.e., mention) of an event to a visual version (scene), or vice versa. There are two CMSMs of different directions included in the inference process: forward CMSM and backward CMSM. They are methodologically the same, both measuring similarity between captions and mentions. The difference lies in the use of CMSM. The forward CMSM is used to transform mentions into scenes, while the backward CMSM is applied in a scene-to-mention direction.

Image Matching is put forth for calculating scene similarity. It is conducted between the scenes of the query events and that of the events in TERB. The events in TERB which have the similar scenes with the queries will be adopted, along with their explicit relations. They are used as the reference samples.

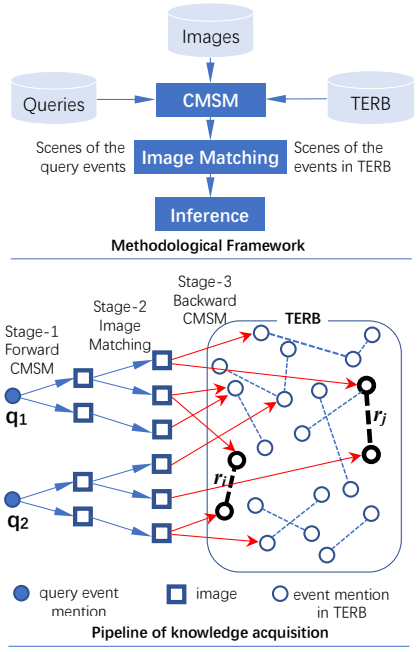


Figure 2: Framework and workflow

3.2 Pipeline for Reference Sample Acquisition

We acquire the reference samples by a three-stage information retrieval system, which consists of three successive one-to-many retrieval stages (see the pipeline in Figure 2):

Stage 1: Mention-to-Scene transformation

The forward CMSM is performed. It is used to retrieve top-n_1 most possible scenes of the query event in a large-scale image database. The images whose captions are most similar to the query event mention will be adopted. The image database (D) we use includes about 5 million images crawled from Wikipedia.

Stage 2: Scene-Scene matching

Each scene obtained in the first stage is used as a query of image search, where image matching is used. There are n_2 most similar images adopted from the image database D, and used as the similar scenes to that of the query event.

Stage 3: Scene-to-Mention transformation

The similar scenes are then used as queries. For each of them, the backward CMSM is performed, so as to acquire n_3 semantically-similar mentions in TERB.

By the knowledge acquisition, for a query event, we can obtain N (N=$n_1 \times n_2 \times n_3$) reference samples in TERB. Thus, given a pair of query events q_1 and q_2, we obtain a collection S of pairs of reference samples (in the size of $N \times N$). Most of the pairs have gone out of use in practice because they are irrelevant and fail to hold a relation. The rest will be reserved as the available reference samples. Their relations are named as reference relations (see r_i and r_j in Figure 2).

3.3 Statistical Inference

Given the set S of reference relations, we use the maximum likelihood estimation to speculate the relation r^* between the query events:

$$r^* = argmax\ p(r) \quad \exists r \in R \qquad p(r) = (\omega_r)^{-1} \times \frac{C(r)}{\sum_{r \in R} C(r)} \qquad \omega_r = \frac{\check{p}(r)}{\lambda_r} \tag{1}$$

where, R denotes all kinds of predefined relation types. We follow Hong et al. (2016)'s relation scheme to discrminate among different relation types. The function $C(r)$ computes the occurrence frequency of the relation r in S. The coefficient ω_r is prodced with a penalty factor λ_r and the prior probability $\check{p}(r)$ of r. The penalty λ_rs of different relation types are inconsistent and need to be fine-tuned on a development set. The prior probability $\check{p}(r)$ of every considered relation type need to be obtained on the training set beforehand. In this paper, we calculate $\check{p}(r)$ using the distribution of r in TERB.

4 Cross-Media Semantic Matching (CMSM)

We implement CMSM between a scene (image) and a mention as the semantic approximation calculation between the caption of the image and the mention. Thus it can be boiled down to a text matching problem.

Activation Function	Loss Function
$tanh_\theta(x) = \frac{e^{\theta^\top x} - e^{\theta^\top x}}{e^{\theta^\top x} + e^{\theta^\top x}}$	$L = -\sum_r y_r \cdot log(s_\theta(x)_r)$
Batch size: 256	**Epoch**: 5
Optimizer: Adadelta **Learning rate**: 10^{-3}	**Dropout ratio**: 0.5 (FC)

Table 1: Hyper-parameter settings for CMSM

Activation Function	Loss Function
$Relu_\theta(x) = ln(1 + e^{\theta^\top x})$	$L = -\sum_r y_r \cdot log(s_\theta(x)_r)$
Batch size: 256	**Epoch**: 74
Momentum: 0.9	**Padding**: 1 pixel
Optimizer: Adadelta **Learning rate**: 10^{-5}	**Dropout ratio**: 0.5 (FC1) **Dropout ratio**: 0.5 (FC2)

Table 2: Hyper-parameter settings for ConvNet

(Note: θ represents all the parameters and $s(*)$ is the ground truth (Vadapalli and Gangashetty, 2016))

For a caption and a mention, we encode each of them as a sentence-level embedding (Sen2vec). We calcualte the cosine similarity of the Sen2vecs and use it as the caption-mention approximation. CNN is utilized for generating the Sen2vecs. As usual, it produces the convolutional features on the concatenated word embeddings of the words in the input short text. There is only 1 hidden layer included in the network. Thus it fails to possess very deep-level perceptions of semantics. Undoubtedly, it can be enhanced by either adding more hidden layers to the network, or instead, using other state-of-the-art network models, such as attention based bidirectional LSTM (Zhou et al., 2016) or gated recurrent unit (Vadapalli and Gangashetty, 2016). In this paper, we choose to use a relatively simple model because we are more willing to verify the validity of the methodological framework.

The configuration of the employed CNN is presented as below: in the **input layer**, a short text (caption or mention) is represented as a fixed-size sequence of real numbers, involving 30 256-dimensional word embeddings. Zero padding is performed when the text length is smaller than 40, otherwise tail clipping. We follow Mikolov et al. (2013) to use skip-gram based word2vec to compute embeddings, and conduct training on the English articles in the latest 2015 Wikipedia dump (Dos Santos and Gatti, 2014). In the **hidden layer**, there are 128 (3×256) filters used for the convolutional computation with a stride of 1. This yields 28 128-dimensional feature vectors. Max pooling is then used to produce a lower dimensional (1×128) vector. As usual, we apply a dense layer to slightly increase the depth. And further, a dropout layer (rate=0.5) is used to produce a 64-dimensional vector. The vector produced in this way is specified as the semantic representation of the text. In the **output layer**, a fully-connected (FC) layer is used, followed by a 18-way softmax classification layer. We train the model on about 50 thousand short texts of 18 domains. Table 1 shows the parameter settings.

It is noteworthy that the domain classification has nothing to do with the task mentioned in this paper. What we really need in the training process is just the well-trained network. The network can be practically used to generate the sentence embeddings (i.e., the ones in FC layer) for captions and mentions.

5 Image Matching

Image matching is used to recognize similar visual scenes and thus enables the events of similar scenes to be acquired. Similarly, we apply the cosine similarity between image embeddings to align images. Simonyan and Zisserman (2014)'s ConvNet is employed to generate the image embeddings.

ConvNet provides an architecture for learning visual features. Each feature indicates a spatial concept of an image patch (only in the input layer) or a multiple-cell receptive field (in hidden layers), preserving the local information about the notions of left, right, top, down and center.

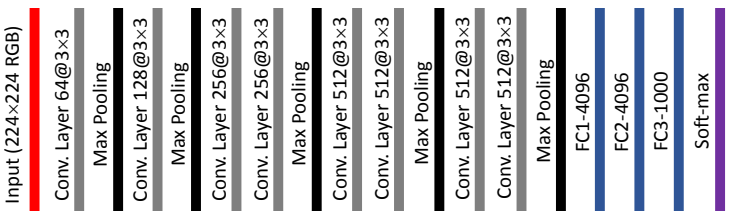


Figure 3: Structure of ImageNet (Plan-A)

In this paper, we follow Simonyan and Zisserman (2014)'s Plan-A, in which a relatively shallower ConvNet is established. The deeper versions in their other plans weren't taken into consideration. It is for the same reason that we mentioned earlier, evaluating the feasibility of our methodological framework when the models we use are weaker than the state of the art.

The ConvNet structure is presented as below and the parameter settings are listed in Table 2. In the **input layer**, a fixed-size 224×224 RGB image is input to the network. The preprocessing we do includes two aspects. One is to normalize images in batches by limiting the size of each to be uniformly 224×224 pixels. The other is to subtract the mean RGB value from each pixel. The mean RGB is computed on the training set. In the **hidden layers**, there are totally 8 convolutional layers deployed, along with 5 max-pooling layers. The number of filters in the convolutional layers is gradually increased (from 64 to 512) with depth. See the layout in Figure 3. In the **output layers**, there are 3 fully-connected (FC.) layers. The first two have 4,096 dimensions each. The third (FC3) is required to perform 1,000-way ILSVRC (Witten et al., 2016) classification during the training procedure and thus contains 1,000 dimensions. A non-linear softmax layer is deployed behind the FCs.

We train the network on ILSVRC-2012. There are about 5 million images used. In our experiments, we pass an image through the well-trained network and employ the FC3 layer as the image embedding. Instead of reproducing the network, we recommend the potential users to use the open source toolkit[1].

6 Textual Event Relation Bank (TERB)

TERB is an event and relation databank we build. There are about 0.74 million pairs of event mentions included in TERB. The mentions are automatically extracted in pairs from Gigaword corpus (Graff and Cieri, 2003). Each pair of events has been exclusively assigned with an exact relation. Below is a standard sample in TERB: <Event Instance 1: "*pollution emission level keeps rising*"> + **causal** relation + <Event Instance 2: "*mask wearing is in fashion*">.

We only take clause-level event mentions into consideration during the process of building TERB. And the eligible clauses for use are limited to the ones that contain a predicate. Syntactic parsing is used for clause segmentation and predicate identification (Björkelund et al., 2010).

There are three patterns used for pairwise mention extraction, including CEE, ECE and EEC, which are distinguishable by means of the position of a connective (C) relative to the neighbor event mentions (E). Below are the examples, where the words in bold font are connectives: $\{CEE-\textbf{\textit{Although}} < Clause1 > , < Clause2 >\}$; $\{ECE- < Clause1 >, \textbf{\textit{but}} < Clause2 >\}$; $\{EEC- < Clause1 >, < Clause2 >, \textbf{\textit{though}}\}$.

The connectives are the words that structurally link sentence constituents. They have been widely used as the reliable clues for explicit semantic relation resolution between text spans. For example, the connective "*because*" frankly signals a causal relation. Nowadays, the utilization of connectives contributes to the high accuracy of explicit relation recognition, reaching a score more than 93% (Pitler and Nenkova, 2009; Wu et al., 2017). Thus, for a pair of event mentions extracted by the patterns, we determine their explicit relation by the one-to-one correspondence between connectives and relation types.

Relational Types	**Num**
Expansion.Conjunctive (Cnj.)	74
Contingency.Conditional (Cnd.)	180
Comparison.Concessive (Cnc.)	1
Temporality.before/after (Baf.)	919
Temporality.during (Dur.)	129
Expansion.Confirming (Cnf.)	19
Contingency.Causal (Cus.)	505
Comparison.Contrastive (Cnt.)	60
Temporality.Equal (Equ.)	107
Coreferential (Cre.)	277

Table 3: Relation scheme

We employ 50 connectives which are elaborately collected from the PDTB corpus (Prasad et al., 2007). The relations they signal have been manually annotated beforehand and double-checked. We filter some ambiguous connectives before use because they generally signal different types of relations. For example, "*since*" may signal a causal relation with the meaning of "*because*", while in some cases, it signals a temporal relation with the meaning of "*from then on*".

7 Experimentation

7.1 Corpus, Settings and Evaluation

Corpus— We use Hong et al. (2016)'s ACE-R2 corpus in our experiments. Table 3 shows the relation scheme, which consists of 5 main types and 10 subtypes. ACE-R2 contains 2,271 pairs of news events.

[1]https://github.com/BVLC/caffe

Each is annotated with a sole relation. The mentions are selected from the Automatic Content Extraction (ACE) corpus (Doddington et al., 2004).

Settings— Both the CNN models and the captioning model that we use in the experiment have been pretrained with external data. What we need to fine-tune is the number of the reference samples (section 3.2), i.e., N, which is produced by the parameters n_1, n_2 and n_3 (N=$n_1 \times n_2 \times n_3$) in the three-stage knowledge acquisition process. A larger value of N will introduce many noises in our inference process. By contrast, a smaller value causes the lack of reference samples and thus the statistical uncertainty.

Evaluation— The methods we concern are evaluated by the metrics of macro-average Precision (Mac-P), Recall (Mac-R) and F-score (Mac-F). For a particular type t, the positive examples are defined as the event pairs that hold the relation t. Thus for t, the precision score is calculated as the ratio of the positive examples in all the output examples of type t. The recall is defined as the ratio of the output positive examples in all the ground-truth ones.

7.2 Compared Methods

Based on the framework mentioned in section 3, we implement an event relation predictor, named as **Holmes**. For the purpose of validating statistically non-random effect, Holmes is compared with the weighted random sampling (**Baseline 1**). By the baseline method, a test sample is most probably determined to hold one of the widely-distributed relations (such as Temporality). The distribution is computed over the development set. Second, Holmes is required to compete with a pure text based approach (**Baseline 2**). In this approach, the statistical inference (section 3.3) is still followed, and similarly to Holmes, the CNN based Sen2vec (section 3) is involved, and devoted to representing the query event mentions and those in TERB. But unlike Holmes, it skips the steps of CMSM and image matching, and instead, directly acquires n_{bas} similar events from TERB by sentence-level embedding similarity.

In addition, we compare Holmes with two discourse classification models, which are based on more general models and trained in a fully supervised fashion. One is Qin et al. (2016)'s **CNN** model. The other is Chen et al. (2016)'s bidirectional LSTM (**Bi-LSTM**) based recurrent model. There are two kinds of performance of the competitors are reported. One is achieved by training the competitors over the standard PDTB corpus (sections 1-20), which is consisted of no more than 6,234 handmade sentence-level argument pairs and relations. The other is achieved by training the competitors over the TERB corpus, a large-scale set of automatically-extracted mentions and explicit relations.

Most of the relation types in PDTB are compatible with those in ACE-R2 and TERB. For example, the `Temporality` has been divided in two subtypes, `Synchronous` and `Asynchronous`, which are compatible with the `Temproality.Equal&During` and `Temporality.Before&After`. However PDTB fails to include the `Coreference` relation. Therefore, it isn't considered in the experiments. Besides, in ACE-R2, the numbers of available instances of the subtypes `Confirming` and `Concessive` are far from large (see Table 3). This makes it difficult to develop Holmes with a high-level confidence. Therefore, the two types aren't taken into consideration too. Thus the total number of available instances in ACE-R2 is 1,974. We use 1,579 as the development set, occupying about 80 percent of all, while the rest (395) for test. We run 5-fold cross validation and report the best performance, while the performance in every validation experiment will be presented in the discussion subsection.

7.3 Experimental Results

Knowledge Acquisition Performance — In the development process, we fine-tune the parameters n_1, n_2 and n_3 for Holmes, and the parameter n_{bas} for the Baseline 2. The NDCG@N metric is used to access the quality of the obtained reference samples (i.e., the retrieved similar events). For the image based retrieval approach in Holmes, N is the product of n_1, n_2 and n_3, while for the text based approach in Baseline 2, N equals to n_{bas}. The high value of NDCG@N implies that most of the N references highly ranked by similarity scores are reliable, holding the ground-truth relation of the query events.

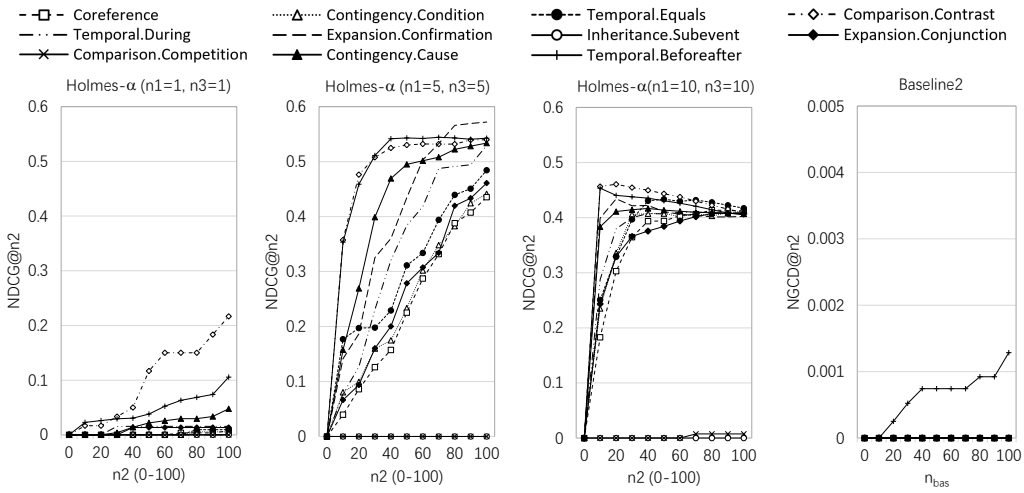


Figure 4: Partial development stages for Holmes-α

Type	Mac-P	Mac-R	Mac-F
Expansion	0.17	0.08	0.11
Contingency	0.82	0.80	0.81
Comparison	0.04	0.07	0.05

Table 4: Best Performance of Holmes for the three main relation types

Subtypes	Mac-P	Mac-R	Mac-F
Contingency.Cus	0.79	0.64	0.71
Contingency.Cnd	0.35	0.53	0.42

Table 5: Results on subtypes of `Contingency`

Subtypes	Mac-P	Mac-R	Mac-F
Temporality.Baf	0.79	0.97	0.87
Temporality.Equ&Dur	0.25	0.04	0.07

Table 6: Results on subtypes of `Temporality`

Figure 4 illustrates partial stages in the development process before the performance remains steady. For example, the subgraph 2 shows the changing tendency of NDCG when n_2 is gradually adjusted, and n_1 and n_3 are kept unchanged. It can be observed from the NDCG curves that the image based approach performs better than the text based (i.e., Baseline 2), obtaining more reliable reference samples. Using such samples, as presented below, we enable Holmes to achieve comparable performance to the sophisticated supervised learning models. It proves that the image based approach performs better than the text based approach in acquiring reference samples. Using such samples, as presented below, we enable the simple statistical inference to outperform some sophisticated supervised learning models.

On the basis, we fine-tune the penalty factor λ for different types of relations. In each-fold cross validation, the penalty factor of a particular relation type will be adopted if it effectively cooperates with the penalty factors of other relation types. The effectiveness is ensured by verifying whether the factors enable the recognizer to achieve the best performance on the development set. We have made the source codes, penalty factors and datasets publicly available[2], so as to enable the reproduction of the whole experiments.

Relation Recognition Performance — We test Holmes by setting the parameters n_1, n_2 and n_3 as 10, 85 and 10. Table 4 shows the performance for the three main relations `Contingency`, `Expansion` and `Comparison`. It is the best performance Holmes achieves in the 5-fold cross validations. Note that the performance for the `Temporal` relation type is evaluated separately and only a 2-way classification is conducted for its subtypes. It is because that every event pair can be regarded to be related temporally, although some of relations fail to be annotated in ACE-R2. The lack of annotations on `Temporality` definitely causes biased assertions on the performance of automated relation prediction.

[2]https://github.com/HuiBinR/VSRB

Method	Training	3-way	Temp	Cont
Baseline1	PDTB	0.11	N/A	N/A
Baseline2	N/A	0.12	N/A	N/A
CNN	PDTB	0.38	0.17	N/A
CNN	**TERB**	0.41	**0.54**	0.60
Bi-LSTM	PDTB	0.39	0.17	N/A
Bi-LSTM	**TERB**	**0.42**	0.48	**0.63**
Holmes	N/A	0.33	0.51	**0.63**

Table 7: Evaluated by General Macro-F

Method	Training	3-way	Temp	Cont
Baseline1	PDTB	0.11	N/A	N/A
Baseline2	N/A	0.12	N/A	N/A
CNN	PDTB	0.36	0.17	N/A
CNN	**TERB**	0.32	**0.48**	0.56
Bi-LSTM	PDTB	0.34	0.17	N/A
Bi-LSTM	**TERB**	**0.40**	0.45	**0.58**
Holmes	N/A	0.31	0.47	0.56

Table 8: Evaluated by Average Macro-F

It is observed that Holmes precisely predicts the `Contingency` relation. Most of the contingently-related instances have been successfully recalled. By contrast, Holmes performs much worse for `Expansion` and `Comparison`. We overview the errors and analyze the reasons as below.

Due to the omission of `Expansion.Confirming`, the performance of Holmes on `Expansion` actually derives from that on `Expansion.Conjunctive`. In general, `Conjunction` appears as a rhetoric method. By `Conjunction`, a series of similar events can be enumerated, such as *earthquake*, *tsunami* and *storm*. Co-occurrence probabilities of such events play important roles for predicting `Conjunction`. However, there are few connectives can be used for pursuing the co-occurred events or they are extraordinarily general. For example, the connective "*and*" is frequently used to signal the co-occurred events, but frankly it nearly connects kinds of linguistic units or even acts as a pause from the perspective of mood. If we use "*and*" to collect sample events which hold conjunctive relations, TERB will be full of pseudo-instances, such as "*sunshine **and** beach*". This will reduce the precision more seriously. By contrast, if we neglect a connective "*and*", there will be lack of available sample events. This inevitably results in the incomplete statistics for the frequently-cooccurred events. We didn't overcome the bottleneck in this paper and use no more than 6 uncommon connectives to collect the sample events. Both the diversity and scale of the events are far from expectation. This makes it difficult to effectively recognize the `Conjunctive` relationship using a statistical approach.

We encounter a similar problem when treating with `Comparison`. A connective like "*than*" is effective to signal a pair of `Contrastive` events. However, due to the limitation of literal expression, in general, such event mentions aren't directly connected by "*than*", but instead the concrete elements in the events are. For example, in a story about *economic competition*, the contrastive events are introduced respectively in different sentences, on the contrary the *profit forecasts* in the events are connected by *than* in a sentence. The neglect of commonly-used connectives (e.g., "*than*") during TERB building causes the lack of sample events. Similarly, the statistic strategy we use fails to perform better under this condition.

Table 5 shows the performance of Holmes on the `Contingent` sub-type relations, i.e., `Cause` and `Condition`. It can be observed that Holmes is adept in predicting causal relations. Table 6 exhibits the performance on the `Temporal` sub-type relations, i.e., `Before&After` and `Equal&During`. It is illustrated that Holmes effectively predicts the synchronous events, though fails to infer the asynchronous.

7.4 Discussion

We carefully evaluate Holmes and the competitors with the general Macro F and average Macro F scores respectively. The former is calculated with the average Macro P and Macro R on the concerned relation types, while the latter is the average value of Macro F on the types. The average Macro F score plays a role of assistance because it is able to reveal the unbalance problem. If a system shows significantly different performance on different types of relations. the average Macro F will be substantially lower than the general. Tables 7 and 8 respectively exhibit the performance of the entrants for the 3 main relation types (3-way), `Temporal` subtypes (Temp) and `Contingent` subtypes (Cont). The tables only exhibit the best performance the competitors achieves through the 5-fold cross validations. The performance in every validation experiment is shown in Figure 5 and 6.

It can be observed that Holmes performs worse than CNN and LSTM for the 3-way main relation classification. Nevertheless, Holmes obtains a smaller gap between general and average Macro F scores. Besides, it achieves a comparable performance to the well-trained CNN and LSTM for the fine-grained sub-type relations. For the case of `Contingency`, Holmes reaches the top together with LSTM. Note

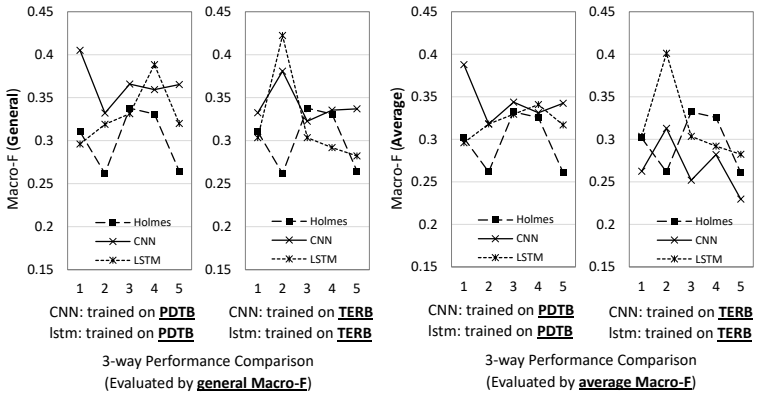


Figure 5: Cross validation for main relation types (`Contingency`, `Comparison` and `Expansion`)

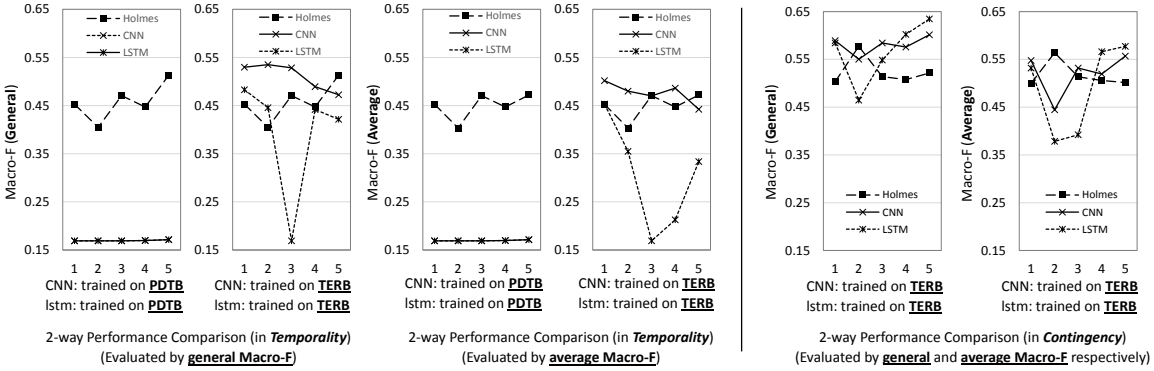


Figure 6: Cross validation for the subtypes in `Temporality` and `Contingency`

that CNN and LSTM fail to be trained on PDTB for classifying the `Contingent` sub-type relations. It is because PDTB contains only one implicit conditionally-related argument pair. The scale of training data and domain adaptation impose great influences on the performance. For example, for the subtypes, CNN and LSTM perform significantly worse than Holmes when trained on PDTB. In addition, Holmes outperforms Baseline 2. It demonstrates that, in TERB, there is lack of semantically-consistent events to the queries. Holmes bypasses the bottleneck, using CMSM and image matching to acquire visually-similar events. This contributes to the reference sample based statistical inference.

When we look through the 5-fold cross validation, we find that Holmes actually performs better than expectation. As shown in Figures 5 and 6, Holmes seldom shows an exceptionally high or an unusually low Marcro F score. And the mean level of Macro F scores of Holmes appears to be no less than that of LSTM for the subtypes of both `Temporality` and `Contingency`.

8 Conclusion

We demonstrate that the use of images also contributes to knowledge acquisition in the field of linguistic computing. By image matching, we open up a new perspective for the identification of similar events, i.e., measuring the similarity of visual scenes. On the basis, we have successfully acquired a variety of events which have comparable scenes with the queries. Using the obtained events and their explicit relations, we enable a simple statistical inference model to achieve competitive performance for event relation recognition. In the future we will introduce active learning into the refinement of the collected event instances, and expand the existing training data to strengthen the supervised classification models.

Acknowledgements

We thank Siyuan Ding and Liang Yao who have made great efforts on this work. This work was supported by the national Natural Science Foundation of China (Nos. 61751206, 61672368, 61672367).

References

Shuya Abe, Kentaro Inui, and Yuji Matsumoto. 2008. Acquiring event relation knowledge by learning cooccurrence patterns and fertilizing cooccurrence samples with verbal nouns. In *IJCNLP*, pages 497–504.

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8. Association for Computational Linguistics.

Steven Bethard and James H Martin. 2007. Cu-tmp: Temporal relation classification using syntactic and semantic features. In *Workshop on Semantic Evaluations*, pages 129–132.

Steven Bethard. 2013. Cleartk-timeml: A minimalist approach to tempeval 2013. In *Second Joint Conference on Lexical and Computational Semantics*, volume 2, pages 10–14.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 33–36. Association for Computational Linguistics.

Eduardo Blanco, Nuria Castell, and Dan I Moldovan. 2008. Causal relation extraction. In *LREC*.

Chloé Braud and Pascal Denis. 2016. Learning connective-based word representations for implicit discourse relation identification. In *EMNLP*, pages 203–213.

Tommaso Caselli, Antske Fokkens, Roser Morante, and Piek Vossen. 2015. Spinoza vu: An nlp pipeline for cross document timelines. *SemEval-2015*, page 787.

Du-Seong Chang and Key-Sun Choi. 2004. Causal relation extraction using cue phrase and lexical pair probabilities. In *IJCNLP*, pages 61–70. Springer.

Jifan Chen, Qi Zhang, Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Implicit discourse relation detection via a deep architecture with gated relevance network. In *ACL (1)*.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 409–419.

Yuchang Cheng, Masayuki Asahara, and Yuji Matsumoto. 2007. Naist. japan: temporal relation identification using dependency parsed tree. In *Semantic Evaluations*, pages 245–248.

Siyuan Ding, Yu Hong, Shanshan Zhu, Jianmin Yao, and Qiaoming Zhu. 2016. Combining event-level and cross-event semantic information for event-oriented relation classification by scnn. In *China National Conference on Chinese Computational Linguistics*, pages 216–224. Springer.

Quang Xuan Do, Yee Seng Chan, and Dan Roth. 2011. Minimally supervised event causality identification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 294–303. Association for Computational Linguistics.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC*, volume 2, page 1.

Cícero Nogueira Dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78.

Joe Ellis, Jeremy Getman, Dana Fore, Neil Kuster, Zhiyi Song, Ann Bies, and Stephanie Strassel. 2015. Overview of linguistic resources for the tac kbp 2015 evaluations: Methodologies and results. In *Proceedings of TAC KBP 2015 Workshop, National Institute of Standards and Technology*, pages 16–17.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 66–71.

Reza Ghaeini, Xiaoli Fern, Liang Huang, and Prasad Tadepalli. 2016. Event nugget detection with forward-backward recurrent neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 369–373.

Roxana Girju, Dan I Moldovan, et al. 2002. Text mining for causal relations. In *FLAIRS*, pages 360–364.

Roxana Girju. 2003. Automatic detection of causal relations for question answering. In *ACL*, pages 76–83.

David Graff and C Cieri. 2003. English gigaword corpus. *Linguistic Data Consortium*.

Chikara Hashimoto, Kentaro Torisawa, Julien Kloetzer, and Jong-Hoon Oh. 2015. Generating event causality hypotheses through semantic relations. In *AAAI*, pages 2396–2403.

Mark Hepple, Andrea Setzer, and Rob Gaizauskas. 2007. Usfd: preliminary exploration of features and classifiers for the tempeval-2007 tasks. In *Workshop on Semantic Evaluations*, pages 438–441.

Yu Hong, Tongtao Zhang, Tim OGorman, Sharone Horowit-Hendler, Heng Ji, and Martha Palmer. 2016. Building a cross-document event-event relation corpus. *LAW X*, page 1.

Takashi Inui, Kentaro Inui, and Yuji Matsumoto. 2005. Acquiring causal knowledge from text using the connective marker tame. *TALIP*, 4(4):435–474.

Ashwin Ittoo and Gosse Bouma. 2011. Extracting explicit and implicit causal relations from sparse, domain-specific texts. In *Natural Language Processing and Information Systems*, pages 52–63. Springer.

Maria Lapata and Alex Lascarides. 2006. Learning sentence-internal temporal relations. *J. Artif. Intell. Res.(JAIR)*, 27:85–117.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 343–351. Association for Computational Linguistics.

Yang Liu and Sujian Li. 2016. Recognizing implicit discourse relations via repeated reading: Neural networks with multi-level attention. *arXiv preprint arXiv:1609.06380*.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural networks. In *AAAI*, pages 2750–2756.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. 1:1789–1797.

Hector Llorens, Estela Saquete, and Borja Navarro. 2010. Tipsem (english and spanish): Evaluating crfs and semantic roles in tempeval-2. In *Workshop on Semantic Evaluation*, pages 284–291.

Annie Louis, Aravind Joshi, Rashmi Prasad, and Ani Nenkova. 2010. Using entity features to classify implicit discourse relations. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 59–62. Association for Computational Linguistics.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *COLING and ACL*, pages 753–760.

Daniel Marcu and Abdessamad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 368–375. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Congmin Min, Munirathnam Srikanth, and Abraham Fowler. 2007. Lcc-te: a hybrid approach to temporal relation identification in news text. In *Workshop on Semantic Evaluations*, pages 219–222.

Paramita Mirza and Anne-Lyse Minard. 2015. Hlt-fbk: a complete temporal processing system for qa tempeval. *SemEval-2015*, page 801.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *EMNLP*, pages 886–891.

Joonsuk Park and Claire Cardie. 2012. Improving implicit discourse relation recognition through feature set optimization. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 108–112. Association for Computational Linguistics.

Haoruo Peng, Yangqiu Song, and Dan Roth. 2016. Event detection and co-reference with minimal supervision. In *EMNLP*, pages 392–402.

Emily Pitler and Ani Nenkova. 2009. Using syntax to disambiguate explicit discourse connectives in text. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 13–16. Association for Computational Linguistics.

Edoardo Maria Ponti and Anna Korhonen. 2017. Event-related features in feedforward neural networks contribute to identifying causal relations in discourse. *LSDSem 2017*, page 25.

Rashmi Prasad, Eleni Miltsakaki, Nikhil Dinesh, Alan Lee, Aravind Joshi, Livio Robaldo, and Bonnie L Webber. 2007. The penn discourse treebank 2.0 annotation manual.

Georgiana Puşcaşu. 2007. Wvali: Temporal relation identification by syntactico-semantic analysis. In *Workshop on Semantic Evaluations*, pages 484–487.

James Pustejovsky and Marc Verhagen. 2009. Semeval-2010 task 13: evaluating events, time expressions, and temporal relations (tempeval-2). In *Workshop on Semantic Evaluations*, pages 112–116.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016. A stacking gated neural architecture for implicit discourse relation classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2263–2270.

Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric P Xing. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. *arXiv preprint arXiv:1704.00217*.

Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. 2012. Learning causality for news events prediction. In *WWW*, pages 909–918.

Attapol Rutherford and Nianwen Xue. 2014. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *EACL*, volume 645, page 2014.

Attapol Rutherford and Nianwen Xue. 2015. Improving the inference of implicit discourse relations via classifying explicit discourse connectives. In *HLT-NAACL*, pages 799–808.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Naushad UzZaman and James F Allen. 2010. Trips and trios system for tempeval-2: Extracting temporal information from text. In *Workshop on Semantic Evaluation*, pages 276–283.

Anandaswarup Vadapalli and Suryakanth V Gangashetty. 2016. An investigation of recurrent neural network architectures using word embeddings for phrase break prediction. In *Interspeech*, pages 2308–2312.

Sumithra Velupillai, Danielle L Mowery, Samir Abdelrahman, Lee Christensen, and Wendy W Chapman. 2015. Blulab: Temporal information extraction for the 2015 clinical tempeval challenge.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. Semeval-2007 task 15: Tempeval temporal relation identification. In *Proceedings of the 4th international workshop on semantic evaluations*, pages 75–80. Association for Computational Linguistics.

Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. 2016. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Changxing Wu, Xiaodong Shi, Yidong Chen, Jinsong Su, and Boli Wang. 2017. Improving implicit discourse relation recognition with discourse-specific word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 269–274.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Katsumasa Yoshikawa, Sebastian Riedel, Masayuki Asahara, and Yuji Matsumoto. 2009. Jointly identifying temporal relations with markov logic. In *ACL and AFNLP*, pages 405–413.

Biao Zhang, Jinsong Su, Deyi Xiong, Yaojie Lu, Hong Duan, and Junfeng Yao. 2015. Shallow convolutional neural network for implicit discourse relation recognition. In *EMNLP*, pages 2230–2235.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212.

Representation Learning of Entities and Documents from Knowledge Base Descriptions

Ikuya Yamada[1,3] ikuya@ousia.jp
Hiroyuki Shindo[2,3] shindo@is.naist.jp
Yoshiyasu Takefuji[4] takefuji@sfc.keio.ac.jp

[1] Studio Ousia, [2] Nara Institute of Science and Technology,
[3] RIKEN AIP, [4] Keio University

Abstract

In this paper, we describe *TextEnt*, a neural network model that learns distributed representations of entities and documents directly from a knowledge base (KB). Given a document in a KB consisting of words and entity annotations, we train our model to predict the entity that the document describes and map the document and its target entity close to each other in a continuous vector space. Our model is trained using a large number of documents extracted from Wikipedia. The performance of the proposed model is evaluated using two tasks, namely fine-grained entity typing and multiclass text classification. The results demonstrate that our model achieves state-of-the-art performance on both tasks. The code and the trained representations are made available online for further academic research.

1 Introduction

The problem of learning distributed representations (or embeddings) from a knowledge base (KB) has recently attracted considerable attention. These representations enable us to use the large-scale, human-edited information of a KB in machine learning models, and can be applied in various natural language tasks such as entity linking (Hu et al., 2015; Yamada et al., 2016; Yamada et al., 2017), entity search (Hu et al., 2015), and link prediction (Bordes et al., 2013; Wang et al., 2014).

In this paper, we describe *TextEnt*, a simple neural network model that learns distributed representations of entities and documents from a KB. Specifically, given a document in a KB consisting of words and *contextual* entities (i.e., entities referred from entity annotations in the document), our model predicts the *target* entity explained by the document (see Figure 1), and maps the document and its target entity close to each other in a continuous vector space. Here, words, contextual entities, and target entities are mapped into continuous vectors that are updated throughout the training. In this study, we train the model using documents retrieved from Wikipedia.

One key characteristic of our model is that it enables us to combine the semantic signals obtained from both words and entities in a straightforward manner. The main motivation for using entities in addition to words is to address the problems of ambiguity (i.e., the same words or phrases may have different meanings) and variety (i.e., the same meaning may be expressed using different words or phrases) in natural language. For example, the word *Washington* is ambiguous because it can refer to a US state, or the capital city of the US, or the first US president *George Washington*, and so on. Further, *New York* is sometimes referred to as *NY* or by its nickname, the *Big Apple*. Obviously, entities do not have these problems, because they are uniquely identified in the KB.

To evaluate our model, we address two important tasks using the proposed representations. Firstly, we consider a fine-grained entity typing task (Yaghoobzadeh and Schutze, 2015) to evaluate the quality of the learned entity representations. In this task, the aim is to infer one or more types of each entity (e.g., *athlete*, *airport*, *sports_team*) from a predefined type set. We perform this task using the simple multilayer perceptron (MLP) classifier with the learned entity representations as features. The results show that our method outperforms the state-of-the-art methods by a wide margin.

Proceedings of the 27th International Conference on Computational Linguistics, pages 190–201
Santa Fe, New Mexico, USA, August 20-26, 2018.

Saturn is the sixth planet from the Sun and the second-largest in the Solar System, after Jupiter. It is a gas giant with an average radius about nine times that of Earth.

Contextual entities: *Planet, Sun, Solar System, Jupiter, Gas giant, Earth*

Target entity: *Saturn*

Figure 1: Example of a KB document with entity annotations.

Secondly, we consider a multiclass text classification task, which aims to classify documents into a set of predefined classes. This task examines the ability of our model as a generic encoder of arbitrary documents. One important approach adopted here is that we automatically annotate entities appearing in the target documents using a publicly available entity linking system and encode the documents to the document representations in the same manner as the documents in the KB. For this task, the logistic regression classifier is applied to the document representations. Because of the quality of semantic signals obtained from the entities, our method outperforms strong state-of-the-art methods on two popular datasets (i.e., the 20 newsgroups dataset (Lang, 1995) and R8 dataset (Debole and Sebastiani, 2005)). To facilitate further research, our code and the trained representations are available online at `https://github.com/studio-ousia/textent/`.

Our contributions can be summarized as follows:

- We propose *TextEnt*, a simple neural network model that learns distributed representations of entities and documents from a KB. Given a document in a KB consisting of words and contextual entities, our model learns the representations by predicting the target entity explained by the document (see Figure 1). We train our model using large-scale documents extracted from Wikipedia.
- Our proposed model allows us to effectively combine the semantic signals retrieved from both words and entities in a straightforward manner. We demonstrate the effectiveness of this feature by addressing two important tasks: fine-grained entity typing and text classification. Despite the simplicity of our approach, we achieve state-of-the-art results in both tasks.
- We have published our code and the trained representations online to facilitate further academic research.

2 Our Method

In this section, we describe our approach of learning distributed representations of entities and documents from a KB.

2.1 Model

Given a document D in a KB consisting of a set of words $w_1, ..., w_N$ and a set of contextual entities $e_1, ..., e_K$, we train our model to predict the target entity that the document is explaining. We first derive two vector representations of document D: the word-based representation $\mathbf{v}_{D_w}$ and the contextual entity-based representation $\mathbf{v}_{D_e}$. For simplicity, we compute $\mathbf{v}_{D_w}$ and $\mathbf{v}_{D_e}$ by averaging the vector representations of words and those of contextual entities, respectively.

$$\mathbf{v}_{D_w} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{a}_{w_n},\ \mathbf{v}_{D_e} = \frac{1}{K}\sum_{n=1}^{K}\mathbf{b}_{e_n}, \tag{1}$$

where $\mathbf{a}_w \in \mathbb{R}^d$ and $\mathbf{b}_e \in \mathbb{R}^d$ are the vector representations of words and contextual entities, respectively.

We define a probability that represents the likelihood of entity e_t being the target entity of document D as the following softmax function:

$$P(e_t|D) = \frac{\exp({\mathbf{c}_{e_t}}^{\top}\mathbf{v}_D)}{\sum_{e' \in E_{KB}}\exp({\mathbf{c}_{e'}}^{\top}\mathbf{v}_D)}, \tag{2}$$

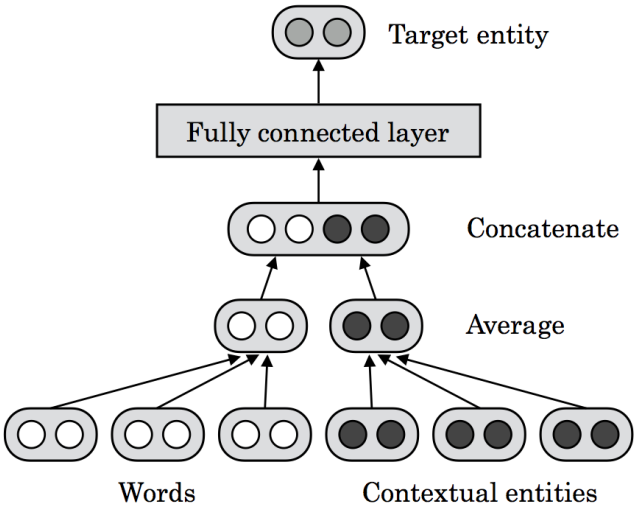


Figure 2: Model architecture of TextEnt.

where E_{KB} is a set of all entities in the KB, $\mathbf{c}_e \in \mathbb{R}^d$ denotes the vector representation of target entity e, and $\mathbf{v}_D \in \mathbb{R}^d$ is the vector representation of document D.

Here, $\mathbf{v}_D$ is computed using a fully connected hidden layer with $\mathbf{v}_{D_w}$ and $\mathbf{v}_{D_e}$ as inputs:

$$\mathbf{v}_D = \mathbf{W}[\mathbf{v}_{D_w}, \mathbf{v}_{D_e}] \tag{3}$$

where $\mathbf{W} \in \mathbb{R}^{d \times 2d}$ is a weight matrix, and $[\mathbf{v}_i, \mathbf{v}_j]$ is the concatenation of $\mathbf{v}_i$ and $\mathbf{v}_j$. This layer projects the input vector ($[\mathbf{v}_{D_w}, \mathbf{v}_{D_e}]$) down to d dimensions, and captures the interactions between $\mathbf{v}_{D_w}$ and $\mathbf{v}_{D_e}$.

We use the categorical cross-entropy loss to train the model:

$$\mathcal{L} = -\sum_{(D,e_t)\in\Gamma} \log P(e_t|D), \tag{4}$$

where Γ represents a set of pairs consisting of a document D and its target entity e_t in the KB.

When training our model, the denominator in Eq. (2) is computationally expensive because it involves summation over all KB entities. To address this, we use negative sampling (Mikolov et al., 2013b); specifically, we replace E_{KB} in Eq. (2) with a set consisting of the target entity e_t and k randomly chosen negative entities. Furthermore, to avoid overfitting, we use *word dropout* (Iyyer et al., 2015), which randomly excludes words and contextual entities with a probability p during the training.

We also test models trained using only words (denoted by *TextEnt-word*) and only contextual entities (denoted by *TextEnt-entity*) in our experiments. These variants are created by replacing $\mathbf{v}_D$ in Eq. (2) with $\mathbf{v}_{D_w}$ (*TextEnt-word*) and $\mathbf{v}_{D_e}$ (*TextEnt-entity*). Hereafter, our original model is referred to as *TextEnt-full*.

2.2 Dataset

We trained our model using documents obtained from the April 2016 version of the DBpedia NIF abstract dataset[1], which contains the texts and entity annotations in the first introductory sections of Wikipedia articles.

For computational efficiency, we limited the size of our dataset. In particular, we excluded documents with fewer than five incoming links from other documents if the corresponding entity of the document is not contained in the dataset used in our fine-grained entity typing experiments, presented in Section 3.1. As a result, the number of target documents was 702,388.

We also modified all words to lowercase, and excluded words that make fewer than five appearances and contextual entities that make fewer than three appearances in the documents. Thus, the final dataset contained 242,771 unique words and 327,263 unique contextual entities.

[1] `http://wiki.dbpedia.org`

2.3 Parameters

The parameters to be trained in our model are the weight matrix $\mathbf{W}$ in the fully connected layer and the vector representations of the words, contextual entities, and target entities. The weight matrix was initialized at random and the vector representations were initialized using pre-trained representations. The pre-trained representations of words and entities were learned jointly using the skip-gram model (Mikolov et al., 2013a; Mikolov et al., 2013b) with negative sampling[2]. The corpus was automatically generated by replacing the name of each entity annotation in the Wikipedia documents with a unique identifier of the entity corresponding to that annotation. Note that we used the same pre-trained entity representations to initialize the representations of the contextual entities and the target entities. Additionally, we used all Wikipedia documents obtained from the July 2016 version of Wikipedia dump[3] to build the corpus.

2.4 Implementation Details

The proposed model was implemented using PyTorch[4] and trained with mini-batch stochastic gradient descent (SGD). The mini-batch size was fixed at 100 and the learning rate was automatically controlled by Adadelta (Zeiler, 2012). We trained the model by iterating over the documents in the KB in random order for 50 epochs[5]. For computational efficiency, we used only the first 2,000 words and first 300 entities in the documents. The training took approximately 25 h on an NVIDIA GTX 1080 Ti GPU. Regarding the other hyper-parameters, the representations were set to have $d = 300$ dimensions, the size of the negative entities was $k = 100$, and the dropout probability was set to $p = 0.5$, as recommended in Srivastava et al. (2014)

3 Experiments

To evaluate the models described in the previous section, we conducted fine-grained entity typing and text classification tasks using the learned representations. A description of each task is given in the following subsections. Finally, we qualitatively analyze the learned representations.

3.1 Fine-grained Entity Typing

This section describes the task of fine-grained entity typing (Yaghoobzadeh and Schutze, 2015; Neelakantan and Chang, 2015; Yaghoobzadeh and Schütze, 2017) using the entity representations learned by our proposed models. The aim of this task is to assign each entity with one or more fine-grained types such as *musician* and *film*. Because an entity typing model is capable of predicting the entity types that are missing from the KB, this can be seen as a *knowledge base completion* problem. The task is important because entity type information is often missing from KBs, but is known to be beneficial for various downstream natural language tasks such as entity linking (Ling et al., 2015), coreference resolution (Hajishirzi et al., 2013), and semantic parsing (Liu et al., 2015).

Setup

Our experimental setup follows that of Yaghoobzadeh and Schutze (2015). In particular, we use their entity dataset of 201,933 Freebase[6] entities mapped to 102 entity types based on the FIGER type set (Ling and Weld, 2012). The dataset consists of a training set (50%), development set (20%), and test set (30%). Because the dataset is constructed based on Freebase, we preprocessed the data by mapping each entity to the corresponding entry in Wikipedia and excluded those entities that did not exist in Wikipedia.[7] As a result, we successfully mapped approximately 92% of the entities to Wikipedia, and obtained training,

[2]We used the skip-gram model implemented in the open-source Gensim library with $size = 300$, $window = 10$, $negative = 15$, $min\_count = 3$, and $iter = 5$. Default values were used for other parameters.

[3]We obtained the Wikipedia dump from Wikimedia Downloads: `https://dumps.wikimedia.org/`

[4]`http://pytorch.org`

[5]We experimented using 10, 20, 30, and 50 epochs. All numbers achieved similar performance in our experiments. We used the model trained for 50 epochs because it achieved the best P@1 performance in our fine-grained entity typing task.

[6]`https://developers.google.com/freebase/`

[7]We used the *wikipedia.en_title* property contained in the Freebase dump to create the mapping.

development, and test sets containing 93,350, 37,036, and 55,715 entities, respectively. We publicized the dataset and the code used to generate the dataset at `https://github.com/studio-ousia/textent/`.

Following Yaghoobzadeh and Schutze (2015), we evaluated the models using ranking and classification measures. The ranking measures test how well a model ranks entity types. In particular, we ranked the entity types based on the probabilities assigned by the model and evaluated the ranked list using the precision at 1 (P@1) and breakeven point (BEP)[8].

The classification measures evaluate the quality of the thresholded assignment decisions of a model. The assignment decisions are based on thresholding the probability assigned to each type. The threshold is selected per type by maximizing the F1 score of entities assigned to the type in the development set. We used the accuracy (an entity is correct if all its types and no incorrect types are assigned to it), micro-average F1 (F1 score of all type–entity assignment decisions), and macro-average F1 (F1 score of types assigned to an entity, averaged over entities). These ranking and classification measures are exactly the same as those used in Yaghoobzadeh and Schutze (2015).

Method

We used an MLP classifier with the entity representations as inputs to predict the probability of entity e being a member of type t in the set of possible types T. In particular, we used an MLP with a single hidden layer and the tanh activation function, and an output layer that contains, for each possible type $t \in T$, a logistic regression classifier that predicts the probability of t:

$$\big[P(t_1|e), ..., P(t_{|T|}|e)\big] = \sigma\big(\mathbf{W}_o \, tanh\,(\mathbf{W}_h \mathbf{c}_e)\big), \tag{5}$$

where $\mathbf{c}_e \in \mathbb{R}^d$ is the vector representation of entity e, σ is the sigmoid function, and $\mathbf{W}_h \in \mathbb{R}^{h \times d}$ and $\mathbf{W}_o \in \mathbb{R}^{|T| \times h}$ are the weight matrices corresponding to the hidden layer and the output layer, respectively. The model was trained to minimize the binary cross-entropy loss summed over all entities and types:

$$-\sum_e \sum_t \big(y_{e,t} \log p_{e,t} + (1 - y_{e,t}) \log(1 - p_{e,t})\big), \tag{6}$$

where $y_{e,t} \in \{0, 1\}$ and $p_{e,t}$ denote the ground-truth label and predicted probability, respectively, of entity e being type t. The parameters in $\mathbf{W}_h$ and $\mathbf{W}_o$ are updated in the training stage. Note that the model described here is equivalent to that proposed in Yaghoobzadeh and Schutze (2015).

The model was trained using mini-batch SGD, with the learning rate controlled by Adam (Kingma and Ba, 2014) and the mini-batch size set to 32. The model was trained using the training set and evaluated using the test set. Following Yaghoobzadeh and Schutze (2015), the number of hidden units was set to 200. We also measured P@1 on the development set to locate the best epoch for testing.

Baselines

The performance of our models is compared with that of the following three entity representation models.

- **Figment-GM** (Yaghoobzadeh and Schutze, 2015) is based on the skip-gram model (Mikolov et al., 2013a; Mikolov et al., 2013b) trained using a large corpus with automatically generated entity annotations (i.e., FACC1 (Gabrilovich et al., 2013)). In this experiment, we used the entity representations publicized by the authors[9].

- **Skip-Gram-Wiki** is equivalent to Figment-GM, except that Wikipedia is used as the entity-annotated corpus. This model is also the same as our pre-trained representations described in Section 2.3.

- **Wikipedia2Vec** (Yamada et al., 2016) extends the skip-gram model to learn entity representations based on the contextual words of link anchors in Wikipedia and the internal link structure

[8] BEP is the F1 score at the point in the ranked list at which the precision and recall have the same value.

[9] `https://github.com/yyaghoobzadeh/figment`

	P@1	BEP	Acc.	Mic.	Mac.
TextEnt-full	**.932**	**.948**	**.626**	**.857**	**.842**
TextEnt-word	.909	.933	.611	.838	.820
TextEnt-entity	.882	.912	.560	.702	.770
Figment-GM	.813	.858	.421	.719	.683
Wikipedia2Vec	.925	.943	.600	.844	.822
Wikipedia2Vec (all)	.897	.917	.554	.798	.787
Skip-Gram-Wiki	.900	.927	.576	.831	.804
Skip-Gram-Wiki (all)	.852	.881	.510	.764	.740

Table 1: Results of the entity typing task.

of Wikipedia entities. We used the entity representations trained using the code publicized by the authors[10] and the Wikipedia dump used to train the Skip-Gram-Wiki model.[11]

We used the entity typing method presented above with the entity representations of each baseline model as inputs. Note that, because the Wikipedia2Vec and Skip-Gram-Wiki models were trained using the link anchors in Wikipedia, they do not contain entities that do not appear or are very rare as the link anchor destinations in Wikipedia. To address this, we evaluated these models in the following two settings: (1) using only the entities that exist in the model, and (2) using all entities, including non-existent ones. In the latter setting, we used the zero vector as the representation of non-existent entities. Similar to the latter setting, the former is not a fair comparison because it is typically more difficult to learn good entity representations of rare entities than those of popular entities (Yaghoobzadeh and Schutze, 2015).

Results

Table 1 compares the results of our models with those of the baseline models. Our TextEnt-full model outperforms the baseline models in all measures. In particular, the TextEnt-full model achieves a strong P@1 score of 93.2%, which clearly shows the effectiveness of our entity typing model for many downstream NLP tasks. Moreover, the TextEnt-full model generally performs better than both the TextEnt-word and TextEnt-entity models. This demonstrates the effectiveness of combining the semantic signals obtained from words and entities.

3.2 Multiclass Text Classification

This section describes the multiclass text classification task, which tests the ability of our proposed representations to encode arbitrary documents. Our key assumption here is that, because our proposed representations are trained to predict the corresponding entity of a given document in the KB, they can also classify non-KB documents into classes that are much more *coarse-grained* than entities.

Setup

Following Jin et al. (2016), we used two standard text classification datasets: the 20 newsgroups dataset[12] (denoted by 20NG) (Lang, 1995) and the R8 dataset (Debole and Sebastiani, 2005). The 20NG dataset consists of 11,314 training documents and 7,532 test documents retrieved from 20 different newsgroups. The documents are partitioned nearly equally across the classes. The R8 dataset contains documents from the eight most frequent classes of the Reuters-21578 corpus (Lewis, 1992), which consists of labeled news articles from the 1987 Reuters newswire. The R8 dataset contains 5,485 documents for training and 2,189 documents for testing. Unlike the 20NG dataset, the R8 dataset is imbalanced; the largest class contains 3,923 documents and the smallest class contains 51 documents. For both datasets, we report the

[10] `https://github.com/wikipedia2vec/wikipedia2vec`

[11] We trained the representations with $dim\_size = 300$, $window = 10$, $negative = 15$, $min\_entity\_count = 3$, and $iteration = 5$. Default values were used for other parameters.

[12] We used the *by-date* version of the dataset obtained from `http://qwone.com/~jason/20Newsgroups/`.

accuracy and macro-average F1 score. Furthermore, the development set was formed by selecting 10% of the documents in the training set at random for both datasets.

As preprocessing, we lowercased all words and removed words and entities appearing fewer than five times. Furthermore, we automatically annotated entity mentions in the documents using an entity linking system. In particular, we used TAGME[13] (Ferragina and Scaiella, 2010), a state-of-the-art entity linking system that is freely available and has been frequently used in recent studies (Xiong et al., 2016; Hasibi et al., 2016). However, TAGME returned many irrelevant entity mentions that would act as noise (e.g., *I like* refers to an entity *I Like (Keri Hilson song)*). Thus, we excluded mentions having relevance scores[14] of less than 0.05[15].

Method

For this task, we simply stacked a logistic regression layer onto our TextEnt model to classify documents into the predefined classes. First, we encoded each document (words with entity annotations) and used the resulting document representation (i.e., $\mathbf{v}_D$ in the TextEnt-full model, $\mathbf{v}_{D_w}$ in the TextEnt-word model, and $\mathbf{v}_{D_e}$ in the TextEnt-entity model) as the feature of the logistic regression classifier.

We trained the classifier using the training set of each dataset, and evaluated the classification performance using the corresponding test set. The classifier was trained using mini-batch SGD, with the learning rate controlled by Adam (Kingma and Ba, 2014) and the mini-batch size set to 32. The accuracy on the development set of each dataset was used to locate the best epoch for testing.

Baselines

We adopted the following state-of-the-art models as our baselines.

- **BoW-SVM** is based on a linear support vector machine (SVM) classifier with bag-of-words (BoW) features as inputs. This model outperforms the conventional naive Bayes model (Jin et al., 2016).

- **BoE** (Jin et al., 2016) is an extension of the skip-gram model that learns different word representations per target class. A linear model based on learned word representations was used to classify documents. This model achieves state-of-the-art results on both the 20NG and R8 datasets.

We also used the **Wikipedia2Vec** and **Skip-Gram-Wiki** models described in Section 3.1 as baselines. For this experiment, we simply input the representations of words and entities in these models to our text classification model described in the previous section.

Results

Table 2 compares the results of our proposed models with those of the baseline models. We obtained the BoW-SVM and BoE results from Jin et al. (2016). Our TextEnt-full model outperforms the state-of-the-art models in terms of accuracy and macro F1 score on both the 20NG and R8 datasets. Furthermore, similar to the results of our previous experiment, the TextEnt-full model generally performs better than both the TextEnt-word and TextEnt-entity models. This shows that combining semantic signals obtained from words and entities is also beneficial for text classification tasks.

Furthermore, we conducted a detailed comparison of the BoW-SVM model, BoE model, and TextEnt-full model using the class-level F1 scores on the 20NG dataset (Table 3) and the R8 dataset (Table 4). On the 20NG dataset, our model achieves the best scores in more than half of the classes and provides comparable performance in the other classes. Moreover, our model achieves strong performance in classes with relatively few documents on the R8 dataset. This is because our model successfully captures the strong semantic signals that can only be obtained from entities.

	20NG		R8	
	Acc.	F1	Acc.	F1
TextEnt-full	**.845**	**.839**	**.967**	**.910**
TextEnt-word	.836	.828	.965	.860
TextEnt-entity	.831	.824	.957	.878
BoW-SVM	.790	.783	.947	.851
BoE	.831	.827	.965	.886
Wikipedia2Vec	.829	.823	.965	.881
Skip-Gram-Wiki	.822	.815	.963	.879

Table 2: Results of the text classification task.

Class	SVM	BoE	TextEnt
alt.atheism	.699	.712	**.783**
comp.graphics	.702	.724	**.773**
comp.os.ms-windows.misc	.714	.724	**.742**
comp.sys.ibm.pc.hardware	.673	.706	**.721**
comp.sys.mac.hardware	.778	.792	**.840**
comp.windows.x	.779	**.853**	.846
misc.forsale	.846	**.852**	.829
rec.autos	.817	**.910**	.909
rec.motorcycles	.900	.942	**.943**
rec.sport.baseball	.895	**.947**	.941
rec.sport.hockey	.935	**.967**	.960
sci.crypt	.890	.926	**.934**
sci.electronics	.721	.737	**.757**
sci.med	.803	.869	**.891**
sci.space	.892	.885	**.900**
soc.religion.christian	.823	.877	**.904**
talk.politics.guns	.781	**.833**	.810
talk.politics.mideast	.837	.920	**.944**
talk.politics.misc	**.699**	.687	.678
talk.religion.misc	.590	**.676**	.672

Table 3: Class-level F1 scores in each class on the 20NG dataset.

4 Qualitative Analysis

To investigate how our model encodes documents and entities into the same continuous vector space, we extracted five example sentences from the 20NG dataset and encoded each sentence into a vector using our model. The closest entities to this vector based on the cosine similarity are presented in Table 5. We automatically annotated the entity mentions using TAGME[16], and fed the words and detected entities into the TextEnt-full model. Table 5 presents the sentences, nearest entities, and their corresponding classes in the 20NG dataset. Our model successfully encodes the sentences into vectors that are close to their relevant entities. For example, all nearest entities of the first sentence "*At one time there was speculation that the first spacewalk (Alexei Leonov?) was a staged fake*" are strongly related to the historic Soviet space program. Similar results can be observed in the other four examples.

[13]We used the public Web API service available at `https://services.d4science.org/`.

[14]We used the ρ scores assigned by TAGME.

[15]Excluding entity mentions using the relevance scores is the recommended practice described in the documentation: `https://services.d4science.org/web/tagme/documentation`

[16]We used the same configuration as described in Section 3.2.

Class	Count	SVM	BoE	TextEnt
grain	51	.824	.818	**.889**
ship	144	.781	.783	**.829**
interest	271	.745	.832	**.873**
money-fx	293	.687	.853	**.876**
trade	326	.897	.879	**.918**
crude	374	.929	**.958**	.929
acq	2,292	.956	**.978**	.977
earn	3,923	.986	**.990**	.988

Table 4: Class-level F1 scores with the number of documents in each class on the R8 dataset.

Class	Sentence	Nearest entities
sci.space	At one time there was speculation that the first spacewalk (Alexei Leonov?) was a staged fake.	Sputnik 1 (0.39), Soviet space program (0.38), Soyuz 5 (0.38), Vostok 1 (0.37)
rec.autos	I prefer a manual to an automatic as it should be.	Manual transmission (0.45), Automatic transmission (0.45), Dual-clutch transmission (0.43), Semi-automatic transmission (0.41)
sci.crypt	I change login passwords every couple of months.	Password (0.49), Login (0.46), Privilege (computing) (0.44), Privilege escalation (0.43)
soc.religion.christian	Which version of the Bible do you consider to be the most accurate translation?	Bible translations (0.37), King James Only movement (0.37), Biblical poetry (0.36), The Living Bible (0.36)
sci.med	The blood tests have shown that I have a little too much Hemoglobin	Blood (0.38), Introduction to genetics (0.38), Hemoglobin (0.37), Blood transfusion (0.35)

Table 5: Five example sentences with their top nearest entities using the TextEnt model.

5 Related Work

In recent years, various models for computing distributed representations of text (e.g., sentences and documents) have been proposed (Le and Mikolov, 2014; Kiros et al., 2015; Wieting et al., 2016; Hill et al., 2016). These models typically use large, unstructured corpora for training; however, certain models attempt to learn text representations from structured data. For instance, Hill et al. (2016) proposed a neural network model that learns text representations from online public dictionaries by predicting each dictionary word from its description. Further, Wieting et al. (2016) used a large set of paraphrase pairs obtained from the Paraphrase Database (Ganitkevitch et al., 2013) to learn text representations.

A number of recent models have attempted to learn distributed representations of entities from a KB. For example, Hu et al. (2015) extended the skip-gram model (Mikolov et al., 2013a) to learn entity representations using the hierarchical structure of the KB, and Li et al. (2016) modified the model by Hu et al. to learn both the category representations and entity representations using the category information of the KB. Additionally, *relational embedding* models (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015) learn the entity representations for link prediction tasks.

Furthermore, some models learn the representations of both words and entities from the KB. A simple method reported in the literature (Yaghoobzadeh and Schutze, 2015; Yamada et al., 2017) is used to derive the pre-trained representations in this study (i.e., preprocessing an entity-annotated corpus by replacing the name of each annotation with the unique identifier of the entity and feeding the corpus into a word embedding model (e.g., skip-gram)). Yamada et al. (2016) proposed Wikipedia2Vec, which extends this idea by using neighboring entities in the internal link graph of the KB as additional contexts for training the model. Note that we used Wikipedia2Vec as a baseline method in the two experiments

conducted in this study. Similarly, in their subsequent work (Yamada et al., 2017), they proposed a neural network model that takes entity-annotated text as input and learns word and entity representations by predicting the annotated entities contained in each text. Furthermore, Mancini et al. (2017) proposed a model that maps words and entities in a lexical dictionary (i.e., BabelNet (Navigli and Ponzetto, 2012)) to a single vector space by extending the CBOW model. Unlike our proposed model, these models require users to design a composition function (e.g., vector averaging) to model the semantics of a document using words and entities in it. Moreover, we showed that our approach is highly effective for the two important tasks of fine-grained entity typing and multiclass text classification.

6 Conclusions

In this paper, we described *TextEnt*, a simple neural network model that learns distributed representations of entities and documents from large-scale KB descriptions. We evaluated the performance of the proposed model on fine-grained entity typing and text classification tasks, and achieved state-of-the-art results in both cases, which clearly demonstrates the effectiveness of our approach. In future work, we will explore the applicability of our model to broader NLP tasks such as entity search and KB-based question answering.

Acknowledgements

We would like to thank the anonymous reviewers for their careful reading of our manuscript and their helpful comments and suggestions.

References

[Bordes et al.2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26*, pages 2787–2795.

[Debole and Sebastiani2005] Franca Debole and Fabrizio Sebastiani. 2005. An Analysis of the Relative Hardness of Reuters-21578 Subsets: Research Articles. *Journal of the American Society for Information Science and Technology*, 56(6):584–596.

[Ferragina and Scaiella2010] Paolo Ferragina and Ugo Scaiella. 2010. TAGME: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities). In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1625–1628.

[Gabrilovich et al.2013] Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase Annotation of ClueWeb Corpora, Version 1 (Release date 2013-06-26, Format version 1, Correction level 0).

[Ganitkevitch et al.2013] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764.

[Hajishirzi et al.2013] Hannaneh Hajishirzi, Leila Zilles, Daniel S Weld, and Luke Zettlemoyer. 2013. Joint Coreference Resolution and Named-Entity Linking with Multi-Pass Sieves. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 289–299.

[Hasibi et al.2016] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2016. Exploiting Entity Linking in Queries for Entity Retrieval. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, pages 209–218.

[Hill et al.2016] Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. 2016. Learning to Understand Phrases by Embedding the Dictionary. *Transactions of the Association for Computational Linguistics*, 4:17–30.

[Hu et al.2015] Zhiting Hu, Poyao Huang, Yuntian Deng, Yingkai Gao, and Eric Xing. 2015. Entity Hierarchy Embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1292–1300.

[Iyyer et al.2015] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691.

[Jin et al.2016] Peng Jin, Yue Zhang, Xingyuan Chen, and Yunqing Xia. 2016. Bag-of-embeddings for Text Classification. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2824–2830.

[Kingma and Ba2014] Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980v9*.

[Kiros et al.2015] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-Thought Vectors. In *Advances in Neural Information Processing Systems 28*, pages 3294–3302.

[Lang1995] Ken Lang. 1995. NewsWeeder: Learning to Filter Netnews. *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339.

[Le and Mikolov2014] Quoc V Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the 31th International Conference on Machine Learning*, volume 32, pages 1188–1196.

[Lewis1992] David D. Lewis. 1992. An Evaluation of Phrasal and Clustered Representations on a Text Categorization Task. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 37–50.

[Li et al.2016] Yuezhang Li, Ronghuo Zheng, Tian Tian, Zhiting Hu, Rahul Iyer, and Katia Sycara. 2016. Joint Embedding of Hierarchical Categories and Entities for Concept Categorization and Dataless Classification. In *Proceedings of the 26th International Conference on Computational Linguistics*, pages 2678–2688.

[Lin et al.2015] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2181–2187.

[Ling and Weld2012] Xiao Ling and Daniel S. Weld. 2012. Fine-Grained Entity Recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 94–100.

[Ling et al.2015] Xiao Ling, Sameer Singh, and Daniel S. Weld. 2015. Design Challenges for Entity Linking. *Transactions of the Association for Computational Linguistics*, 3:315–328.

[Liu et al.2015] Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical Word Embeddings. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2418–2424.

[Mancini et al.2017] Massimiliano Mancini, Jose Camacho-Collados, Ignacio Iacobacci, and Roberto Navigli. 2017. Embedding Words and Senses Together via Joint Knowledge-Enhanced Training. In *Proceedings of the 21st Conference on Computational Natural Language Learning*, pages 100–111.

[Mikolov et al.2013a] Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the 2013 International Conference on Learning Representations*, pages 1–12.

[Mikolov et al.2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

[Navigli and Ponzetto2012] Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. *Artificial Intelligence*, 193(Supplement C):217–250.

[Neelakantan and Chang2015] Arvind Neelakantan and Ming-Wei Chang. 2015. Inferring Missing Entity Type Instances for Knowledge Base Completion: New Dataset and Methods. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 515–525.

[Srivastava et al.2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

[Wang et al.2014] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph and Text Jointly Embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1591–1601.

[Wieting et al.2016] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards Universal Paraphrastic Sentence Embeddings. *Proceedings of the 2016 International Conference on Learning Representations*.

[Xiong et al.2016] Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. 2016. Bag-of-Entities Representation for Ranking. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, pages 181–184.

[Yaghoobzadeh and Schutze2015] Yadollah Yaghoobzadeh and Hinrich Schutze. 2015. Corpus-level Fine-grained Entity Typing Using Contextual Information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 715–725.

[Yaghoobzadeh and Schütze2017] Yadollah Yaghoobzadeh and Hinrich Schütze. 2017. Multi-level Representations for Fine-Grained Typing of Knowledge Base Entities. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 578–589.

[Yamada et al.2016] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 250–259.

[Yamada et al.2017] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2017. Learning Distributed Representations of Texts and Entities from Knowledge Base. *Transactions of the Association for Computational Linguistics*, 5:397–411.

[Zeiler2012] Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701v1*.

Simple Neologism Based Domain Independent Models to Predict Year of Authorship

Vivek Kulkarni
Department of Computer Science
University of California, Santa Barbara
vvkulkarni@cs.ucsb.edu

Yingtao Tian
Department of Computer Science
Stony Brook University
yittian@cs.stonybrook.edu

Parth Dandiwala
Department of Computer Science
Stony Brook University
pdandiwala@cs.stonybrook.edu

Steven Skiena
Department of Computer Science
Stony Brook University
skiena@cs.stonybrook.edu

Abstract

We present domain independent models to date documents based only on neologism usage patterns. Our models capture patterns of neologism usage over time to date texts, provide insights into temporal locality of word usage over a span of 150 years, and generalize to various domains like News, Fiction, and Non-Fiction with competitive performance. Quite intriguingly, we show that by modeling only the distribution of usage counts over neologisms (the model being *agnostic* of the particular words themselves), we achieve competitive performance using several orders of magnitude fewer features (only 200 input features) compared to state of the art models some of which use 200K features.

1 Introduction

Determining when a document is written is an important task in historical linguistics and temporal information retrieval. For instance, several works attempt to date historical biblical texts like the *The Book of Isiah* (Rooker, 1996; Ehrensvärd, 1997; Hurvitz, 2000; Young and Rezetko, 2016) or ancient texts like *Beowoulf* (Chase, 1997). Likewise, in the field of information retrieval, establishing the dates of documents is an important pre-requisite to returning temporally relevant documents and provides important information for a large number of search tasks (Ostroumova Prokhorenkova et al., 2016; Efron, 2013).

Most efforts to automatically date texts adopt a learning based approach and rely on several linguistic features that are time-relevant (Garcia-Fernandez et al., 2011; Jatowt and Tanaka, 2012; Zampieri et al., 2015; Ostroumova Prokhorenkova et al., 2016). Such time-relevant features include neologisms/archaisms, political events, spelling variations, and the presence of named entities as well as external knowledge bases. In addition to the large input feature dimensionality (of the order of the input vocabulary), these approaches all focus on particular domains (for example, primarily News articles). While such feature rich models which are arguably domain specific perform very well, these features are rarely generalizable across domains (for example. models that use political events can perform very well when evaluated in the News domain, but do not necessarily generalize to Fiction). Developing models that generalize across domains without further fine tuning generally entails modeling linguistic cues that are stable across domains.[1]

Consequently, in this work, we propose the first set of models that effectively generalize across domains by leveraging insights into the temporal usage of language. Our models rely on a key insight which we term the **temporal locality of neologisms**: *Documents written at time t tend to use neologisms invented shortly before t.* By effectively modeling this usage of neologisms (as a feature class cumulatively) we propose models that not only reduce our input feature dimensionality by at-least an order of magnitude, but also effectively generalize across a variety of domains (Fiction, Non-Fiction, and News)

[1] It is worth noting that ultimately most domain adaptation techniques attempt to find such a general feature set by either attempting to map source domain features to the target domain or learning a more abstract feature set.

Proceedings of the 27th International Conference on Computational Linguistics, pages 202–212
Santa Fe, New Mexico, USA, August 20-26, 2018.

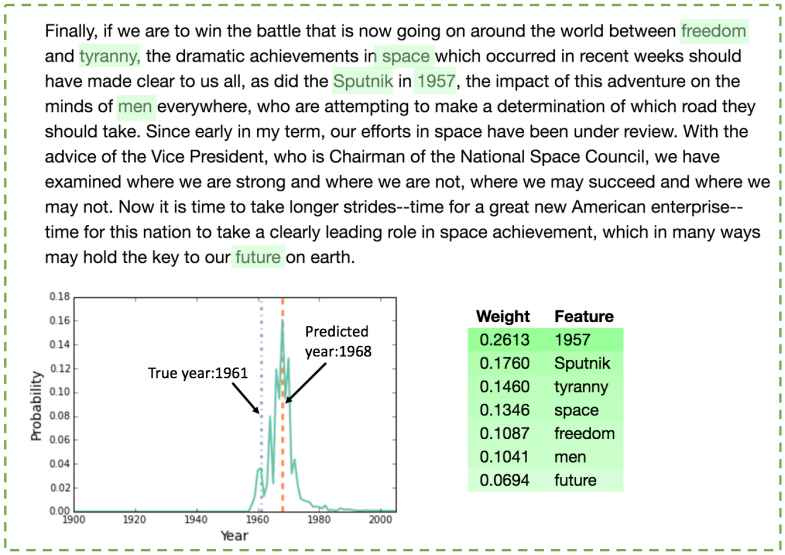


Figure 1: Sample output predictions of a Naïve Bayes model (NB) on a portion of a speech given by President John F.Kennedy in 1961. Note that this model outputs a probability distribution over years with a MAP estimate of year 1968. Note also that the model was trained *only using* Google Book Ngrams and not on the domain it is evaluated on. Finally, observe that words like `1957, Sputnik, tyranny and space` were most influential in this prediction thus providing insight into linguistic patterns the model has captured only from Google Book Ngrams. Moreover, the model is generic and can be applied to multiple domains like Fiction, News or Non Fiction. This motivates our hypothesis that neologisms can be effectively used to date documents across multiple domains.

without further tuning and perform competitively with more complex models that capture fine-grained linguistic cues. Intriguingly, we demonstrate that neologism-based models that use only ~ 200 features achieve a performance within 5 units of mean absolute error (21.58 on NonFiction) over the best Naïve Bayes model (18.25 on NonFiction) which uses more than 200K features.

In a nutshell, we make the following contributions:

- We propose domain independent models for the task of dating documents. We emphasize that our goal is not to necessarily outperform the state of the art domain specific models, but to demonstrate the effectiveness of simple models drawing on linguistic insights that generalize across domains.

- We leverage cumulative usage patterns of neologisms over time to propose the first set of simple generalizable models for this task while revealing insights into the cumulative usage patterns of neologisms over time.

- We empirically evaluate our models against several competing methods including neural models like LSTMs on three different domains (News, Fiction and Non Fiction).

2 Datasets

Here, we describe data-sets which we use for learning and evaluating our models.

Training Dataset We train all of our models using only the Google Book Ngrams dataset spanning the time $1850 - 2005$. Since the Google Book Ngrams spans multiple domains, we can capture domain agnostic linguistic cues to learn generalizable models for our task. Since our neologism models need only the frequency of occurrences, the Google Book Ngrams is an ideal dataset providing not only a large sample size for robust parameter estimation but also inherently spanning multiple domains.

Evaluation Datasets To evaluate our models, we consider the following datasets[2]:

- **NYTimes**: We consider a random sample of 10000 leading paragraphs of NEW YORK TIMES articles from the range $1850 - 2005$ constructed by scraping the New York Times website. Note that this dataset is primarily from the NEWS domain.
- **Corpus of Historical American English (COHA)**: We consider a random sample of 10000 articles from each genre, namely FICTION, NONFICTION, and NEWS from the COHA corpus (Davies, 2002). The COHA corpus is an ideal dataset to evaluate our models since it spans a wide time range, with multiple domains where the dates have been validated by human experts and is easily available for research purposes[3].

We emphasize that all of our models only use the Google Book Ngrams data to learn parameters. The models are then evaluated on the evaluation datasets which span multiple domains without any further fine-tuning.

3 Baselines

Before we describe our proposed models, we introduce two baseline methods to evaluate against on our task.

BOOKPROP We estimate the probability of a document written in a given year y, by computing the fraction of books written in year y over all books written in the time period under consideration. Formally, we estimate the following probability:

$$\mathsf{P}(Y = y) = \frac{\#(\text{books}, y)}{\sum_y \#(\text{books}, y)}$$

We estimate the number of books in English written in year y, as the number of distinct books the word `the` was mentioned in a given year y as per Google Book Ngrams (Michel and others, 2011) data. As expected, the distribution is skewed towards the right with more books written in the 20^{th} century than in the 18^{th} century (see Figure 2). Given a document to date, a random sample drawn from this distribution is then taken as the predicted estimate of the date of the document. A limitation of BOOKPROP is that it does not model language.

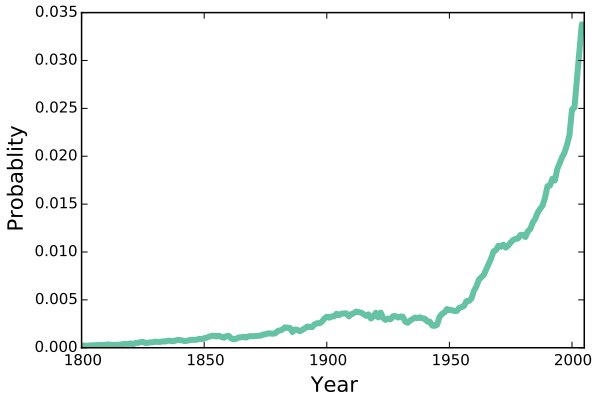


Figure 2: Estimate of the probability of English books being from a given year using the Google Book Ngrams data. As expected, the distribution is skewed towards the right with more books written in the 20^{th} century than in the 18^{th} century.

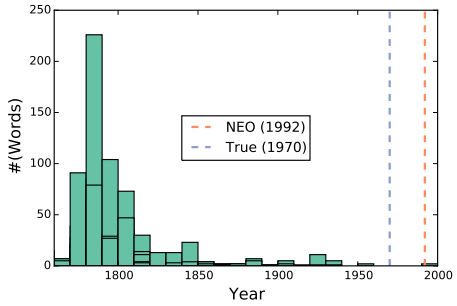
[2]While there has been work on dating documents (Zampieri et al., 2016; Kumar et al., 2012; Graliński et al., 2017), there are no standard publicly available evaluation datasets in the community for this task especially spanning multiple domains in English.

[3]We note that another option to construct such a dataset spanning multiple domains would be to use books from the HathiTrust. However, accessing a large clean dataset required institutional access unavailable to us at this point in time.

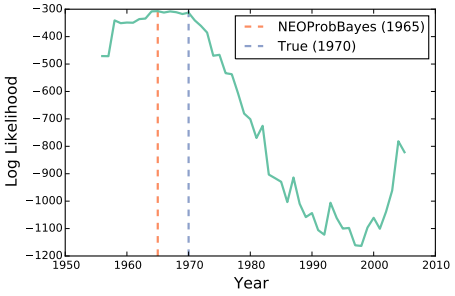
WORD	Estimated Year	Actual First Usage
HIV	1987	1986
Hitler	1933	1934
LSD	1955	1950
Obama	2007	2006
SARS	2003	2003
⋮	⋮	⋮

WORD	Estimated Year	Actual First Usage
Sputnik	1958	1957
electron	1905	1891
radio	1904	1907
television	1931	1907
transistor	1950	1948
walkman	1993	1979

Table 1: Example cases of estimated year of popular usage (MR) and actual year of first use (FU) obtained from `http://www.etymonline.com/` for different words from Google Book Ngrams data. Note that in the majority of these words, our estimated year is close to the year of first usage and shows a small lag from the year of first use as expected.



(a) NEO



(b) NEOProbBayes

Figure 3: Figure illustrating NEO and NEOPROBBAYES on the same document which was written in 1970. NEO is easily misled by outlier words and predicts the date to be 1992 ignoring other evidence like counts of other words. NEOPROBBAYES, in contrast incorporates all observed evidence to estimate more accurately that the document was written in 1965.

NEO A more sophisticated approach to assigning dates to documents is based on the following observation: If we observe a word which first came into popular usage in a year y, then the document is very likely written after year y. A simple model based on this hypothesis is to output the year of the most recent word found in the document. For example, in Figure 1, NEO estimates the date of a document to be 1958, since it is clear that `Sputnik`, the most recent word used in the document, sprung into popular use in 1958.

We estimate the year in which a word came into popular usage MR(w) from Google Book Ngrams as follows: (a) Compute the cumulative usage of a word w through every year in the Google Book Ngrams. (b) Compute the first year in which the cumulative usage of the word w exceeds a small fraction α (empirically set to $1/250.0$) of the total cumulative usage. As an example, our method estimates MR(`Obama`) $= 2007$ while the year of actual first usage is 2006. We validate our approach on a small set of manually curated words which we show in Table 1.

Observe that for the majority of these words, our estimated year is close to the year of first usage and shows a small lag, which is expected since we seek to estimate the year in which the word came into popular usage and not its year of first usage.

While NEO serves as a strong baseline, there are two limitations of this method: (a) The document could be written long after the time period corresponding to the most recent word observed in the document. (b) It ignores evidence of other words seen in the document and bases its decision on the occurrence of a single word.

Figure 3a illustrates these drawbacks. First, note that only one word `vinaya` with an MR(w) in the 1990's was observed in this document. NEO is easily misled by this single erroneous estimation of MR(`vinaya`) and estimates the date of this document to be 1992. It ignores other evidence that suggests that the document was written after 1940, but is unlikely to be written in the 1990's since words

with $\mathrm{MR}(w)$ a few decades before 1990's are not observed at all.

4 Proposed Neologism Based Model

We now describe a probabilistic model that effectively uses new words incorporated into popular usage to estimate when the document was written. In particular, our model computes the likelihood of observing a set of words that came into popular usage after year x given the document was written in year y to estimate when the document was written. Our method has two key steps:

1. **Ensemble Model Construction**: We construct an ensemble of probabilistic models where model M_i outputs $\mathsf{P}(y|X_i)$ and X_i is a discrete random variable counting the words observed in a document which came into popular usage after year i.
2. **Combining Ensemble Predictions**: Each model M_i outputs $\mathsf{P}(y|X_i)$, so we investigate multiple methods to combine predictions from individual models.

Ensemble Model Construction Let $\mathsf{F}(o, n)$ be the probability of observing a word that came into popular usage after year o in year n, where $n > o$. For every year pair (o, n), we estimate $\mathsf{F}(o, n)$ from the Google Books Ngrams Corpus by computing the fraction of words with $\mathrm{MR}(w) > o$ in the Google Book Ngrams of year n.

Given a text T of length N, let $N'(i)$ denote a realization of X_i in T. Each model M_i models the probability of T written in year y based on X_i as follows:

$$\mathsf{P}(y|X_i) \propto \begin{cases} \mathsf{P}(X_i|y)\mathsf{P}(y), & \text{if } i < y \\ 0, & \text{otherwise} \end{cases}$$

$\mathsf{P}(X_i|y)$ follows a binomial distribution with success probability $\mathsf{F}(i, y)$ which can be computed knowing the length of the document N and $N'(i)$ a realization of X_i. We assume the prior $\mathsf{P}(y)$ to be uniform.

Combining Ensemble Predictions Each model M_i computes a probability distribution over years, namely $\mathsf{P}(y|X_i)$. We now describe three methods to combine these individual model predictions to output a final prediction:

1. **NEOProbMean**: We output the mean of individual MAP predictions as the predicted year of authorship.
2. **NEOProbMedian**: We output the median of individual MAP predictions as the predicted year of authorship.
3. **NEOProbBayes**: We use a Bayesian scheme to incorporate all the observed evidence as follows: Let $\boldsymbol{X} = \{X_i \text{ for each year } i\}$. Specifically we compute the following:

$$\begin{aligned} \mathsf{P}(y|\boldsymbol{X}) &\propto \mathsf{P}(\boldsymbol{X}|y)\mathsf{P}(y) \\ &= \left(\prod_i \mathsf{P}(X_i|y)\right)\mathsf{P}(y) \end{aligned}$$

   where we make the *Naïve Bayes assumption* that each X_i is independent of any other X_j, when conditioned on the year y. We output the MAP estimate of $\mathsf{P}(y|\boldsymbol{X})$ as the final prediction.

Figure 3b shows this approach for a document and also contrasts it with the baseline NEO. Observe how NEOPROBBAYES enables a more accurate prediction by incorporating observed evidence ignored by NEO.

As we will show empirically, performance of each of the above methods is a function of the length of the document it is evaluated on since the accuracy of the individual probability estimates depends on the length. [4] While NEOPROBBAYES is better for large documents, it is quite sensitive to errors in estimates for small documents. Therefore, NEOPROBMEDIAN and NEOPROBMEAN which are less sensitive in the presence of outliers are better than NEOPROBBAYES for short documents (100 tokens) but are outperformed by NEOPROBBAYES for larger documents (2000 tokens) when individual model estimates are much more accurate.

[4] The larger the length, the more accurate the individual estimates.

5 Comparison to more Feature Rich Models

To place our models in the context of prior work which use a large set of linguistic features, we consider two linguistically feature rich models. Specifically, we consider a bag-of-words based Naïve Bayes' model as well as a neural network based model NEURALDATE for this task.

5.1 NAÏVEBAYES

We consider a simple, standard Multinomial Naïve Bayes classifier learned using Google Book Ngrams to date the year a document was written. We use unigram bag-of-words (we restrict our vocabulary to 200K tokens and discard out-of-vocabulary words) features and Laplace smoothing. It is worth noting that Naïve Bayes uses 200K features which is orders of magnitude higher than **NEO-Prob** approaches.

5.2 NEURALDATE

We propose a neural model NEURALDATE, to date texts. NEURALDATE operates on short sequences of words (n-grams), and outputs a probability distribution over years, $\mathsf{P}(y|\boldsymbol{x}_i)$ for each ngram $\boldsymbol{x}_i$ in the document $\boldsymbol{D}$,

Our model consists of a bi-directional LSTM with an embedding layer, two hidden layers and an output layer[5]. The embedding layer maps the input (one hot encoding of the word) to a dense embedding of size 200 dimensions. The implementation of the LSTM hidden layers are as described in (Graves and others, 2012) and therefore not described in this paper. The output layer is a soft-max layer over the years within the time-range considered. We use ADAM optimizer (Kingma and Ba, 2014) with a learning rate of $\eta = 0.001$.[6]

In order to date a document $\boldsymbol{D}$, we use the model to compute $\mathsf{P}(y|\boldsymbol{x}_i)$ for each n-gram (we use n=5) in $\boldsymbol{D}$. We then compute $\mathsf{P}(y|\boldsymbol{D})$ to be the mean of these individual probability distributions. Finally, we use the MAP estimate of $\mathsf{P}(y|\boldsymbol{D})$ as our point estimate of the year.

Autocorrelation Regularizer The model described above does not explicitly leverage structure of the label space, namely temporal structure (linear sequential structure). Observe the high variance in probability scores around the mode in Figure 4. Therefore, for a given n-gram $\boldsymbol{x}_i$ it would be preferable to learn model parameters such that $\mathsf{P}(y|\boldsymbol{x}_i)$ is "smooth" around any given label. This captures the insight that classes (years) in the neighborhood of a label l should be assigned probabilities similar to that assigned to l.[7]

We can formalize this notion of smoothness as follows: Let p_l be the probability assigned to label l. Given a neighborhood k, let $\boldsymbol{d}$ be the vector of first order differences: $p_i - p_{i+1}$ for $i \in [l-k \cdots l+k]$. We define the distribution to be L-smooth at l around neighborhood k if $\omega(\boldsymbol{d}) = \frac{\sigma(\boldsymbol{d})}{\text{mean}(|\boldsymbol{d}|)} \leq L$, for some small constant $L > 0$, where smaller values of L indicate smoother distributions.

We therefore propose to add the following cost to the original cost function:

$$\Omega(\boldsymbol{\theta}; \boldsymbol{X}^{\text{train}}) = \sum_j \omega\left(\boldsymbol{d}_j\right) ,$$

where $\boldsymbol{d}_j$ are differences between predicted probabilities for neighboring years for example j.

In summary, the final loss function including this regularization is $J(\boldsymbol{\theta}; \boldsymbol{X}^{\text{train}}, \boldsymbol{y}^{\text{train}}) + \alpha\Omega(\boldsymbol{\theta}; \boldsymbol{X}^{\text{train}})$, where $J(\boldsymbol{\theta}; \boldsymbol{X}^{\text{train}}, \boldsymbol{y}^{\text{train}})$ is the standard cross-entropy loss and α is a hyper-parameter weighing the regularization.

Figure 4 shows the effect of incorporating label smoothness constraints in the cost function for a sample n-gram. Note that incorporating the temporal structure of labels in the cost function produces markedly smoother and realistic distributions than a model not exploiting label structure.To investigate

[5]While more sophisticated and complex sequence models are being developed even as we write this paper, our goal here is only to place the performance of our proposed neologism models in context of other sophisticated methods. Therefore, Bidirectional LSTMs serve as a good lower bound for complex models.

[6]Hyper-parameter settings were chosen based on a validation set.

[7]We initially also experimented with using mean square loss and modeling this as a regression problem, but the resulting model did not perform well empirically.

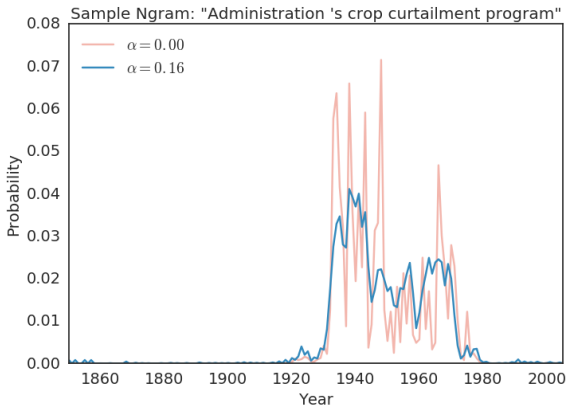


Figure 4: Predicted distribution over years for a given 5-gram (shown above the figure), motivating the need for the auto-correlation regularizer. Note that when $\alpha = 0$, the regularizer is disabled and the output probability distribution is very noisy and neighboring values have large variance. In contrast, when the regularizer is properly enabled ($\alpha = 0.16$), observe how the output probability distribution is much smoother and neighboring probability values are more similar.

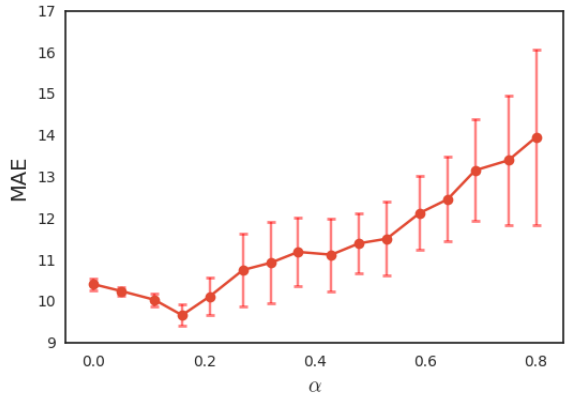


Figure 5: MAE from cross validation for candidates of α which controls the strength of the auto-correlation regularizer. The means and the standard deviations over 20 independent runs are shown. When the model is properly regularized ($\alpha = 0.16$), observe the improvements over model without regularization ($\alpha = 0$). Also, note that when the model is over-regularized ($\alpha > 0.25$), the performance is worse and demonstrates larger variance.

the effect of α, we measure the MAE (Mean Absolute Error) over n-grams and use cross-validation by selecting α from a set of candidates in $[0, 0.8]$ (see Figure 5). Based on these observations, we set α at 0.16 empirically for training our model.

6 Experiments

We evaluate all of our methods against several baselines on diverse data sets spanning multiple domains. We consider the time period of $1850 - 2005$ for the purpose of dating documents and evaluate our models on the evaluation data-sets described in Section 2. Since the tasks should get easier on long documents, we measure the performance of our models as a function of the length. Since the NYTIMES dataset only consists of the first paragraph of articles (about 100 tokens on average) we use the entire paragraph for evaluation on this dataset. Tables 2 and 3 show the Mean Absolute Error (MAE) over the NYTIMES and COHA datasets, from which we make the following observations:

- **Neologism Methods need relatively large documents to perform competitively**: Overall, the neologism based models perform competitively with Naïve Bayes using 1000 times fewer features (see

#(Tokens)	#(Features)	MAE
BOOKPROP	-	43.46
NEO	$\leq$ **200**	58.77
NEOPROBMEAN	$\leq$ **200**	27.55
NEOPROBMEDIAN	$\leq$ **200**	28.22
NEOPROBBAYES	$\leq$ **200**	67.14
NAÏVEBAYES	$\sim 200K$	23.69
NEURALDATE (w/o reg.)	1000	22.80
NEURALDATE	1000	**20.54**

Table 2: Mean Absolute Error on New York Times data. Note that neologism based methods (highlighted) whose feature set size is much smaller, perform competitively with NAÏVEBAYES with a feature space of dimension of 200K. NEURALDATE uses a 200 dimension embedding for each word in a 5-gram and so has an effective feature input size of 1000. It is worth noting that our neologism based models do not directly rely on the actual words themselves but on the number of neologisms used at a given time thus drastically reducing the feature size while yielding competitive performance.

Dataset	#(Tokens)	100	500	1000	2000
COHA-Fiction	BOOKPROP	44.57	44.54	43.95	44.21
	NEO	66.99	34.74	27.39	24.80
	NEOPROBMEAN	32.40	30.76	31.45	31.13
	NEOPROBMEDIAN	36.90	32.96	32.03	31.73
	NEOPROBBAYES	78.99	41.90	33.77	27.92
	NAÏVEBAYES	**26.61**	**23.98**	**22.62**	**21.93**
	NEURALDATE (w/o reg.)	37.56	30.71	28.96	27.97
	NEURALDATE	35.66	30.02	27.81	26.96
COHA-NonFiction	BOOKPROP	45.19	45.07	45.04	45.51
	NEO	57.99	30.75	24.84	22.86
	NEOPROBMEAN	31.13	26.90	26.02	25.39
	NEOPROBMEDIAN	30.68	26.60	25.73	25.13
	NEOPROBBAYES	56.58	30.73	25.46	21.58
	NAÏVEBAYES	**24.28**	**19.83**	**18.36**	**18.25**
	NEURALDATE (w/o reg.)	27.91	23.57	22.29	21.60
	NEURALDATE	25.21	20.07	20.38	20.09
COHA-News	BOOKPROP	44.97	45.34	44.99	45.02
	NEO	39.86	20.39	19.80	20.26
	NEOPROBMEAN	24.36	23.40	23.30	23.31
	NEOPROBMEDIAN	25.22	22.88	22.45	22.39
	NEOPROBBAYES	48.30	22.79	20.97	20.82
	NAÏVEBAYES	21.35	17.21	16.64	16.60
	NEURALDATE (w/o reg.)	20.40	16.04	15.43	15.34
	NEURALDATE	**19.30**	**15.33**	**14.72**	**14.59**

Table 3: Mean Absolute Error of different models on COHA datasets as a function of number of tokens used for evaluation in each document. Note that our proposed neologism based methods (highlighted) use a much smaller feature set, generalize across domains without any further fine-tuning, and perform competitively with feature rich models like NAÏVEBAYES and NEURALDATE for long documents (greater than 500 tokens).

Table 2) suggesting that effective usage of neologism usage patterns can serve as strong baselines.

Furthermore, from Table 3, the baseline NEO generally performs very poorly on short documents

(of length 100 tokens). For example, on the COHA-FICTION dataset using 100 tokens, the MAE is 66.99 compared to BOOKPROP which yields an MAE of 44.57. On short documents NEO is easily misled due to lack of effective sample size. In contrast, observe that as the length of the document increases NEO's error reduces significantly (note Table 3 that for 2000 word documents on COHA-FICTION the mean absolute error is now 24.80).

Finally, the probabilistic models we propose extending NEO also perform better than NEO especially on short documents (for example, on COHA-FICTION for documents with 100 tokens the MAE for NEOPROBMEAN is 32.40 compared to 66.99 for NEO). Similarly, NEOPROBMEAN and NEOPROBMEDIAN outperform NEOPROBBAYES on documents of up to 1000 tokens but NEOPROBBAYES almost always outperforms all of these on documents of length 2000, suggesting that NEOPROBBAYES needs a larger sample size to make effective predictions.

- **Deeper linguistic features boost performance**: We finally observe that including linguistic features like the words used in a simple Naïve Bayes classifier consistently outperforms methods relying solely on neologisms. Further, observe that the NEURALDATE with the auto-correlation regularizer demonstrates superior performance over NEURALDATE without regularization. Finally, note that NEURALDATE also performs competitively and sometimes out-performs Naïve Bayes.

Altogether, our proposed neologism based models generalize well across domains, reduce the input feature size significantly while performing competitively with more complex feature rich models.

7 Related Work

A large body of related work on the task of automatically dating texts exists in the field of temporal information retrieval (De Jong et al., 2005; Popescu and Strapparava, 2015; Kanhabua and Nørvåg, 2009; Garcia-Fernandez et al., 2011; Niculae et al., 2014; Zampieri et al., 2015; Zampieri et al., 2016; Bamman et al., 2017; Jatowt and others, 2017; Kumar et al., 2012; Kumar, 2013; Graliński et al., 2017). The community also has two shared tasks (Popescu and Strapparava, 2015) and (Graliński et al., 2017). However, both of these shared tasks differ from our setting. Popescu and Strapparava (2015) is a shared task for diachronic text evaluation but only focuses on the Newspaper domain in contrast to our work which focuses on generalizable models across domains. Graliński et al. (2017) is the most recent challenge on dating texts but it focuses on Polish texts. De Jong et al. (2005) proposed using temporal language models based on unigrams to date texts on Dutch newspaper articles. Several works incorporate additional features like lexical features, part-of-speech tagging, extraction of concepts and word sense disambiguation and use external knowledge bases (Kanhabua and Nørvåg, 2009; Garcia-Fernandez et al., 2011; Niculae et al., 2014; Zampieri et al., 2015; Zampieri et al., 2016). Recently Jatowt and others (2017) propose an interactive system to estimate the age of document using moment statistics of n-grams focusing on only a qualitative analysis.

Our work is most closely related to the works of (Kumar et al., 2012), (Garcia-Fernandez et al., 2011),(Zampieri et al., 2015) and (Bamman et al., 2017). Kumar et al. (2012) propose a model for predicting the dates of documents without explicit temporal cues. This model essentially learns temporal language models on the temporal corpus where explicit temporal expressions are removed. It then assesses the likelihood of a given document under each time point to make a prediction. It thus relies on implicit temporal cues and words and typically has an input feature dimensionality of the order of the vocabulary (in this case 300K words). Furthermore, this model has only been evaluated in different settings (like predicting the mid-point of an individual's lifetime using their Wikipedia biography). Garcia-Fernandez et al. (2011) develop a model to date documents using both chronological methods with external knowledge and classification methods like using an SVM to date documents on a French Newspaper corpus while Zampieri et al. (2015) propose a ranking based approach to temporal text classification. These methods learn models on the respective domains they are evaluated on. Bamman et al. (2017) proposed bag-of words based models (using Ridge Regression as well as Naïve Bayes) to predict the date of first publication over books obtained from the Hathi Trust.

Differing from these works, our proposed method seeks to learn a global model that can be applied across multiple domains without further tuning. We propose new probabilistic models to date texts by analyzing statistical patterns of the introduction of neologisms over time. Our models are simple, domain-independent, use several orders of magnitude fewer features and yet achieve competitive performance.

8 Conclusion

In this paper, we investigated the task of dating books on a large fine-grained time scale (spanning 150 years) through the lens of neologisms introduced over time. We propose probabilistic models that effectively analyze the usage of neologisms. We demonstrate that our methods perform competitively with models that use deeper linguistic cues (which could use a feature space of more than thousands of features). Furthermore, our models are learned using only the Google Book Ngrams, do not need any further tuning when evaluated on other domains and potentially enable researchers to obtain literary insights into the language of authors over time.

Acknowledgments

The authors thank the anonymous reviewers for their valuable feedback. This work was partially supported by NSF grants DBI-1355990 and IIS-1546113.

References

David Bamman, Michelle Carney, Jon Gillick, Cody Hennesy, and Vijitha Sridhar. 2017. Estimating the date of first publication in a large-scale digital library. In *ACM/IEEE Joint Conference on Digital Libraries (JCDL), 2017*, pages 1–10. IEEE.

Colin Chase. 1997. *The dating of Beowulf*. Number 6. University of Toronto Press.

Mark Davies. 2002. *The Corpus of Historical American English (COHA): 400 million words, 1810-2009*.

Franciska De Jong, Henning Rode, and Djoerd Hiemstra. 2005. Temporal language models for the disclosure of historical text.

Miles Efron. 2013. Query representation for cross-temporal information retrieval. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM.

Martin Ehrensvärd. 1997. Once again: The problem of dating biblical Hebrew. *Scandinavian Journal of the Old Testament*, 11(1).

Anne Garcia-Fernandez, Anne-Laure Ligozat, Marco Dinarelli, and Delphine Bernhard. 2011. When was it written? Automatically determining publication dates. In *International Symposium on String Processing and Information Retrieval*. Springer.

Filip Graliński, Rafał Jaworski, Łukasz Borchmann, and Piotr Wierzchoń. 2017. The RetroC challenge: how to guess the publication year of a text? In *Proceedings of the 2nd International Conference on Digital Access to Textual Cultural Heritage*, pages 29–34. ACM.

Alex Graves et al. 2012. *Supervised sequence labeling with recurrent neural networks*, volume 385. Springer.

Avi Hurvitz. 2000. Can biblical texts be dated linguistically? Chronological perspectives in the historical study of biblical hebrew. *Vetus Testamentum-Supplements 80*, 80.

Adam Jatowt et al. 2017. Interactive system for reasoning about document age. In *CIKM*. ACM.

Adam Jatowt and Katsumi Tanaka. 2012. Large scale analysis of changes in English vocabulary over recent time. In *CIKM*. ACM.

Nattiya Kanhabua and Kjetil Nørvåg. 2009. Using temporal language models for document dating. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Abhimanu Kumar, Jason Baldridge, Matthew Lease, and Joydeep Ghosh. 2012. Dating texts without explicit temporal cues. *arXiv preprint arXiv:1211.2290*.

Abhimanu Kumar. 2013. *Supervised language models for temporal resolution of text in absence of explicit temporal cues*. Ph.D. thesis.

Jean-Baptiste Michel et al. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014).

Vlad Niculae, Marcos Zampieri, Liviu P Dinu, and Alina Maria Ciobanu. 2014. Temporal text ranking and automatic dating of texts. In *EACL*.

Liudmila Ostroumova Prokhorenkova, Petr Prokhorenkov, Egor Samosvat, and Pavel Serdyukov. 2016. Publication date prediction through reverse engineering of the web. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM.

Octavian Popescu and Carlo Strapparava. 2015. Semeval-2015 Task 7: Diachronic Text Evaluation. *Proceedings of SemEval*.

Mark F Rooker. 1996. Dating Isaiah 40-66: What does the linguistic evidence say?

Ian Young and Robert Rezetko. 2016. *Linguistic dating of biblical texts*, volume 1. Routledge.

Marcos Zampieri, Alina Maria Ciobanu, Vlad Niculae, and Liviu P Dinu. 2015. Ambra: A ranking approach to temporal text classification. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015). Denver, CO, USA*.

Marcos Zampieri, Shervin Malmasi, and Mark Dras. 2016. Modeling language change in historical corpora: the case of Portuguese. *arXiv preprint arXiv:1610.00030*.

Neural Math Word Problem Solver with Reinforcement Learning

Danqing Huang[1]*, **Jing Liu**[2]*, **Chin-Yew Lin**[3], **and Jian Yin**[1]
{huangdq2@mail2,issjyin@mail}.sysu.edu.cn
liujing46@baidu.com
cyl@microsoft.com
[1] The School of Data and Computer Science, Sun Yat-sen University.
Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, P.R.China
[2]Baidu Inc. [3]Microsoft Research

Abstract

Sequence-to-sequence model has been applied to solve math word problems. The model takes math problem descriptions as input and generates equations as output. The advantage of sequence-to-sequence model requires no feature engineering and can generate equations that do not exist in training data. However, our experimental analysis reveals that this model suffers from two shortcomings: (1) generate spurious numbers; (2) generate numbers at wrong positions. In this paper, we propose incorporating copy and alignment mechanism to the sequence-to-sequence model (namely CASS) to address these shortcomings. To train our model, we apply reinforcement learning to directly optimize the solution accuracy. It overcomes the "train-test discrepancy" issue of maximum likelihood estimation, which uses the surrogate objective of maximizing equation likelihood during training while the evaluation metric is solution accuracy (non-differentiable) at test time. Furthermore, to explore the effectiveness of our neural model, we use our model output as a feature and incorporate it into the feature-based model. Experimental results show that (1) The copy and alignment mechanism is effective to address the two issues; (2) Reinforcement learning leads to better performance than maximum likelihood on this task; (3) Our neural model is complementary to the feature-based model and their combination significantly outperforms the state-of-the-art results.

1 Introduction

The task of math word problem solving aims to automatically solve a math problem by reading the text description of the problem and generating the answer. This task requires the machine to have the ability of natural language understanding and reasoning.

In the past years, most of the proposed methods heavily rely on predefined rules or feature engineering. On one hand, the rule-based approaches (Bakman, 2007; Liguda and Pfeiffer, 2012; Shi et al., 2015) predefine a structured representation and maps the problem description into the structure by rules. These approaches usually accept only well-formed input and are difficult to scale to other problem types. On the other hand, the feature-based statistical learning approaches (Kushman et al., 2014; Roy and Roth, 2018) generate equation candidates and find the most probable equation by using predefined features. These approaches have two major drawbacks: (1) Their ability of equation generation is weak. An equation is generated by either replacing the numbers of existing equations in the training data, or enumerating possible combinations of math operators, numbers and variables, which leads to intractably huge search space. (2) They need manually designed features that are specific to math word problems.

Recent attempts (Ling et al., 2017; Wang et al., 2017) use sequence-to-sequence (seq2seq) model for math word problem solving and they have shown promising results. Wang et al. (2017) apply a standard seq2seq model to generate equations. They have shown that seq2seq models have the power to generate equations of which the problem types do not exist in the training data.

*Work was done at Microsoft Research.

Proceedings of the 27th International Conference on Computational Linguistics, pages 213–223
Santa Fe, New Mexico, USA, August 20-26, 2018.

However, we have observed two major shortcomings of the existing seq2seq model with attention mechanism. First, the model may generate spurious numbers, which are detrimental for math problem solving. As shown in Figure 1, the model generates a number "424" in the equation for *Problem 1*, which does not exist in the problem description. Since some of the numbers are unavoidably rare or do not appear in the training data, the existing models have difficulty generalizing to the long tail numbers. Second, the model is prone to generate numbers at wrong positions. In *Problem 2*, although the equation template is correctly generated by the model, the last token in the equation should be aligned to "1170" instead of "30" in the problem description. These two behaviors mentioned above are undesirable and will lead to wrong solutions.

In this paper, we focus on addressing the above two issues in the existing model (Wang et al., 2017) by incorporating Copy and Alignment mechanism to the seq2seq model (namely CASS). (1) Copy means directly copying the numbers in the problem description to the equations. In this way, we can avoid generating spurious numbers that are not in the problem description. (2) Alignment means that there is alignment information between the numbers in the equations and the numbers in the problem description. The model could learn the alignment in a supervised way.

Problem 1: Two busses leave Pleasant Grove High School at the same time going in opposite directions. One bus travels 43 mi/h and the other travels 57 mi/h. In how many hours will they be 350 miles apart?
Equation: (57 + 43) * x = 350
Baseline Seq2Seq + Attention: (424 + 43) * x = 350

Problem 2: A mortgage payment is $30 less than 3 times the property tax payment. The sum of the mortgage payment and the property tax payment is $1170. How much is the mortgage payment.
Equation: x = 3 * y – 30 ; x + y = 1170
Baseline Seq2Seq + Attention: x = 3* y - 30 ; x + y = 30

Figure 1: The baseline model generates a spurious number "*424*" in problem 1 and align numbers wrongly in problem 2.

When training the model, we adopt the reinforcement learning technique, specifically policy gradient. Because maximum likelihood estimation (MLE) suffers from the issue of "train-test discrepancy". It means that MLE uses a surrogate objective of maximizing equation likelihood during training, while the evaluation metric of the task is solution accuracy, which is non-differentiable. Therefore we use policy gradient to directly optimize the solution accuracy, which is more capable for this task.

Furthermore, we observe that the neural model and the traditional feature-based model are complementary. To take the advantage of both approaches, we add the result of our neural model as a simple feature to the feature-based model (Huang et al., 2017) to create a combined model.

We test our model on three publicly available datasets. The experimental results show that the copy and alignment mechanism is effective. Reinforcement learning leads to better performance than MLE. When combining our neural model with the feature-based model, we achieve the state-of-the-art results on all publicly available datasets.

The contributions of this paper are as follows:

1) We incorporate copy and alignment mechanism that augment the standard seq2seq model to address two types of errors: generating spurious numbers and generating numbers in wrong positions.

2) We adopt the reinforcement learning to optimize the solution accuracy, which is more capable for this task and leads to better performance.

3) We propose a simple but effective way to combine the neural model with a traditional feature-based model. The combined model outperforms the state-of-the-art models.

2 Problem Statement and Datasets

Given a math word problem P, the goal is to predict its answer A_p. In the training phase, we have annotations of both equation system E_p and answer A_p for each problem. In the testing phase, we obtain the final answer by generating equation and executing it with a math solver. We evaluate the task using solution accuracy.

Equation template is a unique form of an equation system. For example, given an equation system $\{2 * x + 4 * y = 34, x + 4 = y\}$, we replace the numbers with four number tokens $\{n_1, n_2, n_3, n_4\}$ and generalize the equations as the following equation template $\{n_1 * x + n_2 * y = n_3, x + n_4 = y\}$.

We can see that an equation system includes one or more equations and it is a solution for a specific

math word problems. In contrast, an equation template can correspond to several math problems. The number of templates in a dataset reflects the diversity of problem types. Specifically, we have a subset setting $T6$, which represents problems for which the associated template appeared equal to or more than six times in the subset. Note that T6 is a soft constraint of previous feature-based models.

We evaluate different models on three publicly available math word problem datasets[1].

- **Algebra 514** (Alg514) is created by Kushman et al. (2014). It contains 514 algebra word problems from Algebra.com. In the dataset, each template corresponds to at least 6 problems (T6 setting). It only contains 28 templates in total.

- **Number Word Problem** (NumWord) is created by Shi et al. (2015). It contains 2,871 number word problems (i.e., verbally expressed number problems) with 1,183 templates. The T6 subset contains 348 problems. One example problem is *"The sum of two numbers is 10. Their difference is 4. What are the two numbers?"*. We use its linear subset, which contains 986 problems. The vocabulary of this dataset is the smallest among the three, but contains more templates than Alg514.

- **Dolphin18K** (Dophin18K) is created by Huang et al. (2016). It contains 18,711 math word problems from Yahoo! Answers with 5,738 templates. It has much more problem types than the previous two datasets. This dataset is the most challenging of the three. We use its subset with equation annotation, which contains 10,644 problems. The T6 subset contains 6,827 problems.

3 Modeling

Math word problem solving can be formulated as a sequence prediction problem. We set x as the sequence of words in a math word problem description and we want to generate y, the sequence of tokens in an equation system. In this section, we describe (1) the baseline seq2seq model, (2) our two number-related mechanisms to address the issues in existing model.

3.1 Basic Sequence-to-Sequence Model

Following Wang et al. (2017), we first map numbers in the problem description to a list of number tokens $\{n_1,...,n_m\}$ and replace the corresponding numbers in the equation system with the number tokens. As shown in Figure 1, the problem description of *Problem 1* will be as follows after number mapping:
Two buses leave Pleasant Grove High School at the same time going in opposite directions. One bus travels n_1 mi/h and the other travels n_2 mi/h. In how many hours will they be n_3 miles apart?
And the equation system is normalized to: *(n_2 + n_1) * x = n_3*.

The baseline model of seq2seq with attention is based on the work by Bahdanau et al. (2015). It can be viewed as an encoder-decoder model. Basically, the encoder (that is implemented as a single-layer bidirectional GRU) reads the source tokens in problem description one-by-one and produces a sequence of hidden states $h_i = [h_i^F, h_i^B]$ with:

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

$$h_i^B = GRU(\phi^{in}(x_i), h_{i+1}^B) \quad (2)$$

where ϕ^{in} maps each token x_i to a fixed-dimensional vector.

At each decoding step j, the decoder receives the embedding of the previous generated token, the current decoder hidden state and the context vector to produce the target vocabulary distribution as described in the following equation:

$$P_{vocab}(w_j) = softmax(U[\phi^{out}(y_{j-1}), c_j, s_j] + b) \quad (3)$$

[1] (Wang et al., 2017) create a dataset containing 23,161 Chinese algebra math problems which has not been public yet.

where s_j is the decoder state and $\phi^{out}(y_{j-1})$ is the previous output embedding. c_j is the context vector and we calculate it as follows:

$$e_{ji} = v^T tanh(W_h h_i + W_s s_j + b_{attn}) \tag{4}$$

$$a_{ji} = \frac{\exp(e_{ji})}{\sum_{i'=1}^{m} \exp(e_{ji'})} \tag{5}$$

$$c_j = \sum_{i=1}^{m} a_{ji} h_i \tag{6}$$

Intuitively, a_{ji} defines the probability distribution over the input tokens. They are computed from the unnormalized attention scores e_{ji}. c_j is the weighted sum of the encoder hidden states. Specifically, W_h, W_s, U, b_{attn}, and b are parameters of the model.

Once we obtain the decoding result, a post-processing step recovers all tokens n_i to their corresponding numbers in the problem description.

3.2 Copy Numbers

In the basic model, the output token y_j is chosen via a softmax over all words in the output vocabulary. However, this model has the problem of generating spurious number tokens. For example, *Problem 1* in Figure 1 only contains three numbers, replaced with tokens $\{n_1, n_2, n_3\}$. Since the output vocabulary contains other tokens, the model may generate a token "n_4", which cannot be recovered and results in generating wrong equations.

To address this problem, we incorporate an attention-based copy mechanism into the basic model similar to Jia and Liang (2016). In our case, we only copy numbers from the source problem.

At each decoding step j, the model has to decide whether to *generate a token* from target vocabulary or *copy a number* from the problem description. The generation probability p_{gen} is modeled by:

$$p_{gen} = \sigma(W_c c_j + W'_s s_j + W_y y_{j-1} + b_{gen}) \tag{7}$$

where W_c, W'_s, W_y, and b_{gen} are parameters of the model and σ is the sigmoid function.

Then the output candidates are extended to the concatenation of the target vocabulary and the numbers in the math problem. We can obtain the output probability distribution:

$$P(w_j = w) = p_{gen} * P_{vocab}(w_j) + (1 - p_{gen}) * \frac{\sum_{i:w_i=w} a_{ji}}{\sum_{k:w_n} a_{jk}} \tag{8}$$

w_n is the set of numbers in the problem description. In this way, the model is capable of eliminating the spurious word error.

3.3 Align Numbers

In the decoding phase, the model often generates equations with numbers in wrong positions. *Problem 2* in Figure 1 is an example. The output equation structure is correct by the basic model. However, the numbers are aligned wrongly. When decoding the last token, the model should pay more attention to the source token "1170". Instead, the model wrongly aligned to the source token "30".

One characteristic of math word problems is that they have explicit alignment information between numbers in the problems and numbers in the equations. To improve the alignment in math problems, we use a supervised align mechanism to guide the training of our seq2seq model. Similar to Mi et al. (2016), the basic idea is minimize the cost of distance between the "true" alignment and the model predicted attention. We use the cross entropy loss function in the following:

$$\delta(a^i, \hat{a}^i) = -\sum_m \sum_n \hat{a}^i_{m,n} \times \log a^i_{m,n} \tag{9}$$

Please note that the disagreement only exists in numbers. The actual distribution $\hat{a}^i_{m,n}$ will be 1 if the *m*th token in the source is a number and equals to the *n*th tokens in the target sequence, otherwise it is 0.

4 Reinforcement Learning

As previously mentioned, MLE optimizes the surrogate objective of maximizing equation likelihood, while the evaluation metric of the task is solution accuracy. Besides, MLE assumes ground truth is provided at each timestep to predict the next token during training, which is not the case at test time. To remedy the discrepancy, we adopt the reinforcement learning technique, which directly optimizes the solution accuracy.

4.1 Policy Learning

The goal of the REINFORCE (Williams, 1992) is to find an agent that maximizes the task-level expected reward. We view our seq2seq model as a RL agent, which takes math problem description as input and then at each step, outputs a token y_j either by generating from the vocabulary or by copying numbers from the input problem. The agent follows a policy $\pi(y_j|\cdot)$, which we define as Equ 8.

The loss function and the gradient in the reinforcement learning are:

$$L_{RL} = -\sum_i \mathbb{E}_{p_\theta(\mathbf{y}^i|\mathbf{x}^i)}[R(\mathbf{x}^i, \mathbf{y}^i)] \tag{10}$$

$$\nabla_\theta L_{RL} = -\sum_i \mathbb{E}_{p_\theta(\mathbf{y}^i|\mathbf{x}^i)}[R(\mathbf{x}^i, \mathbf{y}^i)\nabla_\theta \log p_\theta(\mathbf{y}^i|\mathbf{x}^i)] \tag{11}$$

$$\approx -\sum_i \sum_{\mathbf{y}^i} p_\theta(\mathbf{y}^i|\mathbf{x}^i)R(\mathbf{x}^i, \mathbf{y}^i)\nabla_\theta \log p_\theta(\mathbf{y}^i|\mathbf{x}^i) \tag{12}$$

where $R(\mathbf{x}^i, \mathbf{y}^i)$ is the reward function. We define it as +1 if $\mathbf{y}^i$ yields to the correct solution, and -1 if $\mathbf{y}^i$ is not a valid equation or yields to a wrong solution.

Gradient Approximation It is often intractable to compute the gradient (Equ 11) because it involves taking an expectation over all possible equations. Therefore we sample from the model by using the top-k equations in the beam to approximate the gradient (Equ 12). Note that the gradient weights $p_\theta(\mathbf{y}^i|\mathbf{x}^i)$ of our sampling equations are renormalized to be summed up to 1.

In practice, the REINFORCE algorithm is unstable and converges slowly when the search space is large. Thus we pre-train our model based on maximum-likelihood for a few iterations before starting reinforcement learning.

4.2 Mixed Objective Function

To consider the alignment loss in Section 3.3, we define a mixed learning objective function:

$$L = L_{RL} + \lambda * \delta(a^i, \hat{a}^i) \tag{13}$$

where λ is a hyper-parameter that controls the magnitude of number alignment disagreement in the loss. In the pre-training step based on MLE, we replace the loss L_{RL} with the negative log likelihood:

$$L_{MLE} = -\sum_i \log p(\mathbf{y}^i|\mathbf{x}^i; \theta) \tag{14}$$

5 Model Ensemble

In this section, we observe and discuss that the neural model and the traditional feature-based model are complementary. To explore the advantage of both approaches, we combine two models by adding the result of our neural model as a feature to the feature-based model.

5.1 Feature-based Model

We use the state-of-the-art feature-based model (Huang et al., 2017) in our experiments. It contains two stages:

(1) Template retrieval. Candidate templates are derived from the training data. Given a problem p_i, they create a feature vector $f(p_i, t_j)$ for each candidate template t_j, and learn to rank the templates. They retrieve the top N templates.

(2) Equation ranking. Given the top N templates, they generate candidate equations with all possible number alignments. For example, there are two candidate equations given the template $x = n_1 - n_2$ and the numbers $\{3, 5\}$: $x = 3 - 5$ and $x = 5 - 3$. Similar to the previous stage, they create a feature vector $f(p_i, e_k)$ for each candidate equation e_k, and learn to rank the equations.

5.2 Generalization Ability of Neural Model

In Figure 2 we show the statistic of problems solved by our neural model and the feature-based model on Dolphin18K. 10.4% of problems can be solved correctly by both models, 18.0% can only be solved correctly by the feature-based model and 5.5% can only be solved correctly by our neural model.

To explore the generalization ability of our neural model, we further look into the 5.5% of problems that can only be solved correctly by our neural model (blue area in Figure 2). They can be summarized into two categories (examples are shown in Table 1):

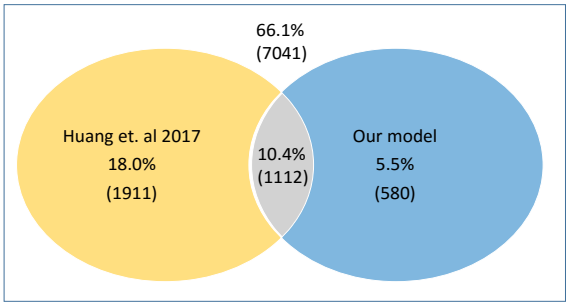


Figure 2: Statistic of problems solved by our neural model and feature-based model.

(1) **Ability to generate new equation templates**. Among the 5.5% of problems, there are 32.3% for which the template does not exist in the training data. Note that the feature-based model can only retrieve candidate templates from the training data. It means that the neural model has the ability to generate new equation templates, similar to the observation in Wang et al. (2017).

(2) **Ability to capture novel features**. For the remaining 67.7% of problems that can only be solved correctly by our neural model, their templates exist at least one time in the training data. The feature-based model should be able to retrieve the correct template and get the correct ranking for the equation, but it fails. This indicates that the neural model can capture novel features that the feature-based model is missing.

(1) **Problem**: Find 2 consecutive even integer such that 5 time the small integer be 10 more than 3 time the large integer. **Equation**: $5 * (2 * x) = 3 * (2 * x + 2) + 10$ (Our model generates the equation of which template does not exist in the training data)
(2) **Problem**: The area of a rectangular garden is 4472 ft$\hat{2}$. If the length of the garden is 86 feet, what is its width? **Equation**: $86 * x = 4472$ (its template exists in the training data)

Table 1: Example problems that are only solved correctly by our neural model.

5.3 Model Ensemble

In previous section, we observe that neural model is complementary to the feature-based approach when the templates are sparse or do not exist in the training data. Hence, it is intuitive to combine the neural model and feature-based approach to obtain better performance. To ensemble both models, We incorporate the neural model output in the two stages of the feature-based approach by (Huang et al., 2017):

(1) In the template retrieval stage, we add a feature of neural template. Given a problem, we derive the template t_{seq} of our neural output equation. For each candidate template t_i, if it is equivalent to t_{seq}, we set the value of the neural template feature to 1, otherwise set to 0.

(2) In the equation ranking stage, we add a feature of neural answer. Given a problem, we calculate the answer of our output equation a_{seq}. For each candidate equation e_j, if its answer a_j is equal to a_{seq}, we set the value of the neural answer feature to 1, otherwise set to 0.

6 Experiments

In this section, we test the performance of our model on three datasets. Furthermore, we conduct experiments to examine the effectiveness of our neural model as a feature in the hybrid model.

6.1 Implementation Details

Experiments are done in 5-fold cross-validation: in each run, 70% is used for training, 10% for validation, and 20% for testing. We report answer accuracy. The dimension of encoder hidden state, decoder hidden state and embeddings are 100 in NumWord and Alg514, 512 in Dolphin18K. All model parameters are initialized randomly with Gaussian distribution. The hyper-parameter λ for supervised attention of alignment is set to 1.0. We use SGD optimizer with decaying learning rate initialized as 0.5. Dropout rate is set to 0.5. The criterion for learning to stop is answer accuracy in validation set. The vocabulary consists of words observed in the training data more than or equal to N times. We set $N = 1$ for NumWord and $N = 5$ for the other two datasets. The beam size is set to 20 in the decoding stage. For reinforcement learning, we utilize a pre-training with maximum likelihood for 50 iterations. We set the beam size for sampling to 10. We tune all the hyper-parameters using a separate dev set.

6.2 Results

We implement the approach in Wang et al. (2017) as the baseline (**Seq2SeqAttn**). Its performance on Alg514 is 19.4%, versus 17.2% they reported.

Models	Alg514	NumWordT6	NumWordT1	Dolphin18KT6	Dolphin18KT1
Seq2SeqAttn (MLE)	19.4%	19.7%	11.0%	13.0%	10.2%
+Copy (MLE)	41.4%	59.9%	23.0%	20.2%	12.9%
+Copy+Align (MLE)	41.8%	60.4%	26.8%	21.0%	13.1%
+Copy+Align (RL)	44.5%	64.0%	29.2%	23.3%	15.9%
Huang et al. (2017)	81.6%	42.0%	20.8%	30.6%	28.4%
+ CASS^{RL} (hybrid)	**82.5%**	**65.8%**	**29.7%**	**33.2%**	**29.0%**

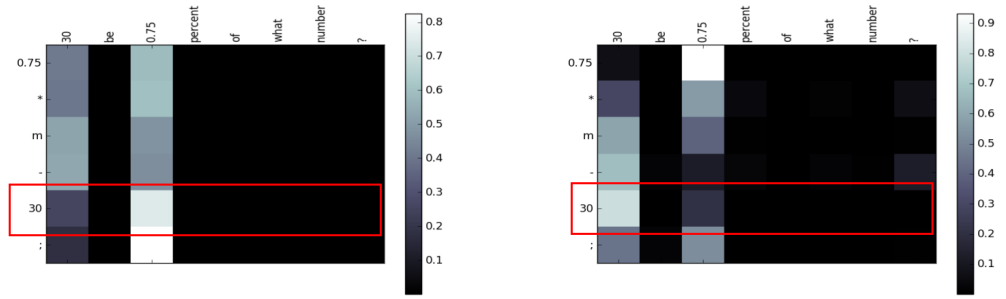
Table 2: Performances on three datasets. "*CASS*" means Seq2SeqAttn + Copy + Align.

From Table 2, we can see that our approach significantly improves the performance over the baseline. Especially on NumWord dataset, our model greatly exceeds the baseline with 44.3% increase on $T6$ and 18.2% increase on $T1$, and is already better than current state-of-the-art model (Huang et al., 2017).

Copy impact The copy mechanism contributes greatly on all three datasets as shown in Table 2. It achieves a 40.2% accuracy on NumWordT6, even outperforms the feature-based models. In the most challenging dataset Dolphin18K among the three, with more diverse problems and larger problem size, our model still gets a 7.2% increase on Dolphin18KT6 and 2.7% increase on Dolphin18KT1.

Align impact From Table 2, we can see our model achieves a consistent improvements over all three datasets using number alignment. In the math problem in Figure 3a, for the target number "30", Seq2SeqAttn + Copy model has a higher attention weight of the source number "0.75" than the source number "30". Thus the model wrongly aligned the target number "30" to the source number "0.75". After incorporating the alignment mechanism, the target number "30" is now aligned correctly as shown in Figure 3b.

Reinforcement learning We can see RL performs substantially better than MLE. As the objective of RL is to optimize the solution accuracy, we further investigate the model's capability of considering multiple admissible equations. We compare the annotated equations with the equations generated by our model on Dolphin18KT1. Using RL training, there are **7.1%** of problems of which the predicted equations are different from the annotated ones but still yield to correct solutions, compared to 4.4% using MLE.



(a) **Seq2SeqAttn+Copy** (b) **Seq2SeqAttn+Copy+Align**

Figure 3: Example alignments of (a) Seq2SeqAttn + Copy, (b) Seq2SeqAttn + Copy + Align.

6.3 Model Ensemble

We further test the performance of the hybrid model that combines our neural model into the feature-based model.

From Table 2, we can see that when our neural model is incorporated into the feature-based model, we achieve the state of the art. This indicates that the two models are complementary and the hybrid model takes advantages of both models.

Furthermore, we show detailed results when our neural model is incorporated in each of the two stages:

(1) Template retrieval. Same as Huang et al. (2017), we report Hit@N accuracy, which means the correct template for a problem is included in the top N list returned by the model. Table 3 shows the Hit@1/Hit@3 accuracy.

We can see that incorporating neural model helps increase template Hit@N accuracy, which means it can retrieve accurate templates for more problems. Except for Alg514, adding the neural feature drops Hit@3 by 1.3%. On Alg514, the feature-based model already outperforms our neural model. In addition, its data size is the smallest among the three datasets, which might be challenging for neural model to learn from. Therefore, adding neural feature on this dataset might bring much noise that leads to performance decrease.

Dataset	Huang et al. (2017)	+Neural
Alg514	62.8/80.9	60.3/79.6
NumWordT6	38.6/70.0	**38.9/72.9**
NumWordT1	20.0/35.1	20.4/35.1
Dolphin18KT6	27.3/39.7	**27.3/41.1**
Dolphin18KT1	17.5/26.3	**18.2/27.1**

Table 3: Accuracy (%) of template retrieval with top 1 / top 3 templates retrieved.

(2) Equation ranking (final accuracy). After retrieving the top N templates from the previous stage, we align numbers with template slots to generate candidate equations for ranking. The final results are shown in the last row of Table 2.

From the table, we can see that the neural feature is effective, achieving the state of the art on all three datasets. Especially, on NumWord dataset, incorporating neural model contributes to 23.8% accuracy increase on T6 and 8.9% increase on T1. Surprisingly, the performance on Alg514 is still improved, though the template retrieval accuracy in previous stage decreased. This indicates that the feature-based model has included some errors that caused by wrong number alignment in the second stage, and our neural feature eliminates this type of errors.

7 Related Work

Our work is related to three research areas: math word problem solving, the seq2seq model, and reinforcement learning for sequence generation.

7.1 Math Word Problem Solving

The approaches to solve math word problems can be divided into three categories: rule-based approach, feature-based approach, and neural-based approach.

Rule-based approaches (Bobrow, 1964a; Bobrow, 1964b; Bakman, 2007; Liguda and Pfeiffer, 2012; Shi et al., 2015) accept only well formed input sentences and map them into predefined structures by rules. These methods require strict constraints on both input text and math problem types.

Feature-based approaches design features and learn rankers to rank equation candidates to the math problems. Hosseini et al. (2014) design features to classify verbs to addition or subtraction. Kushman et al. (2014), Zhou et al. (2015), Upadhyay et al. (2016) use features such as dependency path between two numbers. Koncel-Kedziorski et al. (2015), Roy and Roth (2015) extract quantity information as features. Wang et al. (2018) extract features for quantity pairs and uses a reinforcement framework to construct an equation tree with the constraint of one unknown variable. Roy and Roth (2015), Roy et al. (2016) leverage the tree structure of equations. Mitra and Baral (2016), Roy and Roth (2018) design features for a few math concepts (e.g. Part-Whole, Comparison). Huang et al. (2017), Roy and Roth (2017) focus on the use of fine-grained expression and number units. These approaches requires manual feature design and it is difficult to generalizing the features to other problem types.

Recently, researchers try to build end-to-end neural models to solve math word problems. Ling et al. (2017) focus on multiple-choice problems. It takes a problem description as input and outputs the rationale and the final choice. Wang et al. (2017) apply a standard seq2seq model to generate equations under the constraint of one variable. However, the model is prone to generate numbers that do not exist in the problem or in wrong positions. Our model addresses these issues by incorporating copy and align.

7.2 Sequence-to-Sequence Models

Recent applications of seq2seq model in many areas have shown promising results.

Copy mechanism is proved effective in dealing with rare or unknown words. The main idea is to decide when and what to copy from the source words in the decoding phase. Jia and Liang (2016) use an attention-based copy mechanism to copy arguments from natural language query to logical form for the task of semantic parsing. Gulcehre et al. (2016) design a pointer network to select tokens in the input. Different from previous work, we consider copying only numbers from the problem description.

Some recent work have been proposed to improve the alignment for neural machine translation (Liu et al., 2016; Mi et al., 2016). They introduced a supervised attention mechanism to utilize the word alignment information between sentence pairs in the training data. They first obtain soft word alignments from conventional alignment models. Then they try minimize the distance between the soft word alignments and the word alignments based on model attention in the training procedure. In our math problems, the alignment information is explicit and can be directly obtained.

7.3 Reinforcement Learning for Sequence Generation

The classic REINFORCE algorithm (Williams, 1992) has been applied to solve a wide variety of tasks: machine translation (Norouzi et al., 2016), image captioning (Rennie et al., 2017), semantic parsing (Liang et al., 2017; Guu et al., 2017) and summarization (Paulus et al., 2018). Reinforcement learning is applied usually when the evaluation metric is non-differentiable, or there are multiple candidates that yields to the ground truth despite of token orders in target sequence. It trains an agent with a given environment to directly optimize the task evaluation metric (e.g., BLEU or ROUGE). We apply reinforcement learning in math problem solving for two reasons: (1) we evaluate our task with solution accuracy which is not directly optimized in maximum likelihood estimation; (2) there are multiple equations that yield to the correct solution which maximum likelihood estimation would ignore.

8 Conclusion

In this paper, we follow previous work that applies seq2seq model to solve math word problems. We augment the model with two mechanisms: copy and align. When training the model, we adopt the reinforcement learning to directly optimize the solution accuracy. Our model significantly improves the

performance. To further explore the effectiveness of both neural approach and feature-based approach, we add our model output as a feature into the feature-based model. The combined model leverages advantages of both approaches and achieves the-state-of-the-art result.

In the future work, we will design more methods to combine the two models to better leverage each approach. Furthermore, for math problems, several modules are important, such as mathematical concept and commonsense knowledge, which we plan to incorporate into our model in the future.

Acknowledgements

Thanks to the anonymous reviewers for their helpful comments and suggestions. This work is supported by the National Natural Science Foundation of China (61472453, U1401256, U1501252, U1611264,U1711261,U1711262). Jian Yin is the corresponding author.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conferene on Learning Representation*.

Yefim Bakman. 2007. Robust understanding of word problems with extraneous information. http://arxiv.org/abs/math/0701393.

Daniel G. Bobrow. 1964a. Natural language input for a computer problem solving system. Technical report, Cambridge, MA, USA.

Daniel G. Bobrow. 1964b. Natural language input for a computer problem solving system. Ph.D. Thesis.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, October.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Danqing Huang, Shuming Shi, Chin-Yew Lin, and Jian Yin. 2017. Learning fine-grained expressions to solve math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Robin Jia and Percy Liang. 2016. Data recombinatin for neural semantic parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Chen Liang, Jonathan Berant, Quoc Le, Kennet D.Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Christian Liguda and Thies Pfeiffer. 2012. Modeling math word problems with augmented semantic networks. In *Natural Language Processing and Information Systems. International Conference on Applications of Natural Language to Information Systems (NLDB-2012)*, pages 247–252.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. In *Proceedings of the COLING 2016*.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

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

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *Sixth International Conference on Learning Representations*.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *2017 Conference on Computer Vision and Pattern Recognition*.

Subhro Roy and Subhro Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752. The Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2017. Unit dependency graph and its application to arithmetic word problem solving. In *Proceedings of the 2017 Conference on Association for the Advancement of Artificial Intelligence*.

Subhro Roy and Dan Roth. 2018. Mapping to declarative knowledge for word problem solving. In *Transactions of the Association for Computational Linguistic*.

Subhro Roy, Shyam Upadhyay, and Dan Roth. 2016. Equation parsing: Mapping sentences to grounded equations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Shuming Shi, Wang Yuehui, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang, and Wen tau Yih. 2016. Learning from explicit and implicit supervision jointly for algebra word problems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural model for math word problem problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. 2018. Mathdqn: Solving arithmetic word problems via deep reinforcement learning. In *Thrity-Second AAAI Conference on Artificial Intelligence*.

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

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Personalizing Lexical Simplification

John Lee, Chak Yan Yeung
Department of Linguistics and Translation
City University of Hong Kong
jsylee@cityu.edu.hk, chak.yeung@my.cityu.edu.hk

Abstract

Given an input text from the user, a lexical simplification (LS) system makes the text easier to understand by substituting difficult words with simpler words. The best substitution may vary from one user to another, given individual differences in vocabulary proficiency level. Most current systems, however, do not consider these variations, and are instead trained to find one optimal substitution or list of substitutions for all users. This paper measures the benefits of using complex word identification (CWI) models to personalize an LS system. Experimental results show that even a simple CWI model, based on graded vocabulary lists, can help reduce the number of unnecessary simplifications and complex words in the output for learners of English at different proficiency levels.

1 Introduction

Lexical simplification (LS) is the task of replacing difficult words with simple words in a text, while preserving its meaning and grammaticality. It aims to produce output text that is easier to understand for readers with special needs, such as language learners, children (Kajiwara et al., 2013), and those with language disabilities (Devlin and Tait, 1998; Carroll et al., 1999). Table 1 shows an example input sentence to an LS system, and the ranked list of possible substitutions for the target word, i.e., the word that should be simplified. Most LS systems first perform complex word identification (CWI) to detect target words (i.e., "avoid" in this case), and then find appropriate substitutions for them (i.e., "prevent", "stop", etc., in order of preference).

CWI is thus an important first step in the LS pipeline. On the one hand, an overly conservative CWI model would fail to detect many complex words, leaving them unsimplified and limiting the utility of the LS system. On the other hand, an overly aggressive CWI model would be prone to misidentify simple words as complex, leading to unnecessary simplifications and increasing the risk of substitution errors. In an error analysis on LS systems, CWI-related error categories turned out to be among the most frequent (Shardlow, 2014). CWI has been receiving increasing attention in recent years, including a recent SemEval shared task (Paetzold and Specia, 2016b). Since the test set was annotated by a single learner, however, CWI performance on language learners at different levels of vocabulary proficiency continues to be under-explored.

Indeed, most LS evaluations assume one best substitution or one fixed ranked list of substitutions (cf. Table 1), and do not take into account variations in vocabulary knowledge among users. This "one-size-fits-all" approach is suboptimal since word complexity is in the eye of the beholder: a word that is complex for a low-proficiency user may be perfectly familiar to a high-proficiency user, or even to a low-proficiency user whose native language has a cognate word. As a case in point, consider the dataset in the SemEval 2016 CWI shared task. The Krippendorff's Alpha agreement was 0.244 among the 20 annotators. Further, suppose one builds an oracle CWI system on the test set in the shared task, and apply it on the Japanese learners of English in the dataset constructed by Ehara et al. (2010). For the

Proceedings of the 27th International Conference on Computational Linguistics, pages 224–232
Santa Fe, New Mexico, USA, August 20-26, 2018.

Input sentence	Gold ranked list of substitutions
Typically, a fast shutter speed will require a larger aperture to ensure sufficient light exposure, and a slow shutter speed will require a smaller aperture to *avoid* excessive exposure.	1. prevent 2. stop 3. {dodge, miss, evade, escape} 4. {elude, limit, avert, bypass, deter}

Table 1: An input sentence to a lexical simplification system, and the gold ranked list of substitutions for the target word, "avoid". This example is taken from the BenchLS dataset (Paetzold and Specia, 2016a).

least proficient learner in this dataset, the oracle would fail to identify 35.30% of the complex words; for the most proficient learner, it would cause false alarm for 93.67% of the non-complex words.

To address "the expected heterogeneity among non-native speakers with different language backgrounds and proficiency levels" (Paetzold and Specia, 2016b), this paper argues for the use of personalized CWI models to improve LS performance. We present the first quantitative evaluation of personalized LS on learners at varying levels of English proficiency. Further, we demonstrate that even a simple CWI model, based on graded vocabulary lists, can reduce the number of unnecessary simplifications and complex words in the output text.

The rest of the paper is organized as follows. The next section summarizes previous LS research, focusing on CWI and Substitution Ranking, where we will attempt personalization. Section 3 gives details on our data. Section 4 describes our approach and baselines. Section 5 defines the evaluation metrics. Section 6 presents experimental results and discusses the extent to which LS systems can benefit from personalized CWI. Finally, Section 7 concludes.

2 Previous work

Most lexical simplification (LS) systems adopt a pipeline architecture (Shardlow, 2014; Paetzold and Specia, 2016b). The pipeline typically begins with **Complex Word Identification** (CWI) to find target words to be simplified. A **Substitution Generation** component then generates candidate replacements for these complex words. These substitutions can be learned, for example, from standard Wikipedia and Simple Wikipedia (Horn et al., 2014), or with word embedding models (Glavaš and Štajner, 2015; Paetzold and Specia, 2016c). The **Substitution Selection** step then discards candidates that may distort the meaning of the text or affect its grammaticality, and retains those that best fits the context. Lastly, **Substitution Ranking** determines the best output by ranking the remaining candidates by simplicity.

LS research has mostly adopted the user-independent approach. We now review previous work in two components of the pipeline to which we will attempt to add personalization: CWI (Section 2.1) and Substitution Ranking (Section 2.2).

2.1 Complex word identification

The complex word identification (CWI) task classifies words in a text as either "complex" or "non-complex". Complex words are those that are difficult for a non-native speaker to understand; non-complex words are those that are not (Paetzold and Specia, 2016b; Yimam et al., 2017). In the 2016 SemEval CWI shared task, the best team, which combined various lexicon-based, threshold-based and machine learning voter sub-systems, achieved a precision of 0.147 and recall of 0.769 (Paetzold and Specia, 2016b). Overall, word frequencies were found to give the most reliable prediction for word complexity. The shared task was not designed to test performance on users at different proficiency levels, since the test set was annotated by a single learner.

To date, most CWI research has taken the user-independent approach, with only a few published studies on personalized CWI. Zeng et al. (2005) showed that demographic features can help improve CWI performance for individual users in the medical domain. Laufer and Nation (1999) proposed the "word sampling" method. Using a ten-level proficiency scale, with 1000 words at each level, this method samples a fixed number of words from the learner as the training set, and then labels unseen words based on their proximity to these words. Ehara et al. (2012; 2014) described a two-step algorithm, mainly using

word frequency statistics as features. In the first step, all words are organized as a multiple complete graph. The k most informative nodes, or words, are selected by a graph-based active learning approach. The learner then rates his/her knowledge of these k words on a five-point scale (see Section 3.1). Lee and Yeung (2018) followed the same procedure to create a training set for Chinese CWI. In the second step, a personal CWI classifier is trained for each learner. Using a 50-word training set, Ehara et al. (2014) achieve 76.44% accuracy in English CWI with Local and Global Consistency, a label propagation algorithm. Lee and Yeung (2018) reported 78.0% accuracy for Chinese CWI with an SVM classifier.

An alternative approach is to build only a fixed number of CWI models. After soliciting annotation of vocabulary knowledge on a small number of sample words from the user, the system predicts the most suitable model. These models can correspond to graded vocabulary lists, such as the New General Service Lists (http://www.newgeneralservicelist.org), or they may potentially be trained on graded text corpora, such as the Newsela corpus (Xu et al., 2015). This approach thus offers more coarse-grained personalization, akin to graded readers, and not every user necessarily fits neatly into one of the predetermined levels.

2.2 Substitution ranking

Given a set of candidates from the Substitution Selection step, the Substitution Ranking step chooses the simplest candidate. Most current approaches impose the same notion of simplicity on all users. Recent systems have applied machine learning approaches, such as the SVM (Horn et al., 2014) and neural models (Paetzold and Specia, 2017), on a range of features including word frequencies in large corpora and human rankings in LS datasets. This step can potentially be enhanced with CWI to filter out candidates that are complex words.

3 Data

In this section, we first describe our dataset of language learners (Section 3.1), and then explain how we used it to create personalized versions of an existing, user-independent dataset of lexical simplification (Section 3.2).

3.1 User dataset

Our user dataset was annotated by 15 learners of English as a foreign language who were native speakers of Japanese (Ehara et al., 2010). Each learner rated their knowledge of 12,000 English words on a five-point scale: (1) Never seen the word before; (2) Probably seen the word before; (3) Absolutely seen the word before but do not know its meaning, or tried to learn the word before but forgot its meaning; (4) Probably know, or able to guess, the words meaning; and (5) Absolutely know the words meaning. Following Ehara et al. (2014), we collapsed these five categories into either "complex" (score 1 through 4) or "non-complex" (score 5). Table 2 shows some example annotations.

These 15 learners covered a wide range of proficiency levels. The least proficient learner rated only 17.97% of the words as "non-complex", while the most proficient one rated 94.26% of the words as "non-complex". To help analyze the effect of personalized LS at different proficiency levels, we define two subsets of learners based on vocabulary proficiency:

- **Low Proficiency.** The four least proficient learners, all of whom knew less than 41% of the words, constitute the "Low" proficiency subset.
- **High Proficiency.** The four most proficient learners, all of whom knew more than 75% of the words, constitute the "High" proficiency subset.

3.2 Personalized lexical simplification dataset

The BenchLS dataset contains 929 instances of target words and their gold simple words, annotated by English speakers from the U.S. (Paetzold and Specia, 2016a). For the 15 users (Section 3.1), we created 15 personalized versions of BenchLS with the following steps:

Word	User A	User B	User C	User D
avert				
avoid				√
bypass	√	√	√	√
deter				√
dodge				√
elude				
escape	√	√		√
evade				√
limit	√	√	√	√
miss	√	√		√
prevent		√		√
stop	√	√	√	√

Table 2: Annotations on 12 example words by four users in the user dataset (Section 3.1). Non-complex words are indicated with a checkmark (√); all other words are complex.

User A	User B	User C	User D
1. stop 2. {miss, escape} 3. {limit, bypass}	1. prevent 2. stop 3. {miss, escape} 4. {limit, bypass}	1. stop 2. {limit, bypass}	*null*

Table 3: Personalized gold ranked list of substitutions for the target word "avoid" in Table 1, based on annotations in the user dataset shown in Table 2.

- When the target word is non-complex for the user, we set the gold answer to *null*. Since the user already understands the word, in the interest of meaning preservation, the system should not simplify it. Consider the target word "avoid" in Table 1. Since it is non-complex for User D (Table 2), the gold answer for this target word should be *null* for User D (Table 3).

- When the target word is complex for the user, the system should attempt simplification on it. We retrieve the gold ranked list of substitutions in BenchLS, and remove all complex words from the list, since they would not be helpful for the user. If the list becomes empty, we exclude this instance from our evaluation. Consider again the target word and its gold list of substitutions in Table 1. When editing this list for Users A, B, and C, we keep only those words that are non-complex for them, according to their annotations in Table 2. Notably, the first-ranked substitution is no longer "prevent" for Users A and C, since they do not know this word. The resulting personal gold lists are shown in Table 3.

After filtering, our evaluation dataset contained 883 instances.

4 Approach

We propose a lexical simplification (LS) algorithm that aims to turn complex words in a text into non-complex ones for the user, while keeping intact the non-complex words in the text. This algorithm applies a personalized complex word identification (CWI) model in two steps in the LS pipeline:

- **CWI for detection**: Most current approaches deploy a user-independent CWI model as the first step in their pipeline to detect words that should be simplified (Section 2.1). In contrast, we train a personalized CWI model for this purpose, such that the choice of target words can vary from one user to another. For example, given the input sentence in Table 1, the system is expected to simplify "avoid" for Users A, B, and C, but not for User D (Table 3).

Level	# Words	Content of vocabulary List
1	1,000	First 1,000 words in the New General Service List (NGSL)
2	2,000	First 2,000 words in the NGSL
3	2,800	All words in the NGSL
4	6,777	All words in the NGSL, the TOEIC Service List (TSL), the New Academic Word List (NAWL), and the Business Service List (BSL)

Table 4: Vocabulary lists corresponding to the four CWI models used in the Graded Vocabulary List approach (Section 4).

- **CWI for ranking**: The Substitution Ranking step of the pipeline ranks the substitution candidates according to their simplicity (Section 2.2). In addition, we apply the personalized CWI model to filter out candidates that are complex for the user. For example, given the list of candidate substitutions in Table 1, the system is expected to reject the word "prevent" for Users A and C, since they do not know it. If the model predicts all candidates to be complex, it still returns the first-ranked candidate as the suggested substitution.

Our experiments apply various configurations of the following three models[1] to perform CWI for detection and CWI for ranking, respectively:

- **Baseline** (`nil`): When used as CWI for detection, this baseline always predicts a word to be complex regardless of the user, so the system always attempts simplification. When used as CWI for ranking, it always predicts a word to be non-complex, so the system never removes any word from the user-independent list of substitutions.

- **Oracle** (`gold`): The oracle performs perfect CWI on each user, according to his/her annotation in the user dataset (Section 3.1). When the oracle is used as CWI for detection, the system attempts simplification if and only if the word is complex. When it is used as CWI for ranking, the system returns the highest-ranked substitution that is non-complex for the user.

- **Graded Vocabulary List** (`auto`): This model automatically predicts a word as complex or non-complex, based on graded vocabulary lists. As shown in Table 4, we define four vocabulary proficiency levels, based on 6,777 words covered by a number of vocabulary lists. We then construct four CWI models corresponding to these four levels; each model predicts all words in its vocabulary list to be "non-complex", and all other words to be "complex".

  Next, we select n out of the 6,777 words in the dataset as the training set, with the n words divided evenly among the four levels. For each user, based on his/her annotation in the user dataset (Section 3.1), we calculate the precision and recall of each of the four CWI models. We then assign the user to the model with the highest F-score. In our evaluation, we set $n = 40$, meaning that each user would have to annotate 40 words as "complex" or "non-complex" in order to personalize the LS system.[2]

5 Evaluation metrics

We report two metrics used in previous LS research (Horn et al., 2014; Glavaš and Štajner, 2015):

- *Precision* is the ratio of correct simplifications out of all simplifications made by the system.

- *Accuracy* is the ratio of correct simplifications out of all target words that should be simplified, i.e., in our context, out of all complex target words.

[1] We did not evaluate the model proposed by Ehara et al. (2014) since we were not able to get access to its system output.
[2] Users who do not know any of the 40 words are assigned to the level-1 model.

CWI for detection	CWI for ranking	Precision	Accuracy	Readability
nil	nil	21.39%	89.95%	91.47%
		43.14% (low only)	76.31% (low only)	80.02% (low only)
		4.25% (high only)	100% (high only)	99.01% (high only)
auto	nil	31.97%	76.19%	89.40%
nil	auto	23.36%	**94.19%**	**94.57%**
auto	auto	**34.81%**	80.36%	91.67%
gold	nil	89.95%	89.95%	95.55%
nil	gold	26.31%	100%	100%
gold	gold	100%	100%	100%

Table 5: Performance on the personalized LS dataset (Section 3.2), based on gold substitution lists in BenchLS (Paetzold and Specia, 2016b) and on three methods for CWI for detection and CWI for ranking: nil predicts all target words to be complex, and all candidate substitutions to be non-complex; gold returns the annotation in the user dataset (Section 3.1); auto is the automatic CWI model based on graded vocabulary lists. Low and high refer to proficiency level (Section 3.1).

Correctness of a simplification is based on the personalized LS dataset (Section 3.2) rather than the BenchLS dataset (Paetzold and Specia, 2016a). For instance, in the sentence in Table 1, it is correct to substitute "avoid" with "prevent" for User B, but incorrect to do so for User A and C. Further, it is deemed incorrect to make any substitution for User D (Table 3).

Note that precision penalizes simplifications of non-complex words, even if the substitution is also a non-complex word in the gold list in BenchLS. For some users, this penalty may be reasonable since few substitutions fully preserve the meaning and intent of the original text. For others, unnecessary simplifications may be perfectly acceptable, given the overriding goal of minimizing the number of unknown words in the output text.

To represent the latter perspective, we also report the *readability* metric, which computes the proportion of words in the output text that can be understood by the user and do not distort the original meaning. More precisely, a word in the output text is *readable* if it satisfies two conditions: (1) it is non-complex for the user; and (2) it is either included in the original gold substitution list in BenchLS, or it is unsimplified. Since this metric does not consider whether the original word is complex or non-complex, it allows unnecessary simplification as long as it is appropriate.

6 Experiments

We conducted two experiments to evaluate the effect of adding personalization to a lexical simplification (LS) system. In both experiments, the baseline is a user-independent ranked list of substitutions. We manipulate the list with various combinations of CWI for detection and/or CWI for ranking (see Section 4), and then measure any gain in LS performance. All results are averaged among the 15 users in the dataset (Section 3.1).

6.1 Experiment 1: Personalization with gold substitutions

To better isolate the performance gain as a result of personalization, the first experiment used the gold substitution lists in BenchLS. This design ensures that the performance gain would not be influenced by the extent and nature of the particular substitution errors made by the LS system chosen as baseline.

Table 5 reports performance on the personalized LS dataset (Section 3.2). Because of the use of gold substitutions in BenchLS, the absolute level of performance is overestimated. We will focus on the *difference* between the baseline and the personalized systems, and will verify if the difference holds in realistic conditions in the second experiment.

Precision. The oracle (detect=gold, rank=gold), by definition, achieved the perfect score in all metrics. In contrast, the user-independent approach (detect=nil, rank=nil), even with perfect substi-

CWI for detection	CWI for ranking	Precision	Accuracy	Readability
nil	nil	8.09% (low only) 14.87% (high only) 1.96%	39.17% (low only) 27.06% (high only) 51.25%	37.94% (low only) 27.53% (high only) 45.94%
auto	nil	12.03%	32.59%	61.27%
nil	auto	12.37%	**50.91%**	53.45%
auto	auto	**18.01%**	42.25%	**69.39%**
gold	nil	39.28%	39.17%	83.38%
nil	gold	14.57%	52.52%	56.01%
gold	gold	58.57%	52.52%	87.44%

Table 6: Performance on the personalized LS dataset (Section 3.2), based on output from a user-independent LS system (Paetzold and Specia, 2017), and on three methods for CWI for detection and CWI for ranking: nil predicts all target words to be complex, and all candidate substitutions to be non-complex; gold returns the annotation in the user dataset (Section 3.1); auto is the automatic CWI model based on graded vocabulary lists. Low and high refer to proficiency level (Section 3.1).

tutions, hit a ceiling at 21.39% precision. One source of error for precision was the simplification of non-complex words in the input text, since the system always attempted simplification. Naturally, this was especially problematic for high-proficiency users, as reflected in the lower precision (4.25%), but had less impact on low-proficiency users (43.14% precision). Personalized CWI reduced these unnecessary simplifications, raising precision to as high as 89.95% with oracle CWI for detection (detect=gold, rank=nil). The automatic CWI approach (detect=auto, rank=nil), based on vocabulary lists, also succeeded in reducing them and attained 31.97% precision, a 10% absolute improvement over the baseline.

Accuracy. Another source of error for the user-independent approach was the fact that some gold substitutions were complex; in other words, while these substitutions were considered simpler than the target words, they were still too difficult for the user. This phenomenon resulted in the 89.95% accuracy rate for the user-independent approach (detect=nil, rank=nil). As expected, the phenomenon was magnified among low-proficiency users, as shown by the lower accuracy (76.31%), but it barely affected the high-proficiency users (100% accuracy). Personalized CWI helped steer the system to choose non-complex words as output. Oracle CWI for ranking (detect=nil, rank=gold), by definition, achieved 100% accuracy. The automatic CWI approach (detect=nil, rank=auto) yielded smaller improvement but, at 94.19% accuracy, still outperformed the baseline by over 4% absolute.

Readability. With respect to the readability measure, which accepts unnecessary simplifications, personalized approaches still produced better output than the user-independent baseline. Among configurations that did not involve gold, the highest readability score (94.57%) was achieved by the system that always attempted simplification but used automatic CWI for ranking (detect=nil, rank=auto); this represented a 3% improvement over the baseline. It also achieved the highest accuracy (94.19%). Using automatic CWI for both detection and ranking (detect=auto, rank=auto) produced the best precision (34.81%), an absolute improvement of over 13% over the baseline. However, its accuracy and readability were suboptimal because, by making fewer simplifications, it left more complex words in the input text unsimplified.

6.2 Experiment 2: Personalization with automatically generated substitutions

Results from the first experiment, which assumed perfect substitutions, showed consistent performance gains as a result of personalization. The second experiment investigated whether these gains would hold under more realistic conditions. Instead of gold substitutions from BenchLS, we used the output of a state-of-the-art, user-independent LS system (Paetzold and Specia, 2017). Table 6 shows the performance on the personalized LS dataset (Section 3.2). In this setting, the oracle (detect=gold, rank=gold)

attained only 58.57% precision and 52.52% accuracy. As will be discussed below, the gap between the baseline and the personalized systems persisted.

Precision. The user-independent approach (detect=`nil`, rank=`nil`) achieved only 8.09% precision. A major source of error for precision, as observed in the first experiment, was the simplification of non-complex words in the input text. Oracle CWI for detection (detect=`gold`, rank=`nil`) raised the precision to 39.28%. Automatic CWI for detection (detect=`auto`, rank=`nil`) yielded 12.03% precision, improving the baseline by almost 4% absolute.

Accuracy. The ability of personalization to reduce the other major source of error — selection of complex words as substitutions — was also observed in this experiment. While the accuracy of the user-independent approach (detect=`nil`, rank=`nil`) was only 39.17%, automatic CWI for ranking (detect=`nil`, rank=`auto`) improved it by over 11% to reach 50.91%. This level of performance was very close to the upper bound of 52.52% accuracy suggested by oracle CWI for ranking (detect=`nil`, rank=`gold`).

Readability. In terms of readability, after excluding configurations that involve `gold`, the best readability score (69.39%) was achieved by using automatic CWI for both detection and ranking (detect=`auto`, rank=`auto`). This represented an absolute improvement of over 31% in comparison to the user-independent baseline. Unlike in the first experiment, the system's conservativeness in making simplifications worked in its favor because of the presence of substitution errors. This configuration also yielded the highest precision (18.01%), outperforming the baseline by almost 10%. However, the highest accuracy (50.91%), similar to the first experiment, was obtained by using automatic CWI for ranking only (detect=`nil`, rank=`auto`).

7 Conclusion

Most current approaches to lexical simplification (LS) are user-independent. This paper proposed the use of personalized models of complex word identification (CWI) to tailor LS systems to the vocabulary proficiency of the user. We presented the first study on the effect of personalized CWI in two steps of the LS pipeline: to detect which words require simplification, and to reject substitution candidates that are still too difficult for the user. We measured both the upper bounds of performance gains with an oracle CWI model, as well as the actual gains of a simple automatic CWI model that required only a 40-word training set per user, based on graded vocabulary lists.

Experimental results with oracle CWI demonstrated much room for improving LS systems through personalization. Further, systems that used the automatic CWI model consistently outperformed the user-independent baseline, by reducing both the number of unnecessary simplifications and the number of complex words in the output. The performance gains persisted regardless of whether the substitutions were gold or automatically generated, yielding improvement in precision and accuracy ranging from 4% to 13%. As CWI research produces higher-performing models, future LS systems can expect to derive even greater benefits from personalization.

Acknowledgements

This work was supported by the Innovation and Technology Fund (Ref: ITS/132/15) of the Innovation and Technology Commission, the Government of the Hong Kong Special Administrative Region; and by CityU Internal Funds for ITF Projects (no. 9678104). We thank Dr. Lis Pereira for assisting with the experiments.

References

John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying Text for Language-Impaired Readers. In *Proc. 9th EACL.*

Siobhan Devlin and John Tait. 1998. The Use of a Psycholinguistic Database in the Simplification of Text for Aphasic Readers. *Linguistic Databases*, pages 161–173.

Yo Ehara, Nobuyuki Shimizu, Takashi Ninomiya, and Hiroshi Nakagawa. 2010. Personalized Reading Support for Second-Language Web Documents by Collective Intelligence. In *Proc. 15th International Conference on Intelligent User Interfaces*, pages 51–60.

Yo Ehara, Issei Sato, Hidekazu Oiwa, and Hiroshi Nakagawa. 2012. Mining words in the minds of second language learners: learner-specific word difficulty. In *Proc. COLING*.

Yo Ehara, Yusuke Miyao, Hidekazu Oiwa, Issei Sato, and Hiroshi Nakagawa. 2014. Formalizing Word Sampling for Vocabulary Prediction as Graph-based Active Learning. In *Proc. EMNLP*.

Goran Glavaš and Sanja Štajner. 2015. Simplifying Lexical Simplification: Do We Need Simplified Corpora? In *Proc. ACL*.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a Lexical Simplifier Using Wikipedia. In *Proc. ACL*.

Tomoyuki Kajiwara, Hiroshi Matsumoto, and Kazuhide Yamamoto. 2013. Selecting Proper Lexical Paraphrase for Children. In *Proc. 25th Conference on Computational Linguistics and Speech Processing (ROCLING)*, pages 59–73.

Batia Laufer and Paul Nation. 1999. A Vocabulary-size Test of Controlled Productive Ability. *Language Testing*, 16(1):33–51.

John Lee and Chak Yan Yeung. 2018. Automatic Prediction of Vocabulary Knowledge for Learners of Chinese as a Foreign Language. In *Proc. International Conference on Natural Language and Speech Processing (ICNLSP)*.

Gustavo H. Paetzold and Lucia Specia. 2016a. Benchmarking Lexical Simplification Systems. In *Proc. LREC*.

Gustavo H. Paetzold and Lucia Specia. 2016b. SemEval 2016 Task 11: Complex Word Identification. In *Proc. 10th International Workshop on Semantic Evaluation (SemEval-2016)*.

Gustavo H. Paetzold and Lucia Specia. 2016c. Unsupervised Lexical Simplification for Non-native Speakers. In *Proc. AAAI*.

Gustavo H. Paetzold and Lucia Specia. 2017. Lexical Simplification with Neural Ranking. In *Proc. EACL*.

Matthew Shardlow. 2014. Out in the Open: Finding and Categorising Errors in the Lexical Simplification Pipeline. In *Proc. LREC*.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in Current Text Simplification Research: New Data Can Help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Seid Muhie Yimam, Sanja Stajner, Martin Riedl, and Chris Biemann. 2017. Multilingual and Cross-Lingual Complex Word Identification. In *Proc. Recent Advances in Natural Language Processing (RANLP)*.

Qing Zeng, Eunjung Kim, Jon Crowell, and Tony Tse. 2005. A Text Corpora-based Estimation of the Familiarity of Health Terminology. In J.L. Oliveira et al., editor, *ISBMDA 2005, LNBI 3745*, pages 184–192.

From Text to Lexicon: Bridging the Gap between Word Embeddings and Lexical Resources

Ilia Kuznetsov and **Iryna Gurevych**
Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science
Technische Universität Darmstadt
http://www.ukp.tu-darmstadt.de/

Abstract

Distributional word representations (often referred to as word embeddings) are omnipresent in modern NLP. Early work has focused on building representations for word types, and recent studies show that lemmatization and part of speech (POS) disambiguation of targets in isolation improve the performance of word embeddings on a range of downstream tasks. However, the reasons behind these improvements, the qualitative effects of these operations and the combined performance of lemmatized and POS disambiguated targets are less studied. This work aims to close this gap and puts previous findings into a general perspective. We examine the effect of lemmatization and POS typing on word embedding performance in a novel resource-based evaluation scenario, as well as on standard similarity benchmarks. We show that these two operations have complementary qualitative and vocabulary-level effects and are best used in combination. We find that the improvement is more pronounced for verbs and show how lemmatization and POS typing implicitly target some of the verb-specific issues. We claim that the observed improvement is a result of better conceptual alignment between word embeddings and lexical resources, stressing the need for conceptually plausible modeling of word embedding targets.

1 Introduction

Word embeddings learned from unlabeled corpora are one of the cornerstones of modern natural language processing. They offer important advantages over their sparse counterparts, allowing efficient computation and capturing a surprising amount of lexical information about words based solely on usage data.

Since precise word-related terminology is important for this paper, we introduce it early on. A **token** is a unique word occurrence within context. Tokens that are represented by the same sequence of characters belong to the same word **type**. A type signifies one or — in case of morphological ambiguity — several **word forms**. Word forms encode the grammatical information carried by the word and are therefore POS-specific. One of the word forms is considered the **base form** - or **lemma** of the word. All possible word forms of a word represent the **lexeme** of this word. From a semantic perspective, a word is assigned to a minimal sense-bearing **lexical unit** (LU). In general, multi-word and sub-word lexical units are possible, but in this paper we focus on single-word units. LUs are the reference point to the semantics of the word and might be used to describe further properties of the words within a specific **lexicon**, e.g. get assigned one or several senses, related to other LUs via lexical relations etc. Figure 1 illustrates these concepts and positions some of the relevant approaches w.r.t. the level they operate on.

In general, the training objective in the unsupervised word embedding setup (e.g. (Mikolov et al., 2013)) is to induce vector representations for **targets** based on their occurrences in an unlabeled reference corpus. For each occurrence of the target, **context** representation is extracted. The goal is to encode targets so that targets appearing in similar contexts are close in the resulting **vector space model** (VSM). We further refer to the set of targets in a given VSM as **target vocabulary** of this VSM.

The applications of word embeddings can be roughly grouped in two categories, **occurrence-based** and **vocabulary-based**: the former aims to classify tokens (e.g. parsing, tagging, coreference resolution), the

Proceedings of the 27th International Conference on Computational Linguistics, pages 233–244
Santa Fe, New Mexico, USA, August 20-26, 2018.

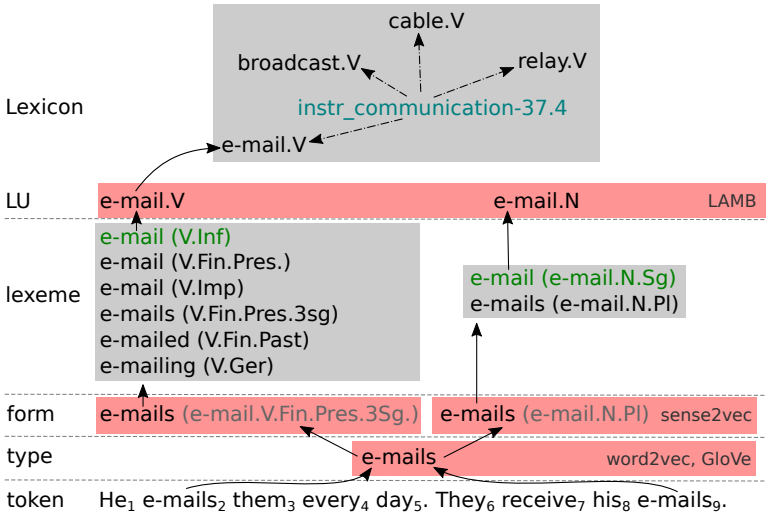


Figure 1: Hierarchy of word-related concepts for *emails*, with VerbNet as lexicon example.

latter aims to classify lexemes (e.g. word clustering, thesaurus construction, lexicon completion). Lexical resources mostly operate on lexeme or lexical unit level. This includes most of the similarity benchmarks (Finkelstein et al., 2001; Bruni et al., 2014; Hill et al., 2015; Gerz et al., 2016) that implicitly provide similarity scores for lexemes and not word types. Traditional word embeddings approaches (Mikolov et al., 2013; Pennington et al., 2014) induce representations on word type level. As our example in Figure 1 shows, this leads to a conceptual gap between the lexicon and the VSM, which has practical consequences. Consider the standard scenario when a type-based VSM is evaluated against a lexeme-based benchmark. One of the types contained in the VSM vocabulary corresponds to the base form (lemma), and the vector for the base word form is used for evaluation. This particular form, however, is selected based on *grammatical* considerations and, as we later demonstrate, is neither the most frequent nor the most representative in terms of contexts, nor the most unambiguous. As a result, (1) the contexts for a given lexical unit are under-represented, ignoring other word forms than the base form; (2) in case of ambiguity the same context-based representation is learned for several different lexemes sharing the same word type.

Recent studies demonstrate that even partially addressing these problems leads to improved performance. Ebert et al. (2016) introduce the lemma-based LAMB embeddings and show that lemmatization of the targets improves the results cross-linguistically on similarity benchmarks and in a WordNet-based evaluation scenario. In our scheme this represents a step up in the conceptual hierarchy but still leaves room for ambiguity on the LU level. Trask et al. (2015) experiment with POS-disambiguated targets, and also report improved performance on a variety of tasks.

While prior work shows that lemmatization and POS-typing of targets in isolation are beneficial for downstream tasks, it does not provide a detailed investigation on why it is the case and does not study the effects of combining the two preprocessing techniques. This paper aims to close this gap. We evaluate the effects of lemmatization and POS disambiguation separately and combined on similarity benchmarks, and further refine our results using a novel resource-based word class suggestion scenario which measures how well a VSM represents VerbNet (Schuler, 2005) and WordNet supersense (Ciaramita and Johnson, 2003) class membership. We find that POS-typing and lemmatization have complementary qualitative and vocabulary-level effects and are best used in combination. We observe that English verb similarity is harder to model and show that using lemmatized and disambiguated embeddings implicitly targets some of the verb-specific issues. In summary, the contributions of this paper are as follows:

- We suggest using lemmatized *and* POS disambiguated targets as a conceptually plausible alternative to type, word form and lemma-based VSMs;
- We introduce the suggestion-based evaluation scenario applicable to a wide range of lexical resources;
- We show that lemmatization and POS disambiguation improve both benchmark and resource-based performance by implicitly targeting some of the grammar-level issues.

2 Related work

The starting point for our study are the results from the lemma-based word embedding approach by Ebert et al. (2016), the POS-disambiguated *sense2vec* embeddings by Trask et al. (2015) and the dependency-based contexts by Levy and Goldberg (2014). Our primary focus is the effect of word embedding targets on vocabulary-based task performance. However, to put our work in context, we provide an extended overview of the issues related to traditional type-based approaches to VSM construction, along with relevant solutions.

Type-based VSMs (Mikolov et al., 2013; Pennington et al., 2014) have several limitations that have been extensively studied. They have **restricted vocabularies** and hence lack the ability to represent out-of-vocabulary words. Several recent approaches treat this issue by learning vector representations for sub-word units, e.g. *fastText* (Bojanowski et al., 2016) and *charagram* (Wieting et al., 2016).

Type-based VSMs do not abstract away from **word inflection**: different forms of the same word are assigned different representations in the VSM. Ebert et al. (2016) introduces lemmatized LAMB embeddings as a way to address this issue and shows that lemmatization is highly beneficial for word similarity and lexical modeling tasks even for morphologically poor English, while using *stems* doesn't lead to significant improvements. Their approach is similar to our `lemma-w2` setup (Section 3). Another recent attempt to address this problem is the study by Vulić et al. (2017b), which uses a small set of morphological rules to specialize the vector space bringing the forms of the same word closer together while setting the derivational antonyms further apart. We relate to this approach in Section 4.

Finally, type-based VSMs neglect the problem of **polysemy**: all senses of a word are encoded as a single entry in the VSM. This issue has been recently approached by multi-modal word embeddings (Athiwaratkun and Wilson, 2017) and Gaussian embeddings (Vilnis and McCallum, 2014). Partially disambiguating the source data via POS tagging has been employed in (Trask et al., 2015). Here, instead of constructing vectors for word forms, POS information is integrated into the vector space as well. This is similar to our `type.POS-w2` setup (Section 3) which, as we show, introduces additional sparsity and ignores morphological inflection.

An alternative line of research aims to learn embeddings for lexical units by using an external WSD tool to preprocess the corpus and applying standard word embedding machinery to induce distributed representations for lexical units, e.g. (Iacobacci et al., 2015; Flekova and Gurevych, 2016). Such approaches require an external WSD tool which introduces additional bias and might not be available for lower-resourced languages. Moreover, to query such VSMs it is necessary to either apply WSD to the input, or to align the inputs with the senses in some other way, which is not always feasible.

From the evaluation perspective, a popular method to assess the performance of a particular vector space model is **similarity benchmarking**. A similarity benchmark consists of word pairs along with human-assessed similarity scores, which are compared to cosine similarities returned by VSMs via rank correlation. Some common examples include WordSim-353 (Finkelstein et al., 2001) and MEN (Bruni et al., 2014). The recent SimLex-999 (Hill et al., 2015) and SimVerb-3500 (Gerz et al., 2016) particularly focus on word similarity as opposed to word relatedness (e.g. *bank* and *money* would be related, but not similar) and have been shown difficult for word embedding models to tackle.

Being attractive from the practical perspective, similarity benchmarks are expensive to create and do not provide detailed insights about the lexical properties encoded by the word embeddings. Motivated by that, **resource-based benchmarking** strategies have been suggested. (Ebert et al., 2016) represents WordNet as a graph and measures the correspondence of similarity ranking to the distances between query words in the graph via mean reciprocal rank. Vulić et al. (2017a) and Sun et al. (2008) apply a clustering algorithm to the input words and measure how well the clusters correspond to the word groupings in VerbNet via purity and collocation. Both methods have certain limitations, which we discuss and attempt to resolve via a novel general suggestion-based approach in Section 5.

3 General setup

Distributional word representations aim to encode word targets based on the context words they co-occur with. One popular general-purpose model for inducing such representations is skip-gram with negative

sampling (SGNS), exemplified by word2vec (Mikolov et al., 2013).

In the original SGNS targets and contexts belong to the same type-based vocabulary, but this is not required by the model. In this study we experiment with the following targets: `type` (*going*), `lemma` (*go*), `type.POS`[1] (*going.V*) and `lemma.POS` (*go.V*). Word embeddings demonstrate qualitative differences depending on context definition (Levy and Goldberg, 2014), and we additionally report the results on 2-word window-based (`w2`) and dependency-based (`dep-W`) contexts. To keep the evaluation setup simple, we only experiment with word type-based contexts.

We train 300-dimensional VSMs on a recent English Wikipedia dump, preprocessed using the Stanford Core NLP pipeline (Manning et al., 2014) with Universal Dependencies 1.0. The text is extracted using the *wikiextractor* module[2] with minor additional cleanup routines. Training the VSM is performed with the skip-gram based *word2vecf* implementation from (Levy and Goldberg, 2014) with default algorithm parameters.

4 Similarity Benchmarks

Following previous work, we first evaluate the effect of lemmatization and POS-typing on similarity benchmarks SimLex-999 (Hill et al., 2015) and SimVerb-3500 (Gerz et al., 2016). Both benchmarks provide POS information, which is required by POS-enriched VSMs. SimLex-999 contains nouns (60%), verbs (30%) and adjectives (10%); SimVerb-3500 is verbs-only. Table 1 summarizes the performance of the VSMs in question on similarity benchmarks.

Context: `w2`

	SimLex				SimVerb
target	N	V	A	all	V
type	.334	**.336**	**.518**	.348	.307
+ POS	.342	.323	.513	.350	.279
lemma	**.362**	.333	.497	**.351**	.400
+ POS	.354	**.336**	.504	.345	**.406**
* type	-	-	-	.339	.277
* type MFit-A	-	-	-	.385	-
* type MFit-AR	-	-	-	.439	.381
Context: `dep-W`					
type	.366	.365	.489	.362	.314
+ POS	.364	.351	.482	.359	.287
lemma	**.391**	.380	**.522**	**.379**	.401
+ POS	.384	**.388**	.480	.366	**.431**
* type	-	-	-	.376	.313
* type MFit-AR	-	-	-	.434	.418

Table 1: Benchmark performance, Spearman's ρ. SGNS results with * taken from (Vulić et al., 2017b). Best results per column (benchmark) annotated for our setup only.

Several observations can be made. Lemmatized targets generally perform better, with the boost being more pronounced on SimVerb. English verbs have richer morphology than other parts of speech and benefit more from lemmatization. Adding POS information benefits the SimVerb and SimLex verb performance, which can be attributed to the **coarse disambiguation** of verb-noun and verb-adjective homonyms. However, the `type.POS` targets show a considerable performance drop on SimVerb and SimLex verbs. This is due to **vocabulary fragmentation**, when the same word type is split into several entries based on the POS, e.g. *acts (110)* $\rightarrow$ *acts.V (80) + acts.N (30)*. As a result, some word types do not exceed the frequency threshold and are not included into the final VSM. Another way to see it

[1] We use coarse POS classes obtained by selecting the first character in the Penn POS tags assigned by the tagger

[2] `https://github.com/attardi/wikiextractor`

is that POS disambiguation increases the sparsity, which lemmatization, in turn, aims to reduce. Using dep-W contexts proves beneficial for both datasets since modeling the context via syntactic dependencies results in more similarity-driven (as opposed to relatedness-driven) word embeddings (Levy and Goldberg, 2014).

We provide the Morph-Fitting scores (Vulić et al., 2017b) as a state-of-the-art reference; a direct comparison is not possible due to the differences in the training data and information available to the models. This approach uses word type-based VSMs specialized via Morph-Fitting (MFit), which can be seen as an alternative to lemmatization. Morph-Fitting consists of two stages: the Attract (A) stage brings word forms of the same word closer in the VSM, while the Repel (R) stage sets the derivational antonyms further apart. Lemma grouping is similar to the Attract stage. However, as the comparison of MFit-A and -AR shows, a major part of the Morph-Fitting performance gain on SimLex comes from the derivational Repel stage[3], which is out of the scope of this paper.

While some properties of lemmatized and POS disambiguated embeddings are visible on the similarity benchmarks, the results are still inconclusive, and we proceed to a more detailed evaluation scenario.

5 Word Class Suggestion

5.1 Background

Similarity benchmarks serve as a standard evaluation tool for word embeddings, but provide little insights into the nature of relationships encoded by the word representations. A more fine-grained context-free evaluation strategy is to assess how well the relationships in a certain **lexical resource** are represented by the given VSM. Two recent approaches to achieve this are rank-based and clustering-based evaluation.

Rank-based evaluation treats the lexical resource as a graph with entries as nodes and lexical relations as edges, and estimates how well the similarities between the VSM targets represent the distances in this graph via mean reciprocal rank (MRR), e.g. Ebert et al. (2016) use WordNet (Miller, 1995). It requires the target lexical resource to have a dense linked structure which might be not present.

Clustering-based evaluation, e.g. Vulić et al. (2017a) utilize the VSM to produce target clusters which are compared to the groupings from the lexical resource via collocation and purity. This approach only requires lexical entries to be grouped into classes. However, it doesn't model word ambiguity: a word can only be assigned to a single cluster. Moreover, it introduces an additional level of complexity (clustering algorithm and its parameters) which might obscure the VSM performance details.

In this paper we propose an alternative, suggestion-based evaluation approach: we use the source VSM directly to generate **word class suggestions** (WCS) for a given input term. Many lexical resources group words into intersecting word classes, providing a compact way to describe word properties on the class level. For example, in VerbNet (Schuler, 2005) verbs can belong to one or more Levin classes (Levin, 1993) based on their syntactic behavior, FrameNet (Baker et al., 1998) groups its entries by the semantic frames they can evoke, and WordNet (Miller, 1995) provides coarse-grained supersense groupings. Suggestion-based evaluation can be seen as flexible alternative to clustering-based evaluation, which intrinsically takes ambiguity into account and does not require an additional clustering layer. The following section describes the suggestion-based evaluation in more detail[4].

5.2 Task formulation

Abstracting away from the resource specifics, a **lexicon** L defines a mapping from a set of **members** $m_1, m_2, ...m_i \in M$ to a set of **word classes** $c_1, c_2...c_j \in C$. We further denote the set of classes available for a member m as $L(m)$ and the set of members for a given class c as $L'(c)$. Given a **query** q, our task is to provide a set of word class suggestions $WCS_L(q) = \{c_a, c_b...\} \in C$. Note that we aim to predict all *potential* classes for a member given its vector representation on vocabulary level, independent of context.

[3] See also (Vulić et al., 2017b), Table 5.

[4] Our implementation is available at https://github.com/UKPLab/coling2018-wcs

5.3 Suggestion strategies

Given an input target w, the source vector space VSM provides its vector representation $VSM(w)$. We use cosine similarity between targets $sim(w_i, w_j)$ to rank the word classes.

A lexical resource might already contain a substantial number of members, and a natural solution for word class suggestion is to find the **prototype** member m_{proto} closest to the query q in the VSM, and use its classes as suggestions. This scenario mimics human interaction with the lexicon. If $q \in M$, this is equivalent to a lexicon lookup. More formally,

$$m_{proto} = argmax_{m \in M} sim(q, m) \qquad score_{proto}(q, c, L) = \begin{cases} 1, & \text{if } c \in L(m_{proto}) \\ 0, & \text{otherwise} \end{cases}$$

The prototype strategy per se is sensitive to the coverage gaps in the lexicon and inconsistencies in the VSM. We generalize it by ranking *each* word class $c \in C$ using the similarity between the query q and its closest member in c:

$$score_{top}(q, c, L) = max_{m \in L'(c)} sim(q, m)$$

This is equivalent to performing the prototype search on each word class, and scoring each class by the closest prototype among its members, given the query. We use this generalized strategy for our WCS experiments throughout this paper. The output of the model $WCS_L(q, VSM)$ is a set of classes ranked using the input VSM, as illustrated by Table 2.

query	top classes / prototypes	query	top classes / prototypes
dog→	animal (*cat*), food (*rabbit*) ...	idea→	cognition (*concept*), artifact (*notion*) ...
crane→	artifact (*derrick*), animal (*skimmer*) ...	bug→	animal (*worm*), state (*flaw*) ...

Table 2: WCS output for WordNet supersenses

5.4 Evaluation procedure

For each member m in the lexicon in turn, we remove it from the lexicon, resulting in a reduced lexicon L_{-m}. We aim to reconstruct its classes using the suggestion algorithm and the remaining mappings.

The performance is measured via precision (P) and recall (R) at rank k with the list of original classes for a given lexical unit serving as *gold* reference. Formally, given m we compute the ranking $WCS_{L_{-m}}(m, VSM)$. Let S_k be the set of classes suggested by the system up to the rank k, and T be the true set of classes for a given member in the original lexicon. Then

$$P_{@k} = \frac{|S_k \cap T|}{|S_k|} \qquad R_{@k} = \frac{|S_k \cap T|}{|T|}$$

To get a single score, we average individual members' $P_{@k}$ and $R_{@k}$ for each value of k, resulting in scores $\overline{P}_{@k}$ and $\overline{R}_{@k}$. F-measure might be then calculated using the standard formula

$$F_{@k} = \frac{2\overline{P}_{@k}\overline{R}_{@k}}{\overline{P}_{@k} + \overline{R}_{@k}}$$

Upper bound Since the number of gold classes is not known in advance, the evaluation is always performed on k ranks, which leads to a resource-specific upper bound on P. For example, if a member only has one class, the ranked list of 10 suggestions will inevitably show lower precision. When the member set is not fully covered by the VSM target vocabulary, additional upper bound on R applies.

6 Experiment

6.1 Resources

We use two lexical resources for suggestion-based evaluation: VerbNet 3.3 (Schuler, 2005) and WordNet 3.1 (Miller, 1995). VerbNet groups verbs into classes[5] so that verbs in the same class share syntactic behavior, predicate semantics, semantic roles and restrictions imposed on these roles. For example, the verb *buy* belongs to the class *get-13.5.1*. The class specifies a set of available roles, e.g. an animate `Agent` (buyer), an `Asset` (price paid) and a `Theme` (thing bought), and lists available syntactic constructions, e.g. the `Asset V Theme` construction (*"$50 won't buy a dress"*). A verb might appear in several classes, indicating different verb senses. For example, the verb *hit* allows several readings: as *hurt* (*John hit his leg*), as *throw* (*John hit Mary the ball*) and as *bump* (*The cart hit against the wall*). VerbNet has been successfully used to support semantic role labeling (Giuglea and Moschitti, 2006), information extraction (Mausam et al., 2012) and semantic parsing (Shi and Mihalcea, 2005).

WordNet, besides providing a dense network of lexical relations, groups its entries into coarse-grained supersense classes, e.g. `noun.animal` (*aardvark*, *koala*), `noun.location` (*park*, *senegal*), `noun.time` (*forties*, *nanosecond*). WordNet supersense tags have been applied to a range of downstream tasks, e.g. metaphor identification and sentiment polarity classification (Flekova and Gurevych, 2016). WordNet differs from VerbNet in terms of granularity, member-class distribution and part of speech coverage, and allows us to estimate VSM performance on nominal as well as verbal supersenses, which we evaluate separately. Table 3 provides the statistics for the resources. We henceforth denote VerbNet as `VN`, WordNet nominal supersense lexicon as `WN-N` and WordNet verbal supersense lexicon as `WN-V`.

	classes	members	ambig	%ambig
VN	329	4 569	1 366	30%
WN-V	15	8 702	3 326	38%
WN-N	26	57 616	9 907	17%

Table 3: Lexicon statistics, single-word members

6.2 Results

We use the suggestion-based evaluation to examine the effect of lemmatization and POS-disambiguation. We first analyze the coverage of the VSMs in question with respect to the lexica at hand, see Table 4. For brevity we only report coverage on `w2` contexts[6].

target	VN	WN-V	WN-N
type	81	66	47
+POS	54	39	43
lemma	88	76	53
+POS	79	63	50
shared	54	39	41

Table 4: Lexicon member coverage (%)

Coverage analysis on lexica confirms our previous observations: lemmatization allows more targets to exceed the SGNS frequency threshold, which results in consistently better coverage. POS-disambiguation, in turn, fragments the vocabulary and consistently reduces the coverage with the effect being less pronounced for lemmatized targets. `WN-N` shows low coverage containing many low-frequency members. Due to significant discrepancies in VSM coverage, we conduct our experiments on **shared vocabulary**, only including members found in all VSMs to analyze the qualitative differences between VSMs.

[5]For simplicity in this study we ignore subclass divisions.

[6]We have observed slight coverage differences for `dep` contexts, and attribute this to the context vocabulary fragmentation caused by dependency typing of the contexts, similar to the POS fragmentation effect described earlier.

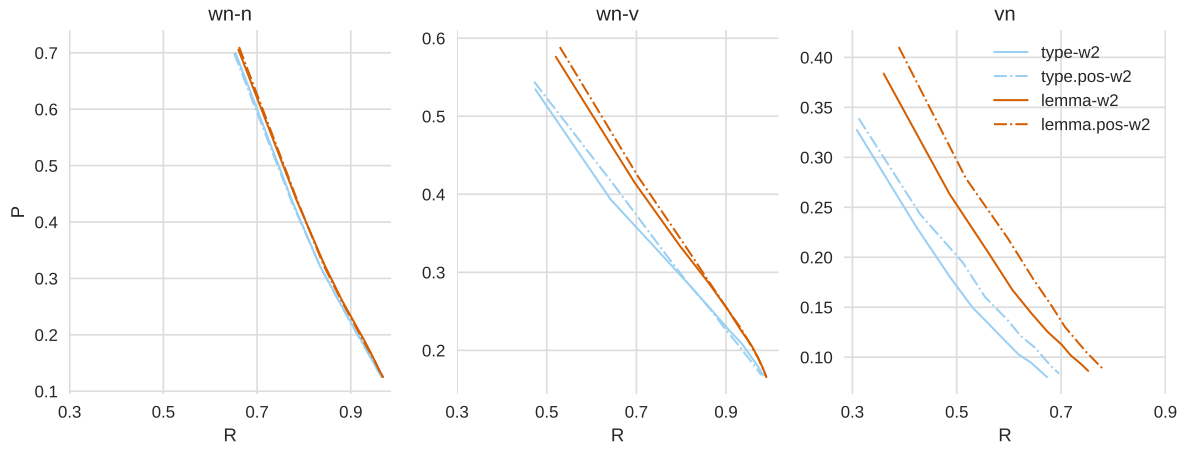


Figure 2: WCS PR-curve, shared vocabulary, w2 contexts.

We treat the cutoff rank k as a parameter that specifies the Precision-Recall trade-off. As Figure 2 demonstrates, lemmatized targets consistently outperform their word form-based counterparts on the WCS task. The magnitude of improvements varies between resources: verb-based WN-V and VN benefit more from lemmatization, and VN gains most from POS-disambiguation. This aligns with the similarity benchmarking results where the verb-based SimVerb benefits more from addition of lemma and POS information.

	WN-N			WN-V			VN		
	P	R	F	P	R	F	P	R	F
Context: w2									
type	.700	.654	.676	.535	.474	.503	.327	.309	.318
+POS	.699	.651	.674	.544	.472	.505	.339	.312	.325
lemma	.706	.660	.682	.576	.520	.547	.384	.360	.371
+POS	**.710**	**.662**	**.685**	**.589**	**.529**	**.557**	**.410**	**.389**	**.399**
Context: dep									
type	.712	.661	.686	.545	.457	.497	.324	.296	.310
+POS	.715	.659	.686	.560	.464	.508	.349	.320	.334
lemma	**.725**	**.668**	**.696**	.591	.512	.548	.408	.371	.388
+POS	.722	.666	.693	**.609**	**.527**	**.565**	**.412**	**.381**	**.396**

Table 5: WCS performance, shared vocabulary, $k = 1$. Best results across VSMs in bold.

Table 5 provides exact scores for reference. Note that the shared vocabulary setup puts the type and type.POS VSMs at advantage since it eliminates the effect of low coverage. Still, lemma-based targets significantly[7] ($p \leq .005$) outperform type-based targets in terms of F-measure in all cases. For window-based w2 contexts POS disambiguation yields significantly better F scores on lemmatized targets for VN ($p \leq .005$) with borderline significance for WN-N and WN-V ($p \approx .05$). When dependency-based dep contexts are used, the effect of POS disambiguation is only statistically significant on type targets for VN ($p \leq .005$) and on lemma-based targets for WN-V ($p \leq .005$). We attribute this to the fact that dependency relations used by dep contexts are highly POS-specific, reducing the effect of explicit disambiguation. Lemma-based targets without POS disambiguation perform best on WN-N when dependency-based contexts are used; however, the difference to lemmatized *and* disambiguated targets is not statistically significant ($p > .1$).

Our results are in line with the previous observations (Gerz et al., 2016) in that the verb similarity is

[7]Wilcoxon signed-rank test over individual lexicon members' F scores

	SimLex			SimVerb	WN-N	WN-V	VN
	N	V	A	V	N	V	V
base	.80	.26	1.0	.24	.86	.21	.22
avg #POS	1.08	1.01	1.39	1.50	1.15	1.37	1.42
single POS	.93	.99	.62	.51	.85	.65	.59

Table 6: Base form ratio and available POS averaged over members; % members with single POS.

hard to capture with standard VSMs. To investigate why verbs benefit most from lemmatization and POS disambiguation, we analyze some relevant statistics based on our Wikipedia data. Table 6 shows the ratio of base form to total occurrences in our corpus[8]. As we can see, the base form (lemma) is by far not the dominating form for verbs. Practically this means that for our resources verbal `type` targets have direct access to only 20-25% of the corpus occurrence data on average. Individual verbs and nouns also differ in terms of preferred word forms, which introduces additional bias into the evaluation. This effect is countered by lemmatization.

The second important difference between the noun- and verb-based lexica is the number of POS available for the lexicon members' lemmas. As Table 6 further demonstrates, nouns are less ambiguous in terms of POS: for example, `VN` member lemmas appear with 1.42 distinct POS categories on average, compared to 1.15 categories for `WN-N`. Individual members might differ in terms of POS frequency distribution, again biasing the evaluation. One regular phenomenon accountable for this is the *verbification* of nouns and adjectives, when a verbal form is constructed without adding any derivational markers. While these derivations might to some extent preserve the similarities between words (e.g. *e-mail.N→e-mail.V* is similar to *fax.N→fax.V*), many cases are less transparent and benefit from POS separation (e.g. the meaning shift in *air.N→air.V* is different from *water.N→water.V*). One exception is the verb subset of SimLex which has a particularly low POS ambiguity.

7 Future work

Lexical unit-based modeling. In this work we have shown that lemmatization and subsequent POS disambiguation benefit both benchmark- and resource-based performance of word embeddings. While verb semantics is notoriously hard to pinpoint per se, we show that modeling verbs via type-based distributed representations also meets grammar-related challenges which can be partially addressed with lemmatization and POS-disambiguation of the inputs. From the conceptual perspective, lemmatized and POS-tagged targets can be seen as another step towards conceptually plausible lexical unit-based modeling of word usage. In this work we focused on single-word entities, and analyzing the effect of including multi-word expressions into the VSM vocabulary is an important direction for future studies.

Word embedding methods. To ensure fair comparison and to keep our evaluation setup compact, we have consciously restricted the scope of the study to a single word embedding model (SGNS), single context size (`w2`) and a single parameter set (*word2vecf* SGNS default). Our results could be further validated by experimenting with alternative context definitions and word embedding models, e.g. GloVe (Pennington et al., 2014) and CBOW (Mikolov et al., 2013). Experiments on character-based models, e.g. *fasttext* (Bojanowski et al., 2016) or *charagram* (Wieting et al., 2016) could be another interesting extension to our work. However, it is not clear how to integrate lemma and POS information with character-based representations in an elegant way.

Cross-linguistic studies. POS tagging and lemmatization are general and well-defined language-independent operations. We have focused on English and have shown that POS-typing and lemmatization implicitly target several grammar-level issues in the context of word embeddings. (Ebert et al., 2016) demonstrate that the improvements from lemmatization hold in a cross-lingual setup. While we believe our results to generally hold cross-linguistically, the relative contribution of POS disambiguation and

[8]We use all lemmas that appear more than 100 times in the corpus; to smooth the effect of tagging errors we only count POS that appear with the target lemma in more than 10% of total lemma occurrences. All statistics averaged over individual lemmas.

lemmatization will inevitably depend on the grammatical properties of the language, constituting another topic for further research.

Suggestion-based evaluation. Our evaluation procedure has clear advantages: it is word class-based, polysemy-aware, does not introduce additional complexity and does not require an annotated corpus. It is resource-agnostic and only requires the target lexical resource to group words into classes. However, several issues must be addressed before it can be used on large scale. Our leave-one-out scenario excludes *singleton classes*, i.e. classes that only have one member. This issue will become less severe with resource coverage increasing over time. The evaluation depends on resource and VSM vocabulary coverage, and for qualitative comparison between VSMs vocabulary intersection should always be taken into account. Alternative suggestion strategies might be explored, e.g. averaging among class members instead of selecting the closest prototype.

Application scenarios. We have introduced word class suggestion as an evaluation benchmark for word embeddings. However, the WCS output might be used in vocabulary-based **application scenarios**, e.g. as annotation study support in cases when the lexicon is available, but the usage corpora are scarce; as a lexicographer tool for finding the gaps in existing lexica; and as a source for context-independent unknown word class candidates in a word sense disambiguation setup.

8 Conclusion

This paper has investigated the effect of lemmatized and POS-disambiguated targets on vector space model performance. We have shown that these preprocessing operations lead to improved performance on the standard benchmarks as well as in a novel word class suggestion scenario. Our results stress the need for more conceptually sound distributional models which align better to lexical resources. This would help to abstract away from the grammar-level issues and allow to further focus on bridging the gap between distributional models and lexical resources on the semantic level.

Acknowledgements

This work has been supported by the German Research Foundation as part of the QA-EduInf project (grant GU 798/18-1 and grant RI 803/12-1) and by the FAZIT Stiftung. We would like to thank the anonymous reviewers for useful comments and future research suggestions.

References

Ben Athiwaratkun and Andrew Wilson. 2017. Multimodal word distributions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1645–1656, Vancouver, Canada, July. ACL.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, pages 86–90, Stroudsburg, PA, USA. ACL.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Elia Bruni, Nam Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49(1):1–47, January.

Massimiliano Ciaramita and Mark Johnson. 2003. Supersense tagging of unknown nouns in wordnet. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 168–175, Stroudsburg, PA, USA. ACL.

Sebastian Ebert, Thomas Müller, and Hinrich Schütze. 2016. Lamb: A good shepherd of morphologically rich languages. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, USA, November. ACL.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Lucie Flekova and Iryna Gurevych. 2016. Supersense embeddings: A unified model for supersense interpretation, prediction and utilization. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, volume 1, pages 2029–2041. ACL.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simverb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2173–2182. ACL.

Ana-Maria Giuglea and Alessandro Moschitti. 2006. Semantic role labeling via framenet, verbnet and propbank. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 929–936, Stroudsburg, PA, USA. ACL.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with genuine similarity estimation. *Computational Linguistics*, 41(4):665–695, December.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Sensembed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015*, pages 95–105. ACL.

Beth Levin. 1993. *English verb classes and alternations : a preliminary investigation*. University of Chicago Press, Chicago, IL.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland, June. ACL.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60. ACL.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534, Stroudsburg, PA, USA. ACL.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, Doha, Qatar. ACL.

Karin Kipper Schuler. 2005. *Verbnet: A Broad-coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA.

Lei Shi and Rada Mihalcea. 2005. Putting pieces together: Combining framenet, verbnet and wordnet for robust semantic parsing. In *Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing'05, pages 100–111, Mexico City, Mexico. Springer-Verlag.

Lin Sun, Anna Korhonen, and Yuval Krymolowski. 2008. Verb class discovery from rich syntactic data. In *Proceedings of the 9th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing'08, pages 16–27, Haifa, Israel. Springer-Verlag.

Andrew Trask, Phil Michalak, and John Liu. 2015. sense2vec-a fast and accurate method for word sense disambiguation in neural word embeddings. *arXiv preprint arXiv:1511.06388*.

Luke Vilnis and Andrew McCallum. 2014. Word representations via gaussian embedding. *arXiv preprint arXiv:1412.6623*.

Ivan Vulić, Nikola Mrkšić, and Anna Korhonen. 2017a. Cross-lingual induction and transfer of verb classes based on word vector space specialisation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2546–2558, Copenhagen, Denmark, September. ACL.

Ivan Vulić, Nikola Mrkšić, Roi Reichart, Diarmuid Ó Séaghdha, Steve Young, and Anna Korhonen. 2017b. Morph-fitting: Fine-tuning word vector spaces with simple language-specific rules. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 56–68, Vancouver, Canada, July. ACL.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515, Austin, Texas, US, November. ACL.

Lexi: A tool for adaptive, personalized text simplification

Joachim Bingel[1] **Gustavo H. Paetzold**[2] **Anders Søgaard**[1]
[1] Department of Computer Science, University of Copenhagen, Denmark
[2] Federal University of Technology - Paraná, Brazil
{bingel,soegaard}@di.ku.dk, ghpaetzold@utfpr.edu.br

Abstract

Most previous research in text simplification has aimed to develop generic solutions, assuming very homogeneous target audiences with consistent intra-group simplification needs. We argue that this assumption does not hold, and that instead we need to develop simplification systems that adapt to the individual needs of specific users. As a first step towards personalized simplification, we propose a framework for adaptive lexical simplification and introduce Lexi, a free open-source and easily extensible tool for adaptive, personalized text simplification. Lexi is easily installed as a browser extension, enabling easy access to the service for its users.

1 Introduction

Many a research paper on text simplification starts out by sketching the problem of text simplification as rewriting a text such that it becomes easier to read, changing or removing as little of its informational content as possible (Zhu et al., 2010; Coster and Kauchak, 2011; De Belder and Moens, 2010; Paetzold and Specia, 2015; Bingel and Søgaard, 2016). Such a statement may describe the essence of simplification as a research task, but it hides the fact that it is not always easy to decide what is easy for a particular user. This paper discusses why we need custom-tailored simplifications for individual users, and argues that previous research on non-adaptive text simplification has been too generic to unfold the full potential of text simplification.

Even when limiting ourselves to lexical substitution, i.e. the task of reducing the complexity of a document by replacing difficult words with easier-to-read synonyms, we see plenty of evidence that, for instance, dyslexics are highly individual in what material is deemed easy and complex (Ziegler et al., 2008). Lexi, which we introduce in this paper, is a free, open-source and easily extensible tool for adaptively learning what items specific users find difficult, using this information to provide better (lexical) simplification. Our system initially serves Danish, but is easily extended to further languages. For surveys of text simplification, including resources across languages, see Siddharthan (2014), Shardlow (2014b) and Collins-Thompson (2014).

1.1 There is no one-size-fits-all solution to text simplification

Text simplification is a diverse task, or perhaps rather a family of tasks, with a number of different target audiences that different papers and research projects have focused on. Among the most prominent target audiences are foreign language learners, for whom various approaches to simplifying text have been pursued, often focusing on lexical (Tweissi, 1998) but also sentence-level simplification (Liu and Matsumoto, 2016). Other notable groups that have been specifically targeted in text simplification research include dyslexics (Rello et al., 2013), and the aphasic (Carroll et al., 1998), for whom particularly long words and sentences, but also certain surface forms such as specific character combinations, may pose difficulties. People on the autism spectrum have also been addressed, with the focus lying on reducing the amount of figurative expressions in a text or reducing syntactic complexity (Evans et al., 2014). Reading beginners (both children and adults) are another group with very particular needs, and text simplification

Proceedings of the 27th International Conference on Computational Linguistics, pages 245–258
Santa Fe, New Mexico, USA, August 20-26, 2018.

research has tried to provide this group with methods to reduce the amount of high-register language and non-frequent words (De Belder and Moens, 2010).

Evidently, each target group has its own simplification needs, and there is considerable variation as to how well the specifics of what makes a text difficult is defined for each group and simplification strategy. While difficult items in a text may be identified more easily and generally for problems such as resolving pronoun reference, questions such as what makes a French word difficult for a native speaker of Japanese, or what dyslexic children consider a difficult character combination or an overly long sentence, are much harder to answer. Nevertheless, there is a vast body of work (Yatskar et al., 2010; Biran et al., 2011; Horn et al., 2014) that ventures to build very general-purpose simplification models from simplification corpora such as the Simple English Wikipedia corpus (Coster and Kauchak, 2011), which has been edited by amateurs without explicit regard to a specific audience, and with rather vague guidelines as to what constitutes difficult or simple language.

Other work in simplification attempts to answer the above questions by inducing models from specifically compiled datasets, which for instance may have been collected by surveying specific target groups and asking them to indicate difficult material in a text. Yet even those approaches often cannot live up to the real challenges in simplification, seeing that we find very heterogeneous simplification needs also within target groups. Foreign language learners with different linguistic backgrounds (pertaining both to their native and second languages) will find very different aspects of the same foreign language difficult. Young readers in different school grades will quickly advance their reading habits and skills, and also within the same class or age reading levels may differ greatly. Likewise, people with autism exhibit very different manifestations of the type and degree of their condition (Alexander et al., 2016), also with respect to reading (Evans et al., 2014), just as there exist many different forms of cognitive impairments affecting literacy, including many different forms of dyslexia (Watson and Goldgar, 1988; Bakker, 1992; Ziegler et al., 2008). In fact, while there is a relatively strong agreement on the existence of some typologies of dyslexia or autism, specific typologies that have been proposed are heavily debated, such that it would not even be straightforward to create simplification tools for specific subtypes of these conditions.

From this it becomes apparent that in order to build simplification systems that truly help specific individuals, those systems have to be personalized or personalizable. Further, due to the frequent lack of insight into what an individual's specific reading problems are (and because any introspection is difficult to verify), such systems need to be able to learn themselves what those individual challenges are, and ultimately adapt to those.

1.2 Obtaining individual data

In order to learn specific reading challenges for an individual person, a simplification system needs individual data for this person, from which a personalized model can then be induced. This brings up the question of how best to obtain such data. A straightforward approach would be to ask each individual to provide ratings for some number of stimuli as they start using a simplification system. However, this would pose a relatively unnatural reading scenario, which might introduce a certain bias in the data and thus distort the induced model. Further, it might create a dissatisfying user experience, and users might not be willing to invest much time into such a calibration phase, especially when they perceive reading as a particularly strenuous activity. Yet perhaps most importantly, the model will not necessarily be well-adapted to the specific domains and genres that a specific user typically consumes text from.

As an alternative, we propose to collect data as the system is used, and to continuously update the system with feedback it collects from the user. In this way, the system can base its model on exactly those text types the user consumes. We discuss how feedback can be incorporated into a system in Section 3 and provide details on how this is implemented in our proposed system in Section 4.

1.3 Contributions

We present Lexi, an open source and easily extensible tool for adaptive, personalized text simplification. Lexi is based on an adaptive framework for lexical simplification that we also describe in this paper. This framework incorporates feedback from users, updating personalized simplification models such as to

meet their individual simplification needs. Lexi is made publicly available under a CC-BY-NC license[1] at https://www.readwithlexi.net.

2 Related Work

Perhaps the earliest contribution that focuses on on-demand lexical simplification is the work of Devlin and Unthank (2006), who present HAPPI, a web platform that allows users to request simplified versions of words, as well as other "memory jogging" pieces of information, such as related images.

Another example is the work of Azab et al. (2015), who present a web platform that allows users to select words they do not comprehend, then presents them with synonyms in order to facilitate comprehension. Notice that their approach does not simplify the selected complex words directly, it simply shows semantically equivalent alternatives that could be within the vocabulary known by the user.

The recent work of Paetzold and Specia (2016a) describes Anita, yet another web platform of this kind. It allows users to select complex words and then request a simplified version, related images, synonyms, definitions and translations. Paetzold and Specia (2016a) claim that their approach outputs customized simplifications depending on the user's profile, and evolves as users provide feedback on the output produced. However, they provide no details of the approach they use to do so, nor do they present any results showcasing its effectiveness.

Therefore not counting Paetzold and Specia (2016a) as work in *personalized* simplification, we are not aware of any previous approaches that address this. We further refer to related work on specific aspects of text simplification as they become relevant in the course of this paper.

3 Adaptive text simplification

As we mapped out in the introduction, we devise a simplification system that continuously learns from user feedback and adapts to the user's simplification needs. This section discusses how such feedback can be incorporated into a *lexical simplification* model via online learning, and where in the lexical simplification pipeline it is sensible to implement adaptivity.

3.1 Adaptivity in the lexical simplification pipeline

Lexical simplification, i.e. replacing single words with simpler synonyms, classically employs a pipeline approach illustrated in Figure 1 (Shardlow, 2014a; Paetzold and Specia, 2015). This pipeline consists of a four-step process, the first step of which is to identify simplification targets, i.e. words that the model believes will pose a difficulty for the user. This step is called *Complex Word Identification* (CWI) and has received a great deal of attention in the community, including two shared tasks (Paetzold and Specia, 2016b; Yimam et al., 2018). In a second step, known as *Substitution Generation*, synonyms are retrieved as candidate replacements for the target These are then filtered to match the context, resolving word sense ambiguities or stylistic mismatches, in *Substitution Selection*. Finally, those filtered candidate are ranked in order of simplicity in what is known as *Substitution Ranking* (SR).

Out of these four steps, we consider CWI and SR as the most natural ones to make adaptive, whereas generation and selecting candidates can be regarded as relatively independent from a specific user. In order to implement adaptivity, we propose to make use of online learning methods as discussed below and, departing from a seed model, train and maintain *user-specific models* as we collect feedback.

3.1.1 Adaptive CWI

Complex Word Identification is usually approached as a binary classification task, where the goal is to decide for some word *in context* whether or not it poses a difficulty to a reader. Existing datasets, for instance the ones used at previous CWI shared tasks (Paetzold and Specia, 2016b; Yimam et al., 2018), therefore provide a sentence and a target word (or multi-word expression) together with a binary label.

A model trained on this data with a learning algorithm based on gradient descent on t examples can now easily integrate newly collected data points into its parameters θ using an update rule such as

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_{\theta^{(t)}} J(\theta^{(t)}; x, y), \quad (1)$$

[1] https://creativecommons.org/licenses/by-nc/4.0/

where x is a representation of a target word in context and y is a binary complexity label we receive from user feedback. As an alternative to gradient descent based algorithms, we can use other online learning models, e.g. the Perceptron algorithm. CWI datasets are typically not very large (between 2,500 and 5,500 positive examples per dataset in the mentioned shared tasks), such that data points sampled from users can quickly have an impact on a generic base model.[2]

3.1.2 Adaptive Substitution Ranking

Substitution Ranking has received relatively little attention in the community compared to CWI. Most lexical simplifiers rank candidates using unsupervised approaches. The earliest example is the approach of Carroll et al. (1998), who rank candidates according to their Kucera-Francis coefficients, which are calculated based on frequencies extracted from the Brown corpus (Rudell, 1993). Other unsupervised approaches, such as those of Ligozat et al. (2012) and Glavaš and Štajner (2015), go a step further and use metrics that incorporate multiple aspects of word complexity, including context-aware features such as n-gram frequencies and language model probabilities. But even though unsupervised rankers perform well in the task, they are incapable of learning from data, which makes them unsuitable for adaptive SR.

Our approach to adaptive SR is similar to our approach to adaptive CWI, namely to train an initial model over manually produced simplicity rankings, then continuously update them with new knowledge as Lexi users provide feedback on the simplifications they receive. The feedback in this scenario is composed of a complex word in context, a simplification produced by Lexi, and a binary rank provided by the user determining which word (complex or simplification) makes the sentence easier to understand. For that purpose, we need a supervised model that (i) supports online learning so that it can be efficiently updated after each session, and (ii) can learn from binary ranks.

Paetzold and Specia (2017) offer some intuition on how this can be done. They exploit the fact that one can decompose a sequence of elements $\{e_1, e_2, ..., e_n\}$ with ranks $\{r_1, r_2, ..., r_n\}$ into a matrix $m \in \mathbb{R}^{n \times n}$, such that $m(i,j) = f(r_i, r_j)$, and function $f(r_i, r_j)$ estimates a value that describes the relationship between the ranks of elements e_i and e_j. For example, f could be described as:

$$f(r_i, r_j) \begin{cases} 1 \text{ if } r_i < r_j \\ -1 \text{ if } r_i > r_j \\ 0 \text{ otherwise} \end{cases} \quad (2)$$

The ranker of Paetzold and Specia (2017) uses a deep multi-layer perceptron that predicts each value of m individually. It takes as input feature representations of e_i and e_j, and produces a function f similar to the one depicted in Equation 2. Their approach would be perfectly capable of learning from the feedback produced by Lexi users, but it would be very difficult to train it through online learning, given that deep multi-layer perceptrons are characterized by a large number of parameters that are costly to optimize in an on-demand basis. We instead propose to employ an online learning model that has fewer parameters, e.g. logistic regression.

4 Implementation

Lexi consists of a client-side frontend and a server-side backend that communicate with each other via a RESTful API (Fielding, 2000), exchanging requests and responses as described further in 4.3. The client-server architecture allows for easy portability of the software to users, minimizing user-side installation efforts, hardware usage and dependencies on other libraries. It also centralizes the simplification engine, such that amendments to and maintenance of the latter need only be implemented on the server side.

Lexi is currently limited to performing lexical simplification. Note, however, that this is merely a limitation of the backend system, which only implements a lexical simplification system for now. From the frontend perspective, however, there are no limitations as to the nature and length of the simplified items in a text, and extending Lexi to support higher-level modes of simplification simply amounts to

[2] An alternative to traditional, one-size-fits-all approaches has recently been proposed by Bingel et al. (2018), who use eye-tracking measures to induce personalized models to predict misreadings in children with reading difficulties.

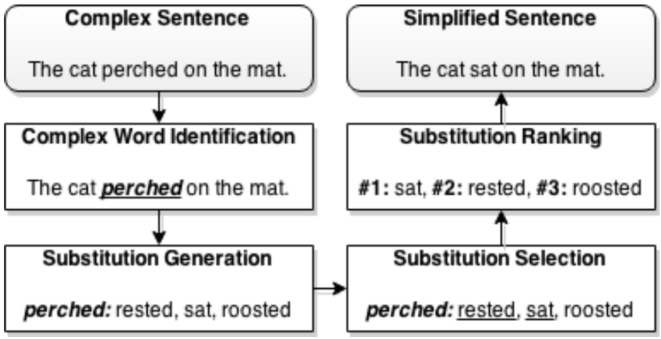


Figure 1: Lexical simplification pipeline as identified by Paetzold and Specia (2015). The simplification workflow consists of identifying simplification targets, i.e. words that pose a challenge to the reader. In the generation step, possible alternatives for each target are retrieved, which are then filtered in the selection step, eliminating words that do not fit the context. In the ranking step, the system finally orders the candidates by simplicity.

implementing a backend system supporting this.[3]

We initially focus on lexical simplification for a number of reasons: (i) We have existing baseline models that we expect to work well in a real-world setting. (ii) Given a relatively small number of parameters in those models, we expect fast adaptation to individual users from relatively little feedback. (iii) Compared to other forms of simplification, lexical simplification needs to make a selection from a relatively limited search space that is still reasonably diverse, such that we expect personalized models to make a difference more easily.

4.1 Frontend

Lexi's frontend is implemented in JavaScript and jQuery under the Mozilla WebExtension framework, supported by most modern browsers.[4] WebExtensions employ *content scripts* to modify a webpage upon certain specified events, for instance a click on some page element. The remainder of this section describes Lexi's basic usage as the user registers an account and asks the system for simplifications, thereby illustrating the user interface and sketching the inner workings of the frontend.

4.1.1 User log-in and registration

Upon installation of the Lexi extension in the browser, the user is prompted to register an account, providing an email address as well as basic demographic information (year of birth and educational level, see Figure 2). This information is sent to the backend using its registration endpoint (see Table 1). If the user has previously created an account and simply reinstalled the extension, they may also just provide their email address to keep using their existing profile. The user's email address is stored locally in the browser, where it is kept until the browser storage is cleared or the extension is uninstalled.

4.1.2 Simplification requests and display

Whenever the user visits a webpage, the extension injects an event listener into the page, which triggers upon the selection of some text and offers the user to simplify the selected content in the form of a small button that is displayed just above the selection. When this button is clicked, the extension retrieves the user's email address from the browser storage (prompting the user to log in if no email address is stored) and verifies that a user with that email address exists in the backend's database, using the login endpoint as given in Table 1. The script then submits a *simplification request* to the backend's simplification endpoint, enclosing a JSON object that contains the user's email address (used by the backend to retrieve

[3]Note that, in general, this paper describes the Lexi frontend and backend versions 1.0. Both parts of Lexi are under ongoing development, with details pertaining to the implementation possibly subject to change.

[4]`https://developer.mozilla.org/en-US/Add-ons/WebExtensions`

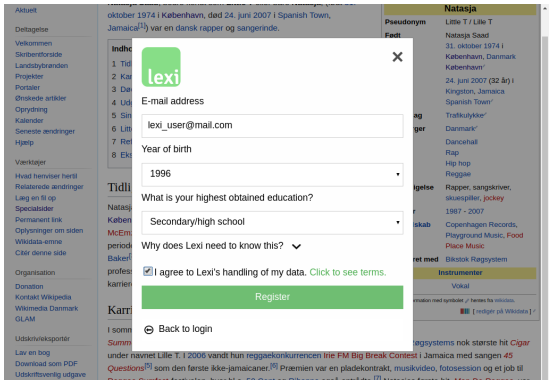


Figure 2: User account registration form

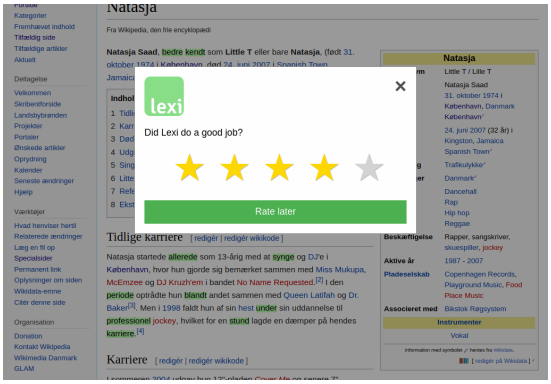


Figure 3: Five-point rating form

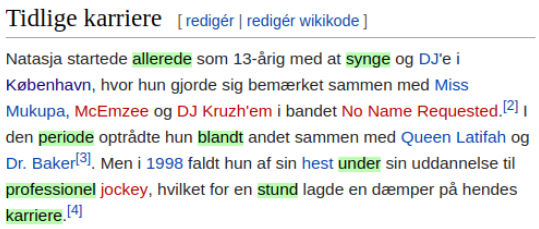


Figure 4: Simplification spans are marked up in light green. As the user clicks on a simplification span, the currently displayed word is replaced with an alternative.

the personal simplification model) and the HTML code of the element containing the text selection. See Appendix A.1 for an example.

The response from the backend then transmits a JSON object with augmented HTML, where elements with unique IDs are wrapped around simplification targets. The response object further contains an array of *simplification objects*, each of which in turn contains a list of synonyms ordered by simplicity ranking (including the target). An example is given in Appendix A.2. The content script replaces the original source with the augmented HTML and displays each simplification span with a light green background color (see Figure 4). The script then shifts through the simplification alternatives for a given target whenever the user clicks on the respective span on the page, advancing one alternative per click and reverting to the first alternative at the end of list. The original item is marked in a slightly but discernibly darker shade than the proposed simplifications.

4.1.3 User feedback

In order to provide personalized simplifications and to adapt to individual users, Lexi needs to be able to decide which alternative a user prefers over the others for every target. In a classical, controlled annotation setting, one would probably present subjects with a set of alternatives and have them rank these or pick a single favorite. However, as Lexi aims to provide as natural and smooth a reading scenario as possible to its users, explicitly asking for such feedback would critically obstruct the reading process.

Lexi therefore interprets whatever final selection a user makes for some simplification span as their preferred alternative in this context.[5] As the user finally navigates away from the webpage that Lexi was invoked on, Lexi solicits feedback from the user on a five-point scale (see Figure 3) and submits this rating along with the simplification objects and their final selections (and click-through counts) to the

[5]In the instructions, users are made aware of this. The frontend further keeps track of how many times the user clicked on a given simplification span, thus providing the backend with information such as how many times the user clicked through the entire list, or whether perhaps no alternatives were solicited for some item.

feedback endpoint of the backend.[6] See Appendix A.3 for an example of the feedback.

4.1.4 Qualitative evaluation of usability

The frontend design was developed in close collaboration with Nota, the Danish Library and Expertise Center for people with reading disabilities.[7] In February 2018, the software was intensively tested by four dyslexic members of Nota, all female students in secondary/higher education and aged between 20 and 30. Each test started with a short preliminary interview in which the subjects were asked about their age, occupation/study field, reading habits, degree of dyslexia and use of browser extensions. The subjects were then given the possibility to watch an introduction video (of 1:30 min length) outlining Lexi's basic functionality and user interface. Two of the four subjects opted for this, while the other two decided to skip the video as they do not usually watch introduction videos when using new software. Next, the subjects were asked to locate Lexi in the Chrome Webshop, install it in the browser and create a user account. Once set up, each subject navigated to a site of her choice and used Lexi to receive simplifications as outlined in 4.1.2. The two subjects who had not watched the video did so now, and both declared they gained further insight into Lexi's functionality through the video, but that it was not crucial in order to understand its basic usage.

In qualitative interviews directly succeeding each test, the test subjects overall reacted very positively to the prospect of a personalized simplification tool in general, and to Lexi and its design in particular.[8] The test subjects suggested a number of improvements, most of which have now been implemented. One suggested improvement, which we have not been able to implement but intend to do so for a future version, is the support for multilingual simplification. Two subjects said they would greatly appreciate this, as much of their study material is only available in English.

4.2 Backend

Lexi's backend consists of a simplification system, implemented in Python 3.5, and a database that stores user information and their simplification histories.

4.2.1 Simplification system

As stated above, Lexi's simplification system currently focuses on lexical simplification, abiding to the *de-facto* standard pipeline depicted in Figure 1. Since Lexi lets users choose which words they wish to have simplified, it does not employ any automatic CWI.[9] Below we sketch Lexi's simplification system as it receives simplification requests from the frontend. As our lexical simplification approach is sensitive to the context of a word, Lexi's first step is to preprocess the HTML source transmitted from the frontend, identifying the boundaries of the sentence that contains the target word, if any.[10]

For Substitution Generation, Lexi implements the embeddings-based approach inspired by the contributions of Glavaš and Štajner (2015) and Paetzold and Specia (2016c). In their work, they extract as candidate substitutions the N words with the highest cosine similarity with a target word. As Danish, the language currently served by Lexi, is not as well-resourced as for example English, Lexi extends the embedding-based Substitution Generation approach by using an ensemble of embeddings models that are trained independently on different text sources, the Danish Wikipedia and a news corpus.[11] The overall similarity score for a target-candidate pair is then defined as the mean score across these embeddings models. Lexi returns the ten most similar candidates whose mean similarity score exceeds some

[6] More correctly, feedback is not solicited when the user actual navigates away from the page, as security restrictions in browsers disallow custom scripts to run upon closing a page. Instead, Lexi asks for feedback via a small notification box in the upper right corner of the page, which pops up as the operating system's *focus* changes to a different window, or when the mouse leaves the browser's viewport (e.g. for the address bar).

[7] `http://www.nota.dk`

[8] An informal evaluation of the software on a 5-point scale (with 1 being worst and 5 best) yielded two ratings of 5, one 4 and one 3.

[9] We do plan, however, to implement CWI as the user solicits simplifications for longer text passages or entire pages.

[10] In order to reduce bandwidth and modify the page more easily, the frontend only transmits the HTML source of the least HTML node fully containing the selection, which typically is a paragraph (`<p>`), but may also be a single word contained in a heading (e.g. `<h1>`), in which case no context is available. Sentence boundaries are identified using NLTK.

[11] `https://ordnet.dk/korpusdk`

configurable threshold. Alternatively, Lexi allows to generate synonyms from a simple dictionary, in the case of Danish using the Danish WordNet (Pedersen et al., 2009), yet this approach suffers from severely reduced coverage compared to word embeddings.

Once generated, the candidates are filtered during Substitution Selection by an unsupervised boundary ranker (Paetzold and Specia, 2016c). In this approach, a supervised ranker is trained with instances gathered in an unsupervised fashion: we generate candidate substitutions for complex words using our generation approach, then assign label 1 to the complex words and 0 to the generated candidates. The boundary between the two classes is then used to rank and filter candidates. Paetzold and Specia (2016c) show that this is a state-of-the-art approach that outperforms all earlier supervised and unsupervised strategies. Given a target word and a set of generated candidate substitutions, the model ranks the candidates based on how far in the positive side of the data they are, then selects 65% of the highest ranking ones.

Finally, the selected candidates are ranked with a supervised Substitution Ranking model following the approach we outlined in Section 3.1.2. It is during this step that Lexi is capable of producing customized output based on the user's needs, and to evolve based on the user's feedback. Lexi employs a pairwise online logistic regression model that learns to quantify the simplicity difference between two candidate substitutions. Given an unseen set of candidate substitutions, the regressor estimates the simplicity difference between each candidate pair, then ranks all candidates based on their average score.

Note that the user's feedback, sent by the frontend, consists of a set S and an index i, where S is the full set of suggested synonyms, including the target, and i is the index of the item in S that the user finally selected. As the regressor, however, learns from pairwise rankings, Lexi passes all pairs $\{\langle S_i, S_j \rangle | j \neq i\}$ to the regressor, i.e. it pairs the selected item with all others and updates the ranker accordingly, postulating that the selected item is easier for this user than each other suggestion.

Using a seed dataset of complex-simple word correspondences in context, we train a default model that produces initial simplifications as a user solicits simplifications for the first time.[12] As Lexi receives feedback for this user for the first time, the seed model is copied and personalized with the first batch of feedback, then this model is saved for later requests by this user.

4.2.2 Database

Lexi stores user information and simplification histories in a PostgreSQL database. More specifically, it employs three different tables, called `users`, `models` and `sessions`. In the first of these, it links a unique, numerical user ID to a user email address, and stores when the user first and last used Lexi. It further contains the demographic information the user provides at registration, i.e. their year of birth and educational status. The `models` table stores a path to the serialized personal model for each user ID. Finally, the `sessions` table stores each simplification request issued to the backend with a unique session ID, the respective user ID, a time stamp for the session start and one for the submission of feedback, the webpage URL, simplification objects serialized as JSON, the provided rating and finally the frontend version number used in this session.

4.3 Communication between backend and frontend

Lexi's backend offers a RESTful API implemented in Python 3.5, using the Flask package.[13] The services available through HTTP POST requests, with their URI paths listed in Table 1. Input and output values are communicated via a JSON-based protocol exemplified in the appendix. Lexi further defines a set of error codes for easier troubleshooting and flexible internationalization of the frontend via the i18n API used by WebExtensions.

[12] Such a seed dataset is not necessarily available for any language. However, in its absence, a seed model could either be trained with simple heuristics, e.g. replacing infrequent words with higher-frequency synonyms. Alternatively, the system could choose to initially rank candidates with such a heuristic and only start learning once the first feedback is available.

[13] `http://flask.pocoo.org/`

URI path	Input	Returns
`/simplify`	User ID; page HTML	Augmented HTML, simplification objects
`/login`	User email address	If successful: User ID, else error code
`/register`	User information	If successful: User ID, else error code
`/feedback`	User ID; simplification objects updated with selections; rating	Status code (successful update or error)

Table 1: RESTful API endpoints defined by Lexi's backend.

4.4 Language support and extensibility

Lexi's design does not impose any restrictions on the support of new (written) languages, including right-to-left or non-alphabetic writing systems. In fact, supporting a new language simply amounts to providing a new language-specific simplification pipeline as illustrated in Figure 1.

Depending on the specific implementation of the simplification system, certain resources are however needed to induce a first seed model for simplification. Most centrally, this pertains to Substitution Generation, where a synonym database or good word embeddings are required in the case of lexical simplification, or a reliable paraphrase module in the case of higher-level simplification. With respect to Substitution Ranking, the availability of resources such as simplification corpora is less critical, as simple heuristics (e.g. simplicity proxies such as length and frequency) might give a reasonable baseline upon which the system can then improve through user feedback.

Lexi currently does not offer multilingual support, but is confined to one language per backend instance. Supporting multilingual simplification could be implemented through a language identification module upstream to the set of simplification pipelines, consisting of one pipeline per language. This raises the interesting question whether knowledge about one user's simplification preferences in one language could be transferred to another language. Support for this hypothesis comes, among others, from the cross-lingual track in the recent CWI shared task by Yimam et al. (2018).

4.5 Ethical and legal considerations

As any software interacting with users and storing information on them, Lexi is naturally subject to ethical and legal concerns, especially those regarding privacy. The EU General Data Protection Regulation (GDPR), for instance, defines a number of regulations such as the clear statement of terms and conditions or that users need to be provided, upon request, with full access to whatever data is stored on them. Lexi does not explicitly store users' names, but in many cases they will be encoded in email addresses. Personally identifiable information may also be stored in the form of simplified text that is logged in the database, for instance if Lexi is used on a user's personal social media profile. The above also highlights the need for encrypted communication between the client and the server, which is safeguarded through TLS encryption over the HTTPS protocol.

Ethical concerns pertaining to text simplification arise when infelicitous simplifications distort the meaning of a text and thus potentially misinform the reader. This is difficult to completely rule out, such that the user should clearly be informed of this possibility. Other concerns revolve around the hypothesis that reducing text complexity will "dumb down" the material and keep users at a low reading level by under-challenging them (Long and Ross, 1993). However, as Rello et al. (2013) point out, "anything which might help [dyslexics] to subjectively perceive reading as being easier, can potentially help them to avoid this vicious circle [of reading less and staying on a low reading level], even if no significant improvement in readability can be demonstrated."

5 Availability and applications

The Lexi software and code, including its backend and frontend, are freely available for non-commercial use under a CC-BY-NC license, obtainable at `https://www.readwithlexi.net`. Researchers can set up their own, customized version of the software and distribute the browser extension to users. It is

straightforward to modify features of the software such as offered languages or the exact resources used to induce the initial models.

Besides its core functionality, which we mapped out in the previous sections, Lexi has a number of alternate use cases, which we discuss in this section.

Preloaded simplifications Lexi's primary use case, as described earlier, is to provide simplifications to users as they select a span of text, which circumvents the need for a CWI module as only such items are simplified that the user explicitly solicits replacements for. Alternatively, users may wish to have the entire page simplified before they start reading. Lexi currently implements this functionality, letting the user solicit simplifications for the entire site via a click on the Lexi icon. As there is no personalized CWI module implemented yet, simplification targets are identified via a confidence threshold during Substitution Generation.

Evaluation of simplification quality Via its rating function (Figure 3), Lexi continuously tracks user satisfaction as a means of evaluating synchronic simplification quality as well as the diachronic development of model adaptation. An adaptive model that is continuously customized is expected to gradually improve the average rating it receives from the user.

Data collection Lexi makes it possible to collect user choices over a longer period in order to create bigger simplification datasets. If sufficiently homogeneous subgroups can be identified across users, this data may give insight into their simplification needs, to build better simplification models for them.

Other plausible approaches may understand different users as different *tasks* and apply multi-task learning methods to transfer knowledge between users, thus both regularizing the models for the individual user and increasing the available amount of data that the individual models can be learned from.

6 Conclusion and Future Work

This paper is a first work in personalized, adaptive text simplification, a direction of research motivated by the observation that generic, user-independent simplification systems cannot fully unfold their potential in making text simpler for specific end users. We propose a framework for adaptive lexical simplification, outlining how user feedback can be used to gradually enhance and personalize text simplification. As a concrete first solution to the problem, we present Lexi, an open-source tool for personalized, adaptive text simplification that has been very positively evaluated in a first usability test. In its current implementation, Lexi focuses on lexical simplification in Danish. An extension to other languages is simple, requiring only a medium-sized monolingual corpus on which a language model and word embeddings can be trained.

In future work, we aim to extend the proposed framework to sentence-level simplifications. We further plan to implement support for multilingual simplification.

References

Regi Alexander, Peter E Langdon, Verity Chester, Magali Barnoux, Ignatius Gunaratna, and Sudeep Hoare. 2016. Heterogeneity within autism spectrum disorder in forensic mental health: the introduction of typologies. *Advances in Autism*, 2(4):201–209.

Mahmoud Azab, Chris Hokamp, and Rada Mihalcea. 2015. Using word semantics to assist english as a second language learners. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 116–120, Denver, Colorado, June. Association for Computational Linguistics.

Dirk J Bakker. 1992. Neuropsychological classification and treatment of dyslexia. *Journal of learning disabilities*, 25(2):102–109.

Joachim Bingel and Anders Søgaard. 2016. Text simplification as tree labeling. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 337–343.

Joachim Bingel, Maria Barrett, and Sigrid Klerke. 2018. Predicting misreadings from gaze in children with reading difficulties. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 24–34.

Or Biran, Samuel Brody, and Noémie Elhadad. 2011. Putting it simply: a context-aware approach to lexical simplification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 496–501. Association for Computational Linguistics.

John Carroll, Guido Minnen, Yvonne Canning, Siobhan Devlin, and John Tait. 1998. Practical simplification of english newspaper text to assist aphasic readers. In *Proceedings of the AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, pages 7–10.

Kevyn Collins-Thompson. 2014. Computational assessment of text readability: A survey of current and future research. *ITL-International Journal of Applied Linguistics*, 165(2):97–135.

William Coster and David Kauchak. 2011. Simple english wikipedia: a new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 665–669. Association for Computational Linguistics.

Jan De Belder and Marie-Francine Moens. 2010. Text simplification for children. In *Prroceedings of the SIGIR workshop on accessible search systems*, pages 19–26. ACM.

Siobhan Devlin and Gary Unthank. 2006. Helping aphasic people process online information. In *Proceedings of the 8th SIGACCESS*, pages 225–226.

Richard Evans, Constantin Orasan, and Iustin Dornescu. 2014. An evaluation of syntactic simplification rules for people with autism. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 131–140.

Roy Thomas Fielding. 2000. Rest: architectural styles and the design of network-based software architectures. *Doctoral dissertation, University of California*.

Goran Glavaš and Sanja Štajner. 2015. Simplifying lexical simplification: Do we need simplified corpora? In *Proceedings of the 53rd ACL*, pages 63–68, Beijing, China, July. Association for Computational Linguistics.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a lexical simplifier using wikipedia. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 458–463.

Anne-Laure Ligozat, Anne Garcia-Fernandez, Cyril Grouin, and Delphine Bernhard. 2012. Annlor: a naïve notation-system for lexical outputs ranking. In *Proceedings of the 6th International Workshop on Semantic Evaluation*, pages 487–492. Association for Computational Linguistics.

Jun Liu and Yuji Matsumoto. 2016. Simplification of example sentences for learners of japanese functional expressions. In *Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA2016)*, pages 1–5.

Michael H Long and Steven Ross. 1993. Modifications that preserve language and content. *Technical Report (ERIC)*.

Gustavo Paetzold and Lucia Specia. 2015. Lexenstein: A framework for lexical simplification. *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 85–90.

Gustavo Paetzold and Lucia Specia. 2016a. Anita: An intelligent text adaptation tool. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 79–83.

Gustavo Paetzold and Lucia Specia. 2016b. Semeval 2016 task 11: Complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569.

Gustavo Henrique Paetzold and Lucia Specia. 2016c. Unsupervised lexical simplification for non-native speakers. In *Proceedings of the 13th AAAI*, pages 3761–3767. AAAI Press.

Gustavo Paetzold and Lucia Specia. 2017. Lexical simplification with neural ranking. In *Proceedings of the 15th EACL*, pages 34–40. Association for Computational Linguistics.

Bolette Sandford Pedersen, Sanni Nimb, Jørg Asmussen, Nicolai Hartvig Sørensen, Lars Trap-Jensen, and Henrik Lorentzen. 2009. Dannet: the challenge of compiling a wordnet for danish by reusing a monolingual dictionary. *Language resources and evaluation*, 43(3):269–299.

Luz Rello, Ricardo Baeza-Yates, Stefan Bott, and Horacio Saggion. 2013. Simplify or help?: text simplification strategies for people with dyslexia. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, page 15. ACM.

Allan Peter Rudell. 1993. Frequency of word usage and perceived word difficulty: Ratings of kucera and francis words. *Behavior Research Methods*, pages 455–463.

Matthew Shardlow. 2014a. Out in the open: Finding and categorising errors in the lexical simplification pipeline. In *LREC*, pages 1583–1590.

Matthew Shardlow. 2014b. A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications*, 4(1):58–70.

Advaith Siddharthan. 2014. A survey of research on text simplification. *ITL-International Journal of Applied Linguistics*, 165(2):259–298.

Adel I Tweissi. 1998. The effects of the amount and type of simplification on foreign language reading comprehension. *Reading in a foreign language*, 11(2):191–204.

Betty U Watson and David E Goldgar. 1988. Evaluation of a typology of reading disability. *Journal of clinical and experimental neuropsychology*, 10(4):432–450.

Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: Unsupervised extraction of lexical simplifications from wikipedia. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 365–368. Association for Computational Linguistics.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A Report on the Complex Word Identification Shared Task 2018. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, United States, June. Association for Computational Linguistics.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd international conference on computational linguistics*, pages 1353–1361. Association for Computational Linguistics.

Johannes C Ziegler, Caroline Castel, Catherine Pech-Georgel, Florence George, F-Xavier Alario, and Conrad Perry. 2008. Developmental dyslexia and the dual route model of reading: Simulating individual differences and subtypes. *Cognition*, 107(1):151–178.

A Examples of JSON protocol

A.1 Request sent from frontend to backend

```
{
  'frontend_version': '1.0.0',
  'user': 'lexi-user@mail.com',
  'html': '<p>Natasja startede allerede som 13-årig med at synge og
     DJ\'e  ... </p>',
  'startOffset': 17,
  'endOffset': 25,
  'url': 'https://da.wikipedia.org/wiki/Natasja'
}
```

A.2 Reply from backend to frontend

```
{
  'backend_version': '1.0.0',
  'html': '<p>Natasja startede <span id="lexi_254_1" class="lexi-
     simplify">allerede</span> som 13-årig med at synge og DJ\'e ...
     </p>',
  'session_id': 254,
  'status': 200,
  'message': 'Simplification successful',
  'simplifications': {
    'lexi_254_1': {
       'choices': ['allerede', 'bare'],
       'selection': 0,
       'sentence': "Natasja startede allerede som 13-årig med at
          synge og DJ'e i København, hvor hun gjorde sig bemærket
          sammen med Miss Mukupa, McEmzee og DJ Kruzh'em i bandet No
          Name Requested.",
       'word_index': 2
    }
  }
}
```

A.3 Feedback sent from frontend to backend

```
{
  'frontend_version': '1.0.0',
  'user': 'lexi-user@mail.com',
  'html': '<p>Natasja startede <span id="lexi_254_1" class="lexi-
     simplify">bare</span> som 13-årig med at synge og DJ\'e ...
     </p>',
  'session_id': 254,
  'simplifications': {
    'lexi_254_1': {
      'choices': ['allerede', 'bare'],
      'selection': 1,
      'sentence': "Natasja startede allerede som 13-årig med at synge
        og DJ'e i København, hvor hun gjorde sig bemærket sammen med
        Miss Mukupa, McEmzee og DJ Kruzh'em i bandet No Name
        Requested.",
      'word_index': 2
    },
  },
  'rating': 4,
  'url': 'https://da.wikipedia.org/wiki/Natasja',
  'session_id': 254
}
}
```

Identifying Emergent Research Trends by Key Authors and Phrases

Shenhao Jiang[†], **Animesh Prasad**[†], **Min-Yen Kan**[†], **Kazunari Sugiyama**[†‡]
[†]School of Computing, National University of Singapore
shenhao_jiang@u.nus.edu
{animesh|kanmy|sugiyama}@comp.nus.edu.sg
[‡]National Institute of Informatics, Japan
zakugus@nii.ac.jp

Abstract

Identifying emergent research trends is a key issue for both primary researchers as well as secondary research managers. Such processes can uncover the historical development of an area, and yield insight on developing topics. We propose an embedded trend detection framework for this task which incorporates our bijunctive hypothesis that important phrases are written by important authors within a field and vice versa. By ranking both author and phrase information in a multigraph, our method jointly determines key phrases and authoritative authors. We represent this intermediate output as phrasal embeddings, and feed this to a recurrent neural network (RNN) to compute trend scores that identify research trends. Over two large datasets of scientific articles, we demonstrate that our approach successfully detects past trends from the field, outperforming baselines based solely on text centrality or citation.

1 Introduction

At no time prior to now has the advance of science – as measured by the rate of growth in scientific publications – bloomed more rapidly. Even practitioners of narrowly-defined, specific fields need help to scan the masses of literature to summarize work and to identify developments likely to spur long-term impact due to the potential problem of information excess (Fujita et al., 2012). The identification of such **emergent research trends** — "radically novel and relatively fast growing research topic characterized by a certain degree of coherence, and a considerable scientific impact" (Wang, 2017) — is a task that is growing in importance and urgency with the increasingly large scale of scholarly literature.

Wang (2017)'s survey reveals two important techniques that characterize the state-of-the-art work on this task. First, text mining — specifically in the guise of topic models — have been employed to yield a summative form of large corpora. Both the Dynamic Topic Model (Blei and Lafferty, 2006) and Author–Topic Model (Rosen-Zvi et al., 2004) cater towards different aspects of the special temporal and authoring aspects of topics in scientific documents, specializing the basic Latent Dirichlet allocation (LDA; Blei et al. (2003)) topic model. Second, dense citations among papers has been used to measure the coherence contributing to a well-defined research trend (Price, 1965). Through appropriate metrics, core works can be identified, from which extracted keywords are treated as a trend's central concepts. Analyses often establish a co-citation network, as typified by Shibata et al. (2008).

These works point to the importance of side information external from the content itself, such as publication date or author identity, and its necessary incorporation into a detection methodology. Our observation is that these attributes often correlate with each other, naturally hinting to a framework where such information can reinforce their signal.

We thus take inspiration from other natural language processing and information retrieval scenarios where such mutual recursion has found application. Hyperlink Induced Topic Search (HITS; (Kleinberg, 1999)) showed web pages arranged as nodes of a hyperlinked graph can be mutually related and ranked by an authority and a hub (popularity) score, where the two scores are calculated using mutual recursion.

Proceedings of the 27th International Conference on Computational Linguistics, pages 259–269
Santa Fe, New Mexico, USA, August 20-26, 2018.

Yan et al. (2017) applied a similar logic to the task of ranking sentences and words with mutual recursion, where it is assumed that important sentences and important words are correlated. They achieved better performance for keyphrase extraction compared with TextRank (Mihalcea and Tarau, 2004), a non-recursive formulation.

Furthermore, with the current advent of deep learning and better semantic representations via embeddings, there is opportunity to utilize both advances in trend detection.

Applying these observations to the research trend task, we make the following assumptions when formulating our model detailed in Section 3:

- Important authors often collaborate together, *i.e.*, the co-authorship among key-authors contributes to the research trend emergence.
- Important authors often write important keyphrases. Such phrases are then often trending phrases for a research area.
- High centrality values for nodes are key features to identify the core items within a graph. For identifying emergent trends, these metrics must exhibit values that connote both importance and urgency.
- We neuralize the traditional trend detection pipeline. Specifically, we represent obtained keyphrases as averaged *tf-idf* word embeddings and use a recurrent neural network (RNN) model to perform supervised learning for trend status.

We use a multigraph ranking (MGR) core component to compute trending scores for keyphrases. We take these MGR outputs as input to our neuralized embedded trend detection (EDM) framework: namely, we represent the outputs as word or phrase embeddings, and feed their ranking scores into a recursive neural network to identify trends per time slice in a target dataset. This results in our final embedded trend detection model – ETD (MGR), or "ETD-M" for short.

2 Prior Work

Previous studies have addressed both of our tasks of obtaining keyphrases and identifying research fronts from scientific publications. We review this area first, then review mutually recursive algorithms used in practice.

Research Trend Detection. Wang (2017) classified research trend detection models into two categories, namely, text- and bibliometric-based approaches. In text-based approaches, keywords and terms representing the core topics have been the focus of study. Hall et al. (2008) performed post-hoc analysis by applying the LDA model to the batch of publications in the same year; top keywords generated in different years are compared in an effort to identify the transition of topics. Mörchen et al. (2008) attempts to combine keywords and the time of their appearances: the variation of term frequencies in two different timestamps provides the insights on how significant the term is becoming. Sessions within conferences are another perspective that have been investigated. Furukawa et al. (2014) constructed *tf-idf* vectors (Salton et al., 1983) from publications' abstracts to obtain the relative importance of sessions, and then applied the vectors to assess the evolution of a publication venue.

In bibliometric approaches, citations are used to connect papers. Hopcroft et al. (2004) have proposed the concept of co-citation, where papers are connected if they are both cited by another paper published later; following the co-citation relation, the graph of papers can be used for further studies (Shibata et al., 2008): topological clusters inside the graph was identified, where tightly knit clusters with densely distributed edges represent the general topics. Subsequently, NLP techniques such as *tf-idf* are employed to generate the top terms associated with the topic. Similar to textual approaches, top keywords from clusters in different years can then be used for research fronts analysis. Elkiss et al. (2008) tried to analyze the cohesion of texts based on the citing sentences and co-citation metrics, which provides insights on the

focused perspective of the cited paper. He et al. (2009) proposed an inheritance topic model to identify topic evolution by fully utilizing the impact of citations.

We note that both textual and bibliometric approaches expose shortcomings: they are restricted to historical data, and human interpretations are often required to establish the link between different timestamps; additionally, without a comprehensive keywords extraction method, the final output can be ambiguous and uninformative.

Mutual Recursive Algorithms. Hyperlink Induced Topic Search (HITS) Algorithm (Kleinberg, 1999) is a prototypical example and used for ranking pages on the Internet, where each page is assigned with two values: an "authority" score and a "hub/popularity" score. The rationale states that a good hub page points to many other pages, where a good authority page is pointed to by many hubs. Therefore, the two scores are computed dependently on each other.

In natural language processing, the implication of mutual recursion can apply to keyphrase extraction models: Yan et al. (2017) implemented a ranking model by constructing separate sentence and word graphs. They assume that important sentences and words are correlated, hence by linking the two graphs together and ranking sentences and words recursively, salient keywords can be extracted. Their model outperforms standard TextRank algorithm (Mihalcea and Tarau, 2004). Zhang et al. (2017) also proposed a graph-based keyphrase extraction approach by capturing mutual influences among different types of information on a candidate word, which have been widely used in existing supervised or graph-based approaches.

3 Methodology

Our goal is to mine and assign high fidelity importance scores to keyphrases, and then to use these to predict emerging trends, based on the abstracts for a large target scientific publication dataset. Our embedded trend detection (ETD) model consists of the following three components: 1) extraction of keyphrases, 2) representativeness enhancement with word embeddings, and 3) predictions and rankings of phrase with RNN. In the following, we detail each of the components.

3.1 Multigraph Ranking for Key Phrase and Author Extraction

Our ETD framework admits any means of identifying the keyphrases, so any keyphrase generation algorithm can be employed (*e.g.*, TextRank). However, since our task is slightly different in that we want to generate keyphrases for overall trend detection of an area (as opposed to the more typical characterization of a single publication), we need to introduce several refinements.

As a result, we propose our multigraph ranking (MGR) component for salient keyphrase extraction. Our MGR model starts with the extraction of keyphrases and key-authors based on the assumption that important authors are more likely to write potentially trending phrases. We extract them via iterative score ranking in a pair of jointly-connected graphs.

We construct the graphs as follows: documents are first grouped into year-based clusters. For each of the cluster, we build an "author" graph, which contains individual authors who have published at least one paper in that year; authors are connected by undirected edges whose weights are determined by the number of publications two authors have made together during the year. We also construct a "phrase" graph following the logic of TextRank with modifications. Compare this to the standard TextRank model, where the nodes in the graph are tokenized words from the raw text, and the edges conceptualize the adjacency when tokens occur within a predefined window size.

Our empirical results demonstrate that domain-specific single-word terms dominate the final keyphrase list. Hence in our model, we pre-process the input to extract noun phrases via regular expression over parts-of-speech[1] before TextRank processing.

Consider the sample document in Figure 1: after transforming the raw text to a set of noun phrases, each noun phrase is split further: for {coherency protocols}, two separate nodes will be added to the

[1]Using the Natural Language Toolkit (NLTK) via `<DT>?<JJ|NN|NNS>*<NN|NNS>` (where `DT` stands for determiner; `JJ` for adjectives, and `NN|NNS` for nouns).

Title: Evaluating the performance of four snooping cache coherency protocols.
Authors: S. Eggers and R. Katz.
Publication Year: 1989.
Abstract: ... **coherency protocols** have been criticized for being unable to achieve **good bus performance** across **all cache configurations** ...

Transformed Token List: {coherency protocols, good bus performance, all cache configurations}

Figure 1: Example text from 1989, with noun phrases bolded.

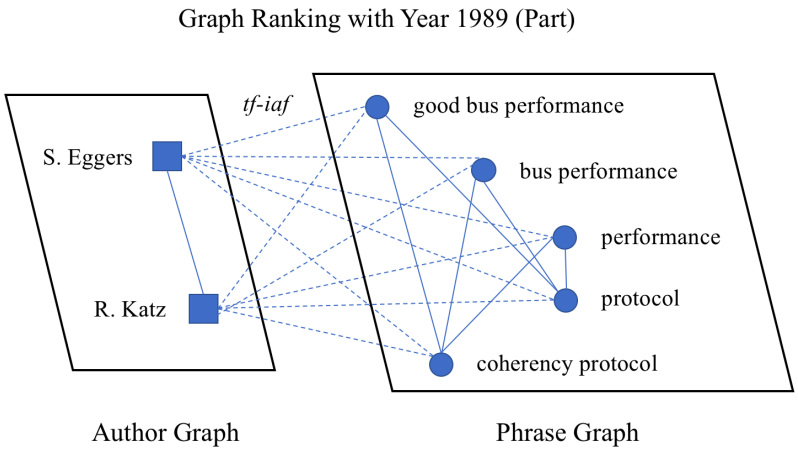


Figure 2: Portion of the multigraph derived from the text in Figure 1. Note that in the complete graph, nodes may connect with other unseen authors and phrases; we only show the part constructed by the sample document.

"phrase" graph: **coherency protocol** and **protocol**; similarly, **good bus performance**, **bus performance**, and **performance** from the next noun phrase. Subsequently, each node from {coherency protocol} will be linked with another node from {good bus performance}. Therefore, single nodes in our "phrase" graph can be entire noun phrases.

The "author" and "phrase" graphs are jointly connected based on the "author-writes-phrase" relationship, in which an author–phrase pair is denoted as { author a_i – phrase p_j}. We then propose *tf-iaf*, which is a variant of *tf-idf* (Salton et al., 1983) to calculate the weight of the edges:

$$\begin{aligned} tf\text{-}iaf_{a_i,p_j} &= tf_{a_i,p_j} \times iaf_{p_j} \\ &= \frac{Occ(a_i,p_j)}{\sum_{z=1}^{n} Occ(a_i,p_z)} \times \log \frac{|A|}{|A(p_j)|}, \end{aligned} \tag{1}$$

where $Occ(a_i, p_j)$, $|A(p_j)|$, and $|A|$ denote the number of occurrences of p_j by a_i, the number of authors who write p_j, and the total number of authors, respectively.

Returning to our running example, for the document group for year 1989, a portion of the graph is demonstrated in Figure 2.

Ranking and extraction of the key-authors and keyphrases can then be implemented through an iterative method, similar to the HITS algorithm. We use three matrices to store the "author", "phrase", and the *tf-iaf* graphs. More specifically,

· $\boldsymbol{N}_{m\times m}$ is a matrix for the set of authors $A = \{a_i | 1 \le i \le m\}$. Each element $n_{i,j}$ denotes how many papers auhtors a_i and a_j co-authored during the year. In the sample document presented above, if the authors "S. Eggers" and "R. Katz" have only collaborated on this paper in 1989, then we have $n_{Eggers,Katz} = 1$ for matrix $\boldsymbol{N}$.

· $\boldsymbol{W}_{n\times n}$ is a matrix for the set of phrases $P = \{p_j | 1 \le j \le n\}$. Each element $w_{i,j}$ denotes that how

often phrase p_i and p_j have co-occurred across all documents since that year. Note that for matrices $\boldsymbol{N}$ and $\boldsymbol{W}$, we have $\boldsymbol{N} = \boldsymbol{N}^T$ and $\boldsymbol{W} = \boldsymbol{W}^T$.

· $\boldsymbol{L}_{m \times n}$ is a matrix for the *tf-iaf* scores defined by Equation (1). Each element $l_{i,j}$ denotes the *tf-iaf* value, in other words, $l_{i,j} = tf\text{-}iaf_{a_i,p_j}$.

Additionally, we use a $m \times 1$ vector $\vec{i}$ and a $n \times 1$ vector $\vec{j}$ to represent the scores of the authors and phrases, respectively. Both vectors are initialized with 1 for each entry; subsequently, the score vectors $\vec{i}$ and $\vec{j}$ are updated using the following rule:

$$\begin{aligned} \vec{i} &= (\alpha \times \boldsymbol{N}\vec{i} + (1-\alpha) \times \boldsymbol{L}\vec{j}).normalized(), \\ \vec{j} &= (\alpha \times \boldsymbol{W}\vec{j} + (1-\alpha) \times \boldsymbol{L}^T\vec{i}).normalized(). \end{aligned}$$

where α is a tuning coefficient to determine the relative importance of authors and phrases during ranking.

Therefore, after the iterative ranking, noun phrases are assigned a score based on both the adjacency with other phrases and the author information for a particular year. Consolidating all scores for all phrases produces a $2D$ array, of which each row stands for the time series scores for a phrase.

3.2 Representativeness with Word Embeddings

A key point here is that the meaning of a phrase and its relation to other phrases evolves over time. For instance, during the development of the topic *Machine Learning*, the phrase "machine learning" was likely to be associated with popular supervised approaches such as "decision tree" and "SVM" around 2000, but after 2010 co-occurs more often with "deep neural networks". These changes indicate that the representativeness of a phrase to a topic also shifts over time. To model such dynamics, we use word embeddings to further refine the phrase scores from the previous step in our ETD pipeline.

Our model assigns a vector per phrase, per unit time step (year). We construct phrase vectors by averaging its individual word embeddings (originating from the 300-dimensional version of Google's Word2Vec model (Mikolov et al., 2013)). The phrase vector is calculated by taking the *tf-idf* weighted average of the vectors of individual words, where the *tf-idf* is calculated on the basis of the yearly-divided corpus. We note that we tried training dynamic word embeddings based on recalibrating the vectors as per many standard NLP application, but that technique did not lead to better performance.

As individual phrases are too fine-grained to be minted as trends, we apply a form of dimensionality reduction to find key phrases that can serve as a label for a trend. We group the vectors into clusters using the k-means algorithm, where each phrase is associated with a representativeness value calculated by taking the cosine similarity between its vector and its respective cluster center. The phrase score obtained from the graph ranking step is multiplied with the representativeness score. In this way, we assign phrases with the list of scores indicating their significance and representativeness over the years.

3.3 Predictions with RNN and Trending Phrase Ranking

For a particular phrase, given its scores over a certain period of time, we expect our model to predict the phrase score in the following time interval. In our model, we implement a basic recurrent neural networks (RNNs) to makethese predictions.

The model employs scores from the first k sequential years to predict $(k+1)^{th}$ score defined as:

$$x_{k+1} = f(x_1, x_2, x_3, ..., x_k).$$

For the array of scores from 1958 to 2015, this is a list of training samples where $\boldsymbol{x}$ is score vector that consists of k sequential years and $\boldsymbol{y}$ is the score in the $(k+1)^{th}$ year. The input of the RNN is $1 \times k$ representing the same feature (score of a certain year) spanned across k years, and the output is a single value representing the score in the $(k+1)^{th}$ year.

With the scores for each phrase predicted from year 1958 to 2015, it is then possible to investigate further into the phrases. As illustrated previously, it is expected that our model could make predictions of trending research fronts. Specifically, trends refer to research areas which are gradually being studied,

hence, there should be a strong foundation for them, meaning that there are certain level of existing work associated with them; yet they should not be the mostly-discussed issue at the same time. Based on these assumptions, we have defined a *trending score* for each phrase, which can be defined as follows:

$$trend = \beta \times (x_{k+1} - \max(x_1, \ldots, x_k)) + (1 - \beta) \times \max(x_1, \ldots, x_k),$$

where β is another tuning weight similar to that used in the graph ranking algorithm. Coupled with the previous steps' score computations, we can then determine the most influential and trending phrases in each year with the predictions from RNN. Our model selects the top ranked phrases based on their actual scores for each year, and produces another list based on predictions from RNN. It is expected that when comparing these two lists, phrases appearing near the top of the actual list also appear at the top of the prediction list.

4 Experiments

Evaluation of research trend detection is largely indicative, since the task is subjective, sources various, and replicability of prior work is difficult at best. Despite these problems, we propose a workflow that allows better future replicability, including the standardization of the datasets and the availability of the algorithms' prediction output[2].

In our experiments, we use the abstract of each paper and the published year as one document entry and the timestamp, respectively. After describing specific parameter settings, we describe our evaluation for the two different datasets in turn.

4.1 Parameter Settings

For the three sub-components of our model, we set the parameters as follows:

In the multi-graph ranking component, when iterating through the texts, phrases appearing within a window size of 3 are identified as adjacent tokens and are hence connected in the graph, when updating the final score vectors with Equation (1). We set α to 0.5.

When incorporating the Word2Vec model into phrase scores, we use the k-means algorithm to cluster the phrase vectors. We compare the performance of the model on various settings of $k = \{10, 15, 20, 25\}$. For the recurrent neural network predictions, we set a (context) block size of 3 for the input shape (*i.e.*, in the $2D$ array of scores, for each phrase, its time series scores incorporate the three prior values: $x_4 \leftarrow < x_1, x_2, x_3 >$).

The generated training samples are then fed into the RNN, which includes 2 hidden layers, where each layer has 4 neurons. We use mean squared error as the loss function (as the result is continuous), which naturally dictates the use of RMSProp as the optimizer. Following typical training regimes, we use a batch size of 10, and 80 epochs before termination. We also empirically tuned β to 0.8, emphasizing the positive and increasing trend in importance in defining trending topics and research fronts.

4.2 ACM Periodical Dataset

We use this dataset to evaluate the performance of multigraph ranking (MGR) component as we hypothesize that its mutual recursion will improve trend detection over the baseline TextRank source. This dataset includes around 9,740 XML files representing publications copyrighted by the Association of Computing Machinery (ACM), in the domain of computer science, published from 1958 to 2015. It is available from the ACM used by explicit permission. We use the abstract and metadata (specifically, publication time, authors, and keywords) for each paper. For space, in this presentation, we restrict the field of study to "software engineering"[3].

We evaluate our model with **Recall@n** and **nDCG@n**, and then compare it with the standard TextRank algorithm (Mihalcea and Tarau, 2004) as a baseline.

[2]Code is available at: `https://github.com/RichardeJiang/racode`.

[3]Filtered using the keyword "software engineering" and "software and its engineering" on the `keywords` and `concept description` component of each publication.

Recall @ n^{th} rank												
System	$k = 10$			$k = 15$			$k = 20$			$k = 25$		
	@10	@20	@30	@10	@20	@30	@10	@20	@30	@10	@20	@30
ETD-TR	11.3%	10.5%	8.5%	13.0%	12.9%	10.4%	7.6%	7.4%	6.5%	14.1%	13.2%	**10.5%**
ETD-M	**24.0%**	**21.9%**	**16.5%**	**25.9%**	**23.5%**	**21.1%**	**17.8%**	**15.6%**	**12.9%**	**16.7%**	**13.6%**	**10.5%**

Normalized Discounted Cumulative Gain (nDCG) @ n^{th} rank												
System	$k = 10$			$k = 15$			$k = 20$			$k = 25$		
	@10	@20	@30	@10	@20	@30	@10	@20	@30	@10	@20	@30
ETD-TR	0.105	0.127	0.138	0.107	0.136	0.143	0.078	0.090	0.095	**0.120**	**0.151**	0.161
ETD-M	**0.158**	**0.220**	**0.251**	**0.226**	**0.261**	**0.289**	**0.147**	**0.180**	**0.199**	0.107	0.136	**0.174**

Table 1: Performance comparison of Recall@n (top) and nDCG@n (bottom), varying the number of clusters $k = \{10, 15, 20, 25\}$ generated by k-means. Best system figures are bolded. "ETD-TR" and "ETD-M" stand for embedded trend detection with TextRank (Mihalcea and Tarau, 2004) (baseline) and our multigraph ranking, respectively.

Recall @ 20, $k = 20$					
Run	1	2	3	4	5
ETD-TR	7.4%	10.3%	8.6%	11.1%	9.2%
ETD-M	15.6%	14.0%	15.6%	13.2%	20.1%

Table 2: Comparison of Recall@20 with $k = 20$ clusters over different runs of our model.

Our embedded trend detection workflow consists of the three stages: keyphrase extraction, embedding representation, and final prediction with RNN. Our evaluation here tests the effectiveness when we change the first component to our proposed MGR. We replace the MGR with the standard TextRank algorithm and use the output as the baseline, and compare the resulting ranking model.

Recall@n measures how many predicted phrases are the actual top terms computed by the system. For each year, the first component (MGR and baseline TextRank) returns a list of the top terms. These are further enhanced through Word2Vec embeddings and fed to the RNN to generate top trending phrases. We first compute average recall based on this final output for all available years, and then compute normalized discounted cumulative gain (nDCG) (Järvelin et al., 2000) to further assess the quality of the element ranking. A system performs better if it returns true positive trending phrases towards the top of its rankings, as opposed to the bottom.

We test different settings of $k \in \{10, 15, 20, 25\}$ for the number of clusters when performing the k-means algorithm in the second Word2Vec stage for incorporation, as shown in Table 1.

From Table 1, we note that when extracting the keyphrases and make value predictions, our model consistently outperforms the baseline TextRank algorithm, in both the number of correctly predicted phrases (Recall) and their ordering (nDCG). Since the only difference between the baseline TextRank and our MGR model is how the phrases are ranked, we conclude that our multi-graph ranking algorithm yields more consistent scores in this dataset. We take this as evidence that in our MGR model, random interference is reduced when compared against the standard TextRank algorithm. This provides the empirical justification for our assumption when formulating the multi-graph ranking step – that important authors are more likely to cooperate with each other; and that words written by them are more likely to be trending in their respective fields. Additionally, the use of graphical techniques which find central components does reveal useful core nodes (concepts).

As the k-means algorithm does not guarantee to produce the same cluster result when we run the algorithm for several times, in order to show the consistency of performance, we have tested the system multiple times with the same set of parameters. Here, we show the example when $k = 20$ (number of clusters), and we compute the **Recall@20**, as illustrated in Table 2. We observe that results will be different when testing the system in different runs, which conform to the nature of k-means algorithm; however, our model (ETD-M) consistently outperforms the baseline ETD-TR with the same parameter settings.

Clusters	*Top 10 Keywords*
G_1	degree, growth, substrate, film, ga, gaas, gan, nh, si, surface
G_2	degree, contact, gan, al, ni, ga, ti, au, physics, american institute
G_3	gan, mg, layers, ga, physics, american institute, structure, defect, photoluminescence, strain

Table 3: Top 10 keywords detected by citation network methodology from (Shibata et al., 2008), for the year of 2000. Copied verbatim from the source.

Year	*Top 20 Keyphrases*
1999	molecular beam epitaxy, beam epitaxy, **substrate**, **surface**, epitaxy, deposition, results, quality, diffraction, metalorganic chemical vapor deposition, vapor deposition, **sapphire substrates**, conditions, cm, properties, measurements, chemical vapor deposition, electron microscopy, microscopy, spectra
2000	**growth**, **layers**, **substrate**, epitaxy, **surface**, diffraction, **substrates**, molecular beam epitaxy, electron microscopy, measurements, spectra, spectroscopy, microscopy, cm, properties, **structure**, quality, results, conditions, characteristics
2001	**growth**, **layers**, temperature, temperatures, **substrates**, emission, **layer**, epitaxy, measurements, spectroscopy, **surface**, molecular beam epitaxy, properties, diffraction, **films**, microscopy, beam epitaxy, cm, **photoluminescence**, **structure**

Table 4: Top keyphrases predicted by our ETD-M model. Phrases detected by both ETD-M and Shibata et al.'s (2008) model marked as bolded.

We also tested the sub-systems by eliminating the Word2Vec component from our pipeline of ETD: meaning that after we obtained the phrases together with their time series scores using graph ranking algorithms (either with TextRank or our proposed multigraph ranking, MGR), we feed the data into RNN and evaluate the performance. The resulting Recall@20 for TextRank and MGR are **14.5%** and **17.1%**, respectively. The values illustrate two inferences: firstly, even without the reinforcement of word embeddings, our proposed MGR still produces more consistent ranking scores. Secondly, if comparing the result with average Recall@20 from Table 1 (10.5%, 12.9%, 7.4%, 13.2% for TextRank, 21.9%, 23.5%, 15.6%, 13.6% for MGR), we observe that when the embedding component is added to our system, ETD-M outperforms ETD-TR when the number of k-means clusters is relatively small. By comparison, TextRank considers all available tokens, hence the adjacency relationship includes more information on the representativeness of each word. Therefore, in ETD-TR, we believe that word embeddings introduce noise to the original values. On the other hand, in MGR, the spatial information is diluted and Word2Vec helps to centralize the popularity of each phrase, especially when the number of clusters is small. This indicates that word embeddings are effective in our ETD-M model.

4.3 Science and Social Science Citation Index Datasets

In their paper, Shibata et al. (2008) provided the research fronts detected by their system for the domain of Gallium Nitride (GaN) – a prominent research field in applied physics and material science – for the year 2000 (Table 3). We wish to compare directly with their reported results, contrasting the top terms generated for phrases in the year range of 1999–2001, as shown in Table 4, hypothesizing that our ETD-M system can outperform their inferred keyphrase in terms of subjective keyphrase quality.

They used data gleaned from the Science Citation Index (SCI) and Social Science Citation Index (SSCI) original compiled by the Institute for Scientific Information, which are available from Web of Science databases. To compare with their results as best as possible, we have retrieved the papers using the same database query provided by their research: "GaN OR Gallium Nitride" to compare our results with theirs. We collected papers whose years of publication range from 1970 to 2004, to exactly match the parameters used in Shibata et al.'s work. The total number of the papers is around 15,200.

We note that about half of the keyphrases obtained by Shibata et al.'s work are also predicted suc-

cessfully by the ETD-M model. Examining the two tables carefully, we believe that phrases from our model provide better quality insight: element names such as "ga", "ni", and "al" recognized by Shibata et al.'s work unfortunately provide no detail on the study; furthermore, some terms recognized by them like "american institute" are also irrelevant to the research topic.

Since Shibata et al.'s work filters out non-core papers first, potential keywords can only originate from core papers. According to their results, out of 4,000 publications collected in the year 2000, around 30 are selected based on the roles of each paper within its cluster. Although we cannot replicate their work exactly, we surmise that these central papers may manifest such terms with sufficiently high frequency to be extracted, resulting in the large involvement of element names and non-relevant terms in the output. Due to the mutual recursion element of ETD-M, we can propagate the influence of keyphrases beyond such a core, while assigning an appropriate level of credit. Such results allow ETD-M to convey and rank longer, meaningful phrases such as "molecular beam epitaxy" and "metalorganic chemical vapor deposition" within that domain.

5 Future Work

The three-stage pipeline in our ETD-M model is designed to provide insights on the trending research keywords and has been proven to work better than the standard TextRank and bibliometric approaches. Additionally, the model can be improved from various perspectives.

In our multigraph ranking (MGR) step, the score vectors for authors and phrases are normalized after each iteration of updates to limit the values within the range of $[0, 1]$; the matrices representing author and phrase mutual relations ($\boldsymbol{N}, \boldsymbol{W}, \boldsymbol{L}$) are non-smoothed. We expect by normalizing matrices $\boldsymbol{N}, \boldsymbol{W}, \boldsymbol{L}$ to $\hat{\boldsymbol{N}}, \hat{\boldsymbol{W}}, \hat{\boldsymbol{L}}$ in the first place, and then using the normalized matrices in the ranking, the output score vector will therefore indicate the relative importance and the value will be smoothed. The update rules hence become:

$$\begin{aligned} \vec{i} &= \alpha \times \hat{\boldsymbol{N}}\vec{i} + (1-\alpha) \times \hat{\boldsymbol{L}}\vec{j}, \\ \vec{j} &= \alpha \times \hat{\boldsymbol{W}}\vec{j} + (1-\alpha) \times \hat{\boldsymbol{L}}^T\vec{i}. \end{aligned}$$

Our model uses Google's Word2Vec to enhance the representativeness of each phrase in every year by taking the cosine similarities between the vector representation of a phrase and its respective cluster center. Given a specific collection of documents, the embedding for its vocabulary can be rather different from the fixed, pretrained Word2Vec, so in the future we may consider further training the existing Word2Vec with our yearly-divided data. In addition, the representativeness of phrases can also be extended by incorporating topic models, so as to assign a topic and a value to each phrase would admit the relative importance of the phrase, while providing an intuitive approach in clustering.

The evaluation of our model is largely subjective: whether the output keyphrase list provides insights with better quality depends on human interpretation. To better understand the efficacy of our model, it can be applied to other disciplines such as using the PubMed data, and then we employ domain experts to manually evaluate the keyphrase list.

One common issue with a pipeline model is that error may propagate through different stages and accumulate. Therefore, the overall structure of the model can be redesigned such that the graphs are already time-aware during ranking. Wang et al. (2013) have provided an article ranking model which takes publication time into account. When applying to our model, we could introduce a "time" graph to the existing MGR model, where the heterogeneous graph incorporates publication times.

6 Conclusion

We have presented our ETD-M model for detecting research fronts and trends from scientific publications. Unlike prior work that emphasize evidence from citation links and topological measures, our model mines salient phrases from the abstract of the publications themselves. Our three-stage ETD-M pipeline features a multigraph keyphrase ranking (MGR) algorithm, followed by k-means clustering and keyphrase representation utilizing word embeddings, with a final stage of detecting the topics using an RNN for the final, temporally-sensitive inference of trending topics.

A key contribution from our work lies in the first stage of evidence gathering, in the form of our keyphrase extraction. We structure this task as a mutual recurrence between important authors and their keyphrases. We compare our MGR against the TextRank algorithm within the ETD framework, which clearly showed the advantages of utilizing author information in identifying salient keyphrases. Additionally, we feel that our MGR model's inferred research fronts align well with historical fact.

Our ETD-M model can be extended further for a more comprehensive structure. Applying normalized author and phrase matrices to the update rule makes the scores more reasonable; Mikolov's Word2Vec can be adjusted to fit in our own dataset for each year. Additionally, we can introduce human evaluation by feeding in data from various fields of study, to further evaluate the efficacy of our model while discovering trend insights to these other scientific research areas.

Acknowledgments

This research is supported in part by the National Research Foundation, Prime Ministers Office, Singapore under its International Research Centres in Singapore Funding Initiative. In addition, our experiments are carried out based on the ACM Digital Library dataset (with the metadata state dated 11 September 2015) furnished in collaboration with the Association for Computing Machinery.

References

David M. Blei, Andrew Ng, Michael Jordan, and John Lafferty. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research (JMLR)*, 3:993–1022.

David M. Blei and John Lafferty. 2006. Dynamic Topic Models. In *Proc. of the 23rd Conference on Machine Learning (ICML'06)*, pages 113–120.

Qi He, Bi Chen, Jian Pei, Baojun Qiu, Prasenjit Mitra, and C. Lee Giles. 2009. Detecting Topic Evolution in Scientific Literature: How Can Citations Help? In *Proc. of the 18th International Conference on Information and Knowledge Management (CIKM'09)*, pages 957–966.

Aaron Elkiss, Siwei Shen, Anthony Fader, Gunes Erkan, David States, and Dragomir Radev. 2008. Blind Men and Elephants: What Do Citation Summaries Tell Us About a Research Article? *Journal of the American Society for Information Science and Technology (JASIST)*, 59(1):51-62, 2008.

Katsuhide Fujita, Yuya Kajikawa, Junichiro Mori, and Ichiro Sakata. 2012. Detecting Research Fronts Using Different Types of Weighted Citation Networks. *Journal of Enginnering and Technology Management (JET-M)*, 32:129–146.

Takao Furukawa, Kaori Mori, Kazuma Arino, Kazuhiro Hayashi, and Nobuyuki Shirakawa. 2014. Identifying the Evolutionary Process of Emerging Technologies: A Chronological Network Analysis of World Wide Web Conference Sessions. *Technological Forecasting and Social Changes*, 91:280–294.

David Hall, Daniel Jurafsky, and Christopher Manning. 2008. Studying the History of Ideas Using Topic Models. In *Proc. of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP 2008)*, pages 363–371.

John Hopcroft, Omar Khan, Brian Kulis, and Bart Selman. 2004. Tracking Evolving Community in Large Linked Networks. In *Proc. of the National Academy of Sciences (PNAS)*, 101(suppl.1):5249–5253.

Kalervo. Järvelin and Jaana. Kekäläinen. 2000. IR Evaluation Methods for Retrieving Highly Relevant Documents. In *Proc. of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2000)*, pages 41–48.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1746–1751.

Jon M. Kleinberg. 1999. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM (JACM)*, 46(5):604-632, 1999.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Texts. In *Proc. of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, pages 404–411.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proc. of the Workshop at the 1st International Conference on Learning Representations (ICLR 2013)*.

Fabian Mörchen, Mathäus Dejori, Dmitriy Fradkin, Julien Etienne, Bernd Wachmann, and Markus Bundschus. 2008. Anticipating Annotations and Emerging Trends in Biomedical Literature. In *Proc. of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08)*, pages 954–962.

Mark E. J. Newman. 2004. Fast Algorithm for Detecting Community Structure in Networks. *Physical Review E*, 69(6):066133.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1998. The PageRank Citation Ranking: Bringing Order to the Web. In *Technical Report SIDL-WP-1999-0120*, Stanford Digital Library Technological Project.

Derek J. De Solla Price. 1965. Networks of Scientific Papers. *Science*, 149(3683):510–515.

Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The Author-Topic Model for Authors and Documents. In *Proc. of the 20th Conference on Uncertainty in Artificial Intelligence (UAI 2004)*, pages 487–494.

Gerald Salton and Michael J. McGill. 1983. Introduction to Modern Information Retrieval. McGraw-Hill, 1983.

Naoki Shibata, Yuya Kajikawa, Yoshiyuki Takeda, and Katsumori Matsushima. 2008. Detecting Emerging Research Fronts Based on Topological Measures in Citation Networks of Scientific Publications. *Technovation*, pages 758–775.

Qi Wang. 2017. A Bibliometric Model for Identifying Emerging Research Topics. *Journal of the Association for Information Science and Technology*, 69(2):290–304.

Yujing Wang, Yunhai Tong, and Ming Zeng. 2013. Ranking Scientific Articles by Exploiting Citations, Authors, Journals, and Time Information. In *Proc. of the 27th AAAI Conference on Artificial Intelligence (AAAI-13)*, pages 933–939.

Ian Witten, Gordon Paynter, Eibe Frank, Carl Gutwin, and Craig Nevill-Manning. 1999. KEA: Practical Automatic Keyphrase Extraction. In *Proc. of the 4th ACM Conference on Digital Libraries (DL '99)*, pages 254–255.

Ying Yan, Qingping Tan, Qinzheng Xie, Ping Zeng, and Panpan Li. 2017. A Graph-based Approach of Automatic Keyphrase Extraction. *Procedia Computer Science, 107 (C)*:248–255.

Yuxiang Zhang, Yaocheng Chang, Xiaoqing Liu, Sujatha Das Gollapalli, Xiaoli Li, and Chunjing Xiao. 2017. MIKE: Keyphrase Extraction by Integrating Multidimensional Information In *Proc. of the 27th International Conference on Information and Knowledge Management (CIKM'17)*, pages 1349–1358.

Embedding WordNet Knowledge for Textual Entailment

Yunshi Lan
School of Information Systems
Singapore Management University
yslan.2015@phdis.smu.edu.sg

Jing Jiang
School of Information Systems
Singapore Management University
jingjiang@smu.edu.sg

Abstract

In this paper, we study how we can improve a deep learning approach to textual entailment by incorporating lexical entailment relations from WordNet. Our idea is to embed the lexical entailment knowledge contained in WordNet in specially-learned word vectors, which we call "entailment vectors." We present a standard neural network model and a novel set-theoretic model to learn these entailment vectors from word pairs with known lexical entailment relations derived from WordNet. We further incorporate these entailment vectors into a decomposable attention model for textual entailment and evaluate the model on the SICK and the SNLI dataset. We find that using these special entailment word vectors, we can significantly improve the performance of textual entailment compared with a baseline that uses only standard word2vec vectors. The final performance of our model is close to or above the state of the art, but our method does not rely on any manually-crafted rules or extensive syntactic features.

1 Introduction

Recognizing textual entailment (RTE) is the task of determining whether a hypothesis sentence can be inferred from a given premise sentence. The task has been well studied since it was first introduced by Dagan et al. (2006). Recently, there has been much interest in applying deep learning models to RTE (Bowman et al., 2015; Rocktaschel et al., 2016; Wang and Jiang, 2016; Parikh et al., 2016; Sha et al., 2016). These models usually do not perform any linguistic analysis or require any feature engineering but have been shown to perform very well on this task.

Intuitively, lexical-level entailment relations should help sentence-level RTE. For example, Table 1 shows a premise and a hypothesis taken from the test data of the SICK dataset (Marelli et al., 2014). We can see that in this example the premise entails the hypothesis, and in order to correctly identify this relation, one has to know that the word *kettle* entails the word *pot*. However, if we train a neural network model on a set of labeled sentence pairs, and if the training dataset does not contain the word pair *kettle* and *pot* anywhere, it would be hard for the learned model to know that *kettle* entails *pot* and subsequently predict the relation between the premise and the hypothesis to be *entailment*. On the other hand, from WordNet we can easily find out that *pot* is a direct hypernym of *kettle* and therefore *kettle* should entail *pot*. If this kind of prior knowledge can be injected into a trained RTE model, then for sentence pairs with words that do not occur in the training data but can be found in WordNet, the RTE predictions could potentially be made easier.

Premise: *Someone is stirring chili in a* ***kettle****.*
Hypothesis: *Someone is stirring chili in a* ***pot****.*

Table 1: An example of a pair of premise and hypothesis from SICK.

Indeed, WordNet (Miller, 1995) knowledge has been used to help RTE in a number of previous studies (Corley and Mihalcea, 2005; Beltagy et al., 2016; Martinez-Gomez et al., 2017). However, these

Proceedings of the 27th International Conference on Computational Linguistics, pages 270–281
Santa Fe, New Mexico, USA, August 20-26, 2018.

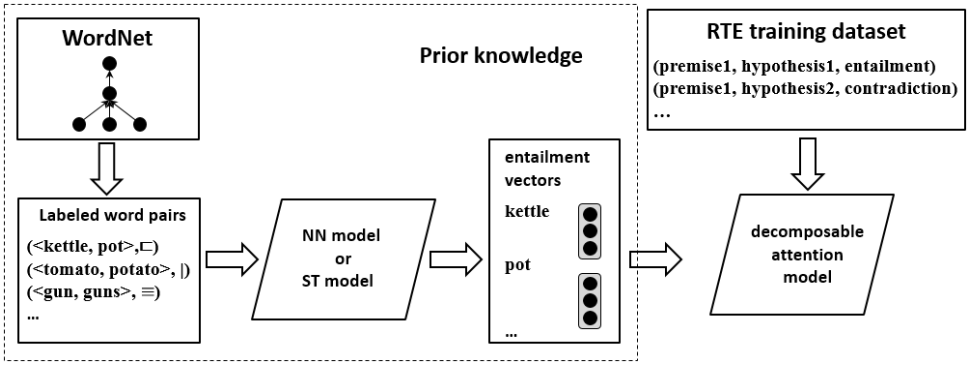


Figure 1: An overview of our approach.

previous studies were not based on neural network models, and thus their base models without using WordNet may not represent the state of the art or may require much human effort. For example, our baseline model based on a neural network model without using any WordNet knowledge can achieve an accuracy of 84.1% on the SICK test data, but the baseline model by Martinez-Gomez et al. (2017), which uses logical semantic representations and a theorem prover, can only achieve an accuracy of 76.7%. Given the advantages of deep learning approaches to RTE, it is therefore desirable to incorporate WordNet knowledge into deep learning based solutions to RTE. However, it is not immediately clear how WordNet knowledge could be easily brought into neural network models. To the best of our knowledge, there has not been any previous work in this direction.

In this paper, we propose to first embed lexical entailment knowledge contained in WordNet into word vectors. We call these special word vectors "entailment vectors." We then incorporate these entailment vectors into a recently proposed decomposable attention model for RTE. To learn these entailment vectors that encode lexical entailment relations, we can use a standard neural network. We also propose a set-theoretic model that has better theoretical justification. Our experiments show that using these entailment vectors learned from WordNet can indeed significantly improve the performance of RTE on the SICK dataset and the SNLI dataset. The performance of our method is also better compared with the state of the art on SICK.

2 Method

In this section, we present our method that learns the entailment vectors that encode prior knowledge of lexical entailment relations. We also present how we incorporate these entailment vectors into a neural network model for RTE. The overall framework of our method is illustrated in Figure 1.

2.1 Learning Entailment Vectors

We assume that our prior knowledge of lexical entailment relations is contained in word pairs and their entailment relations. Specifically, let $\mathcal{R}$ denote a set of lexical entailment relations. For example, $\mathcal{R}$ may contain *entailment* and *reverse-entailment*. Let $\mathcal{L} = \{(w_{i,1}, w_{i,2}, r_i)\}_{i=1}^{N}$ denote a set of N word pairs together with their relation labels, where $w_{i,1}$ and $w_{i,2}$ are two words and $r_i \in \mathcal{R}$. For example, $\mathcal{L}$ may contain the triplet (<*pot*, *kettle*>, *reverse-entailment*). Let $\mathcal{V}$ denote the set of all unique words found in $\mathcal{L}$.

In order to encode the lexical entailment knowledge contained in $\mathcal{L}$, we propose to learn a dense vector for each word $w \in \mathcal{V}$ such that the entailment relation between two words in $\mathcal{V}$ can be easily detected from their word vectors. We refer to these dense word vectors as "entailment vectors." Note that these entailment vectors are different from commonly-used word embeddings such as GloVe and word2vec, because the entailment vectors are not learned from a large corpus and thus do not encode the distributional properties of words. They are learned from labeled word pairs in $\mathcal{L}$. We will show later that they can be used in combination with common word embeddings such as word2vec to help RTE.

In what follows, we first describe what lexical entailment relations we include in $\mathcal{R}$. We then present

Name	Symbol	Set-theoretic definition	Example
(strict) entailment	$x_1 \sqsubset x_2$	$x_1 \subset x_2$	woman, person
(strict) reverse entailment	$x_1 \sqsupset x_2$	$x_1 \supset x_2$	person, woman
equivalence	$x_1 \equiv x_2$	$x_1 = x_2$	couch, sofa
alternation	$x_1 \mid x_2$	$x_1 \cap x_2 = \varnothing \wedge x_1 \cup x_2 \neq \mathcal{D}$	woman, man
negation	$x_1 \wedge x_2$	$x_1 \cap x_2 = \varnothing \wedge x_1 \cup x_2 = \mathcal{D}$	able, unable
cover	$x_1 \smile x_2$	$x_1 \cap x_2 \neq \varnothing \wedge x_1 \cup x_2 = \mathcal{D}$	person, non-woman
independence	$x_1 \# x_2$	(else)	woman, doctor

(a)

	$c_{0,0}$	$c_{0,1}$	$c_{1,0}$	$c_{1,1}$
$x_1 \sqsubset x_2$	–	> 0	$= 0$	–
$x_1 \sqsupset x_2$	–	$= 0$	> 0	–
$x_1 \equiv x_2$	–	$= 0$	$= 0$	–
$x_1 \mid x_2$	> 0	–	–	$= 0$
$x_1 \wedge x_2$	$= 0$	–	–	$= 0$
$x_1 \smile x_2$	$= 0$	–	–	> 0
$x_1 \# x_2$	> 0	–	–	> 0

(b)

Table 2: (a) Seven basic semantic relations defined by MacCartney and Manning (2009). Here $\mathcal{D}$ is the universe that contains all entities. Each x_1 (or x_2) is a language unit that represents a subset of $\mathcal{D}$. (b) Criteria for different basic semantic relations based on the counters $c_{0,0}$, $c_{0,1}$, $c_{1,0}$ and $c_{1,1}$.

two different neural network models used to learn the entailment vectors from $\mathcal{L}$. We defer the description of how we derive labeled word pairs $\mathcal{L}$ from WordNet until Section 2.3.

Lexical Entailment Relations

Recall that eventually the entailment vectors will be used for textual entailment. In standard textual entailment datasets such as SICK and SNLI, there are three sentence-level entailment relations: *entailment*, *contradiction* and *neutral*. However, at word-level the entailment relations are different.

In a previous study, MacCartney and Manning (2009) developed a framework for natural language inference. As the basis of their framework, they defined seven basic semantic relations between language units, which we show in Table 2a. These relations cover all the possible relations between two language units x_1 and x_2. We can see that each relation has an equivalent set-theoretic definition, which dates back to Montague Semantics (Janssen, 2011). Each language unit x_1 or x_2 is modeled as a set, and is supposedly a subset of the universe $\mathcal{D}$. Therefore their relations can be determined by the relation between the two sets. For example, if the language units we consider are nouns, we can regard $\mathcal{D}$, the universe, as the set of all entities in the world. If x_1 is the word "person," we can regard x_1 as a set that contains all people. If x_2 is the word "woman," we can regard x_2 as a set that contains all people who are female. Since x_1 is a superset of x_2, based on the definition, x_1 reversely entails x_2, or $x_1 \sqsupset x_2$. This makes sense because "person" reversely entails "woman," or in other words, "woman" entails "person."

In this work, we only use a subset of these relations, namely, *entailment* ($\sqsubset$), *reverse-entailment* ($\sqsupset$), *alternation* ($|$) and *equivalence* ($\equiv$). In other words, $\mathcal{R} = \{\sqsubset, \sqsupset, |, \equiv\}$. We do not include *negation*, *cover* or *independence* because we find it hard to automatically derive word pairs with these relations from WordNet.

Standard Neural Network Model

To learn entailment vectors from $\mathcal{L}$, we first consider a standard neural network model. Let $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^d$ denote the entailment vectors of two words, which we are trying to learn, where d is the number of dimensions of the vectors. We can combine the two vectors into a single vector as follows:

$$\mathbf{h} = \tanh(\mathbf{M}\begin{bmatrix}\mathbf{w}_1 \\ \mathbf{w}_2\end{bmatrix} + \mathbf{b}),$$

where $\mathbf{M} \in \mathbb{R}^{l \times 2d}$ is a weight matrix, $\mathbf{b} \in \mathbb{R}^l$ is a weight vector, $\tanh(\cdot)$ is applied element-wise to the vector, $\mathbf{h} \in \mathbb{R}^l$ is a hidden vector and l is the number of dimensions of the hidden vector.

The hidden vector $\mathbf{h}$ can then go through a linear transform followed by a softmax layer to be used to predict the relation label r between the two words:

$$p(r \mid \mathbf{h}) = \text{softmax}(\mathbf{M}'\mathbf{h} + \mathbf{b}'), \quad (1)$$

where $\mathbf{M}' \in \mathbb{R}^{k \times l}$ and $\mathbf{b}' \in \mathbb{R}^k$ denote a weight matrix and a bias vector, and $k = |\mathcal{R}|$.

Given all the labeled word pairs in $\mathcal{L}$, we can use the cross entropy loss as the objective function to learn the entailment vector for each $w \in \mathcal{V}$ as well as the various parameters above.

Set-Theoretic Model

Although the standard neural network model above is straightforward, it is hard to explain how the learned word vectors can encode the lexical entailment relations. Inspired both by the set-theoretic definitions of the basic semantic relations shown in Table 2a and by some recent work on formal distributional semantics (Grefenstette, 2013; Rocktaschel et al., 2014), we hypothesize that a good entailment vector for a word essentially encodes which elements in $\mathcal{D}$ are members of the set representing this word. We now present a novel set-theoretic model to learn the entailment vectors.

To illustrate our idea, we first show that in the extreme case when word vectors we try to learn are binary vectors precisely representing set memberships, the semantic relation between two words can be determined by comparing the two vectors element-wise. Specifically, let D denote the size of the universe $\mathcal{D}$. Since a word x can be regarded as a subset of $\mathcal{D}$, we assume that the entailment vector of x is a D-dimensional binary vector $\mathbf{x}$, where $\mathbf{x}_i$ is 1 if the i^{th} element in $\mathcal{D}$ is inside the set x and 0 otherwise. Given two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ representing two words, in order to determine the relationship between them, we need to check the relationship between the two sets. To do so, we define the following counters for $p, q \in \{0, 1\}$:

$$c_{p,q} \quad = \quad \sum_{i=1}^{D} \delta(x_{1,i}, p)\delta(x_{2,i}, q),$$

where $\delta(s, t)$ is 1 if s is equal to t and 0 otherwise. Essentially if we compare $\mathbf{x}_1$ and $\mathbf{x}_2$ element-wise, then $c_{0,0}$, $c_{0,1}$, $c_{1,0}$ and $c_{1,1}$ count the number of times we see (0, 0), (0, 1), (1, 0) and (1, 1), respectively. It is not hard to show that the seven basic semantic relations in Table 2a correspond to different values of these $c_{p,q}$. For example, if $c_{0,1}$ is 0 and $c_{1,0}$ is 0, then the two sets (words) are equivalent. If $c_{0,1}$ is positive and $c_{1,0}$ is 0, then $\mathbf{x}_1$ is a subset of $\mathbf{x}_2$, and therefore the first word entails the second word. Table 2b shows the criteria for each basic semantic relation based on the values of these counters $c_{0,0}$, $c_{0,1}$, $c_{1,0}$ and $c_{1,1}$.

We can use a simple example to illustrate the idea above. Suppose the universe contains four elements: $\mathcal{D} = \{dog, cat, tiger, computer\}$. The word *animal* is a set that contains *dog*, *cat* and *tiger*, and thus can be represented by the vector $[1, 1, 1, 0]$. The word *pet* should contain only *dog* and *cat*, so it can be represented by the vector $[1, 1, 0, 0]$. In this case, the values of the four counters are the following: $c_{0,0} = 1$, $c_{0,1} = 0$, $c_{1,0} = 1$, and $c_{1,1} = 2$. According to Table 2b, we can determine that *animal* reversely entails *pet* based on the values of these counters.

Generally speaking, however, the number of elements in the universe is huge and therefore it is not feasible to learn word vectors that precisely represent the memberships of all these elements. In the following model we propose, we do not strictly force the entailment vectors to be binary. We also introduce a hidden layer that is deterministically derived from the entailment vectors. This hidden layer essentially represents $c_{0,0}$, $c_{0,1}$, $c_{1,0}$ and $c_{1,1}$, and it is used for the prediction of the relation between two words.

Specifically, let $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^d$ represent the entailment vectors corresponding to words w_1 and w_2. Next, we introduce two complementary vectors:

$$\bar{\mathbf{w}}_1 = \mathbf{1} - \mathbf{w}_1, \qquad \bar{\mathbf{w}}_2 = \mathbf{1} - \mathbf{w}_2,$$

where $\mathbf{1}$ is a d-dimensional vector of 1s.

We then define a hidden vector $\mathbf{h}$ to be a 4-dimensional vector as follows:

$$\mathbf{h} = \frac{1}{d}\begin{bmatrix} \mathbf{w}_1^\top \mathbf{w}_2 & \mathbf{w}_1^\top \bar{\mathbf{w}}_2 & \bar{\mathbf{w}}_1^\top \mathbf{w}_2 & \bar{\mathbf{w}}_1^\top \bar{\mathbf{w}}_2 \end{bmatrix}^\top. \tag{2}$$

We can see that this hidden vector $\mathbf{h}$ roughly correspond to the four counters $c_{0,0}$, $c_{0,1}$, $c_{1,0}$ and $c_{1,1}$ defined above if the entailment vectors are binary vectors. But since we do not have the binary constraint, $\mathbf{h}$ actually contains real values rather than integer values. We also perform normalization by dividing the counts by d.

The same as shown in Eqn. (1), we can use $\mathbf{h}$ to predict the relation label between two words through a linear transform and a softmax layer. We can then again use the cross entropy loss as the objective function to learn the entailment vectors and the model parameters.

2.2 Using Entailment Vectors for RTE

Given the entailment vectors learned from $\mathcal{L}$ using either the standard neural network model or our proposed set-theoretic model, we can use them in a neural network model for textual entailment. In this paper we use a slightly modified version of the decomposable attention model proposed by Parikh et al. (2016) because of its relative simplicity and good performance on the SNLI (Bowman et al., 2015) dataset.

The setup of the textual entailment task is as follows. We are given two input sentences: a premise $X = ((\mathbf{x}_1, \mathbf{x}'_1), (\mathbf{x}_2, \mathbf{x}'_2), \ldots, (\mathbf{x}_m, \mathbf{x}'_m))$ and a hypothesis $Z = ((\mathbf{z}_1, \mathbf{z}'_1), (\mathbf{z}_2, \mathbf{z}'_2), \ldots, (\mathbf{z}_n, \mathbf{z}'_n))$, where each $\mathbf{x}_i$ (or $\mathbf{z}_j$) is a standard word embedding such as the word2vec embedding of the i^{th} (or j^{th}) word in the premise (or hypothesis), and $\mathbf{x}'_i$ (or $\mathbf{z}'_j$) is the newly-learned entailment vector of the i^{th} (or j^{th}) word in the premise (or hypothesis), as described in Section 2.1.[1] The goal is to predict a label $y \in \{\textit{entailment}, \textit{contradiction}, \textit{neutral}\}$ from X and Z.

The Modified Decomposable Attention Model

The decomposable attention model consists of the following three steps.

Attend: At this step, we derive attention weights from the word embeddings from X and Z. We first define

$$e_{ij} = F(\mathbf{x}_i)^\top F(\mathbf{z}_j),$$
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'=1}^{n} \exp(e_{ij'})}, \qquad \beta_{ij} = \frac{\exp(e_{ij})}{\sum_{i'=1}^{m} \exp(e_{i'j})},$$

where $F(\cdot)$ is a standard single-layer feed-forward neural network with ReLU activations (Glorot et al., 2011). We then define

$$\tilde{\mathbf{x}}_i = \sum_{j=1}^{n} \alpha_{ij} \cdot \mathbf{z}_j, \qquad \tilde{\mathbf{x}}'_i = \sum_{j=1}^{n} \alpha_{ij} \cdot \mathbf{z}'_j,$$
$$\tilde{\mathbf{z}}_j = \sum_{i=1}^{m} \beta_{ij} \cdot \mathbf{x}_i, \qquad \tilde{\mathbf{z}}'_j = \sum_{i=1}^{m} \beta_{ij} \cdot \mathbf{x}'_i.$$

$\tilde{\mathbf{x}}_i$ is a weighted version of the word embeddings from Z to match $\mathbf{x}_i$, and similarly $\tilde{\mathbf{x}}'_i$ is a weighted version of the entailment vectors from Z to match $\mathbf{x}'_i$. The same idea applies to $\tilde{\mathbf{z}}_j$ and $\tilde{\mathbf{z}}'_j$.

Compare: At this step, another single-layer feed-forward neural network G with ReLU activations is used to compare the aligned words. Normally, without considering the entailment vectors, the comparison is done as follows:

$$\mathbf{v}_{1,i} = G(\mathbf{x}_i \oplus \tilde{\mathbf{x}}_i), \qquad \mathbf{v}_{2,j} = G(\mathbf{z}_j \oplus \tilde{\mathbf{z}}_j),$$

where $\oplus$ is concatenation, and $\mathbf{v}_{1,i}$ and $\mathbf{v}_{2,j}$ represent the comparison results for the i-th word in X and the j-th word in Z, respectively.

When we do consider the entailment vectors $\mathbf{x}'_i$ and $\mathbf{z}'_j$, as well as their counterparts $\tilde{\mathbf{x}}'_i$ and $\tilde{\mathbf{z}}'_j$ from the **Attend** step, we need to perform the comparison slightly differently. Here we use two different ways to perform the comparison, depending on whether the entailment vectors are learned using the standard neural network model or the set-theoretic model. If the standard neural network model is used to learn the entailment vectors, we simply concatenate all the vectors to perform comparison as follows:

$$\mathbf{v}_{1,i} = G(\mathbf{x}_i \oplus \tilde{\mathbf{x}}_i \oplus \mathbf{x}'_i \oplus \tilde{\mathbf{x}}'_i), \qquad \mathbf{v}_{2,j} = G(\mathbf{z}_j \oplus \tilde{\mathbf{z}}_j \oplus \mathbf{z}'_j \oplus \tilde{\mathbf{z}}'_j).$$

[1] For those words which do not appear in $\mathcal{L}$ (the set of unique words in the training data), we randomly generate their entailment vectors in the range $(-0.1, 0.1)$.

However, if the entailment vectors are learned using the set-theoretic model, because of the special properties of the entailment vectors as explained in Section 2.1, we perform comparison as follows:

$$\mathbf{v}_{1,i} = G(\mathbf{x}_i \oplus \tilde{\mathbf{x}}_i \oplus (\mathbf{x}'_i \odot \tilde{\mathbf{x}}'_i) \oplus (\mathbf{x}'_i \odot (1 - \tilde{\mathbf{x}}'_i)) \oplus ((1 - \mathbf{x}'_i) \odot \tilde{\mathbf{x}}'_i) \oplus ((1 - \mathbf{x}'_i) \odot (1 - \tilde{\mathbf{x}}'_i))),$$
$$\mathbf{v}_{2,j} = G(\mathbf{z}_j \oplus \tilde{\mathbf{z}}_j \oplus (\mathbf{z}'_j \odot \tilde{\mathbf{z}}'_j) \oplus (\mathbf{z}'_j \odot (1 - \tilde{\mathbf{z}}'_j)) \oplus ((1 - \mathbf{z}'_j) \odot \tilde{\mathbf{z}}'_j) \oplus ((1 - \mathbf{z}'_j) \odot (1 - \tilde{\mathbf{z}}'_j))),$$

where $\odot$ is element-wise multiplication of two vectors.

Aggregate: Given $\mathbf{v}_{1,i}$ $(1 \le i \le m)$ and $\mathbf{v}_{2,j}$ $(1 \le j \le n)$ as defined above, we first use a standard LSTM model to process the two sequences separately, and then we use MaxPooling to aggregate the derived hidden vectors in order to obtain a single vector for each sequence:

$$\mathbf{h}_{1,i} = \text{LSTM}(\mathbf{v}_{1,i}, \mathbf{h}_{1,i-1}), \qquad \mathbf{h}_{2,j} = \text{LSTM}(\mathbf{v}_{2,j}, \mathbf{h}_{2,j-1}),$$
$$\mathbf{v}_1 = \text{MaxPooling}(\mathbf{h}_{1,1}, \mathbf{h}_{1,2}, \ldots, \mathbf{h}_{1,m}), \qquad \mathbf{v}_2 = \text{MaxPooling}(\mathbf{h}_{2,1}, \mathbf{h}_{2,2}, \ldots, \mathbf{h}_{2,n}).$$

We then use $\mathbf{v}_1$ and $\mathbf{v}_2$ to make the final prediction:

$$p(y|\mathbf{v}_1, \mathbf{v}_2) = \text{softmax}(\mathbf{W}(\mathbf{v}_1 \oplus \mathbf{v}_2) + \mathbf{c}),$$

where $\mathbf{W}$ and $\mathbf{c}$ are parameters to be learned.

2.3 Implementation Details

In this section, we describe how we derive labeled word pairs from WordNet to construct $\mathcal{L}$. Because eventually we will test the learned entailment vectors on the SICK and SNLI textual entailment datasets, we focus on extracting word pairs with at least one word appearing in SICK or SNLI.

Specifically, we follow the following steps to construct $\mathcal{L}$. In all the steps, w_1 and w_2 refer to nouns found in WordNet.

- If the synset of w_1 is a hypernym of the synset of w_2, we add $(w_1, w_2, \sqsupset)$ and $(w_2, w_1, \sqsubset)$ to $\mathcal{L}$.
- If w_1 and w_2 are in the same synset, or w_1 (w_2) is the plural form of w_2 (w_1), we add $(w_1, w_2, \equiv)$ and $(w_2, w_1, \equiv)$ to $\mathcal{L}$.
- If the synsets of w_1 and w_2 share the same parent synset, we add $(w_1, w_2, |)$ and $(w_2, w_1, |)$ to $\mathcal{L}$.
- We remove word pairs from $\mathcal{L}$ where neither word occurs in the SICK or the SNLI dataset.

3 Experiments

3.1 Direct Evaluation of Entailment Vectors

As a first step to evaluate our method, we first test whether our entailment vectors learned from labeled word pairs can indeed encode lexical entailment relations. To do this, we report the prediction accuracy on $\mathcal{L}$. Recall that $\mathcal{L}$ is derived from nouns in WordNet. In total we have 13,029 unique words and 67,935 labeled word pairs in $\mathcal{L}$. The four entailment relations are quite evenly distributed except for alternation, which has roughly twice as many word pairs as the other relations.

We compare the entailment vectors learned from the standard neural network model and from the set-theoretic model. We refer to these two as **NN** and **ST**. In addition, we also consider one baseline method, which train an SVM with RBF kernel on the word2vec embeddings of two words to predict their semantic relation.[2] We refer to this baseline as **word2vec**.

For all the methods, we take two thirds of $\mathcal{L}$ for training/validation and the remaining one third for testing. We repeat this three times and report the average accuracies. We tune the hyperparameters as follows: The dimensionality of the entailment vectors is chosen from $\{25, 50, 100, 200\}$ and the learning rate from $\{0.05, 0.1, 0.15\}$. We applied stochastic gradient descent with a mini-batch size of 5 for optimization.

[2]We have also tried to use a standard neural network model instead of SVM on the word2vec embeddings, and the performance is 68.5%, which is similar to using SVM.

	Acc±SD	F1±SD	d	Acc(CS)	F1(CS)
NN	90.19±0.005	89.24±0.458	50	13.35	23.56
ST	**94.51**±0.002	**93.96**±0.106	200	**93.32**	**96.55**
word2vec	69.01±0.002	67.70±0.354	300	53.00	69.28

(a)

Dataset	SICK(ss)	SICK(ns)	SNLI(ss)	SNLI(ns)
train	4439	7163	549367	460451
dev	485	795	9842	51695
test	4906	1882	9824	61054

(b)

Model	SICK (ss)		SICK (ns)		SNLI (ss)		SNLI (ns)	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test
w2v	84.2	84.1	85.4	81.8	86.5	86.2	84.7	82.8
w2v+NN	84.4	85.1*	**86.3**	84.9**	86.6	86.3	85.0	83.3**
w2v+ST	**84.6**	**85.4****	85.8	**85.1****	**86.7**	86.5	**85.3**	**83.6****
MLN-eclassif	-	85.1	-	-	-	-	-	-
DAM	-	-	-	-	-	86.3	-	-
CAFE	-	-	-	-	-	**88.5**	-	-

(c)

Table 3: (a) Results of direct evaluation on $\mathcal{L}$. d is the optimal dimensionality of the entailment vectors (for NN and ST) or of the word embeddings (for word2vec) based on validation. SD stands for standard deviation. (b) Numbers of sentence pairs in the SICK and the SNLI datasets. (c) Textual entailment results on the two datasets by different methods. MLN-eclassif (Beltagy et al., 2016) and CAFE (Tay et al., 2018) represent the state of the art on the SICK and the SNLI datasets, respectively. DAM (Beltagy et al., 2016) is the original decomposable attention model. The numbers shown in the table are what was reported in these papers. ** and * indicate statistical significance ($p \leq 0.01$) and ($p \leq 0.05$) compared with w2v by McNemars test, respectively.

We show the results in terms of accuracy and F1 in Table 3a. We can observe the following from the table. (1) The entailment vectors give better accuracies for predicting the entailment relations compared with using word2vec embeddings. (2) Between the standard neural network model and our set-theoretic model, the set-theoretic model achieves better performance.

Levy et al. (2015) previously showed that for some models using word embedding vectors to identify hypernyms, the model is essentially memorizing "prototypical" hypernyms. In order to check whether this happens to our model, we create a special set of labeled word pairs in which we introduce some "confusing" word pairs. We use the following rule to generate these word pairs: If we know ($\langle w_2, w_1 \rangle, \sqsubset$) and ($\langle w_1, w_3 \rangle, |$), then we can infer that ($\langle w_2, w_3 \rangle, |$). We obtain 719 labeled word pairs in this way. For example, ($\langle$*staple gun, musical instrument*$\rangle, |$) is one in this set. We denote this dataset as confusion set (CS). The accuracy and F1 scores are shown in the last two columns of Table 3a, respectively. We can see that the entailment vectors learned using the standard neural network models perform very poorly on this special set, suggesting that the learned entailment vectors may be memorizing the prototypical hypernyms. Using word2vec embeddings performs poorly, too. However, entailment vectors learned from the set-theoretic model perform very well, suggesting that the set-theoretic model we proposed does not simply learn "prototypical" hypernyms.

3.2 Evaluation on Textual Entailment

In this section, we evaluate whether the learned entailment vectors can help RTE. Here the entailment vectors are trained using the entire set of labeled word pairs derived from WordNet as we have described in Section 2.3. We use the SICK dataset and SNLI dataset for this experiment.

Recall that we have earlier hypothesized that when there are many word pairs in the test data that are not found in the training data, external lexical entailment knowledge about these word pairs is especially important. In order to verify this hypothesis, we also construct a different split of the data. We use the following steps to build this split:

- For every sentence pair, find all possible matching word pairs that occur in the WordNet data. E.g., we extract ⟨someone, someone⟩, ⟨stirring, stirring⟩, ⟨chili, chili⟩ and ⟨kettle, pot⟩ from the sentence pair in Table 1.
- For every word pair, randomly assign all sentence pairs containing such word pair into either the

training set or the test set. This is to ensure that the test set contains word pairs that have not appeared in the training data at all.
- Randomly split the training set into the final training set and a development set with a ratio of 9:1.

We denote the original split of the data as the "standard split" (ss) and the new split of the data as the "non-overlap split" (ns). This results in two additional specific-split datasets. Table 3b shows the numbers of sentence pairs of all the four datasets.

We compare the following settings of using entailment vectors in the decomposable attention model for RTE:

- w2v: This is the original decomposable attention model for RTE without using the entailment vectors. The word embeddings used are word2vec.
- w2v+NN: This setting uses the modified decomposable attention model with both word2vec word embeddings and the entailment vectors learned from the standard neural network model.
- w2v+ST: This setting uses the modified decomposable attention model with both word2vec word embeddings and the entailment vectors learned from the set-theoretic model.

The SICK dataset has a vocabulary of 2,331 words, 1,560 of which can be found in the WordNet dataset. For the SNLI dataset, the vocabulary contains 32,497 words, of which only 3,599 appear in the WordNet. During training, we experiment with 2-layer feed-forward neural networks with 200 neurons in all sets. Learning rate is set to be 0.03 and batch size is 30. In order to better compare different models, we use the McNemar test to test the statistical significance of the performance differences between different models.

We show the experiment results in Table 3c. We have the following observations. (1) The result of our baseline w2v on the standard split of the SNLI dataset is almost the same as the one reported by Parikh et al. (2016). This shows that our baseline implement is as strong as the DAM model. (2) For the standard split of the datsets, both w2v+NN and w2v+ST can outperform w2v, and w2v+ST's accuracy is significantly better than w2v. In particular, our model outperforms the state-of-the-art performance on the SICK dataset. Even though the performance on SNLI is not better than the best result reported (Tay et al., 2018), where extensive syntactic features were used, we still can outperform the basic model (Parikh et al., 2016) by injecting WordNet knowledge into the model. (3) For the non-overlap split, especially on the SICK dataset, using only word2vec, the RTE performance on the development set is quite high, but when it comes to the test set, the performance drops drastically. This demonstrates that without observing the necessary word pairs from the training set, it is hard to make the right predictions on the test set. In contrast, using entailment vectors performs better in the non-overlap split setting. For the SNLI dataset, the improvement is not so obvious. This may be because SNLI has a low percentage of words that can be found in WordNet. (4) Comparing w2v+NN and w2v+ST, we can see that w2v+ST performs slightly better than w2v+NN most of the time. This shows the advantage of our set-theoretic model for learning the entailment vectors.

3.3 Further Analyses

We also conduct some further analyses on the results to better understand our method.

Analysis of the Set-Theoretic Model

Recall that for the set-theoretic model, the hidden vectors $\mathbf{h}$ as defined in Eqn. (2) roughly correspond to the four counters $c_{p,q}$, and these counters are essential to how the set-theoretic model works. To see whether the set-theoretic model indeed works as we have expected, we plot the values of $\mathbf{h}$ for some example word pairs. Figure 2a shows the $\mathbf{h}$ vectors for four word pairs from WordNet with different lexical entailment relations. Figure 2b shows the weights applied to $\mathbf{h}$ as defined in Eqn. (1) to make the final predictions of lexical entailment relations. We can see in Figure 2a that for the word pair $(\langle \text{child, juvenile} \rangle, \sqsubset)$, it has a large value for the counter $c_{0,1}$, and this counter has a positive weight for relation $\sqsubset$ in Figure 2b. On the other hand, for $\sqsupset$, the counter $c_{1,0}$ has a positive weight. Also, $|$ needs positive weights for $c_{1,0}$ and $c_{0,1}$ while $\equiv$ needs negative weights for them. All these match our expectations from Table 2b.

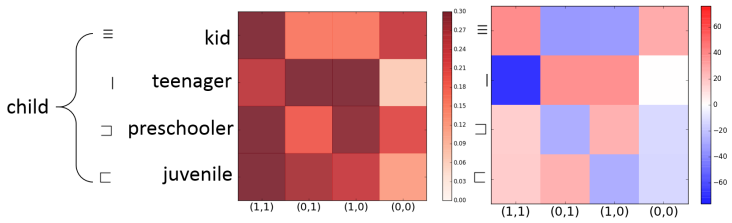


Figure 2: (a) Visualization of the 4-dimensional $\mathbf{h}$ for four types of relations with the corresponding word pairs labeled on the left and the meaning of the 4 dimensions labeled at the bottom. (b) Visualization of the weights applied to $\mathbf{h}$ with the corresponding relations labeled on the left and the meanings of the 4 dimensions labeled at the bottom.

	w2v	w2v+ST(concrete)	w2v+ST(abstract)	w2v+ST
SICK(ns)	81.8%	84.4%	84.0%	85.1%

Table 4: Evaluations of concrete concepts and abstract concepts on SICK(ns) data.

Analysis of Concrete and Abstract concepts

In Section 2.1, we show that our entailment vectors are mainly inspired by the set-theoretic definitions of basic semantic relations. Intuitively they work mostly for nouns presenting concrete concepts such as "animal" and "dog." It would be interesting to see how well they work for abstract nouns such as "idea" and "proposal." In order to check this, we conduct the following analysis.

We make use of an existing dataset[3] that contains concreteness ratings for 40,000 common English lemmas, collected using Amazon Mechanical Turk (Brysbaert et al., 2014). The ratings are from 0 to 5. We select words with scores 4.5 and higher to form our concrete concept list and words with scores 2.5 and lower to form another abstract concept list. This results in 596 concrete words and 250 abstract words. We then re-run the textual entailment experiments on the SICK data with the "non-overlap split" (ns). But instead of incorporating the entailment vectors of all words found in $\mathcal{L}$, we try two new settings, where we incorporate only the entailment vectors of the words found in the concrete word list or the abstract word list. We use entailment vectors learned from the set-theoretic model. We refer to these two new settings as w2v+ST(concrete) and w2v+ST(abstract). We show the performance in terms of accuracy in Table 4. For comparison, we also show w2v, which is our baseline that does not use any entailment vectors, and w2v+ST, which uses entailment vectors of all words. As we can see from the table, incorporating entailment vectors of concrete words works slightly better than of abstract words, but both settings perform better than the baseline w2v. This shows that even for abstract concepts, our model is still able to capture lexical-level entailment knowledge and incorporate such knowledge into the textual entailment model. Overall, incorporating entailment vectors of all words still works the best.

Case Studies

In Table 5 we show some successful as well as erroneous RTE predictions made by our method using the entailment vectors trained by the set-theoretic model. In the first example, since the words *kettle* and *pot* never co-occur in the training set, using only word2vec embeddings makes a wrong prediction. But injecting the entailment vectors trained using the set-theoretic model from the WordNet labeled word pairs, the knowledge that *kettle* entails *pot* is brought into the model, and therefore the w2v+ST method can correctly make a prediction. We can see that the w2v+NN method could not inject such knowledge either. The same thing happens in the next example, where WordNet provides the knowledge $(\langle \textit{inside, outside} \rangle, |)$, which is never learned from the training set.

We also provide two negative examples where w2v+ST makes wrong predictions. In the third example, the knowledge $(\langle \textit{lawn, field} \rangle, \sqsubset)$ provided by WordNet is not applicable, because here *lawn* and *field* are

[3]Available at `http://crr.ugent.be/archives/1330`

ID	sentence pair	ground truth	w2v	w2v+NN	w2v+ST	WordNet triplet
2727	Someone is stirring chili in a kettle Someone is stirring chili in a pot	*entailment*	*neutral*	*neutral*	*entailment*	$(\langle$*kettle, pot*$\rangle, \sqsubset)$
4633	The black and white dog is running inside The black and white dog is running outside	*contradiction*	*neutral*	*neutral*	*contradiction*	$(\langle$*inside, outside*$\rangle, \mid)$
4149	A shirtless man is playing football on a lawn A shirtless man is playing football on a field	*neutral*	*entailment*	*entailment*	*entailment*	$(\langle$*lawn, field*$\rangle, \sqsubset)$
686	A man is hanging up the phone A man is making a call on a cell phone	*contradiction*	*neutral*	*neutral*	*neutral*	

Table 5: Examples of successful and erroneous predictions.

considered to be different. The last example shows that for paraphrases not at the lexical level, our entailment vectors cannot help.

4 Related Work

Lexical entailment: Our work is related to recognizing lexical entailment. Many existing methods for lexical entailment are based on the distributional inclusion hypothesis and leverage co-occurrence information of words from a large corpus (Weeds et al., 2004; Geffet and Dagan, 2005; Kotlerman et al., 2010; Bernardi et al., 2012). In contrast, our work uses labeled word pairs derived from WordNet to learn entailment vectors to encode the lexical entailment relations. The semantic relations we consider are also designed specifically for natural language inference.

Word embeddings: Recently, neural-network-based approaches have been developed to learn vector representations of words (Mikolov et al., 2013; Pennington et al., 2014). These word embeddings are trained from an unlabeled large corpus and for general usage. In contrast, our entailment word vectors are not meant to be used as general-purpose word embeddings. They are designed specifically for incorporating external knowledge from WordNet into neural network models for RTE. Although there has been some work attempting to inject external lexical knowledge into word embeddings (Faruqui et al., 2015; Chen et al., 2015), they have not investigated the usage of these enhanced word embeddings for RTE.

Textual entailment with lexical knowledge: Many studies have attempted to inject lexical knowledge into the RTE task. Balkir et al. (2015) proposed compositional distributional models to extend representations from words to sentences. By mapping from a premise to a hypothesis by dependency trees or proposition extraction and then comparing lexical entailment between nodes or words, Herrera et al. (2005) and Ofoghi and Yearwood (2009) tried to solve RTE via WordNet. Beltagy et al. (2013), Beltagy et al. (2016) and Martinez-Gomez et al. (2017) first changed sentences to logical forms and then leveraged logic inference engines combined with lexical entailment axioms to make entailment judgments. But we are not aware of any work on how to apply lexical knowledge to textual entailment based on neural networks.

5 Conclusions

In this paper, we propose to learn entailment vectors that encode lexical entailment knowledge from WordNet. We incorporate such entailment vectors into a decomposable attention model for RTE. Our empirical evaluation shows that the entailment vectors can indeed improve the performance of RTE when evaluated on the SICK and the SNLI datasets. In the future, we plan to study how to automatically obtain more training data for learning entailment vectors and how to encode phrase-level entailment relations in such entailment vectors. Our data and code have been made publicly available.[4]

Acknowledgment

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its International Research Centres in Singapore Funding Initiative.

[4] https://github.com/lanyunshi/embedding-for-textual-entailment

References

Esma Balkir, Dimitri Kartsaklis, and Mehrnoosh Sadrzadeh. 2015. Sentence entailment in compositional distributional semantics. In *Proceedings of Conference on International Symposium on Artificial Intelligence and Mathematics*.

Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013. Montague meets Markov: Deep semantics with probabilistic logical form. In *Proceedings of the Joint Conference on Lexical and Computational Semantics*.

Islam Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J. Mooney. 2016. Representing meaning with a combination of logical and distributional models. *Computational Linguistics*, 42:763–808.

Macro Baroni Raffaella Bernardi, Ngoc-Quynh Do, and Chung chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of Conference of the European Chapter of the ACL*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior Research Methods*, 46:904–911.

Zhigang Chen, Wei Lin, Qian Chen, Xiaoping Chen, Si Wei, Hui Jiang, and Xiaodan Zhu. 2015. Revisiting word embedding for contrasting meaning. In *Proceedings of Annual Conference of the Association for Computational Linguistics*.

Courtney Corley and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proceedings of ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. *Machine Learning Challenges*, 3944:177–190.

Manaal Faruqui, Jesse Dodge, Sujay K.Jauhar, Chris Dyer Eduard Hovy, and Noah A.Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of Conference of the North American Chapter of the ACL*.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of Annual Conference of the Association for Computational Linguistics*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of International Conference on Artificial Intelligence and Statistics*.

Edward Grefenstette. 2013. Towards a formal distributional semantics: Simulating logical calculi with tensors. In *Proceedings of Joint Conference on Lexical and Computational Semantics*.

Jesus Herrera, Anselmo Penas, and Felisa Verdejo. 2005. Textual entailment recognition based on dependency analysis and WordNet. In *Proceedings of the 1st PASCAL Recognising Textual Entailment*.

Theo M. V. Janssen. 2011. Montague semantics. The Stanford Encyclopedia of Philosophy. `http://plato.stanford.edu/archives/win2011/entries/montague-semantics/`.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexcial inference. *Natural Language Engineering*, 16:359–389.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of Conference of the North American Chapter of the ACL*.

Bill MacCartney and Christopher D. Manning. 2009. An extended model of natural logic. In *Proceedings of International Conference on Computational Semantics*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of International Conference on Language Resources and Evaluation*.

Pascual Martinez-Gomez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2017. On-demand injection of lexical knowledge for recognising textual entailment. In *Proceedings of Conference of the European Chapter of the ACL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of International Conference on Learning Representations*.

George A. Miller. 1995. WordNet: A lexical database for english. *Communications of the ACM*, 38:39–41.

Bahadorreza Ofoghi and John Yearwood. 2009. From lexical entailment to recognizing textual entailment using linguistic resources. In *Proceedings of the Australasian Language Technology Association Workshop*.

Ankur P. Parikh, Osar Taskstrom, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Tim Rocktaschel, Matko Bosnjak, Sameer Singh, and Sebastian Riedel. 2014. Low-dimensional embeddings of logic. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*.

Tim Rocktaschel, Edward Grefenstette, Karl Moritz Hermann, Tomas Kocisky, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of International Conference on Learning Representations*.

Lei Sha, Baobao Chang, Zhifang Sui, and Sujian Li. 2016. Reading and thinking: Re-read LSTM unit for textual entailment recognition. In *Proceedings of the 26th International Conference on Computational Linguistics*.

Y. Tay, L. A. Tuan, and S. Cheung Hui. 2018. A Compare-Propagate Architecture with Alignment Factorization for Natural Language Inference. *ArXiv e-prints: 1801.00102*.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with LSTM. In *Proceedings of Conference of the North American Chapter of the ACL.*

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of International Conference on Computational Linguistics*.

Attributed and Predictive Entity Embedding for Fine-Grained Entity Typing in Knowledge Bases

Hailong Jin and **Lei Hou** and **Juanzi Li**
DCST, Tsinghua University
Beijing 100084, China
{jinhl15@mails.,houlei@mail.,lijuanzi@}tsinghua.edu.cn

Tiansi Dong
B-IT, Bonn University
53113 Bonn, Germany
dongt@bit.uni-bonn.de

Abstract

Fine-grained entity typing aims at identifying the semantic type of an entity in $\mathcal{KB}$. Type information is very important in knowledge bases, but is unfortunately incomplete even in some large knowledge bases. Limitations of existing methods are either ignoring the structure and type information in $\mathcal{KB}$ or requiring large scale annotated corpus. To address these issues, we propose an attributed and predictive entity embedding method, which can fully utilize various kinds of information comprehensively. Extensive experiments on real DBpedia dataset show that our proposed method significantly outperforms 8 state-of-the-art methods, with 3.4% and 2.9% improvement in Mi-F1 and Ma-F1, respectively.

1 Introduction

Knowledge about entities in Knowledge Base ($\mathcal{KB}$) is essential for language understanding. This knowledge can be attributional (e.g., *canFly*, *hasAge*), relational (e.g., *marriedTo*, *bornIn*) or type-based (e.g., *isFood*, *isLocation*). Type information, which clusters a group of entities with same properties, is valuable in $\mathcal{KB}$, because it is the glue that holds our mental world together (Murphy, 2004). Types in $\mathcal{KB}$ are usually organized as a hierarchical structure, namely *type hierarchy*. **Entity typing** assigns types (e.g., *Person*, *Athlete*, *BasketballPlayer*) to an entity (e.g., *Yao Ming*) in $\mathcal{KB}$, and is a fundamental task in knowledge base construction.

Traditional entity typing focuses on a small set of types, such as *Person*, *Location* and *Organization* (Ratinov and Roth, 2009; Nadeau and Sekine, 2007), while **fine-grained entity typing** assigns more specific types to an entity, which normally forms a *type-path* in the type hierarchy in $\mathcal{KB}$ (Ren et al., 2016a). As shown in Fig. 1, *Yao Ming* is associated with a type-path */Thing/Agent/Person/Athlete/BasketballPlayer*. Fine-grained types (e.g., *Athlete* and *BasketballPlayer*) are more informative than coarse-grained types (e.g., *Person*), because they provide more specific semantic information about the entity (Xu et al., 2016). Characterizing an entity with fine-grained types (type-paths) benefits many real applications, such as *knowledge base completion* (Dong et al., 2014), *entity linking* (Ling et al., 2015; Durrett and Klein, 2014), *relation extraction* (Liu et al., 2014), and *question answering* (Yahya et al., 2014).

Many researches have been carried out on *fine-grained entity typing **in text***, which detects local types for an entity mention in a particular plain text from predefined fine-grained entity types (Ling and Weld, 2012; Yogatama et al., 2015; Ren et al., 2016a; Shimaoka et al., 2017; Xin et al., 2018). However, only a few works aim at *fine-grained entity typing **in knowledge base***, that is, inferring missing types for an entity in $\mathcal{KB}$, which is the focus of this paper. Most $\mathcal{KB}$s are incomplete and lack of type information, in part because the world is always changing. For example, 36.53% entities in DBpedia do not have type information. Therefore, $\mathcal{KB}$s are aways in need of updating and completing. To this end, there are two main methods in the literature as follow: One method exploits various kinds of information to construct the feature representation of an entity, such as entity textual description, property and category (Neelakantan and Chang, 2015; Xu et al., 2016). After that, a predict function $P(t|e)$ is learned to infer

Corresponding author: Lei Hou (houlei@mail.tsinghua.edu.cn)

Proceedings of the 27th International Conference on Computational Linguistics, pages 282–292
Santa Fe, New Mexico, USA, August 20-26, 2018.

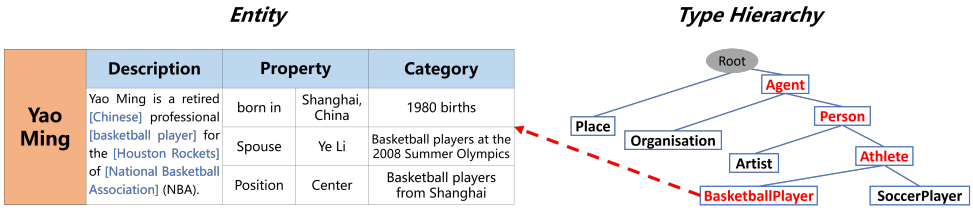


Figure 1: An example of fine-grained entity typing in $\mathcal{KB}$. The left part is an entity with various information. The red ones are assigned types, which forms a type-path.

whether entity e is an instance of type t. This method ignores rich structural information in $\mathcal{KB}$ (i.e., link relation between entities). It is a promising and challenging problem to utilize structural information for type inference. The second method utilizes **annotated corpus** to learn entity low-dimensional representations then makes type inference based on the learned embeddings (Yaghoobzadeh and Schütze, 2015; Yaghoobzadeh and Schütze, 2017). Apparently, it requires large scale corpus in which entity mentions are annotated. But large scale highly-qualified annotated corpus is often difficult to obtain.

To address the above issues, we propose a novel *attributed and predictive entity embedding* method, which can be described as follow: We first construct a partially-labeled attributed entity network, which contains structural, attribute and type information for entities. Here "partially-labeled" means some entities have been labeled with type information, while others not. Then, we employ a deep neural network to integrate the structural and attribute information for entity representation learning. For an unlabeled entity, we jointly utilize its structural and attribute information for neighbor prediction. For a labeled entity, besides neighbors, we also predict its types and type hierarchy via a margin-based loss function. Finally, a multi-label classification model is used for type inference based on the learned entity embeddings.

The main contributions of this paper can be summarized as follows:

- We encode various kinds of information into a partially-labeled entity network and propose a deep neural network model to effectively integrate the structural and attribute information for entity representation learning.
- We utilize type information of labeled entities to equip the learned embeddings with strong predictive power in entity typing task, and further incorporate the structure of type hierarchy via a margin-based loss function to meet the fine-grained demand.
- We construct a dataset from DBpedia. Extensive experiments show that our proposed method significantly outperforms 8 state-of-the-art methods, with 3.4% and 2.9% improvement in Mi-F1 and Ma-F1, respectively.

The rest of this paper is organized as follows. In Section 2, we formally define the problem of fine-grained entity typing in knowledge bases. Section 3 demonstrates the proposed approach in detail. The dataset description and evaluation are presented in Section 4. Section 5 outlines some related works. Section 6 concludes our work.

2 Problem Formulation

In this section, we formally define the problem of *Fine-Grained Entity Typing* in $\mathcal{KB}$.

Definition 1 ***Knowledge Base*** *is defined a tuple of entities $\mathcal{E}$ and a type hierarchy $\mathcal{H}$, formally, $\mathcal{KB} = \langle \mathcal{E}, \mathcal{H} \rangle$, where $\mathcal{H}$ is used to organize entities in $\mathcal{E}$. Each entity $e \in \mathcal{E}$ denotes a specific thing in the real world.*

Definition 2 ***Type Hierarchy*** *$\mathcal{H}$ is defined as a tree or a directed acyclic graph (DAG). It provides a natural way to categorize and organize entities in large $\mathcal{KB}$. It is formalized $(\mathcal{T}, \mathcal{R})$, where $\mathcal{T}$ is the type set and $\mathcal{R} = \{(t_i, t_j) | t_i, t_j \in \mathcal{T} \wedge i \neq j\}$ is the relation set, in which (t_i, t_j) means that t_i is the subtype of t_j.*

We distinguish three kinds of entity knowledge which are beneficial to *fine-grained entity typing* task.

- **Entity Link Relation:** $\mathcal{KB}$ contains a large number of link relations between entities. It is natural to convert entities in $\mathcal{E}$ into a network structure via these link relations. If an entity e_i links to another entity e_j in $\mathcal{KB}$, there will be a directed edge from entity e_i to entity e_j. These link relations reflect the overall structure of $\mathcal{KB}$ and provide important semantic relatedness between entities.

- **Entity Attribute:** Besides the above link structure, each entity $e \in \mathcal{E}$ also has textual description, properties and categories (shown in Fig. 1). We term all such auxiliary information of an entity as attributes. The attributes of an entity provide valuable clues to predict its types.

- **Entity Type:** Part of entities in $\mathcal{E}$ have been assigned with types, we refer them as $\mathcal{E}^l$. The assigned types for an entity form a type-path in type hierarchy $\mathcal{H}$, which starts from the root and ends at one specific type (not necessary to end at the leaf type). Those entities in $\mathcal{E}$ without type information are referred as $\mathcal{E}^u$. The type information observed in $\mathcal{E}^l$ can be a useful signal to infer missing type information in $\mathcal{E}^u$.

From the above definitions, we achieve a **partially-labeled attributed entity network**, including link structure, entity attributes, and type information. We formally define this entity network as follow:

Definition 3 ***Entity Network:*** *Let $\mathcal{G} = (\mathcal{E}, \mathcal{L})$ be an entity network. $\mathcal{E} = \mathcal{E}^l \cup \mathcal{E}^u$ is the entity set, where $|\mathcal{E}| = N$ and $|\mathcal{E}^l| = N^l$. $\mathcal{L} = \{(e_i, e_j) \,|e_i, e_j \in \mathcal{E} \text{ and } i \neq j\}$ is the link relation set.*

For the convenience of formula deduction, we redefine this entity network in matrix form, $\mathbf{G} = (\mathbf{A}, \mathbf{X}, \mathbf{Y})$. Adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ is equivalent to link set $\mathcal{L}$, $\mathbf{A}_{ij} = 1$ if there is a link from e_i to e_j, otherwise 0. Each row in $\mathbf{A}$ refers to an entity's link vector $\mathbf{A}_i$. Attribute matrix $\mathbf{X} \in \mathbb{R}^{N \times M}$ collects attributes for all entities, where *M* is the number of attributes. Each row in $\mathbf{X}$ refers to an entity's attribute vector $\mathbf{X}_i$. $\mathbf{X}$ can either be binary or take any real value. Type matrix $\mathbf{Y} \in \mathbb{R}^{N^l \times K}$ collects type information for entities in $\mathcal{E}^l$, where *K* is the number of types. Each row in $\mathbf{Y}$ refers to the type vector $\mathbf{Y}_i$ of a labeled entity in $\mathcal{E}^l$. $\mathbf{Y}_{ij} = 1$ if e_i has been assigned with type t_j, otherwise 0.

Based on the terminologies described above, we formally define the problem of *fine-grained entity typing in $\mathcal{KB}$* as follows:

Definition 4 ***Fine-grained entity typing in KB:*** *Given a partially-labeled attributed entity network $\mathcal{G} = (\mathcal{E}, \mathcal{L})$, we aim at learning a type predictor from $\mathcal{E}^l$, which comprehensively takes link structure, entity attributes and type information into account. Then we utilize the learned type predictor to predict the type of $e \in \mathcal{E}^u$, i.e., infer whether entity $e \in \mathcal{E}^u$ is an instance of type $t \in \mathcal{T}$.*

3 The Proposed Approach

3.1 General Architecture

A general state-of-the-art architecture for fine-grained entity typing consists of two components.

- **Entity Representation:** It learns distributed representation $\vec{v}(e)$ for each entity $e \in \mathcal{E}$.

- **Type Predictor:** It learns a predict function $P(t|e)$ using $\mathcal{E}^l$ as training examples. $P(t|e)$ denotes the probability that entity *e* belongs to type *t*, and it is often used to infer types of those unlabeled entities in $\mathcal{E}^u$.

Type predictor is regarded as a multi-label classification model. $P(t|e)$ is calculated through a two-layer Multi-Layer Perceptron (MLP).

$$[P(t_1|e) \cdots P(t_K|e)] = \sigma\left(\mathbf{W}_{out} f\left(\mathbf{W}_{in} \vec{v}(e)\right)\right) \tag{1}$$

Each entry of output layer indicates the predicted probability of type t_i for a given entity *e*. $\mathbf{W}_{in}$ and $\mathbf{W}_{out}$ are MLP parameter matrices. σ is the sigmoid function. f is the nonlinear activation function, such as *ReLU* and *tanh*.

The loss L for a prediction $\mathbf{y} = [P(t_1|e) \cdots P(t_K|e)]$ when the true labels are encoded in a binary vector $\mathbf{y}^* \in \{0,1\}^{1\times K}$ is the following cross entropy loss function:

$$L(\mathbf{y}, \mathbf{y}^*) = \sum_{i=1}^{K} -\mathbf{y}^{*(i)} \log \mathbf{y}^{(i)} - \left(1 - \mathbf{y}^{*(i)}\right) \log\left(1 - \mathbf{y}^{(i)}\right) \tag{2}$$

The key difficulty in computing $P(t|e)$ is to learn a good representation $\vec{v}(e)$ for entity e. Several models have achieved the state-of-the-art performance. However, these models still face the following non-trivial challenges:

- Entity attributes and link structure are typically regraded as separated features without considering their correlations. Recent research shows the importance of integrating link structure and entity attributes for learning more informative representations (Liao et al., 2017; Li et al., 2017). Could we seamlessly integrate entity attributes and link structure into a unified entity network? Since the link structure and entity attributes offer different sources of information, how shall we capture the complex non-linear relationship between them?

- The type information of the labeled entities in $\mathcal{E}^l$ is only used in the learning phase of the type predictor, but totally ignored in the representation learning phase. Recent research shows incorporating labeled information in representation learning could enhance the predictive power of the learning embeddings for specific tasks (Tang et al., 2015a; Yang et al., 2016). In our task, the entity types and their formed type hierarchy $\mathcal{H}$ are valuable labeled information, could we design a predictive model to make full use of the type information in $\mathcal{E}^l$ and the structure of type hierarchy?

To address the above challenges, we propose a novel attributed and predictive entity embedding method to learn entity representations. It combines the advantages of attributed network embedding method and semi-supervised learning. We integrate entity attributes and link structure via a deep neural network model (Section 3.2). For an entity e_i in $\mathcal{E}^l$, we jointly predict its type and its neighbor entities in the entity network (Section 3.3). In this way, we can fully utilize the type information in $\mathcal{E}^l$.

3.2 Attributed Entity Embedding

We aim at learning low-dimensional vector representations for each entity in the entity network, such that original link structure and entity attribute proximity can be preserved in vectors. The most intuitive idea is to directly concatenate the embeddings learned from different parts. However, this way can not capture the complex non-linear relationship between them, because each part is trained separately without interaction. To address this issue, we present a neural network model to integrate link structure (structural information) and entity attributes (attribute information). For an entity $e_i \in \mathcal{E}$, we jointly utilize its link vector $\mathbf{A}_i$ and attribute vector $\mathbf{X}_i$ to predict the link probability between e_i and other entities (shown in Fig. 2). Our model integrates structure and attribute information on the input layer, so that these two parts closely interact with each other and all parameters can be optimized simultaneously.

In our design, each entity *e* has two vector representations, *self representation* and *context representation*. $\mathbf{e}$ is self representation of *e* when it is treated as an entity in entity network $\mathcal{G}$. While $\tilde{\mathbf{e}}$ is context representation of *e* when it is treated as a specific context. To comprehensively represent an entity *e*, we add $\mathbf{e}$ and $\tilde{\mathbf{e}}$ as the final representation $\vec{v}(e)$.

Input and Embeddings. For an entity $e_i \in \mathcal{E}$, $\mathbf{A}_i$ and $\mathbf{X}_i$ denote its link vector and attribute vector as defined in Section 2. Our model takes both structural information ($\mathbf{A}_i$) and attribute information ($\mathbf{X}_i$) as input, and utilizes two embedding weight matrices to achieve the final input embeddings. $\mathbf{W}_{stru}$ projects the link vector $\mathbf{A}_i$ to a dense vector $\mathbf{u}_{stru}$ which captures structural information. $\mathbf{W}_{attr}$ encodes the attribute vector $\mathbf{X}_i$ and generates a compact vector $\mathbf{u}_{attr}$ which aggregates attribute information. We concatenate $\mathbf{u}_{stru}$ and $\mathbf{u}_{attr}$, then feed this concatenated vector into a Multi-Layer Perceptron.

Information Fusion. We employ a Multi-Layer Perceptron to integrate structural and attribute information. Recent research shows stacked multiple non-linear layers can help learn better representations

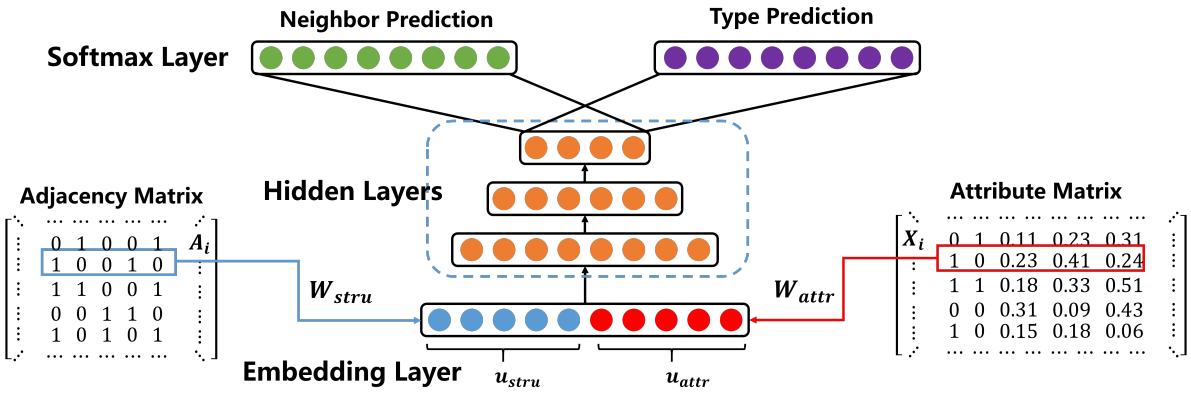


Figure 2: Attributed entity embedding framework

of data (Liao et al., 2017). Each layer is calculated as follows:

$$\mathbf{h}^{(1)} = f\left(\mathbf{W}^{(1)}\left[\mathbf{u}_{stru}, \lambda\mathbf{u}_{attr}\right] + \mathbf{b}^{(1)}\right) \tag{3}$$

$$\mathbf{h}^{(k)} = f\left(\mathbf{W}^{(k)}\mathbf{h}^{(k-1)} + \mathbf{b}^{(k)}\right), k = 1, 2, \cdots, n \tag{4}$$

λ is the trade-off factor for structure and attribute information. f denotes the activation function. n is the number of hidden layers. From the last hidden layer, we obtain an informative representation $\mathbf{h}^{(n)}$, which is regarded as the self representation $\mathbf{e}_i$ of the input entity e_i.

Neighbor Prediction. Finally, we predict neighbors for an input entity e_i (if there is a link from e_i to e_j, e_j is e_i's neighbor). We call it neighbor prediction, which is widely used to model network structure (Tang et al., 2015a; Liao et al., 2017). Specifically, we utilize the self representation $\mathbf{e}_i$ to predict the link probability between entity e_i and other entities. The predictive link probability is defined through a softmax layer as follows:

$$p\left(e_j|e_i\right) = \frac{\exp\left(\tilde{\mathbf{e}}_j \cdot \mathbf{e}_i\right)}{\sum_{j'=1}^{N}\exp\left(\tilde{\mathbf{e}}_{j'} \cdot \mathbf{e}_i\right)} \tag{5}$$

$\tilde{\mathbf{e}}_j$ denotes the context representation of entity e_j. Let $\mathbf{y}_i$ denotes the predicted link probability vector for entity e_i, $\mathbf{y}_i^{(j)}$ is the j-th entry of the vector $\mathbf{y}_i$. $\mathbf{A}_i$ is e_i's link vector (in Section 2), $\mathbf{A}_{ij}$ indicates whether there is a link from e_i to e_j. The loss function L_{attr} is defined as cross entropy over all entities in $\mathcal{E}$

$$L_{attr} = \sum_{i=1}^{N}\sum_{j=1}^{N} -\mathbf{A}_{ij}\log\mathbf{y}_i^{(j)} - \left(1 - \mathbf{A}_{ij}\right)\log\left(1 - \mathbf{y}_i^{(j)}\right) \tag{6}$$

3.3 Predictive Entity Embedding

In this section, we propose predictive entity embedding learning that makes full use of the type information in $\mathcal{E}^l$, which is often ignored by existing methods in representation learning phase. Inspired by (Tang et al., 2015a; Yang et al., 2016), we first introduce a type prediction loss to equip learned embeddings with strong predictive power for entity typing task (Section 3.3.1). Then, to meet the fine-grained demand, we model the tree-structured type hierarchy in a margin-based framework (Section 3.3.2).

3.3.1 Type Prediction

We fully utilize type information in $\mathcal{E}^l$ to make the learned embeddings more predictive. For each entity $e_i \in \mathcal{E}^l$, we not only predict its neighbors in $\mathcal{G}$ (Section 3.2), but also predict its types. We add another softmax layer, the probability of entity e_i being an instance of type t_j is defined as follows:

$$p\left(t_j|e_i\right) = \frac{\exp\left(\mathbf{t}_j \cdot \mathbf{e}_i\right)}{\sum_{j'=1}^{K}\exp\left(\mathbf{t}_{j'} \cdot \mathbf{e}_i\right)} \tag{7}$$

$\mathbf{t}_j$ denotes the representation of type t_j. Let $\mathbf{y}_i$ denotes the predicted type probability vector for entity e_i, $\mathbf{y}_i^{(j)}$ is the *j*-th entry of the vector $\mathbf{y}_i$. $\mathbf{Y}_i$ is e_i's type vector (in Section 2), $\mathbf{Y}_{ij}$ indicates whether e_i is an instance of t_j. The loss function L_{type} is defined as cross entropy over all entities in $\mathcal{E}^l$:

$$L_{type} = \sum_{i=1}^{N^l} \sum_{j=1}^{K} -\mathbf{Y}_{ij} \log \mathbf{y}_i^{(j)} - (1 - \mathbf{Y}_{ij}) \log \left(1 - \mathbf{y}_i^{(j)}\right) \tag{8}$$

3.3.2 Type Hierarchy Modeling

Fine-grained entity typing requires us to pay more attention to specific types (fine-grained types) than to general types (coarse-grained types), because specific types are more informative. For example, *Yao Ming* belongs to *BasketballPlayer* and *Athlete*, and it should get higher score on *BasketballPlayer* than *Athlete*, because the former characterizes *Yao Ming* more accurately. Besides, *Yao Ming* should get higher score on the correct type *BasketballPlayer* than its wrong sibling type *SoccerPlayer* although both types belong to *Athlete*. If different types are treated equally, the structural information between types will be lost. Therefore, we use *type order* to model different relatedness between an entity and its related types, which reflects the structural information of type hierarchy to a certain degree. In particular, we define two kinds of *type order* as follow:

Definition 5 ***Ancestor order:*** *In the type-path of an entity, specific types should get higher score than its ancestor. For example, Yao Ming should get higher score on BasketballPlayer than Athlete.*

Ancestor order reflects the relationship between a type and its ancestor types. For each triple (e, t, t^a), where *t* is one of the type for entity *e*, and t^a is *t*'s ancestor in type hierarchy. We define adaptive margin $\gamma_{tt^a} = \xi \times \frac{l(t^a,t)}{l(root,t)}$, where $l(t^a, t)$ denotes the number of steps going down from type t^a to type t. The hinge loss is defined as:

$$\max\left(0, s\left(e, t^a\right) - s\left(e, t\right) + \gamma_{tt^a}\right) \tag{9}$$

$$L_{ancestor} = \mathbb{E}_{(e,t,t^a)}\left[\max\left(0, s\left(e, t^a\right) - s\left(e, t\right) + \gamma_{tt^a}\right)\right] \tag{10}$$

where $s\left(e, t\right)$ is the score function, defined as the inner product of $\mathbf{e}$ and $\mathbf{t}$.

Definition 6 ***Sibling order:*** *For an entity, the correct type should get higher score than its wrong sibling type in type hierarchy. For example, Yao Ming should get higher score on BasketballPlayer than SoccerPlayer.*

Sibling order reflects the relationship between a type and its sibling types. For each triple (e, t, t^s), where *t* is one of the true type for entity *e*, and t^s is its wrong sibling type of entity *e*. We use the fixed margin ξ. The hinge loss is defined as:

$$\max\left(0, s\left(e, t^s\right) - s\left(e, t\right) + \xi\right) \tag{11}$$

$$L_{sibling} = \mathbb{E}_{(e,t,t^s)}\left[\max\left(0, s\left(e, t^s\right) - s\left(e, t\right) + \xi\right)\right] \tag{12}$$

We achieve the loss function L_{hier} for type hierarchy modeling: $L_{hier} = L_{ancestor} + L_{slibing}$.

Objective and Training. Finally, we sum these three loss function to get the final objective function of entity representation component.

$$Loss = L_{attr} + \lambda_1 \cdot L_{type} + \lambda_2 \cdot L_{hier} \tag{13}$$

λ_1 and λ_2 are weight parameters. We apply negative sampling (Mikolov et al., 2013; Tang et al., 2015b), where only a very small subset of entities are sampled from $\mathcal{E}$. Parameters are optimized through Adaptive Moment Estimation (Adam). In the embedding layer and each hidden layer, we also add dropout component to alleviate over fitting.

4 Experiments and Analysis

In this section, we evaluate the proposed method using real world dataset collected from DBpedia. We will introduce the dataset and experiment settings in Section 4.1, present the comparison results in Section 4.2 and investigate some method details in Section 4.3. Our source code is available[1] for reference.

4.1 Datasets and Experimental Setup

Datasets: To the best of our knowledge, there is no public large-scale dataset available for fine-grained entity typing in $\mathcal{KB}$. We construct a dataset from DBpedia. The type hierarchy in DBpedia is of tree structure. We extract link relation and attribute information from DBpedia[2]. For each entity, we use the single type-path assigned in DBpedia as the ground truth.

For this dataset, we remove those entities which are only labeled as *Thing*. Because they do not provide any semantic information. In representation learning phase, we utilize the whole dataset with only 50% entities are labeled with types. In type predictor phase, we follow a 50/30/20 ratio to split the dataset into training, dev and test sets. Table 1 shows the detailed statistics. Each entity has 3.27 types on average.

Table 1: Statistics of the dataset

Type	Entity	Link	Attribute		
			word	property	category
214	300,000	5,243,230	5,350	200	350

Metrics: To evaluate the performance of our proposed method, we use Accuracy (**Strict-F1**), Micro-averaged F1 (**Mi-F1**) and Macro-averaged F1 (**Ma-F1**), which have been used in many fine-grained typing systems (Yaghoobzadeh and Schütze, 2015; Yaghoobzadeh and Schütze, 2017; Ren et al., 2016a).
Parameter Settings: Our implementation of **APE** is based on TensorFlow[3]. Regarding the choice of activation function of hidden layers, we have tried *ReLU*, *softsign* and *tanh*, finding *softsign* leads to the best performance in general. We randomly initialize model parameters with a Gaussian distribution (with a mean of 0.0 and standard deviation of 0.01), optimizing the model with mini-batch Adam (Kingma and Ba, 2014). We set fixed margin $\xi = 1$ in our experiments. Both λ_1 and λ_2 are chosen among $\{0.2, 0.4, 0.6, 0.8\}$, we found $\lambda_1 = 0.8$ and $\lambda_2 = 0.4$ achieve best performance.
Baseline: We denote our model as **APE**, and compare it with three sets of methods:

- **Entity typing methods**: We compare **APE** with five state-of-the-art entity typing methods. **FIGMENT** uses entity representation learned from annotated corpus for type inference (Yaghoobzadeh and Schütze, 2015). **CUTE** employs a multi-label hierarchical classification method to address entity typing task (Xu et al., 2016). **MuLR** uses multi-level representations of entities (character, word and entity) to make type inference in $\mathcal{KB}$ (Yaghoobzadeh and Schütze, 2017). **Global** utilizes entity text description to infer missing entity type instances (Neelakantan and Chang, 2015). **Corpus** designs global and context model to make type inference jointly (Yaghoobzadeh et al., 2017).

- **Network Embedding methods**: Attributed and predictive network embedding methods can learn informative entity representations, so we compare **APE** with them. **PTE** learns predictive text embeddings from heterogeneous network (Tang et al., 2015a). **Planetoid** is a semi-supervised network embedding method which makes neighbor and label prediction jointly (Yang et al., 2016). **ASNE** integrates structural and attribute information via a deep neural architecture (Liao et al., 2017).

- **APE's variants**: If we ignore the type information, that is, utilize the embeddings learned from section 3.2 to make type inference, the method is denoted as $\textbf{APE}_{no\_type}$. Similarly, if we omit the type hierarchy modeling in Section 3.3.2, the method is denoted as $\textbf{APE}_{no\_hierarchy}$.

4.2 Overall Comparison Results

Table 2 shows the overall performance and we have the following conclusions:

[1] https://github.com/Tsinghua-PhD/APE
[2] http://wiki.dbpedia.org/downloads-2016-10
[3] https://www.tensorflow.org/

Table 2: Typing performance on DBpedia dataset

Type	Algorithm	Acc	Ma-F1	Mi-F1
Entity Typing	**FIGMENT**	0.463	0.619	0.628
	CUTE	0.515	0.673	0.677
	MuLR	0.501	0.654	0.662
	Corpus	0.488	0.662	0.659
	Global	0.457	0.608	0.615
Network Embedding	**Planetoid**	0.434	0.585	0.590
	ASNE	0.417	0.568	0.573
	PTE	0.352	0.512	0.518
Our Variants	$\mathbf{APE}_{no\_type}$	0.492	0.649	0.657
	$\mathbf{APE}_{no\_hierarchy}$	0.531	0.694	0.689
	APE	**0.545**	**0.702**	**0.711**

Comparison with Entity Typing methods. Our model outperforms the state-of-the-art entity typing methods, and achieves 3.4% and 2.9% improvement in Mi-F1 and Ma-F1 respectively, because it employs multiple kinds of information for type inference. Each part can provide complementary information about entities for type inference. Our model leverages label information (type information) in representation learning phase, equip the learned embeddings with strong predictive power in entity typing task.

Comparison with Network Embedding methods. Our model outperforms the state-of-the-art network embedding methods (i.e., **Planetoid**, **ASNE** and **PTE**), and achieves 12.1% and 11.7% improvement in Mi-F1 and Ma-F1 respectively. We model the structure of type hierarchy in a margin-based loss function. Specifically, ancestor order makes entities closer to specific types than general types, and sibling order makes the learned entity embeddings have strong discriminative power between sibling types.

Comparison with Variants. **APE** consistently outperforms $\mathbf{APE}_{no\_type}$ and $\mathbf{APE}_{no\_hierarchy}$, and the simplest version $\mathbf{APE}_{no\_type}$ could achieve comparable results with state-of-the-art methods. The results verify that each of these information sources contributes complementary information for *fine-grained typing of entities*. Structural information (link relation) provides important semantic relatedness between entities. Attribute information (text, property and category) provides additional clues for type inference. Type information (type hierarchy) makes our model more powerful in classification task.

4.3 Result Analysis

4.3.1 Impact of Features

We investigate the effect of information sources used to learn entity embeddings. In Table 3, **Stru**, **T**, **P** and **C** is short for structural information, text description, property and category, respectively. From Table 3, we find different information sources play different roles in this task. *Structural information* is most common in $\mathcal{KB}$, it makes related entities have similar representations. It models the relation between entities, while neglecting entities's self-information. *Attribute information* (i.e., text description, property and category) can be viewed as entity features, which is often informative for type inference. We compare our model **APE**, $\mathbf{APE}_{no\_type}$ and $\mathbf{APE}_{no\_hierarchy}$ with different kinds of information combination. For attribute information, category information is more import than property information and text descriptions.

4.3.2 Effects of Labeled Data

We investigate the effect of labeled data proportion. Intuitively, our model can learn better embeddings and achieve better performance if we have more labeled entities. The results in Fig. 3 show that three metrics strikingly increase with more labeled data added, and gradually become stable when the proportion is over 0.5. It proves that our model can achieve a satisfactory result with not too much labeled data, and this advantage benefits from the entity link relations and network embedding.

Table 3: Performance on different information source combination.

Information Type	APE			APE_{no_type}			$\text{APE}_{no\_hierarchy}$		
	Acc	Ma-F1	Mi-F1	Acc	Ma-F1	Mi-F1	Acc	Ma-F1	Mi-F1
Stru+T	0.467	0.623	0.631	0.463	0.625	0.634	0.472	0.632	0.633
Stru+P	0.475	0.633	0.637	0.480	0.638	0.642	0.489	0.651	0.653
Stru+C	0.481	0.641	0.646	0.487	0.649	0.647	0.486	0.650	0.647
Stru+P+C	0.479	0.642	0.649	0.520	0.685	0.679	0.523	0.677	0.669
Stru+T+P	0.484	0.647	0.655	0.507	0.658	0.659	0.510	0.657	0.660
Stru+T+C	0.481	0.643	0.653	0.518	0.683	0.681	0.525	0.678	0.673
Stru+T+P+C	**0.492**	**0.649**	**0.657**	**0.531**	**0.694**	**0.689**	**0.545**	**0.702**	**0.711**

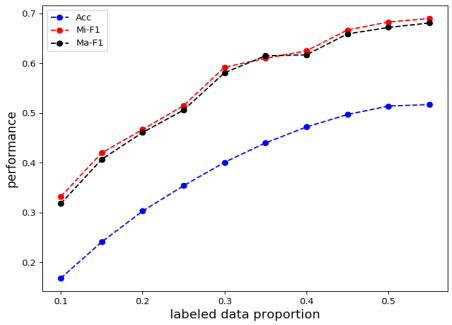


Figure 3: Performance on different labeled data proportion.

Table 4: Type macro average F1 on test for all, frequent and infrequent types, *#training/test* means the average entity number for each type.

Type	infrequent	frequent	all
# training	1013	13,513	8,673
# test	417	5,217	3,462
FIGMENT	0.327	0.620	0.557
MuLR	0.374	0.718	0.636
Corpus	0.447	0.783	0.675
CUTE	0.418	0.730	0.652
APE	**0.461**	**0.792**	**0.681**

4.3.3 Results of Frequent/Infrequent Types

We use type macro average F1 proposed by (Yosef et al., 2012) to evaluate the performance on *frequent types* and *infrequent types*. Note that it is different from Ma-F1 reported in Table 2. The results for infrequent types (occur less than 2,000 times) and frequent types (occur more than 10,000 times) are shown in Table 4. Generally, the performance on infrequent types is worse than frequent ones. Our model consistently outperforms the other methods on infrequent types, which demonstrates its ability on dealing with rare types.

5 Related Work

Fine-grained entity typing in $\mathcal{KB}$ is an important sub-task of knowledge base completion. One way to address this problem is making type inference based on entity's attribute information in $\mathcal{KB}$, such as textual description, property, category and so on. (Neelakantan and Chang, 2015) infer missing types for entities in $\mathcal{KB}$ based on text descriptions. (Xu et al., 2016) employ a multi-label hierarchical classification method to assign Chinese entities with DBpedia types based on property and category information. The other way relies on large-scale annotated corpus, then extracts type information from the annotated corpus for entities in $\mathcal{KB}$. (Yaghoobzadeh and Schütze, 2015) first propose FIGMENT to address this problem. They only used contextual information to assign types for entities in $\mathcal{KB}$. After that, they present FIGMENT-Multi to learn multi-level representations of entities on three complementary levels (character, word and entity). FIGMENT-Multi predicts whether an entity is a member of a type based on the learned embeddings (Yaghoobzadeh and Schütze, 2017). Finally, they propose an embedding based method which combines a global model with a context model. A global model that scores based on aggregated context information and a context model that aggregates the scores of individual contexts (Yaghoobzadeh et al., 2017).

Fine-grained entity typing in text is related to our task. Different from the works mentioned above, it

seeks to detect local types of an entity mention inside an individual sentence, and the same entity could have different types in different sentences (Ling and Weld, 2012; Yosef et al., 2012; Gillick et al., 2014; Dong et al., 2015; Del Corro et al., 2015; Yogatama et al., 2015; Ren et al., 2016b; Ren et al., 2016a; Shimaoka et al., 2017; Rabinovich and Dan, 2017).

6 Conclusion

In this paper, we propose an attributed and predictive entity embedding method to address the task of fine-grained entity typing in $\mathcal{KB}$. It combines the advantages of attributed network embedding method and semi-supervised learning. Specifically, it employs a deep neural network architecture to integrate structure and attribute information of an entity. We jointly predict a labeled entity's neighbors and types, and model the structure of type hierarchy in a margin-based way. Experiments on real world datasets demonstrate the effectiveness of the proposed model.

Acknowledgements

The work is supported by National Key Research and Development Program of China (2017YFB1002101), NSFC key project (U1736204), Ministry of Education and China Mobile Research Fund (No. 20181770250), and THUNUS NExT Co-Lab.

References

Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. In *Conference on Empirical Methods in Natural Language Processing*, pages 868–878.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM.

Li Dong, Furu Wei, Hong Sun, Ming Zhou, and Ke Xu. 2015. A hybrid neural model for type classification of entity mentions. In *International Conference on Artificial Intelligence*, pages 1243–1249.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis : Coreference , typing , and linking. *TACL*, pages 477–490.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *arXiv:1412.1820*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Hang Li, Haozheng Wang, Zhenglu Yang, and Masato Odagaki. 2017. Variation autoencoder based network representation learning for classification. In *ACL 2017, Student Research Workshop*, pages 56–61.

Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat Seng Chua. 2017. Attributed social network embedding. *arXiv:1705.04969*.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *AAAI*, pages 94–100.

Xiao Ling, Sameer Singh, and Daniel S Weld. 2015. Design challenges for entity linking. *TACL*, pages 315–328.

Yang Liu, Kang Liu, Liheng Xu, and Jun Zhao. 2014. Exploring fine-grained entity type constraints for distantly supervised relation extraction. In *COLING*, pages 2107–2116.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *International Conference on Neural Information Processing Systems*, pages 3111–3119.

Gregory Murphy. 2004. *The big book of concepts*. MIT press.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Arvind Neelakantan and Ming Wei Chang. 2015. Inferring missing entity type instances for knowledge base completion: New dataset and methods. *Computer Science*, pages 35–40.

Maxim Rabinovich and Klein Dan. 2017. Fine-grained entity typing with high-multiplicity assignments. In *Meeting of the Association for Computational Linguistics*, pages 330–334.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016a. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1369–1378.

Xiang Ren, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, and Jiawei Han. 2016b. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1825–1834.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural architectures for fine-grained entity type classification. In *EACL*, pages 1271–1280.

Jian Tang, Meng Qu, and Qiaozhu Mei. 2015a. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1165–1174. ACM.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015b. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. ACM.

Ji Xin, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2018. Improving neural fine-grained entity typing with knowledge attention. In *AAAI*.

Bo Xu, Yi Zhang, Jiaqing Liang, Yanghua Xiao, Seung-won Hwang, and Wei Wang. 2016. Cross-lingual type inference. In *International Conference on Database Systems for Advanced Applications*, pages 447–462.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 715–725.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2017. Multi-level representations for fine-grained typing of knowledge base entities. *arXiv:1701.02025*.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. 2017. Corpus-level fine-grained entity typing. *arXiv:1708.02275*.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. 2014. Robust question answering over the web of linked data. In *CIKM*, pages 1107–1116.

Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML*, pages 40–48.

Dani Yogatama, Dan Gillick, and Nevena Lazic. 2015. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL*, pages 26–31.

M Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. Hyena: Hierarchical type classification for entity names. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 1361–1370.

Joint Learning from Labeled and Unlabeled Data for Information Retrieval

Bo Li, Ping Cheng, Le Jia
School of Computer Science
Central China Normal University
Wuhan, China
libo@mail.ccnu.edu.cn

Abstract

Recently, a significant number of studies have focused on neural information retrieval (IR) models. One category of works use unlabeled data to train general word embeddings based on term proximity. The general embeddings can be integrated into traditional IR models. The other category employs labeled data (e.g. click-through data) to train *end-to-end* neural IR models consisting of layers for target-specific representation learning. The latter idea accounts better for the IR task and is favored by recent research works, which is the one we will follow in this paper. We hypothesize that general semantics learned from unlabeled data can complement task-specific representation learned from labeled data of limited quality, and that a combination of the two is favorable. To this end, we propose a learning framework which can benefit from both labeled and more abundant unlabeled data for representation learning in the context of IR. Through a joint learning fashion in a single neural framework, the learned representation is optimized to minimize both the supervised loss on query-document matching and the unsupervised loss on text reconstruction. Standard retrieval experiments on TREC collections indicate that the joint learning methodology leads to significant better performance of retrieval over several strong baselines for IR.

1 Introduction

In recent years, the research community has noticed the great success of neural networks in computer vision (Krizhevsky et al., 2012), speech recognition (Hinton et al., 2012) and natural language processing (Mikolov et al., 2013) tasks. However, the potential of neural networks has not been fully investigated in the IR field. Although a significant number of studies (e.g. (Huang et al., 2013; Ganguly et al., 2015; Zheng and Callan, 2015; Guo et al., 2016; Zamani and Croft, 2016; Dehghani et al., 2017; Mitra et al., 2017)) try to apply neural networks in IR, there have been few studies reporting the performance that is comparable to state-of-the-art IR models. These approaches rely on the general idea that neural network can provide a low-dimensional and semantics-rich representation for both queries and documents. Such a representation can bridge lexical and semantic gaps in traditional IR models. Depending on if the embeddings are trained with discriminative information for IR tasks, existing works can be broadly divided into two categories (Zhang et al., 2016; Mitra and Craswell, 2017).

The first category of approaches extend traditional IR models to incorporate word embeddings that are trained on huge and unlabeled corpora with existing models such as *Word2vec* (Mikolov et al., 2013) and *GloVe* (Pennington et al., 2014) in an unsupervised manner. These approaches (e.g. (Zheng and Callan, 2015; Nalisnick et al., 2016)) leverage semantic information captured by word embeddings in order to enhance traditional IR models. We note that such models trained without references to the retrieval task model term proximity and do not contain discriminative information adapted for IR (Zamani and Croft, 2017). The second category (e.g. (Huang et al., 2013; Guo et al., 2016)) tries to incorporate word embedding learning within neural models for IR, which reflects a more significant shift toward an *end-to-end* framework. These approaches treat word embeddings as layers in neural IR models, to be learned

Proceedings of the 27th International Conference on Computational Linguistics, pages 293–302
Santa Fe, New Mexico, USA, August 20-26, 2018.

along with all model parameters in a supervised manner. Most studies in the second category rely on click-through data for relevance judgment between queries and documents. Text representation learned with relevance information captures relevance rather than term proximity, which clearly accounts better for IR requirements (Zamani and Croft, 2017). However, supervised signals such as click-through data are often limited outside of large industrial research labs, probably due to user privacy concerns. It is thus not surprising to see that many authors following this methodology have industrial background (e.g. (Huang et al., 2013; Shen et al., 2014b; Nalisnick et al., 2016; Mitra et al., 2017)). In addition, Ye et al. (2015) point out that previous studies using click-through data make implicit but strong assumptions about clicked query-document pairs which are not necessarily met in practice.

Neural networks are hungry for data, a fact which also holds for neural IR tasks. One can find from above discussions that the second category of approaches suffer from the data spareness problem, although there have been recent attempts (Gupta et al., 2017; Dehghani et al., 2017) trying to pseudo label query-document pairs automatically with unsupervised retrieval models such as BM25. Using pseudo labels as relevance signals relieves data spareness in terms of quantity but not quality. The idea of using unsupervised learning to complement supervision has been practiced successfully in computer vision (Yang et al., 2013) and natural language processing (Rasmus et al., 2015) tasks. In such a background, we hypothesize that semantics learned from unlabeled data can complement task-specific representation learned from pseudo-labeled data of limited quality, and a combination of the two is favorable in IR. To the best of our knowledge, such a combination has never been investigated in neural IR models.

In this paper, we propose a learning framework which can benefit from both labeled and more abundant unlabeled data for representation learning in IR. Through joint learning in a single neural network, the learned representation can account for task-specific characteristics via supervised loss optimization on query-document matching, as well as preserving general semantics via unsupervised loss optimization on text reconstruction. We demonstrate by experiments that the joint learning model leads to significantly better performance over state-of-the-art IR models.

2 Related work

Representation learning approaches based on neural networks have gained in prominence in recent years due to their extreme efficiency. They motivate the emerging research field of Neural IR. Neural approaches have attracted increasing interests of the IR community in very recent years. Apart from learning to rank approaches that train their models over a set of hand-crafted features (Liu, 2009), neural IR models typically accept the raw text of queries and documents as input. The dense representations of words or texts can then be learned with or without reference to retrieval tasks, respectively corresponding to the two categories of methods summarized in section 1.

Unsupervised approaches learn general text representation without query and document interaction information. Embeddings pre-trained on unlabeled text with tools such as *Word2vec* (Mikolov et al., 2013) and *Glove* (Pennington et al., 2014) have been used to extend traditional IR models. Ganguly et al. (2015) develop a generalized language model with query-likelihood language modeling for integrating word embeddings as additional smoothing. Zheng and Callan (2015) represent term and query as vectors in the same latent space based on word embeddings so as to learn a model to reweight terms. Nalisnick et al. (2016) retain both input and output embeddings of *Word2vec* and map query words into the input space and document words into the output space. Zamani and Croft (2016) propose to use word embeddings to incorporate and weight terms not present in the query, acting as smoothing and query expansion. There are also studies developing their own embedding learning algorithms instead of using standard tools for embedding learning. For instance, Salakhutdinov and Hinton (2009) propose a deep auto-encoder model to generate a condensed binary vector representation of documents. Clinchant and Perronnin (2013) use latent semantic indexing to induce word embeddings for IR. Vulić and Moens (2015) propose to learn from document-aligned comparable corpora the embeddings that can be used for both monolingual IR and cross-lingual IR.

Supervised approaches use query-document relevance information to learn the representation that is optimized *end-to-end* for the task at hand. With click-through data, Huang et al. (2013) develop DSSM, a

feed forward neural network with a word hashing phrase as the first layer to predict the click probability given a query string and a document title. DSSM is extended in (Shen et al., 2014a; Shen et al., 2014b) by incorporating convolutional neural network and max-pooling layers to extract the most salient local features. Since the DSSM related methods make implicit but strong assumptions about clicked data, Ye et al. (2015) try to relax the assumptions in their model. Guo et al. (2016) develop the DRMM model that takes the histogram-based features representing interactions between queries and documents as input into neural networks. DRMM is one of the first neural IR models to show improvement over traditional IR models. Mitra et al. (2017) aim to simultaneously learn local and distributional representation to capture both lexical matching and semantic matching in IR. Following the discussion in section 1, we note that click-through data are not always available in massive amount outside of industrial labs. More recent works propose to use unsupervised IR models to pseudo label query-document pairs that provide weak supervision for representation learning. Dehghani et al. (2017) use BM25 to obtain relevant documents for a large set of AOL queries (Pass et al., 2006) which are then used as weakly supervised signals for joint embedding and ranking model training. Zamani and Croft (2017) employ similar supervision signals as (Dehghani et al., 2017) to train an embedding network similar to *Word2vec* and use the obtained embeddings for query expansion and query classification. Gupta et al. (2017) develop a cross-lingual IR model based on weak supervision. Luo et al. (2017) propose to train deep ranking models with weak relevance labels generated by click model based on click behavior of real users.

We can conclude from above discussions that supervised approaches account better for task-specific features and are superior in IR. They rely on relevance information between query-document pairs of which the quality is relatively low in practice. In this paper, we follow successful practice in CV and NLP tasks and hypothesize that general and rich semantics learned from unlabeled data can complement task-specific representation learned from labeled data of limited quality. We will propose in section 3 a learning framework which can simultaneously learn from labeled and more abundant unlabeled data in the context of IR. By the way, we note that the joint learning framework resembles those studies (e.g. (Liu et al., 2015)) which couple IR with another supervised learning task. Our framework differs from those studies in that we do not require additional data that are labeled for another supervised learning task.

3 Joint learning framework for IR

In this section, we will develop a joint framework to learn low-dimensional representation of queries and documents from both labeled and unlabeled data.

3.1 Learning framework

The joint learning framework is illustrated in figure 1. It consists of three crucial components:

- **An encoding network.** It embeds the raw input into low-dimensional representations that are designed to capture target-specific characteristics of IR.
- **A decoding network.** It tries to reconstruct the input so as to benefit from unlabeled data.
- **A pairwise ranking model.** It makes use of supervision signals from labeled query-document pairs to perform document ranking.

On top of the network structure, we perform joint optimization of both supervised loss and unsupervised loss. The unsupervised learning process uses all the text collection (e.g. queries and documents) for learning rich and general semantics. The supervised learning process learns, from labeled query-document pairs, discriminative representations adapted for IR. The joint training fashion makes two learning processes complement each other via co-tuning the shared hidden layers in the encoding networks to help the representation generalize better in the IR task.

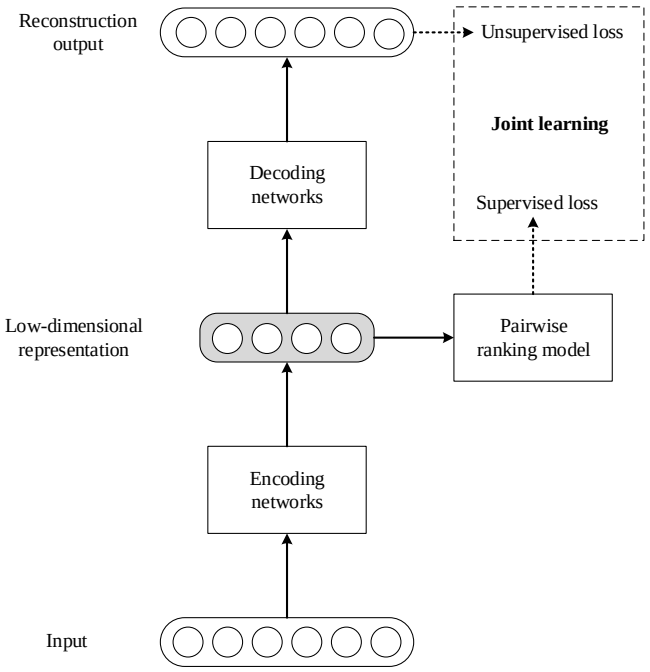


Figure 1: The joint learning framework with labeled and unlabeled data. It consists of an encoding network, a decoding network and a pairwise ranking model. We impose an unsupervised loss and a supervised loss respectively on the reconstruction output and the pairwise ranking output, which is learned in a joint fashion in this paper. The low-dimensional representation is the one our model aims to learn.

3.2 Unsupervised learning

The unsupervised part learns the low-dimensional representation of text via an autoencoder style, which uses all the available text data. Following previous studies on text autoencoder (Chen and Zaki, 2017), we opt for the simple feed-forward neural network architecture for both the encoding and decoding parts in figure 1. For each layer of the encoding/decoding networks, we use Rectified Linear Unit (ReLU) as the activation function, a function recommended by many works in deep learning (LeCun et al., 2015). In the feed-forward step, each layer $l(l \geq 1)$ is a fully-connected layer and its activation potential $\boldsymbol{z}_l$ is given by:

$$\boldsymbol{z}_l = \max(0, \boldsymbol{W}_l \boldsymbol{z}_{l-1} + \boldsymbol{b}_l)$$

where $\boldsymbol{W}_l$ is the weight matrix at layer l and $\boldsymbol{b}_l$ is the corresponding bias.

The input layer (corresponding to $l = 0$) maps the input text into fixed-length vector. There have been two methodologies we can employ to represent the input text: one is the one-hot representation (Gupta et al., 2017) and its variants (Zhai and Zhang, 2016); the other one is the dense and semantically rich representations (He et al., 2017). Empirical results do not indicate that one is always better than the other and we will make use of the former one in this paper. Given the set of text T, we follow previous studies such as (Zhai and Zhang, 2016; Chen and Zaki, 2017) and represent each input text t in T as log-normalized word count vector $\boldsymbol{x} \in R^{|V|}$ where $|V|$ is the size of the vocabulary V. Each dimension of the input vector $\boldsymbol{x}$ is represented by:

$$\boldsymbol{x}_i = \frac{\log[1 + tf(i)]}{\max_{i \in V} \log[1 + tf(i)]}, \text{ for } i \in V$$

where $tf(i)$ is the term frequency of the i-th word in the vocabulary. Since the unsupervised learning part of the framework is modeled as an autoencoder, we want the unsupervised output $\boldsymbol{x}'$ to resemble the

input $\boldsymbol{x}$, leading to the binary cross-entropy loss function l_u on t that can be defined as:

$$l_u(t) = -\sum_{i \in V} [\boldsymbol{x}_i \log(\boldsymbol{x}'_i) + (1 - \boldsymbol{x}_i) \log(1 - \boldsymbol{x}'_i)] \tag{1}$$

3.3 Supervised learning

The document ranking problem can not be modeled with the standard classification or regression framework. Following the methodology in learning to rank (Liu, 2009), we model document ranking in the *pairwise* style where the relevance information is in the form of preferences between pairs of documents with respect to individual queries. In addition, we follow previous studies (Gupta et al., 2017) and make use of well-performing unsupervised retrieval models (e.g. BM25) to pseudo-label query and document pairs so as to obtain the relevance information. More details will be given in section 4.1.

From figure 1 one can note that the hidden layers in the encoding networks are shared by unsupervised and supervised learning, and one can refer to the unsupervised learning part for details of the layers in the encoding networks. The supervised model, on top of the top-level representation layer (i.e. low-dimensional representation), tries to learn a model that, given the query q, assigns a larger score to document d_1 than document d_2 if the ground truth is that d_1 matches to q better. The supervised model is implemented as a pairwise ranking model in figure 1, which is again a feed forward neural networks. Inspired by such studies as (Yih et al., 2011), we can derive the probability $P'(d_1 \succ_q d_2)$ that d_1 is ranked higher than d_2 with respect to the query q via a logistic function:

$$P'(d_1 \succ_q d_2) = \frac{1}{1 + e^{-\sigma[\text{score}(q,d_1) - \text{score}(q,d_2)]}}$$

where the *score* function is computed with the pairwise ranking model, and the parameter σ is used to determine the shape of the sigmoid. The supervised training objective l_s on a triplet of query-document pair (q, d_1, d_2) can then be defined as the cross entropy loss, which is:

$$\begin{aligned} l_s(q, d_1, d_2) = &- P(d_1 \succ_q d_2) \log P'(d_1 \succ_q d_2) \\ &- [1 - P(d_1 \succ_q d_2)] \log[1 - P'(d_1 \succ_q d_2)] \end{aligned} \tag{2}$$

where $P(d_1 \succ_q d_2)$ is the actual probability that d_1 is ranked higher than d_2 according to annotations (i.e. pseudo-labels of query-document pairs). The actual probability in this paper is estimated in a similar way as in (Dehghani et al., 2017), which is:

$$P(d_1 \succ_q d_2) = \frac{1}{1 + e^{-\sigma[\text{S}(q,d_1) - \text{S}(q,d_2)]}}$$

where s denotes the relevance scores obtained from training instances. In the training process, the positive sample d_1 for the query q can be chosen as the most relevant documents according to annotated relevance scores. The negative sample d_2 is selected randomly from the document collection.

3.4 Joint learning with regularization

Combining the unsupervised loss l_u in equation 1 on all text data, the supervised loss l_s in equation 2 on all labeled query-document pairs, and the $L2$ norm regularization for weight matrices, one finally arrives at the objective function for the joint learning model, which is:

$$\begin{aligned} L(T, DS) = &\frac{\alpha}{|T|} \sum_{t \in T} l_u(t) + \frac{\beta}{|QD|} \sum_{(q,d_1,d_2) \in QD} l_s(q, d_1, d_2) \\ &+ \sum_{l \in LY} \|W_l\|_F^2 \end{aligned} \tag{3}$$

where T and $|T|$ denote the set of text data and its size, QD and $|QD|$ denote the set of labeled query-document pairs and its size, LY stands for all the hidden and output layers of the framework in figure 1, and W_l is the weight matrix of the layer l in the network. The hyper-parameters α, β control the importance of the unsupervised loss and the supervised loss. The joint loss function $L(T, DS)$ can be optimized in the gradient-based way, and we use the Adam algorithm (Kingma and Ba, 2015) to compute the gradients.

4 Experiments and results

In this section, we conduct IR experiments to demonstrate the effectiveness of our proposed model.

4.1 Data sets

The IR experiments are carried out against standard TREC collections consisting of one Robust track and one Web track, which represent different sizes and genres of heterogeneous text collections. These collections have been broadly used in recent studies (Zheng and Callan, 2015; Guo et al., 2016; Dehghani et al., 2017). The details of these collections and corresponding queries are given in table 1. The Robust dataset is used in the standard form without change. The ClueWeb-09-Cat-B collection (or *ClueWeb* for short) is filtered to the set of documents with spam scores in the 60-th percentile with Waterloo Fusion spam scores[1]. For all TREC queries, we only make use of the title fields for retrieval.

Table 1: IR collection statistics (M = million, B=Billion).

Collections	Doc count	Word count	TREC topics
Robust04	0.5M	252M	301-450, 601-700
ClueWeb	34.0M	26.1B	1-200

In order to build the labeled query-document pairs for supervised learning, we choose to use the more general methodology in (Gupta et al., 2017) instead of the one in (Dehghani et al., 2017) to relieve from data (i.e. AOL queries) only available from industrial labs. We fetch a set of news titles from the China Daily website[2] and use these titles as training queries to produce annotated query-document pairs. We use these training queries to retrieve the document collection with BM25. We make sure that no training queries appear in the evaluation query set in table 1. For each training query, we take the top 500 retrieved documents as positive samples. The negative samples are picked randomly from the document collection. There are other strategies for choosing negative samples (Wieting et al., 2015), which is out of the scope of this paper. For unsupervised learning, we make use of training queries and evaluation document sets listed in table 1, as well as the Wikipedia articles[3] as the external resource.

4.2 Experimental setup

We set the hyper-parameters of our model by following similar tasks such as (Dehghani et al., 2017). The size and number of hidden layers are respectively selected from $\{64, 128, 256, 512, 1024\}$ and $\{1, 2, 3, 4\}$. The values of α, β in equation 3 are chosen from $\{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$. We select the initial learning rate from $\{10^{-3}, 10^{-4}, 5 * 10^{-4}, 10^{-5}, 5 * 10^{-5}\}$. The batch size for learning is selected from $\{64, 128, 256, 512\}$. These model hyper-parameters are tuned on the validation set (20% of the training queries used for validation).

For IR evaluation, we make use of mean average precision (MAP) of top-ranked 1000 documents , precision at rank 20 (P20), and normalized discounted cumulative gain at rank 20 (nDCG20). Statistically significant differences between various models are determined using the two-tailed paired t-test with $p < 0.05$.

We compare the retrieval performance of our joint learning retrieval model with two categories of IR models: classic IR models showing state-of-the-art performance, and the recent neural ranking models for IR. Since our model is representation-focused rather than interaction-focused, we do not plan to compare our model with those based on relevance matching (Guo et al., 2016) in this paper. More importantly, since our model learns from weakly supervised signals by BM25, we are more interested in the comparisons to BM25 and similar models using weakly supervised signals, an experimental strategy also employed in (Dehghani et al., 2017). Under such considerations, we perform experiments with the following baselines:

[1] https://plg.uwaterloo.ca/~gvcormac/clueweb09spam

[2] http://www.chinadaily.com.cn

[3] The wikipedia dump on September 1, 2017 can be obtained from https://dumps.wikimedia.org

Table 2: Retrieval performance of all models on TREC collections. Significant improvement or degradation at the level 0.05 with respect to *BM25* is indicated as (+/-). The other significance comparisons are given in the text.

	Robust04			**ClueWeb**		
	MAP	P20	nDCG20	MAP	P20	nDCG20
BM25	0.248	0.351	0.406	0.091	0.237	0.190
QL	0.245	0.352	0.404	0.092	0.239	0.193
DSSM	0.088^{-}	0.163^{-}	0.184^{-}	0.037^{-}	0.126^{-}	0.104^{-}
NRMS	0.275^{+}	0.378^{+}	0.441^{+}	0.127^{+}	0.302^{+}	0.236^{+}
Our Model	$\mathbf{0.287}^{+}$	$\mathbf{0.391}^{+}$	$\mathbf{0.450}^{+}$	$\mathbf{0.136}^{+}$	$\mathbf{0.317}^{+}$	$\mathbf{0.251}^{+}$

- *Classic models:* The probabilistic *BM25* model and query likelihood (*QL*) model based on Dirichlet smoothing are highly efficient IR models.
- *DSSM:* It is a representative deep matching model proposed in (Huang et al., 2013), which is a representation-focused model. The model is framed as a feed forward neural network with a word hashing layer.
- *NRMS:* It is a weakly-supervised neural IR model learned with automatically annotated query-document pairs (Dehghani et al., 2017). NRMS shows significant improvement over traditional IR models.

4.3 Results and analysis

Comparisons to classic models. We use here the recommended settings of the baseline models according to their original papers. Table 2 reports the experimental results on TREC datasets for our model and all the baseline models. One can find from the results that classic IR models BM25 and QL perform similarly on the two collections, a conclusion that is coincident with previous findings. Since BM25 is the model we employ to produce pseudo labels for supervised learning, we will not compare neural models with QL in the following discussions. The neural IR model DSSM performs significantly worse than the traditional BM25 model, due to its unsuitability for relevance matching and for handling the diverse matching requirements in long documents (Guo et al., 2016). NRMS is a neural ranking model learned from automatically labeled data, which resembles our model. NRMS shows all the significant improvements over BM25. Our model proposed in this paper, by jointly learning from the labeled and unlabeled data, achieves the best overall performance. Our model always significantly outperforms BM25 by a large margin.

Comparisons to neural models. We further compare our model with the neural IR models DSSM and NRMS. We find that our model performs better than DSSM and NRMS on all collections. Our model significantly outperforms DSSM in all the cases considered above. Our model significantly outperforms NRMS with only one exception that is not significant on Robust04 with nDCG20. By the way, we find that NRMS is also always significantly better than DSSM on all collections. The experimental conclusion is that our model is always significantly better than traditional IR models and mostly outperforms neural IR models considered above. Furthermore, we find that using unlabeled data for training in neural IR models is useful, since it leads to significant improvement over the neural models only using labeled data.

Impact of unsupervised learning. It has been confirmed above that our model shows the best performance overall. However, it is not clear how much unsupervised learning contributes to the retrieval performance. We thus compare representations learned in a different setting without the help of unsupervised loss, which amounts to removing the unsupervised loss l_u from equation 3. We perform IR experiments with the new model over data sets in table 1 and list results in table 3. From the results one can find that the performance of the model without unsupervised loss decreases from the joint model with significance in all the cases considered. It indicates that it is beneficial to combine unsupervised

Table 3: Retrieval performance of the model without unsupervised loss. Significant degradation at the level 0.05 with respect to our original model is indicated as -.

	Robust04			**ClueWeb**		
	MAP	P20	nDCG20	MAP	P20	nDCG20
Original Model	0.287	0.391	0.450	0.136	0.317	0.251
Without unsupervised loss	0.262^{-}	0.356^{-}	0.413^{-}	0.114^{-}	0.298^{-}	0.231^{-}

learning with supervised learning in neural IR. Empirical results in this part support our claim in this paper that learning from unlabeled data complements knowledge learned from labeled data in neural IR.

5 Conclusions

In this paper, we propose a neural IR model which jointly learns from labeled and unlabeled data to benefit from both the rich and general semantics in unlabeled data and target-specific features in labeled data. As far as we can tell, it is the first time such a combination is investigated in neural IR. Experiments on TREC collections show that our model, without any human annotation, is significantly better than traditional IR models and recently proposed models based on neural networks. Experiments also show that using unsupervised learning to complement supervised learning with weak supervision is important in IR. A future direction to follow would be to use more expressive architectures such as LSTM to replace feed-forward networks used in this paper.

Acknowledgements

We thank the anonymous reviewers for their valuable comments. This work was supported by the Fundamental Research Funds for Central Universities of CCNU (No. CCNU15A05062).

References

Yu Chen and Mohammed J. Zaki. 2017. Kate: K-competitive autoencoder for text. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD, pages 85–94.

Stephane Clinchant and Florent Perronnin. 2013. Aggregating continuous word embeddings for information retrieval. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 100–109.

Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural ranking models with weak supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, pages 65–74.

Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth J.F. Jones. 2015. Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, pages 795–798.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, CIKM, pages 55–64.

Parth Gupta, Rafael E. Banchs, and Paolo Rosso. 2017. Continuous space models for clir. *Information Processing and Management*, 53(2):359 – 370.

L. He, X. Xu, H. Lu, Y. Yang, F. Shen, and H. T. Shen. 2017. Unsupervised cross-modal retrieval through adversarial learning. In *Proceedings of the 2017 IEEE International Conference on Multimedia and Expo*, ICME, pages 1153–1158.

G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, CIKM, pages 2333–2338.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, ICLR.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS, pages 1097–1105.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521:436–444.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL, pages 912–921.

Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331.

Cheng Luo, Yukun Zheng, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2017. Training deep ranking model with weak relevance labels. In *Proceedings of the Australasian Database Conference*, pages 205–216.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS, pages 3111–3119.

Bhaskar Mitra and Nick Craswell. 2017. Neural models for information retrieval. *CoRR*, abs/1705.01509.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, WWW, pages 1291–1299.

Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. 2016. Improving document ranking with dual word embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW, pages 83–84.

Greg Pass, Abdur Chowdhury, and Cayley Torgeson. 2006. A picture of search. In *Proceedings of the 1st International Conference on Scalable Information Systems*, InfoScale.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 1532–1543.

Antti Rasmus, Harri Valpola, Mikko Honkala, Mathias Berglund, and Tapani Raiko. 2015. Semi-supervised learning with ladder networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS, pages 3546–3554.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014a. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*, CIKM, pages 101–110.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014b. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW, pages 373–374.

Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, pages 363–372.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 3:345–358.

Y. Yang, G. Shu, and M. Shah. 2013. Semi-supervised learning of feature hierarchies for object detection in a video. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR, pages 1650–1657.

X. Ye, Z. Qi, and D. Massey. 2015. Learning relevance from click data via neural network based similarity models. In *Proceedings of the 2015 IEEE International Conference on Big Data*, pages 801–806.

Wen-tau Yih, Kristina Toutanova, John C. Platt, and Christopher Meek. 2011. Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, CoNLL, pages 247–256.

Hamed Zamani and W. Bruce Croft. 2016. Embedding-based query language models. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ICTIR, pages 147–156.

Hamed Zamani and W. Bruce Croft. 2017. Relevance-based word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, pages 505–514.

Shuangfei Zhai and Zhongfei Mark Zhang. 2016. Semisupervised autoencoder for sentiment analysis. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI, pages 1394–1400.

Ye Zhang, Md. Mustafizur Rahman, Alex Braylan, Brandon Dang, Heng-Lu Chang, Henna Kim, Quinten McNamara, Aaron Angert, Edward Banner, Vivek Khetan, Tyler McDonnell, An Thanh Nguyen, Dan Xu, Byron C. Wallace, and Matthew Lease. 2016. Neural information retrieval: A literature review. *CoRR*, abs/1611.06792.

Guoqing Zheng and Jamie Callan. 2015. Learning to reweight terms with distributed representations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, pages 575–584.

Modeling the Readability of German Targeting Adults and Children: An empirically broad analysis and its cross-corpus validation

Zarah Weiß **Detmar Meurers**
ICALL-Research.com
Department of Linguistics
University of Tübingen
{zweiss,dm}@sfs.uni-tuebingen.de

Abstract

We analyze two novel data sets of German educational media texts targeting adults and children. The analysis is based on 400 automatically extracted measures of linguistic complexity from a wide range of linguistic domains. We show that both data sets exhibit broad linguistic adaptation to the target audience, which generalizes across both data sets. Our most successful binary classification model for German readability robustly shows high accuracy between 89.4%–98.9% for both data sets. To our knowledge, this comprehensive German readability model is the first for which robust cross-corpus performance has been shown. The research also contributes resources for German readability assessment that are externally validated as successful for different target audiences: we compiled a new corpus of German news broadcast subtitles, the *Tagesschau/Logo* corpus, and crawled a *GEO/GEOlino* corpus substantially enlarging the data compiled by Hancke et al. (2012).

Zusammenfassung

Wir untersuchen zwei neue Datensätze deutscher Bildungs- und Mediensprache für Kinder und Erwachsene. Die Analyse basiert auf 400 automatisch extrahierten Maßen sprachlicher Komplexität, die verschiedene linguistische Domänen abdecken. Unsere Ergebnisse zeigen, dass in beiden Datensätzen die sprachliche Gestaltung der Texte in ähnlicher Weise breitflächig an ihr jeweiliges Zielpublikum angepasst wird. Unser erfolgreichstes binäres Klassifikationsmodell erzielt Genauigkeitswerte von 89,4% und 98,9% über beide Datensätze hinweg. Unseres Wissens handelt es sich bei diesem umfassend durch verschiedene linguistische Bereiche informtiertem Modell deutscher Text-Lesbarkeit um das erste, für das robuste Ergebnisse in einer korpusübergreifenden Evaluation dokumentiert sind. Darüber hinaus tragen wir mit unserer Arbeit zwei neue Datensätze zur Erforschung deutscher Text-Lesbarkeit bei, die auf Texten basieren, deren Eignung für ihre respektiven Zielgruppen extern durch wiederholte Rezeption validiert wurde: Wir haben aus Untertiteln deutscher Nachrichtenbeiträge das *Tagesschau/Logo* Korpus erstellt. Weiterhin haben wir das *GEO/GEOlino* Korpus beträchtlich erweitert, das ursprünglich von Hancke et al. (2012) erstellt wurde.

1 Introduction

Readability assessment refers to the task of (automatically) linking a text to the appropriate target audience based on its complexity. A diverse spectrum of potential application domains has been identified for this task in the literature, ranging from the design and evaluation of education materials, to information retrieval, and text simplification. Given the increasing need for learning material adapted to different audiences and the barrier-free access to information required for political and social participation, automatic readability assessment is of immediate social relevance. Accordingly, it has attracted considerable

Proceedings of the 27th International Conference on Computational Linguistics, pages 303–317
Santa Fe, New Mexico, USA, August 20-26, 2018.

research interest over the last decades, particularly for the assessment of English (Crossley et al., 2011; Chen and Meurers, 2017; Feng et al., 2010).

For German readability assessment, however, little progress has been made in recent years, despite a series of promising results published around the turn of the decade (Vor der Brück et al., 2008; Hancke et al., 2012). In particular, German readability research has suffered from the lack of a shared reference corpus and sufficiently comparable corpora for cross-corpus testing of readability models: While for English research, the *Common Core* corpus consisting of examples from the English Language Arts Standards of the Common Core State Standards, and the *WeeklyReader* corpus of online news articles have been widely used in studies on English readability and text simplification (Vajjala and Meurers, 2014; Petersen and Ostendorf, 2009; Feng et al., 2010), there are no comparable resources for German. This is particularly problematic, as over-fitting is a potential issue for classification algorithms, especially given the limited size of the typical data sets.

To address these issues, we first present two new data sets for German readability assessment in Section 3: a set of German news broadcast subtitles based on the primary German TV news outlet *Tagesschau* and the children's counterpart *Logo!*, and a GEO/GEOlino corpus crawled from the educational GEO magazine's web site, a source first identified by Hancke et al. (2012), but double in size.[1] The longstanding success of these outlets with their target audiences provides some external validity to the nature of the implicit linguistic adaptation of the language used. As Bryant et al. (2017) showed for German secondary school textbooks, this is not necessarily the case across all linguistic dimensions and adjustments may even be limited to only the surface level of text, sentence, and word length. We conducted a series of analyses on these two data sets to accomplish the following objectives:

1. Investigate how instances of German educational news language differ in terms of language complexity across adult and child target audiences.

2. Build a binary readability model for German educational language targeting adults and children that shows high, robust classification performance across corpora.

For the purposes of our studies, we operationalize child target audience of German educational news language as children aged between 8 and 14. This is the typical audience age range of the child-targeting news media we analyzed.[2] Adult target audience then is defined as over 14 years of age.

To address our first research question, after introducing a broad set of complexity measures in Section 4, we compare their informativeness for distinguishing adult and child level in the two data sets in Section 5. In Section 6, we define a series of readability models for German, including one showing high classification accuracy between 89.4% and 98.9% on both data sets. The paper closes with a discussion of the implications of our results for the current research discussion and an outlook on future work.

2 Related Work

For over a century, text readability has been assessed using surface measure-based readability formula such as the Flesch-Kincaid formula (Kincaid et al., 1975) or the Dale-Chall readability formula (Chall and Dale, 1995), see for an overview DuBay (2004). While these formula are still used in some non-linguistic studies (Woodmansey, 2010; Grootens-Wiegers et al., 2015; Esfahani et al., 2016), a decade ago research shifted towards using more elaborate statistical modeling approaches based on larger sets of linguistically more informed features. Automatic readability assessment has benefited from the use of Natural Language Processing tools for the assessment of syntactic, lexical, and discourse measures and from adapting complexity measures employed in Second Language Acquisition research (Vajjala and Meurers, 2012; Feng et al., 2010). There has also been extensive research on the relevance of cohesion

[1] We are currently negotiating with the broadcasters of *Tagesschau* and *Logo!* and the publishers of *GEO*/*GEOlino* to make the data freely available to other researchers and will make it available from `http://www.icall-research.de` in that case.

[2] The *GEOlino* magazine is advertised as targeting children between 8 and 14 years (cf. `http://www.geo.de/magazine/geolino-magazin`, accessed 11.06.18, 15:49). *Logo!* does not specify the age of its target audience, but has been reported to be particularly popular with children from age 8 to 12 (vom Orde, 2015).

and discourse measures for readability assessment that have successfully been employed for proficiency assessment in the *CohMetrix* project (Crossley et al., 2008; Crossley et al., 2011). Another example is the work by Feng et al. (2010), who evaluate which of the typically proposed measures of text readability are most promising by studying their relevance on primary school students reading material. They find language model features and cohesion in terms of entity density to be particularly useful, as well as measures of nouns. Interestingly, they also observe overall sentence length to be more informative than more elaborate syntactic features. While Feng et al. (2010) do not elaborate further on other lexical measures than POS features, Chen and Meurers (2017) conduct an elaborate cross-corpus study on the use of word frequency features for readability assessment. They show, that the typical aggregation of word frequencies across documents are less informative than richer representations including frequency standard deviations.

In contrast to English, research on readability assessment for other languages, such as German, is more limited. There was a series of articles on this issue from the late 2000s to the early 2010s that demonstrated the benefits of broad linguistic modeling, in particular the use of morphological complexity measures for languages with rich morphological systems like German (Vor der Brück et al., 2008; Hancke et al., 2012), but also Russian (Reynolds, 2016) or French (François and Fairon, 2012). The readability checker *DeLite* of Vor der Brück et al. (2008) is one of the first more sophisticated approaches that went beyond using simple readability formulas for German. The tool employs morphological, lexical, syntactical, semantic, and discourse measures, which they trained on municipal administration texts rated for their readability by humans in an online readability study involving 500 texts and 300 participant, resulting in overall 3,000 ratings. However, due to the specific nature of the data, the robustness of the approach across genres is unclear. Municipal administration language is so particular that results are unlikely to generalize to educational or literary materials, which are more attractive in first and second language acquisition contexts.

Later work by Hancke et al. (2012) also combines traditional readability formula measures, such as text or word length, with more sophisticated lexical, syntactic, and language model, and morphological features to assess German readability, but they employ an overall broader and more diverse feature set than *DeLite*. They investigate readability of educational magazines on the GEO/GEOlino data set, which they compiled from online articles freely available at the GEO magazine's web page. Their work illustrates the relevance of rich linguistic modeling for readability assessment and in particular the value of morphological complexity features for German.

The latest large scale research endeavor for the assessment of German text readability has focused more on identifying linguistic differences between texts targeting different audiences than on building readability models: In the *Reading Demands* project, complexity differences in German secondary school book texts across grade levels and school types were investigated. Berendes et al. (2017) and Bryant et al. (2017) analyze to which extent publishers successfully adapt their reading material to their target audiences. They find a lack of consistent adaptation for passive constructions, concessive and adversative connectives, and relative clauses, and only some limited adaptation in terms of lexical variation, noun complexity, and dependency length measures.

3 Data Sets

3.1 GEO/GEOlino

The GEO/GEOlino data set consists of online articles from one of the leading German monthly educational magazines, *GEO*, and the counterpart for children, *GEOlino*.[3] They are comparable to the *National Geographic* magazine and cover a variety of topics ranging from culture and history to technology and nature. Hancke et al. (2012) first compiled and analyzed a data set from this web resource. We followed their lead and crawled 8,263 articles from the GEO/GEOlino online archive, almost doubling the size of the original corpus. We removed all material flagged as non-article contents by GEO as well as all articles that contained less than 15 words. We further cleaned our data from crawling artifacts and performed near-duplicate detection with the Simhash algorithm. We then grouped all texts into topic categories

[3] `http://www.geo.de` and `http://www.geo.de/geolino`

based on the subdomains they were published under, following the web page topic structure.[4] Table 1 shows the composition of the corpus in terms of the topic groups. Since the number of documents in the different topic groups differ between GEO and the smaller GEOlino set, we created a more balanced subset (GEO/GEOlino$_S$). For this, we included only topic categories existing in both GEO and GEOlino, included all GEOlino texts in those categories and sampled from the GEO texts in those categories until we reached the same overall size of 2480 texts each.

Topic	GEO	GEOlino	$\sum$	GEO$_S$	GEOlino$_S$	$\sum_S$
Do It Yourself	0	663	663	0	0	0
Humanity	1,476	1,168	2,644	1,047	1,168	2,215
Nature	1,704	576	2,280	1,218	576	1,794
Reviews	300	736	1,036	215	736	951
Technology	0	121	121	0	0	0
Travel	1,519	0	1,519	0	0	0
$\sum$	4,999	3,264	8,263	2,480	2,480	4,960

Table 1: Distribution of topics in the full and sampled GEO/GEOlino data set.

3.2 Tagesschau/Logo

The Tagesschau/Logo data set is compiled from subtitles of German daily news broadcasts of *Tagesschau* and its children's counterpart *Logo!*. *Tagesschau* is the dominant national television news service of Germany, produced by the German public-service television network ARD. It broadcasts multiple updated editions of daily news throughout the day. *Logo!* is a television news service for children produced by the German public-service television broadcaster ZDF airing once a day. The data set consists of subtitles for all editions of both news outlets that have been broadcasted from December 2015 to January 2017. For this paper, we limited the *Tagesschau* data to the main edition broadcasted at 8pm. This amounts to overall 421 editions for *Tagesschau* and 415 editions for *Logo!*, with the small difference arising from a lack of Logo! broadcasts on some public holidays or due to special broadcasts. We cleaned the subtitles by removing non-spoken comments (e.g., * *music playing* * or * *cheering* *).

3.3 Characteristics of the two data sets

Table 2 compares the profiles of the GEO/GEOlino$_S$ and the Tagesschau/Logo data sets that we used.

	GEO$_S$	GEOlino$_S$	Tagesschau	Logo
Num. Documents (total)	2,480	2,480	421	415
Num. Words (median)	383	350	1631	1322
Num. Sentences (median)	23	25	167	125

Table 2: Corpus profile for sampled GEO/GEOlino data set and the Tagesschau/Logo data set.

While GEO/GEOlino contains more documents than Tagesschau/Logo, they are considerably shorter in terms of the number of words and sentences they contain. Another difference arises in terms of the medium: GEO/GEOlino articles are self-contained reading material and Tagesschau/Logo subtitles

[4]Subdomains were mapped to topic groups in the following way based on the URL components following `http://www.geo.de` and `http://www.geo.de/geolino`: building ("basteln"), learning ("lernen"), children's recipes ("kinderrezepte"), and competitions ("wettbewerbe") were categorized as DO IT YOURSELF. Jobs ("berufe"), extras ("extras"), photography ("fotografie"), creativity ("kreativ"), info ("info"), love ("liebe"), magazines ("magazine"), human ("mensch"), idioms ("redewendungen"), and knowledge ("wissen") were categorized as HUMANITY. Nature ("natur"), nature and environment ("natur-und-umwelt"), and animal encyclopedia ("tierlexikon") were labeled as NATURE. Book reviews ("buechertipps"), movie reviews ("filmtipps"), game reviews ("spieletest"), and GEO television ("geo-tv") were labeled as REVIEWS. Research and technology ("forschung-und-technik") was labeled as TECHNOLOGY. Travel ("reisen") was labeled as TRAVEL.

complement video material. At the same time, they consist of German educational media language and share the functional goal of conveying information to the reader, so that we consider them to be sufficiently similar to support a cross-corpus analysis.

4 Complexity Analysis

For the assessment of German language complexity, we extract 400 complexity measures using state of the art NLP techniques. All features are theoretically grounded in the contemporary research in linguistic subdisciplines, in particular Second Language Acquisition research, where Complexity is one of three dimensions of language proficiency, together with Accuracy and Fluency (Housen et al., 2012). SLA research has a rich tradition of analyzing the complexity development of learner language, see Lu (2010; 2012) for an overview. Vajjala and Meurers (2012) show that these measures can be successfully applied to readability research. Building on these findings, we follow the SLA definition of complexity as the elaborateness and variability of language (Ellis and Barkhuizen, 2005). Our measures can be grouped into seven categories: i) lexical complexity, ii) clausal complexity, iii) phrasal complexity, iv) morphological complexity, v) discourse complexity, vi) cognitive complexity, and vii) language use. While the former five groups are rooted in the linguistic system, the latter two categories were derived from psycholinguistic research. The resulting complexity assessment covers a broad variety of measures. To the best of our knowledge, this is currently the most extensive feature collection for German complexity assessment.[5] Table 3 gives an overview of the feature categories and how much they contribute to our assessment.[6]

Category	#	Description
Descriptive	2	Total number of sentences and words.
Lexical	73	Lexical diversity measures such as general and POS-specific type-token ratios as well as semantic relatedness measures.
Sentential	119	Ratios measuring sentential elaboration and variation, such as clauses per sentence.
Phrasal	41	Ratios measuring phrasal elaboration and variation, such as modifiers per noun phrase.
Morphological	39	Ratios of inflection, derivation, and composition measures.
Cohesion	48	Subsequent (local) or across text (global) use of implicit or explicit cohesion markers such as connectives, pronouns, or grammatical transitions.
Cognitive	23	Dependency lengths, verb-argument distances, and ratios of cognitive integration costs assessing cognitive processing load based on Gibson's (2000) Dependency Locality Theory.
Language Use	54	Word frequency ratios based on Subtlex-DE (Brysbaert et al., 2011), dlexDB (Heister et al., 2011), Karlsruhe Children's Texts (Lavalley et al., 2015) Approximation of age of active use based on Karlsruhe Children's Texts.

Table 3: Overview over complexity measures grouped by feature categories.

In order to extract these measures, we employ an elaborate analysis pipeline which relies on a number of NLP tools and external linguistic resources. We use OpenNLP 1.6.0 for tokenization and sentence segmentation.[7] This serves as input for the Mate tools 3.6.0 (Bohnet and Nivre, 2012), which perform a morphological analysis, lemmatization, POS tagging, and dependency parsing. We then use the JWord-

[5] Our feature collection draws from varying perspectives on language complexity including SLA and human language processing research. While the confirmation or refutation of specific theories underlying these measures is an interesting research endeavor, our empirical questions focus on which of these features support the distinction of texts targeting different audiences.

[6] We are working on integrating our German complexity analysis pipeline into CTAP (Chen and Meurers, 2016) to make it generally available and will include an online documentation for each feature.

[7] `http://opennlp.apache.org`

Splitter 3.4.0 for compound analysis.[8] The Mate POS tags are further used to inform the Stanford PCFG parser 3.6.0 (Rafferty and Manning, 2008) and the Berkeley parser 1.7.0 (Petrov and Klein, 2007), which we use for constituency and topological field parsing. For all tools, we use the German default models that were provided with them, except for the Berkeley parser, for which we use the topological field model by Ziai and Meurers (2018). With these annotations, we extract all instances of the linguistic constructs that we need to calculate the final 400 complexity ratios.[9]

5 Study 1: Which complexity measures are informative?

5.1 Set-Up

We first want to determine the informativeness of each measure for distinguishing between adult and child target audience. For this, we calculate the information gain of each measure on both data sets using 10-folds cross-validation for training and testing. We then compare across both data sets i) the number of features that are informative, and ii) the 20 most informative measures that show a Pearson correlation smaller than ± 0.8 with each other.[10] This allows us to gain insights into the range of linguistic properties of the documents targeting adults and children. We used WEKA (Hall et al., 2009) to calculate information gain and R for the correlation analysis.

5.2 Results and Discussion

Table 4 shows the percentage of measures that exhibited an average information gain above zero.

Data Set	Percentage	Informative to Total
GEO/GEOlino	79.00%	316/400
Tagesschau/Logo	88.25%	353/400

Table 4: Percentage of informative measures based on 10-folds cross-validated information gain.

Overall, 79.00% of the measures are informative for the GEO/GEOlino data and 88.25% for the Tagesschau/Logo data. This shows, that the documents are adjusted to their different target audiences in terms of a broad range of dimensions of linguistic complexity.

Table 5 provides a deeper look into the linguistic design of the documents by showing the 20 most informative measures distinguishing adult from child targeted documents, including only measures with a correlation less than ± 0.8. The table shows the original rank of each measure before removal of correlated measures, the average merit of each measure for the distinction of the target audience, the type of complexity measures it belongs to, and the feature name.

The results for both data sets show a diverse collection of features, some of which are similar for both data sets, but also some interesting differences. In total the measures seem to be more informative for Tagesschau/Logo, as indicated by the higher average merit, and more correlated, as can be seen from the wider range of original ranks. Language use as captured by frequency measures is particularly relevant for both data sets. The table includes seven measures of word frequency for GEO/GEOlino and five for Tagesschau/Logo. For both data sets, the most informative measure is one of language use: For GEO/GEOlino it is the average minimal age of active use of lexical types found in the Karlsruhe Children's Corpus (KCT) of Lavalley et al. (2015). For Tagesschau/Logo it is the average log lexical type frequency based on Google Books 2000. The other language-use measures are very similar across data sets: Lexical types unknown to the Subtlex-DE data base (Brysbaert et al., 2011), for example, rank 4th and 2nd on both data sets and while on Tagesschau/Logo the lemma frequency per lexical type found in KCT is the 12th most informative measure, its log counterpart ranks 8th on GEO/GEOlino.

[8] `http://www.danielnaber.de/jwordsplitter`

[9] To support transparent comparison with other complexity studies, we include a description of the operationalization of all linguistic units that allow for varying definitions in Appendix A, as has been suggested by Bulté and Housen (2014).

[10] We set the Pearson correlation threshold relatively high since we primarily are interested in qualitatively inspecting the types of measures that are informative, not in removing all correlations.

GEO/GEOlino				Tagesschau/Logo			
Rank	Average Merit	Group	Feature	Rank	Average Merit	Group	Feature
1	0.332 (±0.004)	USE	sumTypesMinAoAPerTypeInKCT	1	0.978 (±0.004)	USE	logTypeFreqsPerTypeInGoogle00
4	0.327 (±0.005)	LEX	syllablesPerToken	31	0.899 (±0.006)	USE	typesNotInSubtlexPerLexicalType
11	0.288 (±0.004)	USE	logTypeFreqsPerTypeInSubtlex	50	0.825 (±0.009)	COH	2PPersPronounsPerNoun
13	0.231 (±0.003)	USE	typesNotInSubtlexPerLexicalType	71	0.754 (±0.012)	COH	probNotSubsPerTransition
15	0.205 (±0.003)	COH	2PPersAndPossPronounsPerToken	82	0.716 (±0.010)	COH	causalConnectivePerSentence
21	0.164 (±0.004)	USE	typesNotInDlexPerLexicalType	85	0.689 (±0.009)	COH	localArgOverlapsPerSentence
23	0.147 (±0.004)	PHR	complexNominalsPerTUnit	88	0.667 (±0.008)	SEN	sumParseTreeHeightsPerFiniteClause
24	0.143 (±0.002)	USE	logLemmaFreqsPerTypeInKCT	89	0.662 (±0.008)	SEN	NPsPerTUnit
25	0.143 (±0.003)	SEN	syllablesInMiddleFieldPerMiddleField	90	0.656 (±0.007)	COH	1PPersPronounsPerToken
28	0.133 (±0.003)	COH	persPronounsPerToken	91	0.657 (±0.010)	MOR	genitivesPerNoun
31	0.133 (±0.003)	SEN	PPsPerTUnit	95	0.633 (±0.011)	PHR	determinersPerNP
33	0.132 (±0.003)	MOR	secondPersonMarkingsPerFiniteVerb	100	0.622 (±0.011)	USE	lemmaFreqsPerTypeInKCT
35	0.123 (±0.004)	MOR	ionTPerToken	101	0.620 (±0.014)	COG	sumLongestDependenciesPerClause
36	0.122 (±0.003)	LEX	synsetPerTypeInGnet	102	0.617 (±0.008)	MOR	compundNounsPerNP
37	0.121 (±0.004)	PHR	complexNominalsPerFiniteClause	103	0.609 (±0.012)	USE	typeFreqsPerTypeInDlex
38	0.120 (±0.004)	SEN	sumNonTerminalNodesPerTUnit	109	0.568 (±0.008)	LEX	MTLD
40	0.118 (±0.004)	USE	typeFreqsPerTypeInSubtlex	110	0.560 (±0.010)	LEX	nonAuxVerbTypesPerNonAuxVerbToken
43	0.114 (±0.002)	COH	pronounsPerNoun	111	0.550 (±0.013)	COH	globalStemOverlapsPerSentence
44	0.113 (±0.002)	USE	logAnnoTypeFreqBand5PerTypeInKCT	117	0.505 (±0.011)	SEN	conjunctionalClausesPerSentence
49	0.111 (±0.002)	COH	3PPersAndPossPronounsPerNoun	119	0.500 (±0.014)	USE	logAnnoTypeFreqBd4PerTypeInDlex

Table 5: Top 20 most informative measures on balanced GEO/GEOlino and Tagesschau/Logo data based on information gain with $r \leq 0.8$.

Cohesion measures are highly informative, too, although more so for Tagesschau/Logo. In particular the use of certain personal or possessive pronouns is highly informative for GEO/GEOlino. The use of second person pronouns ranks highly for both data sets, which may easily be explained by it being used for the informal German address appropriate when speaking to children. This is further corroborated by the ratio of second person verb inflections being ranked as the 13th most important measure. For Tagesschau/Logo, other implicit measures of textual cohesion based on content overlap are also informative as well as the use of causal connectives. Overall 55% of the most informative 20 measures for both data set are captured by these two categories.

The other feature groups are less frequently represented, but provide some interpretable insights into the data. First, both data sets show indications of differences in the degree of nominalization used in language targeting adults and children: For GEO/GEOlino, complex noun phrases per t-unit and finite clause are highly informative as well as the use of the nominalization suffix *-ion*. On Tagesschau/Logo, genitive case, determiners per noun phrase, and the percentage of compound nouns indicate a similar relevance of differences regarding the organisation of the nominal domain. Lexical and sentential complexity seems to be less homogeneous for the distinction of adult and child targeted language across data sets: There are two measures of lexical complexity assessing word length in syllables and the semantic inter-relatedness of words ranked high for GEO/GEOlino, while on Tagesschau/Logo, lexical diversity and verb variation are particularly informative. For sentential complexity, constituency tree complexity, the average length of the middle field, and the use of prepositional phrases per t-unit are particularly informative on GEO/GEOlino. On Tagesschau/Logo, parse tree height and the use of conjunctional clauses are relevant. Cognitive measures do not seem to play an important role on either data set, except for the sum of longest dependencies per clause on Tagesschau/Logo.

Overall, these results clearly show that for both data sets the distinction between target audiences is not just made based on surface modifications such as sentence or word length. In fact, these measures do not occur among the most informative measures at all. Rather, measures of language use and cohesion are predominantly informative for the distinction of adult and child targeting texts, but also measures of phrasal, sentential, lexical, and morphological complexity. The adjustment of the data to their audience observed here thus seems to be more linguistically refined than that found in the *ReadingDemands* textbook data, where Berendes et al. (2017) found only few adjustment across dimensions.

6 Study 2: Can we successfully model readability for German, also across data sets?

6.1 Set-Up

Our second objective is the design of a robust model of educational media language that distinguishes robustly between language targeting adults and children across corpora and genres. For this, we train two binary Sequential Minimal Optimization (SMO) support vector classifier (Platt, 1998) with linear kernels using the WEKA machine learning toolkit (Hall et al., 2009). Each model is tested i) on the same corpus it is trained on, using 10-folds cross-validation, and afterwards ii) on the other data set for cross-corpus testing after training on the full data set. For model performance evaluation, we report classification accuracy and the classification confusion matrices, and random baselines as reference point.

6.2 Results and Discussion

Table 6 shows the accuracy of our SMO models on both data sets and compares them with a random baseline. Both models clearly outperform the baseline of 50.0%. On GEO/GEOlino$_S$, the performance is comparable to the performance observed by Hancke et al. (2012) on the original GEO/GEOlino data.[11]

As Table 7a shows, erroneous classifications are roughly balanced across both classes, showing that the model does not prefer one class over the other. When training a model using only the 20 most informative measures identified in Study 1, we reach an accuracy of 85.1%, i.e., the additional measures only account only for 3.3%.[12] When testing the models on the Tagesschau/Logo corpus, accuracy increases to 98.8% for both models. The confusion matrix for the model using 400 measures in Table 7b seems to indicate

[11] After observing these results, we obtained the original GEO/GEOlino data set from Hancke et al. (2012) and trained and tested a model with 10-folds cross-validation on it. When using the same data, our model outperforms their best performing

Model	Training	Testing	Features	Accuracy	SD
Baseline		GEO/GEOlino$_S$		50.0	
		Tagesschau/Logo		50.0	
10-folds CV	GEO/GEOlino$_S$	GEO/GEOlino$_S$	400	89.4	$\pm$0.09
			20	85.1	$\pm$0.09
	Tagesschau/Logo	Tagesschau/Logo	400	99.9	$\pm$0.04
			20	99.8	$\pm$0.03
Cross-Corpus	GEO/GEOlino$_S$	Tagesschau/Logo	400	98.9	
			20	98.8	
	Tagesschau/Logo	GEO/GEOlino$_S$	400	52.2	
			20	56.7	

Table 6: Classification performance of model on GEO/GEOlino$_S$ and Tagesschau/Logo data

$\downarrow$Obs./Prd.$\rightarrow$	Child$_{GEOlino}$	Adult$_{GEO}$
Child$_{GEOlino}$	2,222	258
Adult$_{GEO}$	267	2,213

(a) 10-folds CV on GEO/GEOlino$_S$

$\downarrow$Obs./Prd.$\rightarrow$	Child$_{GEOlino}$	Adult$_{GEO}$
Child$_{Logo!}$	408	7
Adult$_{TS}$	2	419

(b) Cross-corpus testing on Tagesschau/Logo

Table 7: Confusion matrices for testing models with 400 features trained on GEO/GEOlino$_S$.

a minor tendency towards classifying *Logo!* texts as Tagesschau texts, but due to the low number of incorrect classifications this is not conclusive.

Overall, performance of both models trained on GEO/GEOlino$_S$ on the Tagesschau/Logo data is comparable to the performance of both models trained and tested on Tagesschau/Logo with 10-folds cross-validation, although the confusion matrix for the cross-validated Tagesschau/Logo model using 400 measures does not exhibit any tendency towards predicting one class preferred over the other, as may be seen in Table 8a.

$\downarrow$Obs./Prd.$\rightarrow$	Child$_{Logo!}$	Adult$_{TS}$
Child$_{Logo!}$	415	0
Adult$_{TS}$	1	420

(a) 10-folds CV on Tagesschau/Logo

$\downarrow$Obs./Prd.$\rightarrow$	Child$_{Logo!}$	Adult$_{TS}$
Child$_{GEOlino}$	2,472	8
Adult$_{GEO}$	2,362	118

(b) Cross-corpus testing on GEO/GEOlino$_S$

Table 8: Confusion matrices for testing models with 400 features trained on Tagesschau/Logo

The model trained and tested on Tagesschau/Logo reaches an unexpectedly high accuracy of 99.9% for using 400 measures and 99.8% when using only the 20 most informative measures reported in Study 1. Since the performance remains high when using only 20 measures and the standard deviation across folds is very low, the results seem not to be due to over-fitting. The model learns linguistic properties of the data set that generalize across. It is important to stress here than none of our measures include n-gram language models or any other lexical content features but only complexity measures aggregated over each document.[13]

model with 91.1%, confirming that our approach is in fact competitive with the state of the art.

[12]We do not show the confusion matrices for the models with 20 features, because they are equivalent to the matrices in Table 7. The same holds for the models tested on Tagesschau/Logo and their matrices in Table 8.

[13]Content features are problematic since they can pick up recurring phrases that are characteristic of particular media outlets rather than generalizable linguistic complexity characteristics. E.g., the *Tagesschau* always starts with the greeting "Hier ist das Erste Deutsche Fernsehen mit der Tagesschau." (*Here is the first public German TV channel with the daily news.*).

When testing the models trained on the Tagesschau/Logo data set on the GEO/GEOlino$_S$ data, it becomes apparent that the characteristics learned from the Tagesschau/Logo data set do not generalize, with the model based on 400 measures performing only marginally above chance, and the model using the 20 measures performing slightly better with 56.2%. When considering the confusion matrix for this model in Table 8b, we see that most texts are classified as GEOlino texts, irrespective of whether they belong to GEO or GEOlino. The Tagesschau/Logo trained models do not generalize well to the other adult/child corpus. Since the model trained on GEO/GEOlino$_S$ is highly successful when tested on Tagesschau/Logo, this cannot be due to an actual lack of generalizable differences in the linguistic characteristics of the adult and child targeting texts contained in both data sets. One possible reason for these results may be that, as Study 1 showed, the measures are considerably more informative on Tagesschau/Logo than on GEO/GEOlino$_S$. It could be, that the differences between the news subtitles designed for different target audiences are more extreme than those observable for the GEO magazines. This would explain the surprisingly good performance of the GEO/GEOlino$_S$ model on the Tagesschau/Logo data, which would then be easier to distinguish, while also accounting for the poor performance in the opposite case.

7 Summary and Outlook

We presented a study of the difference between German targeting adults and children, as far as we know the most broadly based linguistic complexity analysis to date. We created and analyzed a novel data set compiled from German news subtitles that consists of news broadcasts for adults and children from the same days, ensuring a relatively parallel selection of topics. We compared this with a newly compiled GEO/GEOlino corpus consisting of online articles of two magazines for adults and children by the same publisher discussing the similar topics. Based on these two data sets, we presented within-corpus (10-fold CV) and cross-corpus experiments and built binary classification models of German educational media text readability that perform with very high accuracy across both data sets. The model is based on a broad range of features that are highly informative for both data sets. This model is a valuable contribution since i) it is based on a considerably broader data basis than previous approaches to German readability, and ii) it successfully generalizes across the data sets, illustrating surprising robustness across rather different text types. The approach presented thus extends the state-of-the-art in Hancke et al. (2012) in terms of the breadth of features integrated and the accuracy and generalizability of the model – and provides two new data sources for this line of research.

The paper also contributes some new insights into the linguistic characteristics of German media language targeting adults and children. Since all the language is produced by adults, it is not necessarily clear how well it is in fact adjusted to the target audience. As demonstrated by Berendes et al. (2017), German textbook publishers indeed do not seem to be adjusting the complexity of the language used according to school type and grade level in any systematic way. Our results for educational media language indicate, that i) both data sets are successfully and broadly adapted towards their target audiences; and ii) that they form two distinct, cross-corpus generalizable constructs of German educational media language for children and adults. In a next step, we plan to test to which extent this linguistically diverse and generalized construct matches the language competence of the intended children target group by comparing it with the Karlsruhe Children's Text corpus (Lavalley et al., 2015). We also plan to further investigate the linguistic properties of our two data sets. In particular, the Tagesschau/Logo data set requires further statistical and qualitative analyses to investigate why its linguistic characteristics generalize well across all folds of the data set itself but not across GEO/GEOlino. We also plan to conduct more analyses of the informativeness of the different complexity feature groups for the target audience distinction.

Acknowledgements

We would like to thank Peter Lindner of the NDR and the editorial office of *ARD Aktuell* for providing us with the news subtitles of the *Tagesschau* broadcasts and Christiane Müller of the ZDF for giving us access to the subtitles from *Logo!*.

References

Stefanie Albert, Jan Anderssen, Regine Bader, Stephanie Becker, Tobias Bracht, Sabine Brants, Thorsten Brants, Vera Demberg, Stefanie Dipper, Peter Eisenberg, Silvia Hansen, Hagen Hirschmann, Juliane Janitzek, Carolin Kirstein, Robert Langner, Lukas Michelbacher, Oliver Plaehn, Cordula Preis, Marcus Pußel, Marco Rower, Bettina Schrader, Anne Schwartz, George Smith, and Hans Uszkoreit, 2003. *TIGER Annotationsschema*. Universität des Saarlandes and Universität Stuttgart and Universität Potsdam.

Karin Berendes, Sowmya Vajjala, Detmar Meurers, Doreen Bryant, Wolfgang Wagner, Maria Chinkina, and Ulrich Trautwein. 2017. Reading demands in secondary school: Does the linguistic complexity of textbooks increase with grade level and the academic orientation of the school track? *Journal of Educational Psychology*.

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

Doreen Bryant, Karin Berendes, Detmar Meurers, and Zarah Weiß. 2017. Schulbuchtexte der Sekundarstufe auf dem linguistischen Prüfstand. Analyse der bildungs- und fachsprachlichen Komplexität in Abhängigkeit von Schultyp und Jahrgangsstufe. In Mathilde Hennig, editor, *Linguistische Komplexität – ein Phantom?* Stauffenburg Verlag, Tübingen.

Marc Brysbaert, Matthias Buchmeier, Markus Conrad, Arthur M. Jacobs, Jens Bölte, and Andrea Böhl. 2011. The word frequency effect: A review of recent developments and implications for the choice of frequency estimates in german. *Experimental Psychology*, 58:412–424.

Bram Bulté and Alex Housen. 2014. Conceptualizing and measuring short-term changes in L2 writing complexity. *Journal of Second Language Writing*, 26(0):42 – 65. Comparing perspectives on L2 writing: Multiple analyses of a common corpus.

Jeanne S. Chall and Edgar Dale. 1995. *Readability revisited: the new Dale-Chall Readability Formula*. Brookline Books.

Xiaobin Chen and Detmar Meurers. 2016. CTAP: A web-based tool supporting automatic complexity analysis. In *Proceedings of the Workshop on Computational Linguistics for Linguistic Complexity*, Osaka, Japan, December. The International Committee on Computational Linguistics.

Xiaobin Chen and Detmar Meurers. 2017. Word frequency and readability: Predicting the text-level readability with a lexical-level attribute. *Journal of Research in Reading*. JRIR-2017-01-0006.R1.

Scott A. Crossley, Jerry Greenfield, and Danielle S. McNamara, 2008. *Assessing text readability using cognitively based indices*, pages 475–493. Teachers of English to Speakers of Other Languages, Inc. 700 South Washington Street Suite 200, Alexandria, VA 22314.

Scott A. Crossley, David B. Allen, and Danielle McNamara. 2011. Text readability and intuitive simplification: A comparison of readability formulas. *Reading in a Foreign Language*, 23(1):84–101, April.

William H. DuBay. 2004. *The Principles of Readability*. Impact Information, Costa Mesa, California.

Duden. 2009. *Deutsche Grammatik*, volume 4. Dudenverlag, 4 edition.

Rod Ellis and Gary Barkhuizen. 2005. *Analysing learner language*. Oxford University Press.

B. Janghorban Esfahani, A. Faron, K. S. Roth, P. P. Grimminger, and J. C. Luers. 2016. Systematic readability analysis of medical texts on websites of german university clinics for general and abdominal surgery. *Zentralblatt fur Chirurgie*, 141(6):639–644.

Lijun Feng, Martin Jansche, Matt Huenerfauth, and Noémie Elhadad. 2010. A comparison of features for automatic readability assessment. In *In Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010), Beijing, China.*

Thomas François and Cedrick Fairon. 2012. An "AI readability" formula for French as a foreign language. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Edward Gibson. 2000. The dependency locality theory: A distance-based theory of linguistic complexity. *Image, language, brain*, pages 95–126.

Petronella Grootens-Wiegers, Martine C. De Vries, Tessa E. Vossen, and Jos M. Van den Broek. 2015. Readability and visuals in medical research information forms for children and adolescents. *Science Communication*, 37(1):89–117.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. In *The SIGKDD Explorations*, volume 11, pages 10–18.

Julia Hancke, Detmar Meurers, and Sowmya Vajjala. 2012. Readability classification for German using lexical, syntactic, and morphological features. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, pages 1063–1080, Mumbay, India.

Julian Heister, Kay-Michael Würzner, Johannes Bubenzer, Edmund Pohl, Thomas Hanneforth, Alexander Geyken, and Reinhold Kliegl. 2011. dlexDB - eine lexikalische Datenbank für die psychologische und linguistische Forschung. *Psychologische Rundschau*, 62:10–20.

Alexis Housen, Folkert Kuiken, and Ineke Vedder. 2012. Complexity, accuracy and fluency: Definitions, measurement and research. In Alex Housen, Folkert Kuiken, and Ineke Vedder, editors, *Dimensions of L2 Performance and Proficiency*, Language Learning & Language Teaching, pages 1–20. John Benjamins.

Kellogg W. Hunt. 1970. Do sentences in the second language grow like those in the first? *TESOL Quarterly*, 4(3):195–202.

J. Peter Kincaid, Robert P. Fishburne, Richard L. Rogers, and Brad S. Chissom. 1975. Derivation of new readability formulas (Automated Readability Index, Fog Count and Flesch Reading Ease formula) for Navy enlisted personnel. Research Branch Report 8-75, Naval Technical Training Command, Millington, TN.

Rémi Lavalley, Kay Berkling, and Sebastian Stüker. 2015. Preparing children's writing database for automated processing. In *LTLT@ SLaTE*, pages 9–15.

Xiaofei Lu. 2010. Automatic analysis of syntactic complexity in second language writing. *International Journal of Corpus Linguistics*, 15(4):474–496.

Xiaofei Lu. 2012. The relationship of lexical richness to the quality of ESL learners' oral narratives. *The Modern Languages Journal*, pages 190–208.

Sarah E. Petersen and Mari Ostendorf. 2009. A machine learning approach to reading level assessment. *Computer Speech and Language*, 23:86–106.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April.

John C. Platt. 1998. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research.

Anna N. Rafferty and Christopher D. Manning. 2008. Parsing three German treebanks: lexicalized and unlexicalized baselines. In *Proceedings of the Workshop on Parsing German*, PaGe '08, pages 40–46, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marga Reis. 2001. Bilden Modalverben im Deutschen eine syntaktische Klasse? *Modalität und Modalverben im Deutschen*.

Robert Reynolds. 2016. Insights from Russian second language readability classification: complexity-dependent training requirements, and feature evaluation of multiple categories. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 289–300, San Diego, CA, June. Association for Computational Linguistics.

Karsten Schmidt. 2016. Der graphematische Satz. *Zeitschrift für germanistische Linguistik*, 44(2):215–256.

Sowmya Vajjala and Detmar Meurers. 2012. On improving the accuracy of readability classification using insights from second language acquisition. In Joel Tetreault, Jill Burstein, and Claudial Leacock, editors, *In Proceedings of the 7th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–173, Montréal, Canada, June. Association for Computational Linguistics.

Sowmya Vajjala and Detmar Meurers. 2014. Assessing the relative reading level of sentence pairs for text simplification. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 288–297, Gothenburg, Sweden, April. ACL, Association for Computational Linguistics.

Heike vom Orde. 2015. Kindernachrichten im Fernsehen. Eine Zusammenfassung zentraler Forschungsergebnisse zum Format logo! *Televizion*, 28/2015/2:40–42.

Tim Vor der Brück, Sven Hartrumpf, and Hermann Helbig. 2008. A readability checker with supervised learning using deep syntactic and semantic indicators. *Informatica*, 32(4):429–435.

Karl Woodmansey. 2010. Readability of educational materials for endodontic patients. *Journal of Endodontics*, 36:1703–1706.

Ramon Ziai and Detmar Meurers. 2018. Automatic focus annotation: Bringing formal pragmatics alive in analyzing the Information Structure of authentic data. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, New Orleans, LA. ACL. To appear.

Appendix A. Definitions of Linguistic Units

Clauses are all maximal projections of finite verbs and elliptical constructions with sentential status (i.e. all sub-trees tagged with *S*), as well as *to* infinitives that have a sentential status (*satzwertige zu Infinitive*).

Complex t-units are t-units that include subordinate clauses.

Conjunctional clauses are all dependent clauses that are introduced by a subordinating conjunction such as *dass*, *weil*, or *wenn*.

Dependent clauses with conjunction are all conjunctional clauses, but also interrogative and relative clauses. Dependent clauses without conjunction are mostly dependent main clauses, such as *Ich weiß, es ist spät*.

(Graphematic) sentences are strings of at least one token that are ended by sentence ending punctuation marks: *!*, ., *?*. There is a broad discussion on alternative sentence definitions, see for example Schmidt (2016) for a more elaborate theoretical account. However, since sentences are identified by sentence segmentation tools, which are primarily based on punctuation, sentences are always defined as graphematic sentences.

Half modals are *haben*, *sein*, *scheinen*, *drohen*, *versprechen*, if they govern an infinitive with *zu* (Duden, 2009, §101), e.g. *ist zu machen*, *droht zu schneien*.

Lexical words are all nouns, adjectives, adverbs, foreign words, numbers, main verbs, and modal verbs. Note that there is an ongoing discussion on whether modals actually qualify as lexical words (Reis, 2001), hence there is also a subset of lexical words excluding modals employed throughout the system.

Parts-of-Speech are operationalized following the Tiger POS tags (Albert et al., 2003, 121).

Quasi passives are *bekommen*, *erhalten* or *kriegen* if they govern a past participle (Duden, 2009, §179), e.g. *bekommt gemacht*, *kriegt eröffnet*.

T-units are "one main clause plus any subordinate clause or non clausal structure that is attached to or embedded in it" (Hunt, 1970, 4).

Appendix B. Example Extracts from *Tagesschau* and *Logo!* subtitles

Report on New Years shooting in Istanbul by *Tagesschau*, extracted from the subtitles for the broadcast on 01.01.2017, 20:00.

> In der Türkei ist der Jahreswechsel von einem Anschlag in Istanbul überschattet: Mind. ein bewaffneter Angreifer drang in einen Nachtclub ein und schoss um sich. 39 Menschen wurden getötet und mehr als 60 verletzt. Unter den Todesopfern sind zahlreiche Ausländer. Ob Deutsche betroffen sind, ist unklar. Die Suche nach dem Täter dauert an, bekannt hat sich niemand. Das Attentat ereignete sich im europäischen Teil Istanbuls. Dort liegt direkt am Bosporus der Club "Reina", der bei Prominenten beliebt ist. Nur eine Stunde währte in der Türkei die Hoffnung, 2017 könnte ruhiger werden als 2016, das von Bombenanschlägen geprägt war. Doch um 1.15 Uhr Ortszeit macht im Istanbuler Nachtclub "Reina" ein Attentäter mit einem Gewehr Jagd auf Gäste einer Silvesterparty. Zuvor wurde vorm Club ein Polizist erschossen. Der Täter konnte fliehen, eine Großfahndung läuft. Bis zu 800 Personen sollen sich in der Diskothek aufgehalten haben. Gäste berichten, Panik sei ausgebrochen. Einige Besucher sollen in den Bosporus gesprungen sein. Unter den Toten und Verletzten sind Ausländer. Bekannt hat sich niemand zu der Tat. Türkische Medien vermuten den IS hinter dem Terrorakt. Die Regierung verhängte eine Nachrichtensperre. Wir lassen uns vom Terror nicht beirren. Was hier passierte, kann morgen an einem anderem Ort geschehen. Es gibt keine Garantien. Der Nachtclub "Reina" liegt am Bosporus, im Stadtteil Ortaköy. Er ist der berühmteste der Türkei, teuer und bei Touristen beliebt. Die Sicherheitsvorkehrungen waren landesweit erhöht worden. In Istanbul waren 17.000 Polizisten im Einsatz. Trotz Großaufgebot der Polizei, hochaktiver Geheimdienste, Ausnahmezustand und markiger Politikerworte: Die Sicherheitslage in der Türkei spitzt sich zu. Beängstigende Aussichten für Wirtschaft und Menschen.

Report on New Years shooting in Istanbul by *Logo!*, extracted from the subtitles for the broadcast on 02.01.2017, 19:50 (no broadcast on 01.01.2017).

> In der türkischen Großstadt Istanbul hat es an Silvester einen Anschlag in einer Disco gegeben. Ein Mann stürmte mit einem Gewehr in den Club und hat 39 Menschen getötet, darunter auch zwei Männer, die in Deutschland gelebt haben. Die türkische Polizei sucht jetzt nach dem Täter. Er ist seit dem Anschlag auf der Flucht. Auch am zweiten Tag nach dem Anschlag kamen viele Menschen an die Polizeiabsperrung, um Blumen für die Opfer niederzulegen. Der Terrorist stürmte dort in der Silvesternacht mit einem Gewehr in die Disco. Ich war völlig geschockt, konnte mich nicht bewegen. Der Täter schoss erst auf einen Polizisten und dann auf die Gäste. Wir hörten plötzlich Schüsse, da sind wir raus aus dem Ballsaal auf die Terrasse und haben uns dort versteckt. Im Internet behauptet die Terrorgruppe IS, Islamischer Staat, dass sie hinter dem Anschlag stecke. Die Kämpfer dieser Terrorgruppe wollen, dass alle Menschen nach ihren strengen religiösen Regeln leben. Wer sich nicht daran hält, wird sogar umgebracht. Besonders aktiv ist der IS in Teilen von Syrien und dem Irak. Beide Länder grenzen an die Türkei. Dort haben die Kämpfer in letzter Zeit schon öfter Anschläge verübt. In der ganzen Türkei sucht die Polizei jetzt nach dem Attentäter. Acht Verdächtige wurden schon festgenommen. Auf logo.de könnt ihr mehr zur Terrorgruppe Islamischer Staat lesen und da gibt es auch viele Infos zu unserem nächsten Thema.

Appendix C. Example Articles from GEO and GEOlino

GEO article titled "Was ist ein Planet?" (What is a Planet?).[14] It discusses criteria celestial bodies need to fulfill to be considered a planet.

> Lange bezeichneten Menschen alle Lichtpunkte, die über den Nachthimmel wanderten, als Planeten (griech. planáomai = umherirren) – gleich, ob es sich um Venus, Mars, Mond oder Asteroiden handelte. In der Neuzeit durften den Titel nur noch die großen Himmelskörper tragen, die um die Sonne kreisten, aber keine Monde waren – also nicht ihrerseits einen anderen Planeten umrundeten. Als Astronomen von 1992 an in den Randbezirken des Sonnensystems immer neue Objekte entdeckten, manche ähnlich groß wie Pluto (bis dahin der neunte Planet), sah sich die Internationale Astronomische Union genötigt, erstmals zu definieren, was ein Planet genau ist. Nach heftigen Diskussionen beschlossen die Astronomen 2006 die Resolution B5. Demnach muss ein Planet drei Kriterien erfüllen: Er muss um die Sonne kreisen. Er muss ausreichend Masse aufweisen, sodass er unter seiner eigenen Schwerkraft eine nahezu runde Form angenommen hat. Und er muss die Umgebung seiner Umlaufbahn freigeräumt haben. Objekte, die ihm auf seiner Bahn nahekommen, "schluckt" er in einer Kollision oder schleudert sie in einen anderen Orbit. Pluto, Eris und andere große Himmelskörper zählen nun zu den Zwergplaneten, da sie es nicht schaffen, ihre Bahn zu bereinigen, sondern sie sich mit anderen Objekten teilen. Damit kreisen nach derzeitigem Stand acht Planeten um die Sonne. Die Astronomen unterteilen sie in die vier terrestrischen Planeten Merkur, Venus, Erde, Mars (sie werden wegen ihrer festen Oberfläche häufig steinige Planeten genannt) und in die vier jovianischen – jupiterähnlichen – Planeten Jupiter, Saturn, Uranus, Neptun (aufgrund ihrer Zusammensetzung oft als Gasplaneten oder Gasriesen bezeichnet). Wobei Uranus und Neptun manchmal auch als "Eisriesen" beschrieben werden, da sie weniger Wasserstoff als Jupiter und Saturn enthalten, dafür mehr gefrorenes Methan, Wasser und Ammoniak.

GEOlino article titled "Sieben erdähnliche Planeten entdeckt" (Seven Earth-Like Planets Discovered).[15] It reports on the discovery of seven new planets that orbit Trappist-1.

> Dass neue Planeten entdeckt werden, ist erstmal nichts ungewöhnliches. Doch der Fund dieser sieben sogenannten Exoplaneten (Planeten wie Kepler-452b , die sich um einen Stern - außerhalb des Einflusses unserer Sonne - bewegen) ist etwas ganz Besonderes: Denn sechs der neu entdeckten Planeten liegen in einer Temperaturzone, in der Leben möglich ist. Auf den meisten Planeten ist es entweder kochend heiß oder eiskalt - schwierige Bedingungen für die Entwicklung von Leben. Die Sonne der Exoplaneten, der Zwergstern Trappist-1 , ist viel kleiner als die Sonne unseres Sonnensystems : Trappist-1 besitzt nur acht Prozent der Masse unserer Sonne und zwölf Prozent ihres Durchmessers. Auf drei der entdeckten Exoplaneten könnte sogar Wasser existieren, denn ihr Abstand zur Zwergsonne liegt in einem Temperaturbereich, in dem Wasser weder gefrieren noch verdampfen würde. Hier wäre also eine Art von Leben möglich, wie wir es auf unserer Erde kennen.
>
> Die sieben Planeten haben in etwa die Größe unserer Erde und sind wahrscheinlich Gesteinsplaneten. Sie alle umkreisen ihre Sonne, den Stern Trappist-1, der knappe 40 Lichtjahre (ein Lichtjahr ist die Strecke, die Licht in einem Jahr zurücklegt) von uns entfernt im Sternenbild Wassermann liegt. Weil die Sonne des Trappist-1-Systems so klein ist, können die Planeten diese wesentlich schneller umkreisen als wie es in unserem Sonnensystem möglich ist. Die sechs Planeten, die dem Zwergstern am nächsten sind, umrunden ihn in eineinhalb bis zwölf Tagen. Sie haben damit einen engeren Orbit als der Merkur um die Sonne .

[14] `http://www.geo.de/wissen/weltall/15396-rtkl-definitionssache-was-ist-ein-planet`, accessed 11.06.18, 16:06.

[15] `http://www.geo.de/geolino/wissen/weltraum/sieben-erdaehnliche-planeten-entdeckt`, last accessed 11.06.18, 16:06.

Automatic Assessment of Conceptual Text Complexity Using Knowledge Graphs

Sanja Štajner and **Ioana Hulpuş**
Data and Web Science Group
University of Mannheim, Germany
{sanja,ioana}@informatik.uni-mannheim.de

Abstract

Complexity of texts is usually assessed only at the lexical and syntactic levels. Although it is known that conceptual complexity plays a significant role in text understanding, no attempts have been made at assessing it automatically. We propose to automatically estimate the conceptual complexity of texts by exploiting a number of graph-based measures on a large knowledge base. By using a high-quality language learners corpus for English, we show that graph-based measures of individual text concepts, as well as the way they relate to each other in the knowledge graph, have a high discriminative power when distinguishing between two versions of the same text. Furthermore, when used as features in a binary classification task aiming to choose the simpler of two versions of the same text, our measures achieve high performance even in a default setup.

1 Introduction

A text can be complex on various levels. It can contain many infrequent words that are unknown to the reader (lexical complexity). It can contain long sentences with difficult syntactic structures or syntactic structures unknown to the reader (syntactic complexity), especially in the case of children or non-native language learners. A much less studied level of text complexity is the conceptual or semantic level, which accounts for the amount of background knowledge required to understand the meaning of the text. While many methods have been proposed so far for automatic assessment of lexical and syntactic text complexity (Vajjala and Meurers, 2014; Vajjala and Meurers, 2015; Ambati et al., 2016), automatic assessment of conceptual text complexity has never been attempted so far. Similarly, in the task of automatic text simplification, only two systems attempted at simplifying the overall text structure by eliminating irrelevant pieces of information (Narayan and Gardent, 2014; Štajner and Glavaš, 2017).

Texts that are conceptually complex tend to contain difficult and abstract concepts which are not easy to relate to each other, or they mention numerous entities which are not closely related, requiring substantial background knowledge. Kintsch and van Dijk (1978) distinguish between the micro- and macrostructure of text semantics. The former takes into account the individual concepts and their local relations, while the latter considers their organization at a global level. Conceptual complexity is reflected in both those components and thus conceptual simplification must also be performed on both levels.

Conceptual complexity is especially important for low-knowledge readers, who might have problems accessing their prior knowledge (Denton et al., 2015; McNamara et al., 2012). They benefit particularly from texts that require less inferences, and not so much from simplifications at the linguistic level (Arfé et al., 2017). The amount of inferences that readers need to make in order to fill in the gaps in text coherence, and the effort needed to understand the text is a measure of cognitive or conceptual text complexity (Arfé et al., 2017).

Our work sets up a new task of automatic assessment of conceptual complexity. We define conceptual complexity as the level of background knowledge necessary to understand the text. We pose a number of hypotheses on how structured encyclopedic knowledge bases can be used for this task, and propose a number of graph-based features (Section 4) which capture both micro- and macrostructure of conceptual

Proceedings of the 27th International Conference on Computational Linguistics, pages 318–330
Santa Fe, New Mexico, USA, August 20-26, 2018.

text complexity. Our experiments, carried out over documents from a high-quality language learners corpus for English, show that the graph structure of encyclopedic knowledge can be leveraged in order to accurately assess conceptual complexity of informational text (i.e. news stories). This is particularly important as simplification of such texts improves social inclusion of many target audiences (Carroll et al., 1998; Aluísio and Gasperin, 2010; Saggion et al., 2011; Štajner et al., 2014).

2 Knowledge Graphs for Measuring Conceptual Complexity

Language as a networked system has been vastly researched over the last decades, especially with the purpose of capturing word semantics (Borge-Holthoefer and Arenas, 2010). One of the core concepts that have been proposed is that of *semantic networks*. Various types of semantic networks have been studied, with the aim of explaining the organization of human semantic knowledge: hierarchical networks (Collins and Quillian, 1969), associative networks (Nelson et al., 2004), thesauri such as Roget's thesaurus (Jarmasz and Szpakowicz, 2012), terminological ontologies such as WordNet (Miller, 1995), as well as word networks resulted from corpus analysis, e.g. co-occurrence networks (Stubbs, 1996).

Most of these networks were shown to have very similar large-scale structures, and furthermore, also similar to the structure of the World-Wide-Web (Steyvers and Tenenbaum, 2005). Especially in the case of associative networks which are built based on human memory tasks, and corpus based networks that are based on language use, their similarity to the structure of online information networks is very promising as it enables the analysis of information networks as a proxy for human memory.

In this direction, Griffiths *et al*. (2007) hypothesize that if the knowledge in human memory is organized similarly to the knowledge on the World-Wide-Web, then the retrieval processes solve the same problem: identifying items that are relevant to a query from a large network of interconnected pieces of information. They show that PageRank (Brin and Page, 1998), the web page ranking measure, when applied to words in semantic networks, is correlated with prominence of concepts in human memory (Griffiths et al., 2007). This provides us with the intuition that texts that refer to concepts that have high PageRank in semantic networks are easier to understand on a conceptual level.

Grounding our work on those previous findings, we go even further and hypothesize that the background knowledge required during text reading, and which impacts the conceptual complexity of the text, can be estimated by analyzing such networked knowledge resources. However, news articles are well known for their abundance in named entities, therefore semantic networks based on lexical resources cannot cover much of the required background knowledge for news understanding. We address this by focusing on a particular type of semantic networks, specifically *knowledge graphs*. Like all semantic networks, knowledge graphs represent human knowledge by means of a network. But while traditional semantic networks represent knowledge starting from use of language, having nodes as words and relations as untyped word associations, knowledge graphs capture real world concepts and entities, related by facts. We therefore propose a series of measures that leverage the background knowledge graph in order to assess conceptual text complexity.

We focus on a particular knowledge graph that is freely available, DBpedia[1], a network representation of Wikipedia. Its purpose is to provide Wikipedia knowledge in a structured, automatically queryable format. At its core lies the DBpedia Ontology (Bizer et al., 2009), which contains more than 4 million entities, structured based on a hierarchy of 685 classes and 2795 properties. The DBpedia entities correspond to Wikipedia articles. The entities are connected by relations extracted from the Wikipedia article Infobox sections, resulting in a networked structure.

An important characteristic of DBpedia that renders it particularly suitable for conceptual text complexity of informational text such as news, is the fact that, having been created out of Wikipedia's human created Infoboxes, it covers a very broad range of topics. Moreover, being backed by a very active community of Wikipedia editors, Wikipedia's set of article entries is very up-to-date, which is another advantage it has over alternative knowledge bases when dealing with news articles. This makes DBpedia very promising for the task of assessing the amount of required background knowledge for informational text understanding.

[1] www.dbpedia.org

Knowledge graphs such as DBpedia have been widely and successfully used in a series of language understanding tasks such as computing semantic relatedness / similarity between words (Milne and Witten, 2008; Zhu and Iglesias, 2017), word-sense disambiguation and entity linking (Usbeck et al., 2014; Hulpuş et al., 2015), topic/cluster labeling (Hulpuş et al., 2013), document similarity (Schuhmacher and Ponzetto, 2014), etc.

In this paper, we propose to use knowledge graphs for automatically assessing conceptual complexity of text. We argue that the amount of required background knowledge can be estimated by analyzing the parts of knowledge graph that are activated by the text.

3 Easy-to-read Texts

Many initiatives have proposed guidelines for producing plain, easy to read text, which would be more accessible to wider audiences. The "Federal Plain Language Guidelines"[2] instruct how to write governmental documents for the general public. "Make it Simple" guidelines (Freyhoff et al., 1998) assist writers in producing texts for people with intellectual disabilities and other people who cannot read complex texts. These guidelines were proven to improve efficiency (search and reading time) and effectiveness (comprehension scores) in both intellectually disabled people, and the control group (Karreman et al., 2007). Some of the main advises, that appear in all guidelines are: (1) omit unnecessary words; (2) cover only one main idea per sentence; (3) avoid abstract concepts; (4) and avoid definitions as much as possible by removing or replacing a complicated concept, unless the concept is crucial for document understanding.

Ideas from such guidelines are also incorporated in instructions for manual simplification of already existing texts for autistic readers (Orasan et al., 2013), people with Down syndrome (Saggion et al., 2011), or language learners (Newsela, 2016). Several examples of such simplifications found in Newsela's articles are presented in Table 1.

We notice that the above mentioned text requirements resonate with structural properties of knowledge graphs-based representations of text. This motivates the graph-based measures proposed in the next section for assessing text conceptual complexity.

4 Graph-Based Features for Capturing Conceptual Complexity

In our aim for devising features for measuring conceptual complexity, an important objective is to make them independent of syntactical and lexical complexity measures. For instance, one indicator of syntactical complexity is the length of sentences. The length of a sentence influences the number of concepts in the sentence. Therefore, we suggest only measures that are not influenced by the number of concepts in the sentence.

With respect to lexical complexity, the use of an entity linker (see Section 5.2), ensures that purely lexical transformations, that for instance replace a less common word with a more common synonym (e.g. *murder* $\rightarrow$ *killing*, in example #4 in Table 1) are not captured by our measures (as they link to the same entity[3]). In contrast, the conceptual simplification in which a concept is replaced with a more common and more general concept (e.g. *domestic abuse* $\rightarrow$ *beating*, in example #4 in Table 1) is captured by our features, as those two words link to different concepts in DBpedia.[4]

We explore three types of graph-based measures, each focusing on capturing different particularities of conceptual complexity.

4.1 Single Node Measures

The first type of graph measures that we analyze refers to single concepts.[5] These measures target three types of transformations that occur during simplification: (i) removal of non-essential concepts that demand more background knowledge (example #1 in Table 1), (ii) replacement of non-essential

[2] http://www.plainlanguage.gov

[3] http://dbpedia.org/resource/Murder

[4] http://dbpedia.org/resource/Domestic_violence and http://dbpedia.org/page/Strike_(attack)

[5] We use terms concept, node and entity interchangeably, to refer to nodes in the knowledge graphs.

Idx.	Original	Simplified
#1	A January 2013 study of **sediment cores** from lakes near **Alberta** oil sand mines showed...	A January 2013 study on lakes near oil sand mines showed...
#2	Fishermen say **pickerel** and **northern pike** in the lake show bulging eyes and other deformities.	Fishermen say **fish** in the lake show bulging eyes and other deformities.
#3	He says this new technology could be as important as Google Maps, which revolutionized **navigation**.	He says this new technology could be as important as Google Maps, which revolutionized **maps and directions**.
#4	The researchers said crimes like murder and **domestic abuse** could rise 16 percent.	Crimes such as **beatings** and killing could climb as much as 16 percent, the researchers said.
#5	But when we tried to reintroduce Keeva, it didn't go as we had hoped. So the final decision was made and we started looking for a surrogate.	The reintroduction did not go as hoped. So the decision was made to look for a surrogate. **A surrogate takes on the responsibility of parenting a child to whom they are not related by blood.**
#6	Noah Madson remembers being exhausted after hours of tests for his attention deficit hyperactivity disorder.	Noah Madson remembers being exhausted after hours of tests for ADHD, **a disorder that makes concentration difficult**
#7	The truck driver, who doubles as cook, had donned a hair net and plastic gloves and was grilling burgers as smoke billowed from an exhaust fitted to the roof.	The truck driver, who doubles as cook, had donned a hair net and plastic gloves and was grilling burgers. Smoke blew from an exhaust fitted to the roof.
#8	Lions are vanishing in Africa, where they have long been a **symbol** of the continent's **wild beauty**, **power** and **freedom**. The disappearance has researchers worried ...	Lions are disappearing from the African continent, and that has researchers worried.
#9	... said Mark Southerland, **a private consulting** ecologist **who has worked with the Department of Natural Resources**.	... said Mark Southerland, an ecologist.
#10	"I don't think we can win. I don't even know what winning looks like," Rigney said. **As she spoke, she looked out her kitchen window where whitecaps had formed on the lake.** "This oil is just too important for the rest of Canada."	"I don't think we can win. I don't even know what winning looks like," said Rigney. "This oil is just too important for the rest of Canada."

Table 1: Examples of original and simplified text snippets (Newsela, 2016)

demanding concepts with more commonly known ones (examples $\#2$, $\#3$ and $\#4$ in Table 1), and (iii) avoidance of abstract concepts (example $\#8$).

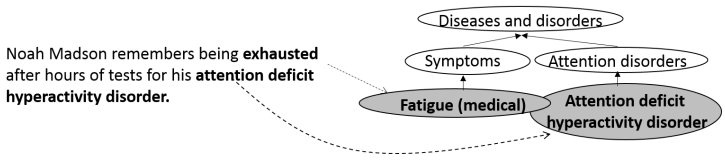
We claim that these edits lead to nodes with different graph properties being favored in simplified texts. There are multiple node properties that come to mind, but we chose to focus on a few most commonly used ones in the network analysis literature.

Node degree is one of the most straightforward measures that describe the popularity of a concept in a network. In knowledge graphs, this translates to the number of concepts that have a direct relation to the concept in question. When referring to the node degree, we consider both incoming and outgoing edges. Under the assumption that the concepts with high degree are more commonly known, our hypothesis is that *when texts are simplified, the high degree nodes are kept, while the lower degree nodes might be omitted, resulting in a higher average node degree in simplified texts (H1).*

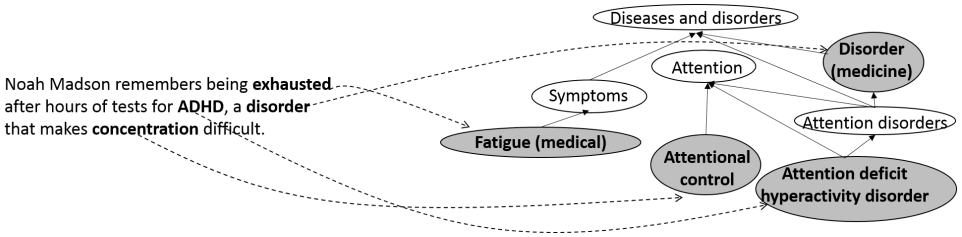
Node clustering coefficient computes how many of the node's neighbors are directly connected among themselves. Therefore, if D is the number of neighbors of a node (equal to its degree), then the clustering coefficient is computed as $\frac{2\times\#related\_neighbors}{D(D-1)}$ The nodes with a low clustering coefficient can be assumed to be more general, since their neighbors tend to be unrelated. Our hypothesis is that *simpler texts tend to favor nodes with low clustering coefficient (H2).*

Node PageRank makes use of a couple of key ideas: (i) a recursive definition of importance such that nodes are important if they are connected to the important nodes; (ii) the importance that a node transmits to a neighbor is diluted by the total number of neighbors that the node has; (iii) a probability governs the decision of transmitting information to a neighbor or randomly to any node in the network. Since PageRank is a measure of node importance in a network, our hypothesis is that concepts with high PageRank in the knowledge graph are prominent in human collective knowledge. We therefore hypothesize that *the conceptually more complex texts will contain more concepts with lower PageRank as compared to the conceptually simpler texts which would, in turn, favor high PageRank concepts (H3).*

We extract the above described measures for all the entities in the text, and use their arithmetic means as conceptual complexity features of the documents. As such, these features should contribute to capturing the semantic *macrostructure* of the text, as defined by Kintsch and van Dijk (1978).



(a) Example sentence before simplification



(b) Example sentence after simplification

Figure 1: Example of simplification through explicitly defining complex but crucial concepts. The simplification leads to a reduced average pairwise shortest path length between the text concepts. The graphs shown in this illustration are drastically reduced, as the complete graphs would be much too complex for visual analysis.

4.2 Pairwise Measures

Besides the properties of the individual entities found in texts, we find it of interest to understand how the entities found within the same sentence or paragraph relate to each other, therefore capturing the *microstructure* of conceptual complexity. We expect that in simpler texts, as a result of the simplification process, the concepts that occur in the same sentence/paragraph are more semantically related.

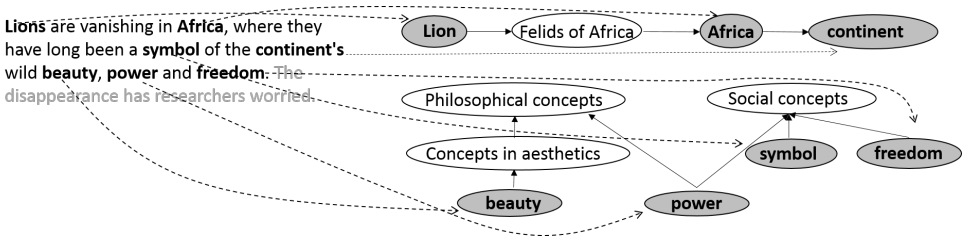
One transformation that is targeted by these measures is that of adding definitions to difficult but essential concepts (examples $\#5$ and $\#6$ in Table 1). The intuition is that when definitions are provided, these explanations usually contain terms that are strongly related to the term they define, increasing the overall relatedness of the simplified text segment.

Another transformation targeted by the pairwise similarity measures is the sentence splitting, that follows the "one main idea per sentence" principle (see Section 3). Each newly formed sentence has a reduced scope, referring to closely related concepts. This is illustrated in the example $\#7$ in Table 1. The original sentence was split so that the smoke topic is mentioned in its own sentence. As a result, the newly formed sentence maintains the particularly high relatedness between its concepts *smoke* and *exhaust*, while separating these concepts from the ones they are not related to such as *driver, cook, hair net, gloves, burger*.

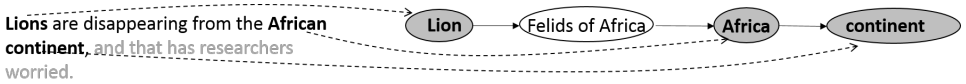
To study those phenomena, we extract the following two measures.

Length of the shortest path is a straightforward measure that we expect to be negatively correlated to two entities' semantic relatedness. When we refer to the paths in the knowledge graph, we consider them undirected. Given a text segment, we compute the length of the shortest path in the knowledge graph between all pairs of entities found in it, and hypothesize that *in a simpler text, the average shortest path lengths between concepts occurring in the same sentence or paragraph tend to be shorter than in conceptually more complex texts (H4).* Figures 1a and 1b visualize the paths between the main concepts extracted from example $\#6$. As shown, by adding a short definition, the average length of shortest path between seed nodes (concepts referred to by the text) becomes shorter: from 4.00 in Figure 1a, to 3.16 in Figure 1b.

Exclusivity-based semantic relatedness is a state-of-the-art relatedness measure (Hulpuş et al., 2015; Zhu and Iglesias, 2017), that considers more paths that connect two concepts rather than just the shortest one. It is based on two key ideas: (i) the direct relations between two nodes are not equally important; their importance is instead governed by an *exclusivity* measure that gives a higher weight to relations whose relation type is less common among the relations of the two target nodes; (ii) the semantic relat-



(a) An original sentence (Newsela, 2016) producing a graph with two disconnected components.



(b) A simplified sentence (Newsela, 2016) with an entire graph connected component removed.

Figure 2: Example of simplification that leads to removal of a connected component from the text segment graph. The graphs shown in this illustration are drastically reduced, as the complete graphs would be much too complex for visual analysis.

edness between two nodes is revealed by more paths between them, not just the shortest path, but the weights of paths are governed by a *length decay* parameter. Using this semantic relatedness measure, our hypothesis is that *the pairwise semantic relatedness between concepts that occur in the same sentence or paragraph tends to be higher in simpler texts (H5).*

4.3 Global Properties of the Graph

The last type of properties of the text segments that we compute over the knowledge graph, refers to the global properties of the graph which is formed by extracting all the paths, up to a certain length, that connect all the entities of the text segment. The resulting graph is called *text segment graph*. The intuition is that the way all the entities of the text segment are connected, including the connecting entities and relation paths, can provide insights into how complex the text is. These features aim to capture the *microstructure* of the conceptual complexity of text, as they analyze the graph produced by concepts that occur in the same sentence/paragraph.

They target similar transformations as the pairwise measures, as well as the removal of abstract concepts and text segments that are not essential for following the main story thread (examples #8, #9, and #10). The graphs that the removed concepts produce are usually disconnected from, or very weakly connected to, the concepts of the main story. Their removal thus results in a more tightly connected graph corresponding to the simplified text. This phenomenon is illustrated in Figures 2a and 2b, using the example #8 in Table 1. We assess the connectivity of the graph by extracting three features.

Number of connected components is the number of groups of interconnected nodes, such that paths exist between all pairs of nodes within the groups, but no paths exists between the nodes in different groups. Multiple connected components usually appear when the text segment deals with very distinct topics resulting in a higher cognitive load on the readers, since they must keep in memory all these disconnected topics simultaneously. For example, the original text in Figure 2a contains two components, and the simplification completely removes one component as seen in Figure 2b. This leads to our hypothesis that *texts whose sentences or paragraphs produce graphs that contain more connected components, are conceptually more complex (H6).*

Average local clustering coefficient computes the mean clustering coefficient of all the nodes in a graph. We compute this measure on the text segment graph. The intuition is similar to that behind the node's clustering coefficient. If the nodes of the text segment graph have high clustering coefficients, they might be quite specific, and hence uncommon. We hypothesize thus that *text segment graphs with low average clustering coefficient tend to be the result of conceptually simpler texts (H7).*

Density of a graph measures how many edges exist between the nodes, out of the total number of possible edges given the number of nodes. Since in knowledge graphs multiple relations can exist be-

tween the same two concepts, when we compute this measure, we do not count the number of effective relations, but the number of related pairs of concepts. The intuition behind using this measure is quite straightforward, since a denser text segment graph can be interpreted as a text in which all concepts are highly related, and accessible from multiple directions. Our hypothesis is therefore that *the denser the graph of a text segment, the simpler the text (H8).*

5 Experimental Setup

5.1 Data

For our experiments, we use the freely available English portion of Newsela language learners corpus (Newsela, 2016). Unlike the original Wikipedia and Simple English Wikipedia corpus (Coster and Kauchak, 2011), where the simplified version is not the result of a direct simplification of the original article, the simplified Newsela articles are direct simplifications of the original articles, manually simplified by trained human editors following strict simplification guidelines (Xu et al., 2016). This ensured the quality of simplifications in the Newsela dataset. More importantly, as the target readers are children and second language learners, and each text is provided in five different difficulty levels, conceptual simplification is expected to be an important component in the Newsela dataset. Additionally, the Newsela dataset allows us to control for the topic in our experiments, which is particularly important in order to isolate the effect of simplification from the influence of the story topics. Different topics naturally lead to different parts of the knowledge graph being activated, thus affecting graph properties and diluting the actual effect of conceptual simplification that we are trying to capture.

5.2 Implementation Details

Entity Linking. The first prerequisite for analyzing the knowledge graph behind text concepts is that the words in the text are linked to the concepts in the knowledge graph. Multiple systems capable of linking words to knowledge graphs, particularly DBpedia, have been researched and are freely available. In this work, we have chosen to use Kan-Dis (Hulpuş et al., 2015) because it links both common nouns and named entities. However, the ideas and measures that make the contribution of this paper do not depend on the linking process, so any linker can be used instead.

Outlier removal. Entity linking must often disambiguate the words, a process that can lead to errors in linking. Mistakes from the entity linking system are capable of affecting our measures, since properties of wrong concepts are then analyzed. To minimize the effects of linking mistakes, we remove from the linked entities those concepts that have a very weak relatedness to the other concepts, under the assumption that this will target particularly wrongly linked concepts. The threshold value chosen for deciding outliers was set empirically to 10^{-6}, computed with exclusivity based relatedness. This outlier removal strategy also makes the analysis less influenced by extreme cases, even if they are correct.

Text Segment Graph Extraction. For the extraction of the text segment graph, we focus on the paths that connect the linked concepts. For all pairs of linked concepts (by the entity linker), we extract all the paths that connect them in the knowledge graph, of length at most four. Most previous work that employs knowledge graphs have converged to the conclusion that a path length of at most four between concepts is sufficient for extracting valuable information about the concepts (Zhu and Iglesias, 2017). Additionally, for all the extracted nodes, we also include in the graph all the direct relations that exist between them.

6 Results

To test our hypotheses (H1–H8), we extract all proposed features on 200 original English Newsela articles and their four manually simplified versions, and examine:

- Whether the values of our 13 graph-based features differ significantly between each two versions of the same text, each corresponding to a different Newsela complexity level (Section 6.1);

- Whether our features can be used for automatically choosing the conceptually simpler of the two given versions of the same text (Section 6.2).

	1	2	3	4
0	+103%	+83%	**+238%**	**+326%**
1		+40%	+75%	+106%
2			+77%	+138%
3				+138%

(a) Page Rank

	1	2	3	4
0	+124%	+103%	**+327%**	**+431%**
1		+45%	**+86%**	+120%
2			+86%	+150%
3				+150%

(b) Seed Degree

	1	2	3	4
0	+5%	+8%	**+3%**	+2%
1		+2%	−3%	**−4%**
2			0%	0%
3				0%

(c) Seed Cl. Coef.

	1	2	3	4
0	**+1%**	**−1%**	**−3%**	**−4%**
1		**−1%**	**−3%**	**−4%**
2			**−1%**	**−2%**
3				−2%

(d) Pairwise Distance in Sent.

	1	2	3	4
0	**+18%**	**+39%**	**+78%**	**+100%**
1		**+34%**	**+182%**	**+156%**
2			**+148%**	**+129%**
3				+129%

(e) Pairwise SemRel in Sent.

	1	2	3	4
0	**−1%**	**−6%**	**−13%**	**−17%**
1		**−2%**	**−8%**	**−14%**
2			**−4%**	**−9%**
3				−9%

(f) Avg Local Cl. Coef. of Sent.

	1	2	3	4
0	**−1%**	**−2%**	**−4%**	**−6%**
1		**−1%**	**−3%**	**−6%**
2			**−2%**	**−4%**
3				**−4%**

(g) Pairwise Distance in Paragraphs

	1	2	3	4
0	**+15%**	**+38%**	**+80%**	**+122%**
1		**+28%**	**+62%**	**+106%**
2			**+35%**	**+68%**
3				**+68%**

(h) Pairwise SemRel in Paragraphs

	1	2	3	4
0	+4%	+2%	**−1%**	**−3%**
1		+2%	0%	**−3%**
2			+3%	**−1%**
3				−1%

(i) Avg Local Cl. Coef. of Paragraphs

	1	2	3	4
0	**+25%**	**+42%**	**+51%**	**+59%**
1		**+24%**	**+30%**	**+38%**
2			+13%	**+17%**
3				+17%

(j) Graph Density of Sent.

	1	2	3	4
0	**−4%**	**−6%**	**−12%**	**−15%**
1		**−2%**	**−8%**	**−11%**
2			**−6%**	**−9%**
3				**−9%**

(k) Graph ConnComp of Sent.

	1	2	3	4
0	**−2%**	**−3%**	**−6%**	**−6%**
1		0%	**−3%**	**−3%**
2			**−2%**	**−2%**
3				−2%

(l) Graph ConnComp of Paragraphs

	1	2	3	4
0	+43%	**+51%**	**+91%**	**+111%**
1		+37%	**+64%**	**+96%**
2			+37%	**+64%**
3				**+64%**

(m) Graph Density of Paragraphs

Table 2: The average change in the feature value (represented as a percentage of the initial value) for each pair of Newsela complexity levels (the original stories belong to the level 0, while the simplest versions belong to the level 4). The cases in which the mean value of the feature was significantly (at a 0.01 level of significance) different (Wilcoxon's signed rank test for paired samples in SPSS) in the two corresponding levels are presented in bold.

6.1 Pairwise Comparison of Texts on Different Complexity Levels

The goal of this analysis is to explore the discriminative power of our features to distinguish across different versions of the same text. For each two versions of the same story, we compute the average change in feature value (as a percentage of the feature value in initial/more complex text). The results are presented in Table 2, where each cell (m, n) corresponds to the change in feature value when simplifying from the complexity level m to the complexity level n. Those cases in which the mean value of the feature is significantly different (at a 0.01 level of significance, measured by Wilcoxon's test for repeated measures) in the corresponding two levels are presented in bold.

Overall results. We note that overall, the pairwise graph-based measures ((d), (e), (g), and (h)) are the most sensitive ones to the complexity level difference, while the single node measures ((a), (b), and (c)) have the lowest discriminative power across all level pairs. This indicates that the conceptual complexity

of the Newsela texts, when computed over DBpedia graph, resides especially in the way the concepts relate to one another (microstructure), rather than in the particular properties of the concepts taken one by one (macrostructure). As expected, the differences are significant especially between the far away levels such as 0-3, 0-4, 1-3 and 1-4, while many measures have a problem to discriminate between the nearby levels, particularly the levels 3-4.[6]

Single node measures. Among the single node measures, the hypothesis H1 and H3 have been proven correct for the distant level pairs, showing the biggest discriminative power for the seed degree (H1) followed by the PageRank (H3). Nevertheless, none of the measures can capture the subtle differences between the nearby levels. This is to be expected since during the simplification of texts the omission or replacement of concepts are measures to be taken very cautiously as the original meaning of the texts should not be noticeably altered. A too abrupt change in this set of measures would rather be indicative of flawed simplification. Regarding the clustering coefficient of the seeds (H2), no trend can be established.

Pairwise graph-based measures. The pairwise graph-based measures perform similarly on both sentences and paragraphs, and are good at distinguishing between any two text complexity levels. Both hypotheses H4 and H5 have been proven to hold.

Global graph-based measures. Regarding the global graph-based measures, we notice that in general, all three measures perform better when computed over sentences than over paragraphs. Computed over sentences, these measures are able to discriminate between the nearby levels, while computed over paragraphs, this does not always hold. This might be due to the fact that, in general, the grouping of the concepts over sentences changes during simplification, but their grouping over paragraphs does not. The global graph-based measure that shows the best discriminative power between each two text complexity levels is the number of connected components of the sentence graph, followed closely by the average local clustering coefficient of the sentence graph. Our hypotheses related to global-graph based measures (H6, H7, and H8) have been proven to hold, except H7 (concerned with the average clustering coefficient) over paragraphs, which only holds for the faraway complexity levels.

This analysis has allowed us to draw conclusions as to how sensitive the proposed measures are to the complexity level of texts. Given the very promising results, in the next section we analyze how these measures perform in a pairwise text classification task.

6.2 Classification Experiments

We perform the pairwise comparison of each two documents corresponding to the same story to assign the label "simpler" if the first document is simpler than the second, and "more complex" if otherwise.

Features. For this binary classification task, each instance has 26 features, obtained by concatenating 13 features of the second text to the 13 features of the first text. Additionally, for each document, we compute a baseline feature as the ratio of unique entities over mentions. On purpose, we avoid using any features which are influenced by the length of the text segment (sentence, paragraph, or entire text). We do so as we are only interested in discriminating texts according to their conceptual complexity, and features that reflect syntactic complexity (i.e. any features that include text segment length) could blur the results.

Dataset. We construct the dataset by pairing texts with the same title and different Newsela complexity level. As we have 200 titles in our collection, that results in a total of 2000 instances (ten for each title, i.e. levels 0-1, 0-2, 0-3, 0-4, 1-2, 1-3, 1-4, 2-3, 2-4, 3-4). To have balanced classes, we randomize the order of the two texts in each pair. This resulted in having 1025 instances in which the first text is simpler than the second, and 975 instances in which the first text is more complex than the second (according to their Newsela complexity levels).

[6]The original stories belong to the level 0, while the simplest versions belong to the level 4.

Algorithm	Feature type (our graph-based features)								Baseline feature (#Entities/#Mentions)	
	All		Single		Pairwise		Global			
	F	Acc.	F	Acc.	F	Acc.	F	Acc.	F	Acc.
NaïveBayes	0.80	80.42	0.57	58.38	0.75	75.67	0.72	72.14	0.61	62.11
Logistic	0.84	84.13	0.56	56.49	0.77	77.43	0.77	77.15	0.60	60.00
SVM-n	0.84	83.73	0.46	53.74	0.77	76.82	0.77	77.40	0.61	61.43
SVM-s	0.84	84.27	0.52	55.19	0.77	77.25	0.77	77.23	0.60	60.53
JRip	0.77	77.44	0.57	57.70	0.76	75.60	0.72	72.09	0.58	58.50
J48 (C4.5)	0.77	77.20	0.51	55.90	0.75	75.52	0.73	73.47	0.57	60.45
RandomForest	**0.85**	**85.00**	**0.70**	**69.85**	**0.81**	**81.11**	**0.79**	**78.95**	**0.64**	**64.36**
Majority class	0.34	50.96	0.34	50.96	0.34	50.96	0.34	50.96	0.34	50.96

Table 3: The weighted F-measure (F) and accuracy (Acc.) for the binary classification task (10-fold cross-validation with 10 repetitions). The best results for each set of features are presented in bold. All classifiers significantly (paired t-test) outperform the majority class baseline.

Algorithms. Our goal is to demonstrate that our feature set has good predictive power in such tasks, and not to build the highest performing classification system. Therefore, we experiment with a range of classification algorithms, and use them in their default parameter setup (without any tuning). We use a ten-fold cross-validation setup with ten repetitions in Weka Experimenter (Hall et al., 2009) to train and test seven classification algorithms: Naïve Bayes (John and Langley, 1995), Logistic (le Cessie and van Houwelingen, 1992), Support Vector Machines (Keerthi et al., 2001) with normalisation (SVM-n) or standardisation (SVM-s), JRip rule learner (Cohen, 1995), J48 - a Weka implementation of the C4.5 decision tree (Quinlan, 1993), and Random Forest (Breiman, 2001).

Results. The results of this set of experiments are presented in Table 3. As can be seen, even without any parameter tuning or feature selection, our set of 13 graph-based conceptual complexity features lead to high performances, both in terms of the weighted F-measure and the accuracy, always significantly (paired t-test at a 0.01 level of significance) outperforming the baseline feature. We also observe that the single node features (Single) perform significantly (paired t-test at 0.01 level of significance) worse than the full set of features, or even any of the other two subsets of features (Pairwise and Global).

7 Conclusion

Text complexity plays an important role in text accessibility. Apart from the usually addressed lexical and syntactic complexity, recent studies emphasize the importance of conceptual complexity, both on micro- and macrostructural levels. Building up on those findings, we set up a new task of automatic assessment of conceptual complexity.

We proposed to measure conceptual complexity by leveraging the structure of knowledge graphs such as DBpedia. We found that properties of individual concepts, as well as the way concepts relate to each other in the graph, successfully capture the conceptual complexity of text. Furthermore, our experiments show that our graph-based measures perform well when used as features in a binary classification task for automatically choosing the simpler of two versions of the same text.

Our research sets a new field and benchmarks the task of automatic assessment of conceptual text complexity. It also shows that knowledge graphs can successfully be used for this task, thus combining recent advances across multiple research areas.

In our future work, we plan to analyze the impact of the entity linking step over the proposed measures, as well as to extend our analysis to other knowledge graphs such as ConceptNet[7] and WikiData[8]. We also plan to explore additional measures for computing conceptual complexity, as well as measures for scoring the relevance of background knowledge facts with respect to making a given text more or less complex. This would naturally lead to methods for extraction of knowledge base facts that would help users understand complex texts.

[7] http://conceptnet.io/

[8] https://www.wikidata.org/wiki/Wikidata:Main_Page

References

Sandra Maria Aluísio and Caroline Gasperin. 2010. Fostering digital inclusion and accessibility: The porsimples project for simplification of portuguese texts. In *Proceedings of YIWCALA Workshop at NAACL HLT 2010*, pages 46–53.

Bharat Ram Ambati, Siva Reddy, and Mark Steedman. 2016. Assessing Relative Sentence Complexity using an Incremental CCG Parser. In *Proceedings of NAACL-HLT*, pages 1051–1057.

Barbara Arfé, Lucia Mason, and Inmaculada Fajardo. 2017. Simplifying informational text structure for struggling readers. *Reading and Writing*, Oct.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sren Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. Dbpedia - a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154 – 165. The Web of Data.

Javier Borge-Holthoefer and Alex Arenas. 2010. Semantic networks: Structure and dynamics. *Entropy*, 12(5):1264–1302.

Leo Breiman. 2001. Random Forests. *Machine Learning*, 45(1):5–32.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International Conference on World Wide Web 7*, WWW7, pages 107–117, Amsterdam, The Netherlands, The Netherlands. Elsevier Science Publishers B. V.

John Carroll, Guido Minnen, Yvonne Canning, Siobhan Devlin, and John Tait. 1998. Practical Simplification of English Newspaper Text to Assist Aphasic Readers. In *Proceedings of AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, pages 7–10.

William W. Cohen. 1995. Fast Effective Rule Induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123.

Allan M. Collins and M. Ross Quillian. 1969. Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior*, 8(2):240 – 247.

William Coster and David Kauchak. 2011. Simple English Wikipedia: a new text simplification task. In *Proceedings of ACL&HLT*, pages 665–669.

Carolyn A. Denton, Mischa Enos, Mary J. York, David J. Francis, Marcia A. Barnes, Paulina A. Kulesz, Jack M. Fletcher, and Suzanne Carter. 2015. Text-processing differences in adolescent adequate and poor comprehenders reading accessible and challenging narrative and informational text. *Reading Research Quarterly*, 50(4):393–416.

Geert Freyhoff, Gerhard Hess, Linda Kerr, Bror Tronbacke, and Kathy Van Der Veken, 1998. *Make it Simple, European Guidelines for the Production of Easy-toRead Information for People with Learning Disability*. ILSMH European Association, Brussels.

Thomas L. Griffiths, Mark Steyvers, and Alana Firl. 2007. Google and the mind: Predicting fluency with pagerank. *Psychological Science*, 18(12):1069–1076. PMID: 18031414.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11:10–18.

Ioana Hulpuş, Conor Hayes, Marcel Karnstedt, and Derek Greene. 2013. Unsupervised graph-based topic labelling using dbpedia. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 465–474, New York, NY, USA. ACM.

Ioana Hulpuş, Narumol Prangnawarat, and Conor Hayes. 2015. Path-based semantic relatedness on linked data and its use to word and entity disambiguation. In *The Semantic Web - ISWC 2015*, pages 442–457, Cham. Springer International Publishing.

Mario Jarmasz and Stan Szpakowicz. 2012. Roget's thesaurus and semantic similarity. *CoRR*, abs/1204.0245.

George. H. John and Pat Langley. 1995. Estimating Continuous Distributions in Bayesian Classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345.

Joyce Karreman, Thea van der Geest, and Esmee Buursink. 2007. Accessible website content guidelines for users with intellectual disabilities. *Journal of Applied Research in Intellectual Disabilities*, 20:510–518.

Sathiya. S. Keerthi, Shirish. K. Shevade, Chiranjib Bhattacharyya, and Krishna R. K. Murthy. 2001. Improvements to Platt's SMO Algorithm for SVM Classifier Design. *Neural Computation*, 13(3):637–649.

Walter Kintsch and Teun A. van Dijk. 1978. Towards a model of text comprehension and production. *Psychological Review*, 85:363–394.

Saskia le Cessie and Johannes C. van Houwelingen. 1992. Ridge Estimators in Logistic Regression. *Applied Statistics*, 41(1):191–201.

Danielle S. McNamara, Arthur Graesser, and Max Louwerse, 2012. *Sources of text difficulty: Across the ages and genres*. Lanham, MD: Rowman & Littlefield Education.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November.

David Milne and Ian H. Witten. 2008. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *In Proceedings of AAAI 2008*.

Shashi Narayan and Claire Gardent. 2014. Hybrid simplification using deep semantics and machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 435–445.

Douglas L. Nelson, Cathy L. McEvoy, and Thomas A. Schreiber. 2004. The university of south florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers*, 36(3):402–407, Aug.

Newsela. 2016. Newsela article corpus. `https://newsela.com/data`. Version: 2016-01-29.

Constantin Orasan, Richard Evans, and Iustin Dornescu, 2013. *Towards Multilingual Europe 2020: A Romanian Perspective*, chapter Text Simplification for People with Autistic Spectrum Disorders, pages 287–312. Romanian Academy Publishing House, Bucharest.

Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA.

Horacio Saggion, Elena Gómez Martínez, Alberto Anula, Lorena Bourg, and Esteban Etayo. 2011. Text Simplification in Simplext: Making Text More Accessible. *Revista de la Sociedad Española para el Procesamiento del Lenguaje Natural*, 47:341–342.

Michael Schuhmacher and Simone Paolo Ponzetto. 2014. Knowledge-based graph document modeling. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 543–552, New York, NY, USA. ACM.

Mark Steyvers and Joshua B. Tenenbaum. 2005. The large scale structure of semantic networks: Statistical analyses and a model of semantic growth. *Cognitive Science*, 29(1):41–78.

Michael Stubbs. 1996. *Text and Corpus Analysis: Computer Assisted Studies of Language and Culture*.

Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Michael Röder, Daniel Gerber, Sandro Athaide Coelho, Sören Auer, and Andreas Both. 2014. Agdistis - graph-based disambiguation of named entities using linked data. In Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig Knoblock, Denny Vrandečić, Paul Groth, Natasha Noy, Krzysztof Janowicz, and Carole Goble, editors, *The Semantic Web – ISWC 2014*, pages 457–471, Cham. Springer International Publishing.

Sowmya Vajjala and Detmar Meurers. 2014. Assessing the relative reading level of sentence pairs for text simplification. In *Proceedings of the EACL 2014*, pages 288–297.

Sowmya Vajjala and Detmar Meurers. 2015. Readability-based Sentence Ranking for Evaluating Text Simplification. Unpublished technical report, arXiv:1603.06009 [cs.CL].

Sanja Štajner and Goran Glavaš. 2017. Leveraging event-based semantics for automated text simplification. *Expert Systems With Applications, Elsevier*, 82:383–395.

Sanja Štajner, Richard Evans, and Iustin Dornescu. 2014. Assessing conformance of manually simplified corpora with user requirements: the case of autistic readers. In *Proceedings of the Workshop on Automatic Text Simplification - Methods and Applications in the Multilingual Society (ATS-MA 2014)*, pages 53–63, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Ganggao Zhu and Carlos. A. Iglesias. 2017. Computing semantic similarity of concepts in knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):72–85, Jan.

Par4Sim – Adaptive Paraphrasing for Text Simplification

Seid Muhie Yimam **Chris Biemann**
Language Technology Group
Department of Informatics, MIN Faculty
Universität Hamburg, Germany
{yimam,biemann}@informatik.uni-hamburg.de

Abstract

Learning from a real-world data stream and continuously updating the model without explicit supervision is a new challenge for NLP applications with machine learning components. In this work, we have developed an adaptive learning system for text simplification, which improves the underlying learning-to-rank model from usage data, i.e. how users have employed the system for the task of simplification. Our experimental result shows that, over a period of time, the performance of the embedded paraphrase ranking model increases steadily improving from a score of 62.88% up to 75.70% based on the NDCG@10 evaluation metrics. To our knowledge, this is the first study where an NLP component is adaptively improved through usage.

Title and Abstract in Amharic

Par4Sim -- **የሚጣጣም መልሶ መጻፍ፤ ፅሁፍን ለማቅለል**

ከገሃዱ ዓለም የመረጃ ፍሰት መማርና ቀጥተኛ ድጋፍ ሳያስፈልገው ሞዴሉን በተከታታይ ለማሻሻል ማሽን ለርኒንግ (በራሱ መማር የሚችል የኮምፒዩተር መርሃ ግብር) በሚያካትቱ የተፈጥሯዊ የቋንቋ ቴክኖሎጂ (ተ.ቋ.ቴ - NLP) ጥናት አዲስ ፈተና ሆኗል። በዚህ ምርምር፤ ተጠቃሚዎች እንድን መተግበሪያ በመጠቀም ላይ ሳሉ ማለትም በተጠቃሚዎች የተለያዩ ድርጊቶች መሰረት ውህቦችን ከበስተጀርባ ሆኖ በመሰብሰብ የማሽን ለርኒንጉን ሞዴል በማዘመን እየተጣጣመ መማር የሚችል የጽሁፍ አቅላይ ስርዓት ሰርተናል። የሲስተማችንን ውጤታማነት ለማረጋገጥ፤ አንድን ፅሁፍ ለተለያዩ ተጠቃሚዎች በቀላሉ እንዲገባቸው ለማድረግ ተጣጣሚ ትርጓሜዎችን እንደ ምሳሌ ወስደናል። በመሆኑም፤ አንድ ተጠቃሚ አንድን ፅሁፍ መረዳት ሲያቅተው፤ የኛን ተንታኝ መተግበሪያ ተጠቅሞ አማራጭ ቃል ወይንም ሐረግ በመተካት ፅሁፍን የበለጠ ቀላል ማድረግ ይችላል።

የሙከራ ውጤታችን እንደሚያሳየው ከሆነ፤ በተከታታይ ጊዜ ውስጥ እንዲማር/እንዲላመድ በሲስተሙ ውስጥ የተሸነጠው ደረጃ መዳቢ ሞዴል፤ በNDCG@10 የግምገማ ልኬቶች መሰረት፣ ሳያቋርጥ በመጨመር ከ62.88% እስከ 75.70% ድረስ መሻሻል አሳይቷል። እኛ እስከምናውቀው ድረስ፤ ይህ ጥናት፤ የተፈጥሯዊ የቋንቋ ቴክኖሎጂ (ተ.ቋ.ቴ - NLP) ትግበራዎች በአግልግሎት አጠቃቀም ጊዜ ውህቦችን በመሰብሰብ የማሽን ለርኒንግ ስልተ ቀመር ሞዴል አገልግሎት በሂደት ውስጥ እያለ እንዲሻሻል/እንዲላመድ የሚያደርግ የመጀመሪያው ጥናት ነው።

1 Introduction

Traditionally, training data for machine learning-based applications are collected amass, using a specific annotation tool. There are a number of issues regarding this approach of data collection. Most importantly, if the behavior of the target application changes over time, this makes the training data outdated and obsolete, an issue known as *concept drift* (Žliobaitė et al., 2016).

In this regard, we opt to design an approach where data can be collected interactively, iteratively and continuously using an adaptive learning model in a live and a real-world application. By adaptive, we mean that the learning model gets signal from the usage data over a period of time and it automatically

Proceedings of the 27th International Conference on Computational Linguistics, pages 331–342
Santa Fe, New Mexico, USA, August 20-26, 2018.

adapts to the need of the application. An adaptive learning model has a spectrum of advantages. First of all, the model gets updated continuously. The model also provides suggestions through usage and the user can correct the suggestions, which in turn improves the model's performance. On top of this, there is no need to collect a large set of training examples a priori, which might be difficult and expensive. We also believe that instead of collecting training data in advance, it is a more natural way to get the training examples from usage data by embedding the adaptive model in the application.

In this premise, we choose the advanced NLP task of text simplification in order to experiment with an adaptive learning approach. Text simplification aims at reducing the complexity (syntactic and lexical) of a text for a given target reader. According to the survey by Shardlow (2014), different approaches such as lexical simplification, explanation generation, and machine translation are used for text simplification. As far as our knowledge is concerned, there is no work regarding the use of an adaptive model, more specifically *adaptive paraphrasing based on usage data* for the task of text simplification.

The main technical challenge is the integration of an adaptive paraphrasing model into a text simplification writing aid tool in order to be able to attain whether it is possible to gain NLP component quality through adaptive learning on usage data. The writing aid tool for text simplification using an adaptive paraphrasing model (Par4Sim) consists of several components. The first component in the pipeline determines the complex or difficult words or phrases (CPs), which is based on the work of Yimam et al. (2017). Once the CPs are identified, the second component produces candidate suggestions from different paraphrase resources. Candidates that do not fit the context in the sentence are filtered or excluded based on a language model score. Finally, an adaptive ranking model reorders candidates based on their simplicity and provides the ranked candidates to the user within an interactive writing aid tool.

This work is aiming to answer the following three research questions:
RQ1: How can an adaptive paraphrase ranking model be integrated into a text simplification writing aid tool?
RQ2: How can an adaptive paraphrase ranking model be evaluated?
RQ3: Can we demonstrate the adaptivity of the approach?

This paper is organized as follows. In Section 2 we briefly review state-of-the-art works in adaptive learning and text simplification. Section 3 outlines the design procedures we have followed in the development of the Par4Sim tool for the integration of adaptive paraphrasing in a text simplification application in detail. In Section 4, a brief description of the data collection and statistics of the collected data is presented. Section 5 describes the learning-to-rank machine learning algorithm employed to build an adaptive ranking model, including the definition of employed feature representation and the evaluation metrics. The experimental results obtained and the contributions of our research work are presented in Section 6 and Section 7. Finally, Section 8 presents the conclusions of our work and indicates future directions on the integration of adaptive approaches for NLP applications.

2 Related Work

Supervised machine learning approaches commonly rely on the existence of a fixed training set, which is collected in advance.

The survey by Parisi et al. (2018) indicates that *continual lifelong learning*, the ability to learn continuously by acquiring new knowledge is one of the challenges of modern computational models. The survey further explains that one of the main problems of continual learning is that training model with new information interferes with previously learned knowledge. The work of To et al. (2009) indicates that the integration of user feedback in an adaptive machine learning approach helps to improve the system's performance. It further shows that the approach helps to reduce the manual annotation efforts. Žliobaitė et al. (2016) revealed that supervised machine learning approaches stationed with static datasets face problems during deployment in a real-world application due to concept drift (Tsymbal, 2004) as applications start generating data streams continually. Applications with such properties include spam filtering and intrusion detection.

Stream-based learning and online learning (Bottou, 1998) are alternative setups to a batch-mode adap-

tive learning system. For example, the work by Levenberg et al. (2010) shows that the deployment of stream-based learning for statistical machine translation improves the performance of their system when new sentence pairs are incorporated from a stream. The work by Wang et al. (2015) presents SOLAR, a framework of scalable online learning algorithms for ranking, to tackle the poor scalability problem of batch and offline learning models. The work by Yimam et al. (2016a) uses MIRA (Margin Infused Relaxed Algorithm) (Crammer and Singer, 2003), a perceptron-based online learning algorithm to generate suggestions in an iterative and interactive annotation setup for a biomedical entity annotation task.

Most text simplification approaches employ basic machine translation models from parallel corpora (Xu et al., 2016; Štajner et al., 2017) and using simple Wikipedia for English (Coster and Kauchak, 2011). Simple PPDB (Pavlick and Callison-Burch, 2016) is such a resource that is built automatically, using machine translation techniques on a large number of parallel corpora.

The work by Lasecki et al. (2015) shows that using crowdsourcing for text simplification is a valid approach. We also conducted our adaptive text simplification experiment on the Amazon Mechanical Turk crowdsourcing platform using a specialized text simplification tool.

3 Design of the Par4Sim System

Par4Sim is a text simplification tool based on an adaptive paraphrasing paradigm. Unlike the traditional text simplification approaches, Par4Sim mimics a normal text editor (writing aid tool). The tool embeds basic text simplification functionalities such as providing suggestions for complex phrases, allowing editing of the texts in place and so on. Our setup is that authors wishing to simplify text can use the tool, without a priori machine learning model, but the system learns from the user interactions in an iterative and interactive manner. In order to run a distributed and web-scale simplification experiment, we have integrated the tool into an existing crowdsourcing platform.

Hence, our targeted users for the Par4Sim experiments are workers from the Amazon Mechanical Turk (MTurk) crowdsourcing platform. As a large number of workers are available in the MTurk platform, we paid special attention in the development of the tool regarding response time, accessibility, and reliability. The tool comprises a front-end component to edit texts and a back-end component, which exposes most of the requests using REST API services. Figure 1 shows the main components of the tool. The detailed instructions for the MTurk task are displayed in Figure 2.

While it is impossible to host complex systems inside the MTurk infrastructure, MTurk supports external human intelligence tasks (HITs) where workers can easily access our system through the MTurk interface. Par4Sim's user interface is embedded into the MTurk web page, but every activity is handled by our own server. This design gives us full control on the collection of the dataset, for training new ranker models, and for updating the model iteratively and seamlessly while we still use the MTurk infrastructure to recruit workers and pay rewards for a web-scale experiment.

As it can be seen in Figure 1, complex words or phrases (CPs) are automatically highlighted (yellow background color and underlined in cyan color). The highlighting of the CPs is based on the work of Yimam et al. (2017). Furthermore, the users can highlight their own CPs and our system will provide ranked candidates. Details about the generation of candidates and the ranking model are provided in Section 4. Besides the highlighting of CPs and providing ranked candidates, the system further provides the following functionalities.

- **Reload text**: If the worker wants to get the original text with the CPs re-highlighted, she can reload the content using the *Reload* button, subject to confirmation.
- **Undo and redo**: At any particular time, workers can undo or redo the changes they have made using the *Undo* and *Redo* buttons.
- **Highlight difficult words**: It is also possible to request our system to automatically highlight difficult words. This functionality is particularly important if the worker has changed the original text but she is not sure if the amended text is in fact simpler. Once the system highlights some words or phrases, she can still check if the suggestions provided by the system could still simplify the text further.

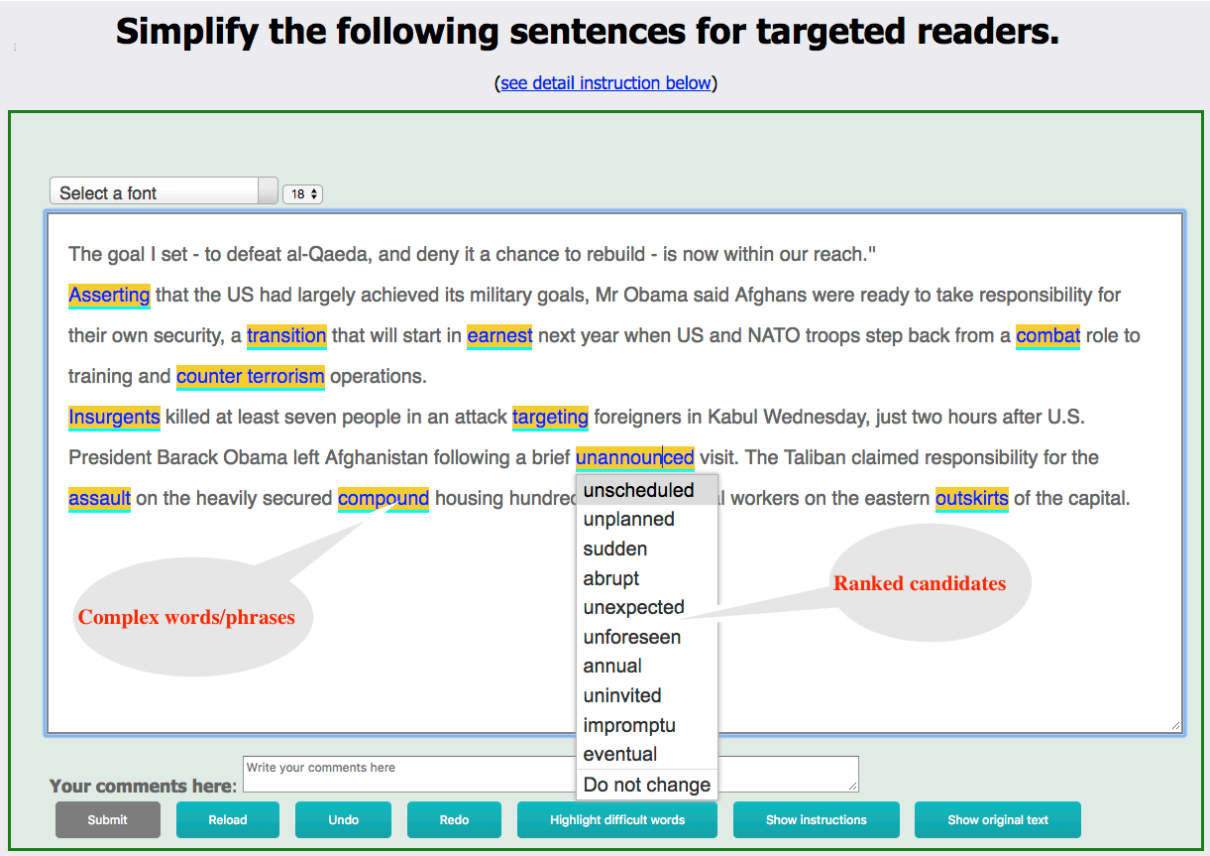


Figure 1: The Par4Sim UI as it is displayed inside the MTurk webpage with the instruction *"Simplify the following sentences for targeted readers"*. The *targeted readers* are explained in the detailed instruction as *children*, *language learner*, and *people with reading impairments*.

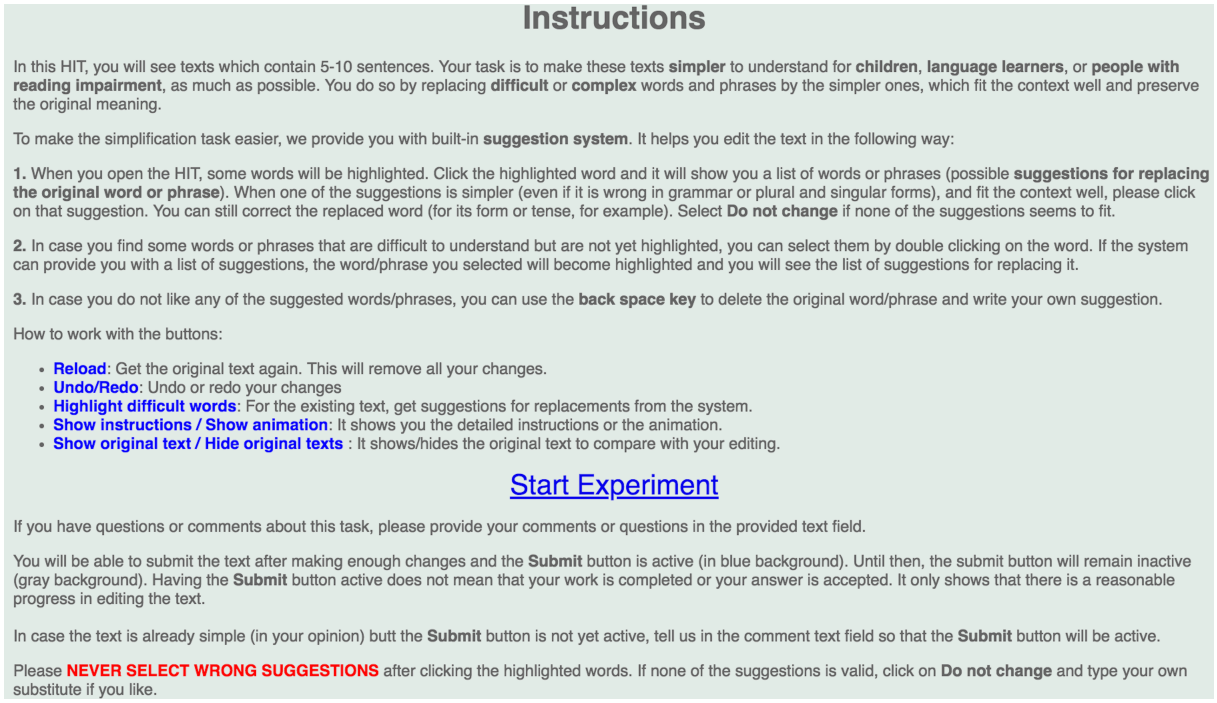
- **Show instruction / Original texts**: The instructions (see Figure 2) are visible at the beginning of each task. Once workers have accepted the task, the instructions will be hidden automatically so that the workers can focus on the simplification task in a clean window. During simplifying the text, the worker can display the instructions below the text editor if she wants to refer to the instructions. Workers can also compare the simplified version of the text with the original text.

- **Show animation**: Crowdworkers prefer a very short task description or the task should be easy enough to be understood by most workers. Since the simplification task is difficult, we decided to include an animated video showing important steps in the text simplification process. The video animation starts as soon as the task description is displayed and is hidden once the task is started. The worker can refer to the animation while completing the task.

With these functionalities, we provided a text simplification aid that comes very close to how a simplification application would look like. The goal was to provide a realistic environment in order to test the adaptive approach within a user-centric scenario. While we embedded the application into MTurk in order to attract paid users, it is straightforward to provide this web-based application online or locally.

4 Task Description and Dataset

In this paper, we address text simplification, which is the task to simplify a given text that is assumed to be difficult to understand for particular readers such as language learners, children or people with reading impairments. A text simplification pipeline usually starts with the complex word or phrase identification.



Instructions

In this HIT, you will see texts which contain 5-10 sentences. Your task is to make these texts **simpler** to understand for **children**, **language learners**, or **people with reading impairment**, as much as possible. You do so by replacing **difficult** or **complex** words and phrases by the simpler ones, which fit the context well and preserve the original meaning.

To make the simplification task easier, we provide you with built-in **suggestion system**. It helps you edit the text in the following way:

1. When you open the HIT, some words will be highlighted. Click the highlighted word and it will show you a list of words or phrases (possible **suggestions for replacing the original word or phrase**). When one of the suggestions is simpler (even if it is wrong in grammar or plural and singular forms), and fit the context well, please click on that suggestion. You can still correct the replaced word (for its form or tense, for example). Select **Do not change** if none of the suggestions seems to fit.

2. In case you find some words or phrases that are difficult to understand but are not yet highlighted, you can select them by double clicking on the word. If the system can provide you with a list of suggestions, the word/phrase you selected will become highlighted and you will see the list of suggestions for replacing it.

3. In case you do not like any of the suggested words/phrases, you can use the **back space key** to delete the original word/phrase and write your own suggestion.

How to work with the buttons:

- **Reload**: Get the original text again. This will remove all your changes.
- **Undo/Redo**: Undo or redo your changes
- **Highlight difficult words**: For the existing text, get suggestions for replacements from the system.
- **Show instructions / Show animation**: It shows you the detailed instructions or the animation.
- **Show original text / Hide original texts** : It shows/hides the original text to compare with your editing.

Start Experiment

If you have questions or comments about this task, please provide your comments or questions in the provided text field.

You will be able to submit the text after making enough changes and the **Submit** button is active (in blue background). Until then, the submit button will remain inactive (gray background). Having the **Submit** button active does not mean that your work is completed or your answer is accepted. It only shows that there is a reasonable progress in editing the text.

In case the text is already simple (in your opinion) butt the **Submit** button is not yet active, tell us in the comment text field so that the **Submit** button will be active.

Please **NEVER SELECT WRONG SUGGESTIONS** after clicking the highlighted words. If none of the suggestions is valid, click on **Do not change** and type your own substitute if you like.

Figure 2: The detailed instructions of Par4Sim, which are displayed at the beginning of the task and hidden afterwards. The worker can display the instruction when required.

For this experiment, we have used parts of the dataset from Yimam et al. (2017), which already contains manually identified complex phrases (CP). In this dataset, the complex phrases are manually identified by 10 native and 10 non-native English speakers. The dataset has been already used for the complex word identification (CWI) shared task 2018[1]. Please refer Yimam et al. (2017) for the details of the dataset. We purposely used the manually identified and verified CPs because 1) we do not want to mix the identification and the simplification tasks, and 2) we want to test the adaptive learning approach in a controlled setting.

We have generated candidate suggestions from different paraphrase resources. The following resources are used to generate candidate suggestions:

- **Lexical and Distributional resources**: We use WordNet (Miller, 1995) and distributional thesaurus (Biemann et al., 2013) to produce candidate results for CPs. We apply lemmatization to reduce the CPs into their base forms and retrieve the top 10 synonyms respectively the top 10 similar words from the lexical resources.

- **PPDB 2.0 and Simple PPDB**: PPDB (Pavlick et al., 2015) is the largest paraphrase resource to date. The recently released simple PPDB (Pavlick and Callison-Burch, 2016) is a particularly relevant resource for the task of text simplification. For each CPs (source entry in PPDB), we retrieve the top 10 candidates (target entry in PPDB).

- **Phrase2Vec**: We have trained a Phrase2Vec model (Mikolov et al., 2013) using English Wikipedia and the AQUAINT corpus of English news text (Graff, 2002). Mikolov et al. (2013) pointed out that it is possible to extend the word-based model to a phrase-based model using a data-driven approach where each phrase or multi-word expressions are considered as individual tokens during the training process. We have used a total of 79,349 multiword expression and phrase resources, which are obtained from the work of Yimam et al. (2016b). We trained the Phrase2Vec embeddings with 200

[1] https://sites.google.com/view/cwisharedtask2018/ (Yimam et al., 2018)

dimensions using skip-gram training and a window size of 5. We retrieve the top 10 similar words to the CPs as candidates.

Obviously, the number of candidate suggestions obtained from these different resources is enormous and we should limit the size before providing to the ranker model. The candidates are ordered by a language model score. We trained a tri-gram language-model (Pauls and Klein, 2011) using the Wikipedia articles. The number of candidates is limited to 10; these are re-ranked using the learning-to-rank model.

For each HIT, we provide between 5 and 10 sentences for simplification. A HIT is then assigned to 10 workers as we need a graded relevance to train the learning-to-rank model (see Section 5). In the experiment, we make sure that a HIT is submitted only in one iteration and most importantly, during the evaluation of the ranking model performance, we make sure that the training data from previous iterations should not contain HITs from the current iteration.

In this experiment, a total of 18,036 training instances have been collected. Figure 3 shows how the training instances collected from the usage data looks like. The number at the end of the simplified sentence shows the number of workers provided the same simplified sentence.

Complex Sentence: *Hajar said his cousin was not* ***affiliated*** *with any terrorist group.*
Simplified Sentence 1: *Hajar said his cousin was not* ***associated*** *with any terrorist group.* → *6*
Simplified Sentence 2: *Hajar said his cousin was not* ***merged*** *with any terrorist group.* → *2*
Simplified Sentence 3: *Hajar said his cousin was not* ***aligned*** *with any terrorist group.* → *1*
Simplified Sentence 4: *Hajar said his cousin was not* ***partnered*** *with any terrorist group.* → *1*

Figure 3: Examples of usage data as training instances. Here affiliated is a CP and associated, merged, aligned, and partnered are the simpler options provided by 6, 2, 1, and 1 workers respectively.

More detailed statistics are shown in Table 1. From Table 1, we see that around 70% of the workers (mainly from India and the US) have successfully completed the task.

	#workers	#visitors
instances	18036	10758
workers	164	71
countries	11	3

Table 1: Statistics of workers and simplification instances collected during all 9 iterations in the experiment. The column #workers shows the number of workers who have accepted and submitted the result while the column #visitors shows the number of workers who did not submit their results.

5 Learning-to-Rank

Learning-to-rank refers to a machine learning technique for training a model based on existing labels or user feedback for ranking task in areas like information retrieval, natural language processing, and data mining (Li, 2014). Learning-to-rank consists of a learning and ranking system. The system is trained by providing pairs of requests/queries and a target/ideal ranking for retrieved items. The learning model then constructs a ranking model on the basis of the training data ranking lists.

Ranklib[2], a well-known learning-to-rank library in Java from the Lemur Project is used to build the ranking models. Specifically, we have used the LambdaMART algorithm to train our learning and ranking models. LambdaMART combines LambdaRank and MART (Multiple Additive Regression Trees) (Burges, 2010; Donmez et al., 2009). While MART uses gradient boosted decision trees for prediction tasks, LambdaMART uses gradient boosted decision trees using a cost function based on NDCG for solving a ranking task.

[2] https://sourceforge.net/p/lemur/wiki/RankLib/

Normalized Discounted Cumulative Gain (NDCG) (Järvelin and Kekäläinen, 2002; Wang et al., 2013) is a family of ranking measures such as mean average precision (MAP) and Precision at K. NDCG is well-suited for our experiment for its capability of incorporating graded judgments. The graded judgments are obtained from the number of workers suggesting the candidate for the given CP target.

5.1 Features

To train the learning-to-rank model, we have designed a set of features that are important for text simplification. Based on the works of (Yimam et al., 2017; Horn et al., 2014), we have implemented the following list of features for the ranking model, which are partially derived from the candidate generating resources.

- **Frequency and length**: Due to the common use of these features in selecting the most simple lexical substitution candidate (Bott et al., 2012), we use three length features specifically the number of vowels, syllables, and characters and three frequency features: the frequency of the word in Simple Wikipedia, the frequency of the word in the document, and the frequency of the word in the Google Web 1T 5-Grams.

- **Lexical and distributional thesaurus resources**: We also use the number of similar words to the CPs and candidate suggestion based on lexical resources such as WordNet and distributional thesaurus as possible features. The features are normalized and scaled using the *featran's*[3] min-max scaler tool.

- **PPDB 2.0 and simple PPDB**: From the PPDB 2.0 and simple PPDB resources (cf. Sect. 4), we use associated scores as given by the resource, i.e.: *ppdb2score*, *ppdb1score*, *paraphraseScore*, and *simplificationScore*.

- **Word embeddings feature**: We use the Phrase2Vec embeddings as described in Section 4 to obtain vector representations for targets (CPs) and candidates. The cosine similarity of the candidates with the whole sentences as well as the cosine similarity of the candidates with the tri-gram words (one word to the left and one word to the right of the target CP) are used as features. The vector representations of the sentences and the tri-grams are the average of the individual vector representations of words in the sentences and the tri-grams.

6 Experiments

6.1 Baseline system

Our baseline system is built using a general purpose paraphrasing dataset from Yimam et al. (2016b). The dataset is based on essay sentences from the ANC[4] and BAWE corpora (Alsop and Nesi, 2009). We use the same feature extraction approach (see Section 5) for the development of the baseline model.

As it can be seen in Table 2, the results from each iteration are compared with the baseline system. We noted that the generic paraphrase datasets do not quite fit the task of text simplification as the need of the task is different. The lower performance on the baseline system can be attributed to the fact that the texts for the baseline system are collected from a different genre (essay sentences). We have to make clear that the first and all the subsequent iterations in the adaptive process do not use the baseline system.

6.2 Adaptive systems

We start with an empty ranking model (iteration 1), where candidates obtained from the resources are provided to the workers without an implied ranking. After collecting enough usage data from iteration 1, we trained a ranking model, which is used to re-rank candidates for the texts in the next iteration (iteration 2). Texts in iteration 2, which are exclusively different from those in iteration 1 are provided to the workers. Once workers completed the simplification task at iteration 2, we re-evaluate the ranking of candidates in iteration 2 based on the usage data (using NDCG@10 metric). An NDCG@10 score of

[3] https://github.com/spotify/featran
[4] http://www.anc.org/

Testing	NDCG@10									
	Training instances on previous iterations									
	#sentences	baseline	1	≤ 2	≤3	≤4	≤5	≤6	≤7	≤8
1	115	-	-	-	-	-	-	-	-	-
2	214	60.66	62.88	-	-	-	-	-	-	-
3	207	61.05	63.39	65.52	-	-	-	-	-	-
4	210	58.21	60.73	65.93	67.46	-	-	-	-	-
5	233	56.10	62.53	65.66	66.00	70.72	-	-	-	-
6	215	62.18	61.05	66.51	67.86	69.88	72.36	-	-	-
7	213	57.00	62.07	64.02	64.88	67.28	69.27	74.14	-	-
8	195	56.56	59.53	62.11	63.03	64.54	67.40	71.05	75.83	-
9	224	56.14	63.48	65.58	65.87	69.18	69.51	71.31	71.40	75.70

Table 2: NDCG@10 results for each iteration of the testing instances using training instances from the previous iteration. For example, for testing at iteration 2, the NDCG@10 result using training data from the previous iteration, i.e. iteration 1, is 62.88. The baseline column shows the performance in each iteration using the generic paraphrasing dataset used to train the baseline ranking model.

	Iteration 2			Iteration 3			Iteration 4		
Workers	Instances (#)	positive (%)	NDCG score	Instances (#)	positive (%)	NDCG score	Instances (#)	positive (%)	NDCG score
AXXXL5	950	10.21	51.35	2661	10.11	55.87	2771	9.78	56.57
AXXX3N	1591	10.31	45.45	3130	10.29	48.72	5367	10.23	47.98
AXXXMY	1117	10.12	55.15	2753	10.10	61.35	4809	10.13	63.76
AXXXI7	70	10.00	49.33	2162	10.59	64.10	3988	10.38	66.82
AXXX56	1190	10.42	54.63	2468	10.29	56.24	4477	10.27	58.79
AXXXS1	824	10.19	54.45	1845	10.24	55.28	3045	10.15	58.78
AXXXM9	448	10.04	55.25	896	10.04	56.00	2669	11.09	58.61
AXXXAM	1594	10.16	60.59	2999	10.17	61.96	4611	10.13	63.28
AXXX3E	615	10.73	59.44	1038	10.69	59.51	3451	10.66	62.59
AXXXGI	100	24.00	45.05	1979	11.22	56.72	3160	10.79	57.35

Table 3: The NDCG result for 10 different workers. *Instances* shows the total number of training instances used form the previous iteration while *positive* shows the total number of positive feedback provided by the user. The workers ID are obscured to protect their privacy.

62.88 is obtained (see Table 2), which is already better than using the baseline system (60.66). Figure 4 shows the learning curve over the different iterations conducted in the experiment.

Similarly, training instances collected from iteration 1 and iteration 2 are used to train a ranking model, which is used to re-rank candidates for texts in the next iteration (iteration 3). We continued the experiment for nine iterations and we record the performance at each iteration.

We have observed that the ranking model substantially improves on every iteration based on the NDCG@10 ranking evaluation measure. Table 2 also shows that if we test the performance on each of the models from the earlier iterations, the performance of the system declines, thus the system can make good use of more usage data if available. For example, on iteration 6, testing on a ranking model that is trained based on training instances from iteration 1 up to iteration 5 (≤**5**) produces an NDCG@10 score of 72.36 while testing on the ranking model trained based on training instances from iteration 1 up to iteration 4 (≤**4**) produces an NDCG@10 score of 69.88.

Furthermore, we have explored the effect of the adaptive system for individual workers. In this case, we have built simulated unique models for the 10 top workers, who have participated in at least 4 different batches (iterations) of the task. We use the first iteration the worker has participated as an initial model, and start using the model for the subsequent iterations. As we can see from Table 3 and Figure 5, the NDCG scores improve consistently over the iterations. The results also revealed that text simplification can be modeled differently based on the user needs (personalization).

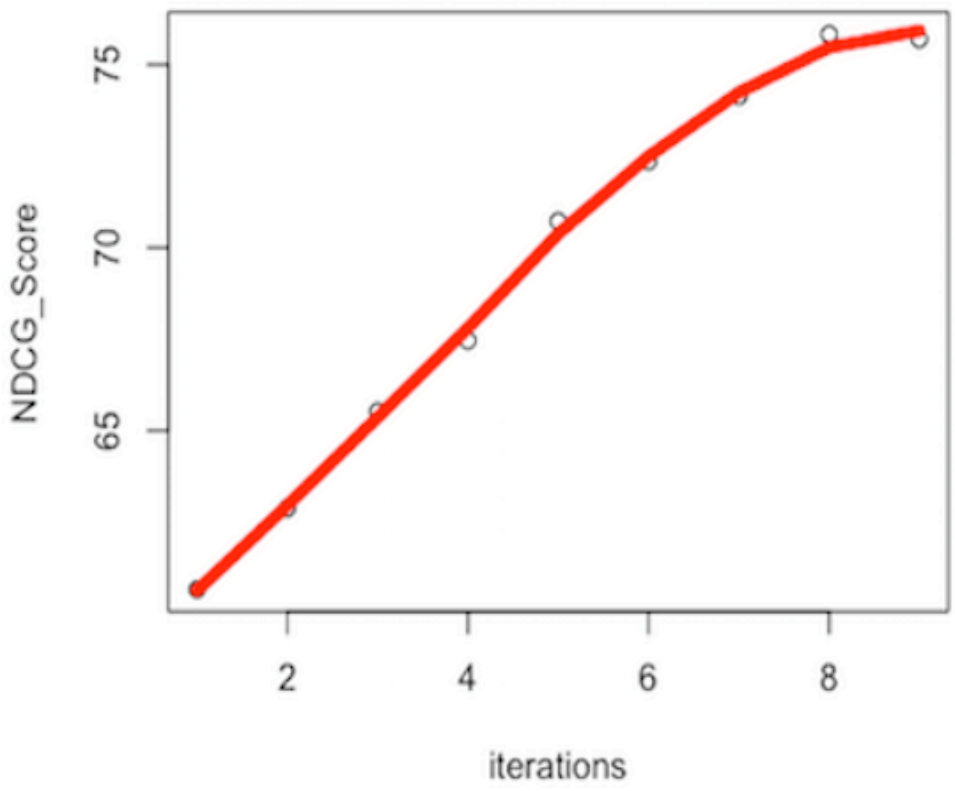


Figure 4: Learning curve showing the increase of NDCG@10 score over 9 iterations.

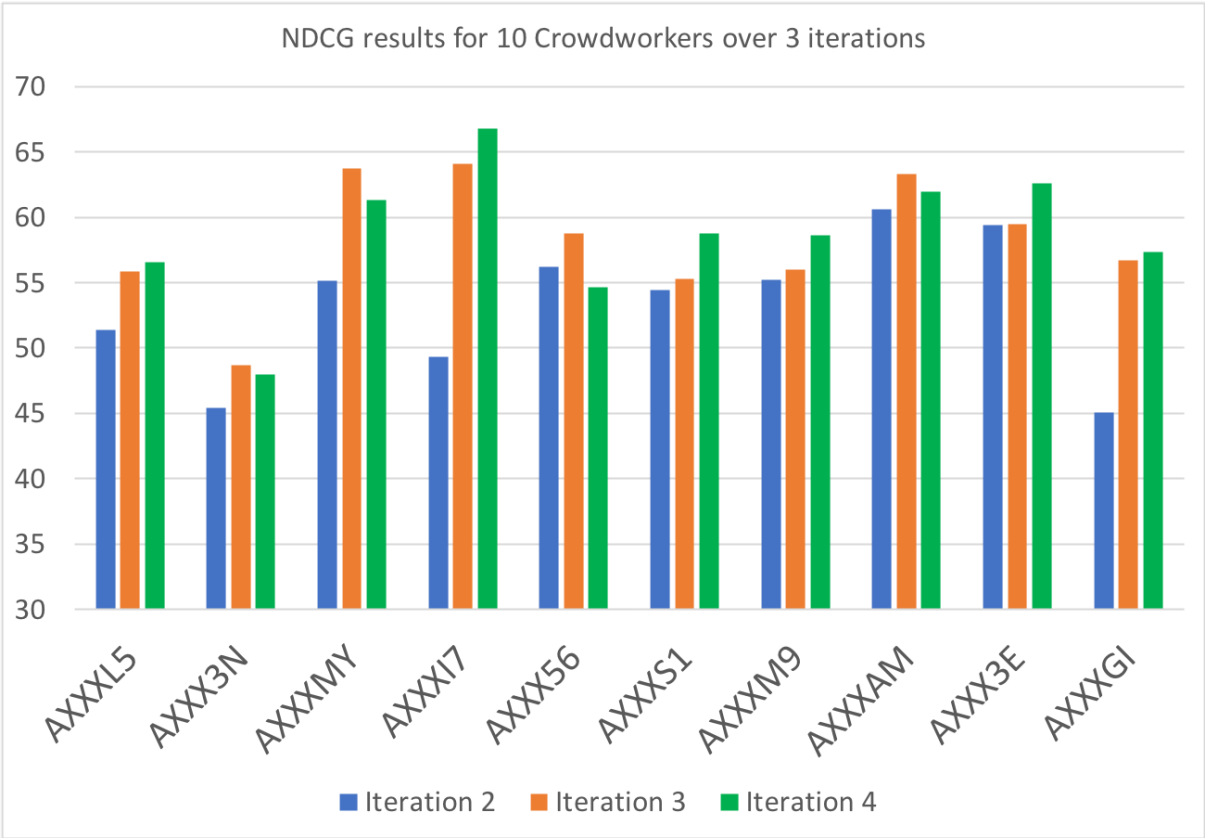


Figure 5: The increase of NDCG@10 score over 3 iterations for the top 10 workers ordered by their productiveness (who have have completed most HITs over several iterations).

7 Discussion

Most text simplification systems, and for that matter, most NLP models, are based on a traditional *collect and train* approach where first all the required training data are annotated , then training and evaluation is carried out after the data collection. To our knowledge, this experiment is the first scientific work to conduct an adaptive approach for text simplification where signals from usage data are collected in an interactive and iterative approach to improve the model of an NLP component.

We have demonstrated that our approach is noble in many aspects: 1) the adaptive learning model is integrated in the real-world NLP application (*live-usage – RQ1*), 2) the performance of the integrated adaptive model improves through usage data of the NLP application (*adaptability – RQ3*), 3) the integrated learning model potentially adapts to the needs of the user or user groups through usage data (*personalized NLP –RQ3*), and 4) we also have shown that adaptive systems can be evaluated incrementally, by comparing the system's suggestions by the ranking model to the actual ranking provided by the users (*incremental evaluation – RQ2*) .

In this research, we also have showcased how to perform web-scaled and real-time adaptive data collection using the Amazon MTurk crowdsourcing platform. The MTurk crowdsourcing platform has been mainly used to collect datasets for tasks that are not complex and difficult to complete such as identifying named entities or biomedical entities in a text, categorizing texts for spam, labeling an image with appropriate captions and so on. Using MTurk's external HIT, we are able to show that the MTurk crowdsourcing platform can be successfully used for complex NLP applications such as text simplification with a writing aid tool, which normally is limited to a lab-based experiment.

8 Conclusion and Future Directions

In this work, we have shown that the integration of an adaptive paraphrase ranking model effectively improves the performance of text simplification task. We have designed a full-fledged, web-scale based text simplification system where we have integrated an adaptive paraphrase ranking model into the tool.

Our tool is integrated with the Amazon Mechanical Turk crowdsourcing platform to collect usage data for text simplification.

To evaluate the performance of the adaptive system on the collected usage data, we have evaluated ranking model performance in an iterative way. In every iteration, we use the usage data exclusively from the previous iterations (except the first iteration that is used solely as a training data and we do not evaluate it) to training the learning-to-rank model. The result shows that, in every iteration, there is a large increase in performance based on the NDCG@10 evaluation metric.

We believe that this experiment is a showcase on how to develop a personalized NLP application. Using a similar approach, one can effectively deploy Par4Sim for a different purpose such as to write technical documents. The research also sheds light on a domain or task adaptions. One can use datasets collected for general purpose domains and it is possible to adapt the model based on the usage data over a period of time. This is a much cheaper alternative than collecting labeled datasets anew.

In the future, we would like to run a long-turn study with arbitrary users and arbitrary texts using a freely available online tool. Specifically for text simplification, the approach can be employed to provided graded complexity level of texts, as it is done for instance in the *Newsela* instructional content platform[5]. We also envision further possible tasks where adaptive learning helps, such as collaborative text composing and recommender systems.

The software is openly available under ASL 2.0 license and the resources and datasets used in this paper are released under CC-BY[6]. The demo of the tool as it was used inside the MTurk browser can be accessed online.[7]

[5] https://newsela.com/data/

[6] https://uhh-lt.github.io/par4sim/

[7] https://ltmaggie.informatik.uni-hamburg.de/par4sim/

Acknowledgement

This work has been partially supported by the SEMSCH project at the University of Hamburg, funded by the German Research Foundation (DFG).

We would like to thank the PC chairs, ACs, and reviewers for their detailed comments and suggestions for our paper. We also would like to thank colleagues at LT lab for testing the user interface. Special thanks goes to Rawda Assefa and Sisay Adugna for the proofreading of the Amharic abstract translation.

References

Sian Alsop and Hilary Nesi. 2009. Issues in the development of the British Academic Written English (BAWE) corpus. *Corpora*, 4(1):71–83.

Biemann, C., Riedl, and M. 2013. Text: Now in 2D! A Framework for Lexical Expansion with Contextual Similarity. *Journal of Language Modelling*, 1(1):55–95.

Stefan Bott, Luz Rello, Biljana Drndarevic, and Horacio Saggion. 2012. Can Spanish Be Simpler? LexSiS: Lexical Simplification for Spanish. In *Proceedings of COLING 2012*, pages 357–374, Mumbai, India.

Léon Bottou. 1998. On-line Learning in Neural Networks. chapter On-line Learning and Stochastic Approximations, pages 9–42. New York City, NY, USA.

Christopher J. C. Burges. 2010. From RankNet to LambdaRank to LambdaMART: An Overview. Technical report, Microsoft Research.

William Coster and David Kauchak. 2011. Simple English Wikipedia: A New Text Simplification Task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 665–669, Portland, OR, USA.

Koby Crammer and Yoram Singer. 2003. Ultraconservative Online Algorithms for Multiclass Problems. *J. Mach. Learn. Res.*, 3:951–991.

Pinar Donmez, Krysta M. Svore, and Christopher J.C. Burges. 2009. On the Local Optimality of LambdaRank. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 460–467, Boston, MA, USA.

David Graff. 2002. The AQUAINT Corpus of English News Text LDC2002T31. In *Web Download. Philadelphia: Linguistic Data Consortium*.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. A Lexical Simplifier Using Wikipedia. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 458–463, Baltimore, MA, USA.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Transactions on Information and System Security*, 20(4):422–446.

Walter S. Lasecki, Luz Rello, and Jeffrey P. Bigham. 2015. Measuring Text Simplification with the Crowd. In *Proceedings of the 12th Web for All Conference*, W4A '15, pages 4:1–4:9, Florence, Italy.

Abby Levenberg, Chris Callison-Burch, and Miles Osborne. 2010. Stream-based Translation Models for Statistical Machine Translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 394–402, Los Angeles, CA, USA.

Hang Li. 2014. *Learning to Rank for Information Retrieval and Natural Language Processing: Second Edition*. Morgan & Claypool Publishers, 2nd edition.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013*, pages 3111–3119, Stateline, NV, USA.

George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM*, 38(11):39–41.

G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. 2018. Continual Lifelong Learning with Neural Networks: A Review. *ArXiv e-prints, 1802.07569*.

Adam Pauls and Dan Klein. 2011. Faster and Smaller N-gram Language Models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 258–267, Portland, OR, USA.

Ellie Pavlick and Chris Callison-Burch. 2016. Simple PPDB: A Paraphrase Database for Simplification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 143–148, Berlin, Germany.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430, Beijing, China.

Matthew Shardlow. 2014. A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications*, 4(1):58–70.

Hoai-Viet To, Ryutaro Ichise, and Hoai-Bac Le. 2009. An Adaptive Machine Learning Framework with User Interaction for Ontology Matching. In *the IJCAI 2009 Workshop on Information Integration on the Web*, pages 35–40, Pasadena, CA, USA.

Alexey Tsymbal. 2004. The Problem of Concept Drift: Definitions and Related Work. Technical report, Department of Computer Science, Trinity College: Dublin, Ireland.

Sanja Štajner, Marc Franco-Salvador, Simone Paolo Ponzetto, Paolo Rosso, and Heiner Stuckenschmidt. 2017. Sentence Alignment Methods for Improving Text Simplification Systems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 97–102, Vancouver, Canada.

Yining Wang, Liwei Wang, Yuanzhi Li, Di He, Wei Chen, and Tie-Yan Liu. 2013. A Theoretical Analysis of Normalized Discounted Cumulative Gain (NDCG) Ranking Measures. In *Machine Learning Research*, pages 25–54, Princeton, NJ, USA.

Jialei Wang, Ji Wan, Yongdong Zhang, and Steven Hoi. 2015. SOLAR: Scalable Online Learning Algorithms for Ranking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1692–1701, Beijing, China.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing Statistical Machine Translation for Text Simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Seid Muhie Yimam, Chris Biemann, Ljiljana Majnaric, Šefket Šabanović, and Andreas Holzinger. 2016a. An adaptive annotation approach for biomedical entity and relation recognition. *Brain Informatics*, 3(3):157–168.

Seid Muhie Yimam, Héctor Martínez Alonso, Martin Riedl, and Chris Biemann. 2016b. Learning Paraphrasing for Multiword Expressions. In *Proceedings of the 12th Workshop on Multiword Expressions*, pages 1–10, Berlin, Germany.

Seid Muhie Yimam, Sanja Štajner, Martin Riedl, and Chris Biemann. 2017. CWIG3G2 - Complex Word Identification Task across Three Text Genres and Two User Groups. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 401–407, Taipei, Taiwan.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo H. Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Zampieri Marcos. 2018. A Report on the Complex Word Identification Shared Task 2018. In *The 13th Workshop on Innovative Use of NLP for Building Educational Applications, NAACL 2018 Workshops*, pages 66–78, New Orleans, LA, USA.

Indrė Žliobaitė, Mykola Pechenizkiy, and João Gama. 2016. An Overview of Concept Drift Applications. In Nathalie Japkowicz and Jerzy Stefanowski, editors, *Big Data Analysis: New Algorithms for a New Society*, pages 91–114.

Topic or Style? Exploring the Most Useful Features for Authorship Attribution

Yunita Sari, Mark Stevenson and **Andreas Vlachos**
Department of Computer Science
University of Sheffield, UK
{y.sari, mark.stevenson, a.vlachos}@sheffield.ac.uk

Abstract

Approaches to authorship attribution, the task of identifying the author of a document, are based on analysis of individuals' writing style and/or preferred topics. Although the problem has been widely explored, no previous studies have analysed the relationship between dataset characteristics and effectiveness of different types of features. This study carries out an analysis of four widely used datasets to explore how different types of features affect authorship attribution accuracy under varying conditions. The results of the analysis are applied to authorship attribution models based on both discrete and continuous representations. We apply the conclusions from our analysis to an extension of an existing approach to authorship attribution and outperform the prior state-of-the-art on two out of the four datasets used.

1 Introduction

Authorship attribution plays an important role in many applications, including plagiarism detection and forensic investigation. Approaches to this problem attempt to identify a document's author through analysis of individual's writing style and/or topics they tend to write about. The problem has been extensively studied and a wide range of features has been explored (Stamatatos, 2013; Schwartz et al., 2013; Seroussi et al., 2013; Hürlimann et al., 2015). However, there has been a lack of analysis of the behavior of features across multiple datasets or using a range of classifiers. Consequently, it is difficult to determine which types of features will be most useful for a particular authorship attribution dataset.

Authorship attribution is a unique task which is closely related to both the representation of individuals' writing style and text categorization. In some cases, where there is a clear topical distinction between the documents written by different authors, content-related features such as those used in text categorization may be effective. However, style-based features are more likely to be effective for datasets containing a more homogeneous set of topics. Previous work on feature exploration for authorship attribution, focused on the overall effectiveness of features without considering the characteristics of the datasets to which they were applied, e.g. (Grieve, 2007; Guthrie, 2008; Stamatatos, 2009; Brennan et al., 2012; Sapkota et al., 2015). A wide range of features have been applied to the authorship attribution problem and many previous studies concluded that using character n-grams is often effective, e.g. (Peng et al., 2003; Koppel et al., 2011; Schwartz et al., 2013; Sapkota et al., 2015; Sari et al., 2017; Shrestha et al., 2017). Thus, character n-grams have become the *go-to* features for this task to capture both an author's topical preferences and writing style.

This study explores how the characteristics of a dataset affect the usefulness of different types of features for the authorship attribution task. Experiments are carried out using four datasets that have previously been widely used for this task. Three types of features are considered: *style*, *content* and *hybrid* (a mixture of the previous two types). In contrast to previous work, this study finds that character n-grams do not perform equally well in all datasets. The analysis holds for authorship attribution models using discrete and continuous representations. Using topic modeling and feature analysis, the most effective features can be successfully predicted for three of the four datasets. The results of this analysis

Proceedings of the 27th International Conference on Computational Linguistics, pages 343–353
Santa Fe, New Mexico, USA, August 20-26, 2018.

are applied via a novel extension of a recently proposed neural approach (Sari et al., 2017) and improved state-of-the-art performance are obtained for two of the four datasets.

2 Related Work

Authorship attribution features are often referred to *stylometric features* since the main goal of the task is often thought to be modeling the authors' writing style. Grieve (2007) conducted experiments which involved thirty-nine different types of textual measurements commonly used in attribution studies. His experiments, which were performed using the Chi-squared test on The Telegraph Columnist corpus, concluded that the combination of word and punctuation mark profiles are effective features for representing authors. Similar to Grieve, Guthrie (2008) carried out an exploration of 166 features used for authorship attribution including commonly used stylistic features and several others intended to capture emotional tone. He reported that fifteen features, including punctuation marks, pronouns, fog index and average sentence length to be the most useful. Stamatatos (2009) divided authorship attribution features into five groups: lexical, character, syntactic, semantic and application-specific features. Compared to others, lexical and character features are commonly used in authorship attribution work as they provide rich information about the author's topical preferences and writing style. In addition, both types of features can be extracted in many languages and datasets with little effort.

Simple lexical features (e.g. word frequencies, word n-grams, function words, hapax legomena, word/sentence length) have been widely used since early attribution work (Mendenhall, 1887). Function words have been proved to be effective features and several studies have reported their usefulness, e.g. (Mosteller and Wallace, 1964; Argamon and Levitan, 2005; Koppel et al., 2005; Juola and Baayen, 2005; Zhao and Zobel, 2005). Bag-of-words approaches have also been reported as being useful for authorship attribution (Koppel et al., 2011). These approaches are also commonly applied for sentiment analysis and topic classification tasks (Zhang et al., 2015; Heap et al., 2017).

The usefulness of character n-grams has been highlighted in several studies including (Peng et al., 2003; Stamatatos, 2013; Schwartz et al., 2013; Sapkota et al., 2015). Koppel et al. (2011) argued that this effectiveness comes from their ability to capture both content and stylistic information. Similar conclusions were reported by Sapkota et al. (2015). They analysed subgroups of character n-grams in both single and cross-domain settings. They concluded that affixes and punctuation n-grams make a significant contribution towards the effectiveness of character n-grams.

Our study differs from previous work in that we perform dataset analysis using topic modeling followed by feature ablation experiments. Thus, we are able to determine the level that each type of feature affects authorship attribution accuracy.

3 Datasets

Experiments[1] are performed using four authorship attribution datasets: Judgment (Seroussi et al., 2011), CCAT10, CCAT50 (Stamatatos, 2008), and IMDb62 (Seroussi et al., 2010). These datasets were chosen because they are all commonly used in previous literature and represent a range of characteristics in terms of the number of authors, topic/genre and document length (see Table 1).

	Judgment	CCAT10	CCAT50	IMDb62
genre	legal judgments	newswire		movie reviews
# authors	3	10	50	62
# total documents	1,342	1,000	5,000	79,550
avg characters per document	11,957	3,089	3,058	1,401
avg words per document	2367	580	584	288

Table 1: Dataset statistics.

[1]Code to reproduce the experiments is available from `https://github.com/yunitata/coling2018`

Judgment consists of legal judgments from three Australian High Court judges while both CCAT datasets are subsets of Reuters Corpus Volume 1 (RCV1) (Rose et al., 2002). The IMDb62 dataset was collected from movie reviews and message board posts of the Internet Movie database. Train/test partitions are provided for both CCAT datasets by the respective authors. For Judgment and IMDb62 we follow previous work (Seroussi et al., 2013) by using 10-fold cross validation in our experiments. We do not make use of datasets from recent authorship attribution shared task events, e.g. PAN (Juola, 2012), due to their relatively small size and fact that they provide a very small number of documents per author.

4 Dataset Analysis

The aim of this analysis is to quantify the topical similarity between authors in each of the datasets considered. The motivation for this is that certain datasets may have clear topical preferences between authors which cause authorship attribution to be biased towards topic classification. Therefore, topic modeling can help assess the topical (dis-)similarity among authors. We use Latent Dirichlet Allocation (LDA) (Blei et al., 2003), a generative probabilistic model for text collections. Documents are represented as mixtures over latent topics, where each topic is characterized by a distribution over words. Assuming a trained topic model over an authorship attribution dataset D, if C_α is the set of documents written by author α and Φ_i is the topic distribution for the i-th document in C_α, then we estimate the topic distribution for a particular author, θ_α, as follows:

$$\theta_\alpha = \frac{\sum_{i=1}^{|C_\alpha|} \Phi_i}{|C_\alpha|} \tag{1}$$

Following this, the difference between two author's topic probability distributions θ_α and θ_β is calculated using the Jensen-Shannon Divergence (JSD) (Cover and Thomas, 2006):

$$JSD(\theta_\alpha, \theta_\beta) = \frac{1}{2} D_{KL}(\theta_\alpha || M) + \frac{1}{2} D_{KL}(\theta_\beta || M) \tag{2}$$

where $M = \frac{1}{2}(\theta_\alpha + \theta_\beta)$ and D_{KL} is Kullback-Leibler divergence.

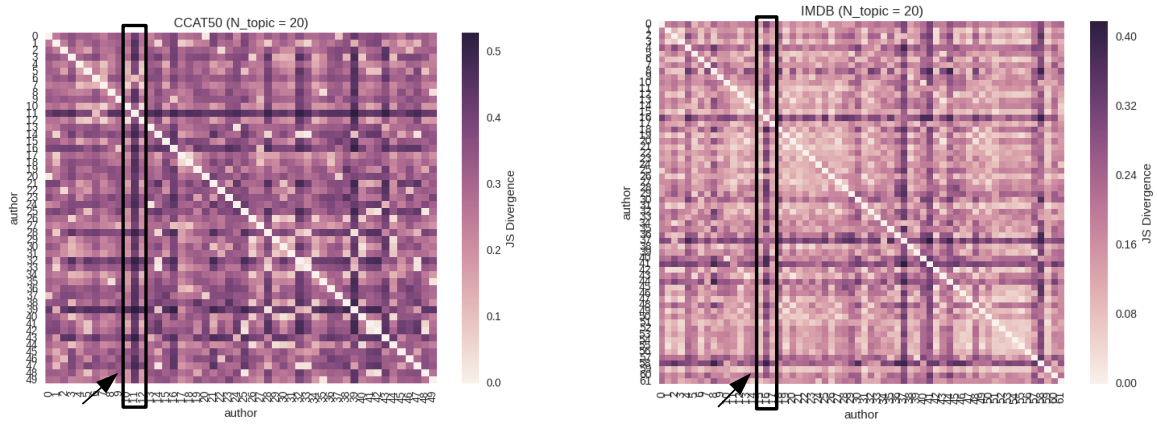
Table 2 shows the average of JSD for all author pairs in each of the datasets having trained a topic model with varying numbers of topics. High JSD scores indicate more topical dis-similarity between authors in the dataset.[2] The CCAT datasets, which contain on-line news, have higher scores compared to Judgment and IMDb62. The scores for CCAT50 and CCAT10 are similar, despite the fact that the first dataset contains five times the number of authors of the second. The consistency of this comparison across different numbers of topics indicates that this method of assessing content similarity between authors is robust with respect to tuning this parameter. Judgment has the lowest score across the four datasets indicating that the authors discuss the most similar topics. Finally, scores for the IMDb62 dataset obtained were higher than those for Judgment but lower than both CCAT's scores. Differences in scores for IMDb62 are due to the authors' preferences, as some comment on the story while others comment on the characters of the movie. Overall, we observe that the genre of the datasets influences the topical dis-similarity between authors.

Confusion matrices were created to further analyse the differences between authors. These matrices were generated after running LDA with 20 topics for 1000 iterations. Similar patterns were observed using different numbers of topics. Darker color indicates higher JSD score between two authors. In the CCAT50 dataset (Figure 1a), one author (number 11, indicated by an arrow) has very different topic preferences compared to the others. Articles written by author 11 mainly discuss topics related to *gold, exploration, Canada, Indonesia* which are rarely picked by the other authors. A similar pattern is found in IMDb62 (see Figure 1b), where reviews by author 16 (also indicated by an arrow) are dominated by positive comments about movies unlike other authors who tended to write negative reviews or discuss the story and/or the characters. However, unlike the aforementioned datasets, authors in Judgment wrote about relatively similar topics.

[2] We do not assume linear scaling.

#topic	Judgment	CCAT10	CCAT50	IMDb62
3	0.0056	0.2053	0.1785	0.1000
10	0.0148	0.3010	0.2867	0.1471
20	0.0180	0.3193	0.3279	0.1617
30	0.0256	0.3414	0.3269	0.1627
40	0.0272	0.3417	0.3291	0.1681
50	0.0281	0.3403	0.3326	0.1634

Table 2: Average JS Divergence



(a) CCAT50 (train)

(b) IMDb62

Figure 1: Author topic distribution (20 topics).

5 Feature Analysis

An ablation study was carried out to determine the contribution of different features for each dataset. Following previous studies (Abbasi and Chen, 2008; Stamatatos, 2009), feature groups are divided into three types (see Table 3):

- **Style:** Style-based features, such as usage of function words, digits and punctuation, capture an author's writing style. We used pre-defined sets of 174 function words and 12 punctuation marks.
- **Content:** Content-based features, e.g. bags of n-grams, represent the author's topical preferences. All function words are removed when extracting these features.
- **Hybrid:** The final feature type, hybrid, are character n-grams which are intended to capture both writing style and topical preferences (Koppel et al., 2011; Sapkota et al., 2015).

Both character and word *n*-grams are limited to bi- and tri-grams. As the purpose of these ablation experiments is not to outperform previous work, only the 100 most common n-grams are used for each feature type.

Authorship attribution experiments were carried out using two classifiers: a single hidden layer Feed-forward Neural Network model (FNN) and Logistic Regression (LR). The FNN hyper-parameters including the number of neurons and dropout rates were tuned on the development set for each of the datasets. For Judgment, CCAT10 and CCAT50, we set the number of epochs to 250 and 100 for IMDb62. For all datasets, early stopping was used on the development sets and the models were optimized with the Adam update rule (Kingma and Ba, 2015). Since none of the datasets have a standard development set, we randomly selected 10% of the training data for this purpose. For LR, we found that using the default parameters from Scikit-Learn (Pedregosa et al., 2011) resulted in comparable performances to the FNN. Accuracy was used as the evaluation metric to measure authorship attribution performance.

Type	Group	Category	#	Description
Style	Lexical	Word-level	2	Average word length, number of short words
		Char-level	2	Percentage of digits, percentage of uppercase letters
		Letters	26	Letter frequency
		Digits	10	Digit frequency
		Vocabulary richness	2	Richness (hapax-legomena and dislegomena)
	Syntactic	Function words	174	Frequency of function words
		Punctuation	12	Occurrence of punctuation
Content	Word *n*-gram	Words unigrams	100	Frequency of 100 most common word unigrams
		Words bigrams	100	Frequency of 100 most common word bigrams
		Word trigrams	100	Frequency of 100 most common word trigrams
Hybrid	Char *n*-gram	Char bigrams	100	Frequency of 100 most common character bigrams
		Char trigrams	100	Frequency of 100 most common character trigrams

Table 3: Authorship attribution feature sets.

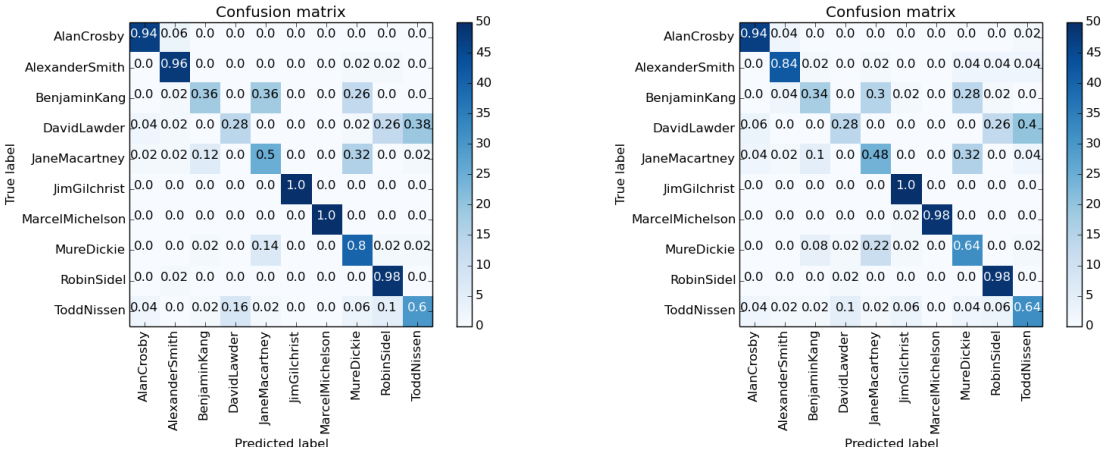
5.1 Feature Ablation Results

Results are presented in Table 4. The $(-)$ symbol indicates that the respective feature type is excluded. The results confirm our topic model-based analysis (see Section 4). Style-based features are more effective for datasets in which authors discuss similar topics, e.g. Judgment and IMDb62. As expected, content-based features are generally more effective when there is more dis-similarity between the topics discussed by the authors in the dataset, e.g. CCAT10 and CCAT50, but are of limited usefulness when the topics are similar (particularly for the Judgment dataset). The hybrid features appear to behave similarly to the content-based features since they are most useful when the topic dis-similarity between authors is high.

Features	Judgment		CCAT10		CCAT50		IMDb62	
	FNN	LR	FNN	LR	FNN	LR	FNN	LR
all features	89.43	90.02	75.40	74.20	60.20	60.56	85.25	85.00
(–) Style	**-3.87**	**-4.32**	-3.00	+0.40	-3.40	-2.60	**-6.91**	**-8.39**
(–) Content	-1.43	+0.30	**-3.60**	**-3.00**	**-4.52**	-4.08	-2.77	-2.68
(–) Hybrid	-0.83	-0.29	-3.40	-1.00	-1.28	**-4.68**	-2.02	-5.32

Table 4: Feature ablation results.

To examine the results further, we generated confusion matrices for the Logistic Regression (LR) classifier applied on CCAT10 dataset (Figure 2). The effect of removing content-based features is shown in Figure 2b where the prediction accuracy for authors *Alexander Smith* and *Mure Dickie* drops from 96% and 80% (see Figure 2a) to 84% and 64% respectively. Content-based features are essential in this particular genre (newswire) dataset, since each author usually has different topical interests. For example, among the ten authors in the dataset, *Alexander Smith* mostly discussed topic related to *investment and*



(a) All features (b) (−) Content

Figure 2: Confusion Matrices of LR classifier with different features types on CCAT10.

finance, while *China* was dominantly written about by *Mure Dickie, Benjamin Kang* and *Jane Macartney*. In addition, writing style between authors in this genre can be very similar. Thus, applying style-based or hybrid features alone may not be effective.

Additional feature exploration was carried out to analyse what types of features are more important to the classifier overall. We performed an analysis using LIME (Ribeiro et al., 2016), a model agnostic framework for intepretability. LIME provides explanations about how a classifier made a prediction by identifying important input features. We selected a document from each of the datasets and analysed what kind of features were used in its prediction. Figures 3, 4, 5 and 6 present the predictions of LR trained on 1000 word unigrams in Judgment, CCAT10, CCAT50, and IMDb62 respectively. In these experiment function words are not removed. For each of the documents presented, LR made correct author predictions with probability close to 100%. The darker shade indicates more important words in the attribution prediction. In the two CCAT datasets the classifier put more weight on content-based words such as *Thomson, Canada* and *Toronto*. In contrast, function words e.g. *at, had, and, was* appear to be more salient in Judgment and IMDb62.

.) . but in bracton 's account of the great convention at merton we read only of those qui nati and not those qui geniti , fuerunt ante sponsalia vel matrimonium , (fol . :) and we learn elsewhere from him (fol .) that it mattered not for legitimacy whether the child was begotten and born after the marriage or begotten before but born into the marriage or begotten in the marriage and born after the marriage had been dissolved and whether dissolved by death or by divorce and that it did not matter whether the union was by matrimonium or (subject to certain exceptions) by sponsalia . but we find that he recognizes adulterine bastardy . the child is to be presumed a bastard , it appears , if feebleness , frigidity or impotence of the husband is proved per multum tempus or absence for two years from the kingdom or from the county or shire is shown . if he returns and finds his wife pregnant or that she has a child of a year or less , whether he avows it and nurtures it or not , the child may rightly be excluded from succession because it could not be heir . but , if it was possible to presume that he could have engendered the child , it seems that decisive importance was given to avowal and nurture by

Figure 3: Important word unigrams features in Judgment.

Alcatel Alsthom said on Monday it was in talks with Aerospatiale and Dassault about a joint offer for the government's 58-percent stake in defence electronics group Thomson-CSF.
Prime Minister Alain Juppe said a decision about the procedure for the privatisation of Thomson-CSF would be made before the end of February.
"There are discussions with the companies mentioned in the press," an Alcatel spokesman said when asked to react to newspaper reports about a joint bid.
Industry sources said that Alcatel chairman Serge Tchuruk had kept the government informed about his plans to form an alliance with Aerospatiale and Dassault in order to win the Thomson-CSF stake.
Alcatel in October lost out to Lagardere Groupe in bidding for state-controlled Thomson SA, which has the stake in Thomson-CSF as well as 100 percent of consumer electronics group Thomson Multimedia (TMM).
But the government had to suspend the sale on December 4 after the independent Privatisation Commission balked against the terms of the sale by Lagardere of TMM to Daewoo Electronics of South Korea.

Figure 4: Important word unigrams features in CCAT10.

We also observed a document in the IMDb62 dataset where the classifier assigns similar prediction

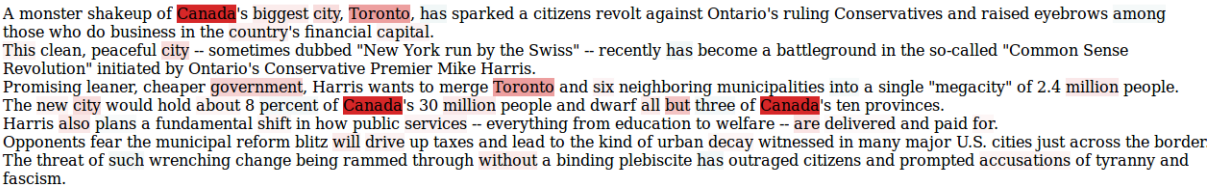


Figure 5: Important word unigrams features in CCAT50.

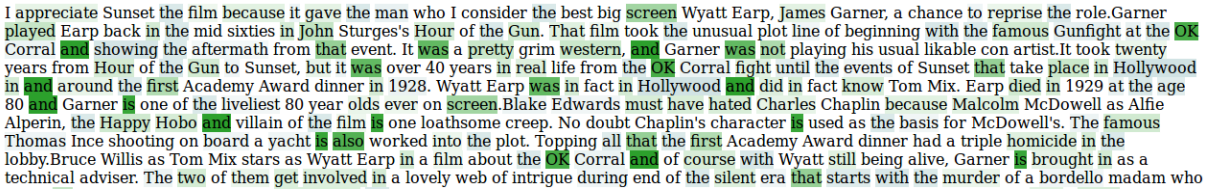


Figure 6: Important word unigrams features in IMDb62.

probabilities to two authors as presented in Figure 7. The classifier put the same weight to function words *and* and *to* which represent two different classes of authors, 26 and not 26 (the LR classifier uses a one-versus-all scheme). The correct decision of the classifier is more likely helped by the presence of some less significant features such as *is, becomes, There, usual* and *could*.

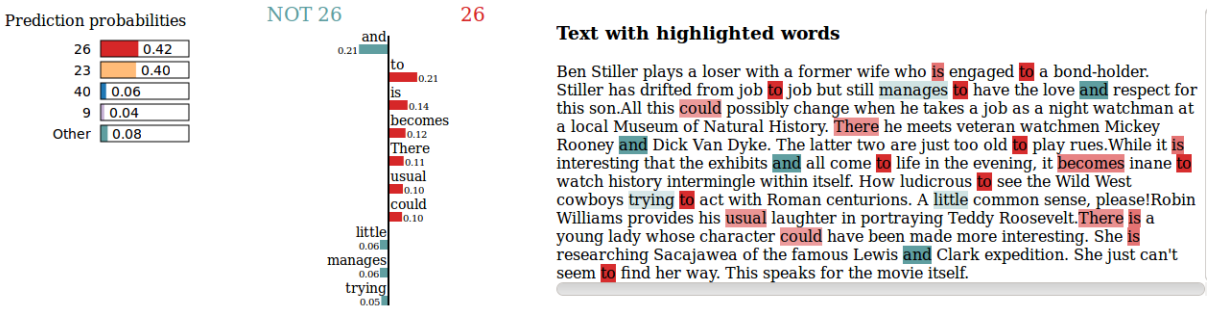


Figure 7: Explanation of individual predictions of Logistic Regression classifier on an IMDb62 document using LIME. The bar chart represent the weight given to the most relevant words which are also highlighted in the text.

6 Extending a Neural Model

These findings are further validated by applying them to a continuous n-grams-based authorship attribution model recently proposed by Sari et al. (2017). They represented a document as a bag of n-grams features and learned the continuous representation of each feature jointly with the classifier in a shallow feed-forward neural network (Joulin et al., 2017). Sari et al. conducted experiments with three different feature choices: characters, words and their combination. The character-based model outperformed the state-of-the-art on the CCAT50 and IMDb62 datasets, while producing comparable results on the remaining two.

We extend their character-based model by incorporating each feature type (*style, content* and *hybrid*) as an auxiliary feature represented in discrete form. Auxiliary features provide additional information related to the dataset characteristics. Given x_{aux} as a normalized auxiliary features frequency vector, V is the weight applied to the features and f is the activation function (ReLu), the hidden layer h performs the following computation:

$$h = f(Vx_{aux}) \tag{3}$$

The probability distribution over the label for a document then can be described as:

$$p(y|x) = softmax(W_{out}[Ax, h]) \tag{4}$$

where x is the frequency vector of features for the document, A is the embedding matrix, W_{out} is the weight matrix of the output layer and $[Ax, h]$ is the concatenation of Ax and h.

All the character n-gram embeddings and hidden layer in the model were initialized using Glorot uniform initialization (Glorot and Bengio, 2010). We used the best hyper-parameters values for each of the datasets which have been tuned in the development set via a small grid search over all combinations of embedding size and dropout rate (specifically dropout in the concatenation layer). The hidden size of hidden auxiliary layer was set to 2. For the rest of the hyper-parameters, we used the values from the baseline model (Sari et al., 2017). For Judgment, CCAT10 and CCAT50, we set the number of epochs to 250 and 100 for IMDb62. For all datasets, early stopping was used on the development sets and the models were optimized with the Adam update rule (Kingma and Ba, 2015).

6.1 Results

Table 5 presents the results of the experiment and compares them against previously reported ones on the same data sets. In the bottom portion of the table it can be seen that for each of the four data sets there is at least one feature type which leads to improved results when it is incorporated into the model. Our results demonstrate that better performance can be achieved by taking the data characteristic into account on choosing authorship attribution features. Moreover, the results provide evidence that character n-grams which have been known as typical *go-to* features do not perform equally well in all types of datasets. For three datasets (CCAT10, CCAT50 and IMDb62) the best result is obtained using the feature type identified as being most useful in Section 5. However, we find that using the style features does not improve results on the Judgment dataset as we had expected. The relatively poor performance of the style features may be due to the baseline model (the continuous character n-grams) which effectively captured all the author's writing style. Thus the addition of auxiliary style features did not lead to any improvement.

The results reported here for the CCAT50 and IMDb62 datasets outperform the previously best reported results (Sari et al., 2017) and the model reported here therefore represents a new state-of-the-art performance. The improvements for IMDb62 are statistically significant ($p < 0.05$, paired t-test).

7 Conclusions

This paper describes experiments on the relationship between the effectiveness of different types of features for authorship attribution and characteristics of datasets. We find that the most effective features for datasets can be predicted by applying topic modeling and feature analysis. Content-based features tend to be suitable for datasets with high topical diversity such as the one constructed from on-line news. Datasets with less topical variance, e.g. legal judgments and movie reviews, benefit more from style-based features. The effectiveness of our proposed analysis is further validated by the performance of our proposed neural model which achieved the state-of-the-art results on two datasets.

Model	Judgment	CCAT10	CCAT50	IMDb62
Previous work				
SVM with affix+punctuation 3-grams (Sapkota et al., 2015)	-	78.80	69.30	-
SVM with 2,500 most frequent 3-grams (Plakias and Stamatatos, 2008)	-	80.80	-	-
STM-Asymmetric cross (Plakias and Stamatatos, 2008)	-	78.00	-	-
SVM with bag of local histogram (Escalante et al., 2011)	-	**86.40**	-	-
Token SVM (Seroussi et al., 2013)	91.15	-	-	92.52
Authorship attribution with topic models (Seroussi et al., 2013)	**93.64**	-	-	91.79
Baseline model				
Continuous character n-gram (Sari et al., 2017)	91.29	74.80	72.60	94.80
Proposed model				
(+) style	91.07	76.00	72.72	**95.93***
(+) content	91.51	76.20	**72.88**	95.59
(+) hybrid	91.21	74.80	71.76	95.26

Table 5: Results with comparison against baseline and previous work.
* denotes significant improvement over baseline model ($p < 0.05$).

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments. The first author would like to acknowledge Indonesia Endowment Fund for Education (LPDP) for support in the form of a doctoral studentship.

References

Ahmed Abbasi and Hsinchun Chen. 2008. Writeprints: A Stylometric Approach to Identity-level Identification and Similarity Detection in Cyberspace. *ACM Transactions on Information Systems (TOIS)*, 26(2).

Shlomo Argamon and Shlomo Levitan. 2005. Measuring the Usefulness of Function Words for Authorship Attribution. In *Proceedings of the 2005 ACH/ALLC Conference*, pages 4–7.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3:993–1022, March.

Michael Brennan, Sadia Afroz, and Rachel Greenstadt. 2012. Circumventing Authorship Recognition to Preserve Privacy and Anonymity. *ACM Transactions on Information and System Security*, 15(3):1–22, November.

Thomas M. Cover and Joy A. Thomas. 2006. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience.

Hugo Jair Escalante, Thamar Solorio, and Manuel Montes-y Gómez. 2011. Local Histograms of Character N-grams for Authorship Attribution. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 288–298, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding The Difficulty of Training Deep Feedforward Neural Networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics, AISTATS 2010*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. Society for Artificial Intelligence and Statistics.

Jack. Grieve. 2007. Quantitative Authorship Attribution: An Evaluation of Techniques. *Literary and Linguistic Computing*, 22(3):251–270, May.

David Guthrie. 2008. *Unsupervised Detection of Anomalous Text*. Ph.D. thesis, University of Sheffield.

Bradford Heap, Michael Bain, Wayne Wobcke, Alfred Krzywicki, and Susanne Schmeidl. 2017. Word vector enrichment of low frequency words in the bag-of-words model for short text multi-class classification problems. *CoRR*, abs/1709.05778.

Manuela Hürlimann, Benno Weck, Esther van den Berg, Simon Šuster, and Malvina Nissim. 2015. GLAD: Groningen Lightweight Authorship Detection—Notebook for PAN at CLEF 2015. In Linda Cappellato, Nicola Ferro, Gareth Jones, and Eric San Juan, editors, *CLEF 2015 Evaluation Labs and Workshop – Working Notes Papers, 8-11 September, Toulouse, France*. CEUR-WS.org, September.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain, April. Association for Computational Linguistics.

Patrick Juola and RH Baayen. 2005. A Controlled-corpus Experiment in Authorship Identification by Cross-entropy. *Literary and Linguistic Computing*, pages 1–10.

Patrick Juola. 2012. An Overview of the Traditional Authorship Attribution Subtask. In Pamela Forner, Jussi Karlgren, and Christa Womser-Hacker, editors, *CLEF 2012 Evaluation Labs and Workshop – Working Notes Papers, 17-20 September, Rome, Italy*, September.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceeding of the 3rd International Conference for Learning Representations, ICLR 2015*, San Diego, CA, May.

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. 2005. Automatically Determining An Anonymous Author's Native Language. *Intelligence and Security Informatics*, pages 209–217.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2011. Authorship Attribution in The Wild. *Language Resources and Evaluation*, 45(1):83–94, March.

T.C. Mendenhall. 1887. The Characteristic Curves of Composition. *Science*, IX:37–49.

F Mosteller and D. L Wallace. 1964. Inference and Disputed Authorship: The Federalist. Addison Wesley.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Fuchun Peng, Dale Schuurmanst, Vlado Kesel, and Shaojun Wan. 2003. Language Independent Authorship Attribution using Character Level Language Models. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics*, Budapest, Hungary.

Spyridon Plakias and Efstathios Stamatatos. 2008. Tensor Space Models for Authorship Identification. In *Proceedings of the 5th Hellenic Conference on Artificial Intelligence: Theories, Models and Applications*, SETN '08, pages 239–249, Berlin, Heidelberg. Springer-Verlag.

Marco Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 97–101, San Diego, California, June. Association for Computational Linguistics.

T. Rose, M. Stevenson, and M. Whitehead. 2002. The Reuters Corpus - from Yesterday's News to Tomorrow's Language Resources. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-02)*, pages 827–832, Las Palmas, Canary Islands.

Upendra Sapkota, Steven Bethard, Manuel Montes, and Thamar Solorio. 2015. Not All Character N-grams Are Created Equal: A Study in Authorship Attribution. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–102, Denver, Colorado, May–June. Association for Computational Linguistics.

Yunita Sari, Andreas Vlachos, and Mark Stevenson. 2017. Continuous N-gram Representations for Authorship Attribution. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 267–273, Valencia, Spain, April. Association for Computational Linguistics.

Roy Schwartz, Oren Tsur, Ari Rappoport, and Moshe Koppel. 2013. Authorship Attribution of Micro-Messages. In *Proceeding of Conference on Empirical Methods in Natural Language Processing*, pages 1880–1891, Seattle, USA.

Yanir Seroussi, Ingrid Zukerman, and Fabian Bohnert, 2010. *Collaborative Inference of Sentiments from Texts*, pages 195–206. Springer Berlin Heidelberg, Berlin, Heidelberg.

Yanir Seroussi, Russell Smyth, and Ingrid Zukerman. 2011. Ghosts from the High Courts past: Evidence from computational linguistics for Dixon ghosting for McTiernan and Rich. *University of New South Wales Law Journal*, 34(3):984–1005.

Yanir Seroussi, Ingrid Zukerman, and Fabian Bohnert. 2013. Authorship Attribution with Topic Models. *Journal Computational Linguistics*, 40(2):269–310.

Prasha Shrestha, Sebastian Sierra, Fabio Gonzalez, Manuel Montes, Paolo Rosso, and Thamar Solorio. 2017. Convolutional Neural Networks for Authorship Attribution of Short Texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 669–674, Valencia, Spain, April. Association for Computational Linguistics.

Efstathios Stamatatos. 2008. Author identification: Using Text Sampling to Handle The Class Imbalance Problem. *Information Processing and Management*, 44(2):790 – 799.

Efstathios Stamatatos. 2009. A Survey of Modern Authorship Attribution Methods. *Journal of the American Society for Information Science and Technology*, 60(3):538–556, March.

Efstathios Stamatatos. 2013. On The Robustness of Authorship Attribution based on Character n-gram Features. *Journal of Law and Policy*, 21(2):421–439.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 649–657, Montreal, Canada. MIT Press.

Ying Zhao and Justin Zobel. 2005. Effective and Scalable Authorship Attribution Using Function Words. *Information Retrieval Technology*, pages 174–189.

A Deep Dive into Word Sense Disambiguation with LSTM

Minh Le[†], **Marten Postma**[†], **Jacopo Urbani**[‡] and **Piek Vossen**[†]
[†] Department of Language, Literature and Communication, Vrije Universiteit Amsterdam
[‡] Department of Computer Science, Vrije Universiteit Amsterdam
{m.n.le,m.c.postma,piek.vossen}@vu.nl, jacopo@cs.vu.nl

Abstract

LSTM-based language models have been shown effective in Word Sense Disambiguation (WSD). In particular, the technique proposed by Yuan et al. (2016) returned state-of-the-art performance in several benchmarks, but neither the training data nor the source code was released. This paper presents the results of a reproduction study and analysis of this technique using only openly available datasets (GigaWord, SemCor, OMSTI) and software (TensorFlow). Our study showed that similar results can be obtained with much less data than hinted at by Yuan et al. (2016). Detailed analyses shed light on the strengths and weaknesses of this method. First, adding more unannotated training data is useful, but is subject to diminishing returns. Second, the model can correctly identify both popular and unpopular meanings. Finally, the limited sense coverage in the annotated datasets is a major limitation. All code and trained models are made freely available.

1 Introduction

Word Sense Disambiguation (WSD) is a long-established task in the NLP community (see Navigli (2009) for a survey) which goal is to annotate lemmas in text with the most appropriate meaning from a lexical database like WordNet (Fellbaum, 1998). Many approaches have been proposed – the more popular ones include the usage of Support Vector Machine (SVM) (Zhong and Ng, 2010), SVM combined with unsupervised trained embeddings (Iacobacci et al., 2016; Rothe and Schütze, 2017), and graph-based approaches (Agirre et al., 2014; Weissenborn et al., 2015).

In recent years, there has been a surge in interest in using Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) to perform WSD (Raganato et al., 2017b; Melamud et al., 2016). These approaches are characterized by their high performance, simplicity and their ability to extract a lot of information from raw text. Among the best-performing ones is the approach by Yuan et al. (2016), in which an LSTM language model trained on a corpus with 100 billion tokens was coupled with small sense-annotated datasets to achieve state-of-the-art performance in all-words WSD.

Even though the results obtained by Yuan et al. (2016) outperform the previous state-of-the-art, neither the used datasets nor the constructed models are available to the community. This is unfortunate because this makes the re-application of this technique a non-trivial process, and it hinders further studies for understanding which limitations prevent even higher accuracies. These could be, for instance, of algorithmic nature or relate to the input (either size or quality), and a deeper understanding is crucial for enabling further improvements. In addition, some details are not reported, and this could prevent other attempts from replicating the results.

To address these issues, we reimplemented Yuan et al. (2016)'s method with the goal of: 1) reproducing and making available the code, trained models, and results and 2) understanding which are the main factors that constitute the strengths and weaknesses of this method. While a full replication is not possible due to the unavailability of the original data, we nevertheless managed to reproduce their approach with other public text corpora, and this allowed us to perform a deeper investigation on the

Proceedings of the 27th International Conference on Computational Linguistics, pages 354–365
Santa Fe, New Mexico, USA, August 20-26, 2018.

performance of this technique. This investigation aimed at understanding how sensitive the WSD approach is w.r.t. the amount of unannotated data (i.e., raw text) used for training, model complexity, how biased the method is towards the choice of the most frequent senses (MFS), and identifying limitations that cannot be overcome with bigger unannotated datasets.

The contribution of this paper is thus two-fold: On the one hand, we present a reproduction study whose results are publicly available and hence can be freely used by the community. Notice that the lack of available models has been explicitly mentioned, in a recent work, as the cause for the missing comparison of this technique with other competitors (Raganato et al., 2017b, footnote 10). On the other hand, we present other experiments to shed more light on the value of this and similar methods.

We anticipate some conclusions. First, a positive result is that we were able to reproduce the method from Yuan et al. (2016) and obtain similar results to the ones originally published. However, to our surprise, these results were obtained using a much smaller corpus of 1.8 billion tokens (Gigaword), which is less than 2% of the data used in the original study. In addition, we observe that the amount of unannotated data is important, but that the relationship between its size and the improvement is not linear, meaning that exponentially more unannotated data is needed in order to improve the performance. Moreover, we show that the percentage of correct sense assignments is more balanced w.r.t sense popularity, meaning that the system has a less-strong bias towards the most-frequent sense (MFS) and is better at recognizing both popular and unpopular meanings. Finally, we show that the limited sense coverage in the annotated datasets is a major limitation, as shown by the fact that resulting model does not have a representation for more than 30% of the meanings which should have been considered for disambiguating the test sets.

2 Background

Current WSD systems can be categorized according to two dimensions: whether they use raw text without any preassigned meaning (unannotated data henceforth), and whether they exploit the relations between synsets in WordNet (synset relations henceforth). One prominent state-of-the-art system that does not rely on unannotated data nor exploits synset relations is It Makes Sense (IMS) (Zhong and Ng, 2010; Taghipour and Ng, 2015). This system uses an SVM to train classifiers for each lemma using only annotated data as training evidence.

In contrast, graph-based WSD systems do not use (un)annotated data but rely on the synset relations. The system UKB (Agirre et al., 2014) represents WordNet as a graph where the synsets are the nodes and the relations are the edges. After the node weights have been initialized using the Personalized Page Rank algorithm, they are updated depending on context information. Then, the synset with the highest weight is chosen. Babelfy (Moro et al., 2014) and the system by Weissenborn et al. (2015) both represent the whole input document as a graph with synset relations as edges and jointly disambiguate nouns and verbs. In the case of Babelfy, a densest-subgraph heuristic is used to compute the high-coherence semantic interpretations of the text. Instead, in Weissenborn et al. (2015) a set of complementary objectives, which include sense probabilities and type classification, are combined together to perform WSD.

A number of systems make use of both unannotated data and synset relations. Both Tripodi and Pelillo (2017) and Camacho-Collados et al. (2016) make use of statistical information from unannotated data to weigh the relevance of nodes in a graph, which is then used to perform WSD. Rothe and Schütze (2017) use word embeddings as a starting point and then rely on the formal constraints in a lexical resource to create synset embeddings.

Recently, there has been a surge in WSD approaches that use unannotated data but do not consider synset relations. One example is provided by Iacobacci et al. (2016), who investigated the role of word embeddings as features in a WSD system. Four methods (concatenation, average, fractional decay, and exponential decay) are used to extract features from the sentential context using word embeddings. The features are then added to the default feature set of IMS (Zhong and Ng, 2010). Moreover, Raganato et al. (2017b) present a number of end-to-end neural WSD architectures. The best performing one is based on a bidirectional Long Short-Term Memory (BLSTM) with attention and two auxiliary loss functions (part-of-speech and the WordNet coarse-grained semantic labels). Melamud et al. (2016) also make use of unannotated data to train a BLSTM. The work by Yuan et al. (2016), which we consider in this paper,

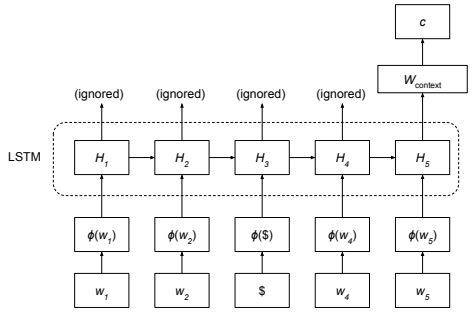


Figure 1: The LSTM model used to perform language modeling and compute context embeddings. At training time, a softmax layer is added, allowing it to predict the omitted word; at test time, the context embeddings are used for WSD in a nearest-neighbor or label-propagation procedure.

belongs to this last category. Different from Melamud et al. (2016), it uses significantly more unannotated data, the model contains more hidden units (2048 vs. 600), and the sense assignment is more elaborated. We describe this approach in more detail in the following section.

3 WSD with Language Models

The method proposed by Yuan et al. (2016) performs WSD by annotating each lemma in a text with one WordNet synset that is associated with its meaning. Broadly speaking, the disambiguation is done by: 1) constructing a language model from a large unannotated dataset; 2) extracting sense embeddings from this model using a much smaller annotated dataset; 3) relying on the sense embeddings to make predictions on the lemmas in unseen sentences. Each operation is described below.

Constructing Language Models. Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is a celebrated recurrent neural network architecture that has proven to be effective in many natural language processing tasks (Sutskever et al., 2014; Dyer et al., 2015; He et al., 2017, among others). Different from previous architectures, LSTM is equipped with trainable gates that control the flow of information, allowing the neural networks to learn both short- and long-range dependencies.

In Yuan et al. (2016), the first operation consists of constructing an LSTM language model to capture the meaning of words in context. They use an LSTM network with a single hidden layer of h nodes. Given a sentence $s = (w_1, w_2, \ldots, w_n)$, they replace word $w_k\,(1 \leq k \leq n)$ by a special token \$. The model takes this new sentence as input and produces a context vector $\mathbf{c}$ of dimensionality p (see Figure 1).[1]

Each word w in the vocabulary $\mathcal{V}$ is associated with an embedding $\phi_{\text{o}}(w)$ of the same dimensionality. The model is trained to predict the omitted word, minimizing the softmax loss over a big collection $\mathcal{D}$ of sentences.

$$\ell = -\sum_{s \in \mathcal{D}} \sum_{k=1}^{|s|} \log \frac{\exp(\mathbf{c} \cdot \phi_{\text{o}}(w_k))}{\sum_{w' \in \mathcal{V}} \exp(\mathbf{c} \cdot \phi_{\text{o}}(w'))}$$

After the model is trained, we can use it to extract *context* embeddings, i.e., latent numerical representations of the sentence surrounding a given word.

Calculating Sense Embeddings. The model produced by the LSTM network is meant to capture the "meaning" of words in the context they are mentioned. In order to perform the sense disambiguation, we need to extract from it a suitable representation for word senses. To this purpose, the method relies on another corpus where each word is annotated with the corresponding sense.

The main intuition is that words used with the same sense are mentioned in contexts which are very similar to each other as well. This suggests a simple way to calculate sense embeddings. First, the LSTM model is invoked to compute the context vector for each occurrence of one sense in the annotated dataset. Once all context vectors are computed, the sense embedding is defined as the average of all vectors. Let

[1] As usual, vectors are indicated with boldface to distinguish them to scalar and other symbols.

us assume, for instance, that the sense $horse_n^2$ (that is, the second sense of horse as a noun) appears in the two sentences:

(1) The move of the $horse_n^2$ to the corner forced the checkmate.

(2) Karjakin makes up for his lost bishop a few moves later, trading rooks and winning black's $horse_n^2$.

In this case, the method will replace the sense by \$ in the sentences and feed them to the trained LSTM model to calculate two context vectors $\mathbf{c_1}$ and $\mathbf{c_2}$. The sense embedding $\mathbf{s_{horse_n^2}}$ is then computed as:

$$\mathbf{s_{horse_n^2}} = \frac{\mathbf{c_1} + \mathbf{c_2}}{2}$$

This procedure is computed for every sense that appears in the annotated corpus.

Averaging technique to predict senses. After all sense embeddings are computed, the method is ready to disambiguate target words. This procedure proceeds as follows:

1. Given an input sentence and a target word, it replaces the occurrence of the target word by \$ and uses the LSTM model to predict a context vector $\mathbf{c_t}$.
2. The lemma of the target word is used to retrieve from WordNet the candidate synsets $s_1, \ldots, s_n$ where n is the number of synsets. Then, the procedure looks up the corresponding sense embeddings $\mathbf{s_1}, \ldots, \mathbf{s_n}$ computed in the previous step.
3. The procedure invokes a subroutine to choose one of the n senses for the context vector $\mathbf{c_t}$. It selects the sense whose vector is closest to $\mathbf{c_t}$ using cosine as the similarity function.

Label Propagation. Yuan et al. (2016) argue that the averaging procedure is suboptimal because of two reasons. First, the distribution of occurrences of senses is unknown whereas averaging is only suitable for spherical clusters. Second, averaging reduces the representation of occurrences of each sense to a single vector and therefore ignores sense prior. For this reason, they propose to use label propagation for inference as an alternative to averaging. Label propagation (Zhu and Ghahramani, 2002) is a classic semi-supervised algorithm that has been employed in WSD (Niu et al., 2005) and other NLP tasks (Chen et al., 2006; Zhou, 2011). The procedure involves predicting senses for not only the target cases but also for unannotated words queried from a corpus. It represents both the target cases and unannotated words as points in a vector space and iteratively propagates classification labels from the target classes to the words. In this way, it can be used to construct non-spherical clusters and to give more influence to frequent senses.

Overall algorithm. The overall disambiguation procedure that we implemented proceeds as follows:

1. *Monosemous:* First, the WSD algorithm checks whether the target lemma is monosemous (i.e., there is only one synset). In this case, the disambiguation is trivial.
2. *Label propagation:* If the label propagation is enabled, then it checks whether the target lemma occurs at least once in the annotated dataset and at least once in the auxiliary unannotated dataset. In this case, the procedure applies the *label propagation* technique for selecting the candidate synset.
3. *Averaging:* If the previous strategies are not applicable and there is at least one occurrence of the target lemma in the annotated dataset, then we apply the *averaging* technique for selecting the candidate synset.
4. *MFS fallback:* If the target lemma does not appear in the annotated dataset, then the system picks the most-frequent synset.[2]

[2]Notice that Yuan et al. (2016) did not report if they use an MFS-fallback strategy or simply returned no answer.

4 Reproduction Study: Methodology

Before we report the results of our experiments, we describe the datasets used and give some details regarding our implementation.

Training data. The 100-billion-token corpus used in the original publication is not publicly available. Therefore, for the training of the LSTM models, we used the English Gigaword Fifth Edition (Linguistic Data Consortium (LDC) catalog number LDC2011T07). The corpus consists of 1.8 billion tokens in 4.1 million documents, originated from four major news agencies. We leave the study of bigger corpora for future work.

For the training of the sense embeddings, we use the same two corpora used by Yuan et al. (2016):

1. *SemCor* (Miller et al., 1993) is a corpus containing approximately 240,000 sense annotated words. The tagged documents originate from the Brown corpus (Francis and Kucera, 1979) and cover various genres.
2. *OMSTI* (Taghipour and Ng, 2015) contains one million sense annotations automatically tagged by exploiting the English-Chinese part of the parallel MultiUN corpus (Eisele and Chen, 2010). A list of English translations were manually created for each WordNet sense. If the Chinese translation of an English word matches one of the manually curated translations for a WordNet sense, that sense is selected.

Implementation. We used the BeautifulSoup HTML parser to extract plain text from the Gigaword corpus. Then, we used the English models[3] of Spacy 1.8.2 for sentence boundary detection and tokenization. The LSTM model is implemented using TensorFlow 1.2.1 (Abadi et al., 2015). We chose TensorFlow because of its industrial-grade quality and because it can train large-scale models.

The main computational bottleneck of the entire process is the training of the LSTM model. Although we do not use a 100-billion-token corpus, training the model on Gigaword can already take years if not optimized properly. To reduce training time, we assumed that all (padded) sentences in the batch have the same length. This optimization increases the speed by 17% as measured on a smaller model ($h = 100, p = 10$). Second, following Yuan et al., we use the sampled softmax loss function (Jean et al., 2015). Third, we grouped sentences of similar length together while varying the number of sentences in a batch to fully utilize GPU RAM. Together, these heuristics increased training speed by 42 times.

Although Yuan et al. proposed to use a distributed implementation of label propagation (Ravi and Diao, 2015), we found that scikit-learn (Pedregosa et al., 2011) was fast enough for our experiments. For hyperparameter tuning, we use the annotations in OMSTI (which are not used at test time). After measuring the performance of some variations of label propagation (scikit-learn implementation: *LabelPropagation* or *LabelSpreading*; similarity measure: *inner product* or radial basis function with different values of γ), we found that the combination of *LabelSpreading* and *inner product* similarity leads to the best result which is also better than averaging on the development set.

Evaluation framework. For evaluating the WSD predictions, we selected two test sets: one from the Senseval2 (Palmer et al., 2001) competition, which tests the disambiguation of nouns, verbs, adjectives and adverbs, and one from the 2013 edition (Navigli et al., 2013), which focuses only on nouns.

The test set from Senseval-2 is the English All-Words Task; *senseval2* henceforth. This dataset contains 2,282 annotations from three articles from the Wall Street Journal. Most of the annotations are nominal, but the competition also contains annotations for verbs, adjectives, and adverbs. In this test set, 66.8% of all target words are annotated with the most-frequent sense (MFS) of the lemma. This means that the simple strategy of always selecting the MFS would score 66.8% F_1 on this dataset.

The test set from SemEval-2013 is the one taken from task 12: Multilingual Word Sense Disambiguation; *semeval2013* henceforth. This task consists of two disambiguation tasks: Entity Linking and Word Sense Disambiguation for English, German, French, Italian, and Spanish. This test set contains 13 articles from previous editions of the workshop on Statistical Machine Translation.[4] The articles contain

[3] `en_core_web_md-1.2.1`

[4] `http://www.statmt.org`

Model/Method	Datasets	Scorer	senseval2 F_1	semeval2013 F_1
Our LSTM	T: SemCor	framework	**0.720**	0.647
Our LSTM	T: OMSTI	framework	0.675	0.651
Our LSTM	T: SemCor+OMSTI	framework	0.699	**0.656**
Our LSTMLP	T: SemCor, U:OMSTI	framework	0.714	0.642
Yuan et al. (2016) LSTM	T: SemCor	mapping to WN3.0	0.736	0.670
Yuan et al. (2016) LSTM	T: OMSTI	mapping to WN3.0	0.724	0.673
Yuan et al. (2016) LSTMLP	T: SemCor, U: OMSTI	mapping to WN3.0	**0.739**	**0.679**
Raganato et al. (2017b)	T: SemCor	framework	0.720	0.669
Iacobacci et al. (2016) IMS+emb	T: SemCor	framework	0.710	0.673
Iacobacci et al. (2016) IMS+emb	T: SemCor+OMSTI	framework	0.708	0.663
Iacobacci et al. (2016) IMS-s+emb	T: SemCor	framework	0.722	0.659
Iacobacci et al. (2016) IMS-s+emb	T: SemCor+OMSTI	framework	**0.733**	0.667
Melamud et al. (2016)	T: SemCor	framework	0.718	0.656
Melamud et al. (2016)	T: SemCor+OMSTI	framework	0.723	0.672
Agirre et al. (2014) UKB-g*	P: SemCor	framework	0.688	0.688
Weissenborn et al. (2015)	P: SemCor	competition	-	**0.728**

Table 1: Performance of our implementation compared to already published results. We report the model/method used to perform WSD, the used annotated dataset and scorer, and F_1 for each test set. In the naming of our models, *LSTM* indicates that the *averaging* technique was used for the sense assignment, while *LSTMLP* refers to the results obtained using *label propagation* (see Section 3). The datasets following *T:* indicate the annotated corpus used to represent the senses while *U:OMSTI* stands for using OMSTI as unlabeled sentences in case label propagation is used. *P: SemCor* indicates that sense distributions from SemCor are used in the system architecture. Three scorers are used: *"framework"* refers to the WSD evaluation framework from Raganato et al. (2017a); *"mapping to WN3.0"* refers to the evaluation used by Yuan et al. (2016) while *"competition"* refers to the scorer provided by the competition itself (e.g., semeval2013).

1,644 test instances in total, which are all nouns. The application of the MFS baseline on this dataset yields an F_1 score of 63.0%.

5 Results

In this section, we report our reproduction of the results of Yuan et al. (2016) and additional experiments to gain a deeper insight into the strengths and weaknesses of the approach. These experiments focus on the performance on the most- and less-frequent senses, coverage of the annotated dataset and the consequent impact on the overall predictions, the granularity of the sense representation, and the impact of the unannotated data and model complexity on the accuracy of WSD.

Reproduction results. We trained the LSTM model with the best reported settings in Yuan et al. (2016) (hidden layer size $h = 2048$, embedding dimensionality $p = 512$) using a machine equipped with an Intel Xeon E5-2650, 256GB of RAM, 8TB of disk space, and two nVIDIA GeForce GTX 1080 Ti GPUs. During our training, one epoch took about one day to finish with TensorFlow fully utilizing one GPU. The whole training process took four months. We tested the performance of the downstream WSD task three times during the training and observed that the best performance is obtained at the 65$^{\text{th}}$ epoch, despite a later model producing a lower negative log-likelihood. Thus, we used the model produced at the 65$^{\text{th}}$ epoch for our experiments below.

Table 1 presents the results using the test sets *senseval2* and *semeval2013*, respectively. The top part of the table presents our reproduction results, the middle part reports the results from Yuan et al. (2016), while the bottom part reports a representative sample of the other state-of-the-art approaches.

It should be noted that with the test set *semeval2013*, all scorers use WordNet 3.0, therefore the performance of the various methods can be directly compared. However, not all answers in *senseval2* can be mapped to WN3.0 and we do not know how Yuan et al. (2016) handled these cases. In the WSD evaluation framework (Moro et al., 2014) that we selected for evaluation, these cases were either re-annotated or removed. Thus, our F_1 on *senseval2* cannot be directly compared with the F_1 in the original paper.

From a first glance at Table 1, we observe that if we use SemCor to train the synset embeddings, then our results come close to the state-of-the-art on *senseval2* (0.720 vs. 0.733). On *semeval2013*, we achieve results comparable to other embeddings-based approaches (Raganato et al., 2017b; Iacobacci et

Model	senseval2			semeval2013		
	F_1	R_mfs (n=1524)	R_lfs (n=758)	F_1	R_mfs (n=1035)	R_lfs (n=609)
Our LSTM (T: SemCor)	**0.72**	0.88	**0.41**	0.65	0.84	0.33
Our LSTM (T: OMSTI)	0.67	0.87	0.27	0.65	**0.86**	0.29
Our LSTM (T: SemCor+OMSTI)	0.70	0.85	0.40	**0.66**	0.82	**0.38**
Our LSTMLP (T: SemCor+OMSTI)	0.71	**0.91**	0.32	0.64	0.85	0.29

Table 2: Performance of our implementation with respect to MFS and LFS recall. *R_mfs* and *R_lfs* are the recall on the most-frequent-sense and least-frequent-sense instances respectively. n represents the number of considered instances.

al., 2016; Melamud et al., 2016). However, the gap with the graph-based approach of Weissenborn et al. (2015) is still significant. When we use both SemCor and OMSTI for the annotated data, our results drop 0.02 point for *senseval2*, whereas they increase by almost 0.01 for *semeval2013*. Different from Yuan et al. (2016), we did not observe improvement by using label propagation (comparing *T: SemCor, U: OMSTI* against *T:SemCor* without propagation). However, the performance of the label propagation strategy is still competitive on both test sets.

Most- vs. less-frequent-sense instances. The original paper only analyses the performance on the whole test sets. We extend this analysis by looking at the performance for disambiguating the most frequent-sense (MFS) and less-frequent-sense (LFS) instances. The first type of instances are the ones for which the correct link is the most-frequent sense, whereas the second subset consists of the remaining ones. This analysis is important because it is well-known that the simple strategy of always choosing the MFS is a strong baseline in WSD, thus there is a tendency for WSD systems to overfit towards the MFS (Postma et al., 2016).

Table 2 shows that the method by Yuan et al. (2016) does not overfit towards the MFS to the same extent as other supervised systems since the recall on LFS instances is still quite high 0.41 (a lower recall on LFS instances than on MFS ones is expected due to the reduced training data for them).

On semeval13, the recall on LFS is already relatively high using only SemCor (0.33), and reaches 0.38 when using both SemCor and OMSTI. For comparison, the default system IMS (Zhong and Ng, 2010) trained on SemCor only obtains an *R_lfs* of 0.15 on semeval13 (Postma et al., 2016) and only reaches 0.33 with a large amount of annotated data.

Finally, our implementation of the label propagation does seem to slightly overfit towards the MFS. When we compare the results of the *averaging technique* using SemCor and OMSTI versus when we use *label propagation*, we notice an increase in the MFS recall (from 0.85 to 0.91), whereas the LFS recall drops from 0.40 to 0.32.

Meaning coverage in annotated datasets. The WSD procedure depends on an annotated corpus to compose its sense representations, making missing annotations an insurmountable obstacle. In fact, annotated datasets only contain annotations for a proper subset of the possible candidate synsets listed in WordNet. We analyze this phenomenon using four statistics:

1. *Candidate Coverage:* For each test set, we performed a lookup in WordNet to determine the unique candidate synsets of all target lemmas. We then determined what percentage of these candidate synsets that have *at least one* annotation in the annotated dataset.
2. *Lemma Coverage:* Given a target lemma in a test set, we performed a lookup in WordNet to determine the unique candidate synsets. If *all* candidate synsets of that target lemma have at least one annotation in the annotated dataset, we claim that the lemma is covered. The lemma coverage is then the percentage of all covered target lemmas. A high lemma coverage indicates that annotated dataset covers most of the meanings in the test set.
3. *Gold Coverage:* We calculate the percentage of the *correct* answers in the test set that has at least one annotation in the annotated dataset.

The column "Candidate Coverage" of Table 3 shows that SemCor only contains less than 70% of all candidate synsets for *senseval2* and *semeval2013*, meaning that a model will never have a representation

Datasets	senseval2 Candidate Coverage	senseval2 Lemma Coverage	senseval2 Gold Coverage	semeval2013 Candidate Coverage	semeval2013 Lemma Coverage	semeval2013 Gold Coverage
SemCor	63.51%	29.72%	80.92%	66.64%	28.42%	83.08%
OMSTI	29.98%	5.9%	43.82%	31.22%	5.94%	45.23%
SemCor+OMSTI	**66.26%**	**32.25%**	**81.98%**	**69.89%**	**31.83%**	**84.76%**

Table 3: Statistics about the coverage of annotated datasets in WordNet.

Competition	Model	MFS fallback	Averaging	Label propagation
senseval2	Our LSTM (T: SemCor)	0.64 (n=135)	0.66 (n=1712)	-
	Our LSTM (T: OMSTI)	0.66 (n=775)	0.56 (n=1072)	-
	Our LSTM (T: SemCor+OMSTI)	0.64 (n=135)	0.63 (n=1712)	-
	Our LSTMLP (T:SemCor, U:OMSTI)	0.64 (n=135)	0.71 (n=644)	0.61 (n=1068)
semeval2013	Our LSTM (T: SemCor)	0.39 (n=41)	0.56 (n=1262)	-
	Our LSTM (T: OMSTI)	0.58 (n=427)	0.55 (n=876)	-
	Our LSTM (T: SemCor+OMSTI)	0.40 (n=40)	0.57 (n=1263)	-
	Our LSTMLP (T:SemCor, U:OMSTI)	0.40 (n=40)	0.57 (n=397)	0.55 (n=866)

Table 4: The recall is shown for three WSD strategies and n indicates the number of instances it was actually used. The recall on all monosemous instances was 1.0 for both competitions and is hence not shown in the table. There are 435 monosemous instances in *senseval2* and 341 in *semeval2013*.

for more than 30% of the candidate synsets. Even with the addition of OMSTI, the coverage does not exceed 70%, meaning that we lack evidence for a significant number of potential annotations. Moreover, the column "Lemma Coverage" illustrates that we have evidence for all potential solutions for only 30% of the lemmas in both WSD competitions, meaning that in the large majority of the cases some solutions are never seen. The column "Gold coverage" measures whether the right answers are at least seen in the annotated dataset. The numbers illustrate that 20% of the solutions in the test sets do not have any annotations. With our approach, these answers can only be returned if the lemma is monosemous or by random guess otherwise.

To further investigate these issues, Table 4 reports the recall of the various disambiguation strategies which could be invoked depending on the coverage of the lemma (these can be: *monosemous*, *averaging*, *label propagation*, *MFS* – see the overall procedure reported in Section 3).

We observe that the MFS fallback plays a significant role in obtaining the overall high accuracy since it is invoked many times, especially with OMSTI due to the low coverage of the dataset (in this case it is invoked in 775 cases vs. 1072 of averaging). For example, if we had not applied the MFS fallback strategy for *senseval2* using SemCor as the annotated corpus, our performance would have dropped from 0.72 to 0.66, below the MFS baseline of 0.67 for this task.[5] Label propagation was indeed applied on half of the cases, but leads to lower results. From these results, we learn that the effectiveness of this method strongly depends on the coverage of the annotated datasets: If it is not high, as it is with OMSTI, then the performance of this method reduces to the one of choosing the MFS.

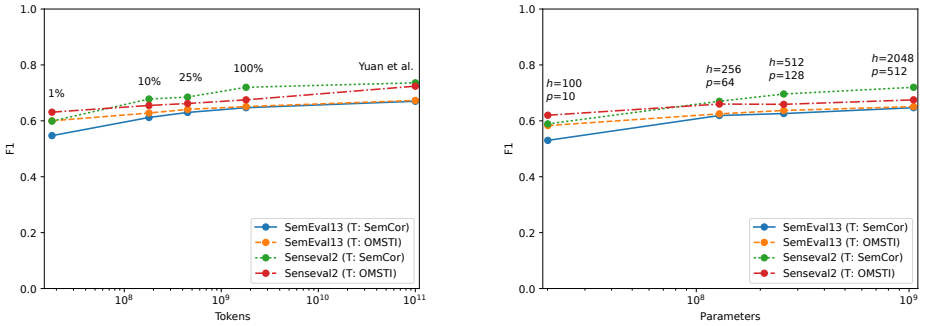
Granularity of sense representation. Rothe and Schütze (2017) provided evidence for the claim that the granularity of the sense representations has an influence on WSD performance. More in particular, their WSD system performed better when trained on sensekeys (called *lexemes* in their paper) than on synsets. Although a sensekey-based disambiguation results in less annotated data per target lemma, the sensekey representation is more precise (since it is a lemma associated with a particular meaning) than at the synset level.

The reimplementation discussed in this paper allows us to answer the question: "How will LSTM models work if we lower the disambiguation level from synset to sensekey?" Table 5 presents the results of this experiment. As we can see from the table, our method also returns better performance on both test sets. This behavior is interesting and one possible explanation is that sensekeys are more discriminative than synsets and this favors the disambiguation.

[5] *senseval2* contains 2,282 instances, of which the system would answer incorrectly 135 instances if the MFS fallback strategy is not used, hence dropping 0.06 in performance.

Model	senseval2		semeval2013	
	sensekey F_1	synset F_1	sensekey F_1	synset F_1
Our LSTM (T: SemCor)	**0.726**	**0.720**	0.661	0.647
Our LSTM (T: OMSTI)	0.688	0.675	0.655	0.651
Our LSTM (T: SemCor+OMSTI)	0.703	0.699	**0.667**	**0.656**

Table 5: Comparison of F_1 scores of our implementation using either synset or sensekey level to represent meanings.



(a) Performance vs. unannotated corpus size

(b) Performance vs. number of parameters

Figure 2: The effect of (a) the size of unannotated corpus and (b) the number of parameters on WSD performance. Number of parameters includes the weights of the hidden layer, the weights of the projection layer, and the input and output embeddings. Notice that the horizontal axis is in log scale.

Impact of unannotated data and model size. Since unannotated data is abundant, it is tempting to use more and more data to train language models, hoping that better word embeddings would translate into improved WSD performance. The fact that Yuan et al. (2016) used a 100-billion-token corpus only reinforces this intuition. We empirically evaluate the effectiveness of unlabeled data by varying the size of the corpus used to train LSTM models and measure the corresponding WSD performance. More in particular, the size of the training data was set at 1%, 10%, 25%, and 100% of the GigaWord corpus (which contains 1.8×10^7, 1.8×10^8, 4.5×10^8 and 1.8×10^9 words, respectively).

Figure 2a shows the effect of unannotated data volume on WSD performance. The data points at 100 billion (10^{11}) tokens correspond to Yuan et al. (2016)'s reported results. As might be expected, a bigger corpus leads to more meaningful context vectors and therefore higher performance on WSD. However, the amount of data needed for 1% of improvement in F_1 grows exponentially fast (notice that the horizontal axis is in log scale). Extrapolating from this graph, to get a performance of 0.8 F_1 by adding more unannotated data, one would need a corpus of 10^{12} tokens. This observation also applies to the balance of the sense assignment. Using only 25% of the unannotated data already yields a recall of 35% on the less-frequent senses.

In addition, one might expect to push the performance further by increasing the capacity of the LSTM models. To evaluate this possibility, we performed an experiment in which we varied the sizes of LSTM models trained on 100% of the GigaWord corpus and evaluated against *senseval2* and *semeval2013*, respectively. Figure 2b suggests that it is possible but one would need exponentially bigger models.

Finally, Reimers and Gurevych (2017) have showed that it is crucial to report the distribution of test scores instead of only one score as this practice might lead to wrong conclusions. As pointed out at the beginning of Section 5, our biggest models take months to train, making training multiple versions of them impractical. However, we trained our smallest model ($h = 100, p = 10$) ten times and our second smallest model ($h = 256, p = 64$) five times and observed that as the number of parameters increased, the standard deviation of F_1 decreased from 0.008 to 0.003. We, therefore, believe random fluctuation does not affect the interpretation of the results.

6 Conclusions

This paper reports the results of a reproduction study of the model proposed by Yuan et al. (2016) and an additional analysis to gain a deeper understanding of the impact of various factors on its performance.

A number of interesting conclusions can be drawn from our results. First, we observed that we do not need a very large unannotated dataset to achieve state-of-the-art all-words WSD performance since we used the Gigaword corpus, which is two orders of magnitude smaller than Yuan et al. (2016)'s proprietary corpus, and got similar performance on *senseval2* and *semeval2013*. A more detailed analysis hints that adding more unannotated data and increasing model capacity are subject to diminishing returns. Moreover, we observed that this approach has a more balanced sense assignment than other techniques, as shown by the relatively good performance on less-frequent-sense instances. In addition, we identified that the limited sense coverage in annotated dataset places a potentially upper bound for the overall performance.

The code with detailed replication instructions is available at: `https://github.com/cltl/wsd-dynamic-sense-vector` and the trained models at: `https://figshare.com/articles/A_Deep_Dive_into_Word_Sense_Disambiguation_with_LSTM/6352964`.

Acknowledgements

The research for this paper was supported by the Netherlands Organisation for Scientific Research (NWO) via the Spinoza-prize Vossen projects (SPI 30-673, 2014-2019). We thank the support of the NWO project scilens (`https://projects.cwi.nl/scilens`) for providing the hardware to run some of the experiments. Experiments were also carried out on the Dutch national e-infrastructure with the support of SURF Cooperative. We would like to thank our friends and colleagues Antske Fokkens, Emiel van Miltenburg, Chantal van Son, Pia Sommerauer, and Roxane Segers for many useful comments and discussions.

References

Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Man, Rajat Monga, Sherry Moore, Derek Murray, Jon Shlens, Benoit Steiner, Ilya Sutskever, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Oriol Vinyals, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467v2*.

Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence*, 240:36–64.

Jinxiu Chen, Donghong Ji, Chew Lim Tan, and Zhengyu Niu. 2006. Relation extraction using label propagation based semi-supervised learning. In *ACL 2016*, pages 129–136.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack Long Short-Term Memory. In *ACL 2015*, pages 334–343.

Andreas Eisele and Yu Chen. 2010. MultiUN: A multilingual corpus from united nation documents. In *LREC 2010*.

Christiane Fellbaum, editor. 1998. *WordNet An Electronic Lexical Database*. The MIT Press, Cambridge, MA ; London, May.

Winthrop Nelson Francis and Henry Kucera. 1979. Brown corpus manual. *Brown University*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. *ACL 2017*.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *ACL 2016*, pages 897–907. Association for Computational Linguistics.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *ACL-IJCNLP 2015*, pages 1–10.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. Context2vec: Learning generic context embedding with bidirectional LSTM. In *CoNLL*, pages 51–61.

George Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Human language technology: Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *TACL 2014*, 2:231–244.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. SemEval-2013 task 12: Multilingual Word Sense Disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 222–231, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10.

Zheng-Yu Niu, Dong-Hong Ji, and Chew Lim Tan. 2005. Word Sense Disambiguation Using Label Propagation Based Semi-supervised Learning. In *ACL 2005*, pages 395–402, Stroudsburg, PA, USA. Association for Computational Linguistics.

Martha Palmer, Christiane Fellbaum, Scott Cotton, Lauren Delfs, and Hoa Trang Dang. 2001. English tasks: All-words and verb lexical sample. In *Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems (SENSEVAL-2)*, pages 21–24, Toulouse, France, July. Association for Computational Linguistics.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Marten Postma, Ruben Izquierdo Bevia, and Piek Vossen. 2016. More is not always better: balancing sense distributions for all-words word sense disambiguation. In *COLING 2016: Technical Papers*, pages 3496–3506.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017a. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *EACL 2017*, pages 99–110. Association for Computational Linguistics.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017b. Neural sequence learning models for word sense disambiguation. In *EMNLP 2017*, pages 1156–1167. Association for Computational Linguistics.

Sujith Ravi and Qiming Diao. 2015. Large Scale Distributed Semi-Supervised Learning Using Streaming Approximation. *arXiv:1512.01752 [cs]*, 51.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *EMNLP 2017*, pages 338–348.

Sascha Rothe and Hinrich Schütze. 2017. Autoextend: Combining word embeddings with semantic resources. *Computational Linguistics*, 43(3):593–617.

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

Kaveh Taghipour and Hwee Tou Ng. 2015. One million sense-tagged instances for word sense disambiguation and induction. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 338–344. Association for Computational Linguistics.

Rocco Tripodi and Marcello Pelillo. 2017. A game-theoretic approach to word sense disambiguation. *Computational Linguistics*, 43(1):31–70.

Dirk Weissenborn, Leonhard Hennig, Feiyu Xu, and Hans Uszkoreit. 2015. Multi-objective optimization for the joint disambiguation of nouns and named entities. In *ACL 2015*, pages 596–605. Association for Computational Linguistics.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. In *COLING 2016: Technical Papers*, pages 1374–1385. The COLING 2016 Organizing Committee.

Zhi Zhong and Hwee Tou Ng. 2010. It Makes Sense: A wide-coverage word sense disambiguation system for free text. In *ACL 2010: System Demonstrations*, pages 78–83. Association for Computational Linguistics.

Guo-Dong Zhou. 2011. Learning noun phrase anaphoricity in coreference resolution via label propagation. *Journal of Computer Science and Technology*, 26(1):34–44.

Xiaojin Zhu and Z Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. *CMU CALD tech report*, (CMU-CALD-02-107).

Enriching Word Embeddings with Domain Knowledge for Readability Assessment

Zhiwei Jiang and **Qing Gu*** and **Yafeng Yin** and **Daoxu Chen**
State Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210023, China
jiangzhiwei@outlook.com, {guq,yafeng,cdx}@nju.edu.cn

Abstract

In this paper, we present a method which learns the word embedding for readability assessment. For the existing word embedding models, they typically focus on the syntactic or semantic relations of words, while ignoring the reading difficulty, thus they may not be suitable for readability assessment. Hence, we provide the knowledge-enriched word embedding (KEWE), which encodes the knowledge on reading difficulty into the representation of words. Specifically, we extract the knowledge on word-level difficulty from three perspectives to construct a knowledge graph, and develop two word embedding models to incorporate the difficulty context derived from the knowledge graph to define the loss functions. Experiments are designed to apply KEWE for readability assessment on both English and Chinese datasets, and the results demonstrate both effectiveness and potential of KEWE.

1 Introduction

Readability assessment is a classic problem in natural language processing, which attracts many researchers' attention in recent years (Todirascu et al., 2016; Schumacher et al., 2016; Cha et al., 2017). The objective is to evaluate the readability of texts by levels or scores. The majority of recent readability assessment methods are based on the framework of supervised learning (Schwarm and Ostendorf, 2005) and build classifiers from hand-crafted features extracted from the texts. The performance of these methods depends on designing effective features to build high-quality classifiers.

Designing hand-crafted features are essential but labor-intensive. It is desirable to learn representative features from the texts automatically. For document-level readability assessment, an effective feature learning method is to construct the representation of documents by combining the representation of the words contained (Kim, 2014). For the representation of word, a useful technique is to learn the word representation as a dense and low-dimensional vector, which is called word embedding. Existing word embedding models (Collobert et al., 2011; Mikolov et al., 2013; Pennington et al., 2014) can be used for readability assessment, but the effectiveness is compromised by the fact that these models typically focus on the syntactic or semantic relations of words, while ignoring the reading difficulty. As a result, words with similar functions or topics, such as "man" and "gentleman", are mapped into close vectors although their reading difficulties are different. It calls for incorporating the knowledge on reading difficulty when training the word embedding.

In this paper, we provide the knowledge-enriched word embedding (KEWE) for readability assessment, which encodes the knowledge on reading difficulty into the representation of words. Specifically, we define the word-level difficulty from three perspectives, and use the extracted knowledge to construct a knowledge graph. After that, we derive the difficulty context of words from the knowledge graph, and develop two word embedding models to incorporate the difficulty context to define the loss functions.

We apply KEWE for document-level readability assessment under the supervised framework. The experiments are conducted on four datasets of either English or Chinese. The results demonstrate that

*Corresponding Author

Proceedings of the 27th International Conference on Computational Linguistics, pages 366–378
Santa Fe, New Mexico, USA, August 20-26, 2018.

our method can outperform other well-known readability assessment methods, and the classic text-based word embedding models on all the datasets. By concatenating our knowledge-enriched word embedding with the hand-crafted features, the performance can be further improved.

The rest of the paper is organized as follows. Section 2 provides the related work for readability assessment. Section 3 describes the details of KEWE. Section 4 presents the experiments and results. Finally Section 5 concludes the paper with future work.

2 Related Work

In this section, we briefly introduce three research topics relevant to our work: readability assessment, word embedding, and graph embedding.

Readability Assessment. The researches on readability assessment have a relatively long history from the beginning of last century (Collinsthompson, 2014). Early studies mainly focused on designing readability formulas to evaluate the reading scores of texts. Some of the well-known readability formulas include the SMOG formula (McLaughlin, 1969), the FK formula (Kincaid et al., 1975), and the Dale-Chall formula (Chall, 1995). At the beginning of the 21th century, supervised approaches have been introduced and then explored for readability assessment (Si and Callan, 2001; Collins-Thompson and Callan, 2004; Schwarm and Ostendorf, 2005). Researchers have focused on improving the performance by designing highly effective features (Pitler and Nenkova, 2008; Heilman et al., 2008; Feng et al., 2010; Vajjala and Meurers, 2012) and employing effective classification models (Heilman et al., 2007; Kate et al., 2010; Ma et al., 2012; Jiang et al., 2015; Cha et al., 2017). While most studies are conducted for English, there are studies for other languages, such as French (François and Fairon, 2012), German (Hancke et al., 2012), Bangla (Sinha et al., 2014), Basque (Gonzalez-Dios et al., 2014), Chinese (Jiang et al., 2014), and Japanese (Wang and Andersen, 2016).

Word Embedding. Researchers have proposed various methods on word embedding, which mainly include two broad categories: neural network based methods (Bengio et al., 2003; Collobert et al., 2011; Mikolov et al., 2013) and co-occurrence matrix based methods (Turney and Pantel, 2010; Levy and Goldberg, 2014b; Pennington et al., 2014). Neural network based methods learn word embedding through training neural network models, which include NNLM (Bengio et al., 2003), C&W (Collobert and Weston, 2008), and word2vec (Mikolov et al., 2013). Co-occurrence matrix based methods learn word embedding based on the co-occurrence matrices, which include LSA (Deerwester, 1990), Implicit Matrix Factorization (Levy and Goldberg, 2014b), and GloVe (Pennington et al., 2014). Besides the general word embedding learning methods, researchers have also proposed methods to learn word embedding to include certain properties (Liu et al., 2015; Shen and Liu, 2016) or for certain domains (Tang et al., 2014; Ren et al., 2016; Alikaniotis et al., 2016; Wu et al., 2017).

Graph embedding. Graph embedding aims to learn continuous representations of the nodes or edges based on the structure of a graph. The graph embedding methods can be classified into three categories (Goyal and Ferrara, 2017): factorization based (Roweis and Saul, 2000; Belkin and Niyogi, 2001), random walk based (Perozzi et al., 2014; Grover and Leskovec, 2016), and deep learning based (Wang et al., 2016). Among them, the random walk based methods are easy to comprehend and can effectively reserve the centrality and similarity of the nodes. Deepwalks (Perozzi et al., 2014) and node2vec (Grover and Leskovec, 2016) are two representatives of the random walk based methods. The basic idea of Deepwalk is viewing random walk paths as sentences, and feeding them to a general word embedding model. node2vec is similar to Deepwalk, although it simulates a biased random walk over graphs, and often provides efficient random walk paths.

3 Learning Knowledge-Enriched Word Embedding for Readability Assessment

In this section, we present the details of Knowledge-Enriched Word Embedding (KEWE) for readability assessment. By incorporating the word-level readability knowledge, we extend the existing word embedding model and design two models with different learning structures. As shown in Figure 1, the above one is the knowledge-only word embedding model (KEWE_k) which only takes in the domain knowledge,

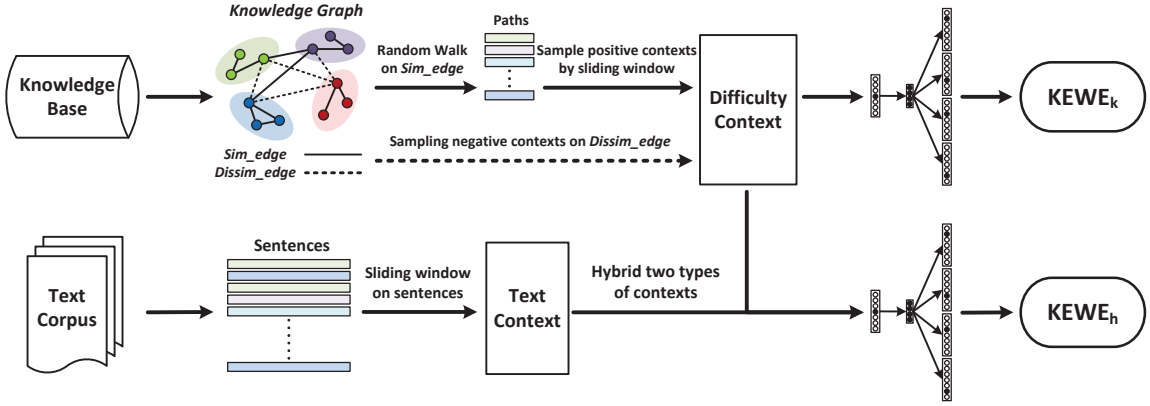


Figure 1: Illustration of the knowledge-enriched word embedding models. KEWE$_k$ is based on the difficulty context, while KEWE$_h$ is based on both the difficulty and text contexts.

the other is the hybrid word embedding model (KEWE$_h$) which compensates the domain knowledge with text corpus.

3.1 The Knowledge-only Word Embedding Model (KEWE$_k$)

In the classic word embedding models, such as C&W, CBOW, and Skip-Gram, the context of a word is represented by its surrounding words in the text corpus. Levy and Goldberg (2014a) have incorporated the syntactic context from the dependency parse-trees and found that the trained word embedding could capture more functional and less topical similarity. For readability assessment, reading difficulty other than function or topic becomes more important. Hence, we introduce a kind of difficulty context, and try to learn a difficulty-focusing word embedding, which leads to KEWE$_k$. In the following, we describe this model in three steps: domain knowledge extraction, knowledge graph construction, and graph-based word embedding learning. The former two steps focus on modeling the relationship among words on reading difficulty, and the final step on deriving the difficulty context and learning the word embedding.

3.1.1 Domain Knowledge Extraction

To model the relationship among words on reading difficulty, we first introduce how to extract the knowledge on word-level difficulty from different perspectives. Specifically, we consider three types of word-level difficulty: acquisition difficulty, usage difficulty, and structure difficulty.

Acquisition difficulty. Word acquisition refers to the temporal stage at which children learn the meaning of new words. Researchers have shown that the information on word acquisition is useful for readability assessment (Kidwell et al., 2009; Schumacher et al., 2016). Generally, the words acquired at primary school are easier than those acquired at high school. We call the reading difficulty reflected by word acquisition as the acquisition difficulty. Formally, given a word w, its acquisition difficulty is described by a distribution K_w^A over the age-of-acquisition (AoA) (Kidwell et al., 2009).

Since the rating on AoA is an unsolved problem in cognitive science (Brysbaert and Biemiller, 2016) and not available for many languages, we explore extra materials to describe the acquisition difficulty. In particular, we collect three kinds of knowledge teaching materials, i.e., in-class teaching material, extra-curricular teaching material, and proficiency test material. These materials are arranged as lists of words, each of which contains words learned in the same time period and hence corresponds to a certain level of acquisition difficulty. For example, given a lists of words, we can define $K_w^A \in \mathbb{R}^a$, where $K_{w,i}^A = 1$ if a word w belongs to the list i, and $K_{w,i}^A = 0$ otherwise.

Usage difficulty. Researchers used to count the usage frequency to measure the difficulty of words (Dale and Chall, 1948), which can separate the words which are frequently used from those rarely used. We call the difficulty reflected by usage preference as the usage difficulty. Formally, given a word w, its usage difficulty is described by a distribution K_w^U over the usage preferences.

We provide two ways to measure the usage difficulty. One way is estimating the level of words'

usage frequency by counting the word frequency lists from the text corpus. The other way is estimating the probability distribution of words over the sentence-level difficulties, which is motivated by Jiang et al. (2015). Usage difficulty is defined on both. By discretizing the range of word frequency into b intervals of equal size, the usage frequency level of a word w is i, if its frequency resides in the ith intervals. By estimating the probability distribution vector P_w from sentence-level difficulties, we can define $K_w^U \in \mathbb{R}^{1+|P_w|}$, and $K_{w,i}^U = [i, P_w]$.

Structure difficulty. When building readability formulas, researchers have found that the structure of words could imply its difficulty (Flesch, 1948; Gunning, 1952; McLaughlin, 1969). For example, words with more syllables are usually more difficult than words with less syllables. We call the difficulty reflected by structure of words as the structure difficulty. Formally, given a word w, its structure difficulty can be described by a distribution K_w^S over the word structures.

Words in different languages may have their own special structural characteristics. For example, in English, the structural characteristics of words relate to syllables, characters, affixes, and subwords. Whereas in Chinese, the structural characteristics of words relate to strokes and radicals of Chinese characters. Here we use the number of syllables (strokes for Chinese) and characters in a word w to describe its structure difficulty. By discretizing the range of each number into intervals, K_w^S is obtained by counting the interval in which w resides, respectively.

3.1.2 Knowledge Graph Construction

After extracting the domain knowledge on word-level difficulty, we then quantitatively represent the knowledge by a graph. We define the knowledge graph as an undirected graph $G = (V, E)$, where V is the set of vertices, each of which represents a word, and E is the set of edges, each of which represents the relation (i.e., similarity) between two words on difficulty. Each edge $e \in E$ is a vertex pair (w_i, w_j) and is associated with a weight z_{ij}, which indicates the strength of the relation. If no edge exists between w_i and w_j, the weight $z_{ij} = 0$. We define two edge types in the graph: *Sim_edge* and *Dissim_edge*. The former indicates that its end words have similar difficulty and is associated with a positive weight. The latter indicates that its end words have significant different difficulty and is associated with a negative weight. We derived the edges from the similarities computed between pairs of the words' knowledge vectors. Formally, given the extracted knowledge vector $K_w = [K_w^A, K_w^U, K_w^S]$ of a word w, E can be constructed using the similarity between pairs of words (w_i, w_j) as follows:

$$z_{ij} = \begin{cases} sim(K_{w_i}, K_{w_j}) & w_j \in \mathcal{N}_p(w_i) \\ -sim(K_{w_i}, K_{w_j}) & w_j \in \mathcal{N}_n(w_i) \\ 0 & otherwise \end{cases} \quad (1)$$

where $sim()$ is a similarity function (e.g., cosine similarity), $\mathcal{N}_p(w_i)$ refers to the set of k most similar (i.e., greatest similarity) neighbors of w_i, and $\mathcal{N}_n(w_i)$ refers to the set of k most dissimilar (i.e., least similarity) neighbors of w_i.

3.1.3 Knowledge Graph-based Word Embedding

After constructing the knowledge graph, which models the relationship among words on difficulty, we can derive the difficulty context from the graph and train the word embedding focused on reading difficulty. For the graph-based difficulty context, given a word w, we define its difficulty context as the set of other words that have relevance to w on difficulty. Specifically, we define two types of difficulty context, positive context and negative context, corresponding to the two types of edges in the knowledge graph (i.e., *Sim_edge* and *Dissim_edge*).

Unlike the context defined on texts, which can be sampled by sliding windows over consecutive words, the context defined on a graph requires special sampling strategies. Different sampling strategies may define the context differently. For difficulty context, we design two relatively intuitive strategies, the random walk strategy and the immediate neighbors strategy, for the sampling of either positive or negative context.

From the type *Sim_edge*, we sample the positive target-context pairs where the target word and the context words are similar on difficulty. Since the similarity is generally transitive, we adopt the random

walk strategy to sample the positive context. Following the idea of node2vec (Grover and Leskovec, 2016), we sample the positive contexts of words by simulating a 2^{nd} order random walk on the knowledge graph with only *Sim_edge*. After that, by applying a sliding window of fixed length s over the sampled random walk paths, we can get the positive target-context pairs $\{(w_t, w_c)\}$.

From the type *Dissim_edge*, we sample the negative target-context pairs where the target word and the context words are dissimilar on difficulty. Since dissimilarity is generally not transitive, we adopt the immediate neighbor strategy to sample the negative context. Specifically, on the knowledge graph with only *Dissim_edge*, we collect the negative context from the immediate neighbors of the target node w_t and get the negative context list $C_n(w_t)$.

By replacing the text-based linear context with our graph-based difficulty context, we can train the word embedding using the classic word embedding models, such as C&W, CBOW, and Skip-Gram. Here we use the Skip-Gram model with Negative Sampling (SGNS) proposed by Mikolov et al. (2013). Specifically, given N positive target-context pairs (w_t, w_c) and the negative context list of the target word $C_n(w_t)$, the objective of KEWE$_k$ is to minimize the loss function $\mathcal{L}_k$, which is defined as follows:

$$\mathcal{L}_k = -\frac{1}{N} \sum_{(w_t, w_c)} \Big[\log \sigma(\mathbf{u}_{w_c}^\top \mathbf{v}_{w_t}) + \mathbb{E}_{w_i \in C_n(w_t)} \log \sigma(-\mathbf{u}_{w_i}^\top \mathbf{v}_{w_t}) \Big] \tag{2}$$

where $\mathbf{v}_w$ and $\mathbf{u}_w$ are the "input" and "output" vector representation of w, and σ is the sigmoid function defined as $\sigma(x) = \frac{1}{(1+e^{-x})}$. This loss function enables the positive context (e.g., w_c) to be distinguished from the negative context (e.g., w_i).

3.2 The Hybrid Word Embedding Model (KEWE$_h$)

The classic text-based word embedding models yield word embedding focusing on syntactic and semantic contexts, while ignoring the word difficulty. By contrast, KEWE$_k$ trains the word embedding focusing on the word difficulty, while leaving out the syntactic and semantic information. Since readability may also relate to both syntax and semantics, we develop a hybrid word embedding model (KEWE$_h$), to incorporate both domain knowledge and text corpus. The loss function of the hybrid model $\mathcal{L}_h$ can be expressed as follows:

$$\mathcal{L}_h = \lambda \mathcal{L}_k + (1 - \lambda) \mathcal{L}_t \tag{3}$$

where $\mathcal{L}_k$ is the loss of predicting the knowledge graph-based difficulty contexts, $\mathcal{L}_t$ is the loss of predicting the text-based syntactic and semantic contexts, and $\lambda \in [0, 1]$ is a weighting factor. Clearly, the case of $\lambda = 1$ reduces the hybrid model to the knowledge-only model.

As there are many text-based word embedding models, the text-based loss $\mathcal{L}_t$ can be defined in various ways. To be consistent with KEWE$_k$, we formalize $\mathcal{L}_t$ based on the Skip-Gram model. Given a text corpus, the Skip-Gram model aims to find word representations that are good at predicting the context words. Specifically, given a sequence of training words, denoted as $w_1, w_2, \cdots, w_T$, the objective of Skip-Gram model is to minimize the log loss of predicting the context using target word embedding, which can be expressed as follows:

$$\mathcal{L}_t = -\frac{1}{T} \sum_{t=1}^{T} \sum_{-s \le j \le s, j \neq 0} \log p(w_{t+j} | w_t) \tag{4}$$

where s is the window size of the context sampling. Since the full softmax function used to define $p(w_{t+j}|w_t)$ is computationally expensive, we employ the negative sampling strategy (Mikolov et al., 2013) and replace every $\log p(w_c|w_t)$ in $\mathcal{L}_t$ by the following formula:

$$\log p(w_c | w_t) = \log \sigma(\mathbf{u}_{w_c}^\top \mathbf{v}_{w_t}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \log \sigma(-\mathbf{u}_{w_i}^\top \mathbf{v}_{w_t}) \tag{5}$$

where $\mathbf{v}_w$, $\mathbf{u}_w$, and σ are of the same meanings as in Eq. 2, k is the number of negative samples, and $P_n(w)$ is the noise distribution. This strategy enables the actual context w_c to be distinguished from the noise context w_i drawn from the noise distribution $P_n(w)$.

3.3 Model Training

We adopt the stochastic gradient descent to train our models. Specifically, for the hybrid model (KEWE$_h$), we adopt the mini-batch mode used in (Yang et al., 2016). Firstly, we sample a batch of random walk paths of size N_1 and take a gradient step to optimize the loss function $\mathcal{L}_k$. Secondly, we sample a batch of text sentences of size N_2 and take a gradient step to optimize the loss function $\mathcal{L}_t$. We repeat the above procedures until the model converged or the predefined number of iterations is reached. The ratio $\frac{N_1}{N_1+N_2}$ is used to approximate the weighting factor λ in Eq. 3. For training the knowledge-only model, the process is the same without $\mathcal{L}_t$ and λ.

3.4 Readability Assessment

We apply KEWE for document-level readability assessment under a supervised learning framework proposed by Schwarm and Ostendorf (2005). The classifier for readability assessment is built on documents with annotated reading levels. Instead of using the hand-crafted features, we use the word embedding to produce the features of documents.

To extract high level features from documents using word embedding, we design the max layer similar to the one used in the convolutional sentence approach proposed by Collobert et al. (Collobert et al., 2011). The max layer is used to generate a fix-size feature vector from variant length sequences. Specifically, given a document represented by a matrix $M \in \mathbb{R}^{m \times n}$, where the kth column is the word embedding of the kth word in the document, the max layer output a vector $f_{max}(M)$:

$$[f_{max}(M)]_i = \max_t [M]_{i,t} \qquad 1 \leq i \leq m \tag{6}$$

where $t = \{1, 2, \ldots, n\}$ represents "time" of a sequence, and m is the dimension of embedding vectors. Besides the max layer, the min and $average$ layers are also used to extract features. By concatenating all three feature vectors, we get the final feature vector $f(M)$ of the document M as follows, which can be fed to the classifier for readability assessment.

$$f(M) = [f_{max}(M), f_{min}(M), f_{avg}(M)] \tag{7}$$

4 Experiments

In this section, we conduct experiments based on four datasets of two languages, to investigate the following two research questions:

RQ1: Whether KEWE is effective for readability assessment, compared with other well-known readability assessment methods, and other word embedding models?

RQ2: What are the effects of the quality of input (i.e., the quality of the knowledge base and text corpus) and the hybrid ratio (i.e., the weighting factor λ) on the prediction performance of KEWE?

4.1 The Datasets and Knowledge Base

The experiments are conducted on four datasets, including two English datasets: ENCT and EHCT, two Chinese datasets: CPT and CPC. ENCT, CPT (Jiang et al., 2015), and EHCT are extracted from textbooks [1], where the documents have already been leveled into grades; CPC is extracted from the students' compositions [2] written by Chinese primary school students, where the documents are leveled by the six grades of authors. EHCT is collected from the English textbooks of Chinese high schools and colleges, which contains 4 levels corresponding to the 3 grades of high school plus undergraduates. The details of the four datasets are shown in Table 1.

Since the experiments are conducted on two languages, we collect the knowledge bases for both English and Chinese, which are used for extracting domain knowledge and constructing the knowledge graphs, as described in Section 3.1. The details are shown in Table 2.

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

[2] http://www.eduxiao.com

Langauage	Dataset	#Level	#doc	#sent/doc	#word/doc	|V|	#doc in reading levels
English	ENCT	4	276	16.92	266.62	7747	[72, 96, 60, 48]
	EHCT	4	252	40.82	646.70	10485	[67, 67, 58, 60]
Chinese	CPT	6	637	25.47	448.16	19254	[96, 110, 106, 108, 113, 104]
	CPC	6	300	14.91	305.16	9459	[50, 50, 50, 50, 50, 50]

Table 1: Statistics of the Four Datasets

Perspective	Type of Material	English	Chinese
Acqusition	AoA rating	AoA norms for over 50,000 English Words	–
	In-class teaching	New word list (primary, junior, high)	New word list (primary, junior, high)
	Extra-curricular teaching	New Concept English vocabulary	Overseas Chinese Language vocabulary
	Proficiency test	CET(4,6), PETS(1,2,5) vocabulary	HSK(1-6) vocabulary
Usage	Frequency List	Word Frequency List of American English	Chinese High Frequency Word List
	Sentence Corpus	English Wikipedia	Chinese Wikipedia
Structure	Dictionary	Syllabary	Stroke dictionary

Table 2: Details of the collected knowledge bases

4.2 Experiment Settings

For readability assessment, we design experiments on the four datasets using the hold-out validation, which randomly divides a dataset into training and test sets by stratified sampling. The test ratio is set as 0.3. To reduce randomness, under each case, 100 rounds of hold-out validations are performed, and the average results are reported. To tune the hyper-parameters, we randomly choose three-tenths of the training set as development set. We choose three widely used metrics (Jiang et al., 2014): Accuracy (Acc), Adjacent Accuracy ($\pm$Acc) and Pearson's Correlation Coefficient (PCC), to measure the performance on readability assessment.

For the training of word embedding, we use all the sentences in the target dataset as the training corpus (denote as the internal corpus), to ensure sufficient word coverage. To avoid mixing the level information into the internal corpus, we shuffle the sentences in it before feeding to the model. For the hyper-parameters of text-based word embedding, we set the dimension as 300, the window size as 5, the number of negative samples as 5, and the iteration number as 5. For KEWE, the default length of random walk path is set as 80 (Grover and Leskovec, 2016). k and λ are tuned using the development set.

4.3 Comparison with Other Methods

To address RQ1, we firstly compare KEWE with the following readability assessment methods:

- SMOG (McLaughlin, 1969) and FK (Kincaid et al., 1975) are two widely-used readability formulas.
- SUM (Collins-Thompson and Callan, 2004) is a smoothed unigram model.
- HCF is the hand-crafted feature based method. For English and Chinese, we employ the state-of-the-art feature set proposed by Vajjala and Meurers (2012) and Jiang et al. (2014) respectively.
- BOW refers to the bag-of-words model.

For the feature-based methods (i.e., HCF, BOW, and KEWE), we use both Logistic Regression (LR) and Random Forest (RF) as the classifiers for readability assessment . Table 3 lists the performance measures of all these methods on four datasets. The value marked in bold in each column refers to the maximum (best) measures acquired by the methods on each dataset by certain metric.

From Table 3, we can see that the performances of readability formulas (SMOG and FK) are not good on all the datasets, except the adjacent accuracies on ENCT. The smoothed unigram model (SUM) outperforms SOMG and FK on all the datasets, and on EHCT, its accuracy is only slightly inferior to KEWE$_h$. HCF performs the best among methods other than KEWE. Even compared with KEWE, it achieves the best performance on CPC and the best adjacent accuracy on ENCT. KEWE is only slightly inferior to HCF in four columns, but outperforms all the other methods in the other eight columns.

Method		Dataset											
		ENCT			EHCT			CPT			CPC		
		Acc	±Acc	PCC	Acc	±Acc	PCC	Acc	±Acc	PCC	Acc	±Acc	PCC
SMOG		46.04	94.13	0.7441	28.47	84.05	0.5517	30.95	70.21	0.6063	19.58	60.10	0.4076
FK		50.98	97.67	0.7870	25.17	76.16	0.3417	25.15	64.61	0.4912	16.93	50.49	0.0808
SUM		66.64	97.19	0.8345	70.41	91.41	0.7257	33.61	72.04	0.5866	26.71	56.59	0.2734
HCF	LR	87.24	98.01	0.9136	53.69	89.85	0.6937	43.71	81.69	0.7652	27.08	61.54	0.4275
	RF	90.96	**100**	0.9592	60.36	91.24	0.7421	47.97	87.99	0.8159	**35.09**	**72.49**	**0.5880**
BOW	LR	81.57	99.28	0.9049	65.88	89.61	0.7257	34.82	74.67	0.6593	29.01	56.71	0.3083
	RF	78.89	94.71	0.8294	59.03	89.96	0.7384	39.56	80.73	0.7486	31.13	62.74	0.5247
KEWE$_k$	LR	91.12	99.72	0.9545	64.03	91.43	0.7745	52.69	87.35	0.8091	30.59	61.32	0.5272
	RF	92.34	99.71	0.9606	69.13	95.28	0.8251	52.94	88.39	0.8167	30.73	66.14	0.5278
KEWE$_h$	LR	93.67	99.58	0.9654	65.25	93.40	0.8129	54.33	88.03	0.8233	34.81	65.11	0.5507
	RF	**94.48**	99.72	**0.9705**	**71.20**	**96.12**	**0.8449**	**54.83**	**88.94**	**0.8287**	33.26	65.63	0.5111
HCF+KEWE$_h$	LR	87.37	98.07	0.9153	53.96	90.05	0.6976	45.92	83.53	0.7876	27.31	61.84	0.4300
	RF	**95.87**	99.89	**0.9794**	70.05	95.85	0.8380	**60.23**	**92.28**	**0.8776**	**35.52**	70.24	0.5801

Table 3: Performance comparison between KEWE and other readability assessment methods

Model	Dataset											
	ENCT			EHCT			CPT			CPC		
	Acc	±Acc	PCC	Acc	±Acc	PCC	Acc	±Acc	PCC	Acc	±Acc	PCC
Random	31.73	75.57	0.0834	24.96	61.35	0.0786	16.96	45.96	0.0590	16.81	45.59	0.0743
NNLM	79.80	98.66	0.8843	58.43	88.36	0.7046	43.47	85.22	0.7806	30.68	65.07	0.5195
C&W	82.35	99.05	0.9039	59.05	90.06	0.7160	45.59	84.70	0.7874	30.60	64.83	0.5023
GloVe	82.66	99.52	0.9129	58.15	90.53	0.7302	43.63	85.80	0.7882	30.33	64.48	0.4933
CBOW	73.94	96.86	0.8299	62.40	91.29	0.7518	40.49	82.83	0.7564	27.40	59.48	0.3929
SG	82.20	98.60	0.9001	62.73	62.90	0.7563	44.31	85.54	0.7869	31.92	63.04	0.4527
KV	89.10	98.71	0.9295	64.15	90.85	0.7483	50.63	84.84	0.7806	28.19	61.90	0.4658
SG+KV	84.71	98.81	0.9136	66.00	92.73	0.7833	46.49	86.08	0.7921	32.03	63.34	0.4729
KEWE$_k$	92.34	99.71	0.9606	69.13	95.28	0.8251	52.94	88.39	0.8167	30.73	**66.14**	**0.5278**
KEWE$_h$	**94.48**	**99.72**	**0.9705**	**71.20**	**96.12**	**0.8449**	**54.83**	**88.94**	**0.8287**	**33.26**	65.63	0.5111

Table 4: Performance comparison between KEWE and other word embedding models

Overall, KEWE is competitive for readability assessment. In addition, by combining KEWE with the hand-crafted feature set of HCF, the performance can be further improved in many columns.

Secondly, we compare KEWE with other text-based word embedding models for readability assessment. These models include NNLM (Bengio et al., 2003), C&W (Collobert and Weston, 2008), GloVe (Pennington et al., 2014), CBOW and Skip-Gram (SG) (Mikolov et al., 2013). Random embedding (Random) and knowledge vectors from the three perspectives (KV) are also used as baselines. All the text-based models mentioned are trained on the four datasets respectively. Table 4 lists the results of applying word embedding for readability assessment using Random Forest as the classifier. From Table 4, the model KEWE$_h$ gets the best performance among all the embedding baselines, including SG+KV, which also takes in both knowledge and text corpus. The results demonstrate the superiority of hybridizing the knowledge graph and text corpus to learn the representation of words.

4.4 Model Analysis

To address RQ2, we analyze the effects of the knowledge graph, the text corpus, and the hybrid ratio on the performance of KEWE. Firstly, we study the impacts of the knowledge graph on the performance of KEWE$_k$. Specifically, we study the following three parameters of the knowledge graph: the knowledge vectors generated from the three word-level difficulties (i.e., acquisition difficulty, usage difficulty, and structure difficulty), the similarity function (i.e., $sim()$), and the number of neighbors of each node in the knowledge graph (i.e., k). Figure 2 shows the performance measures of using each of the three

knowledge vectors respectively in bar chart. It can be found that the acquisition difficulty outperforms the other two on all four datasets. By combining the three knowledge vectors, the performance can be further improved. Figure 3 shows the performance measures of using different similarity functions and different values of k in line charts. It can be found that the knowledge graphs achieve a relatively good performance, when the cosine similarity is used and k is close to 50 or 100.

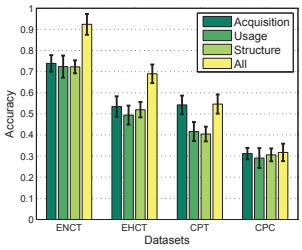


Figure 2: The performance of KEWE$_k$ using different types of knowledge vectors

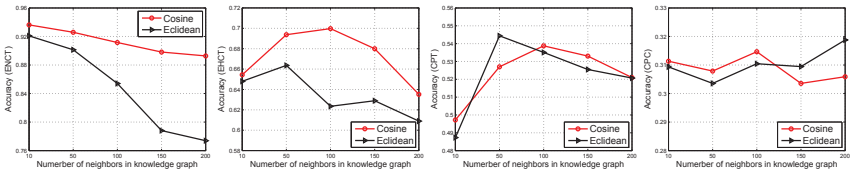


Figure 3: The performance of KEWE$_k$ using knowledge graphs with different similarity functions and neighbor numbers

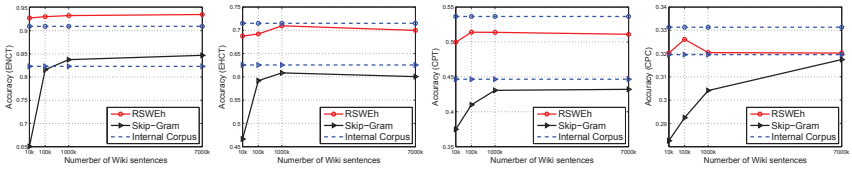


Figure 4: The performance of KEWE$_h$ using different volumes of external corpus

Secondly, we study the impacts of using external corpus on the performance of KEWE$_h$. To learn text-based word embedding for both languages, we collect the external corpus from both English Wikipedia (Ewiki) and Chinese Wikipedia (Cwiki). After preprocessing, Ewiki contains $7M$ sentences and $172M$ words, and Cwiki contains $7.4M$ sentences and $160M$ words. We sample different number of sentences from the corpus for training word embedding, and then measure the performance of KEWE$_h$ on readability assessment. Figure 4 shows the performance measures by using different volumes of external corpus in line charts, with Skip-Gram trained for comparison. Both models are also trained using the internal corpus, and the performance measures (dotted lines) are depicted as baselines. From Figure 4, it can be found that the performance of both Skip-Gram and KEWE$_h$ increases as the volume of external corpus increases, and keeps stable when the corpus is large enough. Besides, on English datasets, word embedding trained using external corpus achieves comparable performance with that using internal corpus. The above suggests that external corpus is a good substitution for the internal corpus, especially on English datasets, and a relative large volume of corpus is required to achieve stable performance.

Finally, we study the impacts of the weighting factor λ on the performance of KEWE$_h$. Since λ is approximated by $N_1/(N_1+N_2)$ in our model, we vary λ from 0 to 1 by setting $N_1+N_2=100$ and then varying N_1 from 0 to 100 stepping by 10. Figure 5 shows the performance of KEWE$_h$ with varied λ in line charts with error bars, where $\lambda=0$ and 1 correspond to Skip-Gram and KEWE$_k$ respectively. From Figure 5, it can be found that KEWE$_h$ can outperform its base models (i.e., KEWE$_k$ and Skip-Gram), by setting λ with a suitable value near the base model which performs better. However, it requires further study to find a suitable λ for training KEWE$_h$.

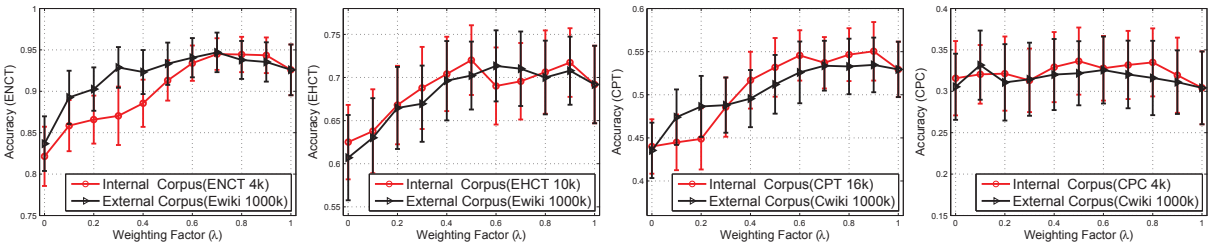


Figure 5: The performance of KEWE$_h$ with varied weighting factor λ

4.5 Error Analysis

To better understand our method, we perform error analysis on the classification results. We mainly describe two sources of errors: word representation failure and word order capturing failure. From the perspective of word representation, the classification error is caused by the fact that the difficulty information may not be properly encoded into the word embedding. Table 4 shows that KEWE performs better than the classical text-based word embeddings in encoding difficulty information into representation, but in the final results there still exists the failure of word representation in KEWE. From the perspective of word order capturing, the classification error is caused by the fact that the order among words may be neglected, so that the syntactic difficulty, pragmatic difficulty, and discourse difficulty of documents are ignored during the process of readability assessment. Table 3 shows that KEWE$_h$ can be further improved by combining with the features related to the word order (i.e., HCF+KEWE$_h$), which means that there exists the failure of word order capturing in KEWE. These two kinds of error sources reveal the limitation of our method. In future work, the neural network accompanied with word embedding is a good alternative, which can produce better representation of documents.

5 Conclusion

In this paper, we propose the knowledge-enriched word embedding (KEWE) for readability assessment. We extract the domain knowledge on word-level difficulty from three different perspectives and construct a knowledge graph. Based on the difficulty context derived from the knowledge graph, we develop two word embedding models (i.e., KEWE$_k$ and KEWE$_h$). The experimental results on English and Chinese datasets demonstrate that KEWE can outperform other well-known readability assessment methods, and the classic text-based word embedding models. Future work is planned to involve extra datasets and additional word embedding strategies so that the soundness of KEWE can be further approved.

Acknowledgements

This work is supported by the National Key R&D Program of China under Grant No. 2018YFB1003800; National Natural Science Foundation of China under Grant Nos. 61373012, 61321491, 91218302; the Fundamental Research Funds for the Central Universities under Grant No. 020214380049. This work is partially supported by the Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

Dimitrios Alikaniotis, Helen Yannakoudakis, and Marek Rei. 2016. Automatic text scoring using neural networks. In *Proceedings of the 54th Annual Meeting of the Association for ComputationalLinguistics*, pages 715–725.

Mikhail Belkin and Partha Niyogi. 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in Neural Information Processing Systems*, 14(6).

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Marc Brysbaert and Andrew Biemiller. 2016. Test-based age-of-acquisition norms for 44 thousand english word meanings. *Behavior Research Methods*, pages 1–4.

Miriam Cha, Youngjune Gwon, and H. T. Kung. 2017. Language modeling by clustering with word embeddings for text readability assessment. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017*, pages 2003–2006.

Jeanne Sternlicht Chall. 1995. *Readability revisited: The new Dale-Chall readability formula*, volume 118. Brookline Books Cambridge, MA.

Kevyn Collins-Thompson and James P Callan. 2004. A language modeling approach to predicting reading difficulty. In *Proceedings of the 2004 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 193–200.

Kevyn Collinsthompson. 2014. Computational assessment of text readability: A survey of current and future research. *International Journal of Applied Linguistics*, 165(2):97–135.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference(ICML 2008)*, pages 160–167.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Edgar Dale and Jeanne S. Chall. 1948. A formula for predicting readability. *Educational Research Bulletin*, 27(1):11–28.

Scott Deerwester. 1990. Indexing by latent semantic analysis. *Journal of the Association for Information Science & Technology*, 41(6):391–407.

Lijun Feng, Martin Jansche, Matt Huenerfauth, and Noémie Elhadad. 2010. A comparison of features for automatic readability assessment. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 276–284.

Rudolph Flesch. 1948. A new readability yardstick. *Journal of applied psychology*, 32(3):221.

Thomas François and Cédrick Fairon. 2012. An ai readability formula for french as a foreign language. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 466–477.

Itziar Gonzalez-Dios, Marıa Jesús Aranzabe, Arantza Dıaz de Ilarraza, and Haritz Salaberri. 2014. Simple or complex? assessing the readability of basque texts. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 334–344.

Palash Goyal and Emilio Ferrara. 2017. Graph embedding techniques, applications, and performance: A survey. *arXiv preprint arXiv:1705.02801*.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

Robert Gunning. 1952. *The technique of clear writing*. McGraw-Hill.

Julia Hancke, Sowmya Vajjala, and Detmar Meurers. 2012. Readability classification for german using lexical, syntactic, and morphological features. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 1063–1080.

Michael Heilman, Collins-Thompson, Jamie Callan, and Maxine Eskenazi. 2007. Combining lexical and grammatical features to improve readability measures for first and second language texts. In *Proceedings of NAACL HLT*, pages 460–467.

Michael Heilman, Kevyn Collins-Thompson, and Maxine Eskenazi. 2008. An analysis of statistical models and features for reading difficulty prediction. In *Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, pages 71–79.

Zhiwei Jiang, Gang Sun, Qing Gu, and Daoxu Chen. 2014. An ordinal multi-class classification method for readability assessment of chinese documents. In *Knowledge Science, Engineering and Management*, pages 61–72. Springer.

Zhiwei Jiang, Gang Sun, Qing Gu, Tao Bai, and Daoxu Chen. 2015. A graph-based readability assessment method using word coupling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 411–420. Association for Computational Linguistics.

Rohit J Kate, Xiaoqiang Luo, Siddharth Patwardhan, Martin Franz, Radu Florian, Raymond J Mooney, Salim Roukos, and Chris Welty. 2010. Learning to predict readability using diverse linguistic features. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 546–554.

Paul Kidwell, Guy Lebanon, and Kevyn Collins-Thompson. 2009. Statistical estimation of word acquisition with application to readability prediction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 900–909.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Air Station, Memphis, TN.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *Meeting of the Association for Computational Linguistics*, pages 302–308.

Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1501–1511.

Yi Ma, Eric Fosler-Lussier, and Robert Lofthus. 2012. Ranking-based readability assessment for early primary children's literature. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 548–552.

G Harry McLaughlin. 1969. Smog grading: A new readability formula. *Journal of reading*, 12(8):639–646.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 186–195.

Yafeng Ren, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016. Improving twitter sentiment classification using topic-enriched multi-prototype word embeddings. In *Thirtieth AAAI Conference on Artificial Intelligence*, pages 3038–3044.

Sam T. Roweis and Lawrence K. Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–6.

Elliot Schumacher, Maxine Eskenazi, Gwen Frishkoff, and Kevyn Collins-Thompson. 2016. Predicting the relative difficulty of single sentences with and without surrounding context. In *Conference on Empirical Methods in Natural Language Processing*, pages 1871–1881.

Sarah E Schwarm and Mari Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 523–530.

Jie Shen and Cong Liu. 2016. Improved word embeddings with implicit structure information. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference*, pages 2408–2417.

Luo Si and James P. Callan. 2001. A statistical model for scientific readability. In *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management*, pages 574–576.

Manjira Sinha, Tirthankar Dasgupta, and Anupam Basu. 2014. Influence of target reader background and text features on text readability in bangla: A computational approach. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference*, pages 345–354.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Meeting of the Association for Computational Linguistics*, pages 1555–1565.

Amalia Todirascu, Thomas François, Delphine Bernhard, Nuria Gala, and Anne-Laure Ligozat. 2016. Are cohesive features relevant for text readability evaluation? In *26th International Conference on Computational Linguistics (COLING 2016)*, pages 987–997.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188.

Sowmya Vajjala and Detmar Meurers. 2012. On improving the accuracy of readability classification using insights from second language acquisition. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP, BEA@NAACL-HLT 2012, June 7, 2012, Montréal, Canada*, pages 163–173.

Shuhan Wang and Erik Andersen. 2016. Grammatical templates: Improving text difficulty evaluation for language learners. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference*, pages 1692–1702.

Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1225–1234.

Changxing Wu, Xiaodong Shi, Yidong Chen, Jinsong Su, and Boli Wang. 2017. Improving implicit discourse relation recognition with discourse-specific word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 269–274.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. pages 40–48.

WikiRef: Wikilinks as a route to recommending appropriate references for scientific Wikipedia pages

Abhik Jana
IIT Kharagpur, India
abhik.jana@iitkgp.ac.in

Pranjal Kanojiya
IIT Kharagpur, India
QUALCOMM India Private Limited
pranjal989091@gmail.com

Pawan Goyal
IIT Kharagpur, India
pawang@cse.iitkgp.ac.in

Animesh Mukherjee
IIT Kharagpur, India
animeshm@gmail.com

Abstract

The exponential increase in the usage of Wikipedia as a key source of scientific knowledge among the researchers is making it absolutely necessary to metamorphose this knowledge repository into an integral and self-contained source of information for direct utilization. Unfortunately, the references which support the content of each Wikipedia entity page, are far from complete. Why are the reference section ill-formed for most Wikipedia pages? Is this section edited as frequently as the other sections of a page? Can there be appropriate surrogates that can automatically enhance the reference section? In this paper, we propose a novel two step approach – **WikiRef** – that (i) leverages the wikilinks present in a scientific Wikipedia target page and, thereby, (ii) recommends highly relevant references to be included in that target page appropriately and automatically borrowed from the reference section of the wikilinks. In the first step, we build a classifier to ascertain whether a wikilink is a potential source of reference or not. In the following step, we recommend references to the target page from the reference section of the wikilinks that are classified as potential sources of references in the first step. We perform an extensive evaluation of our approach on datasets from two different domains – Computer Science and Physics. For Computer Science we achieve a notably good performance with a *precision@1* of 0.44 for reference recommendation as opposed to 0.38 obtained from the most competitive baseline. For the Physics dataset, we obtain a similar performance boost of 10% with respect to the most competitive baseline.

1 Introduction

Wikipedia, the largest online encyclopedia, is a collaborative work of 33,778,487 users and 1,210 admins, with 5,662,889 English articles and a total of 45,132,517 articles in more than 250 languages as of June 7, 2018[1]. Introduced in 2001, it has become one of the most visited websites having global Alexa Rank – 5[2]. Each Wikipedia article contains a rich collection of concepts along with a set of hyperlinks associating crucial terms to other wiki pages (termed as wikilinks), which makes the article more comprehensive. In addition, the ability to edit the entity pages easily with proper source citations and to view the edits reflected in the pages timely are some of the key reasons behind Wikipedia's growth and popularity. As a consequence, Wikipedia has become a destination of all kinds of information about

[1] https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia
[2] http://www.alexa.com/siteinfo/www.wikipedia.org

Proceedings of the 27th International Conference on Computational Linguistics, pages 379–389
Santa Fe, New Mexico, USA, August 20-26, 2018.

entities and events. Further, Wikipedia is being widely utilized by many applications focused on entity linking and disambiguation (Moro et al., 2014; Hachey et al., 2013; Cucerzan, 2007; Hoffart et al., 2011), named entity disambiguation (Cucerzan, 2007; Barrena et al., 2015), semantic similarity (Gabrilovich and Markovitch, 2007; Wu and Giles, 2015), information extraction (Wu and Weld, 2010) etc.

What is lacking? Nevertheless, there are different competing opinions among the researchers regarding the reliability, integrity and usage of Wikipedia as an authentic source of scientific information. While many researchers consider Wikipedia as a valuable resource, others question the accuracy, comprehensiveness and completeness of the articles. For instance, according to Gorman (2007), Wikipedia is an "unethical resource unworthy of our respect" whereas Orlowski (2005) questions that the technology which makes Wikipedia possible is not a substitute for expert editors, and poor writing will remain unavoidable without any expertise. Chesney (2006) conducted an experiment to evaluate the credibility of Wikipedia and found that the studies focusing only on history articles provide mixed evidence concerning its accuracy. The general solution is to post-process the edits done by the users. In order to ensure the accuracy and quality of writing of articles, Wikipedia has imposed a new policy of requiring senior editors to approve changes to articles.

Even then, the completeness of the article is not fully ensured. Not all Wikipedia pages referring to entities (entity pages) are comprehensive: relevant information can either be missing or added with a delay. In addition, the frequency of editing of different sections in a given Wikipedia article varies extensively. For instance, frequency of editing text is higher than frequency of editing wikilinks or references; also add/edit references is considered as the hardest task[3]. This opens up the requirement for an automated system which increases the number of relevant references by either processing the content of the target article itself or by appropriately collating references from alternative sources so as to make the overall information content of the article more complete.

State-of-the-art: Efforts have been made by the researchers to populate Wikipedia pages automatically. Sauper and Barzilay (2009) propose an approach for populating Wikipedia pages with content coming from external sources by automatically generating whole entity pages for specific entity classes. Taneva and Weikum (2013) propose an approach to generate novel summaries from the external information which could be added to Wikipedia entity pages. Balog and Ramampiaro (2013) and Balog et al. (2013) try to recommend news citations for an entity in Wikipedia. West et al. (2014) focuses on the problem of knowledge base completion, through question answering and tries to complete missing facts in Freebase based on templates. All these works significantly rely on high quality input sources which are utilized to extract textual facts for Wikipedia page population. In one of the recent works, Fetahu et al. (2015) propose a two-stage supervised approach for suggesting news articles corresponding to entity pages. Attempt has also been made to introduce appropriate wikilinks automatically (İkıkat et al., 2015). In a similar line, Raganato et al. (2016) present the automatic construction and evaluation of a Semantically Enriched Wikipedia (SEW) in which the overall number of wikilinks has been more than tripled. Efforts have also been made to generate Wikipedia articles of named entities in a semi-supervised framework (Pochampally et al., 2016).

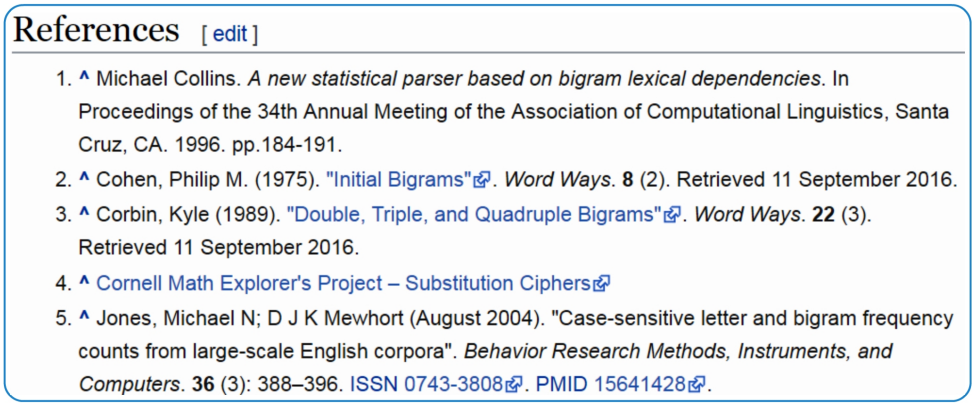
Motivation: Recently, with the increasing number of scientific articles and with the need for getting an overall view of a particular scientific topic within a very less amount of time, researcher's tendency of referring to the Wikipedia pages instead of going through a set of very specific scientific articles is massively increasing. Given that, such a huge community of researchers rely on Wikipedia for scientific information, it becomes absolutely necessary to deeply focus on the completeness of the Wikipedia articles which talk about some scientific topics.

In the following we enlist some initial observations to show that there is a huge scope of improvement of the scientific Wikipedia articles through the inclusion of a proper set of references. Figure 1 illustrates the snapshot of the reference section of the 'Bigram' page[4] in the year 2018. There are only five references present in the page; very relevant scientific articles like, "Statistical Identification of Language" (Dunning, 1994), "Foundations of Statistical Natural Language Processing" (Manning and

[3] https://www.mediawiki.org/wiki/Editing_Tasks_Survey

[4] https://en.wikipedia.org/wiki/Bigram

Schütze, 1999), etc. are missing, without which the article seems to be incomplete. Also from the survey conducted by wikimedia [5] it has been observed that adding or editing the reference section is harder compared to adding or editing simple text or a wikilink which therefore leads to much less frequent changes of the reference section. In order to further confirm this trend, we conduct an analysis on the edit history of 1120 Wikipedia articles from the Computer Science category and observe that till Jan, 2017 on average around 65% edits are in the text content, 32% are wikilink edits and strikingly only 1% are reference edits. Rest correspond to table, template and category edits etc. We present some example Wikipedia articles along with the statistic of their edit history in Table 1. Clearly, all these observations point to the requirement of an automated reference recommendation system to keep the reference section of a scientific Wikipedia article up-to-date.


References [edit]

1. ^ Michael Collins. *A new statistical parser based on bigram lexical dependencies*. In Proceedings of the 34th Annual Meeting of the Association of Computational Linguistics, Santa Cruz, CA. 1996. pp.184-191.
2. ^ Cohen, Philip M. (1975). "Initial Bigrams". *Word Ways*. **8** (2). Retrieved 11 September 2016.
3. ^ Corbin, Kyle (1989). "Double, Triple, and Quadruple Bigrams". *Word Ways*. **22** (3). Retrieved 11 September 2016.
4. ^ Cornell Math Explorer's Project – Substitution Ciphers
5. ^ Jones, Michael N; D J K Mewhort (August 2004). "Case-sensitive letter and bigram frequency counts from large-scale English corpora". *Behavior Research Methods, Instruments, and Computers*. **36** (3): 388–396. ISSN 0743-3808. PMID 15641428.

Figure 1: Snapshot of Wikipedia 'Bigram' in the year 2018.

Wikipage title	Start date	End date	#(Total edits)	#(Wikilink edits)	#(Ref. edits)
Connection Machine	7/8/02	6/1/17	242	98	3
Fuzzy Logic	15/4/02	29/1/17	1343	390	5
Javascript	5/2/02	31/1/17	4141	1146	69

Table 1: Representative scientific Wikipedia articles with detailed statistic of their edit history. Start date indicates the date of creation of Wikipage and end date indicates the time point till which we perform the analysis of edit history.

What we envisage? Motivated by the fact that wikilink edits are less hard and more frequent compared to reference edits, a natural question that arises is that whether it is possible to develop an automatic procedure to add a set of references whenever there is a wikilink edit on a target page. In this paper, we investigate the possibility of enriching the reference section of a scientific Wikipedia article using the wikilinks present in the article as a surrogate. In particular, *we introduce the novel idea of reference inheritance from the source pages pointed to by the wikilinks present on the target page to populate the reference section of the target page with relevant references*.

For this study, we consider 3842 target Wikipedia pages (till June, 2017) from the Computer Science domain. In order to show that our method is quite generic, we also consider an additional dataset comprising 2871 target Wikipedia pages (till February, 2018) from the Physics domain.

We propose a novel two step approach as follows. In the first step, *we determine the potential wikilinks from which references could be inherited*. In the second step, *we recommend the most relevant references from the potential wikilinks obtained in the first step*.

Key contributions: In summary, we make the following contributions:

a) *Novel problem formulation*: We formulate and address the problem of automatically populating the reference section by inheriting the references from the wikilinks present in a target scientific Wikipedia article.

b) *The two step approach*: We propose a two-step reference recommendation system and show its advantage over simple baselines that do not implement the first step of potential wikilink selection from which

[5] https://commons.wikimedia.org/wiki/File:WMF_editing_tasks_survey_July_2015.pdf

references could be inherited.
c) *Evaluation*: We perform extensive experiments to evaluate our system. We evaluate in two different ways - first, by comparing our recommendations with the references already present on the target page (acting as a ground-truth) and second, through human judgement where experts are tasked to judge the appropriateness of the recommended references. For the automatic evaluation we obtain a *precision@1* of 0.44 for the Computer Science dataset compared to 0.38 obtained for the most competitive baseline. For the Physics dataset we obtain a similar performance boost of 10% over the best baseline. Human judgement experiments show that the average Spearman's rank correlation coefficient (ρ) between the ranked list returned by WikiRef and the one produced by averaging the responses of the annotators is **0.203** compared to 0.168 obtained for the most competitive baseline[6].

2 Problem formulation

Let us assume that we have a scientific Wikipedia article A (i.e., target) whose reference section we wish to populate using a set of relevant references. As an input we use only the text content and wikilinks present in A. Suppose, A has a set of n wikilinks $B_1, B_2, \ldots, B_n$ which in this case simply correspond to the n Wikipedia source articles from which A could possibly inherit the references. We assume that while the reference sections of the individual source pages themselves may not be "well populated", together the reference sections of all the source articles may provide a potential set of references, relevant to A. We divide this task in two major steps. First, we obtain the wikilinks (i.e., a set of Bs) that are appropriate for inheriting references from. We pose this as a binary classification problem thus dividing the set of Bs in those that are relevant (appropriate for inheriting references) vs those that are not. Second, we prepare a ranked list of k references taken from each of the correctly classified (i.e., relevant) wikilinks which could be added to the reference section of A. We pose this second step as a "learning-to-rank" problem and use various features to prepare the final list of recommendations.

3 Dataset

We use datasets from two different domains – Computer Science (CS) and Physics (PH) – to evaluate our system. Note that while CS is our primary focus (owing to the background of the authors) we use the PH dataset to show that our method is quite generic.
CS dataset: In order to prepare the CS dataset we crawl a set of 3842 target Wikipedia pages till June, 2017. The topics we span in this crawl are information retrieval, machine learning, automata theory, graph theory etc. In addition, we also crawl all the source pages corresponding to all the wikilinks present in the 3,842 target articles. The total number of wikipages thus crawled is 121,154.
PH dataset: For the PH dataset we crawl a total of 2,871 target articles spanning topics like gravitation, mechanics, motion etc. The source pages pointed to by the wikilinks present on these target articles together make the total crawl size of 107,332 pages.

4 Building WikiRef

Recall, that our approach works in two steps. We present the details of each of these steps below which together refers to our system – WikiRef.

4.1 Classification of wikilinks (Step - I)

As discussed in Section 2, we have a target Wikipedia article A with n wiklinks ($B_1, B_2, \ldots, B_n$) present in it. Our goal in this step is to ascertain how appropriate a source page pointed to by a wikilink B_i is for inheriting references. To estimate this, we measure the extent of similarity between A and the page pointed to by B_i. The hypothesis is that the higher this similarity, the more relevant is the source page to A and thus higher is the propensity of reference inheritance. We define various notions of similarity that act as features to a binary classifier which classifies if the source is relevant or not.

[6]We perform the human judgement experiment only for the CS domain because of the background of the contributing authors of this paper.

- **Tf-idf similarity of article summaries (TIS):** We represent the summaries (first paragraphs) of A and the source page pointed to by B_i as tf-idf vectors and compute the cosine similarity between them. Note that, inverse document frequency (idf) is calculated only from the dataset we have prepared.
- **Outlinks similarity (OS)**: Outlinks of a Wikipedia article (say A) is the set of Wikipedia articles that A hyperlinks to. We compute outlinks similarity as the Jaccard overlap between the outlinks of A and the source page pointed to by B_i.
- **Inlinks similarity (IS)**: Inlinks of a Wikipedia article (say A) is the set of Wikipedia articles that hyperlink to the page A. We compute inlinks similarity as the Jaccard overlap between the inlinks of A and the source page pointed to by B_i. Note that, for computing inlinks we consider the articles present only in our dataset.
- **Out sentence similarity (OSS)**: Here we consider the common outlinks of A and the source page pointed to by B_i. We collate all the sentences in A where these common outlinks occur, to prepare a document OS_A. Similarly, we collate all sentences in the page pointed to by B_i where these common outlinks occur to form a document OS_{B_i}. Now we represent OS_A and OS_{B_i} as tf-idf vectors and compute cosine similarity between them.
- **In sentence similarity (ISS)**: This is same as OSS with the exception that here we construct the documents based on the sentences where the common inlinks of A and the page pointed to by B_i occur.

Additional deep neural representations: Apart from these five similarity measures we use additional variants of TIS, OSS and ISS where instead of representing a text document as a tf-idf vector we represent it as a document vector obtained using an encoder based on a bi-directional LSTM architecture with max pooling, trained on the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015) as proposed by Conneau et al. (2017). Note that, for applying this architecture we use the GloVe vectors trained on Common Crawl data (840B tokens)[7]as seeds for representing words in a document. We name these variants of TIS, OSS and ISS respectively as vector similarity of article summaries (VS), out-sentence vector similarity (OSVS) and in-sentence vector similarity (ISVS) respectively.

We plug in all these features in classifiers like SVM, Random Forest, Logistic Regression etc. We observe that Random Forest performs the best among these and therefore report the results for this classifier only. This step automatically identifies the wikilinks that are potential candidates for reference inheritance.

4.2 Recommending the exact list of references (Step - II)

As an outcome of the previous step, we have a target Wikipedia article A and a wikilink B_i which is classified as either relevant for reference inheritance or not. Let us assume that for a B_i that is classified as relevant in the first step, the source page pointed to by B_i has m references $R_1, R_2, ..., R_m$. However, all of these m references might not be appropriate to inherit. Therefore, the task here is to produce a ranked list of k references which A can inherit depending on the relevance of the reference with respect to the content of A. In order to estimate the relevance of a reference (say R_j) we propose the following features.

- **Similarity between the citation context of the reference R_j in the source page pointed to by B_i and the citation context of B_i in A**: Citation context represents the sentence in which a reference or wikilink gets cited. We represent the citation contexts as simple tf-idf vectors and compute the cosine similarity between the tf-idf vectors of the citation context of R_j in the source page pointed to by B_i and the citation context of B_i in A. We term this feature as **F1-TI**. We also represent citation contexts as sentence vectors using the bi-LSTM architecture proposed by Conneau et al. (2017) (already discussed earlier) and compute the cosine similarity of these vector representations of the citation contexts of R_j in the source page pointed to by B_i and the citation context of B_i in A. We call this feature **F1-Vec**.
- **Similarity between the title of the reference R_j in the source page pointed to by B_i and the citation context of B_i in A**: As the title of any reference contains the most important clue about the

[7] https://nlp.stanford.edu/projects/glove/

reference itself, we represent the title of the reference R_j in the source page pointed to by B_i and citation context of B_i in A as tf-idf vectors and compute the cosine similarity between them. We call this feature **F2-TI**[8].

We use – SVMRank (Joachims, 2006) – a standard learning-to-rank framework and plug in the above three features to obtain a ranked list of k references to recommend. We report the performance our approach for different values of k in Section 5.

5 Evaluation

We first individually evaluate Step - I and Step - II and subsequently report the performance of WikiRef as a whole. We evaluate the performance using the standard precision-recall metrics and compare it with various baseline approaches that we ourselves propose since there is no known work in the literature that could serve as a suitable baseline for this task.

5.1 Gold standard dataset

As discussed in section 3 we have a total of 3,842 target Wikipedia pages from the Computer Science (CS) domain and 2,871 target Wikipedia pages from the Physics (PH) domain for which we aim to come up with a list of references. We consider the existing references in the target pages (i.e., the A pages) at the time-point of crawling as the gold standard references to be evaluated against.

5.2 Baselines

There have been several reference recommendation systems (Huang et al., 2015; Caragea et al., 2013; Huang et al., 2014; Tang et al., 2014; Ren et al., 2014; Meng et al., 2013; Huang et al., 2012) for scientific articles but all of them either use the the text of the cited paper or the underlying citation network which makes these systems inappropriate to be used as baselines to compare with WikiRef. Therefore, we propose some standard baselines to ascertain the importance of each step in our approach.

Baseline I: Here, Step - I of our approach is skipped and we assume that all the wikilinks present in the target Wikipedia article are relevant for reference inheritance. In Step - II we rank the references of each wikilink only on the basis of tf-idf similarity between references' citation context in the page pointed to by the wikilink and the wikilink's citation context in target Wikipedia article.

Baseline II: Here again, Step - I of our approach is skipped and we assume that all the wikilinks present in the target Wikipedia article are relevant for reference inheritance. In Step - II we rank the references of each wikilink only on the basis of tf-idf similarity between the references' title and the wikilink's citation context in the target Wikipedia article.

Baseline III: Step - I is fully retained. In Step - II we rank the references of each wikilink only on the basis of tf-idf similarity between references' citation context in wikilink's content and wikilink's citation context in target Wikipedia article (F1-TI).

Baseline IV: Step - I is fully retained. In Step - II we rank the references of each wikilink only on the basis of tf-idf similarity between references' title and wikilink's citation context in the target Wikipedia article (F2-TI).

Baseline V: Step - I is fully retained. In Step - II we rank the references of each wikilink only on the basis of cosine similarity between sentence vector representation of references' citation context in the page pointed to by the wikilink and the wikilink's citation context in the target Wikipedia article (F1-Vec).

We compare the performance of these baselines with WikiRef in section 5.5.

5.3 Performance analysis of Step - I

We use 70% of the Wikipedia pages for training and rest 30% of them for testing using the gold standard dataset. However, we observe from the gold standard dataset that on average only 10% wikilinks are suitable for reference inheritance while the rest 90% are irrelevant. To tackle this imbalance in the

[8]We do not use the corresponding deep vectors since the reference title are in many cases very small and do not produce appropriate vectors thus negatively affecting the overall performance of our system.

dataset, we follow the under-sampling technique based on the repeated edited nearest neighbour method proposed by Lemaître et al. (2017).

First, in order to understand the importance of the proposed features we perform a χ-square test for the classification. We report the ranking of the features in Table 2. Subsequently, we plug the features according to this rank from top to bottom one by one into the classifier; the performance obtained thereby are presented in Table 3. We observe that appending the features one by one according to the χ-square test ranking helps to improve the overall performance of the classifier except the last feature which causes no improvements.

Rank	1	2	3	4	5	6	7	8
Feature	TIS	OSS	ISS	OS	OSVS	VS	ISVS	IS

Table 2: χ-square test ranking of features for Step - I (wikilink classification).

Features Used	**Precision**	**Recall**	**F-Measure**
TIS	0.11	0.90	0.20
TIS, OSS	0.16	0.36	0.20
TIS, OSS, ISS	0.39	0.37	0.33
TIS, OSS, ISS, OS	0.47	0.44	0.40
TIS, OSS, ISS, OS, OSVS	0.48	0.42	0.40
TIS, OSS, ISS, OS, OSVS, VS	0.49	0.42	0.40
TIS, OSS, ISS, OS, OSVS, VS, ISVS	**0.50**	**0.45**	**0.42**
TIS, OSS, ISS, OS, OSVS, VS, ISVS, IS	**0.50**	**0.45**	**0.42**

Table 3: Step - I (wikilink classification) evaluation with incremental feature set.

Some of the representative Wikipedia articles from the CS dataset along with the relevant/irrelevant wikilinks for reference inheritance as correctly predicted by our classifier are noted in Table 4. We observe that the relevant wikilinks are semantically more close to the target Wikipedia articles compared to the irrelevant ones, thus making the former ones as more probable candidates for reference inheritance.

Target Wikipedia article	**Relevant wikilinks**	**Irrelevant wikilinks**
Feature (machine learning)	Machine learning, Perceptron	Computer vision, Speech recognition
Vanishing gradient problem	Recurrent neural network, Back-propagation	OPTICS algorithm, Cluster analysis
Sensitivity and specificity	True positive rate, Precision and recall	Clinical research, Airport security

Table 4: Representative results of wikilink classification for the CS dataset.

5.4 Performance analysis of Step - II

Table 5 reports the performance of our approach for various values of k. Considering the difficulty of the task we observe that our system achieves very good performance in terms of precision even for higher values of k. Some of the representative recommended references along with the target Wikipedia articles are shown in Table 6. The correct references in the table signify the references which are actually present in the target Wikipedia article's reference section (i.e., in the gold standard dataset). We also observe that the incorrect references which are actually not present in the target Wikipedia article's reference section are not adjudged as relevant by our system. As we cannot compare these references with the gold standard, we perform manual evaluation for only these references in order to analyze how suitable these references are for the target Wikipedia article (see Section 5.6).

k	**Precision**	**Recall**	**F-Measure**
1	0.44	0.21	0.28
2	0.41	0.23	0.30
3	0.40	0.26	0.30
4	0.37	0.27	0.31
5	0.34	0.30	0.31
10	0.25	0.35	0.30

Table 5: Evaluation of Step - II for different values of k.

Target Wikipedia article	Relevant wikilink	Correct reference	Incorrect reference
Heap (data structure)	Binary heap	Introduction to Algorithms	Min-max heaps and generalized priority queues
Sensitivity and specificity	Precision and recall	An Introduction to ROC Analysis	Advanced Data Mining Techniques

Table 6: Representative results of Step - II for the CS dataset.

5.5 WikiRef vs the baselines

The comparisons of the performances of different baselines with WikiRef are noted in Table 7. The table shows that for all the values of k the performance of BL-III and BL-IV is significantly better than BL-I and BL-II respectively leading to the fact that the role of Step - I is very crucial in this context and, in particular, helps boost the performance of the full system. In addition, we observe that the performance we get only using unsupervised ranking in step - II based on any one of the three features (used in BL-III, BL-IV and BL-V respectively) described in section 4.2 cannot beat the performance of our approach where we use a supervised approach like SVMRank that draws advantage of all the three features.

k	Metric	BL-I	BL-II	BL-III	BL-IV	BL-V	WikiRef
1	Precision	0.22	0.15	0.38	0.31	0.36	**0.44**
	Recall	0.04	0.05	0.18	0.09	0.10	**0.21**
	F-Measure	0.068	0.075	0.20	0.11	0.14	**0.28**
3	Precision	0.12	0.09	0.34	0.26	0.27	**0.40**
	Recall	0.08	0.1	0.22	0.17	0.16	**0.26**
	F-Measure	0.096	0.075	0.28	0.17	0.20	**0.30**
5	Precision	0.1	0.07	0.31	0.23	0.25	**0.34**
	Recall	0.1	0.12	0.26	0.21	0.22	**0.30**
	F-Measure	0.1	0.08	0.27	0.18	0.21	**0.31**
10	Precision	0.06	0.06	**0.25**	0.22	0.23	**0.25**
	Recall	0.19	0.19	0.30	0.25	0.26	**0.35**
	F-Measure	0.09	0.09	0.26	0.20	0.24	**0.30**

Table 7: Comparison of the performance of proposed baselines along with WikiRef. Best results are in green cells and the most competing baseline results are in blue cells.

5.6 Manual evaluation

The aim of this manual evaluation is to understand the quality of the recommendations returned by WikiRef for the cases which are not present in the gold standard dataset. In the survey, we provide a total of 25 Wikipedia pages from the Computer Science (CS) dataset along with five recommended references which are not already present in the gold standard and ask the annotators to choose the appropriate references which should be there in the reference section of the given Wikipedia article to make it more resourceful. A sample evaluation page can be seen in this link[9]. The list of scientific Wikipedia articles that we use for manual evaluation are noted in Table 8. 10 participants with Computer Science background have taken part in this survey. First, we compute absolute performance of our system which shows on an average, the fraction of references actually found to be relevant by the annotators. We compute the performance score per Wikipage as the fraction of the proposed references found to be relevant by the annotators (averaged over all the annotators). We then compute the average over the full list of 25 Wikipages, obtaining a very good performance score of 0.63 leading to the fact that out of 5 recommended references, close to 3 references are found to be relevant by the annotators.

Next, we also attempt to correlate the ranking of the competing systems with the implicit ranking imposed by the annotators. For each Wikipedia article, we rank the five references depending on the number of participants voting for a particular reference, using standard fractional ranking method. Then we compute the Spearman's rank correlation coefficient (ρ) between these ranks and the ranked list returned by WikiRef as well as other baselines. Table 9 shows that WikiRef beats the most competitive baseline by a significant margin.

[9] https://goo.gl/forms/N0rmN5xPRzhXsoCL2

Algebric modeling language, Bayesian model of computational anatomy, Object-based language, Recursive ordinal, Graph isomorphism, Recurrence plot, RossFahroo pseudospectral method, Deterministic finite automaton, Clustal, Graph structure theorem, Bidirected graph, Algebraic graph theory, Belt machine, HadwigerNelson problem, Finite-state machine, Two-phase locking, Urquhart graph, Software transactional memory, Radial basis function kernel, Semi-symmetric graph, Algorithm characterizations, Elliott formula, Just-in-time compilation, Abstract family of languages, PAdES.

Table 8: Target Wikipedia pages for manual evaluation.

Metric	BL-I	BL-II	BL-III	BL-IV	BL-V	WikiRef
Average ρ	0.099	0.16	0.089	0.168	0.104	**0.203**

Table 9: Human judgement experiments comparing the proposed baselines with WikiRef. The best result is highlighted in a green cell and the most competing baseline result is highlighted in a blue cell.

5.7 Performance analysis for the Physics (PH) dataset

So far, we have reported all the performance figures for the Computer science (CS) dataset. In order to investigate the applicability of our system in other domains, we repeat the experiments on the Physics (PH) dataset as well. The results for both step - I and step - II are noted in Table 10. We use the same eight features (TIS, OSS, ISS, OS, OSVS, VS, ISVS, IS) with Random Forest classifier for step - I and for step - II we again choose the same SVMRank framework fed in with the three features (F1-TI, F1-Vec and F2-TI) as discussed earlier. From Table 11 we observe that for almost all the cases WikiRef performs better than the baselines.

Step		Precision	Recall	F-Measure
Step-I		0.41	0.44	0.37
Step-II	k=1	0.45	0.1	0.16
	k=2	0.41	0.13	0.19
	k=3	0.38	0.16	0.18
	k=4	0.35	0.20	0.25
	k=5	0.32	0.22	0.23
	k=10	0.30	0.25	0.26

Table 10: Performance of WikiRef on the Physics (PH) dataset

k	Metric	BL-I	BL-II	BL-III	BL-IV	BL-V	WikiRef
1	Precision	0.20	0.24	0.36	0.36	0.41	**0.45**
	Recall	**0.11**	0.07	0.09	0.09	0.09	0.1
	F-Measure	0.14	0.09	0.12	0.12	0.13	**0.16**
3	Precision	0.18	0.18	0.28	0.26	0.29	**0.38**
	Recall	**0.18**	0.13	0.15	0.15	0.15	0.16
	F-Measure	0.16	0.11	0.15	0.15	0.16	**0.18**
5	Precision	0.15	0.15	0.24	0.24	0.25	**0.32**
	Recall	**0.22**	0.18	0.19	0.19	0.18	**0.22**
	F-Measure	0.16	0.12	0.16	0.16	0.16	**0.23**
10	Precision	0.17	0.12	0.22	0.22	0.22	**0.30**
	Recall	**0.29**	0.27	0.23	0.23	0.22	0.25
	F-Measure	0.15	0.13	0.17	0.18	0.17	**0.26**

Table 11: Comparison of the performance of proposed baselines along with WikiRef for Physics (PH) dataset. Best results are in green cells and the most competing baseline results are in blue cells.

6 Conclusion

This paper presented a novel two-step recommendation system for enhancing the reference section of the Wikipedia entity pages, dealing with scientific concepts by inheriting the references from the wikilinks present in the page itself. In the first stage we obtain relevant wikilinks for an entity page via a supervised

classification approach. In the second stage we recommend a ranked list of references from the relevant wikilinks obtained from the first stage. WikiRef achieves an overall *precision@1* of 0.44 for the CS dataset and 0.45 for the PH dataset. These are very significant improvements over the most competing baselines in both cases. In addition, manual evaluations over 25 wikipages show that WikiRef also recommends the most relevant references that are absent in the gold standard dataset. We have made our code and data publicly available[10].

Immediate future work would be to focus on further evaluations on various datasets including biology and medicine. Also, the proposed approach can be adapted to filter any type of entity pages' reference section. We further plan to prepare an online tool that triggers WikiRef to recommend references as soon as relvant wikilinks are added to a target Wikipedia article.

References

[Balog and Ramampiaro2013] Krisztian Balog and Heri Ramampiaro. 2013. Cumulative citation recommendation: Classification vs. ranking. In *Proc. of ACM-SIGIR*. ACM.

[Balog et al.2013] Krisztian Balog, Heri Ramampiaro, Naimdjon Takhirov, and Kjetil Nørvåg. 2013. Multi-step classification approaches to cumulative citation recommendation. In *Proc. of OAIR*.

[Barrena et al.2015] Ander Barrena, Aitor Soroa, and Eneko Agirre. 2015. Combining mention context and hyperlinks from wikipedia for named entity disambiguation. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 101–105.

[Bowman et al.2015] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

[Caragea et al.2013] Cornelia Caragea, Adrian Silvescu, Prasenjit Mitra, and C Lee Giles. 2013. Can't see the forest for the trees?: a citation recommendation system. In *Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries*, pages 111–114. ACM.

[Chesney2006] Thomas Chesney. 2006. An empirical examination of wikipedia's credibility. *First Monday*.

[Conneau et al.2017] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680. Association for Computational Linguistics.

[Cucerzan2007] Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL*.

[Dunning1994] Ted Dunning. 1994. *Statistical identification of language*. Computing Research Laboratory, New Mexico State University.

[Fetahu et al.2015] Besnik Fetahu, Katja Markert, and Avishek Anand. 2015. Automated news suggestions for populating wikipedia entity pages. *CIKM*.

[Gabrilovich and Markovitch2007] Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJcAI*, volume 7, pages 1606–1611.

[Gorman2007] GE Gorman. 2007. A tale of information ethics and encyclopaedias; or, is wikipedia just another internet scam? *Online Information Review*.

[Hachey et al.2013] Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R Curran. 2013. Evaluating entity linking with wikipedia. *Artificial intelligence*.

[Hoffart et al.2011] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *EMNLP*.

[Huang et al.2012] Wenyi Huang, Saurabh Kataria, Cornelia Caragea, Prasenjit Mitra, C Lee Giles, and Lior Rokach. 2012. Recommending citations: translating papers into references. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1910–1914. ACM.

[10] `https://github.com/KingOfThePirate/Wikiref`

[Huang et al.2014] Wenyi Huang, Zhaohui Wu, Prasenjit Mitra, and C Lee Giles. 2014. Refseer: A citation recommendation system. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 371–374. IEEE Press.

[Huang et al.2015] Wenyi Huang, Zhaohui Wu, Liang Chen, Prasenjit Mitra, and C Lee Giles. 2015. A neural probabilistic model for context based citation recommendation. In *AAAI*, pages 2404–2410.

[İkıkat et al.2015] F Yesjm İkıkat, Behire Gürhan, and Banu Dırı. 2015. Automatic linking of wikipedia pages by their semantic similarity. In *Innovations in Intelligent SysTems and Applications (INISTA), 2015 International Symposium on*, pages 1–5. IEEE.

[Joachims2006] Thorsten Joachims. 2006. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM.

[Lemaître et al.2017] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. 2017. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5.

[Manning and Schütze1999] Christopher D Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT press.

[Meng et al.2013] Fanqi Meng, Dehong Gao, Wenjie Li, Xu Sun, and Yuexian Hou. 2013. A unified graph model for personalized query-oriented reference paper recommendation. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 1509–1512. ACM.

[Moro et al.2014] Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *TACL*.

[Orlowski2005] Andrew Orlowski. 2005. Wikipedia science 31% more cronky than britannica's. *The Register*.

[Pochampally et al.2016] Yashaswi Pochampally, Kamalakar Karlapalem, and Navya Yarrabelly. 2016. Semi-supervised automatic generation of wikipedia articles for named entities. In *Wiki@ICWSM*.

[Raganato et al.2016] Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2016. Automatic construction and evaluation of a large semantically enriched wikipedia. In *IJCAI*, pages 2894–2900.

[Ren et al.2014] Xiang Ren, Jialu Liu, Xiao Yu, Urvashi Khandelwal, Quanquan Gu, Lidan Wang, and Jiawei Han. 2014. Cluscite: Effective citation recommendation by information network-based clustering. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 821–830. ACM.

[Sauper and Barzilay2009] Christina Sauper and Regina Barzilay. 2009. Automatically generating wikipedia articles: A structure-aware approach. In *Proc. of ACL-IJCNLP*.

[Taneva and Weikum2013] Bilyana Taneva and Gerhard Weikum. 2013. Gem-based entity-knowledge maintenance. In *Proc. of ACM-CIKM*, pages 149–158.

[Tang et al.2014] Xuewei Tang, Xiaojun Wan, and Xun Zhang. 2014. Cross-language context-aware citation recommendation in scientific articles. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 817–826. ACM.

[West et al.2014] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *WWW*.

[Wu and Giles2015] Zhaohui Wu and C Lee Giles. 2015. Sense-aaware semantic analysis: A multi-prototype word representation model using wikipedia. In *AAAI*, pages 2188–2194.

[Wu and Weld2010] Fei Wu and Daniel S Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127. Association for Computational Linguistics.

Authorship Identification for Literary Book Recommendations

Haifa Alharthi, Diana Inkpen, Stan Szpakowicz
University of Ottawa, Ottawa, Ontario, Canada
{halha060,Diana.Inkpen,szpak@eecs}.uottawa.ca

Abstract

Book recommender systems can help promote the practice of reading for pleasure, which has been declining in recent years. One factor that influences reading preferences is writing style. We propose a system that recommends books after learning their authors' style. To our knowledge, this is the first work that applies the information learned by an author-identification model to book recommendations. We evaluated the system according to a top-k recommendation scenario. Our system gives better accuracy when compared with many state-of-the-art methods. We also conducted a qualitative analysis by checking if similar books/authors were annotated similarly by experts.

1 Introduction

Recommender systems (RSs) are useful for internet users who may find it hard to choose from the multitude of available products and services. RSs predict how likely the target user is to be interested in an item which might have been unknown to her. In this work, we consider book recommender systems, which could be useful in libraries, schools, and on e-learning portals. Nowadays, with the introduction of e-books, readers can access inexpensive resources with little effort. It was expected that the act of reading for pleasure would become widespread, but statistics demonstrate the opposite; it is declining, particularly among young people.[1]

A number of studies have shown the benefits of reading. In a comparison between the well-being of 7,500 Canadian adult readers and non-readers, the former have been found statistically significantly more likely to report better health or mental health, to volunteer, and to feel strongly satisfied with life (Hill, 2013). Fiction has been shown to stimulate profound social communication (Mar and Oatley, 2008). Exposure to fiction correlates with a greater ability for empathy and social support (Mar et al., 2009). It also has a lingering biological effect, notably in the connectivity of the brain (Berns et al., 2013). It is therefore a worthwhile endeavor to deploy artificial intelligence techniques to spark people's interest in books by recommending the right type of books.

There are two main types of traditional recommender systems: collaborative filtering (CF) and content-based recommendation (CB). When a CF system predicts if a user likes a book, it relies on the ratings of all active users in the system. CF depends on a rating matrix which could be large and sparse, and that might lead to inferior recommendations. Sparsity occurs when items/users do not have enough ratings. It can be noticed in library records that many books are never checked out (as is, *e.g.*, the case of 75% of the books in the library of Changsha University of Science and Technology (Yang et al., 2009)). A few other problems face CF systems: *new items* which are difficult to recommend until enough ratings have been received, *the long tail* when most items are overlooked because of few very popular ones, *shilling attack* when items are rated as a way of promotion, and the *gray sheep problem* of users who have tastes entirely different from other users (Alharthi et al., 2017).

Content-based recommendation, on the other hand, relies on the content of items (*e.g.*, a book's genre and author). It is a classifier that finds patterns in the target user's past reading preferences, and predicts

[1] http://tinyurl.com/jgunwfx

Proceedings of the 27th International Conference on Computational Linguistics, pages 390–400
Santa Fe, New Mexico, USA, August 20-26, 2018.

future interests. CB does not require a rating matrix; that eliminates all the aforementioned issues with CF. A newly added book is recommended if it is similar to the target user's preferences. CB also makes recommendations that are interpretable (*e.g.*, a book is recommended because the user tends to like a specific genre). However, it is difficult for CB to recommend items that have inadequate content. Another weakness, *overspecialization*, occurs when the system makes non-diverse recommendations (Alharthi et al., 2017). Hybrid RSs can combine CF and CB to tackle such issues and enhance the recommendation accuracy.

A content-based RS is proposed that analyzes the texts of books to learn users' reading interests. A survey on book recommender systems (Alharthi et al., 2017) describes only a few RSs that take the actual text of books into account. There is a lot of work that explores literary works, including automatic genre identification (Ardanuy and Sporleder, 2016) and learning the narrative structure (Elson et al., 2010). It is quite surprising, then, that the analysis of the textual content of books to improve their recommendations is still quite limited. A user's reading preferences might be influenced by many elements specific to books. For example, recommendations given by the readers' advisory (a library service that suggests books to patrons) rely on multiple appeal factors. The factors are *characterization, frame, language and writing style, pacing, special topics, storyline,* and *tone*. To obtain information about books' appeal factors, librarians can subscribe (at a fee) to reader-advisory databases such as NoveList, established by professionals (Pera and Ng, 2014a).

Commercial applications that perform natural language processing (NLP) on the text of books have begun to appear; one example is BookLamp which was acquired by Apple in 2014.[2] Also, the well-known Google Books provide a retrieval system that searches inside the full text of books for terms that appear in a given query. Moreover, e-book recommendations are already provided via WIFI-connected Kindle, Kobo and other e-readers. Our proposed system could therefore be deployed by e-book providers, and could help current online stores to boost their deployed collaborative filtering systems.

The basic approach in book retrieval systems is to adopt the bag-of-words method, which creates book representations based on term frequencies. In this case, cosine similarity is often used to retrieve the most similar book representations. More advanced document retrieval methods such as latent Dirichlet allocation (LDA) (Blei et al., 2003) and latent semantic indexing (LSI) (Deerwester et al., 1990) also take advantage of term frequencies. Recently, doc2vec models (Le and Mikolov, 2014) have achieved state-of-the-art results in document classification and recommendations (Gupta and Varma, 2017; Wang et al., 2016).

In this paper, we focus on learning authors' writing styles in aid of book recommendation. Our book RS transfers information learned by an authorship identification (AuthId) classifier to a book recommendation module. It is common in neural network literature, especially in image processing, to train a model (source model) on a dataset for a specific task and then transfer the learned features to another model (target model) working with a different dataset and task. The features are considered *general* if they are learned from first layers in neural networks. *General* features tend to hold basic information (*e.g.*, color blobs in image processing), and are suitable to work on a different dataset/task. *Specific* features, on the other hand, generated by the last layers, are very dependent on the dataset/task (Yosinski et al., 2014). Transferability in NLP applications is explored in (Mou et al., 2016) which concludes its usefulness only for tasks that are mutually semantically similar.

The system has two components: authorship identification and book recommendation. For the former, we adopt two approaches similar to (Solorio et al., 2017); they use convolutional neural networks (CNN) over a sequence of words and sequence of character bigrams. Given the text of a book as input, the AuthId classifier predicts its author, and once it has achieved good accuracy, we extract features from the last hidden layer (before the output layer) and use them for another task (book recommendation). These book features, then, are *specific*. In fact, the first task, author identification, is just a way of representing books as vectors that encode information about their authors' writing styles. The recommendation module, on the other hand, is a content-based RS which makes recommendations after finding patterns in the representations of books read by the target user. To achieve this, a regressor is trained over book AuthId

[2]http://www.businessinsider.com/apple-buys-booklamp-2014-7

features associated with the target user ratings to generate a list of ranked books.

This paper makes several contributions. To our knowledge, this is the first work to transfer the information learned from an author identification model to book recommendations. We evaluated the system according to a top-k recommendation scenario. Our system gives better accuracy when compared against many state-of-the-art methods, namely the vector space model (VSM) (Salton and McGill, 1986), latent Dirichlet allocation (LDA) (Blei et al., 2003), latent semantic indexing (LSI) (Deerwester et al., 1990) and paragraph2vec (Le and Mikolov, 2014). Moreover, qualitative analysis was conducted on the vectors generated by the author identification model. The books with similar vectors have been found to have similar description in NoveList.

The remainder of this paper is organized as follows. Section 2 surveys related work in recommender systems and author identification. Section 3 explains the proposed system in detail. Section 4 describes the evaluation process. Section 5 illustrates the results and analyzes them. Section 6 concludes, and highlights future work.

2 Related work

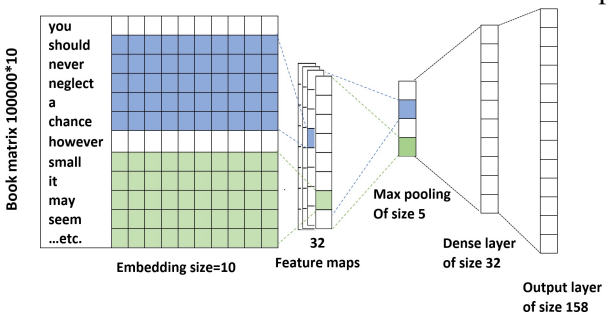
Only a few of more than thirty book RSs surveyed in (Alharthi et al., 2017) take the actual text of books into account. In addition to library circulation analysis, many researchers took advantage of the textual reviews shared by individuals and communities. Another category of book RSs considers stylometry features; they are learned from an author's writing, and can help in authorship identification. Stamatatos (2009) categorizes stylometry features into five classes: lexical, syntactic, semantic, application-specific and character-based. In the stylometric content-based (CB) book RS proposed by (Vaz et al., 2012a), a few features are taken into account. Books were represented in two ways: (1) as vectors of LDA topics, (2) as vectors of stylometric features (document length, vocabulary richness, part-of-speech bigrams and most frequent words). Experiments conducted using the Rocchio algorithm (Manning et al., 2008) on LitRec dataset (Vaz et al., 2012c) showed that the combination of stylometric CB with collaborative filtering (CF) provides better accuracy than an individual CF or CB system. Compared to other representations, the LDA-based vectors showed the best results. Unlike the method we propose, (Vaz et al., 2012a) does not associate stylometric features found in books with authors.

The author's writing style was also considered in (Pera and Ng, 2014a; Pera and Ng, 2014b; Pera and Ng, 2015), yet it is learned from the online reviewers' point of view and not automatically from the textual content of books. Appealing terms (from a fixed number of terms found in (Pera and Ng, 2014a)) are extracted from readers' reviews automatically collected from websites such as Amazon.com. The resemblance between vectors of appealing terms of the preferred and candidate books is calculated, and a recommendation is made accordingly.

Upon receiving the name of the user's favourite author, (Zhang and Chow, 2015) exploit the whole text of an e-book to suggest authors and e-books. To overcome the spatial distribution issue—when words are treated without taking into consideration their order—every author is represented in a hierarchical structure which consists of four layers. The first layer is reserved for the author's related information (*e.g.*, education and political views), while the following three levels are dedicated to the author's books, pages of every book, and paragraphs on each page. The author representations are developed using a multilayer self-organizing map (MLSOM) (Rahman et al., 2007). To recommend books, the system considers the last three layers. On the other hand, to suggest authors, the RS exploits all the layers. The evaluation is conducted by computing the relevance of two books. If a queried book has the same genres as the recommended book, this is considered a relevant recommendation. The results show higher performance when compared with other CB systems, including those using VSM and LSI.

The baseline systems are commonly used in recommender systems. The vector space model, which is the standard approach in information retrieval, was applied to descriptions of books in (Tsuji et al., 2014; Pera and Ng, 2014b), but not to the whole text of books. Both latent semantic indexing and latent Dirichlet allocation are topic-modelling techniques widely used in information retrieval and recommender systems. To name a few, they were used to recommend movies (Bergamaschi and Po, 2015) and articles (Lin, 2017; Nagori and Aghila, 2011). Recently, paragraph2vec—or, as it is frequently

Figure 1: The use of a convolutional neural network for authorship identification



named, Doc2vec—was proposed in (Le and Mikolov, 2014) to offer a fixed-size representation for texts (paragraphs or documents). An unsupervised algorithm predicts the words in a document and generates a vector for every document. Doc2vec was applied in RSs to suggest scientific articles in (Gupta and Varma, 2017) and answers in question-answering systems in (Wang et al., 2016).

A simple system using CNN is proposed in (Solorio et al., 2017). It takes a sequence of characters of a tweet as input and predicts its author, a Twitter user. The system performance with different inputs (including character bigrams, character unigrams and words) is evaluated. The character bigram input gave the best accuracy. Although it has been proposed for short texts, we noticed that it gives acceptable accuracy when predicting authors of books; that is why we have adopted it. As explained in section 3.1, we modified the network in (Solorio et al., 2017) by adding a dense layer which helps extract book AuthId features.

Recent research also shows how the use of recurrent neural networks (RNN) has led to accurate author identification. The work in (Qian et al., 2017) achieved 89% accuracy using Gated Recurrent Unit (GRU), an RNN algorithm, on a dataset from the Gutenberg Project (Lahiri, 2014). Their model represents a sequence of words (initialized as GloVe pretrained embeddings) in a sequence followed by average pooling, and then another GRU that represents the sequence of sentences in an article. A simple recursive neural network adopted in (Macke and Hirshman, 2015) showed high accuracy only when the number of classes was small: 10 authors. Our preliminary experiments show that RNN-based author identification models have poor accuracy, so we do not use such models in this work.

3 Methodology

This section describes the two components of the system. It first illustrates and explains the author identification system, which was proposed by (Solorio et al., 2017), and our modifications to the system. Next, the recommendation procedure is described.

3.1 Author Identification

As shown in Figure 1, drawn similarly to (Kim, 2014), the neural network has the following layers.
Embedding layer (also called *lookup table*). It takes a sequence of words or character bigrams as input, and maps each word or bigram to an embedding of size *k*. The embedding of the *i*th word/character is a dense *k*-dimensional vector $x_i \in R^k$. Each book is represented as a matrix, with one word/character embedding per row. A book has sequence length *n* (*n* is the number of words/characters) where a sequence $x_{1:n} = x_1 \oplus x_2 ... \oplus x_n$ is a concatenation of all words from x_1 to x_n.
Convolution layer. This layer consists of one or more filters (also called kernels) that are applied to windows of words to generate feature maps. Let a filter $w \in R^{hk}$ slide over a window of *h* words $x_{i:i+h-1}$ to generate feature $c_i = f(w \cdot x_{i:i+h-1} + b)$. f is the activation function (non-linear), and b is a bias. A feature map $\mathbf{c} = [c_1, c_2, ...c_{n-h+1}]$ is created by a filter *w* that slides over all the possible windows of words in a book.
Pooling layer. This layer decreases the dimensionality of the feature map. One approach is max-over-time pooling. Given a feature map **c**, it returns $max\{\boldsymbol{c}\}$; as a result, only the features with the highest importance (maximum value) are kept (Kim, 2014).

Fully-connected layer (also called *dense layer*). Each neuron in this layer is connected to each neuron in the previous layer. This layer is essential in our framework, because it generates the book representation. The pooling layer's output is flattened first, then fed to the dense layer which produces a fixed-size vector with dimensions equal to the number of neurons in the layer.
Output layer. It is a fully-connected softmax layer which outputs vector with dimensions equal to the number of authors (labels). Each dimension represents the probability that the input book belongs to a specific author. The values of all elements of one vector sum to 1.

After the model has been trained and achieved accurate predictions, it is used to extract the features of an intermediate layer. Given the text of a book as an input, the fixed-size vector generated by the fourth layer is extracted. This vector is considered as the AuthId book representation which is used in the next step to make recommendations.

3.2 Book Recommendations

Given a user's reading history associated with AuthId book representations, a regressor predicts her future ratings. Book recommendations are ranked according to the predicted rating values. We applied Support Vector Regression (SVR) which is an extension of Support Vector Machine (SVM) (Vapnik et al., 1996). In a non-linear SVR, the training samples X are mapped to a high-dimensional feature space which allows it to learn a linear model in that space. The mapping is achieved by a kernel function such as Radial Basis Function (RBF)—see Equation 1. For every AuthId book representation x_i that has the rating y_i, the SVR algorithm aims to learn a function f(x) as in Equation 2 with α_i^*, where α_i are Lagrange multipliers and N is the number of data points (Gunn, 1997; Basak et al., 2007).

$$k(x_i, x_j) = exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \tag{1}$$

$$f(x) = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)k(x_i, x) + b \tag{2}$$

4 Evaluation

4.1 Dataset and Preprocessing

We use the Litrec dataset (Vaz et al., 2012c) which contains data on 1,927 users who rated 3,710 literary works. The dataset incorporates data from Goodreads and from Project Gutenberg. It also has the complete texts of books, labelled with part-of-speech tags (Vaz et al., 2012b). In Goodreads, books are rated on a scale of 1-5 where 1-2 indicate a dislike and 3-5 a like. A book can have a rating of 0 to indicate that the user has read the book but not rated it. We filtered out all zero-rated books (17,976 ratings). Also, in order to train the author identification network, we need multiple books per author. Thus, we only kept authors with a minimum of three books. Data for users with fewer than 10 ratings are also deleted. The lowest number of ratings needed to develop CB with quality recommendations is 10, a threshold adopted by many researchers, including (Wang et al., 2009). The remaining 351 users rated 1010 unique items authored by 157 distinct authors.

The Gutenberg texts have copyright information at the beginning and the end, which we removed using heuristics or manually. The part-of-speech tags are also removed. Some books are very long; the maximum is 565,570 words, while the average is 99,601 words per book. Because of high memory requirement, the processing of large books resulted in a system crash. That is why we considered only the first 100,000 words of each book—slightly higher than the average length.

4.2 The Experimental Setting

A recent survey (Zhang et al., 2017) states that the percentage of RS publications that report ranking metrics is 66%, rating prediction metrics is 28%, and usage metrics 6%. We adopt ranking metrics to evaluate our system using a top-k recommendation scenario where systems provide a user with a ranked list of books. Books rated by a target user are divided into training and test set. The system learns from

the books in the training set and ranks the remaining books in the dataset (we call it the ranked list). The books in the test set are considered more relevant than the unread books on the ranked list; so, an ideal system would rank items from the test set in the top-k list of books. Similar to many related projects, we set k to 10. Three-fold cross-validation is adopted per user, and the results are averaged. We measure the statistical difference in results, using the t-test at a p-value of 0.05 or 0.01.

Metrics. We compute the recommendation accuracy by precision at k (P@k) and recall at k (R@k)—Equations 3-4—where a *relevant* book means to a preferred book, and *recommended* means ranked in the top k list.

$$P@k = \frac{\#\ relevant\ books\ in\ top\ k\ recommended}{k} \tag{3}$$

$$R@k = \frac{\#\ relevant\ books\ in\ top\ k\ recommended}{\#\ of\ relevant\ books} \tag{4}$$

Baselines. To implement LDA, LSI, VSM and Doc2vec, we used gensim,[3] a Python library. The texts of books were tokenized and down-cased, and NLTK stopwords and least frequent words were filtered out. In the first three systems, relevant books in the user training set are compiled and considered as one query. Using cosine similarity, the top-k books most similar to the query are recommended.

For LDA and LSI, we experimented with 10, 50, 100 and 200 topics; the best accuracy is reported. The Doc2vec model was trained on book texts; a book id is considered as the label. For training the model, we tried dimensions of 100 versus 300 and window size of 10 versus 5, and we report the best results. The average of document vectors of relevant books in the user training set is considered as a query. The books with highest cosine similarity to this vector are recommended. We also compare with a plain author-based RS which uses SVR similarly to our proposed system, with one difference: instead of book AuthId representations, author ids are used.

Parameter Settings. One question usually raised with transfer learning is this: when does one stop training the source model? We stop training the author identification model at the point when the validation accuracy stops increasing for five epochs, or if validation accuracy keeps increasing while the training accuracy becomes close to 100%. The network is trained using the RMSprop optimizer over 32 batches. For the non-linearity, rectified linear units are adopted. The number of neurons in the first fully connected layer is decided empirically to be 32. We also experimented with various combinations of parameters: embedding size = 1, 5, 10, number of filters = 16, 32, kernel size = 2, 5, and maximal pooling size = 2, 5.

In the char-bigram CNN, the combination that provides the best validation accuracy (64%) is reached at the 7^{th} epoch with embedding size of 10, 16 filters and kernel size and max-pooling sizes of 2. The sequence character bigrams can be very lengthy (this causes the system to crash); therefore, a maximum of 150,000 tokens is imposed. For the word CNN, a validation accuracy (65%) was achieved at the 17^{th} epoch when using embedding size of 10 with 32 filters and 5 for kernel size and max-pooling size. These experiments were implemented using keras,[4] a Python library, on a NVIDIA GeForce Titan X Pascal GPU with memory of 12,184 MiB.

The SVR was developed using scikit-learn.[5] The Radial Basis Function (RBF) is used with gamma=0.001. For some users who do not have negative ratings, the regressor ended up not distinguishing between relevant and irrelevant items. To solve this issue, we include in the training stage some randomly selected books not read by the target user to work as irrelevant books (excluded from baselines as well). The final number of irrelevant books in the training set is the same as relevant ones.[6]

5 Results and Analysis

Figure 2 illustrates how the proposed system, whether based on words (AuthId_words) or character bigrams (AuthId_char), retrieves relevant books more than the baselines, with the former achieving statis-

[3]https://radimrehurek.com/gensim/

[4]https://keras.io/

[5]http://scikit-learn.org/stable/

[6]To download the code and dataset, visit `https://tinyurl.com/ybsdxg5a`

Figure 2: Precision@10 and recall@10 generated by our system (AuthId_words and AuthId_char), and the baselines.

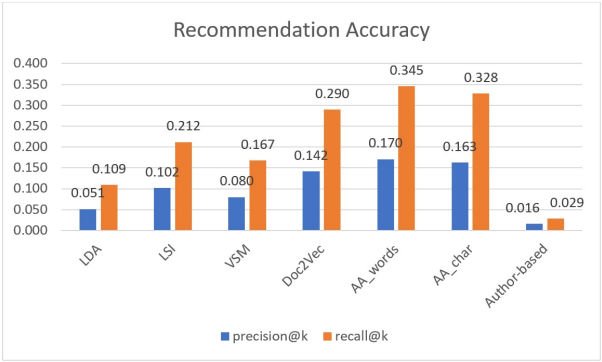
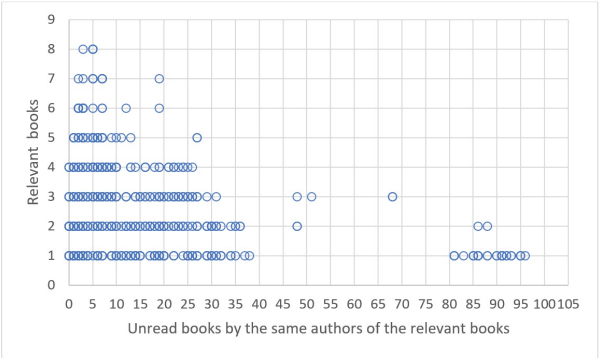


Figure 3: Users' relevant recommended books versus unread books by the same authors



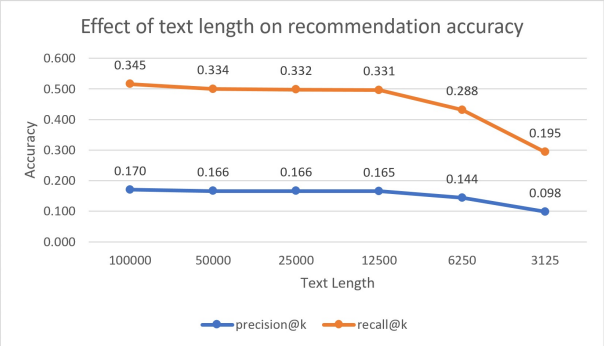
tically higher accuracy at a p-value of 0.01. In fact, AuthId_words score better precision and recall at a p-value of 0.05 when compared with AuthId_char. One possible reason is that in the latter we capped sequence length at 150,000. We also believe that the use of words could capture topical similarity, as highlighted by the qualitative analysis (Table 1).

Many users have fewer than 10 books in their test set. This means that P@10 would never become 1, and also explains why R@10 is greater than P@10. On the other hand, some users have more than 10 books in their test set, making the R@10 very low even for an ideal system. The best baseline is doc2vec, followed by LSI. LDA's low accuracy is surprising, yet it is possible that more preprocessing is required for LDA to work properly. We expected the author-based system to have high recall@k by just assigning high predictions to the target user's favorite authors, but the performance is the poorest. A closer look shows that a preferred author might have many books not read by the user, and when the author-based system recommends a random sample of these books, many of them are considered irrelevant (not read by the user).

This observation has led us to ask how many unread books there are for the authors of books retrieved by our system. To investigate, for each user we obtain the authors of her relevant recommended books, and count the unread books they wrote on the list of books to rank. This analysis is shown in Figure 3 where one circle refers to one test case (one fold for one user). The y-axis represents the number of relevant books in the top 10 recommended list. The x-axis refers to the number of unread books by the same authors who wrote the relevant books. For example, the mark at (3, 68) means that the AuthId_words system could recommend three books relevant to a target user from a list of items containing 68 books written by the same authors as the three retrieved books. In 302 cases, the system could retrieve one or more relevant books from a list with more than ten irrelevant books by the same authors. In 12 cases, the number of unread books exceeded 80. The system recommended at most eight relevant books; that was achieved when three and five irrelevant books were on the list of books to rank.

To assess the effect of text length on the accuracy of recommendations, we developed book AuthId

Figure 4: Effect of text length on recommendation accuracy



representations using fewer texts. We iteratively divided the length by half and fed it to the author identification model. The model was chosen after searching for the most accurate combination of embedding size, number of filters, kernel size and pooling size as in section 4.2. In the author identification model, the validation accuracy of the 100,000 words is similar to 3125 words. However, figure 4 shows that *the shorter the text length, the less accurate the recommendations*. Yet, a statistical difference only occurs when a great deal of the text is removed (*i.e.*, 6250 and 3125 words).

We went further to analyze the quality of AuthId book representations by measuring if similar representations have overlapping descriptions in NoveList. Using cosine similarity, we studied the 10 books most similar to "The Snow-Image: A Childish Miracle" by Nathaniel Hawthorne. We selected this book because NoveList has information on all its related authors. In Tables 1-2, which represent book AuthId using CNN_words and CNN_char_bigrams respectively, one can see Gutenberg book IDs in descending order according to their similarity values, as well as author information.

In both tables, Hawthorne himself authored the first four books. The following list of books by the two methods, however, are entirely different from each other. Table 1 has books by only three authors with many common points with the description of Hawthorne's profile (in Bold). The authors of the first four books and the last five books write mostly about related topics (subject headings). On the other hand, Table 2 contains information on seven unique authors, one sharing the same genre and two having the opposite storyline and pace. Yet, most of them write about different topics than Hawthorne. It is noticeable that the use of words rather than characters captures similarity at the topic level. Solorio et al. (2017) further assess the AuthId model.

6 Conclusion

In the work discussed in this paper, we represent books as vectors learned in relation to authors. Such representations are not only useful in content-based RSs as the results have shown, but can be used to enrich current collaborative filtering systems. The experiments show that AuthId book representation gives better precision and recall compared to LSI, LDA, VSM and Doc2vec. Author writing style may change with time or when writing in different genres. Here, we trained the model to predict the authors regardless of the genre, topics or time of their writing. Taking into consideration these factors may help develop a more accurate author identification model, which is expected to result in better book representations. Other ways for authorship identification should be investigated in the context of book recommendations. More advanced ranking methodologies could be adopted, such as pairwise or list-wise approaches. One possible approach is to adopt multi-task learning where a neural network has two outputs, namely author name and user preferences. We have tried to use NN models to predict user reading preferences from the text of books, but we could not achieve high accuracy in the top-k scenario. We think that performance would improve if the system could work on a larger dataset with more user-item interactions.

Table 1: Author information of books similar to book #30376 using CNN_words

Book id (similarity)	Author information on NoveList
30376 (1)	Author: Nathaniel Hawthorne Genre: **Classics**; **Historical fiction** Character: Brooding; **Complex**; Flawed Storyline: Intricately plotted Pace: Leisurely paced; Tone: **Atmospheric**; Melancholy; **Thought-provoking**; Writing Style: **Descriptive**; **Richly detailed**; Stylistically complex Subject headings: Married women, Puritans – New England, Revenge,Sin, Physicians, Husband and wife, Pariahs, Clergy, Extramarital relations – New England, **Love triangles**, Atonement, Secrets, Ostracism, Villages – New England, Prynne, Curses, **Families**, Haunted houses Location:Massachusetts–History–Colonial period, Massachusetts, **New England**
1916 (0.98) 13707 (0.92) 25344 (0.95)	
2015 (0.88)	Author: G. K. Chesterton Genre: **Classics**; Literary fiction; Mysteries; Mystery classics; Short stories; Surrealist fiction Storyline: Unconventional Tone: **Thought-provoking** Writing Style: Compelling; **Descriptive**; Witty Subject headings: Anarchists, Conspiracies, Secret societies ... etc; Location: London, England, England
11104 (0.87) 4519 (0.87) 4518 (0.86) 4517 (0.86) 1263 (0.85)	Author: Edith Wharton Genre: Literary fiction; Modern classics Character: **Complex**; Storyline: Character-driven Tone: **Atmospheric**; Bittersweet; Strong sense of place Writing Style: **Descriptive**; Lyrical; **Richly detailed** Subject headings: Men/women relations, **Love triangles**, Married men, Marriage, Socialites, Separated women (Marital relations), Family relationships, Manners and customs Location: New York City – Social life and customs – 19th century, **New England**

Table 2: Author information of books similar to book #30376 using CNN_char_bigrams

Book id (similarity)	Author info on NoveList
30376 (1) 1916 (0.96) 25344 (0.959) 13707(0.914)	Author: Nathaniel Hawthorne previous table
22629 (0.775)	Author: Edward Elmer Smith Genre: Science fiction; Science fiction classics Storyline: Plot-driven; World-building Pace: Fast-paced Tone: Dramatic Subject headings: Space warfare, Human/alien encounters, Genocide, Life on other planets, Sexism ... etc.
2729 (0.759)	Author: H. Rider Haggard Genre: Adventure stories; **Classics; Historical fiction**; Science fiction; Science fiction classics Storyline: Action-packed; Plot-driven Pace: Fast-paced Tone: Strong sense of place Writing Style: **Richly detailed**
3735 (0.753)	Translated from author: Nicolay Gogol Genre Anthologies, **Classics**, Short stories, Translations Location: Russia, St. Petersburg, Russia
11605 (0.748)	Author: G.K. Chesterton previous table
2786 (0.739)	Author: Louisa May Alcott Genre: **Classics** Storyline: Character-driven Tone: Feel-good; Moving Subject headings: **Families**, Sisters, Young women, Girls – New England–History Location: **New England** – Social life and customs – 19th century
12204 (0.73)	Author: W.W. Jacobs Genre: **Classics**; Ghost stories; Horror; Horror classics Subject headings: Supernatural, Ghosts, Wishing and wishes, Fate and fatalism, Dead, Mummified animals . . . etc.

References

Haifa Alharthi, Diana Inkpen, and Stan Szpakowicz. 2017. A survey of book recommender systems. *Journal of Intelligent Information Systems*, pages 1–22, 9.

Mariona Coll Ardanuy and Caroline Sporleder. 2016. Clustering of Novels Represented as Social Networks. *LiLT (Linguistic Issues in Language Technology)*, 12(4).

Debasish Basak, Srimanta Pal, and Dipak Chandra Patranabis. 2007. Support Vector Regression. *Neural Information Processing – Letters and Reviews*, 11(10):478–486.

Sonia Bergamaschi and Laura Po. 2015. Comparing lda and lsa topic models for content-based movie recommendation systems. In Valérie Monfort and Karl-Heinz Krempels, editors, *Web Information Systems and Technologies: 10th International Conference, 2014, Revised Selected Papers*, pages 247–263. Springer.

Gregory S. Berns, Kristina Blaine, Michael J. Prietula, and Brandon E. Pye. 2013. Short- and Long-Term Effects of a Novel on Connectivity in the Brain. *Brain Connectivity*, 3(6):590–600.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3:993–1022, March.

S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R.A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41*, pages 391–407.

David K. Elson, Nicholas Dames, and Kathleen R. McKeown. 2010. Extracting Social Networks from Literary Fiction. In *Proc. 48th Annual Meeting of the ACL*, pages 138–147, Stroudsburg, PA, USA. Association for Computational Linguistics.

Steve R. Gunn. 1997. Support Vector Machines for Classification and Regression (Image Speech & Intelligent Systems Group, University of Southampton). `http://m.svms.org/tutorials/Gunn1997.pdf`.

Shashank Gupta and Vasudeva Varma. 2017. Scientific Article Recommendation by Using Distributed Representations of Text and Graph. In *Proc. 26th International Conference on World Wide Web Companion*, WWW '17 Companion, pages 1267–1268.

Kelly Hill. 2013. The Arts and Individual Well-Being in Canada, February. [Online; posted 13 February 2013].

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proc. 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Shibamouli Lahiri. 2014. Complexity of Word Collocation Networks: A Preliminary Structural Analysis. In *Proc. Student Research Workshop at the 14th Conference of the European Chapter of the ACL*, pages 96–105.

Quoc Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proc. 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org.

Sheng-Ting Lin. 2017. *Latent semantic analysis for retrieving related biomedical articles*. Ph.D. thesis, University of British Columbia.

Stephen Macke and Jason Hirshman. 2015. Deep sentence-level authorship attribution. `https://cs224d.stanford.edu/reports/MackeStephen.pdf`.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Raymond A. Mar and Keith Oatley. 2008. The Function of Fiction is the Abstraction and Simulation of Social Experience. *Perspectives on Psychological Science*, 3(3):173–192.

Raymond A. Mar, Keith Oatley, and Jordan B. Peterson. 2009. Exploring the link between reading fiction and empathy: Ruling out individual differences and examining outcomes. *Communications*, 34(4):407–428.

L. Mou, Z. Meng, R. Yan, G. Li, Y. Xu, L. Zhang, and Z. Jin. 2016. How Transferable are Neural Networks in NLP Applications? *ArXiv e-prints*, March.

R. Nagori and G. Aghila. 2011. LDA-based integrated document recommendation model for e-learning systems. In *Proc. 2011 International Conference on Emerging Trends in Networks and Computer Communications*, pages 230–233.

Maria Soledad Pera and Yiu-Kai Ng. 2014a. Automating Readers' Advisory to Make Book Recommendations for K-12 Readers. In *Proc. 8th ACM Conference on Recommender Systems*, RecSys '14, pages 9–16.

Maria Soledad Pera and Yiu Kai Ng. 2014b. How Can We Help Our K-12 Teachers?: Using a Recommender to Make Personalized Book Suggestions. In *Proc. 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) – Volume 2*, pages 335–342. IEEE Computer Society.

Maria Soledad Pera and Yiu-Kai Ng. 2015. Analyzing Book-Related Features to Recommend Books for Emergent Readers. In *Proc. 26th ACM Conference on Hypertext & Social Media*, HT '15, pages 221–230.

Chen Qian, Tianchang He, and Rao Zhang. 2017. Deep Learning based Authorship Identification (report, Stanford University.

M. K. M. Rahman, Wang Pi Yang, Tommy W. S. Chow, and Sitao Wu. 2007. A Flexible Multi-layer Self-organizing Map for Generic Processing of Tree-structured Data. *Pattern Recognition*, 40(5):1406–1424, May.

Gerard Salton and Michael J. McGill. 1986. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA.

Thamar Solorio, Paolo Rosso, Manuel Montes-y-Gómez, Prasha Shrestha, Sebastián Sierra, and Fabio A. González. 2017. Convolutional Neural Networks for Authorship Attribution of Short Texts. In *Proc. 15th Conference of the European Chapter of the ACL, EACL 2017, Volume 2: Short Papers*, pages 669–674.

Efstathios Stamatatos. 2009. A Survey of Modern Authorship Attribution Methods. *J. Am. Soc. Inf. Sci. Technol.*, 60(3):538–556, March.

Keita Tsuji, Nobuya Takizawa, Sho Sato, Ui Ikeuchi, Atsushi Ikeuchi, Fuyuki Yoshikane, and Hiroshi Itsumura. 2014. Book Recommendation Based on Library Loan Records and Bibliographic Information. *Procedia - Social and Behavioral Sciences*, pages 478–486. Proc. 3rd International Conference on Integrated Information.

Vladimir Vapnik, Steven E. Golowich, and Alex Smola. 1996. Support Vector Method for Function Approximation, Regression Estimation and Signal Processing. In *Proc. 9th International Conference on Neural Information Processing Systems*, NIPS'96, pages 281–287, Cambridge, MA, USA. MIT Press.

Paula Cristina Vaz, David Martins de Matos, and Bruno Martins. 2012a. Stylometric Relevance-feedback Towards a Hybrid Book Recommendation Algorithm. In *Proc. Fifth ACM Workshop on Research Advances in Large Digital Book Repositories and Complementary Media*, BooksOnline '12, pages 13–16.

Paula Cristina Vaz, David Martins de Matos, Bruno Martins, and Pavel Calado. 2012b. Improving a Hybrid Literary Book Recommendation System Through Author Ranking. In *Proc. 12th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '12, pages 387–388, New York, NY, USA. ACM.

Paula Cristina Vaz, Ricardo Ribeiro, and David Martins de Matos. 2012c. LitRec vs. Movielens – A Comparative Study. In *KDIR 2012 – Proc. International Conference on Knowledge Discovery and Information Retrieval, Barcelona, Spain, 4-7 October, 2012*, pages 370–373.

Yiwen Wang, Natalia Stash, Lora Aroyo, Laura Hollink, and Guus Schreiber. 2009. Using Semantic Relations for Content-based Recommender Systems in Cultural Heritage. In *Proc. 2009 International Conference on Ontology Patterns - Volume 516*, WOP'09, pages 16–28, Aachen, Germany, Germany. CEUR-WS.org.

J. Wang, C. Man, Y. Zhao, and F. Wang. 2016. An answer recommendation algorithm for medical community question answering systems. In *Proc. 2016 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, pages 139–144, July.

Xuejun Yang, Hongchun Zeng, and Weihong Huang. 2009. ARTMAP-Based Data Mining Approach and Its Application to Library Book Recommendation. In *Proc. 2009 International Symposium on Intelligent Ubiquitous Computing and Education*, pages 26–29, May.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How Transferable Are Features in Deep Neural Networks? In *Proc. 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3320–3328, Cambridge, MA, USA. MIT Press.

Haijun Zhang and Tommy W. S. Chow. 2015. Organizing Books and Authors by Multilayer SOM. *Neural Networks and Learning Systems, IEEE Transactions on*, PP(99):1–14.

S. Zhang, L. Yao, and A. Sun. 2017. Deep Learning based Recommender System: A Survey and New Perspectives. *ArXiv e-prints*, July.

A Nontrivial Sentence Corpus for the Task of Sentence Readability Assessment in Portuguese

Sidney Evaldo Leal sidleal@gmail.com
Magali Sanches Duran magali.duran@uol.com.br
Sandra Maria Aluísio sandra@icmc.usp.br
Institute of Mathematical and Computer Sciences - University of São Paulo
Av. do Trabalhador Saocarlense, 400, São Carlos - SP - Brazil

Abstract

Effective textual communication depends on readers being proficient enough to comprehend texts, and texts being clear enough to be understood by the intended audience, in a reading task. When the meaning of textual information and instructions is not well conveyed, many losses and damages may occur. Among the solutions to alleviate this problem is the automatic evaluation of sentence readability, task which has been receiving a lot of attention due to its large applicability. However, a shortage of resources, such as corpora for training and evaluation, hinders the full development of this task. In this paper, we generate a nontrivial sentence corpus in Portuguese. We evaluate three scenarios for building it, taking advantage of a parallel corpus of simplification, in which each sentence triplet is aligned and has simplification operations annotated, being ideal for justifying possible mistakes of future methods. The best scenario of our corpus PorSimplesSent is composed of 4,888 pairs, which is bigger than a similar corpus for English; all the three versions of it are publicly available. We created four baselines for PorSimplesSent and made available a pairwise ranking method, using 17 linguistic and psycholinguistic features, which correctly identifies the ranking of sentence pairs with an accuracy of 74.2%.

Title and Abstract in Portuguese

Um Corpus Não Trivial de Sentenças para a Tarefa de Avaliação de Complexidade Sentencial em Português

Uma comunicação textual eficaz depende de os leitores serem proficientes o suficiente para compreenderem o texto e de o texto ser claro o suficiente para ser compreendido pelo público-alvo, em uma tarefa de leitura. Quando o significado das informações e instruções textuais não é bem transmitido, muitas perdas e danos podem ocorrer. Entre as soluções para aliviar este problema está a avaliação automática da complexidade sentencial, tarefa que vem recebendo muita atenção devido a sua grande aplicabilidade. No entanto, a escassez de recursos, como corpora para treinamento e avaliação, dificulta o pleno desenvolvimento dessa tarefa. Neste artigo, geramos um corpus de sentenças não triviais em Português. Avaliamos três cenários para construí-lo, aproveitando um corpus paralelo de simplificação textual, no qual cada trio de sentenças está alinhado e possui operações de simplificação anotadas, sendo ideal para justificar possíveis erros de métodos futuros. O nosso melhor cenário do corpus PorSimplesSent é composto por 4.888 pares, que é maior que um corpus similar para o inglês; todas as três versões do corpus PorSimplesSent estão disponibilizadas publicamente. Criamos quatro métricas *baselines* para o PorSimplesSent e um método de ranqueamento por pares, utilizando 17 métricas linguísticas e psicolinguísticas, que identificam corretamente o ranqueamento dos pares de sentenças com uma acurácia de 74.2%.

Proceedings of the 27th International Conference on Computational Linguistics, pages 401–413
Santa Fe, New Mexico, USA, August 20-26, 2018.

1 Introduction

Readability is an issue of great social and economic impact. Effective textual communication depends on readers being proficient enough to comprehend texts, and texts being clear enough to be understood by the intended audience. When the meaning of textual information and instructions is not well conveyed, many losses and damages may occur (Dubay, 2007). In Brazil, for example, only 8% of adult population has reading proficiency (IPM, 2016). The situation is worse in the agriculture and livestock sectors, where only 1% of the surveyed are proficient readers. For this reason, most of rural producers do not have access to new technologies, undermining the development of agribusiness, which accounts for 22% of gross internal product and 30% of Brazilian jobs[1]. Research investments in these sectors, therefore, do not cause as much impact as they potentially might. Identifying which sentences of a text are more complex may help writers of newsletters, manuals and instructions, for example, to adequate their texts to their audiences.

Among the solutions to alleviate this problem is the simplification or adaptation of complex texts, a task that has been partially or fully automatized by Natural Language Processing (NLP) applications. For Brazilian Portuguese, various applications, methods and resources aiming to support simplification in several levels of readability were developed in the Project PorSimples (Aluísio and Gasperin, 2010). Among these resources there is a parallel and aligned corpus with two levels of simplification and annotated simplification operations (Caseli et al., 2009). PorSimples corpus has been used to train readability classifiers for texts (Scarton et al., 2010). Table 1 shows examples of an original sentence of PorSimples corpus (O), its natural simplification (N) and its strong simplification (S). The natural simplification had a substitution of "Uma parcela" by "Alguns" and the strong simplification, shorter than the natural, had a clause removed.

(O) Uma parcela critica o uniforme, porque acredita que ele ameaçaria a individualidade de cada um. (One parcel criticizes the uniform, because it believes that it would threaten the individuality of each one.)
(N) Alguns criticam o uniforme, porque acreditam que ele ameaça a individualidade de cada um. (Some criticize the uniform because they believe that it threatens the individuality of each one.)
(S) Alguns acreditam que o uniforme ameaça a individualidade de cada um. (Some believe that the uniform threatens the individuality of each one.)

Table 1: Examples of simplification in PorSimples.

However, we know that even complex texts have simple sentences, what makes it difficult to identify precisely where complexity lies. In an automatic simplification task, as well, it is difficult to decide which sentence is complex and requires simplification. To address these difficulties, a new task has received attention recently: the prediction of sentences readability, also known by sentence-based readability or sentential complexity task. The first studies on this subject emerged in the beginning of the last decade (Dell'Orletta et al., 2011; Sjöholm, 2012; Del'Orletta et al., 2014).

This task may support simplification systems at least in three applications: (i) to evaluate whether the simplification of a sentence (manual or automatic) is truly simpler than the original sentence or not; (ii) to inform the level of complexity of an original sentence; (iii) to rank the results of several simplification methods, according to their level of complexity. Besides supporting text simplification applications, computer-aided language learning (CALL) systems can benefit from sentence-level readability methods to predict which sentences of a text the students will struggle to read. Furthermore, Open Educational Resources repositories Wiley et

[1] http://www.ibge.gov.br/home/estatistica/economia/agropecuaria/censoagro/

al. (2014) may also take profit of such methods in order to return not merely relevant educational resources, but documents appropriate to the reading level of the user.

Due to its several applications, sentential complexity has been a focus of interest in the NLP studies in recent years, such as Vajjala and Meurers (2014), Vajjala and Meurers (2016), Ambati et al. (2016), Singh et al. (2016), Howcroft and Demberg (2017), Gonzalez-Garduño and Søgaard (2017).

The lack of a sentence-based corpus annotated with regards to readability is a major obstacle to research in this area for Portuguese. Even the English language suffers some drawbacks in what concerns the evaluation of sentential complexity. One of them is the use of benchmarks built from adapted corpora which are automatically aligned, such as Wikipedia and Simple Wikipedia (Zhu et al., 2010). This corpus has some problems to be used as benchmark for text simplification which also prevents its use for the sentential complexity task, for example, automatic sentence alignment errors, inadequate simplifications generating sentences which are not simple, and poor generalization for other genre than encyclopedia (Xu et al., 2015). Other benchmarks for sentential complexity, such as OneStopEnglish corpus (Vajjala and Meurers, 2016), have several positive points — the use of news articles which generalize better for other genres, not having sentence length as high predictive feature, as well as being available by requisition — but also can suffer from errors generated by automatic alignment. Newsela parallel corpus (cf. (Xu et al., 2015)), composed of news articles rewritten by professional editors to be read for children at multiple grade levels, is very beneficial for studying text simplification and could serve as benchmark for sentential complexity if the resulting sentence corpus could be publicly available. Moreover, Scarton et al. (2018) made available the SimPA, an English sentence level corpus for the Public Administration domain with 1,100 original sentences simplified in the lexical (3,300 pairs) and syntactic levels (another 1,100 pairs), annotated by 176 volunteers.

In this paper, we aim at obtaining nontrivial sentence pairs in Portuguese in order to create a gold standard corpus, publicly available. By nontrivial we mean that the pairs are not significantly different in length to avoid the easy judgment that the shorter sentences are the simpler ones. Although it is natural to expect that the simplified sentences are smaller, we found that it is not always true. An example of this is when, in order to simplify a content, one inserts an explanation, examples, or a list of synonyms.

We evaluated three scenarios for building our gold standard corpus from PorSimples corpus, with special care for the split operation, because splitting can generate several short sentences from an original one. The first scenario is a corpus formed of pairs of original and simplified sentences in which, if the split operation is used, we repeat the original sentence to form pairs with each of the simplified sentences. In the second scenario we include pairs with all but the simplified sentences from the split operation. The last scenario is a corpus in which all simplification operations are allowed, but for splitting we only bring the longest simplified sentences to compose the pair original-simplified.

The remainder of this paper is organized as follows. Section 2 reviews the literature on sentence-based readability assessment and its evaluation corpora. Section 3 presents the parallel and aligned corpus of the PorSimples project and explains how we built three evaluation scenarios to create the PorSimplesSent, our corpus for sentence-based readability assessment in Portuguese. In Section 4 we discuss our baselines, our method and features extracted to evaluate the three evaluation scenarios. Conclusions and future work are presented in Section 5.

2 Sentence-based Readability Assessment and its Evaluation Corpora

Initially, sentence-based readability task was considered in isolation by several authors, each one studying a set of features and evaluating in specific corpora. Dell'Orletta et al. (2011) were the first to consider the task of complexity for the sentential level, comparing its difficulty in relation to the textual level, for Italian. They used the SVM method of the LIBSVM library to train a model with 7,000 sentences, half selected in the newspaper *La Repubblica* and half of

the newspaper *Due Parole*, the latter considered simple reading. Interestingly, features at the syntactic level had little influence on the classification of documents, but were very important for the sentential level. Training with 6,000 and testing against 1,000 sentences, they reached 78.2% accuracy at the sentential level. Sjöholm (2012) addressed the task for the Swedish, also using two sets of sentences. For evaluation, 3,500 sentences were taken from the Swedish corpus LäSBarT, considered simple, and 3,500 from the GP2006 (Göteborgsposten journal), considered complex, divided into seven parts, each part used for testing with the model trained in the other six. The best method was Sequential Minimal Optimization (SMO), which reached 83% accuracy. It is important to mention that using the same set of features to evaluate documents (simple and complex) instead of sentences, in the same corpus, they obtained 97% accuracy. Dell'Orletta et al. (2014) returned to the task, addressing the issue of textual genres. They used the same sets of features from the previous article (Dell'Orletta et al., 2011), but now adding three new corpora of different genres to the original journalistic genre: literary, didactic and scientific.

Vajjala and Meurers (2014) made the first evaluation using Wikipedia-Simple Wikipedia corpus, automatically aligned by Zhu et al. (2010). This corpus became the most-used resource for sentential complexity evaluation in the English language. It was created with the matching of the sentences of 65,133 articles of Simple Wikipedia and Wikipedia, using the measure TF-IDF with cosine similarity. For the choice of the alignment measure, they evaluated the performance of three similarity measures: TF-IDF, word overlap and Minimum Edit Distance (MED), against 120 pairs of manually annotated sentences. The accuracy of TF-IDF was above 90%. As a final result, they created 108,016 aligned sentences, annotated in two classes: complex or simple, and a complex sentence may be mapped to one or more simple sentences to handle sentence splitting. This corpus was updated by Hwang et al. (2015), reaching 150,000 pairs of aligned sentences.

Table 2 shows the state-of-the-art (SotA) results we were able to compile, which use Wikipedia-Simple Wikipedia corpus. In the table, the name of each study is listed with the method/baseline used and the accuracy results.

Study	**Method**	**Accuracy (%)**
Flesch-Kincaid	Baseline	72.30
Vajjala and Meurers (2014)	SMOReg	66.00
Vajjala and Meurers (2016)	RankSVM	74.58
Ambati et al. (2016)	SMO	78.87
Singh et al. (2016)	Logistic Regression	75.21
Howcroft and Demberg (2017)	Rank as Classification (RasC)	73.22
Gonzalez-Garduño and Søgaard (2017)	MultiTask MLP	**86.45**

Table 2: SotA results using Wikipedia-SimpleWikipedia corpus.

Vajjala and Meurers (2014) trained a SMO regression model for document complexity, which reached about 90% accuracy. They then applied the model at the sentence level, and even testing in datasets of several sizes, they only achieved 66% accuracy, creating a new *baseline* for the task. They concluded the reason for this low accuracy lies in the incorrect assumption that all Wikipedia sentences are more complex than Simple Wikipedia. Even so, this dataset has been used by several studies of sentence readability. As far as we could see, Gonzalez-Garduño and Søgaard (2017) presents the state-of-the-art for the task, using eye-tracking features together with linguistic and psycholinguistic ones.

Vajjala and Meurers (2016) returned to the task, proposing a new method for evaluating paired sentences based on *ranking*. They contributed with a new corpus of English sentences aligned in three levels, called OneStopEnglish (OSE), used for training and testing. The OSE corpus is a corpus of aligned sentences created from articles rewritten by teaching experts for English language learners at three reading levels (elementary, intermediate, advanced). They used 76 triplets of articles published between 2012 and 2014, resulting in a total of 837 written

sentences with three levels (OSE3). For the alignment, TF-IDF and cosine similarity were used, with values above of 0.7. In addition to OSE3, a second corpus (OSE2) was compiled, which resulted in 3,113 sentence pairs: elementary-intermediate, intermediate-advanced, and elementary-advanced. This corpus was divided in two parts: 65% of pairs for training and the rest for testing.

In addition to significantly improving the accuracy of the task (over 80%), they assessed the impact of linguistic (lexical, syntactic, morphosyntactic) and psycholinguistic features, confirming the importance of eight features in OSE2: AoA (Age of acquisition), CTTR (corrected Type-token ratio), number of subtrees, average length of clause, average word imagery rating, average word familiarity rating, average Colorado meaningfulness rating of words, average concreteness rating. It is important to note that sentence length was not predictive in OSE2 corpus, as in this dataset rewriting and paraphrasing were the most used simplification operations.

As may be seen in Section 3, for our corpus, traditional psycholinguistic features such as AoA, imagery, concreteness, familiarity, have not been used to rank the three types of sentence pairs of PorSimplesSent. We have, indeed, analyzed their contribution to distinguish the three sentence levels, using the resource created by Santos et al. (2017). However, the results were not discriminative. We hypothesize two reasons for this. One of them is related to characteristics of the resource, which has been created automatically based on existing psycholinguistic norms and may contain some bias. The other reason is related to characteristics of the corpus. The corpus PorSimples contain a lot of explanation relating to difficult words (this is a simplification strategy to deal with lexical complexity). However, once explained, the difficult words are repeated along the text. In PorSimplesSent, when there is a split operation, the explanations remain isolated, benefiting only the sentence they appear, whereas the other sentences containing the repetitions of difficult words remain lexically complex. In fact, the psycholinguistic features did not perform well in our corpus and, therefore, they were not chosen as best features for our method.

Table 3 shows SotA results we were able to compile, which use OSE2 corpus, automatically aligned by Vajjala and Meurers (2016). In the table, the name of each study is listed with the method used and the accuracy results, separated by OSE2 subcorpus. OSE(A-E) stands for pairs at the levels Advanced and Elementary; OSE(A-I) for pairs at Advanced and Intermediate levels; OSE(I-E) for Intermediate and Elementary, and OSE(All) for all three pairs. Howcroft and Demberg (2017) joined the subcorpus OSE(A-I) and OSE(I-E), calling it OSE_{near}.

Study	**Method**	OSE(A-E)	OSE(A-I)	OSE(I-E)	OSE(All)
			OSE_{near}		
Flesch-Kincaid	Baseline				69.6
Vajjala and Meurers (2016)	RankSVM				**81.5**
Howcroft and Demberg (2017)	RasC	**85.3**	74.6		77.9
Gonzalez-Garduño and Søgaard (2017)	Multitask MLP	68.5	61.9		

Table 3: SotA accuracy results using OSE2 corpus.

Vajjala and Meurers (2016) explored whether the types of simplification operations are different between Advanced sentences simplified to Intermediate, and Intermediate sentences simplified to Elementary, using OSE3 corpus. That is why we don't have explicit evaluation between these pairs nor between Advanced and Elementary sentence pairs in Table 3.

3 PorSimplesSent Corpus

3.1 PorSimples Corpus

In order to create the PorSimplesSent, our corpus for sentence-based readability assessment in Portuguese, and to train and evaluate methods to predict sentential complexity for this language, we took advantage of PorSimples corpus (Caseli et al., 2009; Aluísio and Gasperin, 2010).

PorSimples corpus consists of 2,915 original sentences simplified into two levels of complexity:

Natural and Strong. All the sentences are from informational texts, being 30% of scientific issues from newspaper Folha de São Paulo[2] and 70% of other issues from newspaper Zero Hora[3].

PorSimples corpus contains complete annotation of each operation made during the simplification process. This was facilitated by the Simplification Annotation Editor, developed in PorSimples project (Caseli et al., 2009). The editor allows the human simplifier to register decisions of lexical and syntactic simplifications, which include substituting words, merging and splitting sentences, deleting part of the sentence, rewriting sentences with other words, and changing constituents order. The editor has a list of operations that may be chosen by the human simplifier. Simplification process in PorSimples was instructed by simplification guidelines, advising how to turn sentences simpler (Specia et al., 2008). Examples show how to tackle with complex structures, like apposition, subordinate clauses, clauses initiated by non-finite verbs, passive voice, inversion of constituents order and embedded clauses.

In a totally annotated process, the alignment between the simplified sentences and their respective simplifications is systematically ensured. This ensured alignment, added to the fact that the corpus contains a large variety of simplification strategies, makes PorSimples a unique corpus, entirely appropriate to evaluate readability predictors.

3.2 Methodology

We created 4,968 pairs and 1,141 triplets of sentences, combining the three levels of PorSimples corpus: Original, Natural and Strong. Pairs and triplets have two or three different sentences aligned, being the Original the more complex in Original-Natural and Original-Strong pairs, and Natural the more complex in Natural-Strong pairs.

In theory, there should be 8,745 pairs (an original-natural, an original-strong and a natural-strong pairing for each of the 2,915 sentences) and 2,915 triplets (original-natural-strong). However, it occurred 3,777 pairs and 1,774 triplets containing at least two identical sentences, because some of the sentences were simplified only in one level or were not simplified at all (they were considered originally simple). Such pairs and triplets were removed from the corpus, which remained with 4,968 pairs and 1,141 triplets.

Table 4 shows what happened with the original sentences of the texts during the simplification process that gave origin to PorSimples corpus. Part of the sentences has not been simplified, possibly because the sentences were considered already simple. The other part is composed of the simplified sentences, which followed one of three possible paths: simplification in both levels (Natural and Strong) or in only one of them (Natural or Strong).

Application of Simplification Operations in PorSimples Sentences	Number of Sentences
NOT simplified in any level	372
Simplified in two levels	1,105
Simplified only in Natural Level	1,268
Simplified only in Strong Level	170
TOTAL	2,915

Table 4: Distribution of original sentences according to the level of simplification.

Additionally, in the PorSimples corpus, 3,873 sentences were simplified into two or more sentences, generating 5,938 sentences, distributed as shown in Table 5. The split leads to an increase of 53% in the overall quantity of simplified sentences.

Each of the resulting sentences is obviously simpler than the split sentence, however, differently from the other pairs, the sentences deriving from split are part and not an integral simplified version of the respective simplified sentence. To evaluate the effect of splitting on the accuracy of

[2]https://www.folha.uol.com.br
[3]https://gauchazh.clicrbs.com.br

Input/Output Levels	Input (A+B)	Non-split sentences (A)	Split sentences (B)	Sentences resulting from split (C)	Output (A+C)	Percentage Increase
Original/ Natural	2,372	1,543	829	1,992	3,535	49%
Natural/ Strong	1,501	782	719	1,621	2,403	60%
TOTAL	3,873	2,325	1,548	3,613	5,938	53%

Table 5: Distribution of sentences increase due to split.

the complexity assessment task, we created three versions of PorSimplesSent. The three versions are very similar, as they pair all the sentences with their respective simplified sentences. They differ in what concerns split sentences.

As we can see in Table 6, the first version, PorSimplesSent1, has 10,616 pairs, including a pair for each sentence resulting from split. The second version, PorSimplesSent2, has 4,968 pairs and, for split sentences, selects only the simplification with greatest score after applying a linear combination of total number of words and word overlapping count, as exemplified in the following. The third version, PorSimplesSent3, disregard all the split sentences and has 2,600 pairs.

Types of Pairs	PorSimplesSent1	PorSimplesSent2	PorSimplesSent3
Original-Natural	3,535	2,372	1,543
Natural-Strong	4,976	1,501	782
Original-Strong	2,105	1,095	275
TOTAL	10,616	4,968	2,600

Table 6: Distribution of pairs by level in the three versions of PorSimplesSent.

For example, given an Original sentence (O) simplified into two sentences in Natural level (N1 and N2):

- (O): O dormitório, de aproximadamente cinco metros por cinco metros, completa-se com um guarda-roupas de duas portas, uma mesa, um frigobar e um aparelho de ar-condicionado. (The dormitory, approximately five meters by five meters, is complete with a two-door wardrobe, a table, a minibar and an air-conditioner.)
- (N1): O dormitório tem mais ou menos cinco metros por cinco metros. (length: 11 words; overlapping: 7 words; score: 11+7=18) (The dormitory is about five meters by five meters.)
- (N2): O dormitório se completa com um guarda-roupas de duas portas, uma mesa, um frigobar e um aparelho de ar-condicionado. (length: 19 words; overlapping: 19 words; score: 19+19=38) (The dormitory is complete with a two-door wardrobe, a table, a minibar and an air-conditioning unit.)

For PorSimplesSent1, we generated 2 pairs: O-N1 and O-N2. For PorSimplesSent2, we generated 1 pair: O-N2. The original was paired with the sentence N2, which presented a score of 38, against a score of 18 of the sentence N1. For PorSimplesSent3 we did not generate any pair with these sentences.

4 Corpus Validation

4.1 Method

To validate the corpus and to contribute with an initial baseline for the task in Portuguese, we evaluated a simple, but successful approach, inspired by Vajjala and Meurers (2016) —

the pair-wise ranking. For sentential complexity, each sentence should receive a score from an ordinal list of complexity, which could be 1 to n, being n the most difficult. Once the ranking method receives a pair of sentences (with feature vectors) it will predict which one is simpler than the other. The problem of sentential complexity is reduced to the comparison of sentences pairs taken from a pool of sentences where the objective is to rank them according to their complexity, trying to minimize inversion of ranks. As these authors, we also chose the RankSVM algorithm implemented in SVM$^{\text{Rank}}$ (Joachims, 2006)[4], which presented the best results among the algorithms tested for the task in English. We gave the rank value 2 to the complex side and value 1 to the simplified side of each sentence pair.

4.2 Features

For this experiment, we evaluated previously the sets of Original, Natural and Strong simplified sentences of PorSimples Corpus, using two publicly available NLP tools for Portuguese to extract textual metrics, which can be used to aid the automated analysis of text readability: Coh-Metrix-Port 2.0[5] (Scarton et al., 2010; Aluísio and Gasperin, 2010) and Coh-Metrix-Dementia[6] (Aluísio et al., 2016), both based on Coh-Metrix (Graesser et al., 2004). Also, we were inspired by another tool named AIC[7], built in PorSimples project which defined several syntactic metrics to be used in evaluation of text readability. Then we chose the 17 features that presented a clear tendency (increase or decrease, depending on the feature) in the three levels compared (see Table 7 and 8) in order to train a predictor.

Table 7 shows mean values of syntactic metrics for Original (O), Natural (N) and Strong (S) sentence levels in PorSimples corpus. In the table, S stands for Number of Sentences, CpS for Clauses per Sentence, ApC for Apposition per Clause, DD for Dependency Distance, MaxNP and MeanNP for Max and Mean Noun Phrase, SC for Subordinate Clauses, MVPpS for Mean Verb Phrase per Sentence, NIV for Non Inflected Verbs, PSR for Postponed Subject Ratio and ISC for Infinite Subordinate Clauses.

Table 8 shows mean values of lexical and psycholinguistic metrics for Original (O), Natural (N) and Strong (S) sentence levels in PorSimples corpus. In the table, WpS stands for Words per sentence, SpCW for Syllables per Content Words and WbMV for Words before Main Verbs.

L	S	CpS	ApC	DD	MaxNP	MeanNP	SC	MVPpS	NIV	PSR	ISC
O	2372	2.62	0.07	48.24	9.87	5.84	0.38	2.24	0.31	0.085	0.179
N	3535	1.95	0.02	28.39	7.35	4.79	0.26	1.71	0.22	0.051	0.124
S	2402	1.74	0.01	22.16	6.48	4.39	0.24	1.55	0.21	0.052	0.117

Table 7: Distribution of corpus sentences according to the level (L) of simplification - Syntactic Metrics.

L	WpS	SpCW	WbMV	Yngve	Frazier	Honoré	Brunet
O	21.01	2.86	6.16	2.89	7.38	1214.16	40.29
N	14.77	2.74	4.09	2.43	6.64	727.87	51.44
S	12.79	2.76	3.73	2.32	6.48	563.98	52.14

Table 8: Distribution of corpus sentences according to the level (L) of simplification - Lexical, Psycholinguistic and the Classic Syntactic Metrics of Yngve and Frazier.

The features are from three different groups: 1-4 are lexical; 5-16 measures syntactic complexity, and the last one is a psycholinguistic measure of working memory overload:

[4] `https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html`
[5] http://143.107.183.175:22680
[6] http://143.107.183.175:22380
[7] http://conteudo.icmc.usp.br/pessoas/taspardo/NILCTR0808.pdf

1. **Syllables per content word**: Average number of syllables per content word;
2. **Words per sentence**: Number of words in the sentence;
3. **Brunet**: Classic formula, its a type token ratio form less sensitive to text size (Thomas et al., 2005);
4. **Honoré**: Classic formula similar to Brunet but vocabulary-based (Thomas et al., 2005);
5. **Mean verb phrase per sentence**: Measures the quantity of verb phrases per sentence (implemented via tagger, counts verbs in a sentence);
6. **Yngve**: Measures how much a syntactic tree escapes from the pattern that tend to have branches to the right (Yngve, 1960);
7. **Frazier**: A bottom-up approach to calculate syntactic complexity of a sentence (Frazier, 1985);
8. **Dependency distance**: Calculates dependency distances in the syntactic tree; as dependency distances grows, the text complexity grows together;
9. **Apposition per clause**: Number of appositions in the sentence divided per number of clauses;
10. **Clauses per sentence**: Number of clauses in a sentence (implemented via parser Palavras (Bick, 2000); counts main verbs, excluding auxiliary verbs);
11. **Max noun phrase**: Maximum length of noun phrase in a sentence, calculated in words;
12. **Mean noun phrase**: Mean of noun phrase length in a sentence, calculated in words;
13. **Postponed subject ratio**: Occurrence of Verb-Subject order instead of canonical Subject-Verb order, calculated in relation to the total number of clauses;
14. **Subordinate clauses**: Proportion of subordinate clauses to the total number of clauses;
15. **Infinite subordinate clauses**: Proportion of subordinate clauses made by verbs in infinitive, gerund and past participle form;
16. **Non-inflected verbs**: Number of verbs that have not been inflected, that is, which are in infinite form: infinitive, gerund and past participle;
17. **Words before main verb**: Number of words before the main verbal phrase.

4.3 Evaluation

The 10-fold cross validation accuracy results are displayed in Table 9. As baselines for our tests, we chose four unique features and evaluated them individually on SVM^{Rank}: a) Words before main verb, b) Clauses per sentence, c) Syllables per content word and d) Tokens per sentence. The last line shows the results of our method with 17 features, detailed in Section 4.2.

Features	PorSimplesSent1	PorSimplesSent2	PorSimplesSent3
Words before main verb	45.13%	36.29%	23.06%
Clauses per sentence	59.02%	41.28%	11.32%
Syllables per content word	54.80%	50.90%	46.33%
Tokens per sentence	80.74%	69.35%	40.76%
All 17 features	**83.39%**	**74.20%**	**53.67%**

Table 9: Baselines and first experiment results (accuracy), using SVMRank.

In **PorSimplesSent1**, as expected, using just the number of tokens per sentence it is possible to achieve more than 80% of accuracy. This is because this dataset includes all sentences that are result of split operations, so the majority of simplified sentences are small parts from the original ones. The **PorSimplesSent3**, which has only full sentences, disregarding those that suffered split, is the most difficult to rank. Besides having the smallest number of pairs, PorSimplesSent3 has some simplified sentences that are bigger than the original ones. The **PorSimplesSent2**, on its turn, is a middle term between the previous two: it has split sentences, but only the longest sentence derived from the split is paired with the original sentence. Therefore, we have chosen the dataset PorSimplesSent2 to be our gold standard for sentential complexity task in Portuguese.

Our model with 17 features presents improvement over the strongest baseline (Tokens per Sentence): 2.65 in PorSimplesSent1, achieving 83.39% accuracy; 4.85 in PorSimplesSent2, achieving 74.20% accuracy; and 12.91 in PorSimplesSent3, achieving 53.67% accuracy.

4.4 Error Analysis

We performed a manual analysis, trying to understand the errors made by our model, in order to improve it with new features. Building on the syntactic and lexical operations used to annotated the PorSimples corpus, but now with focus on operations at the sentence level, we proposed a set of 14 labels to annotate the errors. Table 10 shows the errors found after this analysis.

Label Description	**Qty**	**%**
1 Replacement by word of the same grammatical class, including multiword discourse markers	169	28.89
2 Replacement by word of different grammatical class, without specifying the classes involved	19	3.25
3 Replacement by paraphrase (one word by several words)	111	18.97
4 Removal of clause	6	1.03
5 Removal of syntactic constituent (subject, adverbial adjunct, etc.)	8	1.37
6 Removal of words	31	5.30
7 Removal of parentheses	10	1.71
8 Insertion of words	33	5.64
9 Change in the order of constituents (such as putting the subject first and the adverb last)	44	7.52
10 Change to active voice	21	3.59
11 Change to synthetic (shortest) passive voice form (by means of passivizating particle “se”)	3	0.51
12 Change from direct to indirect speech	2	0.34
13 Rephrasing	48	8.21
14 ERROR (equal sentences or alignment error, which will be excluded from the corpus)	80	13.68

Table 10: List of Errors used to annotate 418 sentence pairs of PorSimplesSent3.

We annotated 209 of the 418 sentence pairs of PorSimplesSent3 for which our model missed the prediction. The annotation performed by two annotators was double blind and multi-label. A discussion on the pairs presenting annotation disagreement helped to clarify doubts on the annotation process and to assign commonly agreed labels. After that, the remaining sentence pairs were divided into two parts and each part was assigned to only one annotator.

The analysis of these numbers lead us to cogitate which features and metrics might be significant to improve the performance of our ranking model, initially trained with 17 linguistic and psycholinguistic features. Both most frequent labels, 1 and 3, relate to lexical substitution. Example 1 below shows a pair of sentences annotated only with the label 1.

Example 1

- (O): Quem é contra diz que os cães sujam a praia e colocam em risco a saúde dos veranistas. (Those who are against say that the dogs dirty the beach and put at risk the health of the vacationers.)

- (N): Quem é contra diz que os cães sujam a praia e colocam em risco a saúde das pessoas. (Those who are against say that the dogs dirty the beach and put at risk the health of the people.)

The only difference between the two sentences is the pair of words "veranistas" versus "pessoas", in a hyponym relationship. Example 2 brings a pair annotated with label 3. It shows 2 substitutions by paraphases, here understood as a word replaced by several ones, similar in meaning: "possibilitar" by "tornará possível" and "hepática" by "do fígado".

Example 2

- (O): A descoberta possibilitará que pessoas com dano no fígado usem as próprias células-tronco para produzir células hepáticas. (The discovery will enable people with liver damage to use their own stem cells to produce hepatic cells.)
- (N): A descoberta tornará possível que pessoas com dano no fígado usem as próprias células-tronco para produzir células do fígado. (The discovery will make it possible for people with liver damage to use their own stem cells to produce liver cells.)

As many sentence pairs differ by only one word, readability measures to compare words are essential to decide which is the easiest sentence. Word frequency and psycholinguistic properties of words (as age of acquisition, familiarity, concreteness, imageability) may be useful for this purpose. Additionally, there are several resources that may be used to design new metrics to deal with similar words and paraphases. For Portuguese, there are different similar projects of wordnets, among which stand out the OpenWordNet-PT (de Paiva et al., 2012), as the most complete with manual revision, and the CONTO.PT (Gonçalo Oliveira, 2016), built semi-automatically in order to comprise a greater number of words, and which describes itself as a diffuse wordnet. There is also the PPDB (Paraphrase Database), a resource that contains paraphrases in several languages, including Portuguese, automatically extracted from bilingual corpora (Ganitkevitch and Callison-Burch, 2014). Paraphrase in the context of PPDB refers to expressions or equivalent words. As it was generated automatically, the PPDB also contains some false positives. The resource is available in six different sizes: the difference is that larger sets extracted paraphrase rules with less confidence.

For features other than the lexical ones, a very promising research avenue is to test simplified sentences with human readers to confirm whether they are simpler than their original counterparts or not (using eye-trackers). This is relevant because many simplification operations we use are inspired in the literature regarding English language simplification and we need more evidence related to Portuguese language. The error analysis, therefore, provided important insights for future work aiming to increase the accuracy of our model in the dataset made available with this paper. Besides that, 80 pairs were dropped from our dataset because they contain nearly identical sentences or completely different sentences (improperly paired due to alignment error). Therefore, all the three totals in Table 6 were reduced by 80, resulting in 10,536, 4,888, and 2,520 sentences, respectively.

5 Conclusions

In this paper, we presented a new resource to evaluate the task of sentence readability for Portuguese language - the corpus PorSimplesSent. This dataset is larger, in terms of sentence pairs, than a similar corpus for the English language (cg. (Vajjala and Meurers, 2016)), and it is the first resource of this kind for Portuguese language, therefore we believe we can have a blossom of future research for this task. Moreover, we made available four baselines for the corpus and an approach based on pairwise ranking to compare two versions of a sentence. Our model uses 17 lexical, syntactic and psycholinguistic features and identifies the readability level of sentence pairs with an accuracy of 74.2%; an improvement of 2.65 on the strongest baseline. We believe there is plenty of room for improvement of our model and we hope this task receive a lot of attention from researchers devoted to Portuguese language NLP as well. The corpus is made publicly available at `http://www.nilc.icmc.usp.br/nilc/index.php/tools-and-resources`. As for future work, we will enlarge the number of features to build an improved model to evaluate the task and organize a shared task using it in an NLP conference.

References

Sandra M. Aluísio, Andre Cunha, and Carolina Scarton. 2016. Evaluating progression of alzheimer's disease by regression and classification methods in a narrative language test in portuguese. In *12th International Conference on Computational Processing of the Portuguese Language (PROPOR 2016)*, volume 9727 of *Lecture Notes in Computer Science*, pages 109–114. Springer Cham.

Sandra M. Aluísio and Caroline Gasperin. 2010. Fostering digital inclusion and accessibility: the Porsimples project for simplication of portuguese texts. In *Proceedings of the NAACL HLT 2010 Young Investigators Workshop on Computational Approaches to Languages of the Americas - Association for Computational Linguistics*, pages 46–53, Stroudsburg, PA.

Bharat Ram Ambati, Siva Reddy, and Mark Steedman. 2016. Assessing relative sentence complexity using an incremental CCG parser. In *HLT-NAACL*, pages 1051–1057, Stroudsburg, PA. The Association for Computational Linguistics.

Eckhard Bick. 2000. The parsing system Palavras: Automatic grammatical analysis of Portuguese in a Constraint Grammar Framework. *Aarhus University Press.*

Helena M. Caseli, Tiago F. Pereira, Lúcia Specia, Thiago A. S. Pardo, Caroline Gasperin, and Sandra M. Aluísio. 2009. Building a Brazilian Portuguese parallel corpus of original and simplified texts. In *Advances in Computational Linguistics, Research in Computer Science (CICLing-2009)*, volume 41, pages 59–70.

Valeria de Paiva, Alexandre Rademaker, and Gerard de Melo. 2012. OpenWordNet-PT: An open Brazilian Wordnet for reasoning. In *Proceedings of COLING 2012: Demonstration Papers*, pages 353–360, Mumbai, India, December. The COLING 2012 Organizing Committee. Published also as Techreport http://hdl.handle.net/10438/10274.

Felice Dell'Orletta, Simonetta Montemagni, and Giulia Venturi. 2011. Read-it: Assessing readability of Italian texts with a view to text simplification. In *SLPAT '11 Proceedings of the Second Workshop on Speech and Language Processing for Assistive Technologies*, pages 73–83, Stroudsburg, PA. Association for Computational Linguistics.

Felice Del'Orletta, Simonetta Montemagni, and Giulia Venturi. 2014. Assessing document and sentence readability in less resourced languages and across textual genres. *International Journal of Applied Linguistics (ITL). Special Issue on Readability and Text Simplification.*

Leandro Borges dos Santos, Magali Sanches Duran, Nathan Siegle Hartmann, Gustavo Henrique Paetzold Arnaldo Candido, and Sandra Maria Aluisio. 2017. A lightweight regression method to infer psycholinguistic properties for Brazilian Portuguese. In *International Conference on Text, Speech, and Dialogue (TSD 2017)*, volume 10415 of *Lecture Notes in Computer Science*, pages 281–289. Springer, Cham.

William H. Dubay. 2007. *Smart Language: Readers, Readability, and the Grading of Text.* Impact Information, Costa Mesa, CA.

Lyn Frazier. 1985. Syntactic complexity. *D.R. Dowty, L. Karttunen and A.M. Zwicky (eds.), Natural Language Parsing, Cambridge University Press.*

Juri Ganitkevitch and Chris Callison-Burch. 2014. The multilingual paraphrase database. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may. European Language Resources Association (ELRA).

Ana Valeria Gonzalez-Garduño and Anders Søgaard. 2017. Using gaze to predict text readability. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 438–443, Stroudsburg, PA. The Association for Computational Linguistics.

Hugo Gonçalo Oliveira. 2016. Conto.pt: Groundwork for the automatic creation of a fuzzy portuguese wordnet. In *12th International Conference on Computational Processing of the Portuguese Language (PROPOR 2016)*, volume 9727 of *Lecture Notes in Computer Science*, pages 283–295. Springer Cham.

Arthur C. Graesser, Danielle S. McNamara, Max M. Louwerse, and Zhiqiang Cai. 2004. Coh-metrix: Analysis of text on cohesion and language. *Behavior Research Methods, Instruments, & Computers*, 36:193–202.

David M. Howcroft and Vera Demberg. 2017. Psycholinguistic models of sentence processing improve sentence readability ranking. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 958–968, Stroudsburg, PA. The Association for Computational Linguistics.

William Hwang, Hannaneh Hajishirzi, Mari Ostendorf, and Wei Wu. 2015. Aligning sentences from Standard Wikipedia to Simple Wikipedia. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 211–217, Stroudsburg, PA. The Association for Computational Linguistics.

IPM. 2016. Inaf brasil 2015: Indicador de alfabetismo funcional - alfabetismo no mundo do trabalho. *Instituto Paulo Montenegro*.

Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, volume 3, pages 217–226. ACM Press.

Carolina Scarton, Caroline Gasperin, and Sandra Aluísio. 2010. Revisiting the readability assessment of texts in Portuguese. In Simari G.R. Kuri-Morales A., editor, *12th Ibero-American Conference on AI, Advances in Artificial Intelligence – IBERAMIA 2010*, volume 6433 of *Lecture Notes in Computer Science*, pages 306–315, Berlin, Heidelberg. Springer.

Carolina Scarton, Gustavo Paetzold, and Lucia Specia. 2018. Simpa: A sentence-level simplification corpus for the public administration domain. In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France, may. European Language Resources Association (ELRA).

Abhinav Deep Singh, Poojan Mehta, Samar Husain, and Rajakrishnan Rajkumar. 2016. Quantifying sentence complexity based on eye-tracking measures. In *Proceedings of the Workshop on Computational Linguistics for Linguistic Complexity*, pages 202–212, Osaka, Japan. The COLING 2016 Organizing Committee.

Johan Sjöholm. 2012. *Probability as readability: A new machine learning approach to readability assessment for written Swedish*. LiU Electronic Press.

Lucia Specia, Sandra M. Aluísio, and Thiago A. S. Pardo. 2008. Manual de simplificação sintática para o português. NILC Technical Report 08-06, ICMC-USP, jun. Série de Relatórios do Núcleo Interinstitucional de Lingüística Computacional (NILC-TR-08-06), 27 p.

Calvin Thomas, Vlado Keselj, Nick Cercone, Kenneth Rockwood, and Elissa Asp. 2005. Automatic detection and rating of dementia of alzheimer type through lexical analysis of spontaneous speech. In *Proceedings of IEEE ICMA 2005*, volume 3, pages 1569–1574. IEEE.

Sowmya Vajjala and Detmar Meurers. 2014. Assessing the relative reading level of sentence pairs for text simplification. *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 288–297.

Sowmya Vajjala and Detmar Meurers. 2016. Readability-based sentence ranking for evaluating text simplification. *CoRR*, abs/1603.06009.

David Wiley, T.J. Bliss, and Mary McEwen. 2014. Open educational resources: A review of the literature. In Spector J., Merrill M., Elen J., and Bishop M., editors, *Handbook of Research on Educational Communications and Technology: Fourth Edition*, pages 781–789, New York, NY. Springer.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Victor H Yngve. 1960. A model and hypothesis for language structure. *Proceedings of the American Philosophical Association*, 104(5):444–466.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of The 23rd International Conference on Computational Linguistics (COLING), August 2010. Beijing, China*, pages 1353–1361. The COLING 2010 Organizing Committee.

Adopting the Word-Pair-Dependency-Triplets with Individual Comparison for Natural Language Inference

Qianlong Du[1,2], Chengqing Zong[1,2,3], Keh-Yih Su[4]
[1]National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Science
[2]University of Chinese Academy of Science
[3]CAS Center for Excellence in Brain Science and Intelligence Technology
[4]Institute of Information Science, Academia Sinica
{qianlong.du, cqzong}@nlpr.ia.ac.cn
kysu@iis.sinica.edu.tw

Abstract

This paper proposes to perform natural language inference with *Word-Pair-Dependency-Triplets*. Most previous DNN-based approaches either ignore syntactic dependency among words, or directly use tree-LSTM to generate sentence representation with irrelevant information. To overcome the problems mentioned above, we adopt *Word-Pair-Dependency-Triplets* to improve alignment and inference judgment. To be specific, instead of comparing each triplet from one passage with the merged information of another passage, we first propose to perform comparison directly between the triplets of the given passage-pair to make the judgment more interpretable. Experimental results show that the performance of our approach is better than most of the approaches that use tree structures, and is comparable to other state-of-the-art approaches.

1 Introduction

Natural language inference (NLI) refers to the following task: given a text passage *P* (*Premise*) (which might have more than one sentence) and a text passage *H* (*Hypothesis*), whether we can infer *H* from *P*, i.e., identifying a specific relationship among *entailment*, *neutral* and *contradiction*. It has many applications such as question answering (Bhaskar et al., 2013; Harabagiu and Hickl, 2006), information extraction (Romano et al., 2006), machine translation (Pado et al., 2009), automatic text summarization (Harabagiu et al., 2007) and so on. Some evaluations about this task have been organized in the past decades, such as the PASCAL Recognizing Textual Entailment (RTE) Challenge (Dagan et al., 2005), SemEval-2014 (Marelli et al., 2014) and RITE (Shima et al., 2011).

Many previous approaches adopt statistical frameworks (Heilman et al., 2010; Kouylekov and Magnini, 2005). However, neural network approaches have emerged after Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015) was released. Most of them adopt an increasingly complicated network structure to represent text passages, and then predict the relationship between them (Bowman et al., 2016; Liu et al., 2016b). However, *P* might include extra words which are not directly related to *H*. Actually, only the words in *P* that are associated with the words in *H* should be paid attention to. Those relevant words should be emphasized more while the irrelevant words should be less weighted during decision making. Therefore, some approaches (Parikh et al., 2016; Chen et al., 2017a) adopt attention mechanism to implicitly align the words between two passages to yield a better performance. This idea is very similar to how human make the entailment judgment, and the result shows that it is very effective for performing natural language inference on SNLI corpus in which most words in *H* can find their corresponding ones in *P*.

Proceedings of the 27th International Conference on Computational Linguistics, pages 414–425
Santa Fe, New Mexico, USA, August 20-26, 2018.

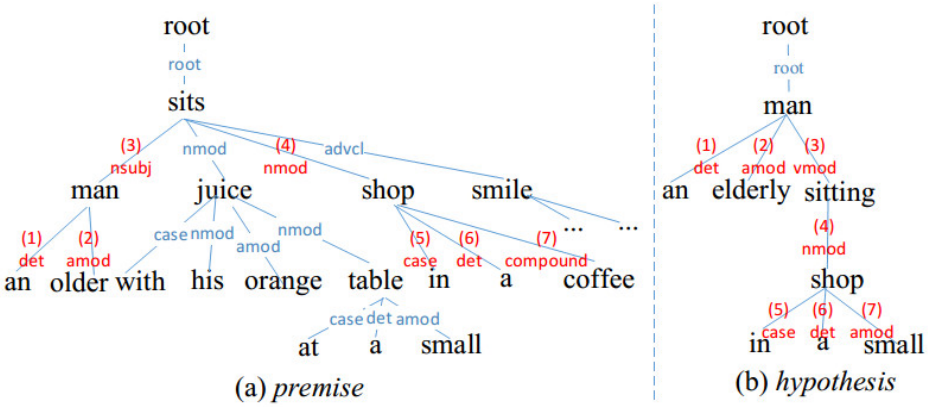


Figure 1. Dependency trees[1] for **Premise** "*An older man sits with his orange juice at a small table in a coffee shop while employees in bright colored shirts smile in the background."* and **Hypothesis** *"An elderly man sitting in a small shop.*" The relationship between them is "*neutral*".

However, after having analyzed some errors generated from an attention-based approach (Parikh et al., 2016), we find that it will introduce mis-alignment and might cause wrong inference. Although context information is used to alleviate this problem, it still cannot handle long distance dependency. Take the following sentence pair as an example (The benchmark is *neutral*):

Premise: *An older man sits with his orange juice at a small table in a coffee shop while employees in bright colored shirts smile in the background.*

Hypothesis: *An elderly man sitting in a small shop.*

The attention-based approach (Parikh et al., 2016) cannot catch the relation between "shop" and "small" in P precisely (which is important for the prediction). In this example, the relationship between *P* and *H* will be predicted as "*entailment*", because all the words in *H* can be found in *P*, although the word "*small*" in these two sentences does not modify the same thing (e.g., "*small*" modifies "*table*" in *premise* while modifies "*shop*" in *hypothesis*).

The above example clearly shows that a sentence is not a set of independent words. It is a sequence of words with syntactic relationship. Based on this observation, we propose to adopt the *Word-Pair Relation-Head-Dependent* (*RHD*) triplet[2] for conducting alignment and comparison. Furthermore, Parikh et al. (2016) and other previous models only compare each word in *H* with the vector merged from all the words in *P* according to their associated alignment scores, and vice versa. However, as shown in Figure 1, human compares *H* and *P* mainly based on *Structure Analogy* (Du et al., 2016; Gentner 1983; Gentner & Markman, 1997) instead of the merged meaning which will not only import irrelevant text but also lose information during merging. Consequently, only words with closely related syntactic/semantic roles (of the aligned predicates) should be compared.

Therefore, we first create two sets of *RHD* from *P* and *H* to denote their corresponding structures, and then perform comparison between triplets in *P* and those in *H*. Accordingly, two *RHD* triplets should be aligned if their relations are related and their head-words are aligned (e.g., the triplets with the same indexes are aligned in Figure 1). Particularly, when two *RHD* triplets are compared, each part of *RHD* triplet (i.e., *Relation, Head*, and *Dependent*) should be compared separately. Besides, as the words of some triplet pairs possess reversed head-dependent relations (e.g., the *Dependent* in "(*nsubj, sits, man*)" is linked to the *Head* in "(*vmod, man, sitting*)", as shown by the triplet pair with index 3 in Figure 1), we introduce *cross-comparison* to compare the *Head* from "(*nsubj, sits, man*)" with the *Dependent* from "(*vmod, man, sitting*)", and vice versa.

[1] The words in black represent the nodes of the dependency tree, and the string on each line represents the dependency relation between two nodes. The red relation denotes that its associated triplet is important in making the judgment. Links with the same indexes indicates that they are aligned and compared when judged the result by human.

[2] Each *RHD* triplet is denoted by "(*rel, head, dep*)", where *head* and *dep* denote the *head-word* and the *dependent-word* of the dependency relation, respectively; and *rel* denotes the dependency relation between *head* and *dependent*.

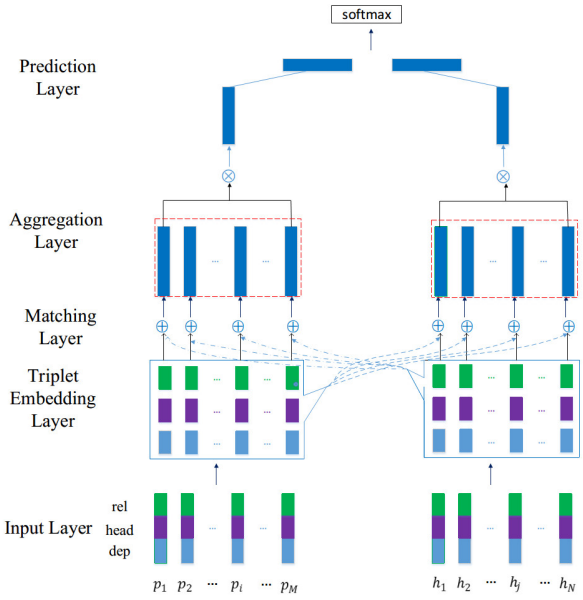


Figure 2: The skeleton of the proposed approach. The *rel*, *head* and *dep* of the triplet are represented with green, purple, and light-blue colors, respectively. Also, $\oplus$ denotes the *Comparison* operation between triplets (see section 2.2.2 for "Matching layer"), while $\otimes$ denotes the *Aggregation* operation (see section 2.2.3 for "Aggregation layer"). Besides, the left part and the right part in *Matching Layer* represent *P-aligned-to-H* and *H-aligned-to-P*, respectively.

Our contributions are summarized as follows: (1) The *RHD* triplet is first proposed to be the alignment and comparison unit in the neural network for NLI. In this way, the corresponding words could be aligned and compared more precisely. (2) Instead of comparing one *RHD* triplet of *H* with the merged meaning of all the *RHD* triplets in *P* (and vice versa), we propose to directly compare each *RHD* triplet of *H* with another *RHD* triplet of *P*; and each part in *RHD* triplet is compared separately. (3) We propose to use *cross-comparison* to compare the related words with different syntactic/semantic roles.

2 Proposed Approach

In our model, we first transform *P* and *H* into two sets of *RHD* triplets. For each *RHD* triplet of *P*, we compare it with another *RHD* triplet of *H* (without merging) to generate an *individual comparison vector* (and vice versa). Afterwards, we use a *self-attention* mechanism to align and sum them to yield the *one-side merged comparison vector* between a triplet and the triplet set of the other side. Last, we aggregate those *one-side merged comparison vectors* to give the overall entailment judgment.

Figure 2 shows the skeleton of the proposed approach. It consists of the following 5 layers: (1) *Input Layer*, which initializes the embedding of the words and relations; (2) *Triplet Embedding Layer*, which is used to adapt the input embedding to yield a better representation for this task; (3) *Matching Layer*, which performs comparison within each *RHD* triplet pair, scores the alignment weights, and sum those *individual comparison vectors* to generate the *one-side merged comparison vector* between each triplet with the triplet set of the other side; (4) *Aggregation Layer*, which aggregates the *one-side merged comparison vectors* to get a *directional comparison vector* for each comparing direction; and (5) *Prediction Layer*, which uses two separated *directional comparison vectors* and a feed-forward neural network classifier to predict the overall judgment.

Our model is symmetric about *P* and *H*. So for simplicity, we only describe the left parts which mainly about comparing each unit of *P* with *H*. Right part is exactly the same except that the roles of *P* and *H* are exchanged.

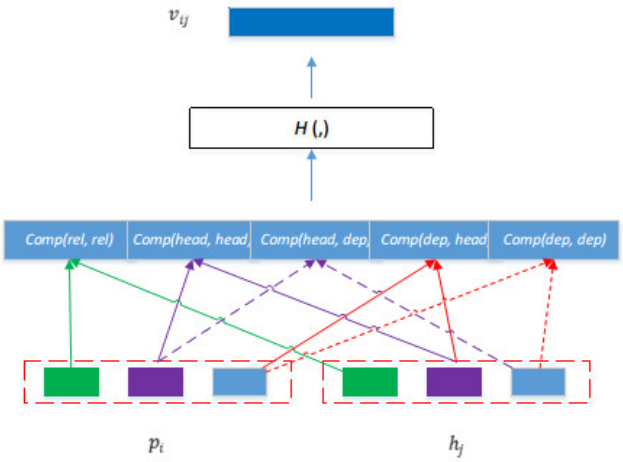


Figure 3: An illustration for the comparison between a triplet p_i in *Premise* and a triplet h_j in *Hypothesis*. The *rel*, *head* and *dep* of the triplet are represented with green, purple, and light-blue colors, respectively. *Comp(,)* indicate the comparison function denoted in equation (2). *H*(,) is a multilayer perception denoted in equation (4). The green solid-line, purple solid-line and red solid-line represent the comparison of pair (*rel, rel*), (*head, head*) and (*dep, dep*), respectively. The purple dot-line represents the cross-comparison of pair (*head, dep*), and red dot-line represents the cross-comparison of pair (*dep, head*).

2.1 Input Layer Generation

We first use a dependency parser[3] to transform *P* and *H* into two sets of *RHD* triplets. We define $\hat{P} := (p_1, p_2, \cdots, p_{\mathrm{m}})$ and $\hat{H} := (h_1, h_2, \cdots, h_{\mathrm{n}})$ be two sets of *RHD* triplets, while p_i and h_j denote the i^{th} *RHD* triplet and the j^{th} *RHD* triplet in *P* and *H*, respectively; also, *m* and *n* indicate the number of associated triplets in *P* and *H*, respectively. We then instantiate the *Input Layer* with the corresponding *word-embeddings* and *rel-embedding* of *RHD* triplets (For conciseness, we will let *rel/head/dep* denote both the original meaning and the corresponding embedding interchangeably from now on). Each $head, dep \in R^{d_w}$ is a word embedding of dimension d_w which is initialized with pre-trained GloVe word embedding (Pennington et al., 2014), while $rel \in R^{d_r}$ is a relation embedding vector of dimension d_r and is initialized randomly with a standard normal distribution (Please note, only rel will be tuned later during training). Each triplet-embedding will be presented as a triplet which contains three embedding corresponding to *rel*, *head* and *dep* respectively.

2.2 Network Architecture

2.2.1 Triplet Embedding Layer

As we fix the value of word embedding during training, in order to obtain better relation/word embedding representations to compare for this task, we use a simple feed-forward structure to adapt the three parts of the triplet to the task. The computations are as follows:

$$\begin{aligned} rel &= W_r * rel_{in} + b_r \\ head &= W_w * head_{in} + b_w \\ dep &= W_w * dep_{in} + b_w \end{aligned} \tag{1}$$

where * is the multiplication of matrices, $rel_{in} \in R^{d_r}$, $head_{in} \in R^{d_w}$ and $dep_{in} \in R^{d_w}$ are the input embedding-vectors from *Input Layer*, $rel \in R^{d_r}$, $head \in R^{d_w}$ and $dep \in R^{d_w}$ are the new representations generated, $W_r \in R^{d_r \times d_r}$, $W_w \in R^{d_w \times d_w}$, $b_r \in R^{d_r}$, $b_w \in R^{d_w}$ are the weight matrices to be learned. Here, all the *rel* share the same weight matrices, while all the words share the same weight matrices.

[3] http://nlp.stanford.edu/software/lex-parser.shtml#Download

2.2.2 Matching Layer

This layer is the core of our model. It is mainly used to perform the individual comparison between two triplets and use associated alignment weights to focus on the individual-comparisons of the preferred alignments. In this step, we will use a *one-side merged comparison vector* to represent each comparison result of one triplet with the triplet-set of another side. For a triplet p_i from *Triplet Embedding Layer*, it has three parts: $rel_{p_i} \in R^{d_r}$, $head_{p_i} \in R^{d_w}$ and $dep_{p_i} \in R^{d_w}$; and for h_j, it has $rel_{h_j} \in R^{d_r}$, $head_{h_j} \in R^{d_w}$ and $dep_{h_j} \in R^{d_w}$.

Figure 3 shows how the *individual comparison-vector* of one triplet-pair is generated. The vector $v_{ij} \in R^d$ denotes the comparison result between triplet p_i and a triplet h_j from $\hat{H}$, while *d* is the dimension of hidden layer. During comparison, each component of the triplet (i.e., *rel*, *head* and *dep*) is compared independently, as shown in the figure. Here, the comparing function is denoted as *comp(,)* in equation (2). *G* is a multi-layer perceptron with one hidden layer and a *Relu* activation.

$$comp(v_1, v_2) = G([v_1; v_2; v_1 - v_2; v_1 \odot v_2]) \tag{2}$$

Where v_1 and v_2 are any two embedding vectors. The notation ";" (within the bracket-pair in the above equation) denotes *Concatenation*; also, '$-$' and '$\odot$' are the *difference* and *element-wise product* of vectors respectively. Then we can get the comparison results in Figure 3 as follows:

$$\begin{aligned}
\overline{rel_{ij}} &:= comp\left(rel_{p_i}, rel_{h_j}\right) \\
\overline{head_{ij}} &:= comp\left(head_{p_i}, head_{h_j}\right) \\
\overline{head_x_{ij}} &:= comp\left(head_{p_i}, dep_{h_j}\right) \\
\overline{dep_{ij}} &:= comp\left(dep_{p_i}, dep_{h_j}\right) \\
\overline{dep_x_{ij}} &:= comp\left(dep_{p_i}, head_{h_j}\right)
\end{aligned} \tag{3}$$

Where $\overline{head\_x_{ij}}$ and $\overline{dep\_x_{ij}}$ are the results of *cross-comparison*. Please note that the comparison functions with the same input arguments share the same set of parameters in *G*. For example, all the functions *comp*(*head*, *head*) share a set of parameters while all *comp*(*head*, *dep*) share another set of parameters. After we obtain the comparison results of these components, we can incorporate them to yield the triplet *individual comparison vector* v_{ij} between p_i and h_j as follow.

$$v_{ij} = H([\overline{rel_{ij}}; \overline{head_{ij}}; \overline{head\_x_{ij}}; \overline{dep_{ij}}; \overline{dep\_x_{ij}}]) \tag{4}$$

Here *H* is a multi-layer perceptron with two hidden layers and a *Relu* activation. Afterwards, we need to generate the *alignment weight* e_{ij} between triplet p_i and triplet h_j to extract the key information for judgment. Most of the previous models (Chen et al., 2017a; Parikh et al., 2016) use the multiplication of two semantic unit vectors as their alignment weights. However, we find the individual comparison here can describe the relatedness of those triplet-pairs better. Consequently, we generate the *alignment weights* using these *individual comparison vectors* with a self-attention mechanism as follows.

$$e_{ij} = W_{s2}\tanh(W_{s1}v_{ij}) \tag{5}$$

Where $W_{s1} \in R^{d \times d}$ and $W_{s2} \in R^{d \times 1}$ are weight matrices to be learned. Then we use e_{ij} to obtain the *one-side merged comparison vectors* $O_{i,H}$ from various v_{ij} as follow.

$$O_{i,H} = \sum_{j=1}^{l_h} \frac{exp(e_{ij})}{\sum_{k=1}^{l_h} exp(e_{ik}))} v_{ij} \tag{6}$$

Where $O_{i,H}$ is the *one-side merged comparison vector* between p_i and the whole set H, l_h is the number of *RHD* triplets in H; $O_{P,j}$ (the right part in Figure 2) is defined similarly between h_j and the whole P.

2.2.3 Aggregation Layer

In this layer, we aggregate all the *one-side merged comparison vectors* $O_{i,H}$ and $O_{P,j}$ (obtained above) to generate the final comparison vector for these two different directions between P and H. Like the previous approaches (Chen et al., 2017a; Parikh et al., 2016), we aggregate the information by summation and max pooling:

$$\begin{aligned} O_{P,sum} &= \sum_{i=1}^{l_p} O_{i,H}, \qquad O_{P,max} = \max_{i=1}^{l_p} O_{i,H}, \qquad O_P = [O_{P,sum}; O_{P,max}] \\ O_{H,sum} &= \sum_{j=1}^{l_h} O_{P,j}, \qquad O_{H,max} = \max_{j=1}^{l_h} O_{P,j}, \qquad O_H = [O_{H,sum}; O_{H,max}] \end{aligned} \tag{7}$$

Where l_p and l_h are the numbers of triplets in P and H, respectively. We get the *overall comparison vector* O_P by concatenating the summation $O_{P,sum}$ and the max pooling $O_{P,max}$ using the *one-side merged comparison vectors of P* (and vice versa). Afterwards, we use these *overall comparison vectors* to predict the relationship in the next section.

2.2.4 Prediction Layer

From the above section, we have obtained O_P and O_H, which are the *overall comparison vectors* from $\hat{P}$ to $\hat{H}$ and from $\hat{H}$ to $\hat{P}$, respectively (i.e., the *overall comparisons* in two different directions). We then concatenate them and use a multi-layer perceptron classifier Q (it has two hidden layers with *Relu* activation and a *softmax* output layer) to generate the final overall judgment vector $\hat{y}$ (as shown in Eq. (8)), where $\hat{y} \in R^C$ (C equals the number of classes) are the scores for each class. The predicted class can be got by setting $y = \text{argmax}_i \hat{y}_i$. When we train the model, we use multi-class cross-entropy loss with dropout regularization (Srivastava et al., 2014).

$$\hat{y} = Q([o_P; o_H]) \tag{8}$$

3 Experiments

3.1 Dataset

We adopt both *SNLI* (Bowman et al., 2015) corpus[4] and *MultiNLI* (Williams et al., 2018) corpus[5] to test the performance. They are briefly introduced as follows.

SNLI - It contains 570k sentence pairs. The sentence pairs in this corpus are labelled with one of the following relationships: *entailment*, *contradiction*, *neutral* and "-", where "-" means that it lacks of consensus from human annotators. In our experiments, we follow Bowman et al. (2015) to delete those sentence pairs labelled with "-". Consequently, we end up with 549,367 pairs for training, 9,842 pairs for development and 9,824 pairs for testing.

MultiNLI - This corpus has 433k sentence pairs, which are collected from broad range of genre of American English such as written non-fiction genres (e.g. SLATE, OUP), spoken genres (TELEPHONE, FACE-TO-FACE), less formal written genres (FICTION, LETTERS), and that specialized on for 9/11. For the training set of this corpus, it selects half of the genres to create in-domain (matched) and out-domain (mismatched) development/test sets. Since the test set labels of this corpus are not released, the test performance is obtained through submission to Kaggle.com[6].

[4] https://nlp.stanford.edu/projects/snli/
[5] http://www.nyu.edu/projects/bowman/multinli/
[6] Matched : https://www.kaggle.com/c/multinli-matched-open-evaluation ; Mismatched: https://www.kaggle.com/c/multinli-mismatched-open-evaluation

Model	Training Acc.	Test Acc.
(1) LSTM (Bowman et al., 2015)	84.4	77.6
(2) Classifier (Bowman et al., 2015)	99.7	78.2
(3) 300D tree-based CNN encoders (Mou et al., 2016)	83.3	82.1
(4) 300D SPINN-PI encoders (Bowman et al. 2016)	89.2	83.2
(5) 100D LSTMs w/ word-by-word attention (Rocktaschel et al., 2015)	85.3	83.5
(6) 300D mLSTM word-by-word attention model (Wang & Jiang, 2016)	92.0	86.1
(7) 200D decomposable attention model (Parikh et al., 2015)	89.5	86.3
(8) 200D decomposable attention model with intra-sentence attention (Parikh et al., 2015)	90.5	86.8
(9) Binary Tree-LSTM + Structured Attention & Composition + dual-attention (Zhao et al., 2016)	87.7	87.2
(10) 300D Full tree matching NTI-SLSTM-LSTM w/ global attention (Munkhdalai and Yu, 2016)	88.5	87.3
(11) 300D Syntactic Tree-LSTM (Chen et al., 2017a)	92.9	87.8
Human Performance (Gong et al., 2017)	97.2	87.7
Our model	90.3	87.4

Table 1. Performance on SNLI

Model	Test Acc.	
	Matched	Mismatched
(1)BiLSTM (Williams et al., 2018)	67.0	67.6
(2) Inner Att (Balazs et al., 2017)	72.1	72.1
(3) ESIM (Williams et al., 2018)	72.3	72.1
(4) Gated-Att BiLSTM (Chen et al., 2017b)	73.2	73.6
(5) Shortcut-Stacked encoder (Nie & Bansal, 2017)	74.6	73.6
(6) DIIN (Gong et al., 2017)	78.8	77.8
(7) Inner Att (ensemble) (Balazs et al., 2017)	72.2	72.8
(8) Gated-Att BiLSTM (ensemble) (Chen et al., 2017b)	74.9	74.9
(9) DIIN (ensemble) (Gong et al., 2017)	80.0	78.7
Human Performance (Gong et al., 2017)	88.5	89.2
Our model	75.1	74.7

Table 2. Performance on MultiNLI

3.2 Details of training

In order to initialize the words in the triplets, we used 300 dimensional Glove embedding (Pennington et al., 2014). For the relation vectors (the dimension is set to 20), we use a standard normal distribution to randomly initialize the values and then normalize each vector. Besides, for OOV words, we follow (Parikh et al., 2016) to initialize them by randomly selecting one from 100 random vectors. During the training, the word embedding including the 100 random vectors for OOV words are fixed while the embedding of the relations will be updated. The dimensions of the Triplet embedding Layer for relation and head/dependent-words are set to 20 and 300, respectively. And the dimensions of other hidden layers are set to 1,024. Besides, *Adadelta* method (Zeiler, 2012) is adopted for optimization, and the batch size is 32, the dropout ratio is 0.2, while the learning rate is 0.1.

3.3 Results and Analysis

Table 1 shows the results of different models on SNLI. The first group includes two baseline classifiers presented by Bowman et al. (2015). In model (1), they use LSTM to learn a representation for the passage, and then use the representation of the passage-pair to predict the judgment label. Model (2) uses a

traditional statistical classifier to predict the label with some handcrafted features such as the overlapped words, negation detection, etc.

In the second group, Model (3) and model (4) are two models based on passage encoding with tree structure. In model (3), Mou et al. (2016) considers tree-based CNN to capture passage-level semantics, while Bowman et al. (2016) use parsing and interpretation within a single tree-sequence hybrid model in model (4).

In the next group, models (5)-(11) are inter-passage attention-based models which are similar to ours. Model (5) and model (6) are LSTMs with word-by-word attention. In model (7), Parikh et al. (2016) decompose each passage into a set of words and then compare two word-sets of the passage-pair. They further improve the performance by adding intra-passage attentions in model (8). Our model is inspired by their approach. Specifically, in models (9)-(11): Zhao et al. (2016) adopt tree-LSTM and attention mechanism based on binary tree to generate semantic units and compare; Munkhdalai and Yu (2016) constructs a full n-ary tree to improve the performance, while Chen et al. (2017a) use a syntactic tree-LSTM to extract the inner dependency relations in the passage and compare between the passage-pair.

Table 1 shows that our model achieves an accuracy of 87.4%, which outperforms most of those models with tree structure (even those with complicated network architectures (Munkhdalai and Yu, 2016; Zhao et al., 2016)), and our model is more interpretable than the other models which are based tree structure. Specifically, our performance is better than Parikh et al. (2016) significantly though our model is inspired by theirs.

In the first group of Table 2, models (1)-(6) show some published best performances on MultiNLI. And models (7)-(9) are some ensemble models which have a very complicated network architecture. From this table, we can see that our performance is better than models (1)-(5) on matched set (even outperforms the ensemble models (7) and (8)), and better than models (1)-(5) on mismatched set. Please note that the structure of our model is more interpretable than that of others.

	Train Acc.	Test Acc.
Original model	90.3	87.4
(1) merged comparison	89.3	84.5
(2) replace self-attention with inter-attention	92.1	86.6
(3) remove cross-comparison	88.3	86.5

Table 3. Ablation study on SNLI set

3.4 Ablation analysis

Table 3 examines the effect of each major component. In order to check whether the strategy of performing individual comparison between *RHD* triplets works well in this task, model (1) directly compares one triplet with the merged representation from the triplet-set of the other side (in the Matching Layer). In this model, we compute one representation for each triplet by concatenating its three parts, align between these triplet representations of the sentence pair, and then compare each triplet with the merged representation from the triplet-set of the other sentence as in (Parikh et al., 2016). It shows that the proposed individual comparison strategy gives a better result.

In model (2), instead of using self-attention to generate the alignment weight between one triplet pair, we first yield a semantic vector representation for each triplet by concatenating their three parts and processing it with a linear transform. And then we use the multiplication of the vector representations of each triplet-pair as their alignment weight (as adopted in (Parikh et al., 2016)). When we do this alteration, the accuracy drops to 86.6% on the test set. This again proves that adopting individual comparison for each triplet-pair can describe their relatedness more accurately.

Last, we remove *cross-comparison* from our approach and list the result in model (3) to check the effect of this component. This component is mainly used to compare the words which are similar in semantics but play different syntactic/semantic roles. It shows that when we remove this component, the accuracy drops to 86.5%.

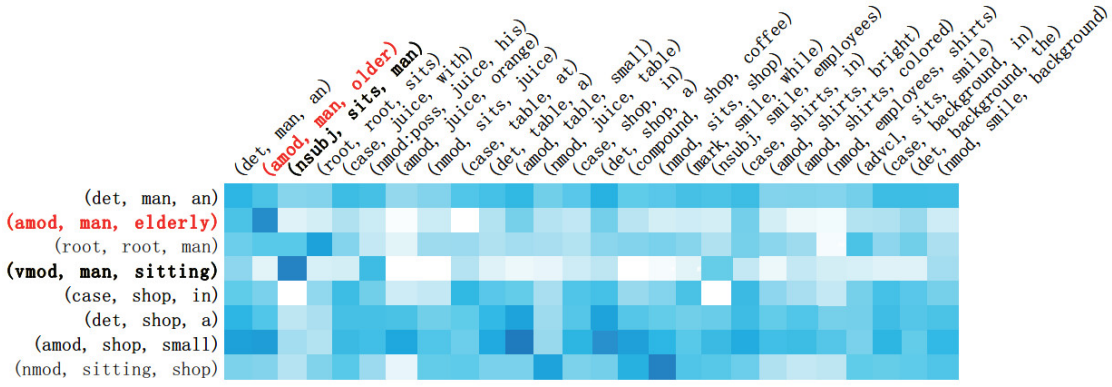


Figure 4. Triplet alignment weights for the triplet-pair in Figure 1. The darker color represents greater value. The triplets for *P* are on the top, and the triplets for *H* are on the left.

3.5 Visualization of Triplet Alignment

Figure 4 shows the alignment weights e_{ij} (i.e., the degree of relatedness between the triplets within the same sentence pair) of the example in Figure 1. In this figure, a darker color corresponds to a larger value of e_{ij}. From this figure, we can see that the related triplets do have larger alignment weights between them. For example, compared with other triplets of *H*, triplet (*amod, man, elderly*) in *H* (the red entry at the left of Figure 4) is more related to (*amod, man, older*) of *P* (the red entry at the top), which is echoed with a darker corresponding cell in Figure 4. Similarly, the corresponding cell for (*nsubj, sits, man*) of *P* and (*vmod, man, sitting*) of *H* shows a darker color, which also meets human judgment. This clearly shows that the alignment weights between these two triplet sets reflect the human interpretation closely.

4 Related Work

Early approaches for natural language inference usually adopted statistical models such as SVM (Joachims, 1998), CRF (Hatoriet et al., 2009) and so on, which employed hand-crafted features, and utilized various external resources and specialized sub-components such as *negation detection* (Lai and Hockenmaier, 2014; Levy et al., 2013). Besides, all the adopted datasets are very small.

After the SNLI corpus (Bowman et al., 2015) was released, a lot of work about natural language inference based on neural networks have been published in recent years (Bowman et al., 2016; Liu et al., 2016; Liu et al., 2016b; Munkhdalai and Yu, 2016; Mou et al., 2015; Sha et al., 2016). Basically, those neural network based approaches could be classified into 2 categories: (1) Merely computing the passage-embedding without introducing alignment (between the words in the sentences), and then comparing these passage-embedding to get the prediction (Bowman et al., 2016, Mou et al., 2016). (2) Decomposing the passage into some semantic units, comparing each semantic unit with the passage of the other side and then aggregating these comparisons (Parikh et al., 2016; Wang et al., 2017).

Within the first category, Bowman et al. (2015) used two LSTMs to get the representations of *P* and *H,* respectively. They then compare the vector representation of these two passages to predict the relationship between *P* and *H*. Besides, Bowman et al. (2016) used tree-LSTM to encode the representation of the passage. Although it uses the dependency relation in the passages to generate the representation, it cannot point out the difference among different words and is unable to catch the key information in the prediction step.

For the second category, Zhao et al. (2017) used the intermediate representations from tree-LSTM to compare *P* and *H*. It compares each node representation with the merged representation of the other passage, and generates the result bottom-up. In this way, it can extract the dependency relations in the passages and compare semantics in different granularities. However, they adopt the tree-LSTM to bottom-up generate the intermediate representations, and then directly compare each unit with the merged representation of the other passage. As the result, they cannot catch the key information and filter out irrelevant text.

Different from them, we transform *P* and *H* into two sets of *RHD* triplets, instead of comparing one *RHD* triplet of *P* with the merged information of the whole *H* (and vice versa), we directly compare a

RHD triplet of *P* with another *RHD* triplet of *H* to obtain the *individual comparison vector* (without merging). Specifically, when comparing two *RHD* triplets, each part in *RHD* triplet (i.e., *rel, head*, and *dep*) is compared separately and *cross-comparison* is adopted between nodes. Furthermore, we generate more precise alignment weights with these *individual comparison vectors* and self-attention mechanism. Consequently, our model is more interpretable than the previous models.

5 Conclusion

Inspired by how human judge the entailment relationship between the given *Premise (P)* and *Hypothesis (H)* text passages, we propose to transform the passages into two sets of word-pair dependency triplets, and then directly compare each triplet with every triplet from the other side to get the *individual comparison vector*. In this way, we can filter out the irrelevant information and judge the relationship between two passages more precisely. In order to further improve the comparison precision, we compare each part (i.e., "*rel*, *head*, and *dep*") of triplet separately and adopt *cross-comparison* to compare the related words which are with different syntactic/semantic roles. As these *individual comparison vectors* can describe the relatedness of the triplet-pair well, we use them and self-attention mechanism to generate the alignment weights between triplets from each passage. Afterwards, we use the alignment weights to incorporate these *individual comparison vectors* to yield the *one-side merged comparison vector* of one *RHD* triplet with the *RHD* triplet set of the whole other side. Finally, we aggregate those *one-side merged comparison vectors* to conduct the final overall entailment decision.

Acknowledgements

The research work descried in this paper has been supported by the National Key Research and Development Program of China under Grant No. 2017YFC0820700 and the Natural Science Foundation of China under Grant No. 61333018.

References

Jorge A Balazs, Edison Marrese-Taylor, Pablo Loyola, and Yutaka Matsuo. 2017. Refining Raw Sentence Representations for Textual Entailment Recognition via Attention. In *Proceedings of the 2nd Workshop on Evaluating Vector-Space Representations for NLP*, pages 51-55, Copenhagen, Denmark, September 7-11, 2017.

Pinaki Bhaskar, Somnath Banerjee, Partha Pakray, Samadrita Banerjee, Sivaji Bandyopadhyay and Alexander Gelbukh. 2013. A hybrid question answering system for Multiple Choice Question (MCQ). In: *Question Answering for Machine Reading Evaluation (QA4MRE) at CLEF 2013 Conference and Labs of the Evaluation Forum*, Valencia, Spain.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 628-635, Vancouver, October 2005.

Samuel R. Bowman, Gabor Angeli, Christopher Potts and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632-642, Lisbon, Portugal, 17-21 September 2015.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1466-1477, Berlin, Germany, August 7-12, 2016.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang and Diana Inkpen. 2017a. Enhanced LSTM for Natural Language Inference. In *Proceedings of the 55th Annual Meeting of the Association for Computional Linguistics*, pages 1657-1668 Vancouver, Canada.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017b. Recurrent Neural Network-Based Sentence Encoder with Gated Attention for Natural Language Inference. In *Proceedings of the 2nd Workshop on Evaluating Vector-Space Representations for NLP*, pages 36-40, Copenhagen, Denmark, September 7-11, 2017.

Peter Clark and Qren Etzioni. 2016. My computer is an honor student-but how intelligent is it? Standard tests as measure of ai. *AI Magazine 37(1):5-12.*

Ido Dagan, Oren Glickman and Bernardo Magnini. 2005. The PASCAL Recognising Textual Entailment Challenge. In *International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment,* pages 177-190.

Qianlong Du, Chengqing Zong and Keh-Yih Su. Intergrating Structural Context and Local Context for Disambiguating Word Senses. The *Fifth Conference On Natural Language Processing and Chinese Computing & The Twenty Fourth International Conference On Computer Processing of Oriental Languages(NLPCC-ICCPOL 2016)*，Kunming, China, December 2-6, 2016, pages. 3-15

Dedre Gentner and Arthur B. Markman. 1997. Structure mapping in analogy and similarity. *American Psychologist, 52(1), 45-56*.

Dedre Gentner. 1983. Structure-mapping: A theoretical framework for analogy. *Congnitive science 7(2):155-170.*

Yichen Gong, Heng Luo and Jian Zhang. 2017. Natural Language Inference Over Interaction Space. CoRR abs/1709.04348. http://arxiv.org/abs/1709.04348.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the 14th International Conference on Artifical Intelligence and Statistics (AISTATS)* 2011, Fort Lauderdale, FL, USA. Volume 15 of JMLR: W&CP 15.

Sanda Harabagiu and Andrew Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Sydney, Australia, 2006:905-912*

Sanda Harabagiu, Andrew Hickl, and Finley Lacatusu. 2007. Satisfying information needs with multi-document summaries. *Information Processing & Management* 43.6 (2007): 1619-1642.

Jun Hatori, Yusuke Miyao, and Jun'ichi Tsujii. On contribution of sense dependencies to word sense disambiguation. 2009. *Information and Media Technologies* 4.4 (2009): 1129-1155.

Michael Heilman, and Noah A. Smith. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. 2010. *Human Language Technologies*: *The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics.*

Thorsten Joachims. Making large-scale SVM learning practical. 1998. No. 1998, 28. *Technical Report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund.*

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

Milen Kouylekov, and Bernardo Magnini. 2005. Recognizing textual entailment with tree edit distance algorithms. In *Proceedings of the First Challenge Workshop Recognising Textual Entailment.*

Alice Lai, and Julia Hockenmaier. 2014. Illinois-LH: A Denotational and Distributional Approach to Semantics. In Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), pages 329-334, Dublin, Ireland, August 23-24, 2014.

Omer Levy, Torsten Zesch, Ido Dagan and Iryna Gurevych. 2013. Recognizing partial textual entailment. In *Proceedings of the 51st Annual Meeting of the Asociation for Computational Linguistics*, pages 451-455.

Pengfei Liu, Xipeng Qiu, Jifan Chen and Xuanjing Huang. 2016a. Deep Fusion LSTMs for Text Semantic Matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1034-1043.

Yang Liu, Chengjie Sun, Lei Lin and Xiaolong Wang. 2016b. Learning Natural Language Inference using Bidirectional LSTM model and Inner-Attention. In *arXiv preprint arXiv*: 1605.09090.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learing Research*, 9(Nov):2579-2605.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini and Roberto Zamparelli. 2014. SemEval-2014 Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014),* pages 1-8, Dublin, Ireland, August 23-24 2014.

.Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan and Zhi Jin. 2016. Natural Language Inference by Tree-Based Convoluation and Heuristic Matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics,* pages 130-136, Berlin, Germany, August 7-12, 2016.

Tsendsuren Munkhdalai and Hong Yu. 2016. Neural tree indexers for text understanding. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers,* pages 11-21, Valencia, Spain, April 3-7, 2017.

Yixin Nie and Mohit Bansal. Shortcut-Stacked Sentence Encoder for Multi-Domain Inference. In *Proceedings of the 2nd Workshop on Evaluating Vector-Space Representation for NLP*, pages 41-45, Copenhagen, Denmark, September 7-11, 2017.

Rodney D. Nielsen, Wayne Ward, and James H. Martin. 2009. Recognizing entailment in intelligent tutoring systems. *Natural Language Engineering 15.4 (2009): 479-501.*

Lorenza Romano, Milen Kouylekov, Idan Szpektor, Ido Dagan and Alberto Lavelli. 2006. Investigating a generic paraphrase-based approach for relation extraction. In *Proceedings of EACL*, pages 409-416, Trento.

Sebastian Padó, et al. 2009. Measuring machine translation quality as semantic equivalence: A metric based on entailment features. *Machine Translation* 23.2-3 (2009): 181-193.

Ankur P. Parikh, Oscar Tackstrom, Dipanjan Das and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249-2255.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP),* pages 1532-1543, October 25-29, 2014, Doha, Qatar.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský and Phil Blunsom. 2015. Reasoning about entailment with neural attention. In *arXiv-2015*. https://arxiv.org/abs/1509.06664

Lei Sha, Baobao Chang, Zhifang Sui and Sujian Li. 2016. Reading and Thinking: Reread LSTM Unit for Textual Entailment Recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2870-2879.

Nidhi Sharma, Richa Sharma and Kanad K. Biswas. 2015. Recognizing Textual Entailment using Dependency Analysis and Machine Learning. In *Proceedings of NAACL-HLT 2015 Student Research Workshop (SRW),* pages 147-153.

Hideki Shima, Hiroshi Kanayama, Cheng-Wei Lee, Chuan-Jie Lin, Teruko Mitamura, Yusuke Miyao, Shuming Shi and Koichi Takeda. 2011. Overview of NTCIR-9 RITE: Recognizing Inference in Text. In *Proceedings of NTCIR-9 Workshop Meeting*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutshever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural network from overfintting. The *Journal of Machine Learning Research*, 15(1):1929-1958.

Shuohang Wang and Jing Jiang. 2016. Learning Natural Language Inference with LSTM. In *Proceedings of NAACL-HLT 2016,* pages 1442-1451, San Diego, California, June 12-17, 2016.

Zhiguo Wang, Wael Hamza and Radu Florian. Bilateral Multi-Perspective Matching for Natural Language Sentences. *IJCAI(2017)*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of NAACL-HLT 2018*, pages 1112-1122, New Orleans, Louisiana, June 1-6, 2018.

Kai Zhao, Liang Huang and Mingbo Ma. Textual Entailment with Structured Attentions and Composition. In *Proceedings of Coling 2016, the 26th International Computational Linguistics: Technical Papers*, pages 2248-2258, Osaka, Japan, December 11-17 2016.

Jiang Zhao, Tian Tian Zhu and Man Lan. 2014. Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 271-277, Dublin, Ireland, August 23-24, 2014.

Cooperative Denoising for Distantly Supervised Relation Extraction

Kai Lei[1,*], **Daoyuan Chen**[1,*], **Yaliang Li**[2], **Nan Du**[2], **Min Yang**[3], **Wei Fan**[2], **Ying Shen**[1,†]
[1]School of Electronics and Computer Engineering, Peking University Shenzhen Graduate School
[2]Tencent Medical AI Lab
[3]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[1]{leik, shenying}@pkusz.edu.cn, [1]chendaoyuan@pku.edu.cn
[2]{yaliangli,ndu,davidwfan}@tencent.com, [3]min.yang@siat.ac.cn

Abstract

Distantly supervised relation extraction greatly reduces human efforts in extracting relational facts from unstructured texts. However, it suffers from noisy labeling problem, which can degrade its performance. Meanwhile, the useful information expressed in knowledge graph is still underutilized in the state-of-the-art methods for distantly supervised relation extraction. In the light of these challenges, we propose CORD, a novel **CO**ope**R**ative **D**enoising framework, which consists two base networks leveraging text corpus and knowledge graph respectively, and a cooperative module involving their mutual learning by the adaptive bi-directional knowledge distillation and dynamic ensemble with noisy-varying instances. Experimental results on a real-world dataset demonstrate that the proposed method reduces the noisy labels and achieves substantial improvement over the state-of-the-art methods.

1 Introduction

Relation extraction aims to discover the semantic relationships between entities. Recently, it has attracted increasing attention due to its broad applications in many machine learning and natural language processing tasks such as Knowledge Graph (KG) construction (Shin et al., 2015), information retrieval (Kadry and Dietz, 2017), and question answering (Abujabal et al., 2017).

Distant supervision is one of the most important techniques in practice for relation extraction due to its ability to generate large-scale labeled training data automatically by aligning KGs to text corpus. Despite its effectiveness, it suffers from noisy labeling problem such as false negative examples, which can severely degrade its performance. To alleviate this limitation, multi-instance learning and probabilistic graphical models have been widely explored by existing work (Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012). Due to the success of deep learning, there has been increasing interest in applying deep neural networks to solve this problem. Zeng et al. (2015) proposed a piecewise CNN model combining multi-instance paradigm, Lin et al. (2016) and Lin et al. (2017) introduced sentence-level and multi-lingual attention to alleviate the side-effect caused by noisy instances, and Luo et al. (2017) further modeled noises explicitly.

However, most existing studies reduce noises by leveraging information within text corpus, ignoring the relational facts expressed in other information sources, such as KG triples or semi-structured tables. Leveraging various information sources simultaneously, which takes full advantage of diverse and supplementary information in different sources, is beneficial to reduce noisy labels for distant supervision. To be more specific, we transform each sentence into entity sequence based on KG information, which is helpful to locate critical entities and adjust word-based network when their predictions are not consistent.

Moreover, by incorporating information from other sources, distantly supervised relation extraction methods can better handle "Not A relation" (NA) class, which is the main reason for the noisy label problem. The large proportion of NA instances is typical in distantly supervised relation extraction task, and it's non-trivial to characterize the NA patterns only based on text-corpus information. By considering

* Equal contribution.
† Corresponding author.

Proceedings of the 27th International Conference on Computational Linguistics, pages 426–436
Santa Fe, New Mexico, USA, August 20-26, 2018.

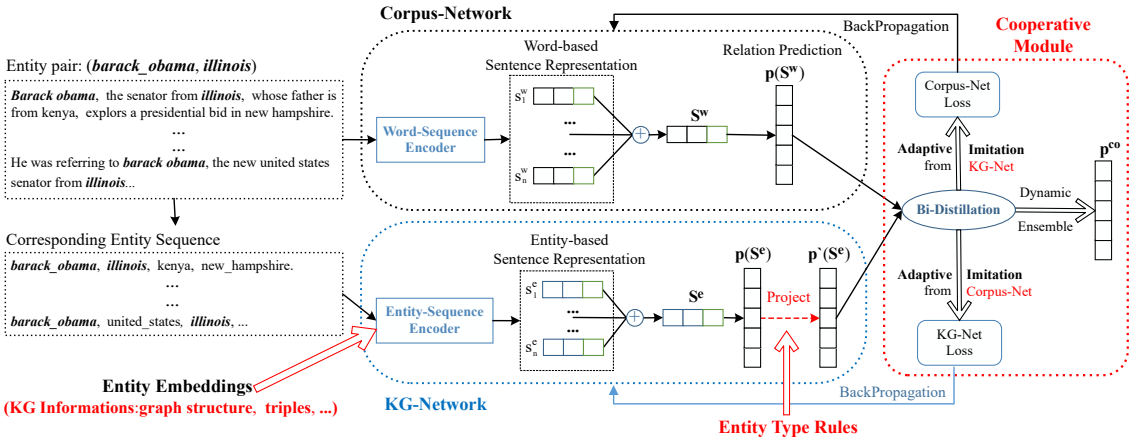


Figure 1: Overview of the proposed Cooperative Denoising Framework.

the information from KG, entity sequence can easily discriminate NA from other relations, since entities appeared in NA sentence usually are not connected in KG.

Motivated by the above observations, we propose a novel cooperative denoising framework (see Figure 1) to leverage the corpus-based and KG-based information. Specifically, we design two base networks, Corpus-Net and KG-Net, which are modeled with two separate Gated Recurrent Unit (GRU) networks, to predict relations using word-sequence and entity-sequence respectively. For KG-Net, we employ network embedding and KG embedding methods to pre-train the entity embeddings, and then project the prediction to a logic rule regularization subspace. Afterward, we design a cooperative module which involves the interactive learning between the two base networks with an adaptive bi-directional knowledge distillation mechanism, and the predictions of the base networks are dynamically integrated by an ensemble method. The key insight is that the base networks trained on different sources can learn complementary information, and thus the cooperative learning can benefit from the complementarity of different expressions of the same relational fact.

Our main contributions are as follows:

- We explore the feasibility of distantly supervised relation extraction by leveraging the information from different sources cooperatively.
- We devise a bi-directional knowledge distillation mechanism to enhance each base network via supplementary supervision.
- We design an adaptive imitation rate setting and a dynamic ensemble strategy to guide the training procedure and help the prediction of noisy-varying instances.
- The experimental results on a benchmark dataset show that the proposed method has robust superiority over compared methods.

2 Methodology

In this paper, we focus on the task of distantly supervised relation extraction. Our goal is to predict relation r for a given entity pair $< e_1, e_2 >$. The proposed framework CORD conducts with multi-instance learning, i.e., we take a bag of sentences mentioning both entity e_1 and e_2 as input, and we compute the probabilities for each relation expressed by this bag as output.

As Figure 1 shows, given a collection of sentences containing the target entity pair, we first transform each sentence into its distributed word-sequence and entity-sequence representations, and predict relation respectively using the attention weighted representation via a multi-instance learning mechanism. We also project the prediction of KG-Net to a logic rule regularization subspace. Then, we train the two base networks simultaneously with a bi-directional knowledge distillation method, in which the predictions of KG-Net and Corpus-Net are used as soft labels for each other. The final prediction is the ensemble

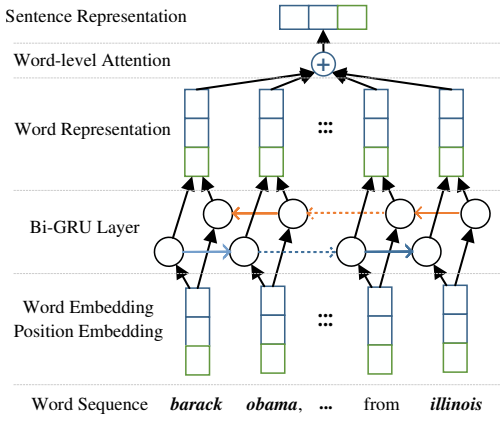


Figure 2: Word-Sequence Encoder.

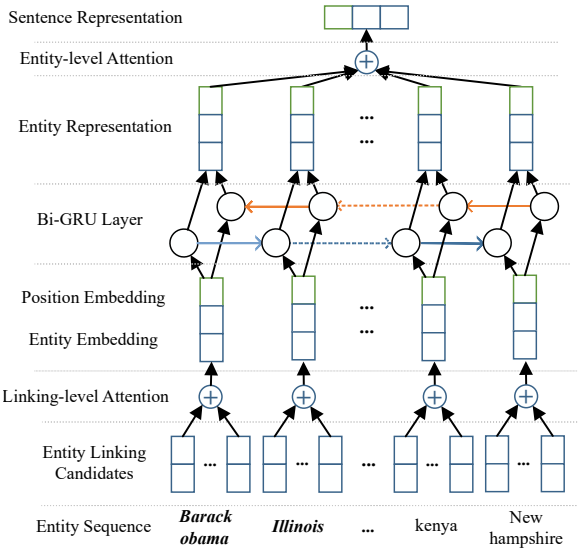


Figure 3: Entity-Sequence Encoder.

of the two base networks, and their weights in the ensemble are dynamically adjusted. In the rest of this section, we elaborate each component of CORD in detail.

2.1 Corpus-Network

We first introduce the word-sequence encoder shown in Figure 2, which transforms each sentence s_i to corresponding word-based sentence representation $\mathbf{s_i^w}$.

Input Representation Each word in the sentence is mapped to a low-dimensional embedding $w_i \in \mathbb{R}^{d_w}$ through a word embedding layer, where d_w denotes the size of the embedding. Similar to Zeng et al. (2014) and Lin et al. (2016), we encode the relative distances to entity e_1 and e_2 as $p_i \in \mathbb{R}^{2d_p}$, where $2d_p$ is the size of the position embedding. The distances are helpful in emphasizing how informative the word is thereby enabling better discrimination for relation extraction. We concatenate word vector w_i and position vector p_i as $\mathbf{w_i} = w_i \| p_i$, $\mathbf{w_i} \in \mathbb{R}^{d_w+2d_p}$, and feed words input $\mathbf{s_i} = \{\mathbf{w_1}, \ldots, \mathbf{w_n}\}$ to Bi-GRU layer.

Bi-directional GRU Layer We employ bi-directional GRU to encode patterns in $\mathbf{s}_i$ into a hidden representation $\mathbf{s_i^w} = \{h_1^f \| h_T^b, \ldots, h_T^f \| h_1^b\}$, which is obtained by concatenating each forward state h_j^f and backward state h_{T-j+1}^b at step j given input $\mathbf{s}_i$, $\mathbf{s_i^w} \in \mathbb{R}^{2d_h}$ and d_h means the hidden size of GRU. The forward sequence $[h_1^f, h_2^f, \ldots, h_T^f]$ and backward sequence $[h_1^b, h_2^b, \ldots, h_T^b]$ are calculated by GRU (Cho et al., 2014).

Hierarchical Attention As Figure 2 and Figure 1 show, we apply hierarchical attention for Corpus-Net involving word and sentence levels respectively. The motivation is that the semantic meaning of a sentence bag representation is gathered by vectors in different levels from the bottom to up, and the vectors in different context do not contribute equally. Some words express targeted relation more relevantly than others in a sentence. Furthermore, since we concatenate position vector to each word, a word with different distance is also importance-varying. As for sentence-level attention, it can reduce the weights of the sentences that suffer from wrong-labeling and improper-bagging (i.e., express inconsistent relations) problems. Inspired by Lin et al. (2016), we calculate the aggregation of different level attentions with unified form as:

$$\mathbf{X} = \sum_{i=1}^{n} a_i \mathbf{x}_i; \qquad a_i = \frac{exp(\mathbf{x_i A r})}{\sum_{j=1}^{n} exp(\mathbf{x_j A r})}, \tag{1}$$

where $\mathbf{x_i}$ is individual input vector in different levels, $\mathbf{X}$ is the corresponding sum of them, $\mathbf{r}$ is randomly initialized global relation vector. Note that we set different $\mathbf{r}$ and weighted diagonal matrix $\mathbf{A}$ for different attention levels.

Finally, the Bi-GRU output of each word $\mathbf{w_i}$ is gathered to sentence representation $\mathbf{s^w}$, and then to the bag-wise representation $\mathbf{S^w}$. We feed the resulting vector $\mathbf{S^w}$ to a *Softmax* classifier.

Prediction With $\mathbf{S^w}$ as input, the condition probability of each relation j is calculated as:

$$\mathbf{p_j}(\mathbf{S^w}) = \frac{\exp(o_j)}{\sum_{i=1}^{n_r} \exp(o_i)}; \qquad \mathbf{o} = \mathbf{W}\mathbf{S^w} + \mathbf{b}, \tag{2}$$

where $\mathbf{o} = [o_1, \ldots, o_{n_r}]$ is calculated with coefficient matrix $\mathbf{W} \in \mathbb{R}^{n_r \times (2d_h)}$ and bias $\mathbf{b} \in \mathbb{R}^{n_r}$. n_r is the number of relations and o_j measures how well $\mathbf{S^w}$ matches relation j.

2.2 KG-Network

Besides Corpus-Net, we propose another base network to incorporate information of KG. The entity-sequence encoder transforms each sentence s_i to entity-based sentence representation $\mathbf{s_i^e}$ as illustrated in Figure 3.

Input Representation One of the key challenges of leveraging KG information is identifying what information to use. Here we employ an extensible manner by using different entity embedding methods. For instance, *network embedding* methods such as DeepWalk (Perozzi et al., 2014) primarily encode graph structure information, while *knowledge embedding* methods such as TransE (Bordes et al., 2013) usually focus on triples information, we can flexibly use one of them or merge them together. Specifically, we link the detected entity names in sentences to the Freebase5M (Bordes et al., 2015) by n-gram text matching, and use the DeepWalk and TransE embeddings of linked entity candidates denoted as $\{e_{1,1}, e_{1,2}, \ldots, e_{m,k-1}, e_{m,k}\}$, where k is the amount of candidates for each entity, m is the amount of entities appearing in sentence.

In addition, we use the position vectors of Corpus-Net for KG-Net because the word-based distances are more discriminative than the entity-based distances. To be specific, the transformation between them is not one-to-one mapping because different word-sequences may result in the same entity-sequence and hence loss of information, e.g., "Obama flew to the US" and "Obama left the US" are both mapped to "Obama, US".

Bi-GRU and Attention Then we employ a similar architecture of the Bi-GRU component and the attention aggregation of Corpus-Net. As shown in Figure 3, the linked candidates $\{e_{i,1}, \ldots, e_{i,k}\}$ are aggregated to e_i for each entity, then entity vectors $\{e_1, \ldots, e_m\}$ and position vectors $\{p_1, \ldots, p_m\}$ are concatenated in element-wise and as input to Bi-GRU layer. The Bi-GRU outputs are gathered to $\mathbf{s^e}$ with entity-level attention, then to $\mathbf{S^e}$ with bag-level attention, and fed to a *Softmax* classifier similar to Corpus-Net.

External Rule Knowledge We regard the patterns learned automatically from word and entity sequences as internal knowledge and manage to transfer external human knowledge such as logic rules into the base network. Furthermore, the KG-specific rules can be incorporated into Corpus-Net gradually by bi-distillation and vice versa.

Here we concentrate on relation-specific type rules because: (1) We observe some typical false predictions which could be corrected with type rules (e.g., it is unreasonable to predict two person entities as relation *place of birth* whose tail type cannot be a *person*). (2) We can automatically obtain large-scale type resources because most KG reserved this information. For example, in Freebase, we can collect the types of entities located in *type/instance* field and the relation-specific type constraints located in *rdf-schema#domain* and *rdf-schema#range* fields.

To be specific, given all types (an entity can have many types) for an entity pair j as $t_{j,1}$ and $t_{j,2}$, we design the logic rule for each relation i as $T_{i,1}\ and\ T_{i,2}\ are\ not\ missing \implies (T_{i,1} \in t_{j,1}) \wedge (T_{i,2} \in t_{j,2})$ where $T_{i,1}$ and $T_{i,2}$ are the relation-specific type constrains for relation i. Here we apply probabilistic soft logic (Bach et al., 2017) to encode rules flexibly, i.e., the rule scores are continuous truth values in the internal $[0, 1]$ rather than $\{0, 1\}$ and logic $\wedge$ denotes averaging of truth values.

Moreover, considering type granularity, i.e., type information that is usually specified with hierarchy form (e.g., /people/person/spouse), we divide the number of matched levels (fields) of the type hierarchy by field amount as the value of $(T_i \in t_j)$ to gain fine-grained features.

Finally, with prediction $\mathbf{p}(\mathbf{S^e})$ from *Softmax* classifier, we use a posterior regularization fashion (Hu et al., 2016) to project $\mathbf{p}(\mathbf{S^e})$ into a constrained subspace $\mathbf{p}'(\mathbf{S^e})$ as follows:

$$(\mathbf{p}'(\mathbf{S^e}))^* \propto \mathbf{p}(\mathbf{S^e}) \times \mathrm{e}^{s_r}, \qquad s_r = -\sum_{i=1}^{l} C\lambda_i(1 - r_i(\mathbf{S^e})), \tag{3}$$

where λ_i is confidence of each rule, C is regularization parameter and s_r is rule score factor, which indicates how $\mathbf{S^e}$ satisfies the rules. This is the closed-form solution obtained by solving an optimization problem which finds the optimal $\mathbf{p}'(\mathbf{S^e})$ fitting rules meanwhile staying to $\mathbf{p}(\mathbf{S^e})$. We set λ_i as $\mathbf{p}_i(\mathbf{S^e})$, i.e., the higher probability classifier predicts (believes), the stronger effect rule-constraint takes for relation i. We can design other rules and enable to scale with similar manner.

2.3 Cooperative Module

In this section, we introduce how to ensemble two base networks cooperatively.

Bi-directional Knowledge Distillation We observe that KG-Net and Corpus-Net have different hard examples and different wrong predictions, i.e., for the same sentence bag, Corpus-Net may predict higher probability than KG-Net and sometimes, the contrary (we demonstrate the differences in Experiments Section). This observation encouraged us to train them cooperatively with mutual knowledge supplementation.

We devise a bi-directional knowledge distillation method to enhance their supervision information in label space. Specifically, the two base networks learn with the hard label $\mathbf{y}$ from distant supervision. Meanwhile, we set the predicted probability of the two base networks $\mathbf{p}^c$ and $\mathbf{p}^k$ as soft label to each other simultaneously:

$$L_c = \sum_{i=1}^{N}(\ell(\mathbf{y}_i, \mathbf{p}_i^c) + \pi_c \ell(\mathbf{p}_i^k, \mathbf{p}_i^c)), \qquad L_k = \sum_{i=1}^{N}(\ell(\mathbf{y}_i, \mathbf{p}_i^k) + \pi_k \ell(\mathbf{p}_i^c, \mathbf{p}_i^k)), \tag{4}$$

where ℓ is cross entropy loss, π is imitation rate, and N is batch size. We update the model parameters by minimizing L_c and L_k with Adam (Kingma and Ba, 2014) optimizer.

The learning process can be regarded as the fact that the two base networks not only learn from the coarse-grained hard label which is one-hot and low entropy, but also learn from the teacher network which expresses specific supplementary knowledge and dependencies between relations with soft label. For example, label $[0.3, 0.2, 0.9]$ is more informative than $[0, 0, 1]$.

Also note that the early base network is not reliable and gives low quality knowledge through soft label, so we pre-train the two base networks separately with certain steps before mutual learning.

Adaptive Imitation Rate The classification difficulty of the two base networks is varying with different entity pair bag instance, sometimes KG-Net is a better qualified teacher for Corpus-Net and sometimes vice versa. To transfer more reliable knowledge of each base network to another and train them more effectively, we set the imitation weights as following:

$$\pi_c = \frac{\ell(\mathbf{y}_i, \mathbf{p}_i^k)}{\ell(\mathbf{y}_i, \mathbf{p}_i^c) + \ell(\mathbf{y}_i, \mathbf{p}_i^k)}, \qquad \pi_k = \frac{\ell(\mathbf{y}_i, \mathbf{p}_i^c)}{\ell(\mathbf{y}_i, \mathbf{p}_i^c) + \ell(\mathbf{y}_i, \mathbf{p}_i^k)}, \tag{5}$$

where π_c and π_k are inversely proportional to the hard-label loss of each other, i.e., the smaller the loss is, the more qualified is the base network as teacher toward each other. In addition, from the perspective of optimization, the adaptive imitation can prevent the gradient from being dominated by ill-classified examples and hence be able to train the model effectively.

Later, in Section 3.4, we will demonstrate the effectiveness of the adaptive imitation rate by comparing with the fixed setting.

Dynamic Ensemble Prediction The final prediction $\mathbf{p}^{co}$ of the CORD framework is an ensemble of the two base networks predictions $\mathbf{p}^c$ and $\mathbf{p}^k$ because each of them has its strong points. We propose a dynamic ensemble strategy considering that (1) A high type-rule score indicates KG-Net may classify current sentence bag well because the predictions of the classifier satisfy the rules; (2) Ideally, entity

name in a sentence should be linked to only one entity in KG, so the KG-Net would be more confused with more linked candidates. Thus, with the above two factors, we can specify the dynamic ensemble prediction $\mathbf{p}^{co}$ as follows:

$$\mathbf{p}^{co} = (1 - w_k)\mathbf{p}^c + w_k\mathbf{p}^k, \qquad w_k = \alpha + \beta(\frac{s_r}{n_r} - \frac{n_c}{n_e \times N_e}), \tag{6}$$

where α is the empirical KG-Net base weight, β is the wave range. They can be set as the ratio of some evaluation indicators (such as F-score) of separate-trained base network. Then the prediction weight of KG-Net $w_k \in [\alpha - \beta, \alpha + \beta]$ depends on the normalized ($\in [0, 1]$) rule score factor and candidates score, i.e., average rule score per-relation and number of candidates per-entity dividing N_e, which is the upper limit on the number of candidates for linking. And s_r is the rule score factor in Eq. 3, n_r, n_c n_e are the amounts of relations, all entities candidates and gathered entities in sentence respectively.

As a comparison, we also deploy a naive baseline using static ensemble weight and report the results in Section 3.4.

3 Experiments

In this section, we aim to evaluate the effectiveness of the proposed CORD framework. We conduct an overall performance comparison with baseline methods and perform a comprehensive examination of the KG-Net and the Cooperative Module.

3.1 Experiment setup

Dataset and Evaluation Metrics We conduct experiments on the widely used benchmark dataset NYT10 (Riedel et al., 2010), which is built by aligning triples in Freebase to the New York Times corpus and contains 53 relations. There are 522,611/172,448 sentences, 281,270/96,678 entity pairs, and 18,252/1,950 relation mentions in train/test dataset respectively. Following previous works (Mintz et al., 2009; Lin et al., 2016), we evaluate our method in the held-out evaluation with P-R curve and P@N metric without expensive human evaluations.

Parameter Settings We set the embedding dimensions as 5, 50, 64, 64 for position, word2vec, DeepWalk and TransE respectively. For both base networks, we set the cell size of GRU as 230, learning rate as 0.001, dropout probability as 0.5 and batch size as 20. For the cooperative module, we set the base weight α and wave range β as 0.4, 0.2 respectively, and fixed w_k as 0.4 for ensemble comparison.

3.2 Comparison with Baseline Methods

We compare our approach with three traditional feature-based methods and two state-of-art neural-based methods.

Feature-based Methods *Mintz* (Mintz et al., 2009) is a multiclass logistic regression model; *MultiR* (Hoffmann et al., 2011) is a probabilistic graphical model which can handle overlapping relations; *MIML* (Surdeanu et al., 2012) is also a probabilistic graphical model but using a multi-instance multi-label paradigm.

Neural-based Methods *CNN+ATT* (Lin et al., 2016) is a sentence-level attention model based on CNN, which can dynamically reduce the weights of noisy instances; *PCNN+ATT* (Lin et al., 2016) achieves state-of-art results by applying sentence-level attention to the piecewise max pooling model, PCNN (Zeng et al., 2015).

The precision-recall curve results are shown in Figure 4, where *Base-Net-Corpus* and *Base-Net-KG* are our best results for the two base networks with independent training, *CORD* is the cooperative training and dynamic ensemble results. For the aforementioned five methods, we directly use the results reported in (Lin et al., 2016).

Figure 4 demonstrates that: (1) The KG-Net achieves higher coverage than the feature-based methods, comparable precision with the *MultiR* and *MIML* and an obvious gap with other neural-based methods. This indicates that the KG-Net and feature-based methods can capture certain patterns effectively but with relatively low coverage. On one hand, the decent precision shows potentials of the KG-Net to capture patterns with entity-sequence and KG information. On the other hand, we suggest that the

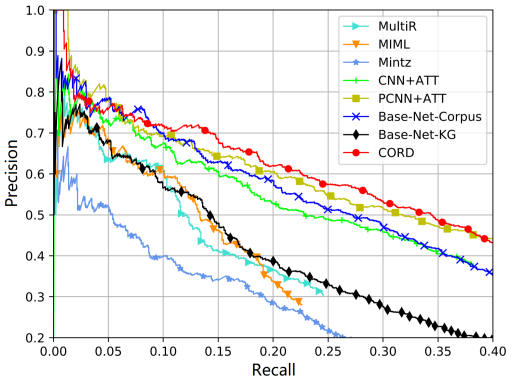


Figure 4: P-R Curve Comparison with Mintz, MIML, MultiR, CNN+ATT and PCNN+ATT.

weakness of the KG-Net might be caused by the sparsity of entity-sequence space (the dataset scales down after word-to-entity mapping), and we can enhance it by exploring other information such as relational paths (Lin et al., 2015; Zeng et al., 2017); (2) Corpus-Net achieves comparable results with *CNN+ATT* and *PCNN+ATT*, which reveals the effectiveness of Corpus-Net that could be the backbone of the CORD framework; (3) The CORD outperforms other methods on most recall area, demonstrating the effectiveness of our methods. Note that the CORD framework is significantly superior to the two separate trained base networks, especially in the rightmost area. This shows the cooperative module takes advantages of two base networks effectively and achieves better generalization. Also note that in the right-side area, CORD is still robust although the separate trained KG-Net is weak, which verifies the effectiveness of the CORD with different strengths of the base networks.

3.3 Performance of the KG Network

To evaluate the effect of incorporating KG information, we first compare P-R curve for different KG-Net setups, then we explore the benefit of using external logic rules and make a case study.

Comparison for Different Setups We experiment three KG-Nets without rule knowledge, using DeepWalk, TransE and their concatenation as entity embedding respectively. Based on whichever yields the best results, we experiment with rule knowledge and report results in Figure 5.

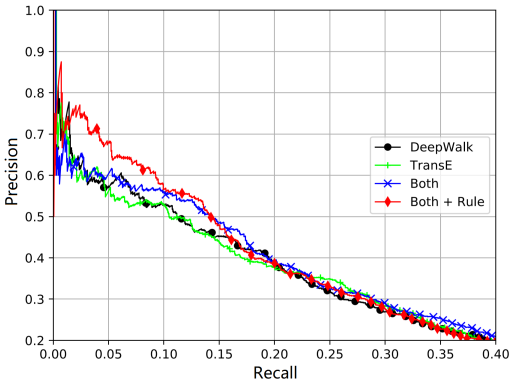


Figure 5: P-R Curve Comparison with Different Setup for the Base KG-Net.

From Figure 5, we can observe that: (1) The results of DeepWalk and TransE are slightly different and their concatenation improves, verifying the extendibility with different kinds of KG information embeddings; (2) After incorporating logic rules, the result is improved significantly as shown in left recall area, indicating that type-constraints helps to capture certain patterns more precisely.

Robustness on Long Tail Situation Efforts based on bag-level denoising such as sentence attention are liable to failure because of the long-tail situation in real-life datasets. For example, we observe that 77.63% entity pairs have only one relation instance in NYT10. We expect that supplementary external knowledge (logic rules) can enhance the robustness of this situation. We evaluate P@N of the KG-Net without rules $\mathbf{p}(\mathbf{S^e})$ and compare the rule-projected $\mathbf{p'}(\mathbf{S^e})$ on two kind of sentence amounts setup in

Table 1, where (≥ 1) means whole test dataset and (> 1) means filtering entity pairs which have only one instance.

	#Entity Pair Sentence	P@100 (%)	P@200 (%)	P@300 (%)	Average (%)
KG-Net	>1	60.2	54.0	46.4	53.5
	≥ 1	**60.3(+0.1)**	**51.5(-2.5)**	**46.0(-0.4)**	**52.6(-0.9)**
KG-Net + Rule	>1	69.2	57.0	48.6	58.3
	≥ 1	**73.5(+4.3)**	**60.4(+3.4)**	**51.7(+3.1)**	**61.9(+3.6)**

Table 1: P@N for Long Tail Situation.

From Table 1, we can observe that the KG-Net gets lower precisions (average reduces 0.9%) on the whole test data comparing with the filtered data. In contrast, the KG-Net with rules gets higher precisions on the whole test data because it can deal with noisy instances effectively in sentence-level and hence be more robust on long tail situation.

Case Study Here we pick an example instance in Table 2 to illustrate the effect of type rules. The KG-Net predicts wrong relation *place of death* probably because the appearance of entity *Joe Williams*. In contrast, it can predict correctly with the help of relation-specific type constraints.

	Predicted Relation	Relation-Specific Type
KG-Net	/people/deceased_person/place_of_death	/people/deceased_person, /location/location
KG-Net+ Rule	/location/location/contains	/location/location, /location/location
text	In ***Suffolk County***, ***Fire Island*** suffered the most damage, according to **Joe Williams**, commissioner of the county 's office of emergency management.	

Table 2: Effect of Type Constraint Rules. Bold indicates entity, italic indicates targeted entity pair.

3.4 Performance of the Cooperative Module

To investigate the effectiveness of the cooperative module, we compare the adaptive imitation rate with the fixed, the dynamic ensemble strategy with the static, and then perform a thorough case study.

Adaptive vs Fixed Imitation Rate We find that the adaptive imitation is crucial for the effective training of the CORD. To demonstrate this, we deploy some fixed imitation rate setups comparing the adaptive imitation rate. We set imitation rate (π_c, π_k) as $\{(0.5, 0.5), (0.6, 0.4), (0.4, 0.6)\}$, and report the loss curves of the dynamic, and only $(0.5, 0.5)$ for the fixed in Figure 6 because the other two have similar results.

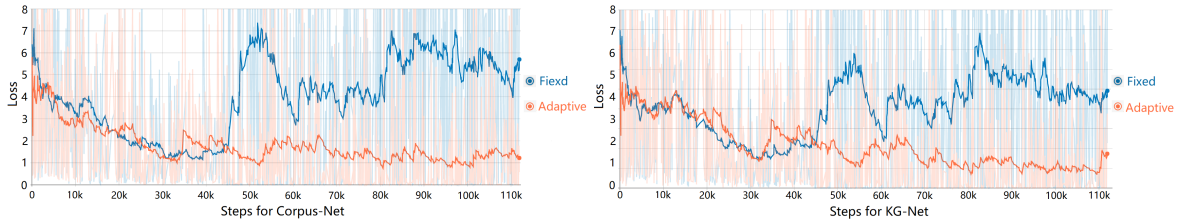


Figure 6: Loss for Adaptive and Fixed Imitation.

Figure 6 shows the remarkable difference between the fixed and the adaptive imitation, where the loss of the adaptive setting reduce gradually while the fixed fluctuates wildly. The waves of fixed KG-Net and fixed Corpus-Net are similar, meanwhile they mislead each other and the gradients are dominated by hard examples, resulting in the non-convergence. Conversely, the adaptive networks are trained effectively because the data speak for itself by providing loss values as clues to reveal the difficulty of predicting current instance. Note that the base networks are pre-trained independently and both descend gradually within the first 10k steps.

Dynamic vs Static Ensemble We also compare the dynamic ensemble strategy with the static and report their P@N results within identical base networks in Table 3. It shows that the dynamic outperforms the static and leverages the two base networks better. Note that the improvements decrease as recall

	P@100 (%)	P@200 (%)	P@300 (%)	Average (%)
Base-KG	64.0	55.2	49.3	56.2
Base-Corpus	77.3	72.0	66.5	73.9
Static	80.7	73.1	64.0	72.8
Dynamic	**84.5**	**76.0**	**66.7**	**76.1**

Table 3: P@N for Dynamic and Static Ensemble.

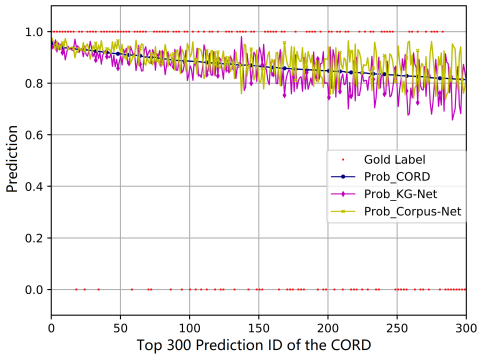


Figure 7: Top300 Predictions of the CORD.

increases, and the degree of decrease for the dynamic is less than the static, demonstrating the dynamic is more robust than the static.

Case Study To gain further insight about how the CORD works, we plot the top 300 predictions of the CORD in descending order, and compare it with its two base networks in Figure 7.

Figure 7 presents the predictions of KG-Net and Corpus-Net which go up and down around the CORD curve. The prediction differences between KG-Net and Corpus-Net are nonuniformly time-varying, indicating that some hard instances for KG-Net are easy to classify by Corpus-Net and vice versa. This supports again our view of employing adaptive imitation and dynamic ensemble.

To further demonstrate the different advantages of KG-Net and Corpus-Net, we choose two points which are predicted correctly and have significantly different values from Figure 7. The result is showed in Table 4, where prediction ID 137 and ID 257 have the different dominating network, KG-Net and Corpus-Net respectively.

ID in Figure 7	Prob-KG	Prob-Corpus	Relation	Text
137	0.9809	0.7591	location contains	when **Julian Resuello**, the mayor of ***San Carlos City*** in the northern ***Philippines***, was killed by gunmen at a campaign rally on April 28, his brother quickly stepped into his shoes.
257	0.6763	0.9583	place of death	***Ernst Haefliger***, a **Swiss** tenor who was most renowned as an interpreter of **German** art song and oratorio roles, died on Saturday in ***Davos***, **Switzerland**.

Table 4: Hard Example of KG and Corpus Net. Bold indicates entity, italic indicates targeted entity pair.

From Table 4, we can see that different networks contribute each other from the view of semantic: (1) KG-Net predicts relation *location contains* more accurately and Corpus-Net may fail if the wording of the sentence doesn't clearly state the relation between two entities. With the help of position and three entity embeddings, KG-Net can capture the relation-dependent features better from the view of graph structure. In contrast, Corpus-Net might be confused by the expression "the mayor of ..." and the uncorrelated latter part, "was killed by ..."; (2) Corpus-Net predicts relation *place of death* more accurately and KG-Net may fail if two entities are already known to be related by more than one relation. Here Corpus-Net provides reliable prediction because of the appearance of featured expression "died on ..." in the last sentence. Contrarily, KG-net may lack discriminative information and be confused by other possible relations such as *place of birth*, because the targeted person entity is followed by too many location entities.

4 Related Work

Relation extraction is one of the most important topics in NLP. Many approaches to relation extraction have been proposed, such as supervised classification (Zelenko et al., 2003; Bunescu and Mooney, 2005), bootstrapping (Carlson et al., 2010), distant supervision (Mintz et al., 2009; Krause et al., 2012; Min et al., 2013; Pershina et al., 2014; Ji et al., 2017), and generative model (Zhang et al., 2018). Among them,

distant supervision is popular as it is efficient to obtain large-scale training data automatically. However, it suffers from noisy labeling problem which severely degrades its performance.

To tackle this problem, Riedel et al. (2010; Hoffmann et al. (2011; Surdeanu et al. (2012) model distant supervision as a multi-instance learning problem under the at-least-one assumption and make it more practical. With the advances in deep learning, Zeng et al. (2015), Lin et al. (2016) and Lin et al. (2017) apply CNN and attention mechanism, Feng et al. (2017) further introduces memory network to reduce noises. Compared with these methods, the proposed framework leverages information from other sources such as KG and combine it with information from text corpus by knowledge distillation.

5 Conclusion

In this paper, we propose a novel neural relation extraction framework with bi-directional knowledge distillation to cooperatively use different information sources and alleviate the noisy label problem in distantly supervised relation extraction. Extensive experiments show that our framework can effectively model relation patterns between text corpus and KG information, and achieve the state-of-the-art results.

Acknowledgements

We thank anonymous reviewers for their helpful comments. This work was financially supported by the National Natural Science Foundation of China (No.61602013), and the Shenzhen Science and Technology Innovation Committee (Grant No. JCYJ20170412151008290 and JCYJ20170818091546869).

References

[Abujabal et al.2017] Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. 2017. Automated template generation for question answering over knowledge graphs. In *WWW*, pages 1191–1200.

[Bach et al.2017] Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2017. Hinge-loss Markov random fields and probabilistic soft logic. *JMLR*, 18(109):1–67.

[Bordes et al.2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795.

[Bordes et al.2015] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. In *NIPS*.

[Bunescu and Mooney2005] Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *HLT/EMNLP*, pages 724–731.

[Carlson et al.2010] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3.

[Cho et al.2014] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

[Feng et al.2017] Xiaocheng Feng, Jiang Guo, Bing Qin, Ting Liu, and Yongjie Liu. 2017. Effective deep memory networks for distant supervised relation extraction. In *IJCAI*, pages 19–25.

[Hoffmann et al.2011] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL*, pages 541–550.

[Hu et al.2016] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. *ACL*.

[Ji et al.2017] Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *AAAI*, pages 3060–3066.

[Kadry and Dietz2017] Amina Kadry and Laura Dietz. 2017. Open relation extraction for support passage retrieval: Merit and open issues. In *SIGIR*, pages 1149–1152.

[Kingma and Ba2014] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Krause et al.2012] Sebastian Krause, Hong Li, Hans Uszkoreit, and Feiyu Xu. 2012. Large-scale learning of relation-extraction rules with distant supervision from the web. In *ISWC*, pages 263–278.

[Lin et al.2015] Yankai Lin, Zhiyuan Liu, Huan-Bo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling relation paths for representation learning of knowledge bases. In *EMNLP*, pages 705–714.

[Lin et al.2016] Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *ACL*, volume 1, pages 2124–2133.

[Lin et al.2017] Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. Neural relation extraction with multi-lingual attention. In *ACL*, volume 1, pages 34–43.

[Luo et al.2017] Bingfeng Luo, Yansong Feng, Zheng Wang, Zhanxing Zhu, Songfang Huang, Rui Yan, and Dongyan Zhao. 2017. Learning with noise: Enhance distantly supervised relation extraction with dynamic transition matrix. In *ACL*, pages 430–439.

[Min et al.2013] Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *NAACL HLT*, pages 777–782.

[Mintz et al.2009] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL*, pages 1003–1011.

[Perozzi et al.2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*, pages 701–710.

[Pershina et al.2014] Maria Pershina, Bonan Min, Wei Xu, and Ralph Grishman. 2014. Infusion of labeled data into distant supervision for relation extraction. In *ACL*, volume 2, pages 732–738.

[Riedel et al.2010] Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *ECML PKDD*, pages 148–163.

[Shin et al.2015] Jaeho Shin, Sen Wu, Feiran Wang, Christopher De Sa, Ce Zhang, and Christopher Ré. 2015. Incremental knowledge base construction using deepdive. *VLDB*, 8(11):1310–1321.

[Surdeanu et al.2012] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *EMNLP*, pages 455–465.

[Zelenko et al.2003] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *JMLR*, 3(Feb):1083–1106.

[Zeng et al.2014] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344.

[Zeng et al.2015] Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*, pages 1753–1762.

[Zeng et al.2017] Wenyuan Zeng, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. Incorporating relation paths in neural relation extraction. In *EMNLP*, pages 1768–1777.

[Zhang et al.2018] Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip S. Yu. 2018. On the generative discovery of structured medical knowledge. In *SIGKDD*.

Adversarial Feature Adaptation for Cross-lingual Relation Classification

Bowei Zou, Zengzhuang Xu, Yu Hong, Guodong Zhou*
School of Computer Science and Technology, Soochow University,
Suzhou, 215006, China
zoubowei@suda.edu.cn, nedxuwork@gmail.com,
{yhong, gdzou}@suda.edu.cn

Abstract

Relation Classification aims to classify the semantic relationship between two marked entities in a given sentence. It plays a vital role in a variety of natural language processing applications. Most existing methods focus on exploiting mono-lingual data, e.g., in English, due to the lack of annotated data in other languages. In this paper, we come up with a feature adaptation approach for cross-lingual relation classification, which employs a generative adversarial network (GAN) to transfer feature representations from one language with rich annotated data to another language with scarce annotated data. Such a feature adaptation approach enables feature imitation via the competition between a relation classification network and a rival discriminator. Experimental results on the ACE 2005 multilingual training corpus, treating English as the source language and Chinese the target, demonstrate the effectiveness of our proposed approach, yielding an improvement of 5.7% over the state-of-the-art.

1 Introduction

Relation classification aims to identify the semantic relationship between two nominals labeled in a given sentence. It is critical to many natural language processing (NLP) applications, such as question answering and knowledge base population. For example, the following sentence contains an instance of the *Content-Container*(e_2,e_1) relation between two labeled entity mentions "e_1=*cartridge*" and "e_2=*ink*".

The [***cartridge***]$_{e_1}$ *was marked as empty, even with* [***ink***]$_{e_2}$ *in both chambers.*

An open challenge is how to train a model which is suitable for languages with insufficient available data of relation classification, since manual annotation is time-consuming and human-intensive. This makes it difficult to transferrably use the existing well-trained classification models in other languages.

To tackle this problem, we propose an adversarial feature adaptation approach to transfer latent feature representations from the source language with rich labeled data to the target language with only unlabeled data. Such an approach are both discriminative for relation classification and invariant across languages. This is largely motivated by the adversarial mechanism which has been effectively applied to measure the similarity between distributions in a variety of scenarios, such as domain adaptation (Bousmalis et al., 2016) and multi-modal representation learning (Park and Im, 2016).

In particular, we build two counterpart networks, using convolutional neural networks (CNNs), for source language and target language, to generate the latent feature representations respectively. Then, we use a rival discriminator to identify the correct source of feature representations. At the training step, the network of the source language is trained to maximize the performance on the annotated dataset, while the network of the target language is trained to imitate the feature representations of the source language by rival confusing the discriminator.

We perform the proposed approach on ACE 2005 multilingual training corpus by regarding English as the richly-labeled language (source) and Chinese as the poorly-labeled language (target). Our approach

*corresponding author

Proceedings of the 27th International Conference on Computational Linguistics, pages 437–448
Santa Fe, New Mexico, USA, August 20-26, 2018.

achieves an F1-score of 70.50% with a significant improvement of 5.7%, compared to the state-of-the-art method.

The main contributions of this study are as follows.

- We present a novel neural feature adaptation framework by leveraging a generative adversarial network to transfer feature representations from a richly-labeled language to a poorly-labeled language. To the best of our knowledge, this is the first study on feature adaptation for cross-lingual relation classification.
- Experimental results show that the latent feature representations can be effectively transferred from the source language to the target language. This enables the adaptation of the existing manually annotated resource in one language to a new language.
- In a slightly better scenario that there are a small-scale annotated data available in the target language, our adversarial feature adaptation approach can also be effectively cooperated with the supervised model to further improve the overall performance.

The rest of this paper is organized as follows. In Section 2, we overview the related work. In Section 3, we introduce details of the proposed adversarial feature adaptation approach. We show our experimental results and discussions in Section 4. Finally, we conclude the paper in Section 5.

2 Related Work

In this section, we briefly review the recent progress in cross-lingual relation classification and existing studies on adversarial adaptation.

2.1 Cross-lingual Relation Classification

Labeled data on relation classification are not evenly distributed among languages. While there are various of annotated datasets in English, such as the SemEval'10 Task-8 dataset (Hendrickx et al., 2010) and the SemEval'18 task-7 dataset (Gábor et al., 2018), annotated datasets in other languages are few.

Traditional studies for relation classification usually perform supervised machine learning models trained on mono-lingual labeled datasets, which either rely on a set of linguistic or semantic features (Kambhatla, 2004; Suchanek et al., 2006), or apply tree kernel-based features to represent the input sentences (Bunescu and Mooney, 2005; Qian et al., 2008). Recently, deep neural networks (Zeng et al., 2015; dos Santos et al., 2015) and attention mechanism (Wang et al., 2016) show the effectiveness in relation classification. However, the training of neural network relies on large-scale labeled instances. This makes it difficult to re-construct such classification models for the poorly-labeled language.

Most of existing studies have attempted to leverage parallel data or a knowledge-based system to transfer effective information from the richly-labeled language to the poorly-labeled language. Qian et al. (2014) proposed a bilingual active learning paradigm for Chinese and English relation classification with pseudo parallel corpora and entity alignment. Kim et al., (2014) proposed a cross-lingual annotation projection strategy by employing parallel corpora for relation detection. Faruqui and Kumar (2015) also present a cross-lingual annotation projection method by using machine translation results, rather than parallel data. Verga et al. (2016) performs multi-lingual relation classification by a knowledge base. Min et al. (2017) drive a classifier to learn discriminative representations by joint supervision of classification (softmax) loss and ideal representation loss.

Instead of exploiting external resources and manually selecting a closeness metric, we come up with an adversarial mechanism to provide an adaptive metric for feature adaptation from the richly-labeled language to the poorly-labeled language.

2.2 Adversarial Adaptation

Recently, the generative adversarial networks (GAN) have become increasingly popular, especially in the area of deep generative unsupervised modeling (Goodfellow et al., 2014; Makhzani et al., 2016). For adversarial adaptation, Ganin et al. (2017) proposed the domain adversarial neural networks (DANN) to

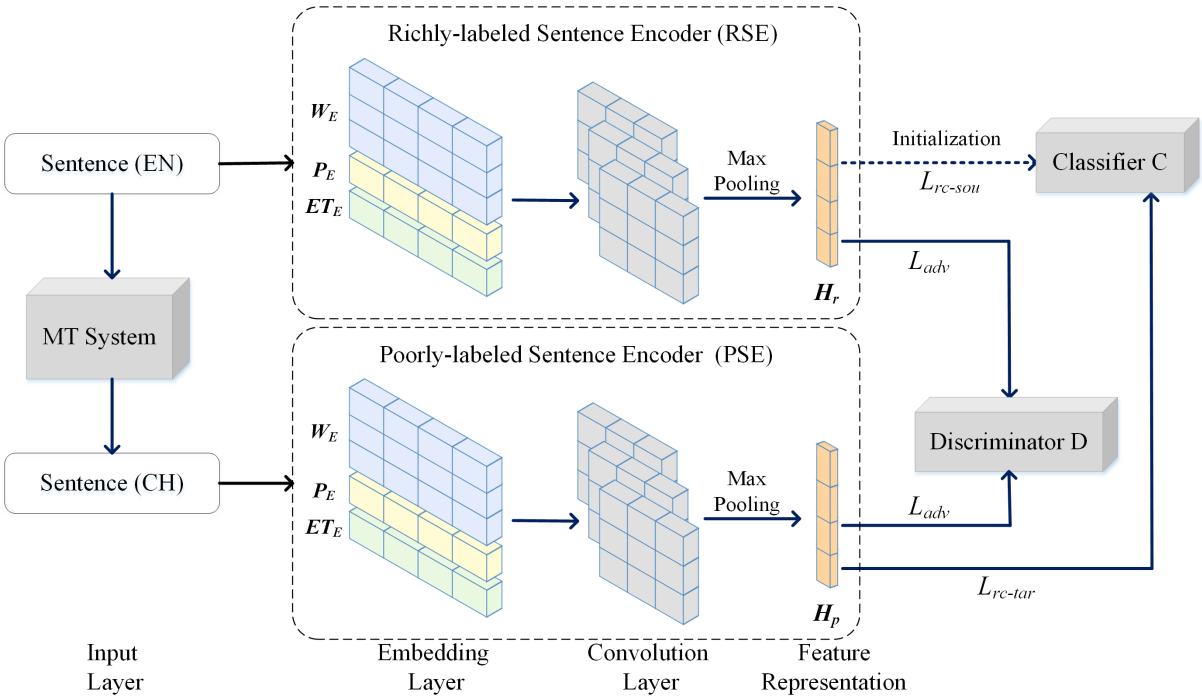


Figure 1: Architecture of the adversarial feature adaptation framework for cross-lingual relation classification.

learn discriminative but domain-invariant representations, transferring the information from the source domain to the target domain. Different from their study, our approach aims to find sharable language-independent latent feature representations for cross-lingual relation classification.

There have been some previous work applying adversarial adaptation technique to NLP tasks, such as that for sentiment analysis (Chen et al., 2016) and parsing (Sato et al., 2017). These studies learn the domain-invariant or domain-specific features by a shared network. Our work differs from them, since we force a second network with identical structure to learn the latent feature representations from the supervised network. Qin et al. (2017) propose a feature imitation approach. An adversarial mechanism is used between explicit and implicit discourse relation samples. Different from their study, we migrates the feature representations from one language to another (non-parallel). To the best of our knowledge, this is the first work to employ the adversarial feature adaptation for cross-lingual relation classification.

3 Adversarial Feature Adaptation for Cross-lingual Relation Classification

The common semantic information between different language motivates our adversarial feature adaptation approach. In this section, let us take a glance at the framework first, and then the relation classification networks (sentence encoders) and the adversarial training procedure.

Figure 1 illustrates the schematic overview of the framework which consists of four key components: 1) a Richly-labeled Sentence Encoder (RSE) with English sentences as the inputs, 2) a Poorly-labeled Sentence Encoder (PSE) which takes translated Chinese sentences as the input, 3) a language discriminator D distinguishing between the feature representations from the above two encoders, and 4) a relation classifier C to predict the relation label. In general, a GAN consists of a generative network G and a discriminator D, in which G generates instances by a distribution $P_{G(x)}$, and D aims to determining whether a instance is from $P_{G(x)}$ or the real data distribution $P_{data(x)}$. In our approach, the PSE is taken as the generative network which generates the feature representations $\boldsymbol{H_p}$ to confuse the discriminator.

In training step, the relation classifier aims to predict the labels, while the language discriminator attempts to distinguish between the feature representations extracted by the two sentence encoders ($\boldsymbol{H_p}$ or $\boldsymbol{H_r}$). In test step, we utilize the PSE to encode the input sentences in target language, and apply the same classifier to predict to relation labels.

3.1 Components

As a common neural network model that yields good performance for monolingual relation classification, we employ CNN to transform a sentence with pairs of entity mentions into a distributed representation $\boldsymbol{H}$. Note that, this plug-in architecture can also be implemented with the other networks, e.g., a long short-term memory network (LSTM)[1].

Embedding Layer

Following Zeng et al. (2014)'s work, we build an embedding layer to encode words, word positions, and entity types by real-valued vectors.

Given an input sentence $S = (w_1, w_2, ..., w_n)$, we first transform every word into a real-valued vector of dimension d_w using a word embedding matrix $\boldsymbol{W_E} \in \mathbb{R}^{d_w \times |V|}$, where V is the input vocabulary. Since the structures of RSE and PSE are the same, it is necessary if the word representations for both languages have a shared vocabulary. Therefore, bilingual word embeddings (Shi et al., 2015) are employed to map words from different languages into the same feature space.

To capture the informative features of the relationship between words and the entity mentions, we map the relative distances to entity mentions of each word to two real-valued vectors of dimension d_p using a position embedding matrix $\boldsymbol{P_E} \in \mathbb{R}^{d_p \times |D|}$, where D is the set of relative distances which are mapped to a vector initialized randomly (dos Santos et al., 2015). For each word, we obtain two position vectors with respect to the two entity mentions.

For each word, we also incorporate its entity type embedding to reflect the relationship between the entity type and the relation type. Each word is mapped to a real-valued vector using embedding matrix $\boldsymbol{ET_E} \in \mathbb{R}^{d_{et} \times |E|}$, where E is the set of entity types.

Finally, we represent a input sentence as a vector sequence $\mathbf{w}=\{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_n\}$ with the embedding dimension $d = (d_w + 2d_p + d_{et})$.

Convolution Layer

After encoding the input sentence, a convolution layer extracts local features by sliding a window of length w over the sentence and perform a convolution within each sliding window. The output for the ith sliding window is

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

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

Max-pooling Layer

We merge all local features via a max-pooling layer and apply a hyperbolic tangent function to obtain a fixed-sized final representations. The ith element of the output vector $\mathbf{x} \in \mathbb{R}^{d^c}$ is

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

Classifier and Discriminator

While the Classifier C is a fully-connected layer followed by a softmax classifier, the Discriminator D is a binary classifier which is implemented as a fully-connected neural network with a sigmoid activation function. Discriminator D takes the feature representations as input, to discriminate whether the feature representation comes from RSE or from PSE.

3.2 Adversarial Training

Although the aforementioned architecture could be applied to cross-lingual relation classification by leveraging the RSE module to train on source language and the MT module to translate the instances from

[1]As an alternative, a bidirectional LSTM (BiLSTM) is tried as the basic network. The experimental results are shown in Subsection 4.2.

Algorithm 1 Adversarial Training Procedure

Input: Training dataset

Output: RSE with classifier C

1: Initialize $\boldsymbol{\theta}_h$ and $\boldsymbol{\theta}_C$ by minimizing Eq.(3).
2: **repeat**
3: Train the discriminator D through Eq.(4)
4: Train PSE and classifier C through Eq.(7)
5: **until** convergence

the target language to the source language, there is no guarantee that the latent feature representations of the source language exist in the target language, or vice versa. On the other hand, the error propagation of MT module also should be considered. Therefore, we introduce adversarial training into our cross-lingual relation classification framework.

Algorithm 1 illustrates the adversarial training procedure. First, we pre-train the RSE and the classifier C by minimizing the relation classification (RC) loss function on source language (step line 1). Then we interleave the optimization of the adversarial loss function and the RC loss function on the target language at each iteration (step line 2-5). Finally, if the discriminator cannot tell the language of a input sentence using the adversarially trained features, then those features from PSE are effectively language-invariant. Upon successful training, the feature representations ($\boldsymbol{H_p}$) are thus encouraged to be both discriminative for relation classification and invariant across languages. Referring to the expression of Qin et al. (2017), the three loss function of our adversarial training are as follows.

RC Loss for Training on Source Language

We denote the parameters of the RSE and classifier C as $\boldsymbol{\theta}_r$ and $\boldsymbol{\theta}_C$, respectively, and the objective can be learned by minimizing the cross-entropy loss as

$$\mathcal{L}_{rc\text{-}sou}(\boldsymbol{\theta}_r, \boldsymbol{\theta}_C) = \mathbb{E}_{(\boldsymbol{x_e}, y)\sim data}\left[\mathcal{J}(C(H_r(\boldsymbol{x_e}; \boldsymbol{\theta}_r); \boldsymbol{\theta}_C), y)\right], \tag{3}$$

where $\mathbb{E}_{(\boldsymbol{x_e}, y)\sim data}[\cdot]$ denotes the expectation in terms of the data distribution, $\mathcal{J}(\boldsymbol{p}, y) = -\sum_k \mathrm{II}(y = k)\log p_k$ is the cross-entropy loss between predictive distribution $\boldsymbol{p}$ and ground-truth label y, $C(\boldsymbol{H_r}(\boldsymbol{x}))$ is the final prediction of classifier C when the input is the feature representation of RSE ($\boldsymbol{H_r}(\boldsymbol{x})$), $(\boldsymbol{x_e}, y)$ is the pair of input and output of relation classification model, where $\boldsymbol{x_e}$ is an English instance, and y is the relation label.

Adversarial Loss

The adversarial loss $\mathcal{L}_{adv}$ is used to train the discriminator D to make a correct estimation that where the feature representation comes from. Formally, the parameters of the discriminator D is denoted as $\boldsymbol{\theta}_D$. The training objective of D is to distinguish the input source of feature representation as far as possible:

$$\min_{\boldsymbol{\theta}_D} \mathcal{L}_{adv} = \mathbb{E}_{(\boldsymbol{x_e}, \boldsymbol{x_c}, y)\sim data}\left[\log(1 - D(\boldsymbol{H_r}(\boldsymbol{x_e}; \boldsymbol{\theta}_D))) + \log D(\boldsymbol{H_p}(\boldsymbol{x_c}; \boldsymbol{\theta}_D))\right], \tag{4}$$

where $D(\boldsymbol{H})$ denotes the output of discriminator D to estimate the probability that $\boldsymbol{H}$ comes from the RSE rather than the PSE, $C(\boldsymbol{H_l}(\boldsymbol{x}))$ is the final prediction of classifier C when the input is the feature representation of PSE ($\boldsymbol{H_l}(\boldsymbol{x})$), and $(\boldsymbol{x_c}, y)$ is the pair of input and output of relation classification model, where $\boldsymbol{x_c}$ is a translated Chinese instance.

RC Loss for Training on Target Language

We denote the parameters of the PSE as $\boldsymbol{\theta}_p$. The training objective is to minimize the discriminator's chance of correctly telling apart the features:

$$\mathcal{L}_p(\boldsymbol{\theta}_p) = \mathbb{E}_{\boldsymbol{x_c}\sim data}\left[\log D(\boldsymbol{H_p}(\boldsymbol{x_c}; \boldsymbol{\theta}_p))\right]. \tag{5}$$

The parameters of classifier C is denoted as $\boldsymbol{\theta}_C$. The training objective of C is to correctly classify

Relation Type	Source (EN)	Target (CH)		
	train	train	dev.	test
PHYS	1,958	1,094	170	314
PART-WHOLE	1,299	1,596	228	454
ART	869	441	63	125
ORG-AFF	2,511	1,528	218	436
PER-SOC	1,087	462	66	132
GEN-AFF	954	1,354	193	386
Other	1,446	1,081	154	308
Total	10,124	7,556	1,092	2,055

Table 1: Discription of our datasets. The data in this tabel denote the number of samples of corresponding sets. The relation type "Other" means that the relation of entity mentions is not among the aforementioned six types.

relations. The objective can be learned by minimizing the cross-entropy loss:

$$\mathcal{L}_C(\boldsymbol{\theta}_C) = \mathbb{E}_{(\boldsymbol{x_c},y)\sim data}\left[\mathcal{J}(C(\boldsymbol{H_p}(\boldsymbol{x_c};\boldsymbol{\theta}_C), y)\right]. \tag{6}$$

Finally, we combine the above objectives Eq.(5) and (6) of the relation classifiers, and minimize the joint loss:

$$\min_{\boldsymbol{\theta}_p,\boldsymbol{\theta}_C} \mathcal{L}_{rc\text{-}tar} = \lambda\mathcal{L}_p(\boldsymbol{\theta}_p) + \mathcal{L}_C(\boldsymbol{\theta}_C) \tag{7}$$

where λ is a balancing parameters calibrating the weights of the classification loss and the feature-regulating loss.

4 Experimentation

In this section, we first describe our datasets, detailed settings, and evaluation metrics used in the experiments. Then we show the effectiveness of our adversarial feature adaptation framework for cross-lingual relation classification. Finally, we further investigate the semi-supervised settings, where a small amount of labeled data of the target language exists.

4.1 Experimental Settings

In this paper we regard English as the richly-labeled (source) language and Chinese as the poorly-labeled (target) language. Note that, in fact, our model could generalize to any pair of source and target languages in principle. We conduct our experiments on the commonly used ACE 2005 multilingual training corpus (Walker et al., 2006)[2] dataset. Table 1 shows the detailed descriptions of the datasets. We utilize all of the seven types of labeled relation mentions, and evaluate all of the systems by using the micro- and macro-F1 scores over the six types of relations excluding "Other". The English and Chinese datasets are not translation of each other.

We pre-train bilingual word embeddings by CLSim[3] (Shi et al., 2015), which provides 50-dimensional embeddings for 800k parallel sentence pairs. We set the dimensions of the position embeddings and the entity type embeddings to 20 and 30, respectively, with random initialization following a continuous uniform distribution. To obtain the translated sentence pairs automatically and easily, we employ the commercial Google Translate engine[4], which is a highly engineered machine translation system. The mention boundaries and the entity type tags are provided by the ACE 2005 multilingual training corpus.

[2]https://catalog.ldc.upenn.edu/LDC2006T06

[3]http://nlp.csai.tsinghua.edu.cn/∼lzy/src/acl2015_bilingual .html

[4]https://translate.google.com

Hyper-parameter	value
Window size w	3
Convolutional filters f	100
Batch size B	100
Maximum iteration I	100
Dropout probability p	0.5

Table 2: Parameter settings.

Model (EN⇒CH)	Micro			Macro		
	P	R	F1	P	R	F1
BI-AL	62.73	64.69	63.69	64.28	65.33	64.80
CNN-MT-Source	**67.11**	66.27	66.68	**70.69**	66.82	68.70
CNN-MT-Target	65.34	66.87	66.10	70.46	66.44	68.39
BiLSTM-MT-Source	65.60	64.54	65.07	68.25	65.97	67.09
BiLSTM-MT-Target	65.52	64.92	65.22	68.96	66.88	67.90
CNN-GAN	66.79	**70.11**	**68.41**	68.73	**72.35**	**70.50**
BiLSTM-GAN	66.22	67.41	66.81	68.02	68.07	68.05

Table 3: Performance of systems on the Chinese test sets of ACE 2005 multilingual training corpus for cross-lingual relation classification. CNN-GAN: our adversarail feature adaptation approach with a CNN sentence encoder; BiLSTM-GAN: similar as CNN-GAN, with a BiLSTM sentence encoder.

All the models are optimized using ADADELTA (Zeiler, 2012). We pick the parameters showing the best performance on the development set (in Column 4, Table 1) via early stopping, and report the scores on the test set (in Column 5, Table 1). Table 2 shows the best settings of model parameters in our experiments.

We compare with the following baselines for cross-lingual relation classification.

- **CNN-MT-Source** All the instances in the English training set (the "Source" Column in Table 1) are translated into Chinese (target language) by Google Translator. A CNN model with the same structure of PSE is trained by leveraging this new translated training data on Chinese.
- **CNN-MT-Target** Contrary to the **CNN-MT-Source**, all the instances in the Chinese test set (the "Target-test" Column in Table 1) are directly translated into English (source language) by Google Translator. A CNN model with the same structure of RSE is trained by leveraging the English training set.
- **BiLSTM-MT-Source** and **BiLSTM-MT-Target** They are similar to the settings of the **CNN-MT-Source** and the **CNN-MT-Target**, respectively. For further examining the robustness of our adversarial feature adaptation framework, we replace the sentence encoder networks (CNNs) with the BiLSTM networks for both RSE and PSE. The BiLSTM model is proposed by Zhang et al. (2015), which is one of the state-of-the-art mono-lingual relation classification system. For fair comparison, we only retain the word embeddings, the position embeddings, and the entity type embeddings.
- **BI-AL** This model is proposed by Qian et al. (2014), which is a bilingual active learning system for Chinese and English relation classification with pseudo parallel corpora and entity alignment.

4.2 Experimental Results

Adversarial Feature Adaptation

Table 3 shows a performance comparison of our adversarial feature adaptation models (*-GAN) with baselines. With the CNN-GAN system, we achieve a micro-F1 score of 68.41% and a macro-F1 score

Model (CH⇒EN)	Micro			Macro		
	P	R	F1	P	R	F1
CNN-MT-Source	64.86	73.53	68.92	64.18	72.13	67.93
CNN-MT-Target	68.08	72.60	70.27	66.60	71.11	68.78
BiLSTM-MT-Source	68.30	73.24	70.68	67.06	70.79	68.88
BiLSTM-MT-Target	66.85	71.56	69.12	66.30	70.74	68.45
CNN-GAN	71.65	**77.53**	**74.47**	69.51	**73.74**	**71.56**
BiLSTM-GAN	**72.20**	75.66	73.89	**69.69**	72.03	70.84

Table 4: Performance of systems on the opposite direction for cross-lingual relation classification, in which Chinese is treated as the source language and English is treated as the target language (contrary to Table 3). The partition of English dataset is the same as that in Table 1 (i.e. the training set 70%, the development set 10%, and the test set 20%).

of 70.50%, which outperform the active learning based system (BI-AL) with a relative improvement of about 5%. It indicates that our adversarial feature adaptation framework substantially outperforms the state-of-the-art model for cross-lingual relation classification without any annotated data on the target language.

Moreover, we pay more attention to evaluate the proposed adversarial feature adaptation model against bidirectional MT-based baselines, including 1) translate the Chinses text into English, then leveraging the English relation classification model (trained on English dataset) to identify the entity type (CNN-MT-Target and BiLSTM-MT-Target), and 2) translate the English training set into Chinese, then train an Chinese relation classification model to classify the entity types (CNN-MT-Source and BiLSTM-MT-Source). Our approach improves about 2% of macro-F1 score over the machine translation based systems. Besides, the performances of CNN-MT-Source and CNN-MT-Target are comparative, which indicates that the translated direction may be insignificant for cross-lingual relation classification via MT.

We can also see that all of the CNN-based adversarial feature adaptation models achieve better results than the corresponding BiLSTM-based ones within the same settings. The reason might be that the BiLSTM is fitter for encoding order information and long-range context dependency for sequence labeling problem, while the CNN is suited to extracting local and position-invariant features. For relation classification, the essential features are always distributed around the entity mentions, which would be better utilized by CNN[5].

To validate the language independence of our feature adaptation framework, we also implement our adversarial feature adaptation systems and the MT-based systems in Table 1 on the same corpus from the opposite direction. Specifically, we regard Chinese as the source language and English as the target, then learn the feature representations from the Chinese dataset to predict the relation labels of the English dataset. Table 4 indicates that our approach also outperforms the baselines on learning the feature representations from Chinese to English. Besides, the results further validate the conclusions mentioned above: 1) Our system outperforms the MT-based systems for cross-lingual relation classification, and 2) the CNN-based systems achieve better performances than the BiLSTM-based systems in the same settings.

To further provide an empirical insight into the relationship between 1) the data size of labeled training set for supervised relation classification (the blue curve in Figure 2), and 2) the data size required by our adversarial feature adaptation system with only unlabeled sentences (the orange dashed curve in Figure 2), we simulate a supervised scenario by adding labeled Chinese instances for training a CNN-based relation classification system (CNN-CH in Table 5). We start from adding 100 labeled sentences and keep adding 100 sentences each time until 900. As shown in Figure 2, when adding the same number of labeled sentences, the CNN-CH system can better utilize the extra supervision. The margin is naturally decreasing as more supervision is incorporated, until the training set contains more than 700 instances.

[5]Yin et al. (2017) also demonstrated this conclusion for relation classification task.

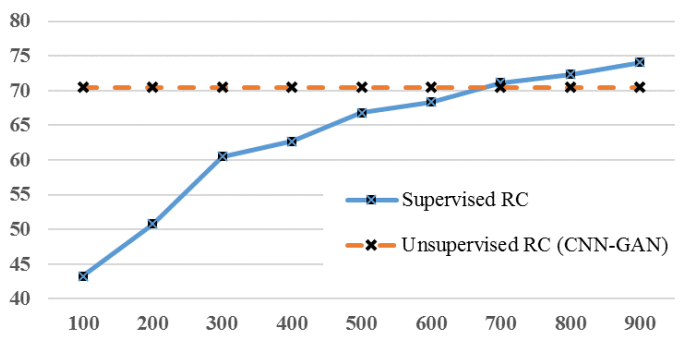


Figure 2: Comparison with a supervised relation classification system.

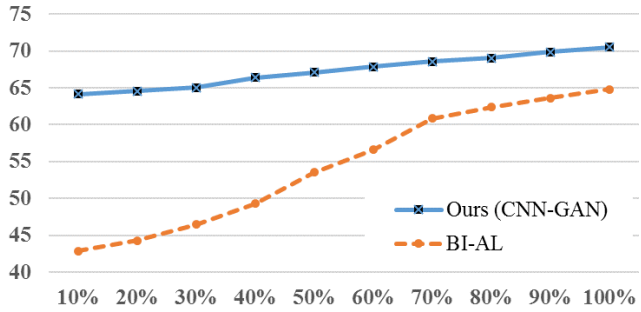


Figure 3: Comparison of the performances (macro-F1) when adding different sizes if labeled data of the source language from 10% to 100%.

It indicates that our adversarial feature adaptation system can achieve comparable performance to the supervised system trained on a small labeled dataset.

Another interesting find is that it seems a very small amount of supervision (e.g., 500 labeled instances) could significantly help the supervised relation classification system. However, it is worth noting that the manual annotation on such amount of dataset is still time-consuming and human-intensive, since the people should annotate not only the entity mentions and their relation, but also the external information such as the lexical or syntactic features if necessary.

Figure 3 compares the performances of our CNN-GAN system (the blue curve in Figure 3) and a bilingual active learning system BI-AL (Qian et al., 2014) (the orange dashed curve in Figure 3) when training with different sizes of labeled data from the source language. As we can see, the margin of our approach is not significant when the size of the source-language instances is relatively small. When using 10% of the training data, our system only declines 6.35% of performance (from 70.50% to 64.15%), while for the bilingual active learning system, the gap widens to 20.91% (from 64.80% to 42.89%). It indicates that our feature adaptation approach can efficiently utilize the translated supervision from the source language.

Semi-supervised Scenario

In this paper, we mainly focus on the the scenario of unsupervised cross-lingual relation classification, i.e., without any labeled dataset. However, for broader comparisons, we also test our framework when a few labeled training instances of the target language are available. Actually, our approach can be easily generalized to a semi-supervised setting. We employ a simple way that directly combine the softmax layers of the CNN-CH system and the CNN-GAN system, to integrate the supervision from target language into relation classification.

Table 5 lists the performances of these semi-supervised relation classification systems. We see that our ensemble model (*-CH-EN) which can employ not only the labeled data from the target language but from the source language, slightly improves over the supervised model (*-CH). It indicates that our

Model	P	R	F1
bilingual-Joint-IRL	**80.9**	77.1	78.9
CNN-CH	79.09	81.15	80.11
BiLSTM-CH	76.23	79.94	78.04
CNN-CH-EN	79.61	**81.65**	**80.62**
BiLSTM-CH-EN	77.55	79.50	78.52

Table 5: Performance of the semi-supervised relation classification systems. bilingual-Joint-IRL system: a bilingual approach by joint supervision of classification loss and ideal representation loss (Min et al., 2017); *-CH systems: A CNN/BiLSTM with the same architecture of our CNN/BiLSTM feature extractor (trained on 7,556 instances of the CH training set); *-CH-EN systems: a simple ensemble way that directly combine the softmax layers of the CNN-CH system (trained on 3,778 instances of the CH training set) and CNN-GAN system (trained on 3,778 instances of the EN training set).

adversarial model can transfer some useful knowledge and information from the source language to the target language for relation classification, by which both the language-specific and language-invariant features could be learned.

Besides, better ensemble methods could be attempted to exploit the information across language for semi-supervised relation classification. In addition, we see that our model has obtained improvement over the previous best-performing system (bilingual-Joint-IRL in Table 5) of semi-supervised relation classification.

5 Conclusion

In this paper we introduce an adversarial feature adaptation approach for cross-lingual relation classification without labeled dataset, which leverages the data on richly-labeled language to help relation classification on the poorly-labeled language. We evaluate our approach on ACE 2005 multilingual training corpus. Experimental results show that this approach can effectively transfer feature representations from a richly-labeled language to another poorly-labeled language, and outperforms several baselines including active learning models and highly competitive MT-based baselines. The code is available at `https://github.com/zoubowei/feature_adaptation4RC`.

Theoretically speaking, our adversarial feature adaptation approach can be flexibly implemented in the scenario of multiple languages, while this paper focuses on two languages of English and Chinese. Thus in future, we will extend this approach to more languages and explore its significance.

Acknowledgements

This research is supported by the National Natural Science Foundation of China (No. 61703293, No. 61751206, and No. 61672368). We would like to thank the anonymous reviewers for their insightful comments and suggestions.

References

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

Razvan C. Bunescu and Raymond J. Mooney. 2005. Subsequence kernels for relation extraction. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS'05)*, pages 171–178.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614*.

Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational*

Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL'15), pages 626–634.

Manaal Faruqui and Shankar Kumar. 2015. Multilingual open relation extraction using cross-lingual projection. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'15)*, pages 1351–1356.

Kata Gábor, Davide Buscaldi, Anne-Kathrin Schumann, Behrang QasemiZadeh, Haïfa Zargayouna, Thierry Charnois. 2018. SemEval-2018 Task 7: Semantic relation extraction and classification in scientific papers. In *Proceedings of International Workshop on Semantic Evaluation (SemEval'18)*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, Mario Marchand, and Victor Lempitsky. 2017. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1):2096–2030.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of Advances in Neural Information Processing Systems Conference (NIPS'14)*, pages 2672–2680.

Iris Hendrickx, Nam Kim Su, Zornitsa Kozareva, Preslav Nakov, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. SemEval-2010 task 8: multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics on Interactive poster and demonstration sessions*.

Seokhwan Kim, Minwoo Jeong, Jonghoon Lee, and Gary Geunbae Lee. 2014. Cross-lingual annotation projection for weakly-supervised relation extraction. *Acm Transactions on Asian Language Information Processing (TALIP)*, 13(1):3.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. 2016. Adversarial autoencoders. In *Proceedings of the International Confer- ence on Learning Representations (ICLR'16)*.

Bonan Min, Zhuolin Jiang, Marjorie Freedman, and Ralph Weischedel. 2017. Learning transferable representation for bilingual relation extraction via convolutional neural networks. In *Proceedings of the The 8th International Joint Conference on Natural Language Processing (IJCNLP'17)*, pages 674–684.

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

Longhua Qian, Haotian Hui, YaNan Hu, Guodong Zhou, and Qiaoming Zhu. 2014. Bilingual active learning for relation classification via pseudo parallel corpora. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL'14)*, pages 582–592.

Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING'08)*, pages 697–704.

Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric P. Xing. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL'17)*, pages 1006–1017.

Motoki Sato, Hitoshi Manabe, Hiroshi Noji, and Yuji Matsumoto. 2017. Adversarial training for cross-domain universal dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies (CoNLL'17)*, pages 71–79.

Tianze Shi, Zhiyuan Liu, Yang Liu, and Maosong Sun. 2015. Learning cross-lingual word embeddings via matrix co-factorization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL'15, Short Papers)*, pages 567–572.

Fabian M. Suchanek, Georgiana Ifrim, and Gerhard Weikum. 2006. Combining linguistic and statistical analysis to extract relations from web documents. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'06)*, pages 712–717.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'16)*, pages 886–896.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. *ACE 2005 multilingual training corpus*. Linguistic Data Consortium.

Linlin Wang, Zhu Cao, Gerard De Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention CNNs. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*, pages 1298–1307.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schtze. 2017. Comparative study of CNN and RNN for natural language processing. *arXiv preprint arXiv:1702.01923*.

Matthew D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*, pages 1753–1762.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers (COLING'14)*, pages 2335–2344.

Shu Zhang, Dequan Zheng, Xinchen Hu, and Ming Yang. 2015. Bidirectional long short-term memory networks for relation classification. In *Proceedings of 29th Pacific Asia Conference on Language, Information and Computation*, pages 73–78.

One-shot Learning for Question-Answering in Gaokao History Challenge

Zhuosheng Zhang[1,2], **Hai Zhao**[1,2,*]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
zhangzs@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn

Abstract

Answering questions from university admission exams (Gaokao in Chinese) is a challenging AI task since it requires effective representation to capture complicated semantic relations between questions and answers. In this work, we propose a hybrid neural model for deep question-answering task from history examinations. Our model employs a cooperative gated neural network to retrieve answers with the assistance of extra labels given by a neural turing machine labeler. Empirical study shows that the labeler works well with only a small training dataset and the gated mechanism is good at fetching the semantic representation of lengthy answers. Experiments on question answering demonstrate the proposed model obtains substantial performance gains over various neural model baselines in terms of multiple evaluation metrics.

1 Introduction

Teaching a machine to pass admission exams is a hot AI challenge which has aroused a growing number of research (Guo et al., 2017; Cheng et al., 2016; Clark, 2015; Fujita et al., 2014; Henaff et al., 2016). Among these studies, deep question-answering (QA) task (Ferrucci et al., 2010; Wang et al., 2014) is especially difficult, aiming to answer complex questions via deep feature learning from multi-source datasets. Recently, a highly challenging deep QA task is from the Chinese University Admission Examination (*Gaokao* in Chinese), which is a national-wide standard examination for all senior middle school students in China and has been known as its large scale and strictness. This work focuses on *comprehensive question-answering* in Gaokao history exams as shown in Table 1[1], which accounts for the major proportion in total scoring and is extremely difficult in the exams. (Cheng et al., 2016) made a preliminary attempt to take up the Gaokao challenge, trying to solve *multiple-choice questions* via retrieving and ranking evidence from *Wikipedia* articles to determine the right choices. Differently, this task is to solve comprehensive questions and has to be based on knowledge representation and semantic computation rather than word form matching in the previous work.

Although deep learning methods shine at various natural language processing tasks (Wang et al., 2015; Zhang et al., 2016b; Qin et al., 2017; Cai et al., 2018; Zhang et al., 2018b; Huang et al., 2018; Zhang et al., 2018a; He et al., 2018; Zhang et al., 2018c; Li et al., 2018), they usually rely on a large scale of dataset for effective learning. The concerned task, unfortunately, cannot receive sufficient training data under ordinary circumstances. Different from previous typical QA tasks such as community QA (Zhang et al., 2016a) which can enjoy the advantage of holding a very large known QA pair set, the concerned task is equal to retrieving a proper answer from textbooks organized as plain texts with guidelines of very limited number of known QA pairs. In addition, the questions are usually given in a quite indirect way to ask students to dig the exactly expected perspective of the concerned facts. If such kind of perspective

*Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100), National Natural Science Foundation of China (No. 61672343 and No. 61733011), Key Project of National Society Science Foundation of China (No. 15-ZDA041), The Art and Science Interdisciplinary Funds of Shanghai Jiao Tong University (No. 14JCRZ04).

[1]These examples are chosen as they are short, the average number of words in questions and answers are 22 and 47.

Proceedings of the 27th International Conference on Computational Linguistics, pages 449–461
Santa Fe, New Mexico, USA, August 20-26, 2018.

fails to fall into the feature representation for either question or answer, the retrieval will hardly be successful.

Q: “农民可能充当一种极端保守的角色，也可能充当一种具有高度革命性的角色。”试结合有关史实评析这一观点。 “Peasants may act as an extremely conservative role, or may be highly revolutionary.” Try to analyze this view in the light of the relevant historical facts. **A**: 农民阶级的经济地位和所处的时代条件决定了它可能充当一种具有高度革命性的角色。以太平天国运动为例，在斗争过程中颁布的《天朝田亩制度》就突出反映了农民阶级要求废除封建土地所有制的强烈愿望，表现出了高度的革命性。农民阶级存在的这种两面性是由其经济地位即受地主阶级压迫和小生产者的地位所决定的。 The economic and social conditions of peasants determine that they may act as a highly revolutionary role. Taking the Taiping Heavenly Kingdom movement as an example, the promulgated regulation “heavenly land system” reflected the peasant class’s demands and strong desire of abolishing the system of feudal land ownership, which shows a strong degree of revolution. The dual nature of the peasant class is also the result of its economic status, that is, under the oppression of the landlord class and the status of the small producers.
Q: 有人说“文艺复兴”是一场复古运动，你如何看待？如何评价其意义？ Some people think the Renaissance is a retro campaign. What’s your opinion and how to evaluate its historic significance? **A**: 文艺复兴运动从表面的含义来看，是一种复兴古希腊罗马时期的哲学、文学和艺术的活动，是一种复古运动，但从其深层的含义看，它却是一场思想解放运动，是一次思想领域里的变革。 The Renaissance is seemingly regarded as a retro movement of philosophy, literature and art of ancient Greece and Rome. However, it is actually an ideological liberation movement and a revolution of thoughts from a deeper inspection.

Table 1: Comprehensive question examples from Gaokao history exams.

Generally speaking, for the Gaokao challenge, knowledge sources are extensive and no sufficient structured dataset is available, while most existing work on knowledge representation focused on structured and semi-structured types (Khot et al., 2017; Khashabi et al., 2016; Vieira, 2016). With regard to the answer retrieval, most current research focused on the factoid QA (Dai et al., 2016; Yin et al., 2016), community QA (Zhang et al., 2017; Lu and Kong, 2017) and episodic QA (Samothrakis et al., 2017; Xiong et al., 2016; Vinyals et al., 2016). Compared with these existing studies, the concerned task is more compositive and more comprehensive and has to be done from unstructured knowledge sources like textbooks. Moreover, the answers of our Gaokao challenge QA task are always a group of detail-riddled sentences with various lengths rather than merely short sentences as focused on in previous work (Yin et al., 2016; Yih et al., 2014).

Recent research has turned to semi-supervised methods to leverage unlabeled texts to enhance the performance of QA tasks via deep neural networks (Yang et al., 2017; Lei et al., 2016). This task is somewhat different from previous ones that the expected extra labels are difficult to be annotated and the entire unlabeled data is kept in a very small scale, so that semi-supervised methods cannot be conveniently applied. Notably, one-shot learning has been proved effective for image recognition with few samples (Li et al., 2006), which is a strategy similar to people learning concept. As an implementation of one-shot learning, neural turing machine (NTM) was proposed (Santoro et al., 2016; Vinyals et al., 2016) and showed great potential by learning effective features from a small amount of data, which caters to our mission requirements. Inspired by the latest advance of one-shot learning, we train a weakly supervised classifier to annotate salient labels for questions using a small amount of examples. For question answering, we propose a cooperative gated neural network (CGNN) to learn the semantic representation and corresponding relations between questions and answers. We also release a Chinese comprehensive deep question answering dataset to facilitate the research.

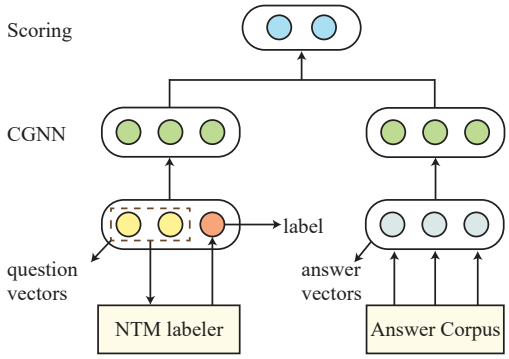


Figure 2: Model architecture.

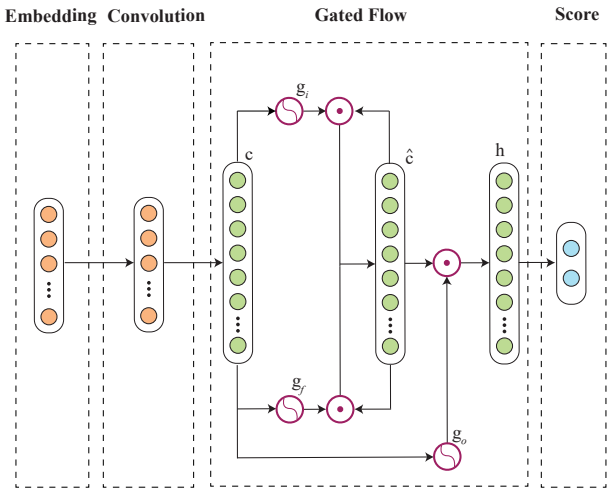


Figure 3: CGNN architecture.

The rest of this paper is organized as follows. The next section introduce our models, including retrieval model, CGNN, and NTM labeler. Task details and experimental results are reported in Section 3, followed by reviews of related work in Section 4 and conclusion in Section 5.

2 Model

The proposed hybrid neural model is composed of two main parts: cooperative gated neural network (CGNN) for feature representation and answer retrieval, and NTM as one-shot learning module for extra labeling. As shown in Figure 2, we use NTM to classify the question type and annotate the corresponding label. Then, the concatenated label vectors and question representation are fed to CGNN for jointly scoring candidate answers.

2.1 Main framework

For the QA task, the key problem is how to effectively capture the semantic connections between a question and the corresponding answer. However, the questions and answers are lengthy noisy, resulting in poor feature extraction. Inspired by the success of the popular gated mechanism (Wang et al., 2017a; Chen et al., 2016a; Lei et al., 2016) and the gradient-easing Highway Network (Srivastava et al., 2015), a CGNN is proposed to score the correspondence of inputted QA pairs. The architecture is shown in Figure 3.

Embedding Our model reads each word of questions and answers as the input. For an input sequence, the embedding is represented as $\mathbf{M} \in \mathbb{R}^{d \times n}$ where d is dimension of word vector and n is the maximum length. When using the NTM module for question type labeling, the question embedding will be refined as the concatenation of the label vectors and its original embedding. Considering the calculation efficiency, we specialize a max number of words for the input and apply truncating or zero-padding when needed.

Convolutional Layer Filter matrices $[\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_k]$ with several variable sizes $[l_1, l_2, \ldots, l_k]$ are utilized to perform the convolution operations for input embeddings. Via parameter sharing, this feature extraction procedure is kept the same for questions and answers. For the sake of simplicity, we will explain the procedure for only one embedding sequence. The embedding will be transformed to sequences $\mathbf{c}_j (j \in [1, k])$:

$$\mathbf{c}_j = [\ldots; \tanh(\mathbf{W}_j \cdot \mathbf{M}_{[i:i+l_j-1]} + \mathbf{b}_j); \ldots]$$

where $[i : i+l_j-1]$ indexes the convolution window. Additionally, we apply wide convolution operation between embedding layer and filter matrices, because it ensures that all weights in the filters reach the entire sentence, including the words at the margins.

Gated Flow To highlight the key information and ignore irrelevant ones during convolution, we adopt an adaptive gated decay mechanism for gated information flow. Three gates are added to optimize the feature representation. These gates are only influenced by the original input through different parameters. Let $\hat{\mathbf{c}}_j^n$ denote n-gram features in c_j, the gates are formulated as

$$\mathbf{g}_i = \sigma(\mathbf{W}_i\hat{\mathbf{c}}_j^n + \mathbf{b}_i)$$
$$\mathbf{g}_f = \sigma(\mathbf{W}_f\hat{\mathbf{c}}_j^n + \mathbf{b}_f)$$
$$\mathbf{g}_o = \sigma(\mathbf{W}_o\hat{\mathbf{c}}_j^n + \mathbf{b}_o)$$

where σ denotes sigmoid function which guarantees the values in the gates are in [0,1] and $\mathbf{W}_i$, $\mathbf{W}_f$, $\mathbf{W}_o$, $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_o$ are model parameters. The transformed inner cell is iteratively calculated by:

$$\hat{\mathbf{c}}_j^n = \hat{\mathbf{c}}_{j-1}^n \odot \mathbf{g}_i + \mathbf{g}_f \odot (\hat{\mathbf{c}}_{j-1}^{n-1} + \mathbf{W}_n)$$

where $\odot$ denotes element-wise multiplication and $\mathbf{W}^o$ is the parameter. Through gated flow, the vector $\mathbf{c}_t$ captures the filtered information. The output of our CGNN is $\mathbf{h}_j = \tanh(\hat{\mathbf{c}}_j^n) \odot \mathbf{g}_o$. The gates are used to route information through the flow. Although these gates are generated independently, they work cooperatively since they jointly control the information flow of inner-cells. This procedure will help select salient features. When there is one gate in our network, the model works similarly to the original highway network (Srivastava et al., 2015).

A *one-max-pooling* operation is adopted after the gated flow and the current state vector $\mathbf{s}_j$ is obtained through concatenating all the mappings for those k filters, we have $\mathbf{s}_j = \mathbf{max}(\mathbf{h}_j)$.

For discriminative training, we use a max-margin framework for learning (or fine-tuning) parameters θ. Specifically, a scoring function $\varphi(\cdot, \cdot; \theta)$ is defined to measure the similarity between the corresponding representations of questions and answers. In this work, we use cosine similarity for the measurement. Let $p = \{p_1, ...p_n\}$ denote the answer corpus and $a \in p$ is the answer to the question q_i, the optimal parameters θ are learned by minimizing the max-margin loss:

$$\max\{\varphi(q_i, p_i; \theta) - \varphi(q_i, a; \theta) + \delta(p_i, a)\}$$

where $\delta(.,.)$ denotes a non-negative margin and $\delta(p_i, a)$ is a small constant when $a \neq p_i$ and 0 otherwise.

2.2 Question Type Labeling

Since examinees studying for exams are required to understand the history knowledge from different sides, we exactly consider using especially-annotated labels as extra indicators for machines to capture such features. With external memory, NTM has shown great potential for one-shot learning (Santoro et al., 2016). As in Figure 3, an NTM consists of two primary components, controller and memory. The controller, often implemented as a recurrent neural network, interacts with an external memory module using soft *read heads* to retrieve representation from memory and *write heads* to store memory, respectively.

Controller The controller of NTMs can be implemented as either a recurrent neural network or a feed-forward neural network. Our model adopts LSTM cells as the implementation for the best performance (Graves et al., 2014).

$$\mathbf{i_t} = \sigma(\mathbf{W}_w^i\mathbf{w}_t + \mathbf{W}_h^i\mathbf{h}_{t-1} + \mathbf{b}_i)$$
$$\mathbf{f_t} = \sigma(\mathbf{W}_w^f\mathbf{w}_t + \mathbf{W}_h^f\mathbf{h}_{t-1} + \mathbf{b}_f)$$
$$\mathbf{u_t} = \sigma(\mathbf{W}_w^u\mathbf{w}_t + \mathbf{W}_h^u\mathbf{h}_{t-1} + \mathbf{b}_u)$$
$$\mathbf{c_t} = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_w^c\mathbf{w}_t + \mathbf{W}_h^c\mathbf{h}_{t-1} + \mathbf{b}_c)$$
$$\mathbf{h_t} = \tanh(\mathbf{c}_t) \odot \mathbf{u}_t$$
$$\mathbf{o_t} = (\mathbf{h}_t \oplus \mathbf{m}_t)$$

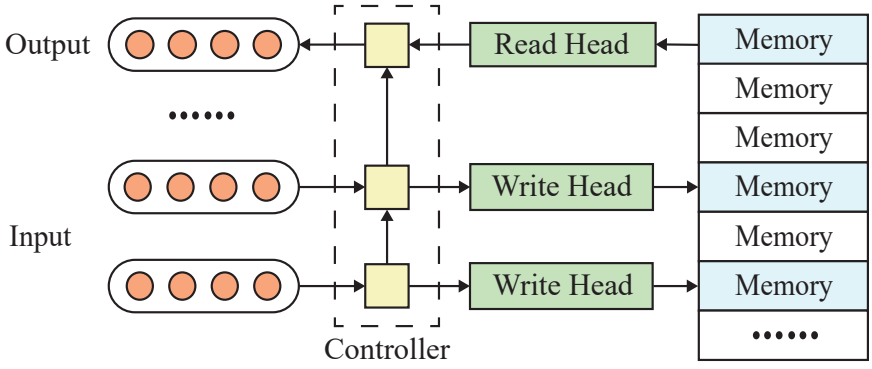


Figure 3: NTM architecture

where $\oplus$ denotes vector concatenation, and $\mathbf{i_t}, \mathbf{f_t}, \mathbf{u_t}, \mathbf{c_t}, \mathbf{h_t}$ are the input gates, forget gates, output gates, cell state and hidden state, respectively.

Given the input sequence, the controller computes the concatenated state sequence $\mathbf{o}_t$ by applying the formulation for each time step. Each word of the sequence is represented as word embedding. In order to simplify the calculation, we apply *one-max-pooling* to each input. As a result, each sequence is represented as a vector with the same dimension of word embedding.

Memory Manipulation We use a rectangular matrix $M \in \mathbb{R}^{N \times M}$ to denote the memory module. Given the read vector $\mathbf{r}_t$, the memory $\mathbf{m}_t$ is retrieved by:

$$\mathbf{m}_t = \mathbf{r}_t \mathbf{M}_t, \mathbf{r}_t = \frac{\exp(\mathbf{K}(\mathbf{k}_t, \mathbf{M}_t))}{\sum_j \exp(\mathbf{K}(\mathbf{k}_t, \mathbf{M}_t)}$$

where $\mathbf{r}_t$ is read vector and $\mathbf{K}$ is the similarity score. For writing, the memory is updated by

$$\mathbf{M}_t = \mathbf{M}_{t-1} \cdot (1 - \mathbf{w}_t \mathbf{e}_t) + \mathbf{w}_t \mathbf{c}_t$$

where $\mathbf{w}_t$, $\mathbf{e}_t$ and $\mathbf{c}_t$ represent the write vectors, erase vectors and content vectors respectively.

There are two categories for memory addressing mechanism of the original NTM: content-based addressing comparing each memory locations by some key $\mathbf{k}_t$ and a location-based addressing by shifting the heads, akin to running along a tape. However, the location-based addressing is not optimal for conjunctive coding of information independent of sequence. Following (Santoro et al., 2016), we use an addressing strategy called the Least Recently Used Access (LRUA).

Addressing The weight $\mathbf{u}_t$ is defined as follows,

$$\mathbf{u}_t = \gamma \mathbf{u}_{t-1} + \mathbf{r}_t + \mathbf{w}_t$$

where γ is a decay parameter. The least-used vectors, $\mathbf{v}_t$, are boolean variables. For a given time-step, each element of $\mathbf{v}_t$ is set to 0 if it is greater than the smallest element of $\mathbf{u}_t$, otherwise is 1. The write weights are obtained by

$$\mathbf{w}_t = \sigma(g_t)\mathbf{r}_{t-1} + (1 - \sigma(g_t))\mathbf{v}_{t-1}$$

where g_t is a scalar interpolation in the range (0,1) that blends the weights of previous read weights and previous least-used weights.

Learning For our labeling task, we define a cost function as the negative log-likelihood:

$$L(\theta) = -\sum_{n=1}^{N} \mathbf{y}_n log(Pr(\hat{\mathbf{y}}_n))$$

where $\mathbf{y}_n$ is the label and $\hat{\mathbf{y}}_n$ is the predicted one. After perspective detection, the label is then fed to the CGNN module as one-hot vectors and concatenated with the question representation.

	Train	Test
# QA pairs	964	965
Avg # words in questions	18	25
Avg # words in answers	46	48
Max # words in questions	478	482
Max # words in answers	4,652	4,387
# Candidate answer set	1,929	
# Vocabulary	10,806	

Table 2: Data statistics of the comprehensive question-answering corpus.

.

3 Experiments

Corpus Gaokao challenge is an imitation of open-book examination for computers. Therefore, only quite a limited number of resources can be used, which makes common semi-supervised methods unlikely beneficial from large scale unlabeled data. In addition, all our source for the answer should come from standard textbook which contains unstructured plain texts as we initialized our system building.

The corpus is from the standard history textbook[2]. First, to compose an answer set, 1,929 text fragments[3] were extracted by human experts. Then equal numbers of questions were collected from past Gaokao exam papers, and their answers were manually assigned from the answer set to give 1,929 annotated QA pairs, which were equally split for training and test. All text fragments in either questions or answers are segmented into words using *BaseSeg* (Zhao et al., 2006). We publish it to research communities to facilitate the research [4]. Data statistics are in Table 2.

Embedding	Max number of words	$n = 100$
	Word embedding size	$d = 200$
CGNN	Hidden unit number	$h = 200$
	Initial learning rate	$lr = 10^{-3}$
	Regularization	$\nabla = 10^{-5}$
	Dropout	$p = 0.2$
	Filter Width	$k = 4$
NTM	Controller size	$s = 200$
	Read heads	$r = 4$
	Memory shape	$m = (128, 100)$
	Initial learning rate	$lr = 10^{-3}$

Table 3: Hyper-parameters of our model.

[2]Standard Middle-school History Textbook (Vol. 1-3), published by the People's Education Press in May, 2015.
[3]1,929 is the number of all must-to-be-mastered history facts required by national education quality control.
[4]Our source is available at: `https://github.com/cooelf/OneshotQA`.

Class Labels	Answers
背景 Background	运用我们从古代诗文、戏曲、民间传说中已经学到的知识，举例说明中国古代自给自足的自然经济的状况。 Using the knowledge learned from ancient poetic proses, operas, and folklores, exemplify the situation of self-sufficiency of natural economy in ancient China.
原因 Cause	春秋战国时期是社会剧烈动荡的历史阶段，为什么在这样的时期会出现思想文化活跃的局面。 The Spring-autumn and Warring States Period was the historical stage of the violent social unrest. Why would there be an active ideological and cultural phenomenon in such a period?
主张 Claim	在启蒙运动中，众多的启蒙思想家的共性思想主张是什么？他们之间有何继承和发展。 During the Enlightenment, what is the common thought of those enlightening thinkers? What is the inheritance and development between them?
事实 Fact	“农民可能充当一种极端保守的角色，也可能充当一种具有高度革命性的角色。”试结合有关史实评析这一观点。 “Farmers may act as an extremely conservative role, or may be highly revolutionary.” Try to analyze this view in the light of the relevant historical facts.
意义 Influence	用历史唯物主义和辩证唯物主义的观点来分析古代雅典民主政治和罗马法发展的历程，了解它们对后世的作用和影响。 From the perspective of historical and dialectical materialism, analyze the history of Athens's democratic politics and the development of Rome law in ancient times, and understand their roles and influences on later generations.

Table 4: Examples of Text Fragments of 5 classes.

Setup For the sake of computational efficiency, a maximal length of 100 words for each text fragment is specialized and truncating or zero-padding will be applied as necessary. Word embedding is trained by word2Vector (Mikolov et al., 2013) using *Wikipedia* corpus[5].

The diagonal variant of AdaGrad (Duchi et al., 2011) is used for neural network training. Table 3 shows the hyper-parameters of our models. For other neural models, we fine-tune the hyper-parameters with the following range of values: learning rate $\in \{1e-3, 1e-2\}$, dropout probability $\in \{0.1, 0.2\}$, CNN filter width $\in \{2, 3, 4\}$. The hidden dimensions for all the neural networks are 200.

In the following experiments, we use NTM to annotate salient labels for each question. Then, the question representation and corresponding label vectors are concatenated and fed to CGNN for feature learning.

Labeling Examinees are required to understand the history knowledge from different sides. Thus, the solver should also be capable of capturing such kind of features. To specify a key perspective on historical questions, five classes[6] (*background, cause, claim, fact and influence*) are annotated as shown in Table 4. Note that such kind of annotation is even not easy for human annotators or history teachers and our complete set for all answers is very small (less than 2,000). At last, we have 80 answers per class annotated for a total number of 400, in which 50 are selected for training[7] and 350 for test. For our detailed case, our data is too precious to be used too many for training.

One-shot learning is supposed to effectively represent the questions in the feature space with a small set of training data. To evaluate its performance, we specify a fixed number of training data. At each epoch, NTM is randomly provided with a specific number (shot) of samples per class from training set according to the typical training setting of one-shot learning (Santoro et al., 2016). One-shot learning, in previous practice, usually still took a large candidate training set (i.e., hundreds) as input, though for

[5] https://dumps.wikimedia.org/

[6] These classes are determined according to national education quality control who requires every student should clearly distinguish these specific perspectives about history facts. More details are in Supplementary Materials.

[7] As our labeling is a five-class classification task, 50 samples therefore allow 10-shot learning per class at most.

Method	1-shot	2-shot	5-shot	10-shot
Naive Bayes	22.1	25.7	30.1	34.1
K-Means	12.5	15.7	15.8	16.1
SVM	19.1	21.8	30.2	**45.2**
Feedforward	20.1	22.7	21.3	21.6
LSTM	23.8	28.5	32.5	38.2
NTMs	**38.7**	**49.1**	**63.9**	**76.6**

Table 5: Accuracy for labeling task.

each class, only a small fixed number of shots (i.e., less than 10) were randomly selected and fed for learning. However, it is quite different from our data reality and we have to keep a least training set (10 for our case) for one-shot learning.

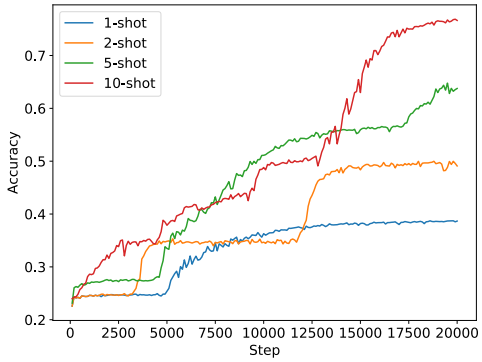


Figure 4: Labeling accuracy curve.

Method	Precision	Recall	F1-score
BM25	8.6 (8.7)	23.2 (23.7)	12.5 (12.7)
CNN	9.8 (27.0)	21.1 (45.0)	13.4 (33.8)
GRU	11.4 (32.9)	24.1 (50.4)	15.5 (39.8)
LSTM	8.4 (30.6)	21.2 (48.0)	12.0 (37.4)
+	**22.2**	**26.8**	**25.4**
CNN+Highway	15.8 (33.2)	30.5 (50.6)	20.8 (40.1)
CNN+GRU	17.4 (32.4)	31.1 (50.8)	22.3 (39.6)
CNN+LSTM	12.7 (27.5)	21.4 (47.0)	16.0 (34.7)
CGNN	**18.2 (33.7)**	**31.6 (51.6)**	**23.1 (40.8)**
+	15.5	20.0	17.7

Table 6: Answer retrieval performance without and with NTM labels (in brackets). "+" means the performance gains with the help of NTM labels.

Table 5 compares NTM with traditional classifiers (Naive Bayes, K-Means and SVM) and neural networks (Feedforward network and LSTM). All the baseline methods are fed the training instances as the same shot style of NTM and parameters are tuned for best performance.

Figure 4 illustrates NTM learning curves with different shots. The results show that NTM works effectively on a small amount of samples. With memory cells as well, LSTM is chosen as the baseline. The hidden layer dimension of the LSTM is 200, which is the same as the NTM controller size. The LSTMs are fed with the training instances as the same shot style of NTM. Results shown in Table 5 indicate only NTM provides satisfactory performance with the least input instances. The effectiveness of NTM may attribute to its capability of slowly learning the representations of raw data through weight updates, and rapidly binding relevant information after a single representation via an external memory. To that extend, the NTM could generalize the meta features of each question type and distinguish the perspectives. This indicates one-shot learning is able to lessen over-fitting issues from sparse features of limited data.

Answer Retrieval For a discriminative training, we add negative QA pairs by randomly selecting 20 false answers for each positive QA pairs in the original training set. For evaluation, all answers in the corpus are ranked to match each question. NTM labeler is from 10-shot training.

We use baselines including BM25 implemented by a standard search engine *Apache Lucene*[8] and other neural models, Convolutional Neural Network (CNN), LSTM, Gated Recurrent Unit (GRU). Our evaluation is based on the following metrics: Precision, Recall and F1-score, which are widely used for relevance evaluation.

Table 6 presents the experimental results of all the models with and without NTM labels. All of the neural networks greatly outperform BM25 which is purely based on word form matching. In addition, NTM labeler boosts the performance of all the methods. For instance, our CGNN model has obtained 17.7% gains on the F1-score metric with the assistance of extra labeling. Furthermore, our model using CGNN and NTM labeler outperforms all the other baselines. This superior performance indicates that the one-shot learning strategy is competent for our deep QA task with only a small amount of data and

[8] http://lucene.apache.org/

Type	Proportion	Precision	Recall	F1-score
Background	16.0	38.0	56.3	45.4
Cause	8.4	48.6	67.5	56.5
Claim	4.7	47.8	65.1	55.1
Fact	51.0	23.0	42.0	29.7
Influence	19.9	39.0	54.5	45.5

Table 7: Question type distribution and model performance of CGNN.

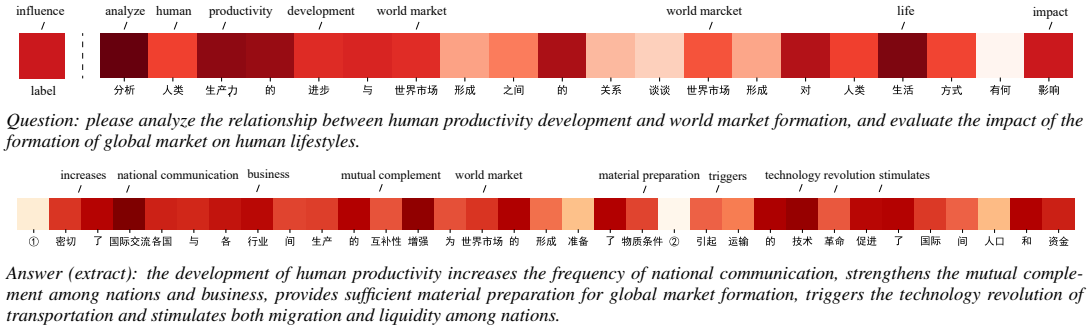


Figure 5: Visualization for question and answer representation weights after gated flow. The darker color means the higher weights.

the adoption of gated mechanism is also well-suited to work with CNN, adaptively transforming and combining local features detected by the individual filters. Table 7 shows question type distribution and the performance of CGNN. Our model performs well on most types but drops on *Fact* since these questions are diverse kinds of facts whose patterns are extremely hard to distinguish, even for human. This also shows our task is challenging.

Model Analysis The primary motivation of highway network is to ease gradient-based training of highly deep networks through utilizing gated units. When remaining one gate in our network, our model works similarly to the highway network. To compare different gated mechanisms, we carry out further experiments with the same CNN setting. As shown in Table 6, CGNN achieves the best performance in terms of all evaluation metrics, indicating our gated mechanism is well-suited to work with CNN, adaptively transforming and combining local features detected by the individual filters.

In order to give an insight into the effectiveness of the gated mechanism on information flow across neural network layers, we visualize the final weights for each word from question and answer after gated flow as shown in Figure 5. We can see that the key information (*human, productivity, development, world market, life and influence*) of the lengthy questions and answers will be given higher weights, especially the common words (*world market*) of the question and corresponding answer. Besides, the label in the question is also assigned a high attention, indicating the label works essentially.

4 Related Work

4.1 Question Answering

Various neural models have been proposed to tackle the tasks of QA task and related knowledge representation (Wang et al., 2017b; Chen et al., 2016b; Todor and Anette, 2018; Kundu and Ng, 2018). Previous work mainly focused on factoid questions, falling into the architecture of knowledge base embedding and question answering from knowledge base (Khashabi et al., 2016; Angeli et al., 2015). Yang et al. (2014) proposed a method that transforms natural questions into their corresponding logical forms and conducted question answering by leveraging semantic associations between lexical representations and knowledge base properties in the latent space. Yin et al. (2016) proposed an end-to-end neural model that can generate answers to simple factoid questions from a knowledge base. Nonetheless, the knowledge base construction requires a lot of human workload, and the related semantic parsing and knowledge representation will become much more complicated as the scale of the knowledge base increases.

For non-factoid QA, most work formulized it as a semantic matching task on sentence pairs by vector modeling. Different from answer retrieval from knowledge bases, this type of studies used vector to represent QA pairs and compare the distance in vector space to match answer text (Tan et al., 2015; Feng et al., 2015). However, the performance of these neural models greatly depends on a large amount of labeled data.

Our concerned task cannot be simply regarded as either factoid or non-factoid. In fact, questions in our task consist of both factoid and non-factoid ones with purely unstructured corresponding answers. Conventional networks might not be sufficient to represent these QA pairs and we need to seek more powerful models. A recent hot comprehensive QA task is the SQuAD challenge (Rajpurkar et al., 2016) which aims to find a text span from given paragraph to answer the question. However, our task is quite different from SQuAD. With sufficient learning data, neural models have shown satisfying performance in the SQuAD challenge. In real world practice, examinees are acquired to learn and comprehend meta-knowledge from textbooks so as to pass the exams through review and reasoning among the whole knowledge space instead of simply finding an answer span from a given paragraph, which shows to be challenging for neural networks.

4.2 Semi-supervised Learning

For real-world applications of specific domains, the datasets are always inadequate. Semi-supervised Learning methods have been extensively studied to make use of unlabeled data. A batch of models have been proposed based on representation learning (Yang et al., 2017; Lei et al., 2016). For question answering, current semi-supervised models generally use auto-encoder framework or generative model to obtain the representation of unlabeled data. However, these semi-supervised models can not be applied to generate labels for our raw QA pairs since our task requires diverse types of labels from discourse to sentence which are difficult to annotate, and it is more serious that the entire unlabeled data is also too small to support an effective semi-supervised learning. Thus, we must very carefully choose a proper learning strategy for our task.

Different from the above research line, one-shot learning belongs to a kind of weakly supervised method (Bordes et al., 2014) that was first proposed to learn information about object categories from one or only a few training images (Li et al., 2006). Recently, memory-based neural network models have greatly expanded the ways of storing and accessing information. Two promising neural networks with external memory have been proposed to connect deep neural network with one-shot learning. One is Neural Turing Machine (Graves et al., 2014), which can read or write the facts to an external differentiable memory. Santoro et al. (2016) proved it to be an effective method for image recognition. The other is Memory Network (Vinyals et al., 2016). The crucial difference between them is that the latter does not have a mechanism to modify the content of the external memory, which lets us choose the former as our one-shot learning implementation. Compared with previous methods for question answering, our model is more weakly supervised with a limited amount of training data. As to our best knowledge, this is the first attempt that adopts one-shot learning for a deep QA task by using NTM as automatic labeler.

5 Conclusion

This paper presents a neural model with NTM for a challenging deep question answering task from history examinations. Our experimental results show that the adopted CGNN together with other neural models works much better than BM25. With an NTM labeler, all the deep neural models are further enhanced. We also release a Chinese comprehensive deep question answering dataset and have launched a new research line for the Gaokao challenge and solve more complicated questions using deep neural networks to learn semantic representation while the previous work focused on choice questions using simple information retrieval methods (Cheng et al., 2016). In fact, open-domain questions that can often be distinguished into different aspects can always benefit from one-shot learnings over a few labeling samples, which has been verified the effectiveness in this paper by only relying on the least task-dependent assumptions.

References

Gabor Angeli, Melvin Johnson Premkumar, and Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *ACL*, pages 344–354.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open question answering with weakly supervised embedding models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 165–180.

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntactic-agnostic or syntactic-aware? In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*.

Jifan Chen, Qi Zhang, Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016a. Implicit discourse relation detection via a deep architecture with gated relevance network. In *ACL*, pages 1726–1735.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016b. Enhancing and combining sequential and tree lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.

Gong Cheng, Weixi Zhu, Ziwei Wang, Jianghui Chen, and Yuzhong Qu. 2016. Taking up the gaokao challenge: An information retrieval approach. *IJCAI*, pages 2479–2485.

Peter Clark. 2015. Elementary school science and math tests as a driver for ai: Take the aristo challenge! In *IAAI*, pages 4019–4021.

Zihang Dai, Lei Li, and Wei Xu. 2016. CFO: Conditional focused neural question answering with large-scale knowledge bases. *arXiv preprint arXiv:1606.01994*.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(39):2121–2159.

Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. In *ASRU*, pages 813–820.

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

Akira Fujita, Akihiro Kameda, Ai Kawazoe, and Yusuke Miyao. 2014. Overview of todai robot project and evaluation framework of its nlp-based problem solving. *ICLR*, pages 2590–2597.

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

Shangmin Guo, Xiangrong Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2017. Which is the effective way for gaokao: Information retrieval or neural networks? In *EACL*, pages 111–120.

Shexia He, Zuchao Li, Hai Zhao, Hongxiao Bai, and Gongshen Liu. 2018. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann Lecun. 2016. Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969*.

Yafang Huang, Zuchao Li, Zhuosheng Zhang, and Hai Zhao. 2018. Moon IME: neural-based chinese pinyin aided input method with customizable association. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018), System Demonstration*.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Peter Clark, Oren Etzioni, and Dan Roth. 2016. Question answering via integer programming over semi-structured knowledge. In *IJCAI*, pages 1145–1152.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2017. Answering complex questions using open information extraction. *ACL*, pages 311–316.

Souvik Kundu and Hwee Tou Ng. 2018. A question-focused multi-factor attention network for question answering. *arXiv preprint arXiv:1801.08290*.

Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi Jaakkola, Katerina Tymoshenko, Alessandro Moschitti, and Lluis Marquez. 2016. Semi-supervised question retrieval with gated convolutions. In *NAACL*, pages 1279–1289.

Fei-Fei Li, R. Fergus, and P. Perona. 2006. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611.

Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. 2018. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*.

Hanqing Lu and Ming Kong. 2017. Community-based question answering via contextual ranking metric network learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 4963–4964.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric P. Xing. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1006–1017.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP 2016*, pages 2383–2392.

Spyridon Samothrakis, Tom Vodopivec, Michael Fairbank, and Maria Fasli. 2017. Convolutional-match networks for question answering. In *IJCAI*, pages 2686–2692.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. One-shot learning with memory-augmented neural networks. In *ICML*, pages 1842–1850.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. LSTM-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.

Mihaylov Todor and Frank Anette. 2018. Enhancing cloze-style reading comprehension with external common knowledge using explicit key-value memory. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*.

Armando Vieira. 2016. Knowledge representation in graphs using convolutional neural networks. *arXiv preprint arXiv:1612.02255*.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638.

Zhenghao Wang, Shengquan Yan, Huaming Wang, and Xuedong Huang. 2014. An overview of microsoft deep qa system on stanford webquestions benchmark. Technical report, Technical report, Microsoft Research.

Rui Wang, Hai Zhao, Bao Liang Lu, Masao Utiyama, and Eiichiro Sumita. 2015. Bilingual continuous-space language model growing for statistical machine translation. *IEEE/ACM Transactions on Audio Speech & Language Processing*, 23(7):1209–1220.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017a. Gated self-matching networks for reading comprehension and question answering. In *ACL*, pages 189–198.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017b. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. *arXiv*, 1603.

Min Chul Yang, Nan Duan, Ming Zhou, and Hae Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *EMNLP*, pages 645–650.

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William W Cohen. 2017. Semi-supervised qa with generative domain-adaptive nets. *arXiv preprint arXiv:1702.02206*.

Wen Tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *ACL*, pages 643–648.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *IJCAI*, pages 2972–2978.

Kai Zhang, Wei Wu, Fang Wang, Ming Zhou, and Zhoujun Li. 2016a. Learning distributed representations of data in community question answering for question retrieval. In *ACM*, pages 533–542.

Zhisong Zhang, Hai Zhao, and Lianhui Qin. 2016b. Probabilistic graph-based dependency parsing with convolutional neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1382–1392.

Xiaodong Zhang, Sujian Li, Lei Sha, and Houfeng Wang. 2017. Attentive interactive neural networks for answer selection in community question answering. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 3525–3531.

Zhuosheng Zhang, Yafang Huang, and Hai Zhao. 2018a. Subword-augmented embedding for cloze reading comprehension. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*.

Zhuosheng Zhang, Jiangtong Li, Hai Zhao, and Bingjie Tang. 2018b. Sjtu-nlp at semeval-2018 task 9: Neural hypernym discovery with term embeddings. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval 2018), Workshop of NAACL-HLT 2018*.

Zhuosheng Zhang, Jiangtong Li, Pengfei Zhu, and Hai Zhao. 2018c. Modeling multi-turn conversation with deep utterance aggregation. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*.

Hai Zhao, Chang-Ning Huang, Mu Li, and Taku Kudo. 2006. An improved Chinese word segmentation system with conditional random field. *Proceedings of the Fifth Sighan Workshop on Chinese Language Processing*, pages 162–165.

Dynamic Multi-Level Multi-Task Learning for Sentence Simplification

Han Guo **Ramakanth Pasunuru** **Mohit Bansal**
UNC Chapel Hill
{hanguo, ram, mbansal}@cs.unc.edu

Abstract

Sentence simplification aims to improve readability and understandability, based on several operations such as splitting, deletion, and paraphrasing. However, a valid simplified sentence should also be logically entailed by its input sentence. In this work, we first present a strong pointer-copy mechanism based sequence-to-sequence sentence simplification model, and then improve its entailment and paraphrasing capabilities via multi-task learning with related auxiliary tasks of entailment and paraphrase generation. Moreover, we propose a novel 'multi-level' layered soft sharing approach where each auxiliary task shares different (higher versus lower) level layers of the sentence simplification model, depending on the task's semantic versus lexico-syntactic nature. We also introduce a novel multi-armed bandit based training approach that dynamically learns how to effectively switch across tasks during multi-task learning. Experiments on multiple popular datasets demonstrate that our model outperforms competitive simplification systems in SARI and FKGL automatic metrics, and human evaluation. Further, we present several ablation analyses on alternative layer sharing methods, soft versus hard sharing, dynamic multi-armed bandit sampling approaches, and our model's learned entailment and paraphrasing skills.

1 Introduction

Sentence simplification is the task of improving the readability and understandability of an input text. This challenging task has been the subject of research interest because it can address automatic ways of improving reading aids for people with limited language skills, or language impairments such as dyslexia (Rello et al., 2013), autism (Evans et al., 2014), and aphasia (Carroll et al., 1999). It also has wide applications in NLP tasks as a preprocessing step, for example, to improve the performance of parsers (Chandrasekar et al., 1996), summarizers (Klebanov et al., 2004), and semantic role labelers (Vickrey and Koller, 2008; Woodsend and Lapata, 2014).

Several sentence simplification systems focus on operations such as splitting a long sentence into shorter sentences (Siddharthan, 2006; Petersen and Ostendorf, 2007), deletion of less important words/phrases (Knight and Marcu, 2002; Clarke and Lapata, 2006; Filippova and Strube, 2008), and paraphrasing (Devlin, 1999; Inui et al., 2003; Kaji et al., 2002). Inspired from machine translation based neural models, recent work has built end-to-end sentence simplification models along with attention mechanism, and further improved it with reinforcement-based policy gradient approaches (Zhang and Lapata, 2017). Our baseline is a novel application of the pointer-copy mechanism (See et al., 2017) for the sentence simplification task, which allows the model to directly copy words and phrases from the input to the output. We further improve this strong baseline by bringing in auxiliary entailment and paraphrasing knowledge via soft and dynamic multi-level, multi-task learning.[1]

Apart from the three simplification operations discussed above, we also ensure that the simplified output is a directed logical entailment w.r.t. the input text, i.e., does not generate any contradictory or unrelated information. We incorporate this entailment skill via multi-task learning (Luong et al., 2015)

[1] All code and pretrained models available at: https://github.com/HanGuo97/MultitaskSimplification.

Proceedings of the 27th International Conference on Computational Linguistics, pages 462–476
Santa Fe, New Mexico, USA, August 20-26, 2018.

with an auxiliary entailment generation task. Further, we also induce word/phrase-level paraphrasing knowledge via a paraphrase generation task, enabling parallel learning of these three tasks in a three-way multi-task learning setup. We employ a novel 'multi-level' layered, soft sharing approach, where the parameters between the tasks are loosely coupled at different levels of layers; we share higher-level semantic layers between the sentence simplification and entailment generation tasks (which teaches the model to generate outputs that are entailed by the full input), while sharing the lower-level lexico-syntactic layers between the sentence simplification and paraphrase generation tasks (which teaches the model to paraphrase only the smaller sub-sentence pieces).

Finally, we also propose a multi-armed bandit approach that dynamically learns an effective schedule (curriculum) of switching between tasks for optimization during multi-task learning, instead of the traditional approach with a manually-tuned static (fixed) mixing ratio (Luong et al., 2015).

Empirically, we evaluate our system on three standard datasets: Newsela, WikiSmall, and WikiLarge. First, we show that our pointer-copy baseline is significantly better than sequence-to-sequence models, and competitive w.r.t. the state-of-the-art. Next, we show that our multi-level, multi-task framework performs significantly better than our strong pointer baseline and other competitive sentence simplification models on both automatic evaluation as well as on human study simplicity criterion. Further, we show that the dynamic multi-armed bandit based switching of tasks during training improves over the traditional manually-tuned static mixing ratio. Lastly, we show several ablation studies based on different layer-sharing approaches (higher versus lower) with auxiliary tasks, hard versus soft sharing, dynamic mixing ratio sampling, as well as our model's learned entailment and paraphrasing skills.

2 Related Work

Previous approaches to sentence simplification systems range from hand-designed rules (Siddharthan, 2006), to syntactic and lexical simplification via synonyms and paraphrases (Siddharthan, 2014; Kaji et al., 2002; Horn et al., 2014; Glavaš and Štajner, 2015), as well as treating simplification as a monolingual MT task, where operations are learned from examples of complex-simple sentence pairs (Specia, 2010; Koehn et al., 2007; Coster and Kauchak, 2011; Zhu et al., 2010; Wubben et al., 2012; Narayan and Gardent, 2014). Recently, Xu et al. (2016) trained a syntax-based MT model using the newly proposed SARI as a simplification-specific objective. Further, Zhang and Lapata (2017) used reinforcement learning in a sequence-to-sequence approach to directly optimize simplification metrics. In this work, we first introduce the pointer-copy mechanism (See et al., 2017) as a novel application to sentence simplification, and then use multi-task learning to bring in auxiliary entailment and paraphrasing skills.

Multi-task learning, known for improving the generalization performance of a task with related tasks, has successful application to many domains of machine learning (Caruana, 1998; Collobert and Weston, 2008; Girshick, 2015; Luong et al., 2015; Pasunuru and Bansal, 2017; Pasunuru et al., 2017). Although there are many variants of multi-task learning (Ruder et al., 2017; Hashimoto et al., 2017; Luong et al., 2015), our approach is similar to Luong et al. (2015), where different tasks share some common model parameters with alternating mini-batches optimization. In this work, we explore a multi-level (i.e., task-specific higher-level semantic versus lower-level lexico-syntactic layer sharing) and soft-sharing mechanism for improving sentence simplification via related tasks of entailment and paraphrase generation.

Recognizing Textual Entailment (RTE) is the task of predicting entailment, contradiction, or neutral relationships, and is useful for many downstream tasks like Q&A, summarization, and information retrieval (Harabagiu and Hickl, 2006; Dagan et al., 2006; Lai and Hockenmaier, 2014; Jimenez et al., 2014). Neural network models (Bowman et al., 2015; Parikh et al., 2016) and large datasets (Bowman et al., 2015; Williams et al., 2017) enabled recent strong progress. Recently, Pasunuru et al. (2017) and Guo et al. (2018) presented results using entailment generation as an auxiliary task for abstractive summarization; however, we use entailment as well as paraphrasing knowledge in a soft and multi-level layer sharing setup to improve sentence simplification.

Previous work (Barzilay and McKeown, 2001; Ganitkevitch et al., 2013; Wieting and Gimpel, 2017a) has developed methods and datasets for generating paraphrase pairs which can be useful for downstream applications such as question answering, semantic parsing, and information extraction (Fader et al., 2013;

Berant and Liang, 2014; Zhang et al., 2015). Wieting and Gimpel (2017a) recently introduced a large sentential paraphrase dataset via back-translation, and showed promising results when applied to learning sentence embeddings. In this work, we use this paraphrase dataset as an auxiliary generation task to improve our sentence simplification model by teaching it about paraphrasing in a multi-task setting.

Many control problems can be cast as a multi-armed bandits algorithm, where the goal of the agent is to select the arm/action from one of the M choices that gives the maximum expected future reward (Bubeck et al., 2012). Optimal control and reinforcement learning have been used to find the trade-off between exploitation and exploration, and yield theoretically-sound regret bounds, e.g., Boltzmann exploration (Kaelbling et al., 1996), UCB (Auer et al., 2002a), Thompson sampling (Chapelle and Li, 2011), adversarial bandits (Auer et al., 2002b), and information gain using variational approaches (Houthooft et al., 2016). Recently, Graves et al. (2017) use a non-stationary multi-armed bandit to automatically select the curriculum or syllabus that a neural network follows so as to maximize learning efficiency. Sharma and Ravindran (2017) use multi-armed bandit sampling to choose which domain data (harder vs. easier) to feed as input to a single model (using different Atari games), whereas we use multi-armed bandit sampling to decide the optimization curriculum (mixing ratio) among our three models for sentence simplification, entailment generation, and paraphrase generation (with different softly-shared layers).

3 Models

In this section, we first describe our sentence simplification baseline model with attention mechanism, which is further improved by pointer-copy mechanism. Later, we introduce our two auxiliary tasks (entailment and paraphrase generation) and discuss how they can share specific lower/higher-level layers/parameters to improve the sentence simplification task in a multi-task learning setting. Finally, we discuss our new multi-armed bandit based dynamic multi-task learning approach.

3.1 Pointer-Copy Baseline Sentence Simplification Model

Our baseline is a 2-layer sequence-to-sequence model with both attention (Bahdanau et al., 2015) and pointer-copy mechanism (See et al., 2017). Given the sequence of input/source tokens $x = \{x_1, ..., x_{T_s}\}$, the model learns an auto-regressive distribution over output/target tokens $y = \{y_1, ..., y_{T_o}\}$, which is defined as $P_{vocab}(y|x;\theta) = \prod_t p(y_t|y_{1:t-1}, x;\theta)$, where θ represents model parameters and $p(y_t|y_{1:t-1}, x;\theta)$ is probability of generating token y_t at decoder time step t given the previous generated tokens $y_{1:t-1}$ and input x. Given encoder hidden states $\{h_i\}$, and decoder's t^{th} time step hidden state (of last layer) s_t, the context vector $c_t = \sum_i \alpha_{t,i} h_i$, where the attention weights $\alpha_{t,i}$ define an attention distribution over encoder hidden states: $\alpha_{t,i} = \exp(e_{t,i}) / \sum_k \exp(e_{t,k})$, where $e_{t,i} = v_a^T \tanh(W_a s_t + U_a h_i + b_a)$. Finally, the conditional distribution at each time step t of the decoder is defined as $p(y_t|y_{1:t-1}, x;\theta) = \text{softmax}(W_s s'_t)$, where the final hidden state s'_t is a combination of context vector c_t and last layer hidden state s_t and is defined as $s'_t = \tanh(W_c[c_t, s_t])$, where W_s and W_c are trained parameters.

Pointer-Copy Mechanism: This helps in directly copying the words from the source inputs to the target outputs via merging the generative distribution and attention distribution (as a proxy of copy distribution). The goal of sentence simplification is to rewrite sentences more simply, while preserving important information; hence, it also involves significant amount of copying from the source. Our pointer mechanism approach is similar to See et al. (2017). At each time step of the decoder, the model makes a (soft) choice between words from the vocabulary distribution P_{vocab} and attention distribution P_{att} (based on words in the input) using the word generation probability $p_g = \sigma(W_g c_t + U_g s_t + V_g d_t + b_g)$, where $\sigma(\cdot)$ is sigmoid, W_g, U_g, V_g and b_g are trainable parameters, and d_t is decoder input. The final vocabulary distribution is defined as the weighted combination of vocabulary and attention distributions:

$$P_f(y) = p_g P_{vocab}(y) + (1 - p_g) P_{att}(y) \tag{1}$$

3.2 Auxiliary Tasks

Entailment Generation The task of entailment generation is to generate a hypothesis which is entailed by the given input premise. A good simplified sentence should be entailed by (follow from) the source

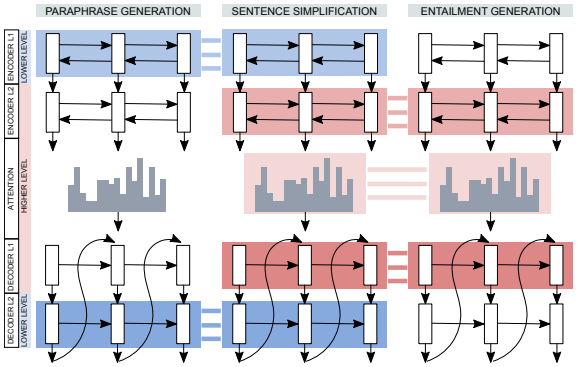


Figure 1: Overview of our 3-way multi-task model. Same color and dashed connections represent soft-shared parameters in different layers.

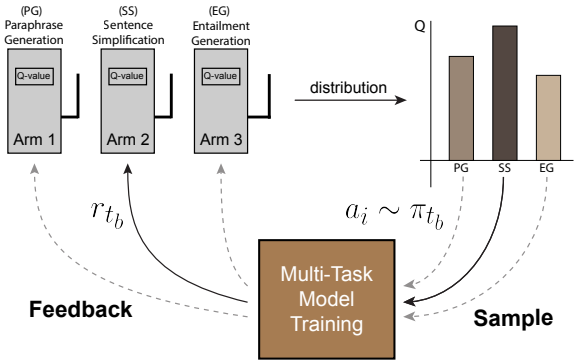


Figure 2: Overview of our multi-armed bandits algorithm for dynamic mixing ratio learning. It consists of a controller with 3 arms/tasks.

sentence, and hence we incorporate such knowledge through an entailment generation task into our sentence simplification task. We share the higher-level semantic layers between the two tasks (see reasoning in Sec. 3.3 below). We use entailment pairs from SNLI (Bowman et al., 2015) and Multi-NLI (Williams et al., 2017) datasets for training our entailment generation model, where we use the same architecture as our sentence simplification model.

Paraphrase Generation Paraphrase generation is the task of generating similar meaning phrases or sentences by reordering and modifying the syntax and/or lexicon. Paraphrasing is one of the common operations used in sentence simplification, i.e, by substituting complex words and phrases with their simpler paraphrase forms. Hence, we add this knowledge to the sentence simplification task via multi-task learning, by sharing the lower-level lexico-syntactic layers between the two tasks (see reasoning in Sec. 3.3 below). For this, we use the paraphrase pairs from ParaNMT (Wieting and Gimpel, 2017a). Here, again, we use the same architecture as our sentence simplification model.

3.3 Multi-Task Learning

In this subsection, we discuss our multi-task, multi-level soft sharing strategy with parallel training of sentence simplification and related auxiliary tasks (entailment and paraphrase generation).

The predominant approach for multi-task learning in sequence-to-sequence models is to directly hard-share all encoder/decoder layers/parameters (Luong et al., 2015; Johnson et al., 2016; Pasunuru and Bansal, 2017; Kaiser et al., 2017). However, this approach places very strong constraints/priors on the primary model to compress knowledge from diverse tasks. We believe that while the auxiliary tasks considered in this work share many similarities with the primary sentence simplification task, they are still different in either lower-level or higher-level representations (e.g., entailment will deal with higher-level, full-sentence logical inference, while paraphrasing will handle the lower-level intermediate word/phrase simplifications). In this section, we propose to relax the priors in two ways: (1) we share the model parameters in a finer-grained scale, i.e. layer-specific sharing, by keeping some of their parameters private, while sharing related representations; and (2) we encourage shared parameters to be close in certain distance metrics with a penalty term instead of hard-parameter-tying (Luong et al., 2015).

Multi-Level Sharing Mechanism Fig. 1 shows our multi-task model with parallel training of three tasks: sentence simplification (primary task), entailment generation (auxiliary task), and paraphrase generation (auxiliary task). Recently, Belinkov et al. (2017) observed that different layers in a sequence-to-sequence model (trained on translation) exhibit different functionalities: lower-layers (closer to inputs) of the encoder learn to represent word structure while higher layers (farther from inputs) are more focused on semantics and meanings (Zeiler and Fergus (2014) observed similar findings for convolutional image features). Based on these findings, we share the higher-level layers[2] between the entailment generation and sentence simplification tasks, since they share higher semantic-level language inference

[2]We found that sharing higher-level semantic layers (farther from input/output), i.e., encoder layer 2, attention, and decoder layer 1 (in Fig. 1), to work well. See Sec. 6 for ablations on alternative layer sharing methods.

skills (for full sentence-to-sentence logical directedness). On the other hand, we share the lower-level lexico-syntactic layers[3] between the paraphrase generation and sentence simplification tasks, since they share more word/phrase and syntactic level paraphrasing knowledge to simplify the smaller, intermediate sentence pieces. Sec. 6 present empirical ablations to support our intuitive layer sharing.[4]

Soft Sharing In multi-task learning, we can do either hard sharing or soft sharing of parameters. Hard sharing directly ties the parameters to be shared, and receives gradient information from multiple tasks. On the other hand, soft sharing only loosely couples the parameters, and encourages them to be close in representation space. Hence the soft sharing approach gives more flexibility for parameter sharing, hence allowing different tasks to choose what parts of their parameters space to share. We minimize the l_2 distance between shared parameters as a regularization along with the cross entropy loss. Hence, the final loss function of the primary task with a related auxiliary task is defined as follows:

$$L(\theta) = -\log P_f(y|x;\theta) + \lambda||\theta_s - \phi_s|| \tag{2}$$

where θ represents the full parameters of the primary task (sentence simplification), θ_s and ϕ_s are the subsets of shared parameters between the primary and auxiliary task resp., and λ is a hyperparameter.

Multi-Task Training We employ multi-task learning with parallel training of related tasks in alternate mini-batches based on a mixing ratio α_{ss}:α_{eg}:α_{pp}, where we alternatively optimize α_{ss}, α_{eg}, α_{pp} mini-batches of sentence simplification, entailment generation, and paraphrase generation, respectively, until all models converge. In the next section, we discuss a new approach to replace this static mixing ratio with dynamically-learned task switching.

3.4 Dynamic Mixing Ratio Learning

Current multi-task models are trained via alternate mini-batch optimization based on a task 'mixing ratio' (Luong et al., 2015; Pasunuru and Bansal, 2017), i.e., how many iterations on each task relative to other tasks (see end of Sec. 3.3). This is usually treated as a very important hyperparameter to be tuned, and the search space scales exponentially with the number of tasks. Hence, we importantly replace this manually-tuned and static mixing ratio with a 'dynamic' mixing ratio learning approach, where a controller automatically switches between the tasks during training, based on the current state of the multi-task model. Specifically, we use a multi-armed bandits based controller with Boltzmann exploration (Kaelbling et al., 1996) with an exponential moving average update rule.

We view the problem of learning the right mixing of tasks as a sequential control problem, where the controller's goal is to decide the next task/action after every n^s training steps in each task-sampling round t_b.[5] Let $\{a_1, ..., a_M\}$ represent the set of 3 tasks in our multi-task setting, i.e., sentence simplification, entailment generation, and paraphrase generation. We model the controller as a M-armed bandits, where it selects a sequence of actions/arms over the current training trajectory to maximize the expected future payoffs (see Fig. 2). At each round t_b, the controller selects an arm based on noisy value estimates and observes rewards r_{t_b} for the selected arm (we use the negative validation loss of the primary task as the reward in our setup). One problem in bandits learning is the trade-off between exploration and exploitation, where the agent needs to make a decision between taking the action that yields the best payoff on current estimates, or explore new actions whose payoffs are not yet certain. For this, we use the Boltzmann exploration (Kaelbling et al., 1996) with exponentially moving action value estimates. Let π_{t_b} be the policy of the bandit controller at round t_b, we define this to be:

$$\pi_{t_b}(a_i) = \exp(Q_{t_b,i}/\tau) \Big/ \sum_{j=1}^{M} \exp(Q_{t_b,j}/\tau) \tag{3}$$

[3]We found that sharing lower-level lexico-syntactic layers (closer to input/output), i.e., encoder layer 1 and decoder layer 2 (in Fig. 1), to work well. See Sec. 6 for ablations on alternative layer sharing methods.

[4]Note that even though entailment just tries to generate shorter, logical-subset sub-sentences, the overall saliency and quality of the simplified output is still balanced because the entailment task is flexibly (softly) shared with the paraphrasing and sentence simplification tasks, and the final model mixture is chosen based on simplification task metrics (see output examples in Fig. 4 where our multi-task model generates entailed sentences with important information).

[5]We set n^s to 10 to reduce variance of estimates, i.e., the bandit controller's task/action will be trained for 10 mini-batches.

where $Q_{t_b,i}$ is the estimated action value of each arm i at round t_b, and τ is the temperature.[6] If $Q_{0,i}$ is the initial value estimate of arm i, then $Q_{t_b,i}$ is the exponentially weighted mean with the decay rate α:

$$Q_{t_b,i} = (1-\alpha)^{t_b} Q_{0,i} + \sum_{k=1}^{t_b} \alpha(1-\alpha)^{t_b-k} r_k \tag{4}$$

To further help the exploration process, we follow the principle of optimism under uncertainty (Sutton and Barto, 1998) and set $Q_{0,i}$ to be above the maximum empirical rewards. Empirically, we show that this approach of 'dynamic mixing ratio' is equal or better than the traditional static mixing ratio (see Table 3). Also, we further show ablation study in Sec. 6 to show that this switching approach is better than the alternative approach of first using multi-armed bandits for finding an optimal 'final' mixing ratio and then re-training the model based on this bandits-selected mixing ratio.

4 Evaluation Setup

Datasets We first describe the three standard sentence simplification datasets we evaluate on: Newsela, WikiSmall, and WikiLarge; next, we describe datasets for our auxiliary entailment and paraphrase generation tasks. **Newsela** (Xu et al., 2015) is acknowledged as a higher-quality dataset for studying sentence simplifications, as opposed to Wikipedia-based datasets which automatically align complex-simple sentence pairs and have generalization issues (Zhang and Lapata, 2017; Xu et al., 2015; Amancio and Specia, 2014; Hwang et al., 2015; Štajner et al., 2015). Newsela consists of $1,130$ news articles, and we follow previous work (Zhang and Lapata, 2017) to use the first $1,070$ documents for training, and 30 documents each for development and test. **WikiSmall** (Zhu et al., 2010) contains automatically-aligned complex-simple sentences from the ordinary-simple English Wikipedias. The data has $89,042$ pairs for training and 100 for test. We use the 205-pairs validation set from Zhang and Lapata (2017). **WikiLarge** (Zhang and Lapata, 2017) is a larger Wikipedia corpus aggregating pairs from Kauchak (2013), Woodsend and Lapata (2011), and WikiSmall. We use the exact training/evaluation sets provided by Zhang and Lapata (2017). **SNLI and MultiNLI**: For the task of entailment generation, we use the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015) and MultiNLI (Williams et al., 2017). We use their entailment labeled pairs for our entailment generation task, following previous work (Pasunuru and Bansal, 2017). The combined SNLI and MultiNLI dataset has $302,879$ entailment pairs, out of which we use $276,720$ pairs for training, and the rest are divided into validation and test sets. **ParaNMT**: For the task of paraphrase generation, we use the back-translated paraphrase dataset provided by Wieting and Gimpel (2017a). The filtered version of the dataset has 5.3 million pairs of paraphrases.[7] We use 99% for training, and the rest are evenly divided into validation and test sets.

Evaluation Metrics Following previous work (Zhang and Lapata, 2017), we report all the standard evaluation metrics: SARI (Xu et al., 2016), FKGL (Kincaid et al., 1975), and BLEU (Papineni et al., 2002). However, several studies have shown that BLEU is poorly correlated w.r.t. simplicity (Zhu et al., 2010; Štajner et al., 2015; Xu et al., 2016). Moreover, Shardlow (2014) argues that FKGL (Kincaid et al., 1975), which measures readability of simpler output (lower is better), favors very short sentences even though longer/less coarse counterparts can be simpler. Further, Xu et al. (2016) argues that BLEU tends to favor conservative systems that do not make many changes, and proposes SARI metric which explicitly measures the quality of words that are added and deleted. SARI is shown to correlate well with human judgment in simplicity (Xu et al., 2016), and hence we primarily focus on this metric in our models' performance analysis.[8] Further, we also do human evaluation based on: Fluency ('is the output grammatical and well formed?'), Adequacy ('to what extent is the meaning expressed in the original sentence preserved in the output?') and Simplicity ('is the output simpler than the original sentence?'), following guidelines suggested by Xu et al. (2016) and Zhang and Lapata (2017).

[6] We tried decaying the temperature variable, but we didn't find this to very beneficial, so we instead fix this to 1.0.

[7] We chose ParaNMT over other paraphrase datasets (e.g. the phrase-to-phrase PPDB dataset (Ganitkevitch et al., 2013)), because ParaNMT is a sentence-to-sentence dataset and hence is a more natural fit for sentence-level multi-task RNN-layer sharing with our sentence-to-sentence simplification task.

[8] We use the JOSHUA package for calculating SARI and BLEU score following Zhang and Lapata (2017) and Xu et al. (2016). Our FKGL implementation is based on `https://github.com/mmautner/readability`.

Models	BLEU	FKGL	SARI
PREVIOUS WORK			
PBMT-R	18.19	7.59	15.77
Hybrid	14.46	4.01	30.00
EncDecA	21.70	5.11	24.12
DRESS	23.21	4.13	27.37
DRESS-LS	24.30	4.21	26.63
OUR MODELS			
Baseline ⊗	23.72	3.25	28.31
⊗ + Ent.	16.82	2.21	31.55
⊗ + Paraphr.	16.29	2.03	31.71
⊗+Ent+Par	11.86	1.38	32.98

Table 1: Newsela (FKGL: lower is better).

Models	WIKISMALL			WIKILARGE		
	BLEU	FKGL	SARI	BLEU	FKGL	SARI
PREVIOUS WORK						
PBMT-R	46.31	11.42	15.97	81.11	8.33	38.56
Hybrid	53.94	9.21	30.46	48.97	4.56	31.40
SBMT-SARI	-	-	-	73.08	7.29	39.96
EncDecA	47.93	11.35	13.61	88.85	8.41	35.66
DRESS	34.53	7.48	27.48	77.18	6.58	37.08
DRESS-LS	36.32	7.55	27.24	80.12	6.62	37.27
OUR MODELS						
Baseline ⊗	36.18	7.69	25.67	82.37	7.84	36.68
⊗+Ent+Par	29.70	6.93	28.24	81.49	7.41	37.45

Table 2: WikiSmall/Large results (FKGL: lower is better).

Training Details All our soft/hard and layer-specific sharing decisions (Sec. 6) were made on the validation/dev set. Our model selection (tuning) criteria is based on the average of our 3 metrics (SARI, BLEU, 1/FKGL) on the validation set. Please refer to the appendix for full training details (vocabulary overlap, mixing ratios and bandit sampler decay rates and reward, WikiLarge pre-training, etc.).

5 Results

We evaluate our models on three datasets and via several automatic metrics plus human evaluation.[9]

Pointer Baseline First, we compare our pointer baseline with various previous works: PBMT-R (Wubben et al., 2012), Hybrid (Narayan and Gardent, 2014), SBMT-SARI (Xu et al., 2016)[10], and EncDecA, DRESS, and DRESS-LS (Zhang and Lapata, 2017). As shown in Table 1, our pointer baseline already achieves the best score in FKGL and the second-best score in SARI on Newsela, and also achieves overall comparable results on both WikiSmall and WikiLarge (see Table 2).

Multi-Task Models We further improve our strong pointer-based sentence simplification baseline model by multi-task learning it with entailment and paraphrase generation. First, we show that our 2-way multi-task models with auxiliary tasks (entailment and paraphrase generation) are statistically significantly better than our pointer baseline and previous works in both SARI and FKGL on Newsela (see Table 1).[11] Next, Table 1 and Table 2 summarize the performance of our final 3-way multi-level, multi-task models with entailment generation and paraphrase generation on all three datasets. Here, our 3-way multi-task models are statistically significantly better than our pointer baselines in both SARI and FKGL (with $p < 0.01$) on Newsela and WikiSmall, and in SARI ($p < 0.01$) on WikiLarge. Also, our 3-way multi-task model is statistically significantly better than the 2-way multi-task models in SARI and FKGL with $p < 0.01$ (see Table 1). In Sec. 6, we further provide a set of detailed ablation experiments investigating the effects of different (higher-level versus lower-level) layer sharing methods and soft- vs. hard-sharing in our multi-level, multi-task models; and we show the superiority of our final choice of higher-level semantic sharing for entailment generation and lower-level lexico-syntactic sharing for paraphrase generation.

Models	BLEU	FKGL	SARI
NEWSELA			
Static Mixing Ratio	11.86	1.38	32.98
Dynamic Mixing Ratio	11.14	1.32	33.22
WIKISMALL			
Static Mixing Ratio	29.70	6.93	28.24
Dynamic Mixing Ratio	27.23	5.86	29.58

Table 3: Results on dynamic vs. static mixing ratio (FKGL: lower is better).

Dynamic Mixing Ratio Models Finally, we present results on our 3-way multi-task model with the new approach of using 'dynamic' mixing ratios based on multi-armed bandits sampling (see Sec. 3.4).

[9] As described in Sec. 4, Newsela is considered as a higher quality dataset for text simplification, and thus we report ablation-style results (e.g., 2-way multi-task models and different layer-sharing ablations) and human evaluation on Newsela (since Wikipedia datasets are automatically-aligned). Moreover, we report SARI, FKGL, and BLEU for completeness, but as described in Sec. 4, SARI is the primary human-correlated metric for sentence simplification.

[10] We borrow the SBMT-SARI results for WikiLarge from Zhang and Lapata (2017).

[11] Stat. significance is computed via bootstrap test (Noreen, 1989; Efron and Tibshirani, 1994). Both our 2-way multi-task models are statistically significantly better in SARI and FKGL with $p < 0.01$ w.r.t. our pointer baseline and previous works. Note the discussion in Sec. 4 about why BLEU is not a good sentence simplification metric.

Models	HUMAN EVALUATION				MATCH WITH INPUT		
	Fluency	Adequacy	Simplicity	Average	BLEU (%)	ROUGE (%)	Exact Match (%)
Ground-truth	4.97	4.08	3.83	4.29	18.25	43.74	0.00
Hybrid	3.88	3.82	3.92	3.87	25.74	56.20	3.34
DRESS-LS	4.84	4.18	3.21	4.08	42.93	67.61	14.48
Pointer Baseline	4.61	3.94	3.99	4.18	30.80	60.56	10.68
3-way Multi-task	4.73	3.18	4.62	4.18	8.74	37.82	2.41

Table 4: Human evaluation results (on left) and closeness-to-input source results (on right), for Newsela.

As shown in Table 3, this dynamic multi-task approach achieves a stat. significant improvement in SARI as compared to the traditional fixed and manually-tuned mixing ratio based 3-way multi-task model: 33.22 vs. 32.98 ($p < 0.05$) on Newsela, and 29.58 vs. 28.24 ($p < 0.001$) on WikiSmall. Hence, this allows us to achieve better results while also avoiding the hassle of tuning on the large space of mixing ratios over several different tasks. In Sec. 6, we further provide ablation analysis to study whether the improvements come from the bandit learning this dynamic curriculum or from the bandit finding the final optimal mixing-ratio at the end of the sampling procedure (and also compare it to a random curriculum).

Human Evaluation We also perform an anonymized human study comparing our pointer baseline, our multi-task model, some previous works (Hybrid (Narayan and Gardent, 2014) and state-of-the-art DRESS-LS (Zhang and Lapata, 2017)), and ground-truth references (see left part of Table 4), based on fluency, adequacy, and simplicity (see Sec. 4 for more details about these criteria) using 5-point Likert scale. We asked annotators to evaluate the models (randomly shuffled to anonymize model identity) based on 200 samples from the representative and cleaner Newsela test set, and their scores are reported in Table 4. Our 3-way multi-task model achieves a significantly higher ($p < 0.001$) simplicity score compared to DRESS-LS, Hybrid, and our pointer baseline models. However, we next observe that our 3-way multi-task model has lower adequacy score as compared to DRESS-LS and the pointer model, but this is because our 3-way multi-task model focuses more strongly on simplification, which is the goal of the given task. Moreover, based on the overall average score of the three human evaluation criteria, our 3-way multi-task model is also significantly better ($p < 0.03$) than the state-of-the-art DRESS-LS model (and $p < 0.001$ w.r.t. Hybrid model).[12] Also, on further investigation, we found that a problem with the adequacy metric is that it gets artificially high scores for output sentences which are exact match (or a very close match) with the input source sentence, i.e., they have very little simplification and hence almost fully retain the exact meaning. In the right part of Table 4, we analyzed the matching scores of the outputs from different models w.r.t. the source input text, based on BLEU, ROUGE (Lin, 2004) and exact match. First, this shows that the ground-truth sentence-simplification references are in fact (as expected) very different from the input source (0% exact match, 18% BLEU, 44% ROUGE). Next, we find that our multi-task model also has low match-with-input scores (2% exact match, 9% BLEU, 38% ROUGE), similar to the behavior of the ground-truth references. On the other hand, DRESS-LS (and pointer baseline) model is generating output sentences which are substantially closer to the input and hence is not making enough changes (14% exact match, 43% BLEU, 68% ROUGE) as compared to the references (which explains their higher adequacy but lower simplicity scores).

6 Ablations and Analysis

In this section, we conduct several ablation analyses to study the different layer-sharing mechanisms (higher semantic vs. lower lexico-syntactic), soft- vs. hard-sharing, two dynamic multi-armed bandit approaches, and our model's learned entailment and paraphrasing skills. We also present and analyze some output examples from several models.[13] Note that all our soft and layer sharing decisions were strictly made on the dev/validation set (see Sec. 4).

[12]Note that our multi-task model is stat. equal to our pointer baseline on the overall-average score, showing the available trade-off between systems that simplify conservatively vs. strongly, based on one's desired downstream task application. Also refer to the high 'match-with-input' issue with the adequacy metric discussed next.

[13]Since Newsela is considered as the more representative dataset for sentence simplification with lesser noise and human quality (Xu et al., 2015; Zhang and Lapata, 2017), we conduct our ablation studies on this dataset, but we observed similar patterns on the other two datasets as well.

Models	Entailment	Paraphrasing
Ground-truth	N/A	62.1
Hybrid	34.8	74.1
DRESS-LS	30.7	77.9
Pointer Baseline	36.9	76.6
2-way Multi-Task	41.4	63.9

Table 6: Analysis: Entailment and paraphrase classification results (avg. probability scores as %) on Newsela.

Models	Deletions	Additions
Hybrid	95.18	0.000
DRESS-LS	85.37	0.047
Pointer Baseline	88.91	0.026
3-way Multi-Task	97.54	0.049

Table 7: Analysis: SARI's sub-operation scores on Newsela dataset.

Models	BLEU	FKGL	SARI
Final (High Ent + Low PP)	**11.86**	**1.38**	**32.98**
Both lower-layer	11.94	1.47	31.92
Both higher-layer	12.26	1.38	32.02
Swapped (Low Ent + High PP)	21.64	2.97	29.07
Hard-sharing	13.01	1.38	32.36

Table 5: Multi-task layer ablation results on Newsela.

Different Layer Sharing Approaches We empirically show that our final multi-level layer sharing method (i.e., higher-level semantic layer sharing with entailment generation, while lower-level lexico-syntactic layer sharing with paraphrase generation) performs better than the following alternative layer sharing methods: (1) both auxiliary tasks with high-level layer sharing, (2) both with low-level layer sharing, and (3) reverse/swapped sharing (i.e. lower-level layer sharing for entailment, and higher-level layer sharing for paraphrasing). Results in Table 5 show that our approach of high-level sharing for entailment generation and low-level sharing for paraphrase generation is statistically significantly better than all other alternative approaches in SARI ($p < 0.01$) (and statistically better or equal in FKGL).

Soft- vs. Hard-Sharing In this work, we use soft-sharing instead of hard-sharing approach (benefits discussed in Sec. 3.3) in all of our models. Table 5 also presents empirical results comparing soft- vs. hard-sharing on our final 3-way multi-task model, and we observe that soft-sharing is statistically significantly better than hard-sharing in SARI with $p < 0.01$.

Quantitative Improvements in Entailment We employ a state-of-the-art entailment classifier (Chen et al., 2017) to calculate the entailment probabilities of output sentence being entailed by the ground-truth.[14] Table 6 summaries the average entailment scores for the Hybrid, DRESS-LS, Pointer baseline, and 2-way multi-task model (with entailment generation auxiliary task), showing that the 2-way multi-task model improves in the aspect of logical entailment ($p < 0.001$), demonstrating the inference skill acquired by the simplification model via the auxiliary knowledge from the entailment generation task.

Quantitative Improvements in Paraphrasing We use the paraphrase classifier from Wieting and Gimpel (2017b) to compute the paraphrase probability score between the generated output and the input source. The results in Table 6 show that our 2-way multi-task model (with paraphrasing generation auxiliary task) is closer to the ground-truth in terms of the amount of paraphrasing (w.r.t. input) required by the sentence-simplification task, while the pointer baseline and previous models have higher scores due to higher amount of copying from input source (see 'Match-with-Input' discussion in Sec. 5, Table 4).

Addition/Deletion Operations We also measured the performance of the various models in terms of the addition and deletion operations using SARI's sub-operation scores computed w.r.t. both the ground-truth and source (Xu et al., 2016). Table 7 shows that our multi-task model is equal or better in terms of both operations.

Two Multi-Armed-Bandit Approaches As described in Sec. 3.4, our multi-armed bandit approach with dynamic mixing ratio during multi-task training learns a sufficiently good curriculum to improve the sentence simplification task (see Sec. 5). Here, we further show an ablation study on another alternative approach of using multi-armed bandits, where we record the last 10% of the actions from the bandit controller[15], then calculate the corresponding mixing ra-

[14]For this entailment analysis, we use ground-truth output as premise instead of input source, because: (1) entailment w.r.t. input source can give artificially high scores even when the output doesn't simplify enough and just copies the source (see the discussion in Sec. 5 and Table 4); (2) By transitivity, if output is entailed by ground-truth, which in turn is entailed by source, then output should also be entailed by source (plus, we want the output to be closer to ground-truth than to input source).

[15]We choose the last 10% to avoid the noisy action-value estimates at the start of the training.

tio based on this 10%, and run another independent model from scratch with this fixed mixing ratio. We found that the curriculum-style dynamic switching of tasks is in fact very effective as compared to this other 2-stage approach (33.22 versus 32.58 in SARI with $p < 0.01$). This is intuitive because the dynamic switching of tasks during multi-task training allows the model to choose the best next task to run based on the current state (as well as the previous curriculum path) of the model, as opposed to a fixed/static single mixing ratio for the full training period. In Fig. 3, we visualize the (moving averages of) probabilities of selecting each task, which shows that in the 0-1000 #rounds range, the bandit initially gives higher weight to the main task, but gradually redistributes the probabilities to the auxiliary tasks; and beyond 1000 #rounds, it then alternates switching among the three different tasks periodically. We also experimented with replacing the bandit controller with random task choices, and our bandit-controller achieves statistically significantly better results than this approach in both SARI and FKGL with $p < 0.01$, which shows that the path learned by the bandit controller is meaningful.

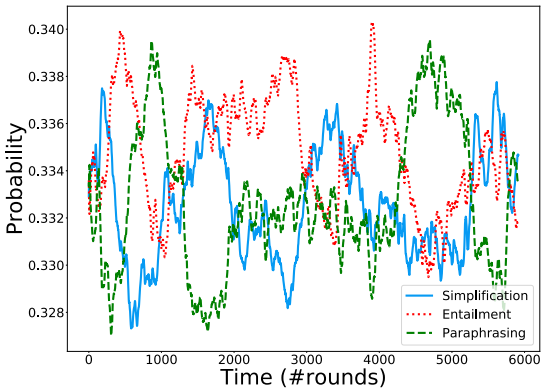


Figure 3: Task selection probability over training trajectory, predicted by bandit controller.

Multi-Task Learning vs. Data Augmentation To verify that our improvements come indeed from the auxiliary tasks' specific character/capabilities and not just due to adding more data, we separately trained word embeddings on each auxiliary dataset (i.e., SNLI+MultiNLI and ParaNMT) and incorporated them into the primary simplification model. We found that both our 2-way multi-task models perform stat. significantly better than these models (which use the auxiliary word-embeddings), suggesting that merely adding more data is not enough. Moreover, Table 5 shows that only specific intuitive (syntactic vs. semantic) layer sharing between the primary and auxiliary tasks helps results and not just adding data.

Output Examples Fig. 4 shows two output examples comparing DRESS-LS, pointer baseline, and multi-task models (and reference). We see that our multi-task model simplifies the input appropriately (similar extent to reference) while also keeping reasonably important information from the source. The pointer baseline and the DRESS-LS models simplify to a lesser extent and keep much more of the original input (as also suggested by our match-with-input investigation in Table 4).

Input: *he put henson in charge of escorting his slaves to his brother 's kentucky plantation .*
Reference: *he sent henson to take his slaves to kentucky .*
DRESS-LS: *he put henson in charge of escorting his slaves to his brother 's kentucky plantation .*
Baseline: *he put his slaves to his brother 's kentucky plantation .*
Multi-Task: *he put henson in charge of escorting .*

Input: *northern states did not allow slavery , but escaped slaves were returned to their owners as property , so henson would have to flee to canada to be free .*
Reference: *states in the north did not allow slavery .*
DRESS-LS: *southern states did not allow slavery , but the guatemalans were returned to their owners as property .*
Baseline: *he slaves were returned to their owners as property .*
Multi-Task: *northern states did not allow slavery .*

Figure 4: Output examples comparing DRESS-LS, our pointer baseline, and multi-task model.

7 Conclusion

We presented a multi-level, multi-task learning approach to incorporate natural language inference and paraphrasing knowledge into sentence simplification models, via soft sharing at higher-level semantic and lower-level lexico-syntactic levels. We also introduced a multi-armed bandits approach for learning a dynamic mixing ratio of tasks. We demonstrated strong simplification improvements on three standard datasets via automatic and human evaluation, and also discussed several ablation and analysis studies.

Acknowledgments

We thank the reviewers for their helpful comments (and Xingxing Zhang for providing preprocessed datasets). This work was supported by DARPA (YFA17-D17AP00022), Google Faculty Research Award, Bloomberg Data Science Research Grant, and NVidia GPU awards. The views contained in this article are those of the authors and not of the funding agency.

References

Marcelo Amancio and Lucia Specia. 2014. An analysis of crowdsourced text simplifications. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 123–130.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002a. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256.

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. 2002b. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77.

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

Regina Barzilay and Kathleen R McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th annual meeting on Association for Computational Linguistics*, pages 50–57. Association for Computational Linguistics.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? *arXiv preprint arXiv:1704.03471*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *ACL (1)*, pages 1415–1425.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122.

John A Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying text for language-impaired readers. In *EACL*, pages 269–270.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, pages 95–133. Springer.

Raman Chandrasekar, Christine Doran, and Bangalore Srinivas. 1996. Motivations and methods for text simplification. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 1041–1044. Association for Computational Linguistics.

Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1657–1668.

James Clarke and Mirella Lapata. 2006. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 377–384. Association for Computational Linguistics.

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

William Coster and David Kauchak. 2011. Learning to simplify sentences using wikipedia. In *Proceedings of the workshop on monolingual text-to-text generation*, pages 1–9. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Siobhan Lucy Devlin. 1999. *Simplifying natural language for aphasic readers.* Ph.D. thesis, University of Sunderland.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Richard Evans, Constantin Orasan, and Iustin Dornescu. 2014. An evaluation of syntactic simplification rules for people with autism. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 131–140.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1608–1618.

Katja Filippova and Michael Strube. 2008. Dependency tree based sentence compression. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 25–32. Association for Computational Linguistics.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *HLT-NAACL*, pages 758–764.

Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Goran Glavaš and Sanja Štajner. 2015. Simplifying lexical simplification: Do we need simplified corpora. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 63–68.

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. 2017. Automated curriculum learning for neural networks. *arXiv preprint arXiv:1704.03003*.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Soft, layer-specific multi-task summarization with entailment and question generation. In *Proceedings of ACL*.

Sanda Harabagiu and Andrew Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *ACL*, pages 905–912.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *EMNLP*.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a lexical simplifier using wikipedia. In *ACL (2)*, pages 458–463.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. 2016. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117.

William Hwang, Hannaneh Hajishirzi, Mari Ostendorf, and Wei Wu. 2015. Aligning sentences from standard wikipedia to simple wikipedia. In *NAACL-HLT*, pages 211–217.

Kentaro Inui, Atsushi Fujita, Tetsuro Takahashi, Ryu Iida, and Tomoya Iwakura. 2003. Text simplification for reading assistance: a project note. In *Proceedings of the second international workshop on Paraphrasing-Volume 16*, pages 9–16. Association for Computational Linguistics.

Sergio Jimenez, George Duenas, Julia Baquero, Alexander Gelbukh, Av Juan Dios Bátiz, and Av Mendizábal. 2014. UNAL-NLP: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *In SemEval*, pages 732–742.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. 2017. One model to learn them all. *arXiv preprint arXiv:1706.05137*.

Nobuhiro Kaji, Daisuke Kawahara, Sadao Kurohash, and Satoshi Sato. 2002. Verb paraphrase based on case frame alignment. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 215–222. Association for Computational Linguistics.

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *ACL (1)*, pages 1537–1546.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Technical Training Command Millington TN Research Branch.

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

Beata Beigman Klebanov, Kevin Knight, and Daniel Marcu. 2004. Text simplification for information-seeking applications. *Lecture Notes in Computer Science*, pages 735–747.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Alice Lai and Julia Hockenmaier. 2014. Illinois-lh: A denotational and distributional approach to semantics. *Proc. SemEval*, 2:5.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

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

Shashi Narayan and Claire Gardent. 2014. Hybrid simplification using deep semantics and machine translation. In *ACL*.

Eric W Noreen. 1989. *Computer-intensive methods for testing hypotheses*. Wiley New York.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Ramakanth Pasunuru and Mohit Bansal. 2017. Multi-task video captioning with video and entailment generation. In *Proceedings of ACL*.

Ramakanth Pasunuru, Han Guo, and Mohit Bansal. 2017. Towards improving abstractive summarization via entailment generation. In *NFiS@EMNLP*.

Sarah E Petersen and Mari Ostendorf. 2007. Text simplification for language learners: a corpus analysis. In *Workshop on Speech and Language Technology in Education*.

Luz Rello, Ricardo Baeza-Yates, and Horacio Saggion. 2013. The impact of lexical simplification by verbal paraphrases for people with and without dyslexia. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 501–512. Springer.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017. Sluice networks: Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Matthew Shardlow. 2014. A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications*, 4(1):58–70.

Sahil Sharma and Balaraman Ravindran. 2017. Online multi-task learning using active sampling. *CoRR*, abs/1702.06053.

Advaith Siddharthan. 2006. Syntactic simplification and text cohesion. *Research on Language and Computation*, 4(1):77–109.

Advaith Siddharthan. 2014. A survey of research on text simplification. *ITL-International Journal of Applied Linguistics*, 165(2):259–298.

Lucia Specia. 2010. Translating from complex to simplified sentences. *Computational Processing of the Portuguese Language*, pages 30–39.

Sanja Štajner, Hannah Béchara, and Horacio Saggion. 2015. A deeper exploration of the standard pb-smt approach to text simplification and its evaluation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

David Vickrey and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *ACL*, pages 344–352.

John Wieting and Kevin Gimpel. 2017a. Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. *CoRR*, abs/1711.05732.

John Wieting and Kevin Gimpel. 2017b. Revisiting recurrent networks for paraphrastic sentence embeddings. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the conference on empirical methods in natural language processing*, pages 409–420. Association for Computational Linguistics.

Kristian Woodsend and Mirella Lapata. 2014. Text rewriting improves semantic role labeling. *Journal of Artificial Intelligence Research*, 51:133–164.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *ACL*.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association of Computational Linguistics*, 3(1):283–297.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. *arXiv preprint arXiv:1703.10931*.

Congle Zhang, Stephen Soderland, and Daniel S Weld. 2015. Exploiting parallel news streams for unsupervised event extraction. *Transactions of the Association for Computational Linguistics*, 3:117–129.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd international conference on computational linguistics*, pages 1353–1361. Association for Computational Linguistics.

A Appendix

A.1 Training Details

All LSTMs use hidden state size of 256. We train word vectors with embedding size of 128 with random initialization. We use gradient clipped norm of 2.0. Our model selection (tuning) criteria is based on the average of our 3 metrics (SARI, BLEU, 1/FKGL) on the validation set. The mixing ratios are α_{ss}:α_{eg}:α_{pp} = 6:1:3 for Newsela, 6:1:3 for WikiSmall, and 7:2:1 for WikiLarge. The soft-sharing coefficient λ is set such that we balance the cross-entropy and regularization losses (at convergence), which is 5×10^{-6} for Newsela, 1×10^{-6} WikiSmall, and 1×10^{-5} for WikiLarge. We train models from scratch for Newsela and WikiSmall (using Adam (Kingma and Ba, 2014) optimizer with learning rate of 0.002 and 0.0015, respectively). However, because of the large size and computation overhead for WikiLarge, we first pre-train both main and auxiliary models on their own domain until they reach 90%

convergence, and use these models to initialize the multi-task models, and set the learning rate to $1/10$ of its original default value (0.001). We set the decay rate α in the bandit controller to be 0.3. We use the negative validation loss as the reward at each sampling step to the bandit algorithm. The validation loss is divided by two as a smoothing technique.[16] All our soft/hard and layer-specific sharing decisions (Sec. 6) were made on the validation/dev set. We follow previous work (Zhang and Lapata, 2017) in their pre-processing and post-processing of named entities. We capped vocabulary size to be 50K and replaced less frequent words with UNK token.[17] Unlike previous work (Zhang and Lapata, 2017), we do not use UNK-replacement at test time, but instead rely on our pointer-copy mechanism. We use beam search with beam size of 5. All other details provided in our released code.

[16] This constant serves the same purpose as the temperature variable in the softmax function.

[17] We measured the vocabulary overlap between the main and auxiliary tasks, and found that "word-form-overlap" (percentage of unique word types in auxiliary task that also appear in the main task) to be 40.7% (entailment) and 41.0% (paraphrase), and "word-count-overlap" (percentage of words in auxiliary task that also appear in the main task, based on token frequency counts) to be 95.2% (entailment) and 94.9% (paraphrase). Hence, this suggests that only rare words (which make up for very few counts) aren't considered in training process, and our pointer mechanism handles these extra UNK words by copying the actual word-form from the source to the output.

Interpretation of Implicit Conditions in Database Search Dialogues

Shun-ya Fukunaga[1] **Hitoshi Nishikawa**[1] **Takenobu Tokunaga**[1]
Hikaru Yokono[2] **Tetsuro Takahashi**[2]
[1]Tokyo Institute of Technology [2]Fujitsu Laboratories Ltd.
{fukunaga.s.ab@m, hitoshi@c, take@c}.titech.ac.jp
{yokono.hikaru, takahashi.tet}@jp.fujitsu.com

Abstract

Targeting the database search dialogue, we propose to utilise information in the user utterances that do not directly mention the database (DB) field of the backend database system but are useful for constructing database queries. We call this kind of information *implicit conditions*. Interpreting the implicit conditions enables the dialogue system more natural and efficient in communicating with humans. We formalised the interpretation of the implicit conditions as classifying user utterances into the related DB field while identifying the evidence for that classification at the same time. Introducing this new task is one of the contributions of this paper. We implemented two models for this task: an SVM-based model and an RCNN-based model. Through the evaluation using a corpus of simulated dialogues between a real estate agent and a customer, we found that the SVM-based model showed better performance than the RCNN-based model.

1 Introduction

The information that a dialogue system needs to extract from user utterances highly depends on its backend application. When being used as a natural language interface of a database system, the dialogue system should be able to extract pieces of information corresponding to the record fields of the database (DB) for constructing a query. There have been several attempts to extract this type of information from user utterances in database search dialogues. For instance, several studies (Raymond and Riccardi, 2007; Mesnil et al., 2015; Liu and Lane, 2016b) tried to extract values for the database field defined in the ATIS (The Air Travel Information System) corpus (Hemphill et al., 1990; Dahl et al., 1994) from the user utterances. The ATIS corpus includes a set of dialogues between users and an air travel system that were collected through the Wizard-of-Oz method. The tags that correspond to the backend database fields, e.g. departure city, arrival date, are annotated to the expressions in the user utterances. However, the utterances in real dialogues include information that does not always directly correspond to database fields but provides useful information for constructing database queries. It will be natural and more efficient if the dialogue system can utilise this type of information for retrieving the database. For instance, in real estate search dialogues, which is our target domain, the number of family members provides useful information for deciding the size of a house, but it is rarely a field of the real estate database as the number of family members is an attribute of the customer rather than that of houses. We call this type of information *implicit conditions* (Fukunaga et al., 2018).

To realise a dialogue system that can utilise the implicit conditions for the database search, we need to tackle the following two tasks.

(1) extracting pairs of a DB field and its value from user utterances that include the implicit conditions

(2) identifying the span in the utterance that represents the evidence for the DB field and value pair extraction

The task (1) provides useful information for efficiently constructing database queries from utterances that include the implicit conditions. The identified utterance span by the task (2) is helpful in constructing

Proceedings of the 27th International Conference on Computational Linguistics, pages 477–486
Santa Fe, New Mexico, USA, August 20-26, 2018.

clarification utterances by the system. As the task (1) requires a kind of inference to derive the DB field and value pair, the system interpretation is not necessarily correct. It will be safer and more natural to confirm the system interpretation with providing the evidence of that interpretation through clarification utterances. For instance, in the following dialogue fragment, the system inferred the DB field "number of rooms" and its value "one" from the user utterance. To ensure that this interpretation is correct, the system can add the evidence of the interpretation in the clarification utterance. This system utterance with the evidence is more natural than that without the evidence, i.e. "One bedroom place is large enough, isn't it?".

...

U: I will live alone.

S: As you live alone, one bedroom place is large enough, isn't it?

...

In such case, identifying that the span "live alone" is the evidence for extracting the DB field "number of rooms" and its value is worthwhile for constructing the clarification utterance.

We tackle the task (1) in two steps: extracting the DB field and extracting its value. In this paper, we first formalise the DB field extraction as a classification problem of user utterances into the DB fields. As we can derive multiple DB fields from a single utterance, the DB field extraction can be formalised as a multi-label classification. Furthermore, we solve the DB field classification together with the task (2), the evidence identification by adopting the method proposed by Lei et al. (2016). We put the DB field value extraction for the future work, as it requires the information of the database structure and its contents.

2 Related Work

The traditional pipeline for task-oriented dialogue systems consists of four modules: Natural Language Understanding (NLU), Dialogue State Tracking, Policy Learning and Natural Language Generation (Chen et al., 2017). The NLU task can be further divided into two tasks: intent detection and slot-filling. The intent detection classifies user utterances into the categories of user intention, e.g. request, question, inform and so on. The slot-filling extracts the semantic contents of user utterances in the form of slot-value pairs, e.g. the slot-value pairs From-Location="New York" and To-Location="Chicago" are extracted from "I'm going to go to Chicago from New York.". Regarding this general framework, our task corresponds to the slot-filling task. The slot-filling task can be formalised as a sequential labelling problem where each word in the target utterance is assigned an IOB tag of semantic slots (Ramshaw and Marcus, 1995). Recently, many studies adopt Recurrent Neural Network (RNN)(Mesnil et al., 2013; Yao et al., 2013; Mesnil et al., 2015; Vu et al., 2016; Jaech et al., 2016; Liu and Lane, 2016b; Liu and Lane, 2016a; Bapna et al., 2017) and Long Short-Term Memory (LSTM) (Yao et al., 2014; Hakkani-t et al., 2016) for the sequential labelling to achieve better performance. These methods, however, cannot identify semantic slots from the implicit condition which we are targeting, because they capture only explicitly mentioned semantic slots through the sequential labelling.

There are very few studies which address the task of predicting the users implicit condition in the task-oriented dialogue system. Celikyilmaz et al. (2012) proposed the method which predicts the user's preferred movie genre in the movie search domain. They estimate the movie genre that the user prefers from the user utterances that do not necessarily mention the movie genre. For instance, they estimate the comedy genre from the utterance "I wanna watch a movie that will make me laugh". Although the motivation of their work is almost the same as ours, their method estimates only a single attribute, i.e. the movie genre, and does not present the evidence for that estimate. Our method provides the evidence of the estimate at the same time and we deal with multiple attributes simultaneously.

In recent years, the end-to-end task-oriented dialogue systems which directly generate system utterances from user utterances are proposed. Eric et al. (2017) proposed an end-to-end dialogue system which accepts queries for the underlying knowledge base and returns the entities related to the users

Location	*Building*	*Facilities*	*Property*
available_railway_lines	building_age	room_facilities	property_type
walking_distance_to_a_station	floor_plan	air_conditioning	rent
nearest_station_facility	floor_area	storage	price
zone	room_placement_in_the_building	bathroom	conditions_for_rent
surrounding_facilities	room_size	kitchen	available_date
land_characteristics	building_structure	TV_and_Internet	target_demographic
distance_to_a_specific_place	sunlight		status
	building_facilities		appearance
	security_system		ownership
	number_of_storeys		available_discount
	number_of_households		subsidy
	renovation		certificate
			warranty

Table 1: Database (DB) field tags

utterance. Their system works without an explicit NLU module. As is often the case for end-to-end systems, their system does not concern the reason why the system returns a certain result to a given query, i.e it does not provide any explanation for the response. Given a user utterance including implicit conditions, our system makes a kind of inference to construct a query to the DB. As there is no guarantee that the system always can make a correct inference, we need to prepare the explanation for the system inference or clarification to the user. Such functions are necessary for natural and efficient dialogues. They are, however, difficult to implement in the current end-to-end framework.

3 Data and Task Setting

We use a Japanese dialogue corpus developed by (Takahashi and Yokono, 2017) in this work. The corpus includes dialogues between pairs of crowd workers who play a real estate agent and their customer each. The dialogues were collected through a keyboard chat system. The goal of the dialogues is finding a dwelling that fulfils the customer needs. The agent does not search in a real database but completes the dialogue when having acquired the necessary information for search. In each dialogue, a customer was assigned one of ten predefined profiles and was instructed to interact with the agent regarding their assigned profile. The customer profile was not open to the agent. An example profile looks like "You are moving to a new place from your current studio apartment to live with your long-standing boyfriend. On this occasion, you want to improve your cooking skills, and thus you prefer a place equipped with an easy-to-use kitchen with a multi-burner range."

The corpus includes 968 dialogues and 29,058 utterances consisting of 14,571 agent utterances and 14,487 customer utterances. The average number of utterances in a dialogue is 29.5.

We assigned the DB field tag shown in Table 1 to each utterance in our past research (Fukunaga et al., 2018). The DB field tag set was designed based on the search conditions that are available for one of the established search sites for real estate in Japan[1]. In addition to these 38 tags, we defined the Other tag that does not fit into any of the DB fields, expecting that utterances with the Other tag contain the implicit conditions. When the annotators assigned the Other tag to an utterance, they were instructed to describe its content in the free format.

The task of the present work is to categorise customer utterances with the Other tag into the 38 DB field tags and to identify the span representing the evidence of that classification at the same time. The target customer utterances are coupled with their preceding consecutive agent utterances[2]. The customer utterance with the Other tag sometimes contains only confirmation, e.g. "Yes, please" for the agent question "You live with your wife?". To collect necessary information for classification in such cases, we add the preceding agent utterances to the target customer utterance. We call this a target utterance chunk. There are 2,642 target utterance chunks in total.

[1] http://suumo.jp

[2] When the target customer utterance with the Other tag is preceded by customer utterances without the Other tag, they are also included in the chunk.

4 Methods

We propose two models for our current task: a Support Vector Machine (SVM)-based model and a Recurrent Convolutional Neural Network (RCNN)-based model.

4.1 SVM-based model

We use SVM with a linear kernel for the first model. We build an SVM binary classifier for each DB field tag that judges whether the input target utterance chunk includes the information for that DB field or not. Thus we have 38 classifiers in total. Given an utterance chunk, the final output of the system is the list of the DB field tags of which classifier returns a positive judgement. As features for SVM, we use a bag-of-words in the utterance chunk. We ran the Japanese morphological analyser MeCab[3] on all utterance chunks and extracted nouns, verbs, adjectives and adverbs appearing more than once in the corpus. We replaced the occurrences of numbers and proper nouns with their abstraction symbol NUM and PROP. The dimension of the feature vector is 1,730.

To identify the evidence span in the utterance chunk, we collect the words of which learnt weight exceeds a certain threshold.

4.2 RCNN-based model

We extend the method proposed by Lei et al. (2016) to solve our task. Given a product review text as input, their method estimates the user rating for each aspect of the product by using regression and specify the text span for the evidence of that regression result. Their system consists of two components: a neural network for regression (Encoder) and the other neural network for identifying the evidence (Generator). These two components are trained simultaneously. The encoder is trained through supervised training being given correct ratings while the generator is trained through unsupervised training. They designed a loss function for the generator so that it prefers a shorter and consecutive span for the evidence. Assuming that the generator correctly identifies the evidence span, the encoder uses only words in the evidence span to estimate the rating. They aim at improving the generator performance indirectly through improving the encoder performance that is trained by supervised training.

To adopt Lei et al. (2016)'s method for our current task, we redesign the loss function for the encoder. We use the cross-entropy instead of the square error for the encoder loss function, as our task is the binary classification while Lei et al. (2016)'s is the regression. Let $\tilde{y}_i = \mathrm{enc}_i(\boldsymbol{z}_i, \boldsymbol{x}) \in \{0, 1\}$ is the result of classification for the i-th DB field tag where $\boldsymbol{x}$ is the input word sequence with length m and $\boldsymbol{z}_i = (z_{i1}, z_{i2}, \ldots, z_{im})$ is the binary vector denoting if the words are selected as the evidence. Note that $\mathrm{enc}_i(\cdot)$ uses only words in $\boldsymbol{x}$ of which corresponding values in $\boldsymbol{z}_i$ is one, and $\boldsymbol{z}_i$ is the output of the generator. Our loss function $\mathcal{L}_i(\boldsymbol{z}_i, \boldsymbol{x}, y_i)$ is defined in equation (1),

$$\mathcal{L}_i(\boldsymbol{z}_i, \boldsymbol{x}, y_i) = -(y_i \log(\tilde{y}_i) + (1 - y_i) \log(1 - \tilde{y}_i)), \tag{1}$$

where y_i is the correct classification. The generator loss function is the same as the original as shown in equation (2).

$$\Omega_i(\boldsymbol{z}_i) = \lambda_1 \|\boldsymbol{z}_i\| + \lambda_2 \sum_{t=1}^{m} |z_{it} - z_{i(t-1)}|. \tag{2}$$

We set z_0 to zero. The first factor promotes the evidence span to be shorter and the second promotes the span to be consecutive. We have two hyper-parameters λ_1 and λ_2 to adjust the balance between these two factors. The total cost function for the i-th DB field is

$$\mathrm{cost}_i(\boldsymbol{z}_i, \boldsymbol{x}, y_i) = \mathcal{L}_i(\boldsymbol{z}_i, \boldsymbol{x}, y_i) + \Omega_i(\boldsymbol{z}_i). \tag{3}$$

[3] http://taku910.github.io/mecab/

5 Experiments

5.1 Data

We divided 986 dialogues of the corpus into ten sets with carefully keeping the distribution of the customer profiles uniform in each set. We use the nine sets for training and the rest one set for testing. We further extracted the utterance chunks as described in 3. We removed the chunks that include the target utterances at the dialogue management level, e.g. greeting, prompting and concluding utterances. The training set has 2,379 utterance chunks and the test set has 263 utterance chunks.

The user utterance in each target utterance chunk has been annotated with the Other tag and its content description in our past work as explained in 3. In the present work, we annotated each utterance chunk with the correct answer of the classification task by mapping its content description onto some of the DB field tags. One utterance chunk can be assigned with more than one tag. For instance, a content description, "the number of people to live", which is assigned to an utterance, "I will live alone.", is mapped onto floor_plan and floor_area. The evidence span for each assigned DB field tag is also annotated in the chunk. These annotation tasks are done by one of authors.

5.2 Evaluation Measures

We use precision, recall and F-measure for the evaluation of the DB field classification. To evaluate the evidence span identification, we calculate F-measure on words, and BLEU (Papineni et al., 2002) and ROUGE (Lin and Hovy, 2003). In the evaluation of the evidence span identification, we evaluated only cases where the DB field was correctly classified. Let binary vectors $\tilde{\boldsymbol{z}} = (\tilde{z_1}, \tilde{z_2}, \cdots, \tilde{z_m})$ and $\boldsymbol{z} = (z_1, z_2, \cdots, z_m)$ be the estimated and true evidence spans respectively, where the vector element one represents that the corresponding word is selected for constituting the evidence. Given $\tilde{\boldsymbol{z}}$ and $\boldsymbol{z}$, the F-measure on words can be calculated by equation (5).

$$\mathrm{TP} = \sum_{t=1}^{m} \tilde{z}_t z_t, \quad \mathrm{FP} = \sum_{t=1}^{m} \tilde{z}_t (1 - z_t), \quad \mathrm{FN} = \sum_{t=1}^{m} (1 - \tilde{z}_t) z_t \tag{4}$$

$$\mathrm{F} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{5}$$

For calculating BLEU and ROUGE, we define the sets of n-grams in the estimated and true evidence span as $\tilde{G_n}$ and G_n respectively in equation (6) and (7).

$$\tilde{G}_n = \left\{ \{j\}_{j=t}^{t+n-1} \middle| \prod_{j=t}^{t+n-1} \tilde{z}_j = 1,\ 1 \le t \le m - n + 1 \right\} \tag{6}$$

$$G_n = \left\{ \{j\}_{j=t}^{t+n-1} \middle| \prod_{j=t}^{t+n-1} z_j = 1,\ 1 \le t \le m - n + 1 \right\} \tag{7}$$

Using these n-gram sets, we can calculate BLEU and ROUGE as the geometric means of P_n and Q_n as shown in equation (8) and (9). We use uni-grams and bi-grams for calculating BLEU and ROUGE, i.e. $n \le 2$.

$$\mathrm{BLEU} = \left(\prod_{n=1}^{2} P_n \right)^{1/2}, \quad P_n = \begin{cases} \dfrac{|\tilde{G}_n \cap G_n|}{|\tilde{G}_n|} & (|\tilde{G}_n| > 0 \wedge n = 1) \\ \dfrac{|\tilde{G}_n \cap G_n| + 1}{|\tilde{G}_n| + 1} & (\text{otherwise}) \end{cases} \tag{8}$$

$$\mathrm{ROUGE} = \left(\prod_{n=1}^{2} Q_n \right)^{1/2}, \quad Q_n = \begin{cases} \dfrac{|\tilde{G}_n \cap G_n|}{|G_n|} & (|G_n| > 0 \wedge n = 1) \\ \dfrac{|\tilde{G}_n \cap G_n| + 1}{|G_n| + 1} & (\text{otherwise}) \end{cases} \tag{9}$$

	surrounding_facilities			floor_plan			floor_area		
Model	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure
SVM	**0.789**	**0.796**	**0.793**	**0.918**	**0.865**	**0.891**	**0.891**	**0.874**	**0.882**
RCNN	0.777	0.770	0.773	0.881	0.856	0.868	0.873	0.864	0.868

Table 2: Result of the DB field classification

	surrounding_facilities			floor_plan			floor_area		
Model	F-measure on words	BLEU	ROUGE	F-measure on words	BLEU	ROUGE	F-measure on words	BLEU	ROUGE
SVM_1	0.514	0.768	0.467	0.440	**0.808**	0.364	0.420	0.763	0.345
SVM	**0.530**	**0.787**	0.552	**0.533**	0.802	0.467	**0.507**	**0.773**	0.462
RCNN	0.458	0.534	**0.576**	0.452	0.625	**0.475**	0.436	0.521	**0.485**

Table 3: Result of the evidence span identification

In equation (8) and (9), we add a smoothing constant. We do not introduce the brevity penalty into BLEU as we also use ROUGE in this evaluation.

5.3 Evaluation Settings

Due to the limitation of the data size, we evaluate the three DB fields: surrounding_facilities, floor_plan and floor_area, which are the three most dominant tags. The numbers of utterance chunks in the training set that are assigned these tags are 1,033 for surrounding_facilities, 974 for floor_plan and 964 for floor_area.

The RCNN-based model has two hyper-parameters λ_1 and λ_2 that control the balance between the span length and the constraints on the word sequence. To decide the values for these hyper-parameters, we randomly chose 200 utterance chunks from the training set of surrounding_facilities to make a development set, and trained the model with the rest of the training set and evaluated it with the development set. As a result, we found that $\lambda_1 = 0.021$ and $\lambda_2 = 0.003$ made the best performance on the development set. We use these hyper-parameter values for the other two DB fields. We also used the parameter values of the neural networks learnt in the development set as the initial parameter values. We used the existing word embeddings made from the Japanese Wikipedia articles[4]. The dimension size is 200.

The SVM-based model also has a hyper-parameter, i.e. the threshold weight for the evidence span identification. The words that have a higher weight than the threshold constitute the evidence. We used the same development set as the RCNN-based model to estimate this threshold. The word weight is normalised by equation (10). The estimated threshold was 0.58.

$$\hat{w} = \frac{w - w_{\min}}{w_{\max} - w_{\min}} \tag{10}$$

5.4 Experimental Results and Discussion

Table 2 and Table 3 shows the results of the DB field classification and the evidence span identification. Table 3 also includes the result of the SVM-based model that adopts a single word with the highest weight for the evidence (SVM_1). In both tasks, the SVM-based model outperformed the RCNN-based model.

DB field classification

Table 2 indicates that the SVM-based model is superior to the RCNN-based model in all evaluation measures. The RCNN-based model is trained to classify the utterance into a DB field by utilising only the fragments in the utterance identified as the classification evidence. That means the performance of the DB field classification depends on that of the evidence span identification. As shown in Table 3,

[4] http://www.cl.ecei.tohoku.ac.jp/~m-suzuki/jawiki_vector/

DB field	surrounding_facilities		floor_plan		floor_area	
overlap	TP	FN	TP	FN	TP	FN
Yes	73	17	86	10	75	11
No	14	9	3	5	14	3

Table 4: Categorisation of cases based on word overlap between the correct and predicted evidence spans

the word-wise F-measure of the evidence span identification is less than 0.5. This suggests that the RCNN-based model has to solve the DB field classification by using insufficient information, i.e. wrong evidence span.

To investigate the relation between the classification performance and the evidence spans, we counted the number of true-positive (TP) and false-negative (FN) cases in the result of the DB field classification by the RCNN-model. Table 4 shows the breakdown of the cases according to whether the predicted and correct evidence spans share at least one word. The row "Yes" shows the numbers of cases where at least one word is shared by both predicted and correct evidence spans and "No" shows that for no word overlap. For instance, in surrounding_facilities, 73 out of 90 (81%) "Yes" cases are correct (TP) while 14 out of 23 (60%) "No" cases are correct. This tendency is the same as in the other two DB fields. Table 4 suggests that the performance of the evidence span identification has a significant impact on that of the DB field classification in the RCNN-model.

However, Table 4 also shows that there are cases where the RCNN-model can correctly classify the DB fields even though it fails to identify the correct evidence spans. We investigated such 14 cases for the surrounding_facilities field (the cell intersecting "No" and "TP") to find words that are useful for the field classification but not included in the evidence spans. For instance, the word "near" appears in the positive examples twice as frequent as in the negative examples in our training data. This word can be a clue for the field classification although not being included in the evidence span. To improve the performance of the DB field classification, we need to extend the current model to incorporate such useful information on top of the information from the evidence spans.

We further investigated the individual error cases of the surrounding_facilities classification. We found errors due to the annotation inconsistency in 10 out of 25 FP cases in the RCNN-based model result and 11 out of 24 FP cases in the SVM-based model result. For instance, an utterance chunk in FP cases, "Do you have any preferences for location? — I prefer a suburb.", is annotated as a negative example, while the utterance chunk which has almost the same content, "What kind of property are you looking for? — I plan to move to a suburb in future.", is annotated as a positive example. When we assign the DB field to the utterance with the Other tag we regarded only the description for the Other tag, without referring to the original utterances. In the above example, the content description of the former utterance chunk is "rough area", while that of the latter is "living environment". Although the content of the utterances is the same, the difference of their description caused such inconsistent field assignment. Our future work includes implementing more solid annotation guideline that enables stable and consistent assignment of the correct DB field.

Evidence span identification

Table 3 shows that the SVM-based model is superior to the RCNN-based model in all DB fields and measures except for ROUGE.

Table 5 shows the average number of words and spans in the predicted and correct evidence of the evidence span identification.

As shown in Table 5, the number of words in the correct evidence spans is more than two on average. Therefore the SVM-based model selecting only one word as the evidence (SVM_1 in Table 3) cannot select enough words for the evidence. Setting an appropriate threshold on the learnt weight of each word (feature) improves both ROUGE and BLUE values. This implies that the learnt word weight can be a clue to choose words to constitute the evidence.

We formalised the evidence span identification as a binary classification of whether each word in the

	surrounding_facilities			floor_plan			floor_area		
	#words	#spans	#w/#s	#words	#spans	#w/#s	#words	#spans	#w/#s
SVM	1.60	1.54	1.04	2.48	2.33	1.07	2.49	2.32	1.07
RCNN	3.21	2.39	1.34	2.97	2.06	1.44	3.57	2.12	1.69
Correct Answer	3.42	1.23	2.78	4.58	1.20	3.81	4.58	1.20	3.81

Table 5: Size of evidence spans

			Correct Answer	
			Positive	Negative
Prediction	Positive	SVM	102	64
		RCNN	92	144
	Negative	SVM	193	2,353
		RCNN	165	2,038

Table 6: Confusion matrix of the evidence span identification

utterance chunks belongs to the evidence or not. Table 6 shows the confusion matrix of the number of words constituting the evidence for the surrounding_facilities field. This matrix indicates that there are more false-negative cases than false-positive cases in both models. A dominant word type in the false-negative cases is Japanese particles, e.g. ga, ni, which indicate case marking. In other words, both models tend to skip these particles and to select the content words such as nouns and verbs. Although the particles are not important features of the classification since they appear in any class, it is desirable for dialogue systems to include them into the evidence so that they are used in the later stage of response generation, e.g. generating clarification utterances.

Through the investigation of the individual extracted evidence spans for the surrounding_facilities field by the RCNN-based model, we found that the model tends to extract the word just after the question mark "?" for all test cases. The question mark is the last word of the question by the real estate agent in most cases. The word "safety" which is the strong clue for surrounding_facilities appears just after the question mark at a rate of 41.3%. The model seems to have learnt this collocation. However, there are only 40.4% of test cases where the word just after a question mark belongs to the evidence span. This over-tuning degrades the performance of the evidence span identification.

The RCNN-based model is inferior to the SVM-based model in both tasks. Probably we need to have more data to train the neural network-based model; we used 2,379 utterance chunks for training classifiers while Lei et al. (2016) used about 80,000 to 90,000 reviews in the experiment.

6 Conclusion

Targeting the database search dialogue, we proposed to utilise information in the user utterances that do not directly correspond to the DB field of the backend database system. We call this kind of information *implicit conditions*. We formalised the interpretation of the implicit conditions as classifying user utterances into the related DB field while identifying the evidence for that classification at the same time. We implemented two models: an SVM-based model and an RCNN-based model. Through evaluation experiments on a corpus of simulated dialogues between a real estate agent and a customer, we found that the SVM-based model showed better performance than the RCNN-based model. This implies that the size of the corpus is too small to train the latter model. Furthermore, the error analysis on the DB field classification revealed that some errors occurred due to the annotation inconsistency. Our future work includes implementing more solid annotation guideline that enables stable and consistent assignment of the correct DB field.

While this paper focused on only finding DB field and its evidence from the utterance, we need to further extract the value for the field in order to generate database queries from the utterances.

References

Ankur Bapna, Gokhan Tur, Dilek Hakkani-tur, and Larry Heck. 2017. Sequential Dialogue Context Modeling for Spoken Language Understanding. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL 2017)*, pages 103–114.

Asli Celikyilmaz, Dilek Hakkani-tür, and Gokhan Tur. 2012. STATISTICAL SEMANTIC INTERPRETATION MODELING FOR SPOKEN LANGUAGE UNDERSTANDING WITH ENRICHED SEMANTIC FEATURES. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 216–221.

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *SIGKDD Explor. Newsl.*, 19(2):25–35, November.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 43–48.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D Manning. 2017. Key-Value Retrieval Networks for Task-Oriented Dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL 2017)*, pages 37–49.

Shun-ya Fukunaga, Hitoshi Nishikawa, Takenobu Tokunaga, Hikaru Yokono, and Tetsuro Takahashi. 2018. Analysis of implicit conditions in database search dialogues. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 2741–2745.

Dilek Hakkani-t, Gokhan Tur, Asli Celikyilmaz, Yun-nung Chen, Jianfeng Gao, Li Deng, and Ye-yi Wang. 2016. Multi-Domain Joint Semantic Frame Parsing Using Bi-Directional RNN-LSTM. In *INTERSPEECH-2016*, pages 715–719.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '90, pages 96–101.

Aaron Jaech, Larry Heck, and Mari Ostendorf. 2016. Domain Adaptation of Recurrent Neural Networks for Natural Language Understanding. In *INTERSPEECH-2016*, pages 690–694.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 107–117.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 71–78.

Bing Liu and Ian Lane. 2016a. Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling. In *INTERSPEECH-2016*, pages 685–689.

Bing Liu and Ian Lane. 2016b. Joint Online Spoken Language Understanding and Language Modeling with Recurrent Neural Networks. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL 2016)*, pages 22–30.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of Recurrent-Neural-Network Architectures and Learning Methods for Spoken Language Understanding. In *INTERSPEECH-2013*, pages 3771–3775.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, and Geoffrey Zweig. 2015. Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Christian Raymond and Giuseppe Riccardi. 2007. Generative and discriminative algorithms for spoken language understanding. In *Eighth Annual Conference of the International Speech Communication Association*, pages 1605–1608.

Tetsuro Takahashi and Hikaru Yokono. 2017. Two persons dialogue corpus made by multiple crowd-workers. In *Proceedings of the 8th International Workshop on Spoken Dialogue Systems (IWSDS 2017)*. 6 pages.

Ngoc Thang Vu, Pankaj Gupta, Heike Adel, and Hinrich Schutze. 2016. BI-DIRECTIONAL RECURRENT NEURAL NETWORK WITH RANKING LOSS FOR SPOKEN LANGUAGE UNDERSTANDING. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 6060–6064.

Kaisheng Yao, Geoffrey Zweig, Mei-yuh Hwang, Yangyang Shi, and Dong Yu. 2013. Recurrent Neural Networks for Language Understanding. In *INTERSPEECH-2013*, pages 2524–2528.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. SPOKEN LANGUAGE UNDERSTANDING USING LONG SHORT-TERM MEMORY NEURAL NETWORKS. In *Spoken Language Technology Workshop (SLT)*, pages 189–194.

Few-Shot Charge Prediction with Discriminative Legal Attributes

Zikun Hu* **Xiang Li*** **Cunchao Tu** **Zhiyuan Liu†** **Maosong Sun**
Department of Computer Science and Technology, Tsinghua University
State Key Lab on Intelligent Technology and Systems, Tsinghua University
Beijing National Research Center for Information Science and Technology
{hzk14, x-l15}@mails.tsinghua.edu.cn
tucunchao@gmail.com, {liuzy, sms}@tsinghua.edu.cn

Abstract

Automatic charge prediction aims to predict the final charges according to the fact descriptions in criminal cases and plays a crucial role in legal assistant systems. Existing works on charge prediction perform adequately on those high-frequency charges but are not yet capable of predicting few-shot charges with limited cases. Moreover, these exist many confusing charge pairs, whose fact descriptions are fairly similar to each other. To address these issues, we introduce several discriminative attributes of charges as the internal mapping between fact descriptions and charges. These attributes provide additional information for few-shot charges, as well as effective signals for distinguishing confusing charges. More specifically, we propose an attribute-attentive charge prediction model to infer the attributes and charges simultaneously. Experimental results on real-work datasets demonstrate that our proposed model achieves significant and consistent improvements than other state-of-the-art baselines. Specifically, our model outperforms other baselines by more than 50% in the few-shot scenario. Our codes and datasets can be obtained from https://github.com/thunlp/attribute_charge.

1 Introduction

The task of automatic charge prediction aims to train a machine judge to determine the final charges (e.g., *theft*, *robbery* or *traffic offence*.) of the defendants in criminal cases. As a representative subtask of legal judgment prediction, charge prediction plays an important role in legal assistant systems and can benefit many real-world applications. For example, it can provide a handy reference for legal experts (e.g., lawyers and judges) and improve their working efficiency. Meanwhile, it can supply ordinary people who are unfamiliar with legal terminology and complex procedures with legal consulting.

As a typical task in legal intelligence, automatic charge prediction has been studied for decades and most existing works formalize this task under the text classification framework. At the early stage, researchers pay great efforts to extract efficient features from text or case profiles. For example, some works (Liu et al., 2004; Liu and Hsieh, 2006) utilize shallow textual features, including characters, words, and phrases, to predict charges. Katz et al. (2017) predict the US Supreme Court's decisions with efficient features extracted from case profiles (e.g., dates, locations, terms, and types). All these approaches require numerous human effort to design features and annotate training instances. Besides, these methods are hard to scale to other scenarios. Inspired by the successful usage of deep neural networks on natural language processing tasks (Kim, 2014; Baharudin et al., 2010; Tang et al., 2015), researchers propose to employ deep neural networks to model legal documents. For example, Luo et al. (2017) propose an attention-based neural network for charge prediction by incorporating the relevant law articles.

However, charge prediction is still confronted with two major challenges which make it non-trivial:

*Indicates equal contribution.
†Corresponding Author.

Proceedings of the 27th International Conference on Computational Linguistics, pages 487–498
Santa Fe, New Mexico, USA, August 20-26, 2018.

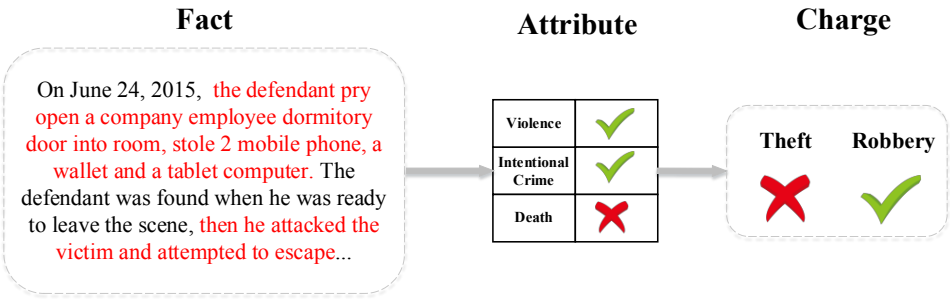


Figure 1: An illustration of the attribute-based charge prediction.

Few-Shot Charges. In practice, the case numbers of various charges are highly imbalanced. According to our statistics on a real-world dataset, the most frequent 10 charges (e.g., *theft*, *intentional injury*, and *traffic offence*.) cover 78.1% cases. On the contrary, the most low-frequency 50 (e.g., *scalping relics*, *disrupting the order of the court*, and *tax-escaping*.) charges only cover less than 0.5% cases and most of these charges own only around ten cases correspondingly. Previous works usually focus on these common charges and ignore the few-shot ones. Though deep neural models advance feature-engineering based charge prediction methods, they are unable to handle few-shot charges well due to the requirement of sufficient training data. Therefore, how to deal with these charges with limited cases is critical to a robust and effective charge prediction system.

Confusing Charges. Besides few-shot charges, there also exist many confusing charge pairs, such as (*theft*, *robbery*) and (*misappropriation of funds*, *embezzlement*). For each confusing pair, the definitions of two charges only differ in the verification of a specific act and the circumstances in corresponding cases are usually similar to each other. As illustrated in Fig. 1, many *robbery* case also contain the act of *theft*, and the existence of violence is the only key factor to distinguish these two charges. Thus, how to capture the crucial factors for distinguishing confusing charges is another challenge of charge prediction.

To address these issues, we propose to introduce discriminative legal attributes of charges into consideration and take these attributes as the internal mapping between facts and charge. More specifically, we select 10 representative attributes of charges, including *violence*, *profit purpose*, *buying and selling* and so on. Afterwards, we conduct a low-cost category-level annotation, i.e., for each charge, we annotate the value (including *yes*, *no*, or *not available*) of each attribute. This annotation indicates if an attribute is the essential condition of a charge.

With the attribute annotation of charges, we propose a novel multi-task learning framework to predict the attributes and charges of each case simultaneously. In this model, we employ attribute attention mechanism to capture the critical factual information relevant to a specific attribute. After that, we combine these attribute-aware representations with an attribute-free fact representation to predict the final charges. There are two reasons for introducing legal attributes into our charge prediction model. On one hand, these attributes can provide explicit knowledge about how to distinguish confusing charges. On the other hand, these attributes are shared by all charges, and the knowledge can transfer from high-frequency charges to low-frequency ones. Even for the few-shot charges, we can learn an efficient attribute-aware representation for prediction.

To investigate the advantage of our model on handling few-shot and confusing charges, we conduct experiments on three real-world datasets of Chinese criminal cases. Experimental results demonstrate that our model significantly and consistently outperforms other state-of-the-art models on all datasets and evaluation metrics. It is worth noting that, our model outperforms other baselines by more than 50% for the few-shot charges.

To summarize, we make three main contributions as follows:

(1) We are the first to focus on the few-shot and confusing problems in charge prediction. To address these issues, we introduce legal attributes of charges into charge prediction task for the first time.

(2) We propose a novel multi-task learning framework to infer the attributes and charges of a case jointly. To achieve it, we employ attribute attention mechanism to learn attribute-aware fact representa-

tions.

(3) We conduct efficient experiments on several real-world datasets, and our model significantly outperforms other baselines and achieves more than 50% improvements for few-shot charges.

2 Related Work

2.1 Zero-Shot Classification

Our work is relevant to zero-shot classification in computer vision. Many attribute-based models have been proposed under this task since attributes are shared among different classes and can offer an intermediate representation. Lampert et al. (2014) introduces direct attribute prediction(DAP) and indirect attribute prediction(IAP), and proposes attribute classifiers which can be pre-trained and dont need retraining when finding new suitable object class. Akata et al. (2013) proposes to transform the task of attribute-based classification to the label-embedding task. Jayaraman and Grauman (2014) introduces a random forest method stressing the unreliability of attribute prediction for unseen classes. They also extend it to the few-shot scenario.

Other than attributes, other external information can also be introduced to promote zero-shot classification. Elhoseiny et al. (2014) makes use of text description of the class label to transfer knowledge between text features and visual features. Zero-shot learning has also been used in applications besides object recognition, such as activity recognition (Zellers and Choi, 2017) and event recognition (Wu et al., 2014).

2.2 Charge Prediction

Researchers in the legal area have been working on automatically making the legal judgment for a long time. Kort (1957) applies quantitative methods to predict judgment by calculation numerical values for factual elements. Nagel (1963) makes use of correlation analysis to make predictions for reapportioning cases. Keown (1980) introduced mathematical models used for legal prediction such as linear models and the scheme of nearest neighbors. These methods are usually mathematical or quantitative, and they are restricted to a small dataset with few labels.

Since machine learning has been proven successful in many areas, researchers begin to formalize charge prediction as a text classification problem and make use of machine learning methods. Such work usually focuses on feature extraction from the case fact. Lin et al. (2012) fetches 21 legal factor labels for case classification. Mackaay and Robillard (1974) extracts N-grams and topics created by clustering semantically similar N-grams as features. Sulea et al. (2017) proposes a system based on SVM ensembles using the case description, ruling and time span of a case as input. However, these methods only extract shallow text features or manual labels which are hard to gather on a larger dataset. Whats more, the conventional models could not catch the subtle difference between similar crimes, thus they wouldnt perform well when the number of classes increases and more similar crimes appear.

With the successful usage of neural network methods on speech (Mikolov et al., 2011; Hinton et al., 2012; Dahl et al., 2012; Sainath et al., 2013), computer vision (CV) (Krizhevsky et al., 2012; Farabet et al., 2013; Tompson et al., 2014; Szegedy et al., 2015) and natural language processing (NLP) (Collobert et al., 2011; Kim, 2014; Bordes et al., 2014; Sutskever et al., 2014; Jean et al., 2015; Yang et al., 2016), researchers propose to employ neural models for legal tasks. Luo et al. (2017) proposes a hierarchical attentional network to predict charges and extract relevant articles jointly. However, this work only focuses on high-frequency charges, without paying attention to few-shot and confusing ones. To address these issues, we propose an attention-based neural model by incorporating several discriminative legal attributes.

3 Method

In this section, we propose a few-shot neural model which jointly models charge prediction task and legal attribute prediction task in a unified framework. In the following parts, we first introduce the discriminative charge attributes. Afterward, we give definitions of charge prediction and attribute prediction. Then

we describe the neural encoder of fact description and the attention-based attribute predictor. At last, we show the output layer and the loss function of our model.

3.1 Discriminative Charge Attributes

To distinguish confusing charges and provide additional knowledge for few-shot charges, we introduce 10 discriminative attributes for all the charges in Chinese criminal law. The detailed descriptions of these attributes are shown in Table 1. For each (*charge, attribute*) pair, it can be labeled as *Yes*, *No* or *NA*. For example, the charge of *manslaughter* should be labeled as *No* on *Intentional Crime*, *Yes* on *Death*, *NA* on *State Organ*. Note that, the fact-findings of a specific case can only be labeled as *Yes* or *No*. When convicting someone of a certain crime, the facts should conform to the description of the certain charge. Thus for a certain attribute, the label of a specific case and the label of the corresponding charge should be the same or not in conflict. In other words, for a certain attribute, the label of a case and the charge can only be (*Yes*, *Yes*), (*No*, *No*), (*Yes*, *NA*), or (*No*, *NA*). In practice, we conduct a low-cost annotation and annotate the attributes of 149 distinct charges manually. Then, we assign each case with the same attributes of its corresponding charge.

Attributes	Description
Profit Purpose	Whether the criminal commits a crime on the purpose of getting profit.
Buying and Selling	Whether the criminal has buying or selling behavior during the commission of the crime.
Death	Whether death is caused by the criminal.
Violence	Whether the criminal has the act of violence.
State Organ	Whether the case or the charge involves State organ or any functionary of a State organ.
Public Place	Whether the criminal commits a crime in a public place.
Illegal Possession	Whether the criminal commits a crime for the purpose of illegal possession.
Physical Injury	Whether a physical injury is caused by the criminal.
Intentional Crime	Whether the criminal commits an intentional crime.
Production	Whether the criminal commits a crime during the production.

Table 1: The descriptions of selected attributes.

3.2 Formalizations

3.2.1 Charge Prediction

The fact description of a case can be seen as a word sequence $\mathbf{x} = \{x_1, \ldots, x_n\}$, where n represents the sequence length, $x_i \in T$, and T is a fixed vocabulary. Given the fact description $\mathbf{x}$, the charge prediction task aims to predict a charge $y \in Y$ from a charge set Y.

3.2.2 Attributes Prediction

The attributes prediction task can be regarded as a binary classification task. It takes the same input sequence $\mathbf{x}$ as in the charge prediction task, and aims to predict the fact-findings of attributes $\mathbf{p} = \{p_1, \ldots, p_k\}$ according to the fact. Here, k is the number of selected attributes, and $p_i \in \{0, 1\}$ is the label for a certain attribute.

3.3 Fact Encoder

As illustrated in Fig. 2, fact encoder encodes the discrete input sequence into continuous hidden states. Here, we employ conventional Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) as fact encoder due to its ability to extract semantic meanings. LSTM is a variation of RNN and is capable of capturing long-term dependencies.

First, LSTM encoder converts each word $x_i \in \mathbf{x}$ into its word embedding $\mathbf{x}_i \in \mathbb{R}^d$, where d is the dimension of word embeddings. Then, we get the corresponding word embedding sequence as

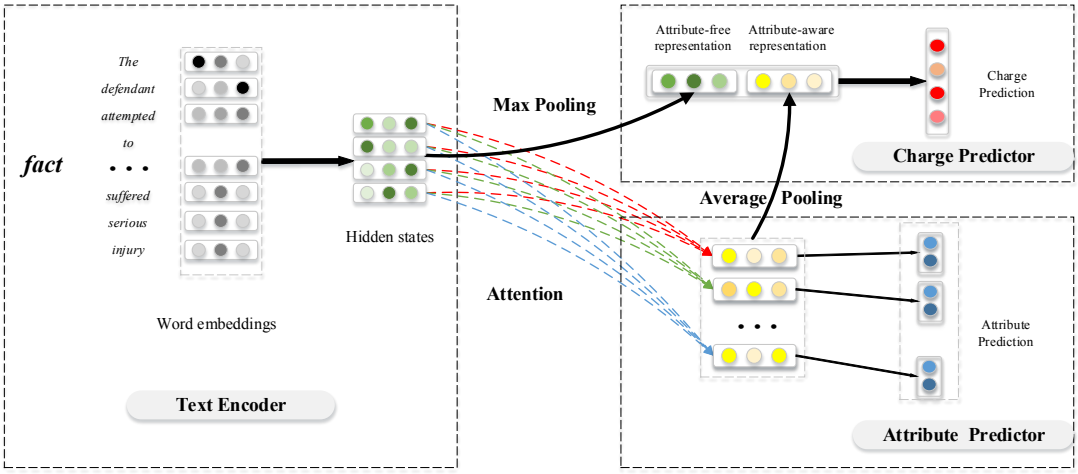


Figure 2: An illustration of the attribute-based charge prediction.

$\hat{\mathbf{x}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.

At each time step $t \in [1, n]$, the LSTM cell intakes $\mathbf{x}_t$, recalculates memory cell $\mathbf{c}_t$, and outputs new hidden state $\mathbf{h}_t$ as follows:

$$\begin{aligned}
\mathbf{f}_t &= \sigma\left(W_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f\right), \\
\mathbf{i}_t &= \sigma\left(W_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i\right), \\
\mathbf{o}_t &= \sigma\left(W_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o\right), \\
\hat{\mathbf{c}}_t &= \tanh\left(W_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c\right), \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t, \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right).
\end{aligned} \tag{1}$$

Here, $\mathbf{f}_t$, $\mathbf{i}_t$ and $\mathbf{o}_t$ represent forget gate, input gate, and output gate respectively. $\odot$ means element-wise multiplication and σ is the sigmoid activation function. W, U, and b are weight matrices and bias vectors. After processing all time steps, we get a hidden state sequence $\mathbf{h} = \{\mathbf{h}_1, \ldots, \mathbf{h}_n\}$. At last, we feed it into a max-pooling layer to get the attribute-free representation $\mathbf{e} = [e_1, \ldots, e_s]$ as

$$e_i = \max(\mathbf{h}_{1,i}, \cdots, \mathbf{h}_{n,i}), \forall i \in [1, s]. \tag{2}$$

Here, s is the dimension of hidden states.

3.4 Attentive Attribute Predictor

Given the fact description $\mathbf{x}$, the attribute predictor aims to predict the label of every attribute. Inspired by (Yang et al., 2016), we employ an attention mechanism to select relevant information from facts and generate attribute-aware fact representations.

As shown in Fig. 2, attribute predictor takes the hidden state sequence $\mathbf{h} = \{\mathbf{h}_1, \ldots, \mathbf{h}_n\}$ as input. Our attentive attribute predictor then calculates attention weights $\mathbf{a} = \{\mathbf{a}_1, \ldots, \mathbf{a}_k\}$ for all attribute, where $\mathbf{a}_i = [a_{i,1}, \ldots, a_{i,n}]$. $\forall i \in [1, k]$ and $j \in [1, n]$, $a_{i,j}$ is calculated by:

$$a_{i,j} = \frac{\exp(\tanh(\mathbf{W}^a \mathbf{h}_j)^T \mathbf{u}_i)}{\sum_t \exp(\tanh(\mathbf{W}^a \mathbf{h}_t)^T \mathbf{u}_i)}. \tag{3}$$

Here, $\mathbf{u}_i$ is the context vector of the i-th attribute to calculate how informative an element is to the attribute i, and $\mathbf{W}^a$ is a weight matrix that all attributes share. Afterwards, we get attribute-aware representations of fact $\mathbf{g} = \{\mathbf{g}_1, \ldots, \mathbf{g}_k\}$, where $\mathbf{g}_i = \sum_t a_{i,t} \mathbf{h}_t$. At last, with representations g, we project it into the label space and use softmax function to get the final prediction results $\mathbf{p} = [p_1, \ldots, p_k]$, where p_i is the prediction result of attribute i and is calculated by:

$$\begin{aligned}
\mathbf{z}_i &= \text{softmax}(\mathbf{W}_i^p \mathbf{g}_i + \mathbf{b}_i^p), \\
p_i &= \arg\max(\mathbf{z}_i).
\end{aligned} \tag{4}$$

Here, $\mathbf{z}_i$ is the prediction probability distribution on *Yes* and *No*. $\mathbf{W}_i^p$ and $\mathbf{b}_i$ are weight matrix and bias vector of attribute i.

3.5 Output Layer

To integrate the fact descriptions and fact-findings of all attributes, we use both attribute-free and attribute-aware representations to predict the final charge of a case in the output layer. The predicted distribution y over all charges is calculated as follows:

$$\begin{aligned} \mathbf{r} &= \frac{\sum_i \mathbf{g_i}}{k}, \\ \mathbf{v} &= \mathbf{e} \oplus \mathbf{r}, \\ y &= \text{softmax}(\mathbf{W}^y \mathbf{v} + \mathbf{b}^y). \end{aligned} \tag{5}$$

Here, $\mathbf{r}$ is the average of attribute-aware representations. $\mathbf{r}$ and $\mathbf{e}$ are concatenated into the final fact representation $\mathbf{v}$. $\mathbf{W}^y$ and $\mathbf{b}^y$ are weight matrix and bias vector in the output layer.

3.6 Optimization

The training objective of our proposed model consists of two parts. The first one is to minimize the cross-entropy between predicted charge distribution y and the ground-truth distribution $\hat{y}$. The other one is to minimize the cross-entropy between predicted distribution and the ground-truth fact-finding of each attribute.

The charge prediction loss can be formalized as:

$$\mathcal{L}_{charge} = -\sum_{i=1}^{C} y_i \cdot log(\hat{y}_i), \tag{6}$$

where y_i is the ground-truth label, $\hat{y}_i$ is prediction probability, and C is the number of charges.

As each attribute is equally important in the model, we can easily calculate the attribution loss by sum up the cross-entropy of all attributes. However, when the attribute of a specific charge is *NA*, the label of the corresponding cases can be *Yes* or *No*. Therefore, we only add up the cross-entropy to the attribute loss when this attribute of the charge belongs to *Yes* or *No*. At last, we formulate the attribute loss as:

$$\mathcal{L}_{attr} = -\sum_{i=1}^{k} I_i \sum_{j=1}^{2} z_{ij} \cdot log(\hat{z}_{ij}), \tag{7}$$

where I_i is an indicator function. $I_i = 1$ if the i-th attribute of current charge is labeled as *Yes* or *No*, and $I_i = 0$ otherwise. Obviously, z_i is the ground-truth label, and $\hat{z}_i$ is predicted probabilities distribution on *Yes* and *No*.

Considering the two objectives, our final loss function $\mathcal{L}$ is obtained by adding $\mathcal{L}_{charge}$ and $\mathcal{L}_{attr}$ as follows:

$$\mathcal{L} = \mathcal{L}_{charge} + \alpha \cdot \mathcal{L}_{attr}, \tag{8}$$

where α is a hyper-parameter to balance the weight of the two parts in the loss function.

4 Experiments

In order to investigate the effectiveness of our model on criminal charges prediction, we conduct experiments on several real-world datasets and compare our model with several state-of-the-art baselines.

4.1 Dataset Construction

Since there are no publicly available datasets in previous works for charges prediction, we collect criminal cases published by the Chinese government from China Judgments Online[1]. As each case is well-structured and divided into several parts such as fact, court view, and penalty result, we select the fact

[1] `http://wenshu.court.gov.cn`.

part of each case as our input. Besides, we can easily extract the charges from the penalty result by regular expression. We have manually checked the extracted charges and there are few mistakes.

Although there are some cases that contain multiple defendants and multiple charges in real-world, considering the task would be too complex to solve if these cases contained, we removed the cases which have more than one charges in a verdict. Besides, in order to examine the performance of our method on few-shot charges, we keep 149 distinct charges (near 3 times as compared with (Luo et al., 2017)) with at least 10 cases.

After preprocessing, we randomly select about $400,000$ cases and construct three datasets with different scales, denoted as **Criminal-S(small)**, **Criminal-M(medium)** and **Criminal-L(large)**. The three different datasets contain the same number of charges but the different number of cases. The detailed statistics are shown in Table 2.

Datasets	**Criminal-S**	**Criminal-M**	**Criminal-L**
train	$61,589$	$153,521$	$306,900$
test	$7,702$	$19,189$	$38,368$
valid	$7,755$	$19,250$	$38,429$

Table 2: The statistics of different datasets.

4.2 Attribute Selection and Annotation

As mentioned in previous part, we propose to introduce discriminative attributes to enhance charge prediction. To select these attributes, we first train a LSTM based charge prediction model and obtain the confusion matrix of predicted charges on validation set. Then, we filter out the confusing charge pairs and provide them to three master students majoring in criminal. According to these confusing charge pairs, they define 10 representative attributes to distinguish these confusing pairs.

With the selected 10 attributes, we conduct a low-cost annotation over all charges. Here, the low-cost annotation means we only need to annotate 10 attributes for 149 charges manually, rather than all cases. As the selected attributes are discriminative and unambiguous, we asked these annotators to reach an agreement for each annotation. Totally, we spent less than 10 hours for annotation.

4.3 Baselines

We employ several typical text classification models and one charge predicting model as baselines:

TFIDF+SVM: We implement term-frequency inverse document frequency (TFIDF) (Salton and Buckley, 1988) to extract features of inputs, and employ SVM (Suykens and Vandewalle, 1999) as the classifier.

CNN: We implement the CNN with multiple filter widths (Kim, 2014) as text classifier.

LSTM: We implement a two-layer LSTM (Hochreiter and Schmidhuber, 1997) with a max-pooling layer as the fact encoder.

Fact-Law Attention Model: Luo et al. (2017) propose an attention-based neural charge prediction model by incorporating relevant law articles.

4.4 Experiment Settings and Evaluation Metrics

As all the case documents are written in Chinese without word cutting, we employ THULAC (Sun et al., 2016) for word segmentation and set the maximum document length to 500. For the TFIDF+SVM model, we set the feature size to $2,000$. For other neural models, we employ Skip-Gram model (Mikolov et al., 2013) to pre-train word embeddings with the embedding size of 100. We set the hidden state size of LSTM to 100.For the CNN based models, we set the filter widths to $(2, 3, 4, 5)$ with each filter size to 25 for consistency. The weight α of the attribute loss is set to 1.

Note that, the representation size of our model turns into 200 after concatenation. For a fair comparison, we add a 100×200 fully connected layer between after the pooling layer in CNN and LSTM,

denoted as CNN-200 and LSTM-200.

We use Adam (Kingma and Ba, 2015) as the optimizer, and set the learning rate to 0.001, the dropout rate (Srivastava et al., 2014) to 0.5 and the batch size to 64. We employ accuracy (Acc.), macro-precision (MP), macro-recall (MR) and macro-F1 as our evaluation metrics.

4.5 Results and Analysis

Datasets	Criminal-S				Criminal-M				Criminal-L			
Metrics	Acc.	MP	MR	F1	Acc.	MP	MR	F1	Acc.	MP	MR	F1
TFIDF+SVM	85.8	49.7	41.9	43.5	89.6	58.8	5 0.1	52.1	91.8	67.5	54.1	57.5
CNN	91.9	50.5	44.9	46.1	93.5	57.6	48.1	50.5	93.9	66.0	50.3	54.7
CNN-200	92.6	51.1	46.3	47.3	92.8	56.2	50.0	50.8	94.1	61.9	50.0	53.1
LSTM	93.5	59.4	58.6	57.3	94.7	65.8	63.0	62.6	95.5	69.8	67.0	66.8
LSTM-200	92.7	60.0	58.4	57.0	94.4	66.5	62.4	62.7	95.1	72.8	66.7	67.9
Fact-Law Att.	92.8	57.0	53.9	53.4	94.7	66.7	60.4	61.8	95.7	73.3	67.1	68.6
Our Model	**93.4**	**66.7**	**69.2**	**64.9**	**94.4**	**68.3**	**69.2**	**67.1**	**95.8**	**75.8**	**73.7**	**73.1**

Table 3: Charge prediction results of three datasets.

As shown in Table 3, we can observe that our model significantly and consistently outperforms all the baselines. Almost all existing methods perform poorly under the macro-F1 metric, which indicates their shortage of predicting few-shot charges. Conversely, our model achieves promising improvements (7.9%, 4.4%, and 5.2% absolutely on three datasets respectively), which demonstrates the robustness and effectiveness of our model.

To further verify the advance of our model on dealing with few-shot charges, we show the performance on charges with different frequencies. As shown in Table 4, we divide the charges into three parts according to their frequencies. Here, the charges with ≤ 10 cases are low-frequency, and the charges with > 100 cases are high-frequency. From this table, we find that our model achieves more than 50% improvements than baseline method for the low-frequency (i.e., few-shot) charges, which verifies the effectiveness of our model on handling few-shot issues.

Charge Type	Low frequency	Medium frequency	High frequency
Charge Number	49	51	49
LSTM-200	32.6	55.0	83.3
Our Model	**49.7 (↑ 17.1%)**	**60.0 (↑ 5.0%)**	**85.2 (↑ 1.9%)**

Table 4: Macro-F1 values of various charges on Criminal-S.

Datasets	Criminal-S				Criminal-M				Criminal-L			
Metrics	Acc.	MP	MR	F1	Acc.	MP	MR	F1	Acc.	MP	MR	F1
Our model	93.4	**66.7**	**69.2**	**64.9**	94.4	68.3	**69.2**	**67.1**	**95.8**	**75.8**	**73.7**	**73.1**
w/o attention	**93.5**	63.4	60.1	60.0	94.7	**68.8**	58.2	60.9	94.9	70.9	54.4	58.6
w/o concatenation	**93.5**	59.3	59.0	57.2	**95.0**	64.6	62.4	62.5	95.7	69.4	64.5	65.4

Table 5: Experimental results of ablation test.

4.6 Ablation Test

Our method is characterized by the incorporation of attention mechanism and attribute-aware representations. Thus, we design ablation test respectively to investigate the effectiveness of these modules. When taken off the attention mechanism, for each attribute we replace attention mechanism with a fully connected layer. When taken off the attribute-aware representations (i.e., without concatenating the averaged

Task	Charge Prediction	Attribute Prediction on *physical injury*
Ground Truth	Intentional Injury	Yes
Our model	Intentional Injury	Yes
LSTM-200	Affray	N/A

Table 6: Charge and attribute prediction result of the selected case.

attribute-aware representation), our method degrades into a typical multi-task learning based on LSTM for both charge and attribute prediction.

As shown in Table 5, we can observe that the performance drops obviously after removing the attention layer or the concatenation. The macro-F1 decreases at least 4%. Therefore, it can be seen that both attention mechanism and attribute-aware fact representation play irreplaceable roles in our model.

4.7 Case Study

In this part, we select a representative case to give an intuitive illustration of how the predicted attributes help to promote the performance of charge prediction. In this case, the defendant is convicted of intentional injury. It is often hard to decide whether to judge a case as affray or intentional injury since they are both related to violence. One important difference between them is that intentional injury has the feature of physical injury, while affray does not.

So we believe the attribute physical injury is essential in the charge prediction of this case. As shown in Table 6, our model correctly predicts the label of *physical injury* as *Yes*, and consequently predicts the charge as *intentional injury*. In contrary, the model LSTM-200 predicts it as *affray* incorrectly. In addition, we visual the heat map of this case when predicting the attribute $intentionalinjury$. Words with deeper background color have higher attention weights. From this figure, we observe that the attention mechanism can capture key patterns and semantics relevant to current attribute.

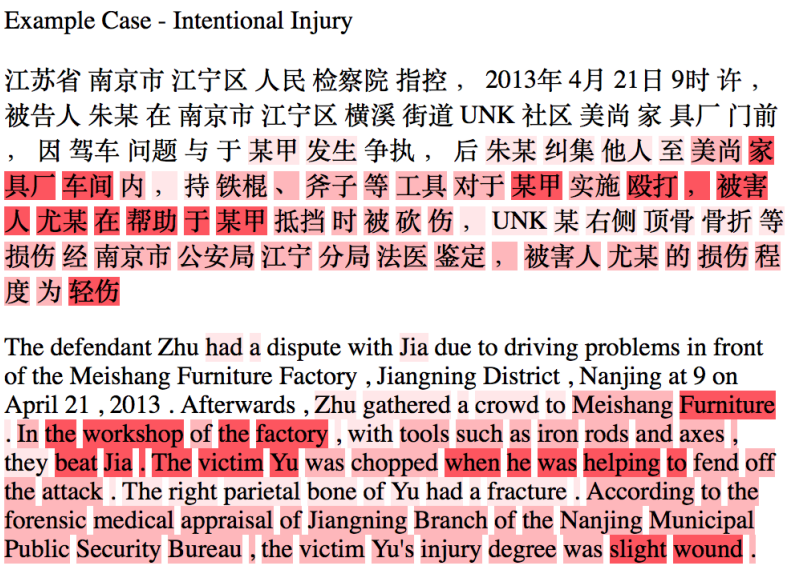


Figure 3: Visualization of attention mechanism.

.

5 Conclusion

In this work, we focus on the task of charge prediction according to the fact descriptions of criminal cases. To address the problem of prediction few-shot and confusing charges, we introduce discriminative legal

attributes into consideration and propose a novel attribute-based multi-task learning model for charge prediction. Specifically, our model learns attribute-free and attribute-aware fact representation jointly by utilizing attribute-based attention mechanism.

In future, we will explore the following directions:

(1) There are more complicated criminal cases, such as multiple defendants and charges. Thus, it is challenging to handle this general form of charge prediction.

(2) In this work, we only utilize several simple attributes of charges, while there exist more complex essential conditions of charges. How to take full usage of essential conditions of charges is expected to improve the interpretability of charge prediction models.

Acknowledgements

We thank all the anonymous reviewers for their insightful comments. This work is supported by the National Natural Science Foundation of China (NSFC No. 61661146007, 61572273) and Tsinghua University Initiative Scientific Research Program (20151080406). This research is part of the NExT++ project, supported by the National Research Foundation, Prime Ministers Office, Singapore under its IRC@Singapore Funding Initiative.

References

Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. 2013. Label-embedding for attribute-based classification. In *Proceedings of CVPR*, pages 819–826.

Baharum Baharudin, Lam Hong Lee, and Khairullah Khan. 2010. A review of machine learning algorithms for text-documents classification. *JAIT*, 1(1):4–20.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of EMNLP*, pages 615–620.

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

George E Dahl, Dong Yu, Li Deng, and Alex Acero. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE TASLP*, 20(1):30–42.

Mohamed Elhoseiny, Babak Saleh, and Ahmed Elgammal. 2014. Write a classifier: Zero-shot learning using purely textual descriptions. In *Proceedings of ICCV*, pages 2584–2591.

Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. 2013. Learning hierarchical features for scene labeling. *IEEE TPAMI*, 35(8):1915–1929.

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

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

Dinesh Jayaraman and Kristen Grauman. 2014. Zero-shot recognition with unreliable attributes. In *Proceedings of NIPS*, pages 3464–3472.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of ACL*, volume 1, pages 1–10.

Daniel Martin Katz, Michael J Bommarito II, and Josh Blackman. 2017. A general approach for predicting the behavior of the supreme court of the united states. *PloS one*, 12(4):e0174698.

R Keown. 1980. Mathematical models for legal prediction. *Computer/Law Journal*, 2:829.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*, pages 1746–1751.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Fred Kort. 1957. Predicting supreme court decisions mathematically: A quantitative analysis of the ”right to counsel” cases. *American Political Science Review*, 51(1):1–12.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of NIPS*, pages 1097–1105.

Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. 2014. Attribute-based classification for zero-shot visual object categorization. *IEEE TPAMI*, 36(3):453–465.

Wan-Chen Lin, Tsung-Ting Kuo, Tung-Jia Chang, Chueh-An Yen, Chao-Ju Chen, and Shou-de Lin. 2012. Exploiting machine learning models for chinese legal documents labeling, case classification, and sentencing prediction. In *Processdings of ROCLING*, pages 140–141.

Chao-Lin Liu and Chwen-Dar Hsieh. 2006. Exploring phrase-based classification of judicial documents for criminal charges in chinese. In *Proceedings of ISMIS*, pages 681–690.

Chao-Lin Liu, Cheng-Tsung Chang, and Jim-How Ho. 2004. Case instance generation and refinement for case-based criminal summary judgments in chinese. *JISE*, pages 783–800.

Bingfeng Luo, Yansong Feng, Jianbo Xu, Xiang Zhang, and Dongyan Zhao. 2017. Learning to predict charges for criminal cases with legal basis. In *Proceedings of EMNLP*, pages 2727–2736.

Ejan Mackaay and Pierre Robillard. 1974. Predicting judicial decisions: The nearest neighbour rule and visual representation of case patterns. *Datenverarbeitung im Recht*, 3(3/4):302–331.

Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. 2011. Strategies for training large scale neural network language models. In *Proceedings of ASRU workshop*, pages 196–201.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Stuart S Nagel. 1963. Applying correlation analysis to case prediction. *Texas Law Review*, 42:1006.

Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. 2013. Deep convolutional neural networks for lvcsr. In *Proceedings of ICASSP*, pages 8614–8618.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.

Octavia Maria Sulea, Marcos Zampieri, Mihaela Vela, and Josef Van Genabith. 2017. Exploring the use of text classi cation in the legal domain. In *Proceedings of ASAIL workshop*.

Maosong Sun, Xinxiong Chen, Kaixu Zhang, Zhipeng Guo, and Zhiyuan Liu. 2016. THULAC: An efficient lexical analyzer for chinese.

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

Johan AK Suykens and Joos Vandewalle. 1999. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, Jen-Hao Rick Chang, et al. 2015. Going deeper with convolutions. In *Proceedings of CVPR*, pages 1–9.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of EMNLP*, pages 1422–1432.

Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. 2014. Joint training of a convolutional network and a graphical model for human pose estimation. In *Proceedings of NIPS*, pages 1799–1807.

Shuang Wu, Sravanthi Bondugula, Florian Luisier, Xiaodan Zhuang, and Pradeep Natarajan. 2014. Zero-shot event detection using multi-modal fusion of weakly supervised concepts. In *Proceedings of CVPR*, pages 2665–2672.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of NAACL*, pages 1480–1489.

Rowan Zellers and Yejin Choi. 2017. Zero-shot activity recognition with verb attribute induction. In *Proceedings of EMNLP*, pages 946–958.

Can Taxonomy Help? Improving Semantic Question Matching using Question Taxonomy

Deepak Gupta[*], **Rajkumar Pujari**[†], **Asif Ekbal**[*], **Pushpak Bhattacharyya**[*],
Anutosh Maitra[‡] **Tom Jain**[‡] and **Shubhashis Sengupta**[‡]
[*]Indian Institute of Technology Patna, India
[†]Purdue University, USA
[‡]Accenture Labs, Bengaluru, India
[*]{deepak.pcs16,asif,pb}@iitp.ac.in, [†]rpujari@purdue.edu
[‡]{anutosh.maitra,tom.geo.jain,shubhashis.sengupta}@accenture.com

Abstract

In this paper, we propose a hybrid technique for semantic question matching. It uses a proposed two-layered taxonomy for English questions by augmenting state-of-the-art deep learning models with question classes obtained from a deep learning based question classifier. Experiments performed on three open-domain datasets demonstrate the effectiveness of our proposed approach. We achieve state-of-the-art results on partial ordering question ranking (POQR) benchmark dataset. Our empirical analysis shows that coupling standard distributional features (provided by the question encoder) with knowledge from taxonomy is more effective than either deep learning (DL) or taxonomy-based knowledge alone.

1 Introduction

Question Answering (QA) is a well investigated research area in Natural Language Processing (NLP). There are several existing QA systems that answer factual questions with short answers (Iyyer et al., 2014; Bian et al., 2008; Ng and Kan, 2015). However, systems which attempt to answer questions that have long answers with several well-formed sentences, are rare in practice. This is mainly due to some of the following challenges: (i) selecting appropriate text fragments from document(s), (ii) generating answer texts with coherent and cohesive sentences, (iii) ensuring the syntactic as well as semantic well-formedness of the answer text. However, when we already have a set of answered questions, reconstructing the answers for semantically similar questions can be bypassed. For each unseen question, the most semantically similar question is identified by comparing the unseen question with the existing set of questions. The question, which is closest to the unseen question can be retrieved as a possible semantically similar question. Thus, accurate semantic question matching can significantly improve a QA system. In the recent past, several deep learning based models such as recurrent neural networks (RNNs), convolution neural network (CNN), gated recurrent units (GRUs) etc. have been explored to obtain representation at the word (Mikolov et al., 2013; Pennington et al., 2014), sentence (Kim, 2014) and paragraph (Zhang et al., 2017) level.

In the proposed semantic question matching framework, we use attention based neural network models to generate question vectors. We create a hierarchical taxonomy by considering different types and subtypes in such a way that questions having similar answers belong to the same taxonomy class. We propose and train a deep learning based question classifier network to classify the taxonomy classes. The taxonomy information is helpful in taking a decision on semantic similarity between them. For example, the questions '*How do scientists work?*' and '*Where do scientists work?*', have very high lexical similarity but they have different answer types. This can be easily identified using a question taxonomy. Taxonomy can provide very useful information when we do not have enough data for generating useful deep learning based representations, which are generally the case with restricted domains. In such scenarios linguistic information obtained from the prior knowledge helps significantly in improving the performance of the system.

Proceedings of the 27th International Conference on Computational Linguistics, pages 499–513
Santa Fe, New Mexico, USA, August 20-26, 2018.

We propose a neural network based algorithm to classify the questions into appropriate taxonomy class(es). The information, thus obtained from taxonomy, is used along with the DL techniques to perform semantic question matching. Empirical evidence establishes that our taxonomy, when used in conjunction with Deep Learning (DL) representations, improves the performance of the system on semantic question (SQ) matching task.

We summarize the contributions of our work as follows: **(i)** We create a two-layered taxonomy for English questions; **(ii)** We propose a deep learning based method to identify taxonomy classes of questions; **(iii)** We propose a dependency parser based technique to identify the *focus* of the question; **(iv)** We propose a framework to integrate semantically rich taxonomy classes with DL based encoder to improve the performance and achieve new state-of-the-art results in semantic question ranking on benchmark dataset and Quora dataset; and finally **(v)** We release two annotated datasets, one for semantically similar questions and the other for question classification.

2 Related Works

Rapid growth of community question and answer (cQA) forums have intensified the necessity for semantic question matching in QA setup. Answer retrieval of semantically similar questions has drawn the attention of researchers in very recent times (Màrquez et al., 2015; Nakov et al., 2016). It solves the problem of *question starvation* in cQA forums by providing a semantically similar question which has already been answered. In literature, there have been attempts to address the problem of finding the most similar match to a given question, for e.g. Burke et al. (1997) and Mlynarczyk and Lytinen (2005). Wang et al. (2009) have presented syntactic tree based matching for finding semantically similar questions. 'Similar question retrieval' has been modeled using various techniques such as topic modeling (Li and Manandhar, 2011), knowledge graph representation (Zhou et al., 2013) and machine translation (Jeon et al., 2005). Semantic kernel based similarity methods for QA have also been proposed in (Filice et al., 2016; Croce et al., 2017; Croce et al., 2011).

Answer selection in QA forums is similar to the question similarity task. In recent times, researchers have been investigating DL-based models for answer selection (Wang and Nyberg, 2015; Severyn and Moschitti, 2015; Feng et al., 2015). Most of the existing works either focus on better representations for questions or linguistic information associated with the questions. On the other hand, the model proposed in this paper is a hybrid model. We also present a thorough empirical study of how sophisticated DL models can be used along with a question taxonomy concepts for semantic question matching.

3 Question Matching Framework

When framed as a computational problem, semantic question (SQ) matching for QA becomes equivalent to ranking questions in the existing question-base according to their semantic similarity to the given input question. Existing state-of-the-art systems use either deep learning models (Lei et al., 2016) or traditional text similarity methods (Jeon et al., 2005; Wang et al., 2009) to obtain the similarity scores. In contrast, our framework of SQ matching efficiently combines deep learning based question encoding and a linguistically motivated taxonomy. Algorithm 1 describes the precise method we follow. $Similarity(.)$ is the standard cosine similarity function. $fsim$ is focus embedding similarity which is described later in Section 4.4.

3.1 Question Encoder Model

Our question encoder model is inspired from the state-of-the-art question encoder architecture proposed by Lei et al. (2016). We extend the question encoder model of Lei et al. (2016) by introducing attention mechanism similar to Bahdanau et al. (2014) and Chopra et al. (2016). We propose the attention based version of two question encoder models, namely Recurrent Convolutional Neural Network (RCNN) (Lei et al., 2016) and Gated Recurrent Unit (GRU) (Chung et al., 2014; Cho et al., 2014).

A question encoder with attention does not need to capture the whole semantics of the question in its final representation. Instead, it is sufficient to capture a part of hidden state vectors of another question it needs to attend while generating the final representation. Let $\mathbf{H} \in \mathbb{R}^{d \times n}$ be a matrix consisting of

Algorithm 1 Semantic Question Matching

procedure SQ MATCHING(QSet)
  RESULTS ← {}
  for (p, q) in QSet **do**
    $\vec{p}, \vec{q} \leftarrow$ *Question-Encoder*(p, q)
    $sim \leftarrow$ `Similarity`($\vec{p}$, $\vec{q}$)
    $T_p^c, T_q^c \leftarrow$ `Taxonomy-Classes`(p, q)
    $F_p, F_q \leftarrow$ `Focus`(p, q)
    $\vec{F_p}, \vec{F_q} \leftarrow$ `Focus-Encoder`(F_p, F_q)
    $fsim \leftarrow$ `Similarity`($\vec{F_p}$, $\vec{F_q}$)
    Feature-Vector=[sim, ${T_p}^c$, ${T_q}^c$, $fsim$]
    result ← Classifier(Feature-Vector)
    RESULTS.append(result)
  return RESULTS

hidden state vectors $[h_1, h_2 \dots h_n]$ that the question encoder (RCNN, GRU) produced when reading the n words of the question, where d is a hyper parameter denoting the size of embeddings and hidden layers. The attention mechanism will produce an attention weight vector $\alpha_t \in \mathbb{R}^n$ and a weighted hidden representation $r_t \in \mathbb{R}^d$.

$$\begin{aligned} C_t &= tanh(W^H H + W^v(v_t \otimes I_n)) \\ \alpha_t &= softmax(w^T C_t) \\ r_t &= H\alpha^T \end{aligned} \tag{1}$$

where W^H, $W^v \in \mathbb{R}^{d \times d}$, are trained projection matrices. w^T is the transpose of the trained vector $w \in \mathbb{R}^d$. $v_t \in \mathbb{R}^d$ shows the embedding of token x_t and $I_n \in \mathbb{R}^n$ is the vector of 1. The product $W^v(v_t \otimes I_n)$ is repeating the linearly transformed v_t as many times (n) as there are words in the candidate question. Similarly we can obtain the attentive hidden state vectors $[r_1, r_2 \dots r_n]$. We apply the averaging pooling strategy to determine the final representation of the question.

Annotated data, $\mathscr{D} = \{(q_i, p_i^+, p_i^-)\}$ is used to optimize $f(p, q, \phi)$, where $f(.)$ is a measure of similarity between the questions p and q, and ϕ is a parameter to be optimized. Here p_i^+ and p_i^- correspond to the similar and non-similar question sets, respectively for question q_i. Maximum margin approach is used to optimize the parameter ϕ. For a particular training example, where q_i is similar to p_i^+, we minimize the max-margin loss $\mathcal{L}(\phi)$ defined as:

$$\mathcal{L}(\phi) = \max_{p \in Q'(q_i)} \left\{ f(q_i, p; \phi) - f(q_i, p_i^+; \phi) + \lambda(p, p_i^+) \right\} \tag{2}$$

where $Q'(q_i) = p_i^+ \cup p_i^-$, $\lambda(p, p_i^+)$ is a positive constant set to 1 when $p \neq p_i^+$, 0 otherwise.

3.2 Question Taxonomy

Questions are ubiquitous in natural language. Questions essentially differ on two fronts: semantic and syntactic. Questions that differ syntactically might still be semantically equivalent. Let us consider the following two questions:

- What is the number of new hires in 2018?
- How many employees were recruited in 2018?

Although the above questions are not syntactically similar but both are semantically equivalent and have the same answer. A well-formed taxonomy and question classification scheme can provide this information which eventually helps in determining the semantic similarity between the questions.

According to Gruber (1995), ontologies are commonly defined as specifications of shared conceptualizations. Informally, conceptualization is the relevant informal knowledge one can extract from their experience, observation or introspection. Specification corresponds to the encoding of this knowledge in representation language. In order to create a taxonomy for questions, we observe and analyze questions

Coarse Classes	Fine Classes
Quantification	Temprature, Time/Duration, Mass, Number, Age Distance, Money, Speed, Size, Percent, Rank/Rating
Entity	Person, Location, Organization, Animal, Technique Flora, Entertainment, Food, Abbreviation, Language Disease, Award/Title, Event, Sport/Game, Policy, Date Publication, Body, Thing, Feature/Attribute, Website Industry Sector, Monuments, Activity/Process, Other Tangible, Other Intangible
Definition	Person, Entity
Description	Reason, Mechanism, Cause & Effect, Describe Compare & Contrast, Analysis
List	Set of fine classes listed in the coarse classes *Quantification* and *Entity*
Selection	Alternative/Choice, True/False

Table 1: Set of proposed coarse and respective fine classes

from Stanford Question Answering Dataset (SQuAD) released by Rajpurkar et al. (2016) and question classifier data from Hovy et al. (2001) and Li and Roth (2002). The SQuAD dataset consists of 100,000+ questions and their answers, along with the text extracts from which the questions were formed. The other question classifier dataset contains $5,500$ questions. In the succeeding sub-section, we describe in details the coarse classes, fine classes and focus of a question. We have included an additional hierarchical taxonomy table with one example question for each class in the appendix section.

3.2.1 Coarse Classes

To choose the correct answer of a question one needs to understand the question and categorize the answer into the appropriate category which could vary from a basic implicit answer (question itself contains the answer) to a more elaborate answer (description). The coarse class of question provides a broader view of the expected answer type. We define the following six coarse class categories: *Quantification, Entity, Definition, Description, List* and *Selection*. *Quantification* class deals with the questions which look for a specific quantity as answer. Similarly *Entity, Definition, Description* class give the evidence that answer type will be entity, definition and a detail description, respectively. *Selection* class defines the question that looks for an answer which needs to be selected from the given set of answers. Few examples of questions along with their coarse class are listed here:

- **Quantity:** *Give the average speed of 1987 solar powered car winner?*
- **Entity:** *Which animal serves as a symbol throughout the book?*

3.2.2 Fine Classes

The coarse class defines the answer type at the broad level such as entity, quantity, description etc. But extracting the actual answer of question needs further classification into more specific answer types. Let us consider the following examples of two questions:

1. **Entity (Flora):** *What is one aquatic plant that remains submerged?*
2. **Entity (Animal):** *Which animal serves as a symbol throughout the book?*

Although both the questions belong to the same coarse class *entity* but they belong to the different fine classes, (*Flora* and *Animal*). Fine class of a question is based on the nature of the expected answer. It is useful in restricting the potential candidate matches. Although, questions belonging to the same fine class need not to be semantically same, questions belonging to the different fine classes rarely match. We show the set of the proposed coarse class and their respective fine classes in Table 1.

3.2.3 Focus of a Question

According to Moldovan et al. (2000), *focus* of a question is a word or a sequence of words, which defines the question and disambiguates it to find the correct answer the question is expecting to retrieve. In

the following example, *Describe the customer service model for Talent and HR BPO*, the term 'model' serves as the *focus*. As per Bunescu and Huang (2010b), *focus* of a question is contained within the noun phrases of a question. In the case of imperatives, the direct object (*dobj*) of the *question word* contains the *focus*. Similarly, in case of interrogatives, there are certain dependencies that capture the relation between the question word and its focus. The *dobj* relation of the root verb or *det* relation of *question word* for interrogatives contain the *focus*. Question word *how* has *advmod* relations that contain *focus* of the question. Priority order of the relations used to extract *focus* is obtained by observation on the SQuAD data. We depict the pseudo-code of the *focus* extraction method in the appendix section.

3.3 Question Classification

Question classification guides a QA system to extract appropriate candidate answer from the document/corpus. For example, the question *'How much does international cricket player get paid?'* should be accurately classified as the coarse class *quantification* and fine class *money* to further extract the appropriate answer. In our problem, we attempt to exploit the taxonomy information to identify the semantically similar questions. Therefore, the question classifier should be capable enough to accurately classify the coarse and fine classes of a reformulated question:

1. *What is the salary of an international level cricketer?*
2. *What is the estimated wage of an international cricketer?*

3.3.1 Question Classification Network

In order to identify the coarse and fine classes of a given question, we employ a deep learning based question classifier. In our question classification network CNN and bidirectional GRU has been applied sequentially. The obtained question vector is passed through a feed forward NN layer, and then through a softmax layer to obtain the final class of the question. We use two separate classifiers for coarse and fine class classification.

Firstly, an embedding layer maps a question $Q = [w_1, w_2 \dots w_n]$, which is a sequence of words w_i, into a sequence of dense, real-valued vectors, $E = [v_1, v_2 \dots v_n]$, $v_i \in \mathbb{R}^d$. Thereafter, a convolution operation is performed over the zero-padded sequence E^p. $F \in \mathbb{R}^{k \times m \times d}$, a set of k filters is applied to the sequence. We obtain convoluted features c_t at given time t for $t = 1, 2, \dots, n$.

$$c_t = tanh(F[v_{t-\frac{m-1}{2}} \dots v_t \dots v_{t+\frac{m-1}{2}}]) \tag{3}$$

Then, we generate the feature vectors $C' = [c'_1, c'_2 \dots c'_n]$, by applying max pooling on C. This sequence of convolution feature vector C' is passed through a bidirectional GRU network. We obtain the forward hidden states $\overrightarrow{h_t}$ and backward hidden states $\overleftarrow{h_t}$ at every step time t. The final output of recurrent layer h is obtained as the concatenation of the last hidden states of forward and backward hidden states.

Finally, the fixed-dimension vector h is fed into the softmax classification layer to compute the predictive probability $p(y = l|Q) = \frac{exp(w_l^T h + b_l)}{\sum_{i=1}^{L} exp(w_i^T h + b_i)}$ for all the question classes (coarse or fine). We assume there are L classes where w_x and b_x denote the weight and bias vectors, respectively and $x \in \{l, i\}$.

3.4 Comparison with Existing Taxonomy

In the Text REtrieval Conference (TREC) task, Li and Roth (2002) proposed a taxonomy to represent a natural semantic classification for a specific set of answers. This was built by analyzing the TREC questions. In contrast to Li and Roth (2002), along with TREC questions we also make a thorough analysis of the most recent question answering dataset (SQuAD) which has a collection of more diversified questions. Unlike Li and Roth (2002), we introduce the list and selection type question classes in our taxonomy. Each of these question types has its own strategy to retrieve an answer, and therefore, we put these separately in our proposed taxonomy. The usefulness of list as a different coarse class in semantic question matching can be understood considering the following questions:

1. *What are some techniques used to improve crop production?*
2. *What is the best technique used to improve crop production ?*

These two questions are not semantically similar as **(1)** and **(2)** belong to *list* and *entity* coarse classes, respectively. Moreover, Li and Roth (2002)'s taxonomy has overlapping classes (*Entity, Human and Location*). In our taxonomy we put all these classes in a single coarse class named *Entity*, which helps in identifying semantically similar questions better. We propose a set of coarse and respective fine classes with more coverage compared to Li and Roth (2002). Li and Roth (2002) taxonomy does not cover many important fine classes such as, *entertainment, award/title, activity, body* etc., under *entity* coarse class. We include these fine classes in our proposed taxonomy. We further redefine *description* type questions by introducing *cause & effect, compare and contrast* and *analysis* fine classes in addition to *reason, mechanism* and *description* classes. This finer categorization helps in choosing a more appropriate answer strategy for descriptive questions.

4 Experiments

4.1 Datasets

We perform experiments on three benchmark datasets, namely Partial Ordered Question Ranking (POQR)-Simple, POQR-Complex (Bunescu and Huang, 2010a) and Quora datasets. In addition to this, we also perform experiments on a new semantic question matching dataset (Semantic SQuAD[1]) created by us. In order to evaluate the system performance, we perform experiments in two different settings. The first setting deals with semantic question ranking (SQR) and the second deals with semantic question classification (SQC) with two classes (match and no-match). We perform SQR experiments on Semantic SQuAD and POQR datasets. For SQC experiments, we use Semantic SQuAD and Quora datasets.

4.1.1 Semantic SQuAD

We built a semantically similar question-pair dataset based on a portion of SQuAD data. SQuAD, a crowd-sourced dataset, consists of 100,000+ answered questions along with the text from which those question-answer pairs were constructed. We randomly selected $6,000$ question-answer pairs from SQuAD dataset and for a given question we asked 12 annotators[2] to formulate semantically similar questions referring to the same answers. Each annotator was asked to formulate 500 questions. We divided this dataset into training, validation and test sets of $2,000$ pairs each. We further constructed $4,000$ *semantically dissimilar* questions automatically. We use these $8,000$ question pairs ($4,000$ semantic similar questions pair from test and validation + $4,000$ semantically dissimilar pairs) to train the semantic question classifier for the SQC setting of the experiments. *Semantically dissimilar* questions are created by maintaining the constraint that questions should be from the different taxonomy classes. We perform 3-fold cross-validation on these $8,000$ question pairs.

4.1.2 POQR Dataset

POQR dataset consists of 60 groups of questions, each having a reference question that is associated with a partially ordered set of questions. Each group has three different sets of questions named as *paraphrase* ($\mathcal{P}$), *useful* ($\mathcal{U}$) and *neutral* ($\mathcal{N}$). For each given reference question q_r we have $q_p \in \mathcal{P}$, $q_u \in \mathcal{U}$, and $q_n \in \mathcal{N}$. As per Bunescu and Huang (2010a) the following two relations hold:

1. $(q_p \succ q_u | q_r)$: A *paraphrase* question is *'more useful than'* useful question.
2. $(q_u \succ q_n | q_r)$: A *useful* question is *'more useful than'* neutral question.

By transitivity, it was assumed by Bunescu and Huang (2010a) that the following ternary relation holds $(q_p \succ q_n | q_r)$: "A *paraphrase* question is *'more useful than'* a neutral question". We show the statistics of these datasets for *Simple* and *Complex* question types for two annotators (1, 2) in Table 2.

4.1.3 Quora Dataset

We perform experiments on semantic question matching dataset consisting of 404,290 pairs released by Quora[3]. The dataset consists of 149,263 matching pairs and 255,027 non-matching pairs.

[1]All the datasets used in the paper are publicly available at `https://figshare.com/articles/Semantic_Question_Classification_Datasets/6470726`

[2]The annotators are the post-graduate students having proficiency in English language.

[3]https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

Datasets	Simple		Complex	
	Simple-1	Simple-2	Complex-1	Complex-2
$\mathcal{P}$	164	134	103	89
$\mathcal{U}$	775	778	766	730
$\mathcal{N}$	594	621	664	714
Pairs	11015	10436	10654	9979

Table 2: Brief statistics of POQR datasets

4.2 Evaluation Scheme

We employ different evaluation schemes for our SQR and SQC evaluation settings. For the **Semantic SQuAD** dataset, we use the following metrics for ranking evaluation: Recall in top-k results (Recall@k) for $k = 1$, 3 and 5, Mean Reciprocal Rank (MRR) and Mean Average Precision (MAP). The set of all candidate questions in $2,000$ pairs of the test set is ranked against each input question. As we have only 1 correct match out of $2,000$ questions for each question in the test set, recall@1 is equivalent to precision@1. Given that we only have one relevant result for each input question, MAP is equivalent to MRR. We evaluate the semantic question classification performance in terms of accuracy. To ensure fair evaluation, we keep the ratio of semantically similar and dissimilar questions to be 1:1. In order to compare the performance on **POQR dataset** with the state-of-the art results, we followed the same evaluation scheme as described in Bunescu and Huang (2010a). It is measured in terms of 10-fold cross validation accuracy on the set of ordered pairs, and the performance is averaged between the two annotators (1,2) for the Simple and Complex datasets. For **Quora dataset**, we perform 3-fold cross validation on the entire dataset evaluating based on the classification accuracy only. We did not perform the semantic question ranking (SQR) experiment on Quora dataset as 149,263 $\times$ 149,263 ranking experiment for matching pairs takes a very long time.

4.3 Baselines

We compare our proposed approach to the following information retrieval (IR) based baselines:
1) TF-IDF: The candidate questions are ranked using cosine similarity value obtained from the TF-IDF based vector representation.
2) Jaccard Similarity: The questions are ranked using Jaccard similarity calculated for each candidate question with the input question.
3) BM-25: The candidate questions are ranked using BM-25 score, provided by Apache Lucene [4]

4.4 Experimental Setup

Question Encoder: We train two different question encoders (hidden size=300) on *Semantic SQuAD* and *Quora* datasets. For Semantic SQuAD dataset, we used $2,000$ training pairs to train the question encoder, as mentioned in Section 4.1.1. For Quora dataset we randomly selected $74,232$ semantically similar question pairs to train the encoder, and $10,000$ question pairs for validation. The best hyper-parameters for the deep learning based attention encoder are identified on validation data. Adam (Kingma and Ba, 2014) is used as the optimization method. Other hyper-parameters used are: learning rate (0.01), dropout probability (Hinton et al., 2012): (0.5), CNN feature width (2), batch size (50), epochs (30) and size of the hidden state vectors (300). This optimal hyper-parameter values are same for the attention based RCNN and GRU encoder. We train two different question encoders trained on *Semantic SQuAD* and *Quora* datasets. We could not train the question encoder on the **POQR dataset** because of the unavailability of sufficient amount of similar question pairs in this dataset. Instead we use the question encoder trained on the Quora dataset to encode the questions from POQR dataset.

Question Classification Network: To train the model we manually label (using 3 English proficient

[4] https://lucene.apache.org/core/

annotators with an inter-annotator agreement of 87.98%) a total of $5,162$ questions[5] with their coarse and fine classes, as proposed in Section 3.2. We release this question classification dataset to the research community. We evaluate the performance of question classification for 5-fold cross-validation in terms of F-Score. Our evaluation shows that we achieve 94.72% and 86.19% F-score on coarse class (6-labels) and fine class (72-labels), respectively. We use this trained model to obtain the coarse and fine classes of questions in all datasets.

We perform the SQC experiments with SVM classifier. We use *libsvm* implementation (Chang and Lin, 2011) with linear kernel and polynomial kernel of degree $\in \{2, 3, 4\}$. Best performance was obtained using linear kernel. Due to the nature of POQR dataset as described in Section $4.1.2$ in the paper we employ SVM^{light} [6] implementation of ranking SVMs, with a linear kernel keeping standard parameters intact. In our experiments, we use pre-trained Google embeddings provided by (Mikolov et al., 2013). The focus embedding is obtained through word vector composition (averaging).

5 Results and Analysis

5.1 Results

We present extensive results of semantic question ranking experiment on the Semantic SQuAD dataset in Table 4. In Tables 3, 4 and 5 the performnce results are reported on the respective dataset using the models **GRU**, **RCNN, GRU-Attention and RCNN-Attention** (c.f. Section 3.1). For all these models the results reported in the tables are based on the cosine similarity of the respective question encoder. The introduction of attention mechansim helps the question encoder in improving the performance. The attention based model obatins the maximum gains of 2.40% and 2.60% in terms of recall and MRR for the *GRU* model. The taxonomy augmented model outperforms the respective baselines and state-of-the-art deep learning question encoder models. We obtain the best improvements for the *Tax+RCNN-Attention* model, 3.75% and 4.15% in terms of Recall and MRR, respectively. Experiments show that taxonomy features assist in consistently improving the R@k and MRR/MAP across all the models.

Models	Simple			Complex		
	Simple-1	Simple-2	Overall	Complex-1	Complex-2	Overall
GRU (Lei et al., 2016)	74.20	73.68	73.94	74.67	75.22	74.94
RCNN (Lei et al., 2016)	76.19	75.81	76.00	75.33	76.44	75.88
GRU-Attention	75.39	74.83	75.11	76.22	76.18	76.20
RCNN-Attention	77.28	77.01	77.14	76.63	77.31	76.97
DNN + Taxonomy based Features						
Tax+GRU	78.29	79.01	78.65	77.63	78.97	78.30
Tax+RCNN	80.92	81.55	81.23	80.15	80.83	80.49
Tax+GRU-Attention	81.69	81.03	81.36	81.22	81.56	81.39
Tax+RCNN-Attention	83.67	83.98	83.82	83.32	84.10	83.71
State-of-the art techniques						
Unsupervised *Cos* (Bunescu and Huang, 2010a)	-	-	73.70	-	-	72.60
Supervised *SVM* (Bunescu and Huang, 2010a)	-	-	82.10	-	-	82.50

Table 3: Semantic question ranking performance of various models on **POQR datasets**. All the numbers shows is in terms of accuracy.

Performance of the proposed model on POQR dataset are shown in Table 3. The '*overall*' column in Table 3 shows the performance average on simple-1,2 and complex-1,2 datasets. We obtain improvements (maximum of 1.55% with *GRU-Attention* model on Complex-1 dataset) in each model by introducing attention mechanism on both simple and complex datasets. The augmentation of taxonomy

[5] $4,000$ questions are the training set of Semantic SQuAD. Remaining $1,162$ questions from the dataset used in Li and Roth (2002)

[6] http://svmlight.joachims.org/

features helps in improving the performance further (8.75% with *Tax+RCNN-Attention* model on Simple dataset).

The system performance on semantic question classification (SQC) experiment with Semantic SQuAD and Quora datasets are shown in Table 5. Similar to ranking results, we obtain significant improvement by introducing attention mechanism and augmenting the taxonomy features on both the datasets.

Models	R@1	R@3	R@5	MRR/MAP
IR based Baselines				
TF-IDF	54.75	66.15	70.25	61.28
Jaccard Similarity	48.95	62.80	67.40	57.26
BM-25	56.40	69.35	71.45	61.93
Deep Neural Network (DNN) based Techniques				
GRU (Lei et al., 2016)	73.25	84.12	86.39	76.77
RCNN (Lei et al., 2016)	75.10	86.35	89.01	78.24
GRU-Attention	74.89	86.02	88.47	78.30
RCNN-Attention	76.41	88.41	91.78	80.28
DNN + Taxonomy based Features				
Tax + GRU	76.19	87.02	88.47	78.98
Tax +RCNN	78.32	88.91	92.35	81.49
Tax + GRU-Attention	77.35	89.22	91.28	80.95
Tax + RCNN-Attention	78.88	90.20	93.25	83.12

Table 4: **Semantic Question Ranking (SQR)** performance of various models on **Semantic SQuAD** dataset, R@k and Tax denote the recall@k & augmentation of taxonomy features.

Models	Semantic SQuAD Dataset	Quora Dataset
IR based Baselines		
TF-IDF	59.28	70.19
Jaccard Similarity	55.76	67.11
BM-25	63.78	73.27
Deep Neural network (DNN) based Techniques		
GRU (Lei et al., 2016)	74.05	77.53
RCNN (Lei et al., 2016)	77.54	79.32
GRU-Attention	75.18	79.22
RCNN-Attention	79.94	80.79
DNN + Taxonomy based Features		
Tax + GRU	77.32	79.21
Tax + RCNN	79.89	81.15
Tax + GRU-Attention	78.11	80.91
Tax + RCNN-Attention	82.25	83.17

Table 5: **Semantic Question Classification (SQC)** performance of various models on **Semantic SQuAD** and **Quora** datasets.

Sr. No.	Datasets	All	-CC	-FC	-Focus Word
1	Semantic SQuAD (SQR)	83.12	81.66	81.84	82.20
2	Semantic SQuAD (SQC)	82.25	80.85	81.19	81.13
3	POQR-Simple	83.82	80.85	81.44	82.57
4	POQR-Complex	83.71	81.04	81.97	82.19
5	Quora	83.17	80.93	81.75	82.24

Table 6: Feature ablation results on all datasets. **SQR** results are in **MAP**. The others results are shown in terms of **Accuracy**.

5.2 Qualitative Analysis

We analyze the obtained results by studying the following effects:

(1) Effect of Attention Mechanism: We analyzed hidden state representation the model is attending to when it is deciding the semantic similarity. We depicted the visualization (**in appendix**) of attention weight between two semantically similar question from Semantic SQuAD dataset. We observed that the improvement due to the attention mechanism in Quora dataset is comparatively less than the Semantic SQuAD dataset. The question pairs from Quora dataset have matching words, and the problem is more focused on difference rather than similar or related word. For example, for the questions "*How magnets are made?*" and "*What are magnets made of?*", the key difference is question words 'how' versus 'what', while the remaining words are similar.

(2) Effect of Taxonomy Features: We performed feature ablation study on all the datasets to analyze the impact of each taxonomy features. Table 4 shows the results[7] with the full features and after removing coarse class (-CC), fine class (-FC) and focus features one by one. We observed from Quora dataset that the starting word of the questions *(what, why, how etc.)* is a deciding factor for semantic similarity. As the taxonomy features categorize these questions into different coarse and fine classes, therefore, it helps the system in distinguishing between semantically similar and dissimilar questions. It can be observed from the results that the augmentation of CC and FC features significantly improves the performance

[7]The results are statistically significant as $p < 0.002$.

especially on Quora dataset. Similar trends were also observed on the other datasets.

5.3 Comparison to State-of-the-Art

We compare the system performance on POQR dataset with state-of-the-art work of Bunescu and Huang (2010a). Bunescu and Huang (2010a) used several cosine similarities as features obtained using bag-of-words, dependency tree, focus, main verb etc. Compared to Bunescu and Huang (2010a), our model achieves better performance with an improvement of 2.1% and 1.46% on simple and complex dataset respectively. A direct comparison to SemEval-2017 Task-3[8] CQA or AskUbuntu (Lei et al., 2016) datasets could not be made due to the difference in the nature of questions. The proposed classification method is designed for well-formed English questions and could not be applied to multi-sentence / ill-formed questions. We evaluate (Lei et al., 2016)'s model (RCNN) on each of our datasets and report the results in Section 5.1. Quora has not released any official test set yet. Hence, we report the performance of 3-fold cross validation on the entire dataset to minimize the variance. We can not directly make any comparisons with others due to the non-availability of an official gold test set.

5.4 Error Analysis

We observed the following as major sources of errors in the proposed system: **(1)** Misclassification at the fine class level is often propagated to semantic question classifier when some of questions contain more than one sentence. For e.g. "*What's the history behind human names? Do non-human species use names?*". **(2)** Semantically dissimilar questions having same function words but different coarse and fine class were incorrectly predicted as similar questions. It is because of the high similarity in the question vector and focus, which forces the classifier to commit mistakes. **(3)** In semantic question ranking (SQR) task, some of the questions with higher lexical similarity to the reference question are selected in prior to the actual similar question due to the high cosine similarity score with the reference question.

6 Conclusion

In this work, we have proposed an efficient model for semantic question matching where DL models are combined with pivotal features obtained from taxonomy. We have created a two layered taxonomy (coarse and fine) for questions in interest and proposed a deep learning based question classifier to classify the questions. We have established the usefulness of our taxonomy on two different task (SQR and SQC) on four different datasets. We have empirically established that effective usage of semantic classification and focus of questions helps in improving the performance of various on semantic question matching. Future work includes the efficient question encoders and handling community forum questions, which are often ill-formed, using taxonomy based features.

7 Acknowledgements

We acknowledge the partial support of Accenture IIT AI Lab. We also thank the reviewers for their insightful comments. Asif Ekbal acknowledges Young Faculty Research Fellowship (YFRF), supported by Visvesvaraya PhD scheme for Electronics and IT, Ministry of Electronics and Information Technology (MeitY), Government of India, being implemented by Digital India Corporation (formerly Media Lab Asia).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Jiang Bian, Yandong Liu, Eugene Agichtein, and Hongyuan Zha. 2008. Finding the right facts in the crowd: factoid question answering over social media. In *Proceedings of the 17th international conference on World Wide Web*, pages 467–476. ACM.

[8] http://alt.qcri.org/semeval2017/task3/

Razvan Bunescu and Yunfeng Huang. 2010a. Learning the relative usefulness of questions in community qa. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 97–107. Association for Computational Linguistics.

Razvan Bunescu and Yunfeng Huang. 2010b. Towards a general model of answer typing: Question focus identification. In *Proceedings of The 11th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2010), RCS Volume*, pages 231–242.

Robin D Burke, Kristian J Hammond, Vladimir Kulyukin, Steven L Lytinen, Noriko Tomuro, and Scott Schoenberg. 1997. Question answering from frequently asked question files: Experiences with the faq finder system. *AI magazine*, 18(2):57.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

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

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *EMNLP*.

Danilo Croce, Simone Filice, Giuseppe Castellucci, and Roberto Basili. 2017. Deep learning in semantic kernel spaces. In *ACL*.

Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 813–820. IEEE.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. Kelp at semeval-2016 task 3: Learning semantic relations between questions and answers. In *SemEval@NAACL-HLT*.

Thomas R. Gruber. 1995. Toward principles for the design of ontologies used for knowledge sharing. *Technical Report KSL 93-04*.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the first international conference on Human language technology research*, pages 1–7. Association for Computational Linguistics.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 633–644, Doha, Qatar, October. Association for Computational Linguistics.

Jiwoon Jeon, W Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 84–90. ACM.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi Jaakkola, Kateryna Tymoshenko, Alessandro Moschitti, and Lluís Màrquez. 2016. Semi-supervised question retrieval with gated convolutions. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1279–1289, San Diego, California, June. Association for Computational Linguistics.

Shuguang Li and Suresh Manandhar. 2011. Improving question recommendation by exploiting information need. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1425–1434. Association for Computational Linguistics.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics, COLING 2002*, pages 1–7. Association for Computational Linguistics.

Lluís Màrquez, James Glass, Walid Magdy, Alessandro Moschitti, Preslav Nakov, and Bilal Randeree. 2015. Semeval-2015 task 3: Answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

S Mlynarczyk and S Lytinen. 2005. Faqfinder question answering improvements using question/answer matching. *Proceedings of L&T-2005-Human Language Technologies as a Challenge for Computer Science and Linguistics*.

D Moldovan, S Harabagiu, M Pasca, R Mihalcea, R Goodrum, R Girji, and V Rus. 2000. Lasso: A tool for surfing the answer net. In *Proceedings 8th Text Retrieval Conference (TREC-8)*.

Preslav Nakov, Lluıs Marquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. Semeval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval*, volume 16.

Jun-Ping Ng and Min-Yen Kan. 2015. Qanus: An open-source question-answering platform. *arXiv preprint arXiv:1501.00311*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382. ACM.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 707–712, Beijing, China, July. Association for Computational Linguistics.

Kai Wang, Zhaoyan Ming, and Tat-Seng Chua. 2009. A syntactic tree matching approach to finding similar questions in community-based qa services. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 187–194. ACM.

Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. 2017. Deconvolutional paragraph representation learning. In *Advances in Neural Information Processing Systems*, pages 4172–4182.

Guangyou Zhou, Yang Liu, Fang Liu, Daojian Zeng, and Jun Zhao. 2013. Improving question retrieval in community question answering using world knowledge. In *IJCAI*, volume 13, pages 2239–2245.

Appendices

A Proposed Taxonomy Table

	Coarse Classes	Fine Classes	Example
Non-Decision	**Quantification**	Temperature	What are the approximate temperatures that can be delivered by phase change materials?
		Time/Duration	How long did Baena worked for the Schwarzenegger/Shriver family?
		Mass	What is the weight in pounds of each of Schwarzenegger 's Hummers?
		Number	How many students are in New York City public schools?
		Distance	How many miles away from London is Plymouth?
		Money	What is the cost to build Cornell Tech?
		Speed	Give the average speed of 1987 solar powered car winner?
		Size	How large is Notre Dame in acres?
		Percent	What is the college graduation percentage among Manhattan residents?
		Age	How old was Schwarzenegger when he won Mr. Universe?
		Rank/Rating	What rank did iPod achieve among various computer products in 2006?
	Entity	Person	Who served as Plymouth 's mayor in 1993?
		Location	In what city does Plymouth 's ferry to Spain terminate?
		Organization	Who did Apple partner with to monitor its labor policies?
		Animal	Which animal serves as a symbol throughout the book?
		Flora	What is one aquatic plant that remains submerged?
		Entertainment	What album caused a lawsuit to be filed in 2001?
		Food	What type of food is NYC 's leading food export?
		Abbreviation	What does AI stand for?
		Technique	What is an example of a passive solar technique?
		Language	What language is used in Macedonia?
		Monuments	Which art museum does Notre Dame administer?
		Activity/Process	What was the name of another activity like the Crusades occurring on the Iberian peninsula?
		Disease	What kind of pain did Phillips endure?
		Award/Title	Which prize did Frederick Buechner create?
		Date	When was the telephone invented?
		Event	What event in the novel was heavily criticized for being a plot device?
		Sport/Game	Twilight Princess uses the control setup first employed in which previous game?
		Policy	What movement in the '60s did the novel help spark?
		Publication	Which book was credited with sparking the US Civil War?
		Body	What was the Executive Council an alternate name for?
		Thing	What is the name of the aircraft circling the globe in 2015 via solar power?
		Feature/Attribute	What part of the iPod is needed to communicate with peripherals?
		Industry Sector	In which industry did the iPod have a major impact?
		Website	Which website criticized Apple 's battery life claims?
		Other Tangible	In what body of water do the rivers Tamar and Plym converge?
		Other Intangible	The French words Notre Dame du Lac translate to what in English?
	Definition	Person	Who was Abraham Lincoln?
		Entity	What is a solar cell?
	Description	Reason	Why are salts good for thermal storage?
		Mechanism	How do the BBC 's non-domestic channels generate revenue?
		Cause & Effect	What caused Notre Dame to become notable in the early 20th century?
		Compare & Contrast	What was not developing as fast as other Soviet Republics?
		Describe	What do greenhouses do with solar energy?
		Analysis	How did the critics view the movie , " The Fighting Temptations "?
	List	Set of fine classes listed in the coarse classes *Quantification* and *Entity*	What are some examples of phase change materials? Which two national basketball teams play in NYC?
Decision	**Selection**	Alternative/Choice	Are the Ewell 's considered rich or poor?
		True/False	Is the Apple SDK available to third-party game publishers?

Table 7: The exemplar description of proposed taxonomy classes

B Algorithms

Algorithm 2 Question word extraction

```
procedure QUESTION WORD(QuesTokens)
    WhTags ← [WDT, WP, WP$, WRB]
    VbTags ← [VB, VBD, VBP, VBZ]
    for t ∈ QuesTokens do
        if t.POS ∈ WhTags then return t
    for t ∈ QuesTokens do
        if t.POS ∈ VbTags then return t
```

Algorithm 3 Focus Word Extraction

```
procedure FOCUS(QuesTokens)
    qw ← QUESTION WORD (QuesTokens)
    depP ← DependencyParse(QuesTokens)
    if qw is 'how' then
        return tail of 'advmod' of qw
    if qw.POS is V* then
        obj ← OBJECT(QuesTokens, qw)
        return obj
    if qw.POS is WH* then
        if 'root' is qw then
            nsubj ← tail of 'nsubj' of qw
            return nsubj
        else
            obj ← OBJECT (QuesTokens, 'root')
            return obj
    return <unk>
```

Algorithm 4 Object Extraction

```
procedure OBJECT(QuesTokens, qw)
    depP ← DependencyParse(QuesTokens)
    obj ← tail of 'det' of qw
    if obj not NULL then return obj
    obj ← tail of 'dobj' of qw
    if obj not NULL then return obj
    qw ← tail of 'conj:*' of qw
    obj ← tail of 'dobj' of qw
    if obj is NULL then
        comp ← tail of 'ccomp'/'xcomp' of qw
        obj ← tail of 'dobj' of comp
    return obj
```

C Additional Results

C.1 K-means Clustering

The k-means clustering was performed on the question representation obtained from the best question (RCNN-Attention) encoder of $2,000$ semantic question pairs. The clustering experiment was evaluated on the test set of Semantic SQuAD dataset (4000 questions). The performance was evaluated using the following metric:

$$\texttt{Recall} = \frac{100 \times \textit{no. of SQ pairs in same cluster}}{\textit{total no. of SQ pairs}} \tag{4}$$

K-means Clustering results are as follows: R@1:50.12, R@3:62.44 and R@5:66.58. As the number of clusters decreases `Recall` is expected to increase as there is higher likelihood of matching questions falling in the same cluster. `Recall` with $2,000$ clusters for $2,000$ SQ pairs i.e. 4,000 questions is comparable to Recall@1 as we have 2 questions per cluster on average, `Recall` with $1,000$ clusters is a proxy for Recall@3 and `Recall` with 667 clusters is comparable to Recall@5.

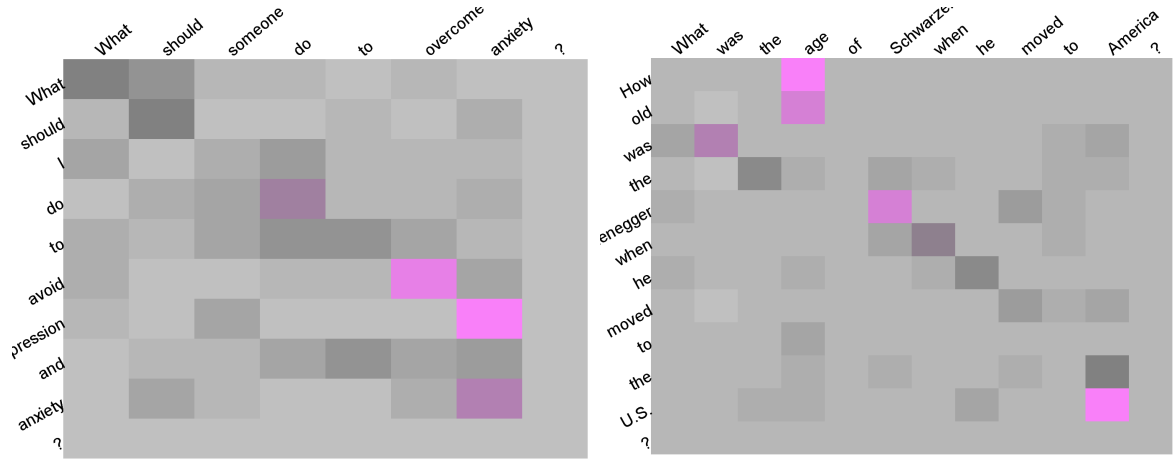
C.2 Semantic question classification (SQC) using IR-based Similarity

We have used TF-IDF, BM-25 and Jaccard similarity to classify a pair of question to similar or non-similar. We calculate the score between the question using the said algorithms thereafter a optimal thresholds are used to label a question pair as 'matching' or 'non-matching'. If the similarity score is greater than or equal to the threshold value we set the label 'matching' otherwise 'non-matching'. The optimal threshold value are calculated using the validation data. The optimal threshold value are given in the table 8.

Algorithm / Dataset	TF-IDF	BM-25	Jaccard Similarity
Semantic SQuAD Dataset	0.72	12.98	0.29
Quora Dataset	0.79	13.18	0.56

Table 8: IR based Optimal threshold value for each dataset

C.3 Attention Visualizations



(a) (b)

Figure 1: In **(a)** Attention mechanism detects semantically similar words (*avoid*, *overcome*). Attention mechanism is also able to align the multi-word expression '*how old*' to '*age*' as shown in **(b)**

Natural Language Interface for Databases Using a Dual-Encoder Model

Ionel Hosu[1], Radu Iacob[1], Florin Brad[2], Stefan Ruseti[1], Traian Rebedea[1]
[1]University Politehnica of Bucharest, Romania
[2]Bitdefender, Bucharest, Romania
{ionel.hosu, radu.iacob,stefan.ruseti,traian.rebedea}@cs.pub.ro
fbrad@bitdefender.com

Abstract

We propose a sketch-based two-step neural model for generating structured queries (SQL) based on a user's request in natural language. The sketch is obtained by using placeholders for specific entities in the SQL query, such as column names, table names, aliases and variables, in a process similar to semantic parsing. The first step is to apply a sequence-to-sequence (SEQ2SEQ) model to determine the most probable SQL sketch based on the request in natural language. Then, a second network designed as a dual-encoder SEQ2SEQ model using both the text query and the previously obtained sketch is employed to generate the final SQL query. Our approach shows improvements over previous approaches on two recent large datasets (WikiSQL and SENLIDB) suitable for data-driven solutions for natural language interfaces for databases.

1 Introduction

The quest for a simpler, more intuitive interface to query databases and other sources of structured information is one of the main practical applications of natural language processing. Developing a natural language interface for databases (NLIDB), able to process text queries directly, would be the optimal solution with regards to ease of use, requiring no additional knowledge and allowing non-technical persons to interact with the data directly. Nevertheless, even after half of decade of research, this is merely a desideratum, as no current solution is able to perform efficiently for complex databases.

From a computational linguistics perspective, natural language interfaces for databases could be treated as a special case of machine translation. Given a text in natural language to query a data source, a NLIDB system needs to output a statement in an artificial language, designed especially by computer scientists to query the database. Several such query languages exist, the most widely used being SQL (Structured Query Language) for structured databases and SPARQL for knowledge bases and ontologies.

Traditional NLIDB solutions have been using mainly a combination of rules and heuristics built upon syntactic dependencies and semantic parsing, with machine learning playing only a marginal role. The lack of large datasets suitable for training data-driven solutions is probably the main reason for the lack of complex machine learning approaches. However, a couple of large datasets (Brad et al., 2017; Zhong et al., 2017) have been recently proposed. These corpora allowed deep learning solutions to be deployed.

An additional obstacle that needs to be overcome by natural language interfaces for databases, especially by data-driven approaches, is the strong dependence of the corpora on the schema of a given database. Thus, even if one model is trained on a large dataset for a given database schema, it will not have good performance on different schemas. As it is impossible to develop a new dataset with pairs of text queries and corresponding SQL statements for each new database, the solution must be looked for in the underlying model. To this extent, we propose using a two-step approach which employs two different SEQ2SEQ models. The first one is schema agnostic and learns to output an SQL sketch (defined in Section 3) without any elements specific to a given database schema. The second model uses both the generated sketch and additional information about the database schema (information encoded

Proceedings of the 27th International Conference on Computational Linguistics, pages 514–524
Santa Fe, New Mexico, USA, August 20-26, 2018.

by the model in its vocabulary) to generate the final SQL statement. We show that this model offers a significant improvement compared to a baseline SEQ2SEQ network with attention.

From a different standpoint, it is important to note that one of the biggest obstacles in generating code, including SQL, is that the output needs to be both syntactically and semantically valid. In previous research on code generation from natural language using neural models, this issue has been tackled by incorporating the syntax into a tree-based decoder (Yin and Neubig, 2017). To this end, our proposed two-step approach aims to learn separately the syntactic and semantic aspects of SQL generation without incorporating any human-defined grammar. The first step involves a SEQ2SEQ network trained on (text query, SQL sketch) pairs. The second step consists of a dual-encoder SEQ2SEQ architecture that generates SQL code based both on the user's request and the previously decoded sketch. Conditioning on the sketch can be seen as injecting syntactic information which helps the second network to focus more on the semantic aspects and on improving the syntax, given additional semantic elements. We show that this model offers a significant improvement compared to a baseline SEQ2SEQ model on two large datasets suitable for data-driven approaches for NLIDB.

The paper continues with an overview of existing solutions for NLIDBs, covering both incipient deep learning approaches and multi-stage conventional ones using a mix of rule-based and machine learning. In Section 3 we briefly present SENLIDB and WikiSQL, two large datasets of text queries and corresponding SQL statements which have already been used to train deep learning models for NLIDB. Section 4 presents the main contribution of our paper, introducing a two-step approach for generating SQL statements from textual input and query sketch using a novel dual-encoder neural model. Section 5 highlights the performance of the proposed two-step architecture compared to previous work. The paper ends with conclusions and future work.

2 Related Work

The first NLIDB solutions used dictionaries, grammars and dialogue to allow users to iteratively refine their query in natural language (Codd, 1974; Hendrix et al., 1978). For decades, complex multi-stage systems were designed for interacting in natural language with a database. They consisted of a mix of hand-crafted rules and heuristics, pattern matching, syntactic parsing, semantic grammar systems, and intermediate representation languages before generating the final statement in the query language (Androutsopoulos et al., 1995).

More recent conventional approaches combine syntactic parsing and semantic alignment to change the order of nodes in the parse trees in order to be correctly interpreted by a semantic analyzer (Popescu et al., 2004). To align the text with the generated SQL candidate statements, they propose using bipartite matching and dictionaries for semantic alignment. NaLIR (Li and Jagadish, 2014) also uses dependency parse trees and several hand-made rules and heuristics to generate candidate SQL statements. Given the input's dependency tree, the database schema and some manual semantic mappings, the system builds candidate query trees, as an intermediate step to SQL generation. The best query tree is computed using a scoring mechanism taking into account the similarity between the dependency and query trees and between adjacent nodes in the query tree, and an additional interaction with the user which is asked to select the best choice from a list of reformulations in natural language of top ranking candidate query trees.

Sqlizer (Yaghmazadeh et al., 2017) uses a mix of traditional rule-based and heuristics approaches combined with machine learning. Similar to our solution, a semantic parser is employed to generate a query sketch, which is then iteratively refined and repaired using rules and heuristics until the score of the generated SQL query cannot be improved. Sqlizer uses machine learning and Word2Vec (Mikolov et al., 2013) to determine the query sketch - a general form of the query, including clauses, but which does not contain any specific database schema information (e.g. table and column names). The reported results are surpassing previous solutions for several standard NLIDB datasets. While the underlying idea of Sqlizer (Yaghmazadeh et al., 2017) is similar to ours (using an intermediate sketch), there are significant differences. First, they do not employ a purely data-driven approach, as they use several hand-written heuristics. Secondly, the query sketch proposed by them is structurally different (encoding

merely column and table names). Thirdly, the sketch is generated by a semantic parser instead of a neural model. However, the neural dual-encoder model employed in the second step of SQL generation is actually the most original part of our work, differentiating our paper from previous proposals employing SQL sketches. This model is novel and provides a substantial improvement over previous rule-based systems for filling the query sketch.

Deep learning approaches for code generation and semantic parsing have allowed the mapping from language to structured output to be learned directly, without the need for intermediate processing steps and hand-crafted rules and heuristics. Iyer et al. (2017) apply an interactive user-based feedback learning scheme to improve the results of a standard SEQ2SEQ model for generating SQL statements. Other approaches (Yin and Neubig, 2017; Rabinovich et al., 2017) incorporate the syntactic structure of the target structured language and decode the Abstract Syntax Tree (AST) corresponding to the surface code. Other recent solutions in semantic parsing avoid the need for ground truth logical forms inferring them automatically from (textual input, answer) pairs and also showing how this process can be generalized to multiple tables (Yin et al., 2015; Neelakantan et al., 2016).

Several large datasets have been recently introduced to facilitate training of deep learning approaches for NLIDB. Brad et al. (2017) propose the SENLIDB dataset together with a baseline SEQ2SEQ architecture for generating complex SQL statements. Zhong et al. (2017) introduce the WikiSQL dataset which contains simpler SQL-like queries involving a single table. They also suggest augmenting a standard SEQ2SEQ model with pointer networks (Vinyals et al., 2015) and highlight the fact that the order of specific clauses from the generated SQL statement affects the accuracy of a SEQ2SEQ model despite having no impact on the actual execution results. To mitigate this issue they apply reinforcement learning and reward queries that diverge from the ground truth but which return the correct results. More recently, Xu et al. (2017) report improved results on the same dataset without the use of reinforcement learning. Their solution, called SQLNet, employs a sketch-based approach designed to align well with the specific structure of SQL statements found in WikiSQL. Their SQL sketch conditions the prediction of a slot only on those particular values it may depend on, instead of all previous predictions. Our sketch-based approach is extended to support full SQL queries, while the two-step approach which uses both the sketch and the description in a dual-encoder to generate the SQL statement differentiates our model from SQLNet.

On another hand, dual-encoder SEQ2SEQ models have been successfully employed used both for machine translation and neural conversational models. In machine translation, dual-encoders have shown to improve the results when using multiple sources as input to generate the target translation (Zoph and Knight, 2016). In addition, dual-attentive decoders using features both from the source sentence and from an additional image related to it have been shown to improve multi-modal machine translation (Calixto et al., 2017). For neural conversational models, while Li et al. (2016) paved the road by introducing a persona-based decoder fed both from an encoder modelling the user input and from a persona specific embedding, only recently dual-encoders have been shown to produce better results than more complex hierarchical neural models (Lowe et al., 2017). Our approach uses a similar dual-encoder model, however we provide as its input two different representations for the desired SQL statement, the textual description supplied by the user and an automatically generated SQL sketch modelling the syntax of the final statement.

3 Two-Step Sketch-Based Model

We define a two-step solution for the problem of generating SQL statements from natural language queries[1]. First, we employ a network that learns to generate an SQL sketch given the user's request. Second, using both the aforementioned generated sketch and the corresponding natural language query, we propose a dual-encoder model to generate the final SQL statement. We consider that the SQL sketch generation is a simpler task, whose intermediate result provides important additional insights for determining the correct SQL statement.

Moreover, as the sketch preserves the SQL syntax of the final statement and only removes semantic

[1]Implementation available at `https://github.com/johnthebrave/nlidb-datasets`

information (e.g. names of tables, columns, constants and other), the proposed two-step model effectively splits the SQL generation problem into two distinct subproblems:

1. The problem of learning the syntax of the SQL query language and generating the most probable SQL sketch given a query in natural language. This refers to learning the surface structure of an SQL stamtement: the order of SQL reserved keywords in a (syntactically) correct query, the precedence of different clauses and subclauses in a the statement, the usage of different operators and so on.

2. The problem of generating semantically correct SQL statements for a given database schema, taking into account both the user's description and the most probable SQL sketch. This task mainly involves choosing the correct table and column names to fill in the sketch, generating the appropriate clauses (WHERE, GROUP BY etc.) given these names, but also correcting the sketch based on the additional semantic and schema information.

3.1 SQL Sketch Generation

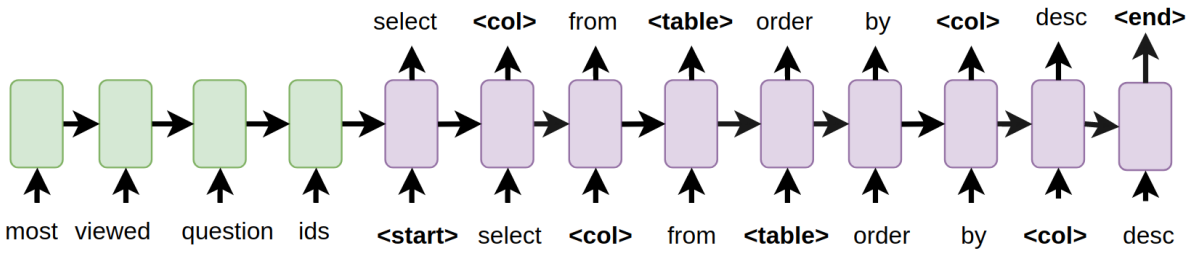


Figure 1: SQL sketch generation using a SEQ2SEQ network

In the first step, the model learns to generate an SQL sketch conditioned on the user's request. In order to obtain the sketches, we altered the dataset by replacing non-SQL keywords, reserved chars and operators within each SQL statement with placeholders (e.g. table names with <table>, similar notations for column names, aliases, variables, and constants). Thus, the SQL sketch only stores the surface form of an SQL statement and is syntactically correct, as we kept all mathematical and logical operators, SQL reserved keywords and special chars. All elements which are specific to the underlying database schema are removed from the sketch, therefore it is schema-agnostic.

To generate the SQL sketch, we have employed a standard SEQ2SEQ model with global-general attention (Luong et al., 2015) which we train on the (description, SQL sketch) pairs extracted from the dataset. An example of generating the SQL schema for a given user input is provided in Figure 1. Using placeholders for schema-related elements in the SQL sketch significantly reduces the output vocabulary for the decoder. This helps in two ways. First, it makes the model focus on the syntactic structure of the SQL statements. Second, it allows training the SEQ2SEQ network using less data - as the syntax is independent from the database schema, this data might even come from different datasets.

We used the open-source neural machine translation toolkit OpenNMT [2]. Both the encoder and the decoder are long short-term memory (LSTM) cells with two hidden layers and 500 neurons. The word embedding layer has 500 neurons. We used batches of maximum size 64. We trained the models with Stochastic Gradient Descent (SGD) for 25 epochs with a learning rate of 1.0 and a learning decay of 0.5 if perplexity did not decrease on the validation set. We generated the SQL sketch using a beam search of size 5.

3.2 Dual-Encoder for SQL Statement Generation

The second stage of the process consists of generating the full SQL statement using a dual-encoder SEQ2SEQ model, as shown in Figure 2. This model receives the natural language description of the query, as well as the SQL sketch obtained in the previous step, which are processed separately by each encoder. The last hidden states of each encoder are concatenated and fed to the decoder to generate the

[2] https://github.com/OpenNMT/OpenNMT-tf

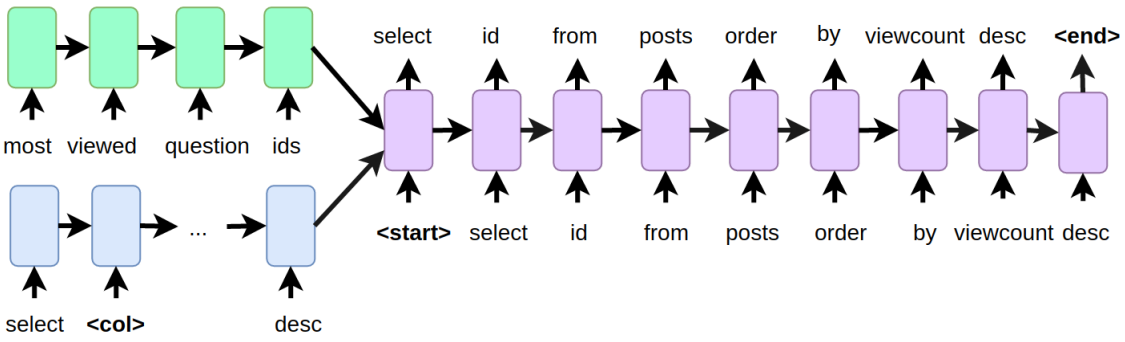


Figure 2: Query generation based on natural language and query sketch

final SQL statement. The decoder uses two attention mechanisms to attend both to the textual description and to the SQL sketch. By relying on the syntactical information provided by the sketch, the dual-decoder can focus more on schema specific entities (e.g. filling in the correct table and column names in the final statement) and on semantic aspects (e.g. generating the correct columns from a selected table). We also expect the dual-encoder model to be able to fix SQL sketches which contain some errors (when compared to the ground truth) after the first step as adding schema and semantic information might improve the syntax as well.

The encoders and the decoder share no weights with the first SEQ2SEQ network used for sketch generation. We used long short-term memory (LSTM) cells with two hidden layers and 500 neurons. The dual-encoder architecture was trained for 25 epochs using batches of 64 samples and a learning rate of 1.0 with learning decay of 0.5. We employed grid search for establishing the number of LSTM cells in the hidden layers and the size of the word embeddings (size 500). Additional hyperparameters that we used are dropout rate (0.3).

4 Results

We evaluated the proposed architecture on the SENLIDB and WikiSQL datasets. We compare our two-step dual-encoder model with a SEQ2SEQ baseline with global-general attention and unknown replacement (Luong et al., 2015). We also perform ablation tests by removing the attention mechanism, either on the textual description encoder or on the SQL sketch encoder, for the dual decoder that generates the final SQL statement.

Model	Dataset	BLEU	Accuracy
Dual-Encoder Seq2Seq (dual attention)	Dev	**83.89**	**39.50**
	Test	**83.2**	**38.99**
Dual-Encoder Seq2Seq (attention on description)	Dev	83.55	38.89
	Test	83.17	38.45
Dual-Encoder Seq2Seq (attention on SQL sketch)	Dev	83.86	38.54
	Test	83.21	38.47
Seq2Seq	Dev	78.90	32.60
	Test	78.50	32.00

Table 1: BLEU scores and accuracy for the generated SQL statements on the WikiSQL dataset

As it can be seen in Table 1 and Table 2, the proposed dual-encoder SEQ2SEQ model greatly outperforms the SEQ2SEQ baseline on both datasets. As expected, the BLEU and accuracy scores on the SENLIDB dataset are significantly smaller than on the WikiSQL dataset, due to the complexity of the SQL statements and database schema from the former dataset. In all experiments, accuracy is computed by checking whether the generated SQL statement matches the ground truth. This process does not take into account that different queries may actually provide the same results. This is especially true for the

Model	Dataset	BLEU	Accuracy
Dual-Encoder SEQ2SEQ	Validation	**21.70**	**3.52**
(dual attention)	Test-annotated	**22.92**	**3.97**
Dual-Encoder SEQ2SEQ	Validation	20.94	3.22
(attention on description)	Test-annotated	21.13	3.33
Dual-Encoder SEQ2SEQ	Validation	21.32	3.47
(attention on SQL sketch)	Test-annotated	21.80	3.60
SEQ2SEQ	Validation	16.90	2.80
	Test-annotated	18.20	2.44

Table 2: BLEU scores and accuracy for the generated SQL statements on SENLIDB dataset

more complex queries in the SENLIDB dataset (e.g. changing the order of the selected columns does not influence the results, but it influences the query match accuracy).

We also notice that removing any of the two attention mechanisms reduces the overall performance of the model on both datasets. This suggests that contextual information from both the SQL sketch and textual description is important to generate the final SQL statement.

Ultimately, all the dual-encoder models outperform the single encoder SEQ2SEQ model on the two datasets, which proves that the proposed two-step model benefits from the previously generated sketch. The added performance of our dual-encoder model comes from the initial task of SQL sketch generation. This demonstrates that even an incorrect, but approximate, syntactic surface form provides important additional information to the dual-encoder which corroborated with the textual description increases the performance of the model.

For the WikiSQL dataset, we also performed experiments with pretrained embeddings (GloVe) (Pennington et al., 2014) and a copying mechanism (See et al., 2017). Using these improvements, the proposed dual encoder model achieves an accuracy of 55.07%, outperforming most previous results.

4.1 Dataset Statistics

We have evaluated our proposed two-step neural architecture on two large datasets: WikiSQL (Zhong et al., 2017) and SENLIDB (Brad et al., 2017). Both datasets consist of several tens of thousand pairs of questions and corresponding SQL statements. However, they are different both with regard to the complexity of the database schemas and to the nature of the corresponding SQL statements. In this section, we briefly explain the characteristics of each dataset to highlight their different nature.

The content of the database tables used in WikiSQL (Zhong et al., 2017) was generated starting from $24,241$ Wikipedia tables. Schema-wise, the database is simplistic as all tables are independent from each other: there are no foreign keys or any other relationships between tables. The natural language queries and corresponding SQL statements were generated in several steps. First, SQL statements were randomly produced using a fixed template. Second, a corresponding interrogation in natural language was then generated automatically for each statement. Afterwards, human annotators from Amazon Mechanical Turk were employed to suggest several paraphrases for the automatically generated interrogations. Other human annotators were further employed to ensure the validity and quality of these paraphrases.

The fixed templates used to generate the SQL statements enforced them to have a specific, simple structure. Each statement contains a single $'select'$ clause requesting a unique column from only one table. An aggregation operator (e.g. $count()$) may be applied on the selected column. The only other component of the SQL statement is the $'where'$ clause, which also follows a very restricted form. Each corresponding subclause is actually a binary operation (e.g. equality operator, '=') applied on a column name and a substring which can always be found in the associated natural language query.

On the other hand, SENLIDB (Brad et al., 2017) consists of real-life SQL statements issued by users on the Data StackExchange Explorer website [3]. Unlike the ones from WikiSQL, these statements have no predefined syntactic limitations and are not generated automatically starting from a specific pattern.

[3] https://data.stackexchange.com/

Datasets / Sample Statistics	SENLIDB		WikiSQL		
	Train	**Test**	**Train**	**Dev**	**Test**
# Samples	24890	780	61297	9145	17284
# Tables	29	15	18471	2704	5200
# Columns	204	98	50207	10144	17642
Avg. Text Length	7.88	10.44	11.73	11.81	11.78
Avg. SQL Length	71.38	7.46	11.50	11.48	11.55
Avg. SQL AST Height	26.11	20.27	19.62	19.63	19.63

Table 3: Brief comparison between SENLIDB and WikiSQL datasets

Moreover, the statements are issued against the StackExchange database which has a complex schema with several relationships and constraints between the various tables. Each SQL statement has a corresponding textual description added by the user who created it. This process resembles a crowd-sourced dataset, with each pair of (textual description, SQL statement) created by a user. To ensure the quality of the dataset, Brad et al. (2017) have preprocessed the data to remove incorrect or uninformative queries. Moreover, a small subset of queries (SENLIDB Test) has been manually annotated by developers with corresponding queries in natural language.

Table 3 provides a brief comparison between the two datasets. Although the size of the training data is similar for both datasets, the test set for SENLIDB is significantly smaller than the corresponding one for WikiSQL. This can be explained by the fact that this data has been manually annotated by developers. Related to the number of tables, although WikiSQL references a significantly larger number of tables than SENLIDB, there is no relation between them and each SQL statement only queries one table. In contrast, the SQL statements in SENLIDB may require data from several tables (thus requiring several $'join'$ operations) or use subqueries or even data extracted from temporary tables created within the statement. Finally, the Abstract Syntax Tree (AST) for the statements in SENLIDB are more complex not only to the number and type of SQL subclauses, but also to the height of the AST.

4.2 Qualitative results

input: what is terrence ross ' nationality **generated sketch:** select <col>from table where <col> = <col> **ground truth sketch:** select <col> from table where <col> = <ct> **generated SQL:** select [nationality] from table where [player] = [terrence ross] **ground truth SQL:** select [nationality] from table where [player] = [terrence ross]
input: which podiums did the williams team have with a margin of defeat of 2 **generated sketch:** select <col> from table where <col> = <ct> **ground truth sketch:** select <col> from table where <col> = <ct> and <col> = <ct> **generated SQL:** select [podiums] from table where [margin of error] = [2] and [team] = [williams] **ground truth SQL:** select [podiums] from table where [team] = [williams] and [margin of defeat] = [2]

Table 4: Examples of SQL sketches and statements generated for the WikiSQL dataset

We list some decoded examples in Table 4 and Table 5. In the first example in Table 4, both the sketch and the final SQL statement are generated correctly by the dual-encoder model. The generated sketch in the second example is incorrect, but the final SQL statement is correct, which means the dual-encoder model actually rectified the surface syntactic form provided by the sketch. This shows that the dual-encoder not only adds specific schema elements and semantic information, but actually is able to modify some incorrect SQL sketches generated in the first step. This result is important as it means that additional semantic information is successfully used by the model to correct the proposed syntax.

In Table 5, we notice that for both examples, the generated sketch is very different from the ground

input: show top 50000 posts which have more than 10000 views **generated sketch:** select top <ct> <alias> . <col> as <alias> , <alias> . <table> , <alias> . <col> , <alias> . <col> from <table> as <alias> inner join <table> as <alias> on <alias> . <col> = <alias> . <col> where <alias> . <col> = <ct> and <alias> . <col> > = <ct> order by <alias> . <col> desc **ground truth sketch:** select top <ct> * from <table> where <table> . <col> > <ct> order by <table> . <col> desc **generated SQL:** select top 10000 posts . id , posts . body from posts where posts . viewcount > 10000 **ground truth SQL:** select top 50000 * from posts where posts . viewcount > 10000 order by posts . viewcount desc
input: list id , body of posts that have the same postid as the id of posttags and tagid of posttags = 17 and id of posts < 1000 **generated sketch:** select <table> . <col> , <col> (*) as <alias> from <table> inner join <table> on <table> . <col> = <table> . <col> inner join <table> on <table> . <col> = <table> . <col> inner join <table> on <table> . <col> = <table> . <col> and <col> = <ct> group by <table> . <col> order by <col> desc **ground truth sketch:** select <alias> . <col> , <alias> . <col> from <table> <alias> inner join <table> <alias> on <alias> . <col> = <alias> . <col> where <alias> . <col> = <ct> and <alias> . <col> < <ct> **generated SQL:** select count (*) from posts inner join posttags on posttags . postid = posts . id **ground truth SQL:** select p . id , p . body from posts p inner join posttags pt on p . id = pt . postid where pt . tagid = 17 and p . id < 1000

Table 5: Examples of SQL sketches and statements generated for the SENLIDB dataset

truth sketch. However, in the first example the generated SQL statement resembles the target SQL ($'top'$ clause, $'viewcount'$ column and $'posts'$ table are correctly identified). In the second example, the network correctly generates a more complex SQL query that contains a join operation between tables $'posts'$ and $'posttags'$. Again, the dissimilarity between sketches and similarity between final SQL statements suggests that the dual-encoder can recover from a incorrect generated sketch.

One of the reasons for which the SQL sketches on the SENLIDB dataset are more complex is related to the fact that the training data contains slightly more complex queries than the test set. This leads the sketch network to have a bias for more complex sketches.

4.3 Sketch accuracy

Dataset	BLEU	Accuracy
WikiSQL test	94.06	82.38
SENLIDB test	28.36	5.12

Table 6: BLEU scores and accuracy of the predicted SQL sketches

We have also investigated the performance of each of the two steps in our proposed model. In this subsection we measure the performance of the first step, the generation of the SQL sketch. In Table 6 we measure how many sketches are correctly predicted by the first SEQ2SEQ model. The BLEU score and accuracy of the generated sketches for the WikiSQL dataset are very high, as expected due to the much simpler nature of the SQL queries featured in this dataset. The accuracy of the final SQL statements is very low compared to the sketch accuracy, which means that the main difficulty with the WikiSQL dataset is correctly instantiating the SQL sketch with the appropriate columns names and constants thus resembling a slot filling task.

On the other hand, the sketch accuracy for SENLIDB is significantly lower compared to WikiSQL.

This is unsurprising, as the SQL queries are more complex. However, the difference between the sketch accuracy and the SQL accuracy for SENLIDB is small, which suggests that getting the right sketch is crucial for these queries and this step is actually more difficult than instantiating them.

4.4 Sketch feeding

We also evaluated the performance of our model when the correct sketch is being provided along with the textual description. To this extent, the dual-encoder in the second step receives the ground truth sketch at test time instead of the decoded sketch from the previous network.

Model	Dataset	BLEU	Accuracy
Dual-Encoder Seq2Seq	WikiSQL test	92.77	39.04
	SENLIDB test	68.20	25.89

Table 7: BLEU scores and accuracy of the generated SQL statements using the ground truth SQL sketch

We observe in Table 7 that the accuracy improves significantly on the SENLIDB corpus (21.92% improvement), which shows again the importance of generating the correct sketch for these difficult SQL statements. The accuracy boost on WikiSQL is small (1.15% improvement), reconfirming our earlier observation that WikiSQL is more challenging as a slot filling task.

4.5 Tables and Column Identification

Model	Task	Precision	Recall	F1 score
Dual-Encoder SEQ2SEQ	Table prediction	0.85	0.74	0.78
	Column prediction	0.59	0.53	0.55
SEQ2SEQ (Brad et al., 2017)	Table prediction	0.82	0.72	0.76
	Column prediction	0.55	0.47	0.50

Table 8: Results (using macro-averaging) for table and column prediction on the SENLIDB dataset

A simpler, but important subtask for the generation of SQL statements is the correct instantiation of table and column names in a query. In Table 8, we present different metrics for this task treated as a classification problem. As it can be observed, the dual-encoder model consistently outperforms the SEQ2SEQ baseline by merely using the SQL query sketch as input.

5 Conclusions

In this paper, we presented a two-step architecture for generating SQL statements starting from a user request expressed in natural language. The novelty resides in generating the underlying SQL sketch first, followed by the generation of the full SQL statement using a dual-encoder model conditioned by both the sketch and the query in natural language. The two-step approach actually separates the problem of generating an SQL statement into first constructing the correct syntactic surface form, which is later used to embed semantic and schema specific elements in the second step.

We demonstrated the validity of this approach by comparing the two-step model with a SEQ2SEQ baseline on two large datasets. The results show that the model significantly improves the results of the baseline. In addition, the best results are obtained when the dual-encoder model uses a dual attention mechanism on both textual description and SQL sketch. Future research should consider improving the two attention mechanisms and how they are combined. Other improvements can be obtained by training the sketch generation network on several datasets, considering that the sketch is schema-agnostic and might benefit from transfer learning between datasets.

A last insight of the paper relates to an important difference between the two datasets. As WikiSQL contains data from a simplistic database schema with artificial constraints for the SQL statements, it resembles a slot filling task. On the other hand, the main challenge for solving the more complex queries in SENLIDB is the generation of the correct SQL sketch corresponding to a query.

References

I. Androutsopoulos, G.D. Ritchie, and P. Thanisch. 1995. Natural language interfaces to databases – an introduction. *Natural Language Engineering*, 1(1):29–81.

Florin Brad, Radu Iacob, Ionel Hosu, and Traian Rebedea. 2017. Dataset for a neural natural language interface for databases (nnlidb). *arXiv preprint arXiv:1707.03172*.

Iacer Calixto, Qun Liu, and Nick Campbell. 2017. Doubly-attentive decoder for multi-modal neural machine translation. *arXiv preprint arXiv:1702.01287*.

E. F. Codd. 1974. Seven steps to rendezvous with the casual user. In *IFIP Working Conference Data Base Management*, pages 179–200, January. IBM Research Report RJ 1333, San Jose, California.

Gary G. Hendrix, Earl D. Sacerdoti, Daniel Sagalowicz, and Jonathan Slocum. 1978. Developing a natural language interface to complex data. *ACM Trans. Database Syst.*, 3(2):105–147, June.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback.

Fei Li and H. V. Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *VLDB*, 8(1):73–84, September.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*.

Ryan Thomas Lowe, Nissan Pow, Iulian Vlad Serban, Laurent Charlin, Chia-Wei Liu, and Joelle Pineau. 2017. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue & Discourse*, 8(1):31–65.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS13, pages 3111–3119, USA. Curran Associates Inc.

Arvind Neelakantan, Quoc V. Le, Martin Abadi, Andrew McCallum, and Dario Amodei. 2016. Learning a Natural Language Interface with Neural Programmer. pages 1–13.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. 2004. Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. *CoRR*, abs/1704.07535.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Type and content-driven synthesis of sql queries from natural language. *CoRR*, abs/1702.01168.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation.

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

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. *arXiv preprint arXiv:1601.00710*.

Employing Text Matching Network to Recognise Nuclearity in Chinese Discourse

Sheng Xu[1], Peifeng Li[1], Guodong Zhou[1], and Qiaoming Zhu[12]
[1]School of Computer Science and Technology, Soochow University, China
[2]Institute of Artificial Intelligence, Soochow University, China
sxu@stu.suda.edu.cn;{pfli, gdzhou, qmzhu}@suda.edu.cn

Abstract

The task of nuclearity recognition in Chinese discourse remains challenging due to the demand for more deep semantic information. In this paper, we propose a novel text matching network (TMN) that encodes the discourse units and the paragraphs by combining Bi-LSTM and CNN to capture both global dependency information and local n-gram information. Moreover, it introduces three components of text matching, the Cosine, Bilinear and Single Layer Network, to incorporate various similarities and interactions among the discourse units. Experimental results on the Chinese Discourse TreeBank show that our proposed TMN model significantly outperforms various strong baselines in both micro-F1 and macro-F1.

1 Introduction

During the past few years, the focus of Natural Language Understanding (NLU) has shifted from the word/sentence level to the discourse level. A challenging task in NLU is discourse parsing, which involves analysing the relations between discourse units and building the document structure. As one of the most influential discourse theories, Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) defines a document as a collection of Elementary Discourse Units (EDUs) with semantic connections and combines adjacent EDUs with rhetorical relations in a hierarchical way to represent an entire document as a discourse tree.

As a critical subtask in discourse parsing, nuclearity recognition involves identifying the nuclearity between the discourse units and thus being able to extract the main information in a document. According to RST, a discourse relation can be divided into mononuclear and multinuclear. A mononuclear relation holds a nucleus and a satellite, where the nucleus expresses the main textual information and the satellite offers additional information about the nucleus (Stede, 2008), while a multinuclear relation holds two or more discourse units, which are all nuclei. Therefore, three types of nuclearity exist: Nucleus-Satellite if the left subtree is the nucleus and the right subtree is the satellite, Satellite-Nucleus if the order of the satellite and nucleus is inverted, and Nucleus-Nucleus for multinuclear relations.

Nuclearity recognition is helpful in detecting discourse relations (Iruskieta et al., 2014) and extracting the main content of a document, and it is widely used in various NLP tasks, including automatic summarisation (Louis et al., 2010; Marcu, 2000), question answering (Verberne et al., 2007) and information extraction (Zou et al., 2014). Consider the following document as an example:

Example 1: 中国机电产品进出口贸易继续增加 $_{a}$，占总进出口的比重继续上升 $_{b}$。其中，出口五十七点九亿美元 $_{c}$，占总出口的百分之三十二点五 $_{d}$；进口八十五点二亿美元 $_{e}$，占总进口的百分之四十六点四 $_{f}$，均比去年同期有所上升 $_{g}$。 *The import and export trade of China's mechanical and electronic products continues to increase* $_{a}$, *and its proportion of the total imports and exports also continues to rise* $_{b}$. *Among them, the exports amounted to 5.79 billion dollars* $_{c}$, *accounting for 32.5 percent of the total exports* $_{d}$; *and the imports of 8.52 billion dollars* $_{e}$, *accounting for 46.4 percent of the total imports* $_{f}$; *all of them were higher than those in the same period last year* $_{g}$.

Example 1 shows a paragraph that includes seven EDUs (*a*-*g*), and its corresponding nuclearity discourse tree is illustrated in Figure 1, where the leaf nodes (*a*-*g*) in Figure 1 are EDUs and the internal

Proceedings of the 27th International Conference on Computational Linguistics, pages 525–535
Santa Fe, New Mexico, USA, August 20-26, 2018.

nodes refer to relational nodes, which represent the combination of the relevant children. When connecting the parent and child nodes, the directed edge indicates that the child is a nucleus in the relationship and the undirected edge indicates that the child is a satellite.

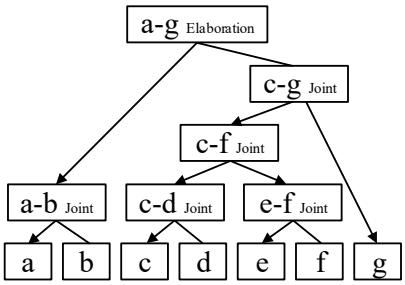


Figure 1: Discourse tree of Example 1.

Starting from the root node, i.e., *a-g* in Figure 1, we can continually select all of the branches labelled as nucleus until the leaf node, i.e., *a* in Figure 1 (中国机电产品进出口贸易继续增加 *The import and export trade of China's mechanical and electronic products continues to increase*), which can be used to represent a summary of this paragraph.

Although there are many studies on discourse parsing due to its vital role in NLP, only a few address nuclearity recognition. Among them only three studies (Li et al., 2015; Chu et al., 2015; Kong and Zhou, 2017) explore nuclearity recognition in Chinese due to the lack of annotated corpus and the abstract nature of Chinese itself. In addition, those studies heavily relied on manual feature engineering (Feng and Hirst, 2014; Heilman and Sagae, 2015; Wang et al., 2017). Only a few studies (Li et al., 2014; Li et al., 2016) used deep neural networks to explore automatic representation learning. One of the disadvantages of previous studies is that they lack deep semantic information extracted from discourse units due to the ineffectiveness of classifier-based models and simple neural network models. Even worse, different from those hypotactic languages such as English, Chinese is a paratactic (discourse-driven and pro-drop) language with a wide spread of ellipsis and open flexible sentence structures. Therefore, the shallow semantic features (syntactic features), which are widely used in English, might not be effective in Chinese. This property makes discourse parsing in Chinese more challenging.

In this paper, we propose a novel text matching network (TMN) for nuclearity recognition. The TMN model encodes the discourse units and paragraphs by combining Bi-LSTM and CNN to capture both the global dependency information and the local n-gram information. Moreover, it introduces three components of text matching, i.e., Cosine, Bilinear and Single Layer Network, to incorporate various similarities and interactions between discourse units and thus provide more useful information to recognise nuclearity. Experimental results on the Chinese Discourse TreeBank (CDTB) (Li et al., 2014) show that our proposed TMN model significantly outperforms various strong baselines in both micro-average and macro-average F1. We summarise the contributions of our work as follows:

- We combine Bi-LSTM and CNN to jointly learn proper representation of the discourse units, which can capture both global dependency information and local n-gram information.
- We introduce three text matching components, i.e., Cosine, Bilinear and Single Layer Network to capture various semantic similarities and interactions among the discourse units.
- We consider the semantic relations between the discourse units and paragraphs. These relations provide an effective supplement to recognise the nuclearity types.

The remainder of this paper is organised as follows: Section 2 introduces the related work, Section 3 gives the details of our model TMN, Section 4 reports the experimental results and Section 5 gives the conclusions.

2 Related Work

Previous studies on nuclearity recognition mainly focused on English, with RST Discourse Treebank (RST-DT) (Carlson et al., 2003) being the most popular corpus. However, most of them only regard

nuclearity recognition as a trivial component of overall discourse parsing, and they ignore its specific characteristics and critical importance.

The algorithms of nuclearity recognition published on RST-DT can mainly be categorised as shift-reduce algorithms (Ji and Eisenstein, 2014; Heilman and Sagae, 2015; Wang et al., 2017), probabilistic CKY-like algorithms (Joty et al., 2013; Li et al., 2014; Li et al., 2016) and greedy bottom-up algorithms (Feng and Hirst, 2014). Li et al. (2016) applied different classifiers to three discourse parsing subtasks separately, but they share the high level representation of discourse units by the same network structures. Wang et al. (2017) used a transition-based system to build discourse trees with nuclearity labels and then used Support Vector Machines (SVMs) to determine the discourse relations at different text levels.

Most of the previous studies used SVMs and variants of Conditional Random Fields (CRFs); only Li et al. (2014) and Li et al. (2016) introduced neural networks into nuclearity recognition. Li et al. (2014) used a two-layer feedforward neural network to determine the relation between text spans and computed the representation for each text span based on the representations of its subtrees by recursive neural models. Li et al. (2016), which is used as one of our baselines, proposed an attention-based hierarchical Bi-LSTM network to learn the representations of the text spans and used a tensor-based transformation function to capture interactions among the features of the text spans.

For recognising nuclearity between Chinese discourse units, there are only three studies. Li et al. (2015), Chu et al. (2015) and Kong and Zhou (2017) have done some preliminary work on Chinese Discourse TreeBank (CDTB) (Li et al., 2014). Li et al. (2015) used contextual features, lexical features and dependency tree features to recognise nuclearity by a Maximum Entropy (ME) classifier. Chu et al. (2015) used similar features to recognise three types of nuclearity by three different ME classifiers and used sampling techniques to obtain a balanced training set and testing set. Kong and Zhou (2017) integrated some previous research and proposed a CDT-styled End-to-End discourse parser, which can automatically detect discourse units and perform all three discourse parsing subtasks in sequence. They used the same model of nuclearity recognition as Li et al. (2015) and reported the same results.

3 Text Matching Network on Nuclearity Recognition

In this section, we propose a novel text matching network (TMN) for nuclearity recognition, and its high-level illustration is shown in Figure 2, which includes three modules: 1) Text Encoding, 2) Text Matching, and 3) Nuclearity Classification.

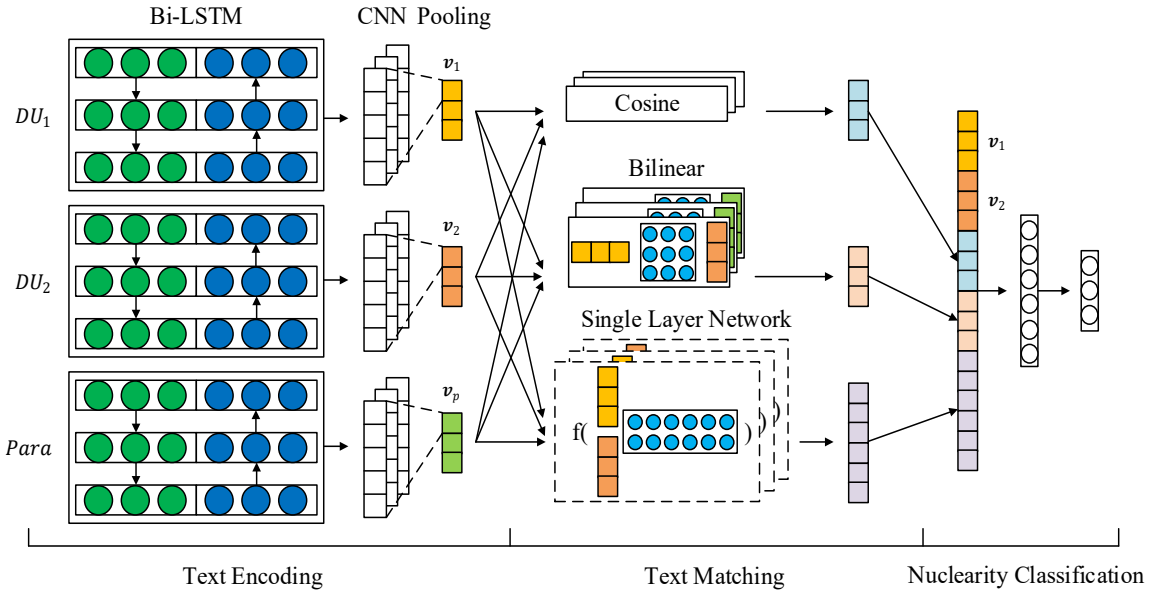


Figure 2: The basic framework of our model, including 1) Text Encoding, 2) Text Matching, and 3) Nuclearity Classification.

To recognise the nuclearity of two discourse units DU_1 and DU_2, their word sequences and the paragraph $Para$ that contains the above two units are the inputs of our model. Taking Example 1 as an instance, DU_1/DU_2 could be one of the EDUs *a*-*g* or their combinations, and $Para$ is the whole paragraph. The Text Encoding module first encodes these word sequences into the semantic vectors $\boldsymbol{v}_1, \boldsymbol{v}_2$ and $\boldsymbol{v}_p$ by Bi-LSTM and CNN. Then, these semantic representations are fed into the Text Matching module, which uses Cosine to calculate the similarity of different semantic vectors and applies Bilinear and Single Layer Network to incorporate the strong linear and nonlinear interactions between different semantic vectors. Finally, the combined feature vector, which is composed of two semantic vectors of two discourse units and one feature vector of all of the interactive information, is sent to the output layer, i.e., the Nuclearity Classification module, through a nonlinear transformation.

Our TMN model is based on two hypotheses. The first hypothesis is that there are strong correlations between the nuclearity and the semantic similarity or interactions of two discourse units. Commonly, the discourse units with similar semantics are multinuclear, and the discourse units with semantic interactions have a mononuclear relation.

The second hypothesis is that the nuclearity of two discourse units is relevant to the topic of the paragraph or document. For example, in the case of a mononuclear relation, the nucleus unit is usually semantically closer to the topic of the paragraph. Therefore, our TMN model makes the semantic match between not only the different discourse units but also the discourse unit and paragraph, by three similarity metrics, namely, the Cosine, Bilinear and Single Layer Network, which can capture the features that are related to nuclearity recognition adequately.

3.1 Text Encoding

Our Text Encoding module combines Bi-LSTM and CNN to encode the discourse unit DU_i and the paragraph $Para$, which is the modification of the Convolutional-pooling LSTM (Tan et al., 2016) in question answering.

Its input is a sequence of words $(t_1, t_2, \dots, t_T)$ in a discourse unit DU_i or a paragraph $Para$ where T is the number of words in the discourse unit or paragraph. Each word t_i in the sequence is represented as the combination of its word embedding $\boldsymbol{e}_i$ and POS (Part-Of-Speech) tag embedding $\boldsymbol{p}_i$ as follows:

$$\boldsymbol{w}_i = [\boldsymbol{e}_i, \boldsymbol{p}_i]. \tag{1}$$

LSTM models successfully keep the useful information from long-range dependency, but they focus more on the words that are behind. Due to the need for our model to treat each word equally, Bi-LSTM is introduced to the Text Encoding module. At each position t, we concatenate the output $\overrightarrow{\boldsymbol{h}_t}, \overleftarrow{\boldsymbol{h}_t} \in \mathbb{R}^l$ of the two inverted LSTMs as the output of the current word as follows:

$$\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}_t}, \overleftarrow{\boldsymbol{h}_t}]. \tag{2}$$

Therefore, each output contains not only the current word information but also the contextual information. Consequently, we choose it as the input of a 1D CNN to capture richer local n-gram information. The 1D CNN is similar to the traditional n-gram model. It can effectively capture the local interaction information between the words in the word window, and thus, it can make up for the lack of LSTM. Finally, all of the features captured by the convolution kernels are collected by the global max pooling operation to obtain the textual representation $\boldsymbol{v}_i \in \mathbb{R}^c$. In addition, the number l of LSTM neurons, the size k and the number c of the CNN convolution kernels are all hyperparameters of our model.

3.2 Text Matching

After obtaining the representations $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_P$ of the discourse units DU_1, DU_2 and the paragraph $Para$ by Bi-LSTM and CNN in the Text Encoding module, we apply the Cosine, Bilinear (Sutskever et al., 2009; Jenatton et al., 2012) and Single Layer Network (Collobert and Weston, 2008) to capture the interactions between different discourse units and between the discourse unit and the paragraph.

The cosine distance calculates the angle between two vectors, which is usually used to measure the degree of similarity. **Cosine** is defined as follows:

$$cos(\boldsymbol{v}_1, \boldsymbol{v}_2) = \frac{\boldsymbol{v}_1^{\mathrm{T}} \boldsymbol{v}_2}{\|\boldsymbol{v}_1\| \cdot \|\boldsymbol{v}_2\|}. \tag{3}$$

Because the discourse units with multinuclear relations are usually similar in content, the cosine similarity $cos(\boldsymbol{v}_1, \boldsymbol{v}_2)$ between the semantic representations of two discourse units can be used as an effective feature to determine their nuclearity. We also calculated the cosine similarities $cos(\boldsymbol{v}_1, \boldsymbol{v}_P)$ and $cos(\boldsymbol{v}_2, \boldsymbol{v}_P)$ between the discourse unit and the paragraph to measure the similarity between the discourse unit and the topic of the paragraph. These two similarities are helpful for identifying the mononuclear relations.

Bilinear is a simple way to incorporate the linear interactions between two vectors and is defined as follows:

$$s(\boldsymbol{v}_1, \boldsymbol{v}_2) = \boldsymbol{v}_1^{\top} \boldsymbol{W} \boldsymbol{v}_2, \tag{4}$$

where $\boldsymbol{W} \in \mathbb{R}^{c \times c}$ is the parameter matrix. Usually, when using the Bilinear model (Chen et al., 2016; Wan et al., 2016; Wu et al., 2017), the Bilinear value $s(\boldsymbol{h}_{x_i}, \boldsymbol{h}_{y_j}) = \boldsymbol{h}_{x_i}^{T} \boldsymbol{W} \boldsymbol{h}_{y_j}$ is calculated for any two words in the two word sequences $\{x_1, x_2, \ldots, x_m\}$ and $\{y_1, y_2, \ldots, y_n\}$ to obtain a matching matrix, where $\boldsymbol{h}_{x_i}, \boldsymbol{h}_{y_i}$ are semantic vectors that correspond to word x_i and y_j.

A discourse unit or a paragraph could contain a larger number of words, and it will lead to generating an enormous matching matrix. However, the number of training samples that can be used in our model is relatively small, which results in great difficulty with training the parameter $\boldsymbol{W}$. Therefore, we simplified this process to calculate the Bilinear values $\boldsymbol{v}_1^{\top} \boldsymbol{W} \boldsymbol{v}_2, \boldsymbol{v}_1^{\top} \boldsymbol{W} \boldsymbol{v}_P$ and $\boldsymbol{v}_2^{\top} \boldsymbol{W} \boldsymbol{v}_P$ directly on the encoded discourse units and encoded paragraphs. Since $\boldsymbol{v}_1, \boldsymbol{v}_2$ and $\boldsymbol{v}_P$ contain the semantic information of the discourse units and the paragraph, Bilinear is equivalent to capturing the linear interaction between DU_1, DU_2 and $Para$ at the textual level.

Single Layer Network is defined as follows:

$$\begin{aligned} s(\boldsymbol{v}_1, \boldsymbol{v}_2) &= f(\boldsymbol{V}_1[\boldsymbol{v}_1, \boldsymbol{v}_2] + \boldsymbol{b}_1) \\ s(\boldsymbol{v}_1, \boldsymbol{v}_P) &= f(\boldsymbol{V}_2[\boldsymbol{v}_1, \boldsymbol{v}_P] + \boldsymbol{b}_2), \end{aligned} \tag{5}$$

where $\boldsymbol{V}_1 \in \mathbb{R}^{w \times 2c}, \boldsymbol{b}_1 \in \mathbb{R}^{w}$ and $\boldsymbol{V}_2 \in \mathbb{R}^{w \times 2c}, \boldsymbol{b}_2 \in \mathbb{R}^{w}$ are parameters to incorporate nonlinear interactions between the discourse units and between the unit and the paragraph, and we choose $tanh$ as the activation function f. The number w of neurons is the hyperparameter of our model. Because of the existence of nonlinear activation functions, the Single Layer Network can capture nonlinear interactions between different discourse units and between the discourse unit and the paragraph. Bilinear focuses on capturing linear interactions, while Single Layer Network focuses on capturing non-linear interactions, and thus, Single Layer Network can make up for the lack of Bilinear to some extent. We can obtain nonlinear feature vectors $\boldsymbol{v}_{S_{12}}, \boldsymbol{v}_{S_{1P}}$ and $\boldsymbol{v}_{S_{2P}} \in \mathbb{R}^{w}$ by Single Layer Network as follows:

$$\begin{aligned} \boldsymbol{v}_{S_{12}} &= f(\boldsymbol{V}_1[\boldsymbol{v}_1, \boldsymbol{v}_2] + \boldsymbol{b}_1) \\ \boldsymbol{v}_{S_{1P}} &= f(\boldsymbol{V}_2[\boldsymbol{v}_1, \boldsymbol{v}_P] + \boldsymbol{b}_2) \\ \boldsymbol{v}_{S_{2P}} &= f(\boldsymbol{V}_2[\boldsymbol{v}_2, \boldsymbol{v}_P] + \boldsymbol{b}_2) \end{aligned} \tag{6}$$

With Cosine, Bilinear, and Single Layer Network, we measure the similarity and capture the linear and non-linear interactions among the discourse units and between the unit and the paragraph, and by training the parameter matrices $\boldsymbol{W}, \boldsymbol{V}_1, \boldsymbol{V}_2$ and the parameter vectors $\boldsymbol{b}_1, \boldsymbol{b}_2$, the matching features that play an important role in recognising nuclearity are extracted. This process is how our Text Matching module identifies important information in the discourse unit and performs text matching methods under supervised learning.

3.3 Nuclearity Classification

Based on the discourse unit DU_1, DU_2 and the paragraph $Para$, we obtain the semantic representation vectors $\boldsymbol{v}_1, \boldsymbol{v}_2$ from the Text Encoding module and the Cosine values, the Bilinear values and the nonlinear feature vectors from the Text Matching module. We concatenate all of these values and the vectors above as the input feature vector $\tilde{\boldsymbol{v}}$ of the Nuclearity Classification module as follows:

$$\begin{aligned} \boldsymbol{v}_{cos} &= [\cos(\boldsymbol{v}_1, \boldsymbol{v}_2), \cos(\boldsymbol{v}_1, \boldsymbol{v}_P), \cos(\boldsymbol{v}_2, \boldsymbol{v}_P)]^{\top} \\ \boldsymbol{v}_{bl} &= [\boldsymbol{v}_1^{\top} \boldsymbol{W} \boldsymbol{v}_2, \boldsymbol{v}_1^{\top} \boldsymbol{W} \boldsymbol{v}_P, \boldsymbol{v}_2^{\top} \boldsymbol{W} \boldsymbol{v}_P]^{\top} \\ \boldsymbol{v}_{sln} &= [\boldsymbol{v}_{S_{12}}, \boldsymbol{v}_{S_{1P}}, \boldsymbol{v}_{S_{2P}}] \\ \tilde{\boldsymbol{v}} &= [\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_{cos}, \boldsymbol{v}_{bl}, \boldsymbol{v}_{sln}] \end{aligned} \tag{7}$$

We implement the Nuclearity Classification module using a two-layer feedforward neural network. The input vector is first sent to a nonlinear transformation and then fed into a standard softmax layer, where the nonlinear transformation uses the $Relu$ function (Nair and Hinton, 2010) as follows:

$$\boldsymbol{t} = Relu(\boldsymbol{W}_t\,\tilde{\boldsymbol{v}} + \boldsymbol{b}_t) \tag{8}$$

$$\tilde{y} = softmax(\boldsymbol{W}_s\boldsymbol{t} + \boldsymbol{b}_s), \tag{9}$$

where $\boldsymbol{W}_t \in \mathbb{R}^{w_t \times (2c+6+3w)}$, $\boldsymbol{b}_t \in \mathbb{R}^{w_t}$ and $\boldsymbol{W}_s \in \mathbb{R}^{3 \times w_t}$, $\boldsymbol{b}_s \in \mathbb{R}^3$ are the parameters in the nonlinear transformation and in the softmax layer, respectively. Additionally, the number w_t of neurons in the nonlinear transformation layer is the hyperparameter of our model. During the training, we use the Adam optimiser (Kingma and Ba, 2014) to optimise the network parameters by maximising the log-likelihood loss function between the predicted label $\tilde{y}$ and the real label y.

4 Experimentation

In this section, we first introduce the CDTB corpus in the Chinese and experimental setting, and then we report and analyse the experimental results.

4.1 CDTB Corpus

Following the tree structure, the representation of nuclearity in RST and the representation of connectives in the Penn Discourse Treebank (PDTB) (Prasad et al., 2008), Li et al. (2014) built the Chinese Discourse TreeBank (CDTB) corpus based on the Chinese Treebank (CTB) (Xue et al., 2005) with a connective-driven dependency tree scheme. In CDTB, each paragraph is marked as a connective-driven dependency tree (CDT), where its leaf nodes are EDUs, its intermediate nodes represent connectives, and EDUs connected by connectives can be combined into higher level discourse units.

Similar to RST-DT, there are three types of nuclearity in CDTB: Nucleus-Satellite, Satellite-Nucleus and Nucleus-Nucleus. However, CDTB labels relations that link two discourse units on their parent node, and thus, a binary tree with *n* EDUs has only *n-1* relations. Moreover, CDTB marks paragraphs instead of documents as discourse trees, which will also lead to fewer annotated relations.

Currently, the CDTB corpus consists of 500 newswire articles, which are further divided into 2342 paragraphs with a CDT representation for one paragraph. CDTB contains 10650 EDUs, and each EDU has 22 Chinese characters on average. There are 7310 annotated relations in CDTB in which 3555 (48.6%) relations are mononuclear relations, with 2110 Nucleus-Satellite and 1445 Satellite-Nucleus, while the remaining 3755 (51.4%) relations are Nucleus-Nucleus. The reason is that COORDINATION (mononuclear/multinuclear: 468/3683) is the largest category (56.8%) in CDTB, which leads to a large number of multinuclear instances.

4.2 Experimental Setup

We evaluate our model on the corpus CDTB. Following the previous work (Kong and Zhou, 2017), we also choose the same 450 documents as the training set and 50 documents as the testing set. The specific division and labelling situation is shown in Table 1. In our evaluation, all of the non-binary trees are transformed into left binary trees, and the numbers of multinuclear relations in the converted training set and testing set are 4257 and 485, respectively. Moreover, we report the Precision (P), Recall (R) and F1 on each nuclearity type and also give the micro-average and macro-average F1.

	Training set	Testing set
Document	0001-0090, 0101-0190, 0201-0290, 0301-0325, 0400-0454, 0500-0509, 0520-0554, 0590-0596, 0600-0647	0091-0100, 0191-0200, 0291-0300, 0510-0519, 0648-0657
Nuclearity	#Nucleus-Satellite/#Satellite-Nucleus: 1901/1343 # Nucleus-Nucleus: 3371	#Nucleus-Satellite/#Satellite-Nucleus: 207/104 # Nucleus-Nucleus: 384

Table 1: Division of dataset. There are 3244 mononuclear relations and 3371 multinuclear relations in the training set, and these figures are 311 and 384 in the testing set, respectively.

The dimension of the word embeddings is set to 300, and the dimension of the POS embeddings is set to 50. We pre-trained the word embeddings with Word2Vec (Mikolov et al., 2013) on the Wikipedia Chinese corpus[1]. We used HanLP[2] to preprocess the texts, including the word segmentation and POS tagging, and we used the Keras[3] library to implement our model. All of the parameters are randomly initialised except for the word embeddings. We adopted the dropout strategy (Hinton et al., 2012) to avoid overfitting and set the dropout rate to 0.5.

We selected one-ninth of the samples from the training set as a development set to tune the hyperparameters by a grid search, and for a fair comparison, all of the models in our experiment use the same parameters. In the Text Encoding module, the number l of LSTM neurons is set to 50, and the size k and the number c of the CNN convolution kernels are set to 5 and 400, respectively, according to the empirical results in Table 2. In the Text Matching module, the number of neurons w in the Single Layer Network is set to 50. In the Nuclearity Classification module, the number w_t of neurons in the nonlinear transformation layer is set to 128.

Filter size	#Feature map				
	100	200	300	400	500
3	55.06	57.73	58.34	58.18	59.38
4	56.47	57.23	58.42	59.21	59.66
5	56.84	57.71	59.17	**59.70**	58.44
6	56.65	58.85	58.22	58.53	57.05

Table 2: Macro-average F1 with different CNN parameter settings on the development set.

4.3 Experimental Results

To exhibit the effectiveness of our TMN model, the experiment results consist of two parts: the baselines and TMN.

Baselines: We collect five baselines for our experiment: ME (Kong and Zhou, 2017), Bi-LSTM, Bi-LSTM(A), Bi-LSTM+CNN and Bi-LSTM(A)+T (Li et al., 2016). The ME model proposed by (Kong and Zhou, 2017) used contextual features, lexical features and dependency tree features to recognise nuclearity by an ME classifier. We obtained their source codes and found a data partition error in their system: some instances appeared in both the training set and the testing set. For a fair comparison, we corrected the data partition following their paper and report the revised results of their system in this paper. Considering that the attention-based hierarchical Bi-LSTM network proposed by (Li et al., 2016) performs better than the recursive neural model (Li et al., 2014), we implemented four neural network-based systems. The first is a Bi-LSTM network model (Bi-LSTM), and the second is a Bi-LSTM network model with the attention mechanisms (Bi-LSTM(A)). The third is a Bi-LSTM network model with the attention mechanisms and the tensor-based transformation function (Bi-LSTM(A)+T) (Li et al., 2016). The fourth is a Bi-LSTM+CNN model that combines Bi-LSTM and CNN.

Model	Nucleus-Satellite P / R / F1	Satellite-Nucleus P / R / F1	Nucleus-Nucleus P / R / F1	Macro-F1	Micro-F1
ME	32.2 / 15.1 / 20.5	40.0 / 15.0 / 21.8	65.6 / **87.8** / 75.0	42.3	60.5
Bi-LSTM	53.6 / 50.2 / 51.9	30.4 / 33.7 / 32.0	74.3 / 74.6 / 74.5	52.8	62.9
Bi-LSTM(A)	55.7 / 44.9 / 49.7	34.9 / 36.5 / 35.7	74.6 / 80.0 / 77.2	54.4	65.2
Bi-LSTM+CNN	59.6 / 46.4 / 52.1	**40.2** / 31.7 / 35.5	73.2 / 83.5 / 78.0	55.7	67.1
Bi-LSTM(A)+T	56.8 / **50.7** / 53.6	37.5 / 43.4 / 40.2	**77.0** / 77.9 / 77.5	57.2	66.3
TMN	**69.1** / 45.4 / **54.8**	39.2 / **49.0** / **43.6**	76.2 / 83.3 / **79.6**	**60.4**	**69.0**

Table 3: The experimental results of five baselines and TMN.

The experimental results of the above models are shown in Table 3, and these results show that our

[1] https://dumps.wikimedia.org/zhwiki/
[2] https://github.com/hankcs/HanLP
[3] https://keras.io/

TMN model outperforms the other five baselines in both the micro-average and macro-average F1. Compared with the traditional method ME, the other five neural network models improve the micro-average and macro-average F1 significantly, especially the macro-average F1, with large gains from 10.5 up to 18.1. These results justify the effectiveness of the neural network models on the nuclearity recognition to capture the deeper semantic information that is hiding in the discourse units.

Compared with Bi-LSTM, Bi-LSTM(A) improves the macro-average and micro-average F1 by 1.6 and 2.3, respectively, because Bi-LSTM(A) can pick up prominent semantic information on the output of Bi-LSTM using the attention mechanism. Furthermore, the performance of Bi-LSTM(A)+T is better than Bi-LSTM(A), and this result ensures that tensor-based transforms are also helpful to Bi-LSTM. Moreover, due to the Bi-LSTM+CNN model combining the ability of capturing the global information by Bi-LSTM and the local information by CNN, it performs better than both Bi-LSTM and Bi-LSTM(A).

Our TMN model outperforms all of the other five models, with large gains from 3.2 up to 18.1 in the macro-average F1 and a significant gain from 1.9 up to 8.5 in the micro-average F1. Compared with the Bi-LSTM, Bi-LSTM(A) and Bi-LSTM+CNN, which focus on obtaining the representations of text, our TMN model can capture the semantic features and incorporates interactions between the representations of the discourse units and the paragraphs. Moreover, compared with the Bi-LSTM(A)+T model, our TMN combines both Bi-LSTM and CNN in the Text Encoding module and uses many simple but efficient methods to incorporate richer interactions in the Text Matching module.

4.4 Analysis

We also compare the performances of the different nuclearity types, and Table 3 shows that the performance of multinuclear (Nucleus-Nucleus) is much higher (>24 in F1) than that of the mononuclear relations (Nucleus-Satellite and Satellite-Nucleus). This result derives from two aspects. The first is that the majority of the training set is Nucleus-Nucleus, which occupies 56.8% of all annotated nuclearity, while the percentages of the Nucleus-Satellite and Satellite-Nucleus are 25.3% and 17.9%, respectively. The second is that our Text Matching module is helpful for identifying similar discourse units via the matching mechanisms of Cosine, Bilinear and Single Layer Network, and then assigns Nucleus-Nucleus to them.

The Bi-LSTM+CNN model is a simplified version of TMN that does not use the Text Matching module. Compared with the Bi-LSTM+CNN model, TMN combines the semantic similarity and the interaction information simultaneously to improve the macro-average and micro-average F1 by 4.7 and 1.9, respectively. These figures justify our first hypothesis that there are strong correlations between the nuclearity and the semantic similarity or the interactions of two different discourse units.

To analyse the contribution of each mechanism in the Text Matching module, we conduct experiments on some variants of TMN, and the results are shown in Table 4. In Table 4, the basic model (TMN-CBS) is equal to Bi-LSTM+CNN, which does not have the Text Matching module. The TMN-BS model refers to the TMN model whose Text Matching module only uses Cosine, while TMN-C refers to the TMN model whose Text Matching module only uses Bilinear and Single Layer Networks.

Model	Nucleus-Satellite P / R / F1	Satellite-Nucleus P / R / F1	Nucleus-Nucleus P / R / F1	Macro-F1	Micro-F1
TMN-CBS	59.6 / 46.4 / 52.1	40.2 / 31.7 / 35.5	73.2 / 83.5 / 78.0	55.7	67.1
TMN-BS	56.5 / 50.7 / 53.4	**47.6** / 28.9 / 35.9	74.2 / **83.7** / 78.7	56.8	68.0
TMN-C	61.5 / **54.1** / **57.6**	39.6 / 34.6 / 36.9	75.7 / 81.7 / 78.6	57.8	68.3
TMN-P	60.9 / 51.2 / 55.6	34.7 / 41.4 / 37.7	**76.9** / 79.0 / 77.9	57.3	66.8
TMN	**69.1** / 45.4 / 54.8	39.2 / **49.0** / **43.6**	76.2 / 83.3 / **79.6**	**60.4**	**69.0**

Table 4: Experimental results of variants of the TMN Model.

Compared with TMN-CBS, the TMN-BS model and the TMN-C model improve the macro-average and micro-average F1, and these improvements show that the semantic similarity or interaction information captured by the Cosine, Bilinear, and Single Layer Network are helpful for nuclearity recognition. Especially after adding the interaction information using the Bilinear and Single Layer Network, the F1 of the relation Nucleus-Satellite achieves a 5.5% improvement.

The TMN-P model is a simplification of the TMN model, which removes the input of the paragraph *Para* in Figure 2. TMN-P captures only the similarity and interaction information between the different discourse units, without the similarity and interaction information between the discourse unit and the paragraph. Compared with TMN-P, TMN significantly improves the macro-average and micro-average F1 by 3.1 and 2.2, respectively, which justifies our second hypothesis that nuclearity recognition is relevant to the topic of the paragraph.

Nuclearity	Nucleus-Satellite	Satellite-Nucleus	Nucleus-Nucleus
Nucleus-Satellite	-	14.5%	40.1%
Satellite-Nucleus	9.6%	-	41.4%
Nucleus-Nucleus	6.6%	10.1%	-

Table 5: The percentages of misclassified samples.

Table 5 shows the error statistics of our TMN model in nuclearity recognition. It shows that 40.1% of the Nucleus-Satellite instances and 41.4% of the Satellite-Nucleus instances are frequently identified as Nucleus-Nucleus by our TMN model. These results show that the errors mainly arise from the judgment of whether an instance is mononuclear or multinuclear. This finding is mainly due to two reasons: 1) the number of Nucleus-Nucleus instances accounts for more than half of the training set; and 2) many discourse units differ in nuclearity but are semantically similar. We consider the following two discourse units as examples.

Example 2: 农业获得较好收成 $_{a}$，全年粮食总产量达七十六点六亿公斤 $_{b}$。 *Farming received good harvests* $_{a}$, *the total grain output in the year amounted to 7.66 billion kg* $_{b}$.

The nuclearity type between the two EDUs *a* and *b* in Example 2 is Nucleus-Satellite due to the content in EDU *a* being more generalised and being able to semantically contain the content described in EDU *b*. However, there is a strong correlation between "农业 *farming*" and "粮食 *grain*" and between "收成 *harvest*" and "产量 *output*", at a semantic level. Therefore, the Text Matching module will misjudge their relation as Nucleus-Nucleus via the similarity and interaction information between the two EDUs *a* and *b*.

5 Conclusions

In this paper, we propose a novel TMN model for nuclearity recognition in Chinese discourse. First, we employ a Text Encoding module to capture both the global dependency information and the local n-gram information via Bi-LSTM and CNN. In this way, the overall discourse semantics can be much better represented. Then, we employ a Text Matching module to capture various similarities and interactions between different discourse units and between the encoded unit and the paragraph by the Cosine, Bilinear and Single Layer Network. Here, while Cosine calculates the semantic similarity, Bilinear and Single Layer Network incorporate the strong linear and nonlinear interactions between the semantic vectors. Experimental results on the CDTB corpus show that our TMN model significantly outperforms various strong baselines both in micro-average and macro-average F1. Our future work will focus on how to better tune the input of our neural network model and apply this model to other languages.

Acknowledgements

The authors would like to thank three anonymous reviewers for their comments on this paper. This research was supported by the National Natural Science Foundation of China under Grant Nos. 61772354, 61773276 and 61673290, and was also supported by the Strategic Pioneer Research Projects of Defense Science and Technology under Grant No. 17-ZLXDXX-02-06-02-04.

Reference

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current and new directions in discourse and dialogue*. Springer, Dordrecht, pages 85-112.

Jifan Chen, Qi Zhang, Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Implicit discourse relation detection via a deep architecture with gated relevance network. In *ACL 2016.* pages 1726-1735.

Xiaomin Chu, Zhongqing Wang, Qiaoming Zhu, and Guodong Zhou. 2015. Recognizing nuclearity between chinese discourse units. In *IALP 2015*. IEEE, pages 197-200.

Ronan Collobert, and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML 2008*. ACM, pages 160-167.

Vanessa Wei Feng, and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *ACL 2014*. pages 511-521.

Michael Heilman, and Kenji Sagae. 2015. Fast rhetorical structure theory discourse parsing. *arXiv preprint arXiv:1505.02425*.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *Computer Science,* 3(4):212-223.

Mikel Iruskieta, Arantza Díaz de Ilarraza, and Mikel Lersundi. 2014. The annotation of the central unit in rhetorical structure trees: A key step in annotating rhetorical relations. In *COLING 2014*. pages 466-475.

Rodolphe Jenatton, Nicolas L. Roux, Antoine Bordes, and Guillaume R. Obozinski. 2012. A latent factor model for highly multi-relational data. In *NIPS 2012*. Curran Associates Inc, pages 3167-3175.

Yangfeng Ji, and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *ACL 2014*. pages 13-24.

Shafiq Joty, Giuseppe Carenini, Raymond Ng, and Yashar Mehdad. 2013. Combining intra-and multi-sentential rhetorical parsing for document-level discourse analysis. In *ACL 2013*. pages 486-496.

Diederik P. Kingma, and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Fang Kong, and Guodong Zhou. 2017. A CDT-styled end-to-end Chinese discourse parser. In *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP 2017)*. 16(4):26.

Jiwei Li, Rumeng Li, and Eduard Hovy. 2014. Recursive deep models for discourse parsing. In *EMNLP 2014*. pages 2061-2069.

Qi Li, Tianshi Li, and Baobao Chang. 2016. Discourse parsing with attention-based hierarchical neural networks. In *EMNLP 2016*. pages 362-371.

Yancui Li, Fang Kong, and Guodong Zhou. 2014. Building Chinese discourse corpus with connective-driven dependency tree structure. In *EMNLP 2014*. pages 2105-2114.

Yancui Li, Jing Sun, Wenhe Feng, Guodong Zhou. 2015. The platform of Chinese discourse structure analysis based on connective-driven dependency tree. In *China National Conference on Computational Linguistics (CCL 2015)*.

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

William C. Mann, and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse* 8(3):243-281.

Daniel Marcu. 2000. *The theory and practice of discourse parsing and summarization*. MIT press.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

Vinod Nair, and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML 2010*. Omnipress, pages 807-814.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K. Joshi, and Bonnie L. Webber. 2008. The Penn Discourse Treebank 2.0. In *LREC 2008*. pages 2961-2968.

Manfred Stede. 2008. RST revisited: Disentangling nuclearity. '*Subordination' versus 'Coordination' in Sentence and Text: A cross-linguistic perspective*. pages 33-59.

Ilya Sutskever, Joshua B. Tenenbaum, and Ruslan R. Salakhutdinov. 2009. Modelling relational data using Bayesian clustered tensor factorization. In *NIPS 2009*. pages 1821-1828.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *ACL 2016*. pages 464-473.

Suzan Verberne, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. 2007. Evaluating discourse-based answer extraction for why-question answering. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pages 735-736.

Yizhong Wang, Sujian Li, and Houfeng Wang. 2017. A two-stage parsing method for text-level discourse analysis. In *ACL 2017*. pages184-188.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *AAAI 2016*. AAAI Press, pages 2835-2841.

Yu Wu, Wei Wu, Chen Xing, Zhoujun Li, and Ming Zhou. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *ACL 2017*. pages 496-505.

Naiwen Xue, Fei Xia, Fudong Chiou, and Marta Palmer. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural language engineering* 11(2): 207-238.

Bowei Zou, Guodong Zhou, and Qiaoming Zhu. 2014. Negation focus identification with contextual discourse information. In *ACL 2014*. pages 522-53.

Joint Modeling of Structure Identification and Nuclearity Recognition in Macro Chinese Discourse Treebank

Xiaomin Chu[1], **Feng Jiang**[1], **Yi Zhou**[1], **Guodong Zhou**[1], **Qiaoming Zhu**[1,2,†]
[1]School of Computer Science and Technology, Soochow University, China
[2]Institute of Artificial Intelligence, Soochow University, China
{xmchu,fjiang,yzhou}@stu.suda.edu.cn; {gdzhou,qmzhu}@suda.edu.cn

Abstract

Discourse parsing is a challenging task and plays a critical role in discourse analysis. This paper focus on the macro level discourse structure analysis, which has been less studied in the previous researches. We explore a macro discourse structure presentation schema to present the macro level discourse structure, and propose a corresponding corpus, named Macro Chinese Discourse Treebank. On these bases, we concentrate on two tasks of macro discourse structure analysis, including structure identification and nuclearity recognition. In order to reduce the error transmission between the associated tasks, we adopt a joint model of the two tasks, and an Integer Linear Programming approach is proposed to achieve global optimization with various kinds of constraints.

1 Introduction

A typical document is usually organized in a coherent way that each discourse unit is relevant to its context and plays a role in the entire semantics. Discourse structure analysis not only helps to understand the discourse structure and semantics, but also can benefit variety of downstream applications including question answering (Sadek and Meziane, 2016), machine translation (Guzmán et al., 2014), text summarization (Ferreira et al., 2014; Cohan and Goharian, 2017), and so forth.

There exist two hierarchical levels of discourse structures: micro level and macro level. The micro level structure refers to the structure and relation among the discourse units in a sentence, or consecutive sentences, or sentences groups. The macro level structure refers to the structure and relation among paragraphs, or chapters, or discourses. Corresponding to related research based on Rhetorical Structure Theory Discourse Treebank (RST-DT), the micro level is similar to the sentence-level discourse parsing, and the macro level is similar to the document-level discourse parsing.

To make a clearer explanation of the macro discourse structure, take the chtb_0019 as an example, which is a typical news article from Chinese Treebank 8.0 (Xue et al., 2013). The macro discourse structure of this article is shown as Figure 1. There are five paragraphs (P1, P2, P3, P4 and P5) in the news "Significant achievements in the construction of Ningbo Bonded Area". The paragraphs are connected by discourse relations (*Elaboration*, *Background*, *Joint*). In this article, paragraph P1 points out the theme of the overall article. Depending on the direction of arrows from the root node to the leaf node in the discourse structure tree, the most important part can be quickly located.

Limited to the length of this paper, the full discourse text of this example is not included, please refer to the corpus. The main contents of the five paragraphs respectively are: P1) Ningbo Bonded Area achieved fruitful results after three years of construction; P2) the basic situation of the Ningbo Bonded Area; P3) the situation of import and export trade, warehouses, storage area, etc. P4) the situation of industrial processing projects and enterprises; P5) the situation of administrative services and information construction.

† The corresponding author is Qiaoming Zhu.

Proceedings of the 27th International Conference on Computational Linguistics, pages 536–546
Santa Fe, New Mexico, USA, August 20-26, 2018.

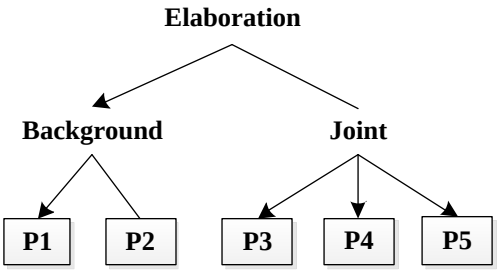


Figure 1: Macro structures tree of chtb_0019.

From the discourse structure tree of this example, we can see that the analysis of discourse structure is beneficial for the understanding of the content and the theme of the discourse. Based on the macro discourse structure analysis, we can further enhance the performance of natural language processing applications. For example, we can use the information of discourse structure to summarize the content and purpose of an article, use the information of discourse relation to assist the construction of question answering system, and improve the performance of automatic summarization by using the nuclearity information.

The existing research on discourse structure is mainly focused on the micro level, and the performance has not yet achieve the level for application. However, the macro level research still stays in the theoretical research, and there is no available corpus resource, nor a corresponding computational model. For the two reasons mentioned above, in this paper, we take the macro discourse structure as the main research object, that is different from the previous research. We explore a macro discourse structure presentation schema to present the macro level discourse structure, and propose a Macro Chinese Discourse Treebank (MCDTB) on the top of existing Chinese Discourse Treebank (CDTB) (Li et al., 2014). On the basis of the presentation schema and annotated corpus, we divide the macro level discourse structure analysis into four tasks, including structure identification, nuclearity recognition, relation classification and discourse tree building.

There are certain differences from the analysis of the micro level discourse structure. For example, macro discourse structure analysis takes paragraphs as the elementary discourse units, and the relations between the units are fairly loose, so it brings difficulties to the task of structure identification. Furthermore, there are virtually no connectives between the discourse units, and the texts of each discourse unit are relatively long, making discourse relation classification lack of effective cue phrase and lexical information.

The task of discourse structure identification is the first and the most crucial step in the macro discourse analysis and also the basis step of further tasks. Nuclearity recognition is only part of the discourse relation classification task in the existing research, and has not been given sufficient attention. However, in our study, we find that the performance improvement of nuclearity recognition contributes to the improvement of the overall performance. Therefore, in this paper, we concentrate on these two tasks of macro discourse structure, structure identification and nuclearity recognition, and take structure identification as the main task.

Our contribution is three-fold. First, we explore a macro discourse structure presentation schema to present the macro level discourse structure, and propose a macro discourse structure corpus, named Macro Chinese Discourse Treebank (MCDTB). The presentation schema and corpus resource can lay the foundation for macro discourse structure analysis. Second, we propose discourse structure identification and nuclearity recognition models on macro level discourse structure analysis. By using CRF models to label sequences of discourse units, we can incorporate contextual information in a more natural way, and achieve a satisfactory performance. Third, we propose a joint model of structure identification and nuclearity recognition to reduce the error transmission between the associated tasks, and achieve global optimization via Inter Linear Programming.

The rest of this paper is organized as follows. Section 2 overviews related work on discourse parsing.

Section 3 introduces our macro discourse structure presentation schema and corpus resources. Section 4 introduce the framework of our joint model of structure identification and nuclearity recognition. Section 5 describes the local models, and the joint approach we used with ILP is introduced in Section 6. Section 7 presents the experimental results. Section 8 gives the conclusion and future work.

2 Related Work

Discourse parsing is the task of discovering the presence and type of the discourse relations between discourse units. The existing discourse parsing researches are mainly based on Rhetorical Structure Theory Discourse Treebank (RST-DT). The RST-DT (Carlson et al., 2003) is built in the framework of Rhetorical Structure Theory, consisting of 385 Wall Street Journal articles from the Penn Treebank (Marcus et al., 1993) and representing over 176,000 words of text.

While recent advances in sentence-level discourse parsing have attained accuracies close to human performance (Joty et al., 2012), discourse parsing at the document-level still poses challenges.

The HILDA discourse parser (Hernault et al., 2010) is the first attempt at document-level discourse parsing on RST-DT. It adopts a pipeline framework, and greedily builds the discourse tree from the bottom-up. In particular, at each step of the tree-building, a binary Support Vector Machine (SVM) classifier is applied to determine which pair of adjacent discourse constituents should be merged to form a larger span, and then another multi-class SVM classifier is applied to assign the type of discourse relation between the chosen pair of constituents.

Joty et al. (2013) approach the document-level discourse parsing using a model trained by Conditional Random Fields (CRF). They decomposed the problem of document-level discourse parsing into two stage: intra-sentential and multi-sentential parsing. Specifically, they employed two separate models for intra- and multi-sentential parsing. They jointly modeled the structure and the relation for a given pair of discourse units, such that information from each aspect can interact with the other.

Feng and Hirst (2014) develop a much faster model whose time complexity is linear in the number of sentences. Their model adopts a greedy bottom-up approach, with two linear-chain CRFs applied as local classifiers. An approach of post-editing is performed, which modified a fully-built tree by considering information from upper-levels, to improve the accuracy.

There is no relevant research on document-level discourse parsing in Chinese so far. For micro level discourse structure analysis, Li (2015) proposes a Connective-driven Dependency Tree (CDT) schema to represent the discourse rhetorical structure in Chinese language, with elementary discourse units as leaf nodes and connectives as non-leaf nodes, largely motivated by the Penn Discourse Treebank and the Rhetorical Structure Theory. On this basis, a Chinese Discourse Treebank (CDTB) consisting of 500 discourses is annotated, and a Chinese discourse structure analysis platform is realized.

3 Macro Chinese Discourse Treebank

3.1 Macro Discourse Structure Representation Schema

Rhetorical Structure Theory (RST) (Mann and Thompson, 1987), one of the most influential theories of discourse, represents a discourse by a hierarchical structure, called discourse tree. The leaves of a discourse tree correspond to Elementary Discourse Units (EDUs). Adjacent EDUs are connected by rhetorical relations, forming larger discourse units. RST defines two different types of discourse units, in which the nucleus is considered as the central part, and the satellite is considered as the peripheral part.

A concept of "macrostructures" is put forward by Van Dijk (1980) in Macrostructure Theory. The point of "macrostructures" is that texts not only have local or micro structural relations between subsequent sentences, but also have overall structures that define their global coherence and organization. But even today, the crucial global structures (including macrostructures, superstructures) that define the overall meaning and form of texts are almost ignored.

Inspired by Van Dijk's Macrostructure Theory and Rhetorical Structure Theory, we explore a macro discourse structure representation schema. In this representation schema, each discourse is represented as a hierarchical discourse tree (as shown in Figure 1). In the macro discourse structure tree, leaf nodes

represent paragraphs, and non-leaf nodes represent discourse relations. The edges connect the discourse units, with the arrows pointing to the "Nucleus" units.

Detailed definitions of macro discourse structure are described as follows.

Leaf nodes: Unlike the definition on the micro level (the elementary units are treated as leaf nodes), we directly treat the paragraphs which are naturally segmented in the discourses as leaf nodes on the macro level.

Non-leaf nodes: Discourse relations connect discourse units, which are treated as non-leaf nodes in our macro discourse structure. We classify the discourse relations into three categories and fifteen subcategories, including **Coordination** (Joint, Sequence, Progression, Contrast, Supplement), **Causality** (Cause-Result, Result-Cause, Background, Behavior-Purpose, Purpose-Behavior), and **Elaboration** (Elaboration, Summary, Evaluation, Statement-Illustration, Illustration-Statement).

Arrow pointing: A discourse unit linked by a discourse relation can be either a "Nucleus" or a "Satellite" depending on how central the message is. In the macro discourse structure, we use the arrows pointing to represent the "Nucleus-Satellite" relations. Specifically, the edges with arrows point to the "Nucleus" units, and the edges without arrows point to the "Satellite" units.

3.2 Corpus Annotating

Guided by the macro discourse structure framework defined in Section 3.1, we have carried out annotating work of macro Chinese discourse structure, which we call Macro Chinese Discourse Treebank (MCDTB)[1]. In the process of annotating, the structure definition and annotating criteria are modified iteratively. After nearly a year of annotation, 720 news wire articles are annotated, the source of which is Chinese Treebank 8.0 (CTB 8.0). (Xue et al., 2002; Xue et al., 2013)

Because the discourse units are not isolated from the overall discourse, it's difficult to judge whether the discourse units are important or not and what relations are between the discourse units simply from the units themselves. It is necessary to have a comprehensive understanding of the overall article before the annotating.

We divide the annotating work into three main stages. **The first stage** lasted four months, with three annotators participating. We selected the first 50 news articles from CTB 8.0, and annotated them together. After a lot of discussions, a preliminary annotating specification was formed. **The second stage** lasted for three months. Three annotators annotated articles independently and discussed the annotating result in groups. At the same time, the imperfect parts of the representation schema and annotating specification were discussed and amended. **The third stage**, which lasted four months, we added three new annotators to improve the efficiency of the annotating. Six annotators were divided into three groups, and each group was composed of a new staff and an old staff. The annotators of each group annotated independently and discussed in groups.

To ensure the quality of our corpus, we adopt the annotating consistency using agreement and kappa. Table 1 illustrates the annotating consistency in detail. We measure the agreement and kappa of discourse structures, nuclearity and discourse relations in the second and the third stage respectively. The method of consistency calculation used in this paper refers to the work of the corpus of RST (Marcu et al., 1999), and the appropriate adjustment is made according to the contents of our annotation.

Annotating Consistency of the Stage 2			**Annotating Consistency of Stage 3**		
Categories	**Agreement**	**Kappa**	**Categories**	**Agreement**	**Kappa**
Structure	88.54%	0.771	Structure	86.07%	0.671
Nuclearity	80.67%	0.694	Nuclearity	83.35%	0.647
Relation	83.05%	0.556	Relation	80.20%	0.597

Table 1: Annotating consistency

After the annotating work finished, the corpus consists of 720 newswire articles with a total of 398,829

[1]The Macro Chinese Discourse TreeBank is available at https://figshare.com/s/250474dba44e4161b040.

Chinese characters. 3,981 paragraphs with 8,391 sentences are annotated. There are 5.53 paragraphs and 554 Chinese characters in each article on average. 2,870 discourse relations are annotated.

4 Overview of Framework

Figure 2 demonstrates the framework of our joint model of structure identification and nuclearity recognition. The test dataset is processed first, and it becomes groups of discourse units' sequences to be labeled. Then we perform two CRF models to identify the structure and recognize the nuclearity using different feature sets respectively. In order to reduce the error transmission between the associated tasks, we adopt a joint model of the two tasks, and an Integer Linear Programming (ILP) approach is proposed to achieve global optimization with various kinds of constraints. In this way, a joint model of two layers is formed.

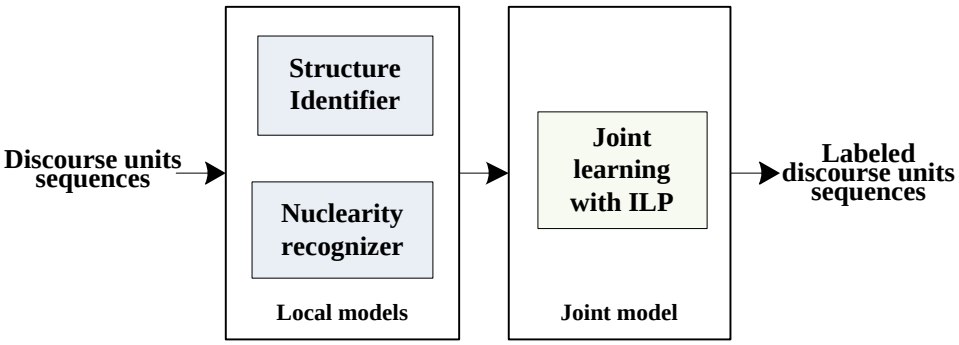


Figure 2: Joint model framework.

5 Local Models

We convert the macro discourse structure and the nuclearity prediction tasks into the sequence labeling problems. It has the following advantages of these conversions. 1) Context information can be fused conveniently. We use a window to capture the features of the previous and next discourse units. 2) Make the prediction process more naturally. In previous studies, in order to classify the structure and relations, methods of left join and right join were used to convert multiple relations to binary relations. There are two shortcomings of these approach, on the one hand, the number of non-original samples is added, and on the other hand, it is difficult to automatically build a complete real structure tree.

We build the local models of structure identification and nuclearity recognition respectively. Our local models are implemented using CRFs. In this way, we are able to take into account the sequential information from contextual discourse units, which cannot be naturally represented with Support Vector Machine (SVM) or Maximum Entropy (ME) as local classifiers.

As shown by Feng and Hirst (2012; 2014), for a pair of discourse units of interest, the sequential information from contextual units is crucial for determining structures. Therefore, it is well motivated to use CRF, which is a discriminative probabilistic graphical model, to make predictions for a sequence of units surrounding the pair of interest.

Figure 3 shows our structure identification model M_{struct} implemented with conditional random field algorithm. The first layer of the chain is composed of discourse units U_j's, and the second layer is composed of nodes of S_j's to indicate the probability of merging adjacent discourse units. When there is a relation between the discourse unit U_j and the previous one U_{j-1}, the structure of S_j is labeled as "1", and on the other hand, when there is no relation between the two consecutive discourse units, the structure of S_j is labeled as "0". To improve the accuracy of the structure identification, we enforce additional commonsense constraints in its Viterbi decoding. In particular, we disallow the existence of all-zero sequences (at least one pair must be merged).

The nuclearity recognition model $M_{nuclear}$ works in a similar way to M_{struct}, in which the first layer of the chain is composed of discourse units U_j's, and the second layer is composed of nodes of N_j's to

indicate the probability of nuclearity between the adjacent discourse units. When the current discourse unit U_j is more important than the previous discourse unit U_{j-1}, N_j is labeled as "1". When the current discourse U_j is less important than U_{j-1}, N_j is labeled as "2". When the two discourse units are equally important, the N_j is labeled as "3".

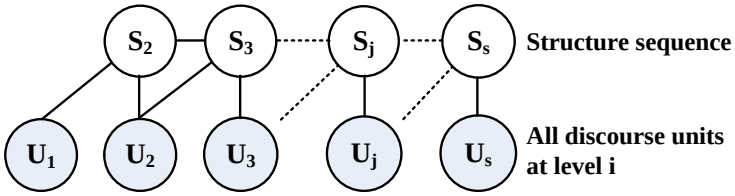


Figure 3: Local structure identification model.

In our local models, to encode two adjacent units, U_j and U_{j+1}, within a CRF chain, we use the following features listed in Table 2, some of which are modified from (Joty et al., 2013)'s and (Feng and Hirst, 2014)'s models.

Some of the helpful features of the RST discourse parsing cannot be used in macro discourse analysis or in Chinese discourse analysis. For example: 1) For macro level discourse analysis, the particles of the N-gram model are too small to represent the information of a paragraph, so the lexical features are not used in our tasks. 2) Syntactic information and dominance set features are very useful for micro level discourse analysis. However, the elementary units of the macro level are paragraphs, and these features are not applicable. 3) Since there is no tense in Chinese, we cannot use temporal features in macro level structure analysis.

Due to the appearance of word vector representation (Mikolov et al., 2013), the methods of co-occurrence (Sporleder and Lascarides, 2004) and word pairs (Feng and Hirst, 2012) are not necessary when the semantic similarity is calculated. We use the word2vec model to train word vector representation on the CTB 8.0, and use the method proposed by Jiang et al. (2018) to calculate the semantic similarity (including the semantic similarity between adjacent discourse units and the similarity between the discourse unit and the topic). In particular, in order to prevent the sparsity of the features, we discretize the semantic similarity into 10 levels.

Features	Used in SI	Used in NR
Organization features		
The beginning and end location of U_j.	Y	Y
Distances of U_j to the beginning and to the end.	Y	Y
Number of sentences (or paragraphs) in U_j.	Y	N
Whether U_j contains more sentences (or paragraphs) than U_{j+1}.	N	Y
Tree structure features		
Whether U_j is a bottom-level constituent.	Y	Y
Whether U_j is a combined unit in the previous step.	Y	Y
Similarity features		
The similarity between U_j and U_{j+1}.	Y	Y
The similarity between U_j and the topic of the discourse.	Y	Y
The similarity between U_{j+1} and the topic of the discourse.	Y	Y
Whether the similarity between U_j and the topic is greater than the similarity between U_{j+1} and the topic.	N	Y

Table 2: Features used in local models.

6 Joint Learning with Integer Linear Programing

While a pipeline model may suffer from the errors propagated from upstream tasks, a joint model can benefit from the close interaction between two or more tasks. Recently, joint modeling has been widely attempted in various NLP tasks, such as joint syntactic parsing and semantic role labeling (Li et al., 2010), joint argument identification and role determination (Li et al., 2013), joint structure identification and relation recognition (Joty et al., 2012), etc.

In our joint model, an ILP (Integer Logic Programming) -based inference framework is introduced to integrate two CRF-based local models, the structure identifier and the nuclearity recognizer. In this section, we propose a joint model of structure identification and nuclearity recognition with some intra-instance and contextual constraints.

We assume $p_{SI}(s_{<i,j>}|seq_i)$ the probability of M_{struct} identifying $s_{<i,j>}$ as a structure of an sequence seq_i, where $s_{<i,j>}$ is the jth structure to be identified in the ith sequence seq_i. We define following assignment costs with $-log$:

$$c^{SI}_{<i,j>} = -log(p_{SI}(s_{<i,j>}|seq_i)) \tag{1}$$

$$c^{-SI}_{<i,j>} = -log(1 - p_{SI}(s_{<i,j>}|seq_i)) \tag{2}$$

where $c^{SI}_{<i,j>}$ and $c^{-SI}_{<i,j>}$ are the cost of $s_{<i,j>}$ whether or not a structure in sequence seq_i respectively.

There are three types of nuclearity labels between discourse units, including "NS", "SN", and "NN". In addition, when there is no structure between the two successive units, we add a "NO-STR" label to distinguish it. The label of nuclearity could be represented as a 4-dimension vector and the value of each element in the vector is either 1 or 0 denoting whether the corresponding label is assigned or not. For instance, $n_{<i,j>}$=[1,0,0,0] denotes the assigned label is "NO-STR" and $n_{<i,j>}$=[0,1,0,0] denotes the assigned label is "NS", where $n_{<i,j>[k]}$ denotes the kth label in the vector and $p_{NR}(n_k|s_{<i,j>})$ denotes the probability belonging the kth label. The cost of nuclearity recognition can be defined as follow:

$$c^{NR}_{<i,j>[k]} = -log(p_{NR}(n_k|s_{<i,j>})) \tag{3}$$

where $c^{NR}_{<i,j>[k]}$ is the cost of assigning or not assigning nuclearity n_k to $s_{<i,j>}$.

Besides, we use indication variable $x_{<i,j>}$ which is set to 1 if $s_{<i,j>}$ is a structure of seq_i, and 0 otherwise. Similar to $x_{<i,j>}$, we use another indicator variable $y_{<i,j,k>}$ which is set to 1 if the $s_{<i,j>}$ has the kth nuclearity label, and 0 otherwise. The objective function for the overall sample set can be represented as follows, where D denotes the overall sample set.

$$min \sum_{seq_i \in D} (\sum_{s_{<i,j>} \in seq_i} (c^{SI}_{<i,j>} x_{<i,j>} + c^{-SI}_{<i,j>}(1 - x_{<i,j>}) + \sum_{k=0}^{3} c^{NR}_{<i,j>[k]} y_{<i,j,k>})) \tag{4}$$

Subject to

$$x_{<i,j>} \in \{0, 1\} \tag{5}$$

$$y_{<i,j,k>} \in \{0, 1\} \tag{6}$$

Constraints (5) and (6) are used to make sure that $x_{<i,j>}$ and $y_{<i,j,k>}$ are binary values.

Furthermore, we enforce following constraints (C1 and C2) on the consistency between SI and NR.

(C1) **Nuclearity type constraints:** the task of nuclearity recognition is a single-label classification problem. That is, the label of an instance could be only one option.

$$\sum_{k=0}^{3} y_{<i,j,k>} = 1 \tag{7}$$

(C2) **Correlation constraints:** if $s_{<i,j>}$ is a structure, it must have a nuclearity label, otherwise, if $s_{<i,j>}$ is not a structure, it must not have a nuclearity label.

$$\sum_{k=1}^{3} y_{<i,j,k>} = x_{<i,j>} \tag{8}$$

Similar to local models, we disallow the existence of all-zero sequences, so we add the constraint C3 to the joint model. This constraint also lays the foundation for subsequent task of discourse tree building.

(C3) **Contextual constraints:** a sequence must have at least one structure. In order to avoid the ILP model optimizing the sequences into all-zero sequences, which has been already constrained by the Viterbi algorithm in the local models, this constraint is added.

$$\sum_{s_{<i,j>} \in seq_i} x_{<i,j>} \geq 1 \tag{9}$$

7 Experiments

In this section, we first describe the experimental setting and then evaluate our joint model of structure identification and nuclearity recognition on MCDTB.

7.1 Experimental Setting

Data: We use the the corpus annotated by ourselves for experiment, and the detailed corpus data is described in the Table 3.

Statistics Items	Value	Statistics Items	Value
Count of documents	720	Amount of sentences	8,391
Count of paragraphs	3,981	Average paragraphs (paragraphs/document)	5.53
Maximal of paragraphs	22	Average sentences (sentences/paragraph)	2.1
Minimal of paragraphs	2	Average characters (characters /paragraph)	554

Table 3: Corpus statistic data

There are 8,863 instances in MCDTB, and we use fivefold cross-validation to ensure the objectivity of the experiment. In particular, we divide the articles into 5 datasets and assign articles of the different lengths (length means the number of paragraphs of a discourse) into the 5 datasets relatively equally, so that the size of each data set is nearly the same. The number of discourses of different lengths is shown in Table 4.

Length	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**>13**
Number	29	112	159	144	91	58	37	33	15	13	14	15

Table 4: Discourses of different lengths

Classification Algorithm: The CRF tool (*CRF++*)[2] is employed to train individual component classifiers and *lp_solver*[3] is used to construct the joint model.

Evaluation Measurement: The performance is evaluated using the standard accuracy measurement.

7.2 Experimental Results

Table 5 compares the performance of local models when different feature sets are leveraged. The features used in local models are mentioned in Section 5. This table indicates the structure features make the greatest contribution to the local models. Although the performances of tree structure and similarity features is not good when used alone, combining with the organization features improves the performance of the model. Especially, when both these two kinds of features are used, the performance reaches the best

[2]http://crfpp.googlecode.com/

[3]http://lpsolve.sourceforge.net/5.5/

Features	Structure		Nuclearity	
	Accuracy	Macro-F1	Accuracy	Macro-F1
Organization	76.13	74.46	74.46	49.17
Tree Structure	74.48	73.38	70.21	36.86
Similarity	74.57	73.32	66.68	39.70
Organization + Tree Structure	76.31	74.64	74.70	47.25
Organization + Tree Structure + Similarity	77.52	75.98	75.50	49.83

Table 5: Comparison of experimental results of different feature sets

accuracy of 77.52% and 75.50% in SI and NR tasks respectively. The Macro-F1 values reach 75.98% and 49.83% in SI and NR tasks respectively. In the following experiments of joint model, we use the best performance for comparison.

Constraint	Accuracy	Macro-F1
Local model	77.52	75.98
ILP(C1)	77.95	77.68
ILP(C1+C2)	78.51	77.81
ILP(C1+C2+C3)	78.54	77.68

Table 6: Performance of structure identification with joint model

Table 6 shows the performance of the ILP approach when different constraints are used. From this table, we can see that the constraints C1,C2 and C3 are capable of improving the performance of structure identification. When all these constraints are utilized, the inference performance reaches the best of 78.54% in accuracy and 77.68% in Macro-F1, 1.02% in accuracy and 1.70% in Macro-F1 better than the best performance of local model. This indicates the beneficiary of label correlations intra- and multi-instance to the task of structure identification.

It is worthwhile to note that our ILP approach could also benefit the task of nuclearity recognition when the constraints are employed. Table 7 shows our ILP approach improve the performance of nuclearity recognition by 0.51% in accuracy and 1.86% in Macro-F1.

	Accuracy	Macro-F1
Local model	75.50	49.83
ILP approach	76.01	51.69

Table 7: Performance of nuclearity recognition with joint model

There are some discoveries in our experiment: 1) The lexical features of connective, such as "therefore", "as a result" etc. are very useful on the micro level discourse analysis. But when used on macro level, the connective may confuse the model. That is because connectives usually occur between successive clauses or consecutive sentences, and are seldom used to express the relationship between paragraphs, especially in Chinese. 2) We have already tried some linguistic features, including lexical and syntactic, but the features are not outstanding in the experiment. There are several sentences in a paragraph, so syntactic information is not easy to use. We will explore other linguistic features and effective expression on macro level discourse structure analysis in the future.

8 Conclusion

In this paper, we present an efficient joint model of structure identification and nuclearity recognition on macro level discourse structure analysis. In particular, various kinds of feature sets are introduced to improve the performance of local models, and various constraints are introduced to improve the performance of joint model.

In future work, we will explore better joint modeling and effective linguistic features in discourse structure analysis. Furthermore, we wish to explore the other two tasks of macro discourse structure analysis, and build an End-to-End analysis platform.

Acknowledgements

We are grateful for the help of Jingjing Wang for his initial discussion. We thank our anonymous reviewers for their constructive comments, which helped to improve the paper. This work is supported by the National Natural Science Foundation of China (61773276, 61751206, 61673290) and Jiangsu Provincial Science and Technology Plan (No. BK20151222).

References

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current and new directions in discourse and dialogue*, pages 85–112. Springer.

Arman Cohan and Nazli Goharian. 2017. Scientific document summarization via citation contextualization and scientific discourse. *International Journal on Digital Libraries*, pages 1–17.

Vanessa Wei Feng and Graeme Hirst. 2012. Text-level discourse parsing with rich linguistic features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 60–68. Association for Computational Linguistics.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *ACL (1)*, pages 511–521.

Rafael Ferreira, Luciano de Souza Cabral, Frederico Freitas, Rafael Dueire Lins, Gabriel de França Silva, Steven J Simske, and Luciano Favaro. 2014. A multi-document summarization system based on statistics and linguistic treatment. *Expert Systems with Applications*, 41(13):5780–5787.

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, and Preslav Nakov. 2014. Using discourse structure improves machine translation evaluation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 687–698.

Hugo Hernault, Helmut Prendinger, David A. duVerle, and Mitsuru Ishizuka. 2010. Hilda: A discourse parser using support vector machine classification. *D&D*, 1(3):1–33.

Feng Jiang, Xiaomin Chu, Sheng Xu, Peifeng Li, and Qiaoming Zhu. 2018. A macro discourse primary and secondary relation recognition method. *Journal of Chinese Information Processing*, 1(32):72–79.

Shafiq Joty, Giuseppe Carenini, and Raymond T Ng. 2012. A novel discriminative framework for sentence-level discourse analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 904–915. Association for Computational Linguistics.

Shafiq R Joty, Giuseppe Carenini, Raymond T Ng, and Yashar Mehdad. 2013. Combining intra-and multi-sentential rhetorical parsing for document-level discourse analysis. In *ACL (1)*, pages 486–496.

Junhui Li, Guodong Zhou, and Hwee Tou Ng. 2010. Joint syntactic and semantic parsing of chinese. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1108–1117. Association for Computational Linguistics.

Peifeng Li, Qiaoming Zhu, and Guodong Zhou. 2013. Joint modeling of argument identification and role determination in chinese event extraction with discourse-level information. In *IJCAI*, pages 2120–2126.

Yancui Li, Wenhe Feng, Jing Sun, Fang Kong, and Guodong Zhou. 2014. Building chinese discourse corpus with connective-driven dependency tree structure. In *EMNLP*, pages 2105–2114. Citeseer.

Yancui Li. 2015. *Research of Chinese discourse structure representation and resource construction*. Ph.D. thesis, Suzhou: Soochow University.

William C Mann and Sandra A Thompson. 1987. Rhetorical structure theory: A theory of text organization (no. isi/rs-87-190). marina del rey. *CA: Information Sciences Institute*.

Daniel Marcu, Estibaliz Amorrortu, and Magdalena Romera. 1999. Experiments in constructing a corpus of discourse trees. *Towards Standards and Tools for Discourse Tagging*.

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

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jawad Sadek and Farid Meziane. 2016. A discourse-based approach for arabic question answering. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 16(2):11.

Caroline Sporleder and Alex Lascarides. 2004. Combining hierarchical clustering and machine learning to predict high-level discourse structure. In *Proceedings of the 20th international conference on Computational Linguistics*, page 43. Association for Computational Linguistics.

Teun Adrianus Van Dijk. 1980. *Macrostructures: An interdisciplinary study of global structures in discourse, interaction, and cognition*. Lawrence Erlbaum Associates.

Nianwen Xue, Fu-Dong Chiou, and Martha Palmer. 2002. Building a large-scale annotated chinese corpus. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–8. Association for Computational Linguistics.

Nianwen Xue, Xiuhong Zhang, Zixin Jiang, Martha Palmer, Fei Xia, Fu-Dong Chiou, and Meiyu Chang. 2013. Chinese treebank 8.0 ldc2013t21. *Linguistic Data Consortium, Philadelphia*.

Implicit Discourse Relation Recognition using Neural Tensor Network with Interactive Attention and Sparse Learning

Fengyu Guo[1,2,*], **Ruifang He**[1,2,*,†], **Di Jin**[1,2], **Jianwu Dang**[1,2,3], **Longbiao Wang**[1,2], **Xiangang Li**[4]
[1]School of Computer Science and Technology, Tianjin University, Tianjin, China.
[2]Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin, China.
[3]Japan Advanced Institute of Science and Technology, Ishikawa, Japan.
[4]AI Labs, Didi Chuxing, Beijing, China.
{fengyuguo,rfhe,jindi,longbiao_wang}@tju.edu.cn
jdang@jaist.ac.jp, lixiangang@didichuxing.com

Abstract

Implicit discourse relation recognition aims to understand and annotate the latent relations between two discourse arguments, such as temporal, comparison, etc. Most previous methods encode two discourse arguments separately, the ones considering pair specific clues ignore the bidirectional interactions between two arguments and the sparsity of pair patterns. In this paper, we propose a novel Neural **T**ensor Network framework with **I**nteractive **A**ttention and **S**parse **L**earning (TIASL) for implicit discourse relation recognition. (1) We mine the most correlated word pairs from two discourse arguments to model pair specific clues, and integrate them as interactive attention into argument representations produced by the bidirectional long short-term memory network. Meanwhile, (2) the neural tensor network with sparse constraint is proposed to explore the deeper and the more important pair patterns so as to fully recognize discourse relations. The experimental results on PDTB show that our proposed TIASL framework is effective.

1 Introduction

Discourse relation describes how two adjacent text units (e.g. clauses, sentences, and larger sentence groups), called arguments, named *Arg*1 and *Arg*2, are connected semantically, such as temporally, causally, etc. Yet implicit discourse relation recognition without explicit connectives (Pitler et al., 2008), which needs to infer the relation from specific context, is still a challenging problem. It can be used in text summarization(Gerani et al., 2014), conversation system (Higashinaka et al., 2014) and so on.

Previous researches mainly include (1) traditional feature-based models and (2) neural network based models. Most feature-based models adopt various linguistic features (such as polarity, word pairs, and position information, etc.) and design complicated rules to recognize implicit discourse relations (Pitler et al., 2009; Zhou et al., 2010; Braud and Denis, 2015). They can not fully use the local and the global context, and the human cost is huge. Neural network based models get the better argument representations and more precisely capture discourse relations (Braud and Denis, 2015; Zhang et al., 2015; Liu et al., 2016). However, they encode two discourse arguments separately, and ignore pair specific clues. The further researches adopt the different hybrid neural models (Chen et al., 2016; Lei et al., 2017) and attention mechanism (Cai and Zhao, 2017) to mine the semantic interactions of argument pairs. Yet, they ignore the bidirectional interactions between two arguments during the representation stage since there is asymmetry from the perspective of human-like reading strategy. And the sparsity of word pair patterns indicating discourse relation is neither considered.

Therefore, a novel neural **T**ensor network model with **I**nteractive **A**ttention and **S**parse **L**earning is proposed for implicit discourse relation recognition, namely TIASL. We imitate the human-like reading strategy, and model the relatedness between two discourse arguments as a kind of interactive attention from bidirectional aspects. It is added into the argument representations with a bidirectional Long Short-Term Memory network (Bi-LSTM), and then plugged in neural tensor network (NTN) with l_1 reg-

* Equal contribution.
† Corresponding author.

Proceedings of the 27th International Conference on Computational Linguistics, pages 547–558
Santa Fe, New Mexico, USA, August 20-26, 2018.

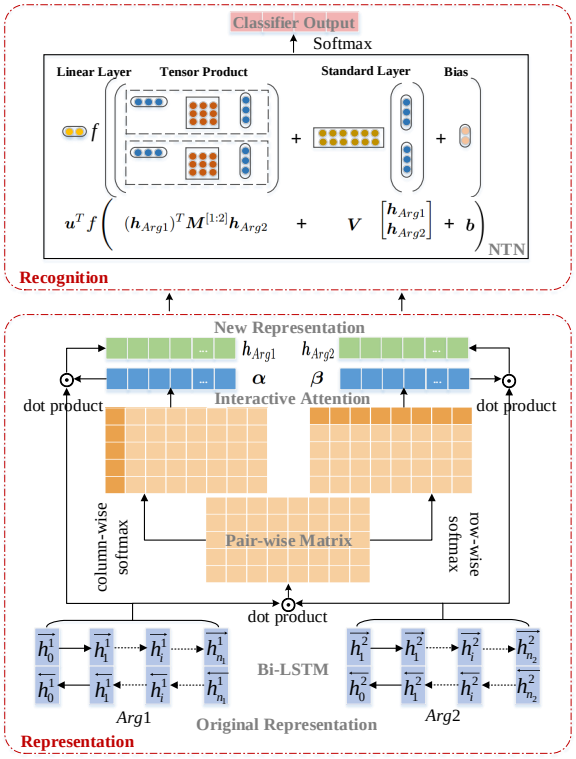


Figure 1: The TIASL framework.

ularization. This helps to mine the different aspects of semantic interactions between two arguments and select the important and the informative word pair patterns.

Our main contributions are as follows:

- Propose a novel TIASL framework from the perspectives of the human-like bidirectional reading strategy and the sparsity of word pair patterns;
- Encode the discourse arguments by the Bi-LSTM with interactive attention for implicit discourse relation recognition;
- Use neural tensor network with sparse constraint to capture the deeper and the more indicative pair patterns;
- Experimental results on PDTB show that our TIASL model is effective.

2 The Proposed Method

We formalize implicit discourse relation recognition as a classification problem. The proposed TIASL framework is shown in Figure 1. The main steps include (1) discourse argument representations with interactive attention based on Bi-LSTM and (2) sparse pair pattern selection and implicit discourse relation recognition.

2.1 Discourse Argument Representations with Interactive Attention

Attention mechanism has achieved great success in image recognition, which is based on the visual attention principle found in humans. Recently, it is widely adopted in many NLP tasks. Inspired by (Herzog et al., 2016), we imitate the human-like bidirectional reading strategy, and propose an interactive attention mechanism to enhance discourse argument representations.

For the original representations of discourse arguments shown in Figure 1, we first associate each word w in the vocabulary with a vector representation $\boldsymbol{x}_w \in \mathbb{R}^d$, where d is the dimension of the embeddings.

Since each argument is viewed as a sequence of word vectors, let $\boldsymbol{x}_i^1(\boldsymbol{x}_i^2)$ be the i-th word vector in *Arg*1 (*Arg*2), thus the arguments in a discourse relation are expressed as,

$$Arg1 : [\boldsymbol{x}_1^1, \boldsymbol{x}_2^1, ..., \boldsymbol{x}_{n_1}^1], \qquad Arg2 : [\boldsymbol{x}_1^2, \boldsymbol{x}_2^2, ..., \boldsymbol{x}_{n_2}^2].$$

where *Arg*1 (*Arg*2) has n_1 (n_2) words.

2.1.1 The Basic Bi-LSTM

Long Short-Term Memory network (LSTM) (Hochreiter and Schmidhuber, 1997) is a variant of recurrent neural network. Considering that it can model long-term dependencies and encode context information, we use it in the basic argument representation. Given the word representations of two arguments as we just described, the LSTM computes the state sequence for each position t using the following equations:

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i[\boldsymbol{x}_t, \boldsymbol{h}_{t-1}] + \boldsymbol{b}_i), \tag{1}$$
$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f[\boldsymbol{x}_t, \boldsymbol{h}_{t-1}] + \boldsymbol{b}_f), \tag{2}$$
$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o[\boldsymbol{x}_t, \boldsymbol{h}_{t-1}] + \boldsymbol{b}_o), \tag{3}$$
$$\tilde{\boldsymbol{c}}_t = \tanh(\boldsymbol{W}_c[\boldsymbol{x}_t, \boldsymbol{h}_{t-1}] + \boldsymbol{b}_c), \tag{4}$$
$$\boldsymbol{c}_t = \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t + \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1}, \tag{5}$$
$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t). \tag{6}$$

where $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, $\boldsymbol{o}_t$, $\boldsymbol{c}_t$, $\boldsymbol{h}_t$ denote the input gate, forget gate, output gate, memory cell and hidden state at position t respectively. $\boldsymbol{W}_i$, $\boldsymbol{W}_f$, $\boldsymbol{W}_o$, $\boldsymbol{W}_c$, $\boldsymbol{b}_i$, $\boldsymbol{b}_f$, $\boldsymbol{b}_o$, $\boldsymbol{b}_c$ are the neural network parameters. [ ] means the concatenation operation of $\boldsymbol{x}_t$, $\boldsymbol{h}_{t-1}$. σ denotes the logistic sigmoid function and $\odot$ denotes the element-wise multiplication.

Since LSTM only considers the context from the previous, we utilize a bidirectional LSTM (Bi-LSTM) preserving both history and future information. Therefore, at each position t of the sequence, we can obtain two representations $\overrightarrow{\boldsymbol{h}_t}$ and $\overleftarrow{\boldsymbol{h}_t}$. Then we concatenate them to get the intermediate state $\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}_t}, \overleftarrow{\boldsymbol{h}_t}]$. For *Arg*1 and *Arg*2, we encode them into the contextual representations by Bi-LSTM. That is, $\boldsymbol{h}_i^1 = [\overrightarrow{\boldsymbol{h}_i^1}, \overleftarrow{\boldsymbol{h}_i^1}]$ and $\boldsymbol{h}_j^2 = [\overrightarrow{\boldsymbol{h}_j^2}, \overleftarrow{\boldsymbol{h}_j^2}]$ are the intermediate states of i-th word in *Arg*1 and j-th word in *Arg*2 respectively, where $\overrightarrow{\boldsymbol{h}_i^1}, \overleftarrow{\boldsymbol{h}_i^1}, \overrightarrow{\boldsymbol{h}_j^2}, \overleftarrow{\boldsymbol{h}_j^2} \in \mathbb{R}^d$.

Separately encoding arguments with Bi-LSTM could not reflect the semantic between two discourse arguments in a discourse relation. In order to fully use their semantic connections, we explore a novel argument representation.

2.1.2 Model the Asymmetry of Reciprocal Attention on Discourse Arguments

Herzog et al. (2016) proposed the two stage model of visual perception which indicated that people's image recognition includes two stages: collecting information and understanding information. In daily life, we have a similar feeling intuitively during reading: a more reasonable strategy is that people may read two discourse arguments back and forth, and find some relevant and informative clues helpful to judge the discourse relation. Due to the different reading order of two arguments, people may get the different focused information and thus have the different decisions. Therefore, we model the reciprocal attention on discourse arguments from two directions.

Firstly, we calculate semantic connections between word pairs in two arguments as a pair-wise matrix shown in Eq.(7), which indicates the relevant score of i-th *Arg*1 word and j-th *Arg*2 word by dot product of their hidden representations.

$$\boldsymbol{S}(i, j) = (\boldsymbol{h}_i^1)^T \cdot \boldsymbol{h}_j^2. \tag{7}$$

where $\boldsymbol{S} \in \mathbb{R}^{n_1 \times n_2}$, n_1 and n_2 are the lengths of *Arg*1 and *Arg*2, respectively.

Secondly, as the concerns of the forward and the reverse reading order are asymmetric when judging the relation of two discourse arguments. For each word in *Arg*2, we apply a column-wise softmax function on the pair-wise matrix $\boldsymbol{S}$ to get a probability distribution $\boldsymbol{\alpha}_t$ over *Arg*1, shown in Eq.(8). Similarly, we conduct a row-wise softmax function to get $\boldsymbol{\beta}_t$ over *Arg*2 when considering one *Arg*1

word, shown in Eq.(9). We denote $\boldsymbol{\alpha}_t \in \mathbb{R}^{n_1}$ as *Arg*2-to-*Arg*1 attention, and $\boldsymbol{\beta}_t \in \mathbb{R}^{n_2}$ as *Arg*1-to-*Arg*2 attention at position t, which are named **interactive attention**.

$$\boldsymbol{\alpha}_t = softmax(\boldsymbol{S}(1,t), ..., \boldsymbol{S}(n_1,t)), \tag{8}$$

$$\boldsymbol{\beta}_t = softmax(\boldsymbol{S}(t,1), ..., \boldsymbol{S}(t,n_2)). \tag{9}$$

where $\boldsymbol{\alpha}_t = [\boldsymbol{\alpha}_t^1, \boldsymbol{\alpha}_t^2, ..., \boldsymbol{\alpha}_t^{n_1}]$, $\boldsymbol{\alpha}_t^i$ means the attention value of i-th word in *Arg*1 at position t. Likewise, $\boldsymbol{\beta}_t = [\boldsymbol{\beta}_t^1, \boldsymbol{\beta}_t^2, ..., \boldsymbol{\beta}_t^{n_2}]$, $\boldsymbol{\beta}_t^j$ is the attention value of j-th word in *Arg*2 at position t.

In order to exploit the overall influence information to represent semantic connection of two discourse arguments, we average all the $\boldsymbol{\alpha}_t$, $\boldsymbol{\beta}_t$ to get the final attention of *Arg*1 and *Arg*2.

$$\boldsymbol{\alpha} = \frac{1}{n_2}\sum_{t=1}^{n_2}\boldsymbol{\alpha}_t, \qquad \boldsymbol{\beta} = \frac{1}{n_1}\sum_{t=1}^{n_1}\boldsymbol{\beta}_t. \tag{10}$$

The new argument representations integrating argument context and interactive attention are shown as Eq.(11), which reflect the human-like bidirectional reading strategy to some extent.

$$\boldsymbol{h}_{Arg1} = \boldsymbol{h}^1\boldsymbol{\alpha}, \qquad \boldsymbol{h}_{Arg2} = \boldsymbol{h}^2\boldsymbol{\beta}. \tag{11}$$

2.2 Sparse Pair Pattern Selection and Discourse Relation Recognition

Observations show that there are some pair patterns in a discourse relation. Once we detect these interactions expressing pair patterns, discriminating discourse relation is obvious. However, how to represent and select this kind of interaction is a problem.

2.2.1 Neural Tensor Network

Conventional methods to measure the relevance between two arguments includes bilinear model (Jenatton et al., 2012), and single layer neural networks (Collobert and Weston, 2008), etc. These methods could hardly model the complex and informative interactions. Success in knowledge graph (Socher et al., 2013a) shows that tensor can model multiple interactions in data. Therefore, we further employ a tensor layer to mine the deeper semantic interactions based on the new argument representations so as to recognize implicit discourse relations.

Tensor is a geometric object that describes relations between vectors, scalars, and others. It can be represented as a multi-dimensional array of numerical values. Following the NTN (Socher et al., 2013a; Pei et al., 2014), we utilize a 3-way tensor $\boldsymbol{M}^{[1:k]} \in \mathbb{R}^{d_h \times d_h \times k}$ to model the interactions shown in Eq.(12).

$$g(\boldsymbol{h}_{Arg1}, \boldsymbol{h}_{Arg2}) = \boldsymbol{u}^T f\left((\boldsymbol{h}_{Arg1})^T \boldsymbol{M}^{[1:k]} \boldsymbol{h}_{Arg2} + \boldsymbol{V}\begin{bmatrix}\boldsymbol{h}_{Arg1}\\ \boldsymbol{h}_{Arg2}\end{bmatrix} + \boldsymbol{b}\right). \tag{12}$$

where f is a standard nonlinearity applied element-wise, $\boldsymbol{M}^{[1:k]} \in \mathbb{R}^{d_h \times d_h \times k}$ is a tensor and the bilinear tensor product $(\boldsymbol{h}_{Arg1})^T \boldsymbol{M}^{[1:k]} \boldsymbol{h}_{Arg2}$ results in a vector $\boldsymbol{m} \in \mathbb{R}^k$, where each entry is computed by one slice $i = 1, 2, ..., k$ of the tensor: $\boldsymbol{m}_i = (\boldsymbol{h}_{Arg1})^T \boldsymbol{M}^{[i]} \boldsymbol{h}_{Arg2}$. The other parameters $\boldsymbol{V} \in \mathbb{R}^{k \times 2d_h}$, $\boldsymbol{u} \in \mathbb{R}^k$, $\boldsymbol{b} \in \mathbb{R}^k$ are the standard form of a neural network. Here, each tensor slice can be seen as a "feature extractor", which extracts the features expressing the *Arg*1-*Arg*2 interactions.

Through the tensor layer, we can obtain the semantic interactions between two arguments as features, which are further reshaped to a vector and fed to a full connection hidden layer. Then we apply a softmax function in the output layer to compute the probabilities of different relations and recognize them.

2.2.2 Model Training with Sparse Constraint

Given a training corpus which contains n instances $\{(\boldsymbol{x}, \boldsymbol{y})\}_{r=1}^n$, $(\boldsymbol{x}, \boldsymbol{y})$ denotes an argument pair and its label. We employ the cross-entropy error to assess how well the predicted relation represents the real relation, defined as:

$$L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_{j=1}^{C} \boldsymbol{y}_j \log(Pr(\hat{\boldsymbol{y}}_j)). \tag{13}$$

where $Pr(\hat{\boldsymbol{y}}_j)$ is the predicted probabilities of labels, C is the class number.

Based on argument representations with interactive attention, tensor embodies the different aspects of semantic interactions between two arguments. However, not all the interactions are useful. There could exist some redundant and noisy interactions influencing the system performance. In order to remove the irrelevant interactions and select the indicative pair patterns, the large portion of $\boldsymbol{M}^{[i]}$ should be zero. Therefore, we introduce the 1-norm regularizer to promote the feature sparsity. This element-wise sparsity can be helpful when most of the features are irrelevant to the learning objective. Furthermore, we also add l_2 regularization to avoid over-fitting issue. And the training objective function is transformed as:

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{r=1}^{n} L(\hat{\boldsymbol{y}}^{(r)}, \boldsymbol{y}^{(r)}) + R(\boldsymbol{\theta}), \tag{14}$$

$$R(\boldsymbol{\theta}) = \lambda_M \|\boldsymbol{\theta}_M\|_1 + \frac{\lambda_O}{2}\|\boldsymbol{\theta}_O\|_2. \tag{15}$$

where $R(\boldsymbol{\theta})$ is the regularization term with respect to $\boldsymbol{\theta}$. We divide $\boldsymbol{\theta}$ into two parts: $\boldsymbol{\theta}_M$ is the tensor term weights, and $\boldsymbol{\theta}_O$ is the other parameters of our model. Especially, $\|\boldsymbol{\theta}_M\|_1$ in Eq.(15) is l_1 regularization for the tensor slices, which is used to filter the important values.

To minimize the objective, we employ the proximal gradient descent method (Parikh and Boyd, 2014) since l_1 regularization is non-differentiable at zero. It is used for optimizing the objective as a combination of both smooth and non-smooth terms. The update formulas is as follows:

$$\boldsymbol{\theta}_i^{(t')} = \boldsymbol{\theta}_i^{(t)} - \gamma(\frac{\partial L}{\partial \boldsymbol{\theta}_i} + \lambda\frac{\partial R}{\partial \boldsymbol{\theta}_i})|_{\boldsymbol{\theta}_i=\boldsymbol{\theta}_i^{(t)}}, \tag{16}$$

$$\frac{\partial R}{\partial \boldsymbol{\theta}_i} = \begin{cases} 2\boldsymbol{\theta}_i^{(t)}, & \text{if } \|\boldsymbol{\theta}\|_2; \\ sign(\boldsymbol{\theta}_i^{(t)}), & \text{if } \|\boldsymbol{\theta}\|_1, \text{ and } \boldsymbol{\theta}_i^{(t)} \neq 0. \end{cases} \tag{17}$$

$$\boldsymbol{\theta}_i^{(t+1)} = prox_\lambda(\boldsymbol{\theta}_i^{t'}) = \tau(\boldsymbol{\theta}_i^{t'}, \gamma\lambda), \tag{18}$$

$$\tau(a, z) = \begin{cases} a - z, & \text{if } a > z; \\ a + z, & \text{if } a < -z; \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

where $prox_\lambda$ is a proximal operator, τ is a soft-thresholding operator, and γ is the learning rate.

3 Experiments

3.1 Data Preparation

Corpus. We use the Penn Discourse TreeBank (PDTB) (Prasad et al., 2008), which is the largest hand-annotated discourse relation corpus annotated on 2312 Wall Street Journal (WSJ) articles. Experiments are conducted on the four top-level classes as in previous work (Rutherford and Xue, 2014; Chen et al., 2016). Following the conventional data splitting, we use Section 2-20 as training set, Section 21-22 as testing set, and Section 0-1 as development set. The relevant statistics is shown in Table 1.

Relation	Train	Dev.	Test
Comparison	1842	393	144
Contingency	3139	610	266
Expansion	6658	1231	537
Temporal	579	83	55

Table 1: Statistics of implicit discourse relations.

Hyper-parameters	Value
Initial learning rate	0.01
Minibatch size	30
Dropout rate	0.1
Number of tensor slice	3

Table 2: Hyper-parameters for our TIASL model.

Experimental Settings. The 50-dimensional pre-trained word embeddings are provided by GloVe (Pennington et al., 2014), which are fixed during our model training. All the discourse arguments are padded

to the same length of 50. And the length of intermediate representation for our network is also 50. The other parameters are initialized by random sampling from uniform distribution in [-0.1,0.1]. We do not present the details of tuning the hyper-parameters and only give their final settings as shown in Table 2.

To evaluate our model, we adopt two kinds of experiment settings, including a four-way classification and four separate one-vs-other binary classification. The former is to observe the overall performance. And the latter is to solve the problem of unbalance data, where each top level class is against the other three discourse relation classes. We use an equal number of positive and negative instances in the training set in each class. The testing set and development set keep the natural state.

3.2 Comparison Methods

We choose the following models as our baselines, which are the state-of-the-art models in argument representation, interaction and attention aspects during implicit discourse relation recognition.

- **Ji2015**: Ji and Eisenstein (2014) utilized two recursive neural networks on the syntactic parse tree to induce argument representation and entity spans.
- **Qin2016**: Qin et al. (2016b) integrated a CNN and a Collaborative Gated Neural Network (CGNN) into argument representation.
- **Chen2016**: Chen et al. (2016) used a Gated Relevance Network (GRN) and incorporated both the linear and non-linear interactions between word pairs.
- **Liu2016**: Liu and Li (2016) designed Neural Networks with Multi-level Attention (NNMA) and selected the important words for recognizing discourse relation. Here, we select the models with two and three levels attention as baselines.

Besides, we also use the following variants of RNN and the proposed TIASL model for comparisons.

- **LSTM**: encode two discourse arguments by LSTM respectively, and concatenate the two representations, feeding them to the full connection hidden layer as the input of softmax classifier.
- **Bi-LSTM**: based on **LSTM**, we consider the bidirectional context information, and use Bi-LSTM to encode two discourse arguments.
- **Bi-LSTM+Interactive Attention**: further integrate the interactive attention to obtain the new argument representations shown in Eq.(11).
- **Bi-LSTM+Tensor Layer**: based on **Bi-LSTM**, adopt the neural tensor network to capture the semantic interaction between two arguments.
- **TIA with k-Max Pooling**: use k-max pooling operation instead of our sparse strategy to select features after tensor layer in neural Tensor network with Interactive Attention model (TIA).
- **Our TIASL**: based on **Bi-LSTM** with **Interactive Attention** and **Tensor Layer**, we add l_1 regularization for tensor parameters in order to capture the most important interactions.

3.3 The Overall Performance

Table 3 shows the overall performance, using F_1 score and accuracy for four-wary classification and F_1 score for binary classification. With respect to four-way classification, we have the following observations:

- Ji2015 gains the lowest performance on both F_1 score and accuracy, which separately computes discourse argument representations by integrating syntactic parse tree into RNN. It indicates a simple neural network, which ignores the interactive context of two discourse arguments, is not sufficient for implicit discourse relation recognition.

Model	Binary Classification				Four-way Classification	
	Comp.	Cont.	Expa.	Temp.	F1	Acc.
Ji2015	35.93%	52.78%	-	27.63%	38.52%	43.56%
Qin2016	41.55%	57.32%	71.50%	35.43%	-	-
Chen2016	40.17%	54.76%	-	31.32%	44.61%	57.84%
Liu2016 (two levels)	36.70%	54.48%	70.43%	38.84%	46.29%	57.17%
Liu2016 (three levels)	39.86%	53.69%	69.71%	37.61%	44.95%	57.57%
Our TIASL	40.35%	56.81%	72.11%	38.65%	47.59%	59.06%

Table 3: Comparisons with the state-of-the-art models.

- The accuracy of Chen2016 model is better than that of other baselines. It verifies the effectiveness word pair information, which uses gate mechanism to control the combination of linear and non-linear interactions between argument pairs. However, there unavoidably exits some noises, and this model has not considered the sparsity of pair patterns. F_1 score of Liu2016 (two levels) model is higher than that of other baselines, which achieves 1.34% than that of three levels attention's. It indicates that attention mechanism is useful, and yet paying more attention may bring the over-fitting problem due to more parameters.

- Our TIASL gains an improvement 1.30% on F_1 score than that of Liu2016 (two levels), and an improvement 1.22% on accuracy than that of Chen2016. The results imply that our model with interactive attention for the bidirectional asymmetry of two arguments and sparse pair pattern selection is useful for recognizing implicit discourse relation.

For binary classification, the observations are as follows:

- F_1 scores of Temporal relation are the lowest in all models. This is reasonable since it accounts for the smallest number of instances (only 5%) in the corpus. With the increase of instance number in different relations, F_1 scores also rise. It proves that the corpus is also crucial to implicit discourse relation recognition.

- Qin2016 gains the best performance on Contingency, and our TIASL model obtains the comparable score with it. Notably, Chen2016 and Liu2016 (two levels) are quite relevant work to ours. Different from them, our model integrates the attention-based interactive information between arguments at representation stage. This may be the main reason why our TIASL model is better than the two models (the improvements of 2.05% and 2.33%, respectively). Similar results are in Comparison relation.

- Our TIASL model achieves state-of-the-art performance in recognition of the Expansion relation. The reasons are two-fold: (1) some argument pairs may have confusable word pairs, which can be effectively mined by asymmetric attention; (2) some complex argument pairs need to be further understood their semantic representation and explore the indicative and interactive patterns. Our TIASL model integrates these two aspects and performs well.

3.4 The Effectiveness of Each Component

In order to verify the effectiveness of attention mechanism, neural tensor layer and l_1 regularization, we design five experiments to compare with our TIASL model. Seen from Table 4, we have the following observations:

- The performance of LSTM is the worst on each relation. Although Bi-LSTM captures more information than LSTM, the results are not very good. The reason is that separately encoding discourse argument by LSTM or Bi-LSTM ignores the local focused words since it equally treats every word.

Model	Binary Classification				Four-way Classification	
	Comp.	Cont.	Expa.	Temp.	F1	Acc.
LSTM	32.95%	43.38%	68.10%	30.80%	36.40%	54.50%
Bi-LSTM	34.01%	44.68%	68.53%	31.27%	36.54%	55.31%
Bi-LSTM + Interactive Attention	35.43%	45.92%	68.57%	32.50%	43.27%	55.68%
Bi-LSTM + Tensor Layer	37.36%	46.73%	69.81%	33.89%	43.61%	55.97%
TIA with k-max pooling	39.78%	55.13%	70.96%	37.65%	46.77%	57.83%
Our TIASL	40.35%	56.81%	72.11%	38.65%	47.59%	59.06%

Table 4: The effects of different components.

- Bi-LSTM with Interactive Attention performs better than the above two simple models. In detail, the F_1 score of this model gains 2.48%, 2.54%, 1.70% improvement on Comparison, Contingency and Temporal than that of LSTM, respectively. We perform significance test for these improvements, and they are both significant under one-tailed t-test ($p < 0.05$). It indicates that the model could find pair specific clues in two arguments by constructing the relevance of word pairs to some extent. And the effectiveness of our attention mechanism for capturing the interactive information between the arguments is crucial at representation stage.
- Bi-LSTM with Tensor Layer slightly achieves better performance. This indicates the effectiveness of tensor layer for capturing complex interactive features. The TIA model, which combines attention mechanism and tensor layer, also performs better, but lower than our TIASL model. It is because that k-max pooling strategy could not guarantee getting the important interaction pairs from the global perspective. How to represent and explain these interactive features mined in our model will be our next research focus.
- Our TIASL model achieves the best performance. It not only encodes discourse arguments with important word pairs by interactive attention, but also captures the more deeper and the more important semantic interactions by NTN with l_1 regularization. The integration of all components is useful for recognizing implicit discourse relations.

The observations of each component's four-way classification are consistent with the binary classification.

3.5 Interactive Attention Analysis

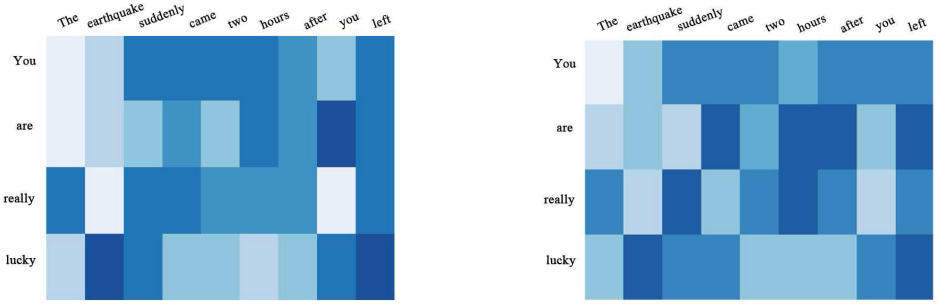
To demonstrate the validity of our interactive attention, we visualize the heat maps of argument pairs shown in Figure 2, which shows the interaction matrices of only using Bi-LSTM and our interactive attention on an example. Every word accompanies with the various background colors. The darker patches denote the correlations of word pairs are higher. The example of *Contingency* relation is listed below:

***Arg*1:** *You are really lucky.*

***Arg*2:** *The earthquake suddenly came two hours after you left.*

However, it might be classified as a *Comparison* relation if we only focus on the informative word pair (lucky, earthquake) with contrasting sentiment polarity. Therefore, we need to consider the context of the whole argument pair to infer the correct relation from two back and forth reading directions.

Seen from Figure 2(a), the word pairs (are, you), (lucky, earthquake), (lucky, left) get the higher scores, the scores on the other pairs are arbitrary. It demonstrates the Bi-LSTM model may be influenced by the word pair frequency in corpus. Meanwhile, it encodes two arguments separately, which ignores the relevant and informative interactions between two arguments. Figure 2(b) as a comparison, we observe that there are the more word pairs obtaining the higher scores, which are ignored in Figure 2(a). This proves that the effectiveness of generating interactive argument representation by our interactive attention, which imitates human-like reading strategy.



(a) Bi-LSTM (only word embedding) (b) Bi-LSTM + Interactive attention

Figure 2: The relatedness between two arguments.

4 Related Work

Traditional methods for implicit discourse relation recognition rely on artificial and shallow features, such as POS, polarity, word position, etc. Recent neural network based methods acquire the better performance, and mainly focus on two aspects:

4.1 Argument Representation

The prerequisite of recognizing discourse relation is to have a good argument representation. Most previous researches use various neural networks, such as CNN, RNN, and hybrid models (Zhang et al., 2015; Qin et al., 2016a; Rutherford et al., 2016) to encode discourse arguments as low-dimensional, dense and continuous representations. Ji and Eisenstein (2014) integrate the linguistic features, including syntactic parsing and coreferent entity mentions into compositional distributed representations.

Though argument representation contains the high-level semantic, it does not embody emphasis during reading comprehension. Some used neural architectures with attention mechanism pick up the important information from discourse arguments (Mnih et al., 2014; Zhang et al., 2016). Li et al. (2016) exploit the hierarchical attention to capture the focus of different granularity. Liu and Li (2016) imitate the repeated reading strategy, and proposes neural networks with multi-level attention to recognize discourse relations. However, these researches have not considered the human-like reading strategy from two directions. The imagination by first reading one argument is different from the other, which has the reciprocal effects on implicit discourse relation recognition.

4.2 Pair Interactions

The emphasis of discourse arguments is partly obtained by attention mechanism. Most studies tend to discover more semantic interactions between two arguments by complex neural networks (Chen et al., 2016; Qin et al., 2016b; Lan et al., 2017). Cai and Zhao (2017) generate discourse argument representations via pair-specified feature extraction. Lei et al. (2017) conduct word interaction score to capture both linear and quadratic relation for argument representation.

Neural tensor network is good at capturing multiple interactions in data, and gets the good performance on entity relation (Socher et al., 2013a), Chinese word segmentation(Pei et al., 2014) and sentiment analysis (Socher et al., 2013b) tasks. And some NTN-like methods learn the semantic interaction between discourse arguments (Chen et al., 2016). Yet they do not discriminate the noises and the redundant information existed in interactions, and ignore the sparsity of pair patterns.

Inspired by sparse learning in deep neural networks (Collins and Kohli, 2014; Yoon and Hwang, 2017; Wen et al., 2017), they use sparse regularization to obtain compact deep networks by removing unnecessary weighs. In our paper, we introduce sparse learning into neural tensor network to select some indicative and informative word pair patterns. To our knowledge, our study is the first to employ the idea of sparse learning in implicit discourse relation recognition.

5 Conclusion

A novel neural tensor network framework with interactive attention and sparse learning (TIASL) is proposed for implicit discourse relation recognition. We imitate human-like bidirectional reading strategy, and encode the semantic representation with reciprocal influence of discourse arguments through interactive attention. And we further adopt neural tensor network with l_1 regularization to capture the indicative and informative interactions between discourse arguments. Our experimental results on PDTB show that the proposed TIASL model is effective.

However, we just take the surface word pairs to express the correlation by calculating the pair-wise matrix in this paper. We will automatically mine the deeper interaction between two arguments and explain the specific patterns of the different relations.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (61472277, 61772361, 61771333). We also thank the anonymous reviewers for their valuable comments.

References

Chloé Braud and Pascal Denis. 2015. Comparing word representations for implicit discourse relation classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2201–2211.

Deng Cai and Hai Zhao. 2017. Pair-aware neural sentence modeling for implicit discourse relation classification. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 458–466. Springer.

Jifan Chen, Qi Zhang, Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Implicit discourse relation detection via a deep architecture with gated relevance network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1726–1735.

Maxwell D Collins and Pushmeet Kohli. 2014. Memory bounded deep convolutional networks. *arXiv preprint arXiv:1412.1442*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 160–167. ACM.

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T. Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1602–1613.

Herzog, Kammer Michael H., and Scharnowski Frank Thomas. 2016. Time slices: What is the duration of a percept? *PLOS Biology*.

Ryuichiro Higashinaka, Kenji Imamura, Toyomi Meguro, Chiaki Miyazaki, Nozomi Kobayashi, Hiroaki Sugiyama, Toru Hirano, Toshiro Makino, and Yoshihiro Matsuo. 2014. Towards an open-domain conversational system fully based on natural language processing. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, pages 928–939.

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

Rodolphe Jenatton, Nicolas L Roux, Antoine Bordes, and Guillaume R Obozinski. 2012. A latent factor model for highly multi-relational data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3167–3175.

Yangfeng Ji and Jacob Eisenstein. 2014. One vector is not enough: Entity-augmented distributional semantics for discourse relations. *Transactions of the Association for Computational Linguistics*, 3:329–344.

Man Lan, Jianxiang Wang, Yuanbin Wu, Zheng-Yu Niu, and Haifeng Wang. 2017. Multi-task attention-based neural networks for implicit discourse relationship representation and identification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1299–1308.

Wenqiang Lei, Xuancong Wang, Meichun Liu, Ilija Ilievski, Xiangnan He, and Min-Yen Kan. 2017. Swim: A simple word interaction model for implicit discourse relation recognition. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4026–4032.

Qi Li, Tianshi Li, and Baobao Chang. 2016. Discourse parsing with attention-based hierarchical neural networks. In *Proceedings of the 2016 Conference on Emirical Methods in Natural Language Processing (EMNLP)*, pages 362–371.

Yang Liu and Sujian Li. 2016. Recognizing implicit discourse relations via repeated reading: Neural networks with multi-level attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1224–1233.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, pages 2750–2756.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2204–2212.

Neal Parikh and Stephen Boyd. 2014. Proximal algorithms. *Foundations and Trends in Optimization.*, 1(3):127–239.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for chinese word segmentation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 293–303.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind K Joshi. 2008. Easily identifiable discourse relations. *Technical Reports (CIS)*.

Emily Pitler, Annie Louis, and Ani and Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 683–691.

Rashmi Prasad, Nikhil Diesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The penn discourse treebank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC)*.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016a. Shallow discourse parsing using convolutional neural network. In *CoNLL Shared Task*, pages 70–77.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016b. A stacking gated neural architecture for implicit discourse relation classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2263–2270.

Attapol Rutherford and Nianwen Xue. 2014. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 645–654.

Attapol T Rutherford, Vera Demberg, and Nianwen Xue. 2016. Neural network models for implicit discourse relation classification in english and chinese without surface features. *arXiv preprint arXiv:1606.01990*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013a. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642.

Wei Wen, Yuxiong He, Samyam Rajbhandari, Wenhan Wang, Fang Liu, Bin Hu, Yiran Chen, and Hai Li. 2017. Learning intrinsic sparse structures within long short-term memory. *arXiv preprint arXiv:1709.05027*.

Jaehong Yoon and Sung Ju Hwang. 2017. Combined group and exclusive sparsity for deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning (PMLR)*, pages 3958–3966.

Biao Zhang, Jinsong Su, Deyi Xiong, Yaojie Lu, Hong Duan, and Junfeng Yao. 2015. Shallow convolutional neural network for implicit discourse relation recognition. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2230–2235.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2016. Neural discourse relation recognition with semantic memory. *arXiv preprint arXiv:1603.03873*.

Zhi-Min Zhou, Yu Xu, Zheng-Yu Niu, Man Lan, Jian Su, and Chew Lim Tan. 2010. Predicting discourse connectives for implicit discourse relation recognition. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1507–1514. Association for Computational Linguistics.

Transition-based Neural RST Parsing with Implicit Syntax Features

Nan Yu, **Meishan Zhang** and **Guohong Fu***
School of Computer Science and Technology, Heilongjiang University, China
yunan.hlju@gmail.com,
mason.zms@gmail.com,
ghfu@hotmail.com

Abstract

Syntax has been a useful source of information for statistical RST discourse parsing. Under the neural setting, a common approach integrates syntax by a recursive neural network (RNN), requiring discrete output trees produced by a supervised syntax parser. In this paper, we propose an implicit syntax feature extraction approach, using hidden-layer vectors extracted from a neural syntax parser. In addition, we propose a simple transition-based model as the baseline, further enhancing it with dynamic oracle. Experiments on the standard dataset show that our baseline model with dynamic oracle is highly competitive. When implicit syntax features are integrated, we are able to obtain further improvements, better than using explicit Tree-RNN.

1 Introduction

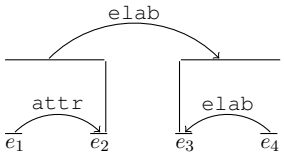
Discourse parsing is an important task in natural language processing (NLP), which aims to identify the relations between text units in documents. It has received great attention (Sagae, 2009; Hernault et al., 2010; Feng and Hirst, 2012; Joty et al., 2013; Feng and Hirst, 2014; Ji and Eisenstein, 2014; Heilman and Sagae, 2015; Li et al., 2016; Wang et al., 2017; Braud et al., 2017; Liu and Lapata, 2017), especially by performing the task based on rhetorical structure theory (RST) (Mann and Thompson, 1988). The RST-based parsing represents a document by a hierarchical tree, where leaf nodes are basic text units referred as elementary discourse unit (EDU), and non-terminal nodes define the discourse relations between adjacent tree nodes. Figure 1 shows an example, where each discourse relation has two parts, specifying its label and indicating its nuclearity, respectively.

Early studies adopt traditional statistical models for this task, using sophisticated manually-designed discrete features (Soricut and Marcu, 2003; Sagae, 2009; Hernault et al., 2010; Feng and Hirst, 2012; Joty et al., 2013; Feng and Hirst, 2014; Heilman and Sagae, 2015; Wang et al., 2017). Recently, inspired by the success of neural network models in NLP (Collobert et al., 2011; Devlin et al., 2014), several neural models for RST discourse parsing have been proposed as well (Li et al., 2014; Li et al., 2015b; Li et al., 2016; Braud et al., 2016; Braud et al., 2017; Liu and Lapata, 2017). Compared with statistical models, neural models exploit low-dimensional dense features, being able to avoid the feature sparsity problem, and on the other hand well-designed neural network structures such as long short term memory (LSTM) (Schmidhuber and Hochreiter, 1997) are capable of capturing high-order compositional features as well as global features automatically. In addition, we can use pre-trained neural word embeddings (Mikolov et al., 2013) on large scale corpus for neural network initialization. These characteristics show that neural models are promising for RST discourse parsing.

Intuitively, syntax is a potential avenue for the task, as it offers long-distance syntax relations between sentential words as well as key components of sentences. Syntax features have been demonstrated helpful in statistical models with human designed features (Soricut and Marcu, 2003; Sagae, 2009; Hernault et al., 2010; Feng and Hirst, 2012; Joty et al., 2013; Feng and Hirst, 2014; Heilman and Sagae, 2015). For neural network models, recursive neural network is a natural choice to represent tree-structural syntax trees globally. Only Li et al. (2015b) apply such a neural structure to incorporate syntax trees for

*Corresponding author.

Proceedings of the 27th International Conference on Computational Linguistics, pages 559–570
Santa Fe, New Mexico, USA, August 20-26, 2018.



e_1: American Telephone & Telegraph Co. said it
e_2: will lay off 75 to 85 technicians here , effective Nov. 1.
e_3: The workers install , maintain and repair its private branch exchanges,
e_4: which are large intracompany telephone networks.

Figure 1: An example of RST discourse tree, where $\{e_1, e_2, e_3, e_4\}$ are EDUs, `attr` and `elab` are discourse relation labels, and arrows indicate the nuclearities of discourse relations.

RST discourse parsing. Other studies still adopt discrete syntax features proposed by statistical models, feeding them into neural network models (Braud et al., 2016; Braud et al., 2017).

The above approaches model syntax trees in an explicit way, requiring discrete syntax parsing outputs as inputs for RST parsing. These approaches may suffer from the error propagation problem. Syntax trees produced by a supervised syntax parsing model could have errors, which may propagate into discourse parsing models. The problem could be extremely serious when inputs of discourse parsing have different distributions with the training data of the supervised syntax parser. Recently, Zhang et al. (2017) suggest an alternative method, which extracts syntax features from a Bi-Affine dependency parser (Dozat and Manning, 2016), and the method gives competitive performances on relation extraction. It actually represents syntax trees implicitly, thus it can reduce the error propagation problem.

In this work, we investigate the implicit syntax feature extraction approach for RST parsing. In addition, we propose a transition-based neural model for this task, which is able to incorporate various features flexibly. We exploit hierarchical bi-directional LSTMs (Bi-LSTMs) to encode texts, and further enhance the transition-based model with dynamic oracle. Based on the proposed model, we study the effectiveness of our proposed implicit syntax features. We conduct experiments on a standard RST discourse TreeBank (Carlson et al., 2003). First, we evaluate the performance of our proposed transition-based baseline, finding that the model is able to achieve strong performances after applying dynamic oracle. Then we evaluate the effectiveness of implicit syntax features extracted from a Bi-Affine dependency parser. Results show that the implicit syntax features are effective, giving better performances than explicit Tree-LSTM (Li et al., 2015b). Our codes will be released for public under the Apache License 2.0 at `https://github.com/yunan4nlp/NNDisParser`.

In summary, we mainly make the following two contributions in this work: (1) we propose a transition-based neural RST discourse parsing model with dynamic oracle, (2) we compare three different syntactic integration approaches proposed by us. The rest of the paper is organized as follows. Section 2 describes our proposed models including the transition-based neural model, the dynamic oracle strategy and the implicit syntax feature extraction approach. Section 3 presents the experiments to evaluate our models. Section 4 shows the related work. Finally, section 5 draws conclusions.

2 Transition-based Discourse Parsing

We follow Ji and Eisenstein (2014), exploiting a transition-based framework for RST discourse parsing. The framework is conceptually simple and flexible to support arbitrary features, which has been widely used in a number of NLP tasks (Zhu et al., 2013; Dyer et al., 2015; Zhang et al., 2016). In addition, a transition-based model formalizes a certain task into predicting a sequence of actions, which is essential similar to sequence-to-sequence models proposed recently (Bahdanau et al., 2014). In the following, we first describe the transition system for RST discourse parsing, and then introduce our neural network model by its encoder and decoder parts, respectively. Thirdly, we present our proposed dynamic oracle strategy aiming to enhance the transition-based model. Then we introduce the integration method of implicit syntax features. Finally we describe the training method of our neural network models.

2.1 The Transition-based System

The transition-based framework converts a structural learning problem into a sequence of action predictions, whose key point is a transition system. A transition system consists of two parts: states and actions. The states are used to store partially-parsed results and the actions are used to control state transitions.

Step	Stack	Queue	Action	Relation
1	$\varnothing$	e_1, e_2, e_3, e_4	SH	$\varnothing$
2	e_1	e_2, e_3, e_4	SH	$\varnothing$
3	e_1, e_2	e_3, e_4	RD (attr, SN)	$\varnothing$
4	$e_{1:2}$	e_3, e_4	SH	$\widehat{e_1\mathbf{e_2}}$
5	$e_{1:2}, e_3$	e_4	SH	$\widehat{e_1\mathbf{e_2}}$
6	$e_{1:2}, e_3, e_4$	$\varnothing$	RD (elab, NS)	$\widehat{e_1\mathbf{e_2}}$
7	$e_{1:2}, e_{3:4}$	$\varnothing$	RD (elab, SN)	$\widehat{e_1\mathbf{e_2}}, \widehat{\mathbf{e_3}e_4}$
8	$e_{1:4}$	$\varnothing$	PR	$\widehat{e_1\mathbf{e_2}}, \widehat{\mathbf{e_3}e_4}, \widehat{e_{1:2}\mathbf{e_{3:4}}}$

Table 1: An example of the transition-based system for RST discourse parsing.

The initial state is an empty state, and the final state represents a full result. There are three kinds of actions in our transition system:

- **Shift** (SH), which removes the first EDU in the queue onto the stack, forming a single-node subtree.
- **Reduce** (RD) (l,d), which merges the top two subtrees on the stack, where l is a discourse relation label, and $d \in \{\texttt{NN}, \texttt{NS}, \texttt{SN}\}$ indicates the relation nuclearity (nuclear (N) or satellite (S)).
- **Pop Root** (PR), which pops out the top tree on the stack, marking the decoding being completed, when the stack holds only one subtree and the queue is empty.

Given the RST tree as shown in Figure 1, it can be generated by the following action sequence: {SH, SH, RD (attr, SN), SH, SH, RD (elab, NS), RD (elab, SN), PR}. Table 1 shows the decoding process in detail. By this way, we naturally convert RST discourse parsing into predicting a sequence of transition actions, where each line includes a state and next step action referring to the tree.

2.2 Encoder-Decoder

Previous transition-based RST discourse parsing studies exploit statistical models, using manually-designed discrete features (Sagae, 2009; Heilman and Sagae, 2015; Wang et al., 2017). In this work, we propose a transition-based neural model for RST discourse parsing, which follows an encoder-decoder framework. Given an input sequence of EDUs $\{e_1, e_2, ..., e_n\}$, the encoder computes the input representations $\{\mathbf{h}_1^e, \mathbf{h}_2^e, ..., \mathbf{h}_n^e\}$, and the decoder predicts next step actions conditioned on the encoder outputs.

2.2.1 Encoder

We follow Li et al. (2016), using hierarchical Bi-LSTMs to encode the source EDU inputs, where the first-layer is used to represent sequencial words inside of EDUs, and the second layer is used to represent sequencial EDUs. Given an input sentence $\{w_1, w_2, ..., w_m\}$, first we represent each word by its form (e.g., w_i) and POS tag (e.g. t_i), concatenating their neural embeddings. By this way, the input vectors of the first-layer Bi-LSTM are $\{\mathbf{x}_1^w, \mathbf{x}_2^w, ..., \mathbf{x}_m^w\}$, where $\mathbf{x}_i^w = emb(w_i) \oplus emb(t_i)$, and then we apply Bi-LSTM directly, obtaining:

$$\{\mathbf{h}_1^w, \mathbf{h}_2^w, ..., \mathbf{h}_m^w\} = \text{Bi-LSTM}(\{\mathbf{x}_1^w, \mathbf{x}_2^w, ..., \mathbf{x}_m^w\}) \quad (1)$$

The second-layer Bi-LSTM is built over sequential EDUs. We should first obtain a suitable representation for each EDU, which is composed by a span of words inside a certain sentence. Assuming an EDU with its words by $\{w_s, w_{s+1}, ..., w_t\}$, after applying the first-layer Bi-LSTM, we obtain their representations by $\{\mathbf{h}_s^w, \mathbf{h}_{s+1}^w ..., \mathbf{h}_t^w\}$, then we calculate the EDU representation by average pooling:

$$\mathbf{x}^e = \frac{1}{t-s+1} \sum_{s}^{t} \mathbf{h}_k^w \quad (2)$$

When the EDU representations are ready, we apply the second-layer Bi-LSTM directly, resulting:

$$\{\mathbf{h}_1^e, \mathbf{h}_2^e, ..., \mathbf{h}_n^e\} = \text{Bi-LSTM}(\{\mathbf{x}_1^e, \mathbf{x}_2^e, ..., \mathbf{x}_n^e\}) \quad (3)$$

Algorithm 1 Dynamic Oracle.

$S \leftarrow empty$;
$a_g \leftarrow Oracle(S, T)$;
$a_p \leftarrow Predict(S)$;
while S is not $terminal$ **do**
    $Train(S, a_g)$;
    if $pick\_gold > \alpha$ **then**
        $S \leftarrow Apply(S, a_g)$;
    else
        $S \leftarrow Apply(S, a_p)$;

2.2.2 Decoder

The decoder part is to predict the next-step action based on a given state. As mentioned before, each transition state has a sequence of subtrees on the stack and a sequence of unprocessed EDUs in the queue. We simply choose the top three stack subtrees (s_0, s_1, s_2) and the first EDU (q_0) in the queue as major clues for prediction, applying a feed-forward neural layer to calculate next-step action scores:

$$\mathbf{o} = \mathbf{W}(\mathbf{h}^{sbt}_{s_0} \oplus \mathbf{h}^{sbt}_{s_1} \oplus \mathbf{h}^{sbt}_{s_2} \oplus \mathbf{h}^{e}_{q_0}) \tag{4}$$

where $\mathbf{o}$ is the output scores and $\mathbf{W}$ is a model parameter.

The hidden vector $\mathbf{h}^{e}_{q_0}$ is obtained directly from the encoder outputs. The hidden vectors $\mathbf{h}^{sbt}_{s_0}$, $\mathbf{h}^{sbt}_{s_1}$ and $\mathbf{h}^{sbt}_{s_2}$ are representations of partially-parsed subtrees. For efficiency, we exploit average pooling over the covering EDUs $\{e_i, e_{i+1}, ..., e_j\}$ of a subtree s to derive its representation:

$$\mathbf{h}^{sbt}_{s} = \frac{1}{j-i+1} \sum_{i}^{j} \mathbf{h}^{e}_{k} \tag{5}$$

2.3 Dynamic Oracle

The transition-based model converts discourse parsing into an incremental action prediction problem. For a given state, we require its answer for training. When the state follows the traces of gold-standard actions, its answer is exactly the next gold-standard action. Majority work trains a classifier based on gold-standard state-answer pairs (Ji and Eisenstein, 2014; Heilman and Sagae, 2015; Braud et al., 2017; Wang et al., 2017). However, we usually have error states during the test stage, which are resulted by historical error actions. These error states have never been seen during training, which makes prediction difficult for them. The phenomenon brings inconsistency between training and testing.

Here we adopt dynamic oracle to alleviate the above problem, which has been firstly exploited by Goldberg and Nivre (2012) for dependency parsing. The main idea is to add error states and their oracle actions into the training corpus randomly. On the one hand, by exploring a percentage of error states, the transition-based model is able to handle these states more confidently. On the other hand, by defining oracle actions over error states, the model is able to make best choices even when errors occur.

In this work, we suggest a similar dynamic oracle strategy for our transition-based neural model. Algorithm 1 shows the pseudo codes. During training, we explore into predicted states randomly by a probability α. At each step, we generate a random number $pick\_gold$ between [0,1]. If the value is less than α, we use the predicted state as the next-step input, making the training process consistent with the testing. The oracle function is defined by minimizing the future decoding loss referring to gold-standard RST trees, which is defined as follows:

$$a_g = \begin{cases} \text{RD},\ s.t.\ \widehat{s_0 s_1} \text{ is a subtree in G (relation labels and nuclearities are neglected)} \\ \text{SH},\ \text{otherwise,} \end{cases} \tag{6}$$

where s_0, s_1 are top two stack subtrees, and G is the set of gold-standard subtrees. Simply speaking, if s_0 and s_1 can be composed as a subtree according to the gold-standard RST-tree, we apply RD, and otherwise we apply SH.

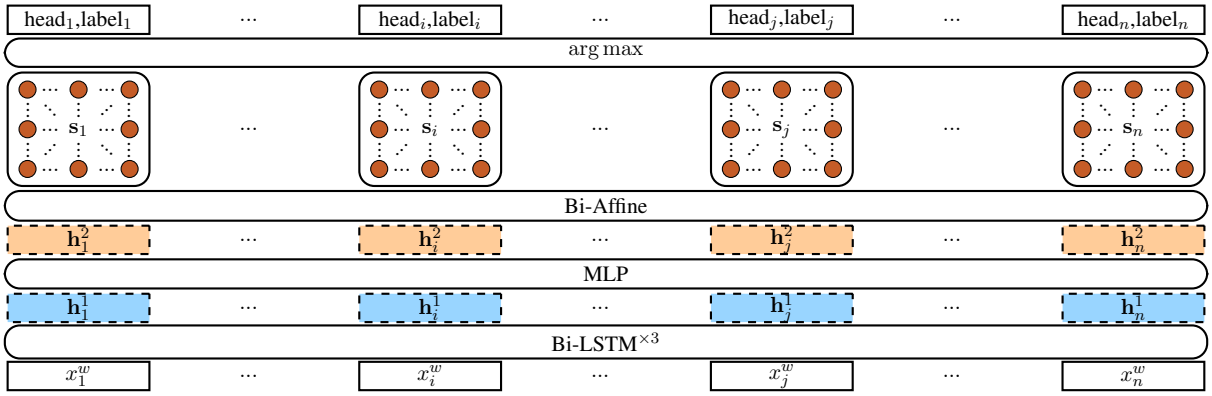


Figure 2: The framework of Bi-Affine neural parser.

2.4 Syntax Features

Intuitively, syntax information could be a valuable source for RST parsing (Soricut and Marcu, 2003; Sagae, 2009; Hernault et al., 2010; Feng and Hirst, 2012; Joty et al., 2013; Feng and Hirst, 2014; Heilman and Sagae, 2015). Previously, the information is integrated into RST discourse parsing by manually-designed discrete features, except the work of Li et al. (2015b), which exploits a bi-directional Tree-LSTM to model syntax trees.

In addition, all previous studies use deterministic parsing outputs as inputs for RST discourse parsing, which may suffer from error propagation problem since syntax parsing outputs can have errors. To alleviate this problem, Zhang et al. (2017) suggest an alternative method, which extracts intermediate hidden outputs of a neural parsing model as inputs for relation-extraction. The method exploits an implicit way to incorporate syntax features without the the need of parsing outputs. Inspired by their work, we investigate the same method for RST discourse parsing.

We follow Zhang et al. (2017) to use the Bi-Affine dependency neural parser of Dozat and Manning (2016) for syntax feature extraction. Zhang et al. (2017) formalize the Bi-Affine parser as an encoder-decoder architecture, and exploit the encoder outputs as inputs for relation extraction. Here we study the Bi-Affine parser in deep, examining other hidden layers as well. Next we will introduce the Bi-Affine parser briefly, and then describe our implicit syntax feature extraction method.

2.4.1 The Bi-Affine Parser

The Bi-Affine neural dependency parser has achieved the top performances (Dozat and Manning, 2016). As shown by Figure 2, the model has four components: (1) a three-layer Bi-LSTM (Bi-LSTM$^{\times 3}$) over the input sentential words; (2) a multi-layer perceptron (MLP) layer; (3) a Bi-Affine layer; (4) an $\arg\max$ decoder. Their outputs are $\{\mathbf{h}_1^1, \mathbf{h}_2^1 ..., \mathbf{h}_n^1\}$, $\{\mathbf{h}_1^2, \mathbf{h}_2^2, ..., \mathbf{h}_n^2\}$, $\{\mathbf{s}_{ij}^l, i \in [1,n], j \in [1,n], l \in L$ (The set of dependency labels.)$\}$, and $\{(\text{head}_1, \text{label}_1), ..., (\text{head}_n, \text{label}_n)\}$, respectively, which are computed by follow formulas:

$$
\begin{aligned}
&\{\mathbf{h}_1^1, \mathbf{h}_2^1, ..., \mathbf{h}_n^1\} = \text{Bi-LSTM}^{\times 3}(\{\mathbf{x}_1^w, \mathbf{x}_2^w, ..., \mathbf{x}_n^w\}) \\
&\{\mathbf{h}_1^2, \mathbf{h}_2^2, ..., \mathbf{h}_n^2\} = \text{MLP}(\{\mathbf{h}_1^1, \mathbf{h}_2^1, ..., \mathbf{h}_n^1\}) \\
&\mathbf{s}_{i,j}^l = \text{Bi-Affine}(\mathbf{h}_i^2, \mathbf{h}_j^2, l) \\
&\text{head}_i, \text{label}_i = argmax_{\{h,l\}}(\mathbf{s}_{i,h}^l)
\end{aligned}
\tag{7}
$$

Noticeably, the above formulas are only an approximate description of the Bi-Affine parser. For more details, one can refer to their paper.

2.4.2 Implicit Syntax Feature Extraction

Zhang et al. (2017) formalize the Bi-Affine parser as an encoder-decoder model, where the encoder part includes the neural structures ending by the Bi-LSTM$^{\times 3}$, and they use the encoder outputs for implicit syntax feature extraction. Additionally, by carefully examining, we find that the MLP outputs have

similar attributes to the Bi-LSTM$^{\times 3}$ outputs, both of which are aligned with the input sentential words perfectly, being closely related to the words at the aligned positions. In addition, the MLP outputs are closer to the final syntax parsing outputs, thus are able to encode syntax information more sufficiently.

Concretely, for a given sentence $\{w_1, w_2, ..., w_n\}$, we feed it into the Bi-Affine parser, extracting the encoder outputs $\{\mathbf{h}_1^1, \mathbf{h}_2^1, ..., \mathbf{h}_n^1\}$ or the MLP outputs $\{\mathbf{h}_1^2, \mathbf{h}_2^2, ..., \mathbf{h}_n^2\}$, respectively. Then we concatenate them with the encoder outputs of our transition-based neural RST parsing model. The resulting vectors are used as inputs for the decoder part. The overall syntax incorporation process can be formalized as follows:

$$\{\mathbf{h}'_1, \mathbf{h}'_2, ..., \mathbf{h}'_n\} = \{\mathbf{h}_1^w \oplus \mathbf{h}_1^*, \mathbf{h}_2^w \oplus \mathbf{h}_2^*, ..., \mathbf{h}_n^w \oplus \mathbf{h}_n^*\} \tag{8}$$

where $*$ could be either 1 or 2, denoting the Bi-LSTM$^{\times 3}$ or the MLP outputs.

2.5 Training

We follow the majority work of transition-based deterministic neural models, using a cross-entropy loss plus with an l_2 regularization as our training objective. Given a state S and its oracle action a, we calculate the decode outputs by using equation (4), and then apply a softmax operation to obtain the oracle action probalility: $p_{a_g} = \frac{exp(\mathbf{o}_{a_g})}{\sum_{a' \in A} exp(\mathbf{o}_{a'})}$, which is then fed into the cross-entropy function:

$$L(\Theta) = -log(p_{a_g}) + \frac{\lambda}{2}||\Theta||^2 \tag{9}$$

where Θ is the set of model parameter and λ is an l_2 regularization factor. We train our model by online learning with mini-batch, updating model parameters by using Adam algorithm (Kingma and Ba, 2014).

3 Experiments

3.1 Settings

3.1.1 Data

We follow previous studies (Ji and Eisenstein, 2014; Li et al., 2014; Feng and Hirst, 2014; Heilman and Sagae, 2015; Li et al., 2016; Wang et al., 2017; Braud et al., 2017), conducting experiments by using the RST discourse Treebank (Carlson et al., 2003). It has annotated 385 documents, where 347 for training and the remaining 38 for testing. All the documents are collected from Wall Street Journal (WSJ), belonging to newswire gene. We use 18 coarse-grained relations in our experiments. For preprocessing, we exploit the StanFord CoreNLP toolkit[1](Manning et al., 2014) to lemmatize words and get their POS tags. To facilitate parameter tuning, we select 35 documents from the training corpus randomly as the development corpus. All experiments are conducted on manually segmented EDUs.

3.1.2 Evaluation

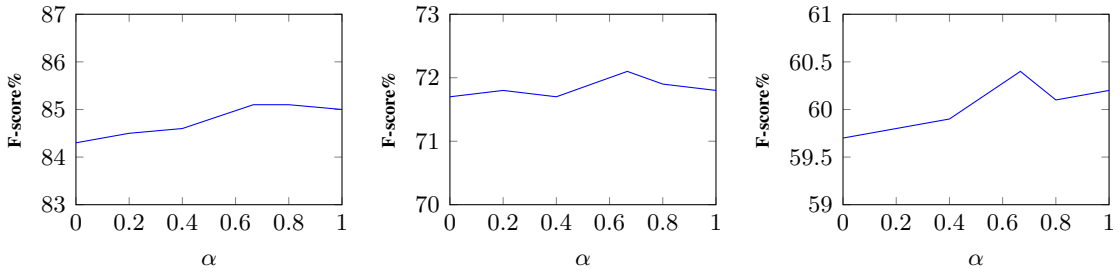
We adopt standard evaluation methods (Marcu, 2000) to test model performances, including three metrics `Span`, `Nuclearity` and `Relation`. The `Span` evaluates the capability of predicting tree skeletons, the `Nuclearity` evaluates tree skeletons and nuclearity indications, and the `Relation` evaluates tree skeletons together with discourse relations, while ignoring nuclearities. The `Full` evaluates the tree skeletons together with both nuclearity indications and discourse relations. We follow (Morey et al., 2017) to report the micro-averaged F scores.

3.1.3 Hyper-Parameters

There are several hyper-parameters in our proposed models. We set the values according to their development performances. We set sizes of all hidden vectors (e.g. the dimension of word embeddings, the Bi-LSTM outputs and etc.) by 200. The word embeddings are initialized by pre-trained GloVe embeddings[2](Pennington et al., 2014). For training, the l_2 regularization factor, the mini-batch size, the initial learning rate, the dropout probability and the max norm of gradient clipping are set by 10^{-6}, 8, 0.001,

[1]https://stanfordnlp.github.io/CoreNLP/

[2]http://nlp.stanford.edu/data/wordvecs/glove.6B.zip



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

Figure 3: The influence of the dynamic oracle strategy in different α.

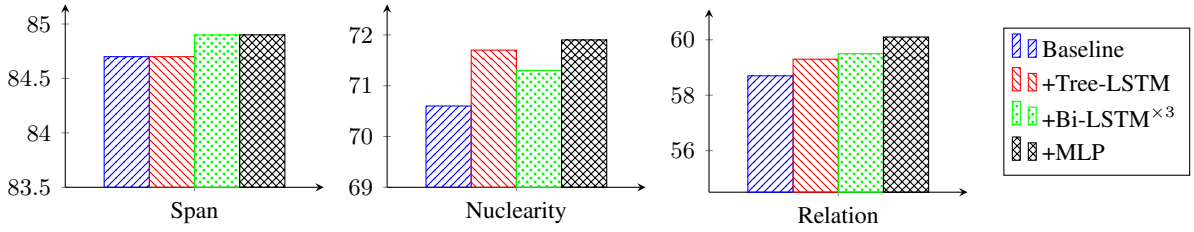


Figure 4: The influence of different syntax feature integration methods.

0.5 and 10, respectively. We train models iteratively on the entire training corpus by 60 rounds, and use the best-performance iteration on the development corpus as the final model.

3.2 Development Results

We conduct two sets of development experiments to show important factors of our proposed model, which are related to dynamic oracle and syntax features, respectively.

3.2.1 Dynamic Orcale

First, we examine the strategy of dynamic oracle. As shown in Algorithm 1, there is a hyper-parameter α to control the exploration, which is used to produce extra training instances. Here we study how α influences the model performances. Figure 3 shows the development performances with respect to α. The performances are relatively stable surrounding 0.7, where all metrics under both settings are close to their peak values. Thus we exploit $\alpha = 0.7$ as the final setting.

3.2.2 Syntax Features

There are two choices of our implicit syntax feature extraction approach. We can use either the encoder Bi-LSTM$^{\times 3}$ outputs or the MLP outputs as described in section 2.4. In addition, we implement a bi-directional dependency based Tree-LSTM (Teng and Zhang, 2017) as well for comparisons. Figure 4 shows the results. First, we find that syntax features[3] are very useful for RST discourse parsing, which is consistent with previous observations (Soricut and Marcu, 2003; Sagae, 2009; Braud et al., 2016). Second, the implicit syntax feature extraction method is slightly better than explicit Tree-LSTM. In particular, the model can achieve the best performances by using the MLP outputs. Thus we exploit the MLP outputs as the final implicit syntax features in our RST discourse parsing model.

[3] We use the English PTB corpus to train the Bi-Affine parser, which achieves 95.88% UAS and 94.16% LAS.

Model	Span	Nuclearity	Relation	Full
(Hayashi et al., 2016)	82.6	66.6	54.6	54.3
(Surdeanu et al., 2015)	82.6	67.1	55.4	54.9
(Joty et al., 2015)	82.6	68.3	55.8	55.4
(Feng and Hirst, 2014)	84.3	69.4	56.9	56.2
(Braud et al., 2016)	79.7	63.6	47.7	47.5
(Li et al., 2016)	82.2	66.5	51.4	50.6
(Braud et al., 2017)	81.3	68.1	56.3	56.0
(Ji and Eisenstein, 2014)	82.0	68.2	57.8	57.6
Baseline	85.0	71.0	57.6	57.1
Our Final Model	**85.5**	**73.1**	**60.2**	**59.9**
Human	88.3	77.3	65.4	64.7

Table 2: Final micro-averaged F scores on the test corpus. The results of other models are borrowed from Morey et al. (2017).

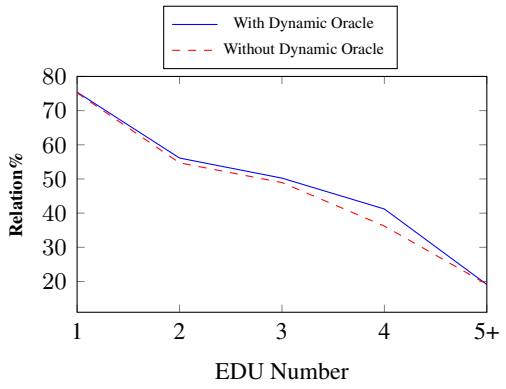


Figure 5: The span-level F-measures with respect to the number of EDU.

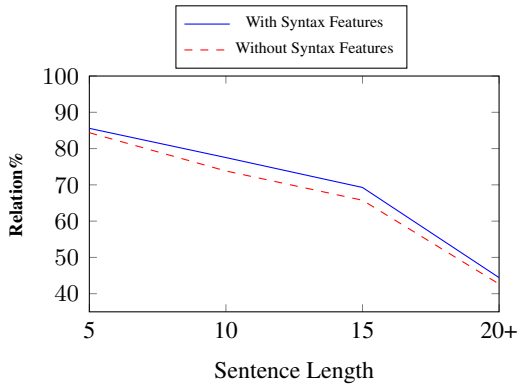


Figure 6: The span-level F-measures with respect to the span length.

3.3 Final Results

Table 2 shows the final results of our proposed neural models on the test corpus. Our baseline model achieves a `Span` F-measure of 85.0 and a `Nuclearity` F-measure of 71.0, which are exceed most models. When dynamic oracle and implicit syntax features are exploited, our final model achieves 60.2 on the `Relation` F-measure, resulting overall improvements 60.2 - 57.6 = 2.6. The `Span`, `Nucleartiy` and `Full` metrics have similar tendencies as well.

We compare the proposed neural model with other state-of-the-art systems as well. (Hayashi et al., 2016) is a greedy bottom-up parser with a linear SVM classification. (Surdeanu et al., 2015) is also a greedy bottom-up parser, which uses the perceptron for tree skeleton building and nuclearity indication, the logistic regression for relation labelling. (Joty et al., 2015) is a two-stage (intra-sentential then multi-sentential parsing) parser with dynamic conditional random field models, and (Feng and Hirst, 2014) is a two-stage parser with linear-chain conditional random field models. (Braud et al., 2016) is a sequence-to-sequence hierarchical neural parser, and (Li et al., 2016) is an attention-based hierarchical neural parser. (Ji and Eisenstein, 2014) is a transition-based statisitical model with the representation learning. (Braud et al., 2017) is a transition-based neural model by embedding and composing a number of manually-designed sophisticated atomic features. As shown by Table 2, we can see that transition-based models are able to achieve state-of-the-art performances under both discrete and neural settings, demonstrating the robustness of this framework. Our final model is able to achieve competitive results compared with these systems.

3.4 Analysis

In this subsection, we perform several analysis experiments on the test corpus to illustrate the benefits from dynamic oracle and syntax features, respectively.

First, in order to understand the influence of dynamic oracle, we evaluate the `Relation` performances with respect to the EDU number. In our transition-based model, a subtree with more EDUs requires more transition actions to be produced. As introduced in section 2.3, our neural model with dynamic oracle is robust to error input states. Figure 5 shows the comparison results. We can find that the model with dynamic oracle performs better on spans of the EDU number between [2,4], which is consistent with our intuitions. However, when the EDU number increases to 5+, both settings perform very poorly.

Second, we investigate the benefits by using our proposed syntax features. Intuitively, the dependency parsing outputs exploit tree structures to re-organize sentential words, thus a number of long-distance word pairs are connected directly. We would expect that spans covering a long range of words could be better handled with syntax features. We verify this assumption by comparing performances with respect to the word number in span. As shown by Figure 6, we can see that syntax features do improve the performances of spans containing more words. The observation is consistent with our assumption.

4 Related Work

RST discourse parsing has been studied intensively since early (Marcu, 1997; Marcu, 1999; Soricut and Marcu, 2003). Initial work focuses on statistical models by using manually-designed feature (Soricut and Marcu, 2003; Sagae, 2009; Hernault et al., 2010; Feng and Hirst, 2012; Joty et al., 2013; Feng and Hirst, 2014; Heilman and Sagae, 2015; Wang et al., 2017). Among these studies, transition-based models have achieved state-of-the-art performances (Sagae, 2009; Ji and Eisenstein, 2014; Heilman and Sagae, 2015; Wang et al., 2017). Recently, neural network models have been investigated. Braud et al. (2017) have proposed a transtion-based neural model by using several well-designed atomic features, embedding them directly and then feeding them into feed-forward neural networks. In this work, we suggest hierarchical Bi-LSTMs, following Li et al. (2016) to encode documents, which are more popular in document modeling (Li et al., 2015a; Chen et al., 2016).

Syntax features have been demonstrated useful for RST discourse parsing by a number of studies (Soricut and Marcu, 2003; Sagae, 2009; Hernault et al., 2010; Feng and Hirst, 2012; Joty et al., 2013; Feng and Hirst, 2014; Heilman and Sagae, 2015). The majority work exploits manually-designed features to achieve this goal. The work of Li et al. (2015b) have compared a Tree-LSTM structrue over syntax trees with a sequential LSTM over input sentences, finding that the syntax tree does not show better performances. In this work, we propose a different implicit feature extraction to represent syntax information, using them as a supplementary for RST discourse parsing.

The exploration of dynamic oracle has been proposed for transition-based dependency parsing (Goldberg and Nivre, 2012), bringing significantly better performances for dependency parsing. Our dynamic oracle is mainly inspired by this work, and we apply it on RST discourse parsing. To our knowledge, it is the first work of transition-based RST parsing by using dynamic oracle.

Zhang et al. (2017) suggest a new method of integrating syntax features implicitly. First they extract a sequence of hidden vectors from a supervised neural dependency parsing model, and then feed these implicit syntax features into a neural relation extraction model. The method has been also applied to targeted sentiment analysis (Gao et al., 2017). We follow their work to investigate the implicit syntax feature extraction for RST discourse parsing, and improve this method accordingly for our task.

5 Conclusion

In this work, we investigated an implicit syntax feature extraction method for neural RST discourse parsing. First, we proposed a transition-based neural baseline, and further enhanced it with a dynamic oracle mechanism. Second, we examined the implicit syntax feature extraction method proposed by Zhang et al. (2017) and suggested use outputs of a different neural layer of a Bi-Affine dependency parsing model (Dozat and Manning, 2016). We conducted experiments on standard RST discourse TreeBank (Carlson et al., 2003) to evaluate our proposed models. Results show that our transition-based parser is very competitive after applying dynamic oracle. Further, we find that the proposed implicit syntax features are highly effective, better than explicit Tree-LSTMs.

Acknowledgments

We thank the anonymous reviewers for their constructive comments, which help to improve the paper. This work is supported by National Natural Science Foundation of China (NSFC) grants 61672211 and 61602160, Natural Science Foundation of Heilongjiang Province (China) grant F2016036, Special business expenses in Heilongjiang Province (China) grant 2016-KYYWF-0183.

References

[Bahdanau et al.2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Braud et al.2016] Chloé Braud, Barbara Plank, and Anders Søgaard. 2016. Multi-view and multi-task training of rst discourse parsers. In *Proceedings of COLING 2016, the 26th ICCL: Technical Papers*, pages 1903–1913.

[Braud et al.2017] Chloé Braud, Maximin Coavoux, and Anders Søgaard. 2017. Cross-lingual rst discourse parsing. *arXiv preprint arXiv:1701.02946*.

[Carlson et al.2003] Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current and new directions in discourse and dialogue*, pages 85–112. Springer.

[Chen et al.2016] Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on EMNLP*, pages 1650–1659.

[Collobert et al.2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

[Devlin et al.2014] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the ACL (Volume 1: Long Papers)*, volume 1, pages 1370–1380.

[Dozat and Manning2016] Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

[Dyer et al.2015] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

[Feng and Hirst2012] Vanessa Wei Feng and Graeme Hirst. 2012. Text-level discourse parsing with rich linguistic features. In *Proceedings of the 50th Annual Meeting of the ACL: Long Papers-Volume 1*, pages 60–68. Association for Computational Linguistics.

[Feng and Hirst2014] Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the 52nd Annual Meeting of the ACL (Volume 1: Long Papers)*, volume 1, pages 511–521.

[Gao et al.2017] Yuze Gao, Yue Zhang, and Tong Xiao. 2017. Implicit syntactic features for target-dependent sentiment analysis. In *Proceedings of the Eighth IJCNLP Processing (Volume 1: Long Papers)*, pages 516–524, Taipei, Taiwan, November. Asian Federation of Natural Language Processing.

[Goldberg and Nivre2012] Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. *Proceedings of COLING 2012*, pages 959–976.

[Hayashi et al.2016] Katsuhiko Hayashi, Tsutomu Hirao, and Masaaki Nagata. 2016. Empirical comparison of dependency conversions for rst discourse trees. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 128–136.

[Heilman and Sagae2015] Michael Heilman and Kenji Sagae. 2015. Fast rhetorical structure theory discourse parsing. *arXiv preprint arXiv:1505.02425*.

[Hernault et al.2010] Hugo Hernault, Helmut Prendinger, Mitsuru Ishizuka, et al. 2010. Hilda: A discourse parser using support vector machine classification. *Dialogue & Discourse*, 1(3).

[Ji and Eisenstein2014] Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the ACL (Volume 1: Long Papers)*, volume 1, pages 13–24.

[Joty et al.2013] Shafiq Joty, Giuseppe Carenini, Raymond Ng, and Yashar Mehdad. 2013. Combining intra-and multi-sentential rhetorical parsing for document-level discourse analysis. In *Proceedings of the 51st Annual Meeting of the ACL (Volume 1: Long Papers)*, volume 1, pages 486–496.

[Joty et al.2015] Shafiq Joty, Giuseppe Carenini, and Raymond T Ng. 2015. Codra: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*, 41(3):385–435.

[Kingma and Ba2014] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Li et al.2014] Jiwei Li, Rumeng Li, and Eduard Hovy. 2014. Recursive deep models for discourse parsing. In *Proceedings of the 2014 Conference on EMNLP*, pages 2061–2069.

[Li et al.2015a] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015a. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

[Li et al.2015b] Jiwei Li, Minh-Thang Luong, Dan Jurafsky, and Eudard Hovy. 2015b. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185*.

[Li et al.2016] Qi Li, Tianshi Li, and Baobao Chang. 2016. Discourse parsing with attention-based hierarchical neural networks. In *Proceedings of the 2016 Conference on EMNLP*, pages 362–371.

[Liu and Lapata2017] Yang Liu and Mirella Lapata. 2017. Learning contextually informed representations for linear-time discourse parsing. In *Proceedings of the 2017 Conference on EMNLP*, pages 1289–1298.

[Mann and Thompson1988] William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

[Manning et al.2014] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL System Demonstrations*, pages 55–60.

[Marcu1997] Daniel Marcu. 1997. The rhetorical parsing of natural language texts. In *Proceedings of the 35th Annual Meeting of the ACL and Eighth Conference of EACL*, pages 96–103. Association for Computational Linguistics.

[Marcu1999] Daniel Marcu. 1999. A decision-based approach to rhetorical parsing. In *Proceedings of the 37th annual meeting of the ACL on Computational Linguistics*, pages 365–372. Association for Computational Linguistics.

[Marcu2000] Daniel Marcu. 2000. *The theory and practice of discourse parsing and summarization*. MIT press.

[Mikolov et al.2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Morey et al.2017] Mathieu Morey, Philippe Muller, and Nicholas Asher. 2017. How much progress have we made on rst discourse parsing? a replication study of recent results on the rst-dt. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1319–1324.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on EMNLP*, pages 1532–1543.

[Sagae2009] Kenji Sagae. 2009. Analysis of discourse structure with syntactic dependencies and data-driven shift-reduce parsing. In *Proceedings of the 11th ICPT*, pages 81–84. Association for Computational Linguistics.

[Schmidhuber and Hochreiter1997] Jürgen Schmidhuber and Sepp Hochreiter. 1997. Long short-term memory. *Neural Comput*, 9(8):1735–1780.

[Soricut and Marcu2003] Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of NAACL on Human Language Technology-Volume 1*, pages 149–156. Association for Computational Linguistics.

[Surdeanu et al.2015] Mihai Surdeanu, Tom Hicks, and Marco Antonio Valenzuela-Escarcega. 2015. Two practical rhetorical structure theory parsers. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: Demonstrations*, pages 1–5.

[Teng and Zhang2017] Zhiyang Teng and Yue Zhang. 2017. Head-lexicalized bidirectional tree lstms. *Transactions of the ACL*, 5:163–177.

[Wang et al.2017] Yizhong Wang, Sujian Li, and Houfeng Wang. 2017. A two-stage parsing method for text-level discourse analysis. In *Proceedings of the 55th Annual Meeting of the ACL (Volume 2: Short Papers)*, volume 2, pages 184–188.

[Zhang et al.2016] Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Transition-based neural word segmentation. In *Proceedings of the 54th Annual Meeting of the ACL (Volume 1: Long Papers)*, volume 1, pages 421–431.

[Zhang et al.2017] Meishan Zhang, Yue Zhang, and Guohong Fu. 2017. End-to-end neural relation extraction with global optimization. In *Proceedings of the 2017 Conference on EMNLP*, pages 1730–1740.

[Zhu et al.2013] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the ACL (Volume 1: Long Papers)*, volume 1, pages 434–443.

Deep Enhanced Representation for Implicit Discourse Relation Recognition

Hongxiao Bai[1,2], **Hai Zhao**[1,2,*]
[1]Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai, China
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, China
baippa@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn

Abstract

Implicit discourse relation recognition is a challenging task as the relation prediction without explicit connectives in discourse parsing needs understanding of text spans and cannot be easily derived from surface features from the input sentence pairs. Thus, properly representing the text is very crucial to this task. In this paper, we propose a model augmented with different grained text representations, including character, subword, word, sentence, and sentence pair levels. The proposed deeper model is evaluated on the benchmark treebank and achieves state-of-the-art accuracy with greater than 48% in 11-way and F_1 score greater than 50% in 4-way classifications for the first time according to our best knowledge.

1 Introduction

Discourse parsing is a fundamental task in natural language processing (NLP) which determines the structure of the whole discourse and identifies the relations between discourse spans such as clauses and sentences. Improving this task can be helpful to many downstream tasks such as machine translation (Li et al., 2014), question answering (Jansen et al., 2014), and so on. As one of the important parts of discourse parsing, implicit discourse relation recognition task is to find the relation between two spans without explicit connectives (e.g., *but, so*, etc.), and needs recovering the relation from semantic understanding of texts.

The Penn Discourse Treebank 2.0 (PDTB 2.0) (Prasad et al., 2008) is a benchmark corpus for discourse relations. In PDTB style, the connectives can be explicit or implicit, and one entry of the data is separated into *Arg1* and *Arg2*, accompanied with a relation sense. Since the release of PDTB 2.0 dataset, many methods have been proposed, ranging from traditional feature-based methods (Lin et al., 2009; Pitler et al., 2009) to latest neural-based methods (Qin et al., 2017; Lan et al., 2017). Especially through many neural network methods used for this task such as convolutional neural network (CNN) (Qin et al., 2016b), recursive neural network (Ji and Eisenstein, 2015), embedding improvement (Wu et al., 2017), attention mechanism (Liu and Li, 2016), gate mechanism (Chen et al., 2016), multi-task method (Lan et al., 2017), the performance of this task has improved a lot since it was first introduced. However, this task is still very challenging with the highest reported accuracy still lower than 50% due to the hardness for the machines to understand the text meaning and the relatively small task corpus.

In this work, we focus on improving the learned representations of sentence pairs to address the implicit discourse relation recognition. It is well known that text representation is the core part of state-of-the-art deep learning methods for NLP tasks, and improving the representation from all perspective will benefit the concerned task.

The representation is improved by two ways in our model through three-level hierarchy. The first way is embedding augmentation. Only with informative embeddings, can the final representations be better. This is implemented in our word-level module. We augment word embeddings with subword-level

*Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100), National Natural Science Foundation of China (No. 61672343 and No. 61733011), Key Project of National Society Science Foundation of China (No. 15-ZDA041), The Art and Science Interdisciplinary Funds of Shanghai Jiao Tong University (No. 14JCRZ04).

Proceedings of the 27th International Conference on Computational Linguistics, pages 571–583
Santa Fe, New Mexico, USA, August 20-26, 2018.

embeddings and pre-trained ELMo embeddings. Subwords coming from unsupervised segmentation demonstrate a better consequent performance than characters for being a better minimal representation unit. The pre-trained contextualized word embeddings (ELMo) can make the embeddings contain more contextual information which is also involved with character-level inputs. The second way is a deep residual bi-attention encoder. Since this task is about classifying sentence pairs, the encoder is implemented in sentence and sentence-pair levels. A deeper model can support richer representations but is hard to train, especially with a small dataset. So we apply residual connections (He et al., 2016) to each module for facilitating signal propagation and alleviating gradient degradation. The stacked encoder blocks make the single sentence representation richer and bi-attention module mixes two sentence representations focusingly. With introducing richer and deeper representation enhancement, we report the deepest model so far for the task.

Our representation enhanced model will be evaluated on the benchmark PDTB 2.0 and demonstrate state-of-the-art performance to verify its effectiveness.

This paper is organized as follows. Section 2 reviews related work. Section 3 introduces our model. Section 4 shows our experiments and analyses the results. Section 5 concludes this work.

2 Related Work

After the release of Penn Discourse Treebank 2.0, many works have been made to solve this concerned task. Lin et al. (2009) is the first work who considered the second-level classification of the task by empirically evaluating the impact of surface features. Feature based methods (Pitler et al., 2009; Zhou et al., 2010; Chen et al., 2015; Li et al., 2016) mainly focused on using linguistic, or semantic features from the discourse units, or the relations between unit pairs and word pairs. Zhang et al. (2015) is the first one who modeled this task using end-to-end neural network and gained great performance improvement. Neural network methods also used by lots of works (Zhang et al., 2016; Cai and Zhao, 2016) for better performance. Since then, a lot of methods have been proposed. Braud and Denis (2015) found that word embeddings trained by neural networks is very useful to this task. Qin et al. (2016a) augmented their system with character-level and contextualized embeddings. Recurrent networks and convolutional networks have been used as basic blocks in many works (Ji et al., 2016; Rönnqvist et al., 2017; Qin et al., 2016b). Ji and Eisenstein (2015) used recursive neural networks. Attention mechanism was used by Liu and Li (2016), Cai and Zhao (2017) and others. Wu et al. (2016) and Lan et al. (2017) applied multi-task component. Qin et al. (2017) utilized adversarial nets to migrate the connective-based features to implicit ones.

Sentence representation is a key component in many NLP tasks. Usually, better representation means better performance. Plenty of work on language modeling has been done, as language modeling can supply better sentence representations. Since the pioneering work of Bengio et al. (2006), neural language models have been well developed (Mikolov and Zweig, 2012; Williams et al., 2015; Kim et al., 2016). Sentence representation is directly handled in a series of work. Lin et al. (2017) used self attention mechanism and used matrix to represent sentence, and Conneau et al. (2017) used encoders pre-trained on SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2017).

Different from all the existing work, for the first time to our best knowledge, this work is devoted to an empirical study on different levels of representation enhancement for implicit discourse relation classification task.

3 Model

3.1 Overview

Figure 1 illustrates an overview of our model, which is mainly consisted of three parts: word-level module, sentence-level module, and pair-level module. Token sequences of sentence pairs (*Arg1* and *Arg2*) are encoded by word-level module first and every token becomes a word embedding augmented by subword and ELMo. Then these embeddings are fed to sentence-level module and processed by stacked encoder blocks (CNN or RNN encoder block). Every block layer outputs representation for each token. Furthermore, the output of each layer is processed by bi-attention module in the pair-level

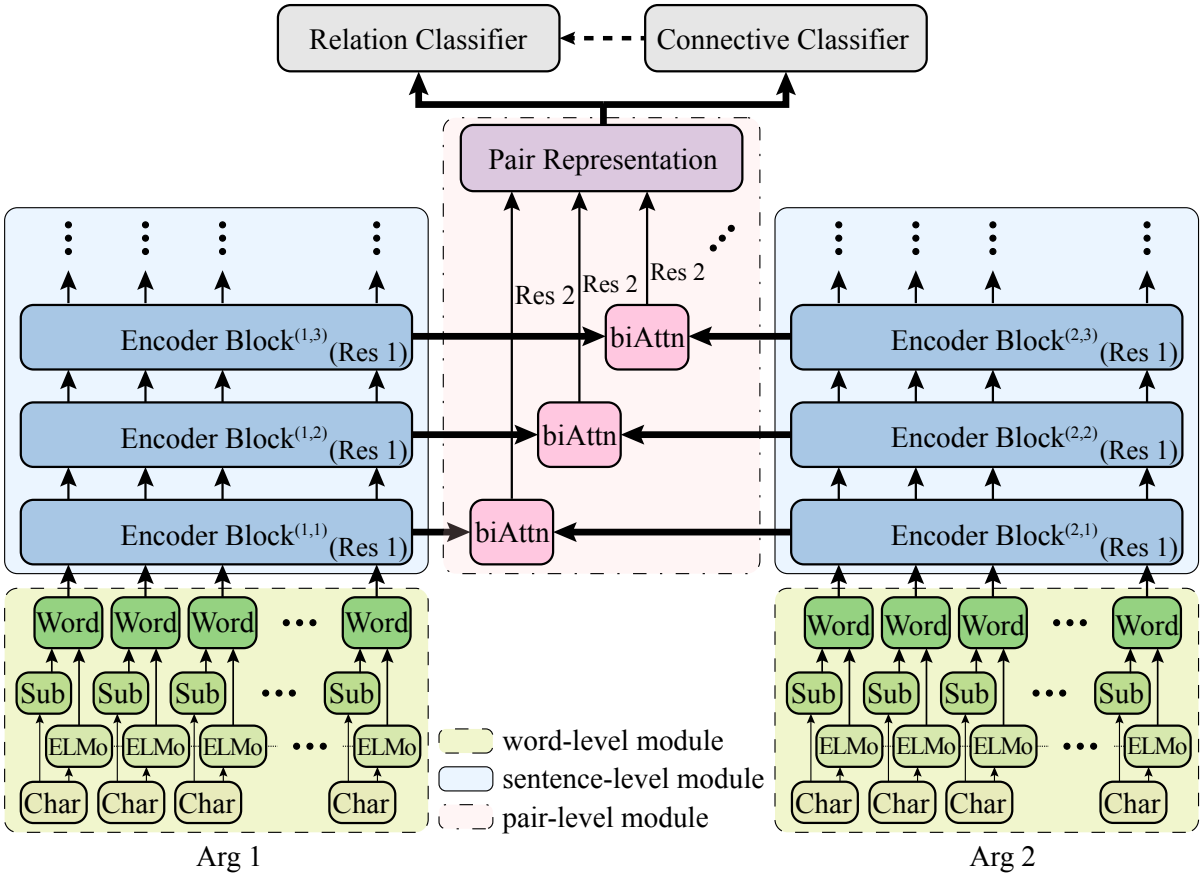


Figure 1: Model overview.

module, and concatenated to pair representation, which is finally sent to classifiers which are multiple layer perceptrons (MLP) with softmax. The model details are given in the rest of this section.

3.2 Word-Level Module

An inputed token sequence of length N is encoded by the word-level module into an embedding sequence $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \cdots, \mathbf{e}_N)$. For each embedded token $\mathbf{e}_i$, it is concatenated from three parts,

$$\mathbf{e}_i = [\mathbf{e}_i^w;\ \mathbf{e}_i^s;\ \mathbf{e}_i^c] \in \mathbb{R}^{d_e} \tag{1}$$

$\mathbf{e}_i^w \in \mathbb{R}^{d_w}$ is pre-trained word embedding for this token, and is fixed during the training procedure. Our experiments show that fine-tuning the embeddings slowed down the training without better performance. $\mathbf{e}_i^s \in \mathbb{R}^{d_s}$ is subword-level embedding encoded by subword encoder. $\mathbf{e}_i^c \in \mathbb{R}^{d_c}$ is contextualized word embedding encoded by pre-trained ELMo encoders, whose parameters are also fixed during training. Subword is merged from single-character segmentation and the input of ELMo encoder is also character.

Subword Encoder

Character-level embeddings have been used widely in lots of works and its effectiveness is verified for out-of-vocabulary (OOV) or rare word representation. However, character is not a natural minimal unit for there exists word internal structure, we thus introduce a subword-level embedding instead.

Subword units can be computationally discovered by unsupervised segmentation over words that are regarded as character sequences. We adopt byte pair encoding (BPE) algorithm introduced by Sennrich et al. (2016) for this segmentation. BPE segmentation actually relies on a series of iterative merging operation over bigrams with the highest frequency. The number of merging operation times is roughly equal to the result subword vocabulary size.

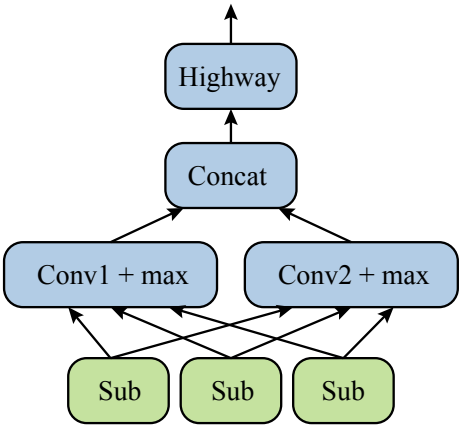


Figure 2: Subword encoder.

For each word, the subword-level embedding is encoded by a subword encoder as in Figure 2. Firstly, the subword sequence (of length n) of the word is mapped to subword embedding sequence $(\mathbf{se}_1, \mathbf{se}_2, \mathbf{se}_3, \cdots, \mathbf{se}_n)$ (after padding), which is randomly initialized. Then K (we empirically set K=2) convolutional operations $Conv_1, Conv_2, \cdots, Conv_K$ followed by max pooling operation are applied to the embedding sequence, and the sequence is padded before the convolutional operation. For the i-th convolution kernel $Conv_i$, suppose the kernel size is k_i, then the output of $Conv_i$ on embeddings $\mathbf{se}_j$ to $\mathbf{se}_{j+k_i-1}$ is

$$\mathbf{C}_j = \tanh\left(Conv_i[\mathbf{se}_j : \mathbf{se}_{j+k_i-1}]\right)$$

The final output of $Conv_i$ after max pooling is

$$\mathbf{u}_i = maxpool\left(\mathbf{C}_1,\ \cdots,\ \mathbf{C}_j,\ \cdots,\ \mathbf{C}_n\right)$$

Finally, the K outputs are concatenated,

$$\mathbf{u} = [\mathbf{u}_1;\ \mathbf{u}_2;\ \cdots;\ \mathbf{u}_K] \in \mathbb{R}^{d_s}$$

to feed a highway network (Srivastava et al., 2015),

$$\begin{aligned} \mathbf{g} &= \sigma(\mathbf{W}_g\mathbf{u}^T + \mathbf{b}_g) \in \mathbb{R}^{d_s} \\ \mathbf{e}_i^s &= \mathbf{g} \odot ReLU(\mathbf{W}_h\mathbf{u}^T + \mathbf{b}_h) + (\mathbf{1} - \mathbf{g}) \odot \mathbf{u} \\ &\in \mathbb{R}^{d_s} \end{aligned} \tag{2}$$

where $\mathbf{g}$ denotes the gate, and $\mathbf{W}_g \in \mathbb{R}^{d_s \times d_s}, \mathbf{b}_g \in \mathbb{R}^{d_s}, \mathbf{W}_h \in \mathbb{R}^{d_s \times d_s}, \mathbf{b}_h \in \mathbb{R}^{d_s}$ are parameters. $\odot$ is element-wise multiplication. The above Eq. 2 gives the subword-level embedding for the i-th word.

ELMo

ELMo (Embeddings from Language Models) (Peters et al., 2018) is a pre-trained contextualized word embeddings involving character-level representation. It is shown useful in some works (He et al., 2018; Lee et al., 2018). This embedding is trained by bidirectional language models on large corpus using character sequence for each word token as input. The ELMo encoder employs CNN and highway networks over characters, whose output is given to a multiple-layer biLSTM with residual connections. Then the output is contextualized embeddings for each word. It is also can be seen as a hybrid encoder for character, word, and sentence. This encoder can add lots of contextual information to each word, and can ease the semantics learning of the model.

For the pre-trained ELMo encoder, the output is the result of the last two biLSTM layers. Suppose $\mathbf{c}_i$ is the character sequence of i-th word in a sentence, then the encoder output is

$$[\cdots, \mathbf{h}_i^0, \cdots; \ \cdots, \mathbf{h}_i^1, \cdots] = ELMo(\cdots, \mathbf{c}_i, \cdots)$$

where $\mathbf{h}_i^0$ and $\mathbf{h}_i^1$ denote the outputs of first and second layers of ELMo encoder for i-th word.

Following Peters et al. (2018), we use a self-adjusted weighted average of $\mathbf{h}_i^0, \mathbf{h}_i^1$,

$$\mathbf{s} = softmax(\mathbf{w}) \in \mathbb{R}^2$$
$$\mathbf{h} = \gamma \sum_{j=0}^{1} s_j \mathbf{h}_i^j \in \mathbb{R}^{d_c'}$$

where $\gamma \in \mathbb{R}$ and $\mathbf{w} \in \mathbb{R}^2$ are parameters tuned during training and d_c' is the dimension of the ELMo encoder's outputs. Then the result is fed to a feed forward network to reduce its dimension,

$$\mathbf{e}_i^c = \mathbf{W}_c \mathbf{h}^T + \mathbf{b}_c \in \mathbb{R}^{d_c} \tag{3}$$

$\mathbf{W}_c \in \mathbb{R}^{d_c' \times d_c}$ and $\mathbf{b}_c \in \mathbb{R}^{d_c}$ are parameters. The above Eq. 3 gives ELMo embedding for the i-th word.

3.3 Sentence-Level Module

The resulting word embeddings $\mathbf{e}_i$ (Eq. 1) are sent to sentence-level module. The sentence-level module is composed of stacked encoder blocks. The block in each layer receives output of the previous layer as input and sends output to next layer. It also sends its output to the pair-level module. Parameters in different layers are not the same.

We consider two encoder types, convolutional type and recurrent type. We only use one encoder type in one experiment.

For the sentence-level module for different arguments (*Arg1* and *Arg2*), many previous works used same parameters to encode different arguments, that is, one encoder for two type arguments. But as indicated by Prasad et al. (2008), *Arg1* and *Arg2* may have different semantic perspective, we thus introduce argument-aware parameter settings for different arguments.

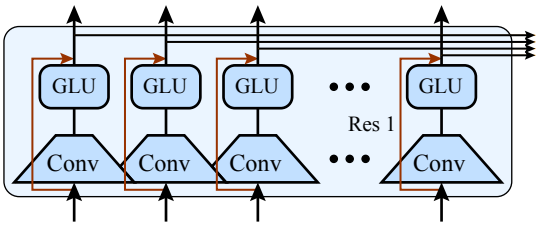


Figure 3: Convolutional encoder block.

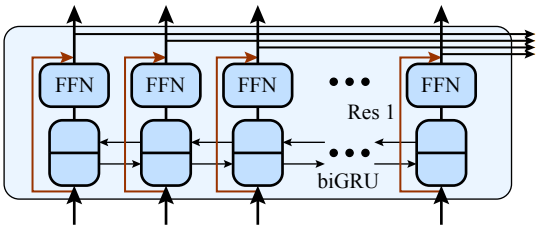


Figure 4: Recurrent encoder block.

Convolutional Encoder Block

Figure 3 is the convolutional encoder block. Suppose the input for the encoder block is $\mathbf{x}_i$ $(i = 1, \cdots, N)$, then $\mathbf{x}_i \in \mathbb{R}^{d_e}$. The input is sent to a convolutional layer and mapped to output $\mathbf{y}_i = [\mathbf{A}_i \ \mathbf{B}_i] \in \mathbb{R}^{2d_e}$. After the convolutional operation, gated linear units (GLU) (Dauphin et al., 2016) is applied, i.e.,

$$\mathbf{z}_i = \mathbf{A}_i \odot \sigma(\mathbf{B}_i) \in \mathbb{R}^{d_e}$$

There is also a residual connection (Res 1) in the block, which means adding the output of GLU and the input of the block as final output, so $\mathbf{z}_i + \mathbf{x}_i$ is the output of the block corresponding to the input $\mathbf{x}_i$. The output $\mathbf{z}_i + \mathbf{x}_i$ for all $i = 1, \cdots, N$ is sent to both the next layer and the pair-level module as input.

Recurrent Encoder Block

Similar to the convolutional one, recurrent encoder block is shown in Figure 4. The input $\mathbf{x}_i$ is encoded by a biGRU (Cho et al., 2014) layer first,

$$\mathbf{y}_i = biGRU(\mathbf{x}_i) \in \mathbb{R}^{2d_e}$$

then this is sent to a feed forword network,

$$\mathbf{z}_i = \mathbf{W}_r \mathbf{y}_i^T + \mathbf{b}_r \in \mathbb{R}^{d_e} \tag{4}$$

$\mathbf{W}_r \in \mathbb{R}^{2d_e \times d_e}$ and $\mathbf{b}_r \in \mathbb{R}^{d_e}$ are parameters. There is also a similar residual connection (Res 1) in the block, so $\mathbf{z}_i + \mathbf{x}_i$ for all $i = 1, \cdots, N$ is the final output of the recurrent encoder block.

3.4 Pair-Level Module

Through the sentence-level module, the word representations are contextualized, and these contextualized representations of each layer are sent to pair-level module.

Suppose the encoder block layer number is l, and the outputs of j-th block layer for *Arg1* and *Arg2* are $\mathbf{v}_1^j, \mathbf{v}_2^j \in \mathbb{R}^{N \times d_e}$, each row of which is the embedding for the corresponding word. N is the length of word sequence (sentence). Each sentence is padded or truncated to let all sentences have the same length. They are sent to a bi-attention module, the attention matrix is

$$\mathbf{M}_j = (FFN(\mathbf{v}_1^j))\mathbf{v}_2^{jT} \in \mathbb{R}^{N \times N}$$

FFN is a feed froward network (similar to Eq. 4) applied to the last dimension corresponding to the word. Then the projected representations are

$$\mathbf{w}_2^j = softmax(\mathbf{M}_j)\mathbf{v}_2^j \in \mathbb{R}^{N \times d_e}$$
$$\mathbf{w}_1^j = softmax(\mathbf{M}_j^T)\mathbf{v}_1^j \in \mathbb{R}^{N \times d_e}$$

where the $softmax$ is applied to each row of the matrix. We apply 2-max pooling on each projected representation and concatenate them as output of the j-th bi-attention module

$$\mathbf{o}_j = [top2(\mathbf{w}_1^j);\ \ top2(\mathbf{w}_2^j)] \in \mathbb{R}^{4d_e}$$

The number of max pooling operation (top-2) is selected from experiments and it is a balance of more salient features and less noise. The final pair representation is

$$\mathbf{o} = [\mathbf{o}_1, \mathbf{o}_2, \cdots, \mathbf{o}_l] \in \mathbb{R}^{4ld_e} \tag{5}$$

Since the output is concatenated from different layers and the outputs of lower layers are sent directly to the final representation, this also can be seen as residual connections (Res 2). Then the output as Eq. 5 is fed to an MLP classifier with softmax. The parameters for bi-attention modules in different levels are shared.

3.5 Classifier

We use two classifiers in our model. One is for relation classification, and another one is for connective classification. The classifier is only a multiple layer perceptron (MLP) with softmax layer. Qin et al. (2017) used adversarial method to utilize the connectives, but this method is not suitable for our adopted attention module since the attended part of a sentence will be distinctly different when the argument is with and without connectives. They also proposed a multi-task method that augments the model with an additional classifier for connective prediction, and the input of it is also the pair representation. It is straightforward and simple enough, and can help the model learn better representations, so we include this module in our model. The implicit connectives are provided by PDTB 2.0 dataset, and the connective classifier is only used during training. The loss function for both classifiers is cross entropy loss, and the total loss is the sum of the two losses, i.e., $Loss = Loss_{relation} + Loss_{connective}$.

4 Experiments[1]

Our model is evaluated on the benchmark PDTB 2.0 for two types of classification tasks.

PDTB 2.0 has three levels of senses: Level-1 *Class*, Level-2 *Type*, and Level-3 *Subtypes*. The first level consists of four major relation Classes: COMPARISON, CONTINGENCY, EXPANSION, and TEMPORAL. The second level contains 16 Types.

All our experiments are implemented by PyTorch[2]. The pre-trained ELMo encoder is from AllenNLP toolkit (Gardner et al., 2017) [3].

4.1 11-way Classification

Dataset Setup

Following the settings of Qin et al. (2017), we use two splitting methods of PDTB dataset for comprehensive comparison. The first is PDTB-Lin (Lin et al., 2009), which uses section 2-21, 22 and 23 as training, dev and test sets respectively. The second is PDTB-Ji (Ji and Eisenstein, 2015), which uses section 2-20, 0-1, and 21-22 as training, dev and test sets respectively. According to Ji and Eisenstein (2015), five relation types have few training instances and no dev and test instance. Removing the five types, there remain 11 second level types. During training, instances with more than one annotated relation types are considered as multiple instances, each of which has one of the annotations. At test time, a prediction that matches one of the gold types is considered as correct. All sentences in the dataset are padded or truncated to keep the same 100-word length.

Model Details

For the results of both splitting methods, we share some hyperparameters. Table 1 is some of the shared hyperparameter settings. The pre-trained word embeddings are 300-dim *word2vec* (Mikolov et al., 2013) pre-trained from Google News[4]. So $d_w = 300, d_s = 100, d_c = 300$, then for the final embedding ($\mathbf{e}_i$), $d_e = 700$. For the encoder block in sentence-level module, kernel size is same for every layer. We use AdaGrad optimization (Duchi et al., 2011).

The encoder block layer number is different for the two splitting methods. The layer number for PDTB-Ji splitting method is 4, and the layer number for PDTB-Lin splitting method is 5.

Hyperparameter	Value	Hyperparameter	Value
BPE merge operation num.	$1k$	classifier layer num.	1
subword embedding dim. (d_s)	50	embedding dropout	0.4
subword CNN kernel num.	2	encoder block dropout	0.4
subword CNN kernel size	[2, 3]	classifier dropout	0.3
reduced ELMo embedding dim.	300	learning rate	0.001
encoder block type	Conv.	batch size	64
encoder block kernel size	5		

Table 1: Shared hyperparameter settings. Before dimension reducing, the dimension of pre-trained ELMo embedding is 1024.

Results

Compared to other recent state-of-the-art systems in Table 2, our model achieves new state-of-the-art performance in two splitting methods with great improvements. As to our best knowledge, our model is the first one that exceeds the 48% accuracy in 11-way classification.

[1]The code for this paper is available at `https://github.com/diccooo/Deep_Enhanced_Repr_for_IDRR`

[2]`http://pytorch.org/`

[3]`http://allennlp.org/`

[4]`https://code.google.com/archive/p/word2vec/`

Model	PDTB-Lin	PDTB-Ji
Lin et al. (2009)	40.20	-
Lin et al. (2009)+Brown clusters	-	40.66
Ji and Eisenstein (2015)	-	44.59
Qin et al. (2016a)	43.81	45.04
Qin et al. (2017)	44.65	46.23
Ours	**45.73**	**48.22**

Table 2: Accuracy (%) comparison with others' results on PDTB 2.0 test set for 11-way classification.

Analysis

Ablation Study

To illustrate the effectiveness of our model and the contribution of each module, we use the PTDB-Ji splitting method to do a group of experiments. For the baseline model, we use 4 layer stacked convolutional encoder blocks without the residual connection in the block with only pre-trained word embeddings. We only use the output of the last layer and the output is processed by 2-max pooling without attention and sent to the relation classifier and connective classifier. Without the two residual connections, using 4 layers may be not the best for baseline model but is more convenient to comparison.

Firstly, we add modules from high level to low level accumulatively to observe the performance improvement. Table 3 is the results, which demonstrate that every module has considerable effect on the performance.

Then we test the effects of the two residual connections on the performance. The results are in Table 4. The baseline$^+$ means baseline + bi-attention, i.e., the second row of Table 3. We find that Res 1 (residual connection in the block) is much more useful than Res 2 (residual connection for pair representation), and they work together can bring even better performance.

Level	Model	Performance
-	baseline	41.48
pair	+ bi-attention	42.25
sentence	+ Res	46.29
word	+ subword	47.03
word	+ ELMo	48.22

Table 3: Accumulatively performance test result.

Model	Performance
baseline$^+$	42.25
baseline$^+$ + Res 1	45.33
baseline$^+$ + Res 2	43.57
baseline$^+$ + Res 1 + Res 2	46.29

Table 4: The effects of residual connections.

Without ELMo (the same setting as 4-th row in Table 3), our data settings is the same as Qin et al. (2017) whose performance was state-of-the-art and will be compared directly. We see that even without the pre-trained ELMo encoder, our performance is better, which is mostly attributed to our better sentence pair representations.

Subword-Level Embedding For the usefulness of subword-level embedding, we compare its performance to a model with character-level embedding, which was ever used in Qin et al. (2016a). We use the same model setting as the 4-th row of Table 3, and then replace subword with character sequence. The subword embedding augmented result is **47.03%**, while the character embedding result is **46.37%**, which verifies that the former is a better input representation for the task.

Parameters for Sentence-Level Module As previously discussed, argument specific parameter settings may result in better sentence-level encoders. We use the model which is the same as the third row in Table 3. If shared parameters are used, the result is **45.97%**, which is lower than argument specific parameter settings (**46.29%**). The comparison shows argument specific parameter settings indeed capture the difference of argument representations and facilitate the sentence pair representation.

Encoder Block Type and Layer Number In section 3.3, we consider two encoder types, here we compare their effects on the model performance. Like the previous part, The model setting is also the same as the third row in Table 3 except for the block type and layer number. The results are shown in

Figure 5.

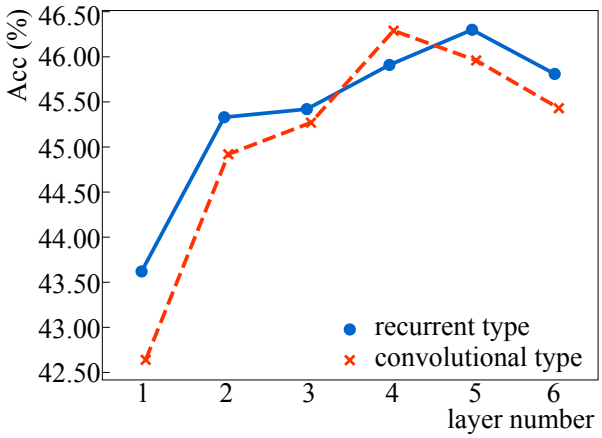


Figure 5: Effects of block type and layer number.

The results in the figure show that both types may reach similar level of top accuracies, as the order of word is not important to the task. We also try to add position information to the convolutional type encoder, and receive a dropped accuracy. This further verifies the order information does not matter too much for the task. For most of the other numbers of layers, the recurrent type shows better, as the number of layers has an impact on the window size of convolutional encoders. When convolutional type is used, the training procedure is much faster, but choosing the suitable kernel size needs extra efforts.

Bi-Attention

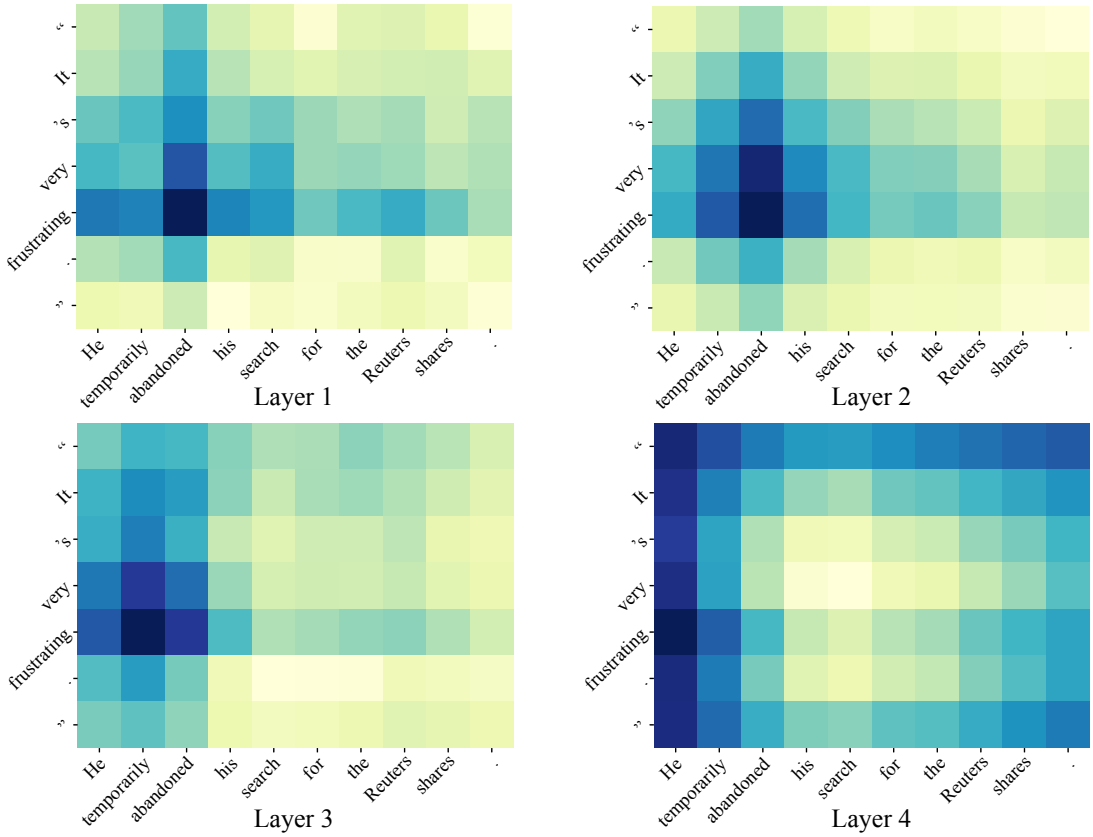


Figure 6: Attention visualization.

We visualize the attention weight of one instance in Figure 6. For lower layers, the attended part is more concentrated. For higher layers, the weights are more average and the attended part moves to the

sentence border. This is because the window size is bigger for higher layers, and the convolutional kernel may have higher weights on words at the window edge.

4.2 Binary and 4-way Classification

Settings For the first level classification, we perform both 4-way classification and one-vs-others binary classification. Following the settings of previous works, the dataset splitting method is the same as PDTB-Ji without removing instances. The model uses 5 block layers with kernel size 3, other details are the same as that for 11-way classification on PDTB-Ji.

Results Table 5 is the result comparison on first level classification. For binary classification, the result is computed by F_1 score (%), and for 4-way classification, the result is computed by macro average F_1 score (%). Our model gives the state-of-the-art performance for 4-way classification by providing an F_1 score greater than 50% for the first time according to our best knowledge.

Model	Comp.	Cont.	Exp.	Temp.	4-way
Zhang et al. (2015)	33.22	52.04	69.59	30.54	-
Ji and Eisenstein (2015)	35.93	52.78	-	27.63	-
Chen et al. (2016)	40.17	54.76	-	31.32	-
Qin et al. (2016c)	41.55	57.32	71.50	35.43	-
Liu and Li (2016)	39.86	54.48	70.43	**38.84**	46.29
Qin et al. (2017)	40.87	54.56	72.38	36.20	-
Lan et al. (2017)	40.73	**58.96**	72.47	38.50	47.80
Lei et al. (2018)	43.24	57.82	**72.88**	29.10	47.15
Ours	**47.85**	54.47	70.60	36.87	**51.06**

Table 5: F_1 score (%) comparison on binary and 4-way classification.

5 Conclusion

In this paper, we propose a deeper neural model augmented by different grained text representations for implicit discourse relation recognition. These different module levels work together and produce task-related representations of the sentence pair. Our experiments show that the model is effective and achieve the state-of-the-art performance. As to our best knowledge, this is the first time that an implicit discourse relation classifier gives an accuracy higher than 48% for 11-way and an F_1 score higher than 50% for 4-way classification tasks.

References

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning: Theory and Applications*, pages 137–186. Berlin, Heidelberg.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 632–642, Lisbon, Portugal, September.

Chloé Braud and Pascal Denis. 2015. Comparing word representations for implicit discourse relation classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2201–2211, Lisbon, Portugal, September.

Deng Cai and Hai Zhao. 2016. Neural Word Segmentation Learning for Chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 409–420, Berlin, Germany, August.

Deng Cai and Hai Zhao. 2017. Pair-aware neural sentence modeling for implicit discourse relation classification. In *Advances in Artificial Intelligence: From Theory to Practice*, pages 458–466, Cham.

Changge Chen, Peilu Wang, and Hai Zhao. 2015. Shallow discourse parsing using constituent parsing tree. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning (CoNLL) - Shared Task*, pages 37–41, Beijing, China, July.

Jifan Chen, Qi Zhang, Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Implicit discourse relation detection via a deep architecture with gated relevance network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers)*, pages 1726–1735, Berlin, Germany, August.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680, Copenhagen, Denmark, September.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2017. AllenNLP: A deep semantic natural language processing platform.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, USA.

Shexia He, Zuchao Li, Hai Zhao, Hongxiao Bai, and Gongshen Liu. 2018. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers)*, Melbourne, Australia, July.

Peter Jansen, Mihai Surdeanu, and Peter Clark. 2014. Discourse complements lexical semantics for non-factoid answer reranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 977–986, Baltimore, Maryland, June.

Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-augmented distributed semantics for discourse relations. *Transactions of the Association for Computational Linguistics (TACL)*, 3:329–344.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse-driven language models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL:HLT)*, pages 332–342, San Diego, California, June.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, pages 2741–2749, Phoenix, USA.

Man Lan, Jianxiang Wang, Yuanbin Wu, Zheng-Yu Niu, and Haifeng Wang. 2017. Multi-task attention-based neural networks for implicit discourse relationship representation and identification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1299–1308, Copenhagen, Denmark, September.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL:HLT)*, New Orleans, Louisiana.

Wenqiang Lei, Yuanxin Xiang, Yuwei Wang, Qian Zhong, Meichun Liu, and Min-Yen Kan. 2018. Linguistic properties matter for implicit discourse relation recognition: Combining semantic interaction, topic continuity and attribution. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, New Orleans, USA.

Junyi Jessy Li, Marine Carpuat, and Ani Nenkova. 2014. Assessing the discourse factors that influence the quality of machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 2: Short Papers)*, pages 283–288, Baltimore, Maryland, June.

Zhongyi Li, Hai Zhao, Chenxi Pang, Lili Wang, and Huan Wang. 2016. A constituent syntactic parse tree based discourse parser. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning (CoNLL) - Shared Task*, pages 60–64, Berlin, Germany, August.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the Penn Discourse Treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 343–351, Singapore, August.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *International Conference on Learning Representations (ICLR)*.

Yang Liu and Sujian Li. 2016. Recognizing implicit discourse relations via repeated reading: Neural networks with multi-level attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1224–1233, Austin, Texas, November.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *Spoken Language Technology (SLT) Workshop*, pages 234–239.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL:HLT)*, New Orleans, Louisiana.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of he Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 683–691, Suntec, Singapore, August.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of the Sixth conference on International Language Resources and Evaluation (LREC-2008)*, pages 2961–2968, Marrakech, Morocco, May.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016a. Implicit discourse relation recognition with context-aware character-enhanced embeddings. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING): Technical Papers*, pages 1914–1924.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016b. Shallow discourse parsing using convolutional neural network. In *Proceedings of the Twentieth Conference on Computational Natural Language Learning (CoNLL) - Shared Task*, pages 70–77, Berlin, Germany, August.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016c. A stacking gated neural architecture for implicit discourse relation classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2263–2270, Austin, Texas, November.

Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric Xing. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers)*, pages 1006–1017, Vancouver, Canada, July.

Samuel Rönnqvist, Niko Schenk, and Christian Chiarcos. 2017. A recurrent neural model with attention for the recognition of Chinese implicit discourse relations. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 2: Short Papers)*, pages 256–262, Vancouver, Canada, July.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. *arXiv preprint arXiv:1507.06228*.

Will Williams, Niranjani Prasad, David Mrva, Tom Ash, and Tony Robinson. 2015. Scaling recurrent neural network language models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5391–5395, South Brisbane, Queensland, Australia.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Changxing Wu, Xiaodong Shi, Yidong Chen, Yanzhou Huang, and Jinsong Su. 2016. Bilingually-constrained synthetic data for implicit discourse relation recognition. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2306–2312, Austin, Texas, November.

Changxing Wu, Xiaodong Shi, Yidong Chen, Jinsong Su, and Boli Wang. 2017. Improving implicit discourse relation recognition with discourse-specific word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 2: Short Papers)*, pages 269–274, Vancouver, Canada, July.

Biao Zhang, Jinsong Su, Deyi Xiong, Yaojie Lu, Hong Duan, and Junfeng Yao. 2015. Shallow convolutional neural network for implicit discourse relation recognition. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2230–2235, Lisbon, Portugal, September.

Zhisong Zhang, Hai Zhao, and Lianhui Qin. 2016. Probabilistic graph-based dependency parsing with convolutional neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1382–1392, Berlin, Germany, August.

Zhi-Min Zhou, Yu Xu, Zheng-Yu Niu, Man Lan, Jian Su, and Chew Lim Tan. 2010. Predicting discourse connectives for implicit discourse relation recognition. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, pages 1507–1514, Beijing, China, August.

A Knowledge-Augmented Neural Network Model for Implicit Discourse Relation Classification

Yudai Kishimoto **Yugo Murawaki** **Sadao Kurohashi**
Graduate School of Informatics, Kyoto University
Yoshida-honmachi, Sakyo-ku, Kyoto, 606-8501, Japan
{kishimoto, murawaki, kuro}@nlp.ist.i.kyoto-u.ac.jp

Abstract

Identifying discourse relations that are not overtly marked with discourse connectives remains a challenging problem. The absence of explicit clues indicates a need for the combination of world knowledge and weak contextual clues, which can hardly be learned from a small amount of manually annotated data. In this paper, we address this problem by augmenting the input text with external knowledge and context and by adopting a neural network model that can effectively handle the augmented text. Experiments show that external knowledge did improve the classification accuracy. On the other hand, contextual information provided no significant gain for implicit discourse relations while it worked for explicit ones.

1 Introduction

Discourse relation recognition has a wide variety of potential applications including summarization (Louis et al., 2010), sentiment analysis (Somasundaran et al., 2009) and machine translation (Meyer et al., 2015). In one of the two most prevalent discourse treebanks, the Penn Discourse TreeBank (PDTB) (Prasad et al., 2008), discourse relations are conventionally divided into two types: explicit and implicit. Explicit relations are overtly marked with discourse connectives such as "because" and "however." Because of these strong cues, explicit relations are relatively easy to classify (Pitler et al., 2008; Xue et al., 2016). By contrast, implicit relations lack discourse connectives and classifying such relations remains a challenging problem.

Recent studies on implicit discourse relation classification have shown success in applying various neural network models including feedforward networks (Zhang et al., 2015; Schenk et al., 2016), convolutional neural networks (Mihaylov and Frank, 2016; Wang and Lan, 2016) and bidirectional LSTM (bi-LSTM) (Chen et al., 2016; Liu and Li, 2016; Dai and Huang, 2018). Although these studies on network engineering report performance improvement, Rutherford et al. (2017) demonstrated that a simple feed-forward neural network was astonishingly competitive, outperforming LSTM- and Tree LSTM-based models. He claimed that training data for implicit discourse relation classification were too small to train powerful neural networks like LSTM. This motivates us to view this task from a different perspective.

We argue that the neural network models need to be provided with world knowledge, which can hardly be learned from a small amount of manually annotated data. To see this, suppose that we want to classify the discourse relation of the following pair of text spans (referred to as `Arg1` (in *italic*) and `Arg2` (in **bold**) throughout this paper):

> `Arg1`: *Not counting the extraordinary charge it would have had a net loss of $3.1 million, or seven cents a share*
> `Arg2`: **A year earlier, it had profit of $7.5 million, or 18 cents a share**
> (Comparison.Contrast)

We can easily recognize that the discourse relation between this pair is Comparison.Contrast partly because we know that "loss" in `Arg1` and "profit" in `Arg2` are antonyms. However, it is difficult for a

Proceedings of the 27th International Conference on Computational Linguistics, pages 584–595
Santa Fe, New Mexico, USA, August 20-26, 2018.

neural network model trained on small training data to recognize the antonymy. Although many recent studies make use of word2vec word embeddings (Mikolov et al., 2013), which are typically trained with a large amount of unannotated data, it is well known that synonyms and antonyms are distributionally similar and thus are hardly distinguishable (Ono et al., 2015). For this reason, we need to look for different knowledge sources as well as an efficient way to integrate them into neural network models.

In this paper, we use MAGE-GRU (Dhingra et al., 2017) to encode external knowledge. It is a straightforward extension to the Gated Recurrent Unit (GRU) (Cho et al., 2014). The input to MAGE-GRU is no longer a sequence of words but a directed acyclic graph in which the word sequence is augmented with edges between arbitrarily distant words for which external knowledge suggests some explicit signals such as antonymy. Thus "loss" and "profit" in the example above are directly connected, making a downstream network layer more easily classify the discourse relation.

In the experiments, we augmented the inputs with ConceptNet (Speer and Havasi, 2012) and coreference resolution. We found that MAGE-GRU significantly outperformed others when ConceptNet was used.

While recurrent neural networks have considerable difficulty in capturing long-range dependencies, MAGE-GRU is expected to mitigate this problem because it creates shortcuts within word sequences. This leads us to explore another question: Do `Arg1` and `Arg2` provide sufficient information to determine discourse relations?

Let us consider the following example:

> Good service programs require recruitment, screening, training and supervision – all of high quality. They involve stipends to participants. Full-time residential programs also require housing and full-time supervision; *they are particularly expensive – more per participant than a year at Stanford or Yale*. **Non-residential programs are cheaper, but good ones still come to some $10,000 a year** (Comparison.Contrast)

In this example, "Non-residential programs" in `Arg2` is contrasted with "they" in `Arg1`. To recognize the fact, we need to resolve the pronoun "they" by looking back at the text preceding `Arg1` to find the antecedent "Full-time residential programs." Although other clues such as the pair of "expensive" and "cheaper" are present, contextual information makes it easier to classify this argument pair as Comparison.Contrast. Although the current standard approach to this task is to use the pair of `Arg1` and `Arg2` out of context, the computer might also benefit from the wider context, given the power of MAGE-GRU.

We compared the performance of MAGE-GRU with and without the text chunk that preceded a given argument pair in the paragraph. It turned out that contextual information provided no significant improvement for implicit discourse relations. However, we also found that contextual information yielded a significant gain for explicit discourse relations. The results appear to strengthen the observation that explicit and implicit discourse relations are dissimilar (Prasad et al., 2014).

2 Related Work

Before neural networks were introduced to the task of implicit discourse relation classification, Lin et al. (2009) proposed a linear classifier that was based on various lexical and syntactic features and was combined with extensive feature selection. Rutherford et al. (2017) built a simple feedforward neural network model where only one pooling layer and one hidden layer were stacked on top of word embeddings. They reported that the simple model outperformed LSTM- and Tree LSTM-based models but lost to Lin et al. (2009).

LSTM has demonstrated success in a wide range of NLP tasks including implicit discourse relation classification. Chen et al. (2016) proposed a combination of a bi-LSTM and a gated relevance network while Liu and Li (2016) combined a bi-LSTM with multi-level attentions. Dai and Huang (2018) proposed a combination of a word-level bi-LSTM and a paragraph-level neural networks. Rutherford et al. (2017) argued that the annotated corpus was too small to train LSTM-based models, however. Note that the present work is complementary to these studies. Since our model is a straightforward extension to

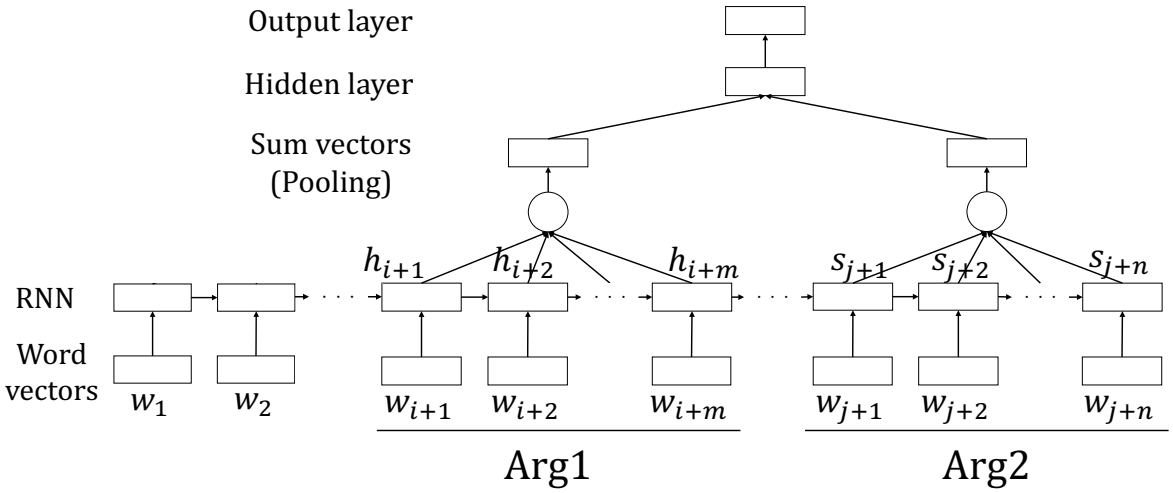


Figure 1: System architecture.

the recurrent neural network, it can easily be combined with other neural network models that are built on top of LSTMs.

Another popular choice of network is convolutional neural networks (CNN). Wang and Lan (2016) proposed an end-to-end shallow discourse parser. In their pipeline system, a CNN is used for non-explicit relation classification.[1] It ranked second on the blind datasets in the CoNLL 2016 Shared Task. Qin et al. (2016) proposed a combination of a bi-LSTM and CNNs. They constructed character-based word representations by transforming character embeddings with CNN and bi-LSTM layers. Another CNN layer was used to extract an argument representation from a sequence of words.

Some recent studies exploited external knowledge sources and some of them were reported to improve the performance of implicit discourse relation classification (Rutherford and Xue, 2014). In the closed track of the CoNLL 2016 Shared Task (Xue et al., 2016), the organizers allowed participants to use a limited set of linguistic resources: Brown Clusters, VerbNet, a sentiment lexicon and an off-the-shelf word2vec model. In addition to these, Inquirer Tags, Levin classes and Modality were tested by Shi and Demberg (2017). They extracted features from these resources and added them to a neural network layer just before the output. Dhingra et al. (2017), who proposed MAGE-GRU, tested a more direct approach as a baseline, in which they appended to word embeddings a sequence of features which are fired by external knowledge. They found that MAGE-GRU consistently outperformed the baseline in various tasks when coreference resolution is used as external knowledge.

A huge performance gap between explicit and implicit relations leads some to transform explicit relations in unannotated corpora to generate pseudo-training data of implicit relations. Sporleder and Lascarides (2008) reported negative results, suggesting that explicit and implicit discourse relations were linguistically dissimilar. Rutherford and Xue (2015) worked on selecting discourse connectives that can be safely dropped. Qin et al. (2017) generated a different kind of pseudo-training data. They inserted to implicit relations *implicit connectives* PDTB annotators assigned to them. They used domain adversarial training to transfer knowledge from the recognition model supplied with implicit connectives to the model without connectives. These methods can be combined with our approach.

3 Proposed Method

The overall system architecture is shown in Figure 1. It is based on the RNN architecture of Rutherford et al. (2017) although we made two major modifications to it, which will be described in Sections 3.1 and 3.3. For each of `Arg1` and `Arg2`, a sequence of words are transformed into a sequence of hidden

[1]Non-explicit relation classification is closely related to implicit discourse relation classification but differs in the treatment of another type of relation, AltLex.

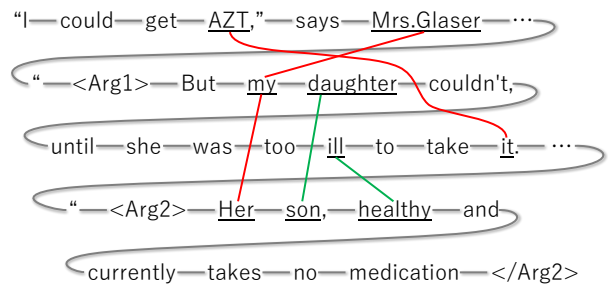


Figure 2: Sequence augmented with extra edges. Words in the sequence are connected by gray edges. In addition, pairs of words are connected by red edges if they are coreferential. Similarly, green edges denote antonymy.

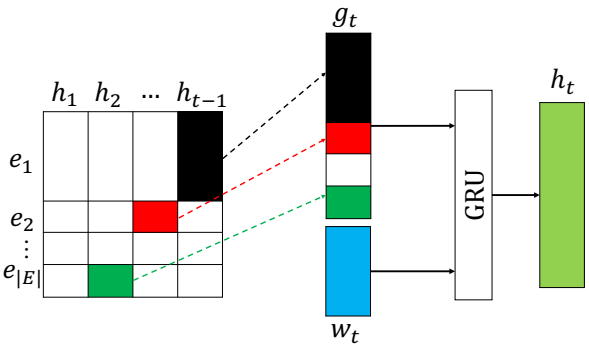


Figure 3: One step of MAGE-GRU.

vectors by an RNN. It is followed by a pooling operation, the concatenation of the two arguments, a linear transformation with non-linear activation, and a softmax classification to predict the relation. The linear transformation is designed to capture the interaction between the arguments.

3.1 MAGE-GRU

One major difference from the architecture of Rutherford et al. (2017) is that whereas Rutherford et al. (2017) used LSTM as the RNN component, we adopt MAGE-GRU (Dhingra et al., 2017). It extends the Gated Recurrent Unit (GRU) (Cho et al., 2014) such that it can incorporate external linguistic knowledge as explicit signals.

The input to MAGE-GRU is no longer a sequence of words but a directed acyclic graph as shown in Figure 2. The graph is constructed by augmenting the word sequence with edges between arbitrarily distant words. Sequential transitions are now treated as a special type of edges. An edge is added if external knowledge suggests an explicit signal for the given word pair. We represent a signal as a triplet (*word A*, *relation type*, *word B*). For example, green edges in Figure 2 denote antonymy: (daughter, `Antonym`, son) and (ill, `Antonym`, healthy). Edges are directed and we assume that *word A* always comes before *word B*. If relation types in external knowledge are asymmetric (e.g., (thinking, `Causes`, acting)), we create reverse relation types to keep the directionality. If external knowledge deals with a phrase, we use the last word for the triplet. We also constrain every node to have at most one incoming edge per relation type because this drastically simplifies computation. If *word B* has two or more candidates for *word A*, the nearest one is chosen.

Given a directed acyclic graph, MAGE-GRU outputs a hidden vector h_t at each time step t. One step of MAGE-GRU is illustrated in Figure 3. The peculiarity of MAGE-GRU is that whereas vanilla GRU receives word vector w_t and hidden vector h_{t-1} as the inputs, MAGE-GRU substitutes h_{t-1} with a special vector g_t. g_t is created by partially and selectively combining the history of hidden vectors, $h_1, \cdots, h_{t-1}$:

$$g_t = [g_t^{e_1}; g_t^{e_2}; \cdots; g_t^{e_{|E|}}]$$
$$g_t^{e_i} = \begin{cases} h_{t'}^{e_i} & \text{if } (x_{t'}, e_i, x_t) \text{ exists} \\ \mathbf{0} & \text{otherwise} \end{cases}$$

where ; denotes vector concatenation. $h_{t'}$ is decomposed into $[h_{t'}^{e_1}; h_{t'}^{e_2}; \cdots; h_{t'}^{e_{|E|}}]$, where e_i $(1 \leq i \leq |E|)$ is a relation type. Thus $h_{t'}^{e_i}$ is a subvector of $h_{t'}$ corresponding to relation type e_i. In Figure 2, for example, when w_t is "healthy", g_t consists of $h_{\text{son}}^{\text{sequence}}$ due to the gray edge and $h_{\text{ill}}^{\text{antonym}}$ due to the green edge and zero vectors for the rest. Note that e_1 is reserved for ordinary sequential transitions.

3.2 External Knowledge

We use ConceptNet (Speer and Havasi, 2012) and coreference resolution as external knowledge. ConceptNet is an open-source project for building in a large linguistic knowledge base. It was constructed through various means including handcrafting (Open Mind Common Sense), automatic extraction from web pages such as Wikipedia, and gamification. ConceptNet covers not only lexical definitions but also common sense. Some relations in ConceptNet appear to be particularly useful for implicit discourse relation classification. For example, the relation `Causes` is closely related to the discourse relation Contingency.Cause. We expect the system to classify such discourse relations more easily when the input is augmented with ConceptNet.

We also use coreference resolution as external knowledge. As we discussed in Section 1, coreference resolution is a straightforward approach to connecting arguments to contextual information. Coreference resolution was also used by Dhingra et al. (2017) in their experiments.

3.3 Input Formats

Another major modification we make to the architecture of Rutherford et al. (2017) is about the formats of input sequences. While Rutherford et al. (2017) among others treated `Arg1` and `Arg2` separately, we combine them into a single sequence. Additionally, we append to the sequence a text chunk that precedes the arguments in the paragraph.[2]

We insert special tags, <Arg1>, </Arg1>, <Arg2>, and </Arg2>, into a given sequence to indicate where the arguments start and end. A similar technique is used in neural machine translation (Johnson et al., 2016). Although the PDTB annotates implicit relations on adjacent sentences, an argument pair does not necessarily form a complete span mainly because sub-sentential spans are sometimes selected as arguments (The PDTB Research Group, 2007). In addition, arguments sometimes contain non-argument text spans as in the following example (underlined):

> *At Quantum*, which is based in New York, *the trouble is magnified by the company's heavy dependence on plastics*. Once known as National Distillers & Chemical Corp., **the company exited the wine and spirits business and plowed more of its resources into plastics after Mr. Stookey took the chief executive's job in 1986**.
> (Contingency.Cause)

To cope with this problem, we enclose a non-argument span with <Skip> and </Skip>.

4 Experiments

4.1 Setup

4.1.1 Penn Discourse TreeBank

We evaluated our model's performance on the Penn Discourse TreeBank (PDTB) (Prasad et al., 2008). It is the most popular and largest corpus of discourse relations in English. The annotation is done as another layer on Wall Street Journal sections of the Penn Treebank. Each discourse relation consists of two text spans (arguments) and a relation label. Arguments are annotated such that they are minimally required to infer the discourse relation.

Relation labels are organized as a 3-level hierarchy in the PDTB. However, it is too difficult for current systems to perform classification in its original form, and previous studies have used more coarse-grained relation labels. Popular settings include top-level one-versus-all binary classification (Pitler et al., 2009), top-level 4-way classification (Pitler et al., 2009; Ji and Eisenstein, 2015), second-level 11-way classification (Lin et al., 2009; Rutherford et al., 2017), and modified second-level classification for the CoNLL 2015 Shared Task (Xue et al., 2015). We used second-level 11-way classification in the experiments.

Following Shi and Demberg (2017), we conducted 10-fold cross validation using the whole corpus of sections 0–24 (referred to as Cross Validation). The standard approach (referred to as Most-used Split)

[2] Rönnqvist et al. (2017) also concatenated `Arg1` and `Arg2` in the task of Chinese implicit discourse relation classification. They did not incorporate contextual information, however.

Sense	Train	Dev	Test
Comparison.Concession	179	19	21
Comparison.Contrast	1,672	185	206
Contingency.Cause	3,332	370	411
Contingency.Pragmatic cause	57	6	6
Expansion.Alternative	146	16	18
Expansion.Conjunction	2,787	309	344
Expansion.Instantiation	1,131	125	139
Expansion.List	314	34	38
Expansion.Restatement	2,519	279	310
Temporal.Asynchronous	527	58	65
Temporal.Synchrony	143	15	17
Total	12,807	1,416	1,575

Table 1: The distribution of relation labels in the Cross Validation dataset.

Sense	Train	Dev	Test
Comparison.Concession	192	5	5
Comparison.Contrast	1,612	82	127
Contingency.Cause	3,376	120	197
Contingency.Pragmatic cause	56	2	5
Expansion.Alternative	153	2	15
Expansion.Conjunction	2,890	115	116
Expansion.Instantiation	1,132	47	69
Expansion.List	337	5	25
Expansion.Restatement	2,486	101	190
Temporal.Asynchronous	543	28	12
Temporal.Synchrony	153	8	5
Total	12,930	515	766

Table 2: The distribution of relation labels in the Most-used Split dataset.

is to use sections 2–21 for the training set, section 22 for the development set and section 23 for the test set (Lin et al., 2009; Rutherford et al., 2017). However, Shi and Demberg (2017) argued that the standard test set was too small for a reliable evaluation especially when second-level classification was employed.

Table 1 shows the distribution of relation labels in the Cross Validation dataset. Note that although we tried to replicate the procedures described by Shi and Demberg (2017) as closely as possible, there remained slight differences in the discourse relation distribution. For comparison, we also trained the model on the Most-used Split dataset. Table 2 shows the relation label distribution in this dataset. We can confirm that the test set distribution diverged from the development set distribution.

4.1.2 Model Configurations

As word embeddings, we used an off-the-shelf word2vec model specified by the CoNLL 2016 Shared Task organizers. Word embeddings were fixed during training except for some words such as special tags <Arg1> and </Skip>.

As we described in Section 3.2, we used `ConceptNet` and coreference resolution (`Coref`) as external knowledge. Note that if none of the external knowledge is used, MAGE-GRU is reduced to vanilla GRU. For ConceptNet, we removed some triplets that contained stop words. We selected all relation types that appeared in the PDTB except `RelatedTo`. The number of relation types is 35. We added reverse relation types for 30 asymmetric relation types (e.g. `AtLocation` and `Causes`). As a result, the number of ConceptNet relation types used in the experiments was increased to 65.

Although ConceptNet was a relatively high-quality and high-coverage knowledge base, it nevertheless (1) contained questionable triplets (e.g., (time, `Antonym`, year)) and (2) failed to cover some important relations (stock, `AtLocation`, market). We mitigated the first problem by checking weights ConceptNet assigned to triplets. We removed triplets whose weight was smaller than 1.0. The second problem might potentially be addressed graph embedding techniques (Xie et al., 2017), but in the experiments, we used a simpler method. For each word in the input, we prepared the top-10 nearest neighbors in terms of cosine similarity of word2vec vectors with the threshold value of 0.6. We searched ConceptNet for all combinations of the original words and neighbors. As a result, the average number of edges given to an argument pair increased from 4.0 to 17.6.

As for a coreference resolution system, we used Stanford CoreNLP[3] (ver.3.7.0). CoreNLP had three different coreference systems. We chose a neural model since it performed the best among the three.

We tested two input formats: `Args` and `Paragraph`. In `Args`, an input sequence started with <Arg1> and ended with </Arg2>. In the longer `Paragraph` format, it started at the beginning of the paragraph that contained `Arg1` and `Arg2`.

Table 4 summarizes the model configurations. We found that mini-batch did not work for our model. Given that training set accuracy was as low as development set accuracy, we conjecture that training

[3] `https://stanfordnlp.github.io/CoreNLP/`

Relation	Count
Synonym	37,689
FormOf	32,547
IsA	27,251
DerivedFrom	12,668
Antonym	10,632
SimilarTo	6,560
DistinctFrom	6,484
HasContext	6,092
AtLocation	5,584
UsedFor	3,287

Table 3: Top 10 ConceptNet relation types, sorted by the frequency counts in the Penn Discourse Treebank.

Description	Values
input	`Args` or `Paragraph`
word embeddings	100,000 words (300 dims)
optimizer	AdaGrad (Duchi et al., 2011)
pooling	summation
hidden layer	1 layer (600 dims)
external knowledge	coreference and/or ConceptNet (10 dims per relation type)
mini-batch size	1
early stopping	yes

Table 4: Configuration of our model.

	Dev	Test
Args	0.3946	0.3820 (±0.013)
+Coref	0.3939	0.3817 (±0.009)
+ConceptNet	0.4000	0.3926 (±0.012)
+Coref+ConceptNet	0.4037	0.3926 (±0.016)
Paragraph	0.3951	0.3814 (±0.010)
+Coref	0.3919	0.3839 (±0.016)
+ConceptNet	0.4060	**0.3980 (±0.009)**
+Coref+ConceptNet	**0.4083**	0.3938 (±0.011)
feedforward (Rutherford et al., 2017)†	-	0.3660 (±0.011)
LSTM (Shi and Demberg, 2017)	-	0.3444 (±0.014)
+Modality	-	0.3767 (±0.018)

Table 5: Accuracy in the Cross Validation dataset. Test result indicates the mean accuracy across folds and the standard deviation. † denotes our reimplementation.

signals in a mini-batch might have canceled out each other.

4.1.3 Models for Comparison

For comparison, we collected model scores from the literature. As for the Cross Validation dataset, we compared the proposed model with that of Shi and Demberg (2017) and a feedforward network (Rutherford et al., 2017). Shi and Demberg (2017) used an LSTM-based model, optionally with surface features derived from Brown Clusters, Modality, etc. The best score was achieved by LSTM+Modality. We reimplemented the feedforward model of Rutherford et al. (2017) because their evaluation was not based on the Cross Validation dataset.

For the Most-used Split dataset, the models for comparison were a feedforward network (Rutherford et al., 2017), a maximum entropy classifier (Lin et al., 2009) and a CNN-based model (Qin et al., 2017).

4.2 Results

Table 5 shows the results for the Cross Validation dataset. In all configurations, our model outperformed Shi and Demberg (2017)'s models. The best score was achieved by `Paragraph+ConseptNet`. The performance gain of `Paragraph+ConceptNet` over `Paragraph` was statistically significant ($p = 2.63 \times 10^{-7}, p < 0.01$) while `Paragraph+Coref` was no different from `Paragraph` ($p = 0.463$). The `Paragraph` models consistently outperformed the corresponding `Args` models when the input was augmented with external knowledge. However, the impact of contextual information was not statistically significant ($p = 0.081$ for `+Coref+ConceptNet`).

A breakdown of the performance by discourse relation is shown in `Paragraph+ConceptNet` are

Sense	F_1
Comparison.Concession	0.0000 (±0.000)
Comparison.Contrast	0.2287 (±0.029)
Contingency.Cause	0.4813 (±0.015)
Contingency.Pragmatic cause	0.0000 (±0.000)
Expansion.Alternative	0.0087 (±0.026)
Expansion.Conjunction	0.4466 (±0.023)
Expansion.Instantiation	0.4498 (±0.037)
Expansion.List	0.2203 (±0.091)
Expansion.Restatement	0.3341 (±0.027)
Temporal.Asynchronous	0.2140 (±0.077)
Temporal.Synchrony	0.0000 (±0.000)
Total	0.3980 (±0.009)

Table 6: F_1 score in `Paragraph+ConceptNet`.

	Dev	Test
Args	0.4214	0.3668
+Coref	0.4214	0.3655
+ConceptNet	0.4252	0.3799
+Coref+ConceptNet	0.4233	0.3877
Paragraph	**0.4408**	0.3708
+Coref	0.4330	0.3642
+ConceptNet	0.4350	0.3603
+Coref+ConceptNet	0.4350	0.3655
Rutherford et al. (2017)	-	0.3956
Lin et al. (2009)	-	0.4020
Qin et al. (2017)	-	**0.4465**

Table 7: Accuracy in the Most-used Split dataset.

shown in Table 6. Comparing Table 6 with Table 1, we can see that the model ended up ignoring very-low-frequency relations such as Comparison.Concession and Expansion.Alternative, which appeared less than 300 times in training set. This problem could possibly be mitigated by leveraging an unlabeled corpus to increase the size of training instances (Jiang et al., 2016).

The results for the Most-used Split dataset are shown in Table 7. In this dataset, the proposed method was outperformed by models in the literature. Although the F-measure of `Args+Coref+ConceptNet` was about 2 points higher than that `Args`, the performance varied too inconsistently to draw meaningful conclusions. For this reason, we support Shi and Demberg (2017)'s argument for the need of cross validation for implicit discourse relation classification.

4.3 Discussion

As we have seen in Table 5, ConceptNet brought performance gain. However, it appears to leave much room for improvement. Consider the following examples:

`Ex1`:
Another, "Jeux Sans Frontieres ," where villagers from assorted European countries make fools of themselves performing pointless tasks, *is a hit in France*. **A U.S.-made imitation under the title "Almost Anything Goes" flopped fast**.
(Comparison.Contrast)

`Ex2`:
HOMESTEAD FINANCIAL CORP., Millbrae, financial services concern, annual revenue of $562 million, OTC, said *three of its 17 Bay-area branches were closed yesterday*. **The company expects all branches to reopen today**.
(Comparison.Contrast)

In `Ex1`, a baseline model wrongly chose Expansion.Conjunction but our model seems to have suppressed the discourse relation presumably because it found the antonym pair "hit" (`Arg1`) and "flopped" (`Arg2`) in ConceptNet. However, our model misclassified `Ex2` as Expansion.Conjunction even though "yesterday" and "today" were correctly identified as antonyms. This indicates that our model might not have given due weight to ConceptNet, possibly because of some noise in the knowledge base.

In our experiments, coreference resolution did not help implicit discourse relation classification. What we relied on was standard pronominal and nominal coreference resolution, but the following example suggests the need for resolving event coreference (Lu et al., 2016):

	Test
Args+Coref+ConceptNet	0.7723 ($\pm$0.008)
Paragraph+Coref+ConceptNet	0.7921 ($\pm$0.006)

Table 8: Accuracy of explicit discourse relation classification. Result indicates the mean accuracy across folds and the standard deviation.

Is an American Secretary of State seriously suggesting that the Khmer Rouge should help govern Cambodia? *Apparently so*. **There are no easy choices in Cambodia, but we can't imagine that it benefits the U.S. to become the catalyst for an all-too-familiar process that could end in another round of horror in Cambodia**.
(Comparison.Contrast)

In this example, `Arg1` is surprisingly uninformative. In order to classify the discourse relation between `Arg1` and `Arg2`, the system would need to identify what "so" in `Arg1` refers to.

To explore the effect of contextual information in detail, we compared the two input formats, `Args` and `Paragraph`, in the task of *explicit* discourse relation classification. The experimental settings are basically the same as in Section 4.1.2, but the Cross Validation dataset now contained explicit relations. The result is shown in Table 8. Contextual information did help in explicit discourse relation classification, with statistical significance at $p < 0.01$.

It is hard to see exactly why we obtained different results for implicit and explicit relations, but a hint is given by the PDTB annotation itself. The PDTB limits arguments to the minimal text needed to interpret a given relation, and provides *supplementary information* to each of `Arg1` and `Arg2` (named `Sup1` and `Sup2`) (The PDTB Research Group, 2007). Consider the following examples:

That pattern hasn't always held, *but recent strong growth in dividends makes some market watchers anxious*. Payouts on the S&P 500 stocks rose 10% in 1988, according to Standard & Poor's Corp., and Wall Street estimates for 1989 growth are generally between 9% and 14%. Many people believe the growth in dividends will slow next year, although a minority see double-digit gains continuing. **Meanwhile, many market watchers say recent dividend trends raise another warning flag: While dividends have risen smartly, their expansion hasn't kept pace with even stronger advances in stock prices**
(Expansion.Conjunction)

The underlined text span is annotated with `Sup1`. This tag is assigned to a text span supplementary to `Arg1` if it appears relevant but not necessary for interpretation. According to Prasad et al. (2014), only 126 implicit relations were annotated with supplementary information while 1,571 explicit relations were. They amounted to about 0.8% and 8% of the whole implicit and explicit relations, respectively. This great gap indicates that explicit relation classification may benefit more from the text chunks outside of the arguments than implicit relation classification. In fact, a baseline model wrongly chose Comparison.Contrast in this example, but our model chose a correct discourse relation. It should be noted that according to Prasad et al. (2014), consistency control over supplementary information annotation was rather weak. They warned that the gap could be an accidental feature of the PDTB annotation. However, our results lend support to the hypothesis that the gap reflects an intrinsic feature of the discourse relations, or at least that of the PDTB's task specifications.

5 Conclusion

In this paper, we adopted MAGE-GRU to efficiently incorporate external knowledge into the task of implicit discourse relation classification. The experiments show that external knowledge improved accuracy in this task. In addition to a pair of arguments, the text chunk that preceded the pair in the paragraph was given to the model with the hope that it could help classifying its relation. The contextual information

yielded a significant improvement not for implicit discourse relations but for explicit discourse relations. Additionally, we reconfirmed the need for cross validation in this task, as argued by Shi and Demberg (2017).

In the future, we would like to work on extending the neural network architecture. The high composability of MAGE-GRU means that it can easily be combined with other neural network models that are built on top of RNNs. A bidirectional extension to MAGE-GRU may be worth trying. Another future direction is to look for different sources of external knowledge. The candidates include other knowledge bases (e.g. Freebase (Bollacker et al., 2008)) and results of high-level NLP analyses (e.g. event coreference).

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA.

Jifan Chen, Qi Zhang, Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Implicit discourse relation detection via a deep architecture with gated relevance network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1726–1735.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Zeyu Dai and Ruihong Huang. 2018. Improving implicit discourse relation classification by modeling inter-dependencies of discourse units in a paragraph. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 141–151.

Bhuwan Dhingra, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2017. Linguistic knowledge as memory for recurrent neural networks. *arXiv*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-augmented distributed semantics for discourse relations. *Transactions of the Association of Computational Linguistics*, 3:329–344.

Kailang Jiang, Giuseppe Carenini, and Raymond Ng. 2016. Training data enrichment for infrequent discourse relations. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2603–2614.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *arXiv*.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the Penn Discourse Treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 343–351.

Yang Liu and Sujian Li. 2016. Recognizing implicit discourse relations via repeated reading: Neural networks with multi-level attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1233.

Annie Louis, Aravind Joshi, and Ani Nenkova. 2010. Discourse indicators for content selection in summarization. In *Proceedings of the SIGDIAL 2010 Conference*, pages 147–156.

Jing Lu, Deepak Venugopal, Vibhav Gogate, and Vincent Ng. 2016. Joint inference for event coreference resolution. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3264–3275, December.

Thomas Meyer, Najeh Hajlaoui, and Andrei Popescu-Belis. 2015. Disambiguating discourse connectives for statistical machine translation. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(7):1184–1197.

Todor Mihaylov and Anette Frank. 2016. Discourse relation sense classification using cross-argument semantic similarity based on word embeddings. In *Proceedings of the CoNLL-16 shared task*, pages 100–107.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Masataka Ono, Makoto Miwa, and Yutaka Sasaki. 2015. Word embedding-based antonym detection using thesauri and distributional information. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 984–989.

Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind K. Joshi. 2008. Easily identifiable discourse relations. Technical report, University of Pennsylvania Department of Computer and Information Science.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 683–691.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn discourse treebank 2.0. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC2008)*, pages 2961–2968.

Rashmi Prasad, Bonnie Webber, and Aravind Joshi. 2014. Reflections on the penn discourse treebank, comparable corpora, and complementary annotation. *Computational Linguistics*, 40(4):921–950.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016. Implicit discourse relation recognition with context-aware character-enhanced embeddings. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1914–1924.

Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric Xing. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1006–1017, July.

Samuel Rönnqvist, Niko Schenk, and Christian Chiarcos. 2017. A recurrent neural model with attention for the recognition of chinese implicit discourse relations. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 256–262, July.

Attapol Rutherford and Nianwen Xue. 2014. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 645–654, April.

Attapol Rutherford and Nianwen Xue. 2015. Improving the inference of implicit discourse relations via classifying explicit discourse connectives. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 799–808.

Attapol Rutherford, Vera Demberg, and Nianwen Xue. 2017. A systematic study of neural discourse models for implicit discourse relation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 281–291.

Niko Schenk, Christian Chiarcos, Kathrin Donandt, Samuel Rönnqvist, Evgeny Stepanov, and Giuseppe Riccardi. 2016. Do we really need all those rich linguistic features? a neural network-based approach to implicit sense labeling. In *Proceedings of the CoNLL-16 shared task*, pages 41–49.

Wei Shi and Vera Demberg. 2017. Do we need cross validation for discourse relation classification? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 150–156.

Swapna Somasundaran, Galileo Namata, Janyce Wiebe, and Lise Getoor. 2009. Supervised and unsupervised methods in employing discourse relations for improving opinion polarity classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 170–179.

Robert Speer and Catherine Havasi. 2012. Representing general relational knowledge in conceptnet 5. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 3679–3686.

Caroline Sporleder and Alex Lascarides. 2008. Using automatically labelled examples to classify rhetorical relations: An assessment. *Natural Language Engineering*, 14(3):369–416.

The PDTB Research Group. 2007. The Penn Discourse Treebank 2.0 Annotation Manual. Technical report, Institute for Research in Cognitive Science, University of Pennsylvania.

Jianxiang Wang and Man Lan. 2016. Two end-to-end shallow discourse parsers for English and Chinese in CoNLL-2016 shared task. In *Proceedings of the CoNLL-16 shared task*, pages 33–40.

Qizhe Xie, Xuezhe Ma, Zihang Dai, and Eduard Hovy. 2017. An interpretable knowledge transfer model for knowledge base completion. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 950–962, July.

Nianwen Xue, Tou Hwee Ng, Sameer Pradhan, Rashmi Prasad, Christopher Bryant, and Attapol Rutherford. 2015. The conll-2015 shared task on shallow discourse parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task*, pages 1–16.

Nianwen Xue, Tou Hwee Ng, Sameer Pradhan, Attapol Rutherford, Bonnie Webber, Chuan Wang, and Hongmin Wang. 2016. CoNLL 2016 shared task on multilingual shallow discourse parsing. In *Proceedings of the CoNLL-16 shared task*, pages 1–19.

Biao Zhang, Jinsong Su, Deyi Xiong, Yaojie Lu, Hong Duan, and Junfeng Yao. 2015. Shallow convolutional neural network for implicit discourse relation recognition. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2230–2235.

Modeling Coherence for Neural Machine Translation with Dynamic and Topic Caches

Shaohui Kuang[†] **Deyi Xiong**[†*] **Weihua Luo**[††] **Guodong Zhou**[†]
[†]School of Computer Science and Technology, Soochow University, Suzhou, China
shkuang@stu.suda.edu.cn, {dyxiong, gdzhou}@suda.edu.cn
[††]Alibaba Group
weihua.luowh@alibaba-inc.com

Abstract

Sentences in a well-formed text are connected to each other via various links to form the cohesive structure of the text. Current neural machine translation (NMT) systems translate a text in a conventional sentence-by-sentence fashion, ignoring such cross-sentence links and dependencies. This may lead to generate an incoherent target text for a coherent source text. In order to handle this issue, we propose a cache-based approach to modeling coherence for neural machine translation by capturing contextual information either from recently translated sentences or the entire document. Particularly, we explore two types of caches: a dynamic cache, which stores words from the best translation hypotheses of preceding sentences, and a topic cache, which maintains a set of target-side topical words that are semantically related to the document to be translated. On this basis, we build a new layer to score target words in these two caches with a cache-based neural model. Here the estimated probabilities from the cache-based neural model are combined with NMT probabilities into the final word prediction probabilities via a gating mechanism. Finally, the proposed cache-based neural model is trained jointly with NMT system in an end-to-end manner. Experiments and analysis presented in this paper demonstrate that the proposed cache-based model achieves substantial improvements over several state-of-the-art SMT and NMT baselines.

1 Introduction

Neural machine translation (Sutskever et al., 2014; Bahdanau et al., 2015) as an emerging machine translation approach, quickly achieves the state-of-the-art translation performance on many language pairs, e.g., English-French (Jean et al., 2015; Luong et al., 2015b), English-German (Shen et al., 2015; Luong et al., 2015a) and so on. In principle, NMT is established on an encoder-decoder framework, where the encoder reads a source sentence and encodes it into a fixed-length semantic vector, and the decoder generates a translation according to this vector.

In spite of its current success, NMT translates sentences of a text independently, ignoring document-level information during translation. This largely limits its success since document-level information imposes constraints on the translations of individual sentences of a text. And such document-level constraints, at least, can be categorized into three aspects: consistency, disambiguation and coherence. First, the same phrases or terms should be translated consistently across the entire text as much as possible, no matter in which sentence they occur. If sentences of a text are translated independent of each other, it will be difficult to maintain the translation consistency across different sentences. Second, document-level information provides a global context that can help disambiguate words with multiple senses if sentence-level local context is not sufficient for disambiguation. Third, the topic of a document is able to keep individual sentences translated in a coherent way.

In the literature, such informative constraints have been occasionally investigated in statistical machine translation and achieved certain success via a variety of document-level models, such as cache-based

[*] Corresponding author

Proceedings of the 27th International Conference on Computational Linguistics, pages 596–606
Santa Fe, New Mexico, USA, August 20-26, 2018.

language and translation models (Tiedemann, 2010; Gong et al., 2011; Nepveu et al., 2004) for the consistency constraint, topic-based coherence model (Xiong and Zhang, 2013; Tam et al., 2007) for the coherence constraint. By contrast, in neural machine translation, to the best of our knowledge, such constraints have not been explored so far.

Partially inspired by the success of cache models in SMT, we propose a cache-based approach to capturing coherence for neural machine translation. Particularly, we incorporate two types of caches into NMT: a static topic cache and a dynamic cache being updated on the fly. For the topic cache, we first use a projection-based bilingual topic learning approach to infer the topic distribution for each document to be translated and obtain the corresponding topical words on the target side. These topical words are then integrated into the topic cache. For the dynamic cache, words are retrieved from the best translation hypotheses of recently translated sentences. While the topic cache builds a global profile for a document, which helps impose the coherence constraint on document translation, the dynamic cache follows an intuition that words occurred previously should have higher probabilities of recurrence even if they are rare words in the training data.

In order to integrate these two caches into neural machine translation, we further propose a cache-based neural model, which adds a new neural network layer on the cache component to score each word in the cache. During decoding, we estimate the probability of a word from the cache according to its score and combine this cache probability with the original probability computed by the decoder via a gating mechanism to obtain the final word prediction probability.

On the NIST Chinese-English translation tasks, our experiment results show that the proposed cache-based neural approach can achieve significant improvements of up to 1.60 BLEU points on average (up to 2.53 BLEU points on NIST04) over the state-of-the-art attention-based NMT baseline.

2 Related Work

In the literature, several cache-based translation models have been proposed for conventional statistical machine translation, besides traditional n-gram language models and neural language models. In this section, we will first introduce related work in cache-based language models and then in translation models.

For traditional n-gram language models, Kuhn and De Mori (1990) propose a cache-based language model, which mixes a large global language model with a small local model estimated from recent items in the history of the input stream for speech recongnition. Della Pietra et al. (1992) introduce a MaxEnt-based cache model by integrating a cache into a smoothed trigram language model, reporting reduction in both perplexity and word error rates. Chueh and Chien (2010) present a new topic cache model for speech recongnition based on latent Dirichlet language model by incorporating a large-span topic cache into the generation of topic mixtures.

For neural language models, Huang et al. (2014) propose a cache-based RNN inference scheme, which avoids repeated computation of identical LM calls and caches previously computed scores and useful intermediate results and thus reduce the computational expense of RNNLM. Grave et al. (2016) extend the neural network language model with a neural cache model, which stores recent hidden activations to be used as contextual representations. Our caches significantly differ from these two caches in that we store linguistic items in the cache rather than scores or activations.

For neural machine translation, Wang et al. (2017) propose a cross-sentence context-aware approach and employ a hierarchy of Recurrent Neural Networks (RNNs) to summarize the cross-sentence context from source-side previous sentences. Jean et al. (2017) propose a novel larger-context neural machine translation model based on the recent works on larger-context language modelling (Wang and Cho, 2016) and employ the method to model the surrounding text in addition to the source sentence.

For cache-based translation models, Nepveu et al. (2004) propose a dynamic adaptive translation model using cache-based implementation for interactive machine translation, and develop a monolingual dynamic adaptive model and a bilingual dynamic adaptive model. Tiedemann (2010) propose a cache-based translation model, filling the cache with bilingual phrase pairs from the best translation hypotheses of previous sentences in a document. Gong et al. (2011) further propose a cache-based approach

to document-level translation, which includes three caches, a dynamic cache, a static cache and a topic cache, to capture various document-level information. Bertoldi et al. (2013) describe a cache mechanism to implement online learning in phrase-based SMT and use a repetition rate measure to predict the utility of cached items expected to be useful for the current translation.

Our caches are similar to those used by Gong et al. (2011) who incorporate these caches into statistical machine translation. We adapt them to neural machine translation with a neural cache model. It is worthwhile to emphasize that such adaptation is nontrivial as shown below because the two translation philosophies and frameworks are significantly different.

3 Attention-based NMT

In this section, we briefly describe the NMT model taken as a baseline. Without loss of generality, we adopt the NMT architecture proposed by Bahdanau et al. (2015), with an encoder-decoder neural network.

3.1 Encoder

The encoder uses bidirectional recurrent neural networks (Bi-RNN) to encode a source sentence with a forward and a backward RNN. The forward RNN takes as input a source sentence $x = (x_1, x_2, ..., x_T)$ from left to right and outputs a hidden state sequence $(\overrightarrow{h_1}, \overrightarrow{h_2}, ..., \overrightarrow{h_T})$ while the backward RNN reads the sentence in an inverse direction and outputs a backward hidden state sequence $(\overleftarrow{h_1}, \overleftarrow{h_2}, ..., \overleftarrow{h_T})$. The context-dependent word representations of the source sentence h_j (also known as word annotation vectors) are the concatenation of hidden states $\overrightarrow{h_j}$ and $\overleftarrow{h_j}$ in the two directions.

3.2 Decoder

The decoder is an RNN that predicts target words y_t via a multi-layer perceptron (MLP) neural network. The prediction is based on the decoder RNN hidden state s_t, the previous predicted word y_{t-1} and a source-side context vector c_t. The hidden state s_t of the decoder at time t and the conditional probability of the next word y_t are computed as follows:

$$s_t = f(s_{t-1}, y_{t-1}, c_t) \tag{1}$$

$$p(y_t|y_{<t}; x) = g(y_{t-1}, s_t, c_t) \tag{2}$$

3.3 Attention Model

In the attention model, the context vector c_t is calculated as a weighted sum over source annotation vectors $(h_1, h_2, ..., h_T)$:

$$c_t = \sum_{j=1}^{T_x} \alpha_{tj} h_j \tag{3}$$

$$\alpha_{tj} = \frac{exp(e_{tj})}{\sum_{k=1}^{T} exp(e_{tk})} \tag{4}$$

$$e_{tj} = a(s_{t-1}, h_j) \tag{5}$$

where α_{tj} is the attention weight of each hidden state h_j computed by the attention model, and a is a feed forward neural network with a single hidden layer.

The dl4mt tutorial[1] presents an improved implementation of the attention-based NMT system, which feeds the previous word y_{t-1} to the attention model. We use the dl4mt tutorial implementation as our baseline, which we will refer to as RNNSearch*.

The proposed cache-based neural approach is implemented on the top of RNNSearch* system, where the encoder-decoder NMT framework is trained to optimize the sum of the conditional log probabilities of correct translations of all source sentences on a parallel corpus as normal.

[1] https://github.com/nyu-dl/dl4mt-tutorial/tree/master/session2

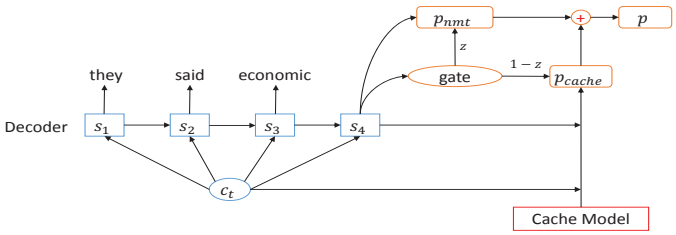


Figure 1: Architecture of NMT with the neural cache model. P_{cache} is the probability for a next target word estimated by the cache-based neural model.

4 The Cache-based Neural Model

In this section, we elaborate on the proposed cache-based neural model and how we integrate it into neural machine translation, Figure 1 shows the entire architecture of our NMT with the cache-based neural model.

4.1 Dynamic Cache and Topic Cache

The aim of cache is to incorporate document-level constraints and therefore to improve the consistency and coherence of document translations. In this section, we introduce our proposed dynamic cache and topic cache in detail.

4.1.1 Dynamic Cache

In order to build the dynamic cache, we dynamically extract words from recently translated sentences and the partial translation of current sentence being translated as words of dynamic cache. We apply the following rules to build the dynamic cache.

a) The max size of the dynamic cache is set to $|c_d|$.

b) According to the first-in-first-out rule, when the dynamic cache is full and a new word is inserted into the cache, the oldest word in the cache will be removed.

c) Duplicate entries into the dynamic cache are not allowed when a word has been already in the cache.

It is worth noting that we also maintain a stop word list, and we added English punctuations and "UNK" into our stop word list. Words in the stop word list would not be inserted into the dynamic cache. So the common words like "a" and "the" cannot appear in the cache. All words in the dynamic cache can be found in the target-side vocabulary of RNNSearch*.

4.1.2 Topic Cache

In order to build the topic cache, we first use an off-the-shelf LDA topic tool[2] to learn topic distributions of source- and target-side documents separately. Then we estimate a topic projection distribution over all target-side topics $p(z_t|z_s)$ for each source topic z_s by collecting events and accumulating counts of (z_s, z_t) from aligned document pairs. Notice that z_s/z_t is the topic with the highest topic probability $p(z_\cdot|d)$ on the source/target side. Then we can use the topic cache as follows:

1) During the training process of NMT, the learned target-side topic model is used to infer the topic distribution for each target document. For a target document d in the training data, we select the topic z with the highest probability $p(z|d)$ as the topic for the document. The $|c_t|$ most probable topical words in topic z are extracted to fill the topic cache for the document d.

2) In the NMT testing process, we first infer the topic distribution for a source document in question with the learned source-side topic model. From the topic distribution, we choose the topic with the highest probability as the topic for the source document. Then we use the learned topic projection function to

[2]http://www.arbylon.net/projects/

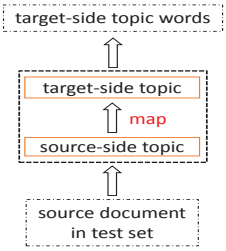


Figure 2: Schematic diagram of the topic projection during the testing process.

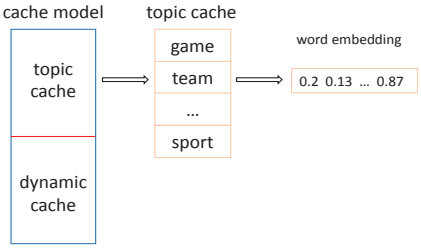


Figure 3: Architecture of the cache model.

map the source topic onto a target topic with the highest projection probability, as illustrated in Figure 2. After that, we use the $|c_t|$ most probable topical words in the projected target topic to fill the topic cache.

The words of topic cache and dynamic cache together form the final cache model. In practice, the cache stores word embeddings, as shown in Figure 3. As we do not want to introduce extra embedding parameters, we let the cache share the same target word embedding matrix with the NMT model. In this case, if a word is not in the target-side vocabulary of NMT, we discard the word from the cache.

4.2 The Model

The cache-based neural model is to evaluate the probabilities of words occurring in the cache and to provide the evaluation results for the decoder via a gating mechanism.

4.2.1 Evaluating Word Entries in the Cache

When the decoder generates the next target word y_t, we hope that the cache can provide helpful information to judge whether y_t is appropriate from the perspective of the document-level cache if y_t occurs in the cache.To achieve this goal, we should appropriately evaluate the word entries in the cache.

In this paper, we build a new neural network layer as the scorer for the cache. At each decoding step t, we use the scorer to score y_t if y_t is in the cache. The inputs to the scorer are the current hidden state s_t of the decoder, previous word y_{t-1}, context vector c_t, and the word y_t from the cache. The score of y_t is calculated as follows:

$$score(y_t|y_{<t}, x) = g_{cache}(s_t, c_t, y_{t-1}, y_t) \tag{6}$$

where g_{cache} is a non-linear function.

This score is further used to estimate the cache probability of y_t as follows:

$$p_{cache}(y_t|y_{<t}, x) = softmax(score(y_t|y_{<t}, x)) \tag{7}$$

4.2.2 Integrating the Cache-based Neural Model into NMT

Since we have two prediction probabilities for the next target word y_t, one from the cache-based neural model p_{cache}, the other originally estimated by the NMT decoder p_{nmt}, how do we integrate these two probabilities? Here, we introduce a gating mechanism to combine them, and word prediction probabilities on the vocabulary of NMT are updated by combining the two probabilities through linear interpolation between the NMT probability and cache-based neural model probability. The final word prediction probability for y_t is calculated as follows:

$$p(y_t|y_{<t}, x) = (1 - \alpha_t)p_{cache}(y_t|y_{<t}, x) + \alpha_t p_{nmt}(y_t|y_{<t}, x) \tag{8}$$

Notice that if y_t is not in the cache, we set $p_{cache}(y_t|y_{<t}, x) = 0$, where α_t is the gate and computed as follows:

$$\alpha_t = g_{gate}(f_{gate}(s_t, c_t, y_{t-1})) \tag{9}$$

where f_{gate} is a non-linear function and g_{gate} is sigmoid function.

We use the contextual elements of s_t, c_t, y_{t-1} to score the current target word occurring in the cache (Eq. (6)) and to estimate the gate (Eq. (9)). If the target word is consistent with the context and in the cache at the same time, the probability of the target word will be high.

Finally, we train the proposed cache model jointly with the NMT model towards minimizing the negative log-likelihood on the training corpus. The cost function is computed as follows:

$$L(\theta) = -\sum_{i=1}^{N}\sum_{t=1}^{T} logp(y_t|y_{<t}, x) \tag{10}$$

where θ are all parameters in the cache-based NMT model.

4.3 Decoding Process

Our cache-based NMT system works as follows:

(1) When the decoder shifts to a new test document, clear the topic and dynamic cache.

(2) Obtain target topical words for the new test document as described in Section 4.1 and fill them in the topic cache.

(3) Clear the dynamic cache when translating the first sentence of the test document.

(4) For each sentence in the new test document, translate it with the proposed cache-based NMT and continuously expands the dynamic cache with newly generated target words and target words obtained from the best translation hypothesis of previous sentences.

In this way, the topic cache can provide useful global information at the beginning of the translation process while the dynamic cache is growing with the progress of translation.

5 Experimentation

We evaluated the effectiveness of the proposed cache-based neural model for neural machine translation on NIST Chinese-English translation tasks.

5.1 Experimental Setting

We selected corpora LDC2003E14, LDC2004T07, LDC2005T06, LDC2005T10 and a portion of data from the corpus LDC2004T08 (Hong Kong Hansards/Laws/News) as our bilingual training data, where document boundaries are explicitly kept. In total, our training data contain 103,236 documents and 2.80M sentences. On average, each document consists of 28.4 sentences. We chose NIST05 dataset (1082 sentence pairs) as our development set, and NIST02, NIST04, NIST06 (878, 1788, 1664 sentence pairs. respectively) as our test sets. We compared our proposed model against the following two systems:

- **Moses** (Koehn et al., 2007): an off-the-shelf phrase-based translation system with its default setting.
- **RNNSearch***: our in-house attention-based NMT system which adopts the feedback attention as described in Section 3 .

For Moses, we used the full training data to train the model. We ran GIZA++ (Och and Ney, 2000) on the training data in both directions, and merged alignments in two directions with "grow-diag-final" refinement rule (Koehn et al., 2005) to obtain final word alignments. We trained a 5-gram language model on the Xinhua portion of GIGA-WORD corpus using SRILM Toolkit with a modified Kneser-Ney smoothing.

For RNNSearch, we used the parallel corpus to train the attention-based NMT model. The encoder of RNNSearch consists of a forward and backward recurrent neural network. The word embedding dimension is 620 and the size of a hidden layer is 1000. The maximum length of sentences that we used to train RNNSearch in our experiments was set to 50 on both Chinese and English side. We used the most

Model	NIST02	NIST04	NIST05	NIST06	Avg
Moses	31.52	32.73	29.52	29.57	30.69
RNNSearch*	36.18	36.36	32.56	30.57	33.92
+ Cd	36.86	37.16	33.10	32.60	34.93
+ Cd, Ct	38.04	38.89	33.31	31.85	35.52

Table 1: Experiment results on the NIST Chinese-English translation tasks. [+Cd] is the proposed model with the dynamic cache. [+Cd, Ct] is the proposed model with both the dynamic and topic cache. Avg means the average BLEU score on all test sets.

frequent 30K words for both Chinese and English. We replaced rare words with a special token "UNK". Dropout was applied only on the output layer and the dropout rate was set to 0.5. All the other settings were the same as those in (Bahdanau et al., 2015). Once the NMT model was trained, we adopted a beam search to find possible translations with high probabilities. We set the beam width to 10.

For the proposed cache-based NMT model, we implemented it on the top of RNNSearch*. We set the size of the dynamic and topic cache $|c_d|$ and $|c_t|$ to 100, 200, respectively. For the dynamic cache, we only kept those most recently-visited items. For the LDA tool, we set the number of topics considered in the model to 100 and set the number of topic words that are used to fill the topic cache to 200. The parameter α and β of LDA were set to 0.5 and 0.1, respectively. We used a feedforward neural network with two hidden layers to define g_{cache} (Equation (6)) and f_{gate} (Equation (9)). For f_{gate}, the number of units in the two hidden layers were set to 500 and 200 respectively. For g_{cache}, the number of units in the two hidden layers were set to 1000 and 500 respectively. We used a pre-training strategy that has been widely used in the literature to train our proposed model: training the regular attention-based NMT model using our implementation of RNNSearch*, and then using its parameters to initialize the parameters of the proposed model, except for those related to the operations of the proposed cache model.

We used the stochastic gradient descent algorithm with mini-batch and Adadelta to train the NMT models. The mini-batch was set to 80 sentences and decay rates ρ and ϵ of Adadelta were set to 0.95 and 10^{-6}.

5.2 Experimental Results

Table 1 shows the results of different models measured in terms of BLEU score[3]. From the table, we can find that our implementation RNNSearch* using the feedback attention and dropout outperforms Moses by 3.23 BLEU points. The proposed model $RNNSearch*_{+Cd}$ achieves an average gain of 1.01 BLEU points over RNNSearch* on all test sets. Further, the model $RNNSearch*_{+Cd,Ct}$ achieves an average gain of 1.60 BLEU points over RNNSearch*, and it outperforms Moses by 4.83 BLEU points. These results strongly suggest that the dynamic and topic cache are very helpful and able to improve translation quality in document translation.

Effect of the Gating Mechanism

In order to validate the effectiveness of the gating mechanism used in the cache-based neural model, we set a fixed gate value for $RNNSearch*_{+Cd,Ct}$, in other words, we use a mixture of probabilities with fixed proportions to replace the gating mechanism that automatically learns weights for probability mixture.

Table 2 displays the result. When we set the gate α to a fixed value 0.3, the performance has an obvious decline comparing with that of $RNNSearch*_{+Cd,Ct}$ in terms of BLEU score. The performance is even worse than RNNSearch* by 10.11 BLEU points. Therefore without a good mechanism, the cache-based neural model cannot be appropriately integrated into NMT. This shows that the gating mechanism plays a important role in $RNNSearch*_{+Cd,Ct}$.

[3]As our model requires document boundaries so as to gurantee that cache words are from the same document, we use all LDC corpora that provide document boundaries. Most training sentences are from Hong Kong Hansards/Laws Parallel Text (accounting for 57.82% of our training data) which are in the law domain rather than the news domain of our test/dev sets. This is the reason that our baseline is lower than other published results obtained using more news-domain training data without document boundaries.

Model	NIST02	NIST04	NIST05	NIST06	Avg
RNNSearch*	36.18	36.36	32.56	30.57	33.92
+ Cd, Ct	38.04	38.89	33.31	31.85	35.52
+ $\alpha = 0.3$	23.39	17.83	31.51	28.90	25.41

Table 2: Effect of the gating mechanism. [+α=0.3] is the [$+Cd, Ct$] with a fixed gate value 0.3.

Model	NIST02	NIST04	NIST05	NIST06	Mean
RNNSearch*	1.90	2.43	1.31	1.50	1.78
+ Ct	2.11	2.51	1.53	1.73	1.96
Reference	2.39	2.51	2.20	1.28	2.09

Table 3: The average number of words in translations of beginning sentences of documents that are also in the topic cache. Reference represents the average number of words in four human translations that are also in the topic cache.

Effect of the Topic Cache

When the NMT decoder translates the first sentence of a document, the dynamic cache is empty. In this case, we hope that the topic cache will provide document-level information for translating the first sentence. We therefore further investigate how the topic cache influence the translation of the first sentence in a document. We count and calculate the average number of words that appear in both the translations of the beginning sentences of documents and the topic cache.

The statistical results are shown in Table 3. Without using the cache model, RNNSearch* generates translations that contain words from the topic cache as these topic words are tightly related to documents being translated. With the topic cache, our neural cache model enables the translations of the first sentences to be more relevant to the global topics of documents being translated as these translations contain more words from the topic cache that describes these documents. As the dynamic cache is empty when the decoder translates the beginning sentences, the topic cache is complementary to such a cold cache at the start. Comparing the numbers of translations generated by our model and human translations (Reference in Table 3), we can find that with the help of the topic cache, translations of the first sentences of documents are becoming closer to human translations.

Analysis on the Cache-based Neural Model

As shown above, the topic cache is able to influence both the translations of beginning sentences and those of subsequent sentences while the dynamic cache built from translations of preceding sentences has an impact on the translations of subsequent sentences. We further study what roles the dynamic and topic cache play in the translation process. For this aim, we calculate the average number of words in translations generated by $RNNSearch*_{+Cd,Ct}$ that are also in the caches. During the counting process, stop words and "UNK" are removed from sentence and document translations.

Table 4 shows the results. If only the topic cache is used ([$document \in [Ct]$,$sentence \in (Ct)$] in Table 4), the cache still can provide useful information to help NMT translate sentences and documents. 28.3 words per document and 2.39 words per sentence are from the topic cache. When both the dynamic and topic cache are used ([$document \in [Ct, Cd]$,$sentence \in (Ct, Cd)$] in Table 4), the numbers of words that both occur in sentence/document translations and the two caches sharply increase from 2.61/30.27 to 6.73/81.16. The reason for this is that words appear in preceding sentences will have a large probability of appearing in subsequent sentences. This shows that the dynamic cache plays a important role in keeping document translations consistent by reusing words from preceding sentences.

We also provide two translation examples in Table 5. We can find that RNNSearch* generates different translations "operations" and "actions" for the same chinese word "行动(xingdong)", while our proposed model produces the same translation "actions".

Model	NIST02	NIST04	NIST05	NIST06	Avg
$document \in (Ct)$	25.70	27.48	26.89	41.00	30.27
$document \in (Cd, Ct)$	59.46	64.24	76.70	124.25	81.16
$sentence \in (Ct)$	2.93	3.07	2.49	1.95	2.61
$sentence \in (Cd, Ct)$	6.77	7.18	7.09	5.89	6.73

Table 4: The average number of words in translations generated by $RNNSearch*_{+Cd,Ct}$ that are also in the dynamic and topic cache. [$document/sentence \in [Ct]$] denote the average number of words that are in both document/sentence translations and the topic cache. [$document/sentence \in [Cd, Ct]$] denote the average number of words occurring in both document/sentence translations and the two caches.

SRC	(1) 并 将 计划 中 的 一 系列 军事 行动 提前 付诸 实施 。 (2) 会议 决定 加大 对 巴方 的 军事 打击 行动 。
REF	(1) and to implement ahead of schedule a series of military actions still being planned . (2) the meeting decided to increase military actions against palestinian .
RNNSearch*	(1) and to implement a series of military operations . (2) the meeting decided to increase military actions against the palestinian side .
+ Cd, Ct	(1) and to implement a series of military actions plans. (2) the meeting decided to increase military actions against the palestinian side .

Table 5: Translation examples on the test set. SRC for source sentences, REF for human translations. These two sentences (1) and (2) are in the same document.

Model	Coherence
RNNSearch*	0.4259
$RNNSearch*_{+Cd,Ct}$	0.4274
Human Reference	0.4347

Table 6: The average cosine similarity of adjacent sentences (coherence) on all test sets.

5.3 Analysis on Translation Coherence

We want to further study how the proposed cache-based neural model influence coherence in document translation. For this, we follow Lapata and Barzilay (2005) to measure coherence as sentence similarity. First, each sentence is represented by the mean of the distributed vectors of its words. Second, the similarity between two sentences is determined by the cosine of their means.

$$sim(S_1, S_2) = cos(\mu(\vec{S_1}), \mu(\vec{S_2})) \tag{11}$$

where $\mu(\vec{S_i}) = \frac{1}{|S_i|} \sum_{\vec{w} \in S_i} \vec{w}$, and $\vec{w}$ is the vector for word w.

We use Word2Vec[4] to get the distributed vectors of words and English Gigaword fourth Edition[5] as training data to train Word2Vec. We consider that embeddings from word2vec trained on large monolingual corpus can well encode semantic information of words. We set the dimensionality of word embeddings to 200. Table 6 shows the average cosine similarity of adjacent sentences on all test sets. From the table, we can find that the $RNNSearch*_{+Cd,Ct}$ model produces better coherence in document translation than RNNSearch* in term of cosine similarity.

6 Conclusion

In this paper, we have presented a novel cache-based neural model for NMT to capture the global topic information and inter-sentence cohesion dependencies. We use a gating mechanism to integrate both the topic and dynamic cache into the proposed neural cache model. Experiment results show that the cache-based neural model achieves consistent and significant improvements in translation quality over several state-of-the-art NMT and SMT baselines. Further analysis reveals that the topic cache and dynamic cache are complementary to each other and that both are able to guide the NMT decoder to use topical words and to reuse words from recently translated sentences as next word predictions.

[4]http://word2vec.googlecode.com/svn/trunk/

[5]https://catalog.ldc.upenn.edu/LDC2009T13

Acknowledgments

The present research was supported by the National Natural Science Foundation of China (Grant No. 61622209). We would like to thank three anonymous reviewers for their insightful comments.

References

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

Nicola Bertoldi, Mauro Cettolo, and Marcello Federico. 2013. Cache-based online adaptation for machine translation enhanced computer assisted translation. *Proceedings of the XIV Machine Translation Summit*, pages 35–42.

Chuang-Hua Chueh and Jen-Tzung Chien. 2010. Topic cache language model for speech recognition. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 5194–5197. IEEE.

Stephen Della Pietra, Vincent Della Pietra, Robert L Mercer, and Salim Roukos. 1992. Adaptive language modeling using minimum discriminant estimation. In *Proceedings of the workshop on Speech and Natural Language*, pages 103–106. Association for Computational Linguistics.

Zhengxian Gong, Min Zhang, and Guodong Zhou. 2011. Cache-based document-level statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 909–919. Association for Computational Linguistics.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2016. Improving neural language models with a continuous cache.

Zhiheng Huang, Geoffrey Zweig, and Benoit Dumoulin. 2014. Cache based recurrent neural network language model inference for first pass speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 6354–6358. IEEE.

Sebastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1–10.

Sebastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. 2017. Does neural machine translation benefit from larger context? *arXiv preprint arXiv:1704.05135*.

Philipp Koehn, Amittai Axelrod, Alexandra Birch, Chris Callison-Burch, Miles Osborne, David Talbot, and Michael White. 2005. Edinburgh system description for the 2005 iwslt speech translation evaluation. In *IWSLT*, pages 68–75.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

R. Kuhn and Renato De Mori. 1990. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):570–583.

Mirella Lapata and Regina Barzilay. 2005. Automatic evaluation of text coherence: models and representations. In *International Joint Conference on Artificial Intelligence*, pages 1085–1090.

Minh Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. *Computer Science*.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. *Association for Computational Linguistics*.

Laurent Nepveu, Guy Lapalme, Philippe Langlais, and George F. Foster. 2004. Adaptive language and translation models for interactive machine translation. In *Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A Meeting of Sigdat, A Special Interest Group of the Acl, Held in Conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*, pages 190–197.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 440–447. Association for Computational Linguistics.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2015. Minimum risk training for neural machine translation. *Computer Science*.

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

Yik Cheung Tam, Ian Lane, and Tanja Schultz. 2007. Bilingual lsa-based adaptation for statistical machine translation. *Machine Translation*, 21(4):187–207.

Jörg Tiedemann. 2010. Context adaptation in statistical machine translation using models with exponentially decaying cache. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 8–15. Association for Computational Linguistics.

Tian Wang and Kyunghyun Cho. 2016. Larger-context language modelling with recurrent neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1319–1329.

Longyue Wang, Zhaopeng Tu, Andy Way, and Qun Liu. 2017. Exploiting cross-sentence context for neural machine translation. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Deyi Xiong and Min Zhang. 2013. A topic-based coherence model for statistical machine translation. In *AAAI*.

Fusing Recency into Neural Machine Translation with an Inter-Sentence Gate Model

Shaohui Kuang **Deyi Xiong***
School of Computer Science and Technology, Soochow University, Suzhou, China
shaohuikuang@foxmail.com, dyxiong@suda.edu.cn

Abstract

Neural machine translation (NMT) systems are usually trained on a large amount of bilingual sentence pairs and translate one sentence at a time, ignoring inter-sentence information. This may make the translation of a sentence ambiguous or even inconsistent with the translations of neighboring sentences. In order to handle this issue, we propose an inter-sentence gate model that uses the same encoder to encode two adjacent sentences and controls the amount of information flowing from the preceding sentence to the translation of the current sentence with an inter-sentence gate. In this way, our proposed model can capture the connection between sentences and fuse recency from neighboring sentences into neural machine translation. On several NIST Chinese-English translation tasks, our experiments demonstrate that the proposed inter-sentence gate model achieves substantial improvements over the baseline.

1 Introduction

In NMT systems (Bahdanau et al., 2015; Cho et al., 2014; Sutskever et al., 2014), an encoder first reads variable-length source sentences and encodes them into a sequence of vectors, then a decoder generates a target translation from the sequence of source vectors. Although NMT is an emerging machine translation approach, the translation process of a document in NMT is similar to conventional statistical machine translation (SMT), treating the document as a bag of sentences and ignoring cross-sentence dependencies.

Sentences are the constituent elements of paragraphs. Harper (1965) argues that sentences possess two attributes: continuity that maintains consistency with other sentences and development that introduces new information. For any adjacent sentences of a well-formed text, they tend to have considerable degree of continuity, which is usually described and measured at two levels: cohesion at the surface level and coherence at the underlying level. These two continuity metrics are two well-known means to estalish such inter-sentence links within a text. Harper (1965) finds that word recurrence, as the most common device of cohesion, occurs in 70% of adjacent sentence pairs. Xiong et al. (2015) show that about 60% of sentences have the same topics (coherence) as those of the documents where these sentences occur.

Such inter-sentence dependencies can and should be used to help document-level machine translation. In the literature, a variety of models have been proposed to capture these dependencies in the context of SMT, such as cache-based language and translation models (Tiedemann, 2010; Gong et al., 2011), topic-based coherence model (Xiong and Zhang, 2013) and lexical cohesion model (Xiong et al., 2013). However, integrating inter-sentence information into an NMT system is still an open problem.

In this paper, we propose a simple yet effective approach to model the inter-sentence information for NMT. In order to capture the connection between two adjacent sentences, i.e. the preceding sentence and current sentence, we first use the same encoder to encode the two adjacent sentences at the same time to form a context vector a for the preceding sentence and a context vector b for current sentence. Then, we introduce a gate mechanism to combine a and b into the final context vector, which is further

* Corresponding author

Proceedings of the 27th International Conference on Computational Linguistics, pages 607–617
Santa Fe, New Mexico, USA, August 20-26, 2018.

used to update the hidden states of the decoder. In this way, we can model the links and dependencies between adjacent sentences. To some extent, our approach models the inter-sentence relationship from the underlying semantic coherence perspective.

On the NIST Chinese-English translation tasks, our experimental results show that the proposed approach can achieve significant improvements of up to 2.0 BLEU points over the NMT baseline.

2 Related Work

In the literature, a series of document-level translation models have been proposed for conventional SMT. Just to name a few, Gong et al. (2011) propose a cache-based approach to document-level translation, which includes three caches, a dynamic cache, a static cache and a topic cache to capture various kind of document-level information. Hardmeier et al. (2012) present a beam search decoding procedure for phrase-based SMT with features modeling cross-sentence dependencies. Xiong and Zhang (2013) propose a topic-based coherence model to produce discourse coherence for document translation. Xiong et al. (2013) present a lexical cohesion model to capture lexical cohesion for document-level translation.

In neural language models, inter-sentence connections can be captured in a contextual model. For example, Lin et al. (2015) propose a hierarchical recurrent neural network (HRNN) language model for document modeling, consisting of a sentence-level and word-level language model, and use the proposed model to model sentence-level coherence. In speech recognition, as input speech signals can contain thousands of frames, Chan et al. (2016) employ Bidirectional Long Short Term Memory with a pyramidal structure to capture the context of a large number of input time steps. Wang and Cho (2016) introduce a late fusion method to incorporate corpus-level discourse information into recurrent language modeling.

In neural conversation systems, links between multi-turn conversations are usually modeled with hierarchical neural networks. Serban et al. (2015) use a hierarchical recurrent encoder-decoder(HRED) to model the dialogue into two-level hierarchy: a sequence of utterances and a sequence of words. The proposed model can track states over many utterances to generate context-aware multiple rounds of dialogue. Serban et al. (2016) further propose a HRED model with an additional component: a high-dimensional stochastic latent variable at every dialogue turn to sample a vaussian variable as input to the decoder.

It is natural to adapt the HRED model to document-level NMT. However, document boundaries are usually missing in bilingual training corpora, indicating that we do not have sufficient data to train the sentence-level hierarchy. In our proposed gate model, we do not need entire documents to train the NMT model. We only use pairs of two adjacent sentences to train the gate. Furthermore, each sentence in a sentence pair can be used twice: one as a preceding sentence and the other as a current sentence. Additionally, in order to reduce the number of extra parameters, we use the same encoder to encode the adjacent sentences, which can also keep the semantic consistency for the same source sentence.

3 Neural Machine Translation

In this section, we briefly describle the atttention-based NMT model proposed in (Bahdanau et al., 2015).

In their framework, the encoder encodes a source sentence as a sequence of vectors with bi-directional RNNs. The forward RNN reads the source sentence $x = (x_1, x_2, ..., x_T)$ from left to right and the backward RNN reads the source sentence in an inverse direction. The hidden states $\overrightarrow{h} = (\overrightarrow{h_1}, \overrightarrow{h_2}, ..., \overrightarrow{h_T})$ in the forward RNN can be computed as follows:

$$\overrightarrow{h_j} = f(\overrightarrow{h_{j-1}}, x_j), \tag{1}$$

where f is a non-linear function, here defined as a gated recurrent unit (GRU) (Chung et al., 2014). Similarly, Hidden states of the backward RNN can be calculated. The forward and backward hidden states are concatenated into the final annotation vectors $h = (h_1, h_2, ..., h_T)$. The decoder is also an RNN that predicts the next word y_t given the context vector c_t , the hidden state s_{t-1} and the previously generated word sequence $y_{<t} = [y_1, y_2, ..., y_{t-1}]$. The probability of the next word y_t is calculated as follows:

$$p(y_t|y_{<t}; x) = g(c_t, y_{t-1}, s_t), \tag{2}$$

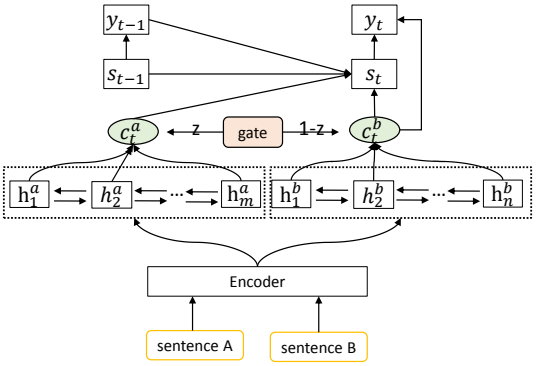


Figure 1: Architecture of NMT with the inter-sentence gate.

where g is a softmax layer, s_t is the state of decoder RNN at time step t computed as

$$s_t = f(s_{t-1}, y_{t-1}, c_t). \tag{3}$$

where f is a function, the same as function used in the encoder. The context vector c_t is calculated as a weighted sum of all hidden states of the encoder as follows:

$$c_t = \sum_{i=1}^{T_x} \alpha_{tj} h_j, \tag{4}$$

$$\alpha_{tj} = \frac{exp(e_{tj})}{\sum_{k=1}^{T_x} exp(e_{tk})}, \tag{5}$$

$$e_{tj} = a(s_{t-1}, h_j). \tag{6}$$

where α_{tj} is the weight of each hidden state h_j computed by the attention model, a is a feedforward neural network with a single hidden layer.

We also implement an NMT system which adopts feedback attention (Wang et al., 2016b; Wang et al., 2016a), which will be referred to as RNNSearch in this paper. In the feedback attention, e_{tj} is computed as follows:

$$e_{tj} = a(\widetilde{s}_{t-1}, h_j), \tag{7}$$

where $\widetilde{s}_{t-1} = GRU(s_{t-1}, y_{t-1})$. The hidden state of the decoder is updated as follows:

$$s_t = GRU(\widetilde{s}_{t-1}, c_t) \tag{8}$$

In this paper, our proposed model is implemented on the top of RNNSearch system.

4 The Inter-Sentence Gate Model

In this section, we will elaborate the proposed inter-sentence gate model, which we refer to as NMT_{ISG}. Figure 1 shows the entire architecture of our NMT with the inter-sentence gate. For notational convenience, we denote two adjacent sentences as A and B: A for the preceding sentence and B for the current sentence.

4.1 Encoder

We employ the same encoder to encode the adjacent sentence A and B into hidden vector representations $[h_1^a, h_2^a, ..., h_m^a]$ and $[h_1^b, h_2^b, ..., h_n^b]$ respectively. We then use the attention network described in Equation (4) of Section 3 to compute their context representations c_t^a and c_t^b.

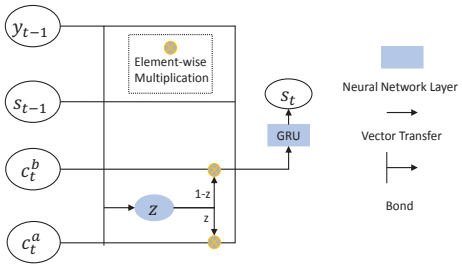


Figure 2: The framework of updating the decoding state using the proposed gate model.

4.2 Inter-Sentence Gate Model

When we translate the current sentence B, we have to make sure that the decoder is provided with sufficient information from sentence B, and at the same time, with helpful imformation from the preceding sentence A. In other words, we need a mechanism to control the scale of information flowing from the sentence A and sentence B to the decoder. Inspired by the success of gated units in RNN (Chung et al., 2014), we propose an inter-sentence gate to control the amount of information flowing from A and B. Formally, we employ a sigmoid neural network layer and an element-wise multiplication operation, as illustrated in Figure 2. Similarly, Tu et al. (2016) also propose a gating mechanism to combine source and target contexts. The gate framework assigns element-wise weights z to the input signals, calculated by

$$z_t = \sigma(U_z s_{t-1} + W_z y_{t-1} + C_b c_t^b + C_a c_t^a) \tag{9}$$

here σ is a logistic sigmoid function, and U_z, W_z, C_b, C_a are the parameter matrix.

4.3 Decoder

Next, we integrate the inter-sentence gate into the decoder to decide the amount of context information used in producing the decoder hidden state at each time step. In this way, we want the hidden states of the decoder to store the inter-sentence context information. The framework of updating the hidden state of the decoder at time step t is illustrated in Figure 2. The decoder hidden state s_t is computed as follows:

$$s_t = GRU(U s_{t-1} + W y_{t-1} + C_1 c_t^a * z_t + C_2 c_t^b * (1 - z_t)) \tag{10}$$

where $*$ is an element-wise multiplication, U, W, C_1, C_2 is the parameter matrix, and z_t is the inter-sentence gate computed by Equation (10).

The conditional probability of the next word y_t is calculated as follows:

$$p(y_t|y_{<t}, x) = g(f(s_t, y_{t-1}, c_t^b)) \tag{11}$$

where c_t^b is the context vector of the current sentence B.

Our aim is to translate the current sentence B with the additional information from the preceding sentence A. We do not want to have the excessive impact of the preceding sentence on the translation output of the current sentence. Therefore, in the stage of generating the next word, we just use the context c_t^b.

5 Experiments

We carried out a series of Chinese-to-English translation experiments to evaluate the effectiveness of the proposed inter-sentence gate model on document-level NMT and conducted in-depth analyses on experiment results and translations.

5.1 Experimental Settings

We selected corpora LDC2003E14, LDC2004T07, LDC2005T06, and LDC2005T10 as our bilingual training data, where document boundaries are kept. We also used all data from the corpus LDC2004T08

Model	NIST05	NIST02	NIST03	NIST04	NIST06	NIST08	Avg
Moses	29.52	31.52	31.68	32.73	29.57	23.09	29.72
RNNSearch	32.56	36.18	34.85	36.36	30.57	23.69	32.37
NMT_{ISG}	$34.58^{\ddagger}$	$36.68^{\ddagger}$	$36.29^{\ddagger}$	$38.15^{\ddagger}$	$31.83^{\ddagger}$	$24.67^{\ddagger}$	33.7

Table 1: Experiment results on the NIST Chinese-English translation task. We adopted the RNNSearch, an in-house NMT system, as our baselines. NMT_{ISG} is the proposed model without replacing UNK words. The BLEU scores are case-insensitive. Avg means the average BLEU score on all the test sets. "‡" : statistically better than RNNSearch ($p < 0.01$).

(Hong Kong Hansards/Laws/News). In total, our training data contain 103,236 documents and 2.80M sentences. Averagely, each document consists of 28.4 sentences. We chose NIST05 dataset as our development set, and NIST02, NIST03, NSIT04, NIST06, NIST08 as our test sets. We used case-insensitive BLEU-4 as our evaluation metric. We compared our NMT_{ISG} with the following two systems:

- **Moses** (Koehn et al., 2007): an open phrase-based translation system with its default setting.
- **RNNSearch**: our new implementation of NMT system with the feedback attention as described in Section 3.

For Moses, we used the full training data (parallel corpus) to train the model. Word alignments were produced by GIZA++ (Och and Ney, 2000). We ran GIZA++ on the corpus in both directions, and merged alignments in two directions with "grow-diag-final" refinement rule (Koehn et al., 2005). We trained a 5-gram language model on the Xinhua portion of the Gigaword corpus using SRILM Toolkit with a modified Kneser-Ney smoothing.

For RNNSearch, we used the parallel corpus to train the attention-based NMT model. The encoder of RNNSearch consists of a forward (1000 hidden unit) and backward (1000 hidden unit) recurrent neural network. The maximum length of sentences that we used to train NMT in our experiments was set to 50 for both the Chinese and English sides. We used the most frequent 30K words for both Chinese and English, covering approximately 99.0% and 99.2% of the data in the two languages respectively. We replaced rare words with a special token "UNK". We also adopted the dropout technique. Dropout is applied only on the output layer and the dropout rate was set to 0.5. All the other settings are the same as the setting up described by Bahdanau et al. (2015). Once the NMT model was trained, we employed a beam search algorithm to find possible translations with high probabilities. We set the beam width to 10.

For the proposed NMT_{ISG} model, we implemented it on the top of RNNSearch. We used tuples (x, before-x, y) as input of NMT_{ISG}, where x and y are a parallel sentence pair, before-x is the previous sentence of source sentence x in the same document[1]. As the first sentence of a document does not have before-x, we used the stop symbol to form the sentence s = (eos, eos, eos) as the before-x. We used a simple pre-training strategy to train our NMT_{ISG} model: training the regular attention-based NMT model using our implementation of RNNSearch, and then using its parameters to initialize the parameters of the proposed model, except for those related to the operations of the inter-sentence gate.

We used the stochastic gradient descent algorithm with mini-batch and Adadelta (Zeiler, 2012) to train the NMT model. The mini-batch was set to 80 sentences and decay rates ρ and ϵ of Adadelta were set to 0.95 and 10^{-6}, respectively.

5.2 Experimental Results

Table 1 shows the results of different NMT systems measured in terms of BLEU score. From the table, we can find that our implementation RNNSearch using the feedback attention and dropout outperforms

[1]We obtain tuples (x, before-x, y) from the training corpus. During the acquisition of these tuples, we follow two constraints. First, we discard the tuple (x,before-x,y) if there is a big difference in the length of the sentence x and before-x. For example, sentence x or before-x is a date expression at the end of a document or an organization name at the beginning of a document. Second, the length of any element in a tuple (x, before-x, y) is not longer than 50.

Model	NIST02	NIST03	NIST04	NIST05	NIST06	NIST08
RNNSearch	32.56	36.18	34.85	36.36	30.57	23.69
RNNSearch + concat	21.81	18.72	19.44	16.11	16.60	9.81

Table 2: The BLEU scores of RNNSearch + concat model which uses the concatenation of two neighboring sentence as input of RNNSearch.

Model	NIST05	NIST02	NIST03	NIST04	NIST06	NIST08	Avg
RNNSearch	32.56	36.18	34.85	36.36	30.57	23.69	32.37
NMT_{ISG}	34.58	36.68	36.29	38.15	31.83	24.67	33.7
+NULL	33.22	36.46	35.27	37.11	30.77	23.80	32.77
+z=**0**	30.49	31.71	31.05	35.10	29.89	22.10	30.06
+RV	31.88	36.21	34.51	35.93	29.96	22.99	31.91

Table 3: Effect of the inter-sentence gate and information of before-x. BLEU scores in the table are case-insensitive. [+NULL] is set the before-x to a NULL sentence. [+z=0] is set the gate vector to all-zero vector. And [+RV] is set the context vector of before-x to a random vector which value of vector is between -1 and 1. Avg meaning the average BLEU score on all the test sets.

Moses by 2.65 BLEU points. The proposed model NMT_{ISG} achieves an average gain of 1.33 BLEU points over RNNSearch on all test sets. And it outperforms Moses by 3.98 BLEU points.

One might use the concatenation of two neighboring source sentences as input of RNNSearch to explore the information of the preceding sentence. However, this will degenerate translation quality as shown in Table 2. The main reason is that the conventional NMT has difficulties in translating long sentences (Pougetabadie et al., 2014). Thus, we conclude that the information of preceding sentences cannot be directly explored via concatenation.

5.3 Effect of the Inter-Sentence Gate

In order to examine the effectiveness of the proposed inter-sentence gate and inter-sentence information from before-x, we also conducted three additional validation experiments in the test sets: (1) we set all before-x to NULL (the before-x sentence consists of only stop symbols). (2) we set z_t to the fixed vector value **0** for NMT_{ISG} to block the inter-sentence gate mechanism. (3) we set the context vector c_t^a of before-x to a random vector, the purpose of which is to test whether the information of the preceding sentence has bad influence on the translation of the current sentence when the preceding sentence is not quite related with the current sentence, for example, a topic change happens between x and before-x. The results are shown in Table 3, from which can find that:

- When we set before-x as NULL (+NULL in Table 3), the performance has an obvious decline comparing with that of NMT_{ISG}, but is still better than RNNSearch in term of BLEU score. We conjecture that the reasons for this are twofold. First, in the training process, when before-x dose not exist for the first sentence of a document, we set before-x to a NULL sentence, which makes NMT_{ISG} model learn the relevant capabilities. That is to say, NMT_{ISG} treats all sentences in the test sets as the first sentences of documents, where some sentences are correctly handled while others not. Second, the gate mechanism assigns a pretty low rate for NULL before-x during context combination.

- When we set the gate weight vector z to a fixed all-zero vector (+z=**0** in Table 3), NMT_{ISG} blocks the inter-sentence information from before-x. From the Table 3, we find a huge loss in performance. The NMT_{ISG} (+z=**0**) is even worse than RNNSearch by 2.31 BLEU points. Although the preceding sentence information is not used, this NMT_{ISG} (+z=**0**) is not exactly the baseline RNNSearch. There are two groups of parameters in NMT_{ISG}: one group of parameters (new parameters) are related with the inter-sentence gate and the other group of parameters (old parameters) are from

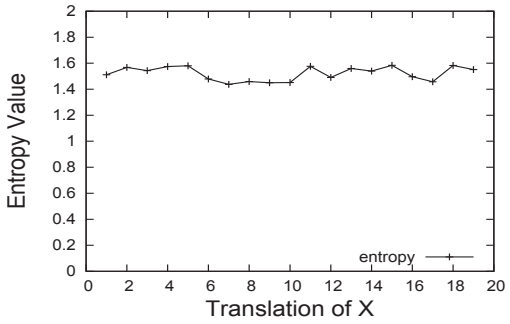


Figure 3: The entropy curve for a NULL preceding sentence. This is a sample generated by NMT_{ISG} in one test set. The horizontal axis represents the translation of a current sentence x. The vertical axis represents the entropy of the preceding sentence before-x obtained from the inter-sentence gate model when translating x.

RNNSearch but have been optimized towards the maximum usage of inter-sentence information. When the inter-sentence gate is closed, old parameters are not able to guide the decoder to generate translations that best use the current sentence information. That is the reason why this result is even worse than RNNSearch.

- When we set the context vector c_t^a of before-x to a random vector (+RV in Table 3), sampled from a uniform distribution between -1 and 1, the performance is worse than the baseline RNNSearch by 0.46 BLEU points. Setting the c_t^a to a random vector, the information from the pseudo preceding sentence becomes meaningless, and even has a bad or uncorrelated impact on the translation of the current sentence. However, the drop of the performance is not as big as that of NMT_{ISG} (+z=**0**). This suggests that the gate mechanism is able to effectively shield these useless and counteractive information from a pseudo and random preceding sentence.

The three experiments further demonstrate that the proposed inter-sentence gate is able to detect useful information for translation and block unrelated information and reconfirm that inter-sentence information is useful for translation.

5.4 Analysis on Inter-Sentence Attention

Many studies (Bahdanau et al., 2015; Luong et al., 2015) on attention-based NMT have proved that attention networks are able to detect alignments between parallel sentence pairs. In our NMT_{ISG} model, we use two attention networks: the first is built for the correspondences between the current source sentence and its target translation and the second for the correspondences between the preceding sentence and the target translation of the current sentence. We are interested in what correspondences the second attention network detect.

We use the entropy as the evaluation criterion to measure how attention weights distribute over words in the preceding sentence before-x for a target word in the translation of the current sentence x. If the attention distribution is even, the entropy will be large. Otherwise, the entropy is small, it suggests that the attention distribution is uneven and that the decoder pays attention to one or several particular words in the preceding sentence when generating a target word for the current sentence. The entropy is computed as follows:

$$H = -\sum_{j=1}^{n} \alpha_j log \alpha_j \tag{12}$$

We calculated entropy values in two cases:

- The before-x is a NULL sentence, being composed of stop symbols.
- The before-x is an ordinary sentence.

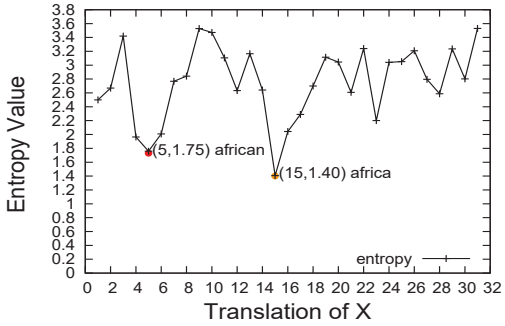


Figure 4: The entropy curve for a normal preceding sentence.

Before-x	进入 新 世纪 后 ， 经济 全球化 成为 非洲 国家面临 的 一个 重大 而 严峻 的 挑战 。
X	非洲 国家领导人 越来越 清楚 地 认识 到,非洲再也 不 能 坐失良机 , 应 奋起 寻求 应对 之 策 。
NMT_{ISG}	the leaders of the african countries have become more and more clearly aware that africa can no longer be able to UNK and take the initiative to pursue countermeasures.

Table 4: Example translation genereted by NMT_{ISG}, before-x is the preceding sentence of x.

If before-x is NULL, it cannot provide useful information except for the indicator of being the first sentence of a document. The attention weight distribution over stop symbols in before-x when generating a target word is supposed to be uniform. Figure 3 exactly visualizes such a case when translating the first sentence in a document. The entropy value in Figure 3 ranges from 1.44 to 1.58 while the entire curve is quite smooth.

Figure 4 demonstrates the second case with an example (in Table 4) where some words in the preceding sentence reoccur in the current sentence. When the NMT_{ISG} model generates the target translation for "非洲国家(feizhou guojia)" and "非洲(feizhou)", the entropy significantly drops as these words have occurred both in the preceding and current sentence. This indicates that the attention network in NMT_{ISG} successfully captures this word repetition (the most common lexical cohesion device) and convey such an inter-sentence relation collectively with the proposed inter-sentence gate to the prediction of target words via hidden states of the decoder.

5.5 Analysis on Translation Coherence

We want to further study how the proposed inter-sentence gate model influence coherence in document translation. For this, we follow Lapata and Barzilay (2005) to measure coherence as sentence similarity. First, each sentence is represented by the mean of the distributed vectors of its words. Second, the similarity between two sentences is determined by the cosine of their means.

$$sim(S_1, S_2) = cos(\mu(\vec{S_1}), \mu(\vec{S_2})) \tag{13}$$

where $\mu(\vec{S_i}) = \frac{1}{|S_i|} \sum_{\vec{w} \in S_i} \vec{w}$, and $\vec{w}$ is the vector for word w.

We use Word2Vec[2] to obtain the distributed vectors of words and English Gigaword fourth Edition as training data to train Word2Vec. We consider that embeddings from word2vec trained on large monolingual corpus can well encode semantic information of words. We set the vectors of words to 400.

Table 5 shows the average cosine similarity of adjacent sentences in test sets. From the table, we can find that the NMT_{ISG} model produces better coherence in document translation than RNNSearch in term of cosine similarity.

[2]https://code.google.com/p/word2vec/

Model	NIST02	NIST03	NIST04	NIST05	NIST06	NIST08
RNNSearch	0.4536	0.4510	0.4761	0.4677	0.3982	0.3716
NMT_{ISG}	0.4744	0.4656	0.4849	0.4816	0.4072	0.3825
Human Reference	0.5090	0.4367	0.5100	0.5073	0.3804	0.3911

Table 5: The average cosine similarity of adjacent sentences in test sets.

before-x	自从 上个月 巴勒斯坦 强人 阿拉法特 去 逝 后 , 国际 社会 重新 继续 恢复 中东 和平 政策 的 推动 , 以期 早日 结束 以 巴 之间 多年 的 流血 冲突 。
x	中东 新闻社 说 , 官员 预测 「 准备 工作 将 会 进行 到 七月 , 然后 再 展开 政治动作 」
RNNSearch	the UNK news agency said that the officials forecast that "preparations will be made in july and then political moves will be taken again."
NMT_{ISG}	the middle east news agency said, the officials forecast that "preparations will be made in july and then another political action will be taken."
before-x	美军 第八 军团 司令 康贝尔 中 将 发表 声明 , 此 一 冻结 调防 军令 旨在 确保 驻 南 韩美军 实力 。
x	根据 南韩 与 美国 签订 的 协 防 条约 , 目前 驻 南 韩 美军 人数 约 三万七 千 人 , 自 去年 十二月 北 韩 发展 核子 计 画 野心 曝光 以来 , 驻 韩美军 一 直 保持 警戒 。
RNNSearch	according to the UNK treaty signed between south korea and the united states, the number of us troops in south korea is about UNK, and the us troops stationed in the rok since december last year have been kept alert.
NMT_{ISG}	according to the UNK treaty signed between south korea and the united states, the number of us troops stationed in south korea is about UNK. the us troops stationed in south korea have been maintaining vigilance since last december last year when north korea's nuclear plan was exposed.

Table 6: Example translations genereted by NMT_{ISG}.

In order to better reflect the performance of the model about coherence, we also provide two examples displayed in Table 6 to verify the impact of inter-sentence information on document-level NMT. In the first example, source x does not have enough context information to correctly translate word "动作(dongzuo)". Fortunately, the before-x provides extra information: the background is about politices, which guide the decoder to select a better translation "action" for "动作(dongzuo)". In the second example, RNNSearch generates different translations for "驻韩(zhuhan)" and "驻南韩(zhunanhan)". NMT_{ISG} generates consistent translations for these two different words but with the same meaning.

6 Conclusion and Future Work

In this paper, we have presented a novel inter-sentence gate model for NMT to deal with document-level translation. Experimental results show that the NMT_{ISG} model achieves consistent and significant improvements in translation quality over strong NMT baselines. In-depth analyses further demonstrate that the proposed model inter-sentence gate is able to capture cross-sentence dependencies and lexical cohesion devices.

The proposed inter-sentence gate model only uses source-side information to capture document-level information for translation. In the future, we would like to integrate target-side information into document-level NMT.

Acknowledgments

The present research was supported by the National Natural Science Foundation of China (Grant No. 61622209). We would like to thank three anonymous reviewers for their insightful comments.

References

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

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964. IEEE.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *In Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Presented in NIPS 2014 Deep Learning and Representation Learning Workshop*.

Zhengxian Gong, Min Zhang, and Guodong Zhou. 2011. Cache-based document-level statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 909–919. Association for Computational Linguistics.

Christian Hardmeier, Joakim Nivre, and Jörg Tiedemann. 2012. Document-wide decoding for phrase-based statistical machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1179–1190. Association for Computational Linguistics.

Kenneth E Harper. 1965. Studies in inter-sentence connection. Technical report, DTIC Document.

Philipp Koehn, Amittai Axelrod, Alexandra Birch, Chris Callison-Burch, Miles Osborne, David Talbot, and Michael White. 2005. Edinburgh system description for the 2005 iwslt speech translation evaluation. In *IWSLT*, pages 68–75.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Mirella Lapata and Regina Barzilay. 2005. Automatic evaluation of text coherence: models and representations. In *International Joint Conference on Artificial Intelligence*, pages 1085–1090.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *Conference on Empirical Methods in Natural Language Processing*, pages 899–907.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 14121421.*

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 440–447. Association for Computational Linguistics.

Jean Pougetabadie, Dzmitry Bahdanau, Bart Van Merrienboer, Kyunghyun Cho, and Yoshua Bengio. 2014. Overcoming the curse of sentence length for neural machine translation using automatic segmentation. *Eprint Arxiv*.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015. Building end-to-end dialogue systems using generative hierarchical neural network models. *Association for the Advancement of Artificial Intelligence*, (4).

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. A hierarchical latent variable encoder-decoder model for generating dialogues.

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

Jörg Tiedemann. 2010. Context adaptation in statistical machine translation using models with exponentially decaying cache. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 8–15. Association for Computational Linguistics.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2016. Context gates for neural machine translation. *arXiv preprint arXiv:1608.06043*.

Tian Wang and Kyunghyun Cho. 2016. Larger-context language modelling with recurrent neural network. In *Meeting of the Association for Computational Linguistics*, pages 1319–1329.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2016a. Memory-enhanced decoder for neural machine translation. *arXiv preprint arXiv:1606.02003*.

Xing Wang, Zhengdong Lu, Zhaopeng Tu, Hang Li, Deyi Xiong, and Min Zhang. 2016b. Neural machine translation advised by statistical machine translation. *AAAI*.

Deyi Xiong and Min Zhang. 2013. A topic-based coherence model for statistical machine translation. In *AAAI*.

Deyi Xiong, Guosheng Ben, Min Zhang, Yajuan Lv, and Qun Liu. 2013. Modeling lexical cohesion for document-level machine translation. In *IJCAI*.

Deyi Xiong, Min Zhang, and Xing Wang. 2015. Topic-based coherence modeling for statistical machine translation. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 23(3):483–493.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Improving Neural Machine Translation by Incorporating Hierarchical Subword Features

Makoto Morishita, Jun Suzuki* and **Masaaki Nagata**
NTT Communication Science Laboratories, NTT Corporation, Japan
{morishita.makoto, nagata.masaaki}@lab.ntt.co.jp
jun.suzuki@ecei.tohoku.ac.jp

Abstract

This paper focuses on subword-based Neural Machine Translation (NMT). We hypothesize that in the NMT model, the appropriate subword units for the following three modules (layers) can differ: (1) the encoder embedding layer, (2) the decoder embedding layer, and (3) the decoder output layer. We find the subword based on Sennrich et al. (2016) has a feature that a large vocabulary is a superset of a small vocabulary and modify the NMT model enables the incorporation of several different subword units in a single embedding layer. We refer these small subword features as hierarchical subword features. To empirically investigate our assumption, we compare the performance of several different subword units and hierarchical subword features for both the encoder and decoder embedding layers. We confirmed that incorporating hierarchical subword features in the encoder consistently improves BLEU scores on the IWSLT evaluation datasets.

Title and Abstract in Japanese

階層的部分単語特徴を用いたニューラル機械翻訳

本稿では、部分単語 (subword) を用いたニューラル機械翻訳 (Neural Machine Translation, NMT) に着目する。NMT モデルでは、エンコーダの埋め込み層、デコーダの埋め込み層およびデコーダの出力層の 3 箇所で部分単語が用いられるが、それぞれの層で適切な部分単語単位は異なるという仮説を立てた。我々は、Sennrich et al. (2016) に基づく部分単語は、大きな語彙集合が小さい語彙集合を必ず包含するという特徴を利用して、複数の異なる部分単語列を同時に一つの埋め込み層として扱えるよう NMT モデルを改良する。以降、この小さな語彙集合特徴を階層的部分単語特徴と呼ぶ。本仮説を検証するために、様々な部分単語単位や階層的部分単語特徴をエンコーダ・デコーダの埋め込み層に適用して、その精度の変化を確認する。IWSLT 評価セットを用いた実験により、エンコーダ側で階層的な部分単語を用いたモデルは BLEU スコアが一貫して向上することが確認できた。

1 Introduction

The approach of end-to-end Neural Machine Translation (NMT) continues to make rapid progress. A simple encoder-decoder model was proposed by Sutskever et al. (2014), and an attentional mechanism was added to better exploit the encoder-side information (Luong et al., 2015; Bahdanau et al., 2015). Compared to traditional Statistical Machine Translation (SMT), NMT has relatively simple architecture, which only uses a large single neural network, but its accuracy surpasses SMT (Junczys-Dowmunt et al., 2016).

A conventional NMT uses a limited vocabulary and a decoder generates a "word" in the vocabulary at each time step, but a problem occurs when it encounters an out-of-vocabulary word. Since NMT cannot correctly encode and generate such out-of-vocabulary words, the task performance is degraded. To solve this problem, Sennrich et al. (2016) proposed a method that expresses a word by combining "subwords."

*His current affiliation is Tohoku University.

Proceedings of the 27th International Conference on Computational Linguistics, pages 618–629
Santa Fe, New Mexico, USA, August 20-26, 2018.

A subword is a fraction of a word, determined by Byte Pair Encoding (BPE) operations. By the BPE operation, a word that appears frequently can be one unit, and rare or uncommon words can be expressed by the combination of several subword units. Thus we can express any words by the combination of small subword vocabularies to alleviate the out-of-vocabulary problem. Several similar works exist to make subword units (e.g., (Schuster and Nakajima, 2012; Kudo, 2018)), but in the following, we denote the units segmented by BPE as subword units unless otherwise noted.

The primary reason why we use subword units is to generate rare or unknown words on the decoder side. In other words, our purpose is to change the vocabulary at the decoder output layer into subwords. Once we decide the vocabulary in the decoder output layer, it is natural to use the same vocabulary in the decoder embedding layer. We also use subword units in the encoder side to maintain consistency with the decoder. As described, NMT has three layers that are related to subwords: the encoder embedding layer, the decoder embedding layer, and the decoder output layer. However, we generally use the same operations to make subword units.

We hypothesize that the optimal subword units can be different among these three layers. Since these layers play different roles in the model, the subword units should be determined based on each role. To validate this hypothesis, we modify the model to simultaneously deal with several different subword units.

We focus on the property that the large subword vocabulary is always a superset of the small subword vocabulary. By taking advantage of this, we propose the model uses these small subword vocabularies as additional features of an embedding layer. We name these small subword vocabulary features as hierarchical subword features. We simply use the sum of the embeddings of each hierarchical subword features to represent each embedding. This simple approach is GPU friendly and does not increase the computational time.

We empirically investigate our assumption and find that incorporating several different subword units for encoder embedding layers consistently improves the BLEU scores on the IWSLT 2012, 2013, and 2014 evaluation datasets.

2 Neural Machine Translation with Subword Units

Among many options for a model architecture of NMT models, our baseline's model architecture was introduced in Luong et al. (2015) with a global attention mechanism and a bi-directional encoder (Bahdanau et al., 2015).

2.1 Formulation

In general, the NMT model receives an input sentence and returns a corresponding (translated) output sentence. Here, to concisely explain the NMT model, its input and output are both sequences of one-hot vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$ that respectively correspond to input and output sentences. This conversion can be performed straightforwardly without loss of generality since each token (word) has a one-to-one correspondence to a one-hot vector.

Let $\mathcal{V}_s$ and $\mathcal{V}_t$ respectively represent the vocabulary sizes of the input and the output. Let $\boldsymbol{x}_i \in \{0,1\}^{V_s}$ represent the one-hot vector of the i-th token in $\boldsymbol{X}$. Similarly, let $\boldsymbol{y}_j \in \{0,1\}^{V_t}$ represent the one-hot vector of the j-th token in $\boldsymbol{Y}$. We introduce notation $\boldsymbol{x}_{1:I}$ to represent a list of one-hot vectors, i.e., $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_I)$, as a short notation where I represents the length (the number of instances) of the list. Then the NMT model approximates the following conditional probability:

$$p(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{j=1}^{J+1} p(\boldsymbol{y}_j|\boldsymbol{y}_{0:j-1}, \boldsymbol{X}), \tag{1}$$

where $\boldsymbol{y}_0$ is a one-hot vector of a special begin-of-sentence (BOS) token and $\boldsymbol{y}_{J+1}$ is a one-hot vector of a special end-of-sentence (EOS) token. Moreover, $\boldsymbol{X} = \boldsymbol{x}_{1:I}$ and $\boldsymbol{Y} = \boldsymbol{y}_{0:J+1}$.

Our baseline NMT model consists of three primary components (modules): encoder, attention, and decoder. The following briefly explains these three components. Hereafter, we assume that the number

of tokens in the input sentence is I, the number of tokens in the output sentence is J, the dimensions of the embedding vectors are D, and the dimensions of the hidden vectors are H.

Encoder: The encoder generates a list of hidden vectors $\boldsymbol{h}_{1:I}$ given an input sequence of one-hot vectors $\boldsymbol{x}_{1:I}$. Let $\boldsymbol{E} \in \mathbb{R}^{D \times |\mathcal{V}_s|}$ represent an (encoder) embedding matrix. Then $\texttt{Enc}(\cdot)$, which denotes a function that returns a list of encoded vectors $\boldsymbol{h}_{1:I}$, is calculated:

$$\boldsymbol{h}_{1:I} = \texttt{Enc}(\boldsymbol{e}_{1:I}), \quad \text{where} \quad \boldsymbol{e}_i = \boldsymbol{E}\boldsymbol{x}_i \quad \text{for all } i. \tag{2}$$

Finally, the encoder outputs $\boldsymbol{h}_{1:I}$.

Decoder and attention mechanism: The decoder estimates the probability of the output sentence given the encoded information generated by the encoder: $\boldsymbol{h}_{1:I}$. The attention mechanism allows the decoder to directly incorporate $\boldsymbol{h}_{1:I}$ at each decoder time step j.

Let $\boldsymbol{F} \in \mathbb{R}^{D \times |\mathcal{V}_t|}$ represent an (decoder) embedding matrix. Let $\texttt{Dec\_Attn}(\cdot)$ be a function that returns the final hidden vector at decoder time step j, namely, $\boldsymbol{z}_j$, which is calculated based on $\boldsymbol{f}_j$, $\boldsymbol{z}_{j-1}$, and $\boldsymbol{h}_{1:I}$ for all j:

$$\boldsymbol{z}_j = \texttt{Dec\_Attn}\big(\boldsymbol{z}_{j-1}, \boldsymbol{f}_j, \boldsymbol{h}_{1:I}\big), \quad \text{where} \quad \boldsymbol{f}_j = \boldsymbol{F}\boldsymbol{y}_{j-1}. \tag{3}$$

Here we assume that both $\boldsymbol{y}_j$ and $\boldsymbol{z}_j$ are zero-vectors if $j = 0$.

Then let $\boldsymbol{W} \in \mathbb{R}^{|\mathcal{V}_t| \times H}$ and $\boldsymbol{b} \in \mathbb{R}^{|\mathcal{V}_t|}$ be a weight matrix and a bias term in the decoder's output layer. Finally, the decoder calculates the probability of $\boldsymbol{y}_j$ at each time step j, which is $p(\boldsymbol{y}_j | \boldsymbol{y}_{0:j-1}, \boldsymbol{X})$ from Eq. 1:

$$p(\boldsymbol{y}_j | \boldsymbol{y}_{0:j-1}, \boldsymbol{X}) = \frac{\exp(\boldsymbol{o}_j) \cdot \boldsymbol{y}_j}{\exp(\boldsymbol{o}_j) \cdot \boldsymbol{1}}, \quad \text{where} \quad \boldsymbol{o}_j = \boldsymbol{W}\boldsymbol{z}_j + \boldsymbol{b}, \tag{4}$$

where $\boldsymbol{1}$ is a vector whose elements are all 1.

In the generation (test) phase, we generally use a K-best beam search to generate output sentences with the (approximated) K-highest probability given input sequence $\boldsymbol{X}$.

2.2 Subword Units Based on Byte-Pair Encoding

Several approaches have been proposed to obtain a set of subword units based on statistics, e.g., (Schuster and Nakajima, 2012; Sennrich et al., 2016). The scheme based on Byte-Pair Encoding (BPE) (Sennrich et al., 2016) is one of the most frequently used methods in current NMT researches. Following this trend, this paper focuses only on a method based on BPE to obtain a set of subword units and refers to Sennrich et al. (2016)'s method as SubW_{BPE} to distinguish it from others for clarity.

The following briefly describes the SubW_{BPE} procedure for building a set of subword units given a set of training data. SubW_{BPE} first splits the input sentences into character units and then combines the two frequently appearing consecutive (character or subword) units into one subword unit. SubW_{BPE} repeats this merge operation predefined m times. For splitting sentences into obtained subword units, we straightforwardly apply merge operations in the obtained order of the above procedure.

Here we revisit several interesting properties of SubW_{BPE}. For example, generated subword units become identical to the character units if and only if we set the number of merge operations m to zero ($m = 0$). All the subword units always recover and match the original word units if we set m to ∞ ($m = \infty$). These two properties imply that SubW_{BPE} naturally involves the methods using either character or word units in a single unified framework in terms of using subword units. Therefore, using SubW_{BPE}, we are not required to distinguish the methods of using character or word units since they are respectively just a special case of SubW_{BPE} with certain hyperparameters $m = 0$ and $m = \infty$. From this perspective, this paper does not explicitly distinguish among character, subword, and word units and treats all of them as subword units.

Another interesting property of SubW_{BPE} is that every subword unit obtained by $m = m'$ can be represented as a series of subword units obtained by $m = m''$ if $m' > m''$. This property is easily provable. SubW_{BPE} consists only of a *merge operation* of two consecutive subwords during the subword

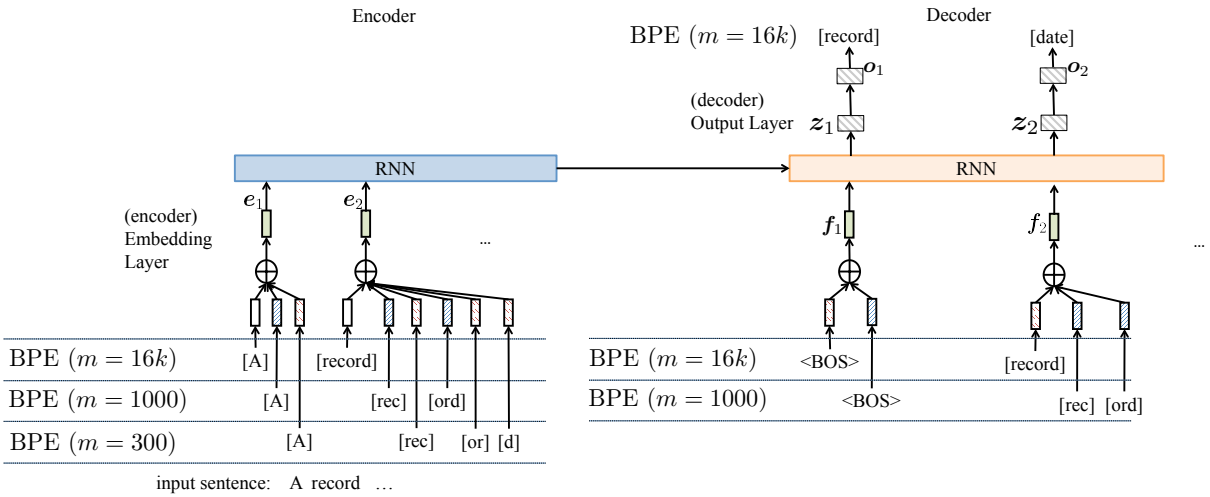


Figure 1: Overview of hierarchical subword features

unit construction, and thus, subword units with m are always identical to a subword with $m-1$ except the merged subwords as the m-th merge operation. Therefore, there is a relation where a subword unit obtained by a larger m is always a concatenation of (several) subword units obtained by a smaller m. This relation is always satisfied by any m' and m'' pairs, indicating a hierarchical *subword in subword* structure for all m from 1 to ∞.

Generally, the number of merge operations (m) is a hyperparameter that is empirically derived. Recent NMT researches usually set this m within a range from 1,000 to 100,000: apparently never less than 1,000 or over 100,000. This is because a set of subword units obtained with few merge operations resembles character units, and therefore the sentence becomes too long to process[1] On the other hand, since a set of subword units obtained with relatively large merge operations becomes nearly identical to the original word units, it lacks the advantage of using subword units. Hence, it is intuitively reasonable to set m in a range from 1,000 to 100,000.

3 Hierarchical Subword Features

Fig. 1 shows an overview of our proposed method. In it, we extend the encoder's and the decoder's embedding layers by modifying our model to work with several subwords units at once.

To formally explain our modification, we first introduce distinct Q encoder embedding matrices and R decoder embedding matrices. Let $\boldsymbol{E}_q$ represent the q-th encoder embedding matrix, where $q \in \{1, \ldots, Q\}$. Similarly, let $\boldsymbol{F}_r$ represent the r-th decoder embedding matrix, where $r \in \{1, \ldots, R\}$. Then we modify the operations to obtain encoder and decoder embedding vectors, which are shown respectively in Eqs. 2 and 3. For the encoder embedding vectors, we introduce the following operator:

$$\boldsymbol{e}_i = \sum_q \boldsymbol{E}_q \phi_q(\boldsymbol{x}_i), \tag{5}$$

where $\phi_q(\boldsymbol{x}_i)$ is a mapping function that returns a binary vector that corresponds to $\boldsymbol{x}_i$. For the decoder embedding vectors, we obtain $\boldsymbol{f}_j$ by the following equation:

$$\boldsymbol{f}_j = \sum_r \boldsymbol{F}_r \psi_r(\boldsymbol{y}_{j-1}), \tag{6}$$

where $\psi_r(\boldsymbol{y}_{j-1})$ is a mapping function that returns a binary vector that corresponds to previously estimated result $\boldsymbol{y}_{j-1}$. For example, if we get `record` as an estimation result of BPE(m =16k), mapping

[1] NMT lacks the ability to translate longer sentences (Koehn and Knowles, 2017).

	DE-EN		FR-EN	
	Tokens	Sentences	Tokens	Sentences
train	3,496,036	189,318	3,800,613	208,323
tst2012 (development)	30,900	1,700	21,653	1,124
tst2013 (test)	21,037	993	21,894	1,024
tst2014 (test)	24,950	1,305	24,950	1,305

Table 1: Cleaned corpora statistics on IWSLT datasets. Number of tokens is English side.

Configurations	Values	Configurations	Values
Embedding dimension D	512	Optimizer	SGD
Hidden dimension H	512	Initial learning rate	1.0
Attention dimension	512	Gradient clipping	5.0
Encoder layer	2	Dropout rate	0.3
Decoder layer	2	Mini-batch size	128 sent

Table 2: Model and optimization configurations

function $\psi_r(\boldsymbol{y}_{j-1})$ returns a binary vector that corresponds to the subwords with smaller merge operations (e.g., `rec` and `ord`). As we described in Section 2.2, every subword unit obtained with $m = m'$ can be represented as a series of subword units obtained with $m = m''$ if $m'' < m'$, and the series of subword units are uniquely determined. Thus, our modification can be interpreted as adding features of smaller subword units, which were derived from previously estimated output $\boldsymbol{y}_{j-1}$. We refer to our method that incorporates smaller subword features as *hierarchical subword features*.

Note that we use different embeddings for each hierarchical subword feature[2]. This means that NMT can learn different features for each hierarchical subword.

The hierarchical subword features slightly increased the number of model parameters (see Section 4.2.1 for more detail). However, increasing the memory requirement and runtime is limited, which allows us to run almost the same speed and memory requirement as the baseline system[3].

We can simultaneously use several subword features. In Fig. 1, we use both BPE (m =1k) and BPE (m=300) for the encoder side. By adding several subword features, the model can use more information with which we expect to improve the task performance.

4 Experiments

4.1 Setup

In this paper, we focused on from/to English (EN) to/from French (FR), German (DE) translations. We carried out our experiments on the IWSLT evaluation campaign dataset (Cettolo et al., 2012), which is based on a TED talk that has been translated into several languages. We used the IWSLT 2016 training set for the training models, tst2012 as the development set, and tst2013 and tst2014 as the test sets. For preprocessing, we used the Moses tokenizer[4] and the truecaser[5]. For the training set, we removed sentences over 50 words to clean the corpus. Table 1 shows the cleaned corpora statistics on the IWSLT datasets. To split words into subwords, we used the scripts[6] provided by Sennrich et al. (2016).

As an NMT framework, we used almost the same structure as Luong et al. (2015), except for our proposed embedding layers. Detailed NMT configurations are shown in Table 2. We set the initial

[2]We also tested using the same embeddings. However, we found through preliminary experiments that it is more effective to use different embeddings.

[3]The number of model parameters and size of the memory requirement largely depend on the vocabulary size. However, we use a smaller m (e.g., 1k or 300) for the hierarchical subword features, and since these features only affect the embedding layer, its effect on computational cost is limited.

[4]`https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl`

[5]`https://github.com/moses-smt/mosesdecoder/blob/master/scripts/recaser/truecase.perl`

[6]`https://github.com/rsennrich/subword-nmt`

	Encoder		Decoder		
	Unit	Feature	Unit	Feature	Description
(a)	16k	16k	16k	16k	baseline, conventional setting
(b)	1k	1k	16k	16k	smaller vocabulary setting for encoder side
(c)	300	300	16k	16k	smaller vocabulary setting for encoder side
(d)	16k	16k, 1k	16k	16k	(a) + Encoder side BPE 1k feature
(e)	16k	16k, 300	16k	16k	(a) + Encoder side BPE 300 feature
(f)	16k	16k, 1k, 300	16k	16k	(a) + Encoder side BPE 1k, 300 features
(g)	∞	∞	16k	16k	substituting Encoder side BPE 16k in (a) to word
(h)	∞	∞, 16k	16k	16k	(g) + Encoder side BPE 16k feature
(i)	∞	∞, 1k	16k	16k	(g) + Encoder side BPE 1k feature
(j)	∞	∞, 300	16k	16k	(g) + Encoder side BPE 300 feature
(k)	∞	∞, 1k, 300	16k	16k	(g) + Encoder side BPE 1k, 300 features
(l)	16k	16k	16k	16k, 1k	(a) + Decoder side BPE 1k feature
(m)	16k	16k	16k	16k, 300	(a) + Decoder side BPE 300 feature
(n)	16k	16k	16k	16k, 1k, 300	(a) + Decoder side BPE 1k, 300 features
(o)	16k	16k, 1k, 300	16k	16k, 1k, 300	(f) + (n)
(p)	∞	∞, 1k, 300	16k	16k, 1k, 300	(k) + (n)

Table 3: Compared experimental settings

learning rate to 1.0, but after 30 epochs we multiplied it by 0.8 for every epoch and continued training until 40 epochs. For decoding, we performed a beam search with a beam size of 20. To prevent the model from outputting short sentences, we applied the length normalization technique by dividing the negative log-likelihood by the sentence length (Cromieres et al., 2016; Morishita et al., 2017). As evaluation metrics, we used case-sensitive[7] BLEU scores (Papineni et al., 2002) using `multi-bleu.perl`[8].

To fix the experimental settings, we carried out a preliminary analysis to find the relation between the sentence length and the vocabulary size ($\simeq$ the number of BPE merge operations). Fig. 2 shows the results on the English sentences of the IWSLT 2016 German-English training set. When we reduce the vocabulary size, the average sentence length rapidly increases. Unfortunately, longer sentences require more computational cost and are time-consuming. Thus in our experiments, we set the baseline system's vocabulary size to 16,000, which balances the sentence length without affecting the advantage of the subwords.

The experimental settings are compared in Table 3. Our experiments answer the following questions:

- Does the hierarchical subword feature improve the model?
- Which part of the model should we use it? The encoder side, decoder side or both?
- How does it affect the translation results?

4.2 Results

Tables 4 and 5 show the experimental results with various word units and hierarchical subword features. All the scores are averages of four independently trained models. We used different parameter initialization and random seeds to train them. The score differences between the baseline system (a) and the proposed system are shown in the brackets.

(b) and (c) show some improvements using smaller subword units. However, as mentioned in Section 4.1 and shown in Fig. 2, these units are too small and lengthen the sentences. Thus computational time is also extended (see Table 6).

Then we added hierarchical subword features to the encoder side. From (d), (e), and (f), we confirmed that these features improved the model. System (f) uses both BPE (m =1k) and (m = 300)

[7] For a reference, we used a true-cased test set.

[8] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

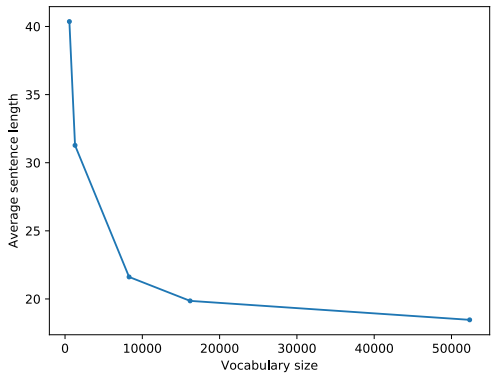


Figure 2: Relation between vocabulary amount ($\simeq$ number of BPE merge operations) and average sentence length in DE-EN training set (EN side). As vocabulary increases, length is shortened.

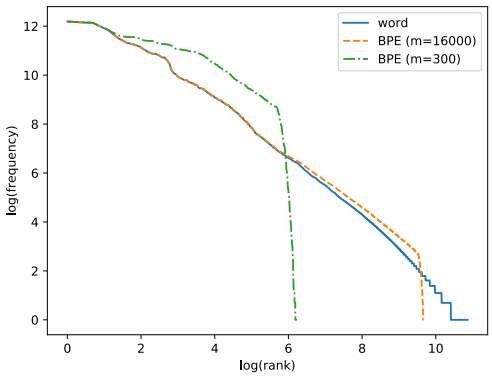


Figure 3: Relation between (sub-)word frequency and its rank in DE-EN training set (EN side). Both axes are in log scale.

	DE-EN						EN-DE					
	tst2012		tst2013		tst2014		tst2012		tst2013		tst2014	
(a)	31.64		33.68		29.18		26.24		28.22		24.17	
(b)	32.23	(+0.59)	34.24	(+0.56)	29.52	(+0.34)	26.18	(−0.06)	28.09	(−0.13)	24.46	(+0.29)
(c)	32.48	(+0.84)	34.60	(+0.92)	29.91	(+0.73)	26.39	(+0.15)	28.75	(+0.53)	24.70	(+0.53)
(d)	32.18	(+0.54)	34.49	(+0.81)	29.60	(+0.42)	26.50	(+0.26)	28.92	(+0.70)	24.74	(+0.57)
(e)	32.15	(+0.51)	34.83	(+1.16)	29.69	(+0.51)	26.78	(+0.54)	28.90	(+0.68)	24.79	(+0.61)
(f)	32.45	(+0.82)	34.67	(+0.99)	29.92	(+0.74)	27.07	(+0.83)	29.10	(+0.88)	24.88	(+0.70)
(g)	30.37	(−1.27)	32.58	(−1.10)	27.53	(−1.65)	25.86	(−0.38)	28.28	(+0.06)	24.28	(+0.11)
(h)	30.79	(−0.85)	33.14	(−0.54)	28.18	(−1.00)	26.06	(−0.18)	28.30	(+0.08)	24.31	(+0.14)
(i)	32.39	(+0.75)	34.76	(+1.09)	29.93	(+0.75)	26.61	(+0.37)	29.13	(+0.91)	24.73	(+0.56)
(j)	32.43	(+0.80)	34.63	(+0.95)	29.78	(+0.60)	26.90	(+0.66)	29.25	(+1.03)	25.15	(+0.98)
(k)	32.71	(+1.07)	34.71	(+1.03)	30.06	(+0.88)	26.99	(+0.75)	29.16	(+0.94)	**25.28**	(+1.11)
(l)	31.62	(−0.02)	33.60	(−0.08)	29.11	(−0.07)	26.11	(−0.13)	28.38	(+0.16)	24.22	(+0.05)
(m)	31.55	(−0.08)	33.65	(−0.03)	29.21	(+0.03)	26.17	(−0.07)	28.26	(+0.04)	24.15	(−0.03)
(n)	31.65	(+0.01)	33.51	(−0.17)	29.00	(−0.18)	25.93	(−0.31)	28.37	(+0.15)	24.13	(−0.04)
(o)	**32.84**	(+1.20)	34.76	(+1.09)	29.99	(+0.81)	**27.27**	(+1.03)	29.14	(+0.92)	24.97	(+0.79)
(p)	32.80	(+1.17)	**34.95**	(+1.27)	**30.11**	(+0.93)	27.11	(+0.87)	**29.64**	(+1.42)	25.20	(+1.03)

Table 4: BLEU scores on IWSLT German-to-English and English-to-German experiments. All scores are *averages* of four independently trained models.

as hierarchical subword features and shows more improvement than using a single feature. Since these models use BPE(m =16k) as a unit, the computational cost is almost the same as the baseline system (a).

In (d), (e), and (f), we used BPE (m =16k) as a unit, but it is more natural to use "word" (BPE ($m = \infty$)) as a unit. For the comparison, we did experiments from word to BPE translation without hierarchical subword features (g). As we expected, that step degraded the accuracy more than the baseline with BPE. This is because its vocabulary contains many rare words, and thus it is difficult to train these word embeddings well. However, by adding hierarchical subword features ((h) to (k)), the accuracy improved at the same level as the system with BPE units. Hierarchical subword features helped the model correctly encode the rare words and improved the accuracy.

For system (h), we saw no improvement, perhaps because large subword units (e.g., BPE (m =16k)) seem too similar to the word units, so it did not help the model very much. Fig. 3 shows the relation

	FR-EN						EN-FR					
	tst2012		tst2013		tst2014		tst2012		tst2013		tst2014	
(a)	42.35		39.61		36.79		43.65		40.09		37.32	
(b)	42.84	(+0.49)	39.53	(−0.08)	36.86	(+0.07)	43.72	(+0.07)	40.43	(+0.34)	37.69	(+0.37)
(c)	43.54	(+1.19)	39.42	(−0.19)	37.16	(+0.37)	43.86	(+0.21)	40.11	(+0.02)	37.54	(+0.22)
(d)	43.13	(+0.78)	39.93	(+0.32)	37.29	(+0.50)	44.59	(+0.94)	40.75	(+0.67)	37.82	(+0.51)
(e)	43.18	(+0.83)	39.42	(−0.19)	37.34	(+0.55)	44.76	(+1.11)	41.25	(+1.16)	38.29	(+0.97)
(f)	43.60	(+1.25)	40.01	(+0.40)	37.42	(+0.62)	45.07	(+1.42)	41.15	(+1.06)	38.50	(+1.18)
(g)	41.70	(−0.65)	38.36	(−1.26)	35.83	(−0.96)	43.13	(−0.52)	39.38	(−0.70)	36.54	(−0.78)
(h)	42.04	(−0.31)	38.56	(−1.05)	36.19	(−0.60)	42.88	(−0.77)	39.40	(−0.69)	36.54	(−0.78)
(i)	43.31	(+0.96)	40.13	(+0.52)	37.17	(+0.38)	44.84	(+1.19)	41.07	(+0.98)	38.11	(+0.79)
(j)	43.34	(+0.99)	40.27	(+0.65)	37.44	(+0.65)	45.01	(+1.36)	41.43	(+1.35)	**38.61**	(+1.29)
(k)	**43.82**	(+1.47)	**40.38**	(+0.77)	**37.94**	(+1.15)	45.32	(+1.67)	**41.58**	(+1.50)	38.51	(+1.20)
(l)	42.47	(+0.12)	39.19	(−0.42)	36.63	(−0.16)	43.86	(+0.21)	40.22	(+0.13)	37.25	(−0.07)
(m)	42.34	(−0.01)	39.52	(−0.09)	36.96	(+0.17)	43.79	(+0.14)	39.85	(−0.24)	37.14	(−0.18)
(n)	42.55	(+0.20)	39.09	(−0.52)	36.87	(+0.08)	43.54	(−0.11)	39.84	(−0.24)	37.20	(−0.12)
(o)	43.62	(+1.27)	40.12	(+0.51)	37.73	(+0.94)	45.22	(+1.57)	41.32	(+1.23)	37.98	(+0.66)
(p)	43.63	(+1.28)	39.93	(+0.31)	37.22	(+0.43)	**45.43**	(+1.78)	41.50	(+1.42)	38.34	(+1.03)

Table 5: BLEU scores on IWSLT French-to-English and English-to-French experiments. All scores are *averages* of four independently trained models.

	Model parameters	Training times for an epoch
(a)	38.9M	1050s
(b)	31.3M	1075s
(c)	30.9M	1101s
(d)	39.7M	981s
(e)	39.2M	992s
(f)	40.0M	1002s
(g)	88.8M	1053s
(h)	97.1M	1069s
(i)	89.5M	1029s
(j)	89.1M	976s
(k)	89.8M	1015s
(l)	39.7M	987s
(m)	39.2M	967s
(n)	39.9M	1004s
(o)	41.1M	1019s
(p)	90.8M	1083s

Table 6: Number of model parameters and required training times for an epoch (DE-EN). We used a single NVIDIA GeForce GTX 1080 Ti GPU for training. Required training time might vary due to server condition.

between the (sub-)word frequency and its rank on the English sentences of the German-English training set. The word's graph follows Zipf's law, and thus these word embeddings that appear a few times in the training set are relatively hard to train. The graph shows that BPE (m =16k) is almost the same as the word, because the frequently appearing subword pairs are connected until a subword to be a word. On the other hand, since each subword in BPE (m = 300) appears in the training set more frequently, its embedding layer is trained well. Perhaps the number of merge operations for the hierarchical subword features should be smaller than the one we commonly use. Our method can use both small and large subword features at once, thus we can cherry pick an advantage of small subword with maintaining the sentence length.

For systems (l), (m), and (n), we also added features to the decoder side, but we did not find as much improvement as the encoder side. A possible reason for this observation is that our method works

	DE-EN						EN-DE					
	tst2012		tst2013		tst2014		tst2012		tst2013		tst2014	
(a)	34.98		37.52		32.54		28.67		30.73		26.51	
(b)	35.03	(+0.05)	37.08	(−0.44)	31.97	(−0.57)	28.79	(+0.12)	30.67	(−0.06)	26.52	(+0.01)
(c)	35.27	(+0.29)	37.20	(−0.32)	32.60	(+0.06)	28.67	(+0.00)	30.94	(+0.21)	26.66	(+0.15)
(d)	35.44	(+0.46)	37.62	(+0.10)	**33.28**	(+0.74)	28.87	(+0.20)	31.21	(+0.48)	27.00	(+0.49)
(e)	35.02	(+0.04)	37.53	(+0.01)	32.99	(+0.45)	29.13	(+0.46)	31.12	(+0.39)	26.72	(+0.21)
(f)	35.46	(+0.48)	**37.88**	(+0.36)	33.07	(+0.53)	29.04	(+0.37)	30.99	(+0.26)	26.94	(+0.43)
(g)	32.49	(−2.49)	34.98	(−2.54)	29.24	(−3.30)	28.13	(−0.54)	30.59	(−0.14)	25.95	(−0.56)
(h)	34.59	(−0.39)	37.59	(+0.07)	32.23	(−0.31)	29.11	(+0.44)	31.20	(+0.47)	**27.01**	(+0.50)
(i)	35.08	(+0.10)	37.64	(+0.12)	32.62	(+0.08)	29.01	(+0.34)	31.23	(+0.50)	26.71	(+0.20)
(j)	35.12	(+0.14)	37.80	(+0.28)	32.69	(+0.15)	28.90	(+0.23)	31.46	(+0.73)	26.74	(+0.23)
(k)	35.08	(+0.10)	37.86	(+0.34)	32.69	(+0.15)	**29.26**	(+0.59)	31.17	(+0.44)	26.73	(+0.22)
(l)	34.90	(−0.08)	37.29	(−0.23)	32.71	(+0.17)	28.77	(+0.10)	30.89	(+0.16)	26.41	(−0.10)
(m)	34.62	(−0.36)	37.61	(+0.09)	32.77	(+0.23)	28.84	(+0.17)	30.35	(−0.38)	26.11	(−0.40)
(n)	34.92	(−0.06)	36.79	(−0.73)	32.10	(−0.44)	28.71	(+0.04)	31.21	(+0.48)	26.37	(−0.14)
(o)	**35.64**	(+0.66)	37.85	(+0.33)	32.99	(+0.45)	29.24	(+0.57)	31.12	(+0.39)	26.67	(+0.16)
(p)	35.25	(+0.27)	37.78	(+0.26)	32.60	(+0.06)	29.03	(+0.36)	**31.62**	(+0.89)	26.72	(+0.21)

Table 7: BLEU scores on IWSLT German-to-English and English-to-German experiments. All scores are *ensembles* of four independently trained models.

as a regularizer of the model and might degrade the decoder's language modeling ability: in other words, its ability to predict the next token given previous tokens.

Even though (o) and (p) systems showed slight improvements from (f) and (k), such insignificant improvements suggest that the usefulness of (o) and (p) is limited.

Our results suggest the following conclusions: (1) add hierarchical subword features to the encoder side but not to the decoder side and (2) use fewer merge operations, e.g., $m=300$ and $m=1$k.

4.2.1 Number of Parameters and Required Training Time

Table 6 shows the number of parameters and required training times per epoch. The number of model parameters significantly increases if we add *word*-level features ((g), (h), (i), (j), (k), and (g)). In contrast, adding *subword*-level features does not significantly increase the number of parameters.

We also checked the required training times for each setting. The training time with hierarchical subword features is comparable to the baseline NMT. These results revealed that our methods do not require further computational costs and can be easily applied to any existing systems.

4.2.2 Model Ensembling Results

Tables 7 and 8 show the BLEU scores of ensembling four independently trained models. Hierarchical subword features consistently improved the BLEU scores even for ensembling. This means that our method can be applied to highly tuned systems such as the one submitted to WMT.

4.2.3 Example of Improved Translation

Table 9 shows an example of an improved translation from French to English. The input includes a rare combination of two words, "Britney Spears", which is a proper noun that is hard to translate. Table 10 shows an example of how the words "Britney Spears" were segmented into subwords. These words have been split into small bits of subwords. The subwords are slightly different based on the number of merge operations.

On the one hand, the baseline system with BPE 16k for both the encoder and decoder side cannot correctly translate the two words. On the other hand, our proposed system with hierarchical subword features did correctly translate them. One significant reason is that the embedding layer of subwords with large merge operations is not trained well, as described in Section 4.2. In contrast, our proposed model can make use of both large and small features for correct translations of such rare words.

	FR-EN						EN-FR					
	tst2012		tst2013		tst2014		tst2012		tst2013		tst2014	
(a)	47.46		43.72		40.48		46.13		42.86		39.79	
(b)	46.07	(−1.39)	41.94	(−1.78)	39.62	(−0.86)	47.19	(+1.06)	**43.57**	(+0.71)	**40.87**	(+1.08)
(c)	46.47	(−0.99)	42.38	(−1.34)	39.51	(−0.97)	46.97	(+0.84)	42.94	(+0.08)	40.71	(+0.92)
(d)	**48.72**	(+1.26)	43.59	(−0.13)	40.98	(+0.50)	46.91	(+0.78)	43.02	(+0.16)	40.07	(+0.28)
(e)	48.16	(+0.70)	43.86	(+0.14)	**41.53**	(+1.05)	46.41	(+0.28)	42.66	(−0.20)	40.38	(+0.59)
(f)	48.18	(+0.72)	43.96	(+0.24)	41.10	(+0.62)	**47.40**	(+1.27)	43.30	(+0.44)	40.37	(+0.58)
(g)	47.61	(+0.15)	42.50	(−1.22)	40.00	(−0.48)	44.88	(−1.25)	41.85	(−1.01)	38.53	(−1.26)
(h)	45.95	(−1.51)	42.22	(−1.50)	39.54	(−0.94)	47.51	(+1.38)	43.54	(+0.68)	40.39	(+0.60)
(i)	48.52	(+1.06)	44.24	(+0.52)	40.67	(+0.19)	46.79	(+0.66)	43.32	(+0.46)	39.82	(+0.03)
(j)	48.61	(+1.15)	43.85	(+0.13)	41.17	(+0.69)	46.24	(+0.11)	42.85	(−0.01)	39.89	(+0.10)
(k)	48.49	(+1.03)	**44.33**	(+0.61)	40.92	(+0.44)	46.76	(+0.63)	42.90	(+0.04)	40.18	(+0.39)
(l)	48.47	(+1.01)	43.99	(+0.27)	40.43	(−0.05)	46.14	(+0.01)	42.86	(+0.00)	39.67	(−0.12)
(m)	48.11	(+0.65)	43.32	(−0.40)	40.68	(+0.20)	46.10	(−0.03)	42.49	(−0.37)	39.92	(+0.13)
(n)	47.94	(+0.48)	43.25	(−0.47)	40.83	(+0.35)	46.62	(+0.49)	42.77	(−0.09)	39.77	(−0.02)
(o)	48.38	(+0.92)	44.09	(+0.37)	40.90	(+0.42)	47.14	(+1.01)	42.98	(+0.12)	40.68	(+0.89)
(p)	48.24	(+0.78)	44.32	(+0.60)	41.00	(+0.52)	46.86	(+0.73)	42.76	(−0.10)	39.96	(+0.17)

Table 8: BLEU scores on IWSLT French-to-English and English-to-French experiments. All scores are *ensembles* of four independently trained models.

Input	J'ai répondu, "Je ne suis pas Britney Spears, mais tu peux peut-être me l'apprendre à moi.
Reference	I was like, "Well I'm not Britney Spears, but maybe you could teach me.
(a) Baseline	I said, "I'm not British Speney Spears, but maybe you can teach me.
(k) Proposed	I said, "I'm not Britney Spears, but maybe you can teach me.

Table 9: Example translation from French to English. Proposed method correctly translated rare words: "Britney Spears."

Merge operations	Subword segmentation
16k	Bri t ney S pe ars
1k	B ri t n e y S pe ars
300	B ri t n e y S p e ar s

Table 10: Example segmentation of "Britney Spears"

5 Related Work

Sennrich and Haddow (2016) added linguistic features to NMT embedding layer and achieved significant improvement. They modified the embedding layer to exploit several features, such as part-of-speech tags and syntactic dependency labels. This method resembles our work in terms of providing more information to the embedding layer. However, to use these linguistic features, since we need to prepare a morphological analyzer and/or a dependency parser, applicable languages are limited. In our method, BPE features are language independent and applicable to all languages.

Our method can also be interpreted as a regularizer to the embedding layer. Recently, Kudo (2018) proposed a subword regularization method that uses different subword segmentation based on its segmentation probability. This method increases NMT's robustness to noise and segmentation errors. Their experiments showed that the subword regularization method is significantly effective when testing accuracy with a different dataset than the training set. This means that their method is effective with open-domain settings. Our method might have a similar tendency, but we will investigate in future work. It might be interesting to verify the effect of combining our method and the subword regularization method.

Several studies incorporated the Recurrent Neural Network (RNN) or the Convolutional Neural Network (CNN) into the embedding layer for encoding character-, subword-, or morpheme-level informa-

tion (Luong and Manning, 2016; Lee et al., 2017). Comparing with these approaches, our work has a significant advantage in terms of the fast computation since our method increases negligible computational cost as clarified in Section 4.2.1.

6 Conclusion

In this paper, we focused on neural machine translation with subword units and experimented with hierarchical subword features. Our experiments confirmed that adding hierarchical subword features to the encoder side consistently improved the BLEU scores. Our method, which is quite simple and easy to adapt to any models that use subwords as a unit (such as text summarization and language modeling), has the potential to be a de-facto standard in the future. Our codes and scripts are available for reproduction and further experiments[9].

In this paper, we just experimented with an RNN-based NMT, even though recently several new NMT architectures have been proposed, including an attentional-based (Vaswani et al., 2017) and a CNN-based NMT (Gehring et al., 2017). As future work, we want to try our method with these new NMT models and see whether it is effective. Future work will also apply hierarchical subword features to a larger dataset.

Acknowledgements

We thank three anonymous reviewers for their insightful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In Proceedings of the 3rd International Conference on Learning Representations (ICLR).

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. WIT3: web inventory of transcribed and translated talks. In Proceedings of the 16th Annual Conference of the European Association for Machine Translation (EAMT), pages 261–268.

Fabien Cromieres, Chenhui Chu, Toshiaki Nakazawa, and Sadao Kurohashi. 2016. Kyoto university participation to WAT 2016. In Proceedings of the 3rd Workshop on Asian Translation (WAT), pages 166–174.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In Proceedings of the 34th International Conference on Machine Learning (ICML), pages 1243–1252.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016. Is neural machine translation ready for deployment? a case study on 30 translation directions. In Proceedings of the International Workshop on Spoken Language Translation (IWSLT).

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In Proceedings of the 1st Workshop on Neural Machine Translation (WNMT), pages 28–39.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL).

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. Transactions of the Association for Computational Linguistics (TACL), 5:365–378.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), pages 1054–1063.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1412–1421.

[9] https://github.com/nttcslab-nlp/hierarchical_subword

Makoto Morishita, Jun Suzuki, and Masaaki Nagata. 2017. NTT neural machine translation systems at WAT 2017. In Proceedings of the 4th Workshop on Asian Translation (WAT), pages 89–94.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), pages 311–318.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pages 5149–5152.

Rico Sennrich and Barry Haddow. 2016. Linguistic Input Features Improve Neural Machine Translation. In Proceedings of the 1st Conference on Machine Translation (WMT), pages 83–91.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), pages 1715–1725.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS), pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st Annual Conference on Neural Information Processing Systems (NIPS), pages 6000–6010.

Design Challenges in Named Entity Transliteration

Yuval Merhav*
Amazon Alexa AI
Cambridge, MA
merhavy@amazon.com

Stephen Ash*
Amazon AWS AI
Seattle, WA
ashstep@amazon.com

Abstract

We analyze some of the fundamental design challenges that impact the development of a multilingual state-of-the-art named entity transliteration system, including curating bilingual named entity datasets and evaluation of multiple transliteration methods. We empirically evaluate the transliteration task using the traditional weighted finite state transducer (WFST) approach against two neural approaches: the encoder-decoder recurrent neural network method and the recent, non-sequential Transformer method. In order to improve availability of bilingual named entity transliteration datasets, we release personal name bilingual dictionaries mined from Wikidata for English to Russian, Hebrew, Arabic, and Japanese Katakana. Our code and dictionaries are publicly available[1].

1 Introduction

Named entity transliteration is the process of converting a named entity from one language script to another, using the correct characters that represent the entity in the target language. It is an important component in many search and language understanding tasks, such as robust cross-language information retrieval (CLIR) and machine translation (MT), among others.

A possible simple transliteration approach is mapping every character (or sequence of characters) in the source language to its most common counterpart in the target language. However, spelling and pronunciation of many languages (e.g., English, Japanese) can be ambiguous and inconsistent (Fushimi et al., 1999). As a result, most transliteration systems are data driven and use context for disambiguation; *e.g.*, (Yan and Nivre, 2016; Haizhou et al., 2004; Ekbal et al., 2006).

While transliteration has been a long studied problem, some important aspects received little attention. There is not clear guidance that addresses a number of common design considerations faced when building a robust multilingual transliteration system, such as data representation and the huge gap in results depending on the language pairs and transliteration direction (Rosca and Breuel, 2016). Like many other NLP fields recently, *neural* transliteration systems have gained popularity. However, it is still unclear if neural systems consistently outperform traditional approaches and what architecture is ideal for this task. For example, in (Yan and Nivre, 2016) the authors applied a stack of convolutional layers and simple recurrent layer on top for English to Chinese transliteration, which achieved competitive results but still below a phrase-based statistical machine translation system. Other works show that bidirectional long short-term memory (LSTM) neural networks and the encoder-decoder architecture (Sutskever et al., 2014) achieve comparable results with WFST based n-gram models that are considered state-of-the-art (Rao et al., 2015; Yao and Zweig, 2015). However, these works only studied grapheme-to-phoneme (G2P) conversion from English to standard English pronunciation sets, such as the CMU pronunciation dictionary (Weide, 2014). Also, there is not empirical evidence comparing the neural encoder-decoder method with LSTM to the recently proposed Transformer (Vaswani et al., 2017) neural method on the

*Y. Merhav and S. Ash contributed equally to this work

[1] https://github.com/steveash/NETransliteration-COLING2018

Proceedings of the 27th International Conference on Computational Linguistics, pages 630–640
Santa Fe, New Mexico, USA, August 20-26, 2018.

transliteration task. The Transformer method uses a simple neural network architecture based solely on attention mechanisms. That motivated us to learn if it can produce strong results on transliteration as it did on translation.

The contributions in this paper are summarized as follows:

- We enumerate and experimentally evaluate a number of design considerations important to building a named entity transliteration system such as: handling infrequent or unique name tokens in training and test data, building a bilingual training corpus from Wikidata[2], and top-1 versus top-k result performance.
- We present empirical results comparing design choices across four different language/script pairs mined from Wikidata: English to Hebrew, English to Russian, English to Arabic, and English to Katakana. We are releasing each of the bilingual datasets and the particular train, development, and test splits to encourage future experimentation and benchmarking.
- We empirically evaluate the traditional Weighted Finite State Transducer (WFST) approach against two neural network-based approaches: the sequence-to-sequence encoder-decoder architecture and the recent Transformer approach based of self-attention.

The rest of this paper is organized as follows: first, we briefly describe the traditional and neural approaches to transliteration present in the literature; second, we describe data collection considerations and our multilingual datasets; lastly, we describe our experimental setup, present a number of empirical results, and provide guidance based on our experience and analysis.

2 Transliteration Approaches

2.1 Traditional Approaches

Early transliteration works utilized dictionaries and phonetic resources to learn probabilistic mapping rules between languages (Knight and Graehl, 1998; Stalls and Knight, 1998). Later works introduced more robust methods that do not require an intermediate phonemic mapping and can learn direct orthographical mapping between two languages given a bilingual dictionary (Haizhou et al., 2004). Traditionally, such works are modeled based on the common Grapheme-to-Phoneme (G2P) joint-sequence modeling technique. Given word-to-pronunciation examples, an initial alignment between corresponding grapheme (word) and phoneme (pronunciation) sequences is learned. Then, a language model is learned on the aligned tokens (see (Bisani and Ney, 2008) for a detailed description of joint-sequence models for G2P). The open source Phonetisaurus (Novak et al., 2012) has shown to achieve state-of-the-art scores on different G2P tasks (Thu et al., 2016). Phonetisaurus implements the common EM-driven sequence alignment algorithm, with a few constraints such as allowing only m-to-one and one-to-m alignments. It trains an n-gram language model from the aligned pairs, which is converted into a Weighted Finite-State Transducer (WFST). Decoding can then be done by extracting the shortest path through the phoneme lattice created via composition with the input word.

2.2 Neural Approaches

Recent work in Neural Machine Translation (NMT) has proposed a number of approaches to use neural networks in variable-length sequence-to-sequence tasks such as transliteration. The encoder-decoder architecture (Sutskever et al., 2014) is a recurrent neural network setup with two parts. An *encoder* is fed input tokens one at a time and encodes them into a hidden state vector. At the end of the input sequence, an end-of-sentence token is fed to signify the end of the encoding phase. Next, the hidden state output of the encoder is fed into the *decoder*. The decoder emits tokens and updated hidden states, which are recursively fed into itself, until there are no more output tokens to produce. An additional mechanism, *attention* (Bahdanau et al., 2014), allows the decoder to focus on different parts of the input sequence and capture long-range dependencies. More recently, the Transformer (Vaswani et al., 2017)

[2] http://wikidata.org

model was proposed, which avoids the need for sequential processing, relying only on self-attention. A benefit of this approach is there is no information bottleneck in the encoded hidden state vector as in the Encoder-Decoder approach. Additionally, because there is no longer a sequential recurrent network, model training can be better parallelized, decreasing model training time.

Previous work applied variations of the encoder-decoder approach to the name transliteration task (Rosca and Breuel, 2016) and grapheme-to-phoneme transduction (Rao et al., 2015), suggesting strong results on both. (Rosca and Breuel, 2016) reports that on English to Japanese Katakana transliteration a unidirectional encoder-decoder with 2 hidden layers using gated recurrent unit (GRU) as the underlying neural cell type achieves the best result: a word error rate (WER) of 0.50.

3 Data

The transliteration shared task, as part of the named entities workshop (NEWS)[3], has been a continuous effort of benchmarking different transliteration approaches and systems across different languages. The workshop released multiple multilingual datasets over the years. However, datasets from previous years are not publicly available and their license is restrictive. That motivated us to create new multilingual datasets based on Wikidata that will be publicly available and free for all.

Wikipedia has been widely used in transliteration works (e.g., (Irvine et al., 2010; Rosca and Breuel, 2016; Pasternack and Roth, 2009)). Wikidata is a central knowledge base in all languages for Wikipedia and other Wikimedia projects. Most Wikidata pages contain labels[4] in one or multiple languages. We automatically collected all "en" (English) labels where they pair with one of the following labels: "ja" (Japanese), "he" (Hebrew), "ar" (Arabic), and "ru" (Russian). We filtered out labels containing characters not belonging to their main script (e.g., English labels containing non-Latin characters). For Japanese, we only included labels with Katakana characters. For English tokens, we did *not* strip out diacritics[5] as these may carry useful information in the context of named entity transliteration. We did convert all characters to lowercase and strip some punctuation such as underscores, braces, exclamation marks, etc.

The first version of our dataset contained many types of entities, such as song and book titles. A quick analysis of the data revealed that many label pairs are translations and not transliterations. Consequently, we used the Wikidata "instance-of" property to only include labels of type "human". There are significantly more pages of this type than any other type on Wikidata, and they are less noisy than other types for the transliteration task. Since no direction specific information was used in data gathering, we evaluate performance on both directions (*e.g.* English $\rightarrow$ Arabic and Arabic $\rightarrow$ English).

We also report performance on two publicly available datasets: (1) The CMU pronunciation dictionary (Weide, 2014); and (2) The Arabic to English dataset extracted from Wikipedia titles in (Rosca and Breuel, 2016). The former is used mainly for evaluating G2P systems and not named entity transliteration, but the two tasks are related and many papers use it solely for evaluation.

4 Experimental Setup

4.1 Data Representation

Pronunciation of English names usually does not carry context across tokens. In the majority of names, the pronunciation of a last name is independent of the previous tokens (middle or first names). Hence, it makes sense to train a transliteration system on name pairs that consist of a single token on each side. Our Wikidata datasets contain many multi-token names. Consequently, for constructing the train and test examples, we learn the word alignments of multi-token names, which is a simple task since the majority of name pairs contain the same word order, and we are not dealing with CJK languages with no token boundaries. The majority of our Katakana names contain a middledot to separate the tokens. We throw away English to Katakana name pairs if the English name contains more than one token and the Katakana

[3] http://workshop.colips.org/news2018/

[4] A Wikidata label is the most common name that the item would be known by. It does not need to be unique, in that multiple items can have the same label.

[5] We did not remove any Latin script characters

Table 1: Dataset statistics. Note that "EN" refers to English labels in Wikidata, but in our setup it actually means Latin script. That is why the source alphabet size exceeds 100 for all the four Wikidata datasets. As expected, the "EN-RU" dataset has the largest Latin alphabet size (source alphabet size).

Dataset	Total Size	Training Set Size	Avg. Source Length	Avg. Target Length	Source Alphabet Size	Target Alphabet Size
WD-EN-RU	164,640	105,371	7.0	6.6	188	62
WD-EN-KA	98,820	63,246	7.0	4.8	170	105
WD-EN-AR	74,973	41,584	6.7	5.9	145	67
WD-EN-HE	50,039	32,036	6.7	5.6	135	57
Rosca and B., 2016	15,898	12,877	6.0	6.8	48	39
CMUdict	126,191	113,438	7.5	6.3	27	39

name has no middledots or spaces. It is worth noting that the majority of Russian names in Wikidata are written as "last, first" and not "first last" as most other languages.

After alignment, we construct our cross-validation splits with only single, unique name tokens. Splitting names into single token examples produces many duplicates. For example, every name that starts with "John" would produce a "John" example. For train, development, and test splits, we only include a single instance for each unique name token. When we encounter multiple possible target script transliterations for the same English token, we only include the most frequent target transliteration[6]. We found that including multiple transliteration targets instead of only the single most-frequent one, only impacts performance in a small way, and thus, we opted for the simpler approach.

We used a typical 64/16/20 cross-validation split of the single token data. We used 64% of the data for training, 16% for a development evaluation set to pick hyperparameters, and a 20% held-out test set for metric reporting in Section 5. Our curated datasets are being released publicly in two forms. We are releasing the original name phrases where each line contains two tab-separated columns with the English name phrase in the first and the target script name phrase in the second column. Additionally, we are releasing the particular 64/16/20 splits of the aligned single token data in order that future researchers can replicate and benchmark against our results. Table 1 provides more information about the aligned, single token versions of each dataset.

For evaluating performance on the Arabic to English dataset from (Rosca and Breuel, 2016), we used the same train, dev, and test splits used in the paper. For evaluating CMUdict performance, we used the same 90% train, 10% split used in (Novak et al., 2012). We used these datasets without making any changes.

4.2 Libraries

We evaluate the previously discussed approaches using the following libraries:

- `Phonetisaurus` library[7] (Novak et al., 2012), the traditional WFST approach to sequence to sequence transduction.
- `seq2seq` library[8], the encoder-decoder recurrent neural network approach (Luong et al., 2017) as implemented in the TensorFlow (Abadi et al., 2015) machine learning platform.
- `tensor2tensor` library[9], the reference implementation of the Transformer approach to neural sequence to sequence transduction tasks (Vaswani et al., 2017), which is implemented on top of TensorFlow.

[6] We resolve ties by picking one of the targets arbitrarily

[7] `https://github.com/AdolfVonKleist/Phonetisaurus`

[8] `https://github.com/tensorflow/nmt`

[9] `https://github.com/tensorflow/tensor2tensor`

For the experiments using Phonetisaurus, we used the default configuration with an 8-order MITLM (Hsu and Glass, 2008) language model. For the many-to-many alignment, we disallowed ϵ-transitions on the source side and allowed them on the target side (with the `-seq2_del` option). For the encoder-decoder experiments, we simply adapted the Seq2Seq tutorial (Luong et al., 2017) to use the library for our character-level transliteration task. We used an LSTM cell-type, 2 hidden layers, 128 units per layer, a dropout probability of 0.2, and default 'luong' attention mechanism. Lastly, for the Tensor2Tensor experiments, we used the default configuration but tested with both 64 units per layer and 128 units per layer. Using 128 units per layer improved WER for every language by $\approx$ 2-3 points, and thus we only report results with 128 units. All of the scripts used in these experiments along with the mined Wikidata datasets are publicly available[10].

4.3 Evaluation Metric

Since many names may have multiple correct transliterations, in most experiments, we report the Word Error Rate (WER) metric, which is based on the proportion of name tokens that were transliterated and exactly match the expected target script from the test set (ground truth). The lower the WER score, the better. We report WER scores as 1-best, 2-best, and 3-best, which reflects when the correct transliteration occurs in the top spot, or in one of the top 2 spots, or in one of the top 3 spots. Since we do not penalize bad transliterations, 2-best is always at least as accurate as 1-best, and 3-best is always at least as accurate as 1-best and 2-best. We feel it is important to report in this way, because in many cases, such as information retrieval, it is feasible to generate multiple possible transliterations and use all of them at search time to improve recall. In other cases, such as text-to-speech, the top-1 result is the only result that matters, because the system only has a single opportunity to provide the correct transliteration.

5 Experimental Results

Table 2 summarizes the results of each method on different datasets. The recent Tensor2Tensor Transformer architecture outperforms the WFST approach and the Seq2Seq approach on every language. However, it is worth noting that the training time using Tensor2Tensor is between 5-8 hours using an AWS p3.2xlarge[11] instance type, which has a Tesla V100 GPU. The Phonetisaurus training time only takes a couple of minutes on a typical dual-core Intel Core-i7 personal laptop. We did not expect the WER difference between T2T and Phonetisaurus to be that significant. Our hypothesis was that given the small number of characters in every language and small average input size, the n-gram based models would be hard to beat. NMT approaches have a clear advantage in handling long term dependencies, but we expected that the small input size of our named entity tokens implied few important long term dependencies. The CMUdict dataset is the only case in which Phonetisaurus (slightly) outperforms the neural Transformer approach. One explanation is that the CMUdict dataset is not that challenging, as can be seen by the difference in WER compared to all other datasets.

Another observation is the WER gap between the languages. Among the English to X tasks, Katakana has the worse WER, followed by Arabic that is only slightly behind Hebrew. Given that Arabic and Hebrew belong to the same language family, it is not surprising that their WER is similar. Russian has the best WER by a large margin. It also has the largest training data and it is the closest language to English among the four evaluated. We were interested to learn the impact of training size on the error rate. Figure 2 shows learning curves for various datasets. We used Phonetisaurus since it was the fastest to train. One interesting observation is that training size is not a big factor. When the number of training examples is equal across all languages, Russian still has the best WER by a large margin and Katakana the worst by a large margin. Another interesting observation is that we reach close to optimal performance with about 50% of the training data. For example, WER on WD-EN-RU (English to Russian) is 0.40 and 0.38 after training on 50K and 100K examples, respectively. CMUdict has the steepest reduction in WER as the number of training examples increases, with WER of 0.33 and 0.28 after training on 50K and 100K examples, respectively.

[10] https://github.com/steveash/NETransliteration-COLING2018

[11] https://aws.amazon.com/ec2/instance-types/p3/

Table 2: Word Error Rate (WER) using different methods for named entity transliteration. "WD" refers to our produced Wikidata datasets. "T2T" refers to the Tensor2Tensor system.

Task	Dataset	Method	1-best WER	2-best WER	3-best WER
English → Arabic	WD-EN-AR	T2T	**0.45**	**0.30**	**0.24**
		Seq2Seq	0.53	0.41	0.36
		Phonetisaurus	0.51	0.37	0.30
English → Hebrew	WD-EN-HE	T2T	**0.44**	**0.27**	**0.22**
		Seq2Seq	0.49	0.36	0.31
		Phonetisaurus	0.49	0.32	0.26
English → Katakana	WD-EN-KA	T2T	**0.51**	**0.35**	**0.29**
		Seq2Seq	0.60	0.48	0.43
		Phonetisaurus	0.57	0.43	0.36
English → Russian	WD-EN-RU	T2T	**0.35**	**0.22**	**0.17**
		Seq2Seq	0.40	0.29	0.25
		Phonetisaurus	0.38	0.24	0.19
Arabic → English	Rosca and B., 2016	T2T	**0.75**	**0.63**	**0.57**
		Seq2Seq	0.81	0.71	0.67
		Phonetisaurus	0.75	0.65	0.59
English → ARPAbet	CMUdict	T2T	0.29	0.16	0.11
		Seq2Seq	0.29	0.18	0.14
		Phonetisaurus	**0.27**	**0.14**	**0.10**

The seq2seq system consistently performed the worst. Based on design guidance in Neural Machine Translation literature (Sutskever et al., 2014), we experimented with feeding the source string in reverse order as this showed a significant improvement in encoder-decoder approaches. In our experiments, this did not meaningfully impact the WER. The resulting score was either identical or within one point for each of the language pairs that we tested.

5.1 Removing tokens that occur only once

One design question that we set out to answer is how to be handle unique name token pairs in dataset curation (e.g., the 'EN-KA' transliteration pair ["escalada", "エスカラーダ"] occurs only once in the data we collected from Wikidata). Name tokens, like other language tokens, follow a Zipfian distribution and thus there are likely to be many tokens that appear infrequently. On the one hand, tokens that occur only once may be erroneous, and it may be better to exclude them from the training and test sets. Taking the English to Hebrew dataset as an example, 77% of the name tokens only occur once. Our hypothesis was that given this distribution of token frequency, excluding them would likely remove more useful information than noise. Table 3 illustrates the impact of excluding single occurrence name tokens from both train and test against a baseline of training and testing on everything. All of these results use the Transformer neural approach with 128 units per layer and 2 hidden layers.

Comparing column 1 (baseline) and column 2 in Table 3 shows that excluding the infrequent tokens from test, results in a surprisingly high reduction in WER. If we also filter the tokens from train (as shown in column 4), we see an increase in WER. The worse WER is achieved when filtering the train set but testing on the full test set. In English to Hebrew (WD-EN-HE), filtering the train set but testing on the full set increases the WER by 7.0 points over the baseline. Industrial transliteration systems will contain dictionary look-ups for frequent named entity transliterations and employ automated methods, as evaluated here, to handle out-of-vocabulary cases. Thus, it stands to reason that a fair evaluation of automated methods should include infrequent names even in the presence of some noise. The results presented in other sections include unique tokens at both train and test time.

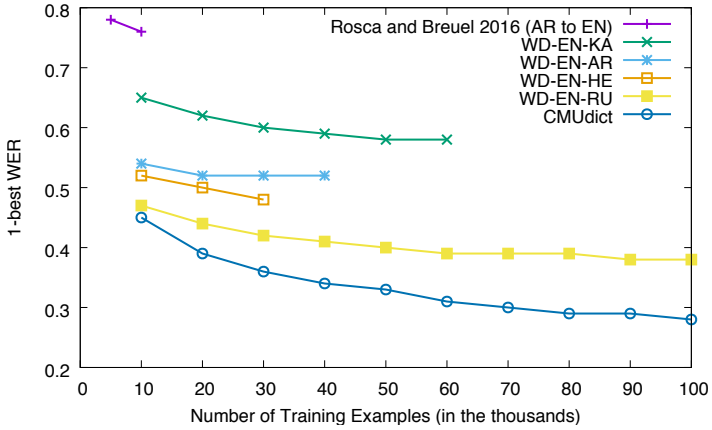


Figure 1: Phonetisaurus: Impact of the number of training examples on WER.

Table 3: T2T 1-Best Word Error Rate (WER) when filtering single occurrence name tokens from training and/or testing.

Dataset	**Full Train, Full Test (Baseline)**	**Full Train, Filtered Test**	**Filtered Train, Full Test**	**Filtered Train, Filtered Test**
WD-EN-AR	0.45	0.32	0.50	0.35
WD-EN-HE	0.43	0.31	0.50	0.35
WD-EN-RU	0.35	0.24	0.39	0.26
WD-EN-KA	0.50	0.36	0.56	0.41

An interesting question is why infrequent names in our data happen to be the hardest instances. One possible factor is that these names are less known to the public so there is no spelling convention to follow, leading to higher ambiguity. Wikidata labels are updated by many people from all over the world, hence name frequency can be thought as a proxy for annotator agreement. We also found that infrequent names in our data are longer on average, with a strong negative correlation between token frequency and token length. For example, the average length of all single occurrence English name tokens based on all of our test datasets is 7.1 characters, while the average length of all other English name tokens (at least two occurrences) is 6.4. Word-based evaluation metrics like we use here might be biased towards shorter names. Some prior works have used character-based metrics for transliteration, such as length-normalized edit distance (Irvine et al., 2010; Noeman and Madkour, 2010), which might be a good alternative (depending on the application).

5.2 English as source or target language

We performed experiments comparing the performance of modeling the transliteration problem as English $\rightarrow X$ versus $X \rightarrow$ English. Our hypothesis was that learning the transliteration with English as the source language was going to result in a lower word error rate due to the loss of information when going to languages like Hebrew or Arabic. Table 4 reports the differences in these two approaches. In every case using English as the source language results in much better word error rate, but the impact varies depending on the language. Hebrew and Arabic are impacted the most, and Russian is the least penalized. Previous works that focus on back-transliterating (recovering names of English origin) report similar findings (Irvine et al., 2010). In the error analysis section we provide some insights on why the reverse task is harder.

Table 4: T2T Word Error Rate (WER) when modeling as English to X versus X to English.

Dataset	Task	1-best WER	2-best WER	3-best WER
WD-EN-AR	English → Arabic	**0.45**	**0.30**	**0.24**
WD-EN-AR	Arabic → English	0.75	0.64	0.58
WD-EN-HE	English → Hebrew	**0.44**	**0.27**	**0.22**
WD-EN-HE	Hebrew → English	0.77	0.65	0.59
WD-EN-KA	English → Katakana	**0.51**	**0.35**	**0.29**
WD-EN-KA	Katakana → English	0.70	0.57	0.50
WD-EN-RU	English → Russian	**0.35**	**0.22**	**0.17**
WD-EN-RU	Russian → English	0.47	0.35	0.30

5.3 Error Analysis

Given the nature of Wikidata, our extracted datasets contain a diverse set of names from different origins, including many foreign names with different linguistic conventions. Names can be pronounced differently depending on origin and context can play a role as well. This explains why only outputting the top transliteration is usually not enough, as our results show. One example showing how hard the task can be is the Irish name "Domhnall", pronounced as DONAL, which none of the models transliterated correctly. The Hebrew, Arabic, and Russian models all included an 'm' in each of their 3-best outputs.

Figure 2 shows error examples in the various tasks. Vowel confusion is a common error. As expected, the Hebrew and Arabic models are affected by it the most. The problem becomes even worse when English is the target language. Hebrew and Arabic names in Wikidata, like in most of the web, do not include diacritical marks. This means that often English names with vowels are written in Arabic and Hebrew without the vowels. For example, "barak" is written as "brk" (in Hebrew letters) in modern Hebrew. When transliterating to English, the model often needs to recover the missing vowels. One of the figure examples is the name "Rashid"/"Rashed" that was transliterated incorrectly since the model failed to recover the letter 'i'/'e' that is not included in the original Arabic name. Vowel confusion is also a problem in Japanese, as the "ewan" example in the figure shows (the name Ewan MacGregor is quite known in Japan). The third best transliteration is "ユーアン", which differs from the correct transliteration only by the Katakana long vowel "ー". There seem to be different vowel variations in Katakana, such as long vs. short vowels. We found it often happens in the word final position. Having a special treatment to handle such cases could significantly boost accuracy. Some errors are more language specific. For example, in English to Russian the model often fails on the letter 'w' that can be ambiguous since the sound does not exist in Russian. The figure shows how the model made the same error in all three transliterations of the name "edwarda", replacing 'w' with the Russian letter 'В' ('v' sound), which is commonly used as a replacement.

Another interesting observation is that the models also fail on common names. This happens in all languages but more frequently occurs when English is the target language. For example, the Hebrew to English model transliterated "hillary" from Hebrew to English as "hilari", "hilarry", and "hillari". We believe this is a result of our problem setup. Since all our test names are unseen in training and every name only occurs once without making use of frequency info, the language model cannot learn that "hillary" is more common than "hilarry", for example, unless it is a sub-token of a longer name in training. Incorporating name frequency or large character-based n-gram language models learned from a large corpus (as in (Al-Onaizan and Knight, 2002; Irvine et al., 2010)) could possibly help such cases.

6 Future Directions

We will continue exploring best practices for multilingual transliteration. We are planning to evaluate design criteria for doing transliteration with CJK languages, where alignment is a more difficult problem. We are also interested to understand better the relationship between token frequency and transliteration

		Ground Truth	Prediction
Task	**Source**	**Target**	**3-best**
English → Russian	edwarda	эдуарда	эдварда эдворда эдварде
Russian → English	анатольевич	anatolyevich	anatolyevich anatolievitch anatoljević
English → Hebrew	hicham	הישאם	היצ'ם היחם היכהאם
Hebrew → English	הילרי	hillary	hilari hilarry hillari
English → Arabic	amos	عاموس	أموس اموس أمس
Arabic → English	راشد	rashid/rashed	rachd rašd -
English → Katakana	ewan	ユアン	エヴァン エワン ユーアン
Katakana → English	マツクイーン	mcqueen	mcquin mcquiin mcquine

Figure 2: Model errors across different languages

quality. Resources like Wikidata are extremely useful for curating large multilingual datasets, but some thought should be given to how we can reduce bias and noise with minimal effort, aiming to achieve quality that is comparable with manually annotated datasets. We believe that token frequency can be used as a proxy for annotator agreement and are planning to study this further. Finally, we are interested to study how well character-based NMT systems such as (Ling et al., 2015) perform on named entity transliteration.

7 Conclusions

We described a number of design considerations that one must address when building a robust, multilingual named entity transliteration system. We empirically demonstrated that the Tensor2Tensor neural Transformer method produces consistently strong results against the LSTM-based encoder-decoder neural method and WFST-based traditional method. We described a number of challenges and design choices when building bilingual dictionaries from mined knowledge bases, such as Wikidata. Our Wikidata curated datasets, including both the name phrase bilingual data and the aligned, single token datasets are being released for further experimentation and benchmarking.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Yaser Al-Onaizan and Kevin Knight. 2002. Translating named entities using monolingual and bilingual resources.

In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 400–408. Association for Computational Linguistics.

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

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech communication*, 50(5):434–451.

Asif Ekbal, Sudip Kumar Naskar, and Sivaji Bandyopadhyay. 2006. A modified joint source-channel model for transliteration. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 191–198. Association for Computational Linguistics.

Takao Fushimi, Matsuo Ijuin, Karalyn Patterson, and Itaru F Tatsumi. 1999. Consistency, frequency, and lexicality effects in naming japanese kanji. *Journal of Experimental Psychology: Human Perception and Performance*, 25(2):382.

Li Haizhou, Zhang Min, and Su Jian. 2004. A joint source-channel model for machine transliteration. In *Proceedings of the 42nd Annual Meeting on association for Computational Linguistics*, page 159. Association for Computational Linguistics.

Bo-June Hsu and James Glass. 2008. Iterative language model estimation: efficient data structure & algorithms. In *Ninth Annual Conference of the International Speech Communication Association*.

Ann Irvine, Chris Callison-Burch, and Alexandre Klementiev. 2010. Transliterating from all languages. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 100–110.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. 2017. Neural machine translation (seq2seq) tutorial. *https://github.com/tensorflow/nmt*.

Sara Noeman and Amgad Madkour. 2010. Language independent transliteration mining system using finite state automata framework. In *Proceedings of the 2010 Named Entities Workshop*, pages 57–61. Association for Computational Linguistics.

Josef R Novak, Nobuaki Minematsu, and Keikichi Hirose. 2012. Wfst-based grapheme-to-phoneme conversion: Open source tools for alignment, model-building and decoding. In *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, pages 45–49.

Jeff Pasternack and Dan Roth. 2009. Learning better transliterations. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 177–186. ACM.

Kanishka Rao, Fuchun Peng, Haşim Sak, and Françoise Beaufays. 2015. Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4225–4229. IEEE.

Mihaela Rosca and Thomas Breuel. 2016. Sequence-to-sequence neural network models for transliteration. *arXiv preprint arXiv:1610.09565*.

Bonnie Glover Stalls and Kevin Knight. 1998. Translating names and technical terms in arabic text. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, pages 34–41. Association for Computational Linguistics.

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

Ye Kyaw Thu, Win Pa Pa, Yoshinori Sagisaka, and Naoto Iwahashi. 2016. Comparison of grapheme-to-phoneme conversion methods on a myanmar pronunciation dictionary. In *Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016)*, pages 11–22.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

R Weide. 2014. The CMU pronunciation dictionary, release 0.7b.

Shao Yan and Joakim Nivre. 2016. Applying neural networks to english-chinese named entity transliteration. In *Sixth Named Entity Workshop, joint with 54th ACL*.

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. *arXiv preprint arXiv:1506.00196*.

A Comparison of Transformer and Recurrent Neural Networks on Multilingual Neural Machine Translation

Surafel M. Lakew
University of Trento
Fondazione Bruno Kessler
lakew@fbk.eu

Mauro Cettolo
Fondazione Bruno Kessler
cettolo@fbk.eu

Marcello Federico
MMT Srl, Trento
Fondazione Bruno Kessler
federico@fbk.eu

Abstract

Recently, neural machine translation (NMT) has been extended to multilinguality, that is to handle more than one translation direction with a single system. Multilingual NMT showed competitive performance against pure bilingual systems. Notably, in low-resource settings, it proved to work effectively and efficiently, thanks to shared representation space that is forced across languages and induces a sort of transfer-learning. Furthermore, multilingual NMT enables so-called zero-shot inference across language pairs never seen at training time. Despite the increasing interest in this framework, an in-depth analysis of what a multilingual NMT model is capable of and what it is not is still missing. Motivated by this, our work (i) provides a quantitative and comparative analysis of the translations produced by bilingual, multilingual and zero-shot systems; (ii) investigates the translation quality of two of the currently dominant neural architectures in MT, which are the Recurrent and the Transformer ones; and (iii) quantitatively explores how the closeness between languages influences the zero-shot translation. Our analysis leverages multiple professional post-edits of automatic translations by several different systems and focuses both on automatic standard metrics (BLEU and TER) and on widely used error categories, which are lexical, morphology, and word order errors.

1 Introduction

As witnessed by recent machine translation evaluation campaigns (IWSLT (Cettolo et al., 2017), WMT (Bojar et al., 2017)), in the past few years several model variants and training procedures have been proposed and tested in neural machine translation (NMT). NMT models were mostly employed in conventional single language-pair settings, where the training process exploits a parallel corpus from a source language to a target language, and the inference involves only those two languages in the same direction. However, there have also been attempts to incorporate multiple languages in the source (Luong et al., 2015a; Zoph and Knight, 2016; Lee et al., 2016), in the target (Dong et al., 2015), or in both sides like Firat et al. (2016) which combines a shared attention mechanism and multiple encoder-decoder layers. Regardless, the simple approach proposed in Johnson et al. (2016) and Ha et al. (2016) remains outstandingly effective: it relies on single "universal" encoder, decoder and attention modules, and manages multilinguality by introducing an artificial token at the beginning of the input sentence to specify the requested target language.

The current NMT state-of-the-art includes the use of recurrent neural networks, initially introduced in Sutskever et al. (2014; Cho et al. (2014), convolutional neural networks, proposed by Gehring et al. (2017), and so-called transformer neural networks, recently proposed by Vaswani et al. (2017). All of them implement an encoder-decoder architecture, suitable for sequence-to-sequence tasks like machine translation, and an attention mechanism (Bahdanau et al., 2014).

Besides specific studies focusing on new architectures and modules, like Luong et al. (2015b) that empirically evaluates different implementations of the attention mechanism, the comprehension of what a model can learn and the errors it makes has been drawing much attention of the research community, as

Proceedings of the 27th International Conference on Computational Linguistics, pages 641–652
Santa Fe, New Mexico, USA, August 20-26, 2018.

evidenced by the number of recent publications aiming at comparing the behavior of neural vs. phrase-based systems (Bentivogli et al., 2016; Toral and Sánchez-Cartagena, 2017; Bentivogli et al., 2018). However, understanding the capability of multilingual NMT models in general and zero-shot translation, in particular, has not been thoroughly analyzed yet. By taking the bilingual model as the reference, this work quantitatively analyzes the translation outputs of multilingual and zero-shot models, aiming at answering the following research questions:

- How do bilingual, multilingual, and zero-shot systems compare in terms of general translation quality? Is there any translation aspect better modeled by each specific system?
- How do Recurrent and Transformer architectures compare in terms of general translation quality? Is there any translation aspect better modeled by each specific system?
- What is the impact of using related languages data in training a zero-shot translation system for a given language pair?

To address these questions, we exploit the data collected in the IWSLT 2017 MT evaluation campaign (Cettolo et al., 2017) and made publicly available by the organizers. The campaign was the first featuring a multilingual shared MT task, spanning five languages (English, Dutch, German, Italian, and Romanian) and all their twenty possible translation directions. In addition to the official external single reference of the test sets, we can also rely on professional post-edits of the outputs of nine Romanian→Italian and of nine Dutch→German participants' systems. Hence, we exploit the availability of multiple Italian and German references to perform a thorough analysis for identifying, comparing and understanding the errors made by different neural system/architectures we are interested in; in particular, we consider pairs of both related languages (Romanian→Italian, Dutch→German) and unrelated languages (Romanian→German and Dutch→Italian). Furthermore, to explore the impact of using data from other related languages, French and Spanish are considered for training purposes as well, in particular for analyzing the behavior of zero-shot x→Italian systems, x representing any source language distant from Italian.

In the following sections, we begin with a brief review of related work on quantitative analysis of MT tasks (§2). Then, we give an overview of NMT (§3) with a contrast between the Recurrent (§3.1) and Transformer (§3.2) approaches, and a summary on multilingual and zero-shot translation (§3.3). Section (§4), describes the dataset and preprocessing pipeline (§4.1), qualitative evaluation data (§4.2), experimental setting (§4.3), models (§4.4) and the evaluation methods (§4.5). In Section (§5), we analyze the overall translation quality for related and unrelated language directions. Before the summary and conclusion, we will focus on lexical, morphological and word-order error types for the fine-grained analysis (§6).

2 Related Work

Recent trends in NMT evaluation show that post-editing helps to identify and address the weakness of systems (Bentivogli et al., 2018). Furthermore, the use of multiple post-edits in addition to the manual reference is gaining more and more ground (Bentivogli et al., 2016; Koehn and Knowles, 2017; Toral and Sánchez-Cartagena, 2017; Bentivogli et al., 2018). For our investigation, we follow the error analysis approach defined in Bentivogli et al. (2018), where multiple post-edits are exploited in order to quantify morphological, lexical, and word order errors, a simplified error classification with respect to that proposed in Vilar et al. (2006), which settles two additional classes, namely missing and extra words.

The first work that compares bilingual, multilingual, and zero-shot systems comes from the IWSLT 2017 evaluation campaign (Cettolo et al., 2017). The authors analyze the outputs of several systems through two human evaluation methods: direct assessment which focuses on the generic assessment of overall translation quality, and post-editing which directly measures the utility of a given MT output to translators. Post-edits are also exploited to run a fine-grained analysis of errors made by the systems. The main findings are that (i) a single multilingual system is an effective alternative to a bunch of bilingual systems, and that (ii) zero-shot translation is a viable solution even in low-resource settings. Motivated by

those outcomes, in this work we explore in more detail the practical feasibility of multilingual and zero-shot approaches. In particular, we explore the benefit of adding training data involving related languages in a zero-shot setting and, in that framework, we compare the behavior of state-of-the-art Transformer and Recurrent NMT models.

3 Neural Machine Translation

A standard state-of-the-art NMT system comprises of an encoder, a decoder and an attention mechanism, which are all trained with maximum likelihood in an end-to-end fashion (Bahdanau et al., 2014). Although there are different variants of the encoder-attention-decoder based approach, this work focuses on the Recurrent LSTM-based variant (Sutskever et al., 2014) and the Transformer model (Vaswani et al., 2017).

In both the Recurrent and Transformer approaches, the encoder is purposed to cipher a source sentence into hidden state vectors, whereas the decoder uses the last representation of the encoder to predict symbols in the target language. In a broad sense, the attention mechanism improves the prediction process by deciding which portion of the source sentence to emphasize at a time (Luong et al., 2015b). In the following two subsections, we briefly summarize the two architecture types.

3.1 Recurrent NMT

In this case, the source words are first mapped to vectors with which the encoder recurrent network is fed. When the <eos> (i.e. end of sentence) symbol is seen, the final time step initializes the decoder recurrent network. At each time step of the decoding, the attention mechanism is applied over the encoder hidden states and combined with the current hidden state of the decoder to predict the next target word. Then, the prediction is fed back to the decoder (i.e. input feeding), to predict the next word, until the <eos> symbol is generated (Sutskever et al., 2014; Cho et al., 2014).

3.2 Transformer NMT

The Transformer architecture works by relying on a self-attention (*intra-attention*) mechanism, removing all the recurrent operations that are found in the previous approach.In other words, the attention mechanism is repurposed to compute the latent space representation of both the encoder and the decoder sides. However, with the absence of recurrence, *positional-encoding* is added to the input and output embeddings. Similarly, as the time-step in a recurrent network, the positional information provides the Transformer network with the order of input and output sequences. In our work, we use the absolute positional encoding, but very recently the use of the relative positional information has been shown to improve performance (Shaw et al., 2018). The model is organized as a stack of encoder-decoder networks that works in an auto-regressive way, using the previously generated symbol as input for the next prediction. Both the decoder and encoder can be composed of uniform layers, each built of two sub-layers, i.e., a multi-head self-attention layer and a position wise feed-forward network (FFN) layer. The multi-head sub-layer enables the use of multiple attention functions with a similar cost of utilizing attention, while the FFN sub-layer is a fully connected network used to process the attention sublayers; as such, FFN applies two linear transformations on each position and a ReLU (Vaswani et al., 2017).

3.3 Multilingual NMT

Recent efforts in multilingual NMT using a single encoder-decoder and an attention mechanism (Johnson et al., 2016; Ha et al., 2016) have shown to improve translation performance with minimal complexity. Multilingual NMT models can be trained with parallel corpora of several language pairs in *many-to-one, one-to-many*, or *many-to-many* translation directions. The main idea that distinguishes multilingual NMT training and inference from a single language pair NMT is that in preprocessing, a *language-flag* is appended to the source side of each segment pair. The flag specifies the target language the source is paired with at training time. Moreover, it enables a zero-shot inference by directing the translation to a target language never seen at training time paired with the source. In addition to reducing training and maintenance complexity of several single language pair systems, the two main advantages of multilingual

	encoder-decoder type	embedding size	hidden units	encoder depth	decoder depth	batch size
Recurrent	LSTM	512	1024	4	4	128 seg
Transformer	Self-Attention	512	512	6	6	2048 tok

Table 1: Hyper-parameters used to train Recurrent and Transformer models, unless differently specified.

NMT is the performance gain for low-resource languages, and the possibility to perform a zero-shot translation.

However, the translations generated by multilingual and zero-shot systems have not been investigated in detail yet. This includes analyzing how the model behaves solely relying on a "language-flag" as a way to redirect the inference. Recent works have shown that the target language-flag is weaker in a low-resource language setting (Lakew et al., 2017). Thus, in addition to analyzing the behavior of bilingual and multilingual models, mainly, the zero-shot task requires a careful investigation.

4 Data and Experiments

4.1 Datasets and preprocessing

The experimental setting comprises seven languages; for each language pair, we use the ≈200,000 parallel sentences made publicly available by the IWSLT 2017 evaluation campaign (Cettolo et al., 2017), partitioned in training, development, and test sets. In the preprocessing pipeline, the raw data is first tokenized and cleaned by removing empty lines. Then, a shared byte pair encoding (BPE) model (Sennrich et al., 2015) is trained using the union of the source and target sides of the training data. The number of BPE segmentation rules is set to $8,000$, following the suggestion of Denkowski and Neubig (2017) for experiments in small training data condition. For the case of Transformer training, the internal sub-word segmentation (Wu et al., 2016) provided by the Tensor2Tensor library[1] is used. Note that prepending the "language-flag" on the source side of the corpus is specific to the multilingual and zero-shot models.

4.2 Evaluation data

For our investigation, we exploit the nine post-edits available from the IWSLT 2017 evaluation campaign. Post-editing regarded the bilingual, multilingual, and zero-shot runs of three different participants to the two tasks Dutch (Nl)→German (De) and Romanian (Ro)→Italian (It). Human evaluation was performed on a subset (603 sentences) of the nine runs, involving professional translators. Details on data preparation and the post-editing task can be found in Cettolo et al. (2017).

The translation directions we consider in this work are Nl/Ro→De and Nl/Ro→It. The choice of *German* and *Italian* as the target languages is motivated by (i) the availability of multiple post-edits for the fine-grained analysis and (ii) the possibility of varying the linguistic distance between the source and the target languages, allowing experimental configurations suitable to answer the research questions raised in Section §1.

As said, for Nl→De and Ro→It the human evaluation sets consist of 603 segments. Since post-editing involved only those two language pairs, for the other two directions considered in this work, namely Nl→It and Ro→De, we tried to exploit at best the available post-edits by looking for all and only the segment pairs of the Nl→It and Ro→De tst2017 sets for which the target side exactly matches (at least) one of the segment pairs of the Ro→It and Nl→De human evaluation sets. This way, we were able to find 478 matches on the Italian sides and 444 on the German sides, which therefore become the sizes of the human evaluation sets of Ro→De and Nl→It, respectively, for which we can re-use the available post-edits.

It is worth to note that in general, the post-edits from the evaluation campaign are not actual post-edits of MT outputs generated in our experiments, with some exceptions discussed later, therefore they should rather be considered as multiple external references.

[1] https://github.com/tensorflow/tensor2tensor

Model	#Directions	System description
NMT	1	*Four pure bilingual models for the Nl→De/It and Ro→De/It directions*
M-NMT	20	*Multilingual, trained on all directions in the set {En,De,Nl,It,Ro}*
ZST	16	*Zero-shot, trained as multilingual but removing also Nl↔De and It↔Ro data*
ZST_A	12	*Zero-shot, trained as ZST but removing also De↔Ro and Nl↔It data*
ZST_B	16	*Zero-shot, trained as ZST_A but adding En↔Fr/Es data*

Table 2: The training setting of 4*bilingual, 1*multilingual, and 3*zero-shot systems.

4.3 Training setting

Each of the three system types, namely bilingual, multilingual and zero-shot, is trained using both Recurrent and Transformer architectures, with the proper training data provided in the IWSLT 2017 evaluation campaign. Meta training parameters were set in a preliminary stage with the aim of maximizing the quality of each approach. Recurrent NMT experiments are carried out using the open source OpenNMT-py[2] (Klein et al., 2017), whereas the Transformer models are trained using the Tensor2Tensor toolkit. Hence, we took the precaution of selecting the optimal training and inference parameters for both approaches and toolkits. For instance, for our low-resource setting characterized by a high data sparsity, the dropout (Srivastava et al., 2014) is set to 0.3 (Gal and Ghahramani, 2016) in Recurrent models and to 0.2 in Transformer models to prevent over-fitting. Similarly, Adam (Kingma and Ba, 2014) optimizer with an initial learning rate of either 0.001 (RNN) or 0.2 (Transformer) is used. If the perplexity does not decrease on the validation set or the number of epochs is above 7, a learning rate decay of 0.7 is applied in the Recurrent case. For the Transformer case, the learning rate is increased linearly in the early stages (*warmup_training_steps*=16000); after that, it is decreased with an inverse square root of training step (Vaswani et al., 2017). Table 1 summarizes the list of hyper-parameters.

4.4 Models

To address the research questions listed in Section §1, we train five types of models using either the Recurrent or the Transformer approaches. All models are trained up to convergence, eventually the best performing checkpoint on the dev set is selected. Table 2 summarizes the systems tested in our experiments. As references, we consider four bilingual systems (in short NMT) trained on the following directions: Nl→De/It and Ro→De/It. The first term of comparison is a many-to-many multilingual system (in short M-NMT) trained in all directions in the set {En,De,Nl,It,Ro}. Then, we test zero-shot translation (ZST) between related languages, namely Nl→De and Ro→It, by training a multilingual NMT without any data for these language pairs. We also test zero-shot translation between unrelated languages (ZST_A), namely Ro→De and Nl→It, by excluding parallel data between these languages. Finally, for the same unrelated zero-shot directions we also train multi-lingual systems (ZST_B) that include data related to Romanian and Italian, namely En↔Fr/Es.

4.5 Evaluation methods

Systems are compared in terms of BLEU (Papineni et al., 2002) (as implemented in *multi-bleu.perl* [3]) and TER (Snover et al., 2006) scores, on the single references of the official IWSLT test sets.
In addition, two TER-based scores are reported, namely the multiple-reference TER (mTER) and a lemma-based TER (lmmTER), which are instead computed on the nine post-edits of the IWSLT 2017 human evaluation set. In mTER, TER is computed by counting, for each segment of the MT output, the minimum number of edits across all the references and dividing by the average length of references. lmmTER is computed similarly to mTER but looking for matches at the lemma level instead of surface forms. Significance tests for all scores are reported using Multeval (Clark et al., 2011) tool.
Systems are also compared in terms of three well known and widely used error categories, that is lexical, morphological, and word order errors, exploiting TER and post-edits as follows. First, the MT outputs

[2]https://github.com/OpenNMT/OpenNMT-py
[3]A script from the Moses SMT toolkithttp://www.statmt.org/moses

Direction	System	Recurrent				Transformer			
		BLEU	TER	mTER	lmmTER	BLEU	TER	mTER	lmmTER
Nl→De	NMT	18.05	64.61	23.70	20.60	**18.37**	**63.74**	27.95	23.86
	M-NMT	17.79	66.18	21.75	18.28	↑**19.95**	**61.90**	23.62	20.05
	ZST	17.06	65.73	26.35	22.29	↑**19.13**	**62.69**	**25.19**	**21.53**
Ro→It	NMT	22.16	59.35	22.99	20.39	**22.48**	**57.34**	26.60	23.36
	M-NMT	21.69	59.50	21.12	18.46	↑**22.12**	**57.51**	25.05	21.57
	ZST	18.72	62.08	29.66	26.15	↑**21.29**	**59.08**	**26.93**	**23.33**

Table 3: Automatic scores on tasks involving related languages. BLEU and TER are computed on test2017, while mTER and lmmTER are reported for human evaluation sets. Best scores of the Transformer model against the Recurrent are highlighted in bold, whereas arrow ↑ indicates statistically significant differences ($p < 0.05$).

and the corresponding post-edits are lemmatized and POS-tagged; for that, we used ParZu (Sennrich et al., 2013) for German and TreeTagger (Schmid, 1994) for Italian. Then, the lemmatized outputs are evaluated against the corresponding post-edits via a variant of the *tercom* implementation[4] of TER: in addition to computing TER, the tool provides complete information about matching lemmas, as well as shift (matches after displacements), insertion, deletion, and substitution operations. Since for each lemma the tool keeps track of the corresponding original word form and POS tag, we are able to measure the number of errors falling in the three error categories, following the scheme described in detail in Bentivogli et al. (2018).

5 Translation Analysis

5.1 Related languages

First, we compare the bilingual (NMT), multilingual (M-NMT), and zero-shot (ZST) systems on the two tasks *Nl→De* and *Ro→It*, implemented as either Recurrent or Transformer networks, in terms of automatic metrics. As stated above, BLEU and TER exploit the official external reference of the whole test sets, while mTER and lmmTER are utilize the multiple post-edits of the (smaller) IWSLT human evaluation test set. Scores are given in Table 3.

Looking at the BLEU/TER scores, it is evident that Transformer performs better in all the three model variants. In particular, for the multilingual and the zero-shot models, the gain is statistically significant. On the contrary, the mTER and lmmTER scores are better for the Recurrent architecture; in this case, the outcome is misleading since the nine post-edits include those generated by correcting the outputs of the three Recurrent systems. As such, the translations of the Recurrent systems are rewarded over the translations produced by the Transformer systems, thus making the comparison not fair.

As far as the models are compared, the bilingual one is the best in three out of four cases, the exception being the Transformer/Nl→De. Nonetheless, it is worth to note the good performance of the multilingual model in terms of mTER and lmmTER. This result holds true in both Recurrent and Transformer approaches, regardless of the BLEU score. We hypothesize that the main reason behind this is the higher number of linguistic phenomena observed in training, thanks to the use of data from multiple languages, which makes the multilingual models more *robust* than the bilingual models.

5.2 Unrelated languages

In unrelated language directions, our experimental setting is aimed at evaluating the impact of source *language-relatedness* with the target. Particularly, we focus on the zero-shot setup given its intrinsic difficulty, by taking the bilingual systems as references. Table 4 provides BLEU and TER based scores for the Ro→De and Nl→It directions.

Concerning the ZST_A training condition, in one case (Recurrent Ro→De) it outstandingly allows to outperform the pure bilingual system, while in the other cases there is no significant difference between

[4]Available at `wit3.fbk.eu/show.php?release=2016-02&page=subjeval`

Direction	System	Recurrent				Transformer			
		BLEU	TER	mTER	lmmTER	BLEU	TER	mTER	lmmTER
Ro→De	NMT	13.99	72.70	61.82	54.61	$^{\uparrow}$**16.52**	**66.71**	**55.68**	**48.44**
	ZST_A	14.93	69.38	58.26	51.08	$^{\uparrow}$**16.46**	**66.88**	**54.72**	**48.25**
	ZST_B	14.75	69.29	58.26	51.37	$^{\uparrow}$**16.55**	**67.18**	**55.29**	**48.03**
Nl→It	NMT	18.88	63.79	58.79	52.16	$^{\uparrow}$**20.22**	**60.88**	**55.52**	**48.56**
	ZST_A	18.77	62.97	58.80	51.32	$^{\uparrow}$**19.80**	**60.24**	**54.06**	**47.16**
	ZST_B	18.87	62.40	57.34	50.17	$^{\uparrow}$**20.61**	**59.41**	**53.04**	**46.17**

Table 4: Evaluation results for the unrelated language directions. BLEU and TER scores are computed with single references, while mTER and lmmTER are computed with nine post-edits. Best scores of the Transformer over the corresponding Recurrent architectures are highlighted in bold, whereas arrow $^{\uparrow}$ indicates statistically significant differences ($p < 0.05$).

ZST_A and NMT, proving once again that zero-shot translation built on the "language-flag" of M-NMT is really effective (Johnson et al., 2016): in fact, at most a slight performance degradation is recorded as the number of pairs used in training decreases (Lakew et al., 2017). Although gains are rather limited, adding training data involving Romance target languages (French and Spanish, ZST_B) close to Italian impacts as hoped: ZST_B scores are in general better than both NMT and ZST_A in Nl→It, while they do not degrade with respect to ZST_A in Ro→De.
Similarly to what is observed for related pairs (Table 3), the Transformer architecture shows definitely higher quality than the RNN one, confirming the capability of the approach to infer unseen directions.
The overall outcomes from Tables 3 and 4 are: (i) multilingual systems have the potential to effectively model the translation either in zero-shot or non zero-shot conditions; (ii) zero-shot translation is a viable option to enable translation without training samples; (iii) the Transformer is the best performing approach, particularly in the zero-shot directions.
The next section is devoted to a fine-grained analysis of errors made by the various systems at hand, with the aim of assessing the outcomes based on automatic metrics.

6 Fine-grained Analysis

Following the error classification defined in Section 4.5, now we focus on lexical, morphological, and reordering error distributions to characterize the behavior of the three types of models and the two sequence-to-sequence learning approaches considered in this work.
As anticipated in the previous section, it is expected that scores computed with reference to post-edits penalize Transformer over Recurrent systems because the outputs of the latter were post-edited, but not those of the former. We try to mitigate this bias by relying on the availability of multiple post-edits which likely allows to better match the Transformer runs than having a single reference would do. For the fine-grained analysis, we use instead the expedient of computing error distributions that are normalized with respect to the error counts observed in a bilingual reference system. In the next two sections, the fine-grained analysis is reported for related and unrelated languages pairs, consecutively.

6.1 Related languages

Table 5 provides the distribution over the error types by the bilingual (NMT), multi-lingual (M-NMT), and zero-shot (ZST) models, implemented either with Recurrent or Transformer architectures, for the Nl→De translation direction. We also report, for each error type and M-NMT and ZST system, the observed relative difference of errors with respect to the bilingual reference model (NMT).
Considering each error category, we observe the same general trend for all systems: the lexical errors represent by far the most frequent category (76-77%), followed by morphology (15-16%) and reordering (3-6%) errors; cases of words whose morphology and positioning are both wrong, represent about 1-2% of the total errors. Beyond the similar error distributions, it is worth to note the variation of errors made by M-NMT and ZST models with respect to those of the NMT model: for the Recurrent

Nl→De	Recurrent					Transformer				
	NMT	M-NMT	Δ_{NMT}	ZST	Δ_{NMT}	NMT	M-NMT	Δ_{NMT}	ZST	Δ_{NMT}
Lexical	77.29	69.65	-7.64	83.73	+6.44	76.47	64.83	-11.64	69.53	-6.94
Morph	15.41	16.51	+1.10	19.1	+3.69	15.70	13.96	-1.74	14.13	-1.57
Reordering	5.53	3.14	-2.39	5.41	-0.12	6.20	4.97	-1.23	5.41	-0.79
Morph. & Reo.	1.76	1.02	-0.74	1.61	-0.15	1.63	1.36	-0.27	1.53	-0.10
Total	100	90.31	-9.69	109.84	+9.84	100	85.12	**-14.88**	90.6	**-9.40**

Table 5: Distribution of lexical, morphological, and reordering error types from the two MT approaches. Reported values are normalized with respect to the total error count of the respective bilingual reference model (NMT). Δ_{NMT} are variations with respect to the bilingual reference models (NMT).

Ro→It	Recurrent					Transformer				
	NMT	M-NMT	Δ_{NMT}	ZST	Δ_{NMT}	NMT	M-NMT	Δ_{NMT}	ZST	Δ_{NMT}
Lexical	80.63	73.81	-6.82	102.79	+22.16	81.97	76.01	-5.96	84.12	+2.15
Morph	12.33	12.86	+0.53	16.00	+3.67	11.49	11.79	+0.30	12.44	+0.95
Reordering	5.74	3.71	-2.03	6.09	+0.35	5.35	4.64	-0.71	4.81	-0.54
Morph. & Reo.	1.30	1.15	-0.15	2.18	+0.88	1.19	1.09	-0.10	1.09	-0.10
Total	100	91.54	**-8.46**	127.07	+27.07	100	93.52	-6.48	102.45	**+2.45**

Table 6: Distribution of the error types in the Ro→It direction for the Recurrent and Transformer approaches. From the variation of errors that compare M-NMT and ZST models with the bilingual reference (NMT), a larger margin of error is observed in case of Transformer ZST model.

Ro→It	Recurrent					Transformer				
	NMT	ZST_A	Δ_{NMT}	ZST_B	Δ_{NMT}	NMT	ZST_A	Δ_{NMT}	ZST_B	Δ_{NMT}
Lexical	80.63	108.27	+27.64	100.31	+19.68	81.97	82.11	+0.14	76.76	-5.21
Morph	12.33	17.11	+4.78	17.23	+4.90	11.49	13.09	+1.60	11.59	+0.10
Reordering	5.74	6.20	+0.46	6.16	+0.42	5.35	5.18	-0.17	5.59	+0.24
Morph. & Reo.	1.30	2.22	+0.92	2.30	+1.00	1.19	1.16	-0.03	1.02	-0.17
Total	100	133.81	+33.81	126	+26.00	100	101.53	**+1.53**	94.96	**-5.04**

Table 7: Error distribution of ZST_A and ZST_B models for the Recurrent and Transformer variants. Transformer achieves the highest error reduction, particularly in the ZST_B model setting.

architecture, there is a decrease of 9.69 and an increase of 9.84 points, respectively. On the contrary, the Transformer architecture yields improvements for both models: total errors reduce by 14.88 and 9.40 points, respectively. The result for the Transformer ZST system is particularly valuable since the average error reduction comes from remarkable improvements across all error categories.

For the Ro→It direction, results are given in Table 6. Although to a different extent, we observe a picture similar to that of Nl→De discussed above: lexical errors is the type of error committed to a greater extent, multilingual models outperform their bilingual correspondents (more for the Recurrent than for the Transformer models), and ZST is competitive with bilingual NMT only if the Transformer architecture is adopted.

Training under the zero-shot conditions ZST_A and ZST_B assume less training data available and permit to measure the impact of introducing additional parallel data from related languages. We considered training conditions ZST_A and ZST_B here to perform Ro→It zero shot translation and report the outcomes in Table 7.

Results show error counts for each condition normalized with respect to the corresponding bilingual reference models (NMT). The most interesting aspect comes from the fact that global variations in the normalized error counts of the zero-shot translation can be here associated with the relatedness and variety of languages in the training data. As recently reported (Lakew et al., 2017), zero-shot performance

Ro→De	Recurrent					Transformer				
	NMT	ZST_A	Δ_{NMT}	ZST_B	Δ_{NMT}	NMT	ZST_A	Δ_{NMT}	ZST_B	Δ_{NMT}
Lexical	79.18	74.42	-4.76	74.09	-5.09	79.21	79.11	-0.10	78.52	-0.69
Morph	9.91	10.35	+0.44	10.07	0.16	9.92	10.05	+0.13	10.87	+0.95
Reordering	7.33	6.16	-1.17	6.16	-1.17	7.19	6.88	-0.31	7.22	+0.03
Morph. & Reo.	3.58	3.47	-0.11	3.47	-0.11	3.68	3.52	-0.16	3.60	-0.08
Total	100	94.4	**-5.60**	93.79	**-6.21**	100	99.55	-0.45	100.21	+0.21

Table 8: Error distribution of the bilingual (NMT), ZST_A and ZST_B model runs for the unrelated Ro→De direction. The Transformer moder shows the smallest sensitivity to the change in the number of training language pairs.

Nl→It	Recurrent					Transformer				
	NMT	ZST_A	Δ_{NMT}	ZST_B	Δ_{NMT}	NMT	ZST_A	Δ_{NMT}	ZST_B	Δ_{NMT}
Lexical	81.08	80.7	-0.38	78.79	-2.29	81.15	79.36	-1.79	77.48	-3.67
Morph	8.47	9.03	+0.56	8.38	-0.09	9.01	9.2	+0.19	9.03	+0.02
Reordering	7.78	6.63	-1.15	6.32	-1.46	7.51	6.74	-0.77	6.51	-1.00
Morph & Reo	2.67	2.54	-0.13	2.38	-0.29	2.34	2.41	+0.07	2.45	+0.11
Total	100	98.89	-1.11	95.86	-4.14	100	97.71	**-2.29**	95.47	**-4.53**

Table 9: Error distribution of the bilingual (NMT), ZST_A and ZST_B model runs. Δ_{NMT} shows the relative change in the error distribution of the zero-shot models with respect to the bilingual reference models.

of Recurrent models in a low resource setting seems highly associated with the number of languages provided in the training data. This is also confirmed by comparing performance of Recurrent models across the ZST (Table 6), ZST_A and ZST_B conditions. In particular, variations from the bilingual reference model, show significant degradation when some language directions are removed (from +27.07 to +33.81) and a significant improvement when two related languages are added (from +33.81 to +26.00). Remarkably, the Transformer zero-shot model seems less sensitive to the removal or addition of languages: actually a slight improvement is observed after removing Nl→It and De→Ro (ZST_A), i.e., from +2.45 to +1.53, followed by a large improvement when En→Fr/Es (ZST_B) are added, i.e. from +1.53 to -5.04. Notice that the latter results outperform the bilingual model. Overall, across all experiments, we see slight changes in the distribution of errors types. On the other hand, increases or drops of specific error types with respect to the bilingual reference model show sharper differences across the different conditions. For instance, the best performing Transformer model (ZST_B in Table 7) seems to gain over the reference bilingual systems only in terms of lexical errors (-5.21). The zero-shot Transformer model trained under the ZST condition (Table 6) although globally worse than the bilingual reference, seems instead slightly better than the reference concerning reordering error (-0.54), which account for 5.35% of the total number of errors.

6.2 Unrelated languages

In our second scenario, we evaluate the relative changes in the error distribution for the unrelated language directions (Ro→De and Nl→It). This section complements the translation results reported in Table 4, analyzing the runs from the ZST_A and ZST_B models in a different manner.

In the Ro→De unrelated direction (Table 8), the Recurrent model shows a reduction in the error rate of 5.60 points (ZST_A) and 6.21 points (ZST_B) with respect to the bilingual (NMT) reference model, while for the Transformer no significant differences are observed. These results confirm what observed in the automatic evaluation on the reference translations (Table 4). The gain observed by the Recurrent model on the ZST_B condition is mainly on lexical (-4.76 points) and reordering errors (-1.17 points) is probably due to the poor performance of its bilingual counterpart.

As far as the the Nl→It unrelated direction (Table 9) is concerned, both Recurrent and Transformer

ZST models show to reduce the error counts over the bilingual reference model. Actually, a similar trend occurs in Ro→De (Table 8), but with a relatively higher error reduction in case of the Transformer model. In particular, the Transformer model shows reductions of -2.29 points for ZST_A and -4.53 for ZST_B, whereas for the Recurrent model the improvements are slightly lower, namely -1.11 (ZST_A) and -4.14 (ZST_B) points. Remarkably, both the Recurrent and Transformer models benefit from additional training data related to Italian (compare ZST_A and ZST_B).

In conclusion, we observe that error counts of the zero-shot models in unrelated directions can increase (Table 8) when compared to the bilingual model. However, in the related language direction the most interesting aspect is observed with the discount of error in the Nl→It direction (Table 9). In particular, the ZST_B zero-shot model showed $>2.0\%$ error reduction over the ZST_A model. This gain is directly related to the newly introduced training data (i.e., English↔French/Spanish) in case of ZST_B.

Summary and Conclusions

In this work, we showed how bilingual, multilingual, and zero-shot models perform in terms of overall translation quality, as well as the errors types produced by each system. Our analysis compared Recurrent models with the recently introduced Transformer architecture. Furthermore, we explored the impact of grouping related languages for a zero-shot translation task. In order to make the overall evaluation more sound, BLEU and TER scores were complemented with mTER and lmmTER, leveraging multiple professional post-edits. Our investigation on the translation quality and the results of the fine-grained analysis shows that:

- Multilingual models consistently outperform bilingual models with respect to all considered error types, i.e., lexical, morphological, and reordering.
- The Transformer approach delivers the best performing multilingual models, with a larger gain over corresponding bilingual models than observed with RNNs.
- Multilingual models between related languages achieve the best performance scores and relative gains over corresponding bilingual models.
- When comparing zero-shot and bilingual models, relatedness of the source and target languages does not play a crucial role.
- The Transformer model delivers the best quality in all considered zero-shot condition and translation directions.

Our fine-grained analysis looking at three types of errors (lexical, reordering, morphology) show significant differences in the error distributions across the different translation directions, even when switching the source language with another source language of the same family. No particular differences in the error distributions were observed across neural MT architectures (Recurrent vs. Transformer), while some marked differences were observed when comparing bilingual, multilingual, and zero-shot systems. A more in-depth analysis of these differences will be carried out in future work.

Acknowledgements

This work has been partially supported by the EC-funded projects ModernMT (H2020 grant agreement no. 645487) and QT21 (H2020 grant agreement no. 645452). We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus phrase-based machine translation quality: a case study. *arXiv preprint arXiv:1608.04631*.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2018. Neural versus phrase-based mt quality: An in-depth analysis on english–german and english–french. *Computer Speech & Language*, 49:52–70.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 169–214, Copenhagen, Denmark, September. Association for Computational Linguistics.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Katsuhito Sudoh, Koichiro Yoshino, and Christian Federmann. 2017. Overview of the IWSLT 2017 Evaluation Campaign. In *Proceedings of the 14th International Workshop on Spoken Language Translation (IWSLT)*, Tokyo, Japan.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Jonathan H Clark, Chris Dyer, Alon Lavie, and Noah A Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 176–181. Association for Computational Linguistics.

Michael Denkowski and Graham Neubig. 2017. Stronger baselines for trustable results in neural machine translation. *arXiv preprint arXiv:1706.09733*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *ACL (1)*, pages 1723–1732.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. *arXiv preprint arXiv:1601.01073*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. *arXiv preprint arXiv:1611.04798*.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

Surafel M Lakew, Mattia A Di Gangi, and Marcello Federico. 2017. Multilingual neural machine translation for low resource languages. In *CLiC-it 2017 4th Italian Conference on Computational linguistics*.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2016. Fully character-level neural machine translation without explicit segmentation. *arXiv preprint arXiv:1610.03017*.

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

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015b. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Helmut Schmid. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 44–49.

Rico Sennrich, Martin Volk, and Gerold Schneider. 2013. Exploiting Synergies Between Open Resources for German Dependency Parsing, POS-tagging, and Morphological Analysis. In *Proceedings of Recent Advances in Natural Language Processing*, number September, pages 601–609.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*, Boston, US-MA, August.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

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

Antonio Toral and Víctor M Sánchez-Cartagena. 2017. A multifaceted evaluation of neural versus phrase-based machine translation for 9 language directions. *arXiv preprint arXiv:1701.02901*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

David Vilar, Jia Xu, Luis Fernando dHaro, and Hermann Ney. 2006. Error analysis of statistical machine translation output. In *Proceedings of LREC*, pages 697–702.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. *arXiv preprint arXiv:1601.00710*.

On Adversarial Examples for Character-Level Neural Machine Translation

Javid Ebrahimi, Daniel Lowd, Dejing Dou
Computer and Information Science Department, University of Oregon, USA
{javid, lowd, dou}@cs.uoregon.edu

Abstract

Evaluating on adversarial examples has become a standard procedure to measure robustness of deep learning models. Due to the difficulty of creating white-box adversarial examples for discrete text input, most analyses of the robustness of NLP models have been done through black-box adversarial examples. We investigate adversarial examples for character-level neural machine translation (NMT), and contrast black-box adversaries with a novel white-box adversary, which employs differentiable string-edit operations to rank adversarial changes. We propose two novel types of attacks which aim to remove or change a word in a translation, rather than simply break the NMT. We demonstrate that white-box adversarial examples are significantly stronger than their black-box counterparts in different attack scenarios, which show more serious vulnerabilities than previously known. In addition, after performing adversarial training, which takes only 3 times longer than regular training, we can improve the model's robustness significantly.

1 Introduction

Last year, a mistranslation by Facebook's machine translation (MT) system led to a wrongful arrest (Berger, 2017). Instead of translating an Arabic phrase to "good morning," Facebook's MT translated it as "attack them." Arabic is a morphologically-rich language, and the MT mistook the input word for another which differs from the input by only one character. As MT is used more and more, it is increasingly important to understand its worst-case failures to prevent incidents like this.

Adversarial examples are inputs designed to make a machine learning model perform poorly, and are often constructed by manipulating real-world examples (Goodfellow et al., 2015). Belinkov and Bisk (2018) investigate the sensitivity of neural machine translation (NMT) to synthetic and natural noise containing common misspellings. They show that state-of-the-art models are vulnerable to adversarial attacks even after a spell-checker is deployed. By performing ensemble adversarial training (Tramèr et al., 2018), they improve an NMT's robustness to adversarial noise.

We explore the space of adversarial examples for NMT in two directions: first, we study untargeted adversarial examples in a *white-box* setting, wherein the adversary has access to model parameters and can use its gradients to inflict more damaging manipulations for a larger decrease in the BLEU score; second, equipped with the developed machinery to do white-box attacks, we can perform more interesting attacks. We propose *controlled* and *targeted* adversaries which create adversarial examples with other goals, instead of merely decreasing the BLEU score. A controlled adversary aims to mute a word in the original translation, while a targeted adversary aims to push a word into it. Table 1 shows one example of each category. In both cases, the adversary, which has no word alignment model, has not drastically changed the rest of the translation, and has been able to reach its goals.

There is growing interest in understanding vulnerabilities of NLP systems (Jia and Liang, 2017; Zhao et al., 2018; Belinkov and Bisk, 2018). Previous work in NLP has focused on creating adversarial examples in a *black-box* setting, wherein the attacker can query a model but does not have access to its

Proceedings of the 27th International Conference on Computational Linguistics, pages 653–663
Santa Fe, New Mexico, USA, August 20-26, 2018.

src	1901 wurde eine Frau namens Auguste in eine medizinische Anstalt in Frankfurt gebracht.
adv	1901 wurde eine Frau namens **Afuiguste** in eine medizinische Anstalt in Frankfurt gebracht.
src-output	In 1931, a woman named **Augustine** was brought into a medical institution in France.
adv-output	In 1931, a woman named Rutgers was brought into a medical institution in France.
src	Das ist Dr. Bob Childs – er ist Geigenbauer und Psychotherapeut.
adv	Das ist Dr. Bob Childs – er ist Geigenbauer und **Psy6hothearpeiut**.
src-output	This is Dr. Bob Childs – he's a wizard maker and a **therapist**'s **therapist**.
adv-output	This is Dr. Bob Childs – he's a brick maker and a **psychopath**.

Table 1: Controlled and Targeted Attack on DE→EN NMT. In the first example, the adversary wants to suppress a person's name, and in the second example, to replace occurrences of *therapist* with *psychopath*

parameters. Black-box attacks often rely on heuristic methods to create adversarial examples. In contrast, white-box attacks approximate the *worst-case* attack for a particular model and input, within some allowed set of perturbations. Therefore, white-box attacks can demonstrate and defend against a model's most serious vulnerabilities, which may not be discovered by black-box heuristics.

After exploring the space of adversarial examples for NMT and proposing new types of attacks, we focus on adversarial training to make our model more robust. We build on HotFlip (Ebrahimi et al., 2018), a recently-introduced white-box method for generating adversarial examples and performing adversarial training for text classification. At the core of HotFlip lies an atomic *flip* operation, which changes one character to another by using the gradients of the model with respect to the one-hot vector input. In this work, we extend it to include a broader set of attacks, and we also improve it with a better beam search heuristic and faster adversarial training.

Our contributions are as follows:

1. We use a gradient-based estimate, which ranks adversarial manipulations, and we search for adversarial examples using greedy search or beam search methods.

2. We propose two translation-specific types of attacks and provide a metric to evaluate adversaries in these scenarios. Our experiments show that white-box adversaries can be significantly stronger than black-box adversarial examples.

3. We investigate the robustness of models trained with white-box adversarial examples and compare their robustness with black-box trained models.

2 Related Work

The need to understand vulnerabilities of NLP systems is only growing. Companies such as Google are using text classifiers to detect abusive language[1], and concerns are increasing over deception (Zubiaga et al., 2016) and safety (Chancellor et al., 2016) in social media. In all of these cases, we need to to better understand the dynamics of how NLP models make mistakes on unusual inputs, in order to improve accuracy, increase robustness, and maintain security or privacy. While this line of research has recently received a lot of attention in the deep learning community, it has a long history in machine learning, going back to adversarial attacks on linear spam classifiers (Dalvi et al., 2004; Lowd and Meek, 2005).

Character-level NMT systems (Lee et al., 2017; Costa-Jussa and Fonollosa, 2016) and those based on sub-word units (Sennrich et al., 2016) are able to extract morphological features which can generalize unseen words. Belinkov and Bisk (2018) show that character-level machine translation systems are overly sensitive to random character manipulations, such as keyboard typos. They use black-box heuristics to generate character-level adversarial examples, without using the model parameters or gradients to generate adversarial examples. The major challenge in creating white-box adversarial examples for text is that optimizing over discrete input is difficult (Miyato et al., 2017), which is why previous work has focused on black-box adversarial examples. Zhao et al. (2018) search for black-box adversarial examples in the space of encoded sentences and generate adversarial examples by perturbing the latent representation until the model is tricked. However, it is not clear how many queries are sent to the model or what the

[1]https://www.perspectiveapi.com

success rate of the adversary is. We contrast black-box and white-box attacks and show how white-box attacks significantly outperform their black-box counterparts in controlled and targeted attack scenarios.

Adversarial training/regularization interleaves training with generation of adversarial examples (Goodfellow et al., 2015). Concretely, after every iteration of training, adversarial examples are created and added to the mini-batches. This technique has been used for text classification (Miyato et al., 2017) using adversarial noise on the word embeddings without creating real-world adversarial examples. Jia and Liang (2017) point out the difficulty of adversarial training with real-world adversarial examples, as it is not easy to create such examples efficiently. In this work, we improve the training time of HotFlip (Ebrahimi et al., 2018) by using a one-shot attack in our inner adversary, which makes the running time of adversarial training only 3 times slower than regular training.

3 White-Box Adversarial Examples

Editing text to trick an NLP model, constrained by the number of characters to change, r, is a combinatorial search problem. We develop a gradient-based optimization method to perform four text edit operations: namely, flip (replacing one character with another), swap (replacing two adjacent characters with each other), delete, and insert. We use derivatives with respect to one-hot representation of the input, to rank candidate changes to text, in order to search for an adversarial example which satisfies the adversary's goal. Compared with a black-box adversary, our method has the overhead of sorting or searching among derivatives, while being considerably more successful.

3.1 Definitions

We use $J(\mathbf{x}, \mathbf{y})$ to refer to the log-loss of the translation model on source sequence $\mathbf{x}$ and target sequence $\mathbf{y}$. Let V be the alphabet, $\mathbf{x}$ be a text of length L characters, and $x_{ij} \in \{0,1\}^{|V|}$ denote a one-hot vector representing the j-th character of the i-th word. The character sequence can be represented by

$$\mathbf{x} = [(x_{11},.. \; x_{1n});(x_{21},.. \; x_{2n});...(x_{m1},.. \; x_{mn})]$$

wherein a semicolon denotes explicit segmentation between words. The number of words is denoted by m, and n is the number of maximum characters allowed for a word[2].

3.2 Derivatives of Operations

We represent text edit operations as vectors in the input space, and estimate the change in loss by directional derivatives with respect to these operations. Based on these derivatives, the adversary can choose the best loss-increasing operation. Our algorithm requires just one function evaluation (forward pass) and one gradient computation (backward pass) to estimate the best possible flip.

A **flip** of the j-th character of the i-th word ($a \rightarrow b$) can be represented by this vector:

$$\vec{v}_{ijb} = (\vec{0},..;(\vec{0},..(0,..\text{-}1,0,..,1,0)_j,..\vec{0})_i; \vec{0},..)$$

where -1 and 1 are in the corresponding positions for the a-th and b-th characters of the alphabet, respectively, and $x_{ij}^{(a)} = 1$. A first-order approximation of change in loss can be obtained from a directional derivative along this vector:

$$\nabla_{\vec{v}_{ijb}} J(\mathbf{x}, \mathbf{y}) = \nabla_x J(\mathbf{x}, \mathbf{y})^T \cdot \vec{v}_{ijb} = \frac{\partial J}{\partial x_{ij}}^{(b)} - \frac{\partial J}{\partial x_{ij}}^{(a)} \quad (1)$$

Using the derivatives as a surrogate loss, we simply need to find the best change by **maximizing** eq. 1, to *estimate* the best character change ($a \rightarrow b$). This requires searching in $|V|mn$ values for a given text of m words with n characters each, in a vocabulary of size $|V|$.

Similarly, character **insertion**, **deletion**, and **swap** of adjacent characters can be represented as vectors which carry the information about the direction and number of flips. Since the magnitudes of operation vectors are different, we normalized them by both L_1 and L_2 norm but found it had little impact.

A black-box adversary can perform similar manipulations which would be randomly picked. For example, Belinkov and Bisk (2018) define `Key` operation, which flips one character with an adjacent one on the keyboard, at random.

[2]Padding is applied if the number of characters is fewer than the maximum.

3.3 Controlled and Targeted Attacks

The previous adversary was untargeted, and its only goal was to increase the loss of the model. However, some corruptions of the output may be much worse than others – translating "good morning" as "attack them" is much worse than translating it as "fish bicycle." By changing the loss function, we can force the adversary to focus on more specific goals.

In a *controlled attack*, the adversary tries to remove a specific word from the translation. This could be used to maintain privacy, by making more sensitive information harder to translate, or to corrupt meaning, by removing key modifiers like "not," "joked,"' or "kidding." Concretely, we maximize the loss function $J(x, y_t)$, where t is the target word. This way, the adversary ignores the rest of the output and focuses on parts of the input that would affect the target word most.

In a *targeted attack*, the adversary aims to not only mute a word but also replace it with another. For example, changing the translation from "good morning"' to "good attack" could lead to an investigation or an arrest. Making specific changes like this is much more dangerous, but also harder for the adversary to do. For this attack, we maximize the loss $-J(x, y_{t'})$, where $t^{'}$ is the new word chosen to replace t. Note that the negation makes this equivalent to minimizing the predictive loss $J(x, y_{t'})$ on $t^{'}$. We represent this as maximization so that it fits in the same framework as the other attacks. Our derivative-based approach from the previous subsection can be then used directly to generate these new attacks, simply by substituting the alternate loss function.

3.4 Multiple Changes

We explained how to estimate the best single change in text to get the maximum increase/decrease in loss. We now discuss approaches to perform multiple changes.

(a) **one-shot:** In this type of attack, the adversary manipulates all the words in the text with the best operation in parallel. That is, the best operation for each word is picked locally and independently of other words. This is efficient, as with only one forward and backward pass, we can collect the gradients for all operations for all words. In addition, compared with the global one-shot method of Ebrahimi et al. (2018), it does not require sorting the gradients globally, and can further reduce the time to create adversarial examples. It is less optimal than the next approaches, which apply changes one by one. Due to its efficiency, this is the approach we choose to do adversarial training. Our experiments in section 6, which are untargeted, follow this approach. We also investigate black-box and white-box variants of one-shot attacks in section 5.1. The budget for the adversary is the number of words, and it is spent in the first shot.

(b) **Greedy:** In this type of attack, after picking the best operation in the whole text, we make another forward and backward pass, and continue our search. Our controlled attack in section 5.2 follows this approach, where we allow a maximum of 20% of the characters in text as the budget for the adversary. As will be explained in 5.2, the adversary spends much less than this amount.

(c) **Beam Search**: And finally, we can strengthen our greedy search by beam search. Our beam search requires only $\mathcal{O}(br)$ forward passes and an equal number of backward passes, with r being the budget and b, the beam width. At every step, the beam will be sorted by the sum of the true loss up to that point, which we have computed, plus the gradient-based estimate of candidate operations. This showed better performance than using the sum of the gradients in the path for sorting the beam, as was previously done (Ebrahimi et al., 2018). Since targeted attacks are the most difficult type of attack, we use this strategy for targeted attacks, as described in section 5.3. We allow a maximum of 20% of the characters in text as the budget for the adversary, and set the beam width to 5.

4 Experiments

We use the TED talks parallel corpus prepared by IWSLT 2016 (Mauro et al., 2016) for three pairs of languages: German to English, Czech to English, and French to English. We use the development sets and test sets of previous years except 2015 as our development set. The statistics of the dataset can be

Pair	Train	Test	Target vocab.
FR-EN	235K	1.1k	69k
DE-EN	210K	1.1k	66k
CS-EN	122K	1.1k	49k

Table 2: Data Statistics

found in Table 2. Throughout our experiments, we only allow character changes if the new word does not exist in the vocabulary, to avoid changes that an MT would respond to as expected. For example, it is not surprising that changing the source German word "nacht" to "nackt" would cause an MT to introduce the word "nude" in the translation.

The architecture we study was first proposed by Kim et al. (2016) for language modeling, and later adapted by Costa-Jussa and Fonollosa (2016) for translation, and is also used by Belinkov and Bisk (2018) for their adversarially-trained models. In this architecture, feature extraction is performed by convolutions over characters, which are passed to layers of highway networks, and finally given to stacks of recurrent neural nets for modeling a sequence of words. Using this architecture, the BLEU scores on our datasets are competitive with the submissions to the IWSLT (Mauro et al., 2016). Our implementation[3] relies largely on Yoon Kim's seq2seq implementation[4], with similar hyper-parameters, which mostly follows the guidelines of Luong et al. (2015) for attentional translation.

For experiments in section 6, where we report the BLEU score for a vanilla model and several adversarially-trained models against different attackers, we use a decoder with a beam width of 4. However, our white-box attacker uses a model with greedy decoding to compute gradients. The reason is that in order to calculate correct gradients, we either need to use greedy decoding, or use models which incorporate beam search in the decoder architecture such that gradients could flow in the beam paths, too (Wiseman and Rush, 2016), which incur more computational cost. Correct gradients are more of an important issue for targeted attacks, where we want to achieve a goal beyond simply breaking the system. For the sake of consistency, we use greedy decoding for both the vanilla model which is being attacked, and the white-box attacker in all experiments of section 5, where we contrast white-box and black-box adversaries in different scenarios.

5 Analysis of Adversaries

We first study whether first-order approximation gives us a good estimator to be employed by white-box adversaries. Figure 1 compares the true increase in log-loss (i.e., $J(x+v) - J(x)$), with our gradient-based estimate (i.e., $\nabla_v J(x)$). We create adversarial examples using the best estimated character flip for every word, over the German test set. Then, we compare the true increase in loss for the created adversarial examples, with our gradient-based estimate. The log-loss is evaluated by summing the log-loss of individual words, and similarly, the gradient-based estimate is the sum of all gradients given by flips which are performed once on every word. Figure 1.a plots the histograms for both of these measures, and Figure 1.b shows a scatter plot of them with the least squares fitting line. Due to linearization bias of the first-order approximation, we have a distribution with smaller variance for the gradient-based estimate measure. We can also observe a moderately positive correlation between the two measures (Spearman coefficient $\rho = 0.61$), which shows we can use the gradient-based estimate for *ranking* adversarial manipulations.

Next, we contrast black-box and white-box adversaries in untargeted, controlled, and targeted scenarios, and demonstrate that white-box adversaries significantly outperform black-box adversaries especially in controlled and targeted scenarios.

5.1 Untargeted Attack

Table 3 shows the BLEU score after one-shot white-box and black-box attacks are performed. Unlike delete and swap, insert and flip have the advantage of making changes to one-letter words, so we expect

[3]https://github.com/jebivid/adversarial-nmt

[4]https://github.com/harvardnlp/seq2seq-attn

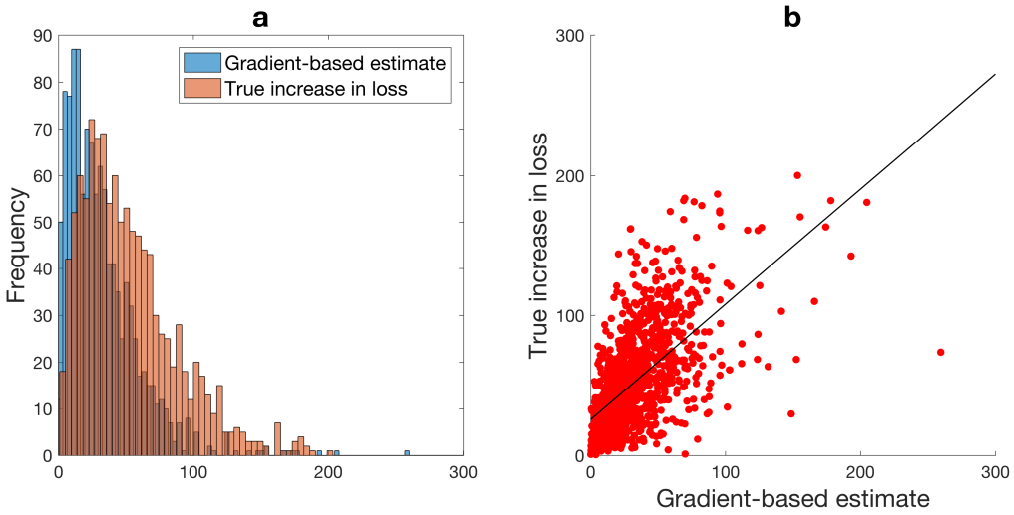


Figure 1: Comparing the distribution of the true increase in loss and its gradient-based estimate, and their correlation, using best flips for each word in a sentence of the German test set.

them to perform better. We see this for the white-box attacks which can pick the best change to every word using the gradients. On the contrary, a black-box adversary performs worst for flip, which is because the black-box attacker is not enabled to pick the best change when more options (possible character flips) are available, as opposed to swap and delete which are governed by the location of the change and contain no additional flip. Nevertheless, a black-box adversary has competitive performance with the white-box one, even though it is simply randomly manipulating words. We argue that evaluating adversaries, based on their performance in an untargeted setting on a brittle system, such as NMT, is not appropriate, and instead suggest using goal-based attacks for evaluation.

Attack	Flip		Insert		Delete		Swap	
	white	black	white	black	white	black	white	black
FR	**4.27**	6.98	**4.74**	4.85	**4.99**	5.86	**4.87**	5.20
DE	**4.50**	6.87	**3.91**	4.31	**5.63**	5.73	4.94	**4.74**
CS	**4.31**	6.09	**4.66**	5.86	**6.30**	6.62	6.05	**5.82**

Table 3: BLEU score after greedy decoding in the existence of different types of untargeted attacks.

5.2 Controlled Attack

We introduce more interesting attacks, in which the adversary targets the MT for more specific goals. A perfect *mute* attack removes a word successfully and keeps the rest of the sentence intact. For example, consider the translation T, containing words w_1, w_2,..., w_t, ...w_n, where w_t is the target word. A perfect mute attack will cause the NMT to create a translation T_p which contains words w_1, w_2,..., UNK, ...w_n, wherein w_t is replaced with UNK. With this observation in mind, we define the success rate of an attack, which generates T_{adv}, as follows:

$$\text{success}(T_{adv}) = \begin{cases} 1, & \text{if } \frac{\text{BLEU}(T, T_{adv})}{\text{BLEU}(T, T_p)} \geq \alpha \\ 0, & \text{otherwise} \end{cases}$$

We can control the quality of an attack with α, for which a larger value punishes the adversary for ad-hoc manipulations, which could cause the NMT to generate a radically different and possibly gibberish translation. Success rate is defined by the number of successful attacks divided by the number of total sentences in the test set. Figure 2 plots the success rate against α. As can be seen, the white-box adversary is significantly more successful than a black-box adversary. By taking advantage of the knowledge of

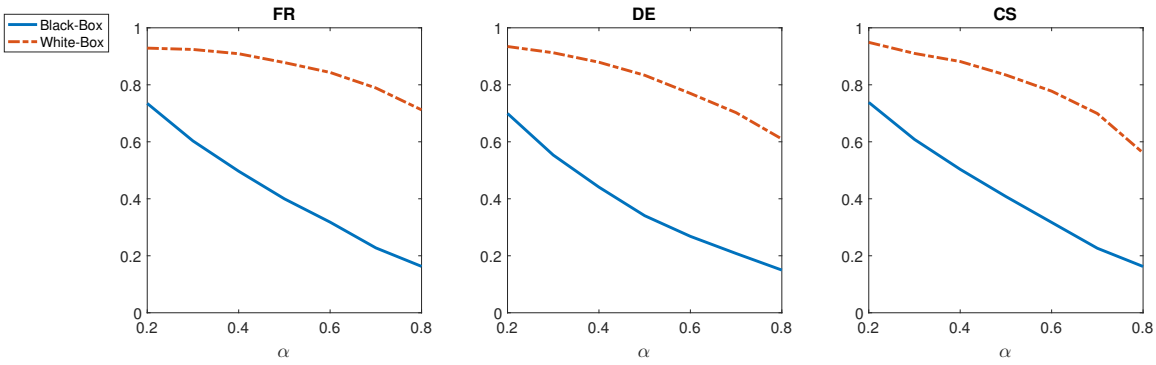


Figure 2: Success rate of white-box and black-box adversaries in a controlled setting as a function of α.

gradients of the model, a white-box adversary can perform better targeted attacks. For this experiment, the black-box attacker uniformly picks from the four possible changes and randomly applies them.

For this attack, we follow the greedy approach, and use a budget of 20% of the characters in text. Table 4 shows the average number of character changes and the number of queries made to the model. The reported character changes are for attacks wherein the attacker only muted a target word successfully, regardless of the quality of the translation. The reported number of queries takes the unsuccessful trials into account too. The white-box adversary is more efficient due to fewer queries and fewer manipulated characters, which can be crucial for a real-word adversary. Nevertheless, unlike a black-box adversary, a white-box adversary requires additional backward passes, and has the overhead of operations on the gradient values, mainly sorting. This makes the running times of the two comparable.

Compared with the results in the previous section, controlled attacks show a more convincing superiority of the white-box attacks over black-box attacks.

	Character Changes		Queries	
source	white	black	white	black
FR	**1.9**	7.7	**2.3k**	8.9k
DE	**1.9**	6.5	**1.9k**	7.8k
CS	**1.5**	5.3	**1.2k**	6.1k

Table 4: Efficiency of attacks.

5.3 Targeted Attack

A more challenging attack is to not only mute a word but also replace it with another one. The evaluation metric for targeted attack is similar to controlled attack with one difference that a perfect *push* attack produces a translation, T_p, which contains words w_1, w_2, $w_{t'}$,..w_n, wherein $w_{t'}$ has replaced w_t. In classification domains with few classes, targeted attacks are relatively simple, since an adversary can perturb the input to move it to the other side of a decision boundary. Whereas in MT, we deal with vocabulary sizes in the order of at least tens of thousands, and it is less likely for an adversary to be successful in targeted attacks for most possible target words. To address this, we evaluate our adversary with n^{th}-most likely class attacks. In the simplest case, we replace a target word with the second-most likely word at decoding time.

As can be seen in Table 3, targeted attacks are much more difficult with a much lower success rate for the adversary. Nevertheless, the white-box adversary still performs significantly better than the black-box adversary. The success rate dramatically goes down for large values of n. For example, for the value of 100, the success rate will be more than ten times smaller than second-most likely attack.

5.4 Some Adversarial Examples

Table 5 shows three adversarial examples. The first example shows a controlled attack, where the adversary has successfully removed a swear word from the sentence. The BLEU ratio, used in our success

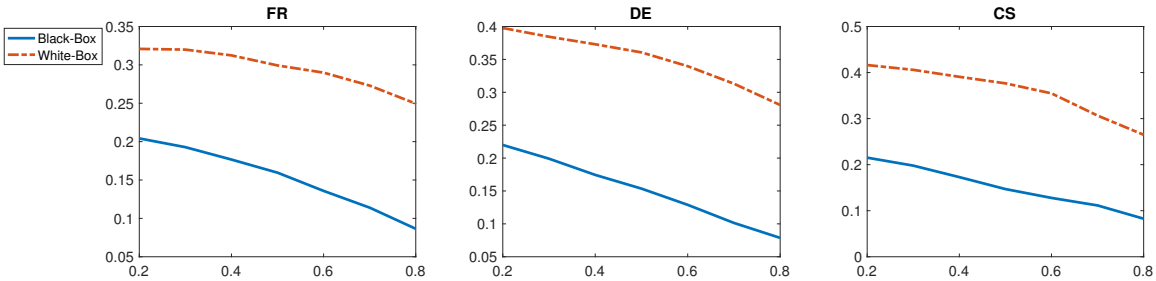


Figure 3: Success rate of white-box and black-box adversaries in the second-most likely targeted attack as a function of α.

src	Wir erwarten Perfektion von Feministinnen, weil wir immer noch für so viel kämpfen, so viel wollen, so verdammt viel brauchen.
adv	Wir erwarten Perfektion von Feministinnen, weil wir immer noch für so viel kämpfen, so viel wollen, so **öberdammt** viel brauchen.
src-output	We expect perfection from feminist, because we're still fighting so much, so **damn**, so **damn**, so **damn**.
adv-output	We expect perfection from feminist, because we still fight for so much, so much of all, so much of all that needs to be.
src	In den letzten Jahren hat sie sich zu einer sichtbaren Feministin entwickelt.
adv	In den letzten Jahren hat sie sich zu einer sichtbaren **FbeminisMin** entwickelt.
src-output	In the last few years, they've evolved to a safe **feminist**.
adv-output	In the last few years, they've evolved to a safe **ruin**.
src	Ein Krieg ist nicht länger ein Wettbewerb zwischen Staaten, so wie es früher war.
adv	Ein Krieg ist nicht länger ein **erkBkaSzeKLlWmrt** zwischen Staaten, so wie es früher war.
src-output	A war is no longer a **competition** between states, like it used to be.
adv-output	A war is no longer a **throwaway** planet between states, as it used to be.

Table 5: A controlled attack and two targeted attacks on our DE-EN NMT. First example shows a controlled attack, the second and third examples show a second-most and a 100_{th}-most likely targeted attack, respectively.

rate measure, for this example is 0.52. The second example shows a second-most likely targeted attack where the new translation has managed to keep the rest of the translation intact and achieve its goal. The BLEU ratio for this example is 1.00. The third example, which has a BLEU ratio of 0.70, shows a 100^{th}-most likely attack, where the word *competition* is replaced with *throwaway*. Due to the difficulty of this change, the adversary has committed a considerably larger number of manipulations.

6 Robustness to Adversarial Examples

6.1 Baselines

We use the black-box training method of Belinkov and Bisk (2018) as our baseline. They train several models using inputs which include noise from different sources. We used their script[5] to generate random (`Rand`), keyboard (`Key`), and natural (`Nat`) noises. Their best model was one which incorporated noise from all three sources (`Rand+Key+Nat`).

Similar to their approach, we train a model which incorporates noisy input scrambled by Flips, Inserts, Deletes, and Swaps in training (`FIDS-B`). Since natural noise was shown to be the most elusive adversarial manipulation (Belinkov and Bisk, 2018), we used this source of noise to determine the proportion of each of the FIDS operations in training. Concretely we found that the majority of the natural noise can be generated by FIDS operations, and we used the ratio of each noise in the corpora to sample from these four operations. Figure 4 shows the distribution of manipulations for each language. A single swap is the least likely operation in all three languages[6]. FIDS operations account for 64%, 80%, and 70% of natural noise for Czech, German, and French, respectively. This can be regarded as a background knowledge incorporated into the adversary.

[5]https://github.com/ybisk/charNMT-noise

[6]Excluding single swaps from manipulations with two flips.

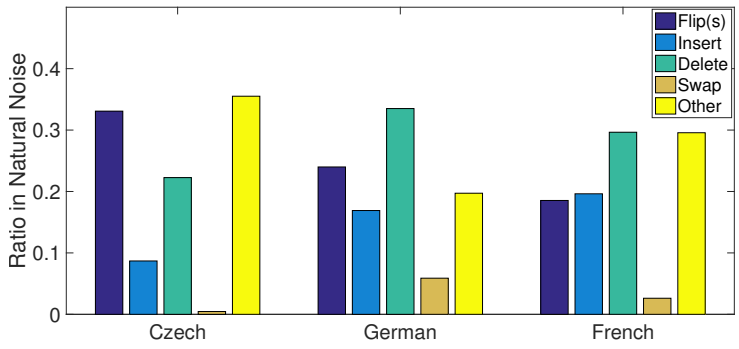


Figure 4: Distribution of types of noise in the natural noise corpora.

6.2 White-Box and Ensemble Methods

Our white-box adversary `FIDS-W` generates adversarial examples using our four text edit operations in accordance with the distribution of operations on the natural noise. At every epoch, adversarial examples are generated for every mini-batch using the one-shot approach. More precisely, all words in the sentence are changed by a single FIDS operation in parallel. While training on both clean and adversarial examples has been the standard approach in adversarial training, some evidence in computer vision (Madry et al., 2017; Shaham et al., 2015) suggests that training on white-box adversarial examples alone can boost models' robustness to adversarial examples, with a minor decrease in accuracy on clean examples. However, we found that in order to get a good BLEU score on the clean dataset, we need to train on both clean and white-box adversarial examples. We also train an ensemble model, `Ensemble`, which incorporates white-box and black-box adversarial examples, with 50/50 share for each. The black-box adversarial examples come from `Nat` and `Rand` sources.

When evaluating models against `White` adversarial examples at test-time, we use the test set which corresponds to their method of training. For instance, the `White` adversarial examples for the `Rand` model, come from the test set which has manipulated clean examples by `Rand` noise first. For `Vanilla`, `FIDS-W`, and `Ensemble` models, the adversarial examples are generated from clean data. This makes the comparison of models, which are trained on different types of data, fair. We use the one-shot attack for our white-box attack evaluation, using the same distribution based on natural noise.

6.3 Discussion

Table 6 shows the results for all models on all types of test data. Overall, our ensemble approach performs the best by a wide margin. As expected, adversarially-trained models usually perform best on the type of noise they have seen during training. However, we can notice that our `FIDS-W` model performs best on the `Nat` noise amongst models which have not been trained on this type of noise. Similarly, while `FIDS-W` has not directly been trained on `Key` noise, it is trained on a more general type of noise, particularly flip, and thus can perform significantly better on the `Key` than on other models which also have not been trained on this type of noise. However, it cannot generalize to `Rand`, which is an extreme case of attack, and we need to use an ensemble approach to perform well on it too. Nevertheless, `FIDS-W` performs best on the `Rand` noise, compared with models which are not trained on `Rand` either. This validates our earlier claim that training on white-box adversarial examples, which are harder adversarial examples, can make the model more robust to weaker types of noise. We also observe that `FIDS-B` performs better on the `White` examples compared with other baselines; although it has not been trained on white-box adversarial examples, it is trained on black-box adversarial examples of the same family of FIDS operations.

Figure 5 shows the training loss of white-box adversarially-trained model on adversarial examples. The model is getting more resilient to adversarial examples, which are created at the start of each epoch.

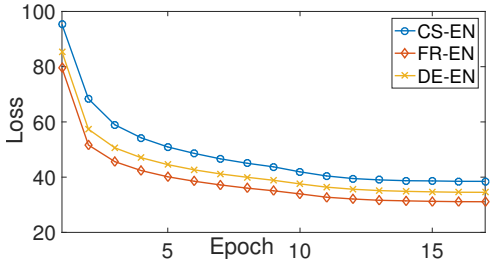


Figure 5: Training loss on adversarial examples for FIDS-W.

Training \ Test		Clean	Nat	Key	Rand	FIDS-B	FIDS-W	Avg.
French	Vanilla	37.54	19.17	12.12	4.75	6.85	5.36	15.95
	Nat	26.35	**33.23**	11.16	5.32	8.28	6.65	15.16
	Key	33.02	17.30	**35.97**	4.63	7.00	5.17	17.17
	Rand	36.06	18.54	8.31	**36.10**	8.76	7.14	19.14
	FIDS-B	34.48	21.59	28.48	6.82	32.62	13.60	22.92
	FIDS-W	37.15	23.65	31.18	7.78	**32.72**	**31.94**	27.40
	Rand+Key+Nat	34.55	30.74	32.82	34.01	12.05	7.08	25.20
	Ensemble	**37.81**	30.27	29.36	34.42	32.00	30.01	**32.30**
German	Vanilla	31.81	17.24	10.36	4.20	6.78	5.50	12.64
	Nat	24.89	**32.14**	10.22	4.61	7.53	5.99	14.23
	Key	27.20	15.98	**30.62**	4.64	7.68	4.74	15.13
	Rand	31.01	17.90	6.59	**30.70**	9.19	5.83	16.86
	FIDS-B	28.27	20.22	23.84	6.29	27.35	10.79	19.45
	FIDS-W	**31.81**	21.72	26.23	7.75	**27.38**	**26.51**	23.56
	Rand+Key+Nat	29.22	29.78	27.83	28.88	10.30	6.14	22.01
	Ensemble	31.54	31.11	23.91	28.95	26.38	25.06	**27.82**
Czech	Vanilla	**26.44**	13.55	9.49	4.78	7.30	5.93	11.24
	Nat	18.73	**23.06**	9.07	4.45	7.36	5.42	11.34
	Key	22.76	13.09	**23.79**	4.83	7.93	5.82	13.03
	Rand	24.23	12.00	7.26	**24.53**	7.24	5.47	13.45
	FIDS-B	22.31	14.15	17.91	6.48	19.67	8.60	14.84
	FIDS-W	25.53	15.57	19.74	7.18	**20.02**	**19.42**	17.90
	Rand+Key+Nat	22.21	20.59	20.60	21.33	10.06	5.89	16.77
	Ensemble	25.45	20.46	17.15	21.39	18.52	17.03	**19.99**

Table 6: BLEU score of models on clean and adversarial examples, using a decoder with beam size of 4. The best result on each test set is shown in bold. FIDS-W performs best on all noisy test sets compared with models which have not been trained on that particular noise (shown in red). FIDS-B performs best on white-box adversarial examples compared with other black-box trained models (shown in blue).

7 Conclusion and Future Work

As MT methods become more effective, more people trust and rely on their translations. This makes the remaining limitations of MT even more critical. Previous work showed that NMT performs poorly in the presence of random noise, and that its performance can be improved through adversarial training. We consider stronger adversaries which are attacking a specific model and may also have specific goals, such as removing or changing words. Our white-box optimization, targeted attacks, and new evaluation methods are a step towards understanding and fixing the vulnerabilities in NMT: we're able to find more effective attacks and train more robust models than previous black-box methods. Next steps include exploring other types of targeted attacks, such as attacks that target more than one word, and other types of constraints, such as better characterizing which character changes affect intelligibility the least.

8 Acknowledgment

This work was funded by ARO grant W911NF-15-1-0265.

References

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *Proceedings of ICLR*.

Yotam Berger. 2017. Israel arrests Palestinian because Facebook translated 'good morning' to 'attack them'. Retrieved from `https://www.haaretz.com`.

Stevie Chancellor, Jessica Annette Pater, Trustin Clear, Eric Gilbert, and Munmun De Choudhury. 2016. # thyghgapp: Instagram content moderation and lexical variation in pro-eating disorder communities. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, pages 1201–1213. ACM.

Marta R Costa-Jussa and José AR Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of ACL*.

Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, Deepak Verma, et al. 2004. Adversarial classification. In *Proceedings of KDD*, pages 99–108.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *Proceedings of ACL*.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of ICLR*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of EMNLP*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. *Proceedings of AAAI*.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *TACL*.

Daniel Lowd and Christopher Meek. 2005. Adversarial learning. In *Proceedings of KDD*, pages 641–647.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Cettolo Mauro, Niehues Jan, Stüker Sebastian, Bentivogli Luisa, Cattoni Roldano, and Federico Marcello. 2016. The iwslt 2016 evaluation campaign. In *International Workshop on Spoken Language Translation*.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *Proceedings of ICLR*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. *Proceedings of ACL*.

Uri Shaham, Yutaro Yamada, and Sahand Negahban. 2015. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. *arXiv preprint arXiv:1511.05432*.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. 2018. Ensemble adversarial training: Attacks and defenses. In *Proceedings of ICLR*.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of EMNLP*.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *Proceedings of ICLR*.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PloS one*, 11(3):e0150989.

Systematic Study of Long Tail Phenomena in Entity Linking

Filip Ilievski, Piek Vossen, Stefan Schlobach
Vrije Universiteit Amsterdam
De Boelelaan 1105, 1081 HV Amsterdam
The Netherlands
{f.ilievski, piek.vossen, k.s.schlobach}@vu.nl

Abstract

State-of-the-art entity linkers achieve high accuracy scores with probabilistic methods. However, these scores should be considered in relation to the properties of the datasets they are evaluated on. Until now, there has not been a systematic investigation of the properties of entity linking datasets and their impact on system performance. In this paper we report on a series of hypotheses regarding the long tail phenomena in entity linking datasets, their interaction, and their impact on system performance. Our systematic study of these hypotheses shows that evaluation datasets mainly capture head entities and only incidentally cover data from the tail, thus encouraging systems to overfit to popular/frequent and non-ambiguous cases. We find the most difficult cases of entity linking among the infrequent candidates of ambiguous forms. With our findings, we hope to inspire future designs of both entity linking systems and evaluation datasets. To support this goal, we provide a list of recommended actions for better inclusion of tail cases.

1 Introduction

The task of Entity Linking (EL) anchors recognized entity mentions in text to their semantic representation, thus establishing identity and facilitating the exploitation of background knowledge, easy integration, and comparison and reuse of systems. The past years featured a plethora of EL systems: DBpedia Spotlight (Daiber et al., 2013), WAT (Piccinno and Ferragina, 2014), AGDISTIS MAG (Moussallem et al., 2017), to name a few. These systems propose various probabilistic algorithms for graph optimization or machine learning, in order to perform disambiguation, i.e., to pick the correct entity candidate for a surface form in a given context. The reported accuracy scores are fairly high, which gives an impression that the task of EL is both well-understood and fairly solved by existing systems.

At the same time, several papers (Ilievski et al., 2016; Van Erp et al., 2016; Esquivel et al., 2017; Ilievski et al., 2017) have argued that state-of-the-art EL systems base their performance on frequent 'head' cases, while performance drops significantly when moving towards the rare 'long tail' entities. This statement seems intuitively obvious, but no previous work has quantified what the 'head' and 'tail' of EL entails. In fact, the lack of definition of head and tail in this task prevents the (in)validation of the hypothesis that interpreting some (classes of) cases is more challenging for systems than others. This, in turn, means that we are currently unable to identify the difficult cases of EL for which current systems need to be adapted, or new approaches need to be developed. Previous linguistic studies which analyze words distributions can not be applied for this purpose, because they do not study reference, nor the relation of the head-tail distribution to system performance.

Understanding the tail cases better and explicitly addressing them in systems design will be beneficial because: 1. a lot of textual data and requirements for processing it are made up of long tail cases, 2. unlike the head entities, the knowledge about tail entities is less accessible (in structured or unstructured form), not redundant, and hard to obtain. 3. to perform well on the tail, systems are required to interpret entity references without relying on statistical priors, but by focusing on high-precision reasoning.

Proceedings of the 27th International Conference on Computational Linguistics, pages 664–674
Santa Fe, New Mexico, USA, August 20-26, 2018.

This paper addresses the question: **which data properties capture the distinction between the head and the tail in entity linking, and to what extent?** Its main contributions are the following:

1. We define the long tail properties of entity linking **(Section 3)**.[1] This is the first work that looks systematically into the relation of surface forms in datasets and their instances in DBpedia, and provides predictions in the form of a series of hypotheses about long tail phenomena.

2. We analyze existing entity linking datasets with respect to these long tail properties, demonstrating that data properties have certain inter-correlations that follow our hypotheses **(Section 4)**.

3. We describe how the performance of systems correlates with head and long tail cases, proving that the long tail phenomena and their interactions influence the performance of systems **(Section 5)**.

4. We provide recommendations on how to address the tail in future EL research (**Section 7**).

2 Related work

Analyzing well-known datasets for semantic tasks, Ilievski et al. (2016) measured very low ambiguity and variation, and a notable bias towards dominance. Overall, tasks and datasets show strong **semantic overfitting** to the head of the distribution (the popular part) and are not representative for the diversity of the long tail. In the related task of Word Sense Disambiguation, Postma et al. (2016) analyzed the impact of frequency on system accuracy, showing that the accuracy on the most frequent words is close to human performance, while the least frequent words can be disambiguated correctly in at most 20% of cases. According to Van Erp et al. (2016), EL datasets contain very little referential ambiguity and evaluation is focused on well-known entities (i.e., with high PageRank (Page et al., 1999) values).

NIL entities are entities without a representation in a knowledge base (Ji and Grishman, 2011). These are typically considered to have low frequencies within a corpus and/or to be domain-specific. Esquivel et al. (2017) report that around 50% of the people mentioned in news articles are not present in Wikipedia. Considering that Wikipedia and its structured data derivatives are almost exclusively used as an anchor in EL, this means that for half of all people mentions, the EL task is nonsensical. While NILs are a challenge that concerns the long tail of EL, in this work we focus on those entities that have been linked to Wikipedia, but are still infrequent, since this provides an entry for extensive analysis of their properties.

In this work, we distinguish dark, emerging, and domain entities. **Dark entities** are those for which no relevant information is present in a given knowledge base (Van Erp et al., 2015). Dark entities thus expand the notion of NIL entities to cases where an entity representation exists, but it is insufficient to reason over. **Emerging entities** are time-bound entities, recently unknown but potentially becoming popular in news in a short time (Hoffart et al., 2014). A body of work has dealt with **domain entities**, whose relevance is restricted within a topic, e.g. biomedical (Zheng et al., 2015), or historical domain (Heino et al., 2017). Dark, emerging, and domain entities mostly make up the tail in EL. Their definition is, however, orthogonal to our work: we strive to provide an umbrella theory of the tail in EL based on linguistic properties and avoid a discussion on defining the distinction between head or tail in a categorical way.

Finally, studying distributional properties of entity expressions and their linking, as we do in this study, is different from the classical linguistic studies on the distribution of words (Zipf, 1935; Corral et al., 2015; Kanwal et al., 2017). Linked entity data provides information on the surface forms, the meaning, and the referent of the surface form, whereas distributional studies on words only provide knowledge on the surface forms and to a limited extent on their sense, but never on their reference.

3 Approach

To address our research goal of **quantifying the long tail of EL**, we first explain the notions of ambiguity, variance, frequency, and popularity. Next, we formulate a set of hypotheses regarding their interaction and our expected influence on system performance. We also describe our choice of data collections and

[1] We consider the following properties: ambiguity of surface forms, variance of instances, frequency of surface forms, frequency of instances, and popularity of instances.

EL systems to analyze. The code of this analysis is available on Github at `https://github.com/cltl/EL-long-tail-phenomena`.

3.1 The long tail phenomena of the entity linking task

Each entity exists only once in the physical world. However, this is different in our communication where: 1. certain surface forms are very prominent and others occur only rarely; 2. certain instances are very prominent and others are mentioned incidentally. The task of EL covers a many-to-many relation between surface forms observed in text and their instances potentially found in a knowledge base. Surface forms and instances both have their own **frequency distribution**, pointing to the same underlying Grice (1975) mechanisms, governed by an envisioned trade-off between efficiency and effectiveness.

Surface forms have various frequency of occurrence in textual documents. Frequent surface forms include "U.S." and "Germany", but also "John Smith". The frequency of a surface form can be explained by its relation to one (or a few) very popular instances (`United States`), but it can also be a result of high **ambiguity** ("John Smith" is a common name, so it simply refers to many possible instances).[2]

Similarly, some **instances** are more popular and therefore more frequently mentioned than others. Note that *frequency* here refers to the number of occurrences in a corpus, while popularity refers to the frequency as a topic and can, for example, be measured by the volume of knowledge about an instance captured with its **PageRank** in a knowledge base. Frequent and popular instances are intuitively quite prominent and relevant, very often across many different circumstances, and are typically referred to by a relatively wide set of surface forms, resulting in a high **variance**. In addition, frequent/popular entities tend to participate in metonymy relations with other entities topically related to them. For instance, `United States` as a country relates to `United States Army` and to `United States Government` - two other entities of a different type, but possibly referenced by the same surface forms.

3.2 Hypotheses on the long tail phenomena of the entity linking task

We look systematically at the relation of surface forms in datasets and their instances in DBpedia, and provide a series of hypotheses regarding the long tail phenomena and their relation to system performance (Table 1). Some of these hypotheses, e.g., D1 and D2, are widely accepted as common knowledge but have rarely been investigated in EL datasets. Others, such as S4 and S5, are entirely new.

ID	Hypothesis	Sec
D1	Only a few forms and a few instances are very frequent in corpora, while most appear only incidentally.	4.1
D2	A few instances in corpora are much more popular (have much higher PageRank) compared to most other.	4.2
D3	Only a small portion of all forms in corpora are ambiguous.	4.3
D4	Only a small portion of all instances in corpora are referred to with multiple forms.	4.4
D5	There is a positive correlation between ambiguity of forms and their frequency.	4.5
D6	There is a positive correlation between variance of instances and their frequency.	4.5
D7	There is a positive correlation between variance of instances and their popularity.	4.5
D8	There is a positive correlation between popularity of instances and their frequency.	4.5
D9	The frequency distribution within all forms that refer to an instance is Zipfian.	4.6
D10	The frequency and the popularity distribution within all instances that refer to a form is Zipfian.	4.6
S1	Systems perform worse on forms that are ambiguous than overall.	5.1
S2	There is a positive correlation between system performance and frequency/popularity.	5.2
S3	Systems perform best on frequent, non-ambiguous forms, and worst on infrequent, highly ambiguous forms.	5.3
S4	Systems perform better on ambiguous forms with imbalanced, compared to balanced, instance distribution.	5.4
S5	Systems perform better on frequent instances of ambiguous forms, compared to their infrequent instances.	5.4
S6	Systems perform better on popular instances of ambiguous forms, compared to their unpopular instances.	5.4

Table 1: Hypotheses on the data properties (D*) and on their relation to system performance (S*)

[2]The notion of ambiguity captures the amount of instances to which a surface form has been observed to refer. Non-ambiguous forms are those that refer to a single instance, whereas ambiguous forms refer to at least two instances. Highly ambiguous forms refer to a wide array of instances.

3.3 Datasets and systems

We focus on well-known EL datasets with news documents, preferring larger sets with open licenses. Many customary EL datasets are however quite small ($< 1,000$ mentions). We opted to perform our analysis on the following two data collections, with five corpora in total:

AIDA-YAGO2 (Hoffart et al., 2011) - we consider its train, test A, and test B sets, summing up to 34,929 entity forms in 1,393 news documents, published by Reuters from August 1996 to August 1997.

N3 (Röder et al., 2014) is a collection of three corpora released under a free license. We consider the two N3 corpora in English: RSS-500 and Reuters-128. Reuters-128 contains economic news published by Reuters, while RSS-500 contains data from RSS feeds, covering various topics such as business, science, and world news. These two corpora consist of 628 documents with 1,880 entity forms in total.

We analyzed the EL performance of recent public and open-sourced entity linkers, as the state-of-the-art: 1. **AGDISTIS MAG** (Moussallem et al., 2017)[3] combines graph algorithms with context-based retrieval over knowledge bases. 2. **DBpedia Spotlight** (Daiber et al., 2013)[4] is based upon cosine similarities and a modification of TF-IDF weights. 3. **WAT** (Piccinno and Ferragina, 2014)[5] combines a set of algorithms, including graph- and vote-based ones.[6]

3.4 Evaluation

We apply the customary metrics of precision, recall, and F1-score to measure system performance in Section 5. In Table 2, we briefly describe the computation of true positives (TPs), false positives (FPs), and false negatives (FNs) per class. For example, if the gold instance belongs to C_1, then we count a TP when the system instance is also C_1. In case the system instance belongs to another class C_i, $i \neq 1$, this leads to a FN for C_1 and a FP for C_N. A special class are the NILs: predicting a NIL case incorrectly by the system results in a FN for the correct class; inversely, if the system was supposed to predict a NIL and it did not, then we count a FP. See the details in later sections for what constitutes a class. In our analysis we exclude the cases referring to NILs.

$G \backslash S$	C_1	...	C_N	NILL
C_1	TP(1)		FP(N), FN(1)	FN(1)
...				
C_N	FP(1), FN(N)		TP(N)	FN(N)
NILL	FP(1)		FP(N)	-

Table 2: Computation of TPs, FPs, and FNs per class C_1, ..., C_N. 'G'=gold instance, 'S'=system instance.

4 Analysis of data properties

4.1 Frequency distribution of forms and instances in datasets

We hypothesize that only a few forms and a few instances are very frequent in corpora, while most appear only incidentally (*D1*). This represents a variation of Zipf's law (Zipf, 1935) for the EL task.

The log-log frequency distributions of forms and instances (Figure 1a and 1b) show an almost ideal Zipfian distribution in the case of AIDA (with a slope coefficient of -0.9085 for forms and -0.9657 for instances) and to a lesser extent for N3 (a slope of -0.4291 for forms and -0.5419 for instances). The less Zipfian curves of N3 are probably because this data collection is significantly smaller than AIDA.

The similar shape of the form and the instance distribution per dataset can be explained by the dependency between the two aspects. Namely, the form 'U.S.' denoting the instance `United_States` 462 times is reflected in both the form and the instance distributions. However, these two distributions are only identical if the ambiguity and variance are both 1. In practice, the mapping between forms and instances is M-to-N, i.e., other forms also denote `United_States` (such as 'America') and there are other instances referred to by a form 'US' (such as `United_States_dollar`).

[3]AGDISTIS API: `http://akswnc9.informatik.uni-leipzig.de:8113/AGDISTIS`, used on 24/05/2018.

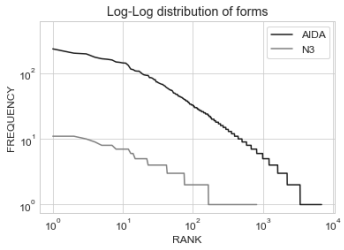
[4]Spotlight API: `http://spotlight.fii800.lod.labs.vu.nl/rest/disambiguate`, used on 24/05/2018.

[5]WAT API: `https://wat.d4science.org/wat/tag/json`, used on 24/05/2018.

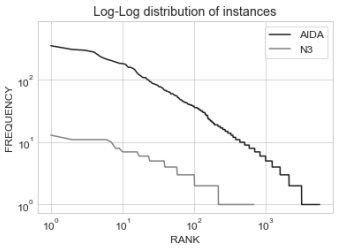
[6]All three APIs link to the Wikipedia dump from April 2016. Since the official DBpedia Spotlight endpoint at `http://model.dbpedia-spotlight.org/en/disambiguate` links to a newer Wikipedia version (February 2018 at the moment of writing of this paper), we set up our own endpoint that performs linking to the model 2016-04, to enable fair comparison with the other two systems. We reached similar conclusions with both versions of DBpedia Spotlight.

Figure 1: Log-log distribution of:

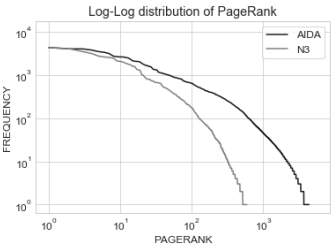
(a) form frequency



(b) instance frequency



(c) PageRank



4.2 PageRank distribution of instances in datasets

Similar to the instance frequency, we expect that a few instances in the corpora have an extremely high PageRank compared to most others (*D2*). Figure 1c shows the PageRank distribution of our both datasets. We observe that most entity mentions in text refer to instances with a low PageRank value, while only a few cases have a high PageRank value. Not surprisingly, the instance with the highest PageRank value (`United_States`) is at the same time the instance with the highest corpus frequency.

We inspect the effect of frequency and PageRank on system performance in Section 5.2.

4.3 Ambiguity distribution of forms

We hypothesize that only a small portion of all forms in a corpus are ambiguous (*D3*). As shown in Table 3, when both datasets are merged and NIL entities excluded, only 508 surface forms (6.73%) are ambiguous, as opposed to 7,037 monosemous forms (93.27%). This extremely high percentage validates our hypothesis. Moreover, in Sections 5.1, 5.3, and 5.4, we show that it also has a strong effect on systems performance.

	1	2	3	4	5	6	..	12
AIDA	6,400	359	78	29	7	3		1
N3	794	18	2	1	0	0		0
BOTH	7,037	381	84	29	10	3		1

Table 3: Ambiguity distribution per dataset, after NILs are excluded. Columns represent degrees of ambiguity.

4.4 Variance distribution of instances

We expect that only a small portion of all instances in a corpus are referred to with multiple forms (*D4*). The results of our variance analysis are given in Table 4. Over both datasets, 1,568 instances (25.61%) are referred to by multiple forms, as opposed to 4,555 instances (74.39%) which are always referred to by the same form. While the distribution of variance is much more even compared to that of ambiguity, we observe that most of the instances have a variance of 1.[7]

	1	2	3	4	5	6	7	8	9	10	11
AIDA	4,156	1,118	230	56	19	10	6	0	1	1	1
N3	550	106	15	7	1	0	0	0	0	0	0
BOTH	4,555	1,206	247	74	22	10	6	0	0	2	1

Table 4: Variance of instances with respect to the number of surface forms that reference them. Columns represent degrees of variance.

4.5 Interaction between frequency, PageRank, and ambiguity/variance

In the previous four Sections we analyzed the frequency distribution of individual data properties. Here we move forward to analyze their interaction. Figure 2 shows these results with mean as an estimator.[8]

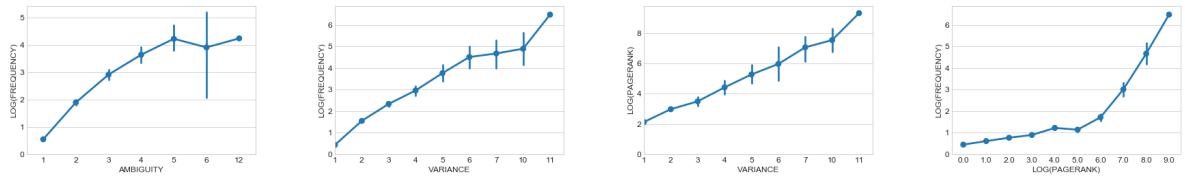
Firstly, we predict a positive dependency between ambiguity of forms and their frequency (*D5*). We expect that frequently used forms tend to receive new meanings, often as a result of metonymy or meronymy

[7] We expect that this skewness will also dominate system performance. For space reasons, we have not included this analysis.

[8] We observed comparable results with median as an estimator.

Figure 2: Correlations between long tail phenomena (estimator=mean):

(a) Ambiguity and frequency. (b) Variance and frequency. (c) Variance and PageRank. (d) PageRank and frequency.



of their dominant meaning. Figure 2a confirms this tendency, the Spearman correlation being 0.3772.

Secondly, we expect a positive correlation between variance of instances and their frequency (*D6*) or popularity (*D7*). Frequently mentioned and popular instances tend to be associated with more forms. Indeed, we observe that instances with higher frequency (Figure 2b) or PageRank (Figure 2c) typically have higher variance. The Spearman correlations are 0.6348 and 0.2542, respectively.

Thirdly, we compare the frequency of instances to their popularity measured with PageRank, predicting a positive correlation (*D8*). On average, this dependency holds (Figure 2d), though there are many frequent instances with low PageRank, or vice versa, leading to a Spearman correlation of 0.3281. The former are instances whose prominence coincides with the creation time of the corpus, but are not very well-known (e.g., the now-retired football player `Predrag_Mijatovic`). The latter are generally popular entities which were not captured sufficiently by the corpus, because their topical domain is marginal to this corpus (e.g., scientists), or they became relevant after the corpus release (emerging entities).

Hence, besides the high corpora skewness in terms of frequency, popularity, ambiguity, and variance, these factors also have positive interdependencies. Section 5 shows their effect on system performance.

4.6 Frequency distribution within a synset

We observed that the distribution of form frequency, instance frequency, and PageRank all have a Zipfian shape. But do we also see this behavior on a single form or instance level?

Supposedly, the frequency distribution within all forms that refer to an instance is Zipfian (*D9*). We test *D9* on the instance with highest variance and frequency, `United_States` (Figure 3). As expected, the vast majority of forms are used only seldom to refer to this instance, while in most cases a dominant short form "U.S." is used to make reference to this entity.

Figure 3: Form frequencies for the instance `United_States`

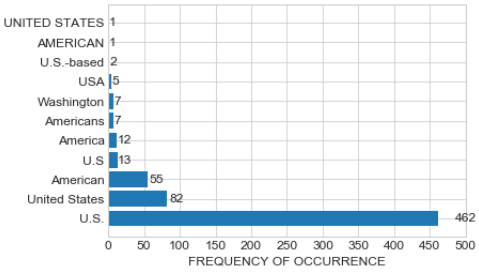


Figure 4a presents the frequency distribution of all instances that are referred to by the most ambiguous form in our data, "World cup". Figure 4b shows their PageRank distribution. In both cases, we observe a long-tail distribution among these instances (*D10*). Comparing them, we observe a clear match between frequency and PageRank, deviating only for instances that were prominent during the corpus creation, like `1998_FIFA_World_Cup`.

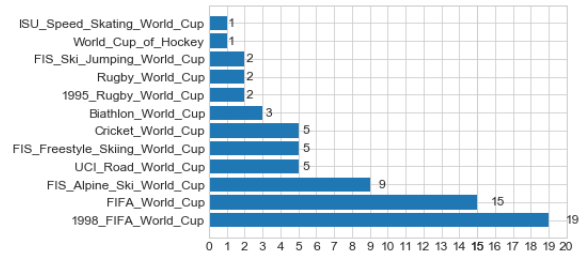
For analysis on the effect of frequency and popularity on system performance, please see Section 5.2.

5 Analysis of system performance and data properties

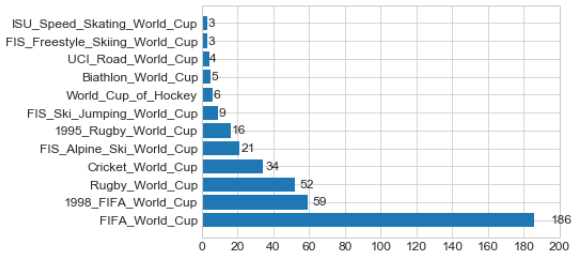
Next, we analyze systems performance in relation to the data properties: ambiguity (Section 5.1), form frequency, instance frequency, and PageRank (Section 5.2), as well as their combinations (5.3 and 5.4).

Figure 4: Distributions of the instances denoted by the most ambiguous form ("World Cup")

(a) Frequency distribution.



(b) PageRank distribution.

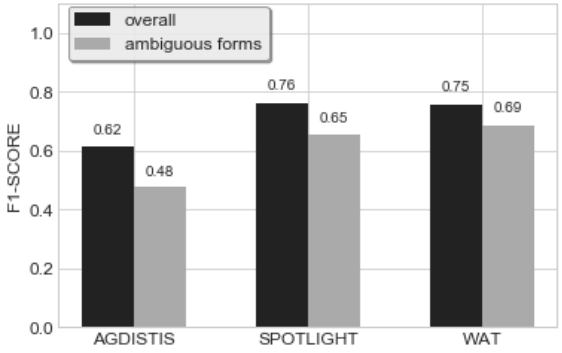


5.1 Correlating system performance with form ambiguity

Figure 5 displays the micro F1-scores of AGDISTIS MAG, Spotlight, and WAT on the two data collections jointly. For each system, we show its overall F1-score and F1-score on ambiguous forms only.

Section 4.3 showed that most of the forms in our corpora are not ambiguous. We expect that these forms lift the performance of systems, i.e., that they can resolve non-ambiguous forms easier than ambiguous ones (*S1*). Figure 5 confirms this for all systems: the F1-score on ambiguous forms is between 6 and 14 absolute points lower than the overall F1-score.

Figure 5: Micro F1-scores of systems: overall and on ambiguous subsets



When computing macro- instead of micro-F1 scores, we observe similar findings for *S1*. Interestingly, the macro-F1 scores are consistently lower than the micro-F1 scores, especially in case of the ambiguous subsets evaluation. Namely, the overall macro-F1 scores are between 0.44 and 0.52, and between 0.14 and 0.34 on the ambiguous forms. This suggests that frequent forms boost system performance, especially on ambiguous surface forms. We investigate this further in the next Sections.

5.2 Correlating system performance with form frequency, instance frequency, and PageRank

Next, we consider frequency of forms and instances, as well as PageRank values of instances in relation to system performance. For each of these, we expect a positive correlation with system performance (*S2*), suggesting that systems perform better on frequent and popular cases, compared to non-popular and infrequent ones.

	all forms			ambiguous forms only		
	FF-F1	**FI-F1**	**PR-F1**	**FF-F1**	**FI-F1**	**PR-F1**
AGDISTIS	0.2739	0.3812	0.1465	0.3550	0.4073	0.3969
Spotlight	0.1321	0.1847	0.1357	0.3986	0.4196	0.3108
WAT	0.4663	0.5050	0.3164	0.5831	0.5319	0.4214

Table 5: Correlation between F1-score and: frequency of forms (FF-F1), frequency of instances (FI-F1), and PageRank (PR-F1). Left: on all forms, right: only on ambiguous forms.

The Spearman correlation for each of the systems and properties, over all forms, are shown in Table 5 (left half). While most of the correlation for frequency and popularity is positive, the values are in general relatively low (WAT being an exception). This shows that frequency/popularity by itself contributes, but is not sufficient to explain system performance. The right half of the Table shows the same metrics when applied to the ambiguous forms. We observe an increase in all values, which means that frequency and popularity are most relevant when multiple instances 'compete' sharing a form. These findings are in line with those in Section 5.1.

5.3 Correlating system performance with ambiguity and frequency of forms jointly

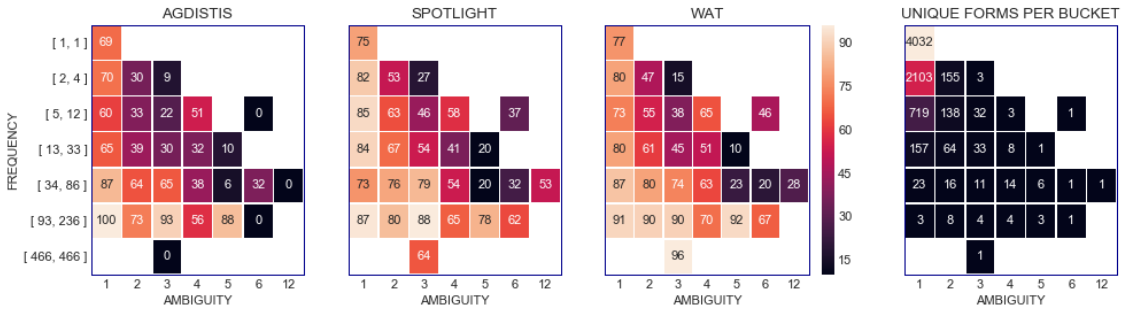


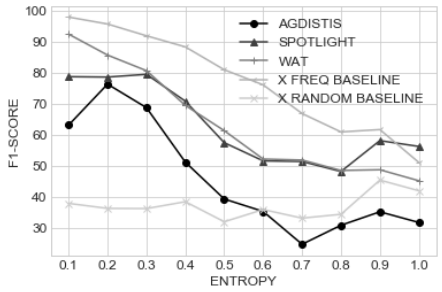
Figure 6: F1 per (ambiguity, frequency) pair. The first three heat maps show the F1-scores of systems per ambiguity degree and frequency range. The last plot shows the amount of unique forms in each cell.

We have shown that system performance is lower on ambiguous than unambiguous forms. We also observed a tendency of systems to perform better on frequent forms and instances as compared to infrequent ones. But how does performance differ across different levels of ambiguity? How do ambiguity and form frequency interact as a joint predictor of performance? The heat maps in Figure 6 show the interplay between ambiguity, frequency, and micro F1-score for each of the systems. Generalizing over the discussion in Sections 5.1 and 5.2, we observe the best scores on frequent, non-ambiguous forms (bottom-left), and worst F1-scores on non-frequent, highly ambiguous forms (top-right) (*S3*).

In Section 5.4, we investigate if some instances within ambiguous forms are more difficult than others.

5.4 Correlating system performance with frequency of instances for ambiguous forms

Figure 7: Micro F1-score per entropy bucket.



To measure the instance distribution within individual forms, we employ the notion of *normalized entropy*, similarly as Ilievski et al. (2016). The entropy of a form i with n_i instances and N_i occurrences is: $H_i = (-\sum_{j=1}^{n_i} p_{i,j} log p_{i,j})/log_2 N_i$, where $p_{i,j}$ is the probability that the instance j is denoted by the form i. For non-ambiguous forms $H_i = 0$, while forms with uniform frequency distribution of instances have a maximum $H_i = 1$. We predict an inverse correlation between system performance and entropy (*S4*). The results in Figure 7 show a dramatic drop in micro F1-score for uniformly distributed cases (high entropy) compared to skewed ones (low entropy). We compare these shapes to two baselines: frequency baseline, that picks the most frequent instance for a form on the gold data, and a random baseline, choosing one of the gold instances for a form at random. All three systems have a similar curve shape to the frequency baseline, whereas out of the three systems Spotlight's curve comes closest to that of the random baseline.

We also compute the macro F1-score per entropy bucket to help us understand whether the drop in performance in Figure 7 is due to: 1. a qualitative difference between low and high entropy forms, or 2. an overfitting of systems to the frequent interpretations of ambiguous forms. The macro F1-score reduces the effect of frequency on performance, by evaluating each form-instance pair once. We observe that the macro F1-scores are much more balanced across the entropy buckets compared to the micro F1-scores, and especially lower on the buckets with higher skewness (low entropy). This suggests that the high micro F1-score for low entropies is heavily based on frequent instances.

As a final analysis, we seek to understand whether frequent/popular instances of a form are indeed resolved easier than less frequent/popular instances of the same form. For that purpose, we pick the set of all ambiguous forms, and we rank their instances by relative frequency/PageRank value.

Considering the most ambiguous form "World Cup" as an example and ranking by frequency, its $r1$ (rank 1) instance is `1998_FIFA_World_Cup`, $r2$ is `FIFA_World_Cup`, ..., and $r12$ is `ISU_Speed_Skating_World_Cup`. We expect systems to perform much better on frequent instances of ambiguous forms, compared to infrequent instances of ambiguous forms, i.e., we expect F1-scores to decrease when moving from higher to lower ranks (*S5*). Figure 8 shows that our hypothesis holds to a large extent for precision, recall, and F1-scores, except for the occasional peaks at $r6$ and $r9$.

Figure 8: Precision, recall, and micro F1-score per instance frequency rank, averaged over the systems.

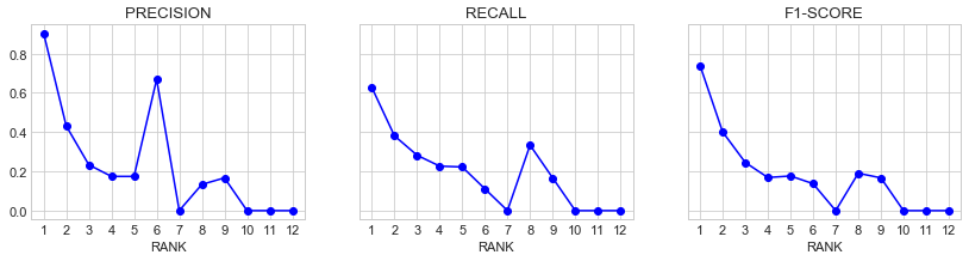
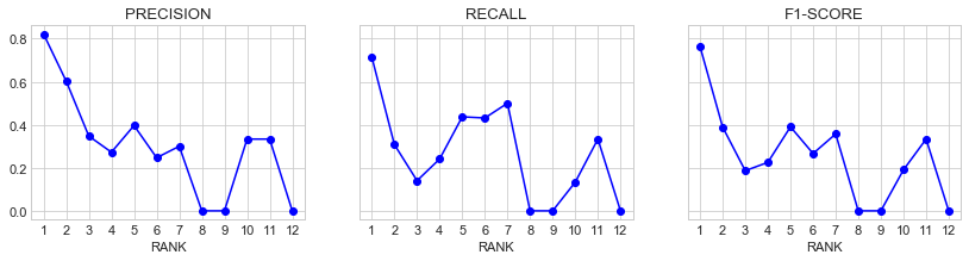


Figure 9: Precision, recall, and micro F1-score per PageRank-based rank, averaged over the systems.



Similarly, we order the instances denoting a form based on their relative PageRank value. We hypothesize that systems perform better on popular instances of ambiguous forms, compared to their unpopular instances (*S6*). Although less monotonic than the frequency ones in Figure 8, the resulting shapes of this analysis in Figure 9 suggest that popularity can also be applied to estimate system performance.

6 Summary of findings

We noted a positive correlation between ambiguity and frequency of forms, as well as between variance and frequency of instances. We noticed that the distribution of instances overall, but also per form, has a Zipfian shape. Similarly, the distribution of forms, both on individual and on aggregated level, is Zipfian. While some of these distributions are well-known in the community for words, this is the first time they have been systematically analyzed for surface forms of entities, their meaning and reference, and empirically connected with the performance of systems.

We observed that ambiguity of forms leads to a notable decline in system performance. Coupling it with frequency, we measured that low-frequent, highly ambiguous forms yield the lowest performance, while very frequent, non-ambiguous forms yield the highest performance. The entropy of forms, capturing the frequency distribution of their denoted instances, revealed that balanced distributions tend to be harder for systems, with the micro F1-value dropping with 20-40 absolute points between the highest and lowest entropy. Finally, the higher performance on skewed cases was shown to be a result of overfitting to the most frequent/popular instances.

Based on these outcomes, we can conclude that the intersection of ambiguity and frequency/popularity is a good estimator of the complexity of the EL task. The hard cases of EL should be sought among the low-frequent and unpopular candidates of highly ambiguous forms.

7 Recommended actions

We have shown that there are systematic differences between the head and the tail of the EL task, and that these reflect on how systems perform. Provided that systems show a weakness on tail cases, and that this weakness is simultaneously hidden by averaged evaluation numbers, how can we overcome this obstacle in practice? Here we list three recommendations:

1. When **creating a dataset**, we propose authors to include statistics on the head and the tail properties (ambiguity, variance, frequency, and popularity) of the data, together with a most-frequent-value baseline. By doing so, the community would be informed about the hard cases in that dataset, as well as about the portion of the dataset that can be resolved by following simple statistical strategies.

2. When **evaluating** a system, we suggest splitting of all cases into head and tail ones. Afterwards, head and tail cases can be evaluated separately, as well as together. This provides a direct insight into the differences in scoring of the tail cases compared to the head cases, potentially signaling aspects of the EL tail that are challenging for the given system. In addition, the frequency skewness of head cases can be largely decreased by employing a macro instead of micro F1-score, as shown in this paper.

3. In addition to the suggestion in 2., when **developing or training** a system, it should be made explicit which heuristics target which cases, and to what extent resources and training data optimize for the target dataset in relation to the head and tail distributions.

8 Conclusions

Although past research has argued that the performance of EL systems declines when moving from the head to the tail of the entity distribution, the long tail has not been quantified so far, preventing one to distinguish head and tail cases in the EL task. Previous linguistic studies on words distributions can also not be applied for this purpose since they do not study reference. This paper is the first one that systematically looks into the relation of surface forms in EL corpora and instances in DBpedia, and provides a series of hypotheses on what long tail phenomena are. We analyzed existing EL datasets with respect to these long tail properties, demonstrating that data properties have certain inter-correlations that follow our hypotheses. Next, we investigated their effect on the performance of three state-of-the-art systems, proving that the long tail phenomena and their interaction consistently predict system performance. Namely, we noted a positive dependency of system performance on frequency and popularity of instances, and a negative one with ambiguity of forms. Our findings in this paper are meant to influence future designs of both EL systems and evaluation datasets. To support this goal, we listed three recommended actions to be considered when creating a dataset, evaluating a system, or developing a system in the future.

We see two directions for future improvement of our analysis: 1. To obtain a corpus-independent inventory of forms and their candidate instances, both with their corresponding frequencies, is a challenge in the case of EL and no existing resource can be assumed to be satisfactory in this regard (for Word Sense Disambiguation, this is usually WordNet). We approximated these based on the corpora we analyzed, but considering the fairly small size of most EL datasets, this poses a limitation to our current analysis. 2. Some of our current numbers are computed only on a handful of cases. This leads to unexpected disturbances in our results, like the occasional peaks for high ranks in Figure 8. We expect the outcome of this analysis to gain significance when more large EL datasets become available in the future.

Acknowledgements

We would like to thank Eduard Hovy, Frank van Harmelen, and Marieke van Erp for their valuable suggestions. In addition, we thank the anonymous reviewers for their feedback. The research for this paper was supported by the Netherlands Organisation for Scientific Research (NWO) via the Spinoza fund.

References

Álvaro Corral, Gemma Boleda, and Ramon Ferrer-i Cancho. 2015. Zipfs law for word frequencies: Word forms versus lemmas in long texts. *PloS one*, 10(7):e0129031.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 121–124. ACM.

José Esquivel, Dyaa Albakour, Miguel Martinez, David Corney, and Samir Moussa. 2017. On the Long-Tail Entities in News. In *European Conference on Information Retrieval*, pages 691–697.

H Paul Grice. 1975. Logic and conversation. *1975*, pages 41–58.

Erkki Heino, Minna Tamper, Eetu Mäkelä, Petri Leskinen, Esko Ikkala, Jouni Tuominen, Mikko Koho, and Eero Hyvönen. 2017. Named entity linking in a complex domain: Case second world war history. In *International Conference on Language, Data and Knowledge*, pages 120–133. Springer.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics.

Johannes Hoffart, Yasemin Altun, and Gerhard Weikum. 2014. Discovering emerging entities with ambiguous names. In *Proceedings of the 23rd international conference on World wide web*, pages 385–396. ACM.

Filip Ilievski, Marten Postma, and Piek Vossen. 2016. Semantic overfitting: what world do we consider when evaluating disambiguation of text? In *proceedings of COLING*.

Filip Ilievski, Piek Vossen, and Marieke Van Erp. 2017. Hunger for Contextual Knowledge and a Road Map to Intelligent Entity Linking. In *International Conference on Language, Data and Knowledge*, pages 143–149. Springer.

Heng Ji and Ralph Grishman. 2011. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1148–1158. Association for Computational Linguistics.

Jasmeen Kanwal, Kenny Smith, Jennifer Culbertson, and Simon Kirby. 2017. Zipfs Law of Abbreviation and the Principle of Least Effort: Language users optimise a miniature lexicon for efficient communication. *Cognition*, 165:45–52.

Diego Moussallem, Ricardo Usbeck, Michael Röeder, and Axel-Cyrille Ngonga Ngomo. 2017. MAG: A Multilingual, Knowledge-base Agnostic and Deterministic Entity Linking Approach. In *Proceedings of the Knowledge Capture Conference*, page 9. ACM.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Francesco Piccinno and Paolo Ferragina. 2014. From TagME to WAT: a new entity annotator. In *ERD@SIGIR*.

Marten Postma, Ruben Izquierdo, Eneko Agirre, German Rigau, and Piek Vossen. 2016. Addressing the MFS Bias in WSD systems. In *Proceedings of LREC 2016*, Paris, France. ELRA.

Michael Röder, Ricardo Usbeck, Sebastian Hellmann, Daniel Gerber, and Andreas Both. 2014. N^3-A Collection of Datasets for Named Entity Recognition and Disambiguation in the NLP Interchange Format. In *LREC*, pages 3529–3533.

Marieke Van Erp, Filip Ilievski, Marco Rospocher, and Piek Vossen. 2015. Missing Mr. Brown and Buying an Abraham Lincoln-Dark Entities and DBpedia. In *NLP-DBPEDIA@ ISWC*, pages 81–86.

Marieke Van Erp, P Mendes, Heiko Paulheim, Filip Ilievski, Julien Plu, Giuseppe Rizzo, and Jörg Waitelonis. 2016. Evaluating entity linking: An analysis of current benchmark datasets and a roadmap for doing a better job. In *LREC. ELRA*.

Jin G Zheng, Daniel Howsmon, Boliang Zhang, Juergen Hahn, Deborah McGuinness, James Hendler, and Heng Ji. 2015. Entity linking for biomedical literature. *BMC medical informatics and decision making*, 15(1):S4.

George Zipf. 1935. *The Psychobiology of Language: An Introduction to Dynamic Philology*. M.I.T. Press, Cambridge, Mass.

Neural Collective Entity Linking

Yixin Cao, Lei Hou*, Juanzi Li, Zhiyuan Liu
Dept. of Computer Science and Technology, Tsinghua University, China 100084
{caoyixin2011,greener2009,lijuanzi2008}@gmail.com
liuzy@tsinghua.edu.cn

Abstract

Entity Linking aims to link entity mentions in texts to knowledge bases, and neural models have achieved recent success in this task. However, most existing methods rely on local contexts to resolve entities independently, which may usually fail due to the data sparsity of local information. To address this issue, we propose a novel neural model for collective entity linking, named as NCEL. NCEL applies Graph Convolutional Network to integrate both local contextual features and global coherence information for entity linking. To improve the computation efficiency, we approximately perform graph convolution on a subgraph of adjacent entity mentions instead of those in the entire text. We further introduce an attention scheme to improve the robustness of NCEL to data noise and train the model on Wikipedia hyperlinks to avoid overfitting and domain bias. In experiments, we evaluate NCEL on five publicly available datasets to verify the linking performance as well as generalization ability. We also conduct an extensive analysis of time complexity, the impact of key modules, and qualitative results, which demonstrate the effectiveness and efficiency of our proposed method.

1 Introduction

Entity linking (EL), mapping entity mentions in texts to a given knowledge base (KB), serves as a fundamental role in many fields, such as question answering (Zhang et al., 2016), semantic search (Blanco et al., 2015), and information extraction (Ji et al., 2015; Ji et al., 2016). However, this task is non-trivial because entity mentions are usually ambiguous. As shown in Figure 1, the mention *England* refers to three entities in KB, and an entity linking system should be capable of identifying the correct entity as *England cricket team* rather than *England* and *England national football team*.

Entity linking is typically broken down into two main phases: (i) candidate generation obtains a set of referent entities in KB for each mention, and (ii) named entity disambiguation selects the possible candidate entity by solving a ranking problem. The key challenge lies in the ranking model that computes the relevance between candidates and the corresponding mentions based on the information both in texts and KBs (Nguyen et al., 2016). In terms of the features used for ranking, we classify existing EL models into two groups: **local models** to resolve mentions independently relying on textual context information from the surrounding words (Chen and Ji, 2011; Chisholm and Hachey, 2015; Lazic et al., 2015; Yamada et al., 2016), and **global (collective) models**, which are the main focus of this paper, that encourage the target entities of all mentions in a document to be topically coherent (Han et al., 2011; Cassidy et al., 2012; He et al., 2013b; Cheng and Roth, 2013; Durrett and Klein, 2014; Huang et al., 2014).

Global models usually build an entity graph based on KBs to capture coherent entities for all identified mentions in a document, where the nodes are entities, and edges denote their relations. The graph provides highly discriminative semantic signals (e.g., entity relatedness) that are unavailable to local model (Eshel et al., 2017). For example (Figure 1), an EL model seemly cannot find sufficient disambiguation clues for the mention *England* from its surrounding words, unless it utilizes the coherence

*Corresponding author.

Proceedings of the 27th International Conference on Computational Linguistics, pages 675–686
Santa Fe, New Mexico, USA, August 20-26, 2018.

information of consistent topic "cricket" among adjacent mentions *England*, *Hussain*, and *Essex*. Although the global model has achieved significant improvements, its limitation is threefold:

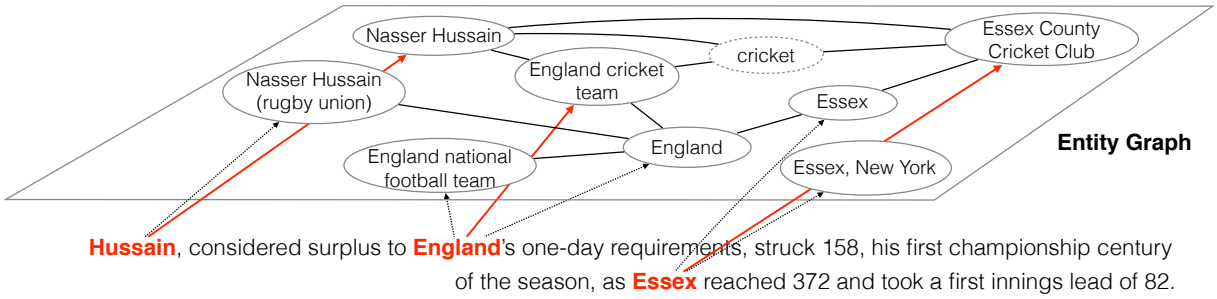


Figure 1: Illustration of named entity disambiguation for three mentions *England*, *Hussain*, and *Essex*. The nodes linked by arrowed lines are the candidate entities, where red solid lines denote target entities.

1. The global approach suffers from the data sparsity issue of unseen words/entities, and the failure to induce underlying discriminative features for EL.

2. The joint inference mechanism in the global approach leads to expensive computations, especially when the entity graph may contain hundreds of nodes in case of long documents.

3. The annotated EL training data is usually expensive to obtain or only available in narrow domains, which results in possible overfitting issue or domain bias.

To mitigate the first limitation, recent EL studies introduce neural network (NN) models due to its amazing feature abstraction and generalization ability. In such models, words/entities are represented by low dimensional vectors in a continuous space, and features for mention as well as candidate entities are automatically learned from data (Nguyen et al., 2016). However, existing NN-based methods for EL are either local models (Yamada et al., 2017; Gupta et al., 2017) or merely use word/entity embeddings for feature extraction and rely on another modules for collective disambiguation, which thus cannot fully utilize the power of NN models for collective EL (Globerson et al., 2016; Guo and Barbosa, 2017; Phan et al., 2018).

The second drawback of the global approach has been alleviated through approximate optimization techniques, such as PageRank/random walks (Pershina et al., 2015), graph pruning (Hoffart et al., 2011), ranking SVMs (Ratinov et al., 2011), or loopy belief propagation (LBP) (Globerson et al., 2016; Ganea and Hofmann, 2017). However, these methods are not differentiable and thus difficult to be integrated into neural network models (the solution for the first limitation).

To overcome the third issue of inadequate training data, (Gupta et al., 2017) has explored a massive amount of hyperlinks in Wikipedia, but these potential annotations for EL contain much noise, which may distract a naive disambiguation model (Chisholm and Hachey, 2015).

In this paper, we propose a novel **N**eural **C**ollective **E**ntity **L**inking model (NCEL), which performs global EL combining deep neural networks with Graph Convolutional Network (GCN) (Defferrard et al., 2016; Kipf and Welling, 2017) that allows flexible encoding of entity graphs. It integrates both local contextual information and global interdependence of mentions in a document, and is efficiently trainable in an end-to-end fashion. Particularly, we introduce attention mechanism to robustly model local contextual information by selecting informative words and filtering out the noise. On the other hand, we apply GCNs to improve discriminative signals of candidate entities by exploiting the rich structure underlying the correct entities. To alleviate the global computations, we propose to convolute on the subgraph of adjacent mentions. Thus, the overall coherence shall be achieved in a chain-like way via a sliding window over the document. To the best of our knowledge, this is the first effort to develop a unified model for neural collective entity linking.

In experiments, we first verify the efficiency of NCEL via theoretically comparing its time complexity with other collective alternatives. Afterwards, we train our neural model using collected Wikipedia hyperlinks instead of dataset-specific annotations, and perform evaluations on five public available benchmarks. The results show that NCEL consistently outperforms various baselines with a favorable generalization ability. Finally, we further present the performance on a challenging dataset WW (Guo and Barbosa, 2017) as well as qualitative results, investigating the effectiveness of each key module.

2 Preliminaries and Framework

We denote $M = \{m_i\}$ as a set of entity mentions in a document $D = \langle x_1, \ldots, x_i, \ldots, x_{|D|} \rangle$, where x_i is either a word w_i or a mention m_i. $G = (E, R)$ is the entity graph for document D derived from the given knowledge base, where $E = \{e_i\}$ is a set of entities, $R = \{r_j^i \in (0, 1]\}$ denotes the relatedness between $\langle e_i, e_j \rangle$ and higher values indicate stronger relations. Based on G, we extract a subgraph G^{ij} for $e_j \in \Phi(m_i)$, where $\Phi(m_i)$ denotes the set of candidate entities for m_i. Note that we don't include the relations among candidates of the same mention in G^{ij} because these candidates are mutually exclusive in disambiguation.

Formally, we define the entity linking problem as follows: Given a set of mentions M in a document D, and an entity graph G, the goal is to find an assignment[1] $\Gamma : M \rightarrow E$.

To collectively find the best assignment, NCEL aims to improve the discriminability of candidates' local features by using entity relatedness within a document via GCN, which is capable of learning a function of features on the graph through shared parameters over all nodes. Figure 2 shows the framework of NCEL including three main components:

1. **Candidate Generation**: we use a pre-built dictionary to generate a set of entities as candidates to be disambiguated for each mention, e.g., for mention *England*, we have $\Phi(m_i) = \{e_1^i, e_2^i, e_3^i\}$, in which the entities refer to *England national football team*, *England* and *England cricket team* in Figure 1, respectively.

2. **Feature Extraction**: based on the document and its entity graph, we extract both local features and global features for each candidate entity to feed our neural model. Concretely, local features reflect the compatibility between a candidate and its mention within the contexts, and global features are to capture the topical coherence among various mentions. These features, including vectorial representations of candidates and a subgraph indicating their relatedness, are highly discriminative for tackling ambiguity in EL.

3. **Neural Model**: given feature vectors and subgraphs of candidates, we first encode the features to represent nodes (i.e., candidates) in the graph, then improve them for disambiguation via multiple graph convolutions by exploiting the structure information, in which the features for correct candidates that are strongly connected (i.e., topical coherent) shall enhance each other, and features for incorrect candidates are weakened due to their sparse relations. Finally, we decode the features of nodes to output a probability indicating how possible the candidate refers to its mention.

Example As shown in Figure 2, for the current mention *England*, we utilize its surrounding words as local contexts (e.g., *surplus*), and adjacent mentions (e.g., *Hussian*) as global information. Collectively, we utilize the candidates of *England* $e_j^i \in \Phi(m_i), j = 1, 2, 3$ as well as those entities of its adjacencies $\Phi(m_{i-1}) \cup \Phi(m_{i+1})$ to construct feature vectors for e_j^i and the subgraph of relatedness as inputs of our neural model. Let darker blue indicate higher probability of being predicted, the correct candidate e_3^i becomes bluer due to its bluer neighbor nodes of other mentions m_{i-1}, m_{i+1}. The dashed lines denote entity relations that have indirect impacts through the sliding adjacent window , and the overall structure shall be achieved via multiple sub-graphs by traversing all mentions.

Before introducing our model, we first describe the component of candidate generation.

[1]Normally, an entity linking system outputs NIL for a mention when no assignment score is higher than a threshold. This is application-specific and thus outside of the scope of this work.

2.1 Candidate Generation

Similar to previous work (Ganea and Hofmann, 2017), we use the prior probability $\hat{p}(e_i|m_j)$ of entity e_i conditioned on mention m_j both as a local feature and to generate candidate entities: $\Phi(m_j) = \{e_i|\hat{p}(e_i|m_j) > 0\}$. We compute $\hat{p}(\cdot)$ based on statistics of mention-entity pairs from: (i) Wikipedia page titles, redirect titles and hyperlinks, (ii) the dictionary derived from a large Web Corpus (Spitkovsky and Chang, 2012), and (iii) the YAGO dictionary with a uniform distribution (Hoffart et al., 2011). We pick up the maximal prior if a mention-entity pair occurs in different resources. In experiments, to optimize for memory and run time, we keep only top n entities based on $\hat{p}(e_i|m_j)$. In the following two sections, we will present the key components of NECL, namely feature extraction and neural network for collective entity linking.

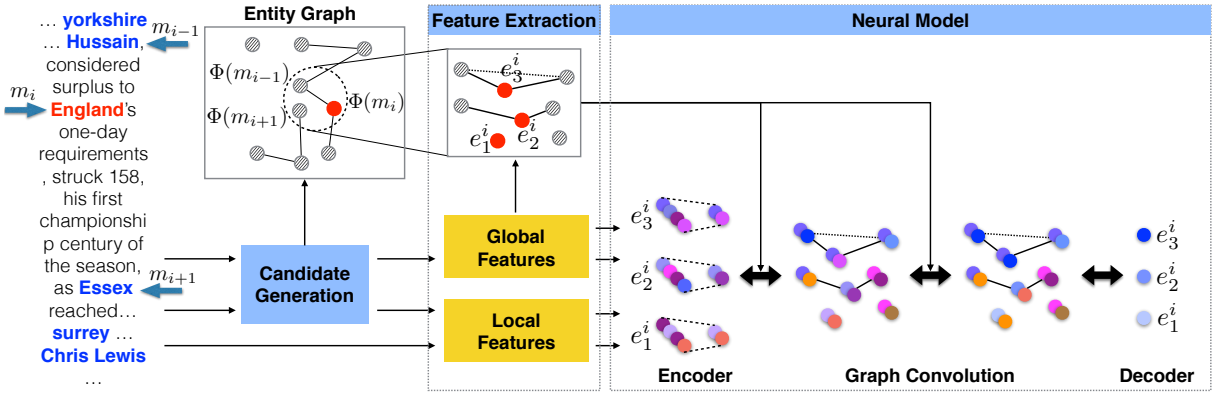


Figure 2: Framework of NCEL. The inputs of a set of mentions in a document are listed in the left side. The words in red indicate the current mention m_i, where m_{i-1}, m_{i+1} are neighbor mentions, and $\Phi(m_i) = \{e_1^i, e_2^i, e_3^i\}$ denotes the candidate entity set for m_i.

3 Feature Extraction

The main goal of NCEL is to find a solution for collective entity linking using an end-to-end neural model, rather than to improve the measurements of local textual similarity or global mention/entity relatedness. Therefore, we use joint embeddings of words and entities at sense level (Cao et al., 2017) to represent mentions and its contexts for feature extraction. In this section, we give a brief description of our embeddings followed by our features used in the neural model.

3.1 Learning Joint Embeddings of Word and Entity

Following (Cao et al., 2017), we use Wikipedia articles, hyperlinks, and entity outlinks to jointly learn word/mention and entity embeddings in a unified vector space, so that similar words/mentions and entities have similar vectors. To address the ambiguity of words/mentions, (Cao et al., 2017) represents each word/mention with multiple vectors, and each vector denotes a sense referring to an entity in KB. The quality of the embeddings is verified on both textual similarity and entity relatedness tasks.

Formally, each word/mention has a global embedding $\mathbf{w}_i/\mathbf{m}_i$, and multiple sense embeddings $\mathcal{S}(x_i) = \{\mathbf{s}_j\}$. Each sense embedding $\mathbf{s}_j$ refers to an entity embedding $\mathbf{e}_j$, while the difference between $\mathbf{s}_j$ and $\mathbf{e}_j$ is that $\mathbf{s}_j$ models the co-occurrence information of an entity in texts (via hyperlinks) and $\mathbf{e}_j$ encodes the structured entity relations in KBs. More details can be found in the original paper.

3.2 Local Features

Local features focus on how compatible the entity is mentioned in a piece of text (i.e., the mention and the context words). Except for the prior probability (Section 2.1), we define two types of local features for each candidate entity $e_j \in \Phi(m_i)$:

String Similarity Similar to (Yamada et al., 2017), we define string based features as follows: the edit distance between mention's surface form and entity title, and boolean features indicating whether they are equivalent, whether the mention is inside, starts with or ends with entity title and vice versa.

Compatibility We also measure the compatibility of e_j with the mention's context words $\mathcal{C}(m_i)$ by computing their similarities based on joint embeddings: $sim(\mathbf{e}_j, \mathbf{c}_{m_i,e_j})$ and $sim(\mathbf{s}_j, \mathbf{c}_{m_i,e_j})$, where $\mathbf{c}_{m_i,e_j}$ is the context embedding of m_i conditioned on candidate e_j and is defined as the average sum of word global vectors weighted by attentions:

$$\mathbf{c}_{m_i,e_j} = \sum_{w_k \in \mathcal{C}(m_i)} \alpha_{kj} \mathbf{w}_k$$

where α_{kj} is the k-th word's attention from e_j. In this way, we automatically select informative words by assigning higher attention weights, and filter out irrelevant noise through small weights. The attention α_{kj} is computed as follows:

$$\alpha_{kj} \propto sim(\mathbf{w}_k, \mathbf{e}_j)$$

where sim is the similarity measurement, and we use cosine similarity in the presented work. We concatenate the prior probability, string based similarities, compatibility similarities and the embeddings of contexts as well as the entity as the local feature vectors.

3.3 Global Features

The key idea of collective EL is to utilize the topical coherence throughout the entire document. The consistency assumption behind it is that: *all mentions in a document shall be on the same topic*. However, this leads to exhaustive computations if the number of mentions is large. Based on the observation that the consistency attenuates along with the distance between two mentions, we argue that the adjacent mentions might be sufficient for supporting the assumption efficiently.

Formally, we define neighbor mentions as q adjacent mentions before and after current mention m_i: $\mathcal{N}(m_i) = \{m_{i-q}, \ldots, m_{i-1}, m_{i+1}, \ldots, m_{i+q}\}$, where $2q$ is the pre-defined window size. Thus, the topical coherence at document level shall be achieved in a chain-like way. As shown in Figure 2 ($q = 1$), mentions *Hussain* and *Essex*, a cricket player and the cricket club, provide adequate disambiguation clues to induce the underlying topic "cricket" for the current mention *England*, which impacts positively on identifying the mention *surrey* as another cricket club via the common neighbor mention *Essex*.

A degraded case happens if q is large enough to cover the entire document, and the mentions used for global features become the same as the previous work, such as (Pershina et al., 2015). In experiments, we heuristically found a suitable $q = 3$ which is much smaller than the total number of mentions. The benefits of efficiency are in two ways: (i) to decrease time complexity, and (ii) to trim the entity graph into a fixed size of subgraph that facilitates computation acceleration through GPUs and batch techniques, which will be discussed in Section 5.2.

Given neighbor mentions $\mathcal{N}(m_i)$, we extract two types of vectorial global features and structured global features for each candidate $e_j \in \Phi(m_i)$:

Neighbor Mention Compatibility Suppose neighbor mentions are topical coherent, a candidate entity shall also be compatible with neighbor mentions if it has a high compatibility score with the current mention, otherwise not. That is, we extract the vectorial global features by computing the similarities between e_j and all neighbor mentions: $\{sim(\mathbf{e}_j, \mathbf{m}_i) | m_i \in \mathcal{N}(m_i)\}$, where $\mathbf{m}_i$ is the mention embedding by averaging the global vectors of words in its surface form: $\mathbf{m}_j = \sum_{w_l \in \mathcal{T}(m_j)} \mathbf{w}_l$, where $\mathcal{T}(m_j)$ are tokenized words of mention m_j.

Subgraph Structure The above features reflect the consistent semantics in texts (i.e., mentions). We now extract structured global features using the relations in KB, which facilitates the inference among candidates to find the most topical coherent subset. For each document, we obtain the entity graph G by taking candidate entities of all mentions $\Phi(M)$ as nodes, and using entity embeddings to compute their

similarities as edges $R = \{r_j^i | r_j^i = sim(\mathbf{e}_i, \mathbf{e}_j)\}$. Then, we extract the subgraph structured features $\mathbf{g}^{i*}$ for each entity $e_*^i \in \Phi(m_i), m_i \in M$ for efficiency.

Formally, we define the subgraph as: $G^{i*} = (e_*^i \bigcup \Phi(\mathcal{N}(m_i)), R^{i*})$, where $R^{i*} = \{r_{jk}^{i*} | e_k^j \in \Phi(m_j), j \in [i-q, i+q] \setminus i\}$. For example (Figure 1), for entity *England cricket team*, the subgraph contains the relation from it to all candidates of neighbor mentions: *England cricket team*, *Nasser Hussain (rugby union)*, *Nasser Hussain*, *Essex*, *Essex County Cricket Club* and *Essex, New York*. To support batch-wise acceleration, we represent G^{i*} in the form of adjacency table based vectors: $\mathbf{g}^{i*} = [r_{i-q,1}^{i*}, \cdots, r_{i+q,n}^{i*}]^T \in \mathbb{R}^{2qn}$, where n is the number of candidates per mention.

Finally, for each candidate e_j^i, we concatenate local features and neighbor mention compatibility scores as the feature vector $\mathbf{f}^{ij}$, and construct the subgraph structure representation $\mathbf{g}^{ij}$ as the inputs of NCEL.

4 Neural Collective Entity Linking

NCEL incorporates GCN into a deep neural network to utilize structured graph information for collectively feature abstraction, while differs from conventional GCN in the way of applying the graph. Instead of the entire graph, only a subset of nodes is "visible" to each node in our proposed method, and then the overall structured information shall be reached in a chain-like way. Fixing the size of the subset, NCEL is further speeded up by batch techniques and GPUs, and is efficient to large-scale data.

4.1 Graph Convolutional Network

GCNs are a type of neural network model that deals with structured data. It takes a graph as an input and output labels for each node. As a simplification of spectral graph convolutions, the main idea of (Kipf and Welling, 2017) is similar to a propagation model: to enhance the features of a node according to its neighbor nodes. The formulation is as follows:

$$H^{l+1} = \sigma(\tilde{A} H^l W^l)$$

where $\tilde{A}$ is a normalized adjacent matrix of the input graph with self-connection, H^l and W^l are the hidden states and weights in the l-th layer, and $\sigma(\cdot)$ is a non-linear activation, such as *ReLu*.

4.2 Model Architecture

As shown in Figure 2, NCEL identifies the correct candidate e_1^i for the mention m_i by using vectorial features as well as structured relatedness with candidates of neighbor mentions $\Phi(m_{i-1}), \Phi(m_{i+1})$. Given feature vector $\mathbf{f}^{ij} \in \mathbb{R}^{d_0}$ and subgraph representation $\mathbf{g}^{ij} \in \mathbb{R}^{2qn}$ of each candidate $e_j^i \in \Phi(m_i)$, we stack them as inputs for mention[2] m_i: $\mathbf{f} = [\mathbf{f}_{i1}, \cdots, \mathbf{f}_{in}]^T \in \mathbb{R}^{n \times d_0}$, and the adjacent matrix $A = [\hat{\mathbf{g}}^1, \cdots, \hat{\mathbf{g}}_n]^T \in \mathbb{R}^{n \times (2qn+1)}$, where $\hat{\mathbf{g}}^j = [\mathbf{g}^j, 1]^T \in \mathbb{R}^{2qn+1}$ denotes the subgraph with self-connection. We normalize A such that all rows sum to one, denoted as $\tilde{A}$, avoiding the change in the scale of the feature vectors.

Given $\mathbf{f}$ and $\tilde{A}$, the goal of NCEL is to find the best assignment:

$$\Gamma^*(m_i) = \underset{\hat{y}}{\text{argmax}}\ P(\hat{y}; \mathbf{f}, \tilde{A}, \omega)$$

where $\hat{y}$ is the output variable of candidates, and $P(\cdot)$ is a probability function as follows:

$$P(\hat{y}; \mathbf{f}, \tilde{A}, \omega) \propto \exp(F(\mathbf{f}, \tilde{A}, \hat{y}; \omega))$$

where $F(\mathbf{f}, \tilde{A}, \hat{y}; \omega)$ is the score function parameters by $\omega \in \mathbb{R}^\omega$. NCEL learns the mapping $F(\cdot)$ through a neural network including three main modules: encoder, sub-graph convolution network (sub-GCN) and decoder. Next, we introduce them in turn.

Encoder The function of this module is to integrate different features by a multi-layer perceptron (MLP):

$$h^1 = \sigma(\mathbf{f} W^1 + b^1)$$

[2]For clarity, we omit the superscripts indicating the mention.

where h^1 is the hidden states of the current mention, $W^1 \in \mathbb{R}^{d_0 \times d_1}$ and $b^1 \in \mathbb{R}^{d_1}$ are trainable parameters and bias. We use ReLu as the non-linear activation $\sigma(\cdot)$.

Sub-Graph Convolution Network Similar to GCN, this module learns to abstract features from the hidden state of the mention itself as well as its neighbors. Suppose $h^t_{m_k}$ is the hidden states of the neighbor m_k, we stack them to expand the current hidden states of m_i as $\tilde{h}^t \in \mathbb{R}^{(2qn+1)\times d_t}$, such that each row corresponds to that in the subgraph adjacent matrix $\tilde{A}$. We define sub-graph convolution as:

$$h^{t+1} = \sigma(\tilde{A}\tilde{h}^t W^t)$$

where $W^t \in \mathbb{R}^{d_t \times d_{t+1}}$ is a trainable parameter.

Decoder After T iterations of sub-graph convolution, the hidden states integrate both features of m_i and its neighbors. A fully connected decoder maps h^{t+1} to the number of candidates as follows:

$$F = h^{T+1} W^{T+1}$$

where $W^{T+1} \in \mathbb{R}^n$.

4.3 Training

The parameters of network are trained to minimize cross-entropy of the predicted and ground truth y^g:

$$\mathcal{L}_m = -\sum_{j=1}^{n} y^g_j log(P(\hat{y} = e_j; \mathbf{f}, \tilde{A}, \omega))$$

Suppose there are $D \in \mathcal{D}$ documents in training corpus, each document has a set of mentions M, leading to totally $M \in \mathcal{M}$ mention sets. The overall objective function is as follows:

$$\mathcal{L} = \sum_{M \in \mathcal{M}} \sum_{m \in M} \mathcal{L}_m$$

5 Experiments

To avoid overfitting with some dataset, we train NCEL using collected Wikipedia hyperlinks instead of specific annotated data. We then evaluate the trained model on five different benchmarks to verify the linking precision as well as the generalization ability. Furthermore, we investigate the effectiveness of key modules in NCEL and give qualitative results for comprehensive analysis[3].

5.1 Baselines and Datasets

We compare NCEL with the following state-of-the-art EL methods including three local models and three types of global models:

1. Local models: He (He et al., 2013a) and Chisholm (Chisholm and Hachey, 2015) beat many global models by using auto-encoders and web links, respectively, and NTEE (Yamada et al., 2017) achieves the best performance based on joint embeddings of words and entities.

2. Iterative model: AIDA (Hoffart et al., 2011) links entities by iteratively finding a dense subgraph.

3. Loopy Belief Propagation: Globerson (Globerson et al., 2016) and PBoH (Ganea et al., 2016) introduce LBP (Murphy et al., 1999) techniques for collective inference, and Ganea (Ganea and Hofmann, 2017) solves the global training problem via truncated fitting LBP.

4. PageRank/Random Walk: Boosting (Kulkarni et al., 2009), AGDISTISG (Usbeck et al., 2014), Babelfy (Moro et al., 2014), WAT (Piccinno and Ferragina, 2014), xLisa (Zhang and Rettinger, 2014) and WNED (Guo and Barbosa, 2017) performs PageRank (Page et al., 1999) or random walk (Tong et al., 2006) on the mention-entity graph and use the convergence score for disambiguation.

[3]Our codes can be found in `https://github.com/TaoMiner/NCEL`

For fairly comparison, we report the original scores of the baselines in the papers. Following these methods, we evaluate NCEL on the following five datasets: (1) **CoNLL-YAGO** (Hoffart et al., 2011): the CoNLL 2003 shared task including testa of 4791 mentions in 216 documents, and testb of 4485 mentions in 213 documents. (2) **TAC2010** (Ji et al., 2010): constructed for the Text Analysis Conference that comprises 676 mentions in 352 documents for testing. (3) **ACE2004** (Ratinov et al., 2011): a subset of ACE2004 co-reference documents including 248 mentions in 35 documents, which is annotated by Amazon Mechanical Turk. (4) **AQUAINT** (Milne and Witten, 2008): 50 news articles including 699 mentions from three different news agencies. (5) **WW** (Guo and Barbosa, 2017): a new benchmark with balanced prior distributions of mentions, leading to a hard case of disambiguation. It has 6374 mentions in 310 documents automatically extracted from Wikipedia.

5.2 Training Details and Running Time Analysis

Training We collect 50,000 Wikipedia articles according to the number of its hyperlinks as our training data. For efficiency, we trim the articles to the first three paragraphs leading to 1,035,665 mentions in total. Using CoNLL-Test A as the development set, we evaluate the trained NCEL on the above benchmarks. We set context window to 20, neighbor mention window to 6, and top $n = 10$ candidates for each mention. We use two layers with 2000 and 1 hidden units in MLP encoder, and 3 layers in sub-GCN. We use early stop and fine tune the embeddings. With a batch size of 16, nearly 3 epochs cost less than 15 minutes on the server with 20 core CPU and the GeForce GTX 1080Ti GPU with 12Gb memory. We use standard Precision, Recall and F1 at mention level (Micro) and at the document level (Macro) as measurements.

Complexity Analysis Compared with local methods, the main disadvantage of collective methods is high complexity and expensive costs. Suppose there are k mentions in documents on average, among these global models, NCEL not surprisingly has the lowest time complexity $\mathcal{O}(T * kn^2)$ since it only considers adjacent mentions, where T is the number of sub-GCN layers indicating the iterations until convergence. AIDA has the highest time complexity k^3n^3 in worst case due to exhaustive iteratively finding and sorting the graph. The LBP and PageRank/random walk based methods achieve similar high time complexity of $\mathcal{O}(T * k^2n^2)$ mainly because of the inference on the entire graph.

5.3 Results on GERBIL

GERBIL (Usbeck et al., 2015) is a benchmark entity annotation framework that aims to provide a unified comparison among different EL methods across datasets including ACE2004, AQUAINT and CoNLL. We compare NCEL with the global models that report the performance on GERBIL.

Datasets	AGDISTIS	AIDA	Babelfy	WAT	xLisa	PBoH	WNED	NCEL
ACE2004	0.66	0.80	0.61	0.76	0.81	0.79	0.81	**0.88**
	0.78	0.89	0.76	0.85	0.88	0.86	**0.90**	0.89
AQUAINT	0.73	0.57	0.70	0.75	0.79	0.84	0.83	**0.87**
	0.59	0.56	0.70	0.76	0.77	0.83	0.83	**0.88**
CoNLL-Test A	0.56	0.74	0.74	0.78	0.52	**0.80**	0.79	0.79
	0.49	0.71	0.68	0.76	0.48	**0.77**	0.76	**0.77**
CoNLL-Test B	0.55	0.77	0.76	**0.80**	0.54	**0.80**	0.79	**0.80**
	0.54	0.78	0.70	**0.80**	0.53	0.79	0.79	**0.80**
Average	0.63	0.72	0.70	0.77	0.67	0.81	0.81	**0.84**
	0.60	0.74	0.71	0.79	0.67	0.81	0.82	**0.84**

Table 1: Micro F1 (above) and Macro F1 (bottom) on GERBIL.

As shown in Table 1, NCEL achieves the best performance in most cases with an average gain of 2% on Micro F1 and 3% Macro F1. The baseline methods also achieve competitive results on some datasets but fail to adapt to the others. For example, AIDA and xLisa perform quite well on ACE2004 but poorly on other datasets, or WAT, PBoH, and WNED have a favorable performance on CoNLL but lower values on ACE2004 and AQUAINT. Our proposed method performs consistently well on all datasets that demonstrates the good generalization ability.

5.4 Results on TAC2010 and WW

In this section, we investigate the effectiveness of NCEL in the "easy" and "hard" datasets, respectively. Particularly, TAC2010, which has two mentions per document on average (Section 5.1) and high prior probabilities of correct candidates (Figure 3), is regarded as the "easy" case for EL, and WW is the "hard" case since it has the most mentions with balanced prior probabilities (Guo and Barbosa, 2017). Besides, we further compare the impact of key modules by removing the following part from NCEL: global features (NCEL-local), attention (NCEL-noatt), embedding features (NCEL-noemb), and the impact of the prior probability (prior).

Table 2: Precision on WW

AIDA	Ganea	WNED	NCEL-local	NCEL
0.63	0.78	0.84	0.81	**0.86**

	Prec	Micro F1	Macro F1
Chisholm	0.81	-	-
He	0.81	-	-
NTEE	0.88	-	-
NCEL-local	0.89	0.89	0.88
AIDA	-	0.55	0.51
Babelfy	-	0.63	0.62
WAT	-	0.75	0.73
Globerson	-	0.84	-
Boosting	-	0.86	0.85
NCEL	**0.91**	**0.91**	**0.92**

Table 3: Results on TAC2010

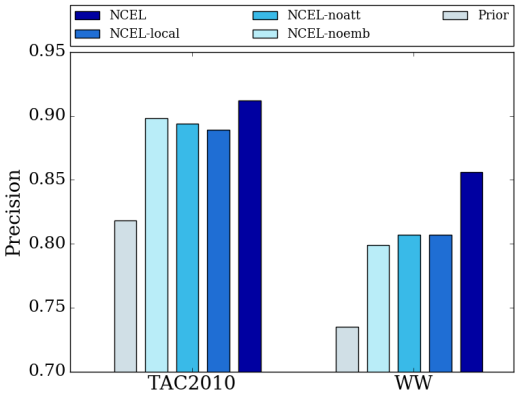


Figure 3: Impacts of NCEL modules

The results are shown in Table 2 and Table 3. We can see the average linking precision (Micro) of WW is lower than that of TAC2010, and NCEL outperforms all baseline methods in both easy and hard cases. In the "easy" case, local models have similar performance with global models since only little global information is available (2 mentions per document). Besides, NN-based models, NTEE and NCEL-local, perform significantly better than others including most global models, demonstrating that the effectiveness of neural models deals with the first limitation in the introduction.

Impact of NCEL Modules

As shown in Figure 3, the prior probability performs quite well in TAC2010 but poorly in WW. Compared with NCEL-local, the global module in NCEL brings more improvements in the "hard" case than that for "easy" dataset, because local features are discriminative enough in most cases of TAC2010, and global information becomes quite helpful when local features cannot handle. That is, our propose collective model is robust and shows a good generalization ability to difficult EL. The improvements by each main module are relatively small in TAC2010, while the modules of attention and embedding features show non-negligible impacts in WW (even worse than local model), mainly because WW contains much noise, and these two modules are effective in improving the robustness to noise and the ability of generalization by selecting informative words and providing more accurate semantics, respectively.

5.5 Qualitative Analysis

Hussain, considered surplus to Englands one-day requirements, struck 158, his first championship century of the season, as Essex reached 372 and took a first innings lead of 82.			
NCEL		NCEL-local	
England	0.23	England	0.42
England cricket team	0.72	England cricket team	0.20
Essex County Cricket Club	0.99	Essex County Cricket Club	0.97

Table 4: Qualitative Analysis of the Example *England*.

The results of example in Figure 1 are shown in Table 4, which is from CoNLL testa dataset. For mention *Essex*, although both NCEL and NCEL-local correctly identify entity *Essex County Cricket*

Club, NCEL outputs higher probability due to the enhancement of neighbor mentions. Moreover, for mention *England*, NCEL-local cannot find enough disambiguation clues from its context words, such as *surplus* and *requirements*, and thus assigns a higher probability of 0.42 to the country *England* according to the prior probability. Collectively, NCEL correctly identifies England cricket team with a probability of 0.72 as compared with 0.20 in NCEL-local with the help of its neighbor mention *Essex*.

6 Conclusion

In this paper, we propose a neural model for collective entity linking that is end-to-end trainable. It applies GCN on subgraphs instead of the entire entity graph to efficiently learn features from both local and global information. We design an attention mechanism that endows NCEL robust to noisy data. Trained on collected Wikipedia hyperlinks, NCEL outperforms the state-of-the-art collective methods across five different datasets. Besides, further analysis of the impacts of main modules as well as qualitative results demonstrates its effectiveness.

In the future, we will extend our method into cross-lingual settings to help link entities in low-resourced languages by exploiting rich knowledge from high-resourced languages, and deal with NIL entities to facilitate specific applications.

7 Acknowledgments

The work is supported by National Key Research and Development Program of China (2017YFB1002101), NSFC key project (U1736204, 61661146007), and THUNUS NExT Co-Lab.

References

Roi Blanco, Giuseppe Ottaviano, and Edgar Meij. 2015. Fast and space-efficient entity linking for queries. In *WSDM*.

Yixin Cao, Lifu Huang, Heng Ji, Xu Chen, and Juan-Zi Li. 2017. Bridge text and knowledge by learning multi-prototype entity mention embedding. In *ACL*.

Taylor Cassidy, Heng Ji, Lev-Arie Ratinov, Arkaitz Zubiaga, and Hongzhao Huang. 2012. Analysis and enhancement of wikification for microblogs with context expansion. In *COLING*.

Zheng Chen and Heng Ji. 2011. Collaborative ranking: A case study on entity linking. In *EMNLP*.

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. In *EMNLP*.

Andrew Chisholm and Ben Hachey. 2015. Entity disambiguation with web links. *TACL*.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *TACL*.

Yotam Eshel, Noam Cohen, Kira Radinsky, Shaul Markovitch, Ikuya Yamada, and Omer Levy. 2017. Named entity disambiguation for noisy text. In *CoNLL*.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *EMNLP*.

Octavian-Eugen Ganea, Marina Ganea, Aurélien Lucchi, Carsten Eickhoff, and Thomas Hofmann. 2016. Probabilistic bag-of-hyperlinks model for entity linking. In *WWW*.

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In *ACL*.

Zhaochen Guo and Denilson Barbosa. 2017. Robust named entity disambiguation with random walks.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *EMNLP*.

Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: a graph-based method. In *SIGIR*.

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013a. Learning entity representation for entity disambiguation. In *ACL*.

Zhengyan He, Shujie Liu, Yang Song, Mu Li, Ming Zhou, and Houfeng Wang. 2013b. Efficient collective entity linking with stacking. In *EMNLP*.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *EMNLP*.

Hongzhao Huang, Yunbo Cao, Xiaojiang Huang, Heng Ji, and Chin-Yew Lin. 2014. Collective tweet wikification based on semi-supervised graph regularization. In *ACL*.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the tac 2010 knowledge base population track.

Heng Ji, Joel Nothman, Ben Hachey, and Radu Florian. 2015. Overview of tac-kbp2015 tri-lingual entity discovery and linking.

Heng Ji, Joel Nothman, H Trang Dang, and Sydney Informatics Hub. 2016. Overview of tac-kbp2016 tri-lingual edl and its impact on end-to-end cold-start kbp. *Proceedings of TAC*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *ICLR*.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective annotation of wikipedia entities in web text. In *KDD*.

Nevena Lazic, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2015. Plato: A selective context model for entity resolution. *TACL*.

David N. Milne and Ian H. Witten. 2008. Learning to link with wikipedia. In *CIKM*.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *TACL*.

Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *UAI*.

Thien Huu Nguyen, Nicolas Fauceglia, Mariano Rodriguez-Muro, Oktie Hassanzadeh, Alfio Massimiliano Gliozzo, and Mohammad Sadoghi. 2016. Joint learning of local and global features for entity linking via neural networks. In *COLING*.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web.

Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized page rank for named entity disambiguation. In *HLT-NAACL*.

Minh C. Phan, Aixin Sun, Yi Tay, Jialong Han, and Chenliang Li. 2018. Pair-linking for collective entity disambiguation: Two could be better than all. *CoRR*.

Francesco Piccinno and Paolo Ferragina. 2014. From tagme to wat: a new entity annotator. In *ERD@SIGIR*.

Lev-Arie Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *ACL*.

Valentin I. Spitkovsky and Angel X. Chang. 2012. A cross-lingual dictionary for english wikipedia concepts. In *LREC*.

Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast random walk with restart and its applications. *ICDM*.

Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Michael Röder, Daniel Gerber, Sandro Coelho, Sören Auer, and Andreas Both. 2014. Agdistis - graph-based disambiguation of named entities using linked data. In *International Semantic Web Conference*.

Ricardo Usbeck, Michael Röder, Axel-Cyrille Ngonga Ngomo, Ciro Baron, Andreas Both, Martin Brümmer, Diego Ceccarelli, Marco Cornolti, Didier Cherix, Bernd Eickmann, et al. 2015. Gerbil: general entity annotator benchmarking framework. In *WWW*.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *CoNLL*.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2017. Learning distributed representations of texts and entities from knowledge base. *TACL*.

Lei Zhang and Achim Rettinger. 2014. X-lisa: Cross-lingual semantic annotation. *PVLDB*.

Yuanzhe Zhang, Shizhu He, Kang Liu, and Jun Zhao. 2016. A joint model for question answering over multiple knowledge bases. In *AAAI*.

Exploiting Structure in Representation of Named Entities using Active Learning

Nikita Bhutani
University of Michigan, Ann Arbor
nbhutani@umich.edu

Kun Qian
IBM Research - Almaden
qian.kun@ibm.com

Yunyao Li
IBM Research - Almaden
yunyaoli@us.ibm.com

H. V. Jagadish
University of Michigan, Ann Arbor
jag@umich.edu

Mauricio A. Hernandez
IBM Research - Almaden
mahernan@us.ibm.com

Mitesh Vasa
IBM Research - Almaden
mitesh.vasa@us.ibm.com

Abstract

Fundamental to several knowledge-centric applications is the need to identify named entities from their textual mentions. However, entities lack a unique representation and their mentions can differ greatly. These variations arise in complex ways that cannot be captured using textual similarity metrics. However, entities have underlying structures, typically shared by entities of the same entity type, that can help reason over their name variations. Discovering, learning and manipulating these structures typically requires high manual effort in the form of large amounts of labeled training data and handwritten transformation programs. In this work, we propose an active-learning based framework that drastically reduces the labeled data required to learn the structures of entities. We show that programs for mapping entity mentions to their structures can be automatically generated using human-comprehensible labels. Our experiments show that our framework consistently outperforms both handwritten programs and supervised learning models. We also demonstrate the utility of our framework in relation extraction and entity resolution tasks.

1 Introduction

Named entities are atomic objects of reference and reasoning in many cognitive applications and knowledge-centric services like deep question answering, text summarization and analytics. A real-world entity may have a great variety of representations (Galárraga et al., 2014; Nakashole et al., 2011). For example, University of California, Santa Cruz could have different string representations or *name variations*: UCSC, UC Santa Cruz, UC–Santa Cruz. Determining if two representations refer to the same entity is an important primitive in entity resolution and entity linking algorithms that drive these knowledge-centric applications (Shen et al., 2015; Arasu and Kaushik, 2009).

Unifying entity representations has been widely studied in record linkage (Christen, 2012), deduplication (Elmagarmid et al., 2007) and reference matching (McCallum et al., 2000). Typically, duplicates are identified using the attributes and/or the contextual information of entities (Zhang et al., 2010; Han et al., 2011; Shen et al., 2015). The string representation or *mention* of an entity forms key evidence: similar mentions likely refer to the same entity. Although deemed as crucial (Dredze et al., 2010), variations in mentions are typically handled using textual similarity like edit distance and cosine similarity (Zheng et al., 2010; Lehmann et al., 2010; Liu et al., 2013), which can be misleading.

Example. Consider the mentions: (a) General Electric Corporation, (b) General Electric China Corporation, (c) GE Corp. Mentions (a) and (b) are textually similar, differing in just one token. However, they refer to different entities, owing to the location detail 'China'. Conversely, textually dissimilar mentions (a) and (c) refer to the same entity.

An entity mention is not merely a sequence of characters (Arasu and Kaushik, 2009). It instead has an internal structure, specific to the type of entity. For example in Table 1, the company mentions have a $\langle name \rangle$, optionally followed by $\langle loc \rangle$ and $\langle suffix \rangle$. Such structural interpretation can help design similarity

Proceedings of the 27th International Conference on Computational Linguistics, pages 687–699
Santa Fe, New Mexico, USA, August 20-26, 2018.

Unlabeled Mention	Structured Representation	Labeled Mention
IBM United Kingdom Ltd.	⟨name⟩⟨loc⟩⟨suffix⟩	IBM→⟨name⟩,United Kingdom→⟨loc⟩, Ltd.→⟨suffix⟩
Alibaba USA	⟨name⟩⟨loc⟩	Alibaba→⟨name⟩, USA→⟨loc⟩
Barclays	⟨name⟩	Barclays→⟨name⟩
Hewlett-Packard Co.	⟨name⟩⟨suffix⟩	Hewlett-Packard→⟨name⟩, Co.→⟨suffix⟩
University of California, Irvine	⟨name⟩,⟨loc⟩	University of California→⟨name⟩, Irvine→⟨loc⟩
UM-Ann Arbor	⟨name⟩-⟨loc⟩	UM→⟨name⟩, Ann Arbor→⟨loc⟩
Stanford University	⟨name⟩	Stanford University→⟨name⟩

Table 1: Example Structured Representations of Company and University Mentions

functions to capture the nuances in name variations of an entity that string similarity functions cannot. For instance, GE can be explained as a mention of General Electric Corporation using transformations like abbreviate $\langle name \rangle$ and drop $\langle suffix \rangle$. These transformations coupled with string similarity functions can augment existing entity linking algorithms (Qian et al., 2017).

The name variations and, consequently, the transformations are highly domain-dependent (Arasu et al., 2008). Designing similarity functions for an entity type that can reason over the structure of entities requires: (a) a comprehensive list of the structured representations (e.g. column 2 in Table 1), and (b) programs that can map mentions to these structures (e.g. column 3 in Table 1). Traditionally, a domain expert would scan a list of mentions of a target entity type to identify the different structured representations and handwrite programs. This requires specialized skills, and is error-prone and expensive, taking up to several person months to tune the programs for a single application (Campos et al., 2015).

To alleviate the high manual effort, some of the prior works (Arasu and Kaushik, 2009) use declarative, programmable frameworks that allow an expert to directly manipulate the mentions. The expert can provide a program as a set of grammar rules to generate the structured representations. While this equips the expert to manipulate the structure of a mention, it is only a partial solution. The rules are not generic, requiring the expert to specify how each mention is parsed to its structure. This is wasteful because a structure, shared by several mentions, can be captured with a generic rule. Another limitation of previous approaches is that they do not handle structural ambiguities. For instance, there can be multiple structures of Apple Inc. such as $\langle name \rangle \langle loc \rangle$ and $\langle name \rangle \langle suffix \rangle$, only one of which is correct.

Since several mentions of an entity type tend to have similar structures, it is possible to learn the various structures from a subset of mentions. Given a good query strategy, active learning offers a promising approach to efficiently select a small set of such mentions. Moreover, the expert need not provide programs that parse the selected mentions and generalize to unseen mentions. These programs can be induced from the labels for the structures of the mentions. Embodying these ideas, we propose LUSTRE, an active-learning based framework that learns structured representations for an entity type from human-comprehensible labels for a small set of mentions. It automatically synthesizes *generalizable* programs from the labels to map new mentions of the entity type to the learned structured representations. In addition, it allows the expert to incorporate domain knowledge and additional feedback to handle structural ambiguities. Our framework significantly reduces the manual effort in labeling mentions and writing programs for the structured representations. We also demonstrate how these structured representations help define similarity functions that benefit entity resolution, and string transformation functions that benefit relation extraction. The intellectual contributions of this work are as follows:

- *Structured Representations.* We present a framework to reason about name variations of entities based on their structured representations.
- *Active-learning.* We present an active-learning approach and a unified query strategy to learn structured representations for a target entity type with minimal human input.
- *Program Synthesis.* We propose to automatically synthesize *generalizable* programs from human-understandable labels to map mentions to their structured representations.
- *Experimental Evaluation.* Our experiments show that our framework achieves an average of 92% precision and 86% recall in predicting structured representations for several entity types, outperforming competing approaches that require high manual effort. We demonstrate the usefulness of the structured representations in two important tasks: entity resolution and relation extraction.

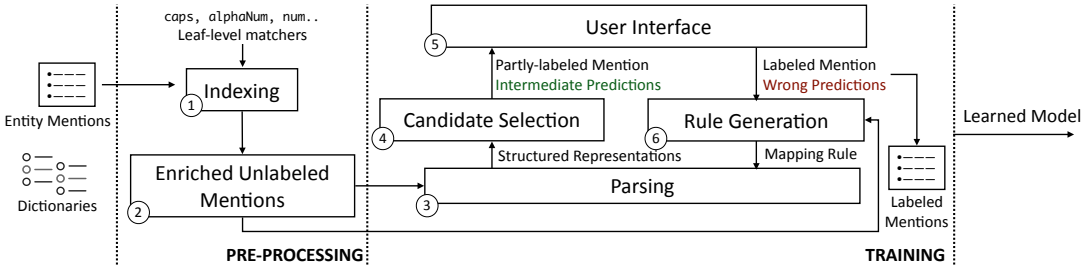


Figure 1: LUSTRE Architecture

2 Problem Formalization and Notations

Given a set of unlabeled mentions (i.e., raw strings) of a target entity type ϵ, our goal is to actively learn a high-quality model M_ϵ (with a small number of user labels) that can map a new mention of type ϵ to its structured representation. Key technical challenges in learning such a model include: (1) designing an effective query strategy to find representative mentions that are structurally similar to several other mentions and to find mentions with diverse structures; (2) automatically inferring mapping programs from the user labels to reduce the human effort, as in (Campos et al., 2015) and (Arasu and Kaushik, 2009); (3) handling ambiguities when an unlabeled mention has multiple candidate structured representations.

Mentions of a target entity type have internal *structured representations* consisting of atomic *semantic units*. Given the labels for the semantic units for a small set of mentions, we want to learn a model M of *mapping rules*. Each mapping rule converts a raw string mention into a structured representation.

Definition 1 (Structured Representation)**.** *A structured representation $\mathcal{S}$ of an entity mention is a sequence of atomic semantic units that compose its structure.*

Textually dissimilar mentions can have the same structure (e.g. Apple Inc. and Hewlett-Packard Co.).

Definition 2 (Semantic Unit)**.** *A semantic unit is a tuple $u = \langle l : p \rangle$ where l is a label for the unit and p is a pattern matching function (e.g., a regular expression), referred to as a **matcher**.*

A matcher describes how a substring in a mention matches a semantic unit. For instance, a matcher ([A-Z][A-Za-z]*[-']?)+ captures $\langle name \rangle$ in mentions Apple Inc. and GE Corp. Name variations of an entity can be explained as transformations on its semantic units (e.g. dropping suffix in Apple Inc. generates Apple).

Definition 3 (Mapping Rule)**.** *The mapping rule $r_\mathcal{S}$ for a structured representation $\mathcal{S}$ consists of matchers which decide how an unlabeled mention is mapped to $\mathcal{S}$, resulting in a **labeled entity mention**.*

Specifically, a mapping rule $r_\mathcal{S}$ is a sequence of matchers in the semantic units in $\mathcal{S}$, denoted by $r_\mathcal{S} = \{p_i \mid \langle l_i, p_i \rangle \in \mathcal{S}\}$. This mapping rule constitutes a program that maps unseen mentions to $\mathcal{S}$. Together the various mapping rules constitute a model M_ϵ for the entity type ϵ.

3 The LUSTRE System

We propose LUSTRE that addresses the aforementioned challenges in learning a model of mapping rules for an entity type by:

- adopting a unified query strategy that combines uncertainty sampling (i.e., selecting a mention whose current structured representation is unknown or uncertain) and density-weighted sampling (i.e., selecting a mention whose structured representation is representative of many unlabeled mentions).
- seeking human-comprehensible labels for the semantic units in the structure of an unlabeled mention, and deducing a mapping rule by combining the matchers for the semantic units.
- handling structural ambiguities of an unlabeled mention by ranking the candidate mapping rules based on their *reliability* and additional user feedback.

Figure 1 depicts the workflow of LUSTRE and Algorithm 1 shows our learning algorithm. It takes as input unlabeled mentions $\mathcal{U}_\epsilon$ and optionally dictionaries $\mathcal{D}_\epsilon$ of a target entity type ϵ. These dictionaries and a set of pre-defined matchers form the building blocks for the mapping rules. Before training, LUSTRE evaluates all the matchers against $\mathcal{U}_\epsilon$ to inform the query strategy and rule generation (Sec 3.1).

Algorithm 1 LUSTRE learning algorithm

input: $\mathcal{U}_\epsilon$ = a pool of unlabeled mentions $\{m\}$
output: $\mathcal{M}_\epsilon$ = a model of mapping rules $\{r_\mathcal{S}\}$

1: **function** TRAIN($\mathcal{U}_\epsilon$)
2: $\mathcal{L}_\epsilon$ = a set of labeled mentions $\{\langle m, l\rangle\}$
3: $\mathcal{F}$ = feedback $\{\langle m, l, f\rangle \mid m \in \mathcal{U}, f \in \{0, 1\}\}$
4: $\mathcal{M} = \varnothing$
5: **for** t = 1,2... **do**
6: $\mathcal{M}$ = UPDATE($\mathcal{M}, \mathcal{L}, \mathcal{F}$)
7: select $m^* \in \mathcal{U}$, mention with highest utility
8: select $\mathcal{F}$ with least confident predictions
9: query label l^* for mention m^*
10: query binary feedback on $\mathcal{F}$
11: $\mathcal{L} = \mathcal{L} \cup \langle m^*, l^*\rangle, \mathcal{U} = \mathcal{U} \backslash \{m^*\}$
12: **return** $\mathcal{M}$
13: **function** UPDATE($\mathcal{M}, \mathcal{L}, \mathcal{F}$)
14: **for** $\langle m, l\rangle \in \mathcal{L}$ **do**
15: $r_\mathcal{S} = generate\_rule(l)$
16: **if** $r_\mathcal{S} \notin \mathcal{M}$ **then**
17: $\mathcal{P}_{r_\mathcal{S}} = reliability(r_\mathcal{S})$
18: $\mathcal{M} = \mathcal{M} \cup \langle r_\mathcal{S}, \mathcal{P}_{r_\mathcal{S}}\rangle$
19: **for** $\langle r_\mathcal{S}, \mathcal{P}_{r_\mathcal{S}}\rangle \in \mathcal{M}$ **do**
20: $\mathcal{P}_{r_\mathcal{S}} = estimate(\mathcal{P}_{r_\mathcal{S}}, \mathcal{F})$
21: **return** $\mathcal{M}$

During training, in each iteration, LUSTRE selects a candidate mention for the user to label (Sec 3.2). Given the user-provided labels for semantic units of the selected mention, it derives a generic mapping rule for the structure of the mention (Sec 3.3). It then updates the model M with the new rule, and predicts the structures of unlabeled mentions (Sec 3.4). It presents a sample of these predictions to the user and integrates the user feedback in M (Sec 3.5). This iterative process continues until all the unlabeled mentions can be mapped to some structure or the user is satisfied with M.

3.1 Indexing

User-provided dictionaries and pre-defined regular expressions constitute the vocabulary of matchers for the mapping rules. The dictionaries help capture key domain-specific terminology but need not correspond to semantic units.

Example. *A suffix dictionary may contain 'Inc.', 'Corp', 'LLC', 'Pvt. Ltd.'* □

Additionally, we use type-independent regex matchers shown in Table 2. In a pre-processing step (Line 1 of Algorithm 1), we evaluate the unlabeled mentions against the matchers to save computational overhead. We refer to these *enriched unlabeled mentions* during training.

Example. *Given dictionaries for country and suffix and a mention IBM UK Ltd., the matchers yield matches UK→*$\langle country\rangle$*, Ltd.→*$\langle suffix\rangle$ *IBM→* $[\langle caps\rangle, \langle alphaNum\rangle]$*, UK→* $[\langle caps\rangle]$. □

Multiple matchers can match the same token (e.g. $\langle country\rangle$ and $\langle caps\rangle$ match 'UK'). We assume that more specific matchers offer higher precision than generic matchers. We further formalize this intuition in Section 3.3. We rank matchers in the following order based on their selectivity: $\mathcal{D}_\epsilon > caps > alphaNum > num > special > wild$[1].

Name	Regex
caps	[A-Z][A-Za-z]*[.-']?
alphaNum	[A-Za-z0-9]+[.-']?
num	[0-9]+
special	[^A-Za-z0-9]+
wild	.+

Table 2: Predefined Matchers

3.2 Candidate Selection

The query strategy to determine what constitutes an informative mention is the central challenge in our active-learning setting (Line 7 of Algorithm 1). We consider a mention *informative* if its represents the structure of several other unlabeled mentions and its current structure is unknown or uncertain. We adopt a unified approach, combining density-weighted sampling (Settles and Craven, 2008) and uncertainty sampling (Culotta and McCallum, 2005) as it is robust to outliers and input distribution.

For each unlabeled mention m_i, we compute a correlation score c_i (based on its structural similarity to other mentions) and an uncertainty score f_i (based on its predicted structure). We then compute a utility score u_i for m_i, combining its correlation and uncertainty scores, and select a mention m^* with the highest utility score for labeling.

To compute correlation score c_i, we need a reliable metric to measure the similarity of the structures of two mentions. Since surface-string similarity metrics won't suffice, we estimate structural similarity of two mentions as a function of the matchers that constitute their structures. Specifically, we compute structural similarity $c(i, j)$ of a pair of mentions (i, j) as the edit distance of their structures, $\mathcal{S}_i$ and $\mathcal{S}_j$.

[1]The order of preference can be inferred in a pre-processing step in which the system counts the selectivity of each matcher against the input mentions

Example. *'IBM Ltd.' and 'Apple Inc.' have the same structure* $\langle caps \rangle \langle suffix \rangle$*, and therefore, have an edit distance of 0. Each has edit distance 1 to 'Microsoft Asia Inc.' with structure* $\langle caps \rangle \langle loc \rangle \langle suffix \rangle$*.* □

Given the pair-wise structural similarity metric, the correlation score c_i of a mention m_i is its average structural similarity to other unlabeled mentions.

$$c_i = \frac{1}{|U|} \sum_{j \in U} c(i,j) \qquad f_i = \begin{cases} 1 & \text{if no rule maps } m_i \\ 1 - \mathcal{P}_{\hat{r}_\mathcal{S}} & \text{otherwise} \end{cases} \qquad u_i = c_i * f_i$$

$$c(i,j) = 1 - \frac{\text{edit distance}(\mathcal{S}_i, \mathcal{S}_j)}{\text{max edit distance}} \qquad \mathcal{P}_{\hat{r}_\mathcal{S}} = \operatorname*{argmax}_r \mathcal{P}_{r_\mathcal{S}} \qquad m^* = \operatorname*{argmax}_{m_i} u_i$$

To estimate the uncertainty score f_i of a mention, we use $\mathcal{P}_{r_\mathcal{S}}$, the reliability of the mapping rules that can parse the mention. If no mapping rule can parse a mention m_i, its structure is unknown i.e. f_i is simply 1. Otherwise, it is the uncertainty of the most reliable rule $\hat{r}_\mathcal{S}$ known to parse the mention. We discuss how the reliability $\mathcal{P}_{r_\mathcal{S}}$ of mapping rules are estimated in Sec. 3.4.

3.3 Rule Generation

Once the user labels a selected mention, LUSTRE next has to synthesize a generic program i.e. a mapping rule for the structure of the mention (Line 15 of Algorithm 1). Deriving a mapping rule is non-trivial as semantic units in the structure can potentially span multiple tokens and matchers. As a result, there are many ways to combine matchers and derive the rule.

Example. *A user can label 'General Motors' as semantic unit* $\langle name \rangle$ *in 'General Motors Co.' Tokens 'General' and 'Motors' map to matchers* $caps$ *and* $alphaNum$*. The most reliable interpretation for* $\langle name \rangle$ *is* $\langle name ::caps\{1,2\} \rangle$*, a combination of two adjacent matchers* $\langle caps \rangle$*.* □

LUSTRE derives a reliable rule as the sequence of most selective matchers, where *selectivity* is the expected number of matches of a matcher over the set of unlabeled mentions $\mathcal{U}$ (Li et al., 2008).

$$sel(p_i) = \mathrm{E}[match(p_i, m \in \mathcal{U})]$$

with $match(p_i, m)$ being number of matches of p_i over mention m.

3.4 Parsing

When a new mapping rule is learned, we want to estimate its reliability in predicting structures of mentions, and update the model M (Line 17-18 of Algorithm 1). These reliability scores are used for estimating utility scores at candidate selection and for resolving ambiguities when multiple rules can parse a mention (with preference given to the most reliable rule). Following the intuition that generic rules are less reliable, we estimate reliability of a rule based on its expected numbers of matches in the unlabeled mentions. Specifically, reliability $\mathcal{P}_{r_\mathcal{S}}$ is a function of the selectivity of the matchers in $r_\mathcal{S}$.

$$p^* = \operatorname*{argmin}_i \{sel(p_i) \mid \langle l_i, p_i \rangle \in \mathcal{S}\}$$

$$\mathcal{P}_{r_\mathcal{S}} = 1 - sel(p^*)$$

3.5 User Interface

LUSTRE uses an easy-to-use interface (Qian et al., 2018) to seek labels for a selected mention and additional feedback on intermediate predictions (Line 9 of Algorithm 1). To reduce the labeling effort, tokens in the mention that match an entry in the dictionary are pre-labeled with the name of the dictionary (e.g. 'IBM Corp' is presented with 'Corp' labeled as suffix). The user can keep these and/or provide new labels (e.g. user can label 'IBM' as name).

In addition, the interface presents the predictions of structures for a sample of unlabeled mentions. It selects 10 least confident predictions based on uncertainty scores (Section 3.2). The user can simply mark a prediction incorrect, without providing labels for its correct structure. LUSTRE effectively integrates this feedback to avoid over-estimating the reliability of learned rules, thereby improving the quality of model M. Specifically, it updates the reliability of a mapping rule $\mathcal{P}^i_{r_\mathcal{S}}$ as a function of number of incorrect predictions $\overline{m_r}$ for the rule in the set of 10 predictions presented to the user (Line 20 of Algorithm 1).

$$\mathcal{P}^j_{r_\mathcal{S}} = \mathcal{P}^i_{r_\mathcal{S}} * (1 - \alpha * \frac{|\overline{m_r}|}{10})$$

with α being the decay constant. We found such estimation, though simple, was effective in estimating the reliability of mapping rules. Training more powerful measures such as a discriminative model would require larger user feedback than is available in this setting.

4 Experiments

We assess the effectiveness of our learning algorithm, and the quality and usefulness of learned structures.[2]

4.1 Experimental Setup

Datasets. We conduct experiments on four entity types of varied complexity and ambiguity.

- Person: Focusing on subtype *Individual*, we randomly select 200 unique mentions for training and another 200 mentions for testing from the ACE 2005 dataset (Walker et al., 2006). For out-of-domain tests, we randomly select 200 person mentions from Freebase (Bollacker et al., 2008).
- Company: Focusing on subtype *Commercial*, we randomly select 200 mentions for training and use the remaining 100 mentions for testing from the ACE dataset. For out-of-domain test, we randomly select 200 company mentions from Freebase.
- Tournament: We use a 50/50 split for the 100 unique mentions of tournaments in Freebase.
- Academic Title: We use a 50/50 split for the 350 unique mentions of academic titles in Freebase.

For each mention, we ask two experts to manually annotate every token with a semantic label to produce the ground truth. The average inter-annotator agreement was 0.89 for Cohen's κ.

Baselines. We compare LUSTRE with the following methods.

- STG: A commercial system (Campos et al., 2015) which requires an expert (with domain knowledge and programming skills) to analyze the structures of an entity type and handwrite mapping programs.
- Linear-chain CRF[3]: A linear-chain CRF model that predicts a sequence of labels for the tokens in a mention. We use matches from the *Indexing* stage as features. We use two different training settings: CRF trained on the entire training set (to compare with best model), and CRF^{L} trained on the subset of training set selected via the query strategy in LUSTRE (to compare with same user effort).
- LUSTRE^{T}: LUSTRE with a native tf-idf based query strategy.

Evaluation Metric. We consider a prediction correct if the model and expert agree on the semantic labels for each token in the mention. We measure *precision* as the fraction of predictions that are correct and *recall* as the fraction of correct structures that are predicted. We define a new metric, which we call the α value, to estimate the role of manual effort on performance of various methods (definition is given below). Intuitively, higher the α value, higher the effectiveness of a method in learning from a user label.

$$\alpha(X,t) = \frac{\text{F-score of method } X \text{ on entity type } t}{\text{number of user labels requested by } X}, \text{where X} \in \{\text{LUSTRE, CRF, CRF}^{\text{L}}\}.$$

4.2 Quality Analysis

The performance results of the different methods are summarized in Table 3. For *Person* mentions, all methods achieve reasonably good results (F-scores are all above 0.85). This is not surprising as person mentions typically have simple structures with few semantic units. STG achieves lowest performance, indicating that manually-crafted programs are not as robust as learned models. CRF models, especially CRF^{L} trained on mentions selected using LUSTRE, show evident improvement over STG. LUSTRE achieves comparable performance. However, it outperforms all other methods on out-of-domain test data, suggesting it can capture the structures in mentions regardless of their data source.

Company mentions are more complex than *Person* mentions. They can have several semantic units such as core name, location, suffix and subsidiary, which can appear in different orders, be separated by special symbols etc. Consequently, capturing structures in company mentions is more difficult, as is reflected in the performance across methods. Learning a reliable model/program would require higher

[2]We will release the code and data from this work. Code is proprietary but can be licensed.

[3]http://mallet.cs.umass.edu/sequences.php

manual effort in STG and more training data in CRF. In contrast, LUSTRE, with small human input, achieves high precision and recall for both in-domain and out-of-domain data.

Tournament and *Academic Title* mentions have even more complex structures that exhibit more variations. Even for these complex types, LUSTRE outperforms CRF. We do not report results of STG for these two entity types because these were not considered when STG was developed. We do not provide an out-of-domain evaluation of the two entity types because only the Freebase data included mentions of these entity types.

In summary, LUSTRE outperforms STG and CRF-based methods in terms of overall F-score with the exception of *Person* where CRF^{L} has a small improvement over LUSTRE. Furthermore, its unified query strategy is more effective than a tf-idf based strategy, as is reflected in the performance of LUSTRE^{T}.

We found two main sources of errors made by LUSTRE. First, akin to all learning methods (such as CRF), it has low recall when the training data is not representative. Second, ranking mapping rules sometimes cannot effectively resolve structural ambiguities. For example, 'The Stanford University Professorship in Nephrology' can be interpreted as $\langle honorary\ \ prefix\rangle\langle title\rangle\langle specialty\rangle$ or $\langle institute\rangle\langle title\rangle\langle specialty\rangle$. The former is incorrect, but being more commonly observed in the training data, is ranked higher.

		In-Domain			Out-of-Domain		
Type	Algorithm	P	R	F_1	P	R	F_1
Person	STG	0.92	0.92	0.92	0.85	0.85	0.85
	CRF	0.97	0.97	0.97	0.90	0.90	0.90
	CRF^{L}	**0.99**	**0.99**	**0.99**	0.91	0.91	0.91
	LUSTRE^{T}	0.98	0.95	0.96	**0.92**	0.90	0.91
	LUSTRE	**0.99**	0.97	0.98	**0.92**	**0.95**	**0.93**
Company	STG	0.83	0.83	0.83	0.79	0.79	0.79
	CRF	0.87	**0.87**	0.87	0.85	**0.85**	0.85
	CRF^{L}	0.81	0.81	0.81	0.73	0.73	0.73
	LUSTRE^{T}	0.84	0.77	0.80	0.78	0.60	0.68
	LUSTRE	**0.95**	0.86	**0.90**	**0.91**	**0.85**	**0.88**
Tournament	CRF	0.70	0.70	0.70	-	-	-
	CRF^{L}	0.68	0.68	0.68	-	-	-
	LUSTRE^{T}	0.96	0.68	0.79	-	-	-
	LUSTRE	**0.96**	**0.90**	**0.93**	-	-	-
Academic Title	CRF	0.69	**0.69**	0.69	-	-	-
	CRF^{L}	0.67	0.67	0.67	-	-	-
	LUSTRE^{T}	0.36	0.23	0.28	-	-	-
	LUSTRE	**0.79**	0.65	**0.72**	-	-	-

Table 3: Performances of LUSTRE, STG and CRF

4.3 Effectiveness

To assess how the quality of structured representations evolves in the learning process, we examine the precision and recall of LUSTRE after each iteration (as shown in Figure 2). We found that the precision remains nearly constant. There are a limited number of structures for the same types of entities. Every time LUSTRE learns a precise rule that improves coverage. There are slight drops in the precision due to the long tail of non-representative cases. The recall generally increases depending on whether the system prefers to gather additional evidence or to discover new structured representations.

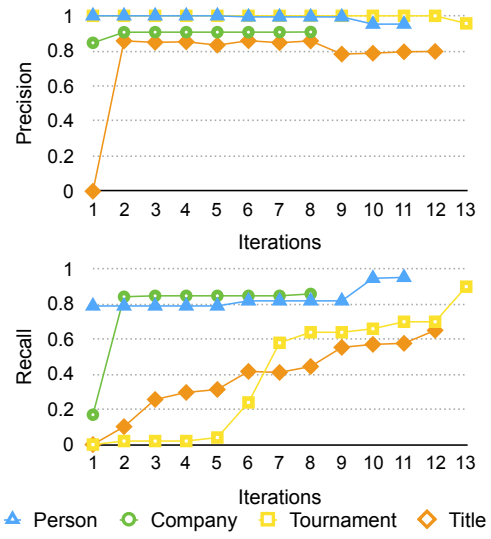


Figure 2: Performance of LUSTRE over iterations

We also examine the number of rules learned, number of incorrect intermediate predictions, and percentage of input training data covered after each learning iteration in LUSTRE (as shown in Figure 3). We found that it took 8-13 iterations to learn almost all different structures of an entity type. The fraction of training data that could be parsed using the mapping rules also generally increased, indicating that our query strategy selected structurally diverse mentions for labeling. Only a few (<5) predictions were

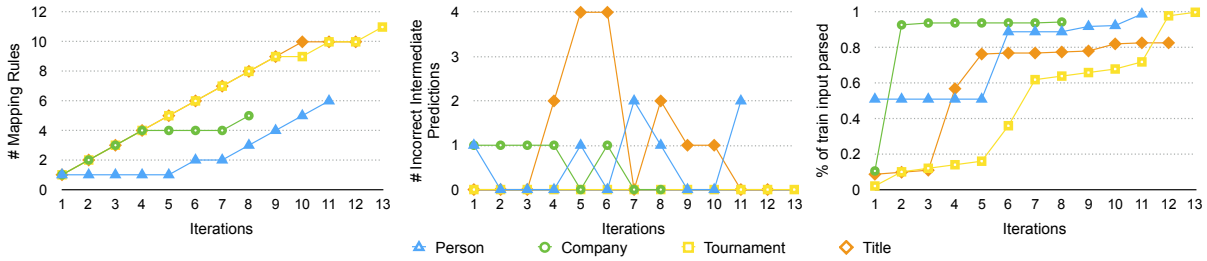


Figure 3: Number of mapping rules, number of incorrect predictions, fraction of training data parsed after each iteration

marked incorrect in each iteration. This suggests that LUSTRE, with a small set of labeled mentions and a little user feedback, learned reliable rules and ranked them correctly.

Next, we compare the nature and amount of human effort across different methods. STG requires high effort and high skill. Even for a simple entity type *Person*, it takes one skilled developer a few months to manually inspect the different structural patterns and write programs. Learned models such as LUSTRE and CRF reduce the skill level and require only human-readable labels for a mention. However, in contrast to CRF, LUSTRE does not require the user to collect and label a set of representative mentions. It guides the user to interactively and iteratively label a few mentions, with each iteration taking less 7 seconds. The optional feedback also is small (<5 predictions) and low-effort (boolean labels).

Quantitatively, we report the α values of the three learning methods. CRF has the lowest α value because it uses all labeled examples. Interestingly, benefiting from the informative mentions selected by LUSTRE, CRF^L have much higher α values. LUSTRE, however, has the highest α values.

	LUSTRE	CRF	CRF^L
`Person (in-domain)`	0.089	0.005	**0.090**
`Person (out-domain)`	**0.084**	0.005	**0.083**
`Company (in-domain)`	**0.125**	0.004	0.101
`Company (out-domain)`	**0.11**	0.004	0.091
`Tournament`	**0.072**	0.014	0.052
`Title`	**0.060**	0.004	0.055

Table 4: The α values of different methods

4.4 Usefulness

We present an extrinsic evaluation on entity resolution and relation extraction tasks, both of which require reasoning over name variations of entities. We implement a configurable variant generation program that uses structure of an entity to generate its name variations. It allows the user to configure a set of string transformation functions for the semantic units in a structure, which are then used to compose name variations. It supports four transformations: DROP (ignore a token), INITIAL (retain first character of a token), INITIAL$_{dot}$ (retain first character followed by a dot) and MAP (replace with user-provided string).

Entity Resolution

We use ERLearn (Qian et al., 2017), a state-of-the-art system for large-scale entity resolution (ER). ERLearn aims to learn a set of *matching rules*, each consisting of *matching functions* for different entity attributes (e.g., to determine duplicate names 'John Smith' and 'J. Smith'). The matching functions are mostly based on textual similarity of attributes. We replace these with matching functions created using LUSTRE[4]. We focus on two scenarios: (1) `Emp-Social` (matching ∼470k employee records with 50 million social network user profiles) (2) `Crystal` (de-duplicating ∼1.3 million company records). We include new matching functions for *person* names and *company* names for the two scenarios respectively.

Type	Algorithm	# links	Precision	# true links
EMP-SOCIAL	ERLearn-CIKM	1088	**0.93**	∼1015
	ERLearn-LUSTRE	**1178**	0.9	**∼1060**
CRYSTAL	ERLearn-CIKM	145,516	0.9	∼130,964
	ERLearn-LUSTRE	**147,232**	0.9	**∼132,508**

Table 5: ERLearn-CIKM vs. ERLearn-LUSTRE

As shown in Table 5, the new matching functions help ERLearn identify 4.40% and 1.17% more true links for `Emp-Social` and `Crystal` scenarios, respectively. This improvement is significant for

[4]Due to space constraints, we request readers to refer to the supplementary material for details

Emp-Social given the extreme low matching ratio for this dataset. ERLearn-CIKM already identifies a significant subset of the true links in such a sparse space that it is non-trivial for ERLearn-LUSTRE to have found additional 45 true links. In contrast, the ER task is more challenging for the Crystal scenario, with more matching functions (158 vs. 68) (Qian et al., 2017). ERLearn-LUSTRE could still identify additional 1544 true links, which is a significant improvement over 130k true links already identified by ERLearn-CIKM. [5]

Relation Extraction

We use MULTIR (Hoffmann et al., 2011), a state-of-the-art relation extractor trained on NY Times text (Riedel et al., 2010) with weak supervision from Freebase (Bollacker et al., 2008). The weak supervision data is generated by exactly matching the textual mentions to canonicalized entities in Freebase. We instead match to the variations of entities of types *Person* and *Company*, generated using the configurations in Table 6. We follow the approach of (Hoffmann et al., 2011) to generate supervision data, compute features and evaluate aggregate extraction.

For the 2 million entities, we generate 5.3 million variations. There were 24,882 sentences where textual mentions exactly matched a canonical Freebase entity and one of the specified relations existed between the entities. This increased to 34,197 sentences when named variations of entities were included, suggesting name variations are useful for entity recognition. By including the variations for only two entity types, we could generate a training data that improved the overall extractor performance (F-1 score increased by 3% from 0.485 to 0.499). The extractor could further benefit from variations for entities of other types.

Person		
⟨first⟩	INITIAL, INITIAL$_{dot}$	*Dr. R. M. Nelson*
⟨middle⟩	INITIAL, INITIAL$_{dot}$, DROP	*Dr. Russell Nelson*
⟨title⟩	DROP	*Russell Nelson II*
⟨suffix⟩	DROP	*Dr. Russell Nelson*
Company		
⟨name⟩	INITIAL, INITIAL$_{dot}$	*HP USA Inc.*
⟨suffix⟩	DROP	*Hewlett-Packard USA*
⟨location⟩	DROP	*Hewlett-Packard Inc.*

Table 6: Configurations for generating variants

5 Related Work

Exploiting the compositional structure of entities and attributes especially from query streams and text has received much attention in databases and NLP literature. It has mostly been used for understanding NL questions (Berant et al., 2013), noun-phrase queries (Li, 2010) or normalizing time expressions (Lee et al., 2014; Bethard, 2013). Consequently, structuring and linking information on the web (Bollacker et al., 2008; Auer et al., 2007) about entities and their attributes, has seen a rise in interest. With this information being automatically extracted from textual data (Fader et al., 2011; Carlson et al., 2010), reconciling variations in entities and attribute names has become an integral part of the effort. Some recent work (Halevy et al., 2016) has attempted to organize attribute names by learning their compositional structure. On the other hand, some have proposed complex normalization frameworks (D'Souza and Ng, 2015) for specific domains. However, we need methods that can learn structured representations for the large scale of entity types found on the web.

Named entities are not atomic units and often contain other entities (Finkel and Manning, 2009). However, entity resolution has relied largely on surface-level match of entity mentions (Riedel et al., 2010; Hoffmann et al., 2011; Xu et al., 2013). Variations are typically handled using similarity functions such as edit distance, jaccard similarity, which have limited customizability. While learning string transformation rules (Arasu et al., 2009; Singh and Gulwani, 2012) to reconcile variations has been studied in different contexts, it typically relies on a set of input-output examples. It is difficult to obtain such data for entities and their variations. We instead propose a different approach to first learn the internal structure of an entity and then enable configurable transformations on its structure to generate its variations.

We focus on the problem of reducing manual effort in selecting a set of representative examples for learning the regular expression patterns, which is different from the problem of synthesizing regular

[5] More results of the entity resolution experiment can be found in the appendix.

expressions from examples (Bartoli et al., 2012; Li et al., 2008).

6 Conclusion

This paper identifies a novel problem of understanding structured representations of entities for handling their name variations. We propose an active-learning based approach, LUSTRE, to iteratively learn the structured representations of an entity type from a few labeled mentions and a large set of unlabeled mentions. With small manual effort, it can learn these structured representations and automatically generate programs to map mentions to their structured representations. Reasoning over such structured representations is useful for entity resolution and relation extraction that require reasoning over name variations of entities. In the future, we plan to extend our approach to learn structures of nested entities, and use sophisticated variant generation algorithms that could rank the variations based on their reliability.

Acknowledgements

This work was supported in part by IBM under contract 4915012629 and a graduate fellowship, and by a grant from the UM Office of Research.

References

Arvind Arasu and Raghav Kaushik. 2009. A grammar-based entity representation framework for data cleaning. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 233–244. ACM.

Arvind Arasu, Surajit Chaudhuri, and Raghav Kaushik. 2008. Transformation-based framework for record matching. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 40–49. IEEE.

Arvind Arasu, Surajit Chaudhuri, and Raghav Kaushik. 2009. Learning string transformations from examples. *Proceedings of the VLDB Endowment*, 2:514–525.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Alberto Bartoli, Giorgio Davanzo, Andrea De Lorenzo, Marco Mauri, Eric Medvet, and Enrico Sorio. 2012. Automatic generation of regular expressions from examples with genetic programming. In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, pages 1477–1478. ACM.

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

Steven Bethard. 2013. A synchronous context free grammar for time normalization. In *EMNLP*, pages 821–826.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM.

Adriano Crestani Campos, Yunyao Li, Sriram Raghavan, and Huaiyu Zhu. 2015. Entity variant generation and normalization, June 23. US Patent 9,063,926.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3.

Peter Christen. 2012. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE transactions on knowledge and data engineering*, 24(9):1537–1555.

Aron Culotta and Andrew McCallum. 2005. Reducing labeling effort for structured prediction tasks. In *AAAI*, volume 5, pages 746–751.

Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. 2010. Entity disambiguation for knowledge base population. In *COLING*.

Jennifer D'Souza and Vincent Ng. 2015. Sieve-based entity linking for the biomedical domain. In *ACL (2)*, pages 297–302.

Ahmed K Elmagarmid, Panagiotis G Ipeirotis, and Vassilios S Verykios. 2007. Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering*, 19(1):1–16.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics.

Jenny Rose Finkel and Christopher D Manning. 2009. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 141–150. Association for Computational Linguistics.

Luis Galárraga, Geremy Heitz, Kevin Murphy, and Fabian M Suchanek. 2014. Canonicalizing open knowledge bases. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1679–1688. ACM.

Alon Halevy, Natalya Noy, Sunita Sarawagi, Steven Euijong Whang, and Xiao Yu. 2016. Discovering structure in the universe of attribute names. In *Proceedings of the 25th International Conference on World Wide Web*, pages 939–949. International World Wide Web Conferences Steering Committee.

Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: a graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 765–774. ACM.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Kenton Lee, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. 2014. Context-dependent semantic parsing for time expressions. In *ACL (1)*, pages 1437–1447.

John Lehmann, Sean Monahan, Luke Nezda, Arnold Jung, and Ying Shi. 2010. Lcc approaches to knowledge base population at tac 2010. In *TAC*.

Yunyao Li, Rajasekar Krishnamurthy, Sriram Raghavan, Shivakumar Vaithyanathan, and HV Jagadish. 2008. Regular expression learning for information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 21–30. Association for Computational Linguistics.

Xiao Li. 2010. Understanding the semantic structure of noun phrase queries. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1337–1345. Association for Computational Linguistics.

Xiaohua Liu, Yitong Li, Haocheng Wu, Ming Zhou, Furu Wei, and Yi Lu. 2013. Entity linking for tweets. In *ACL (1)*, pages 1304–1311.

Andrew McCallum, Kamal Nigam, and Lyle H Ungar. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178. ACM.

Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. 2011. Scalable knowledge harvesting with high precision and high recall. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 227–236. ACM.

Kun Qian, Lucian Popa, and Prithviraj Sen. 2017. Active learning for large-scale entity resolution. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 1379–1388, New York, NY, USA. ACM.

Kun Qian, Nikita Bhutani, Yunyao Li, H. Jagadish, and Hernandez Mauricio. 2018. Lustre: An interactive system for entity structured representation and variant generation. In *Data Engineering (ICDE), 2018 IEEE 34th International Conference on*. IEEE.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics.

Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460.

Rishabh Singh and Sumit Gulwani. 2012. Learning semantic string transformations from examples. *Proceedings of the VLDB Endowment*, 5:740–751.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57.

Wei Xu, Raphael Hoffmann, Le Zhao, and Ralph Grishman. 2013. Filling knowledge base gaps for distant supervision of relation extraction. In *ACL (2)*, pages 665–670.

Wei Zhang, Chew Lim Tan, Yan Chuan Sim, and Jian Su. 2010. Nus-i2r: Learning a combined system for entity linking. In *TAC*.

Zhicheng Zheng, Fangtao Li, Minlie Huang, and Xiaoyan Zhu. 2010. Learning to link entities with knowledge base. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 483–491. Association for Computational Linguistics.

7 Appendix A. Experimental Setup of Entity Resolution

Our entity resolution experiments use ERLearn (Qian et al., 2017) on two scenarios `Emp-Social` and `Crystal`. We aim to provide it supplementary matching functions that reason over structured representations of entity names. To learn structured representations for the `Emp-Social` scenario, we randomly sample 1000 person names from the *Emp* records and 1000 person names from the online social network data. Using LUSTRE over the 2000 names, we learn the structures for *Person* with five different semantic units: $\langle first \rangle$, $\langle last \rangle$, $\langle middle \rangle$, $\langle suffix \rangle$, and $\langle title \rangle$. For the `Crystal` scenario, we learn structures for *Company* names using a sample of 2000 names. These structures have semantic units $\langle name \rangle$, $\langle industry \rangle$, $\langle suffix \rangle$, and $\langle location \rangle$.

Person		
⟨first⟩	INITIAL, INITIAL$_{dot}$	*Dr. R. M. Nelson*
⟨last⟩	INITIAL, INITIAL$_{dot}$	*Dr. Russell N.*
⟨middle⟩	INITIAL, INITIAL$_{dot}$, DROP	*Dr. Russell Nelson*
⟨title⟩	DROP	*Russell Nelson II*
⟨suffix⟩	DROP	*Dr. Russell Nelson*
ORDER	⟨title⟩⟨first⟩⟨middle⟩⟨last⟩⟨suffix⟩	
Company		
⟨name⟩	INITIAL, INITIAL$_{dot}$	*GM U.S. Trading Corp.*
⟨industry⟩	DROP	*General Motors U.S. Corp.*
⟨suffix⟩	DROP	*General Motors U.S.*
⟨location⟩	DROP	*General Motors Trading Corp.*
ORDER	⟨name⟩⟨industry⟩⟨suffix⟩⟨location⟩	

Table 7: Configurations with examples

To design matching functions that reason over structured representations to identify duplicates, we refer to the name variations of mentions generated using their representations. Intuitively, duplicate mentions are likely to share many name variations. Given a mention, we first generate its variations using the configurations in Table 7. We keep both the original string and the transformed string for a semantic unit modified using an operator from the configuration. For instance, dropping the title in a person name would generate Dr. Russell Nelson Sr. and Russell Nelson Sr. as variations of Dr. Russell Nelson Sr.. We define matching functions to identify two mentions as duplicates if they have at least r common variations.

matchPerson(Emp.name, social.name, r),
matchCompany(cp1.name, cp2.name, r),
matchCompany(cp1.parent.name, cp2.parent.name, r),

where $r = 1, 2, \ldots, 20$. These matching functions complement the matching functions used in the original study (Qian et al., 2017). We compare the ER rules learned by ERLearn in two different configurations, ERLearn-LUSTRE and ERLearn-CIKM, with and without the new matching functions respectively. The rules learned by ERLearn-LUSTRE had similar constituent matching functions as rules from ERLearn-CIKM. However, we found they additionally included the new matching functions and had relaxed some of the matching functions. For illustration, consider the following rules:

```
match Emp i, Social s By R:
  matchPerson(i.Name, s.name, 9)
  AND i.name.last = s.name.last
  AND lastNameFreqFilter(s.name.last, 50%)
  AND sameCity(i.CITY, s.city)
  AND countryIsInUS(i.COUNTRY, s.country)
```

(a) ERLearn-LUSTRE

```
match Emp i, Social s By R:
  nameMatch(i.Name.firstNameVars,s.name.first)
  AND i.Name.last = s.name.last
  AND lastNameFreqFilter(s.name.last, 60)
  AND upperCase(i.CITY)= upperCase(s.home.city)
  AND countryIsInUSA(i.COUNTRY)
```

(b) ERLearn-CIKM

There are to key differences: (1) the matching function for first names now uses the new matching function *matchPerson*, (2) the threshold value used in *lastNameFreqFilter* decreased from 60% to 50%. In contrast to the original rules, the lower threshold value for last names in the new rule makes it less conservative, potentially increasing the risk of identifying incorrect links. However, the matching function *matchPerson* makes the rule less susceptible to over-generalization. We found that by including matching functions that exploit the structured representations of entities, ERLearn could learn an ER model with appropriate generalization.

A Practical Incremental Learning Framework For Sparse Entity Extraction

Hussein S. Al-Olimat[1]*, Steven Gustafson[2], Jason Mackay[3]
Krishnaprasad Thirunarayan[1], and Amit Sheth[1]
[1] Kno.e.sis Center, Wright State University, Dayton, OH
{hussein;tkprasad;amit}@knoesis.org
[2] Maana Inc, Bellevue, WA
steven.gustafson@maana.io
[3] GoDaddy Inc, Kirkland, WA
jmackay@godaddy.com

Abstract

This work addresses challenges arising from extracting entities from textual data, including the high cost of data annotation, model accuracy, selecting appropriate evaluation criteria, and the overall quality of annotation. We present a framework that integrates Entity Set Expansion (ESE) and Active Learning (AL) to reduce the annotation cost of sparse data and provide an online evaluation method as feedback. This incremental and interactive learning framework allows for rapid annotation and subsequent extraction of sparse data while maintaining high accuracy.

We evaluate our framework on three publicly available datasets and show that it drastically reduces the cost of sparse entity annotation by an average of 85% and 45% to reach 0.9 and 1.0 F-Scores respectively. Moreover, the method exhibited robust performance across all datasets.

1 Introduction

Entity extraction methods delimit all mentions of an entity class (e.g., Location Names, Proteins, or Auto Parts IDs) in corpora of unstructured text. Supervised machine learning approaches have been effective in extracting such entity mentions, such as (Finkel et al., 2005). However, the prohibitive cost of obtaining a large volume of labeled data for training makes them unattractive and hard to use in realistic settings where resources may be scarce or costly to create (e.g., aviation engineer needing to annotate engine maintenance data). In this paper, we address the problem by developing a practical solution exploiting the strengths of the supervised sequence labeling techniques (Ye et al., 2009) while reducing the high cost of annotation.

Training data and the annotation approach are the two focal points of any supervised method. A model can be built from pre-annotated data or using *de novo annotations*. There are many pre-annotated and publicly available corpora such as CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003), GENIA (Ohta et al., 2001), and BioCreative II (et. al., 2008). However, having such data leaves us with two challenges: (i) no training dataset can be found for many of the domain-specific entity classes in enterprise corpora due to data privacy concerns, IP restrictions, and problem specific use cases (needing *de novo annotations*), and (ii) data sparsity, which hinders the performance of supervised models on unseen data (needing *incrementally augmented annotations* to the annotated sentence pool).

We propose a practical solution to model building through (i) rapid auto-annotation, to create models with reduced cost, and (ii) flexible stopping criteria using an online evaluation method, which calculates the confidence of a model on unseen data without a need for a gold standard. Having such an online feedback method supports incremental learning which allows us to partially overcome the data sparsity problem (see Section 2).

Mainstream entity annotation approaches typically present a large body of text or full documents to annotators, which can be time-consuming as well as lead to low quality annotations. Additionally, the majority of these techniques require labeling for multiple entity classes which makes the annotation task

*Part of this work has been done during the author's internship at Maana Inc.

Proceedings of the 27th International Conference on Computational Linguistics, pages 700–710
Santa Fe, New Mexico, USA, August 20-26, 2018.

harder and more complex. Therefore, sentence-level and single-entity-class annotation are desirable to improve tractability and reduce cost. Imposing such requirements, however, causes a labeling starvation problem (i.e., querying annotators to label sentences containing a low frequency of entities from the desired class). In this work, we develop an Entity Set Expansion (ESE) approach to work side by side with Active Learning (AL) to reduce the labeling starvation problem and improve the learning rate.

Our framework is similar to the work proposed by (Tsuruoka et al., 2008) that aims to annotate all mentions of a single entity class in corpora while relaxing the requirement of full coverage. Our design choice lowers the annotation cost by using flexible stopping criteria, assisting in the corpus annotating procedure and removing the requirement for annotators to scan all sentences. Our approach differs from the above in that we perform ESE by learning the semantics and relevant context to retrieve similar entities by analogy, to accelerate the learning rate. Moreover, our approach provides Fast (FA), Hyper Fast (HFA), and Ultra Fast (UFA) auto-annotation modes for rapid annotation. Our contributions include:

1. A framework to annotate sparse entities rapidly in unstructured corpora using auto-annotation. We then use flexible stopping criteria to learn sequence labeling models incrementally from annotated sentences without compromising the quality of the learned models.

2. A comprehensive empirical evaluation of the proposed framework by testing it on six entity classes from three public datasets.

3. An open source implementation of the framework within the widely used Stanford CoreNLP package (Manning et al., 2014).[1]

In the upcoming sections, we describe our incremental learning solution developed using Active Learning, and how we use ESE to accelerate the annotation. Finally, we will evaluate the proposed method on publicly available datasets showing its effectiveness.

2 Incremental and Interactive Learning

In practical applications, domain experts have datasets from which they want to extract entities and then use these entities to build various applications like identifying the mentions of the same suppliers and materials referred to by similar entities in sourcing databases for the purpose of optimizing supply chains. Next, we will describe how we use AL to accelerate the learning rate, to auto-annotate sentences, and to provide feedback for flexible stopping criteria.

Active Learning (AL) We adopt the linear-chain Conditional Random Field (CRF) implementation by (Finkel et al., 2005) to learn a sequence labeling model to extract entities. However, we use pool-based AL for sequence labeling (Settles and Craven, 2008) to replace the sequential or random samplers during online corpus annotation.

Our iterative AL-enabled framework requires a pre-trained/base model to sample the next batch of sentences to be labeled. This iterative sampling allows us to reach higher accuracies with fewer data points by incrementally incorporating the new knowledge encoded in the trained models. This more informative sampling is achieved due to the added requirement on model training that uses all previously annotated sentences as well as new batches of annotated sentences[2]. In Section 3, we show how learning the base model by annotating sentences sampled using a method called Entity Set Expansion (ESE) is better than learning a model from a random sample.

While the default inductive behavior of CRF is to provide one label sequence (the most probable one), we instead use an n-best sequence method that uses the Viterbi algorithm to return, for each sentence, the top n sequences with their probabilities. We then query the annotator to annotate b number of sentences (equal to a pre-selected batch size) with the highest entropies (Kim et al., 2006; Settles and Craven, 2008) calculated using the following equation:

[1]Codes and data can be found at `https://github.com/halolimat/SpExtor`

[2]We train all CRF models using the default set of features for CoNLL 4 class - `https://rebrand.ly/corenlpProp`

$$nSE(m, s) = -\sum_{\hat{y} \in \hat{Y}} p_m(\hat{y}|s; \theta)\ log\ p_m(\hat{y}|s; \theta) \quad (1)$$

where m is the trained CRF model, s is a sentence, $\hat{Y}$ is the set of n sequences, and θ is the features' weights learned by the model m.

Annotation Modes We devise four AL-enabled annotation modes: **1.** ESE and AL (EAL), **2.** Fast (FA), **3.** Hyper Fast (HFA), and **4.** Ultra Fast (UFA) annotation mode. The EAL mode (which perform ESE to learn the base model, see Section 3) uses model confidence on sentences estimated using nSE for sampling while the rest use a thresholded auto-annotation method that we developed to accelerate the annotation procedure.

Auto-annotation employs a pre-trained CRF model m to annotate all unlabeled sentences in the pool. Then, we accept the most probable annotation of each sentence (i.e., the first label sequence) from the model if the first two label sequences satisfies the following condition: **if** $\frac{SE_1(s)}{SE_2(s)} \leq t$, where $SE_i(s)$ is the entropy of the sequence i of the sentence s and t is a predefined threshold representing the desired margin difference between the entropies of the two sequences 1 and 2. We chose t after running some experiments. What we found was that any threshold below 0.10 usually resulted in no auto-annotations, while a threshold above 0.20 resulted in many incorrect ones. Therefore, we chose 0.10, 0.15, and 0.20 for the FA, HFA, and UFA auto-annotation modes, respectively. In Section 4.4, we evaluate the effectiveness of the devised annotation modes.

Interactive Learning We designed an online evaluation method σ that provides feedback to annotators on the confidence of a model m on a given sentence pool S (Equation 2). This feedback is an alternative to the F-measure and is very valuable in the absence of a gold standard dataset that could otherwise provide this kind of a feedback.

$$\sigma = 1 - \frac{1}{|S|} \sum_{s \in S} nSE(m, s) \quad (2)$$

Typically, $\sigma \in [0, 1]$, where 1 is the highest confidence value. Hence, it gives the annotator a clearer picture of whether to keep the model as is or learn a new model, interactively, using the mean of all nSEs (i.e., $\frac{1}{|S|} \sum_{s \in S} nSE(m, s)$). We use σ both during the incremental learning steps from the same sentence pool and to test the model's accuracy/confidence on unseen sentences. Therefore, using this feedback, we can decide to stop annotating from a certain sentence pool and augment it with sentences from a new pool, or simply stop annotating and train a final model. Consequently, deciding to annotate new sentences that contain novel entities helps reduce the effect of data sparsity and increase the accuracy of models, which highlights the importance of this online feedback method.

We compare our online evaluation method with the Estimated Coverage (EC) method in (Tsuruoka et al., 2008) that computes the expected number of entities as follows:

$$EC = \frac{E}{E + \sum_{u \in U} E_u} \quad (3)$$

where E is the number of annotated entities in sentences, U is the set of all unlabeled sentences, and E_u is the expected number of entities in sentence u. E_u is calculated by summing the probability of each entity in all of the n-best sequences. As shown later in Section 4.4, our online method is more informative than the EC method.

3 Entity Set Expansion Framework

Since AL requires a base (pre-trained) model to work (Settles and Craven, 2008), learning that model from annotated sentences sampled using sequential or random sampling can be very expensive due to the labeling starvation problem mentioned before. Therefore, we developed an Entity Set Expansion (ESE)

method that incorporates and exploits the semantics and relevant context of the desired entity class to more informatively sample sentences that are likely to have entities of the desired class.

We assume that all noun phrases (NPs) are candidate entities and extract all of them from the unstructured text[3]. Then, we record five features for each noun phrase (np) and model the data as a bipartite graph with NPs on one side and features on the opposite side. Ultimately, modeling the edges from multi-modal edge weights allow us to calculate the similarity between NPs and retrieve all sentences containing the most similar ones.

3.1 Noun Phrase Extraction (NPEx)

We POS tag and parse sentences to the form ($[w_i/pos_i \; \forall w \in W]$). We then use the regular expressions below to extract NPs, where JJs are adjectives, NNs are nouns (singular, plural, proper, etc.), and CDs are cardinal numbers (Santorini, 1990).

```
NP = JJs + NNs + CDs
JJs = (?:(?:[A-Z]\\w+ JJ )*)
NNs = (?:[^\\s]* (?:N[A-Z]*)\\s*)+
CDs = (?:\\w+ CD)?
```

3.2 Featurization

We automatically extract and record five kinds of features for each extracted noun phrase from text to create relationships between noun phrases. Following is the list of the diverse lexical, syntactic, and semantic features we derive:

① **Lexical Features (LF):**

- **Orthographic Form (OF)**: We abstract OF features from the actual word and obtain its type (i.e., numeric, alpha, alphanumeric, or other). Additionally, we classify the word as: all upper case, all lower case, title case, or mixed case.
- **Word Shape (WS)**: We define WS to abstract the patterns of letters in a word as short/long word shape (SWS/LWS) features. In LWS, we map each letter to "L", each digit to "D", and retain the others unchanged. On the other hand, for SWS, we remove consecutive character types. For example, LWS("ABC-123") $\rightarrow$ "LLL-DDD" and SWS("ABC-123") $\rightarrow$ "L-D".

② **Lexico-Syntactic Features (LS):** We use a skip-gram method to record the explicit LS-patterns surrounding each NP, i.e., for each np in s we record the pattern $w_{i-1} + np + w_{i+1}$, where $w_{(.)}$ are the two words that precede and follow np.

③ **Syntactic Features (SF):** We use dependency patterns to abstract away from the word-order information that we can capture using contextual and lexico-syntactic features. We use Stanford's English_UD neural network-based dependency parser (Chen and Manning, 2014) to extract universal dependencies of NPs. For each np in s, we record two dependency patterns of the NP that serve in *governor* and *dependent* roles.

④ **Semantic Features (SeF):** To capture the lexical semantics, we use WordNet (Miller, 1995) to get word senses and draw sense relations between NPs if they have the same sense class.

⑤ **Contextual Features (CF):** To capture the latent features, we use a Word2Vec embedding model (Mikolov et al., 2013) trained on the sentence pool S as the only bottom-up distributional semantics method. We exploit the word-context co-occurrence patterns learned by the model to induce the relational similarities between NPs.

The use of semantic (i.e., word senses) and contextual (i.e., word embeddings) features for sparse or domain-specific data (such as enterprise data) is not enough and sometimes not even possible due to unavailability (Tao et al., 2015). Therefore, we tried to use diverse features in a complementary manner

[3]While not all candidate NPs are positive examples of an entity, a human-in-the-loop would interactively give feedback to the system by choosing the positive ones.

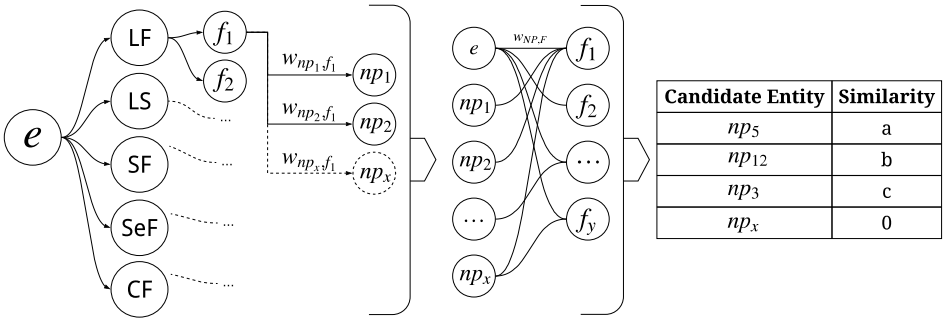


Figure 1: The three stages of graph embedding for candidate entities ranking.

to capture as many meaningful relations between potential entities as possible. For example, the absence of SeF for domain specific entities, such as protein molecules or parts numbers, is compensated by the use of WS features (e.g., by drawing a relation between the NPs "IL-2" and "AP-1" which have the same SWS and LWS features).

3.3 Feature-Graph Embedding

We model edges in the bipartite graph by assigning a weight w between each pair of a noun phrase n and a feature f using one of the following weights:

$$w1_{n,f} = C_{n,f} \tag{4}$$

$$w2_{n,f} = log(1 + C_{n,f})[log|N| - log(|N|_f)] \tag{5}$$

$$w3_{n,f} = log(1 + C_{n,f})[log|N| - log(\sum_{\hat{n}} C_{\hat{n},f})] \tag{6}$$

where $C_{n,f}$ is the co-occurrence count between n and f, $|N|$ is the number of NPs in our dataset, $|N|_f$ is the co-occurrence count of all NPs with f, and $\sum_{\hat{n}} C_{\hat{n},f}$ is the sum of all NP co-occurrences with the feature f. Equations 5 and 6 are two variations of TFIDF where the latter is adapted to weigh the edges in (Shen et al., 2017; Rong et al., 2016).

3.4 Set Expansion

Our method starts by taking a seed entity e from the annotator input, and returns a ranked list of similar NPs (see Algorithm 1). After constructing a graph using the embedding method mentioned in the previous section (see Figure 1), we calculate the similarity between the seed entity e and all other noun phrases NPs in the graph G using one of the following similarity methods:

$$Sim1(n_1, n_2|F) = \frac{\sum_{f \in F} w_{n_1,f} w_{n_2,f}}{\sqrt{\sum_{f \in F} w_{n_1,f}^2} \sqrt{\sum_{f \in F} w_{n_2,f}^2}} \tag{7}$$

$$Sim2(n_1, n_2|F) = \frac{\sum_{f \in F} \min(w_{n_1,f}, w_{n_2,f})}{\sum_{f \in F} \max(w_{n_1,f}, w_{n_2,f})} \tag{8}$$

Data: e: input seed; S: text sentences
Result: $N = \{n\}$: ranked similar noun phrases
Start
$\hat{N}$ = ∅; `// all noun phrases`
F = ∅; `// all features`
for $s \leftarrow S$ **do**
  $\hat{N} = \hat{N} \cup$ **ExtractNounPhrases**(s);
  $F = F \cup$ **Featurize**($\hat{N}$); `// Sec. 3.2`
end
$G =$ **BuildBipartiteGraph**($\hat{N}$, F);
  `// Section 3.3`
$N =$ **CalculateSimilarity**(e, G);
  `// Equation 7 or 8`
return $rank(N)$;

Algorithm 1: Entity Set Expansion

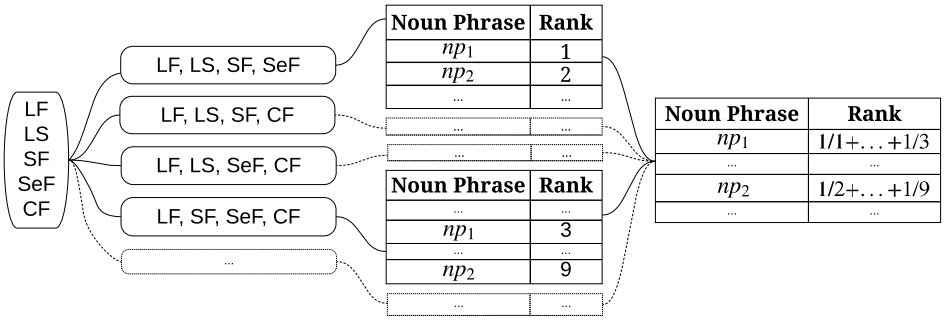


Figure 2: Coarse features ensemble method for ranking candidate entities.

Dataset Name	Entity Class	$\|S\|$ (with Entities)	# Entities	NPEx Precision	Seed Entity (Count)
CoNLL-2003	Location	13519 (36%)	1258	84%	U.S. (296), Washington (26)
	Person	13519 (31%)	3388	76%	Clinton (70), Von Heesen (1)
BioCreAtIvE II	Gene	12500 (51%)	10441	51%	insulin (64), GrpE (1)
GENIA 3.02	Protein Molecule	14838 (55%)	3413	60%	NF-kappa B (552), IL-1RA (1)
	Cell Type	14838 (27%)	1569	24%	T cells (489), MGC (2)
	Virus	14838 (8%)	324	50%	HIV-1 (336), adenovirus E1A (1)

Table 1: Evaluation dataset statistics, noun phrase extraction accuracies, and seed entity counts.

where $Sim(n_1, n_2|F)$ is the similarity between the two noun phrases n_1 and n_2 given the set of features F they have in common, and $w_{(.,.)}$ is the weight of the edge between the NPs and features (defined using one of the Equations 4-6). Equation 7 is the cosine similarity and Equation 8 is the context-dependent similarity by (Shen et al., 2017). The difference between these two similarity measure is that higher cosine similarities implies that the two NPs are better aligned on many feature dimensions, while higher context-dependent similarities implies that the two NPs are significantly similar on some feature dimensions compared to others.

3.5 Feature Ensemble Ranking

Similar to the use in SetExpan (Shen et al., 2017), we ensemble the coarse features[4] and rank NPs in sublists to reduce the effect of inferior features on the final ranking. The size of each sublist is equal to $|\hat{F}| - 1$, where $\hat{F}$ is the set of all features from Section 3.2.

$$MRR(n) = \sum_{q \in Q} \frac{1}{rank(n, q)} \tag{9}$$

We then use the mean reciprocal rank (MRR) from Equation 9 above, a form of rank aggregation, to find the final ranking of each noun phrase n using its rank in each sublist q (see Figure 2).

4 Experiments and Results

4.1 Data Preparation

To test our framework while emulating the full experience of annotators, we use three publicly available gold standard datasets labeled for several entity classes (CoNLL-2003, GENIA 3.02, and BioCreAtIvE II). We tokenize documents into sentences, then query the emulator (which plays the role of annotators) to label sentences, thus avoiding the required annotation of full documents for better user experience. We require the emulator to label only a single entity class from each dataset, to avoid the complexity of multiple class annotations. Therefore, we created six versions of the datasets where only one class is kept in each version. Table 1 includes some statistics about the datasets. The percentage of sentences with entities of a given class shows the sparsity of those entities.

[4]Coarse features are the five groups of features in Section 3.2 as opposed to the fine ones which are part of each of them.

		Location			Person			Gene			Protein			Cell Type			Virus		
		Eq.4	Eq.5	Eq.6	Eq.4	Eq.5	Eq.6	Eq.4	Eq.5	Eq.6	Eq.4	Eq.5	Eq.6	Eq.4	Eq.5	Eq.6	Eq.4	Eq.5	Eq.6
Seed 1	Eq.7	0.37	0.40	0.50	0.23	0.23	0.30	0.00	0.03	0.13	0.17	0.23	0.20	0.27	0.50	0.53	0.20	0.13	0.17
	Eq.8	0.63	**0.73**	0.73	0.03	**0.17**	0.20	0.03	**0.07**	0.07	0.43	**0.43**	0.53	0.17	**0.23**	0.23	0.07	**0.10**	0.07
Seed 2	Eq.7	0.33	0.33	0.57	0.53	0.40	0.30	0.63	0.63	0.63	0.17	0.60	0.27	0.10	0.20	0.13	0.07	0.03	0.03
	Eq.8	0.57	**0.70**	0.63	0.47	**0.37**	0.37	0.60	**0.57**	0.60	0.07	**0.30**	0.30	0.07	**0.07**	0.07	0.03	**0.10**	0.03

Table 2: ESE performance ($p@k$). The use of TFIDF variant without co-occurrence summation (Equation 5), to weight the graph edges, and the context-dependent similarity (Equation 8), to measure the similarity between NPs, led to the bold-faced best performing combination.

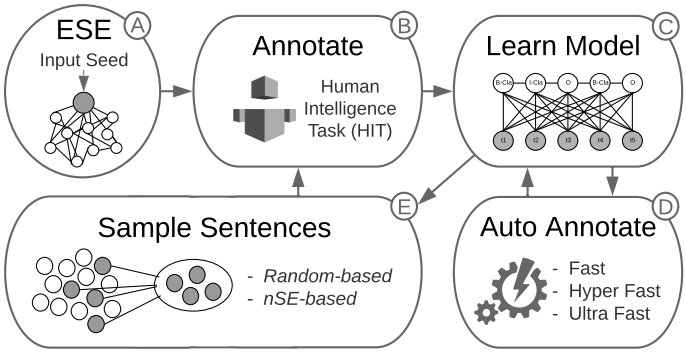


Figure 3: End-to-End system pipeline. Arrows represent paths can be followed to annotate text.

4.2 NPEx Evaluation

Since ESE method operates at an NP level, the performance of Noun Phrase Extraction (NPEx) significantly influences the overall performance of ESE. Table 1 includes the precision of NPEx method. In the future, ESE performance can be improved by enhancing the performance of candidate entities extraction through, for example, extracting noun phrases formed from typed-dependencies.

4.3 Entity Set Expansion (ESE) Evaluation

To test the influence of seed entity frequency on ESE performance, we manually picked two seeds, the most frequent and the least frequent among all noun phrases (See Table 1). Additionally, we varied the weighting measure of the graph edges to reflect different notions of feature relevance using one of Equations 4-6. Finally, we varied the similarity measure when ranking NPs (Equations 7 and 8).

We tested the performance of ESE with and without using the feature ensemble method in Section 3.5. We measured the precision of the method in ranking positive examples of entities similar to the seed entity (Seed 1 and Seed 2) in the top k NPs. We designed ESE to output thirty candidate entities (NPs) ranked based on the similarity to the seed term. Therefore, we calculated precision at k ($p@k$) where k is always 30. Table 2 shows the best results when using the feature ensemble method which is more stable than the non-ensemble one (due to lower standard deviation and non-zero precision). According to the results, the best combination in terms of the mean and standard deviation is obtained when using TFIDF (Eq.5) to weigh the edges and context-dependent similarity (Eq.8) to rank NPs. This shows that the uniqueness and the significant overlap of features between noun phrases were very important.

4.4 Full System Evaluation

We tested our pipeline on three different settings following three different paths in Figure 3:

1. All Random (AR): $(E, B, C)^*$, where E is a random-based sampler.
2. ESE and AL (EAL): $A, B, C, (E, B, C)^*$, where E is an nSE-based sampler.
3. EAL in addition to auto-annotation (EAA): $A, B, C, (D, C, E, B, C)^*$

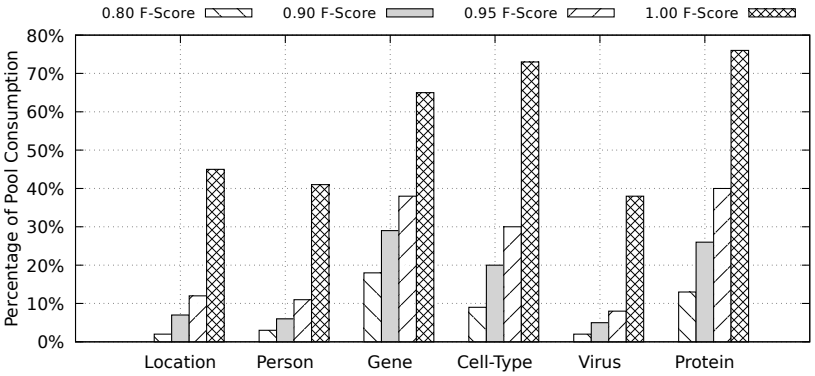


Figure 4: Percentage of sentences annotated while using EAL to reach different F-Scores.

Dataset Name	Entity Class	EAL		EAA Annotation Mode		
		@ 1.0 F		FA	HFA	UFA
		σ	% cut	F-Score (percentage cut)		
CoNLL-2003	Location	0.97	55%	0.99 (46%)	0.93 (83%)	0.82 (91%)
	Person	0.97	59%	0.99 (48%)	0.95 (81%)	0.85 (90%)
BioCreAtIvE II	Gene	0.94	35%	1.00 (35%)	0.96 (50%)	0.89 (69%)
GENIA 3.02	Protein Molecule	0.99	33%	0.98 (36%)	0.87 (71%)	0.74 (85%)
	Cell Type	0.99	62%	0.94 (70%)	0.82 (86%)	0.74 (91%)
	Virus	0.94	24%	0.97 (79%)	0.89 (94%)	0.84 (96%)
Average		0.97	45%	0.98 (52%)	0.90 (78%)	0.81 (87%)

Table 3: Pipeline testing results of EAL and EAA annotation modes (see Section 2) showing the model confidence (σ), F-Scores, and percentage cut from the pool of sentences.

We iterate through the loop paths in the starred parentheses while sampling 100 sentences each time until we finish all sentences in the pool or reach full F-Score. In Figure 5, we show the performance of the first two settings (i.e., AR and EAL) in terms of F-Score. The use of AL and ESE methods outperformed random sampling all the time. ESE increased the performance of the base model by 35% F-Score on average, allowing us to reach 0.5 F-Score while the random sampler reached only 0.37 F-Score.

As shown in Figure 4, the percentage consumption of the sentence pool to reach up to 0.95 F-Score follows almost a linear growth. However, this cost grows exponentially if we want to reach 1.0 F-Score (needing on average around 33% more sentences). Practically speaking, we might want to trade the 5% F-Score improvement to significantly reduce manual labor.[5]

We used Jensen-Shannon divergence (Lin, 1991) to compare the curves corresponding to the performance of the estimated coverage method (Equation 3) and our online evaluation metric σ (Equation 2) with the F-Score curve of EAL (see Figure 5). Our method σ outperformed the estimated coverage method with a decrease of 96% in dissimilarity, which makes our method more reliable. Additionally, as shown in Table 3, on average σ gives an accurate estimation of the F-Score without labeled data, with an error margin of 3% on average, when reaching 1.0 F-Score (i.e., @ 1.0 F).[6]

Our EAL method also outperformed (Tsuruoka et al., 2008) on the three overlapping entity classes we tested on from the two datasets (Genia and CoNLL-2003). EAL needed 2800, 2900, and 400 fewer sentences to reach the same coverage as (Tsuruoka et al., 2008) for the Location, Person, and Cell-Type datasets, respectively.

Finally, for the last setting, we tested the system using the three auto-annotation modes (i.e., FA, HFA,

[5]This would, therefore, make the annotated data a silver standard instead of a gold standard.

[6]The σ curves of GENIA-Cell-Type and GENIA-Virus show high overestimation of the F-Score curve in the first iterations. This is due to missing entity annotations we found in the gold standard, which mistakenly shows that we had many false positives. To list a few, GENIA-Cell-Type have missing annotations for "transformed T cells", "transformed cells", and "T cells", where GENIA-Virus have some missing annotations for "HIV-1" and "HIV-2".

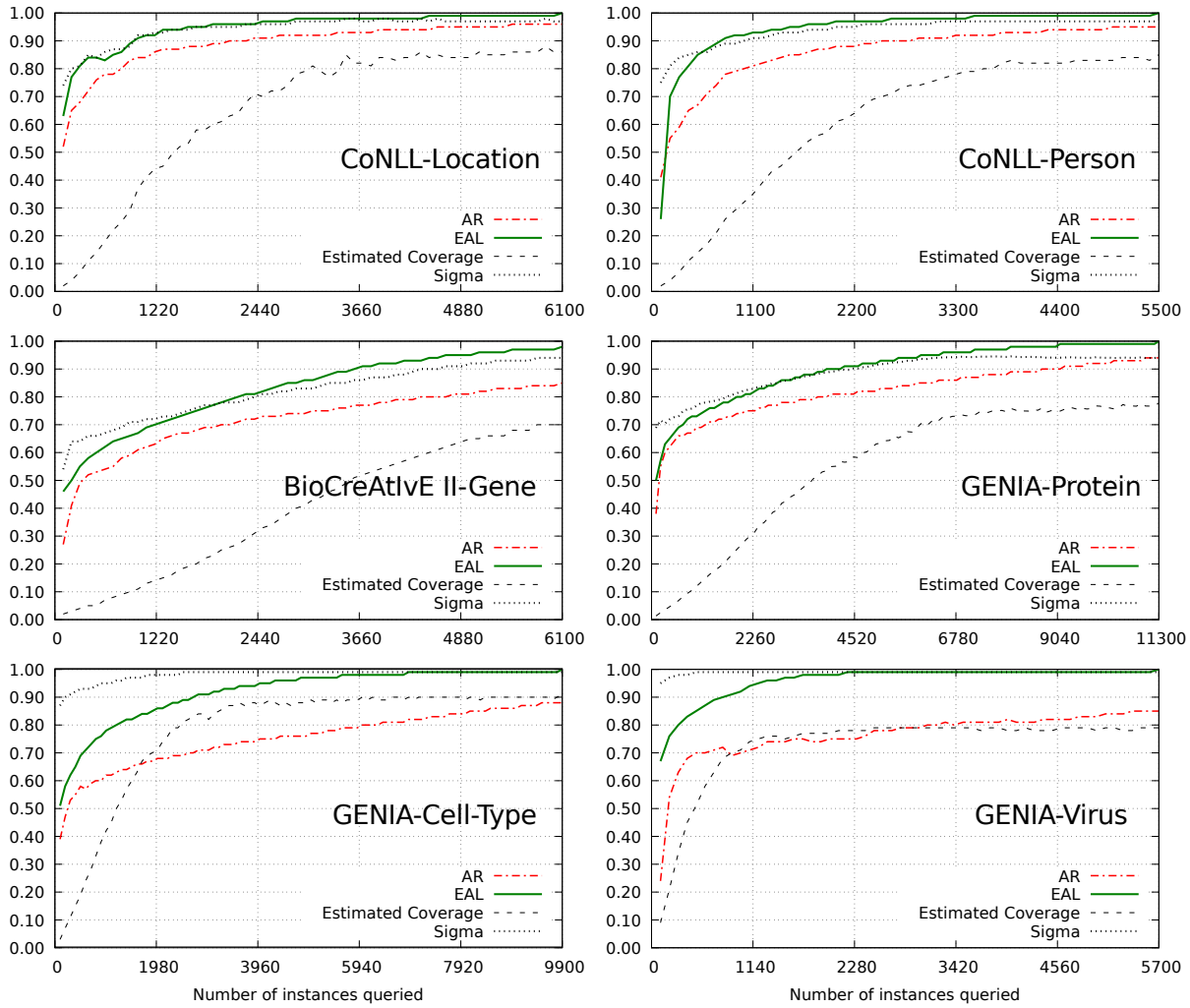


Figure 5: Learning curves with different query strategies. Y-axis value for AR and EAL is the F-Score, and for Sigma and Estimated Coverage is the value from Equations 2 and 3, respectively.

and UFA) as shown in Table 3. While the auto-annotation mode can allow us to reduce up to 87% of the data pool, this drastic saving also reduces the accuracy of the learned model, achieving, on average, around 81% F-Score. Overall, our framework presents a trade off between coverage and annotation cost. The HFA auto-annotation mode shows the benefit, especially in a realistic enterprise setting, when we need to annotate 33% of the data to increase F-Score by only 10% (when comparing the on average performance of HFA with ESA) is unreasonable.

Table 3 appears to show FA being inferior to EAL in terms of the percentage cut for the Location class, for example. In reality FA reduced sentence annotation by 65% to reach 0.99 F-Score. But as our testing criteria demanded that we either reach 1.0 F-Score or finish all sentences from the pool, FA tried to finish the pool without any further performance improvement on the 0.99 F-Score.

5 Related Work

To extract sparse entities from texts and learn sequence models faster, our practical framework uses Active Learning (AL) for sequence labeling (Tomanek and Hahn, 2009; Settles and Craven, 2008), both as a method of sampling and to auto-annotate sentences. Our work encompasses entity extraction, Entity Set Expansion (ESE), corpora pre-annotation, auto-annotation, and AL.

Using a pattern-based ESE (a.k.a., seed set expansion) technique on top of AL helped our approach in discovering rare patterns and rules which might have been hidden when using only a feature-based system (Chiticariu et al., 2013). Our ESE method is similar to (Shen et al., 2017; Tao et al., 2015;

Sadamitsu et al., 2012) where the system starts with a few positive examples of an entity class and tries to extract similar entities. Additionally, we use a richer set of features than those in (Gupta and Manning, 2014; Grishman and He, 2014; Min and Grishman, 2011) while training a CRF model and using it to find additional positive examples of a given entity class.

Methods such as (Sarmento et al., 2007) computes the degree of membership of an entity in a group of predefined entities called seed entities, where the class of each group is predefined. Their method does not focus on entity extraction, rather it focuses on detecting the class of entities during the evaluation step, which makes their method different from ours.

Regarding corpus annotation, many notable previous works such as (Lingren et al., 2013) use dictionaries to pre-annotate texts. However, inaccurate pre-annotations may harm more than improve since they require an added overhead of modifications and deletions (Rehbein et al., 2009). Kholghi et al. (2017) and Tsuruoka et al. (2008) propose AL-based annotation systems. However, their work differs from ours in the following ways: **(1)** we propose a more accurate online evaluation method than theirs, **(2)** we use ESE to bootstrap the learning framework collaboratively with a user-in-the-loop, and finally, **(3)** we provide auto-annotation modes which reduces the number of sentences to be considered for annotation and so allows for better usability of the framework.

6 Conclusions and Future Work

We presented a practical and effective solution to the problem of sparse entity extraction. Our framework builds supervised models and extracts entities with a reduced annotation cost using Entity Set Expansion (ESE), Active Learning (AL), and auto-annotation without compromising extraction quality. Additionally, we provided an online method for evaluating model confidence that enables flexible stopping criteria. In the future, we might consider evaluating different AL querying strategies and compare their performances and try a more sophisticated candidate entity extractors for entity set expansion.

References

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Laura Chiticariu, Yunyao Li, and Frederick R Reiss. 2013. Rule-based information extraction is dead! long live rule-based information extraction systems! In *EMNLP*, number October, pages 827–832.

Larry Smith et. al. 2008. Overview of biocreative ii gene mention recognition. *Genome Biology*, 9(2):S2, Sep.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics.

Ralph Grishman and Yifan He. 2014. An information extraction customizer. In Petr Sojka, Aleš Horák, Ivan Kopeček, and Karel Pala, editors, *Text, Speech and Dialogue*, pages 3–10, Cham. Springer International Publishing.

Sonal Gupta and Christopher D Manning. 2014. Improved pattern learning for bootstrapped entity extraction. In *CoNLL*, pages 98–108.

Mahnoosh Kholghi, Laurianne Sitbon, Guido Zuccon, and Anthony Nguyen. 2017. Active learning reduces annotation time for clinical concept extraction. *International journal of medical informatics*, 106:25–31.

Seokhwan Kim, Yu Song, Kyungduk Kim, Jeong-Won Cha, and Gary Geunbae Lee. 2006. Mmr-based active machine learning for bio named entity recognition. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 69–72. Association for Computational Linguistics.

Jianhua Lin. 1991. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151.

Todd Lingren, Louise Deleger, Katalin Molnar, Haijun Zhai, Jareen Meinzen-Derr, Megan Kaiser, Laura Stoutenborough, Qi Li, and Imre Solti. 2013. Evaluating the impact of pre-annotation on annotation speed and potential bias: natural language processing gold standard development for clinical named entity recognition in clinical trial announcements. *Journal of the American Medical Informatics Association*, 21(3):406–413.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Bonan Min and Ralph Grishman. 2011. Fine-grained entity set refinement with user feedback. *Information Extraction and Knowledge Acquisition*, page 2.

Tomoko Ohta, Yuka Tateisi, Jin-Dong Kim, Sang-Zoo Lee, and Junichi Tsujii. 2001. Genia corpus: A semantically annotated corpus in molecular biology domain. In *Proceedings of the ninth International Conference on Intelligent Systems for Molecular Biology (ISMB 2001) poster session*, volume 68.

Ines Rehbein, Josef Ruppenhofer, and Caroline Sporleder. 2009. Assessing the benefits of partial automatic pre-labeling for frame-semantic annotation. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 19–26. Association for Computational Linguistics.

Xin Rong, Zhe Chen, Qiaozhu Mei, and Eytan Adar. 2016. Egoset: Exploiting word ego-networks and user-generated ontology for multifaceted set expansion. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 645–654. ACM.

Kugatsu Sadamitsu, Kuniko Saito, Kenji Imamura, and Yoshihiro Matsuo. 2012. Entity set expansion using interactive topic information. In *PACLIC*, pages 108–116.

Beatrice Santorini. 1990. Part-of-speech tagging guidelines for the penn treebank project (3rd revision). *Technical Reports (CIS)*, page 570.

Luis Sarmento, Valentin Jijkuon, Maarten de Rijke, and Eugenio Oliveira. 2007. More like these: growing entity classes from seeds. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 959–962. ACM.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics.

Jiaming Shen, Zeqiu Wu, Dongming Lei, Jingbo Shang, Xiang Ren, and Jiawei Han. 2017. Setexpan: Corpus-based set expansion via context feature selection and rank ensemble. In *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2017)*.

Fangbo Tao, Bo Zhao, Ariel Fuxman, Yang Li, and Jiawei Han. 2015. Leveraging pattern semantics for extracting entities in enterprises. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1078–1088. International World Wide Web Conferences Steering Committee.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Katrin Tomanek and Udo Hahn. 2009. Semi-supervised active learning for sequence labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1039–1047. Association for Computational Linguistics.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 2008. Accelerating the annotation of sparse named entities by dynamic sentence selection. *BMC bioinformatics*, 9(11):S8.

Nan Ye, Wee S Lee, Hai L Chieu, and Dan Wu. 2009. Conditional random fields with high-order features for sequence labeling. In *Advances in Neural Information Processing Systems*, pages 2196–2204.

An Empirical Study on Fine-Grained Named Entity Recognition

Khai Mai[1,*] **Thai-Hoang Pham**[1,*] **Nguyen Minh Trung**[1] **Nguyen Tuan Duc**[1]
Danushka Bolegala[2] **Ryohei Sasano**[3] **Satoshi Sekine**[4]
[1)] Alt Inc [2)] University of Liverpool
[3)] Nagoya University [4)] Riken AIP
{mai.tien.khai, pham.thai.hoang, nguyen.minh.trung, nguyen.tuan.duc}@alt.ai,
danushka.bollegala@liverpool.ac.uk, sasano@i.nagoya-u.ac.jp, satoshi.sekine@riken.jp

Abstract

Named entity recognition (NER) has attracted a substantial amount of research. Recently, several neural network-based models have been proposed and achieved high performance. However, there is little research on fine-grained NER (FG-NER), in which hundreds of named entity categories must be recognized, especially for non-English languages. It is still an open question whether there is a model that is robust across various settings or the proper model varies depending on the language, the number of named entity categories, and the size of training datasets. This paper first presents an empirical comparison of FG-NER models for English and Japanese and demonstrates that LSTM+CNN+CRF (Ma and Hovy, 2016), one of the state-of-the-art methods for English NER, also works well for English FG-NER but does not work well for Japanese, a language that has a large number of character types. To tackle this problem, we propose a method to improve the neural network-based Japanese FG-NER performance by removing the CNN layer and utilizing dictionary and category embeddings. Experimental results show that the proposed method improves Japanese FG-NER F-score from 66.76% to 75.18%.

1 Introduction

Named entity recognition (NER) is a well studied topic in natural language processing. There have been many methods proposed for NER, including the conventional methods based on Conditional Random Fields (CRF) (McCallum and Li, 2003), Support Vector Machines (SVM) (Yamada et al., 2002; Takeuchi and Collier, 2002) and Hidden Markov Model (HMM) (Zhou and Su, 2002). Recently, neural network based methods, such as LSTM+CNN+CRF (Ma and Hovy, 2016) or BiLSTM/LSTM-CRF/Stack LSTMs (Lample et al., 2016; Misawa et al., 2017), have achieved state-of-the-art performance. However, while most existing studies mainly focus on recognizing a relatively small number of named entity (NE) categories (e.g., ten or twelve categories) such as Person, Organization, Location, Artifact, etc., modern NLP applications often require domain-specific and fine-grained (FG) NER with hundreds of NE categories. For example, a movie recommendation system might require the recognition of movie names, but does not need to recognize Locations. Similarly, a chatbot software might require not only the recognition of Organization, but also the fine-grained classification to recognize a music band name to answer the question "Which band was Paul in", from the information shown in Figure 1.

A fine-grained named entity recognition (FG-NER) model refers to a NER model that can recognize and classify a large number of entity categories (e.g., hundreds of NE categories). In classical coarse-grained named entity (NE) definition, often less than ten named entity categories are defined. For example, in the CoNLL-2003 Named Entity Recognition task, there are four NE categories: Person, Location, Organization and Miscellaneous (Sang and Meulder, 2003). Ritter et al. (2011) proposed a NER algorithm to recognize ten categories of entities from Twitter text. On the other hand, in FG-NER, there are hundreds of NE categories, which are fine-grained classification of coarse-grained categories.

*) Equally contributed to the paper

Proceedings of the 27th International Conference on Computational Linguistics, pages 711–722
Santa Fe, New Mexico, USA, August 20-26, 2018.

Paul, a former member of The Beatles, known for "Let It Be",
Person *Organization* *Artifact*
will be holding a concert at Carnegie Hall in New York.
Location *Location*

(a) Named entity recognition result

Paul, a former member of The Beatles, known for "Let It Be",
Person *Org > Show_Org* *Product > Art > Music*
will be holding a concert at Carnegie Hall in New York.
Facilty > GOE > Theatre *Location > GPE > City*

(b) Fine-grained NER result

Figure 1: Example of NER and FG-NER

For example, Sekine (2008) divided the coarse-grained category Organization into the fine-grained categories such as Political_Party, Military, Sports_Organization, Show_Organization, as shown in Figure 2.

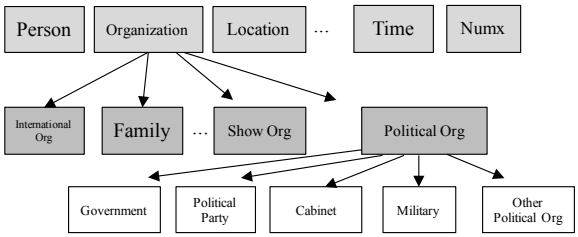


Figure 2: Sekine's Extended Named Entity (ENE) hierarchy

FG-NER is still an open research domain, with little information concerning the state-of-the-art performance, the relation between training data size and performance, and how to select the best model for different settings of training data size and target language. In FG-NER, because the number of NE categories is large, some categories might face with the data sparseness problem, whereas some other categories might have a large number of training samples in a dataset. Hence, it would be worth investigating the relation between dataset size and the performance of the system. The current state-of-the-art method for English NER (coarse-grained NER) is a neural network-based method, which uses convolutional neural network (CNN) to calculate the character level embeddings (Ma and Hovy, 2016). This leads to the question whether this method works well for languages with a large number of character types, such as Japanese.

In this paper, we first investigate the relationship between the F-score of various FG-NER algorithms with the size of training datasets for both English and Japanese. Second, we suggest the direction to choose an appropriate FG-NER algorithm for appropriate target language and training data size. We show that the state-of-the-art method for English NER also performs well for English FG-NER. On the other hand, for Japanese FG-NER, the state-of-the-art method does not work well. To solve this problem, we propose a method to significantly improve the neural network-based Japanese FG-NER performance by removing the CNN layer, as well as utilizing dictionary and category embeddings information. Experiments show that, the proposed method improves the F-score of the Japanese FG-NER system from 66.76% to 75.18%, which is a wide margin. We applied the proposed method to build an FG-NER system that can recognize 200 categories of named entities in the Sekine's Extended Named Entity Hierarchy (ENEH) (Sekine, 2008). To the best of our knowledge, the proposed system achieves the state-of-the-art performance in the task of recognizing 200 NE categories in the Sekine's ENEH. We publish the test dataset[1] and our FG-NER API on the Web to allow other researchers to freely use[2].

The rest of this paper is organized as follows. In Section 2, we describe the fine-grained named entity tag sets and datasets. We present various algorithms for FG-NER and show our proposal to achieve high performance for Japanese FG-NER in Section 3. We present detailed experimental results to find out the relationships between models, training data size and target languages in Section 4. Finally, Section 5 concludes the paper.

[1] https://fgner.alt.ai/duc/ene/testsets/comp/en/
[2] https://fgner.alt.ai/extractor/

2 Fine-Grained Named Entity Tag Sets and Datasets

2.1 FG-NER tag set

The first challenge in FG-NER is defining a comprehensive tag set with a very large number of entity categories (Ling and Weld, 2012). There are two methods for defining a tag set (i.e., set of entity categories to recognize) in previous studies on FG-NER. The first method is to take the entity categories from a knowledge base such as Freebase (Bollacker et al., 2008) or YAGO (Suchanek et al., 2007), filtering out the categories with a small number of entities and merging the categories with similar semantic meaning into one FG-NER category (Ling and Weld, 2012; Yosef et al., 2012; Gillick et al., 2014). The second method is to manually build an entity hierarchy to cover important domains in the real world. Following the second method, Sekine et al. proposed an Extended Named Entity Hierarchy (ENEH), which contains 200 entity categories in a three-layer hierarchy, as shown in Figure 2 (Sekine et al., 2002; Sekine, 2008).

In this paper, we use the entity hierarchy described by Sekine (2008), which contains 200 NE categories at the leaf-level, as our tag set[3]. At the top level of the hierarchy, there are about twenty coarse-grained named entity categories, such as Person, Organization, Location, Facility, Product, Event, ... Each top-level categories is further divided into several second-level categories as shown in Figure 2. Each second-level category is in turn divided into several leaf-level categories.

We use this hierarchy because it is carefully designed by humans, it does not ignore important domains with small numbers of entities (e.g., Continental_Region) and it includes a systematic categorization of date/time and number. Moreover, for subsequent applications such as search engines or chatbot platforms to easily utilize the FG-NER results, we want to classify an entity to exactly one category in the hierarchy based on the context in which the entity appears. Consequently, we need an entity category hierarchy that does not allow overlap between the categories (Sekine, 2008). For all experiments in this paper, we will build Fine-grained NERs to recognize the leaf-level categories in this hierarchy. The parent level categories can be easily inferred once we recognize the leaf-level categories.

2.2 FG-NER Dataset

We hired several human annotators to annotate two text corpora to create FG-NER tagged datasets for English and Japanese. The number of annotators for English is ten and the number of annotators for Japanese is seven. All of the annotators are native speakers of the corresponding language.

For each category in the Sekine's Extended Named Entity (ENE) Hierarchy (Sekine, 2008), the human annotators are first asked to write down 100 entities that belong to each ENE category. We then search the Web for sentences that contain these entities. In the search result, we retrieve the sentences that include at least one entity of the corresponding category. The human annotators then tag the sentences with 200 ENE labels. For example, for the entity "Tokyo", the Web search results might contain the sentence "Tokyo is the capital of Japan". This sentence must then be tagged as "<City>Tokyo</City> is the capital of <Country>Japan</Country>". Consequently, we have 20,000 sentences for English and 20,000 sentences for Japanese (as the number of leaf-level categories defined in Sekine's ENEH is 200). Note that the number of entities is larger than the number of sentences because in one sentence we might have multiple entities, of different NE categories.

After filtering out erroneous sentences (sentences with invalid tag format, e.g., without closing the tag </City>), we obtain totally 19,800 well-formed English sentences and 19,594 well-formed Japanese sentences. Each category has at least 100 sentences containing the entities of that category. We divided the dataset into three subsets: the training set (70% of the total data), development set (10%) and test set (20%), as shown in Table 1. We use the development set to check the stop condition while training our LSTM model.

To have an estimation of the difficulty of the FG-NER annotation task, we measured the coherence between the annotators by calculating two coefficients for some categories in the English dataset: the Fleiss' kappa (κ) and the F-score of an annotator when assuming that another annotator created gold-

[3]The full tag set is here: https://nlp.cs.nyu.edu/ene/version7_1_0Beng.html

Table 1: Statistics of the datasets

Dataset	English		Japanese	
	Sents	Entities	Sents	Entities
Train	13,860	27,107	13,749	37,128
Dev	1,980	3,870	1,948	5,304
Test	3,960	7,739	3,897	10,485

Table 2: Agreement between annotators

Category	Fleiss' κ	F-score
Country	0.977	0.966
Dish	0.890	0.738
Car_Stop	0.873	0.672
Organization_Other	0.835	0.483

standard data[4]. The results are shown in Table 2. We choose these categories to calculate the coefficients because they represent typical categories in the dataset: a category with limited number of frequently used entities (*Country*), a category with entities that are often not proper nouns (*Dish*), a category with ambiguous and complicated location names (*Car_Stop*) and a category that is ambiguously defined (*Organization_Other*).

The κ coefficient is calculated on tokens so it is only slightly different between the four categories in Table 2. This is because the ratio between the number of tokens with label "Other" is very large, compared against the number of tokens with specific NE category labels. We calculated Fleiss' κ on tokens but not on entities because at entity level, there are some cases in which two annotators made overlapping tags, but not identical. For example, Annotator1 might tag "<City>Greater Tokyo Area</City>" and Annotator2 might tag "Greater <City>Tokyo</City> Area". If we calculate on entity level then the score would be 0 but if we calculate on token level, the score is greater than 0. Consequently, we calculate the κ based on B/I/O tags at token level.

On the other hand, when calculating the entity-based F-score, the difference is very large between the category *Country* and *Organization_Other*. This is because the category *Country* is very easy to recognize, as there are only about 200 entities frequently used in this category, whereas, recognizing *Organization_Other* or *Car_Stop* is very difficult because of the ambiguity. This also indicates that the performance of an FG-NER system tends to depend on the categories and we can confirm this in the experimental results in the next sections.

3 Fine-Grained Named Entity Recognition Methods

3.1 Dictionary and Rule-based FG-NER

The simplest method for FG-NER is using a dictionary and a set of rules. Sekine and Nobata (2004) presented a dictionary and rule-based Japanese FG-NER system that contains more than 1400 rules to recognize 140 entity categories.

In this work, we added 200 rules to the existing 1400 rules by Sekine and Nobata to create a rule set of 1600 rules to classify 200 NE categories in the Sekine's Extended Named Entity Hierarchy. We then built a rule-based Japanese FG-NER model to recognize 200 NE categories based on these 1600 rules.

We use a Japanese FG-NER dictionary containing 1.6 million Wikipedia entities in this model. In the 1.6 million entities in the dictionary, only 70 thousand entities are assigned NE tags by human, the rest are assigned by an existing Wikipedia NE labeling algorithm (Suzuki et al., 2016), which gives a score for each (entity, NE category) pair. We created similar rules for English FG-NER and we translated the Japanese dictionary into English by looking up parallel entries in Wikipedia.

We use this method as a baseline for performance evaluation of FG-NER systems.

3.2 CRF+SVM hierarchical classifier for FG-NER

Hierarchical classifiers have been successfully used in previous research for FG-NER (Ling and Weld, 2012; Yosef et al., 2012). Ling and Weld proposed FIGER, an FG-NER system with the entity categories taken from Freebase tags (Ling and Weld, 2012). In FIGER, the entity category is represented as a path, such as `/location/city` or `organization/company`. FIGER divides the entity categories into

[4]In this case, we only have two annotators for each categories, so Fleiss' kappa is equal to Cohen's kappa

a hierarchy of two layers: the categories in the first layer corresponding to the categories in coarse-grained NER systems, whereas, the second layer (the leaf-layer) contains fine-grained entity categories. FIGER uses a hierarchical classifier that contains a CRF at the top layer for sequence labelling and then a Perceptron at the second layer for classification of the entities into fine-grained categories.

In this work, we propose a hierarchical classification method in which CRF is used for sequence labelling at top-level and SVM is used for named entity classification at leaf-level of the Sekine's ENE Hierarchy, as shown in Figure 3. We use SVM at the leaf-level to classify an entity to fine-grained categories because SVM is good for classification tasks.

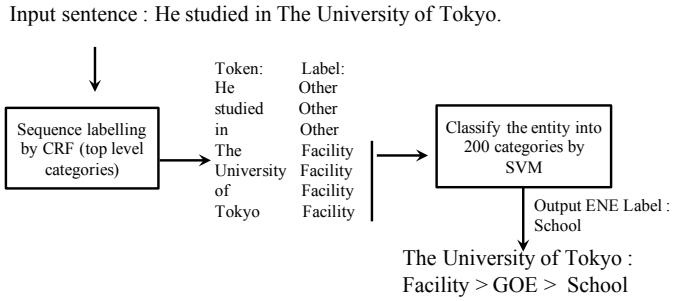


Figure 3: CRF+SVM based hierarchical classifier for FG-NER

Specifically, we use a training dataset (containing FG-NER tagged sentences) to train a CRF model to tag the input sentences with the top-level ENE categories in Figure 2. As illustrated in Figure 2, at the top level, we have only less than 20 categories of entities, thus using a CRF model here would achieve comparable performance with existing NER systems. We cannot directly use CRF for a large number of categories at leaf-level of the hierarchy, because with this number of classes, it would take a huge amount of memory and running time to train the CRF model. Actually, we have tried to use CRF for 200 classes, but the training process took a long time and did not finish. After tagging the sentences with the top-level categories, we can convert the FG-NER problem into a simple classification problem (not a sequence labeling problem anymore), thus we can use SVM to classify the extracted entities at the top level into leaf-level categories. Therefore, we have a CRF model to tag the input sentences with top-level categories, and several SVM models (each for a top-level category) to classify the entities into the leaf-level categories.

We use the following features for both SVM and CRF: bag-of-words, POS-tag, the number of digits in the word, the Brown cluster of the current word, the appearance of the word as a substring of a word in the Wikipedia ENE dictionary, the orthography features (the word is written in Kanji, Hiragana, Katakana or Romanji), is capital letter, and the last 2-3 characters. Those features are proved to be useful in previous work on named entity recognition (Ling and Weld, 2012; Yosef et al., 2012; Yogatama et al., 2015; Suzuki et al., 2016).

Once we have the sequence labelling result, we have already known the surfaces and the top-level categories of the entities in the input sentence. We then use SVM to classify the entities into leaf-level categories. Because the number of leaf-level categories in each top-level categories is also not too large (e.g., less than 15), SVM can achieve a reasonable performance at this step.

We also propose a method to incorporate dictionary information in both CRF and SVM step to improve the entire performance, as described in Section 3.4.

3.3 LSTM+CNN+CRF model for FG-NER

We re-implemented the LSTM+CNN+CRF NER model described by Ma and Hovy (2016) and adjust the model to work with FG-NER. The LSTM+CNN+CRF model originally described by (Ma and Hovy, 2016) is for NER problem with few NE categories. It first uses Convolutional Neural Network (CNN) to learn character level embeddings in the training process. For NLP tasks, previous works have shown that CNN is likely to extract morphological features such as prefix and suffix effectively (Ma and Hovy,

2016; dos Santos and Guimarães, 2015; Chiu and Nichols, 2016). The model then concatenates the character level embeddings with word embeddings to create a feature vector for each token in the input sentence. The input sentence is then fed to a BiLSTM network (Bi-directional Long-Short Term Memory network). Finally, CRF is used at the top layer of the BiLSTM to explore the correlations between outputs and jointly decode the best sequence of labels (i.e., NE categories).

For both English and Japanese FG-NER task, we use pre-trained word embeddings as input for our models. Previous studies have shown that GloVe achieves the best performance for English NER task (Reimers and Gurevych, 2017). Consequently, we use the embeddings based on GloVe for English[5]. For Japanese, we use pretrained word2vec[6] embeddings. The vector dimension is 300 for English and 200 for Japanese.

We use the default hyperparameters by Ma and Hovy (2016) in our model: $learning\_rate = 0.01$, $batch\_size = 10$ and $decay\_rate = 0.09$.

3.4 Incorporating dictionary information

Dictionary information (gazetteer feature) has been proved to be efficient in many NER and FG-NER tasks (McCallum and Li, 2003; Sekine and Nobata, 2004; Yosef et al., 2012). While there are previous studies that use dictionary for CRF (McCallum and Li, 2003) or SVM (Yosef et al., 2012) in the NER/FG-NER tasks, we believe that dictionary information would be useful in both sequence labelling and entity category disambiguation phase in the CRF+SVM method. Furthermore, the dictionary information can also be used in LSTM+CNN+CRF method. Consequently, we propose a method that efficiently utilizes dictionary information in the method LSTM+CNN+CRF and in both sequence labelling (CRF) and entity category disambiguation (SVM) phase of the method CRF+SVM.

We search the dictionary and assign a label in the set $\{B, I, O\}$ for each token in the input sentence $w_1 w_2 \dots w_n$, in which a token w_i is assigned the label *B* if it is a start token of an entity in the dictionary, label *I* if it is a token inside an entity in the dictionary, otherwise, it is assigned the label *O*. If there is a conflict (e.g., overlapping with two different entities in the dictionary) then we take the entities with the largest number of tokens. This is because we want to tag the longest sequence that could be an entity (e.g., if we have "United States" and "United States Army" then we take "United States Army"). We do not directly use these labels as the final results of the sequence labelling phase since they are not reliable as we drop all context information while assigning labels. Instead, we use these labels as features for CRF model for sequence labelling the entire input sentence or we add it to the additional dimensions of the vector representation of a token in the method LSTM+CNN+CRF.

Other than features from previous research on entity disambiguation (Yosef et al., 2012; Suzuki et al., 2016), we propose to use the following feature derived from the dictionary. We tokenize all the entities in the dictionary and calculate the probability that a token w is contained in an entity of type c :

$$P(c|w) = \frac{\text{count}(w, c)}{\text{count}(w, *)} \tag{1}$$

in which $\text{count}(w, c)$ is the number of occurences of w in an entity of type c, whereas, $\text{count}(w, *)$ is the total number of times that w appears. We then use this probability as a feature for SVM to classify the entities into leaf-level categories in the method CRF+SVM and we directly add this feature to the feature vector of a token in the LSTM+CNN+CRF method. This feature is helpful because it represents the likelihood that an entity e (which is a sequence of tokens $w_i w_{i+1} \dots w_{i+m}$) is contained in the category c.

We add these features to the CRF+SVM method to have the CRF+SVM+Dict method. Similarity, while using these features for LSTM+CNN+CRF, we obtained LSTM+CNN+CRF+Dict method.

3.5 Utilize entity category embeddings to improve Japanese FG-NER performance

For languages with a large number of character types such as Japanese, the CNN layer in the LSTM+CNN+CRF+Dict network architecture might not work well. This is because with a large

[5]https://nlp.stanford.edu/projects/glove/

[6]http://www.cl.ecei.tohoku.ac.jp/∼m-suzuki/jawiki_vector/

number of character types (e.g., 2000 frequently used characters in Japanese), it is difficult to calculate high quality character embeddings. Consequently, we propose a modification to the existing LSTM+CNN+CRF network architecture to alleviate this problem: We remove the CNN layer from the LSTM+CNN+CRF+Dict method to have the LSTM+CRF+Dict method. Moreover, we add another additional information concerning the entity category, as described below.

In the spirit of taking advantage of available dictionaries to improve our model as in CRF+SVM, we propose to integrate statistical information from the dictionary to improve the word embeddings. Specifically, for a word, we form a vector in which each element is the probability that this word is contained in an entity category, as follows:

$$C(w) = (x_1, x_2, \ldots, x_{n-1}, x_n) \tag{2}$$

where x_i is the probability that word w is contained in i^{th} category (c_i) :

$$x_i = P(c_i|w) = \frac{\text{count}(w, c_i)}{\text{count}(w, *)} \tag{3}$$

We call these "entity category embeddings". For new words or words that do not appear in any entity category in the dictionary, their entity embeddings are zero at all dimensions. We concatenate this entity category embedding vector with the original word embedding vector to form a new word embedding vector. We call the **LSTM+CRF+Dict** method that also utilizes entity category embeddings as **LSTM+CRF+Dict+Cate**.

4 Experiments

4.1 Performance comparison between FG-NER methods

We evaluated the performance of the FG-NER methods using the test dataset, as described in Table 1. We calculated the Precision, Recall and F1-score for each category and the micro-average Precision and Recall over all 200 entity categories. Finally, we derived the average F1-score from the average Precision and Recall.

Method	English F-score (%)	Japanese F-score (%)
Rule+Dict	24.43	48.29
FIGER(Ling and Weld, 2012)	23.41	-
CRF+SVM+Dict	72.58	**73.30**
LSTM+CNN+CRF (Ma and Hovy, 2016)	80.93	66.76
LSTM+CNN+CRF+Dict	**83.14**	70.34
LSTM+CRF+Dict	81.89	73.05

(a) Average F-score of the FG-NER methods

Method	English	Japanese
LSTM+CNN+CRF (Ma and Hovy, 2016)	80.93	66.76
LSTM+CNN+CRF+Dict	**83.14**	70.34
LSTM+CNN+CRF+Dict+Cate	82.29	-
LSTM+CRF+Dict	-	73.05
LSTM+CRF+Dict+Cate	-	**75.18**

(b) Average F-score of the FG-NER systems with and without entity category embeddings

Category	Precision (%)	Recall (%)	F-score (%)
URL	100.00	100.00	100.00
Phone_Number	100.00	100.0	100.00
Cabinet	95.24	100.00	97.56
Country	90.06	92.67	91.35
...	...	...	...
Award	80.65	92.59	86.21
Car_Stop	80.00	76.19	78.05
Book	87.50	63.64	73.67
Dish	67.31	76.09	71.43
...	...	...	...
Mollusk_Arthropod	65.79	64.10	64.93
Art_Other	63.16	57.14	59.99
Organization_Other	57.14	48.00	52.17
Average	**83.25**	**83.04**	**83.14**

(c) Precision, Recall, F-score of the method LSTM+CNN+CRF+Dict on the English test dataset, sorted by F-score

Table 3: Performance of FG-NER methods

The evaluation results are shown in Table 3a. Rule and dictionary-based methods (**Rule+Dict**) can only achieve an F-score of less than 50%, even we used more than 1000 rules and a huge number of entries in the dictionaries. Because there are many Wikipedia entries that are in Japanese Wikipedia but they are not in English Wikipedia and we translated the dictionary from Japanese by looking up parallel

entries in English Wikipedia, these entities do not appear in the English dictionary. Consequently, the F-score of English **Rule+Dict** is very low.

The method **FIGER** is the CRF+Perceptron hierarchical classifier described by Ling and Weld (Ling and Weld, 2012). We use the published source code of **FIGER**[7] and our training dataset (identical to the training dataset of other methods in this paper) to get an FG-NER model. We evaluated this model with the test dataset described in the previous section. We can only evaluate **FIGER** with English because the source code of **FIGER** can only work with English (e.g., the feature extraction code can only work with English if we don't modify it). Although the architecture of the method **FIGER** is very similar to that of **CRF+SVM+Dict**, the performance is very different (the F-score is 24.43%, compared to 72.58%). This is because of several reasons. First, **FIGER** is designed for classifying the two-layer hierarchy of named entities based on Freebase. This hierarchy is different from the three-layer hierarchy of Sekine's Extended Named Entity Hierarchy (ENEH) that we use in this work. Second, we use SVM at the leaf-level, instead of Perceptron. We believe that SVM gives high performance than Perceptron in multi-class classification tasks. And finally, the training data size is not large enough for **FIGER** to work well. **FIGER** uses millions of training samples (which are automatically extracted from Wikipedia) to train the system (Ling and Weld, 2012), whereas, in our method, we only have about 13 thousand sentences in the training dataset. Automatically creating the NE tagged data is a reasonable approach for FG-NER definitions that are derived from existing knowledge bases like Freebase or Wikipedia. For FG-NER definitions that are not based on existing knowledge bases, it is difficult to automatically extract the NE tagged data from a large text corpus like Wikipedia. Consequently, for FG-NER system working with these definitions, we must design an algorithm that requires only a small amount of training data.

We can observe that the state-of-the-art method for English NER (Ma and Hovy, 2016) also works well with English FG-NER, as **LSTM+CNN+CRF** outperforms **CRF+SVM+Dict** by a wide margin (80.93%, compared to 72.58%). However, for Japanese FG-NER, the situation is different. The method **CRF+SVM+Dict** achieves the best performance, and it significantly outperforms the original **LSTM+CNN+CRF** model in (Ma and Hovy, 2016). Even when we add dictionary information, **CRF+SVM+Dict** still outperforms **LSTM+CNN+CRF+Dict** for Japanese FG-NER (the F-score is 73.30 and 70.34, respectively). We measured the statistical significance of the F-score difference by an approximate randomization test (Chinchor, 1992) with significance level $\alpha = 0.05$ and 99,999 iterations. The randomization test is used because we want to check the difference between the micro-average F-score over all ENE categories. The result of this test indicates that the difference in F-score is statistically significant (the p-value is 10^{-5}). The reason for this difference lies in the character level embeddings that **LSTM+CNN+CRF+Dict** creates in the CNN layer. For English, we only have less than 30 characters in the alphabet, whereas, in Japanese, we have more than 2,000 Hiragana, Katakana and Kanji characters that are frequently used (the total number of Japanese characters is above 10,000). Because the number of characters is large in Japanese, the CNN layer does not work well to create good quality character embeddings.

We verified the above reason by the following experiment: we removed the CNN layer from LSTM+CNN+CRF+Dict and we measured the performance on the test dataset. Removing the CNN layer means disabling the character level embedding features in the LSTM+CNN+CRF+Dict method. The result of this experiment is shown in the last row of Table 3a. We can observe that, removing the CNN layer boosts the performance of the neural network based method from 70.30% (**LSTM+CNN+CRF+Dict**) to 73.05% (**LSTM+CRF+Dict**), making it comparable with the performance of **CRF+SVM+Dict** (73.30%) for Japanese FG-NER. However, for English FG-NER, the performance decreased if we remove the CNN layer. This proves that the CNN layer works well for English, but it does not work for Japanese.

We also verify the difference between **CRF+SVM+Dict** and **LSTM+CRF+Dict** using a randomization test as described previously. For Japanese, the difference between **CRF+SVM+Dict** and **LSTM+CRF+Dict** is not statistically significant. This suggests that with enough amount of training data and dictionary information, we can use both CRF or LSTM for FG-NER problem.

[7] https://github.com/xiaoling/figer

4.2 Result of removing CNN and using entity category embeddings

To verify the effectiveness the proposed method for Japanese FG-NER, we compared the performance of the best neural network-based model for each language with and without the entity category embeddings. The results are shown in Table 3b.

Because removing the CNN layer does not improve the performance of English FG-NER, we compared **LSTM+CNN+CRF** with the method **LSTM+CNN+CRF+Dict** and **LSTM+CNN+CRF+Dict+Cate** for English. We observe that, by adding dictionary information, we can improve the F-score from 80.93% to 83.14%. This improvement is statistically significant under a randomization test (similar to the tests in Section 4.1). However, if we further add the entity category embedding information (in **LSTM+CNN+CRF+Dict+Cate**), the performance slightly decreased (from 83.14% to 82.29%, which is statistically significant under the randomization test). This is because the quality of the translated English dictionary is not good enough to have practical statistics information.

Table 3b also shows a series of improvements we made for Japanese FG-NER. First, if we add the dictionary information, we obtained a similar result with that of English FG-NER (we boost the performance from 66.76% to 70.34%, as shown in the 1st and 2nd row of Table 3b). Second, if we remove the CNN layer from the network, we improve the performance from 70.34% to 73.05%, as described in the previous section. Finally, if we add category entity embeddings, we further improve the performance from 73.05% to 75.18%. This makes the method **LSTM+CRF+Dict+Cate** significantly outperforms the method **CRF+SVM+Dict** (the randomization test result shows that the improvement is statistically significant). This proves the effectiveness of the newly added entity category embedding feature for Japanese FG-NER.

With this result, we can unify the FG-NER model for both English and Japanese: we don't need to use **CRF+SVM** anymore, but we can utilize neural network-based models for both languages. We can have an identical system for both English and Japanese FG-NER by simply setting an option to enable/disable the CNN layer and the entity category embedding information to switch between English and Japanese. This makes the engineering of the system easier and helps to cut the maintenance cost of the FG-NER system.

4.3 Performance of each category

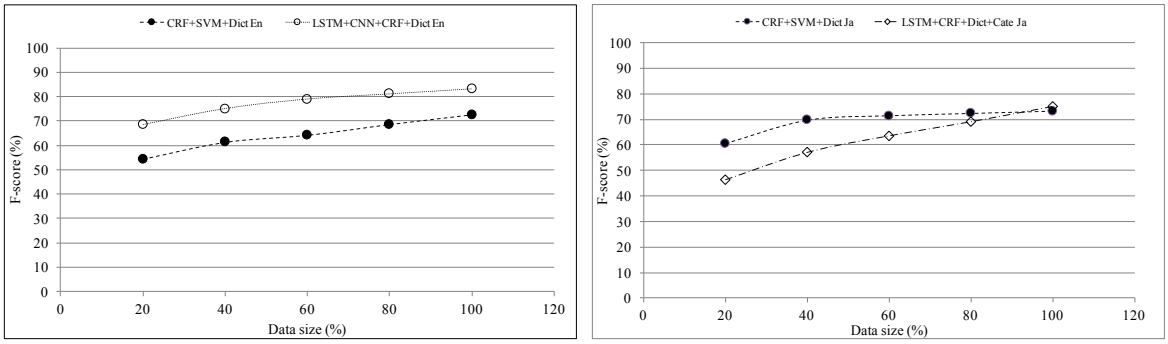
In the 200 NE categories in the Sekine's ENEH (Sekine, 2008), there are several categories that can be recognized by rule-based method (such as URL or Email). There are also some categories that are actually not named entities, and difficult to recognize, such as Mollusk_Arthropod (cellar spider, turbinidae,...) or Art_Other (e.g., "the Venus of Milo", "Genji Monogatari Emaki",...). Consequently, the performance of the FG-NER models for each category will be different. In this section, we investigate the performance some typical entity categories in the method **LSTM+CNN+CRF+Dict** (the best method for English).

Table 3c shows the Precision, Recall and F-score of the **LSTM+CNN+CRF+Dict** method on some specific categories as well as the average evaluation results of the entire 200 categories (in the last row). We achieved very high performance on the categories with a small number of known entities (such as Country) or the categories that the rules can capture almost all entities (such as Intensity, Volume, URL, and Email). For categories with free text names (e.g, printing names) or very short name (e.g., AK-47, a type of weapon) the system can not predict the NE tag very well because these names might appear in various contexts. This result is consistent with the difficulty of the tasks as described in Section 2.2 and in Table 2.

4.4 Training data size and performance

To find out the relation between the training data size with the FG-NER models, we varied the training data size and measured the F-score of each model on the test dataset.

Specifically, we randomly sample the training dataset by the rates of 20%, 40%, 60%, 80% and 100% (i.e., using the entire training dataset). We then use the sampled training dataset to train the FG-NER models and evaluated these models with the test dataset. The results of this experiment are shown in



(a) For English FG-NER models

(b) For Japanese FG-NER models

Figure 4: Relation between training data size and F-score

Figure 4a and Figure 4b.

For English, we observe that the neural network-based method (**LSTM+CNN+CRF+Dict**) consistently outperforms the method **CRF+SVM+Dict** by a wide margin, irrespective of the training data size. If we only use 20% of the training data (i.e., about 4000 training sentences), we can get the F-score of 68.49%, whereas, using the entire of the training dataset (100%), we get the F-score of 83.14%. However, from 60% of the data size, the F-score increases gradually. This indicates that, we can still get a comparable performance even if we do not have a large amount of training data (an F-score of 78.68% for 60% of training data size).

For Japanese, the situation is totally different. When the training data size is small, **CRF+SVM+Dict** outperforms **LSTM+CRF+Dict+Cate** by a wide margin (F-score of 60.60%, compared to 45.43%). However, when the training data size is increased to 100%, **LSTM+CRF+Dict+Cate** is the best method (it achieves an F-score of 75.18%, compared to 73.30% of **CRF+SVM+Dict**). Therefore, if we only have a small amount of training data (e.g., 4,000 sentences), we must use **CRF+SVM+Dict**. If we have a larger amount of training data (e.g., 20,000 sentences), we can use the neural network-based model **LSTM+CRF+Dict+Cate** to achieve the best performance. When the amount of training data is large enough, the LSTM layer can learn the relation between the NE tags and the underlying words (tokens) as well as the grammatical structures. Consequently, it can precisely model the NE tagging problem.

These comparison results give us a suggestion to choose the appropriate model for each setting of target language and training data size: if the target language is English then we should use the neural network-based methods; if the language is Japanese then we should use **CRF+SVM+Dict** if we only have little amount of training data and we use the neural network-based method when we have enough amount of training data.

5 Conclusion

We presented an empirical study of fine-grained named entity recognition (FG-NER) methods. We investigated the relation between the performance of the methods with various settings of target languages and training data sizes. We found that the state-of-the-art method for English NER, which is based on neural network architecture, also works well with English FG-NER. However, for Japanese FG-NER, it does not achieve state-of-the-art performance. We proposed two additional features to the existing method: first, incorporating dictionary information to the model and second, utilizing entity category embeddings. Moreover, we proposed a modification to the architecture of the neural network-based model, that is removing the CNN layer for Japanese FG-NER to alleviate problem concerning a large number of characters in the Japanese alphabet. Experimental results show that, the proposed additional features and network architecture modification improve the performance of Japanese FG-NER by a wide margin.

References

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008*, pages 1247–1250.

Nancy Chinchor. 1992. The Statistical Significance of the MUC-4 Results. In *Proceedings of the 4th Conference on Message Understanding, MUC 1992*, pages 30–50.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics (TACL)*, 4:357–370.

Cicero dos Santos and Victor Guimarães. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of the Fifth Named Entity Workshop, NEWS 2015*, pages 25–33.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-Dependent Fine-Grained Entity Type Tagging. *CoRR*, abs/1412.1820.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016*, pages 260–270.

Xiao Ling and Daniel S. Weld. 2012. Fine-Grained Entity Recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2012*, pages 94–100.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*, pages 1064–1074.

Andrew McCallum and Wei Li. 2003. Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-Enhanced Lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003*, pages 188–191.

Shotaro Misawa, Motoki Taniguchi, Yasuhide Miura, and Tomoko Ohkuma. 2017. Character-based Bidirectional LSTM-CRF with Words and Characters for Japanese Named Entity Recognition. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP, SCLeM 2017*, pages 97–102.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017*, pages 338–348.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named Entity Recognition in Tweets: An Experimental Study. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011*, pages 1524–1534.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003*, pages 142–147.

Satoshi Sekine and Chikashi Nobata. 2004. Definition, Dictionaries and Tagger for Extended Named Entity Hierarchy. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004*, pages 1977–1980.

Satoshi Sekine, Kiyoshi Sudo, and Chikashi Nobata. 2002. Extended Named Entity Hierarchy. In *Proceedings of the Third International Conference on Language Resources and Evaluation, LREC 2002*, pages 1818–1824.

Satoshi Sekine. 2008. Extended Named Entity Ontology with Attribute Information. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation, LREC 2002*, pages 52–57.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007*, pages 697–706.

Masatoshi Suzuki, Koji Matsuda, Satoshi Sekine, Naoaki Okazaki, and Kentaro Inui. 2016. Fine-Grained Named Entity Classification with Wikipedia Article Vectors. In *Proceedings of the 2016 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2016*, pages 483–486.

Koichi Takeuchi and Nigel Collier. 2002. Use of Support Vector Machines in Extended Named Entity Recognition (in Japanese). In *Proceedings of the Sixth Conference on Natural Language Learning, CoNLL 2002*, pages 119–125.

Hiroyasu Yamada, Taku Kudo, and Yuji Matsumoto. 2002. Japanese Named Entity Extraction Using Support Vector Machine. *Transactions of Information Processing Society of Japan (IPSJ)*, 43(1):44–53.

Dani Yogatama, Daniel Gillick, and Nevena Lazic. 2015. Embedding Methods for Fine Grained Entity Type Classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, ACL 2015*, pages 291–296.

Mohamed Amir Yosef, Sandro Bauer, Johannes Hoffart, Marc Spaniol, and Gerhard Weikum. 2012. HYENA: Hierarchical Type Classification for Entity Names. In *Proceedings of the 24th International Conference on Computational Linguistics, COLING 2012*, pages 1361–1370.

Guodong Zhou and Jian Su. 2002. Named Entity Recognition Using an HMM-Based Chunk Tagger. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics, ACL 2002*, pages 473–480.

Does Higher Order LSTM Have Better Accuracy for Segmenting and Labeling Sequence Data?

Yi Zhang, Xu Sun, Shuming Ma, Yang Yang, Xuancheng Ren
MOE Key Lab of Computational Linguistics, School of EECS, Peking University
{zhangyi16, xusun, shumingma, 1200012760, renxc}@pku.edu.cn

Abstract

Existing neural models usually predict the tag of the current token independent of the neighboring tags. The popular LSTM-CRF model considers the tag dependencies between every two consecutive tags. However, it is hard for existing neural models to take longer distance dependencies of tags into consideration. The scalability is mainly limited by the complex model structures and the cost of dynamic programming during training. In our work, we first design a new model called "high order LSTM" to predict multiple tags for the current token which contains not only the current tag but also the previous several tags. We call the number of tags in one prediction as "order". Then we propose a new method called Multi-Order BiLSTM (MO-BiLSTM) which combines low order and high order LSTMs together. MO-BiLSTM keeps the scalability to high order models with a pruning technique. We evaluate MO-BiLSTM on all-phrase chunking and NER datasets. Experiment results show that MO-BiLSTM achieves the state-of-the-art result in chunking and highly competitive results in two NER datasets. [1]

1 Introduction

Chunking and named entity recognition are sequence labeling tasks whose target is to find the correct segments and give them the correct labels. The tags inside a segment have internal dependencies. The tags in consecutive segments may have dependencies, too. Therefore, it is natural to take the tag dependencies into consideration when making a prediction in such sequence labeling tasks.

Recently, methods have been proposed to capture tag dependencies for neural networks. Collobert et al. (2011) proposed a method based on convolutional neural networks, which can use dynamic programming in training and testing stage (like a CRF layer) to capture tag dependencies. Furthermore, Huang et al. (2015) proposed LSTM-CRF by combining LSTM and CRF for structured learning. They use a transition matrix to model the tag dependencies. A similar structure is adopted by Ma and Hovy (2016). Their model also involves an external layer to extract some character level features.

However, it is not explicit how to model the dependencies of more tags or use the dependency information in these lines of work. We then propose a solution to capture long distance tag dependencies and use them for dependency-aware prediction of tags. For clarity, we first give some detailed explanations of the related terms in our work. "order" means the number of tags that a prediction involves in a model. An order-2 tag is a bigram which contains the previous tag and the current tag at a certain time step, as shown in Figure 1. Higher order tags are defined in a similar way.

We first develop a simple method to implement high order models. But these models, which are supposed to capture more tag dependency information, perform worse and worse as the order of models increases. One possible reason is that trying to capture more tag dependencies raises the difficulty of prediction. We name these models as single order models and propose a new method based on them. The proposed **M**ulti-**O**rder **LSTM** (MO-LSTM) combines multi-order information from these single order models to decode. It keeps the scalability with a proposed pruning technique and performs well in our

[1] The code is available at https://github.com/lancopku/Multi-Order-LSTM

Proceedings of the 27th International Conference on Computational Linguistics, pages 723–733
Santa Fe, New Mexico, USA, August 20-26, 2018.

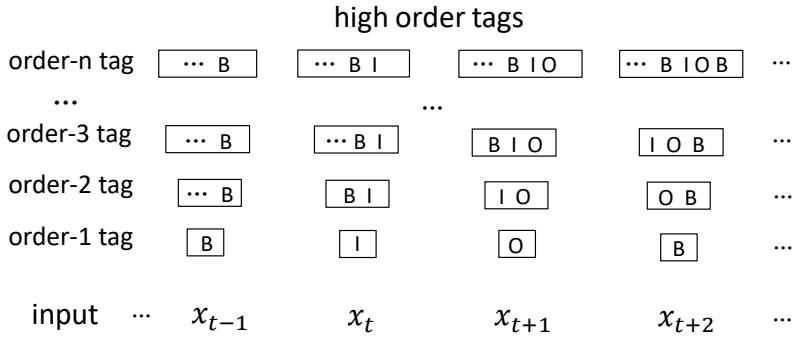


Figure 1: An illustration of tags of different orders.

tasks. Experiments show that MO-LSTM achieves the state-of-the-art F1 score in all-phrase chunking and competitive scores in two NER datasets.

The contributions of this work are as follows:

- We extend the LSTM model to higher order models. However, the performance of the high order models which are supposed to capture longer tag dependencies is getting worse when increasing the order.
- We propose a model integrating low order and high order models. It keeps the scalability in both training and testing stage with a pruning technique.
- The proposed MO-LSTM achieves an evident error reduction in chunking and NER tasks. It produces the state-of-the-art F1 score in chunking and highly competitive results in two NER datasets.

2 Single Order LSTM

We first propose a simple training and decoding method which enables the existing models to extend to higher order models. Take the order-2 model as an example, for each word we combine its previous tag and its current tag to produce a bigram tag as its new tag to predict. Hence, the model can be trained with the "new" bigram (order-2) tag set.

Formally, given an input sequence $\boldsymbol{x} = \{x_1, x_2, \cdots, x_T\}$, where x_t denotes the t-th word in a sentence and T denotes the sentence length. The sequence $\boldsymbol{y} = \{y_1, y_2, \cdots, y_T\}$ represents a possible label sequence for $\boldsymbol{x}$. We denote $\mathcal{Y}^{(1)}$ as the set of all possible order-1 labels, and $y_t \in \mathcal{Y}^{(1)}$. The order-1 model can be represented as:

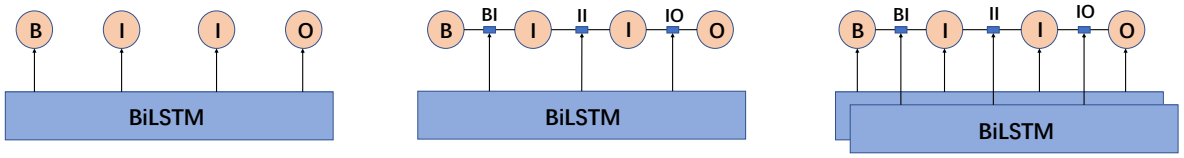
$$s_1(y_1, y_2, \cdots, y_T|\boldsymbol{x}; \theta) = \prod_{t=1}^{T} s(y_t|\boldsymbol{x}; \theta) \tag{1}$$

where θ is the parameters of the model. In implementation, we use a Bi-LSTM with a softmax layer to compute the score $s(y_t|\boldsymbol{x}; \theta)$.

To extend the order-1 model to an order-2 model, we transform the unigram label sequence into a bigram label sequence $y_0y_1, y_1y_2, \cdots, y_{T-1}y_T$, where y_0 is a special START symbol. The bigram label is defined as a combination of two consecutive label y_{t-1} and y_t, and $\mathcal{Y}^{(2)}$ is the set of all possible bigram labels that appear in the training set. The order-2 model can then be written as:

$$s_2(y_1, y_2, \cdots, y_T|\boldsymbol{x}; \theta) = \prod_{t=1}^{T} s(y_{t-1}y_t|\boldsymbol{x}; \theta) \tag{2}$$

Similar to the order-1 model, the score $s(y_{t-1}y_t|\boldsymbol{x}; \theta)$ is computed by a Bi-LSTM with a softmax layer. In implementation, the difference with the order-1 model is that the unigram label is replaced with the



(a) Single Order-1 Model (b) Single Order-2 Model (c) Multi-Order-2 Model

Figure 2: An illustration of the single order model and the multi-order model. The single order-1 model is a BiLSTM.

bigram label. In this way, the model can be further extended to order-n:

$$s_n(y_1, y_2, \cdots, y_T|\boldsymbol{x}; \theta) = \prod_{t=1}^{T} s(y_{t-n+1} \cdots y_t|\boldsymbol{x}; \theta) \tag{3}$$

As the order of the models increases, the models are supposed to learn more tag dependencies. However, according to our experiments, the performance of these models is getting worse, and the detailed results are shown in Section 4. An intuitive reason to explain the experimental phenomena is that the increasing size of the label set makes it more difficult to predict a correct label of the input word. Another potential reason is that the complex structure leads to overfitting problem. Sun (2014) suggests that complex structures are actually harmful to the generalization ability in structured prediction.

3 Multi-Order BiLSTM

The performance of single high order models deteriorates as the order increases. But they might capture some kinds of useful dependency information. To make use of these dependency information, we introduce a multi-order model which combines the low-order and high-order information. The proposed multi-order model consists of several single order models (as described in Section 2) of different orders. At the training stage, these models are trained separately as usual. At the decoding stage, we propose a new decoding method to combine the low order model and the high order model. Since both low order information and high order information is used when decoding, the proposed method is named Multi-Order BiLSTM (MO-BiLSTM). In this section, we first give the details of the training and the decoding process, and then introduce a pruning technique to improve the efficiency of MO-BiLSTM.

3.1 Multi-Order Training

Our proposed multi-order-n model is a mixture of k single order models with different orders, where n is the maximum order of the single order models. When $n = 1$, the multi-order model becomes a single order-1 model, i.e. a BiLSTM. The order set of the single order models is the subset of $\{1, 2, \cdots, n\}$. For example, if the maximum order n is 3, the combination of the single order models can be $[1, 2]$, $[1, 3]$, $[2, 3]$, or $[1, 2, 3]$. Formally, we denote the order set as $\{o_1, o_2, \cdots, o_k\}$, where $o_i < o_j$ and $i < j$. In our implementation, n is equal to k in both training and decoding stage.

At the training stage, we train k single order models separately following Eq. 3:

$$\theta_i = \underset{\theta}{\operatorname{argmax}}\, s_{o_i}(y_1, y_2, \cdots, y_T|\boldsymbol{x}; \theta) = \underset{\theta}{\operatorname{argmax}} \prod_{t=1}^{T} s(y_{t-o_i+1} \cdots y_t|\boldsymbol{x}; \theta) \tag{4}$$

where θ_i is the parameters of the i-th single order model of the order o_i. After training, we obtain a set of k independent models: $\{s(y_{t-o_1+1} \cdots y_t|\boldsymbol{x}; \theta_1), \cdots, s(y_{t-o_i+1} \cdots y_t|\boldsymbol{x}; \theta_i)\}$, which learns the label dependency of different orders.

3.2 Multi-Order Decoding

For the purpose of simplicity and clarity, we first describe the proposed decoding method of MO-BiLSTM in the order-2 case, and then we extend it to the general order-n case.

Algorithm 1 Multi-order decoding with pruning in the order-n case

Input: sentence $\boldsymbol{x}$, trained order-1 LSTM $s_1(y|x)$ in Eq. 1, multi-order-n LSTM $s_n(y|x)$ in Eq. 6
for $t = 1...T$ **do**
  Select the top-k uni-labels by the order-1 scores:
    $\tilde{Y_1} = \text{topkTag}(s_1(y_t|x)), \tilde{Y_2} = \text{topkTag}(s_1(y_{t-1}|x)), \cdots, \tilde{Y_n} = \text{topkTag}(s_1(y_{t-n+1}|x))$
  Combine n top-k uni-label sets into a n-gram label set:
    $Y = \tilde{Y_1} \times \tilde{Y_2} \times \cdots \times \tilde{Y_n}$
  for each $(y^1, y^2, \cdots, y^n) \in Y$ **do**
    Previous tag state $d_{t-1} = y^1 y^2 \cdots y^{n-1}$
    Current tag state $d_t = y^2 y^3 \cdots y^n$
    Compute the transition score $s = s_n(y^1, y^2, \cdots, y^n|x)$ by multi-order-n LSTM
    Compute the maximum score at current state $A[t][d_t] = max(A[t][d_t], A[t-1][d_{t-1}] * s)$
Output: The optimal tag sequence y^* by backtracking the path of the maximum score $A[T][d_T]$

As shown in Figure 2, in the order-2 case the multi-order model is a mixture of 2 single order models, i.e. single order-1 model (Eq. 1) and single order-2 model (Eq. 2). At the decoding stage, the multi-order model takes account of both the order-1 model and the order-2 model. We need a new decoding approach to unify the decisions of both models. Since the order-1 model and order-2 model predict the label sequence independently, we choose to multiply the scores of order-1 model and order-2 model to get a global score, and use a dynamic programming algorithm to search for the label sequence with the maximum score:

$$\begin{aligned} y_1^*, y_2^*, \cdots, y_T^* &= \underset{\boldsymbol{y}}{\text{argmax}}\, s_1(y_1, y_2, \cdots, y_T|\boldsymbol{x}; \theta_1) \times s_2(y_1, y_2, \cdots, y_T|\boldsymbol{x}; \theta_2) \\ &= \underset{\boldsymbol{y}}{\text{argmax}} \prod_{t=1}^{T} s(y_t|\boldsymbol{x}; \theta_1) \times s(y_{t-1}, y_t|\boldsymbol{x}; \theta_2) \end{aligned} \tag{5}$$

where $s(y_t|\boldsymbol{x}; \theta_1)$ and $s(y_{t-1}, y_t|\boldsymbol{x}; \theta_2)$ are the score predictions of the single order-1 model and the single order-2 model, respectively. The details of the dynamic programming algorithm are shown in Section 3.3.

Further, we extend the order-2 case to a general order-n case. The difference with the order-2 case is that there are k single order models to approximate the scores of the generated label sequence. We approximate the scores by multiplying all the scores of these trained single order models, and then decode the sequence with the maximum score. Formally, it can be written as:

$$\begin{aligned} y_1^*, y_2^*, \cdots, y_T^* &= \underset{\boldsymbol{y}}{\text{argmax}} \prod_{i=1}^{k} s_{o_i}(y_1, y_2, \cdots, y_T|\boldsymbol{x}; \theta_i) \\ &= \underset{\boldsymbol{y}}{\text{argmax}} \prod_{i=1}^{k} \prod_{t=1}^{T} s(y_{t-o_i+1} \cdots y_t|\boldsymbol{x}; \theta_i) \end{aligned} \tag{6}$$

where $s(y_{t-o_i+1} \cdots y_t|\boldsymbol{x}; \theta_i)$ is the score prediction of the i-th single order model of the order o_i.

3.3 Scalable Decoding with Pruning

Here, we introduce an efficient dynamic programming algorithm to search for the label sequence with the maximum score. The scores of different n-gram labels are jointly considered in our model. Originally, we should consider all possible n-gram labels at every position of the sentence during dynamic programming. However, it will lead to a huge search space and a lot of time. In order to reduce the time cost, we can prune the unnecessary searching branches. For example, an order-1 model assigns a very low probability to the uni-label "I" of the t-th word, which means the order-1 model is confident that the t-th word can hardly be labeled as "I". Therefore, it is unnecessary to take account of the bi-gram labels "I-B", "I-I", and "I-O" at the next time step.

Model	All-Chunking	English-NER	Dutch-NER
Single Order-1 BiLSTM	93.89	88.23	77.20
Single Order-2 BiLSTM	93.71 (-0.18)	87.61 (-0.62)	76.61 (-0.59)
Single Order-3 BiLSTM	93.34 (-0.55)	87.47 (-0.76)	76.47 (-0.73)
Multi-Order-1 BiLSTM	93.89	88.23	77.20
Multi-Order-2 BiLSTM	94.93 (+1.04)	90.23 (+2.00)	80.95 (+3.75)
Multi-Order-3 BiLSTM	**95.01 (+1.12)**	**90.70 (+2.47)**	**81.76 (+4.56)**

Table 1: Results of single order models and MO-BiLSTM. The number in parentheses means the improvements or reductions compared to the results of order-1 models. All-Chunking denotes All-Phrase-Chunking.

In implementation, we use the order-1 labels with high scores to evaluate whether to prune the high order labels. More precisely, we simply keep the top-k order-1 labels at each position. The order-n labels for a specific position is generated by the top-k labels of n tokens around the position. Suppose a task has totally 50 labels. The order-1 model should compute 50 scores of these labels at each time step. As for the order-3 model, the number of the scores to be computed becomes 50^3. The original search space before pruning for dynamic programming at each time step is 50^3. But if we only keep top-5 order-1 labels at each position and prune the order-n labels, the search space will be reduced from 50^3 to 5^3.

According to our experiments, the pruning technique saves a lot of time in the decoding stage and results in no loss of accuracy, and we find top-5 pruning works the best in order to balance the accuracy and the time cost. Details of the experiments can be found in Section 4. Algorithm 1 shows the detailed process of multi-order decoding with pruning in the order-n case.

4 Experiments

4.1 Datasets

Chunking and named entity recognition are sequence labeling tasks that are sensitive to tag dependencies. The tags inside a segment have internal dependencies. The tags in consecutive segments may have dependencies, too. Thus, we conduct experiments on the chunking and NER tasks to evaluate the proposed method. The test metric is F1-score. The chunking data is from CoNLL-2000 shared task (Sang and Buchholz, 2000), where we need to identify constituent parts of sentences (nouns, verbs, adjectives, etc.). To distinguish it from NP-chunking, it is referred to as the all-phrase chunking. We use the English NER data from the CoNLL-2003 shared task (Sang and Meulder, 2003). There are four types of entities to be recognized: PERSON, LOCATION, ORGANIZATION, and MISC. The other NER dataset is the Dutch-NER dataset from the shared task of CoNLL-2002. The types of entities are the same as the English NER dataset.

4.2 Experimental Details

Our model uses a single layer for the forward and backward LSTMs whose dimensions are set to 200. We use the Adam learning method (Kingma and Ba, 2014) with the default hyper parameters. We set the dropout (Srivastava et al., 2014) rate to 0.5.

Following previous work (Huang et al., 2015), we extract some spelling features and context features. We did not use extra resources, with the exception of using Senna embeddings[2] in Chunking and English-NER tasks. The embeddings in Dutch-NER tasks are randomly initialized with a size of 50. The code is implemented with the python package *Tensorflow* (Abadi et al., 2016).

4.3 Effect of Multi-Order Setting

For simplicity, the single order model of order-n is denoted as single order-n model and the multi-order model in the order-n case is denoted as multi-order-n model. To verify the effectiveness of MO-BiLSTM,

[2]Downloaded from http://ronan.collobert.com/senna/

Model	All-Chunking	English-NER	Dutch-NER
Order-1	14	10	11
Order-2	154	39	44
Order-3	832	138	158

Table 2: The sizes of tag set of different order.

we conduct comparison experiments of single order models and multi-order models. The results are shown in Table 1. The performance of single order BiLSTM models is getting worse with the growing of the order. An intuitive reason is that the increasing size of tag set raises the difficulty to make a correct tag prediction of a word. Although the performance of single high order models is far from satisfactory, the multi-order models perform well with consistent growth of F1-score on three datasets. In chunking, the MO-BiLSTM at order-3 obtains a 18.3% error reduction compared to BiLSTM. It also performs well in the NER tasks, resulting in a 21.6% and a 20.0% error reductions in English-NER and Dutch-NER compared to BiLSTM baselines, respectively.

The results suggest that high order dependency information is indeed beneficial to the prediction. Furthermore, the adopted multi-order setting makes the learned tag dependency specific to the input words. The reason is that the proposed high order model encodes the tag dependency into a single "output tag", and model the "output tag" relations using a BiLSTM conditioned on the input words. The tag dependency in previous work is represented by a transition matrix, which cannot capture the relations of tag dependencies with respect to the input words. Moreover, MO-BiLSTM can take advantage of the subtle tag dependencies captured by single-order models and naturally integrate multi-order information to make tag prediction. The decoding process of MO-BiLSTM finds a global optimum tag sequence, which significantly reduces the risk of mistakes.

MO-BiLSTM also results in a growing size of tag set. The sizes of tag set from order-1 model to order-3 model are given in Table 2 respectively. The tag size of the model is beyond a hundred at order-3 case. Although the size of tag set grows as the order of model increases, it is acceptable in such sequence labeling problems compared to the vocabulary size in machine translation which can be over millions.

4.4 Effect of Pruning

The effect of pruning on speeding up the decoding is presented in Table 3. As shown, the pruning technique has shown a great ability to save time with no loss of accuracy. We then give a detailed analysis of the pruning technique. Original search process of dynamic programming considers all possible high order dependencies. However, most low-order tags have been assigned very low probabilities by low-order models and they will form almost impossible high-order tags. Thus, we only keep a small subset of all low-order tags, which makes the possible combinations shrink rapidly so that the cost of dynamic programing is greatly reduced. We also find that the pruned search space has no effect on the performance of the models. We suppose it is almost unlikely that the best tag sequence is out of the pruned search space. Hence, the accuracy is kept to the full extent, as shown in our experiments.

Model	All-Chunking		English-NER		Dutch-NER	
	Time (s)	F1	Time (s)	F1	Time (s)	F1
Multi-Order-2 BiLSTM w/o pruning	31.59	94.93	19.23	90.23	26.60	80.95
Multi-Order-2 BiLSTM	13.64	94.93	13.13	90.23	18.42	80.95
Multi-Order-3 BiLSTM w/o pruning	215.21	95.01	51.78	90.70	69.79	81.76
Multi-Order-3 BiLSTM	44.81	95.01	20.43	90.70	28.66	81.76

Table 3: Effect of pruning on speeding up the decoding.

All-Chunking	F1
SVM classifier (Kudo and Matsumoto, 2001)	93.91
Second order CRF (Sha and Pereira, 2003)	94.30
Second order CRF (McDonald et al., 2005)	94.29
Specialized HMM + voting scheme (Shen and Sarkar, 2005)	94.01
Second order CRF (Sun et al., 2008)	94.34
Conv network tagger (senna) (Collobert et al., 2011)	94.32
CRF-ADF (Sun et al., 2014)	94.52
BiLSTM-CRF (Senna) (Huang et al., 2015)	94.46
Edge-based CRF (Ma and Sun, 2016)	94.80
Encoder-decoder-pointer framework(Zhai et al., 2017)	94.72
BiLSTM (our implementation)	93.89
MO-BiLSTM (this work)	**95.01**

Table 4: All-Chunking: Comparison with state-of-the-art models.

English-NER	F1
Combination of HMM, Maxent etc. (Florian et al., 2003)	88.76
Semi-supervised model combination (Ando and Zhang, 2005)	89.31
Conv-CRF (Senna + Gazetteer) (Collobert et al., 2011)	89.59
CRF with Lexicon Infused Embeddings (Passos et al., 2014)	90.90
BiLSTM-CRF (Senna) (Huang et al., 2015)	90.10
BiLSTM-CRF (Lample et al., 2016)	90.94
BiLSTM-CNNs-CRF (Ma and Hovy, 2016)	**91.21**
Iterated Dilated CNNs (Strubell et al., 2017)	90.65
CNN-CNN-LSTM (Shen et al., 2018)	90.89
BiLSTM (our implementation)	88.23
MO-BiLSTM (this work)	90.70

Table 5: English-NER: Comparison with state-of-the-art models.

Dutch-NER	F1
AdaBoost (decision trees) (Carreras et al., 2002)	77.05
Semi-structured resources (Nothman et al., 2013)	78.60
Variant of Seq2Seq (Gillick et al., 2015)	78.08
Character-Level Stacked BiLSTM (Kuru et al., 2016)	79.36
BiLSTM-CRF (Lample et al., 2016)	81.74
Special Decoder + Attention (Martins and Kreutzer, 2017)	80.29
BiLSTM (our implementation)	77.20
MO-BiLSTM (this work)	**81.76**

Table 6: Dutch-NER: Comparison with state-of-the-art models. Gillick et al. (2015) reported a F1-score of 82.84 in their work, but this result is based on multilingual resources.

4.5 Comparison with State-of-the-art Systems

Table 4 shows the results on all-phrase chunking task compared with previous work. We achieve the state-of-the-art performance in all-phrase chunking. Our model outperforms the popular method BiLSTM-CRF (Huang et al., 2015) by a large margin. Shen and Sarkar (2005) also reported a 95.23 F1-score in their paper. However, this result is based on noun phrase chunking (NP-chunking). All phrase chunking task contains much more tags to predict than NP-chunking, so it is more difficult.

GOLD	The ministry updated port conditions and shipping warnings for the Gulf of Mexico (LOC), Caribbean and Pacific Coast
BiLSTM	The ministry updated port conditions and shipping warnings for the Gulf (LOC) of Mexico(LOC) , Caribbean and Pacific Coast
MO-BiLSTM	The ministry updated port conditions and shipping warnings for the Gulf of Mexico (LOC), Caribbean and Pacific Coast.
GOLD	About 200 Burmese students marched briefly from troubled Yangon Institute of Technology (ORG) in northern Rangoon on Friday.
BiLSTM	About 200 Burmese students marched briefly from troubled Yangon (LOC) Institute of Technology (ORG) in northern Rangoon on Friday.
MO-BiLSTM	About 200 Burmese students marched briefly from troubled Yangon Institute of Technology (ORG) in northern Rangoon on Friday.

Table 7: Examples of the predictions of BiLSTM and MO-BiLSTM of order-3.

Table 5 shows the comparison results on the English-NER dataset. Ma and Hovy (2016) reported the best result of English NER. The main architecture of their network is BiLSTM-CRF equipped with a CNN layer to extract character-level representations of words. Our model performs slightly worse than it but outperforms BiLSTM-CRFs reported in other papers (Huang et al., 2015; Lample et al., 2016).

The comparison results on Dutch NER are shown in Table 6. Gillick et al. (2015) keeps the best result of Dutch NER. However, the model is trained on four languages. With the monolingual setting, their model achieves 78.08 on F1 score. Another competitive result is reported in the work of Lample et al. (2016). Their model is a BiLSTM-CRF model with an external LSTM layer to extract character-level representations of words. Our model gets the best score when there is no extra resources.

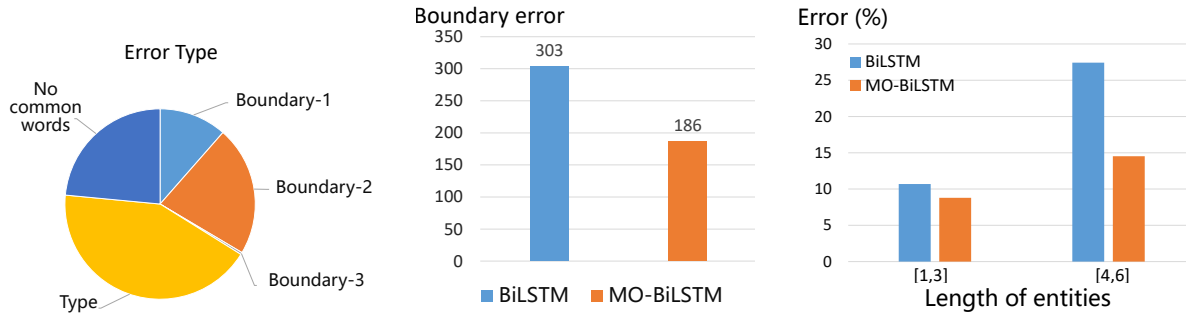
4.6 Case Study

We observe that MO-BiLSTM mainly helps in two aspects: the prediction of boundaries of a segment and the recognition of long segments. Table 7 shows two cases that MO-BiLSTM model predicts correctly but BiLSTM fails to recognize the entities. In the first case, "Gulf of Mexico" should be recognized as the entity "Location". BiLSTM recognizes "Gulf" and "Mexico" as locations, but fails to recognize "of" as a part of the entity, so that an entire entity is split. The reason is that BiLSTM model predicts the tag independently, and it predicts "O" as the tag of "of" regardless of the neighboring tags. On the contrary, MO-BiLSTM takes account of the neighboring tags, and works well in this case. Considering that both the left tag and the right tag are labeled "LOC", the word "of" has a larger probability to be a part of the entity.

The second case contains an entity of type "LOC". BiLSTM succeeds in recognizing the boundary of the entity but predicts a wrong entity type for the word "Yongon". Although "Yangon" is a city, it should not be recognized as a location because it is a part of an organization. BiLSTM does not consider the neighboring tag, and makes a wrong prediction, while MO-BiLSTM succeeds in predicting a correct entity by considering the neighboring tag.

4.7 Error Analysis

To better analyze the basic model and the MO-BiLSTM, we investigate the cases that can not be handled well in English-NER dataset, and the result is summarized in Figure 3. All the unrecognized entities are classified into five categories, which are "boundary-1", "boundary-2", "boundary-3", "type", and "no common words". "Boundary-1" denotes the cases that the gold entity contains a predicted entity, and "boundary-2" means the gold entity is contained by a prediction. "Boundary-3" represents the case that the gold entity and our prediction overlap. "Type" means a entity's boundaries are recognized correctly but its entity type is misclassified. When there are no common words between the predicted entity and any gold entity, it is denoted as "no common words". We count the number of wrongly predicted entities of these different categories, and the result is shown in Figure 3a. The "boundary" error (the



(a) Error types of the predicted entities of MO-BiLSTM.

(b) Number of predicted entities belonging to boundary error.

(c) Percentage of error entities regarding the length of entities.

Figure 3: Error analysis of BiLSTM and MO-BiLSTM on English-NER.

sum of "boundary-1", "boundary-2", and "boundary-3"), which represents the model misidentifies the entity's boundaries, is the major error type of BiLSTM. The reason is that the boundary is made up of two tags, but BiLSTM model predicts each tag independently. Our MO-BiLSTM is able to capture the dependencies between two tags, so it can significantly decrease the number of boundary recognition error. That is also the reason why "boundary" error is not the major error of MO-BiLSTM.

We further compare the number of entities belonging to "boundary" error between BiLSTM and MO-BiLSTM. According to Figure 3b, it shows that the "boundary" error of MO-BiLSTM has a reduction rate of nearly 40% compared with BiLSTM. In order to analyze the influence of the length of entities, we divide the entities into 2 groups according to their lengths, and calculate the recognition error rate of different lengths of entities. The result is shown in Figure 3c. We observe that the MO-BiLSTM model has a significant reduction in the recognition error of long entities from 27.42% to 14.52%. The large reduction in error rate proves that the MO-BiLSTM model is able to capture longer distance tag dependencies compared with BiLSTM.

5 Related Work

Huang et al. (2015) and Lample et al. (2016) stacked a CRF layer on BiLSTM to capture the global tag dependencies. The difference between their work is the way to capture character-level information. Their proposed BiLSTM-CRF performs well in sequence labeling tasks. However, the dynamic programming must be done in both training and testing stage. The MO-BiLSTM does not need dynamic programming during training. Muller et al. (2013) proposed a model that also prunes the tag set using a lower order model, but dynamic programming is required in both training and testing stage like prior work. Besides the difference that we do not need dynamic programing in training stage, the pruning technique is different. We directly model the high order states in the training stage, while Muller et al. (2013) merges lower order states to get higher order states. Soltani and Jiang (2016) propose a model called higher order recurrent neural networks (HORNNs). They proposed to use more memory units to keep track of more preceding RNN states, which are all recurrently fed to the hidden layers as feedback. These structures of Soltani's work are also termed "higher order" models, but the definition is different from ours.

There are several other neural networks that use new techniques to improve sequence labeling. Ling et al. (2015) and Yang et al. (2016) used BiSLTM to compose character embeddings to words representation. Martins and Kreutzer (2017) used an attention mechanism to decide what is the "best" word to focus on next in sequence labeling tasks. Zhai et al. (2017) proposed to separate the segmenting and labeling in chunking. Segmentation is done by a pointer network and a decoder LSTM is used for labeling. Shen et al. (2018) used active learning to strategically choose most useful examples in NER datasets.

6 Conclusions

In this paper, we focus on extending LSTM to higher order models in order to capture more tag dependencies for segmenting and labeling sequence data. We introduce a single order model, which is supposed

to capture more tag dependencies. However, the performance of the single order model is getting worse when increasing the order. To address this problem, we propose to integrate dependency information of different orders to decode. The proposed method, which is called MO-BiLSTM, keeps the scalability to high order models with a pruning technique. Experiments show that MO-BiLSTM achieves better performance than many existing popular methods. It produces the state-of-the-art result in chunking and competitive results in two NER datasets. At the end, we analyze the advantage and limitation of the MO-BiLSTM. We find that MO-BiLSTM mainly helps in the prediction of segment boundaries and the recognition of long segments.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No. 61673028), National High Technology Research and Development Program of China (863 Program, No. 2015AA015404), and the National Thousand Young Talents Program. Xu Sun is the corresponding author of this paper.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Xavier Carreras, Lluis Marquez, and Lluís Padró. 2002. Named entity extraction using adaboost. In *proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–4. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 168–171. Association for Computational Linguistics.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2015. Multilingual language processing from bytes. *CoRR*, abs/1512.00103.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of NAACL'01*, pages 1–8.

Onur Kuru, Ozan Arkan Can, and Deniz Yuret. 2016. Charner: Character-level named entity recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 911–921.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *CoRR*, abs/1603.01360.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernández Astudillo, Silvio Amir, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. *CoRR*, abs/1508.02096.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *CoRR*, abs/1603.01354.

Shuming Ma and Xu Sun. 2016. A new recurrent neural CRF for learning non-linear edge features. *CoRR*, abs/1611.04233.

André FT Martins and Julia Kreutzer. 2017. Learning what's easy: Fully differentiable neural easy-first taggers. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 349–362.

Ryan T. McDonald, Koby Crammer, and Fernando Pereira. 2005. Flexible text segmentation with structured multilabel classification. In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*.

Thomas Muller, Helmut Schmid, and Hinrich Schutze. 2013. Efficient higher-order crfs for morphological tagging. In *EMNLP 2013*.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R Curran. 2013. Learning multilingual named entity recognition from wikipedia. *Artificial Intelligence*, 194:151–175.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *arXiv preprint arXiv:1404.5367*.

Erik Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL'00*, pages 127–132.

E. F. Sang and F. D. Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of CoNLL-2003*, pages 142–147.

Fei Sha and Fernando C. N. Pereira. 2003. Shallow parsing with conditional random fields. In *HLT-NAACL*.

Hong Shen and Anoop Sarkar. 2005. Voting between multiple data representations for text chunking. In *Advances in Artificial Intelligence, 18th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2005, Victoria, Canada, May 9-11, 2005, Proceedings*, pages 389–400.

Yanyao Shen, Hyokun Yun, Zachary C. Lipton, Yakov Kronrod, and Animashree Anandkumar. 2018. Deep active learning for named entity recognition. In *International Conference on Learning Representations*.

Rohollah Soltani and Hui Jiang. 2016. Higher order recurrent neural networks. *CoRR*, abs/1605.00064.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate sequence labeling with iterated dilated convolutions. *arXiv preprint arXiv:1702.02098*.

Xu Sun, Louis-Philippe Morency, Daisuke Okanohara, and Jun'ichi Tsujii. 2008. Modeling latent-dynamic in shallow parsing: A latent conditional model with improved inference. In *Proceedings of COLING'08*, pages 841–848, Manchester, UK.

Xu Sun, Wenjie Li, Houfeng Wang, and Qin Lu. 2014. Feature-frequency-adaptive on-line training for fast and accurate natural language processing. *Computational Linguistics*, 40(3):563–586.

Xu Sun. 2014. Structure regularization for structured prediction. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2402–2410.

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

Feifei Zhai, Saloni Potdar, Bing Xiang, and Bowen Zhou. 2017. Neural models for sequence chunking. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3365–3371.

Ant Colony System for Multi-Document Summarization

Asma Bader Al-Saleh
Department of Computer Science
King Saud University
absaleh@imamu.edu.sa

Mohamed El Bachir Menai
Department of Computer Science
King Saud University
menai@ksu.edu.sa

Abstract

This paper proposes an extractive multi-document summarization approach based on an ant colony system to optimize the information coverage of summary sentences. The implemented system was evaluated on both English and Arabic versions of the corpus of the Text Analysis Conference 2011 MultiLing Pilot by using ROUGE metrics. The evaluation results are promising in comparison to those of the participating systems. Indeed, our system achieved the best scores based on several ROUGE metrics.

1 Introduction

Multi-document summarization (MDS) is a type of summarization in which the contents of a set of documents are represented as a single summary. Today, this type of summarization has become a necessity due to the existence of enormous amounts of information. It reduces the quantity of text by providing a summary that contains the most relevant and important parts. The automatic summarization problem has been studied since the middle of the 20th century. Therefore, the application of several summarization approaches, such as statistical (Radev et al., 2004; Alguliev et al., 2013; Rautray and Balabantaray, 2017) and graph-based (Erkan and Radev, 2004; Shen and Li, 2010; Mosa et al., 2017b) approaches, have been described in the literature.

An extractive summary represents a combination of the most important sentences from the source without modifying them. Summary sentences are selected according to the two following approaches. The first is the greedy selection approach, in which the best textual units (e.g., sentences) are selected one item at a time. This approach is widely used in text summarization and is simple and fast; however, it rarely produces the best summaries (Huang et al., 2010). The second approach is the global optimal selection approach, which searches for the best summary rather than for the best sentences. It reduces the summarization task, or at least the step of selecting sentences, to an optimization problem in which the overall score of the output summary is optimized by searching for the best mixture of sentences. In the literature, several summarization objectives have been studied and optimized, such as information coverage, text diversity, and readability. In addition, different meta-heuristics have been applied in the studies to approximate the solution of the summarization problem. One group of such meta-heuristics is swarm intelligence (SI). SI is a nature-inspired population-based type of meta-heuristics that has been applied successfully to the summarization problem (Alguliev et al., 2013; Peyrard and Eckle-Kohler, 2016). Additionally, ant colony optimization (ACO) algorithms have been successfully applied to short text (Mosa et al., 2017a; Mosa et al., 2017b) and single document summarization (Tefrie and Sohn, 2018).

Motivated by the importance of the summarization task as well as by the promising results of the studies mentioned above, this paper investigates the application of the ACO algorithms to MDS for both the English and Arabic languages. It proposes an extractive MDS algorithm that maximizes the information coverage and saliency and minimizes the redundancy in the resulting summary using ACO. Specifically, it uses an ant colony system (ACS) to search for a good summary that optimizes these

Proceedings of the 27th International Conference on Computational Linguistics, pages 734–744
Santa Fe, New Mexico, USA, August 20-26, 2018.

objectives. The implemented system (called MDS-ACS) has been evaluated on both English and Arabic versions of the corpus of the Text Analysis Conference (TAC) 2011 MultiLing Pilot (hereafter referred to as the 2011 MultiLing Pilot) and using ROUGE metrics (Lin, 2004). The results of the proposed algorithm are promising compared to those of the top-ranked participated systems. The outline of the paper is as follows: Section 2 briefly presents some related studies; Section 3 briefly describes the ACS algorithm; Section 4 presents the formulation of the MDS problem; Section 5 describes the proposed algorithm and its main steps; Section 6 presents the experimental results; and finally, Section 7 presents the main conclusion and considerations for future work.

2 Related Work

Due to space limitations, this section only covers text summarization studies that use SI meta-heuristics. In addition, since the results of the proposed algorithm are compared to the results of the systems that participated in the 2011 MultiLing Pilot, these systems will be briefly presented. During the last decade, SI has accompanied the other meta-heuristics used to solve the text summarization problem. Examples of these meta-heuristics are particle swarm optimization (PSO), which simulates the behavior of a fish school or a bird flock; ACO, which was inspired by the ant society; and the artificial bee colony (ABC) algorithm, which simulates the behavior of honey bees. Recently, new SI meta-heuristics such as cuckoo search (CS) has also been used in the summarization field (see Table 1.)

PSO has been used by several summarization studies. For example, they have been used to assign weights for features extracted from the text to be summarized in Binwahlan et al. (2010) and as a feature selection method for Arabic single document summarization in Al-Zahrani et al. (2015). Aliguliyev (2010) proposed a multi-document method based on sentence clustering, which has been solved using a modified PSO algorithm. PSO has also been used in several summarization studies to perform the actual summary extraction process. Alguliev et al. (2013) proposed an optimization model that uses a discrete PSO algorithm to generate multi-document summaries by maximizing their content coverage and diversity. Asgari et al. (2014) proposed an extractive summarization method based on a multi-agent PSO (Ahmad et al., 2007). Foong and Oxley (2011) proposed an extractive summarization model that combines two kinds of algorithms: PSO and harmony search. PSO has also been included in summarization studies with languages other than English, such as Arabic (Al-Abdallah and Al-Taani, 2017) and Hindi (Dalal and Malik, 2018).

ACO is another SI meta-heuristics that has been used in summarization. Tefrie and Sohn (2018) proposed a summarization model that incorporates several features to calculate the heuristic value of each sentence. There are several differences between Tefrie and Sohn (2018) algorithm and our algorithm. Besides using different extracted features, several aspects related to the ACO are also different, such as the initial pheromone values, the calculated heuristics, the pheromone updating method, and the termination condition. Unfortunately, we could not compare the performance of our algorithm and of their algorithm because the exact values of the results are not provided in their paper. ACO has also been used with short text summarization problems. Mosa et al. (2017a) proposed a technique based on the use of ACO and Jensen-Shannon divergence to summarize a large number of Arabic user-contributed comments. Mosa et al. (2017b) proposed another technique to summarize comments using ACO along with graph coloring and local search to extract the summaries. Peyrard and Eckle-Kohler (2016) used ABC meta-heuristic to create an extractive multi-documents summarizer. They proposed a general optimization framework in which any objective function of input documents and an output summary can be used. Another ABC based summarizer was proposed by Sanchez-Gomez et al. (2017). The main difference between this study and all the previous ones is that the summarization problem is formulated as a multi-objective one. Finally, Rautray and Balabantaray (2017) used the CS for multi-document summarization.

The remainder of this section is devoted to describing the eight systems that participated in the 2011 MultiLing Pilot, in which they were given IDs from 1 to 8. All these systems were applied to both English and Arabic languages except system 5, which was not applied to Arabic. Liu et al. (2011) proposed a solution (ID 1) based on using the hierarchical latent Dirichlet allocation (LDA) topic model along with other traditional features to score the sentences. The CLASSY model (ID 2) (Conroy et al.,

Reference	SI Type	Summarization Type	Language
Binwahlan et al. (2010)	PSO	Single document	English
Al-Zahrani et al. (2015)	PSO	Single document	Arabic
Aliguliyev (2010)	PSO	Multi-document	English
Alguliev et al. (2013)	PSO	Multi-document	English
Asgari et al. (2014)	PSO	Single document	English
Foong and Oxley (2011)	PSO	Single document	English
Al-Abdallah and Al-Taani (2017)	PSO	Single document	Arabic
Dalal and Malik (2018)	PSO	Single document	Hindi
Tefrie and Sohn (2018)	ACO	Single document	English
Mosa et al. (2017a)	ACO	Short text	Arabic
Mosa et al. (2017b)	ACO	Short text	Arabic
Peyrard and Eckle-Kohler (2016)	ABC	Multi-document	English
Sanchez-Gomez et al. (2017)	ABC	Multi-document	English
Rautray and Balabantaray (2017)	CS	Multi-document	English

Table 1: SI meta-heuristics investigated for text summarization.

2011), used a naïve Bayes classifier to give a weight for each term and summary sentences were selected using one of two methods: non-negative matrix factorization and integer programming. Steinberger et al. (2011) proposed a system (ID 3) based on the use of latent semantic analysis (LSA). Hmida and Favre (2011) proposed a summarizer (ID 4) that uses the Maximal Marginal Relevance (MMR) model to select a summary. Varma et al. (2011) proposed a system (ID 5) that uses the hyperspace analogue to language (Lund and Burgess, 1996) model to estimate the probability for each word w that another word w' occurs with w within a window of size K. Based on these probabilities, the system gives a score for each sentence, and the summary is created by selecting the sentences with the highest scores. Saggion (2011) proposed system (ID 6) in which summary sentences are selected based on their similarity to the centroid of the set of documents to be summarized. Das and Srihari (2011) proposed a solution (ID 7) based on the use of LDA. The solution combines several models to solve the summarization problems, including global tag-topic models (i.e., on a corpus- level) and local models (i.e., on a document set level). Finally, El-Haj et al. (2011) proposed a centroid-based summarizer (ID 8) whereby the sentences are ordered and selected based on their similarity to its centroid.

3 Ant Colony System

Ant colony optimization (ACO) is a family of SI meta-heuristics that mimics the collective behavior of real ants. Communication among ants in their colonies is realized by means of pheromone trails laid down while the ants search for food. Because these trails evaporate over time, the shortest path from the colony to the food source attracts more ants because it has a greater amount of pheromone. An example of an ACO method is the ant colony system (ACS) algorithm. Dorigo and Gambardella (1997) proposed this algorithm and applied it to the traveling salesman problems (TSP.) They made three modifications to the ant system (AS) algorithm, which is another example of ACO. The first modification concerns to the state transition rule that balances between the exploration of new paths and the exploitation of old ones. Formally, an ant k moves from city r to city s by following this rule:

$$s = \begin{cases} arg\ max_{u \in J_k(r)}\{[\tau(r,u)].[\eta(r,u)]^{\beta}\} & if \quad q \leq q_0 \ \ (exploitation) \\ S & if \quad q > q_0 \ \ (biased\ exploration), \end{cases} \tag{1}$$

where τ and η represent the pheromone value and the heuristic value, respectively. $J_k(r)$ is a set of cities that can be reached by the ant k. q is a random number that is uniformly distributed over [0,1]. The relative importance of exploration versus exploitation in the algorithm is controlled by the parameter q_0. β is another parameter used to control the relative weight of the pheromone verses the heuristic. S is a

city that is randomly selected according to the following probability distribution:

$$P_k(r,s) = \begin{cases} \frac{[\tau(r,s)].[\eta(r,s)]^\beta}{\sum_{u \in J_k(r)}[\tau(r,u)].[\eta(r,u)]^\beta} & if \quad s \in J_k(r) \\ 0 & otherwise. \end{cases} \tag{2}$$

The second and third modifications concern the pheromone-trail updating process. The second modification adds a local updating rule that is applied to the pheromones of the visited edges while constructing the solutions. The third modification is applied to the global updating rule so only the ant with the best tour is allowed to deposit pheromone.

4 Formulating the MDS Problem

In this step, MDS problem is formulated into an optimization problem and summary sentences are selected in a way that maximizes its overall content coverage score using ACS algorithm. This optimization problem can be formulated as follows. Let D be a set of input documents to be summarized and each of these document is split into sentences. Thus, D can be written as $D = \{s_1, \ldots, s_{|D|}\}$ where $|D|$ is the total number of sentences in D and s_i represents sentence i ($1 \leq k \leq |D|$.) The extractive MDS problem imposes the generation of a sequence of sentences; summary S, with a maximum length L by selecting a number of sentences from D such that the overall information coverage of S is maximized. Formally, it asks to optimize the main objective below:

$$S = max(\sum_{s_k \in D}(c_k.z_k)) \tag{3}$$

$$s.t. \sum_{s_k \in D}(l_k.z_k) \leq L,$$

where c_k and l_k stand for the content coverage score and the length of of sentence k, respectively. The binary variable z_k is equals to 1 if s_k is part of the summary and zero otherwise. The content score of each sentence is based on the weight of the words it contains. However, to maximize the information coverage and saliency as well as to minimize the redundancies, only the weight of the words that have not been covered by other sentences that already selected as a part of S will be considered and the other words are ignored. Thus, even if a word j occurs more than once in S, its weight w_j is added only once to the total content coverage score. Therefore, the overall content coverage score of S can be calculated by summing the weights of words it covers:

$$\sum_{s_k \in D}(c_k.z_k) = \sum_j (b_j.w_j), \tag{4}$$

The binary variable b_j is defined as:

$$b_j = \begin{cases} 1 & if \sum_{s_k \in D}(d_{kj}.z_k) \geq 1 \\ 0 & otherwise. \end{cases} \tag{5}$$

where d_{kj} is a constant that equals 1 if sentence k contains word j and 0 otherwise.

5 The Proposed Algorithm

This study proposes a summarization algorithm that generates extractive MDS summaries where its overall content score is maximized. It starts by preparing the input text using four preprocessing steps. After that, it gives a score for each word. Finally, these scores are used to generate the summary by selecting the sentences that maximize its information coverage score by using an ACS algorithm. The proposed algorithm (denoted as MDS-ACS) is outlined by Algorithm 1.

input: a set of related documents
the maximum summary length
output: a summary
begin
1 Preprocessing
1.1 Segment the text into a set of sentences
1.2 Tokenize the set of sentences
1.3 Remove the stop words
1.4 Replace each word by its stem
2 Scoring of words
2.1 Build the sentence-to-sentence, word-to-word, and sentence-to-word graphs
2.2 Apply the reinforcement algorithm (Wan et al., 2007)
3 Extracting of summary sentences
3.1 Build the graph of the input documents
3.2 Optimize the information coverage of the summary sentences by ACS
end

Algorithm 1: MDS-ACS

5.1 Preprocessing

Four preprocessing steps are applied to the text before conducting the summarization process. First, Stanford CoreNLP[1] (Manning et al., 2014) is used for text segmentation and sentence tokenization. The text segmentation step breaks up the text into sentences while the sentence tokenization step specifies the words in each of these sentences. After that, a stop word elimination step is applied to the text to remove the frequently occurring words that have low semantic weight (Jurafsky and Martin, 2009). A stop word list from the SMART information retrieval system [2] is used for the English text and the general stop-word list provided in El-Khair (2006) is used for the Arabic text. Finally, a stemming step is applied to obtain the stem of each word, using the Porter stemmer[3] and Khoja's stemmer[4] for English and Arabic text, respectively.

5.2 Scoring of words

In this step, the score of each word is computed by following the approach proposed by Wan et al. (2007). This iterative reinforcement approach merges ideas similar to those of two graph-ranking algorithms: PageRank (Brin and Page, 1998) and the HITS (Kleinberg, 1999). It starts by building three graphs. The first one is a bipartite graph that links each word with the sentences in which it appears and its edges are given a score based on the TF-ISF scores and cosine similarity measure. The second graph represents the relationship between each pair of sentences using also the TF-ISF scores and cosine similarity measure while the third one represents the relationship between each pair of words using the longest common substring.

The reinforcement algorithm is then applied to the graphs. Specifically, the score of each word is computed by applying a method similar to that of the HITS algorithm is applied to the first graph and a method similar to that of the PageRank algorithm is applied to the second and third graphs. Formally, each graph is represented by a matrix: W for the first graph, U for the second graph, and V for the third graph. In addition, this approach has two outputs; the score of each word and the score of each sentence. These scores are computed by applying repeatedly the following equations:

$$u^{(n)} = \alpha \widetilde{U}^T u^{(n-1)} + \beta \widehat{W}^T v^{(n-1)} \tag{6}$$

[1] https://stanfordnlp.github.io/CoreNLP/
[2] http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop
[3] https://tartarus.org/martin/PorterStemmer/
[4] http://zeus.cs.pacificu.edu/shereen/research.htm

$$v^{(n)} = \alpha \widetilde{V}^T u^{(n-1)} + \beta \widetilde{W}^T u^{(n-1)}, \tag{7}$$

where v and u are two matrices that hold the scores of the words and the score of the sentences, respectively. $\widetilde{W}$, $\widetilde{U}$, and $\widetilde{V}$ are the normalized versions of W, U, and V, respectively, $\widehat{W}$ is the normalized transposed version of W. $u^{(n)}$ and $v^{(n)}$ are the values of matrix u and matrix v at the iteration n. Finally, $u^{(n-1)}$ and $v^{(n-1)}$ are the values of matrix u and matrix v at the iteration $n-1$. At this point, the score of each word is computed and ready to be used to generate the summary. However, due to the importance of the first sentences in the summarization, the proposed algorithm doubles the weight of the words that exist in the first sentences. Several differences exist between the proposed algorithm and the reinforcement approach. First, the proposed algorithm generates multi-document summaries while the reinforcement approach creates single document summaries. Second, while the reinforcement approach depends on the scores of the sentences to create the summary, the proposed algorithm uses the scores of the words to generate a summary that maximizes information coverage and saliency and minimizes redundancy. Third, the proposed algorithm uses the longest common substring to compute the similarities among the words. Forth, the words of the first sentences are given more weight than the other words. Finally, an ACS algorithm is used to extract summary sentences.

5.3 Extracting of summary sentences

A modified version of the ACS algorithm is used to extract summary sentences (see Algorithm 2.)

```
input : the graph representation of input documents
        the maximum summary length
output: summary sentences
begin
    Initialize the pheromone trails and parameters
    while the maximum number of iterations is not reached do
        Create and position an ant on each node (i.e., sentence)
        Activate the ants
        repeat
            for each active ant do
                Choose the next sentence according to Equation (1) and Equation (2)
                if ant cannot include more sentences then
                    Deactivate the ant
                end
                else
                    Update the ant's current scores of the unvisited sentences
                    Update the current length and score of the ant's partial summary
                end
            end
            for each active ant do
                Apply pheromone local updating rule
            end
        until all ants become inactive;
        Increase the current number of iterations
    end
    Apply pheromone global updating rule using the best solution found in the current iteration
    if a summary with a new higher score is found then
        Update the best-so-far summary
    end
    return the best-so-far summary
end
```

Algorithm 2: ACS

This step begins by building a connected graph from the text to be summarized by adding a node to represent each sentence. Then, a modified version of the ACS algorithm proposed by Dorigo and Gambardella (1997)), adapted to work for MDS instead of TSP, is applied to the graph. In this study, the ACS algorithm starts by creating and placing an ant on each node (i.e., sentence). After each iteration, each ant generates a solution (i.e. a summary) which is a path of nodes that represent the extracted sentences. Regarding the summary length constraint, each ant keeps its own length and stops when it reaches the maximum summary length. Therefore, MDS-ACS assigns a state for each ant; active ant if it can include more sentences to its summary and inactive ant otherwise.

Regarding the maximization of the coverage objective (i.e., minimization of the travel distance), the heuristic value to include a new sentence (i.e., to go to a new node) is the inverse of the content score of this sentence. In other words, the heuristic value to travel from sentence r to sentence u is computed as follows:

$$\eta(r, u) = \frac{1}{c_u} \tag{8}$$

where c_u represents the content score of sentence u. In addition, each ant updates the current scores of its unvisited nodes (i.e., sentences) while constructing its solution based on the words that it has covered so far. Finally, the ACS parameters were set to the same values recommended by Dorigo and Gambardella (1997), except the number of ants, which was set to the total number of sentences in the documents to be summarized.

6 Experiments

The proposed algorithm was implemented in Java programming language and is available online[5]. The evaluations were performed on a machine running Windows 10 with with 12 GB RAM and an Intel(R) Core(TM) i7-6500U CPU 2.5 GHz processor. The corpus of the 2011 MultiLing Pilot was chosen to evaluate MDS-ACS on both English and Arabic languages. The 2011 MultiLing Pilot is a multilingual MDS corpus written in seven languages, including English and Arabic. This pilot asked the participants to test their systems on at least two languages to create multi-document summaries of between 240 and 250 words. In this study, MDS-ACS was applied to the English and Arabic versions of this corpus, each consisting of 10 clusters including 10 documents. The results of the present study were compared to the results of participating systems (eight systems for the English version and seven for the Arabic version, see Table 2) as well as to those of the topline and the baseline summaries. Topline summaries were created using a genetic algorithm with the models (human) summaries, whereas the baseline summaries were created based on the similarity between the text and the cluster centroid.

System ID	Research Group (Participant)	Language
ID1 (Liu et al., 2011)	CIST	English and Arabic
ID2 (Conroy et al., 2011)	CLASSY	English and Arabic
ID3 (Steinberger et al., 2011)	JRC	English and Arabic
ID4 (Hmida and Favre, 2011)	LIF	English and Arabic
ID5 (Varma et al., 2011)	SIEL_IIITH	English
ID6 (Saggion, 2011)	TALN_UPF	English and Arabic
ID7 (Das and Srihari, 2011)	UBSummarizer	English and Arabic
ID8 (El-Haj et al., 2011)	UoEssex	English and Arabic

Table 2: Systems that participated at 2011 MultiLing Pilot.

In this study, ROUGE, specifically the ROUGE-1.5.5 toolkit[6], was used to produce the results. In addition to ROUGE-L, the ROUGE metrics used in the competition (ROUGE-1, ROUGE-2, and ROUGE-SU4) are used in this study for comparison purposes. ROUGE scores are reported in terms of F-measure

[5]https://github.com/asma-b/MDS-ACO

[6]ROUGE-1.5.5 was run with the parameters: -a -2 4 -u -c 95 -r 1000 -n 2 -f A -p 0.5 -t 0

System ID	R-1	R-2	R-SU4	R-L	Improvement of MDS-ACS (%)			
					R-1	R-2	R-SU4	R-L
ID10 (topline)	0.52141	0.25	0.27062	0.46685	-9.10	-29.05	-22.12	-5.72
ID9 (baseline)	0.3791	0.109	0.14728	0.35426	+25.02	+62.73	+43.09	+24.24
MDS-ACS	**0.47397**	0.17737	**0.21075**	**0.440136**	-	-	-	-
ID1	0.40776	0.12247	0.16112	0.38606	+16.24	+44.83	+30.80	+14.01
ID2	0.46062	0.16914	0.20042	0.43446	+2.90	+4.87	+5.15	+1.31
ID3	0.45404	**0.18237**	0.20973	0.42935	+4.39	-2.74	+0.49	+2.51
ID4	0.44691	0.15269	0.192	0.42052	+6.05	+16.17	+9.77	+4.66
ID5	0.42243	0.13985	0.17964	0.39517	+12.20	+26.83	+17.32	+11.38
ID6	0.39617	0.11937	0.16312	0.3659	+19.64	+48.59	+29.20	+20.29
ID7	0.39547	0.09635	0.1448	0.36974	+19.85	+84.09	+45.55	+19.04
ID8	0.38985	0.12219	0.15765	0.36974	+21.58	+45.16	+33.68	+19.04

Table 3: F-measure values of ROUGE -1 (R-1), ROUGE-2 (R-2), ROUGE-SU4 (R-SU4), and ROUGE-L (R-L) for the participating systems, the baseline, the topline, and the proposed algorithm (MDS-ACS) for the English version of the corpus of the 2011 MultiLing Pilot. The highest values among those of participants and MDS-ACS are written in bold.

scores. MDS-ACS scores are reported in terms of mean F-measure of five independent runs. These scores, those of participating systems, baseline, and topline summaries of the English and Arabic versions of the corpus are presented in Tables 3 and 4, respectively. Relative improvements of MDS-ACS over the other systems are also reported in these tables. The relative improvement of MDS-ACS over another system, X, was computed as follows:

$$Relative\ Improvement = \frac{score(MDS - ACS) - score(X)}{score(X)} \times 100 \quad (9)$$

The results of MDS-ACS were promising. When tested on the English version of the corpus, MDS-ACS outperformed all the eight systems based on ROUGE-1, ROUGE-SU4, and ROUGE-L scores. MDS-ACS showed improvements of 2.9%, 0.49%, and 1.31% over the top ranked systems in terms of these metrics, respectively. In addition, MDS-ACS was ranked second among other systems based on ROUGE-2 scores. It outperformed the baseline in terms of ROUGE-1, ROUGE-2, ROUGE-SU4, and ROUGE-L metrics with relative improvements of 25.02%, 62.73%, 43.09%, and 24.24%, respectively. When tested on the Arabic version of the corpus, MDS-ACS outperformed all the participating systems based on ROUGE-1 and ROUGE-L scores. In comparison to the top ranked systems ID3 and ID2, MDS-ACS showed relative improvements of 3.98% and 4.09%, respectively. The former comparison used ROUGE-1 and the latter ROUGE-L. MDS-ACS was ranked second among other systems based on ROUGE-2 scores and third based on ROUGE-SU4 scores. MDS-ACS outperformed baseline summaries based on all four ROUGE metrics used in this study. The relative improvement of MDS-ACS over the baseline was 35% (ROUGE-1), 26.56% (ROUGE-2), 33.12% (ROUGE-SU4), and 34.04% (ROUGE-L).

Paired t-tests (p-value = 0.05) were conducted to check whether performance differences between MDS-ACS and the other systems were statistically significant. The results showed that on the English version of the corpus, MDS-ACS significantly outperformed the systems ID1, ID6, ID7, and ID8 as well as the baseline in terms of all metrics used in this study. MDS-ACS outperformed the ID5 system according to ROUGE-1, ROUGE-2, and ROUGE-L metrics. However, there was no significant difference between MDS-ACS and ID5 according to ROUGE-SU4. In addition, MDS-ACS was significantly outperformed by the topline system (ID10), and there were no statistically significant differences between MDS-ACS and the ID2, ID3, and ID4 systems. Regarding the Arabic version, t-tests showed that the only significant difference was between MDS-ACS and ID9 (the baseline) in terms of ROUGE-L. This may be because ROUGE 1.5.5 has not been adapted to the Arabic language. Overall, these experiments showed that adding more weight to words occurring in the first sentences of input documents signifi-

System ID	R-1	R-2	R-SU4	R-L	Improvement of MDS-ACS (%)			
					R-1	R-2	R-SU4	R-L
ID10 (topline)	0.30786	0.14922	0.15489	0.2695	+1.28	-19.45	-16.3	+5.42
ID9 (baseline)	0.23097	0.09497	0.0974	0.21196	+35	+26.56	+33.12	+34.04
MDS-ACS	**0.3118**	0.12019	0.129646	**0.284108**	-	-	-	-
ID1	0.2319	0.0889	0.09871	0.21956	+34.45	+35.2	+31.34	+29.40
ID2	0.29188	0.10347	0.13309	0.27295	+6.82	+16.16	-2.59	+4.09
ID3	0.29987	**0.1278**	**0.1514**	0.2725	+3.98	-5.95	-14.37	+4.26
ID4	0.26279	0.08634	0.1071	0.23853	+18.65	+39.21	+21.05	+19.11
ID6	0.2763	0.10629	0.12456	0.23801	+12.85	+13.08	+4.08	+19.37
ID7	0.22376	0.08577	0.09874	0.21379	+39.35	+40.13	+31.3	+32.89
ID8	0.26786	0.09653	0.11487	0.24793	+16.4	+24.51	+12.86	+14.59

Table 4: F-measure values of ROUGE -1 (R-1), ROUGE-2 (R-2), ROUGE-SU4 (R-SU4), and ROUGE-L (R-L) for the participating systems, the baseline, the topline, and the proposed algorithm (MDS-ACS) for the Arabic version of the corpus of the 2011 MultiLing Pilot. The highest values among those of participants and MDS-ACS are written in bold.

cantly improved the performance of MDS-ACS. We think that these results could be enhanced by adding other semantic and language-dependent features.

7 Conclusion and Future Work

This study proposed a generic extractive MDS approach based on ACO. The original ACS algorithm was adapted to search for the sentences maximizing the information coverage of the summary generated. The implemented system, MDS-ACS, was evaluated using the corpus of the 2011 MultiLing Pilot (English and Arabic versions) based on four ROUGE metrics. The results show that the performance of MDS-ACS was comparable to the performance of the best participating systems. It outperformed all participating systems based on ROUGE-1, ROUGE-SU4, and ROUGE-L for the English version and ROUGE-1 and ROUGE-L for the Arabic version. As a future work, we plan to study the influence of other semantic features on the performance of MDS-ACS. We also intend to explore other SI meta-heuristics for maximizing the information coverage in the generated summaries.

References

R. Ahmad, Yung-Chuan Lee, S. Rahimi, and B. Gupta. 2007. A multi-agent based approach for particle swarm optimization. In *International Conference on Integration of Knowledge Intensive Multi-Agent Systems, 2007. KIMAS 2007*, pages 267–271, April.

Raed Z. Al-Abdallah and Ahmad T. Al-Taani. 2017. Arabic single-document text summarization using particle swarm optimization algorithm. *Procedia Computer Science*, 117:30 – 37.

Ahmed M. Al-Zahrani, Hassan Mathkour, and Hassan Abdalla. 2015. Pso-based feature selection for arabic text summarization. *Journal of Universal Computer Science*, 21(11):1454–1469, nov.

Rasim M. Alguliev, Ramiz M. Aliguliyev, and Nijat R. Isazade. 2013. Formulation of document summarization as a 0-1 nonlinear programming problem. *Computers & Industrial Engineering*, 64(1):94 – 102.

Ramiz M. Aliguliyev. 2010. Clustering techniques and discrete particle swarm optimization algorithm for multi-document summarization. *Computational Intelligence*, 26(4):420–448.

H. Asgari, B. Masoumi, and O.S. Sheijani. 2014. Automatic text summarization based on multi-agent particle swarm optimization. In *Intelligent Systems (ICIS), 2014 Iranian Conference on*, pages 1–5, Feb.

Mohammed Salem Binwahlan, Naomie Salim, and Ladda Suanmali. 2010. Fuzzy swarm diversity hybrid model for text summarization. *Information Processing & Management*, 46(5):571–588, September.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, April.

John M Conroy, Judith D Schlesinger, Jeff Kubina, Peter A Rankel, and Dianne P OLeary. 2011. CLASSY 2011 at TAC: Guided and multi-lingual summaries and evaluation metrics. In *Proceedings of Text Analysis Conference 2011 (TAC 2011)*, Gaithersburg, Maryland, USA, November.

Vipul Dalal and Latesh Malik, 2018. *Semantic Graph Based Automatic Text Summarization for Hindi Documents Using Particle Swarm Optimization*, pages 284–289. Springer International Publishing, Cham.

Pradipto Das and Rohini Srihari. 2011. Global and local models for multi-document summarization. In *Text analysis conference (TAC) (2011), MultiLing summarisation pilot*, Maryland, USA. TAC.

M. Dorigo and L. M. Gambardella. 1997. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, Apr.

Mahmoud El-Haj, Udo Kruschwitz, and Chris Fox. 2011. University of Essex at the TAC 2011 multilingual summarisation pilot. Maryland, USA. TAC.

Ibrahim Abu El-Khair. 2006. Effects of stop words elimination for arabic information retrieval: A comparative study. *International Journal of Computing & Information Sciences*, 4(3):119 – 133.

Günes Erkan and Dragomir R Radev. 2004. The University of Michigan at DUC 2004. In *Proceedings of the 2004 Document Understanding Conference*, Boston, USA.

O. M. Foong and A. Oxley. 2011. A hybrid pso model in extractive text summarizer. In *2011 IEEE Symposium on Computers Informatics*, pages 130–134, March.

Firas Hmida and Benoit Favre. 2011. LIF at TAC multiling: towards a truly language independent summarizer. In *Text analysis conference (TAC) (2011), MultiLing summarisation pilot*, Maryland, USA. TAC.

Lei Huang, Yanxiang He, Furu Wei, and Wenjie Li. 2010. Modeling document summarization as multi-objective optimization. In *Intelligent Information Technology and Security Informatics (IITSI), 2010 Third International Symposium on*, pages 382–386, April.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2nd edition.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July. Association for Computational Linguistics.

Hongyan Liu, Ping'an Liu, Wei Heng, and Lei Li. 2011. The CIST summarization system at TAC 2011. In *Text analysis conference (TAC) (2011), MultiLing summarisation pilot*, Maryland, USA. TAC.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Mohamed Atef Mosa, Alaa Hamouda, and Mahmoud Marei. 2017a. Ant colony heuristic for user-contributed comments summarization. *Knowledge-Based Systems*, 118:105 – 114.

Mohamed Atef Mosa, Alaa Hamouda, and Mahmoud Marei. 2017b. Graph coloring and aco based summarization for social networks. *Expert Systems with Applications*, 74:115 – 126.

Maxime Peyrard and Judith Eckle-Kohler. 2016. A general optimization framework for multi-document summarization using genetic algorithms and swarm intelligence. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 247–257, Osaka, Japan.

Dragomir R. Radev, Hongyan Jing, Malgorzata Styś, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Inf. Process. Manage.*, 40(6):919–938, November.

Rasmita Rautray and Rakesh Chandra Balabantaray. 2017. An evolutionary framework for multi document summarization using cuckoo search approach: Mdscsa. *Applied Computing and Informatics*.

Horacio Saggion. 2011. Using SUMMA for language independent summarization at TAC 2011. In *Text analysis conference (TAC) (2011), MultiLing summarisation pilot*, Maryland, USA. TAC.

Jesus M. Sanchez-Gomez, Miguel A. Vega-Rodrguez, and Carlos J. Prez. 2017. Extractive multi-document text summarization using a multi-objective artificial bee colony optimization approach. *Knowledge-Based Systems*.

Chao Shen and Tao Li. 2010. Multi-document summarization via the minimum dominating set. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 984–992, Stroudsburg, PA, USA. Association for Computational Linguistics.

Josef Steinberger, Mijail Kabadjov, Ralf Steinberger, Hristo Tanev, Marco Turchi, and Vanni Zavarella. 2011. JRC's participation at TAC 2011: Guided and multilingual summarization tasks. In *Proceedings of Text Analysis Conference 2011 (TAC 2011)*, Gaithersburg, Maryland, USA, November.

Kaleab Getaneh Tefrie and Kyung-Ah Sohn, 2018. *Autonomous Text Summarization Using Collective Intelligence Based on Nature-Inspired Algorithm*, pages 455–464. Springer Singapore, Singapore.

Vasudeva Varma, Sudheer Kovelamudi, Jayant Gupta, and Nikhil Priyatam. 2011. IIIT hyderabad in summarization and knowledge base population at TAC 2011. In *Text analysis conference (TAC) (2011), MultiLing summarisation pilot*, Maryland, USA. TAC.

Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. 2007. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 552–559, Prague, Czech Republic, June. Association for Computational Linguistics.

Multi-task dialog act and sentiment recognition on Mastodon

Christophe Cerisara, Somayeh Jafaritazehjani, Adedayo Oluokun and Hoa T. Le
Université de Lorraine, CNRS, LORIA, F-54000 Nancy, France
cerisara@loria.fr, hoa.le@loria.fr

Abstract

Because of license restrictions, it often becomes impossible to strictly reproduce most research results on Twitter data already a few months after the creation of the corpus. This situation worsened gradually as time passes and tweets become inaccessible. This is a critical issue for reproducible and accountable research on social media. We partly solve this challenge by annotating a new Twitter-like corpus from an alternative large social medium with licenses that are compatible with reproducible experiments: Mastodon. We manually annotate both dialogues and sentiments on this corpus, and train a multi-task hierarchical recurrent network on joint sentiment and dialog act recognition. We experimentally demonstrate that transfer learning may be efficiently achieved between both tasks, and further analyze some specific correlations between sentiments and dialogues on social media. Both the annotated corpus and deep network are released with an open-source license.

1 Introduction

Social media are a gold mine for researchers in many domains and especially in natural language processing, because of the endless stream of linguistic content produced every day. However, a major issue faced by every researcher working with Twitter data concerns the accessibility of datasets previously extracted. Indeed, license restrictions limit the possibility to store tweets in a database for a long period of time, and the proportion of tweets that are continuously deleted from the Twitter company servers makes any Twitter corpus quickly obsolete and impossible to retrieve after a few months. This is a very serious issue for the ethics of experimental research, of which reproducibility is a foundational feature. Despite these limitations, many research and development works use Twitter data for a wide variety of applications.

We propose in this work to exploit another social medium with language data, the Mastodon network, which closely resembles Twitter, except for two important differences: Mastodon is a decentralized social network, and it adopts permissive licenses that are compatible with reproducible research. In particular, everyone may create his own Mastodon server with his cohort of registered users, and have his server automatically federated within the worldwide social network. To enable this, the Mastodon software is released with the open-source AGPL license, and the user-generated content in the Mastodon network typically follows a Creative Commons license[1], which allows redistribution of posts.

Mastodon is very similar to Twitter with regard to content and usage, except that the user posts are limited by default to 500 characters, which gives the user the possibility to write longer sentences than on Twitter. From a sociological point of view, Mastodon users are mostly composed of people who are either attracted by the technical challenge of setting up their own server, or who have been disappointed by Twitter, sometimes because of advertisements, changes of policies and privacy threats, or who feel a strong sense of belonging to a minority group, such as LGBT, because Mastodon naturally enables

[1] see, e.g., https://forum.etalab.gouv.fr/tos#3

Proceedings of the 27th International Conference on Computational Linguistics, pages 745–754
Santa Fe, New Mexico, USA, August 20-26, 2018.

and favors the emergence of local communities. Another difference, which is relevant for researchers in Linguistics and Natural Language Processing, is that Mastodon is more international than Twitter, in the sense that the majority of Mastodon servers are located in Japan and in the West of Europe (France, Spain and Germany in particular), and although these servers are federated together, each of them brings together a community, which is often linguistic. It is thus frequent to observe on some server a majority of posts that are not written in English.

In terms of the amount of language data produced every day, Mastodon is still less developed than Twitter, but it is growing and is already large enough (with one and a half million of users as of February 2018) to be useful for most research works. We demonstrate this by scrapping and annotating a corpus of dialogues in English from Mastodon and training and validating a deep learning model on this data. Both the corpus and software described in this work are distributed freely at `https://github.com/cerisara/DialogSentimentMastodon`. We focus in this work on two aspects of Mastodon posts: dialog acts and sentiment recognition, and on the correlations between both of these tasks.

One of the most studied natural language processing task on Twitter is sentiment recognition, which is for example a recurrent task every year at the SEMEVAL evaluation campaign (Nakov et al., 2016; Bethard et al., 2017). We thus have also annotated our Mastodon corpus with sentiments, in order to maximize the potential impact and usage of this new corpus. Conversely, there are fewer works that study dialogues on Twitter. Dialogues on social media have quite a different form than on other media, such as vocal, sms and meetings interactions, because the participants in the dialogue are often unknown in advance, they may come and leave at any time and live in different time zones. The structure of social media dialogues thus typically forms a tree, or a directed acyclic graph if we take into account @-mentions. We focus next on dialog acts, also known as speech acts, which characterize the function of a phrase in the course of a dialog. Typical dialog acts are questions, answers, disagreements...

The scientific hypothesis that is studied in this work is that sentiments are correlated to dialog acts on social media, and that this correlation may be exploited to enable transfer learning between both tasks. Intuitively, consider a dialog where user A writes something *positive* (sentiment) about a given smart phone brand. Then, user B *disagrees* (dialog act) and points out a *negative* aspect of this brand. An obvious correlation between dialog acts and sentiments can be found on this trivial example. Surprisingly, the majority of works about sentiment analysis ignores such relations between dialog acts and sentiments. We propose next to demonstrate experimentally that this correlation is strong enough to enable transfer learning between both tasks, and we further analyze both quantitatively and qualitatively some of the observed correlation patterns.

2 Corpus annotation

We hereafter call *post* the Mastodon equivalent of a tweet, i.e., a single message that is limited to 500 chars maximum. From the user perspective, Mastodon is very similar to Twitter, and we can thus find the same type of content that is found on Twitter. We have crawled about 800,000 posts from the *octodon.social* Mastodon instance, which is one amongst thousands of existing instances, and have filtered out non English posts automatically with the *langdetect* python library. We have then followed the *reply-to* links to structure all posts into dialog trees: dialogues in social media are structured as trees, because anyone can get involved in a dialog from any post that composes the dialog so far. About half of the dialogues have two posts, $1/4$ three, and so on. The longest dialog is composed of 44 posts. The tree-dialogues are then split into a training and test set.

Before annotation, all dialogues are linearized, following the work of Zarisheva and Scheffler (2015), which means that each branch of the tree forms a unique dialog. Two students with a Master degree in linguistics and fluent in English independently assigned two tags to each post in a dialog:

- A sentiment tag with 3 possible values: positive (26% of the corpus), negative (31%) and neutral (43%); The baseline performances, when always classifying posts as neutral, are F1=24.4%.
- A dialog act with 27 possible values (the bold labels on the right of Table 2).

A typical dialog is shown in table 1.

	Sent	DA	Textual content of segment
0	-	I	because this is getting way too much attention it wasn't even that funny URL
1	+	I	LMAO
2	*	O	why you still on twitter though ?
3	-	W	i'm trying to get all my other friends to get on here before i deactivate !
4	-	I	I might have to do that too . Some ain't migrated yet . It's an issue .
5	-	I	some of my mutuals are waiting for more leftists go get on here
6	*	O	and i'm like what are you waiting for ? ? ?
7	-	I	then again i did have to spend hours to figure out how to work this site haha
8	-	Q	lol was it that hard ? ?
9	-	W	i'm not that good with technology or websites
10	-	A	so yeah ..
11	-	A	well yeah , the whole instances thing is kind of confusing tbh .

Table 1: Example of a typical dialog from the Mastodon corpus.

The dialog act tags have been derived from the seminal work on the Switchboard corpus (Core and Allen, 1997) and have been adapted to take into account specificities of social media, taking inspiration from the work of Zarisheva and Scheffler (2015). The recently proposed ISO standard (Bunt et al., 2017) has also been taken into account in the design phase of the annotation guide. This process has lead to the definition of the tags listed in Table 2. After a first round of annotations and in order to avoid rare labels, these 27 tags have been further merged into 15 tags, whose distribution is shown in Table 2.

The inter-annotator agreement is 88.6% for dialog acts and 90.2% for sentiments. The Cohen's kappa coefficient is 85.1% for dialog acts and 90.2% for sentiments.

The final training corpus is composed of 239 dialogues for a total of 1075 posts, and the test corpus of 266 dialogues for a total of 1142 posts. The vocabulary size is 5330 words.

3 Multi-task model

3.1 Model description

The proposed model is shown in Figure 1. It is a two-level hierarchical recurrent network similar to the one proposed in Sordoni et al. (2015):

- The first level (*post level*) takes as input the sequence of word embeddings in a post. This first level is composed of a bi-LSTM, which outputs a single vector per post.
- The second level (*dialog level*) takes as input the sequence of vectors produced at the first level, i.e., one vector per post for every post within a dialog. The second level is composed of a standard RNN, because the length of dialogues rarely exceeds 10 posts and so the LSTM cells do not offer a clear advantage over standard recurrent cells. The output of this RNN at every post is passed to two MLPs, one for dialog acts and another for sentiment labels.

3.2 Training procedure

Two cross-entropy losses are used to train the model, one for dialog act recognition and another for sentiment classification. In our corpus, every post is manually annotated with both a sentiment and dialog act labels. So standard multi-task training simply consists in backpropagating the gradient of both losses with equal weights. However, we also realize some transfer learning experiments with fewer annotations in one of the tasks. In such cases, we artificially remove the gold sentiment label for training and only backpropagate the gradient of the dialog act loss, and vice versa when transferring from the sentiment to the dialog act task.

A development corpus is extracted using 10-fold cross-validation. Three values of the learning rate (0.1, 0.01 and 0.001) have been tried on this development corpus and the best one has been kept. The number of epochs, up to a maximum of 500 epochs, is also tuned on this development corpus for each experiment. Furthermore, because some badly initialized weights may fail to converge, every experiment

- Communicative functions
  - General purpose functions
    - Information seeking functions
      - Question
        - Propositional question
          - Yes/No question ..... **Q** (8.3%)
          - Check question ..... (merged a priori in Q)
        - Set question
          - Wh* / Open question ..... **O** (7.4%)
        - Choice question
          - Open with choices ..... (merged a priori in O)
    - Information providing functions
      - Inform
        - Statement ..... **I** (49.3%)
        - Agreement ..... **A** (7.9%)
        - Disagreement ..... **D** (1.9%)
          - Correction ..... (merged a priori in D)
        - Answer
          - Open + choice answer ..... **W** (9.9%)
          - Confirm answer ..... **Y** (merged in A)
          - Disconfirm answer ..... **N** (merged in D)
    - Commissive functions
      - Offer ..... **E** (1.4%)
        - Promise ..... (merged a priori in E)
    - Directive functions
      - Request ..... **R** (3.3%)
        - Instruct ..... (merged a priori in R)
      - Suggest ..... **S** (3.0%)
    - Directive & commissive functions
      - Accept offer request suggest ..... **P** (merged in A)
      - Decline offer request suggest ..... **L** (merged in D)
  - Feedback functions
    - auto-positive acknowledgement ..... **F** (0.2%)
    - allo-positive acknowledgement ..... (merged a priori in F)
    - auto-negative acknowledgement ..... **B** (merged in F)
    - allo-negative acknowledgement ..... (merged a priori in B)
  - Social obligation functions
    - Initial greetings ..... **H** (2.0%)
    - Return greetings ..... (merged a priori in H)
    - Initial goodbye ..... **G** (merged in H)
    - Return goodbye ..... (merged a priori in G)
    - Apology ..... **X** (merged in M)
    - Accept apology ..... **C** (merged in A)
    - Thanking ..... **T** (2.0%)
    - Accept thanking ..... **K** (merged in A)
- Exclamation ..... **J** (1.5%)
- Explicit performative (*hope you get better*) ..... **V** (1.6%)
- Sympathy (*I'm sorry to hear that*) ..... **M** (0.6%)
- Miscellaneous / other ..... ***** (absent)
- UNK (too ambiguous to decide) ..... **U** (removed)
- Malformed input ..... **Z** (removed)

Table 2: List of dialog acts with the 27 labels used by annotators (letters in bold) and their distribution in the corpus (percentage within parentheses). Some labels have been merged a priori (before the annotation process), in order to simplify the annotation process. Very rare labels have been merged after the annotation process. The remaining 15 labels used to compute the F1 metric are the ones with a distribution percentage. The baseline recognition performance obtained when always answering **I** is F1=35%.

on the development corpus is run twice with different random initializations, and the best one on the development set is kept. Running only twice does not guarantee that at least one training is correct, but it reduces the chances of failure, without increasing too much the computation requirements. All other hyper-parameters have been set a priori to reasonable values: 100-dim LSTM hidden state size, 100-dim word embeddings, 0.4 dropout, ReLU activations.

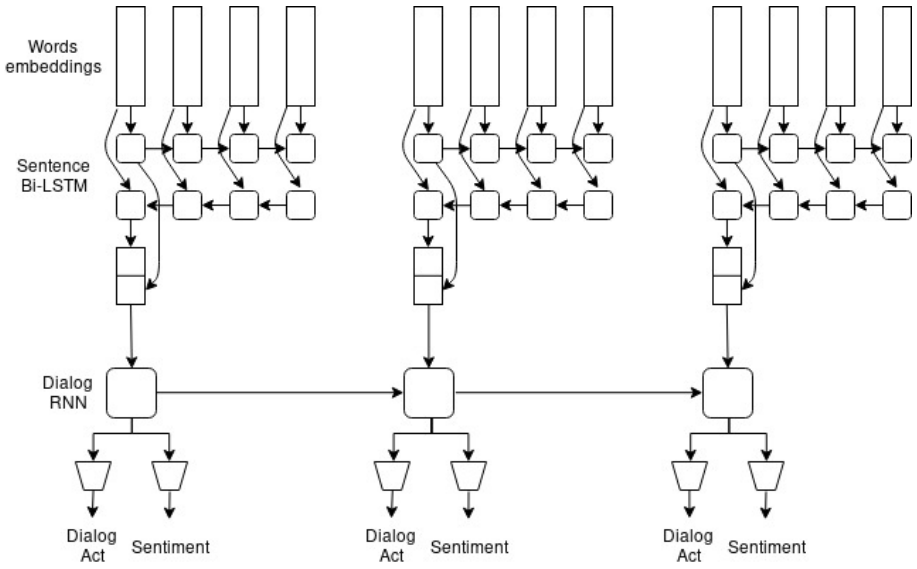


Figure 1: Multi-task hierarchical recurrent model. The number of posts per dialog and the number of words per post may vary and are handled dynamically without any padding.

4 Related work

Pluwak (2016) demonstrates that some expressions of sentiments might not be detected with traditional methods of opinion mining, and that exploiting dialog acts may partly solve this challenge. Nevertheless, very few works have investigated this joint modeling of dialog acts and opinion mining: Clavel and Callejas (2016) study the impact and usage of both dialog act recognition and sentiment analysis in human-agent conversational platforms and affective conversational interfaces, while Novielli and Strapparava (2013) perform a dialog act clustering of a lexicon and study the emotional load of dialog acts. The authors further try to improve dialog act recognition by exploiting affective lexicon, but without convincing results. Conversely, Herzig et al. (2016) include dialog features (e.g., time elapsed between turns...) into an emotion classifier, but do not model the dialog and emotion jointly. Boyer et al. (2011) exploit affects that are visually expressed on faces to better classify dialog acts.

While many works have investigated sentiment analysis on Twitter (Bethard et al., 2017), a few works only handle dialog act recognition on Twitter, such as in (Vosoughi and Roy, 2016), (Ritter et al., 2010), (Forsythand and Martell, 2007) and (Zarisheva and Scheffler, 2015). Each of these works have annotated their own corpus, because, to the best of our knowledge, no large corpus with dialog act annotations on Twitter exist. Zarisheva and Scheffler (2015) describe a detailed and precise procedure for annotating and evaluating dialog acts on German tweets, which we have tried to reproduce as closely as possible for English. Hence, we have manually segmented our own corpus into functional segments - following the ISO definition of a functional segment (Bunt et al., 2017) - and assumed that this gold segmentation is known during training and testing of our models. We have also linearized the dialog trees on Mastodon by splitting and reforming every branch of a dialog tree from the root of the tree as a complete independent dialog. However, because several such dialogues may share some posts at the beginning of dialogues, we have paid attention that our train, dev and test splits never contain the same sub-dialog, and are thus completely independent. We finally took inspiration from the conclusions of (Zarisheva and Scheffler, 2015), where it is shown that dialog act recognition is quite reliable for small taxonomies to design our own taxonomy adapted from the new ISO-standard (Bunt et al., 2017).

A joint model of sentiments and dialog acts is proposed in (Kim and Kim, 2018). It exploits one convolutional network per task, which output vectors are concatenated and passed to one classifier per task. It makes strong simplifying assumptions to merge the tasks, in particular:

- The dialog act at time t does not depend on sentiments, but does depend on the dialog act at time

$t-1$;

- The sentiment at time t only depends on the sentiment and dialog act at time t (no Markov hypothesis);

Despite these strong hypothesis, the authors report better results when considering jointly dialog acts and sentiments on a Korean corpus. As far as we know, their corpus and code is not distributed, which makes comparative evaluations with this model difficult. Compared to this work, our proposed network jointly models both tasks at deeper layers, just after the word embeddings layer. Furthermore, our hypothesis are weaker, because both labels at time t depend on both tasks at time t and $t-1$, and recurrence is applied at both word and dialog levels. Finally, we distribute our corpus and software under an open-source and free license to make future comparison easier.

5 Experimental validation

Model evaluation is performed with metrics used in related works:

- For sentiment recognition, following SemEval 2016 (Nakov et al., 2016), this is the macro-average of the positive and negative F1 scores;
- For dialog act recognition, following (Zarisheva and Scheffler, 2015), this is the average of the dialog-act specific F1 scores weighted by the prevalence of each dialog act.

The model has been written in pytorch. The source code, along with all the data used in this work, is released as open-source and is available at `https://github.com/cerisara/DialogSentimentMastodon`.

5.1 Multi-task experiments

The development corpus is composed of 28 dialogues (average size across all 10 folds). We experiment next with a training corpus of varying size: 1, 10, 50, 100, 150, 200 and the full set of 239 dialogues. The x-axis in Figures 2 and 3 are labeled with the total number of annotated dialogues used at training time (train + dev set). The number of epochs for training is tuned on the development set by optimizing either the sentiment loss (**target sentiment**) or the dialog act loss (**target dialog act**).

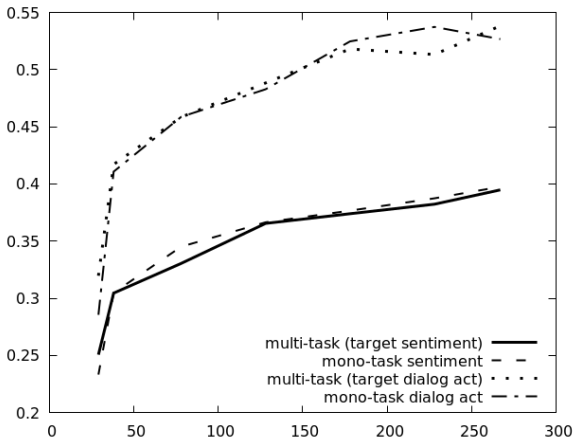


Figure 2: Sentiment and dialog act F1s as a function of the number of dialogues used for training.

In Figure 2, both the mono-task and multi-task models have similar performance. So considering an additional label for the second task does not help, as compared to when only the label of the target task is given. This might be due to the fact that the model already captures all available information with a single task, and the auxiliary label does not bring enough new information to help recognition of the target label.

5.2 Transfer between tasks

We study next unbalanced cases, when the number of annotations differ between tasks. Figure 3 compares the evolution of the sentiment analysis F1 score when:

- Both tasks are trained on the same number of annotated dialogues (**both-rich**); this is the same experiment as realized in Section 5.1.
- The sentiment recognition task is limited to a maximum of 38 (10 dialogues for training plus the development corpus) annotated dialogues, while the training size of the dialog act task is not limited (**sentiment-poor**);
- Both tasks are limited to only 38 annotated dialogues (**both-poor**).

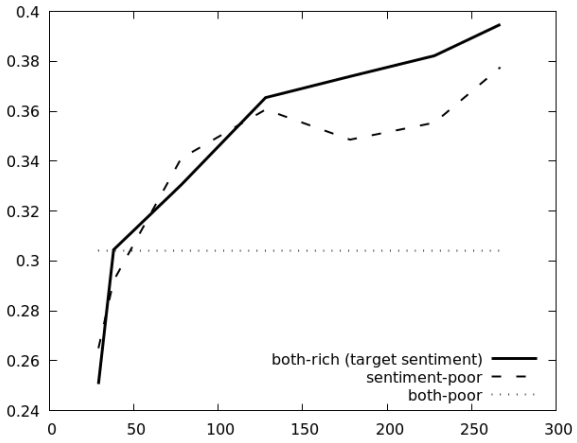


Figure 3: Sentiment F1 as a function of the number of dialogues used for training. **both-rich:** both tasks have the same training size; **sentiment-poor:** only 38 dialogues maximum are annotated with sentiment labels; **both-poor:** both tasks are limited to 38 training dialogues.

The curves show that information is largely transferred from the richer dialog act recognition task to the target sentiment classification task. Hence, although the **sentiment-poor** and **both-poor** systems have only access to 38 dialogues annotated with sentiment labels, the accuracy of the **sentiment-poor** model keeps on increasing when additional dialog act labels are considered. On the right, when the full dialog act training corpus is used, its accuracy is quite close to the one obtained with all sentiment labels, and is better than the **both-poor** model by a large margin, thanks to multi-task transfer learning.

Conversely, we also validate transfer learning from a rich sentiment recognition task to a poor dialog act recognition task. The resulting curves are shown in Figure 4.

Similar gains in performance are obtained through transfer learning for the dialog act recognition task than for the sentiment analysis task. We can thus conclude that transfer learning is efficient between both tasks in both directions.

6 Analysis

Beyond quantitative experiments that demonstrate transfer learning between dialog act and sentiment recognition, we analyze next some properties of our Mastodon corpus.

6.1 Dynamics of sentiments and dialog acts

In the course of a dialog, the sentiment globally changes at a slower rate than the dialog act. While dialog acts change nearly at every segment, often, a single sentiment is expressed per dialog, sometimes two or three, but rarely more. This is understandable, but this also implies that the correlation between both tasks is not very strong, otherwise, they would have a much similar dynamic. On the other hand, they are

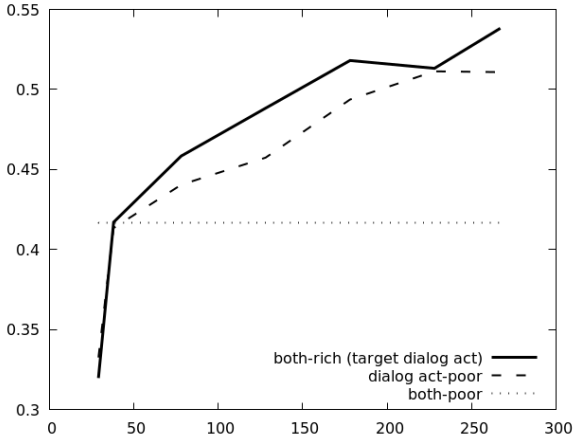


Figure 4: Dialog act F1 as a function of the number of dialogues used for training. **both-rich:** both tasks have the same training size; **dialog act-poor:** only 38 dialogues maximum are annotated with dialog act labels; **both-poor:** both tasks are limited to 38 training dialogues.

not completely independent, as shown in the transfer learning experiments. So we try and exhibit next some patterns where correlation between tasks is explicit in the corpus.

6.2 Both tasks are sparsely correlated

In order to validate our initial hypothesis that, in the course of a dialog, sentiments may be correlated to agreements and disagreements, we have computed the transition log-probabilities between the previous sentiment s_{t-1} and the current sentiment s_t as a function of the current dialog act d_t. These log-probabilities are shown respectively for $s_{t-1} =$ neutral, positive and negative in Figures 5 and 6.

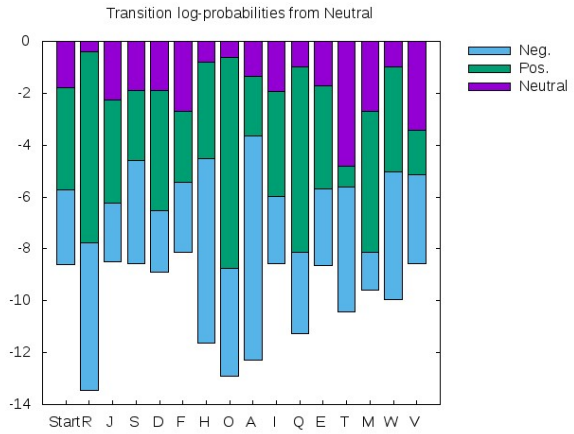


Figure 5: $\log p(s_t|d_t, s_{t-1} = neutral)$. The dialog acts d_t are listed at the bottom of each histogram column. Because log-probabilities are plotted, boxes with the smallest height are the most likely. The start of a dialog, which has no previous sentiment, is a special case that is added on the left.

We observe that:

- With agreements (**A**), a positive sentiment mainly stays positive and a negative sentiment mainly stays negative.
- With disagreements (**D**), a positive sentiment becomes either neutral or negative, a negative sentiment becomes either positive or stays negative, a neutral sentiment either stays neutral or becomes negative.

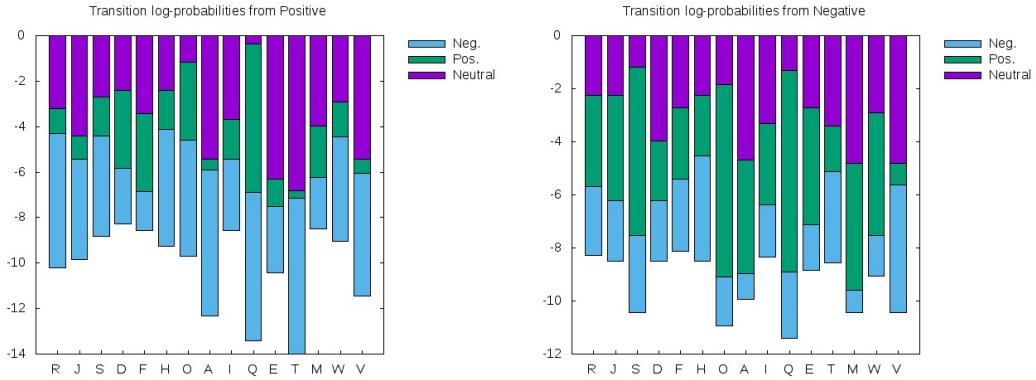


Figure 6: $\log p(s_t|d_t, s_{t-1} = positive)$ (left) and $\log p(s_t|d_t, s_{t-1} = negative)$ (right)

These observations globally follow our intuition, even though more complex patterns than expected seem to occur.

Another observation concerns the evolution of sentiments from the start to the end of a dialog. Previous studies have shown that, with regard to dialogues, the sentiment on social media is related to:

- The topic of discussion: for instance, discussions about politics tend to have more negative sentiments (Thelwall et al., 2012a);
- The length of the dialog: negative sentiments tend to lead to longer dialogues (Thelwall et al., 2012b), which we suggest may be also related to the correlation between sentiments and disagreements that we have observed in this work; indeed, an increased proportion of disagreements may lead to longer dialogues.

Based on our corpus, another pattern seems to emerge, which is in favor of negative sentiments at the beginning of dialogues and positive sentiments at the end. This is intuitively plausible, as at least a small proportion of the sources of discords are likely to be resolved during dialogues, but this interpretation is still to be confirmed on a more diverse set of corpora.

7 Conclusion

We have studied a multi-task model for joint sentiment and dialog act recognition on social media. We have shown that transfer learning is quite efficient when the number of annotated labels for one task is smaller than for the other task. We have further analyzed the correlation between both tasks and shown that although there is enough mutual information to enable transfer learning, both tasks are not strongly correlated globally. They are actually characterized by different dynamics, but we have nevertheless found a few specific patterns that exhibit a strong correlation. All these studies are realized on a new corpus extracted from the Mastodon social network. Conversely to other corpora based on Twitter, this corpus enables reproducible experiments and both the annotated corpus and source code of our model are available with an open-source license on github. Obviously, the privacy issues concerned with extracting corpora from social media are not only related to licenses, and other aspects must be considered, including dissemination, usage and ethical considerations with respect to citizen privacy. Even though the proposed approach does not totally solve all such issues, it initiates an original alternative to the current dominant experimental methodology that hopefully leads to a better experimental reproducibility for natural language researches.

Acknowledgments

The authors thank the “Programme d’Investissements d’Avenir” of the French government, the French National Research Agency (ANR) and the Lorraine Université d’Excellence (LUE) initiative for funding. Experiments presented in this paper were carried out using the Grid’5000 testbed, supported by a

scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see `https://www.grid5000.fr`).

References

Steven Bethard, Marine Carpuat, Marianna Apidianaki, Saif M. Mohammad, Daniel Cer, and David Jurgens, editors. 2017. *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, Vancouver, Canada, August.

Kristy Boyer, Joseph Grafsgaard, Eun Young Ha, Robert Phillips, and James Lester. 2011. An affect-enriched dialogue act classification model for task-oriented dialogue. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1190–1199, Portland, Oregon, USA, June. Association for Computational Linguistics.

Harry Bunt, Volha Petukhova, David Traum, and Jan Alexandersson. 2017. Dialogue act annotation with the iso 24617-2 standard. In *Multimodal Interaction with W3C Standards*, pages 109–135. Springer.

Chlo Clavel and Zoraida Callejas. 2016. Sentiment analysis: from opinion mining to human-agent interaction. *IEEE Transactions on Affective Computing*, pages 74–93.

Mark G Core and James Allen. 1997. Coding dialogs with the damsl annotation scheme. In *AAAI fall symposium on communicative action in humans and machines*, volume 56. Boston, MA.

Eric N Forsythand and Craig H Martell. 2007. Lexical and discourse analysis of online chat dialog. In *Semantic Computing, 2007. ICSC 2007. International Conference on*, pages 19–26. IEEE.

Jonathan Herzig, Guy Feigenblat, Michal Shmueli-Scheuer, David Konopnicki, Anat Rafaeli, Daniel Altman, and David Spivak. 2016. Classifying emotions in customer support dialogues in social media. In *SIGDIAL Conference*, pages 64–73.

Minkyoung Kim and Harksoo Kim. 2018. Integrated neural network model for identifying speech acts, predicators, and sentiments of dialogue utterances. *Pattern Recognition Letters*.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Evaluation measures for the semeval-2016 task 4: Sentiment analysis in twitter (draft: Version 1.12). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*, San Diego, California, June.

Nicole Novielli and Carlo Strapparava. 2013. The role of affect analysis in dialogue act identification. *IEEE Transactions on Affective Computing*, pages 439–451.

Agnieszka Magdalena Pluwak. 2016. Towards application of speech act theory to opinion mining. *Cognitive Studies*, pages 33–44.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180. Association for Computational Linguistics.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562.

M. Thelwall, K. Buckley, and G. Paltoglou. 2012a. Sentiment strength detection for the social web. *Journal of the American Society for Information Science and Technology*, pages 163–173.

M. Thelwall, P. Sud, and F. Vis. 2012b. Commenting on YouTube videos: From Guatemalan rock to El Big Bang. *Journal of the American Society for Information Science and Technology*, pages 616–629.

Soroush Vosoughi and Deb Roy. 2016. Tweet acts: A speech act classifier for twitter. *arXiv preprint arXiv:1605.05156*.

Elina Zarisheva and Tatjana Scheffler. 2015. Dialog act annotation for twitter conversations. In *SIGDIAL Conference*, pages 114–123.

RuSentiment: An Enriched Sentiment Analysis Dataset for Social Media in Russian

Anna Rogers[†], **Alexey Romanov**[†], **Anna Rumshisky**[†],
Svitlana Volkova[‡], **Mikhail Gronas**[§], **Alex Gribov**[†]
[†]Dept. of Computer Science, University of Massachusetts Lowell, Lowell, MA, USA
{arogers,aromanov,arum}@cs.uml.edu,alexander_gribov@student.uml.edu
[‡]Pacific Northwest National Laboratory, Richland, WA, USA
svitlana.volkova@pnnl.gov
[§]Dept. of Russian, Dartmouth College, Hanover, NH, USA
mikhail.gronas@dartmouth.edu

Abstract

This paper presents RuSentiment, a new dataset for sentiment analysis of social media posts in Russian, and a new set of comprehensive annotation guidelines that are extensible to other languages. RuSentiment is currently the largest in its class for Russian, with 31,185 posts annotated with Fleiss' kappa of 0.58 (3 annotations per post). To diversify the dataset, 6,950 posts were pre-selected with an active learning-style strategy. We report baseline classification results, and we also release the best-performing word embeddings trained on 3.2B corpus of Russian social media posts.

1 Introduction

Over the past several years sentiment analysis has been increasingly important in political science (Ceron et al., 2015) and journalism (Jiang et al., 2017). Such applications necessitate resources for languages spoken in the conflict zones. Our study focuses on Russian, which to date has little annotated data (Loukachevitch and Rubtsova, 2016; Koltsova et al., 2016), and no openly available sentiment detection systems beyond the dictionary-based ones. However, lexical features have time and again shown inferior performance compared to the supervised learning approaches using annotated data (Gombar et al., 2017), and lack of such a resource severely limits sentiment analysis applications for Russian.

We present RuSentiment, a dataset of public posts on VKontakte (VK), the largest Russian social network that currently boasts about 100M monthly active users.[1] RuSentiment was developed with new comprehensive guidelines that enabled light and speedy annotation while maintaining consistent coverage of a wide range of explicitly and implicitly expressed sentiment. The overall inter-annotator agreement in terms of Fleiss' kappa stands at 0.58. In total, 31,185 posts were annotated, 21,268 of which were selected randomly (including 2,967 for the test set). 6,950 posts were pre-selected with an active learning-style strategy in order to diversify the data. This makes RuSentiment the largest openly available sentiment dataset for social media, and the largest general domain sentiment dataset for this relatively low-resource language.

We also present baseline classification results on the new dataset. The best results were achieved with a neural network model that made use of word embeddings trained on the VKontakte corpus, which we also release to enable a fair comparison with our baselines in future work. This model achieved an F1 score of 0.728 in a 5-class classification setup. The dataset, the source code for the baseline classifiers, and the in-domain word embeddings are available at the project website[2].

[1] https://vk.com/about

[2] http://text-machine.cs.uml.edu/projects/rusentiment/

Proceedings of the 27th International Conference on Computational Linguistics, pages 755–763
Santa Fe, New Mexico, USA, August 20-26, 2018.

2 Related Work

Russian is generally not as well resourced as English, and that includes the sentiment analysis data. RuSentiLex, the largest sentiment lexicon for Russian[3] (Loukachevitch and Levchik, 2016), currently contains 16,057 words, which exceeds the size of such manually constructed English resources as, for example, SentiStrength (Thelwall and Buckley, 2013). However, there is nothing like SentiWordNet (Baccianella et al., 2010), SentiWords, (Gatti et al., 2016), or SenticNet (Cambria et al., 2018) for Russian.

There are also few annotated datasets. The datasets from the SentiRuEval 2015 and 2016 competitions are the largest resource that has been available to date (Loukachevitch and Rubtsova, 2016). The SentiRuEval 2016 dataset is comprised by 10,890 tweets from the telecom domain and 12,705 from the banking domain. The Linis project (Koltsova et al., 2016) reports to have crowdsourced annotation for 19,831 blog excerpts, but only 3,327 are currently available on the project website.

The choice of VK social network makes RuSentiment qualitatively different from the above resources. Unlike Linis, it contains standalone mini-texts, and unlike SentiRuEval, the postings vary by length and were not pre-selected by topic. We also found the VK data used for our RuSentiment to be more noisy than SentiRuEval tweets.[4] RuSentiment thus fills an existing gap, providing a large annotated dataset of general-domain posts from the largest Russian social network.

3 Annotation

Our VK data was originally collected for research on political bias, and contained the posts from the personal "walls" (i.e., posts on personal pages) of the users that were members of anti-Maidan and pro-Maidan communities during the 2014 Maidan conflict in Ukraine. RuSentiment only includes the posts that were posted outside these communities, and do not contain political keywords.[5] No pre-selection by topic makes RuSentiment currently the largest manually annotated general domain sentiment dataset for Russian, exceeded in size only by automatically annotated silver dataset by Rubtsova (2015).

To remove noisy posts, we used the following selection criteria. The posts included in the dataset were 10-800 characters in length, at least 50% of which were alphabetical, and at least 30% used the Russian Cyrillic alphabet. URLs and VK postcards were excluded. To ensure the meaningfulness of the posts, we also excluded any posts with over 4 hashtags or less than 2 comments. RuSentiment is distributed without VK post ids, and only includes posts that were posted publicly.[6] The annotation was performed by six native speakers with backgrounds in linguistics over the course of 5 months. The average annotation speed was 250-350 posts per hour. A screenshot of our custom web-interface is shown in Figure 1.

3.1 Annotation Policy

Despite the popularity of the sentiment task, the problem of developing comprehensive, yet easy to follow and "light" guidelines that would ensure high enough agreement is far from being solved. Sentiment is an extremely multi-faceted phenomenon, and each research team in the end has to make its own choices about how it would prefer to treat implicit vs. explicit sentiment, subjective feeling and emotion vs. evaluation, and also irony, sarcasm, and other types of mixed sentiment.

We aimed to develop comprehensive guidelines that would cover the most frequent potentially ambiguous cases and would be easy to apply consistently. Most of the categories we have used have been defined and used before (Liu, 2015; Toprak et al., 2010; Wiebe et al., 2005; Thelwall et al., 2010), and our contribution is mainly their combination that enabled the right balance between coverage and ease of application.

[3]There are at least two more projects that attempt to crowdsource sentiment lexicons: SentiBase (`http://web-corpora.net/wsgi/senti\_game.wsgi/rules`), and Sentimeter (`http://sentimeter.ru/assess/instruction/`). At the moment, both appear to be unfinished.

[4]Baldwin et al. (2013) did not find significant grammatical or spelling differences between Twitter, Youtube comments, or blogs, but domain (telecom in SentiRuEval) could impact the ratio of professionally edited commercial or news-like texts.

[5]The list comprised 169 keywords, including political entities (such as *Moscow* or *Putin*) and words coined and used during the Maidan conflict (such as *ukrop* "dill", a Russian derogatory term for Ukrainians).

[6]This work is covered by an IRB protocol at the authors' institution.

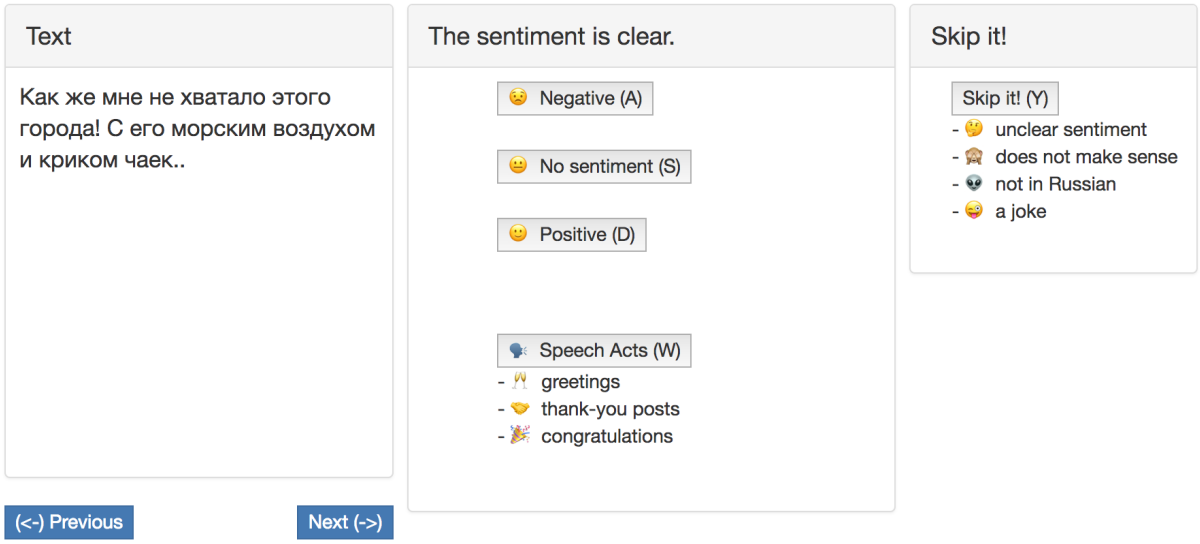


Figure 1: Annotation web-interface.

Finding this point of balance required multiple pilots and extensive linguistic analysis. This difficulty is likely the reason why many sentiment annotation projects, including the Russian crowdsourced initiatives mentioned in Section 2, provide only minimal annotation instructions. Such instructions often yield very inconsistent data. Our guidelines are available at the project website in English, with Russian examples.[7] We hope that they would be useful for subsequent work in other languages and domains.

We prioritized the speed of annotation over detail, opting for a 3-point scale rather than e.g., the 5-point scale in SemEval Twitter datasets (Rosenthal et al., 2017). Thus, the task was to rate the prevailing sentiment in complete posts from VK on a three-point scale ("negative", "neutral", and "positive"). We also defined the "skip" class for excluding the posts that were too noisy, unclear, or not in Russian (e.g., in Ukrainian). We also made the decision to exclude jokes, poems, song lyrics, and other such content that was not generated by the users themselves. It could be argued that posting jokes on social media should be interpreted as an expression of positive mood, but such data is easy to import from existing web collections.

Although the only sentiment classes we annotated are "positive" and "negative", RuSentiment guidelines address both explicit and implicit forms of expressions for the speaker's *internal emotional state* (mood) and *external attitude* (evaluation), as shown in Table 1. These distinctions are often not accounted for in sentiment data, including many of the English datasets (Volkova et al., 2013; Abdul-Mageed and Ungar, 2017). The guidelines cover such cases of implicit sentiment as rhetorical questions, (non-)desirability, recommendations, and descriptions or mentions of the experiences that most people would consider positive or negative.

Additionally, we defined a subcategory of positive posts that covers frequent speech acts, such as expressions of gratitude, greetings, and congratulations. They are very frequent in VK data, and the sentiment they express is overtly positive, but they are also very formulaic. The separate subcategory enables excluding them from the category of positive posts, depending on the practical goals of the analysis. In our binary classification experiments, we chose to exclude this category.

The neutral posts were defined as those that describe something in a matter-of-fact way, without clear sentiment (e.g., *That's a girl I know.*) They also included factual questions, commercial information, plot summaries, descriptions, etc..

We opted to not define a separate "mixed sentiment" class, as this would not be particularly useful, and is also difficult for models to capture (Liu, 2015, p. 77). All cases of mixed sentiment were annotated as either negative or positive. To improve consistency, the guidelines covered 7 frequent cases of mixed sentiment. For example, irony was annotated with the dominant (usually negative) sentiment (e.g. *I*

[7] http://text-machine.cs.uml.edu/projects/rusentiment/

Mode \ Target	Emotional state	Attitude towards some entity
Explicit	• direct statement of one's mood, emotions, feelings - either positive or negative e.g., УРРРРАААААААА!!!!!!)))) *Hooray!!!!!!))))*	• direct statement of one's positive or negative attitude towards the target e.g., годный дабстеп, очень годный. *That's fairly decent dubstep.*
Implicit	• descriptions of experience that most people would consider positive or negative e.g., 15 дней к окончанию сессии... *15 more days of exams...* • rhetorical questions e.g., Вот как я теперь её найду?(((*So how am I supposed to find it now?*	• expressing (non-)desirability of the target, (non-)recommending it e.g., Обязательно попробуй шампусик. *Do taste the champagne.* • rhetorical questions e.g., НАТАШ ТЕБЕ НЕ КАЖЕТСЯ ШО ТЕБЕ ПОРА ЗАНИМАТЬСЯ УЧЕБОЙ? *Natasha, don't you think you should be studying?*

Table 1: Types of positive and negative sentiment covered by RuSentiment annotation guidelines.

broke my heels and got drenched on the way back. What a great day.) Generally neutral posts with some positive formulaic language were annotated as neutral (e.g. *Looking to buy a used electric guitar. Please share this post. Thank you, and have a great day!*) In cases of conflict between a speaker's emotional state and their attitude towards something, we annotated the mood of the speaker. For example, *I miss you* expresses the sadness of the speaker while also implying their high opinion of the addressee, and is annotated as negative.

Hashtags such as *#epicentr* (company name) were considered neutral, but those that could agree or disagree with the general sentiment of the post were treated as sentiment markers. This concerned the hashtags that expressed the sentiment explicitly (e.g. #ihate, #sad) or implicitly, via experiences that most people would consider positive or negative (e.g. #beach, #party).

The annotators were instructed to *not* treat the emoticons as automatic sentiment labels, as done by Go et al. (2009), Davidov et al. (2010), Sahni et al. (2017), and others. Some emoticons do indeed strengthen the message (Derks et al., 2008), but others serve to soften its illocutionary force without changing its content (Ernst and Huschens, 2018).[8] A user may end a post with a "hedging" emoticon just to express friendliness or politeness, and we found that often to be the case for VK data. Therefore, emoticons were not taken into account when no sentiment was expressed verbally or when they aligned with the content of the message. Emoticons were considered relevant only when they contradicted the verbal clues of sentiment. In that case, the annotators were instructed to annotate the dominant overall sentiment of the post, including the emoticons.

In total, five categories were annotated: "Neutral", "Negative", "Positive", "Speech Act", and "Skip".

3.2 Annotating Randomly Selected Posts

In the first stage, 18,453 randomly selected posts were annotated. Fleiss' kappa for three annotators in this sample constituted 0.654. A post was deemed to belong to a class if at least 2 of 3 annotators attributed it to that class. The class distribution is as follows: 41.3% neutral posts, 10.3% negative and 20.5% positive posts, and 9.4% skipped, with an additional 13.9% posts in the speech acts category. In 4.6% cases, all three annotators disagreed, and we included them in the skipped class as unclear. Furthermore, we annotated 2,967 random posts to create a test set (Fleiss' kappa is 0.604).

4 Experiments

4.1 Baseline model selection

We experimented with several classifiers of different types, including logistic regression, linear SVM, and gradient boosting classifier (Pedregosa et al., 2011). We also implemented a simple neural network classifier (NNC) consisting of four fully-connected layers with non-linear activation functions between

[8]This also concurs with classifications drawn in pragmatics and discource analysis, among others, by Kavanagh (2010; Yus and Yus (2014).

(a) Negative posts vs. the rest

(b) Positive posts vs. the rest

Figure 2: Distribution of true labels in probability bins of NNC binary classifier.

them. We used the PyTorch library to implement it.[9] The selected models cover most types of commonly used models, spanning from simple linear models to ensembles and neural networks. The source code for these baselines is available on the project page[10].

The posts were represented either with a sparse TF-IDF representation (Manning et al., 2008) or as an average of word vectors for the constituent tokens. Averaging is by no means the only way to combine word vectors to obtain a representation of a sentence (Mitchell and Lapata, 2010; Baroni et al., 2014; Socher et al., 2012; Hill et al., 2016), but it is one of the computationally cheapest options that still encodes a non-trivial amount of information about the sentences (Adi et al., 2016). FastText (Bojanowski et al., 2017) was chosen for its capacity to represent subword information. This is beneficial for a morphologically rich language such as Russian, especially with a corpus that is as noisy and full of misspellings as our VK corpus.

Model	Feat.	F1	Prec.	Rec.
Logistic Regression	CC	0.526	0.619	0.513
	VK	0.622	0.691	0.611
	Wiki	0.574	0.652	0.559
	TF-IDF	0.626	0.654	0.615
Linear SVM	CC	0.586	0.589	0.610
	VK	0.687	0.690	0.691
	Wiki	0.632	0.628	0.646
	TF-IDF	0.664	0.660	0.670
Gradient Boosting	CC	0.527	0.619	0.509
	VK	0.624	0.692	0.611
	Wiki	0.577	0.646	0.557
	TF-IDF	0.587	0.605	0.588
Neural Net Classifier	CC	0.604	0.603	0.623
	VK	**0.717**	**0.718**	**0.717**
	Wiki	0.661	0.658	0.666
	TF-IDF	0.593	0.599	0.589

Table 2: Classifier performance with different post representations on 5-class classification (20,896 random posts).

We performed experiments using the published FastText embeddings trained on CommonCrawl (CC)[11] and on Russian Wikipedia[12] (Wiki). We also trained our own embeddings on 3.2B tokens of VK data. The training corpus included the posts that are part of RuSentiment, but they constitute less than 0.001% of the data. We trained these embeddings with the default FastText parameters, with vector size 300 and minimum frequency 100. Table 2 shows the performance in 5-way classification, for the models trained using the 20,896 posts annotated without active learning. The best accuracy was achieved by the NNC classifier. The in-domain VK embeddings consistently improved performance of all models, as could be expected.

4.2 Active Learning Data Selection Strategy

Unbalanced datasets present difficulties in classification, especially when the classes of interest – in this case, positive and negative sentiment – are in the minority. With the setup described in Section 4.1

[9] http://pytorch.org
[10] http://text-machine.cs.uml.edu/projects/rusentiment/
[11] https://github.com/facebookresearch/fastText/blob/master/docs/crawl-vectors.md
[12] https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

we conducted preliminary experiments with 13,764 posts for training and 3,441 for testing[13]. In these preliminary experiments, the best-performing model (NNC) reached an F1 of 0.637 for the positive class vs. the rest and F1 of 0.550 for negative vs. the rest in binary classification. The recall was very low, 0.477 and 0.587, respectively). Figure 2 shows that the distribution of correct labels in each probability bin was nearly even, which suggests that the classifier did not have a well-formed representation of the target class. For example, the number of examples correctly assigned to the negative polarity class with probability of 0.9 was nearly equal to the number of misclassified negative polarity examples which were assigned the probability of 0.1. Although there were twice as many training examples, the same trend was present for the positive polarity posts.

To provide the classifier with more examples that it was unsure about, we used NNC to pre-select additional 3,500 "negative" and 2,500 "positive" posts with certainty sampling (Koncz and Paralič, 2013; Fu et al., 2013). We drew an equal number of samples from the probability bins 0.3-0.7, annotating additional 6,950 posts.

4.3 Active Learning Effect

Figure 3 shows that the distribution of positive posts in the pre-selected sample turned out to be similar to the original one. However, the classifier was successful in reducing the number of skipped and speech act posts, and the ratio of negative posts increased.

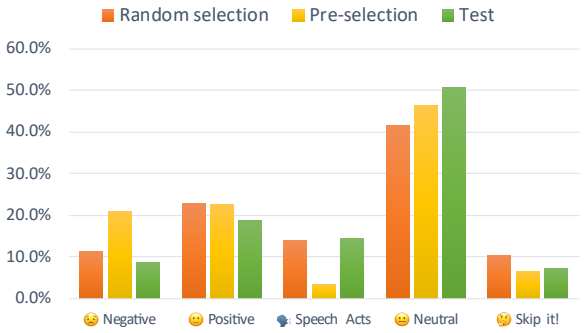


Figure 3: Sentiment class distribution in RuSentiment.

Fleiss' kappa for the pre-selected sample was much lower (0.449), bringing the overall number down. The reason was the higher ratio of posts with agreement of two, rather than three annotators. We interpret this as success in bringing more borderline and diverse cases into the dataset, even at the cost of the overall agreement.

In order to investigate the effects of adding the pre-selected sample to the dataset, we conducted additional 5-way classification experiments using NNC and VK embeddings. Table 3 shows the comparison between the model trained on all the randomly selected examples and the model trained on the data in which 6,950 randomly selected examples were replaced with the ones pre-selected with the active learning strategy described above.[14] Replacing randomly selected examples with pre-selected ones yielded consistent, albeit small, improvement, with the recall being affected the most. Table 3 also shows that the 5-way classification results using the full training data, including the pre-selected sample, produces the best results. The performance is averaged across three runs.

Input data	F1	Prec.	Rec.
Randomly selected posts only (21,268)	0.717	0.718	0.717
6,950 out of 21,268 random posts replaced with pre-selected	0.720	0.722	0.730
21,268 random + 6,950 pre-selected posts	**0.728**	**0.729**	**0.736**

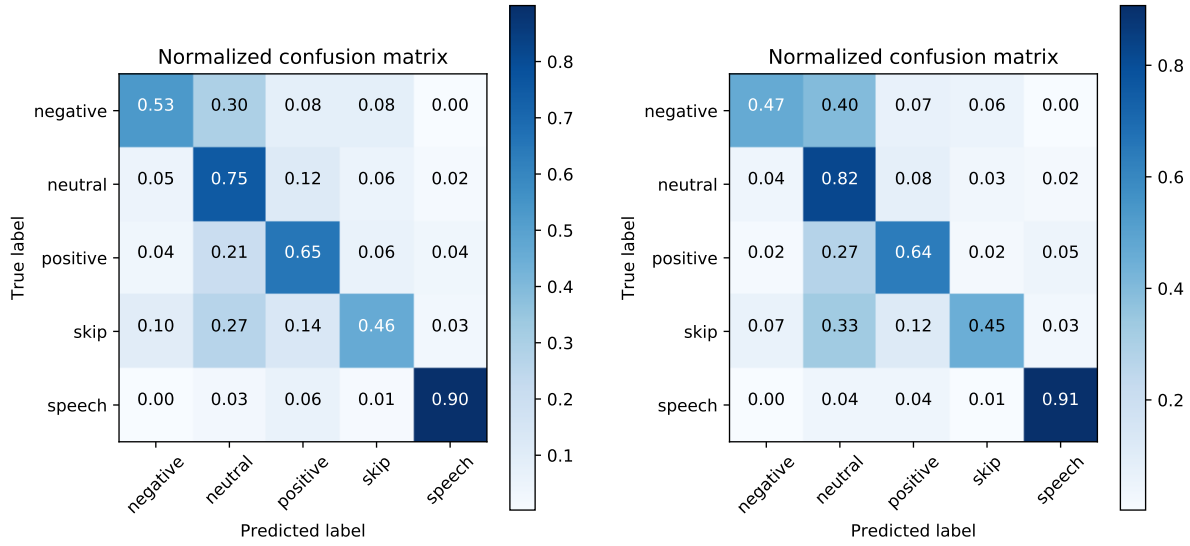
Table 3: 5-class classification with NNC and VK embeddings: the effect of pre-selected posts.

4.4 Error Analysis

Figure 4 presents the confusion matrices for RuSentiment test set with the NNC classifier trained in two settings: (a) using the base randomly selected posts only, and (b) using the additional posts pre-selected with active learning. There is a clear shift in the distribution of neutral and negative posts. Compared to

[13]These samples do not intersect with the final test set.

[14]An additional 463 randomly selected examples were included in this experiment and are released in the final dataset. Out of these, 372 examples were added to the training set and 91 to the test set. These data came in late from the annotation team, and additional experiments confirmed that they did not affect the overall pattern of model performance.



(a) Random posts only

(b) Random and pre-selected posts

Figure 4: Confusion matrix for the RuSentiment test set with different training sets (true labels in rows, predicted labels in columns).

the classifier trained only on randomly selected posts, the classifier trained on the entire dataset makes more errors misclassifying neutral and, to a lesser degree, skipped posts as negative. This suggests that the pre-selected sample makes the classifier more sensitive to borderline cases, since their amount is increased. Interestingly, the positive posts category was not affected in the same way, presumably due to the larger amount of positive posts in the base data.

5 Conclusion

We presented RuSentiment, a new general domain sentiment dataset for Russian social media, built with data from VK, Russia's largest social network. RuSentiment includes 31,185 posts, each carrying 3 annotations with Fleiss' kappa of 0.58, and is currently the largest openly available dataset of its class for Russian. RuSentiment was developed with new guidelines that enabled light, speedy and consistent 5-class annotation of explicit and implicit sentiment, and could be adapted for other languages.

We also presented 4 baseline models using FastText word embeddings as well as TF-IDF representation. The best performance was F1 of 0.728 in 5-class classification. It was achieved by a neural net classifier with in-domain FastText embeddings, which we release to enable fair comparison with future systems.

Acknowledgements

This work was supported in part by the U.S. Army Research Office under Grant No. W911NF-16-1-0174.

References

Muhammad Abdul-Mageed and Lyle Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), volume 1, pages 718–728.

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. arXiv preprint arXiv:1608.04207.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In Language Resources and Evaluation Conference, volume 10, pages 2200–2204.

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. 2013. How noisy social media text, how diffrnt social media sources? In Proceedings of the Sixth International Joint Conference on Natural Language Processing, pages 356–364.

Marco Baroni, Raffaela Bernardi, and Roberto Zamparelli. 2014. Frege in space: A program of compositional distributional semantics. Linguistic Issues in Language Technology, 9.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5(0):135–146, June.

Erik Cambria, Soujanya Poria, Devamanyu Hazarika, and Kenneth Kwok. 2018. SenticNet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings. In AAAI.

Andrea Ceron, Luigi Curini, and Stefano M. Iacus. 2015. Using sentiment analysis to monitor electoral campaigns: Method matters—evidence from the United States and Italy. Social Science Computer Review, 33(1):3–20, February.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In Proceedings of the 23rd International Conference on Computational Linguistics: Posters, pages 241–249. Association for Computational Linguistics.

Daantje Derks, Arjan ER Bos, and Jasper Von Grumbkow. 2008. Emoticons and online message interpretation. Social Science Computer Review, 26(3):379–388.

Claus-Peter Ernst and Martin Huschens. 2018. The effects of different emoticons on the perception of emails in the workplace. In Proceedings of the 51st Hawaii International Conference on System Sciences.

Yifan Fu, Xingquan Zhu, and Bin Li. 2013. A survey on instance selection for active learning. Knowledge and Information Systems, 35(2):249–283, May.

Lorenzo Gatti, Marco Guerini, and Marco Turchi. 2016. SentiWords: Deriving a high precision and high coverage lexicon for sentiment analysis. IEEE Transactions on Affective Computing, 7(4):409–421, October.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford, 1(12).

Paula Gombar, Zoran Medić, Domagoj Alagić, and Jan Šnajder. 2017. Debunking sentiment lexicons: A case of domain-specific sentiment classification for Croatian. In Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing, pages 54–59.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In Proceedings of NAACL-HLT 2016, pages 1367–1377, San Diego, California, June 12-17, 2016. Association for Computational Linguistics.

Ye Jiang, Xingyi Song, Jackie Harrison, Shaun Quegan, and Diana Maynard. 2017. Comparing attitudes to climate change in the media using sentiment analysis based on Latent Dirichlet Allocation. In Proceedings of the 2017 EMNLP Workshop: Natural Language Processing Meets Journalism, pages 25–30.

Barry Kavanagh. 2010. A cross-cultural analysis of Japanese and English non-verbal online communication: The use of emoticons in weblogs. Intercultural Communication Studies, 19(3):65–80.

O.Yu. Koltsova, S.V. Alexeeva, and S.N. Kolcov. 2016. An opinion word lexicon and a training dataset for Russian sentiment analysis of social media. In Proceedings of the International Conference on Computational Linguistics and Intellectual Technologies Dialog 2016, pages 277–287, Moscow, June 1–4, 2016.

Peter Koncz and Ján Paralič. 2013. Active learning enhanced document annotation for sentiment analysis. In Availability, Reliability, and Security in Information Systems and HCI, Lecture Notes in Computer Science, pages 345–353. Springer, Berlin, Heidelberg, September.

Bing Liu. 2015. Sentiment Analysis: Mining Opinions, Sentiments, and Emotions. Cambridge University Press, Cambridge.

Natalia V. Loukachevitch and Anatolii Levchik. 2016. Creating a general russian sentiment lexicon. In Proceedings of the Tenth International Conference on Language Resources and Evaluation, pages 1171–1176, Portorož, Slovenia. ELRA.

N.V. Loukachevitch and Yu.V. Rubtsova. 2016. SentiRuEval-2016: Overcoming time gap and data sparsity in Twitter sentiment analysis. In Proceedings of the International Conference on Computational Linguistics and Intellectual Technologies Dialog 2016, pages 375–384.

Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. Introduction to information retrieval, volume 1. Cambridge university press Cambridge.

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

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12:2825–2830.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), pages 502–518.

Yu.V. Rubtsova. 2015. Constructing a corpus for sentiment classification training. Mezhdunarodnyj zhurnal "Programmnye produkty i sistemy", 27:72–78, March.

Tapan Sahni, Chinmay Chandak, Naveen Reddy Chedeti, and Manish Singh. 2017. Efficient Twitter sentiment classification using subjective distant supervision. In Communication Systems and Networks (COMSNETS), 2017 9th International Conference On, pages 548–553. IEEE.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1201–1211.

Mike Thelwall and Kevan Buckley. 2013. Topic-based sentiment analysis for the social web: The role of mood and issue-related words. Journal of the American Society for Information Science and Technology, 64(8):1608–1617, August.

Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment strength detection in short informal text. Journal of the Association for Information Science and Technology, 61(12):2544–2558.

Cigdem Toprak, Niklas Jakob, and Iryna Gurevych. 2010. Sentence and expression level annotation of opinions in user-generated discourse. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 575–584, Uppsala, Sweden 11-16 June 2010.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring sentiment in social media: Bootstrapping subjectivity clues from multilingual twitter streams. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), volume 2, pages 505–510.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. Language Resources and Evaluation, 39(2-3):165–210, May.

Francisco Yus and Francisco Yus. 2014. Not all emoticons are created equal. Linguagem em (Dis)curso, 14(3):511–529, December.

Self-Normalization Properties of Language Modeling

Jacob Goldberger
Faculty of Engineering
Bar-Ilan University, Israel
jacob.goldberger@biu.ac.il

Oren Melamud
IBM Research
Yorktown Heights, NY, USA
oren.melamud@ibm.com

Abstract

Self-normalizing discriminative models approximate the normalized probability of a class without having to compute the partition function. In the context of language modeling, this property is particularly appealing as it may significantly reduce run-times due to large word vocabularies. In this study, we provide a comprehensive investigation of language modeling self-normalization. First, we theoretically analyze the inherent self-normalization properties of Noise Contrastive Estimation (NCE) language models. Then, we compare them empirically to softmax-based approaches, which are self-normalized using explicit regularization, and suggest a hybrid model with compelling properties. Finally, we uncover a surprising negative correlation between self-normalization and perplexity across the board, as well as some regularity in the observed errors, which may potentially be used for improving self-normalization algorithms in the future.

1 Introduction

The ability of statistical language models (LMs) to estimate the probability of a word given a context of preceding words, plays an important role in many NLP tasks, such as speech recognition and machine translation. Recurrent Neural Network (RNN) language models have recently become the preferred method of choice, having outperformed traditional n-gram LMs across a range of tasks (Jozefowicz et al., 2016). Unfortunately however, they suffer from scalability issues incurred by the computation of the softmax normalization term, which is required to guarantee proper probability predictions. The cost of this computation is linearly proportional to the size of the word vocabulary and has a significant impact on both training and testing run-times.

Several methods have been proposed to cope with this scaling issue by replacing the softmax with a more computationally efficient component at train time. [1] These include importance sampling (Bengio and et al, 2003), hierarchical softmax (Minh and Hinton, 2008), BlackOut (Ji et al., 2016) and Noise Contrastive Estimation (NCE) (Gutmann and Hyvarinen, 2012). NCE has been applied to train neural LMs with large vocabularies (Mnih and Teh, 2012) and more recently was also successfully used to train LSTM-RNN LMs (Vaswani et al., 2013; Chen et al., 2015; Zoph et al., 2016), achieving near state-of-the-art performance on language modeling tasks (Jozefowicz et al., 2016; Chen et al., 2016). All the above works focused on reducing the complexity at train time. However, at test time, the assumption was that one still needs to compute the costly softmax normalization term to obtain a normalized score fit as an estimate for the probability of a word.

Self-normalization was recently proposed to address the test time complexity. A self-normalized discriminative model is trained to produce near-normalized scores in the sense that the sum over the scores of all words is approximately one. If this approximation is close enough, the assumption is that the costly exact normalization can be waived at test time without significantly sacrificing prediction accuracy (Devlin et al., 2014). Two main approaches were proposed to train self-normalizing models. Regularized softmax self-normalization is based on using softmax for training and explicitly encouraging the normalization term of the softmax to be as close to one as possible, thus making its computation redundant

[1] Alleviating this problem using sub-word representations is a parallel line of research not discussed here.

Proceedings of the 27th International Conference on Computational Linguistics, pages 764–773
Santa Fe, New Mexico, USA, August 20-26, 2018.

at test time (Devlin et al., 2014; Andreas and Klein, 2015; Chen et al., 2016). The alternative approach is based on NCE. The original formulation of NCE included a parametrized normalization term Z_c for every context c. However, the first work that applied NCE to language modeling (Mnih and Teh, 2012) discovered empirically that fixing Z_c to a constant did not affect the performance. More recent studies (Vaswani et al., 2013; Zoph et al., 2016; Chen et al., 2015; Oualil and Klakow, 2017) empirically found that models trained using NCE with a fixed Z_c, exhibit self-normalization at test time. This behavior is facilitated by inherent self-normalization properties of NCE LMs that we analyze in this work.

The main contribution of this study is in providing a first comprehensive investigation of self-normalizing language models. This includes a theoretical analysis of the inherent self-normalizing properties of NCE language models, followed by an extensive empirical evaluation of NCE against softmax-based self-normalizing methods. Our results suggest that regularized softmax models perform competitively as long as we are only interested in low test time complexity. However, when train time is also a factor, NCE has a notable advantage. Furthermore, we find, somewhat surprisingly, that larger models that achieve better perplexities tend to have worse self-normalization properties, and perform further analysis in an attempt to better understand this behavior. Finally, we show that downstream tasks may not all be as sensitive to self-normalization as might be expected.

The rest of this paper is organized as follows. In sections 2 and 3, we provide theoretical background and analysis of NCE language modeling that justifies its inherent self-normalizing properties. In Section 4, we review the alternative regularized softmax-based self-normalization methods and introduce a novel regularized NCE hybrid approach. In Section 5, we report on an empirical intrinsic investigation of all the methods above, and finally, in Section 6, we evaluate the compared methods on the Microsoft's Sentence Completion Challenge and compare these results with the intrinsic measures of perplexity and self-normalization.

2 NCE as a Matrix Factorization

In this section, we review the NCE algorithm for language modeling (Gutmann and Hyvarinen, 2012; Mnih and Teh, 2012) and focus on its interpretation as a matrix factorization procedure. This analysis is analogous to the one proposed by Melamud et al. (2017) for their PMI language model. Let $p(w|c)$ be the probability of a word w given a preceding context c, and let $p(w)$ be the word unigram distribution. Assume the distribution $p(w|c)$ has the following parametric form:

$$p_{nce}(w|c) = \frac{1}{Z_c} \exp(m(w, c)) \tag{1}$$

such that $m(w, c) = \vec{w} \cdot \vec{c} + b_w$, where $\vec{w}$ and $\vec{c}$ are d-dimensional vector representations of the word w and its context c, and Z_c is a normalization term.

We can use a simple lookup table for the word representation $\vec{w}$, and a recurrent neural network (RNN) model to obtain a low dimensional representation of the entire preceding context $\vec{c}$. Given a text corpus D, the NCE objective function is:

$$S(m) = \sum_{w,c \in D} \Big[\log \sigma(m(w, c) - \log(p(w)k)) \tag{2}$$

$$+ \sum_{i=1}^{k} \log(1 - \sigma(m(w_i, c) - \log(p(w_i)k))) \Big]$$

such that w, c go over all the word-context co-occurrences in the learning corpus D and $w_1, ..., w_k$ are 'noise' samples drawn from the word unigram distribution. σ denotes the sigmoid function.

Let $\text{pce}(w, c) = \log p(w|c)$ be the Pointwise Conditional Entropy (PCE) matrix, which is the true log probability we are trying to estimate. Gutmann and Hyvarinen (2012) proved that $S(m) \leq S(\text{pce})$ for every matrix m. The rank of the matrix m is at most $d + 1$. Thus, the NCE training goal is finding the best low-rank decomposition of the PCE matrix in the sense that it minimizes the difference $S(\text{pce}) -$

$S(m)$. Following Melamud and Goldberger (2017), we can explicitly write this difference as a Kullback-Leibler (KL) divergence. The NCE derivation was originally based on sampling w and c either from the joint distribution or from the product of marginals according to a binary r.v. denoted by z. For every matrix m, the conditional distribution of z given w and c is:

$$p_m(z{=}1|w,c) = \sigma(m(w,c) - \log(kp(w))).$$

The difference between the NCE score at the PCE matrix and the NCE score at a given matrix m can be written as:

$$S(\text{pce}) - S(m) = \text{KL}(p_{\text{pce}}(z|w,c)||p_m(z|w,c)) \tag{3}$$

$$= \sum_{w,c} p(w,c) \sum_{z=0,1} p_{\text{pce}}(z|w,c) \log \frac{p_{\text{pce}}(z|w,c)}{p_m(z|w,c)}.$$

This view of NCE as a matrix factorization instead of a distribution estimation, makes the normalization factor redundant during training, thereby justifying the heuristics of setting $Z_c = 1$ used by Mnih and Teh (2012). The crux of the matrix decomposition view of NCE is that although the normalization term is not explicitly included here, the optimal low-dimensional model attempts to approximate the true conditional probabilities, which are normalized, and therefore we expect that it will be almost self-normalized. Indeed, in the next section we provide formal guarantees for that.

3 The NCE Self-Normalization property

We now address the test time efficiency of language models, which is the focus of this study. As is the case with other language models, at test time, when we use the low-dimensional matrix learned by NCE to compute the conditional probability $p_{nce}(w|c)$ (1), we need to compute the normalization factor to obtain a valid distribution:

$$Z_c = \sum_w \exp(m(w,c)) = \sum_w \exp(\vec{w} \cdot \vec{c} + b_w). \tag{4}$$

Unfortunately, this computation of Z_c is often very expensive due to the typical large vocabulary size. However, as we next show, for NCE language models this computation may be avoided not only at train time, but also at test time due to self-normalization.

A matrix m is called self-normalized if $\sum_w(\exp(m(w,c)) = 1$ for every c. The full-rank optimal LM obtained from the PCE matrix $\text{pce}(w,c) = \log p(w|c)$, is clearly self normalized:

$$Z_c = \sum_w \exp(\text{pce}(w,c)) = \sum_w p(w|c) = 1.$$

The NCE algorithm seeks the best low-rank unnormalized matrix approximation of the PCE matrix. Hence, we can assume that the matrix m is close to the PCE matrix and therefore defines a LM that should also be close to self-normalized:

$$\sum_w \exp(m(w,c)) \approx \sum_w \exp(\text{pce}(w,c)) = 1. \tag{5}$$

We next formally show that if the matrix m is close to the PCE matrix then the NCE model defined by m is approximately self-normalized.

Theorem 1: Assume that for a given context c there is an $0 < \epsilon$ such that

$$\log \sum_{w \in V} p(w|c) \exp(|m(w,c) - \log p(w|c)|) \leq \epsilon.$$

Let $Z_c = \sum_w \exp(m(w,c))$ be the normalization factor. Then $|\log Z_c| \leq \epsilon$.

Proof:

$$\log Z_c = \log \sum_w \exp(\vec{w} \cdot \vec{c} + b_w)$$

$$= \log \sum_w (p(w|c) \exp(m(w,c) - \log p(w|c)))$$

$$\leq \log \sum_w p(w|c) \exp(|m(w,c) - \log p(w|c)|) \leq \epsilon. \quad (6)$$

The concavity of the log function implies that:

$$-\log Z_c \leq -\sum_w p(w|c)(m(w,c) - \log p(w|c)) \quad (7)$$

$$= \sum_w p(w|c)(-(m(w,c) - \log p(w|c)))$$

The convexity of the exp function implies that:

$$\leq \log \sum_w p(w|c) \exp(-(m(w,c) - \log p(w|c)))$$

$$\leq \log \sum_w p(w|c) \exp(|m(w,c) - \log p(w|c)|) \leq \epsilon$$

Combining Eq. (6) and Eq. (7) we finally obtain that $|\log Z_c| \leq \epsilon$. □

We can also state a global version of Theorem 1 and its proof is similar.

Theorem 2: Assume there is an $0 < \epsilon$ such that

$$\log \sum_{w,c} p(w,c) \exp(|m(w,c) - \log p(w|c)|) \leq \epsilon.$$

Then $|\sum_c p(c) \log Z_c| \leq \epsilon$.

4 Explicit Self-normalization

In this section, we review the two recently proposed language modeling methods that achieve self-normalization via explicit regularization, and then borrow from them to derive an novel regularized version of NCE.

The standard language modeling learning method, which is based on a softmax output layer, is not self-normalized. To encourage its self-normalization, Devlin et al. (2014) proposed to add to its training objective function, an explicit penalty for deviating from self-normalization:

$$S_{Dev} = \sum_{w,c \in D} \left[(\vec{w} \cdot \vec{c} + b_w - \log Z_c) - \alpha (\log Z_c)^2 \right] \quad (8)$$

where $Z_c = \sum_{v \in V} \exp(\vec{v} \cdot \vec{c} + b_v)$ and α is a constant. The drawback of this approach is that at train time you still need to explicitly compute the costly Z_c. Andreas and Klein (2015) proposed a more computationally efficient approximation of (8) that eliminates Z_c in the first term and computes the second term only on a sampled subset D' of the corpus D:

$$S_{And} = \sum_{w,c \in D} (\vec{w} \cdot \vec{c} + b_w) - \frac{\alpha}{\gamma} \sum_{c \in D'} (\log Z_c)^2 \quad (9)$$

where $\gamma < 1$ is an additional constant that determines the sampling rate, i.e. $|D'| = \gamma |D|$. They also provided analysis that justifies computing Z_c only on a subset of the corpus by showing that if a given LM is exactly self-normalized on a dense set of contexts (i.e. each context c is close to a context c' s.t. $\log Z_{c'} = 0$) then $E|\log Z_c|$ is small.

Inspired by this work, we propose a regularized variant of the NCE objective function (2):

$$S_{nce-r}(m) = S_{nce}(m) - \frac{\alpha}{\gamma} \sum_{c \in D'} (\log Z_c)^2 \quad (10)$$

This formulation allows us to further encourage the NCE self-normalization, still without incurring the cost of computing Z_c for every word in the learning corpus.

5 Intrinsic Evaluation

We report here on an empirical investigation of the self-normalization properties of NCE language modeling as compared to the alternative methods described in the previous sections.

5.1 Experimental Settings

We investigated the following language modeling methods:

- *DEV-LM* - the language model proposed by Devlin et al. (2014) (Eq. 8)
- *SM-LM* - a standard softmax language model (DEV-LM with $\alpha = 0$)
- *AND-LM* - the light-weight approximation of DEV-LM proposed by Andreas and Klein (2015) (Eq. 9)
- *NCE-LM* - NCE language model (Eq. 2)
- *NCE-R-LM* - our light-weight regularized NCE method (Eq. 10)

Following Devlin et al. (2014), to make all of the above methods approximately self-normalized at init time, we initialized their output bias terms to $b_w = -\log|V|$, where V is the word vocabulary. We set the negative sampling parameter for the NCE-based LMs to $k = 100$, following Zoph et al. (2016), who showed highly competitive performance with NCE LMs trained with this number of samples, and following Melamud et al. (2017) who used the same with PMI language models. We note that in early experiments with PMI LMs, which can be viewed as a close variant of NCE-LMs, we got very similar results for both of these types of models and therefore did not include PMI-LMs in our final investigation.

All of the compared methods use standard LSTM to represent the preceding (left-side) sequence of words as the context vector $\vec{c}$, and a simple word embedding lookup table to represent the predicted next word as $\vec{w}$. The LSTM hyperparameters and training regimen are similar to Zaremba et al. (2014) who achieved strong perplexity results compared to other standard LSTM-based neural language models. Specifically, we used a 2-layer LSTM with a 50% dropout ratio. During training, we performed truncated back-propagation-through-time, unrolling the LSTM for 20 steps at a time without ever resetting the LSTM state. We trained our model for 20 epochs using Stochastic Gradient Descent (SGD) with a learning rate of 1, which is decreased by a factor of 1.2 after every epoch starting after epoch 6. We clip the norms of the gradient to 5 and use mini-batch size of 20. All models were implemented using the Chainer toolkit ((Tokui et al., 2015)).

We used two popular language modeling datasets in the evaluation. The first dataset, denoted *PTB*, is a version of the Penn Tree Bank, commonly used to evaluate language models.[2] It consists of 929K/73K/82K training/validation/test words respectively and has a 10K word vocabulary. The second dataset, denoted *WIKI*, is the WikiText-2, more recently introduced by Merity et al. (2016). This dataset was extracted from Wikipedia articles and is somewhat larger, with 2,088K/217K/245K train/validation/test tokens, respectively, and a vocabulary size of 33K.

To evaluate self-normalization, we look at two metrics: (1) $\mu_z = E(\log(Z_c))$, which is the mean log value of the normalization term, across the contexts in the evaluated dataset; and (2) $\sigma_z = \sigma(\log(Z_c))$, which is the corresponding standard deviation. The closer these two metrics are to zero, the more self-normalizing the model is considered to be. We note that a model with an observed $|\mu_z| >> 0$ on a dev set, can be 'corrected' to a large extent (as we show later) by subtracting this dev μ_z from the unnormalized scores at test time. However, this is not the case for σ_z. Therefore, from a practical point of view, we consider σ_z to be the more important metric of the two. In addition, we also look at the classic perplexity metric, which is considered a standard intrinsic measure for the quality of the model predictions. Importantly, when measuring perplexity, except where noted otherwise, we first perform exact normalization of the models' unnormalized scores by computing the normalization term.

[2] Available from Tomas Mikolov at: `http://www.fit.vutbr.cz/~imikolov/rnnlm/simple-examples.tgz`

	NCE-LM			*SM-LM*		
d	μ_z	σ_z	perp	μ_z	σ_z	perp
	PTB validation set					
30	-0.18	0.11	267.6	2.29	0.97	243.4
100	-0.19	0.17	150.9	3.03	1.52	145.2
300	-0.15	0.29	100.1	3.77	1.98	97.7
650	-0.17	0.37	87.4	4.36	2.31	87.3
	WIKI validation set					
30	-0.20	0.13	357.4	2.57	1.02	322.2
100	-0.24	0.19	194.3	3.34	1.45	191.1
300	-0.23	0.27	125.6	4.19	1.73	123.3
650	-0.23	0.35	110.5	4.67	1.83	110.7

Table 1: Self-normalization and perplexity results of NCE-LM against the standard softmax language model, SM-LM. d denotes the size of the compared models (units).

5.2 Results

We begin by comparing the results obtained by the two methods that do not include any explicit self-normalization component in their objectives, namely NCE-LM and the standard softmax SM-LM. Table 1 shows that consistent with previous works, NCE-LM is approximately self-normalized as apparent by relatively low $|\mu_z|$ and σ_z values. On the other hand, SM-LM, as expected, is much less self-normalized. In terms of perplexity, we see that SM-LM performs a little better than NCE-LM when model dimensionality is low, but the gap closes entirely at $d = 650$. Curiously, while perplexity improves with higher dimensionality, we see that the quality of NCE-LM's self-normalization, as evident particularly by σ_z, actually degrades. This is surprising, as we would expect that stronger models with more parameters would approximate the true $p(w|c)$ more closely and hence be more self-normalized. A similar behavior was recorded for SM-LM. We further investigate this in Section 5.3.

We also measured model test run-times, running on a single Tesla K20 GPU. We compared run-times for normalized scores that were produced by applying exact normalization versus unnormalized scores. For both SM-LM and NCE-LM, which perform the same operations at test time, we get $\sim$9 seconds for normalized scores vs. $\sim$8 seconds for unnormalized ones on the PTB validation set. Run-times on the x3 larger Wiki validation set are $\sim$38 seconds for normalized and $\sim$24 seconds for unnormalized. We see that the run-time of the unnormalized models seems to scale linearly with the size of the dataset. However, the normalized run-time scales super-linearly, arguably since it depends heavily on the vocabulary size, which is greater for Wiki than for PTB. With typical vocabulary sizes reaching much higher than Wiki's 33K word types, this reinforces the computational motivation for self-normalized language models.

Next, Table 2 compares the self-normalization and perplexity performance of DEV-LM for varied values of the constant α on the validation sets. As could be expected, the larger the value of α is, the better the self-normalization becomes, reaching very good self-normalization for $\alpha = 10.0$. On the other hand, the improvement in self-normalization seems to occur at the expense of perplexity. This is particularly true for the smaller models, but is still evident even for $d = 650$. Interestingly, as with NCE-LM, we see that σ_z grows (i.e. self-normalization becomes worse) with the size of the model, and is negatively correlated with the improvement in perplexity.

Finally, in Table 3, we compare AND-LM against our proposed NCE-R-LM, using a sampling rate of $\gamma = 0.1$ to avoid computing Z_c most of the time, and varied values of α. As can be seen, AND-LM exhibits relatively bad performance. In particular, to make the model converge when trained on the WIKI dataset, we had to follow the heuristic suggested by Chen et al. (2016), applying the following conversion to all of AND-LM's unnormalized scores, $x \rightarrow 10 \tanh(x/5)$. In contrast, we see that NCE-R-LM is able to use the explicit regularization to improve self-normalization at the cost of a relatively small degradation in perplexity.

	DEV-LM								
	$\alpha = 0.1$			$\alpha = 1.0$			$\alpha = 10.0$		
d	μ_z	σ_z	perp	μ_Z	σ_z	perp	μ_z	σ_z	perp
	PTB validation set								
30	-0.12	0.21	242.6	-0.16	0.09	250.9	-0.13	0.060	307.2
100	-0.10	0.28	143.3	-0.17	0.11	149.5	-0.12	0.058	182.0
300	-0.09	0.36	96.3	-0.14	0.14	100.8	-0.16	0.054	121.3
650	-0.14	0.43	85.0	**-0.17**	**0.18**	**86.3**	-0.11	0.071	99.5
	WIKI validation set								
30	-0.10	0.23	334.1	-0.17	0.08	338.7	-0.15	0.055	389.0
100	-0.13	0.28	189.4	-0.22	0.13	191.1	-0.15	0.071	228.3
300	-0.15	0.34	121.9	-0.20	0.17	125.7	-0.13	0.081	143.6
650	-0.23	0.42	109.1	**-0.23**	**0.20**	**110.0**	-0.12	0.089	116.9

Table 2: Self-normalization and perplexity results of the self-normalizing DEV-LM for different values of the normalization factor α. d denotes the size of the compared models (units).

	NCE-R-LM			AND-LM		
α	μ_z	σ_z	perp	μ_z	σ_z	perp
	PTB validation set					
0.1	-0.19	0.34	87.1	6.14	0.56	117.5
1.0	-0.21	0.27	87.2	**0.45**	**0.25**	**119.4**
10.0	**-0.19**	**0.17**	**89.8**	-0.037	0.079	143.7
100.0	-0.089	0.086	112.6	-0.024	0.030	209.5
	WIKI validation set					
0.1	-0.23	0.33	111.1	**4.85**	**0.72**	**201.5**
1.0	-0.24	0.28	107.5	1.02	0.001	1481.3
10.0	**-0.22**	**0.19**	**110.8**	0.41	0.12	33323.1
100.0	-0.12	0.099	131.5	0.413	0.000	33278.0

Table 3: Self-normalization and perplexity results of the self-normalizing DEV-LM for different values of the normalization factor α. d = 650 and γ = 0.1.

Switching to the test-set evaluation, we propose a simple technique to center the $\log(Z)$ values of a self-normalizing model's scores around zero. Let μ_z^{valid} be $E(\log(Z))$ observed on the validation set at train time. The probability estimates of the 'shifted' model at test time are $\log p(w|c) = \vec{w}\cdot\vec{c}+b_w-\mu_z^{valid}$. Table 4 shows the results that we get when evaluating the shifted versions of DEV-LM, NCE-R-LM, NCE-LM and AND-LM with $d = 650$. For each compared model, we chose the α value that showed the best self-normalization performance without sacrificing significant perplexity performance. Specifically, we used $\alpha_{\text{DEV-LM}} = 1.0$ and $\alpha_{\text{NCE-R-LM}} = 10.0$ for both PTB and WIKI datasets, and then $\alpha_{\text{AND-LM}} = 1.0$ and $\alpha_{\text{AND-LM}} = 0.1$ for the PTB and WIKI datasets, respectively. Following Oualil and Klakow (2017), in addition to perplexity, we also report 'unnormalized perplexity', which is computed with the unnormalized model scores. When the unnormalized perplexity measure is close to the real perplexity, this suggests that the unnormalized scores are in fact nearly normalized.

As can be seen, with the shifting method, all models achieve near perfect (zero) μ_z value, and their unnormalized perplexities are almost identical to their respective real perplexities. Also, with the exception of AND-LM, the perplexities of all models are nearly identical. Finally, the standard deviation of the normalization term of DEV-LM and NCE-R-LM is notably better than that of NCE-LM and AND-LM. DEV-LM and NCE-R-LM perform very similar in all respects. However, we note that NCE-R-LM's advantage is that during training, it performs sparse computations of the costly normalization term and therefore its training time depends much less on the size of the vocabulary.

	PTB-test				*WIKI-test*			
	μ_z	σ_z	perp	u-perp	μ_z	σ_z	perp	u-perp
DEV-LM	-0.001	0.17	83.1	83.0	0.002	0.20	104.1	104.2
NCE-R-LM	0.002	0.17	85.9	86.0	-0.003	0.19	105.0	104.7
NCE-LM	-0.004	0.35	83.7	83.4	0.003	0.36	104.3	104.6
AND-LM	0.001	0.30	114.9	115.0	0.018	0.74	185.7	189.1

Table 4: Self-normalization and perplexity results on test sets for 'shifted' models with $d = 650$. 'u-perp' denotes unnormalized perplexity.

	PTB-validation		*WIKI-validation*	
d	NCE-LM	DEV-LM ($\alpha = 1.0$)	NCE-LM	DEV-LM ($\alpha = 1.0$)
30	-0.33	-0.27	-0.50	-0.26
100	-0.29	-0.29	-0.53	-0.49
300	-0.46	-0.41	-0.56	-0.63
650	-0.50	-0.45	-0.53	-0.64

Table 5: Pearson's correlation between H_c (entropy) and $\log(Z_c)$ on samples from the validation sets.

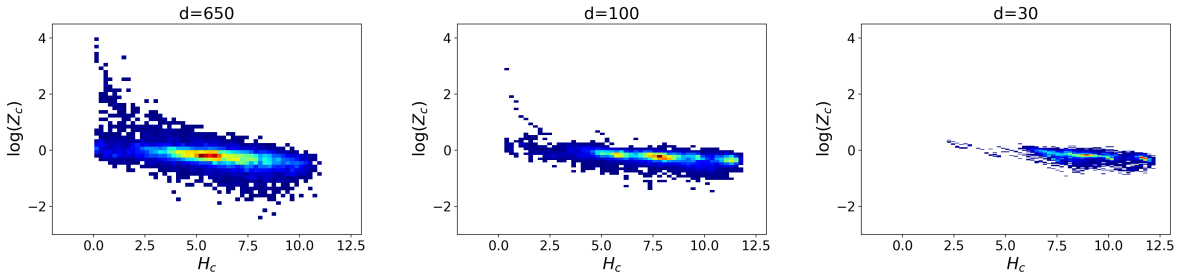


Figure 1: A 2-dimensional histogram of the normalization term of a predicted distribution as a function of its entropy, as measured over a sample from NCE-LM predictions on the WIKI validation set. Brighter colors denote denser areas.

5.3 Analysis

The entropy of the distributions predicted by a language model is a measure of how uncertain it is regarding the identity of the predicted word. Low-entropy distributions would be concentrated around few possible words, while high-entropy ones would be much more spread out. To more carefully analyze the self-normalization properties of NCE-LM and DEV-LM, we computed the Pearson's correlation between the entropy of a predicted distribution $H_c = -\sum_v p(v|c) \log p(v|c)$ and its normalization term, $\log(Z_c)$. As can be seen in Table 5, it appears that a regularity exists, where the value of $\log(Z_c)$ is negatively correlated with entropy. Furthermore, it seems that, to an extent, the correlation is stronger for larger models. To further illustrate this regularity, Figure 1 shows a 2-dimensional histogram of a sample of distributions predicted by NCE-LM . We can see there that particularly low entropy distributions can be associated with very high values of $\log(Z_c)$, deviating a lot from the self-normalization objective of $\log(Z_c) = 0$. Examples for contexts with such low-entropy distributions are: "During the American Civil [War]" and "The United [States]", where the actual word following the preceding context appears in parenthesis. This phenomenon is less evident for smaller models, which tend to produce fewer low entropy predictions.

We hypothesize that the above observations could be a contributing factor to our earlier finding that larger models have larger variance in their normalization terms, though it seems to account only for some of that at best. Furthermore, we hope that this regularity could be exploited to improve self-normalization algorithms in the future.

	2 training iterations				5 training iterations			
	acc-n	Δacc	perp	σ_z	acc-n	Δacc	perp	σ_z
DEV-LM	47.6	+0.4	75.3	0.10	51.9	-0.7	67.5	0.10
NCE-R-LM	46.3	-0.5	78.3	0.11	51.0	-0.2	70.2	0.10
NCE-LM	47.6	-0.4	75.3	0.17	51.6	+1.1	67.1	0.14
SM-LM	47.0	**-2.0**	73.4	1.15	51.0	**-2.3**	66.3	1.19

Table 6: Microsoft Sentence Completion Challenge (MSCC) results for models with $d = 650$ that were trained with 2 and 5 iterations. 'acc-n' denotes the accuracy measure obtained when language model scores are precisely normalized. 'Δacc' denotes the delta in accuracy when unnormalized scores are used instead. 'perp' and σ_z denote the mean perplexity and standard deviation of $log(Z_c)$ recorded for the 1,040 answer sentences.

6 Sentence Completion Challenge

In Section 5, we've seen that there may be some trade-offs between perplexity, self-normalization and run-time complexity of language models. While the language modeling method should ultimately to be optimized for each downstream task individually, we follow Mnih and Teh (2012) and use the Microsoft Sentence Completion Challenge (Zweig and Burges, 2011) as an example use case.

The Microsoft Sentence Completion Challenge (MSCC) (Zweig and Burges, 2011) includes 1,040 items. Each item is a sentence with one word replaced by a gap, and the challenge is to identify the word, out of five choices, that is most meaningful and coherent as the gap-filler. The MSCC includes a learning corpus of approximately 50 million words. To use this corpus for training our language models, we split it into sentences, shuffled the sentence order and considered all words with frequency less than 10 as unknown, yielding a vocabulary of about 50K word types. We used the same settings described in Section 5.1 to train the language models except that due to the larger size of the data, we ran fewer training iterations. [3] Finally, as the gap-filler, we choose the word that maximizes the score of the entire sentence, where a sentence score is the sum of its words' scores. For a normalized language model this score can be interpreted as the estimated log-likelihood of the sentence.

The results of the MSCC experiment appear in Table 6. Accuracy is the standard evaluation metric for this benchmark (simply the proportion of questions answered correctly). We report this metric when performing the costly test-time score normalization and then the delta observed when using unnormalized scores instead. First, we note that given the same number of training iterations, all methods achieved fairly similar normalized-scores accuracies, as well as perplexity values. At the same time, we do see a notable improvement in both accuracies and perplexities when more training iterations are performed. Next, with the exception of SM-LM, all of the compared models exhibit good self-normalization properties, as is evident from the low σ_z values. There does not seem to be a meaningful accuracy performance hit when using unnormalized-scores for these models, suggesting that this level of self-normalization is adequate for the MSCC task. Finally, as expected, SM-LM exhibits worse self-normalization properties. However, somewhat surprisingly, even in this case, we see a relatively small (though more noticeable) hit in accuracy. This suggests that in some use cases, the level of the language model's self-normalization may have a relatively low impact on the performance of a down-stream task.

7 Conclusions

We reviewed and analyzed the two alternative approaches for self-normalization of language models, namely, using Noise Contrastive Estimation (NCE) that is inherently self-normalized, versus adding explicit self-normalizing regularization to a softmax objective function. Our empirical investigation compared these approaches, and by extending NCE language modeling with a light-weight explicit self-normalization, we also introduced a hybrid model that achieved both good self-normalization and perplexity performance, as well as little dependence of train-time on the size of the vocabulary. To put our

[3] We started with a learning rate of 1 and reduced it by a factor of 2 after each iteration beginning with the very first one.

intrinsic evaluation results in perspective, we used the Sentence Completion Challenge as an example use-case. The results suggest that it would be wise to test the sensitivity of the downstream task to self-normalization, in order to choose the most appropriate method. Finally, further analysis revealed unexpected correlations between self-normalization and perplexity performance, as well as between the partition function of self-normalized predictions and the entropy of the respective distribution. We hope that these insights would be useful for improving self-normalizing models in future work.

References

[Andreas and Klein2015] J. Andreas and D. Klein. 2015. When and why are log-linear models self-normalizing? In *NAACL*.

[Bengio and et al2003] Y. Bengio and J. Senecal et al. 2003. Quick training of probabilistic neural nets by importance sampling. In *AISTATS*.

[Chen et al.2015] X. Chen, X. Liu, M. Gales, and P. C. Woodland. 2015. Recurrent neural network language model training with noise contrastive estimation for speech recognition. In *ICASSP*.

[Chen et al.2016] W. Chen, D. Grangier, and M. Auli. 2016. Strategies for training large vocabulary neural language models. *CoRR*, abs/1512.04906.

[Devlin et al.2014] J. Devlin, R. Zbib, Z. Huang, T. Lamar, R. Schwartz, and J. Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of ACL*.

[Gutmann and Hyvarinen2012] M. U. Gutmann and A. Hyvarinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13:307–361.

[Ji et al.2016] S. Ji, S. Vishwanathan, N. Satish, A. Nadathur, J. Michael, and P. Dubey. 2016. Blackout: Speeding up recurrent neural network language models with very large vocabularies. ICLR.

[Jozefowicz et al.2016] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

[Melamud and Goldberger2017] O. Melamud and J. Goldberger. 2017. Information-theory interpretation of the skip-gram negative-sampling objective function. In *Proceedings of ACL*.

[Melamud et al.2017] O. Melamud, I. Dagan, and J. Goldberger. 2017. A simple language model based on pmi matrix approximations. In *EMNLP*.

[Merity et al.2016] S. Merity, C. Xiong, J. Bradbury, and R. Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

[Minh and Hinton2008] A. Minh and G. E. Hinton. 2008. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems*.

[Mnih and Teh2012] A. Mnih and Y. W. Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *ICML*.

[Oualil and Klakow2017] Y. Oualil and D. Klakow. 2017. A batch noise contrastive estimation approach for training large vocabulary language models. In *Interspeech*.

[Tokui et al.2015] S. Tokui, K. Oono, S. Hido, and J. Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Workshop on Machine Learning Systems (LearningSys) in The 29th Annual Conference on Neural Information Processing Systems*.

[Vaswani et al.2013] A. Vaswani, Y. Zhao, V. Fossum, and D. Chiang. 2013. Decoding with large-scale neural language models improves translation. In *EMNLP*.

[Zaremba et al.2014] W. Zaremba, I. Sutskever, and O. Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

[Zoph et al.2016] B. Zoph, A. Vaswani, J. May, and K. Knight. 2016. Simple, fast noise-contrastive estimation for large RNN vocabularies. In *NAACL*.

[Zweig and Burges2011] Geoffrey Zweig and Christopher JC Burges. 2011. The microsoft research sentence completion challenge. Technical report, Technical Report MSR-TR-2011-129, Microsoft.

A Position-aware Bidirectional Attention Network for Aspect-level Sentiment Analysis

Shuqin Gu[1], **Lipeng Zhang**[2], **Yuexian Hou**[1]*and **Yin Song**[1]
[1]School of Computer Science and Technology, Tianjin University, Tianjin, China
[2]School of Computer Software, Tianjin University, Tianjin, China
{shuqingu,lpzhang, yxhou, songyin}@tju.edu.cn

Abstract

Aspect-level sentiment analysis aims to distinguish the sentiment polarity of each specific aspect term in a given sentence. Both industry and academia have realized the importance of the relationship between aspect term and sentence, and made attempts to model the relationship by designing a series of attention models. However, most existing methods usually neglect the fact that the position information is also crucial for identifying the sentiment polarity of the aspect term. When an aspect term occurs in a sentence, its neighboring words should be given more attention than other words with long distance. Therefore, we propose a position-aware bidirectional attention network (PBAN) based on bidirectional GRU. PBAN not only concentrates on the position information of aspect terms, but also mutually models the relation between aspect term and sentence by employing bidirectional attention mechanism. The experimental results on SemEval 2014 Datasets demonstrate the effectiveness of our proposed PBAN model.

1 Introduction

Sentiment analysis, also known as opinion mining (Liu, 2012; Pang et al., 2008), is a vital task in Natural Language Processing (NLP). It divides the text into two or more classes according to the affective states and the subjective information of the text, and has received plenty of attention from both industry and academia. In this paper, we address the aspect-level sentiment analysis, which is a fine-grained task in the field of sentiment analysis. For instance, given the mentioned aspect terms $\{menu, server, specials\}$, and the sentence is *"The menu looked good, except for offering the Chilean Sea Bass, but the server does not offer up the specials that were written on the board outside."*. For aspect term *menu*, the sentiment polarity is *positive*, but for aspect term *server*, the polarity is *negative* while for *specials*, the polarity is *neutral*.

One important challenge in aspect-level sentiment analysis is how to model the semantic relationship between aspect terms and sentences. Traditional approaches have defined rich features about content and syntactic structures so as to capture the sentiment polarity (Jiang et al., 2011). However this kind of feature-based method is labor-intensive and highly depends on the quality of the features. Compared with these methods, neural network architectures are capable of learning features without feature engineering, and have been widely used in a variety of NLP tasks such as machine translation (Cho et al., 2014), question answering (Andreas et al., 2016) and text classification (Lai et al., 2015). Recently, with the development of the neural networks, they are also applied to target-dependent sentiment analysis[1], such as Target-Dependent LSTM (TD-LSTM) (Tang et al., 2015) and Target-Connection LSTM (TC-LSTM) (Tang et al., 2015). However, these neural network-based methods can not effectively identify which words in the sentence are more important. Fortunately, attention mechanisms are an effective way to solve this problem.

*Corresponding author: Yuexian Hou.

[1]The aim of the target-dependent sentiment analysis is similar to aspect-level sentiment analysis. Given a sentence and target/aspect term, the task calls for inferring the sentiment polarity of the sentence towards the target/aspect term.

Proceedings of the 27th International Conference on Computational Linguistics, pages 774–784
Santa Fe, New Mexico, USA, August 20-26, 2018.

Attention, which is widely applied to Computer Vision (CV) and NLP fields, is an effective mechanism and has been demonstrated in image recognition (Mnih et al., 2014), machine translation (Bahdanau et al., 2014; Luong et al., 2015) and reading comprehension (Hermann et al., 2015; Cui et al., 2016). Therefore, some researchers have designed attention networks to address the aspect-level sentiment analysis and have obtained comparable results, such as AE-LSTM (Wang et al., 2016), ATAE-LSTM (Wang et al., 2016) and IAN (Ma et al., 2017).

However, these existing work ignores or does not explicitly model the position information of the aspect term in a sentence, which has been studied for improving performance in information retrieval (IR). In (Liu et al., 2015), the occurrence positions of the query terms were modeled via kernel functions and then integrated into traditional IR models to boost the retrieval performance. By analyzing this aspect-level sentiment analysis task and the corresponding dataset, we find that when an aspect term occurs in a sentence, its neighboring words in the sentence should be given more attention than other words with long distance. Let us take *"It's a perfect place to have an amazing indian food."* as an example, when the aspect term is *indian food*, its corresponding sentiment polarity is *positive*. Intuitively, we can see that the neighboring word of the *indian food* (i.e. *"amazing"*) has a greater contribution to judge the sentiment polarity of the aspect term than other words with long distance such as *"to"* and *"have"*. Sometimes this intuitive idea of judging the sentiment polarity may be interpreted as a cognitive activity, which also can be rephrased in a quantum-like language model (Niu et al., 2017). To be specific, sentiment polarity may be interpreted as a quantum-like cognition state. Inspired by this, we go one step further and propose a position-aware bidirectional attention network (PBAN) based on bidirectional Gated Recurrent Units (Bi-GRU) (Cho et al., 2014).

In addition to utilizing the position information, PBAN also mutually models the relationship between the sentence and different words in the aspect term by adopting a bidirectional attention mechanism. To be specific, our model consists of three components: 1) Obtaining position information of each word in corresponding sentence based on the current aspect term, then converting the position information into position embedding. 2) The PBAN composes of two Bi-GRU networks focusing on extracting the aspect-level features and sentence-level features respectively. 3) Using the bidirectional attention mechanism to model the mutual relation between aspect term and its corresponding sentence. We evaluate our models on SemEval 2014 Datasets, and the results show that our models are more effective than other previous methods.

The main contributions of our work can be summarized as follows: (1) We attempt to explicitly investigate the effectiveness of the position information of aspect term for aspect-level sentiment analysis. (2) We propose a position-aware bidirectional attention network (PBAN) based on Bi-GRU, which has been proved to be effective to improve the sentiment analysis performance. (3) We apply a bidirectional attention mechanism, which can enhance the mutual relation between the aspect term and its corresponding sentence, and prevent the irrelevant words from getting more attention.

2 Model Overview

In this section, we describe the proposed model position-aware bidirectional attention network (PBAN) for aspect-level sentiment analysis and PBAN is shown in Figure 1. In this paper, the set of sentiment polarity of the aspect term is $\{positive, negative, neutral\}$.

2.1 Position Representation

As for how to model the position information of the aspect term in its corresponding sentence, inspired by the position encoding vectors used in (Collobert et al., 2011; Zeng et al., 2014), we define a position index sequence whose length is equal to the length of corresponding sentence. Suppose that if a word in the aspect term occurs in the sentence, then its position index will be marked as "0", and the position index of other words will be represented as the relative distance to the current aspect term.

$$p_i = \begin{cases} \mid i - j_s \mid, & i < j_s \\ 0, & j_s \leq i \leq j_e \\ \mid i - j_e \mid, & i > j_e \end{cases} \tag{1}$$

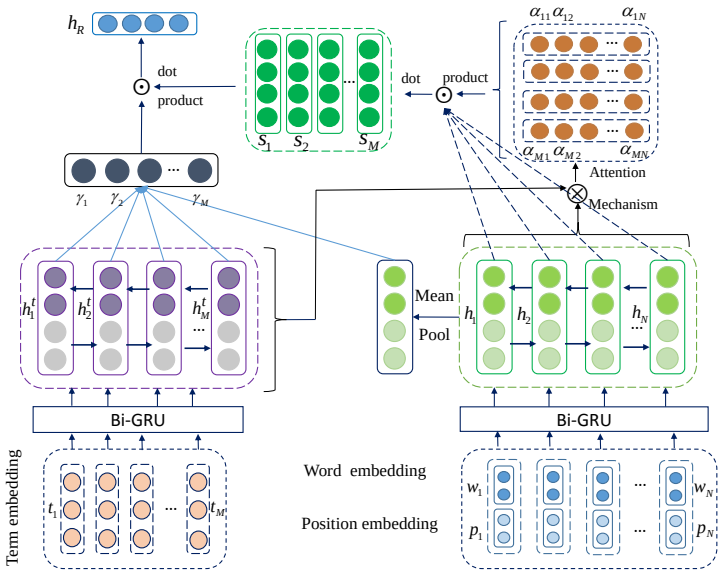


Figure 1: The architecture of position-aware bidirectional attention network for aspect-level sentiment analysis (PBAN). $\{w_1, w_2, ..., w_N\}$ represents the word embedding in a sentence whose length is N, and $\{t_1, t_2, ..., t_M\}$ represents the aspect term embedding whose length is M. $\{p_1, p_2, ..., p_N\}$ is the position embedding of the aspect term, which is concatenated to the word embedding. $\{\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_N\}$ denotes the hidden representation of inputs and $\{\mathbf{h}_1^t, \mathbf{h}_2^t, ..., \mathbf{h}_M^t\}$ indicates the hidden representation of aspect term.

where, j_s and j_e denote the starting and ending indices of the aspect term respectively, and p_i can be viewed as the relative distance of the $i\text{-}th$ word in sentence to the aspect term. For example, given a sentence *"not only was the food outstanding but the little perks were great."*, and the aspect term is *food*, then the position index sequence is represented as $\mathbf{p} = [4, 3, 2, 1, 0, 1, 2, 3, 4, 5, 6, 7]$. And its corresponding position embedding are obtained by looking up a position embedding matrix $\mathbf{P} \in \mathbb{R}^{d_p \times N}$, which is randomly initialized, and updated during the training process. Here, d_p denotes the dimension of position embedding and N indicates the length of the sentence. After the position index is converted to the embedding, the position embedding can model the different weights of words with different distance. From this example, it is obvious that the words with smaller relative distances (such as *"outstanding"*) play an more important role in judging the sentiment polarity of *food*. We can find that this process is basically consistent with the way people judge the sentiment polarity of the aspect term. Because we usually first observe the neighboring words of the aspect term, judging whether the neighboring words can show its sentiment polarity, after that we will focus on those words with long distance.

2.2 Word Representation

Bidirectional LSTMs have been successfully applied to various NLP tasks (Bahdanau et al., 2014), and it models the context dependency with the forward LSTM and the backward LSTM. The forward L-STM handles the sentence from left to right, and the backward LSTM processes it in the reverse order. Therefore, we can obtain two hidden representation, and then concatenate the forward hidden state and backward hidden state of each word. In this paper, we choose to use bidirectional GRU since it performs similarly to bidirectional LSTM but has fewer parameters and lower computational complexity.

Concretely, we firstly obtain the representation of each word in aspect term and sentence, and formalize the notations in our work. We suppose that a sentence consists of N words $[w_1, w_2, ..., w_N]$ and an aspect term contains M words $[t_1, t_2, ..., t_M]$, then we get sentence embedding and aspect term embedding by looking up a word embedding matrix $\mathbf{E} \in \mathbb{R}^{d \times v}$ respectively, where d denotes the dimension of the embedding, and v indicates the vocabulary size.

Then we input aspect term embeddings into the left Bi-GRU to get the hidden contextual representa-

tion, which consists of forward hidden state $\overrightarrow{\mathbf{h}_i^t} \in \mathbb{R}^{d_h}$ and backward hidden state $\overleftarrow{\mathbf{h}_i^t} \in \mathbb{R}^{d_h}$, where d_h denotes the number of hidden units. Finally, the hidden contextual representation of aspect term $\mathbf{h}_i^t$ is obtained by concatenating $\overrightarrow{\mathbf{h}_i^t}$ and $\overleftarrow{\mathbf{h}_i^t}$, i.e., $\mathbf{h}_i^t = [\overrightarrow{\mathbf{h}_i^t}; \overleftarrow{\mathbf{h}_i^t}] \in \mathbb{R}^{2d_h}$. For the right Bi-GRU structure, we take the concatenation of the position embedding and word embedding as the inputs, then we can obtain the final hidden contextual representation of the inputs, i.e., $\mathbf{h}_i = [\overrightarrow{\mathbf{h}_i}; \overleftarrow{\mathbf{h}_i}] \in \mathbb{R}^{2d_h}$.

2.3 Position-aware Bidirectional Attention Network Model (PBAN)

As shown in Figure 1, attention model in PBAN consists of two parts: including the aspect term to the position-aware sentence part and a position-aware sentence to the aspect term part. For the former part, we can obtain the different hidden contextual representation of a sentence according to different word in aspect term. For the later part, we can obtain the attention weights of the words in aspect term according to the position information, which is used for getting the final representation of a sentence. Details will be described in follwing sections.

Aspect term to position-aware sentence attention: A sentence should be represented differently based on different words in aspect term, because different words may have different effects on the final representation of the sentence. We firstly get the hidden contextual representation of the aspect term by the left Bi-GRU, and get the hidden contextual representation of inputs (i.e., the concatenation of word embedding and position embedding) by the right Bi-GRU structure. Here, we regard the position embedding as the part of the inputs, because it intuitively represents the relative distance of words in a sentence to the current aspect term as mentioned in section 2.1. Then we calculate the attention weights by adopting hidden contextual representation of aspect term and inputs, obtaining the attention weight distribution of sentence corresponding to each word in this aspect term. It can be formulated as follows:

$$\mathbf{s}_i = \sum_{j=1}^{N} \alpha_{ij} \mathbf{h}_j \tag{2}$$

$$\alpha_{ij} = \frac{exp(f(\mathbf{h}_j, \mathbf{h}_i^t))}{\sum_{k=1}^{N} exp(f(\mathbf{h}_k, \mathbf{h}_i^t))} \tag{3}$$

$$f(\mathbf{h}_j, \mathbf{h}_i^t) = \tanh(\mathbf{h}_j^T \mathbf{W}_m \mathbf{h}_i^t + b_m) \tag{4}$$

where α_{ij} indicates the attention weights from the word $\mathbf{h}_i^t$ in the aspect term to the j-th word in the inputs, and $tanh$ is a non-liner activation function. $\mathbf{W}_m$ is the weight matrix and b_m is the bias. Subsequently, α_{ij} is used to compute a weighted sum of the hidden representation $\mathbf{s}_i$, producing a semantic vector that represents the input sequence.

Position-aware sentence attention to aspect term: As we mentioned above, different words in aspect term will play different role in judging the sentiment polarity of aspect term. Since we obtain the hidden contextual representation of the inputs by the right Bi-GRU, we utilize both the position and semantic information for calculating the attention weights of different words in aspect term. The process can be formulated as follows:

$$\mathbf{h}_R = \sum_{i=1}^{M} \gamma_i \mathbf{s}_i \tag{5}$$

$$\gamma_i = \frac{exp(f(\overline{\mathbf{h}}, \mathbf{h}_i^t))}{\sum_{k=1}^{M} exp(f(\overline{\mathbf{h}}, \mathbf{h}_k^t))} \tag{6}$$

$$f(\overline{\mathbf{h}}, \mathbf{h}_i^t) = \tanh(\overline{\mathbf{h}}^T \mathbf{W}_n \mathbf{h}_i^t + b_n) \tag{7}$$

$$\overline{\mathbf{h}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{h}_i \tag{8}$$

where γ_i stands for the attention weights from inputs to the words in aspect term, denoting which word in aspect term should be more focused. $\overline{\mathbf{h}}$ is calculated by averagely pooling all Bi-GRU hidden states. Later, the sequence representation $\mathbf{x}$ is obtained by using a non-linear layer:

$$\mathbf{x} = \tanh(\mathbf{W}_R\mathbf{h}_R + \mathbf{b}_R) \tag{9}$$

where $\mathbf{W}_R$ and $\mathbf{b}_R$ are weight matrix and bias respectively.

We feed $\mathbf{x}$ into a linear layer, the length of whose output equals to the number of class labels $\|S\|$. Finally, we add a *softmax* layer to compute the probability distribution for judging the sentiment polarities as *positive*, *negative* or *neutral*:

$$\mathbf{y} = softmax(\mathbf{W}_s\mathbf{x} + \mathbf{b}_s) \tag{10}$$

where $\mathbf{W}_s$ and $\mathbf{b}_s$ are the weight matrix and bias respectively for *softmax* layer.

2.4 Model training

The PBAN model can be trained in an end-to-end way in a supervised learning framework, the aim of the training is to optimize all the parameters so as to minimize the objective function (loss function) as much as possible. In our work, let y_i be the correct sentiment polarity, which is represented by one-hot vector, and $\widehat{y}_i$ denotes the predicted sentiment polarity for the given sentence. We regard the cross-entropy as the loss function, and the formula is as follows:

$$loss = -\sum_{i=1}^{S} y_i log(\widehat{y}_i) + \frac{1}{2}\lambda \parallel \theta \parallel^2 \tag{11}$$

where λ is the regularization factor and θ contains all the parameters. Furthermore, in order to avoid over-fitting, we adopt the dropout strategy to enhance our PBAN model.

3 Experiments

3.1 Experiments Setting

Parameters Setting: In our experiments, all word embedding are initialized by the pre-trained Glove vector[2] (Pennington et al., 2014). All the weight matrices are given the initial value by sampling from the uniform distribution $U(-0.1, 0.1)$, and all the biases are set to zero. The dimension of the word embedding and aspect term embedding are set to 300, and the number of the hidden units are set to 200. The dimension of position embedding is set to 100, which is randomly initialized and updated during the training process. We use Tensorflow (Abadi et al., 2016) to implement our proposed model and employ the Momentum as the training method, whose momentum parameter γ is set to 0.9, λ is set to 10^{-6}, and the initial learning rate is set to 0.01.

Dataset: To evaluate our proposed methods, we conduct experiments on the dataset of SemEval 2014 Task4[3], the SemEval 2014 dataset consists of reviews in *Restaurant* and *Laptop* datasets. Each review contains a list of aspect terms and corresponding polarities, which are labeled with $\{positive, negative, neutral\}$. Particularly, each aspect term has its character index in the sentence, so that when different aspect term have the same word in a sentence, we can mark the relative position distance of a sentence according to the current aspect term without confusion. Table 1 shows the training and test sample numbers in each sentiment polarity.

3.2 Model Comparison

In order to evaluate the performance of our model, we compare our model with several baseline models, including LSTM (Wang et al., 2016), AE-LSTM (Wang et al., 2016), ATAE-LSTM (Wang et al., 2016), IAN (Ma et al., 2017) and MemNet (Tang et al., 2016).

[2]Pre-trained word vectors of Glove can be obtained from http://nlp.stanford.edu/projects/glove/

[3]The detail introduction of this dataset can be seen at: http://alt.qcri.org/semeval2014/task4/

Datasets	Positive		Negative		Neutral	
	Train	Test	Train	Test	Train	Test
Restaurant	2164	728	807	196	637	196
Laptop	994	341	870	128	464	169

Table 1: Samples of SemEval 2014 Dataset.

LSTM: LSTM takes the sentence as input so as to get the hidden representation of each word. Then it regards the average value of all hidden states as the representation of sentence, and puts it into *softmax* layer to predict the probability of each sentiment polarity. However, it can not capture any information of aspect term in sentence (Wang et al., 2016).

AE-LSTM: AE-LSTM first models the words in sentence via LSTM network and concatenate the aspect embedding to the hidden contextual representation for calculating the attention weights, which are employed to produce the final representation for the input sentence to judge the sentiment polarity (Wang et al., 2016).

ATAE-LSTM: ATAE-LSTM extended AE-LSTM by appending the aspect embedding to each word embedding so as to represent the input sentence, which highlights the role of aspect embedding. The other design of ATAE-LSTM is the same as AE-LSTM (Wang et al., 2016).

IAN: IAN considers the separate modeling of aspect terms and sentences respectively. IAN is able to interactively learn attentions in the contexts and aspect terms, and generates the representations for aspect terms and contexts separately. Finally, it concatenates the aspect term representation and context representation for predicting the sentiment polarity of the aspect terms within its contexts (Ma et al., 2017).

MemNet: MemNet applies attention multiple times on the word embedding, so that more abstractive evidences could be selected from the external memory. The output of the last attention layer is fed to a *softmax* layer for predictions (Tang et al., 2016).

Datasets	Restaurant		Laptop	
	Three-class	Two-class	Three-class	Two-class
LSTM	74.28	—	66.45	—
AE-LSTM	76.60	89.60	68.90	87.40
ATAE-LSTM	77.20	90.90	68.70	87.60
IAN	78.60	—	72.10	—
MemNet(9)	80.95	—	72.21	—
PBAN	**81.16**	**91.67**	**74.12**	**87.81**

Table 2: Comparison with baselines. Accuracy on Three-class and Two-class prediction about *Restaurant* and *Laptop* dataset, and Two-class denotes $\{positive, negative\}$. MemNet(9) indicates that MemNet with nine computational layers. Best scores are in bold.

Table 2 shows the performance of our model and other baseline models on datasets *Restaurant* and *Laptop* respectively. We can observe that our proposed PBAN model achieves the best performance among all methods. It is obvious that LSTM method gets the worst performance, because it treats aspect term and other words as the same, so that it can not take full advantage of the aspect term information and predicts the same polarity for different aspect terms in a sentence.

Furthermore, both AE-LSTM and ATAE-LSTM perform better than LSTM model, because they all consider the importance of the aspect term, and utilize the attention mechanism. Specifically, ATAE-LSTM outperforms AE-LSTM since it appends the aspect embedding to each word embedding and takes them as inputs, which helps the model obtain more semantic information related to aspect term. IAN realizes the importance of interaction between aspect term and context, and models aspect term and context using two connected attention networks. Thus, IAN performs better than ATAE-LSTM, and achieves an improvement of 1.40 points and 3.40 points on *Restaurant* and *Laptop* datasets in *Three-class*

respectively. MemNet(9) utilizes a more complex structure that containing nine computational layers, and it achieves better results compared to IAN since MemNet reads the useful information from external memory repeatedly.

Although both IAN and MemNet models performance better than other methods, they all perform less competitive than our PBAN both on *Restaurant* and *Laptop* datasets. For IAN model, it interactively learns the attentions between the aspect term and its corresponding sentence, but this attention mechanism is coarse-grained and it does not fully consider the influence of different words in aspect term on the sentence. For MemNet model, although it utilizes the location information, it is mainly used for calculating the memory vectors. Nevertheless, PBAN utilizes the character index of the aspect term (provided in the raw dataset) and adopts relative distance to represent the position sequence. As we have mentioned in previous sections, an aspect term contains several words and different words in aspect term should have different contributions to the final representation of sentence. In PBAN, the position information is regarded as the inputs of the Bi-GRU, so it can help calculate the weights of different words in aspect term and improve the final representation of the sentence. Moreover, when different aspect terms contain the same word, our proposed position information can effectively identify the current aspect term without confusion while MemNet can not.

Generally speaking, by integrating the position information and the bidirectional attention mechanism, PBAN achieves the state-of-the-art performances, and it can effectively judge the sentiment polarity of different aspect term in its corresponding sentence so as to improve the classification accuracy.

3.3 Analysis of PBAN Model

In this section, we design a series of models to demonstrate the effectiveness of our PBAN model. Firstly, we design an ATAE-Bi-GRU model, whose structure is similar with ATAE-LSTM. The only difference between these two models is that ATAE-Bi-GRU uses the Bi-GRU structure rather than LSTM, and other design is the same as ATAE-LSTM. Next we design a BAN model without modeling position embedding, and it just utilizes the representation of aspect term and sentence. In BAN, we still adopt bidirectional attention mechanism to model the relation between aspect term and sentence as PBAN does. The only difference between BAN and PBAN is that BAN without taking the position embedding as a part of inputs. Moreover, we also design a PAN model, whose structure is similar with the ATAE-Bi-GRU model. PAN takes the concatenation of the aspect term embedding and the word embedding as the inputs of the Bi-GRU structure to obtain the hidden contextual representation, and then PAN utilizes this representation and the position embedding of the aspect term to calculate the attention weights, so as to effectively judge the sentiment polarity of an aspect term. From Table 3, we can find that PBAN achieves the best performance among these models.

Dataset	Restaurant	Laptop
ATAE-LSTM	77.20	68.70
ATAE-Bi-GRU	77.68	69.47
PAN	78.07	71.13
IAN	78.60	72.10
BAN	78.74	72.61
PBAN	**81.16**	**74.12**

Table 3: Analysis of PBAN model.

Because Bi-GRU structure has a big advantage over LSTM, it is obvious that ATAE-Bi-GRU model performs better than ATAE-LSTM model. For PAN model, it outperforms ATAE-LSTM and ATAE-Bi-GRU models, but it is worse than BAN model. Compared with ATAE-Bi-GRU, the most difference is that PAN utilizes the position embedding to calculate the attention weights rather than the aspect term embedding like ATAE-Bi-GRU. Therefore, according to these three experimental results, we can prove the importance of the position information in aspect-level sentiment analysis task.

As for BAN model, it outperforms IAN model while performs worse than PBAN model. Because

Aspect term	Sentence	Polarity
pizza	This is one great place to eat pizza more out but not a good place for take-out pizza.	positive
take-out pizza	This is one great place to eat pizza more out but not a good place for take-out pizza.	negative

Figure 2: Case Study: The visualized attention weights for sentence and aspect term by PBAN.

compared with IAN model, BAN model can learn more semantic relationship between aspect term and sentence via bidirectional attention mechanism. However, it ignores the position information of aspect term when compared with PBAN model.

As we expect, PBAN achieves the best performance among all these models. This is because in addition to fully considering the position information of the aspect term in its corresponding sentence, PBAN also considers the mutual relationship between aspect term and sentence, which is mainly achieved by a bidirectional attention mechanism.

3.4 A Case Study

To have an intuitive understanding of our proposed model, we visualize the attention weights on the aspect term and sentence in Figure 2. The color depth indicates the importance degree of the weight, the darker the more important. In Figure 2, the sentence is *"**This is one great place to eat pizza more out but not a good place for take-out pizza.**"*, the polarities are *positive* and *negative* for ***pizza*** and ***take-out pizza*** respectively. From Figure 2, we can find that our model is more inclined to consider the neighboring words of the aspect term. For example, when the current aspect term is ***pizza***, obviously, its neighboring words such as *"great"*, *"place"* and *"more"* get more attention and play a great role for judging sentiment polarity of ***pizza***. However, those words that are far from the current aspect term such as *"but"*, *"not"* and *"take-out"* obtain less attention, which demonstrates the effectiveness of the position information. For aspect term ***take-out pizza***, it is obvious that the word *"take-out"* is more important to express the aspect term than the word *"pizza"*. From Figure 2, it is worth noting that some words such as *"good"* and *"place"* get less attention even they are closer to the current aspect term than *"but"* and *"not"*. This is because different words in aspect term have different effect on a sentence, and we apply the bidirectional attention mechanism to choose more useful words. For instance, in this case, PBAN should pay more attention on the word *"take-out"*. Therefore, PBAN is capable of figuring out the important part in a sentence for judging the sentiment polarity by modeling the mutual relation between sentence and different words in aspect term.

4 Related Work

In this section, we will briefly review some research on sentiment analysis in recent years. The previous research can be divided into three directions: traditional machine learning methods, neural network methods and attention network methods.

4.1 Machine Learning for Sentiment Analysis

Traditional machine learning approaches mainly involve text representation and feature extraction, such as bag-of-words models and sentiment lexicons features, then training a sentiment classifier (Prez-Rosas et al., 2012). Rao et al. (2010) demonstrated the utility of graph-based semi-supervised learning framework for building sentiment lexicons. Kaji et al. (2007) explored to use structural clues that could extract polar sentences from HTML documents, and built lexicon from the extracted polar sentences. However, these methods are labor-intensive, and usually results in high dimensional and high sparse phenomenon for the text representation.

4.2 Neural Network for Target-dependent Sentiment Analysis

Since a simple and effective method to learn distributed representation was proposed (Mikolov et al., 2013), neural networks enhance target-dependent sentiment analysis significantly. Vo and Zhang (2015) split a tweet into a left context and a right context according to a given target, using distributed word representations and neural pooling functions to extract features. Tang et al. (2015) proposed TD-LSTM and TC-LSTM, where target information is automatically taken into account. These two models integrated the connections between target words and context words so as to significantly boost the classification accuracy. Zhang et al. (2016) proposed two gated neural networks, one was used to capture tweet-level syntactic and semantic information, and the other was used to model the interactions between the left context and the right context of a given target. With the gating mechanism, the target influenced the selection of sentiment signals over the context.

4.3 Attention Network for Aspect-level Sentiment Analysis

With the successful application of the attention mechanism in machine translation and reading comprehension, it is also applied to aspect-level sentiment analysis in recent years. Wang et al. (2016) examined the latent relatedness of the aspect term and sentiment polarity for aspect-level sentiment analysis. They designed an attention-based LSTM to learn aspect term embedding, and let the aspect term embedding participate in calculating the attention weights. Ma et al. (2017) proposed a new attention model IAN, which considered the separate modeling of aspect terms and could interactively learn attention in the contexts and aspect terms.

Despite the effectiveness of these attention mechanisms, they are coarse-grained and it is still challenging to identify different sentiment polarity at a fine-grained aspect level. However, our PBAN model makes full use of the position information of the aspect term, and PBAN uses a fine-grained bidirectional attention mechanism to model the mutual relationship between the sentence and each word in the aspect term, identifying the importance of the word in the aspect term to obtain a more effective sentence representation as described in Section 1.

5 Conclusion

In this paper, we have proposed a position-aware bidirectional network (PBAN) based on Bi-GRU for aspect-level sentiment analysis. The main idea of PBAN is to utilize the position embedding of aspect term for calculating the attention weights. Moreover, PBAN adopts a bidirectional attention mechanism, which is not only capable of mutually modeling the relation between sentence and different words in aspect term, but also takes advantage of the position information to better judge the sentiment polarity of aspect term. Experimental results on SemEval 2014 Datasets demonstrate that our proposed models can learn effective features and obtain superior performance over the baseline models.

Acknowledgements

This work is funded in part by the national key research and development program of China (2017YFE0111900), the Key Project of Tianjin Natural Science Foundation (15JCZDJC31100), the National Natural Science Foundation of China (Key Program, U1636203), the National Natural Science Foundation of China (U1736103) and MSCA-ITN-ETN - European Training Networks Project (QUARTZ).

References

Martn Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, and Michael Isard. 2016. Tensorflow: a system for large-scale machine learning.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. *arXiv preprint arXiv:1601.01705*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *Computer Science*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Ronan Collobert, Jason Weston, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(1):2493–2537.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423*.

Karl Moritz Hermann, Tom Koisk, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *International Conference on Neural Information Processing Systems*, pages 1693–1701.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 151–160. Association for Computational Linguistics.

Nobuhiro Kaji and Masaru Kitsuregawa. 2007. Building lexicon for sentiment analysis from massive collection of html documents. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273.

Baiyan Liu, Xiangdong An, and Jimmy Xiangji Huang. 2015. Using term location information to enhance probabilistic information retrieval. In *International Acm Sigir Conference on Research Development in Information Retrieval*, pages 883–886.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. *arXiv preprint arXiv:1709.00893*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Computer Science*.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212.

Xiaolei Niu, Yuexian Hou, and Panpan Wang. 2017. Bi-directional lstm with quantum attention mechanism for sentence modeling. In *International Conference on Neural Information Processing*, pages 178–188.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Vernica Prez-Rosas, Carmen Banea, and Rada Mihalcea. 2012. Learning sentiment lexicons in spanish. In *Eighth International Conference on Language Resources and Evaluation*.

Delip Rao and Deepak Ravichandran. 2010. Semi-supervised polarity lexicon induction. In *Eacl 2009, Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, March 30 - April3, 2009, Athens, Greece*, pages 675–682.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2015. Effective lstms for target-dependent sentiment classification. *arXiv preprint arXiv:1512.01100*.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. *arXiv preprint arXiv:1605.08900*.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. In *IJCAI*, pages 1347–1353.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *EMNLP*, pages 606–615.

D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao. 2014. Relation classification via convolutional deep neural network.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated neural networks for targeted sentiment analysis. In *AAAI*, pages 3087–3093.

Dynamic Feature Selection with Attention in Incremental Parsing

Ryosuke Kohita†, Hiroshi Noji§ and Yuji Matsumoto‡
†IBM Research
§Artificial Intelligence Research Center,
National Institute of Advanced Industrial Science and Technology (AIST)
†‡Graduate School of Information Science,
Nara Institute of Science and Technology (NAIST)
ryosuke.kohita1@ibm.com, hiroshi.noji@aist.go.jp, matsu@is.naist.jp

Abstract

One main challenge for incremental transition-based parsers, when future inputs are invisible, is to extract good features from a limited local context. In this work, we present a simple technique to maximally utilize the local features with an attention mechanism, which works as context-dependent dynamic feature selection. Our model learns, for example, which tokens should a parser focus on, to decide the next action. Our multilingual experiment shows its effectiveness across many languages. We also present an experiment with augmented test dataset and demonstrate it helps to understand the model's behavior on locally ambiguous points.

1 Introduction

This paper explores better feature representations for incremental dependency parsing. We focus on a system that builds a parse tree incrementally receiving each word of a sentence, which is crucial for interactive systems to achieve fast response or human-like behavior such as understanding from partial input (Baumann, 2013). The most natural way to achieve incremental parsing is using a transition system (Nivre, 2008), and for such parsers, the main challenge is to choose an appropriate action with only the local context information. While some recent transition-based parsers alleviate this difficulty by exploiting the entire input sentence with recurrent neural networks (Kiperwasser and Goldberg, 2016; Shi et al., 2017), one possible disadvantage is to require that all inputs are visible from the beginning, which should be problem when we try more strict incremental conditions such as simultaneous translation. Therefore there are still demands to explore the effective way to extract better feature representation from incomplete inputs.

In this paper, we incorporate a simple attention mechanism (Bahdanau et al., 2015) with an incremental parser and investigate its effectiveness during the feature extraction. Attention mechanism itself has firstly succeeded in machine translation, capturing relative importances of tokens on a certain step for a proper output (Bahdanau et al., 2015; Luong et al., 2015). The characteristic to weight on some features automatically and effectively can be applied to various tasks such as seq-to-seq parsing model (Vinyals et al., 2015), text summarization (Rush et al., 2015), dialogue generation (Shang et al., 2015), image captioning (Xu et al., 2015) in which the systems can enjoy performance gain by attending to specific clues depending on a given situation. We can also expect this behavior is helpful to fix the error which transition-based parsers often commits due to local ambiguities.

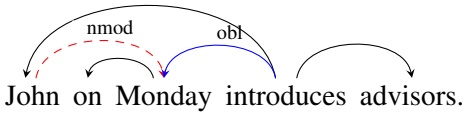


Figure 1: A locally ambiguous sentence. "*Monday*" should be analyzed as oblique of "*introduce*" while tends to be analyzed as a noun modifier of "*John*".

Proceedings of the 27th International Conference on Computational Linguistics, pages 785–794
Santa Fe, New Mexico, USA, August 20-26, 2018.

Figure 1 shows our motivating example, on which the standard transition-based parser fails and attaches "*Monday*" to "*John*", since on the POS level and the usual behavior of "*on*", this sequence is misleading as a typical noun phrase. By introducing attention on feature extraction, we expect the model to attend to important tokens, in this case "*Monday*", which is not likely to attach to a person and suggests the parser to anticipate the following predicate. Our technique can be applied to any models with feed-forward networks on concatenated feature embeddings, and in this work, we apply it on the standard transition-based parser of Chen and Manning (2014).

On the multilingual experiment on Universal Dependencies (UD) 2.0 (Zeman et al., 2017), we find our attention brings performance gain for most languages. To inspect the model's behavior, we also introduce a controlled experiment with manually created data. For this experiment, we prepare a set of sentences for which the parser must attend to the key points for correct disambiguation, as in Figure 1, and see whether the model behaves as expected. There we give detailed error analysis to suggest what makes it difficult to solve the local ambiguities and how attention achieves it. This type of analysis is common in psycholinguistics (Levy, 2008), and a similar idea has recently begun to be explored in NLP neural models (Shekhar et al., 2017).

2 Model

2.1 Base model

Our base model is a transition-based neural parser of Chen and Manning (2014).[1] For each step, this parser first creates feature vectors of words ($\mathbf{x}^w$), POS tags ($\mathbf{x}^p$), and labels ($\mathbf{x}^l$), each of which is a concatenation of embeddings around a stack and a buffer. These vectors are transformed with corresponding weights, i.e., $\mathbf{h} = \mathbf{W}^w\mathbf{x}^w + \mathbf{W}^p\mathbf{x}^p + \mathbf{W}^l\mathbf{x}^l + \mathbf{b}$, followed by nonlinearity. A next softmax layer then provides action probabilities.

Although this method is actually old, the approach which creates the feature vector from independent embeddings becomes useful in our second experiment inspecting our attention behaviors (See section 3.3 in detail). In addition, UDPipe (Straka et al., 2016) which is the baseline parser in the latest shared task (Zeman et al., 2017) also adopts this approach and holds good performance compared to others using recent techniques.

2.2 Attention on local features

We introduce attention in feature computation from the input embeddings to $\mathbf{h}$. Note that three components $\mathbf{W}^w\mathbf{x}^w$, $\mathbf{W}^p\mathbf{w}^p$, and $\mathbf{W}^l\mathbf{x}^l$ are independent; in the following we focus on just one part, abstracted by $\mathbf{W}\mathbf{x}$, and describe how attention is applied for this computation.

Our attention calculates the importance of input elements. First, note that $\mathbf{x}$ is a concatenation of embeddings of input elements, and when the number of elements is n, $\mathbf{W}$ can also be divided into n blocks as in Figure 2. When these parts are denoted by $\mathbf{W}_i$ and $\mathbf{x}_i$, $\mathbf{W}\mathbf{x} = \sum_i \mathbf{W}_i\mathbf{x}_i$ holds. We define $\mathbf{c}_i = \mathbf{W}_i\mathbf{x}_i$, which corresponds to the hidden representation for the i-th input element.

Our core idea is to apply attention on decomposed hidden vectors $\{\mathbf{c}_i\}$. Using attention vector $\mathbf{a} = (a_1, a_2, \cdots, a_n)$, the new hidden representation becomes $\mathbf{h}_g = \sum_i a_i\mathbf{c}_i$. We obtain attention a_i using $\mathbf{c}_i$ and parameters $\mathbf{q}$ as follows:

$$a_i = \frac{\exp(\sigma(\mathbf{q} \cdot \mathbf{c}_i))}{\sum_{i=1}^{n} \exp(\sigma(\mathbf{q} \cdot \mathbf{c}_i))},$$

where σ is a sigmoid function. We use different attention parameters $\mathbf{q}^w$, $\mathbf{q}^p$, and $\mathbf{q}^l$ for word, POS, and label inputs, respectively.

3 Experiments

Our first experiment is on the multilingual UD treebanks used in CoNLL 2017 shared task (Zeman et al., 2017). In addition to this, we present another experiment using augmented test data. This is a set of

[1] As described in Section 3.1 we slightly extend their parser to use additional features. In this section, we first present our model with the original features for simplicity.

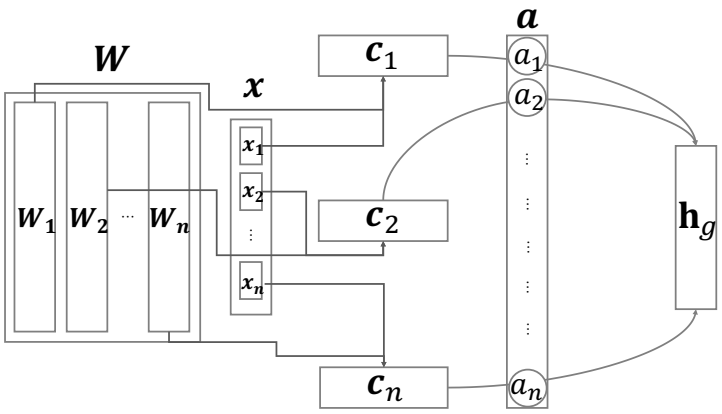


Figure 2: Our attention mechanism on the decomposed hidden vectors $\mathbf{c}_i$, obtained by $\mathbf{W}_i\mathbf{x}_i$.

sentences for which there is a key token for correct disambiguation. We will see whether our model is capable of disambiguating them by attending to the critical points.

For both experiments, our baselines are our parser without attention, and UDPipe v1.1 (Straka et al., 2016), which was the state-of-the-art among transition-based parsers with local features on the shared task.[2]

3.1 Parser

We extract features from the same positions as Chen and Manning (2014); top three tokens from the stack and the buffer, the first and second leftmost or rightmost children of the top two tokens on the stack, and the leftmost or rightmost children of leftmost or rightmost children of the top two tokens on the stack. However, from each position we extract more information such as LEMMA (see also footnote 1). The embedding sizes are: 50 dimensions for WORD, 20 dimensions for LEMMA, UPOS, XPOS, FEATS, and DEPREL. We also extract 32 dimensional character encoding of a token by bi-LSTMs (Ling et al., 2015), though we do not apply attention on this. The size of the hidden dimension is 200, on which we apply 50% dropout. We use pre-trained embeddings used in the baseline UDPipe.[3] To handle non-projectivity, we employ the arc-standard swap algorithm (Nivre et al., 2009). We also use beam search with width 5. To learn the representation for unknown words, we stochastically replace singletons with the dummy token (Dyer et al., 2015). These hyperparameters are the same across languages except Kazakh. This is apart from UDPipe, which tunes the setting for each language. For Kazakh, which is extremely small, we find increasing the model size as 100, 50, and 50 dimensions for WORD, UPOS, and XPOS embeddings works well so we choose this setting.

3.2 Multilingual evaluation

We use 63 treebanks in 45 languages on Universal Dependencies v2.0 (Nivre et al., 2017), with the same data split as the setting of official UDPipe.[3] We evaluate F1 LAS of each treebank and their macro average. For the development sets, we use the gold preprocessed data while for the test sets, we parse the raw text preprocessed by UDPipe.

With respect to the macro averaged score, in the Table 1 below, we can see that our model without attention (w/o Att.) is comparable to UDPipe; with attention, it outperforms both. When inspecting in detail, we see that our attention improves the scores on 54 treebanks on the development set and 57 treebanks on the test set. We also see that the treebanks for which our attention degrades the performance are relatively small, e.g., `en_partut` (1,035 sentences) and `hu` (864 sentences), which indicates our attention may be more data-hungry.

[2]There are three systems (Straka and Straková, 2017; Kanerva et al., 2017; Yu et al., 2017) that outperform UDPipe v1.1 but the improvements come not from parsing models but from preprocessing, such as improvements to the POS tagger.

[3]https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1990

	Development			Test		
Treebank	UDPipe	w/o Att.	w/ Att.	UDPipe	w/o Att.	w/ Att.
ar	78.11	78.73	**79.84**	65.30	64.72	**65.43**
bg	**87.56**	86.90	87.35	**83.64**	83.23	83.44
ca	**88.35**	87.52	88.28	85.39	84.66	**85.43**
cs	**88.19**	86.29	87.06	**82.87**	81.15	82.27
cs_cac	86.57	86.13	**87.01**	**82.46**	81.48	82.15
cs_cltt	78.95	**79.96**	79.22	71.64	72.08	**72.56**
cu	79.44	79.46	**81.69**	62.76	63.19	**65.40**
da	81.13	80.57	**82.01**	73.38	73.16	**74.22**
de	84.06	83.58	**84.25**	**69.11**	67.54	68.66
el	83.71	**84.59**	83.88	79.26	79.56	**80.03**
en	**85.82**	84.96	85.29	**75.84**	74.65	75.06
en_lines	**80.51**	80.40	80.46	72.94	73.75	**74.11**
en_partut	81.29	**81.49**	79.95	**73.64**	73.37	73.15
es	**86.69**	86.17	86.66	81.47	80.55	**81.58**
es_ancora	87.55	86.98	**87.89**	83.78	83.59	**84.39**
et	**76.37**	74.00	75.63	**58.79**	57.62	58.74
eu	76.88	76.31	**77.93**	69.15	68.24	**70.29**
fa	**85.16**	82.69	83.60	**79.24**	77.20	78.28
fi	82.12	81.83	**83.10**	73.75	73.73	**74.73**
fi_ftb	85.14	84.70	**86.20**	74.03	73.45	**74.54**
fr	**89.02**	87.94	88.82	**80.75**	79.87	80.70
fr_partut	80.61	78.81	**82.42**	77.38	77.62	**78.08**
fr_sequoia	**86.66**	**86.66**	86.60	79.98	80.00	**80.29**
ga	71.09	70.49	**72.75**	61.52	62.37	**62.62**
gl	80.55	81.16	**82.22**	77.31	77.82	**78.71**
gl_treegal	74.48	**75.46**	75.13	**65.82**	65.06	65.30
got	76.51	77.32	**77.86**	59.81	60.16	**60.80**
grc	61.65	62.80	**65.51**	**56.04**	54.83	55.66
grc_proiel	75.72	74.58	**76.78**	65.22	64.80	**66.79**
he	**83.18**	81.94	82.87	**57.23**	55.13	55.07
hi	91.07	91.72	**92.15**	**86.77**	86.02	86.46
hr	80.76	79.46	**81.17**	77.18	76.35	**77.59**
hu	73.98	**75.42**	75.36	**64.30**	64.23	64.01
id	78.43	78.24	**79.15**	74.61	74.41	**75.31**
it	88.44	87.27	**88.89**	**85.28**	84.47	85.20
it_partut[a]	**85.16**	84.20	83.85	-	-	-
ja	**95.48**	95.28	95.23	72.21	72.68	**72.69**
kk	34.83	**37.08**	22.47	24.51	**25.14**	22.77
ko	62.06	79.28	**80.10**	59.09	73.52	**74.38**
la	60.04	61.44	**63.11**	43.77	43.78	**46.51**
la_ittb	77.91	77.01	**79.98**	**76.98**	75.78	76.67
la_proiel	74.36	72.48	**75.06**	57.54	57.11	**58.28**
lv	72.71	72.58	**73.37**	59.95	58.65	**60.13**
nl	82.43	81.85	**83.51**	68.90	68.02	**68.93**
nl_lassysmall	80.34	79.22	**80.61**	78.15	76.15	**78.86**
no_bokmaal	**88.78**	87.54	88.70	**83.27**	81.61	82.71
no_nynorsk	87.99	87.56	**88.04**	**81.56**	80.51	80.94
pl	**87.35**	87.79	86.66	78.78	**78.99**	78.65
pt	89.45	92.10	**92.59**	**82.11**	78.79	78.91
pt_br	89.57	88.97	**89.58**	**85.36**	84.91	85.35
ro	82.25	81.80	**82.59**	79.88	78.93	**80.07**
ru	80.84	81.79	**82.53**	74.03	74.79	**75.36**
ru_syntagrus	**89.63**	88.10	89.38	**86.76**	85.55	86.54
sk	83.83	82.95	**84.13**	72.75	72.14	**73.66**
sl	89.15	89.18	**90.05**	81.15	80.06	**81.18**
sl_sst	67.31	66.81	**68.09**	46.45	46.05	**46.50**
sv	80.40	78.94	**80.91**	**76.73**	76.11	76.32
sv_lines	81.38	81.07	**81.73**	**74.29**	73.62	74.07
tr	60.27	59.45	**61.48**	53.19	54.50	**55.50**
ug	**53.85**	58.65	49.04	34.18	**36.26**	35.62
uk	69.30	70.61	**70.68**	60.76	60.91	**61.13**
ur	81.62	85.47	**85.68**	76.69	76.36	**76.98**
vi	66.22	68.13	**69.27**	37.47	37.85	**38.10**
zh	**79.37**	77.45	78.21	**57.40**	56.18	56.22
avg.	79.52	79.65	**80.18**	70.34 [b]	70.06	**70.79**

[a] The test set of it_partut treebank was excluded in the shared task as well.
[b] The UDPipe's official score is 68.35 because it includes the scores for extra treebanks in the shared task, called *surprise language*.

Table 1: Labeled attachment scores of 63 treebanks in UD v2.0

3.3 Augmented data evaluation

Why does attention help for disambiguation? To inspect this, now we perform a controlled experiment by parsing a set of sentences that for correct disambiguation may require attending to some specific points. We present two different sets on English, which differ in the points where the model should attend.

Oblique vs. noun modifier The first set is related to the difficulty of the left of Figure 3, where as we discussed the parser may be confused and attach "*Monday*" to "*John*" as a noun modifier since the POS sequence of "*John on Monday*" and the usual behavior of "*on*" are typical for a noun phrase. The right of Figure 3 shows the step where the parser must decide the head of "*Monday*"; here the correct action is shift and right-arc leads to the wrong analysis. At this step, though the important token for a typical NP is "*on*", we expect the parser to focus more on "*Monday*", which is likely to attach to a subsequent predicate as oblique.

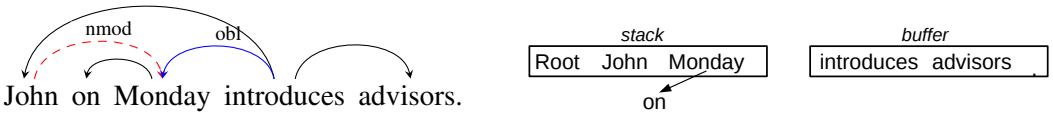


Figure 3: Left - An local ambiguous sentence, reprint of Figure 1. Right - The configuration on which the parser must decide whether "*Monday*" works as an oblique or a modifier.

To inspect the model's ability for correctly handling these ambiguities, we prepare 14 pairs of sentences.[4] Each pair differs minimally, as in "*John on Monday introduces advisors*" and "*John on a balcony introduces advisors*", in which the former should be analyzed as oblique (obl) while the latter as a modifier (nmod). Table 2 contrasts the inputs to parsers when gold preprocessing is given, where the differences always appear at third and forth tokens ("*Monday*" in obl vs. "*a*" and "*balcony*" in nmod). All words in these items occur at least one in the training corpus, therefore no unknown words are used.

	1	2	3	4	5	6	7
obl	John	on	Monday		introduces	advisors	.
	PROPN	ADP	PROPN		VERB	NOUN	PUNCT
	NNP	IN	NNP		VBZ	NNS	.
	Sing	_	Sing		Ind	Plur	_
nmod	John	on	a	balcony	introduces	advisors	.
	PROPN	ADP	DET	NOUN	VERB	NOUN	PUNCT
	NNP	IN	DT	NN	VBZ	NNS	.
	Sing	_	Ind	Sing	Ind	Plur	_

Table 2: Minimal pair in oblique ("*John on Monday introduces advisors*") vs. noun modifier ("*John on a balcony introduces advisors*") experiment

The result is summarized in Table 3, where we count the number of sentences on which the parser outputs are perfect. We can see that nmod sentences are analyzed near perfectly, which is intuitive as this structure is typical. obl sentences are more difficult, but the system with attention is capable of handling them. The other systems fail, even assuming gold tags. For pred tags, all systems receive the same inputs tagged by UDPipe. The accuracy for obl decreases, and we find the errors are due to incorrect POS tags for the predicate at 5th word, which are sometimes tagged as a noun. This suggests our attention parser can handle these local ambiguities unless a crucial tag error occurs, while the other systems cannot at all.

Finally, we show in Figure 4 the attention weights on features at a branching step (The right of Figure 3) for the sample sentences in Table 2. We can see that for the obl sentence the parser attends more

[4] All items are shown in Appendix A.

	Gold tags		Pred tags	
	obl	nmod	obl	nmod
UDPipe	0 / 14	14 / 14	0 / 14	13 / 14
w/o Att.	0 / 14	14 / 14	0 / 14	13 / 14
w/ Att.	**12** / 14	14 / 14	**6** / 14	13 / 14

Table 3: # of correct analysis for obl vs. nmod pairs.

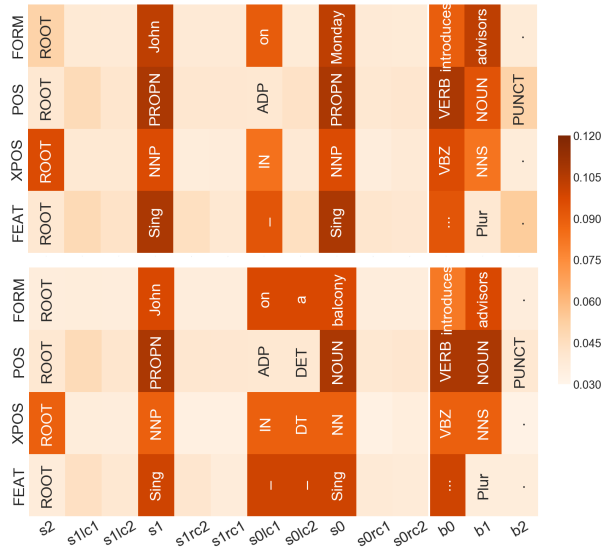


Figure 4: Attention weights on obl sentence (above) and nmod sentence (below). s_i and b_i are the i-th top-most position on the stack and buffer, respectively. lc_i and rc_i are their (inward) i-th left and right child.

on the key tokens of "*Monday*" on the stack and "*introduces*" on the buffer. This suggests the attention mechanism works as we expected and its behavior matches our intuition.

Object complement vs. that clause The second set is about different ambiguities from the previous experiment; an example pair is shown in the left of Figure 5, where to correctly parse the lower one, the parser must recognize the implicit that clause (that), rather than an object-complement (oc). The right of the figure shows the configuration on which the parser must choose the structure, by shift or right-arc. The key token for correct analysis is at the last, which can be accessed as the second token on the buffer.

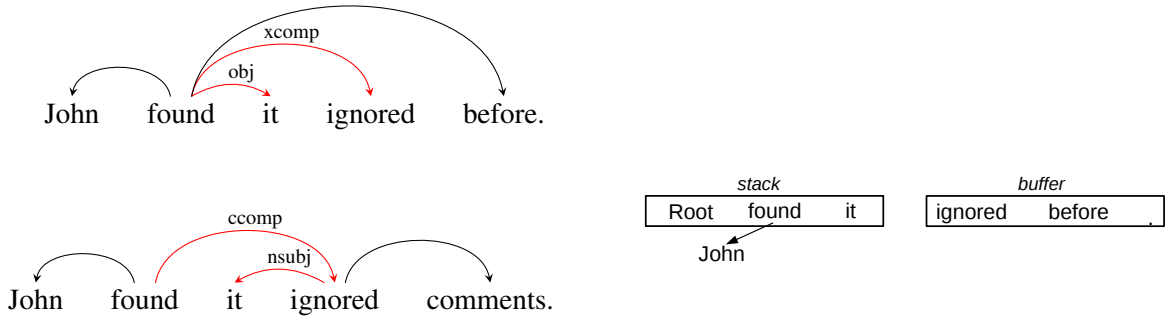


Figure 5: The representative pair for the second set: object-complement (left above) vs. that clause (left below) and the branching configuration (right).

We prepare 24 pairs of sentences. Table 4 shows an example of differences of a pair. In these sentences, tokens from third to fifth differ. Note that contrary to Table 2 these two condition are distinctive with

POS tags (e.g., VBN or VBD), so the main challenge is whether the model can attend to the key tokens when the predicted noisy tags are used.

	1	2	3	4	5	6
oc	John	found	it	ignored	before	.
	PROPN	VERB	PRON	VERB	ADV	PUNCT
	NNP	VBD	PRP	VBN	RB	.
	Sing	Ind	Acc	Past	_	_
that	John	found	it	ignored	comments	.
	PROPN	VERB	PRON	VERB	NOUN	PUNCT
	NNP	VBD	PRP	VBD	NNS	.
	Sing	Ind	Nom	Ind	Plur	_

Table 4: Minimal pair in object-complement ("*John found it ignored before*") vs. that-clause experiment ("*John found it ignored comments*")

Table 5 summarizes the results. As we expected, all systems succeed with gold tags, but perform badly in particular on that sentences, with predicted tags. Inspecting errors, we find that this is due to error propagation from an incorrect tag for *it* (3rd token), on which UDPipe assigns Acc(suative) feature due to that-omission. By this error, another error is induced on the POS tag of the next token (e.g., *ignored*), which becomes participle or adjective. These erroneous tags make it hard for parsers to recognize an implicit *that*.

Though all models fail, we notice that for 30% of sentences (7/24), our attention parser recognizes the existence of that-clause, by wrongly analyzing the last noun (e.g., comments) as the head of the clause (*it* becomes nsubj of the noun). Inspecting the attention weights for succeeded and failed cases (Figure 6), we find the last noun is slightly attended more in the succeeded case (above), which may lead the parser to predict a ccomp arc (but to a wrong word).

	Gold tags		Pred tags	
	oc	that	oc	that
UDPipe	23 / 24	24 / 24	19 / 24	0 (0) / 24
w/o Att.	24 / 24	24 / 24	16 / 24	1 (2) / 24
w/ Att.	23 / 24	24 / 24	18 / 24	1 (7) / 24

Table 5: # of correct sentences for oc vs. that. Numbers in brackets mean the cases where that-omission is correctly predicted but other errors exist (see body).

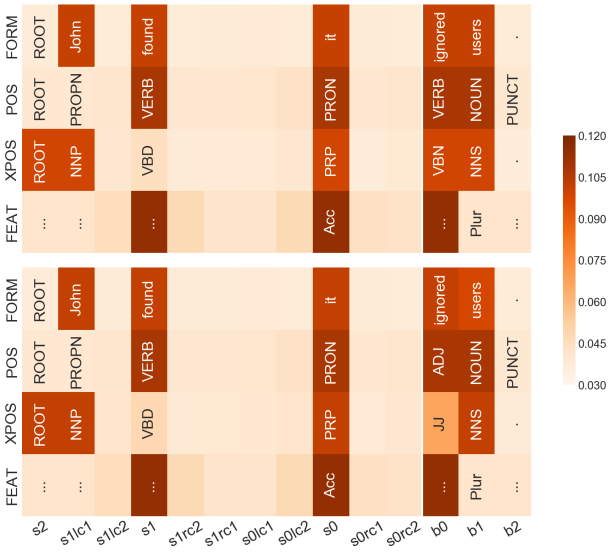


Figure 6: Attention weights on that sentences when that-omission is predicted (above) or failed (below).

4 Conclusion

We have presented an simple attention mechanism for dynamic feature selection, which can be applied to any feed-forward networks on concatenated feature embeddings. When applying to an incremental parser, the parser performance increased across many languages. Also our augmented-data experiment showed that the parser successfully learns where to focus on each context, and becomes more robust to erroneously tagged sentences.

References

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

Timo Baumann. 2013. *Incremental Spoken Dialogue Processing: Architecture and Lower-level Components*. Ph.D. thesis.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October. Association for Computational Linguistics.

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

Jenna Kanerva, Juhani Luotolahti, and Filip Ginter. 2017. Turkunlp: Delexicalized pre-training of word embeddings for dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 119–125, Vancouver, Canada, August. Association for Computational Linguistics.

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

Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106(3):11261177.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon, Portugal, September. Association for Computational Linguistics.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421, Lisbon, Portugal, September. Association for Computational Linguistics.

Joakim Nivre, Marco Kuhlmann, and Johan Hall. 2009. An improved oracle for dependency parsing with online reordering. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT)*, pages 73–76.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Cristina Bosco, Gosse Bouma, Sam Bowman, Marie Candito, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Fabricio Chalub, Jinho Choi, Çağrı Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Linh Hà Mỹ, Dag Haug, Barbora Hladká, Petter Hohle, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Natalia Kotsyba, Simon Krek, Veronika Laippala, Phng Lê H`ông, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Lng Nguy˜ên Thị, Huy`ên Nguy˜ên Thị Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Lilja

Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Livy Real, Laura Rituma, Rudolf Rosa, Shadi Saleh, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jonathan North Washington, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–554.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal, September. Association for Computational Linguistics.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1577–1586, Beijing, China, July. Association for Computational Linguistics.

Ravi Shekhar, Sandro Pezzelle, Yauhen Klimovich, Aurélie Herbelot, Moin Nabi, Enver Sangineto, and Raffaella Bernardi. 2017. Foil it! find one mismatch between image and language caption. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 255–265, Vancouver, Canada, July. Association for Computational Linguistics.

Tianze Shi, Liang Huang, and Lillian Lee. 2017. Fast(er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 12–23, Copenhagen, Denmark, September. Association for Computational Linguistics.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada, August. Association for Computational Linguistics.

Milan Straka, Jan Hajic, and Jana Strakov. 2016. Udpipe: Trainable pipeline for processing conll-u files performing tokenization, morphological analysis, pos tagging and parsing. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Oriol Vinyals, Ł ukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2773–2781. Curran Associates, Inc.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention.

Kuan Yu, Pavel Sofroniev, Erik Schill, and Erhard Hinrichs. 2017. The parse is darc and full of errors: Universal dependency parsing with transition-based and graph-based algorithms. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 126–133, Vancouver, Canada, August. Association for Computational Linguistics.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria dePaiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung

Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19. Association for Computational Linguistics.

A Experiment Items

No.	Type	Sentence
1	obl	John on Monday agrees friends.
	nmod	John on a table agrees friends.
2	obl	John on Monday needs colleagues.
	nmod	John on a chair needs colleagues.
3	obl	John on Tuesday introduces advisors.
	nmod	John on a balcony introduces advisors.
4	obl	John on Tuesday calls leaders.
	nmod	John on a bike calls leaders.
5	obl	John on Wednesday employs people.
	nmod	John on a ship employs people.
6	obl	John on Wednesday meets relatives.
	nmod	John on a bus meets relatives.
7	obl	John on Thursday sees workers.
	nmod	John on a stage sees workers.
8	obl	John on Thursday praises owners.
	nmod	John on a roof praises owners.
9	obl	John on Friday believes teachers.
	nmod	John on a seat believes teachers.
10	obl	John on Friday hits professors.
	nmod	John on a car hits professors.
11	obl	John on Saturday protects soldiers.
	nmod	John on a tank protects soldiers.
12	obl	John on Saturday supports doctors.
	nmod	John on a hill supports doctors.
13	obl	John on Sunday worries visitors.
	nmod	John on a mountain worries visitors.
14	obl	John on Sunday contacts managers.
	nmod	John on a plane contacts managers.

Table 6: Items in the oblique vs. noun modifier experiment.

No.	Type	Sentence
1	oc	John found it ignored before.
	that	John found it ignored comments.
2	oc	John found it ignored again.
	that	John found it ignored opinions.
3	oc	John found it contained before.
	that	John found it contained layers.
4	oc	John found it contained again.
	that	John found it contained plants.
5	oc	John considered it classified before.
	that	John considered it classified species.
6	oc	John considered it classified again.
	that	John considered it classified words.
7	oc	John considered it involved before.
	that	John considered it involved issues.
8	oc	John considered it involved again.
	that	John considered it involved changes.
9	oc	John felt it abandoned before.
	that	John felt it abandoned soldiers.
10	oc	John felt it abandoned again.
	that	John felt it abandoned people.
11	oc	John felt it protected before.
	that	John felt it protected ideas.
12	oc	John felt it protected again.
	that	John felt it protected students.
13	oc	John guessed it reccommended before.
	that	John guessed it reccommended graphics.
14	oc	John guessed it reccommended again.
	that	John guessed it reccommended services.
15	oc	John guessed it employed before.
	that	John guessed it employed officials.
16	oc	John guessed it employed again.
	that	John guessed it employed relatives.
17	oc	John understood it infected before.
	that	John understood it infected computers.
18	oc	John understood it infected again.
	that	John understood it infected animals.
19	oc	John understood it pasted before.
	that	John understood it pasted pictures.
20	oc	John understood it pasted again.
	that	John understood it pasted posters.
21	oc	John believed it transmitted before.
	that	John believed it transmitted signals.
22	oc	John believed it transmitted again.
	that	John believed it transmitted images.
23	oc	John believed it replaced before.
	that	John believed it replaced lights.
24	oc	John believed it replaced again.
	that	John believed it replaced positions.

Table 7: Items in the object complement vs. that clause experiment.

Vocabulary Tailored Summary Generation

Kundan Krishna
Adobe Research
kunkrish@adobe.com

Aniket Murhekar
IIT Bombay
aniket1602@gmail.com

Saumitra Sharma
IIT Guwahati
sharmasaumitra15@gmail.com

Balaji Vasan Srinivasan
Adobe Research
balsrini@adobe.com

Abstract

Neural sequence-to-sequence models have been successfully extended for summary generation. However, existing frameworks generate a single summary for a given input and do not tune the summaries towards any additional constraints/preferences. Such a tunable framework is desirable to account for linguistic preferences of the specific audience who will consume the summary. In this paper, we propose a neural framework to generate summaries constrained to vocabulary-defined linguistic preferences of a target audience. The proposed method accounts for the generation context by tuning the summary words at the time of generation. Our evaluations indicate that the proposed approach tunes summaries to the target vocabulary while still maintaining a superior summary quality against a state-of-the-art word embedding based lexical substitution algorithm, suggesting the feasibility of the proposed approach. We demonstrate two applications of the proposed approach - to generate understandable summaries with simpler words, and readable summaries with shorter words.

1 Introduction

Automatic text summarization (Nenkova and McKeown, 2011) is the task of generating summaries of an input document while retaining the important points. These summaries are used for presenting the most relevant and important information in a long text in a succinct form. They are useful in places where a quick consumption of the information in a long article is preferred. Earlier works in summarization select sentences/textual units from the input article and put them together into an "extractive" summary. However, humans summarize an article by understanding the content and paraphrasing the understood content into the desired summary. Therefore, extractive summarization is unable to produce "human-like" summaries. This has led to efforts towards "abstractive" summarization which paraphrases summaries the input article. Several models have been proposed, with the most recent ones based on neural networks.

Often, it is desirable to tune the summaries to the linguistic preferences of the readers. For example, a medical report may contain a lot of technical jargon beyond the understanding of the common population. When such a report is consumed by a patient, it makes sense to use lesser jargon to suit a patient's knowledge level. Similarly, while reading articles, teenagers would prefer more informal and trendy words, while older people might like a more formal vocabulary. Summaries which are tuned to such "linguistic" preferences of the target population segment are likely to appeal better and catch their attention.

A standard approach to incorporate vocabulary tuning would be to post-process a generated summary to achieve the desired goal by replacing a few words (e.g. replacing words with their simpler alternatives). However, such an approach might not preserve the context and hence can result in a complete change in the meaning of the content. Consider the sentence *"The baseball **pitcher** was seen with a **pitcher** of beer in his hand."* The word *pitcher* means different things in its two occurrences here. The sentence can be rephrased as *"The baseball **player** was seen with a **jug** of beer in his hand."* Since *pitcher* can mean both *player* and *jug* in two independent senses, it is not easy to decide the right replacement without looking

Proceedings of the 27th International Conference on Computational Linguistics, pages 795–805
Santa Fe, New Mexico, USA, August 20-26, 2018.

at the respective context. Post-processing based approaches lose the contextual information from the source article once the summary has been constructed. This is because the algorithm only sees the short snippet of text (summary), and not the large source article. This is a lost opportunity to utilize all that contextual information to make better word substitutions.

In this paper, we propose a neural network based summary generator that generates summaries imposing the desired vocabulary preferences while also maintaining the context and meaning of the content. Neural network based summarizers encode the entire source article, and generate the summary word-by-word. They are trained to predict the next word in a summary given the words in the summary generated so far, as well as the encoded article. This will allow the network to better judge which words will fit in context at any position in the sentence. In our proposed approach, we modify the generation probability of the next word in accordance with the vocabulary-based preferences. A key advantage of the proposed approach is that it does not require re-training a neural network, and relies on modifying the summary generation procedure on a pre-trained network.

2 Related Work

Traditional methods for summarization (Nenkova and McKeown, 2011) extract key sentences from the source text to construct the summary based on various features like descriptiveness of words, word frequencies, etc. Early attempts at abstractive summarization created summary sentences based on templates (Wang and Cardie, 2013; Genest and Lapalme, 2011) or used ILP-based sentence compression to collect parts from various sentences to generate the summary (Filippova, 2010; Berg-Kirkpatrick et al., 2011; Banerjee et al., 2015).

With the advent of deep sequence to sequence networks (Sutskever et al., 2014), attention based models have been proposed for summarizing long sentences (Rush et al., 2015; Chopra et al., 2016). Gulcehre et al. (2016) incorporated the ability to copy out-of-vocabulary words from the article to incorporate rarely seen words like names in the generated text. Tu et al. (2016) included the concept of coverage, to prevent the models from repeating the same phrases while generating a sentence. See et al. (2017) proposed a summarization model which incorporates these improvements, and also learns to switch between generating new words and copying words from the source article. We use this summarization framework as the starting point for our work. However, none of the existing works attempt to tune the summaries to suit preferences of a reader.

One naive way to solve this problem could be to impose the preferences after the summary generation as a post-processing step. As we will show later, this can result in out-of-context replacements while tuning. The words can be substituted based on a standard thesaurus (Bott et al., 2012) using one of the synonyms of the target word. However, since a word can be used in multiple senses, not all of its synonyms can be used in its place, and therefore such an approach is prone to errors. We address these challenges in the proposed approach by optimizing for the vocabulary preferences at the time of summary generation by looking for potential replacements in the synonyms weighted by their contribution to the context (computed from the attention models). As we will show later, such an approach generates better quality summaries and reduces substitution errors.

Lexical substitution deals with deciding textual substitutions that will preserve the meaning and grammatical correctness of the sentence by modeling the overall sentence context and using it for word substitutions. Early methods used co-occurrence statistics of the possible substitutions and the context words to predict whether a substitution is valid in a given context (Szarvas et al., 2013).

Melamud et al. (2015) use the proximity of words in an embedding space to measure the appropriateness of a candidate substitution to replace the target word. They use the embeddings of the dependency relations (De Marneffe and Manning, 2008) of the target word in the same embedding space called 'context embeddings'. The cosine similarity between the embedding of a candidate substitution with embeddings of these dependency relations along with the target was shown to improve substitution performance. Roller and Erk (2016) extended this work further by incorporating a linear transformation of the context embeddings and learning the parameters of the transformation to rank possible substitutions.

These methods work on the hypothesis that words closer to the target word in the embedding space are

its viable replacements. We believe that this hypothesis might not always hold. While it is true that the proximity of word embeddings of two words implies their usage in similar contexts (similar neighboring words or dependency relations), two different words having opposite meanings can also occur nearby in the word embedding space. For example, *good* and *bad* have very close embeddings in the space trained on the Google News corpus[1] by Mikolov et al. (2013), because both are adjectives and used around similar contexts. In this embedding space, the cosine similarity between *good* and *bad* is 0.72, whereas similarity between *good* and *wonderful* is 0.57. However, replacing good with bad will certainly change the meaning of the sentence and despite having a lower similarity, *wonderful* is the better substitute in most cases. Our method does not suffer from such incorrect substitutions because we couple the information from a thesaurus with the contextual information from the neural decoder to generate the summary by picking the appropriate words. We show that compared to contextual word vectors, the neural decoder is able to capture the context better and so our method generates better summaries.

Another related line of work is **text adaptation** which deals with modification of the textual content to suit the needs of a particular audience segment. Text simplification (Paetzold and Specia, 2015; Paetzold and Specia, 2016) is a popular variant of text adaptation where the objective is to modify text to have simpler words so that it is easier to comprehend. Linguistic personalization is another variant of the problem which looks at modifying messages to suit a target segment's linguistic style (Roy et al., 2015). However, all these approaches adapt the text as a post-processing task, and hence do not account for the context with which the text was generated. Our proposed approach is generic and can be extended to address these variants of text adaptation while accounting for the context of generation. In particular, we show the application to the tasks of text simplification and text readability enhancement.

3 Summary Generation with Vocabulary Tuning (VoTing)

Given an input text article as a sequence of n tokens $A = a_1, a_2, ..., a_n$, a vocabulary $V = \{w_1, w_2, ..., w_k\}$ of words with scores indicating the preference of each word given by $q : V \rightarrow \mathbb{R}^+$, the objective is to generate a summary as a sequence of tokens $S = s_1, s_2, ..., s_m$ tuned to the preferences indicated by the vocabulary while preserving the contextual sense.

We extend the **pointer generator network** by See et al. (2017) to generate the summary in a word-by-word fashion. At each step, the algorithm runs a trained decoder neural network to output the probability of each word $w \in V$ being the next generated word. This generation probability of any word is also an indicator of its contextual appropriateness in the current generation.

Our primary contribution in this paper is a **modified decoding algorithm** to incorporate vocabulary preferences. At each decoding step of the trained neural network, instead of adding the word w having the highest generation probability to the summary, we tune it by replacing it with a better preferred word that is also contextually appropriate. We take all synonyms of w and score their contextual appropriateness based on their generation probabilities. We combine the contextual appropriateness with the "vocabulary" scores of the synonyms based on the vocabulary metric q (e.g. simplicity) to select the contextually best candidate that is preferred in the vocabulary, and append it to the summary. Iteratively repeating this builds the complete summary.

3.1 Pointer Generator Network

For the sake of completeness and introducing the notations, we first give an overview of the pointer generator architecture (See et al., 2017) before introducing our vocabulary tuning approach in Section 3.2. The pointer generator network consists of an encoder and a decoder, both based on LSTM architecture. Given an input article, the encoder takes the word embedding vectors of the source text $A = a_1 a_2 ... a_n$ and computes a sequence of encoder hidden states $h_1, h_2, ..h_n$. The final hidden state is passed to a decoder. The decoder computes a hidden state s_t at each decoding time step, and an attention distribution a^t is over all words in the source text,

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + b_{att});\ a^t = \text{softmax}(e^t) \quad (1)$$

[1]Pretrained word embeddings available at `https://code.google.com/archive/p/word2vec/`

where v, W_h, W_s and b_{att} are trained model parameters. The attention model is a probability distribution over the words in the source text, which aids the decoder in generating the next word in the summary using words with higher attention. The context vector h_t^* is a weighted sum of the encoder hidden states and is used to determine the next word that is to be generated.

$$h_t^* = \sum_{i=1}^{n} a_i^t h_i, \tag{2}$$

At each decoding time step, the decoder uses the last word y_t in the summary generated so far and computes a scalar p_{gen} denoting the probability of the network generating a new word from the vocabulary.

$$p_{gen} = \sigma(w_h^T h_t^* + w_s^T s_t + w_y^T y_t + b_{gen}) \tag{3}$$

where w_h, w_s, w_y, b_{gen} are trained vectors. The network probabilistically decides based on p_{gen}, whether to generate a new word from the vocabulary or copy a word from the source text using the attention distribution. For each word w in the vocabulary, the model calculates $P_{vocab}(w)$, the probability of the word getting newly generated next. P_{vocab} is calculated by passing a concatenation s_t and h_t^* through a linear transformation with softmax activation. On the other hand, for each word w' in the input article, its total attention received yields its probability of being copied. The total probability of w being the next word generated in the summary, denoted by $\mathbf{p}$ is given by,

$$\mathbf{p}(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} a_i^t \tag{4}$$

The second term allows the framework to choose a word to copy from the input text using the attention distribution. For our experiments, we utilized the model trained using back-propagation and the Adagrad gradient descent algorithm (Duchi et al., 2011).

3.2 Vocabulary Tuning (VoTing)

We assume that there exists a scoring function $q(w)$ that computes a quality/preference score of a word w in the target vocabulary. The preference can be along different criteria like simplicity, readability, commonality etc. There have been several approaches explored (Paetzold and Specia, 2015) to compute such a score for a target corpus/vocabulary, e.g. normalized word counts in the target vocabulary. Our primary contribution is a method to integrate such preference scores with the generation process for tuning the generated summary.

We encourage the decoder to pick a different word w' in place of w if it has a higher preference score and similar contextual appropriateness ($\mathbf{p}()$). To preserve the meaning in the generation, we restrict the possible replacements to synonyms of w, $w' \in Syn(w)$, where $Syn(w)$ represents the set of synonyms of w. Since the replacement is done to tune the summary to the target vocabulary, it implicitly attempts to maximize the aggregated quality/preference score of the generated summary. We therefore, define the probability of replacement $p(w'|w)$ to be non-zero only when the new word(w') has a higher quality/preference score than the old one(w). We calculate $p(w'|w)$ for each pair of words (w', w) in the vocabulary which is given by,

$$p(w'|w) = \begin{cases} \frac{q(w')}{N(w)} & \text{if } q(w') \geq q(w) \text{ and } w' \in Syn(w) \\ 0 & \text{otherwise} \end{cases} ; \text{ where } N(w) = \sum_{q(w'') \geq q(w), w'' \in Syn(w)} q(w'')$$

$p(w'|w)$ can be seen as the replacement affinities for a word w with respect to other words in the vocabulary. Whenever the decoder adds a token w (the token with the highest generation probability from the network) to the summary, we calculate,

$$w_{tuned} = \underset{w':p(w'|w)>0}{\arg\max} \hat{\mathbf{p}}(w') p(w'|w) \tag{5}$$

$$\hat{\mathbf{p}}(w') = \frac{e^{(\ln \mathbf{p}(w'))/r}}{\sum_{\bar{w}:p(\bar{w}|w)>0} e^{(\ln \mathbf{p}(\bar{w}))/r}} \tag{6}$$

where $\mathbf{p}$ is the distribution from the network in the latest time step, which contains the generation probabilities for each word in the vocabulary. This is an indicator of the current contextual appropriateness of the words in the vocabulary. The vocabulary preferences from $p(w'|w)$ is thus combined with the contextual appropriateness from $\hat{\mathbf{p}}(w')$ (a function of $\mathbf{p}$). The token w_{tuned} thus obtained from Eq. 5 is added to the tuned summary by replacing w. Eq. 6 (inspired by the softmax activation function with temperature often used in reinforcement learning (Sutton and Barto, 1998)) includes a replacement strength parameter r that can be used to tune the replacement levels for the algorithm. The value of r is always kept positive.

When r is close to 0, the distribution of $\hat{\mathbf{p}}(w')$ is more peaked and the value of $\hat{\mathbf{p}}(w')$ is almost 1 for the w' having highest $\mathbf{p}(w')$ and almost 0 for others. Hence $w_{tuned} = \arg\max_{w'} \mathbf{p}(w')$, and there are no replacements to tune for vocabulary. As r increases to 1, we see more replacements. When r goes much higher than 1, $\hat{\mathbf{p}}(w')$ tend to be almost same across all w'. Hence, the output would depend more on the target vocabulary preferences $p(w'|w)$, leading to more aggressive replacements at the cost of contextual appropriateness.

The proposed decoder has a better understanding of context because of the awareness of past words produced in the summary. Besides, attention based decoding has been shown to generate new words while preserving context, like generating the word *beat* by paying attention to words like *victorious* and *win* from the source text of an article about a football match (See et al., 2017). This suggests that there are high probabilities of generation for novel synonyms which can actually be used in the summary while preserving context.

As we will show later in our experiments in Section 4.2, the source article itself might have more than one word appropriate for a given context, which can be used by the decoder. For example, an article about "crime" can have both words - *inexplicable* and *mysterious*. Since generation probabilities of the pointer generator network are influenced partly by its tendency of copying words from source text, these words have a high probability of generation($\mathbf{p}()$). Now if our algorithm has to choose an alternative word for *inexplicable*, it is more likely to generate *mysterious* (if it is better suited for the target vocabulary), because of its higher generation probability than other synonyms of *inexplicable* which are not in the text and may or may not be usable in the given context.

4 Experiments

The proposed approach can be used in applications where the audience's linguistic preference can be quantified. Here, we evaluate the framework on two such applications: enhancing **text simplicity** and enhancing **text readability** of generated summaries. In the former experiment, our objective is to generate a "simple" summary that contains more commonly used words. In the latter, we adapt the summary to use shorter words thereby making it more readable. In both applications, we test the proposed framework against several baselines that are described below.

Pointer-Generator Summary (PGS): These are the summaries generated by the vanilla pointer-generator network (See et al., 2017) without any optimization for vocabulary. Note that the proposed algorithm is aimed at producing summaries with comparable quality to this vanilla generator and better tuned to the vocabulary.

Non-Contextual Post-processed Summary (NCP): Here, we replace a given word with its synonym which has the highest score. All replacements are carried out after the summaries have been generated by the network. Note that this does not consider the context for the summary generation.

Contextual Word Embedding (CWE): This method is based on the hypothesis that cosine similarities in the word/dependency embedding space capture the extent to which a word can contextually replace another as proposed by Melamud et al. (2015). Given a target word t to be replaced in a sentence, $\mathbf{p}(w'|t)$ defines a measure of the appropriateness of the word w' replacing t. If t has m dependency relations $r_1, r_2, ...r_m$ with words $w_1, w_2, .., w_m$, then $\mathbf{p}(w'|t)$ is given by,

$$\mathbf{p}(w'|t) = \frac{1}{2m}(m\langle v(w'), v(t)\rangle + \sum_{i=1}^{m}\langle v(w'), v(r_i, w_i)\rangle) \tag{7}$$

where, $v(w)$ is the embedding vector for word w and $v(r_i, w_i)$ is the embedding vector for a dependency in the same embedding space. We extend this towards our problem by replacing each word w by w_{CWE} in the generated summary S based on,

$$w_{\text{CWE}} = \underset{w':p(w'|w)>0}{\arg\max}\ \hat{\mathbf{p}}(w'|w)p(w'|w) \tag{8}$$

We define $\hat{\mathbf{p}}(w'|w)$ similar to our formulation in Eq. 6 as,

$$\hat{\mathbf{p}}(w'|w) = \frac{e^{\mathbf{p}(w'|w)/r}}{\sum\limits_{\bar{w}:p(\bar{w}|w)>0} e^{\mathbf{p}(\bar{w}|w)/r}} \tag{9}$$

where r is the replacement strength parameter. To enhance the overall quality of replacements and for unbiased comparisons, all compared approaches were set to not replace stopwords.

4.1 Dataset & Evaluation Metric

We use the CNN/Daily Mail dataset (Hermann et al., 2015; Nallapati et al., 2016) which consists of $312,084$ news articles from the CNN and Daily Mail news websites, together with multi-line human-written summaries. The dataset consists of $287,226$ article-summary training pairs, $13,368$ validation pairs and $11,490$ test pairs.

Besides measuring the degree to which the vocabulary has been tuned, we also use **ROUGE** scores (Lin, 2004) to calculate the degree of similarity between the algorithmically generated summary and a human generated summary in terms of overlap of unigrams (ROUGE-1), bi-grams (ROUGE-2) and longest common subsequence (ROUGE-L).

4.2 Simplified Summary Generation

Our first experiment focuses on simplified summary generation. Simplification aims at rewording given text to make it simpler to understand for an audience. Existing works in simplification (Paetzold and Specia, 2015; Paetzold and Specia, 2016) break down the problem into a pipeline with multiple steps: complex word identification, substitution generation, substitution selection and substitution ranking. The generation of possible substitutions can be done in many ways (Paetzold and Specia, 2015), some of which leverage dictionaries (Yamamoto, 2013), while others leverage learned substitutions from a parallel corpus (Horn et al., 2014). However, Paetzold and Specia (2016) found that using nearest neighbors in word2vec embedding space leads to better performance in substitution generation. The substitution selection part is also responsible for ensuring the contextual appropriateness of the new word. The ranking of words is typically based on the frequency of words in a standard simple corpus. The hypothesis here, is that amongst the words with the same meaning, the ones which are used frequently are simpler.

Following existing works, we set the score $q(w)$ of a word w to be its frequency in the SUBLTEX corpus (Brysbaert and New, 2009) and tune the summary generation. We measure the simplicity of a summary S based on the simplicity score defined as,

$$\text{Simplicity}(S) = \frac{1}{m}\sum_{i=1}^{m}\frac{f(s_i)}{1000}, \tag{10}$$

where $f(s_i)$ is the frequency of the i^{th} word of the summary in the SUBTLEX corpus.

Table 1(a) shows the performance of various methods across different metrics. For CWE and Voting, we tune r to yield a comparable simplicity score and report the other metrics for this setting.The NCP method achieves the highest simplicity score at the cost of the summary quality as shown by low ROUGE scores. A higher number of replacements will decrease the ROUGE scores if the newly introduced replacements are out of context and therefore unlikely to be in the ground truth summary. We observed that NCP replaces 18.429 words per summary (across the 11490 test set) in this experiment where the average summary length is 57.436 words. In contrast, CWE replaces 2.082 words and VoTing replaces

Table 1: Performance of the proposed approach against existing baselines

(a) Simplified Summary

Metric	PGS	NCP	VoTing	CWE
rouge-1 F-score	0.3880	0.2940	0.3790	0.3771
rouge-2 F-score	0.1679	0.0799	0.1563	0.1552
rouge-L F-score	0.3569	0.2706	0.3482	0.3466
Simplicity score	9.67	28.05	12.51	12.35

(b) Readable Summary

Metric	PGS	NCP	VoTing	CWE
rouge-1 F-score	0.3880	0.2724	0.3810	0.3802
rouge-2 F-score	0.1679	0.0693	0.1593	0.1593
rouge-L F-score	0.3569	0.2525	0.3504	0.3498
Reading ease	59.23	80.90	64.15	64.12

2.231 words per summary. Despite VoTing replacing more words (around 2000 more words in total), it manages to score higher on the ROUGE scores suggesting that the new words introduced still keep the summary closer to ground truth while simultaneously increasing the desired vocabulary score (which is the simplicity in this case). Table 2 shows the choices made by our proposed approach towards summary generation using simpler words from the source article.

Table 2: Simplified summaries where our method picks up simpler words, highlighted in boldface, from the source article. Baseline summaries used the words given in parantheses instead

Article: hong kong (cnn) six people were **hurt** after an explosion (...) *Summary:* (...) five out of six people were **hurt** (injured) by broken glass (...)
Article: (cnn) five americans who were monitored for three weeks at an omaha , nebraska , hospital after being exposed to ebola in west africa have been **released**, (...) *Summary:* one of the five had a heart-related issue on saturday and has been **released** (discharged).(...)
Article: (...)"it is shameful that so many states around the **world** are essentially playing with people's lives (...) *Summary:* (...) china is also mentioned , as having used the death penalty as a punitive measure across the **world** (globe).
Article: (...) but corliss was not afraid to puncture hype around **big** movies he found overrated, including "titanic" (...) *Summary:* richard corliss died a week after suffering a **big** (major) stroke. (...)

Table 3(a) shows the summaries generated on one of the articles by these methods. We can see that NCP over-replaces words leading to loss of meaning. For example, it replaces the word *march* (which refers to a month here) by *move* since move is a valid synonym of one of the senses of the replaced word (e.g. *"The army contingent marched towards the fortress."*). VoTing makes fewer replacements which seem to be in context, like replacing *mom* with *ma* and *reversed* with *turned*. CWE replaces *mom* by *grandmother* which leads to factual incorrectness.

Our formulation in VoTing and CWE allows to control the strength of replacement in the algorithm (Eq. 6 and Eq. 9). Higher strength increases the simplicity score but at the cost of reduced ROUGE. To better compare VoTing and CWE, we must look at the quality of summaries for different levels of simplicity desired in the output. We show these in Figure 1(a). It is observed that VoTing is able to achieve higher ROUGE for any given level of simplicity.

4.3 Readable summary generation

In our next experiment, we focus on readable summary generation. To make text more readable, it is advisable to use words with fewer syllables (Kincaid et al., 1975). Kincaid et al. (1975) define the Flesch reading ease to quantify readability. Text which scores high on the Flesch reading ease can be understood more easily by students of lower grade levels (Flesch, 1979). The Flesch reading ease of a summary S is given by,

$$206.835 - 1.015\frac{\text{total words}}{\text{total sentences}} - 84.6\frac{\text{total syllables}}{\text{total words}}$$

To generate more readable summaries, we run VoTing with higher scores given to shorter words, as they are likely to have fewer syllables. Here, $q(w)$ is set to be the inverse of the length of the word w. This encourages the algorithm to use shorter words while generating summaries.

Here again, we tune r for VoTing and CWE to yield a comparable reading ease and report the other metrics for this setting. Table 1(b) shows the performance of different algorithms. Our method achieves comparable ROUGE to CWE, indicating that we achieve the target without compromising on the quality.

Table 3(b) shows some sample outputs for an article. NCP replaces most words with their shortest synonyms leading to complete loss of meaning. CWE makes fewer substitutions which are more appro-

Table 3: Summaries generated by different methods on a sample article. Changed words from the baseline are highlighted in boldface

(a) Simple Summary	(b) Readble Summary
Article: Facebook has admitted it made a mistake when a photo of an Alabama boy who was born without a nose was removed from the social media website because it was deemed to be too controversial. The photo of Timothy Eli Thompson that was removed when a pro-life group posted an ad about the infant's story have since been reinstated. (...)	*Article:* A Paratrooper who braved heavy Taliban fire to rescue a wounded comrade received the Victoria Cross from the Queen yesterday. She told Lance Corporal Joshua Leakey: I dont get to give this one out very often. Did you ever imagine youd be standing here? Well done. But in fact the 27-year-old is the second member of his family to receive the highest military decoration for valour a cousin was given the honour 70 years ago (...)
Baseline summary: timothy eli thompson was born without a nose in march in alabama and his mom put his photo on facebook .facebook reversed its decision after public protest and admitted it made a mistake .facebook reversed its decision after public protest .	*Baseline summary:* the 27-year-old is the second member of his family to receive the highest military decoration for valour .it is just the sixth time the queen has given a vc to a living british recipient during the afghanistan campaign but the other two awards were made posthumously .in 2013 he braved heavy gunfire from 20 taliban insurgents in helmand to rush to the aid of a wounded us marine .
VoTing: timothy eli thompson was born without a nose in march in alabama and his **ma** put his photo on facebook . facebook **turned** its decision after public protest and admitted it made a mistake . facebook reversed its decision after public protest .	*VoTing:* the 27-year-old is the second member of his family to **get** the **top** military decoration for valour . it is just the sixth time the queen has given a vc to a living british recipient during the afghanistan campaign but the other two awards were made posthumously . in 2013 he braved heavy gunfire from 20 taliban insurgents in helmand to rush to the aid of a wounded us marine .
CWE: timothy eli thompson was born without a nose in march in alabama and his **grandmother** put his photo on facebook . facebook reversed its decision after public protest and admitted it made a mistake . facebook reversed its decision after public protest .	*CWE:* the 27-year-old is the second member of his family to **get** the highest military decoration for valour . it is just the sixth time the queen has given a vc to a living british recipient during the afghanistan campaign but the other two awards were made posthumously . in 2013 he braved heavy gunfire from 20 taliban insurgents in helmand to rush to the aid of a wounded us marine .
NCP: timothy eli thompson was born **out** a nose in **move** in alabama and his **ma** put his **picture** on facebook . facebook **turned** its **end** after **open question** and **take** it made a **fault** . facebook **turned** its **end** after **open question** .	*NCP:* the 27-year-old is the **twin cut** of his **clan** to **cop** the **top army garnish** for valour . it is just the sixth **go** the **ruler** has **apt** a vc to a **warm** british **heir** during the afghanistan **push** but the other **dos gift** were made posthumously . in 2013 he **firm fat salvo** from 20 taliban **radical** in helmand to **flux** to the aid of a **hurt** us **sea** .

priate like replacing *receive* by *get*. VoTing makes the same replacement and also replaces *highest* by *top* - suggesting that our approach performs more tuning without compromising on the overall quality.

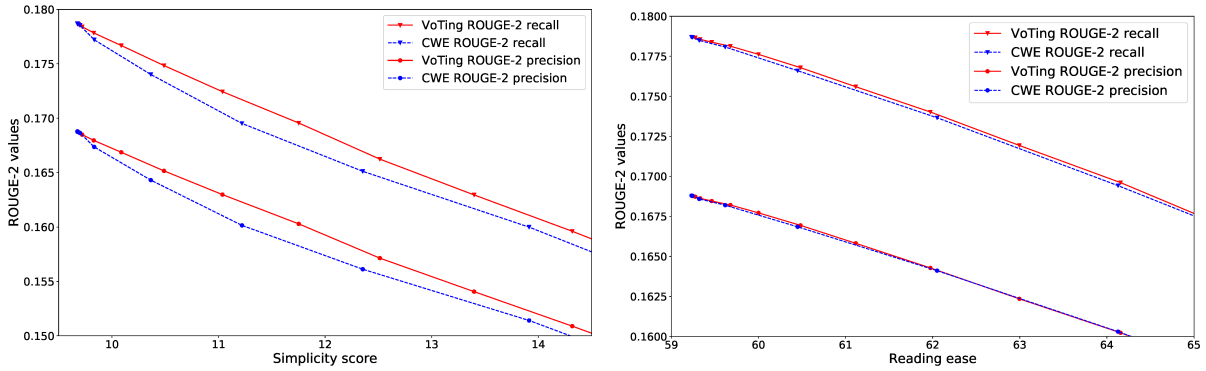
Varying the replacement strength parameter, we find that VoTing again has higher ROUGE-2 scores across different levels of reading ease. This can be seen in Figure 1(b) indicating better contextual tuning by the proposed approach across different reading ease.

4.4 Human evaluation of contextual appropriateness of VoTing

While the level of improvement in the vocabulary tuning achieved in a summary can be measured by using various scores, the contextual appropriateness of the new words added is better judged by humans. We, therefore, conducted a survey, where each annotator was shown the ground truth summary generated by PGS along with the tuned summaries from the three methods - NCP, CWE and VoTing. The annotators were asked to rank the outputs of the three methods according to the extent to which it preserves the meaning of the original summary generated by PGS. The three tuned summaries were shown in random order to remove any positional bias. We used the summaries from 20 randomly chosen articles from the test set for the survey. Each set of summaries was annotated by 4 or 5 different annotators. We had a total of 90 human annotated rankings.

We used the Condorcet fusion (Montague and Aslam, 2002) to aggregate the rankings, which looks at pairwise comparisions between the methods. The results are shown in Table 4, which indicates that VoTing performs the best, beating CWE in 61.11% of responses, and NCP clearly performs the worst. The Krippendorff's alpha (Krippendorff, 2011) for inter-annotator agreement was 0.84 indicating high inter-annotator agreement.

After establishing the comparative superiority of VoTing, we proceeded to objectively analyze the degree to which these three methods preserved the meaning of the original summary. We ran another



(a) Simple Summary

(b) Readable Summary

Figure 1: CWE vs Voting for different simplicity and reading ease levels. ROUGE-2 precision and recall are shown for different levels of tuning achieved.

Table 4: Pairwise comparison of human rankings of different methods. Each row signifies the fraction of times the corresponding method was ranked higher than the method corresponding to the column.

	CWE	**VoTing**	**NCP**
CWE	-	0.3889	0.9667
VoTing	0.6111	-	0.9778
NCP	0.0333	0.0222	-

human evaluation where we showed human raters the summaries from the three methods in random order and asked them to rate the three on a scale of 1 to 5 according to the descriptions given in Table 5, on the extent to which they preserve the meaning of the PGS generated summary. VoTing received an average rating of 3.47 against 3.36 for CWE and 1.83 for NCP, further confirming the contextual appropriateness of the proposed tuning.

Table 5: Description shown to human annotators for ratings on the contextual appropriateness scale

Rating value	**Description**
5	Perfectly captures the original meaning
4	Mostly preserves the meaning
3	Understandably close to the original meaning
2	Changes the meaning by a little bit
1	Completely changes the meaning

5 Conclusions and Future work

We proposed a novel approach to generate summaries of articles while incorporating vocabulary preferences. We showed the application of our algorithm to text simplification and text readability enhancement. We showed that tuning the vocabulary during summary generation leads to fewer out-of-context replacements than post-processing a generated summary. Our findings also suggest that LSTM-based decoders of pointer-generator networks are capable of preserving the local context better than word embeddings trained on vast corpora.

Our current work is limited to replacing words with better synonyms. However, introduction of new words can benefit tuning the generation towards a specific aspect or tone. For example, "Pass me that plate." can be changed to "*Please* pass me that plate." to make it sound more formal. Rephrasing a sequence of words instead of replacing one word at a time or changing the structure of a sentence are other ways to make it suit a target audience's preference. The ability of LSTM based decoders to carry out such transformations is a subject for further explorations.

References

Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document abstractive summarization using ilp based multi-sentence compression. In *International Joint Conference on Artificial Intelligence*.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 481–490. Association for Computational Linguistics.

Stefan Bott, Luz Rello, Biljana Drndarevic, and Horacio Saggion. 2012. Can spanish be simpler? lexsis: Lexical simplification for spanish. *International Conference on Computational Linguistics*, pages 357–374.

Marc Brysbaert and Boris New. 2009. Moving beyond kučera and francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for american english. *Behavior research methods*.

Sumit Chopra, Michael Auli, Alexander M Rush, and SEAS Harvard. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 93–98.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. Stanford typed dependencies manual. Technical report, Technical report, Stanford University.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*.

Katja Filippova. 2010. Multi-sentence compression: Finding shortest paths in word graphs. In *International Conference on Computational Linguistics (COLING)*. Association for Computational Linguistics.

Rudolf Franz Flesch. 1979. *How to write plain English: A book for lawyers and consumers*. Harpercollins.

Pierre-Etienne Genest and Guy Lapalme. 2011. Framework for abstractive summarization using text-to-text generation. In *Workshop on Monolingual Text-To-Text Generation*. Association for Computational Linguistics.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

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

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a lexical simplifier using wikipedia. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Technical Training Command Millington TN Research Branch.

Klaus Krippendorff. 2011. Computing krippendorff's alpha-reliability. *ScholarlyCommons*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*. Barcelona, Spain.

Oren Melamud, Omer Levy, Ido Dagan, and Israel Ramat-Gan. 2015. A simple word embedding model for lexical substitution. In *VS@ HLT-NAACL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (ICLR)*.

Mark Montague and Javed A Aslam. 2002. Condorcet fusion for improved retrieval. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 538–548. ACM.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Ça glar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *CoNLL*.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends® in Information Retrieval*.

Gustavo Henrique Paetzold and Lucia Specia. 2015. Lexenstein: A framework for lexical simplification. In *ACL (System Demonstrations)*.

Gustavo Henrique Paetzold and Lucia Specia. 2016. Unsupervised lexical simplification for non-native speakers. In *30th AAAI Conference (ACL)on Artificial Intelligence*.

Stephen Roller and Katrin Erk. 2016. Pic a different word: A simple model for lexical substitution in context. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Rishiraj Saha Roy, Aishwarya Padmakumar, Guna Prasaad Jeganathan, and Ponnurangam Kumaraguru. 2015. Automated linguistic personalization of targeted marketing messages mining user-generated text on social media. In *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Conference on Empirical Methods in Natural Language Processing*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. MIT press Cambridge.

György Szarvas, Chris Biemann, Iryna Gurevych, et al. 2013. Supervised all-words lexical substitution using delexicalized features. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

T Yamamoto. 2013. Selecting proper lexical paraphrase for children. In *Proceedings of the Twenty-Fifth Conference on Computational Linguistics and Speech Processing (ROCLING 2013)*.

Reading Comprehension with Graph-based Temporal-Casual Reasoning

Yawei Sun, Gong Cheng, Yuzhong Qu
National Key Laboratory for Novel Software Technology, Nanjing University, China
ywsun@smail.nju.edu.cn, gcheng@nju.edu.cn, yzqu@nju.edu.cn

Abstract

Complex questions in reading comprehension tasks require integrating information from multiple sentences. In this work, to answer such questions involving temporal and causal relations, we generate event graphs from text based on dependencies, and rank answers by aligning event graphs. In particular, the alignments are constrained by graph-based reasoning to ensure temporal and causal agreement. Our focused approach self-adaptively complements existing solutions; it is automatically triggered only when applicable. Experiments on RACE and MCTest show that state-of-the-art methods are notably improved by using our approach as an add-on.

1 Introduction

The task of *reading comprehension* has received wide attention from the research community. A machine's ability to understand natural language is tested by answering multiple-choice questions based on a given passage (Hirschman et al., 1999), as illustrated in Fig. 1. State-of-the-art methods have achieved encouraging results on simple factoid questions, but still, it is a challenge to answer a complex question that requires extracting information from more than one sentence in the passage. Typical examples are those *involving temporal or causal relations* spanning multiple sentences (Sugawara et al., 2017; Trischler et al., 2016), like Questions 1 and 2 in Fig. 1. They constitute a considerable proportion of the questions in popular datasets like MCTest (Richardson et al., 2013) and RACE (Lai et al., 2017).

To integrate multiple sentences for answering a complex question, various techniques have emerged, but many of them ignore the semantic relations between sentences (Wang et al., 2016) or superficially use a sliding window scanning over the words of the passage without sentence breaks (Smith et al., 2015; Trischler et al., 2016). To meet the challenge, we *explicitly model the event structure of the passage by a graph representing temporal and causal relations between events* extracted from the passage. Representing a candidate answer in a similar way, we align the two graphs subject to temporal and causal agreement which is verified by *graph-based reasoning*, and rank candidate answers by their degrees of alignment. Our approach, called Graph-based Temporal-Casual Reasoning (GTCR) which focuses on temporal and casual questions, can be used as *an effective add-on to other methods*.

Our research contribution in this paper is threefold.

- To answer complex questions in reading comprehension tasks, we represent the passage and each candidate answer by an event graph to explicitly model temporal and causal relations, which are crucial to the understanding and integration of multiple sentences for answering questions. We generate event graphs from text based on dependencies produced by an off-the-shelf parser, thereby benefiting from methodological progress and tools on dependency parsing.
- To score a candidate answer, we align it with the passage using their event graphs. In an alignment, event matching is constrained by temporal and causal agreement, for which graph-based event reasoning is performed. The alignment problem is formulated and solved using linear programming.

Proceedings of the 27th International Conference on Computational Linguistics, pages 806–817
Santa Fe, New Mexico, USA, August 20-26, 2018.

Passage: Susan held a birthday party. She made a big cake, and hung up some balloons. Meanwhile, her parents bought chocolate ice creams because she enjoyed it. Soon, her friends showed up. Then, Susan hugged her friends. Each friend had a present for Susan. Therefore, Susan was happy and sent each friend a thank you card. So, her friends were happy, too.

Question 1: What did Susan do before her friends came out?
(A) Susan bought ice cream. (B) Susan hung up balloons. (C) Susan hugged her friends. (D) Susan sent friends thank you cards.

Question 2: Why did Susan send out thank you cards to her friends?
(A) Her friends love her. (B) Her friends brought her gifts at the party. (C) Her friends was happy. (D) Her friends came late.

Figure 1: A passage and two questions for reading comprehension.

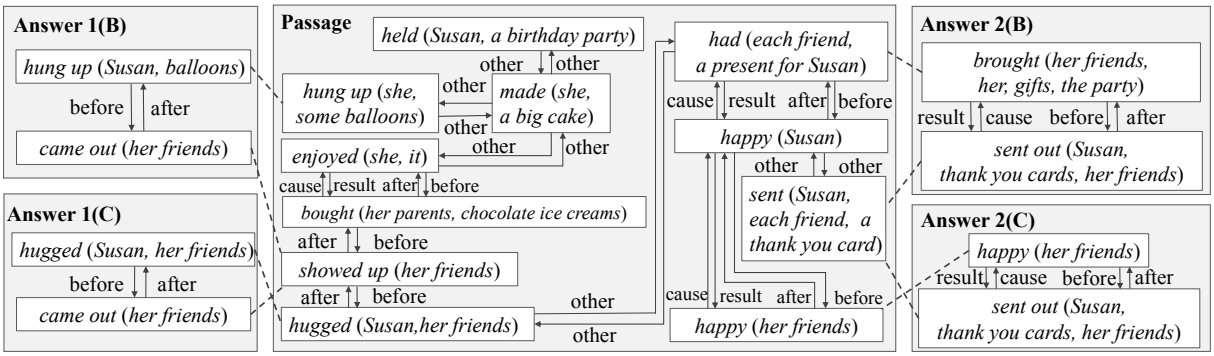


Figure 2: An event graph for the passage and four event graphs for question-answer combinations.

- By automatically detecting question type and determining confidence in scores, our self-adaptive approach is triggered only when appropriate, to enhance existing methods as an add-on. A notable improvement on the state of the art is observed in the experiments on RACE and MCTest.

The paper is structured as follows. Section 2 overviews our approach. Sections 3 and 4 describe event graph generation and alignment, respectively. Section 5 combines our approach with existing methods. Section 6 reports experiments. Section 7 compares related work. Section 8 concludes the paper.

2 Overview of the Approach

Our pipelined GTCR approach is outlined in Fig. 3. The passage is transformed into an event graph called the passage graph. The question and each candidate answer are combined and transformed into an event graph called a QA graph. The two graphs are aligned, and the degree of alignment is assigned as the score of the answer.

2.1 Event Graph Generation

The passage is transformed into an event graph, called the passage graph, representing temporal and causal relations between events extracted from the passage. The transformation firstly recognizes events from each sentence and extracts intra-sentence relations, and then merges the results based on inter-sentence relations. We will define event graph in Section 3.1 and elaborate its generation in Section 3.2.

The question and each candidate answer are combined and transformed in a similar way into an event graph, called a QA graph. We will elaborate this step in Section 3.3.

Example 1 *The passage in Fig. 1 is transformed into the passage graph in Fig. 2. Four out of the eight candidate answers in Fig. 1 are combined with questions and transformed into four QA graphs in Fig. 2.*

2.2 Event Graph Alignment

The passage graph and the QA graph for each candidate answer are aligned. The degree of alignment, which is the maximum sum of the degrees of match between aligned events, is assigned as the score of the answer. Alignment is constrained by the agreement in temporal and causal relations between events, which is verified by graph-based reasoning. We will elaborate this step in Section 4.

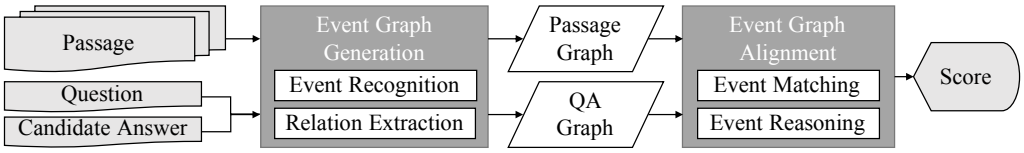


Figure 3: Overview of the GTCR approach.

Example 2 *Dashed lines in Fig. 2 represent exact matches between events, where the alignments for answers 1(C) and 2(C) violate temporal and causal agreement, respectively, thereby being infeasible. The degrees of other (unshown) possible alignments for these answers are not large. Therefore, they are likely to be incorrect answers.*

3 Event Graph Generation

Event graph is a core concept in our approach. In this section we present its definition and detail its generation from the passage. We also describe how to generate an event graph in a similar way from the combination of the question and each candidate answer.

3.1 Event Graph

Definition 1 (Event Graph) *An event graph is a directed multi-graph where vertices represent events which are connected by arcs representing temporal or causal relations between events.*

It is a multi-graph which is permitted to have multiple arcs between two vertices, e.g., both a temporal relation and a causal relation between two events, as illustrated in Fig. 2.

Definition 2 (Event) *An event consists of a predicate and its arguments (Jurafsky and Martin, 2009), and has three attributes (Pustejovsky et al., 2003; Bar-Haim et al., 2015), namely* `aspect` $\in$ $\{$`perfective`, `none`$\}$, `modality` $\in$ $\{$`modal`, `none`$\}$, *and* `polarity` $\in$ $\{$`positive`, `negative`$\}$.

Definition 3 (Relations between Events) *We consider three types of relations.*

Temporal Relation. *An event u can happen* before *another event v, represented by an arc from u to v labeled with* `before`*, and an arc from v to u labeled with* `after`.

Causal Relation. *An event u can* cause *another event v, represented by an arc from u to v labeled with* `result`*, and an arc from v to u labeled with* `cause`.

Other Relation. *We also allow an unspecified relation between two events, represented by a pair of inverted arcs labeled with* `other`.

3.2 Passage Graph Generation

To generate an event graph from the passage, called the *passage graph*, we recognize events from sentences, extract intra-sentence relations to form a local event graph for each sentence, and finally extract inter-sentence relations to merge local event graphs into a global event graph.

3.2.1 Event Recognition

Predicate and Arguments To recognize the predicate and arguments which identify an event, we extend a dependency-based method (Rudinger and Van Durme, 2014). In particular, we consider *embedded events*. Given a dependency relation `ccomp` or `xcomp` whose head is the predicate of an embedding event u and whose dependent is the predicate of an embedded event v, we remove v and assign its predicate and arguments to u's arguments. For example, consider the sentence *Hannah's mother explained that they were moving to Kenya.* The embedded event *moving* is removed; its predicate and arguments, namely *moving* and $\{$*they*, *Kenya*$\}$, become the arguments of the embedding event *explained*.

Attributes To recognize the attributes of an event, we extend a heuristic method (Bar-Haim et al., 2015) which focuses on the attribute `polarity`, and define the following heuristics for recognizing `aspect` and `modality` of an event u based on dependencies.

- Given a dependency relation `aux` whose head is the predicate of u and is POS-tagged with VBN, and whose dependent's lemma is *have*, we have u.`aspect` = `perfective`.
- Given a dependency relation `aux` whose head is the predicate of u and whose dependent is a modal verb (e.g., *will*, *should*), we have u.`modality` = `modal`.

Example 3 *Consider the sentence* Meanwhile, her parents bought chocolate ice creams because she enjoyed it. *As shown in Fig. 2, two events with predicates* bought *and* enjoyed *are recognized. The arguments of* bought *consist of* her parents *and* chocolate ice creams. *The arguments of* enjoyed *consist of* she *and* it. *There is no particular attribute to recognize, so all the three attributes of these events take their default values, i.e.,* `aspect` = `none`, `modality` = `none`, *and* `polarity` = `positive`.

3.2.2 Intra-sentence Relation Extraction

We observe that within a sentence where two or more events are recognized, their relations are usually explicitly indicated by a few syntactic patterns. We define a mapping from dependency patterns to relations as shown in Table 1. Note that some polysemous patterns (e.g., those involving *so*) can give rise to multiple relations. The result is a local event graph generated from each sentence.

Relation	Dependency Pattern
$u \xrightarrow{\texttt{before}} v$ and $u \xleftarrow{\texttt{after}} v$	(i) $u \xrightarrow{\texttt{advcl}} v_{(\texttt{modality=none})} \xrightarrow{\texttt{mark\|advmod}}$ *before|until|till|so|so-that* (ii) $v_{(\texttt{modality=none})} \xrightarrow{\texttt{advcl}} u \xrightarrow{\texttt{mark\|advmod}}$ *after|since|because|for|as* (iii) $v_{(\texttt{modality=none})} \xleftarrow{\texttt{conj}} u \xrightarrow{\texttt{cc}}$ *so*
$u \xrightarrow{\texttt{result}} v$ and $u \xleftarrow{\texttt{cause}} v$	(iv) $u \xrightarrow{\texttt{advcl}} v \xrightarrow{\texttt{mark\|advmod}}$ *so|so-that* (v) $v \xrightarrow{\texttt{advcl}} u \xrightarrow{\texttt{mark\|advmod}}$ *because|for|as* (vi) $v \xleftarrow{\texttt{conj}} u \xrightarrow{\texttt{cc}}$ *so*
$u \xrightarrow{\texttt{other}} v$ and $u \xleftarrow{\texttt{other}} v$	other cases

Table 1: Extraction of intra-sentence relations between two events u and v.

Example 4 *Consider two events* bought *and* enjoyed *recognized in the sentence* Meanwhile, her parents bought chocolate ice creams because she enjoyed it. *Here, two dependency patterns (ii) and (v) in Table 1 are matched, and hence both a temporal relation and a causal relation are extracted, as shown in Fig. 2.*

Example 5 *Consider two events* made *and* hung up *recognized in the sentence* She made a big cake, and hung up some balloons. *An other relation is extracted here, as shown in Fig. 2.*

3.2.3 Inter-sentence Relation Extraction

We extract relations between adjacent sentences, in order to merge local event graphs into a global event graph. For two adjacent sentences, an *exit event* u and an *entrance event* v are selected from the former sentence and the latter sentence, respectively; then relations between u and v are extracted.

Event Selection In the local event graph representing the former sentence, the exit event is one that has `before` or `result` as an incoming arc but never as an outgoing arc. If none of the events in the sentence satisfies this condition, the one that involves the root of the dependency tree (called the *root event*) will be specified as the exit event. Similarly, in the local event graph representing the latter sentence, the entrance event is one that has `before` or `result` as an outgoing arc but never as an incoming arc.

Relation Extraction Having determined the exit event u in the former sentence and the entrance event v in the latter sentence, their relations are extracted in the following way.

- We reuse the set of connective adverbs listed by Lin et al. (2014), and identify two (overlapping) classes: a temporal relation (i.e., $u \xrightarrow{\texttt{before}} v$ and $u \xleftarrow{\texttt{after}} v$) is extracted given *later*, *next*, etc.; or a causal relation (i.e., $u \xrightarrow{\texttt{result}} v$ and $u \xleftarrow{\texttt{cause}} v$) is extracted given *hence*, *therefore*, etc.

- In particular, when $v.\texttt{aspect} = \texttt{perfective}$, a temporal relation in the reverse direction (i.e., $v \xrightarrow{\texttt{before}} u$ and $v \xleftarrow{\texttt{after}} u$) is heuristically extracted.

- In other cases, an other relation (i.e., $u \xrightarrow{\texttt{other}} v$ and $u \xleftarrow{\texttt{other}} v$) is extracted.

Example 6 *Consider two adjacent sentences* Each friend had a present for Susan. Therefore, Susan was happy and sent each friend a thank you card. *The exit event of the former sentence is* had*, and the entrance event of the latter sentence is* happy. *According to the connective adverb* therefore*, both a temporal relation and a causal relation between the two events are extracted, as shown in Fig. 2.*

3.3 QA Graph Generation

We also generate an event graph from the combination of the question q and each candidate answer ans, called a *QA graph*. When q is a fill-in-the-blank question, this QA graph is generated by firstly replacing the blank in q with ans to form a complete text, and then following the method described in Section 3.2 to generate an event graph from this text.

When q is a *wh*-question, we firstly generate an event graph G_q from q using the method described in Section 3.2, and then insert ans into G_q. Here, we differentiate between two types of *wh*-questions.

- For a *why*-question, ans is normally a complete sentence, from which an event graph G_{ans} is generated using the method described in Section 3.2. G_{ans} and G_q are merged by connecting their root events via a causal and a temporal relation, as illustrated by answers 2(B) and 2(C) in Fig. 2.

- For a non-*why*-question, the argument in G_q that contains the interrogative word is replaced by ans. For an exceptional case where ans is a complete sentence and the predicate of the root event in G_q is a general verb like *do* or *happen*, we generate an event graph G_{ans} from ans using the method described in Section 3.2, and then merge G_{ans} and G_q by replacing the event involving the interrogative word in G_q with the root event in G_{ans}, as illustrated by answers 1(B) and 1(C) in Fig. 2.

4 Event Graph Alignment

The degree of alignment between the passage graph G_p and the QA graph G_{qa} for each candidate answer ans is assigned as the score of ans. In this section we formulate and solve the alignment problem, which maximizes event matching and is constrained by graph-based event reasoning.

4.1 Problem Formulation

We define the *degree of alignment* between G_p and G_{qa} as the maximum sum of the degrees of match between aligned events in G_p and G_{qa}, which is formulated as a variant of the maximum weighted bipartite matching problem, which in turn is a special case of a 0-1 linear program.

Specifically, let V_p and V_{qa} be the events in G_p and G_{qa}, respectively. For $u \in V_p$ and $v \in V_{qa}$, let $\texttt{dom}(u, v)$ be the degree of match between u and v, which will be detailed in Section 4.2. The score of ans, i.e., the degree of alignment between G_p and G_{qa}, is given by

$$\begin{aligned} &\text{maximizing} \sum_{u \in V_p, v \in V_{qa}} \texttt{dom}(u, v) x_{uv} \\ &\text{subject to} \sum_{v \in V_{qa}} x_{uv} \leq 1 \text{ for } u \in V_p\,, \quad \sum_{u \in V_p} x_{uv} \leq 1 \text{ for } v \in V_{qa}\,, \\ &\qquad x_{uv} + x_{km} \leq 1 \text{ for } u, k \in V_p,\ v, m \in V_{qa},\ \texttt{conflict}(u, v, k, m) = 1\,, \\ &\qquad x_{uv} \in \{0, 1\} \text{ for } u \in V_p,\ v \in V_{qa}\,, \end{aligned} \tag{1}$$

where $x_{u,v}$ is a binary variable representing the alignment between u and v, taking value 1 if they are aligned, and $\texttt{conflict}(u, v, k, m)$ indicates whether two matches u-v and k-m lead to a conflict in terms of temporal or causal relations between events, which will be detailed in Section 4.3. We further normalize the score of ans by dividing it by the number of events in G_{qa}.

Note that as a special case, we will define the degree of alignment as one minus the above result if the `polarity` of the root event in the question is `negative`.

4.2 Event Matching

The degree of match between two events u and v, denoted by $\texttt{dom}(u, v)$, measures to what extent u and v refer to the same event. For example, each dashed line in Fig. 2 represents an exact match between two coreferential events. Our measure is a combination of element-level matching and word-level matching:

$$\texttt{dom}(u,v) = \begin{cases} \alpha \cdot \texttt{dom}_{\text{elem}}(u,v) + \beta \cdot \texttt{dom}_{\text{word}}(u,v) & \text{if } u.\texttt{polarity} = v.\texttt{polarity}, \\ 0 & \text{if } u.\texttt{polarity} \neq v.\texttt{polarity}, \end{cases} \tag{2}$$

where α, β are weights to be empirically tuned. Note that for two events having opposite values of `polarity`, their degree of match is simply fixed to 0.

Element-level Matching Let $\texttt{Elem}(u)$ be the set of *elements* of event u, consisting of the predicate and arguments of u. For two elements $e_i \in \texttt{Elem}(u)$ and $e_j \in \texttt{Elem}(v)$, we extend the method used by Li and Srikumar (2016) and define 16 features that calculate the similarity between e_i and e_j. Each feature ϕ returns a similarity value in the range of $[0, 1]$. Inspired by Yih et al. (2013), we calculate the overall similarity between e_i and e_j as follows:

$$\texttt{sim}(e_i, e_j) = \max_{\phi} w_\phi \cdot \phi(e_i, e_j)\,, \tag{3}$$

where ϕ is weighted by w_ϕ. The following features and weights are used.

- $w = 1.0$: exact string match, exact string match between roots of dependency trees, existence of trigram overlap, coreference, apposition, sharing synonyms in Wiktionary, sharing synonyms in WordNet, sharing derivations in WordNet, VerbOcean-based similarity;
- $w = 0.5$: Jaccard similarity of words, partial string match, existence of bigram overlap, cosine similarity of Word2vec embeddings, cosine similarity of GloVe embeddings, light-verb-to-verb match;
- $w = 0.1$: existence of unigram overlap.

With `sim` values for all e_i-e_j pairs in $\texttt{Elem}(u) \times \texttt{Elem}(v)$, we find a maximum weighted bipartite matching between $\texttt{Elem}(u)$ and $\texttt{Elem}(v)$, and assign the squared mean weight of this matching to $\texttt{dom}_{\text{elem}}(u, v)$.

Word-level Matching Complementary to element-level matching which relies on dependency-based argument recognition, we measure the degree of word-level match between u and v using a bag-of-words method (Sultan et al., 2016), and assign the result to $\texttt{dom}_{\text{word}}(u, v)$.

4.3 Event Reasoning

For events $u, k \in V_p$ and $v, m \in V_{qa}$, $\texttt{conflict}(u, v, k, m) = 1$ means if two matches u-v and k-m appear in the same alignment, a *conflict* in terms of temporal or causal relations will emerge; such an alignment is not allowed. We consider two types of conflicts by extending the idea in Berant et al. (2014).

Temporal Conflict There is a (directed simple) u-k path in G_p and a v-m path in G_{qa} such that `before` appears in one of them and `after` appears in the other.

Causal Conflict There is a u-k path in G_p and a v-m path in G_{qa} such that `result` appears in one of them and `cause` appears in the other.

For efficiency reasons, our implementation only checks paths that contain not more than 10 arcs.

Example 7 *The alignment represented by the dashed lines between the passage graph and the QA graph for answer 1(C) in Fig. 2 is not feasible due to the temporal conflict induced by the two matches. The alignment for answer 1(B) is feasible. Similarly, the alignment for answer 2(C) is not feasible due to a causal conflict, whereas the alignment for answer 2(B) is feasible.*

5 Combination with Other Methods

Our GTCR approach is specifically developed for temporal and causal questions. Reading comprehension tasks also involve other questions. Therefore, GTCR is to be combined with other methods.

We propose to let GTCR *self-adaptively enhance existing methods as an add-on*. Specifically, GTCR will answer a question only if it accepts the type of the question and is confident in the scores it generates. Otherwise, it will not answer the question, which in turn will be answered by an existing method. In this section we describe how to *automatically* detect question type and determine confidence in scores.

5.1 Question Type

Our current implementation of GTCR can process *wh*-questions and fill-in-the-blank questions. However, it only accepts temporal and causal questions. A temporal question, e.g., Question 1 in Fig. 1, is one that matches dependency patterns (i) or (ii) in Table 1, or in case there are errors in dependency parsing, matches the following POS-level regular expression: .*VB.+(*when*|*after*|*before*|*since*).+VB.*. A causal question, e.g., Question 2 in Fig. 1, is one that begins with *why*, or contains an adverbial clause fronted by *because* (which is common for fill-in-the-blank questions).

5.2 Confidence in Scores

As GTCR relies on the results of dependency parsing, it will not be confident in the scores it generates if it finds errors in the dependency tree of the question. Specifically, inspired by Petrov et al. (2010), our implementation detects five common types of errors in dependency parsing:

- `nsubj`: missing a subject in a question or clause,
- `aux`: placing an auxiliary verb as the root of the dependency tree,
- `dep`: associating `dep` with a subject,
- `acl:relcl`: accompanying `acl:relcl` which indicates a relative clause with a conjunction commonly fronting an adverbial clause (e.g., *when*), and
- `advcl`: missing core arguments (e.g., `nsubj`, `obj`) in an adverbial clause.

When dependency parsing seems correct, GTCR will still reject the question if the score of the answer it selects is below a threshold τ, which will be tuned in the experiments.

6 Experiments

6.1 Datasets

Experiments are performed on two standard datasets for reading comprehension: RACE and MCTest.

RACE (Lai et al., 2017) consists of nearly 28,000 passages and 100,000 questions collected from English exams for middle and high school Chinese students. In particular, 25.8% of these questions require multi-sentence reasoning, though not necessarily temporal or causal reasoning. For our experiments we use middle school data, denoted by RACE-M.

MCTest (Richardson et al., 2013) consists of 660 fictional stories as passages and 2,640 questions that a seven-year-old can understand. It is divided into MC160 where 160 stories have been manually curated for quality, and MC500 where 500 stories may have grammatical or other errors.

6.2 Competing/Collaborating Methods

For RACE we consider two state-of-the-art methods. Stanford Attentive-Reader (Stanford AR) (Chen et al., 2016) is a neural network system based on the Attentive-Reader model. Gated-Attention (GA) Reader (Dhingra et al., 2017) integrates a multi-hop architecture with an attention mechanism.

For MCTest we consider three state-of-the-art methods. Smith et al. (2015) propose a lexical matching method that takes into account multiple context windows, question types, and coreference resolution.

Narasimhan and Barzilay (2015) incorporate discourse information. Trischler et al. (2016) harness simple neural networks arranged in a parallel hierarchy.

Some other methods are not considered in our experiments because their codes or detailed results are not available at the time of experiment so that we could not test the combination of GTCR with them.

6.3 Our Approach

For our implementation of GTCR, we use Stanford CoreNLP (Manning et al., 2014) for dependency parsing and coreference resolution, and use Gurobi (www.gurobi.com) for solving linear programs.

Some parameters are tuned based on the development set in each dataset, which is separate from the test set. Specifically, for Eq. (2) we set $\alpha = \beta = 0.5$ for all the datasets. The confidence threshold τ used for combination with other methods is set to 0.00 for RACE, 0.71 for MC160, and 0.78 for MC500.

Our implementation is open source[1].

	RACE-M
Stanford AR	58/133
GA Reader	50/133
GTCR	**63/133**
GTCR w/o elem. matching	63/133
GTCR w/o word matching	59/133
GTCR w/o reasoning	61/133

Table 2: Accuracy of competition on RACE-M.

	MC160	**MC500**
Smith et al. (2015)	8/13	9/16
Narasimhan et al. (2015)	7/13	8/16
Trischler et al. (2016)	—	10/16
GTCR	**9/13**	**12/16**
GTCR w/o elem. matching	7/13	9/16
GTCR w/o word matching	10/13	11/16
GTCR w/o reasoning	9/13	10/16

Table 3: Accuracy of competition on MCTest.

	RACE-M	
		+GTCR
Stanford AR	44.15	**44.50**
GA Reader	43.73	**44.64**

Table 4: Accuracy (%) of collaboration on RACE-M.

	MC160		**MC500**	
		+GTCR		+GTCR
Smith et al. (2015)	74.27	**74.68**	65.79	**66.29**
Narasimhan et al. (2015)	71.45	**72.29**	63.71	**64.38**
Trischler et al. (2016)	74.58	—	70.41	**70.75**

Table 5: Accuracy (%) of collaboration on MCTest.

6.4 Results

We carry out two experiments. In the first experiment, GTCR *competes* with state-of-the-art methods on complex questions that require temporal or causal reasoning. In the second experiment, GTCR self-adaptively *collaborates* with those methods as an add-on, in order to advance the state of the art.

We measure the accuracy of each method, namely the proportion of correctly answered questions.

Competition On RACE-M, among the 1,436 test questions, 133 are recognized as temporal or causal questions according to question type detection in GTCR. Table 2 shows the accuracy of each method on these questions. Compared with competing methods, GTCR correctly answers 4–10% more of those questions.

On MCTest, among the 240 and 600 test questions in MC160 and MC500, 51 and 99 respectively are recognized as temporal or causal questions. After excluding 10 and 17 questions respectively having dependency errors and those on which GTCR is not confident, Table 3 shows the results on the remaining questions. GTCR notably outperforms competing methods on these questions.

Ablation To evaluate the effectiveness of our event matching and reasoning, we conduct an ablation study. As shown in Tables 2 and 3, the accuracy of our approach decreases by up to 19% and 6% when element-level and word-level event matching are disabled, respectively, showing that the two techniques

[1] http://ws.nju.edu.cn/qa/coling2018

are generally useful; only for one question in MC160, word-level matching has a negative effect. Without performing event reasoning, the accuracy decreases by 2% on RACE-M and by 13% on MC500, showing its usefulness. However, reasoning is not helpful on MC160.

Collaboration By self-adaptively collaborating with existing methods as an add-on, GTCR consistently enhances all the methods and improves the state of the art on both datasets. As shown in Table 4, the overall accuracy is increased by 0.35–0.91% on RACE-M. For MCTest in Table 5, the improvement is in the range of 0.41–0.84% on MC160, and 0.34–0.67% on MC500. On MC160 we do not add GTCR to Trischler et al. (2016) because at the time of experiment we did not obtain its details results.

6.5 Error Analysis and Discussion

In GTCR, errors may occur in event graph generation and event matching. We analyze the 78 questions in Tables 2 and 3 for which GTCR produces incorrect answers.

Event Graph Generation Although GTCR can automatically detect a few common types of errors in dependency parsing, other POS or dependency errors are still observed in the experiments, which lead to imprecise recognition of events and extraction of relations (18%). These errors are propagated along the pipeline framework of GTCR, which has the potential to be significantly improved by benefiting from future advances in dependency parsing.

Another drawback to GTCR is the use of syntactical methods for extracting explicit relations between events. Some temporal and causal relations are implicit, and hence are missing in our generated event graphs (31%). To identify those relations, commonsense knowledge is to be exploited.

Event Matching Event matching is closely related to paraphrasing. Our element-level and word-level matching in GTCR, though utilizing many linguistic resources, still fail to identify some coreferential events that are expressed differently (19%). It inspires us to integrate or develop more effective techniques for paraphrasing and textual entailment in future work.

Another disadvantage of GTCR is its limited expressivity of event alignment. We assume one-to-one event matching between event graphs, but in practice it is sometimes one-to-many (6%).

6.6 Running Time

On an Intel E3-1225v3 (3.2GHz), our approach uses 7.8 seconds, 6.5 seconds, and 7.0 seconds to process a question in RACE-M, MC160, and MC500, respectively, being reasonably fast. Most time is used for: retrieving external resources in element-level event matching, and identifying conflicting event matches in event graph alignment.

7 Related Work

7.1 Reading Comprehension

Many solutions to reading comprehension are learning-based. They either manually define and extract features for learning a model to predict the correct answer (Sachan et al., 2015; Smith et al., 2015; Wang et al., 2015; Sachan and Xing, 2016), or use deep learning with attention mechanisms (Hermann et al., 2015; Trischler et al., 2016; Wang et al., 2016; Chen et al., 2016; Chen et al., 2017; Cui et al., 2017; Dhingra et al., 2017; Samothrakis et al., 2017). These methods have achieved encouraging results on simple factoid questions, but are not optimized for complex questions that require integrating information from multiple sentences. By comparison, our approach is distinguished by the explicit representation and reasoning over temporal and causal relations between events which can span multiple sentences, therefore exactly targeting complex questions and featuring an explainable problem solving process. Due to its unsupervised nature, our approach can be rapidly deployed for a new domain or application.

We are not the first to address complex questions. Sachan and Xing (2016) leverage Abstract Meaning Representation (AMR) to construct graph representations and solve a graph containment problem, but they report that their method suffers from the lack of complex reasoning support (e.g., temporal, causal), which are provided in our approach. Narasimhan and Barzilay (2015) incorporate discourse information

into a discriminative framework with hidden variables that capture relevant sentences and their relations, which is different from our explicit representation of events and their relations extracted from text, and is inferior to our approach in the experiments. Berant et al. (2014) model the relations between events in a biological process using a graph representation, which is similar to our approach. However, to generate a graph, they train dedicated classifiers whereas we use dependencies produced by off-the-shelf parsers. Besides, for question answering, they simply use a few hand-crafted regular expressions over graphs whereas we more thoughtfully formulate and solve a maximum weighted bipartite matching problem.

7.2 Event Extraction

Event extraction has attracted considerable research attention (Pustejovsky et al., 2003; Ahn, 2006; Rudinger and Van Durme, 2014; Mostafazadeh et al., 2016; Mirza and Tonelli, 2016; Judea and Strube, 2016). We extend some of those existing methods (Rudinger and Van Durme, 2014) for event recognition. As to relation extraction, our rule-based method uses dependencies, whereas there are also learning-based solutions (Ahn, 2006; Chambers et al., 2007; Mirza and Tonelli, 2016). Although our focus is on the reading comprehension task, event extraction is at the core of our approach. Advances in this research area will be beneficial to the performance of our approach.

8 Conclusion

To answer complex questions in reading comprehension tasks, we generate event graphs from text for modeling temporal and causal relations spanning multiple sentences. With graph-based reasoning, our self-adaptive GTCR approach helps advance the state of the art on two standard datasets, showing the effectiveness and generalizability of the proposed technical contribution. Furthermore, this graph-based event representation and reasoning complies with human cognition, thereby being capable of not only answering a question but also providing a human-readable explanation. This will expand the scope of application, ranging from explainable question answering to interactive computer-aided education.

Many components of GTCR build upon the results of dependency parsing. Featuring an unsupervised framework, our proposed approach can be conveniently transferred to different domains and applications without the necessity of preparing dedicated training data. Besides, its performance will progress along with the improvement in off-the-shelf dependency parsers. However, as revealed by our post-experiment error analysis, there are still several other directions for improving and extending our approach, to turn it into a more effective add-on to other methods or even a powerful stand-alone solution.

Acknowledgments

This work was supported in part by the NSFC under Grants 61772264 and 61572247, and in part by the Qing Lan and Six Talent Peaks Programs of Jiangsu Province.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8. Association for Computational Linguistics.

Roy Bar-Haim, Ido Dagan, and Jonathan Berant. 2015. Knowledge-based textual inference via parse-tree transformations. *J. Artif. Intell. Res.(JAIR)*, 54:1–57.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 173–176. Association for Computational Linguistics.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2358–2367.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1870–1879.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-attention neural networks for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 593–602.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1832–1846.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Lynette Hirschman, Marc Light, Eric Breck, and John D Burger. 1999. Deep read: A reading comprehension system. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 325–332. Association for Computational Linguistics.

Alex Judea and Michael Strube. 2016. Incremental global event extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2279–2289.

Dan Jurafsky and James H Martin. 2009. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794.

Tao Li and Vivek Srikumar. 2016. Exploiting sentence similarities for better alignments. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2193–2203.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering*, 20(2):151–184.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Paramita Mirza and Sara Tonelli. 2016. Catena: Causal and temporal relation extraction from natural language texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 64–75.

Nasrin Mostafazadeh, Alyson Grealish, Nathanael Chambers, James Allen, and Lucy Vanderwende. 2016. Caters: Causal and temporal relation scheme for semantic annotation of event structures. In *Proceedings of the Fourth Workshop on Events*, pages 51–61.

Karthik Narasimhan and Regina Barzilay. 2015. Machine comprehension with discourse relations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1253–1262.

Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, and Hiyan Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 705–713. Association for Computational Linguistics.

James Pustejovsky, José M Castano, Robert Ingria, Robert J Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R Radev. 2003. Timeml: Robust specification of event and temporal expressions in text.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203.

Rachel Rudinger and Benjamin Van Durme. 2014. Is the stanford dependency representation semantic? In *Proceedings of the Second Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 54–58.

Mrinmaya Sachan and Eric Xing. 2016. Machine comprehension using rich semantic representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 486–492.

Mrinmaya Sachan, Kumar Dubey, Eric Xing, and Matthew Richardson. 2015. Learning answer-entailing structures for machine comprehension. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 239–249.

Spyridon Samothrakis, Tom Vodopivec, Michael Fairbank, and Maria Fasli. 2017. Convolutional-match networks for question answering. International Joint Conferences on Artificial Intelligence.

Ellery Smith, Nicola Greco, Matko Bosnjak, and Andreas Vlachos. 2015. A strong lexical matching method for the machine comprehension test. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1693–1698. Association for Computational Linguistics.

Saku Sugawara, Hikaru Yokono, and Akiko Aizawa. 2017. Prerequisite skills for reading comprehension: Multi-perspective analysis of mctest datasets and systems. In *AAAI*, pages 3089–3096.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2016. Dls @ cu at semeval-2016 task 1: Supervised models of sentence similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 650–655.

Adam Trischler, Zheng Ye, Xingdi Yuan, Jing He, and Philip Bachman. 2016. A parallel-hierarchical model for machine comprehension on sparse data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 432–441.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 700–706.

Bingning Wang, Shangmin Guo, Kang Liu, Shizhu He, and Jun Zhao. 2016. Employing external rich knowledge for machine comprehension. In *IJCAI*, pages 2929–2925.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1744–1753.

Projecting Embeddings for Domain Adaptation: Joint Modeling of Sentiment Analysis in Diverse Domains

Jeremy Barnes[1,2], **Roman Klinger**[1], and **Sabine Schulte im Walde**[1]
[1]Institut für Maschinelle Sprachverarbeitung
University of Stuttgart
Pfaffenwaldring 5b, 70569 Stuttgart, Germany
{barnesjy,klinger,schulte}@ims.uni-stuttgart.de

[2]Grup de Lingüística Computacional
Universitat Pompeu Fabra
Roc Boronat 138, 08018 Barcelona, Spain
jeremy.barnes@upf.edu

Abstract

Domain adaptation for sentiment analysis is challenging due to the fact that supervised classifiers are very sensitive to changes in domain. The two most prominent approaches to this problem are structural correspondence learning and autoencoders. However, they either require long training times or suffer greatly on highly divergent domains. Inspired by recent advances in cross-lingual sentiment analysis, we provide a novel perspective and cast the domain adaptation problem as an embedding projection task. Our model takes as input two mono-domain embedding spaces and learns to project them to a bi-domain space, which is jointly optimized to (1) project across domains and to (2) predict sentiment. We perform domain adaptation experiments on 20 source-target domain pairs for sentiment classification and report novel state-of-the-art results on 11 domain pairs, including the Amazon domain adaptation datasets and SemEval 2013 and 2016 datasets. Our analysis shows that our model performs comparably to state-of-the-art approaches on domains that are similar, while performing significantly better on highly divergent domains. Our code is available at https://github.com/jbarnesspain/domain_blse

Title and Abstract in Basque

Domeinu-Egokitzapenerako Bektore Proiekzioa:
Domeinu Urrunetarako Sentimenduen Analisiko Eredu Bateratua

Sentimenduen analisirako domeinu-egokitzapena erronka handi bat da oraindik, domeinu arteko ezberdintasunek ondorio esanguratsuak izan baititzakete sailkatzaile gainbegiratuentzat. Arazo honi aurre egiteko bi hurbilpen arrakastatsuenak egiturazko kidetasunaren ikasketa (structural correspondence learning) eta autoencoder-ak dira. Hala ere, denbora asko behar dute sistema entrenatzeko edo, domeinu arteko distantzia handia denean, ez dituzte emaitza onak lortzen. Hizkuntza-arteko sentimenduen analisian egindako azken lanetan oinarrituta, ikuspuntu berri bat eskaintzen dugu, domeinuaren egokitzapen ataza bektore proiekzio ataza gisa planteatuta. Gure sistemaren sarrera domeinu banako bi bektore espazio dira, zeinak sistemak espazio berri batera proiektatzen ikasten duen. Sistema hau optimizatuta dago (1) domeinu batetik besterako proiekzioa egiteko eta (2) esaldi baten sentimendua aurresateko. 20 jatorri-xede domeinu pareetan esperimentuak burutu ditugu eta 11 kasutan artearen egoerako emaitzarik onenak lortzen ditugu Amazon-eko domeinu-egokitzapeneko eta SemEval 2013 eta 2016 datu-multzoetan. Gure analisian ikus daitekeenez, gure hurbilpena artearen egoerako sistemen pareko moldatzen da antzeko domeinuetan, baina emaitza hobeak lortzen ditu oso domeinu ezberdinetan. Gure kodea eskuragarri dago helbide honetan: https://github.com/jbarnesspain/domain_blse.

Proceedings of the 27th International Conference on Computational Linguistics, pages 818–830
Santa Fe, New Mexico, USA, August 20-26, 2018.

1 Introduction

One of the main limitations of current approaches to sentiment analysis is that they are sensitive to differences in domain. This leads to classifiers that, after training, perform poorly on new domains (Pang and Lee, 2008; Deriu et al., 2017). Domain adaptation techniques provide a solution to reduce the discrepancy and enable models to perform well across multiple domains (Blitzer et al., 2007). The two main approaches to domain adaptation for sentiment analysis are *pivot-based methods* (Blitzer et al., 2007; Pan et al., 2010; Yu and Jiang, 2016), which augment the feature space with domain-independent features learned on unsupervised data, and *autoencoder* approaches (Glorot et al., 2011; Chen et al., 2012), which seek to create a good general mapping from a sentence to a latent hidden space. While pivot-based domain adaptation methods are well-motivated, they are often outperformed by autoencoder methods. However, both approaches to domain adaptation effectively lead to a loss of information, as they must reduce the effect of discriminant features which are domain-dependent. We argue in this paper that this leads to a decreased performance especially in cases where the similarity between the domains is low.

Unlike previous approaches, in this paper, we propose a domain adaptation approach based on lessons learned from cross-lingual sentiment analysis (Barnes et al., 2018). This approach maintains the domain-dependent features, while adapting them to the target domain. Following state-of-the-art approaches to create bilingual word embeddings (Mikolov et al., 2013; Artetxe et al., 2016; Artetxe et al., 2017), we learn to project a mapping from a source domain vector space to the target domain space, while jointly training a sentiment classifier for the source domain.

We show that our proposed model (1) performs comparably to state-of-the-art models when domains are similar and (2) outperforms state-of-the-art models significantly on divergent domains. We report novel state-of-the-art results on 11 domain pairs. We also contribute a detailed error analysis and compare the effect of different projection lexicons. Our code is available at `https://github.com/jbarnesspain/domain_blse`.

2 Related Work and Motivation

Domain adaptation is an omnipresent challenge in natural language processing. It has been applied for many tasks, such as part-of-speech tagging (Blitzer et al., 2006; Daume III, 2007), parsing (Blitzer et al., 2006; Finkel and Manning, 2009; McClosky et al., 2010), or named entity recognition (Daume III, 2007; Guo et al., 2009; Yu and Jiang, 2015). In the following, we limit our review to adaptation techniques which have been applied to sentiment analysis.

2.1 Pivot-based Approaches

Blitzer et al. (2006) propose *structural correspondence learning* (SCL), which introduces the concept of *pivots*. These are features that behave in the same way for discriminative learning for both domains, *e. g.*, *good* or *terrible* for sentiment analysis. The intuition is that non-pivot domain-dependent features, *e. g.*, *well-written* for the book domain or *reliable* for electronics, which are highly correlated to a pivot should be treated the same by a sentiment classifier.

Blitzer et al. (2007) extend their SCL approach to sentiment analysis and also create one of the benchmark datasets for domain adaptation in sentiment analysis. They crawl between 4000 and 7000 product reviews for each domain, and create balanced datasets of 1000 positive and 1000 negative reviews for four product types (books, DVD, electronics, and kitchen appliances). The remaining reviews serve as unlabeled training data for the SCL approach. For each pivot, they train a binary classifier to predict the existence of the pivot from non-pivot features. They then use these classifiers to create a domain-independent representation of the data. The concatenation of the original representation and the SCL representation are used to train a classifier.

Pan et al. (2010) also exploit the relationship between pivots and non-pivots to span the domain gap, but use a graph-based approach to cluster non-pivot features and augment the original feature space. Yu and Jiang (2016) learn sentence embeddings that are useful across domains through multi-task learning. They jointly train a convolutional recurrent neural network model to predict the sentiment of source domain sentences while at the same time predicting the presence of pivots. Finally, Ziser and Reichart

(2017) propose neural structural correspondence learning (NSCL), which marries SCL and autoencoder techniques by using a neural network to create a hidden representation of a text, and then using this representation to predict the existence of pivots.

NSCL is currently state of the art, but requires a careful choice of pivot features and extensive hyper-parameter search to achieve the best results.

2.2 Autoencoder Approaches

Glorot et al. (2011) adopt a deep learning approach for domain adaptation. They create lower-dimensional representations for their data through the use of *stacked denoising autoencoders* (SDA), which are trained to reconstruct the original sentence from a corrupted version. They then train a linear SVM on the original feature space augmented with the hidden representations obtained from the autoencoder.

Chen et al. (2012) extend this work by proposing *Marginalized Denoising Autoencoders* (MSDA), which are more scalable thanks to a series of linear transformations which are performed in closed-form, with the non-linearity being applied afterwards. This leads to a significant gain in speed, as well as the ability to include more features from the original representations. Autoencoder models perform better than earlier SCL models (excluding NSCL), but have the disadvantages of being less interpretable, requiring long training times, and only utilizing a small amount of the original feature space.

2.3 Domain Specific Word Representations

A third approach is to create word representations that provide useful features for multiple domains. He et al. (2011) propose a joint sentiment-topic model which uses pivots to change the topic-word Dirichlet priors. Bollegala et al. (2015) create domain-specific embeddings for pivots and non-pivots with the constraint that the pivot representations are similar across domains.

The work that is most similar to ours is that of Bollegala et al. (2014). Their method learns to predict differences in word distributions across domains by learning to project lower-dimensional SVD representations of documents across domains. Unlike our work, however, they learn the projection step separately from the classification. They also only learn to project the features that the two domains have in common, which implies discarding information useful for classification. These approaches, however, perform worse than MSDA and NSCL.

3 Projecting Representations

Our approach is motivated by previous success in learning to project embeddings across languages for cross-lingual sentiment analysis, namely *Bilingual Sentiment Embeddings* (Barnes et al., 2018, BLSE). The inputs for this model are (1) a monolingual embedding space for the source and target language, (2) a translation lexicon, and (3) an annotated sentiment corpus for the source language. It jointly learns to project the source and target vectors into a bilingual space and also to classify the sentiment of the sentences in the source corpus. At test time, the classifier is able to use the projected target words as features because they are optimized such that they resemble the source words.

In this work, we cast domain adaptation for sentiment analysis as a version of this cross-lingual adaptation in which the source and target domains have a large shared vocabulary. However, as is the case in domain adaptation, words do not necessarily have the same semantics across domains. Therefore, we will use the aforementioned projection model to learn a word-level projection from one domain to another, while jointly learning to classify the source domain. In the following, we detail the projection objective, the sentiment objective, and the full objective.

Cross-domain Projection

We assume that we have two precomputed vector spaces $S = \mathbb{R}^{v \times d}$ and $T = \mathbb{R}^{v' \times d'}$ for our source and target domains, where v (v') is the length of the source vocabulary (target vocabulary) and d (d') is the dimensionality of the embeddings. We also assume that we have a projection lexicon L of length n which consists of word-to-word pairs $L = \{(s_1, t_1), (s_2, t_2), \ldots, (s_n, t_n)\}$ which map from source to target domains. In this paper, we assume that the words map to themselves across domains, so that

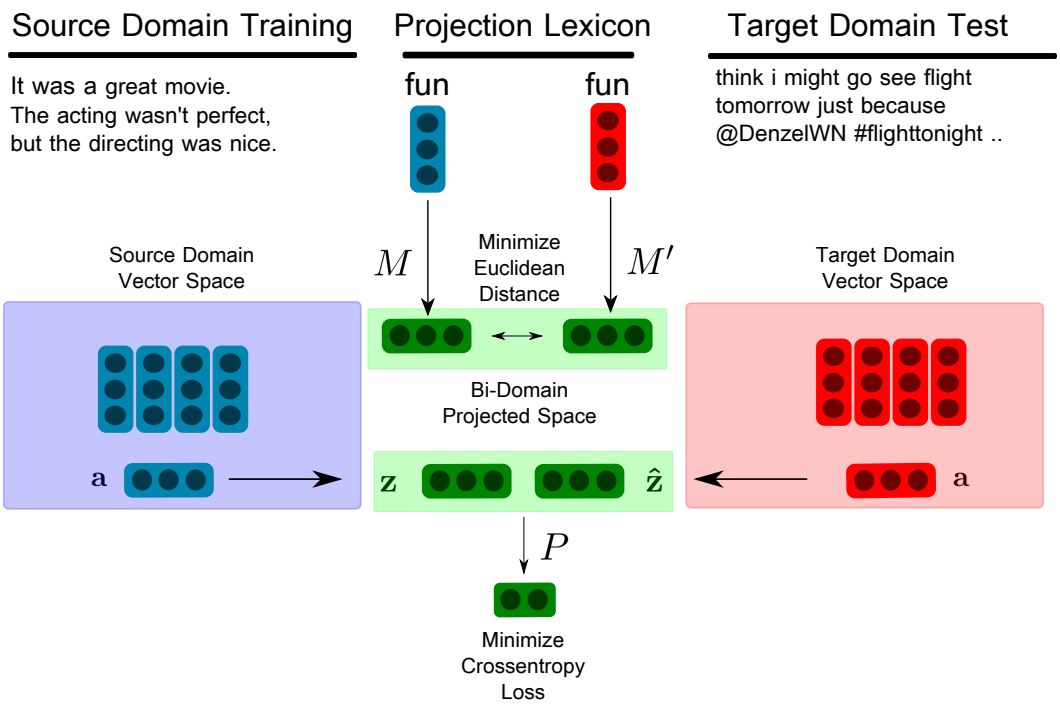


Figure 1: Embedding projection architecture.

$L = \{(s_1, s_1), (s_2, s_2), \ldots, (s_n, s_n)\}$, although other mapping lexicons are possible. One could imagine constructing a lexicon that maps concepts from one domain to those of another, *i. e.* "read" in the books domain and "watch" for movies.

In order to create a mapping from both original vector spaces S and T to shared sentiment-informed bi-domain spaces $\mathbf{z}$ and $\hat{\mathbf{z}}$, we employ two linear projection matrices, M and M'. During training, for each translation pair in L, we first look up their associated vectors, project them through their associated projection matrix and finally minimize the mean squared error of the two projected vectors. This is very similar to the approach taken by Mikolov et al. (2013), but includes an additional target projection matrix.

The projection quality is ensured by minimizing the mean squared error[1]

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{z_i} - \hat{\mathbf{z}}_\mathbf{i})^2\,, \tag{1}$$

where $\mathbf{z_i} = S_{s_i} \cdot M$ is the dot product of the embedding for source word s_i and the source projection matrix and $\hat{\mathbf{z}}_\mathbf{i} = T_{t_i} \cdot M'$ is the same for the target word t_i and target matrix M'.

The intuition for including this second matrix is that a single projection matrix does not support the transfer of sentiment information from the source domain to the target domain. Although this term is degenerate by itself, when coupled with the sentiment objective, it allows the model to learn to project to a sentiment-aware target language space.

Sentiment Classification

We add a second training objective to optimize the projected source vectors to predict the sentiment of source phrases. This inevitably changes the projection characteristics of the matrix M and consequently M', which encourages M' to learn to predict sentiment without any training examples in the target domain.

To train M to predict sentiment, we require a source-domain corpus $C_{\text{source}} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i)\}$ where each sentence x_i is associated with a label y_i.

For classification, we use a multi-layer feed-forward architecture. For a sentence x_i, we take the word embeddings from the source embedding S and average them to $\mathbf{a}_i \in \mathbb{R}^d$. We then project this vector to

[1]We omit parameters in equations for better readability.

		B	D	E	K	S13	S16
Train	+	800	800	800	800	2,225	2,468
	−	800	800	800	800	831	664
Dev	+	200	200	200	200	328	682
	−	200	200	200	200	163	310
Test	+	1000*	1000*	1000*	1000*	946	5,619
	−	1000*	1000*	1000*	1000*	316	2,386
Total		2,000	2,000	2,000	2,000	4,809	12,129

Table 1: Statistics for the Amazon corpora (books, DVD, electronics, kitchen), as well as the SemEval 2013 and 2016 message classification tasks (S13 and S16 respectively). * For the Amazon corpora, we test on the entire target domain corpora.

the joint bi-domain space $\mathbf{z}_i = \mathbf{a}_i \cdot M$. Finally, we pass $\mathbf{z}_i$ through a softmax layer P to get our prediction $\hat{y}_i = \text{softmax}(\mathbf{z}_i \cdot P)$.

To train our model to predict sentiment, we minimize the cross-entropy error of our predictions

$$H = -\sum_{i=1}^{n} y_i \log \hat{y}_i - (1 - y_i) \log(1 - \hat{y}_i)\,. \tag{2}$$

Joint Learning

In order to jointly train both the projection component and the sentiment component, we combine the two loss functions to optimize the parameter matrices M, M', and P by

$$J = \sum_{(x,y) \in C_{\text{source}}} \sum_{(s,t) \in L} \alpha H(x, y) + (1 - \alpha) \cdot \text{MSE}(s, t)\,, \tag{3}$$

where α is a hyperparameter that weights sentiment loss vs. projection loss.

Target-domain Classification

For inference, we classify sentences from a target-domain corpus C_{target}. As in the training procedure, for each sentence, we take the word embeddings from the target embeddings T and average them to $\mathbf{a}_i \in \mathbb{R}^d$. We then project this vector to the joint space $\hat{\mathbf{z}}_i = \mathbf{a}_i \cdot M'$. Finally, we pass $\hat{\mathbf{z}}_i$ through a softmax layer P to get our prediction $\hat{y}_i = \text{softmax}(\hat{\mathbf{z}}_i \cdot P)$.

4 Experiments

We compare our method with two adaptive baselines and one non-adaptive version. We describe the six evaluation corpora in Section 4.1 and the baselines in Section 4.3.

4.1 Datasets

4.1.1 Amazon Corpora

In order to evaluate our proposed method, we use the corpus collected by Blitzer et al. (2007), which consists of Amazon product reviews from four domains: books (B), DVD (D), electronics (E), and kitchen (K). Each subcorpus contains a balanced labeled subset, with 1000 positive and 1000 negative reviews, as well as a much larger set of unlabeled reviews. We use the standard split of 1600 reviews from each domain as training data and the remaining 400 reviews as validation data. For testing, we use all of the 2000 reviews from the target domain (Ziser and Reichart, 2017).

We take the unlabeled data from each domain to create the domain embeddings for our method, as well as to train the domain independent representations for the NSCL and MSDA methods. In order to create embeddings for the Amazon corpora, we concatenate all of the unlabeled data from all domains. The statistics for this corpus are given in Table 1.

4.1.2 SemEval Corpora

Sentiment analysis of Twitter data is common nowadays, with several popular shared tasks organized on the topic (Nakov et al., 2013; Villena-Román et al., 2013; Basile et al., 2014; Nakov et al., 2016, *i. a.*). In order to evaluate how well domain adaptation techniques perform on large domain gaps, we also use the message polarity classification corpora provided by the organizers of SemEval 2013 and 2016 (Nakov et al., 2013; Nakov et al., 2016). We will refer to these as S13 and S16, respectively. These contain tweets which have been annotated for positive, negative, and neutral sentiment. We remove neutral tweets, giving us a binary setup which allows compatibility with the Amazon corpora. The statistics for these corpora are given in Table 1.

4.2 Embeddings

For BLSE, we create mono-domain embeddings using the Word2Vec toolkit[2] by training skip-gram embeddings with 300 dimensions, subsampling of 10^{-4}, window of 5, negative sampling of 15 on the concatenation of the unlabeled Amazon corpora. We also create Twitter-specific embeddings by training on nearly 8 million tokens taken from tweets collected using various hashtags. The parameters were the same as those used to create the Amazon embeddings. For out-of-vocabulary words, a vector initialized randomly between -0.25 and 0.25 approximates the variance of the pretrained vectors.

4.3 Baselines and Model

Domain transfer for sentiment analysis has been widely studied on the Amazon sentiment domain corpus. However, we hypothesize that progress previous approaches have made on this particular corpus may not hold when tested on more divergent domains. Therefore, we compare two state-of-the-art approaches on the Amazon corpus with our method, as well as a standard non-adaptive baseline.

NOAD is a non-adaptive approach which uses a bag-of-words representation from each review as features for a linear SVM.

MSDA is the original implementation of marginalized Stacked Denoising Autoencoders (Chen et al., 2012), one of the state-of-the-art domain adaptation methods on the Amazon sentiment domain corpus. The approach learns a latent hidden representation of the data, which is then concatenated to the original feature space. For our experiments, we use the 30000 most common uni- and bi-grams as features and take the top 5000 features as pivots (Chen et al., 2012). We tune the corruption level (0.5, 0.6, 0.7, 0.8, 0.9) and the C-parameter for the SVM classifier on the source domain validation data, but leave the number of layers at 5.

NSCL is an approach that marries both the pivot-based methods and autoencoders. Specifically, we use the original implementation[3] of the Autoencoder SCL with similarity regularization, which we refer to as NSCL. This approach substitutes the reconstruction weights of the autoencoder with a matrix of the pre-trained word embeddings of pivots. This enables the model to generalize beyond boolean features. We set the hyper-parameters for training the autoencoders with stochastic gradient descent to those from the original paper[4] and tune the number of pivots (100, 200, 300, 400, 500), dimensionality of the hidden layer (100, 300, 500), and C-parameter for logistic regression on the source domain validation data (400 reviews).

BLSE is our approach based on cross-domain vector projection. We use the domain-specific word embeddings to initialize our model and following the embedding literature, we take the most common 20,000 words in the concatenated corpora as a projection dictionary (see Section 5.3). We tune the hyper-parameters training epochs, alpha (0.1–0.9), and batch sizes (20–500) on the source domain validation data.

4.4 Results

Tables 2 and 3 present the results of our experiments. In order to compare with previous work, we report accuracy scores for the balanced Amazon corpora. Because the SemEval corpora are highly imbalanced,

[2] `https://code.google.com/archive/p/word2vec/`

[3] `https://github.com/yftah89/Neural-SCL-Domain-Adaptation`

[4] Learning rate: 0.1, momentum: 0.9, weight-decay regularization: 10^{-5}

	D→B	E→B	K→B	B→D	E→D	K→D	B→E	D→E	K→E	B→K	D→K	E→K
BLSE	**82.2**	71.3	69.0	81.0	**76.8**	**76.5**	71.8	70.3	70.8	73.8	72.3	78.3
NSCL	77.3	71.2	**73.0**	**81.1**	74.5	76.3	**76.8**	**78.1**	**84.0**	**80.1**	**80.3**	**84.6**
MSDA	76.1	**71.9**	70.0	78.3	71.0	71.4	74.6	75.0	82.4	78.8	77.4	84.5
NOAD	73.6	67.9	67.7	76.0	69.2	70.2	70.0	70.9	81.6	74.0	73.2	82.4

Table 2: Sentiment classification accuracy for the Blitzer et al. (2007) task.

	B→S13	D→S13	E→S13	K→S13	B→S16	D→S16	E→S16	K→S16
BLSE	**65.8**	**67.1**	**65.6**	**63.9**	**65.2**	**66.1**	**67.0**	**62.8**
NSCL	62.8	60.6	59.2	50.7	61.5	61.9	60.7	57.6
MSDA	52.2	45.3	48.8	53.2	53.1	43.1	48.2	55.6
NOAD	61.6	61.5	60.9	51.8	59.6	63.2	59.3	54.2

Table 3: Sentiment classification macro F_1 for the SemEval 2013 and 2016 tasks in binary setup.

we instead present macro F_1 scores. We introduce the notation X→Y, where X is the train corpus and Y is the test corpus, to indicate the domain pairs.

On the Amazon corpora, Table 2, NSCL outperforms the other approaches (3.6 percentage points (pp) in F_1 on average compared to BLSE, 2.5 pp compared to MSDA, and 5.1 pp compared to NOAD). BLSE only performs better than NSCL on three setups (DVD to books, electronics to DVD, and kitchen to DVD) and MSDA on four setups (DVD to books, books to DVD, electronics to DVD, and kitchen to DVD). BLSE performs better on the books and DVD test sets than the electronics and kitchen test sets (an average of 3.23 pp). This can be explained by the fact that the corpora used to train the Amazon embeddings contain many more unlabeled reviews for books and electronics (973,194 / 122,438 respectively) than electronics and kitchen (21,009 / 17,856). Consequently, the vector representations for sentiment words that only appear in the books and DVD subcorpora are of higher quality than those that only appear in the electronics and kitchen subcorpora (see Table 4). BLSE relies entirely on the embeddings as input, and if the quality of the embeddings is lower, the model cannot use these features to correctly classify a review. This suggests that the amount of available unlabeled data in the target domain is important, but not limiting. In this paper, we did not decide to crawl more data for the electronics and kitchen domains, but this would be relatively straightforward.

For the SemEval corpora (see Figures 2 and 3), BLSE significantly outperforms all other models (8.2, 15.5, and 6.4 F_1 better on average compared to NSCL, MSDA, and NOAD, respectively). NSCL is better than MSDA on 7 of the 8 setups, but better than the NOAD baseline on only 4. MSDA performs particularly poorly here and only outperforms the baseline on one setup. We suspect that this may be caused by the substantial differences in the source and target corpora and the way this affects the representation given to the classifier, which we explore in more detail in Sections 5.1 and 5.2.

5 Model Analysis

In this section we examine aspects of our model in an attempt to shed light on its strengths and weaknesses. Specifically, we observe how our model performs on highly divergent datasets, perform an error analysis, motivate our choice of projection lexicon, and motivate the need for M'.

5.1 Domain Divergence and Feature Sparsity

From the initial results, it seems that the BLSE model performs better on more divergent domains when compared to other state-of-the-art models. In order to analyze this further, we test the similarity of our domains using the Jensen-Shannon Divergence, which is a smoothed, symmetric version of the Kullback-Leibler Divergence, $D_{KL}(A||B) = \sum_i^N a_i \log \frac{a_i}{b_i}$. Kullback-Leibler Divergence measures the difference between the probability distributions A and B, but is undefined for any event $a_i \in A$ with zero

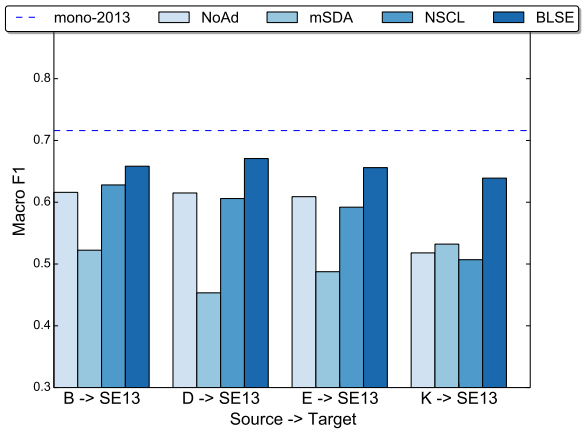


Figure 2: F_1 of approaches trained on the source dataset and tested on the 2013 SemEval corpus.

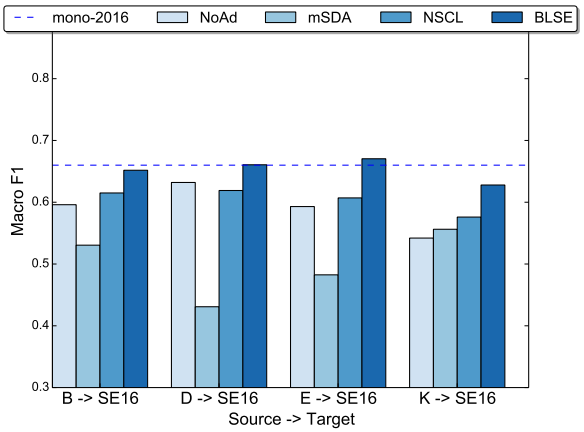


Figure 3: F_1 of approaches trained on the source dataset and tested on the 2016 SemEval corpus.

	books		electronics	
word	admires	conceit	indispensable	cumbersome
neighbors	professes	conceits	career.this	choppiness
	unselfish	macgruffen	non-western	setups
	parminder	pretentiously	mindwalk	forgiveable
	well-liked	contrivance	all-too-rare	unweildy

Table 4: Words and their nearest neighbors for important domain-dependent sentiment words. The nearest neighbors for the two example words from the book domain are more coherent than those of the electronics domain.

probability, which is common in term distributions. Jensen-Shannon Divergence is then

$$D_{JS}(A, B) = \frac{1}{2}\Big[D_{KL}(A||B) + D_{KL}(B||A)\Big] .$$

Our similarity features are probability distributions over terms $t \in \mathbb{R}^{|V|}$, where t_i is the probability of the i-th word in the vocabulary V.

For each domain, we create frequency distributions of the most frequent 10,000 unigrams that all domains have in common and measure the divergence with D_{JS}. The results in Table 5 make it clear that the SemEval datasets are more distant from the Amazon datasets than the Amazon datasets are from each other. This is especially true for the distance between the SemEval datasets from the kitchen dataset (D_{JS} = 0.741 and 0.761, respectively). This suggests that NSCL and MSDA give the best results when the difference between domains is relatively small, whereas BLSE performs better on more divergent datasets.

On the SemEval datasets, BLSE also benefits from using dense representations, rather than the sparse unigram and bigram features of NSCL and MSDA. This is particularly important when you have less domain overlap and smaller texts (the average number of features for the Amazon corpora is 76, compared to 17 for SemEval). BLSE is always able to find useful features, even if the tweet is quite short, whereas a bag-of-words representation can be so sparse that it is not helpful.

5.2 Error Analysis

We perform a label-based error analysis of the models on the SemEval 2013 and 2016 datasets by checking the error rate for the positive and negative classes, which we define as

$$\text{Error Rate} = \frac{e_c}{n_c} , \tag{4}$$

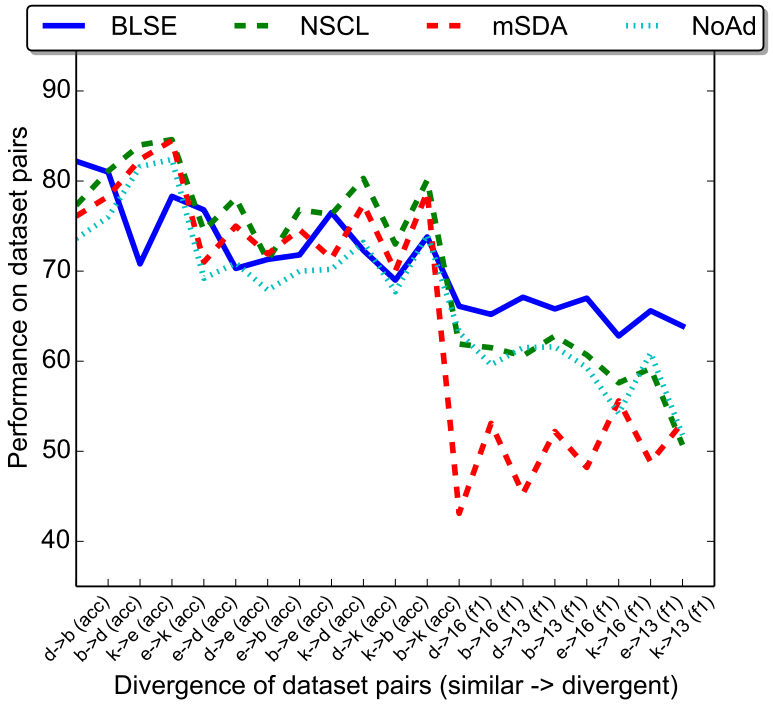


Figure 4: Plot of performance of each model as a function of domain similarity. The x axis plots the rank of similarity from most similar (left) to least similar (right). BLSE maintains its performance as the similarity decreases.

where the number of errors e_c in class c is divided by the total number of examples n_c in the class. We hypothesize that better models will suffer less on minority classes. The results are found in Table 6. In general, BLSE has better overall performance than NSCL or MSDA. In fact, MSDA performs very poorly on the minority negative class, with error rates reaching 98 percent. NSCL almost always favors a single class, with error rates as high as 60.4 on negative and 70.5 on positive.

5.3 Choice of Projection Lexicon

Given that the choice of projection lexicon is one of the key parameters in the BLSE model, we experiment with three approaches to creating a projection lexicon and observe their effect on the books to SemEval 2013 setup.

The **Most Frequent Source Words** are a common source of projection lexicon in the multilingual embedding literature (Faruqui and Dyer, 2014; Lazaridou et al., 2015). For our experiment, we take the 20,000 most frequent tokens from the Brown corpus (Francis and Kučera, 1979). The hypothesis behind using a general corpus is that a large general lexicon will provide more supervision than a smaller task-specific lexicon. This should contribute to learning accurate projection matrices M and M'.

Sentiment Lexicons often contain domain-independent words that convey sentiment. In our model, using a sentiment lexicon as a translation dictionary is equivalent to the use of pivots in other frameworks, as these are usually domain independent words with are good predictors of sentiment. The hypothesis here is that a small task-specific lexicon will help to learn a good projection for the most discriminative words. For our experiment, we take the subset of the sentiment lexicon from Hu and Liu (2004) which is found in the Amazon and SemEval corpora. The final version has 1130 words.

Mutual Information Selected Pivots have been shown to be a good predictor of sentiment across domains (Blitzer et al., 2006; Pan et al., 2010; Ziser and Reichart, 2017). We experiment with using words with the highest mutual information scores as a projection lexicon, although this leads to smaller lexicons.

	book	DVD	electronics	kitchen	SemEval 2013	SemEval 2016
book	1.000	**0.940**	0.870	0.864	0.775	0.802
DVD		1.000	0.873	0.866	0.790	0.814
electronics			1.000	**0.908**	0.748	0.769
kitchen				1.000	0.741	0.761
SemEval 2013					1.000	**0.921**
SemEval 2016						1.000

Table 5: Jensen-Shannon divergence between term distribution representations of datasets. The **bold** numbers represent the most similar domains and underlined numbers represent the most divergent.

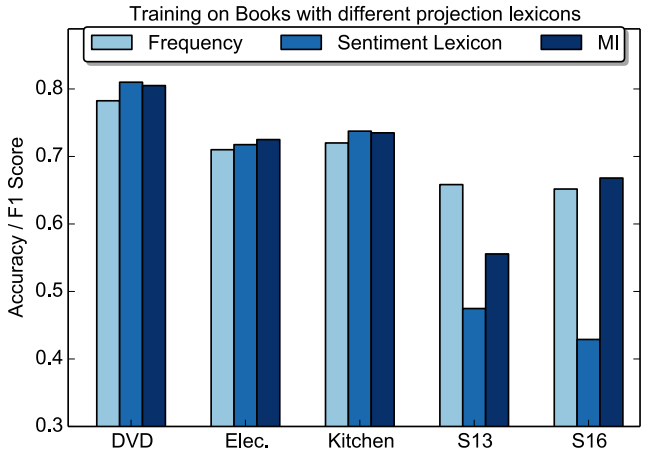


Figure 5: The effect of different projection lexicons for the BLSE method when training on the books domain.

		SemEval 2013		SemEval 2016	
		Pos	Neg	Pos	Neg
books	BLSE	26.9	35.4	31.8	33.0
	NSCL	34.2	43.0	28.9	43.5
	MSDA	1.4	92.7	1.4	89.5
DVD	BLSE	22.8	39.2	28.0	36.5
	NSCL	18.1	60.4	18.7	52.1
	MSDA	0.2	97.8	0.2	98.2
elec.	BLSE	19.1	48.7	27.7	34.6
	NSCL	35.2	41.5	38.9	33.7
	MSDA	1.1	93.7	0.8	91.6
kitchen	BLSE	21.6	49.1	23.3	50.1
	NSCL	63.6	19.3	70.5	13.2
	MSDA	2.4	90.5	2.4	85.8

Table 6: Error rates for positive and negative classes for BLSE, NSCL, and MSDA trained on the Amazon corpora and tested on the SemEval corpora.

Our hypothesis is that specific source-target domain lexicons may provide a better projection between the two specific domains. For each source and target domain pair, we take unigrams and bigrams with high mutual information scores that appear at least 10 times in both domains. The number of pivots differs with each domain pair. The lowest number is 100 (DVD to SemEval 2013) and the highest 955 (books to DVD), with an average of 470 per domain pair.

Figure 5 shows that the frequency-based lexicon gives better results on the more divergent datasets, while the sentiment lexicon performs slightly better on the similar datasets, but poorly on the divergent datasets. The mutual information induced pivot lexicons provide good results on all but the SemEval 2013 dataset. This is likely because the lexicon is too small (103 tokens) to give a good mapping.

5.4 Analysis of M'

The main motivation for using two projection matrices M and M' is to allow the original embeddings to remain stable, while the projection matrices have the flexibility to align translations and separate these into distinct sentiment subspaces. To justify this design decision empirically, we perform an experiment to evaluate the actual need for the target language projection matrix M': We create a simplified version of our model without M', using M to project from the source to target and then P to classify sentiment.

The results of this model are shown in Figure 6. The modified model does learn to predict in the source language, but not in the target language. This confirms that M' is necessary to transfer sentiment in our model.

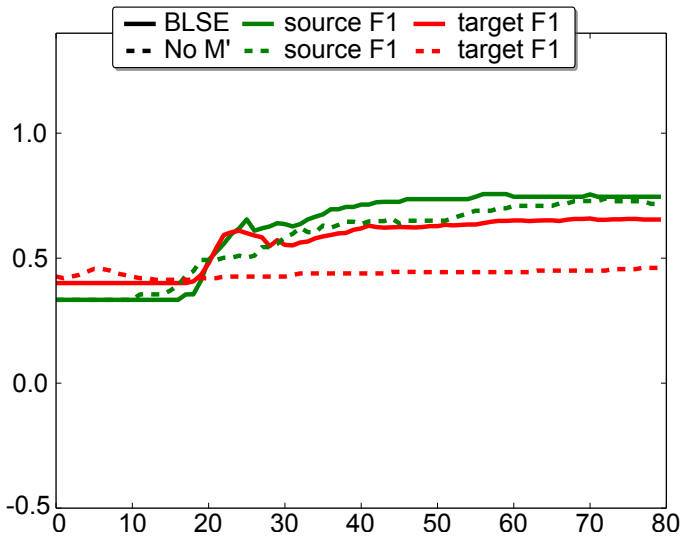


Figure 6: The BLSE model (solid lines) compared to a variant without the target domain projection matrix M' (dashed lines). The red green lines show F_1 on the source books dataset, while the green lines show F_1 on the target SemEval 2013 dataset. The modified model does not learn to predict sentiment on the target dataset (dashed red line).

6 Conclusion and Future Work

We have presented an approach to domain adaptation which learns to project mono-domain embeddings to a bi-domain space and use this bi-domain representation to predict sentiment. We have experimented with 20 domain pairs and shown that for highly divergent domains, our model shows substantial improvement over state-of-the-art methods. Our model constitutes a novel state of the art on 11 of the 20 domain pairs.

One of the main advantages of this approach is that the learned classifier can be used to classify sentiment in either of the two domains without further tuning. In the future, we would like to extend this model to learn multiple domain mappings at a time, effectively permitting zero-shot domain adaptation at a large scale. This would enable a single model to predict sentiment for a number of domains.

Another promising avenue for improvement is to create lexicons that map concepts from the source domain to the target domain, *i. e.*, "read" in the books domain to "watch" in the movies domain. It would be interesting to see if it is possible to use vector algebra (Mikolov et al., 2013) to find similar concepts in different domains, *e. g.*, *read - books + DVD = watch*. It would also be beneficial to map multiword units across domains, *e. g.*, "not particularly exciting" in DVD to "not very reliable" in electronics. This could be particularly helpful for moving beyond a binary view of sentiment at document-level, where domain adaptation would be of particular use, given that the cost of annotation is higher for multi-class, sentence-, or aspect-level classification.

A current disadvantage of our model might be that it uses skip-gram embeddings trained on more than one domain. Therefore, it it would be of interest to investigate if methods which create domain specific embeddings (He et al., 2011; Bollegala et al., 2014; Bollegala et al., 2015) are able to give better results within our framework.

Acknowledgements

We thank Manex Agirrezabal for proofreading the Basque abstract.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294, Austin, Texas, November. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Vancouver, Canada, July. Association for Computational Linguistics.

Jeremy Barnes, Roman Klinger, and Sabine Schulte im Walde. 2018. Bilingual sentiment embeddings: Joint projection of sentiment across languages. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia, July. Association for Computational Linguistics.

Valerio Basile, Andrea Bolioli, Malvina Nissim, Viviana Patti, and Paolo Rosso. 2014. Overview of the evalita 2014 sentiment polarity classification task. In *Proceedings of the 4th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA14)*, pages 50–57, Pisa, Italy. Pisa University Press.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128, Sydney, Australia, July. Association for Computational Linguistics.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, Prague, Czech Republic, June. Association for Computational Linguistics.

Danushka Bollegala, David Weir, and John Carroll. 2014. Learning to predict distributions of words across domains. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 613–623, Baltimore, Maryland, June. Association for Computational Linguistics.

Danushka Bollegala, Takanori Maehara, and Ken-ichi Kawarabayashi. 2015. Unsupervised cross-domain word representation learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 730–740, Beijing, China, July. Association for Computational Linguistics.

Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pages 1627–1634, USA. Omnipress.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June. Association for Computational Linguistics.

Jan Milan Deriu, Martin Weilenmann, Dirk Von Gruenigen, and Mark Cieliebak. 2017. Potential and limitations of cross-domain sentiment classification. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 17–24, Valencia, Spain, April. Association for Computational Linguistics.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, Gothenburg, Sweden, April. Association for Computational Linguistics.

Jenny Rose Finkel and Christopher D. Manning. 2009. Hierarchical bayesian domain adaptation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 602–610, Boulder, Colorado, June. Association for Computational Linguistics.

W. Nelson Francis and Henry Kučera. 1979. The Brown Corpus: A Standard Corpus of Present-Day Edited American English. Brown University Liguistics Department.

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

Honglei Guo, Huijia Zhu, Zhili Guo, Xiaoxun Zhang, Xian Wu, and Zhong Su. 2009. Domain adaptation with latent semantic association for named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 281–289, Boulder, Colorado, June. Association for Computational Linguistics.

Yulan He, Chenghua Lin, and Harith Alani. 2011. Automatically extracting polarity-bearing topics for cross-domain sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 123–131, Portland, Oregon, USA, June. Association for Computational Linguistics.

Minqing Hu and Bing Liu. 2004. Mining opinion features in customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004)*, pages 168–177.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 270–280, Beijing, China, July. Association for Computational Linguistics.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 28–36, Los Angeles, California, June. Association for Computational Linguistics.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168. http://arxiv.org/abs/1309.4168.

Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 312–320, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1–18, San Diego, California, June. Association for Computational Linguistics.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 751–760, New York, NY, USA. ACM.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Julio Villena-Román, Sara Lana-Serrano, Eugenio Martínez-Cámera, and José Carlos González-Cristóbal. 2013. Tass - workshop on sentiment analysis at sepln. *Procesamiento del Lenguaje Natural*, 50(0):37–44.

Jianfei Yu and Jing Jiang. 2015. A hassle-free unsupervised domain adaptation method using instance similarity features. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 168–173, Beijing, China, July. Association for Computational Linguistics.

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 236–246, Austin, Texas, November. Association for Computational Linguistics.

Yftah Ziser and Roi Reichart. 2017. Neural structural correspondence learning for domain adaptation. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 400–410, Vancouver, Canada, August. Association for Computational Linguistics.

Cross-lingual Argumentation Mining: Machine Translation (and a bit of Projection) is All You Need!

Steffen Eger, Johannes Daxenberger, Christian Stab, and Iryna Gurevych
Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
http://www.ukp.tu-darmstadt.de

Abstract

Argumentation mining (AM) requires the identification of complex discourse structures and has lately been applied with success monolingually. In this work, we show that the existing resources are, however, not adequate for assessing cross-lingual AM, due to their heterogeneity or lack of complexity. We therefore create suitable parallel corpora by (human and machine) translating a popular AM dataset consisting of persuasive student essays into German, French, Spanish, and Chinese. We then compare (i) annotation projection and (ii) bilingual word embeddings based direct transfer strategies for cross-lingual AM, finding that the former performs considerably better and almost eliminates the loss from cross-lingual transfer. Moreover, we find that annotation projection works equally well when using either costly human or cheap machine translations. Our code and data are available at http://github.com/UKPLab/coling2018-xling_argument_mining.

1 Introduction

Argumentation mining (AM) is a fast-growing research field with applications in discourse analysis, summarization, debate modeling, and law, among others (Peldszus and Stede, 2013a). Recent studies have successfully applied computational methods to analyze monological argumentation (Wachsmuth et al., 2017; Eger et al., 2017). Most of these studies view arguments as consisting of (at least) claims and premises—and so do we in this work. Thereby, our focus is on token-level *argument component extraction*, that is, the segmentation and typing of argument components.

AM has thus far almost exclusively been performed *monolingually*, e.g. in English (Mochales-Palau and Moens, 2009), German (Eckle-Kohler et al., 2015), or Chinese (Li et al., 2017). Working only monolingually is problematic, however, because AM is a difficult task even for humans due to its dependence on background knowledge and parsing of complex pragmatic relations (Moens, 2017). As a result, acquiring (high-quality) datasets for new languages comes at a high cost—be it in terms of training and/or hiring expert annotators or querying large crowds in crowd-sourcing experiments. It is thus of utmost importance to train NLP systems in AM that are capable of going cross-language, so that annotation efforts do not have to be multiplied by the number of languages of interest. This is in line with current trends in NLP, which increasingly recognize the possibility and the necessity to work cross-lingually, be it in part-of-speech tagging (Zhang et al., 2016), dependency parsing (Agic et al., 2016), sentiment mining (Chen et al., 2016; Zhou et al., 2016), or other fields.

In this work, we address the problem of cross-lingual (token-level) AM for the first time. We initially experiment with available resources in English, German, and Chinese. We show that the existing datasets for analyzing argumentation are not suitable for assessing cross-lingual component extraction due to their heterogeneity or lack of complexity. Given this scarcity of homogeneously annotated high-quality large-scale datasets across different languages, our first contribution is to (1) provide a fully parallel (en-de), *human-translated* version of one of the most popular current AM datasets, namely, the English dataset of persuasive student essays published by Stab and Gurevych (2017).

Proceedings of the 27th International Conference on Computational Linguistics, pages 831–844
Santa Fe, New Mexico, USA, August 20-26, 2018.

We then (2) *machine translate* the 402 student essays into German, Spanish, French, and Chinese. Both our human and machine translations contain argumentation annotations, in the form of either human annotations or automatically projected annotations. Our experiments indicate that both the translations and the projected annotations are of very high quality, cf. examples in Table 2.

Besides contributing new datasets, (3) we perform the first evaluations of cross-lingual (token-level) AM, based on suitable adaptations of two popular cross-lingual techniques, namely, *direct transfer* (McDonald et al., 2011) and *projection* (Yarowsky et al., 2001). We find that projection works considerably better than direct transfer and almost closes the cross-lingual gap, i.e., cross-lingual performance is almost on par with in-language performance when we use parallel data and project annotations to the target language. This holds both for human (translated, HT) parallel data, which is costly to obtain, and machine translated (MT) parallel data, which is very cheap to obtain for dozens of high-resource languages.

Our findings imply that current neural MT has reached a level where it can act as a substitute for costly (non-expert) HT even for problems that operate on the fine-grained token-level.

2 Related Work

In what follows, we briefly summarize the works that most closely relate to our current research.

Argumentation Mining AM seeks to automatically identify argument structures in text and has received a lot of attention in NLP lately. Existing approaches focus, for instance, on specific subtasks like argument unit segmentation (Ajjour et al., 2017), identifying different types of argument components (Mochales-Palau and Moens, 2009; Habernal and Gurevych, 2017), recognizing argumentative discourse relations (Nguyen and Litman, 2016) or extracting argument components and relations end-to-end (Eger et al., 2017). However, most of these approaches are specifically designed for English and there are only few resources for other languages. For German, a few datasets annotated according to the claim-premise scheme are available (Eckle-Kohler et al., 2015; Liebeck et al., 2016). Furthermore, Peldszus and Stede (2015) annotated a small German dataset with claims and premises and translated it to English subsequently. There are very few works studying AM in other languages, e.g. Basile et al. (2016) for Italian, Li et al. (2017) for Chinese and Sardianos et al. (2015) for Greek. There are also two recent papers addressing some form of cross-linguality: Aker and Zhang (2017) map argumentative sentences from English to Mandarin using machine translation in comparable Wikipedia articles. Sliwa et al. (2018) create corpora in Balkan languages and Arabic by labeling the English side of corresponding parallel corpora on the sentence level and then using the same label for the target sentences. In contrast to these works, we work on the more challenging token-level. Moreover, we actually train classifiers for language transfer rather than only creating annotated data in other languages based on parallel data.

As mentioned, we focus on *cross-lingual component extraction*, that is, the segmentation and typing of argument components in a target language (L2), while having only annotated source language (L1) data. We operate on token-level by labeling each token with a BIO label plus its respective component type. The BIO label marks the start, continuation and end of specific argument components. Examples are given in Tables 2 and 3.

Cross-lingual sequence tagging POS tagging and named-entity recognition (NER) are standard tasks in NLP. In recent years, there is increased interest not only in evaluating POS and NER models within multiple individual languages (Plank et al., 2016), but also cross-lingually (Zhang et al., 2016; Tsai et al., 2016; Mayhew et al., 2017; Yang et al., 2017). Two standard approaches are *projection* (Yarowsky et al., 2001; Das and Petrov, 2011; Täckström et al., 2013) and *direct transfer* (Täckström et al., 2012; Zhang et al., 2016). Projection uses parallel data to project annotations from one language to another. In contrast, in direct transfer, a system is trained in L1 using language independent or shared features and applied without modification to L2.

While these approaches are typically *unsupervised*, i.e., they assume no annotations in L2, there are also *supervised* cross-lingual approaches based on multi-task learning (Cotterell and Heigold, 2017; Yang et al., 2017; Kim et al., 2017). These assume small training sets in L2, and a system trained on them is regularized by a larger amount of training data in L1. In our work, we only consider unsupervised

Name	Docum.	Tokens	Sentences	Major Cl.	Cl.	Prem.	Genre	Lang.
MTX	112	8,865 (en)	449	-	112	464	short texts	en, de
CRC	315	21,858	957	135	1,415	684	reviews	zh [en]
PE	402	148,186 (en)	7,141	751	1,506	3,832	persuasive essays	en [de, fr, es, zh]

Table 1: Statistics for datasets used in this work. Languages in brackets added by the current work.

Orig-EN	In the end , I think [**any great success need great work not great luck**] , even though [luck is one factor in reaching goal] but [*its impact is extraneous and we must not reckon on luck in our plans*] .
HT-DE-HumanAnno	Schließlich denke ich , dass [**jeder große Erfolg auf harter Arbeit statt Glück beruht**] , obwohl [Glück ein Faktor in der Erreichung des Ziels ist] , jedoch [*ist dessen Auswirkung unwesentlich und wir sollten uns nicht in unseren Projekten auf unser Glück verlassen*] .
HT-DE-ProjAnno	Schließlich denke ich , dass [**jeder große Erfolg auf harter Arbeit statt Glück beruht , obwohl Glück**] [ein Faktor in der Erreichung des Ziels ist] , jedoch [*ist dessen Auswirkung unwesentlich und wir sollten uns nicht in unseren Projekten auf unser Glück verlassen*] .
MT-DE-ProjAnno	Am Ende denke ich , dass [**jeder große Erfolg große Arbeit erfordert , nicht viel Glück**] , auch wenn [Glück ein Faktor beim Erreichen des Ziels] [*ist , aber seine Auswirkungen sind irrelevant und wir dürfen nicht mit Glück in unseren Plänen rechnen*] .
MT-ES-ProjAnno	Al final , creo que [**cualquier gran éxito requiere un gran trabajo y no mucha suerte**] , aunque la [suerte es un factor para alcanzar el objetivo] , pero [*su impacto es extraño y no debemos tener en cuenta la suerte en nuestros planes*] .
MT-FR-ProjAnno	En fin de compte , je pense que [**tout grand succès a besoin d' un bon travail , pas de chance**] , même si la [chance est un facteur d' atteinte de l' objectif] , mais [*son impact est étranger et nous ne devons pas compter sur la chance dans nos plans*] .
MT-ZH-ProjAnno	最后, 我认为[**任何伟大的成功都需要伟大的工作, 而不是运气好**] , 即使 [运气是达成目标的一个因素] , [但其影响是无关紧要的, 我们不能算计划中的运气] 。

Table 2: Human-annotated English sentence in the PE dataset as well as translations with human-created and projected annotations. Major claims in bold, claims underlined, premises in italics. HT/MT =human/machine translation.

language adaptation, because it is the most realistic cross-lingual scenario for AM, as it may be costly to even produce small amounts of training data in many different languages.

Most cross-lingual sequence tagging approaches address POS tagging, and only few are devoted to NER (Mayhew et al., 2017; Tsai et al., 2016), aspect-based sentiment classification (Lambert, 2015), or even more challenging problems such as discourse parsing (Braud et al., 2017). While POS tagging and NER are in some sense very similar to AM, namely, insofar as both can be modeled as sequential tagging of tokens, there are also important differences. For example, in POS tagging and NER, the label for a current token usually strongly depends on the token itself plus some local context. This strong association between label, token and local context is largely absent in AM, causing some models that perform well on POS and NER to fail blatantly in AM.[1]

Cross-lingual Word Embeddings are the (modern) basis of the direct transfer method. As with monolingual embeddings, there exists a veritable zoo of different approaches, but they often perform very similarly in applications (Upadhyay et al., 2016) and seemingly very different approaches are oftentimes also equivalent on a theoretical level (Ruder et al., 2017).

3 Data

We chose three freely available datasets: a small parallel German-English dataset, and considerably larger English and Chinese datasets using (almost) the same inventory of argument types, which we therefore assumed to be adequate for cross-lingual experiments. We translated the two last named monolingual datasets in other languages, described below. Statistics for all datasets are given in Table 1.

[1]E.g., we had tried out a word embedding based HMM model (Zhang et al., 2016) in initial experiments but found it to perform below our random baseline. The apparent reason is that an HMM cannot deal with long-range dependencies that abound in AM.

Orig-ZH	几次入住中国大饭店, [**感觉都非常不错**] , [新开的豪华阁酒廊非常棒] , [饮料丰富] , [食物也很好吃] , [服务也非常的棒] , [尤其特别感谢杨雪峰和张东静, 他们非常贴心]
MT-EN-ProjAnno	Several times staying at China World Hotel , [**I feel very good**] , the [newly opened Horizon Club Lounge is great] , [rich drinks] , [food is also very good] , [very good service] , [*especially thanks to Yang Xuefeng and Zhang Dongjing , they are very caring*]

Table 3: Review from CRC corpus as well as English machine translation with projected annotations. Major claims in bold, claims underlined, premises in italics (ZH: regular font).

3.1 Microtexts (MTX)

Peldszus and Stede (2015) annotated 112 German short texts (six or less sentences) written in response to questions typically phrased like "Should one do X". These were annotated according to a version of Freeman's theory of argumentation macro-structure (Peldszus and Stede, 2013b). Each microtext consists of one (central) claim and several premises. As opposed to our other datasets, MTX has no "O" (non-argumentative) tokens and no major claims. The German sentences have been professionally translated to English, making this the first parallel corpus for AM in English and German.

3.2 Chinese Review Corpus (CRC)

Li et al. (2017) created the only large-scale argument mining dataset in Chinese, freely available and with annotations on component level according to the claim-premise scheme (Stab and Gurevych, 2017). We thus chose to include this dataset in our experiments, despite differences in the domain of the annotated texts. Li et al. (2017) used crowdsourcing to annotate Chinese hotel reviews from *tripadvisor.com* with four component types (major claim, claim, premise, premise supporting an implicit claim). We consider only those components with direct overlap with the components used by Stab and Gurevych (2017), thus considering components labeled as "premise supporting an implicit claim" as non-argumentative. We applied the CRF-based Chinese word segmenter by Tseng et al. (2005) to split Chinese character streams into tokens. Furthermore, we only use the "Easy Reviews Corpus" from Li et al. (2017). The remaining part of the corpus are isolated sentences from reviews with low overall inter-annotator agreement, which we ignored. An example from CRC can be found in Table 3.

3.3 A Large-Scale Parallel Dataset of Persuasive Essays (PE)

Stab and Gurevych (2017) created a dataset of persuasive essays written by students on *essaysforum.com*. These are about controversial topics such as "competition or cooperation—which is better?". To obtain a human-translated parallel version of this dataset, we asked seven native speakers of German with an attested strong competence in English (all students or university employees) to translate the 402 student essays in the PE corpus sentence-by-sentence. As only requirement, we asked the translators to retain the argumentative structure in their translations: i.e., the translation of an argument component should be connected and not contain non-argumentative tokens. Since German has a freer word order compared to English, this requirement can in virtually all cases be easily fulfilled without producing awkward sounding German translations. Each essay was translated by exactly one translator. Besides translating the essays, we also asked the translators to annotate argument boundaries so that the original mark-up is preserved in the translations. The translators took about 40min on average to translate one essay and indicate the argument structures. Thus, they required about 270 hours to translate the whole PE corpus into German, and the resulting overall cost was roughly 3,000 USD. The motivations to ask translators to translate argument components contiguously were that (i) all monolingual AM datasets we know of have contiguous components, (ii) transfer would have been naturally hampered had components in the source language been contiguous but not in the target language, at least for methods such as direct transfer.[2]

[2]We note that even professional translations typically differ from original, non-translated texts because they retain traces of the source language (Rabinovich et al., 2017). We thus speculate that our reported results are probably slightly upward biased compared to a situation where the test data consists of original German student essays. This latter situation would have been much more costly to produce, in any way: it would have required retrieval (and, if necessary, creation) of original student essays in German as well as induction of all subsequent annotation mark-up.

To obtain further parallel versions of the PE data, we also *automatically* translated them into German, French, Spanish, and Chinese using Google Translate. Of course, we cannot make any demands on how Google Translate translates text into other languages but noticed that it has a tendency to stay rather close to the original text, but nevertheless has a very high perceived quality of translation. We automatically projected argument structures from the English text to the machine translations using our projection algorithm described in §4. It took few hours to automatically translate the PE corpus into the four languages. Examples of the data as well as the human and machine translations can be found in Table 2. Even though we also provide translations of PE in French, Spanish and Chinese, our primary focus in our experiments below is on the languages for which we have gold (human-created) data, i.e., EN↔DE (for PE and MTX) as well as EN↔ZH.

4 Approaches

In what follows, we describe our adaptations of direct transfer and projection to the AM task. Direct transfer focuses on the source language and trains on human-created L1 data as well as human-created L1 labels. In contrast, during training, projection operates directly on the language of interest, viz., L2. This comes at a cost: the labels in L2 are noisy, because they are projected from L1, which is an error-prone process. The success of projection can therefore be expected to largely depend on the quality of this transfer step. Projection makes stronger assumptions than direct transfer: it requires parallel data.[3] When the parallel data is induced via machine translation, then a second source of noise for projection is the 'unreliable' L2 input training data.

Direct Transfer Here, we directly train a system on bilingual representations, which in our case come in the form of bilingual word embeddings. To retain some freedom over the choice and parameters of our word embeddings, we choose to train them ourselves instead of using pre-trained ones. For EN↔DE we induce bilingual word embeddings by training BIVCD (Vulic and Moens, 2015) and BISKIP models (Luong et al., 2015) on >2 million aligned sentences from the Europarl corpus (Koehn, 2005). BIVCD concatenates bilingually aligned sentences (or documents), randomly shuffles each concatenation and trains a standard monolingual word embedding technique on the result; here, we use the word2vec skip-gram model (Mikolov et al., 2013). BIVCD was shown to be competitive to more challenging approaches in Upadhyay et al. (2016). BISKIP is a variant of the standard skip-gram model which predicts mono- and cross-lingual contexts. It requires word alignments between parallel sentences and we use fast-align for this (Dyer et al., 2013). For EN↔ZH we train the same models on the UN corpus (Ziemski et al., 2016), which comprises >11 million parallel sentences. We train embeddings of sizes 100 and 200.

Projection To implement projection for the problem of token-level AM, we proceed as follows. We take our human-labeled L1 data and align it with its corresponding parallel L2 data using fast-align. Once we have word level alignment information, we consider for each argument component $c(s)$ in L1 of type a (e.g., MajorClaim, Claim, Premise) with consecutive words $s_1, \ldots, s_N$: the word t_1 with smallest index in the corresponding L2 sentence that is aligned to some word in $s_1, \ldots, s_N$, and the analogous word t_{-1} with largest such index. We then label all the words in the L2 sentence between t_1 and t_{-1} with type a, using a correct BIO structure, resulting in $c(t)$. We repeat this process for all the components within a sentence in L1 and for all sentences. In case of collision, e.g., if two components in L2 would overlap according to the above-described strategies, we simply increment the beginning counter of one of the components until they are disjoint. If our above strategy fails, i.e., $c(s)$ cannot be projected, e.g., because the words in an L1 component are not aligned to any words in L2, then we simply ignore the projection of $c(s)$ to the L2 sentence, labeling the corresponding words in L2 as non-argumentative instead. We think of this projection strategy as naive because we do not do much to resolve conflicts and instead trust the quality of the alignments and that the subsequent systems trained on the projected data are capable of gracefully recovering from noise in the projections.

[3]Thus, direct transfer is potentially the cheaper approach, even though it also requires bilingual word embeddings, which themselves are based on some form of bilingual signal, e.g., parallel sentence- or word-level data.

5 Experiments

We perform token-level sequence tagging. Our label space is $\mathcal{Y} = \{\text{B,I}\} \times T \cup \{\text{O}\}$ where T is the set of argument types, comprising "claim", "premise", and (if applicable) "major claim".

5.1 Experimental Setup

To perform token-level sequence tagging, we implement a standard bidirectional LSTM with a CRF layer as output layer in TensorFlow. The CRF layer accounts for dependencies between successive labels. We represent words by their respective embeddings. In addition to this word-based information, we also allow the model to learn a character-based representation (via another LSTM) and concatenate this learned representation to the word embedding. Our model is essentially the same as the ones proposed by Ma and Hovy (2016) and Lample et al. (2016); it is also a state-of-the-art model for monolingual AM (Eger et al., 2017). We name it BLCRF+char, when character information is included, and BLCRF when disabled. For all experiments, we use the same architectural setup: we use two LSTM hidden layers with 100 hidden units each. We train for 50 epochs using a patience of 10. We apply dropout on the embeddings as well as on the LSTM units. On character-level, we also use a bidirectional LSTM with 50 hidden units and learn a representation of size 30. As evaluation measure we choose macro-F1 as implemented in scikit-learn (Pedregosa et al., 2011).

Baseline A simple baseline to test successful learning is to choose the majority label in the test data. However, this performs particularly poorly on token-level and for our chosen evaluation metric. We therefore choose a more sophisticated baseline. We first split our datasets by sentences and then compute a probability distribution of how likely each argument component appears in a sentence. At test time, we again split the test data by sentences and then label each token in the test sentence with a randomly drawn argument component (according to the calculated probability distribution on train/dev sets). We label all the tokens in the sentence with the drawn argument component type, keeping valid BIO structure. We label the last token (which is typically a punctuation symbol) with the "O" label in PE and CRC. In essence, our baseline is a random baseline, but has some basic prior knowledge of the BIO format.

Train/dev/test splits For the PE corpus, we use the same split into training and test data as in Stab and Gurevych (2017). In particular, our test data comprises 80 documents ("essays") with a total of 29,537 tokens (en). We choose 10% of the training data as dev set. Thus, we have 286 essays in the train set with a total of 105,988 tokens (en) and 36 essays in the dev set with a total of 12,657 tokens (en). We report averages over five random initializations of our networks. For the CRC corpus, we perform 5-fold cross-validation on document level. Our train sets consist of roughly 15K tokens, our dev sets of 2K tokens, and our test sets of 4K tokens. For each split, we average over five different random initializations and report the average over these averages. For MTX, we also perform 5-fold cross-validation on document level. Our train sets consist of roughly 6K tokens (en), our dev sets of 500 tokens (en), and our test sets of 1,500 tokens (en). We use the same averaging strategy as for CRC, but average over ten random initializations per fold, to account for the smaller dataset sizes.

5.2 Results

We report results for adapting between datasets in different languages and between parallel versions of one and the same dataset. We only consider cases where one of the involved languages is English. Further, we do not transfer between MTX and the other datasets, because MTX has no "O" units (and no major claims). Unless stated otherwise, we always *evaluate* on `HT` for both direct transfer and projection.

5.2.1 Direct Transfer

For all cross-lingual direct transfer experiments, we train on the union of train and dev (train+dev) sets (randomly drawn for the datasets for which we used cross-validation) of the source language and test either on the whole data (train+dev+test) of L2, or, in case of parallel versions of a dataset (such as PE EN↔DE) on the test set of L2. We do not use `MT` for direct transfer at any stage.

PE_{EN}↔PE_{DE}, results in Table 4: English in-language results do not vary much and are on a level of slightly above 69% macro-F1, largely independent of the embedding types and whether or not character

information is available.[4] German in-language results are 4-5 percentage points (pp) below the English ones. One might suspect the presumed inferior quality (or derivative nature) of the student translations as a cause for this, but we hypothesize that German is simply more complex than English, both in morphology and syntax.

We observe a noticeable drop when moving cross-language. This drop is up to >40% for the direction EN→DE (worst case drop from ∼70% to ∼37%) and slightly less for DE→EN. We explain this drop by the discrepancy between training and test distributions. This discrepancy is present even in bilingual embedding spaces: no test word has the exact same representation as the words in the training data. Further, disabling character information typically has a very positive impact cross-language. For example, EN→DE performance increases from ∼40% F1 to ∼50% when disabling character information. The reason is that a system that extracts a character representation based on English characters may get confused from the diverging German character sequences. Surprisingly, characters do not impede so much in the direction DE→EN. The reason seems to be lexical borrowing in modern German from English. For example, ∼17% of the 'active' vocabulary (i.e., frequency >30) of English in PE_{EN} is also contained in PE_{DE}. In contrast, only 6% of the active vocabulary of German in PE_{DE} occurs also in PE_{EN}.

CRC↔PE_{EN}, results in Table 5 (left): In-language CRC results are lower than in-language PE results (∼46% vs. ∼69% for PE). This is unsurprising since CRC is considerably smaller in size than PE. However, we observe that the cross-language drop is much larger than it is for the PE DE↔EN setting. In fact, performance values always lie below our random baseline. We attribute this huge drop not to the larger language distance between English and Chinese (relative to English and German), but primarily to the domain gap between student essays and hotel reviews. In fact, we observe that, e.g., major claims in PE are almost always preceded by specific discourse markers such as "Therefore, I believe that" or "In the end, I think" (cf. Table 2), while hotel reviews completely lack such discourse connectives (cf. Table 3). Since we expect a system that trains on PE to learn the signaling value of these markers, directly applying this system to text where such markers are absent, fails.

MTX_{EN}↔MTX_{DE}, results in Table 5 (right): Even though the dataset is by far smallest in size it yields the highest F1-scores among all our considered datasets. Moreover, the language drop is comparatively small (between 4 and 7pp). Investigating, we notice that argument components are typically separated by punctuation symbols (mostly "." or ",") in MTX, which is easy to learn even cross-lingually. Moreover, we find that claims can often be separated from premises by simple keywords such as "should", which can, apparently, be reliably spotted cross-lingually via the corresponding bilingual word embeddings.

Model	Embedding Type	EN→EN	EN→DE	DE→DE	DE→EN
BLCRF+Char	BIVCD-100	68.87	41.89	65.22	49.91
	BIVCD-200	**70.51**	39.87	**65.92**	49.52
	BISKIP-100	69.27	37.01	63.33	48.23
BLCRF	BIVCD-100	69.27	49.70	65.90	50.14
	BISKIP-100	69.15	**49.76**	64.92	**50.28**
Baseline		20.	20.	20.	20.

Table 4: Direct transfer results for PE_{EN}↔PE_{DE}. Scores are macro-F1.

Error Analysis and Discussion For PE direct transfer experiments, we find that a major source of errors is incorrect classification of tokens labeled "B-". This means that the system has difficulty finding the exact beginning of an argument span. We find, however, that the reason for the language drop is not that the bilingual embedding spaces are bad: among the top-10 neighbors of English words are roughly five German words, and vice versa. Rather, direct transfer induces a situation very similar to standard monolingual out-of-vocabulary (OOV) scenarios, namely as if all test words had been replaced by synonyms that did not occur in the train data. While systems using embeddings as input are more robust to

[4]Our in-language results are slightly below our previous results reported in Eger et al. (2017) (table 6), where we obtained scores of 72-75% for token-level component extraction, even though the architecture is in principle the same. Reasons may be the different word embeddings used as well as that we reported majority performance over different hyperparameter combinations in the previous work, which typically increased performance scores by a few percentage points.

Model	CRC↔PE_{EN} ZH→ZH	ZH→EN	EN→EN	EN→ZH	MTX_{EN}↔MTX_{DE} EN→EN	EN→DE	DE→DE	DE→EN
BLCRF+Char	**46.31**	14.01	**69.27**	9.50	**73.12**	67.03	**73.41**	**66.62**
BLCRF	44.95	16.52	69.15	12.60	72.15	**69.46**	72.52	63.71
Baseline	18.	**17.**	20.	**17.**	45.	46.	50.	50.

Table 5: Direct transfer results for CRC and MTX. Scores are macro-F1. Embeddings are BISKIP-100.

OOV words, they are still affected by them (Ma and Hovy, 2016; Müller et al., 2013). This "blurring effect" at test time then makes it more difficult to detect exact argument component spans. While this is true in general, it is not true for punctuation symbols, which typically have an identical role across languages and, hence, extremely similar representations. For example, German and English "." have more than 97% cosine similarity in BISKIP-100d, which is much higher than for typical monolingually closely related words. Finally, besides semantic shift direct transfer also faces syntactic shift, because the test words may have different word order compared to the train data (e.g., verb final position in German).

The lessons we learn from our above experiments are that (i) the MTX dataset does not provide a real challenge for cross-lingual techniques because argument components can easily be spotted based on punctuation and component typing appears to be just as easily portable across languages. (ii) Language adaptation between the CRC corpus and PE appears, in contrast, too difficult because argumentation units are very differently realized across the two datasets, and hence, the domain shift appears to be the (much) larger obstacle compared to the language shift.[5] Thus, (iii) we mostly focus on the cross-lingual version of the PE corpus in the sequel, which is a difficult enough dataset for cross-lingual AM, without confounding the problem with issues relating to differences of AM domains. In addition, we only use BISKIP-100d embeddings, because the choice of embeddings seemed to have a negligible effect in our case (certainly, for our main focus, namely, cross-lingual evaluations) and because they showed slightly superior results cross-lingually than BIVCD-100d.

5.2.2 Projection

HT-Projection Table 6 (HT columns) shows results when we project PE_{EN} training data annotations on parallel German HT documents and train and evaluate a system directly on German (and vice versa). As said before, we train in this case directly on the same language as we test on, viz., L2. We observe that this improves cross-language results dramatically. From a best cross-language result of 49.76% for BIVEC-100d in the direct transfer setting, we improve by almost 30% to 63.67%. This is only roughly 1pp below the best in-language result for German which was 64.92%. In the direction DE→EN, we observe the same trend: we improve by over 30% relative to the direct transfer results and achieve a macro-F1 score that is only 1.7pp below the in-language upper-bound.

MT-Projection Next, we investigate what happens when we replace the HT translations with MT translations and perform the same projection steps as before. Results for PE_{EN}↔PE_{DE} are shown in Table 6 (MT columns). We see that EN→DE results get slightly better while DE→EN results get slightly worse. On average, it seems, using machine translations is just as good as using human translations. Moreover, we remain very close to the upper-bound in-language results. The reason why MT results could be better than HT results is that the machine might translate more consistently. It might also be better in certain cases in correcting (the sometimes ungrammatical) English original. Another likely reason is that current MT has reached, if not already surpassed, HT of non-expert (but bilingually fluent) human translators.[6]

Motivated by this finding, we also machine translated the CRC corpus into English, projected annotations and then trained a system on this translation and evaluated on the PE_{EN} corpus. Results improve to 23.15% macro-F1 score relative to the best direct transfer result of 16.52%. This indicates, again, that

[5] This is a similar finding as in Daxenberger et al. (2017).

[6] We conducted a formal test if MT can reliably be distinguished from HT in our setup. We trained a system (an adaptation of InferSent (Conneau et al., 2017)) to predict whether for an English original e and a second input sentence it could determine if the second input is a human or machine translation of e. The system's performance of 54% accuracy (which is only slightly better than random guessing) matched our own intuition and introspection into the quality of the machine translations.

training in L2 is better than training in L1, given the high quality machine translations and a suitable projection algorithm. However, 23.15% is still only slightly better than the random baseline of 17%—and far from the best PE_{EN} in-language result of >69%. In addition, in the direction PE_{EN}→CRC macro-F1 (also) improves to 15.33% relative to a best direct transfer score of 12.60%. However, here the performance is still below the random baseline of 17%. We take this as strong evidence that the domain gap between CRC and PE is too large and it is not possible to train a system in one of these two domains and directly apply it in the other, even when the language gap has been eliminated.

	EN→DE			DE→EN		
	HT	MT	In-Lang.	HT	MT	In-Lang.
BLCRF+Char	63.67	**64.00**	63.33	**67.57**	66.39	69.27
BLCRF	61.18	63.34	64.92	64.87	64.68	69.15

Table 6: Projection on HT/MT translations, evaluated on human-created test data. Scores are macro-F1. Embeddings are BISKIP-100.

Other languages We conducted a final experiment in which we considered our MT translations of PE into French, Spanish, and Chinese. Since we have no human-created test data for these languages we could only evaluate on *machine translations and projected annotations*. For our BLCRF+char model, we obtained performance scores of 62.45%, 65.92%, 59.20% for French, Spanish and Chinese, respectively. To see if these numbers give reliable estimates of the systems when evaluated on HT data, we performed the same test with German and English and got scores of 63.20% and 61.45%, respectively, when trained and evaluated on MT. For CRC, we also trained and evaluated on the English (MT translated and automatically projected) data, obtaining a score of 47.92% with BLCRF+char, which is slightly above the in-language value of 46.31% on the original Chinese data (see Table 5). That all these numbers are close to the original in-language results gives a good indication that the MT evaluations very likely strongly correlate to 'true' performances on HT data.

Error analysis The bottleneck of projection is the quality of the cross-lingual projections (which in turn depend on the quality of the word alignments between bi-text). We can directly assess our projections by comparing them to the human-created annotations. Our algorithmic projections match in 97.24% of the cases (token-level) with the human gold standard for the direction EN→DE. The corresponding macro-F1 score is 89.85%. Inspecting the confusion matrices, we observe that most mismatches occur between the "B" and "I" categories of a given component type and with the "O" category. These numbers and the mismatch types indicate that there are only few projection errors and they typically lead to either (slightly) larger or smaller argument components than given in the human-created data. To illustrate, typical projection errors arise in case of missing articles in one of the two languages involved; cf. Table 2: "luck is [...]" vs. "la chance est [...]". Here, it is likely that the alignment algorithm does not align the French determiner "la" to an English word, and thus, "la" is not included in the argument component. Another case of too short argument components is that of verb final position in German which often gets misaligned and the corresponding final verb omitted from the argument component. These misalignments lead to slightly "shifted" argument components in the L2 train set and are the most prominent source of error for the projection technique. To quantify: the German in-language system classifies 58% of all cases when one of the determiners ("der/die/das") begins an argument component correctly, but the system using projections from the English data only classifies 35% of them correctly. Hence, alignment/projection errors indeed propagate to some degree, but these phenomena are rare and have negligible impact on performance. Note that, by design of our projection algorithm, misalignments of words in the 'center' of an argument component are much less likely to be a problem.

Figure 1 plots individual F1 scores for various systems transferring to English on PE. Here, the cross-lingual systems using HT/MT projection perform roughly as well as the in-language system for "O", "I-C" and "I-P". These are the most frequent classes. For claims and major claims, which have lower frequency, the in-language upper bound tends to perform better. Noteworthy, the in-language system is

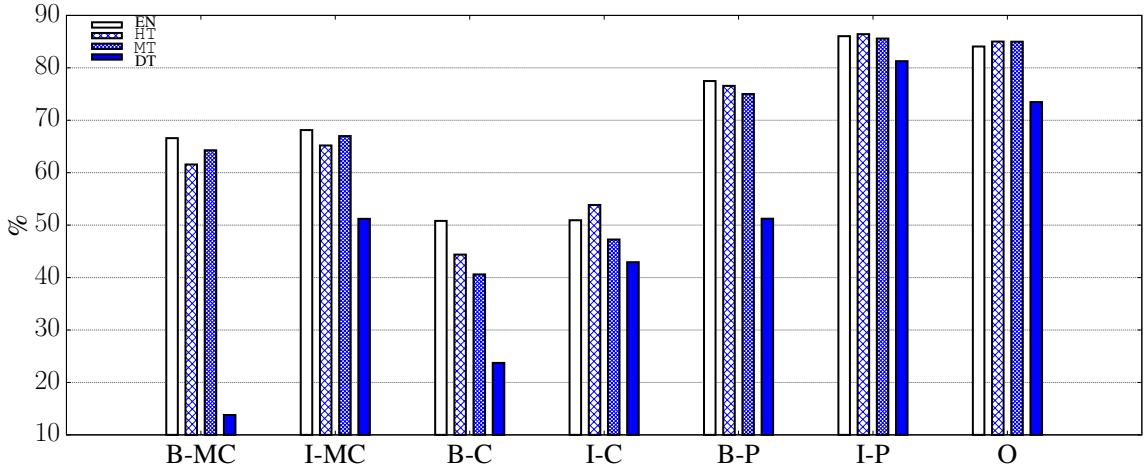


Figure 1: Individual F1-scores for four indicated systems and seven labels. All transfer systems are from PE_{DE} to PE_{EN}; EN is in-language. DT stands for Direct Transfer. HT/MT are projection-based approaches. Embeddings are BISKIP-100. Systems are BLCRF+Char.

always better for the beginnings of an argument component (B-MC,B-C,B-P), which confirms our above analysis. The direct transfer system, in contrast, performs much worse, particularly for major claims and claims, and also for all starts of components, indicating that the blurring ("OOV") effect is here much more severe.

6 Conclusion

Showing that the currently available datasets for AM are not adequate for evaluating cross-lingual AM transfer, we created human and machine translations of one of the most popular current AM datasets, the dataset of persuasive student essays (Stab and Gurevych, 2017). We also machine translated a Chinese corpus of reviews (Li et al., 2017) into English, which provides argumentation structures on hotel reviews. Performing cross-lingual experiments using suitable adaptations of two popular transfer approaches, we have shown that machine translation and (naive) projection work considerably better than direct transfer, even though the former approach contains two sources of noise. Moreover, machine translation in combination with projection almost performs on the level of in-language upper bound results. We think that our findings shed further light on the value—and the huge potential—of current (neural) machine translation systems for cross-lingual transfer. They also cast doubt on current standard use of direct transfer in cross-lingual scenarios. Instead, we propose to simply machine translate the train set, when this is possible, and then project labels to the translated text. This eliminates the (particular) "OOV" and "ordering" problems inherent to direct transfer. Prerequiste to this approach is high quality MT, which, with the advent of neural techniques, appears to be now available.

We hope our new datasets fuel AM research in languages other than English. In this work, we did not consider cross-lingual argumentative relation identification, although relations are available in the newly created parallel PE and CRC datasets. Future work should explore cross-lingual multi-task learning for AM (Schulz et al., 2018) with the source language as main task and small amounts of labeled target language data, as well as adversarial training techniques (Yasunaga et al., 2018), which promise to be beneficial for the particular OOV problem that direct transfer is prone to (though not for the ordering problem). We also want to combine projection with direct transfer by training on the union of projected L2 data as well as the original L1 data using shared representations.

Acknowledgements

This work has been supported by the German Federal Ministry of Education and Research (BMBF) under the promotional reference 01UG1816B (CEDIFOR) and 03VP02540 (ArgumenText).

References

Zeljko Agic, Anders Johannsen, Barbara Plank, Héctor Martínez Alonso, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *TACL*, 4:301–312.

Yamen Ajjour, Wei-Fan Chen, Johannes Kiesel, Henning Wachsmuth, and Benno Stein. 2017. Unit segmentation of argumentative texts. In *Proceedings of the 4th Workshop on Argument Mining*, pages 118–128, Copenhagen, Denmark, September. Association for Computational Linguistics.

Ahmet Aker and Huangpan Zhang. 2017. Projection of argumentative corpora from source to target languages. In *Proceedings of the 4th Workshop on Argument Mining, ArgMining@EMNLP 2017, Copenhagen, Denmark, September 8, 2017*, pages 67–72.

Pierpaolo Basile, Valerio Basile, Elena Cabrio, and Serena Villata. 2016. Argument mining on italian news blogs. In *Proceedings of Third Italian Conference on Computational Linguistics (CLiC-it 2016) & Fifth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2016), Napoli, Italy, December 5-7, 2016.*

Chloé Braud, Ophélie Lacroix, and Anders Søgaard. 2017. Cross-lingual and cross-domain discourse segmentation of entire documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 237–243.

Xilun Chen, Ben Athiwaratkun, Yu Sun, Kilian Q. Weinberger, and Claire Cardie. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. *Arxiv preprint https://arxiv.org/abs/1612.08994*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*, pages 681–691. Association for Computational Linguistics.

Ryan Cotterell and Georg Heigold. 2017. Cross-lingual character-level neural morphological tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 759–770. Association for Computational Linguistics.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 600–609.

Johannes Daxenberger, Steffen Eger, Ivan Habernal, Christian Stab, and Iryna Gurevych. 2017. What is the Essence of a Claim? Cross-Domain Claim Identification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 2055–2066, September.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648. Association for Computational Linguistics.

Judith Eckle-Kohler, Roland Kluge, and Iryna Gurevych. 2015. On the role of discourse markers for discriminating claims and premises in argumentative discourse. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2236–2242, Lisbon, Portugal.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural end-to-end learning for computational argumentation mining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11–22, Vancouver, Canada, July. Association for Computational Linguistics.

Ivan Habernal and Iryna Gurevych. 2017. Argumentation mining in user-generated web discourse. *Computational Linguistics*, 43(1):125–179.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for pos tagging without cross-lingual resources. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2822–2828. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.

Patrik Lambert. 2015. Aspect-level cross-lingual sentiment classification with constrained smt. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 781–787, Beijing, China, July. Association for Computational Linguistics.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270.

Mengxue Li, Shiqiang Geng, Yang Gao, Shuhua Peng, Haijing Liu, and Hao Wang. 2017. Crowdsourcing Argumentation Structures in Chinese Hotel Reviews. In *Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics*, pages 87–92, Banff, Canada.

Matthias Liebeck, Katharina Esau, and Stefan Conrad. 2016. What to Do with an Airport? Mining Arguments in the German Online Participation Project Tempelhofer Feld. In *Proceedings of the Third Workshop on Argument Mining, hosted by the 54th Annual Meeting of the Association for Computational Linguistics, ArgMining@ACL 2016, August 12, Berlin, Germany*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159. The Association for Computational Linguistics.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Stephen Mayhew, Chen-Tse Tsai, and Dan Roth. 2017. Cheap translation for cross-lingual named entity recognition. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 62–72, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Arxiv preprint https://arxiv.org/abs/1301.3781*.

Raquel Mochales-Palau and Marie-Francine Moens. 2009. Argumentation mining: The detection, classification and structure of arguments in text. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law*, pages 98–107, Barcelona, Spain.

Marie-Francine Moens. 2017. Argumentation mining: How can a machine acquire common sense and world knowledge? *Argument & Computation*. Accepted.

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

Huy Nguyen and Diane Litman. 2016. Context-aware argumentative relation mining. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1127–1137, Berlin, Germany, August. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Andreas Peldszus and Manfred Stede. 2013a. From Argument Diagrams to Argumentation Mining in Texts: A Survey. *International Journal of Cognitive Informatics and Natural Intelligence*, 7(1):1–31.

Andreas Peldszus and Manfred Stede. 2013b. Ranking the annotators: An agreement study on argumentation structure. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 196–204, Sofia, Bulgaria.

Andreas Peldszus and Manfred Stede. 2015. An annotated corpus of argumentative microtexts. In *Argumentation and Reasoned Action: Proceedings of the 1st European Conference on Argumentation*, pages 801–815, Lisbon, Portugal.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*.

Ella Rabinovich, Noam Ordan, and Shuly Wintner. 2017. Found in translation: Reconstructing phylogenetic language trees from translations. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 530–540. Association for Computational Linguistics.

Sebastian Ruder, Ivan Vulic, and Anders Sogaard. 2017. A survey of cross-lingual word embedding models. In *Arxiv preprint https://arxiv.org/abs/1706.04902*.

Christos Sardianos, Ioannis Manousos Katakis, Georgios Petasis, and Vangelis Karkaletsis. 2015. Argument extraction from news. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 56–66, Denver, CO.

Claudia Schulz, Steffen Eger, Johannes Daxenberger, Tobias Kahse, and Iryna Gurevych. 2018. Multi-task learning for argumentation mining in low-resource settings. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–41. Association for Computational Linguistics, June.

Alfred Sliwa, Yuan Man, Ruishen Liu, Niravkumar Borad, Seyedeh Ziyaei, Mina Ghobadi, Firas Sabbah, and Ahmet Aker. 2018. Multi-lingual argumentative corpora in english, turkish, greek, albanian, croatian, serbian, macedonian, bulgarian, romanian and arabic. In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France, may. European Language Resources Association (ELRA).

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659, September.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 477–487, Stroudsburg, PA, USA. Association for Computational Linguistics.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan T. McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *TACL*, 1:1–12.

Chen-Tse Tsai, Stephen D. Mayhew, and Dan Roth. 2016. Cross-lingual named entity recognition via wikification. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016*, pages 219–228.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A Conditional Random Field Word Segmenter for Sighan Bakeoff 2005. In *Fourth SIGHAN Workshop on Chinese Language Processing*.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Ivan Vulic and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *ACL (2)*, pages 719–725. The Association for Computer Linguistics.

Henning Wachsmuth, Giovanni Da San Martino, Dora Kiesel, and Benno Stein. 2017. The Impact of Modeling Overall Argumentation with Tree Kernels. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 17)*, pages 2369–2379. Association for Computational Linguistics, September.

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. *Arxiv preprint https://arxiv.org/abs/1703.06345*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the First International Conference on Human Language Technology Research*, HLT '01, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michihiro Yasunaga, Jungo Kasai, and Dragomir R. Radev. 2018. Robust multilingual part-of-speech tagging via adversarial training. In *Proceedings of NAACL*. Association for Computational Linguistics.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi S. Jaakkola. 2016. Ten pairs to tag - multilingual POS tagging via coarse mapping between embeddings. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1307–1317.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016. Cross-lingual sentiment classification with bilingual document representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, page 1403–1412.

Michał Ziemski, Marcin Junczys-Dowmunt, and Bruno Pouliquen. 2016. The united nations parallel corpus v1.0. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may.

HL-EncDec: A Hybrid-Level Encoder-Decoder for Neural Response Generation

Sixing Wu[*], **Dawei Zhang**[†], **Ying Li** [‡§], **Xing Xie**[†], **Zhonghai Wu**[‡]
[*]School of Software and Microelectronics, Peking University, Beijing, China
[‡]National Research Center of Software Engineering, Peking University, Beijing, China
[†]Microsoft Research Asia, Beijing, China
[§]li.ying@pku.edu.cn

Abstract

Recent years have witnessed a surge of interest on response generation for neural conversation systems. Most existing models are implemented by following the Encoder-Decoder framework and operate sentences of conversations at word-level. The word-level model is suffering from the Unknown Words Issue and the Preference Issue, which seriously impact the quality of generated responses, for example, generated responses may become irrelevant or too general (i.e. safe responses). To address these issues, this paper proposes a hybrid-level Encoder-Decoder model (HL-EncDec), which not only utilizes the word-level features but also character-level features. We conduct several experiments to evaluate HL-EncDec on a Chinese corpus, experimental results show our model significantly outperforms other non-word-level models in automatic metrics and human annotations and is able to generate more informative responses. We also conduct experiments with a small-scale English dataset to show the generalization ability.

1 Introduction

Nowadays, conversation systems have gained a great progress due to the rapid development of big-data and deep learning techniques. A conversation system enables a computer to make human-like conversations with users. Massive conversational datasets can be easily accessed on the Web, which highly promotes both the academia and industry to turn to their attention to develop data-driven conversation systems (Vinyals and Le, 2015; Shang et al., 2015; Chen et al., 2017). A common approach to build a generation-based conversation system is to model it as a response generation task. And a response generation model is often learnt within the Encoder-Decoder framework from large-scale conversational data from the Web.

The Encoder-Decoder is a state-of-the-art framework for SEQ2SEQ (sequence-to-sequence) tasks such as machine translation, abstractive summarization and response generation, etc.. Researchers have proposed several Encoder-Decoder based models, and most of them utilize Recurrent Neural Networks to build an Encoder and an attentional Decoder. Due to the fact that a word is the most basic unit in linguistics, and other larger language elements are built on words, the majority of models operate sentences at word-level. Namely, sentences should be segmented as sequences of words, thus an Encoder-Decoder model could read or generate a sentence with their own vocabularies. Since the computational complexity and the capacity limitation of internal high-speed storage such as GPU VRAM, vocabularies could only record finite words. However, there is an issue that the amount of total words is far more than the volume of vocabularies and the out-of-vocabulary words (denoted as OOVs) may appear in the conversations. Meanwhile, another issue is that words already recorded in the vocabulary have different probabilities to be used in human's dialogues, only a small part of high-frequency words may be learnt well and the remaining low-frequency words may be undertrained. We respectively called the first and

This work is partly supported by National Key R&D Program of China (Grant No. 2017YFB1002002).

Proceedings of the 27th International Conference on Computational Linguistics, pages 845–856
Santa Fe, New Mexico, USA, August 20-26, 2018.

the second issue as Unknown Words Issue and Preference Issue. Intuitively, both issues seriously impact the generation quality and diversity of a response generation model.

Operating sentences at character-level could address the above issues, for the amount of a language's all characters is fixed and small, for example, the alphabet size of English is 128 (ASCII), which allows a vocabulary to record all characters easily. By representing a word as several characters, two issues could be naturally addressed. (Kim et al., 2016) obtained a word embedding by utilizing a convolutional network with character embeddings of that word instead of directly looking up the word embedding matrix of vocabulary. (Costajussa and Fonollosa, 2016) borrowed this idea and proposed a character-level Encoder, but the entire model does not fully work at character-level. (Lee et al., 2016) proposed a fully character-level Encoder-Decoder model, and initially addressed the previous issues. Letting a model work at subword-level is another choice. The BPE (byte-pair-encoding) algorithm proposed by (Sennrich et al., 2016) is able to mine a specific number of most frequent subword units from the corpora and utilizes them to completely represent sentences. (Chung et al., 2016) utilized BPE to build a decoder.

However, these non-word-level models could sharpen the long-term dependencies issue (Hochreiter and Schmidhuber, 1997) and lost much semantic and syntactic information. And those non-word-level approaches are proposed for the machine translation or language modelling instead of response generation task. To the best of our knowledge, there is no existing work that tries to address Unknown Words Issue and Preference Issue for response generation task. In this work, we focus on addressing these two issues for Encoder-Decoder based response generation model by proposing a model named HL-EncDec. HL-EncDec is an improved Encoder-Decoder based model that operates sentences at hybrid-level, which employs characters and words to represent sentences at the same time. In the source side, HL-EncDec is able to utilize a convolutional network to calculate an equivalent word-level embedding vector from characters of that word. Since characters are fully recorded , we could calculate a well-trained equivalent embedding for any word. Moreover, HL-EncDec could also utilize the original word-level embeddings recorded in the matrix by using the sum of equivalent one and original one, this is the main reason why we claim our model works at hybrid-level. In the target side, we also propose a hybrid-level approach to represent a sentence without the increasing of the complexities.

We compare our HL-EncDec with the most widely used standard Encoder-Decoder implementation and other state-of-the-art non-word-level methods using both automatic metrics and human evaluations. The experimental result shows HL-EncDec delivers more high-quality and informative responses compared to baseline models.

2 Background and Motivation

Inspired by SEQ2SEQ approaches for neural machine translation (Sutskever et al., 2014; Bahdanau et al., 2014; Jean et al., 2015), researchers have started extending these techniques to implement generation-based response generation model and already harvested significant improvements. A standard SEQ2SQ model takes a source sentence $X = (x_1, x_2, \ldots, x_T)$ as an input, and generates another sentence $Y = (y_1, y_2, \ldots, y_L)$ as an output. The word-level RNN Encoder-Decoder (denoted as EncDec) is the most famous and applied framework.

In general, an EncDec model has two components: an Encoder first reads a message from user and summarizes this message into several internal representations, and a Decoder utilizes these internal representations to generate a new sentence as a response to reply user's message.

2.1 Encoder

Formally, given a tokenized source sentence $X = (x_1, x_2, \ldots, x_T)$, Encoder reads each word in order and generates hidden states $\mathbf{H} = (\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_T})$ as the internal representation of X with a Recurrent Neural Network (RNN). The hidden state $\mathbf{h_t}$ is considered to have already summarized a slice of words $(x_1, \ldots, x_i)$, which is calculated by:

$$\mathbf{h_t} = f(\mathbf{x_t^w}, \mathbf{h_{t-1}}) \tag{1}$$

where function f is always a long-short term memory unit (LSTM) (Hochreiter and Schmidhuber, 1997) or gate recurrent unit (GRU) (Cho et al., 2014), both of them are able to effectively reduce the impact of

long-term dependencies. In this paper, the LSTM is employed as function f. In addition, suppose that x_t is the i-th word of Encoder's vocabulary , thus the embedding vector $\mathbf{x_t^w}$ of word x_t is obtained by an embedding matrix looking up operation:

$$\mathbf{x_t^w} = lookingup(\mathbf{V^e}, x_t) = \mathbf{V^e}[i] \tag{2}$$

where V^e is Encoder's vocabulary and $\mathbf{V^e} \in \mathbb{R}^{|V^e|*d}$ is the corresponding embedding matrix.

Recently, the biRNN (bi-directional RNN) is a common approach to enhance the Encoder (Chung et al., 2014). A biRNN includes a forward RNN and a backward RNN, the forward RNN reads a sentence in its original order while the backward RNN reads it in the reversed order. Thus, the concatenation $\mathbf{h_t^c}$ of the hidden state $\mathbf{h_t^f}$ generated by forward RNN and the hidden state $\mathbf{h_t^b}$ generated by backward RNN is used to replace the single-directional $\mathbf{h_t}$ in the Equation 1.

2.2 Attentional Decoder

Decoder utilizes the hidden states $\mathbf{H}$, and employs another RNN to predict the conditional probability of the next target word $y_{t'}$ being the k-th word of the vocabulary of Decoder V^d at time $t^{'}$:

$$p(y_{t'} = w_k^d | y_{t'-1}, \ldots, y_1, X) = \frac{exp(z(\mathbf{h_t^{'}}, \mathbf{V^d}[k], \mathbf{c_{t'}}))}{\sum_{i=1}^{|V^d|} exp(z(\mathbf{h_t^{'}}, \mathbf{V^d}[i], \mathbf{c_{t'}}))} \tag{3}$$

Where z is a non-linear is function with activation, and $\mathbf{h_{t'}^{'}}$ is hidden state of Decoder's RNN for time $t^{'}$, which also applies a LSTM or GRU to calculate:

$$\mathbf{h_{t'}^{'}} = f(\mathbf{y_{t'-1}}, \mathbf{h_{t'-1}^{'}}, \mathbf{c_{t'}}) \tag{4}$$

The context vector $\mathbf{c_{t'}}$ is computed as a weighted sum of the hidden states $\mathbf{H} = (\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_T})$:

$$\mathbf{c_{t'}} = \sum_{i=1}^{T} \alpha_{t',i} \mathbf{h_i} \tag{5}$$

where weight $\alpha_{t',i}$ is given by:

$$\alpha_{t',i} = \frac{exp(e_{t',i})}{\sum_{j=1}^{T} exp(e_{t',j})} \tag{6}$$

$$e_{t',j} = a(\mathbf{h_{t'-1}^{'}}, \mathbf{h_j}) \tag{7}$$

Where a is an alignment model which is used to score how well the word x_j of the input sentence and the next target word $y_t^{'}$ could be related. (Bahdanau et al., 2014) and (Luong et al., 2015) have proposed two different but effective alignment models, respectively. This paper takes the model of (Luong et al., 2015) as function a, which is computed by employing a bilinear term $\mathbf{W_a}$:

$$a(\mathbf{h_{t'-1}^{'}}, \mathbf{h_j}) = (\mathbf{h_{t'-1}^{'}})^{\mathbf{T}} \mathbf{W_a} \mathbf{h_j} \tag{8}$$

2.3 Training

An Encoder-Decoder model can be trained end-to-end by minimizing the negative conditional log-likelihood of target Y given source X, which is defined as:

$$\pounds = -\frac{1}{N} \sum_{i=1}^{N} \sum_{t'=1}^{|Y^i|} \log(p(y_{t'}^i | y_{t'-1}^i, \ldots, y_1^i, X^i) \tag{9}$$

Where N is the total count of the given corpus D , (X^i, Y^i) is the i-th pair of D, and $y_{t'}^i$ is the $t^{'}$-th word of Y^i. is the $t^{'}$-th word of Y^i.

2.4 Challenges

As described above, a word-level EncDec model operates input messages and output responses at word-level. A word itself could not be utilized by a computer unless it has been converted into the internal representation such as a one-hot vector or an embedding vector. Both of Encoder and Decoder maintain their own vocabulary, such that they are able to understand the meaning or predict the probability of a word by looking up the embedding matrix of their vocabularies.

More or less, due to the complexity of model itself, deep neural network models are suffering from the limited computational power of hardware and the scarce capacity of internal high-speed storages such as the GPU VRAM. An EncDec response generation model is surely not an exception, hence the capacity of a vocabulary of the Encoder or Decoder must be controlled to a finite and relatively small number, for example, 40K. Consequently, a lot of words become out-of-vocabulary words (denoted as OOVs), and an EncDec model could not correctly understand or generate it. It is a common approach to enable an EncDec model to work with OOVs that utilizing a special and universal symbol **unk** to replace all OOVs. In other words, all OOVs share a same embedding vector in the source side and they will never be correctly generated in the target side. Intuitively, the more OOVs a corpus contains, the lower performance an EncDec conversation model has. The challenge of the appearance of OOVs is named **Unknown Words Issue**, for convenience.

If we only consider the words already been recorded in the vocabulary (i.e. known words), another challenge will rise to the surface, which is called **Preference Issue**. All known words could be understood and generated by an EncDec model, but that doesn't mean they have the same chance to appear in the sentences of dialogues. Some words are very potential to appear in our dialogues (for example, what, that, I, etc.) and some words are rare to be adopted while it is not an OOV word. This paper calls the first kind of words as high-frequency words (HF words) and the second kind as low-frequency words (LF words). The embedding vector of HF words would be well learnt but LF words could not, for HF words have more appearance to enable them to be updated by commonly used gradient descent optimizing algorithms. Thus, an EncDec model may not correctly operate LF words, which may sharpen the phenomenon that an EncDec conversation model always generates some too safe, too general but very boring responses (e.g., I think so).

The above two issues have seriously impacted the overall performance of EncDec response generation models, even models to solve other tasks such as neural machine translation and neural abstractive text summarization. In this paper, we propose a novel model HL-EncDec to solve these issues, which has significantly improved the generation quality and diversity.

3 Related Work

In linguistics, a word is the most basic unit, and other larger language elements such as phrases and sentences, etc., are built upon words. Due to this fact, it is natural that most previous works regard a word as the smallest unit in their models, i.e. these models operate at word-level.

The word-level EncDec models have achieved great improvements on neural machine translations (Bahdanau et al., 2014; Jean et al., 2015), which inspired researchers to apply this idea to response generation task to build a generation based neural conversation systems(Vinyals and Le, 2015; Shang et al., 2015; Xing et al., 2016; Chen et al., 2017). This technique enables a conversation system to be end-to-end trained from a large-scale corpus of message-response pairs, but researchers found it tends to generate some general and 'safe' responses(Vinyals and Le, 2015). To address this challenge, (Li et al., 2016) introduced a new objective function with MMI (maximum mutual information) to penalize too general responses. (Shao et al., 2017) presented a novel attentional model to generate long and diverse responses. (Wu et al., 2017) enhanced the quality and diversity of generated responses by constructing a dynamic vocabulary.

In this paper, we focus on two issues discussed in Section 2: Unknown Words Issue and Preference Issue. Actually, many researchers have raised their attention to these two issues, and proposed models operate at non-word-level such as character-level and subword-level. (Kim et al., 2016) solved the representation of unknown words and selection preference for language modelling by utilizing characters

to calculate an equivalent word embedding instead of looking up a word embedding matrix. (Costajussa and Fonollosa, 2016) absorbed this idea and proposed a character-level Encoder for machine translation. Recently, inspired by (Kim et al., 2016; Costajussa and Fonollosa, 2016), (Lee et al., 2016) introduced a fully character-level EncDec model. (Sennrich et al., 2016) mined subword units to represent all words by applying BPE algorithm. (Chung et al., 2016) utilized the BPE to build a subword-level Encoder and proposed a character-level Decoder.

While a character-level model or a subword-level model may address the Unknown Words Issue, but it could sharpen the long-term dependencies issue and lost much semantic and syntactic information due to each word must be decomposed to several characters or subunits. The loss may more than gains, and we consider this is why most models operate sentences at word-level.

In addition, previous models are designed for machine translation task which is actually a different task compared with response generation while they are both SEQ2SEQ tasks. This paper proposes a model HL-EncDec, which utilizes the hybrid-level representation technique to improve the generation quality for response generation task.

In the source side ,for a word x, HL-EncDec not only utilizes the word-level embedding vector $\mathbf{x^w}$ obtained by looking up embedding matrix, but also calculates an equivalent word-level embedding vector $\mathbf{x^c}$ of x. To calculate the $\mathbf{x^c}$, HL-EncDec employs its corresponding character sequence and a Convolutional Neural Network based network. The Encoder of HL-EncDec maintains two vocabularies V_w^e and V_c^e, the first vocabulary V_w^e records a finite number of words and their embedding vectors, the second vocabulary V_c^e records all characters of the language used by current system. In general, the total number of all characters of a language is fixed and small, hence recently GPU devices are very easy to load all characters to its RAM. For example, there are about 128 characters in English alphabet (ASCII). Obviously, the concept of out-vocabulary-character has gone with the wind, and each OOV word could be represented as a sequence of fully known characters.

4 HL-EncDec

4.1 Encoder

HL-EncDec employs CharCNN to calculate the equivalent word-level embedding $\mathbf{x^c}$. CharCNN is initially presented for language modeling by (Kim et al., 2016).

Given a word x , CharCNN first represents x as a sequence of characters $C_x = (c_1, \ldots, c_{N_x}, c_{eos})$ where N_x is the character number of x and c_{eos} is a special symbol to indicate the end of characters. Each character c_i could be represented as an embedding vector $\mathbf{c_i} \in \mathbb{R}^{d_c}$ by looking up the embedding matrix $\mathbf{V_c^e} \in \mathbb{R}^{|V_c^e| * d_c}$. Subsequently, a narrow convolution is applied upon the character embeddings $\mathbf{C_x} = (\mathbf{c_1}, \ldots, \mathbf{c_{N_x}}, \mathbf{c_{eos}}) \in \mathbb{R}^{d_c * (N_x+1)}$. Assuming there is a single convolution filter $\mathbf{f} \in \mathbb{R}^{d_c \times w}$ with a variable width w, and the height of $\mathbf{f}$ is fixed to d_c, thus each character vector could be completely operated. With the padding and convolutional operation, the obtained feature map is denoted as $\mathbf{f^m} \in \mathbb{R}^{N_x+1}$, and the i-th element of $\mathbf{f^m}[i]$ is given by:

$$\mathbf{f^m}[i] = ReLU(\sum \mathbf{C_{x[:,i:i+w-1]}} \otimes \mathbf{f}) \tag{10}$$

Where $\otimes$ denotes element-wise matrix multiplication, $\mathbf{C_{x[:,i:i+w-1]}}$ is the i-to-$(i+w-1)$-th column of $\mathbf{C_x}$, which is corresponding to the i-to-$(i+w-1)$-th characters of x. Finally, we take the max-over-time pooling technique:

$$\mathbf{f^p} = \max(\mathbf{f^m}[i]), i \in [1, N_x + 1] \tag{11}$$

Hence, $\mathbf{f^p}$ is regarded as the feature corresponding to the filter $\mathbf{f}$. Assume there are N_f unique filters in total, then $\mathbf{u} \in \mathbb{R}^{N_f}$ is used to denoted the concatenation of corresponding N_f features. Highway network (Hinton et al., 2012) has been shown to bring significant improvements to many NLP tasks, We apply a 4-layer highway network to $\mathbf{u}$, and a single layer highway network is calculated as:

$$\mathbf{u_h^1} = g \odot ReLU(\mathbf{W_h u} + \mathbf{b_h}) + (1 - g) \odot \mathbf{u} \tag{12}$$

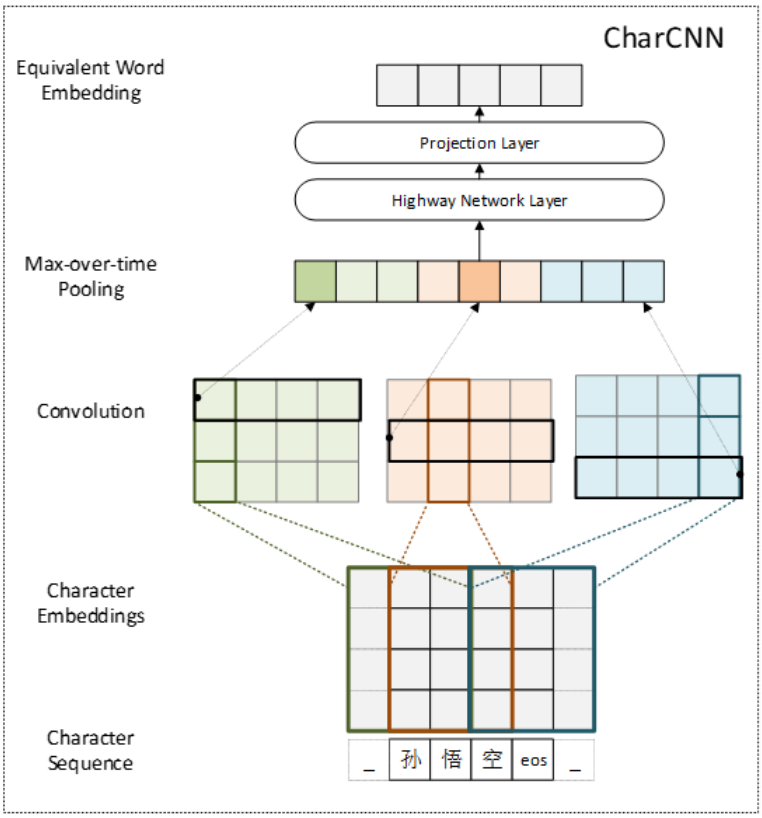


Figure 1: The architecture of CharCNN

where g is called as transform gate, $(1-g)$ is called as carry gate, and $g = sigmod(\mathbf{W_g u} + \mathbf{b_g})$. The equivalent word embedding $\mathbf{x^c}$ is finally worked out by a linear projection:

$$\mathbf{x^c} = \mathbf{u_h^4 W_l} + \mathbf{b_l} \tag{13}$$

Thus, $\mathbf{x^c}$ is the equivalent word embedding calculated by CharCNN, which has the same dimension as $\mathbf{x^w}$ obtained by looking up $\mathbf{V_w^e} \in \mathbb{R}^{|V_w^e| * d_w}$. Then, we calculate the sum of $\mathbf{x^c}$ and $\mathbf{x^w}$:

$$\mathbf{x^h} = \mathbf{x^c} + \mathbf{x^w} \tag{14}$$

Finally, $\mathbf{x^h}$ is the hybrid-level embedding vector for x, and HL-EncDec replaces the word-level $\mathbf{x^w}$ in Equation 2 with $\mathbf{x^h}$. Since $\mathbf{x^c}$ could be generated for any word without the challenge of unknown words and preference, the $\mathbf{x^h}$ could naturally address the Unknown Words Issue and Preference Issue. We also tried to merge $\mathbf{x^w}$ and $\mathbf{x^c}$ with more complex techniques, but this simple summation yield the best result.

4.2 Hybrid-Level Decoder

This paper utilizes a hybrid-level Decoder to avoid the appearance of OOVs and improve the quality and diversity of generated responses. HL-Decoder is able to generate either a word or a character. HL-Decoder has a vocabulary V_d and a corresponding embedding matrix $\mathbf{V_d} \in \mathbb{R}^{|V_d| * d_w}$, which records all characters of the target language, a connector symbol and some HF words. Characters and words are equally recorded in the vocabulary of HL-EncDec, namely HL-Decoder has none explicit operation to judge whether it should generate a character or word, for reducing the complexity of decoding.

While HL-Decoder has another way to segment a sentence into a sequence, HL-Decoder has the same network architecture as the Decoder introduced in Section 2. In the target side of HL-EncDec, a response sentence should be applied a hybrid-level approach to a segment. Assuming there is a tokenized word-level message sequence $Y = (y_1, y_2, \ldots, y_L)$, repeatedly execute the following operation until there is

no OOV word: if y_i is an OOV word, then y_i should be decomposed to several corresponding characters $(c_1, cs, c_2, \ldots, cs, c_{N_y})$, where$(c_1, c_2, \ldots, c_{N_y})$ is the corresponding character sequence of y_i and cs is the connector. Hence, the message sequence could be represented as

$$Y^{'} = (y_1, \ldots, y_{i-1}, c_1, cs, c_2, \ldots, cs, c_{N_y}, y_{i+1}, y_L) \tag{15}$$

Hence, we finally obtain a sequence Y^h without OOVs, and HL-EncDec let the hybrid-level Y^h to replace the word-level Y . Therefore, HL-Decoder is able to generate any word, no matter this word has been recorded in Decoder's vocabulary or not.

5 Experiment

5.1 Model

The widely used word-level model and other famous non-word-level models are selected as baselines:

EncDec: The standard word-level Encoder-Decoder, which has been elaborated in the Section 2. EncDec has a bi-directional LSTM Encoder and a single-directional LSTM Attentional Decoder.

CharEncDec: A character-level EncDec, it has the same network architecture with EncDec, but it fully operates sentences at character-level. Namely, the CharEncDec directly reads or generates a sequence of characters.

BpeEncDec: A subword-level EncDec, and the BPE algorithm is selected to construct the subword units (Sennrich et al., 2016).

CNNEncDec: A fully character-level model proposed by Lee, which employs a character-level convolutional network with max-pooling at Encoder to operate sentence and a character-level Decoder to generate responses (Lee et al., 2016).

For all models, we set the embedding vector dimension as 512, RNN hidden unit size as 256, and batch size as 256. There is an exception in HL-EncDec's source side, the word embedding vector dimension d_w is set to 512 but the character embedding vector dimension d_c is set to 160, for efficiency. The training will be automatically stopped if the perplexity results of dev set are continuously increased in two periods, which is either the end of an epoch or every 10000 global steps. For each model, the beam search with $k = 10$ is applied to infer the responses. All models are implemented by Tensoflow.

5.2 Dataset & Vocabulary

We evaluate HL-EncDec and other models on a Chinese corpus released by Shang, which consists of 4.44 million message-response conversation pairs obtained from Sina Weibo (Shang et al., 2015). Each sentence already been tokenized to words by Stanford Chinese Word Segmenter. After filtering out noisy symbols and duplications, about 3.78 million pairs are finally used. We divide the filtered dataset into 3 sets, there are 20K/20K/3.74M pairs in the test/dev/training sets, respectively.

The vocabulary configurations should be noted here. For character-level models, CharEncDec and CNNEncDec, we keep all characters in vocabularies, and there are 6,690/10,432 characters in the source/target side. For the word-level model EncDec, we individually keep 40,000 most frequent words in the source and target side. For BpeEncDec, we allow BPE algorithm to mine at most (40,000 - 6,454/10,432) subword units in source/target side. For HL-EncDec, we keep 10,432 characters and 29,568 most frequent words in Decoder's vocabulary. In HL-EncDec's source side, we keep 6,690 characters in character vocabulary and keep 38328 words in word vocabulary.

5.3 Metrics

In this paper, we evaluate the performance of models with the following metrics:

BLEU & ROUGE: In previous work, BLEU & ROUGE are already been widely used to evaluate the overall performance of generation quality (Tian et al., 2017; Wu et al., 2017). BLEU is able to configure the max n-gram in the evaluating, hence we configure it from 2 to 4 and respectively denote them as: BLEU-2, BLEU-3 and BLEU-4.

Table 1: Word-level automatic metrics results. A number in bold means the best.

Model	ROUGE	BLEU-2	BLEU-3	BLEU-4	Distinct-1	Distinct-2
EncDec(Dev)	10.75	4.69	3.15	2.44	6.34%	23.48%
EncDec(Test)	10.69	4.60	3.02	2.30	6.28%	23.23%
BpeEncDec(Dev)	**11.45**	5.18	3.40	2.58	6.47%	24.98%
BpeEncDec(Test)	**11.63**	5.15	3.32	2.48	6.37%	24.60%
CNNEncDec(Dev)	9.13	4.00	2.49	1.79	3.85%	16.34%
CNNEncDec(Test)	9.22	3.93	2.39	1.69	3.93%	16.62%
HL-EncDec (Dev)	11.19	**5.28**	**3.55**	**2.74**	**7.08%**	**26.02%**
HL-EncDec (Test)	11.22	**5.26**	**3.53**	**2.74**	**7.03%**	**26.02%**

Table 2: Character-level automatic metrics results. A number in bold means the best.

Model	ROUGE	BLEU-2	BLEU-3	BLEU-4	Distinct-1	Distinct-2
EncDec(Dev)	11.13	6.26	4.42	3.42	1.18%	13.10%
EncDec(Test)	11.09	6.21	4.29	3.24	1.16%	12.87%
CharEncDec(Dev)	11.00	6.18	4.33	3.30	0.86%	12.82%
CharEncDec(Test)	10.83	5.91	4.08	3.07	0.85%	12.69%
BpeEncDec(Dev)	**12.43**	6.68	4.67	3.59	**1.43%**	16.76%
BpeEncDec(Test)	**12.60**	6.69	4.61	3.48	**1.42%**	16.56%
CNNEncDec(Dev)	10.28	5.85	3.83	2.78	0.86%	7.39%
CNNEncDec(Test)	9.22	3.93	2.39	1.69	0.86%	7.39%
HL-EncDec (Dev)	12.37	**6.80**	**4.80**	**3.73**	1.40%	**17.15**%
HL-EncDec (Test)	12.40	**6.83**	**4.77**	**3.68**	1.38%	**17.05**%

DISTINCT-1 & DISTINCT-2: Following (Li et al., 2016; Xing et al., 2016; Wu et al., 2017), we employ the DISTINCT-1/2 to measure how diverse and informative the generated responses are. DISTINCT-1/2 is the ratio of distinct uni-grams/bigrams in generated responses.

Human Evaluation: We employ three native speakers to individually annotate 50 randomly selected groups of responses. Each response may be rated as the following three criteria: +2/Excellent: the response is reasonable, fluent; +1/Acceptable: The response is a little 'safe' or irrelevant or has other small problems; 0/Bad: The response is irrelevant, or ungrammatically or too 'safe'.

5.4 Evaluation Results

Table 1 reports the evaluation results on automatic metrics. The proposed HL-EncDec outperforms baseline models on most metrics. In comparison with the standard EncDec model, HL-EncDec notably outperforms it in all metrics, which demonstrates HL-EncDec is able to generate a more excellent and informative response. Another CNN based model, CNNEncDec, has poor results, we believe the reason behind this is that CNNEncDec could not adapt our Chinese dataset. BPE is one of the state-of-the-art approaches to address the unknown words issue in neural machine translation. Here, BPE keeps this advantage cause BpeEncDec performs better than other baseline models. BpeEncDec only slightly outperforms HL-EncDec in the ROUGE, however, HL-EncDec outperforms BpeEncDec on the other metrics. In the metric BLEU-N, a higher N means it will evaluate the generated responses with more stringent standards. We find that HL-EncDec exceeds BpeEncDec more with the increasement of N, which shows the effectiveness of HL-EncDec from another aspect.

Unlike other languages such as English, etc., a Chinese native speaker always writes or reads a sentence without the explicit segmentation. Considering this fact, we evaluate all models at character-level, namely, we regard each sentence as a character sequence without explicit segmentation. The character-level evaluation results have been reported at Table 2. Clearly, the HL-EncDec still yield the best score on most metrics.

Table 3 reports human evaluation results. HL-EncDec has the highest total score, which means three

Table 3: Human Annotation

Model	EncDec	CharEncDec	BpeEncDec	CNNEncDec	HL-EncDec
0	48.67%	51.33%	32.67%	41.33%	34.00%
1	25.33%	27.33%	30.67%	26.00%	20.00%
2	26.00%	21.33%	36.67%	32.67%	46.00%
Total	38.67	35.00	52.00	45.67	56.00

annotators think our model has the best performance. Similar to the previous results of automatic metrics, although BpeEncDec is behinds HL-EncDec, it still outperforms than other models. EncDec has the lowest score, since EncDec may generate a sentence with unknown words. An unknown word often ruins the whole sentence. In addition, HL-EncDec also has the highest potential to generate a high-quality response, because 46% of responses generated by HL-EncDec are rated as +2/Excellent responses. CNNEncDec beats the EncDec and CharEncDec here, which is different from the automatic metrics results.

Table 4: Case Study

Case	1	2	3
Message	一天两包咖啡已然不够了。	子时已过一小时了，果断睡觉。	晚上吃素，自己动手，啦啦啦。
	It's not enough to drink two packs of coffee a day.	At two in the morning, go to bed	Eat vegetarian at night, DIY， La lala
EncDec	一天一杯，一瓶咖啡。	我还以为是**unk**呢	我也想吃这种蔬菜
	One cup of coffee a day, a bottle of coffee.	I thought it was **unk**	I want to eat this kind of vegetable, too
CharEncDec	一天两包咖啡已然不够了。	子时你过一小时了，果断睡觉。	好可爱啊，好可爱哦
	It's not enough to drink two packs of coffee a day.	At two in the morning, go to bed	It's cute, so cute.
BpeEncDec	一天一天，一天一天	我现在还在睡觉呢。	这就是传说中的胡萝卜?
	Day by day, day by day.	I'm still sleeping now.	This is the legendary carrot?
CNNEncDec	这是什么品种的啊?	时时时刻，果断睡觉	这是什么品种啊？求解
	What kind is this?	Time and time, go to bed	What kind is this?
HL-EncDec	我也想喝一包咖啡。	早睡早起身体健康!	我也想吃。
	I'd like to have a pack of coffee, too.	Early to bed and early to rise makes a man healthy, wealth and wise.	I want to eat, too

We give a qualitative case study in Table 4. By observing, it's easy to find HL-EncDec is able to generate more informative and high-quality responses. In the case 1, sentence generated by EncDec is not fluent, CharEncDec retells the message, BpeEncDec generates a strange response, these three models are unqualified here. In the case 2, EncDec suffers from unknown words, CharEncDec retells message again, CNNEncDec gives an incorrect response. BpeEncDec and HL-EncDec perform well. In the last case, while models except CharEncDec generate acceptable responses, HL-EncDec still gives a most suitable response.

Table 5: Word-level automatic metrics results on Twitter dataset.A number in bold means the best.

Model	ROUGE	BLEU-2	BLEU-3	BLEU-4	Distinct-1	Distinct-2
EncDec	5.84	1.39	0.52	0.23	0.89%	3.19%
BpeEncDec	**6.10**	1.41	0.52	0.23	**1.63%**	**6.06%**
HL-EncDec	5.99	**1.54**	**0.63**	**0.29**	1.34%	4.41%
HL-BPE	5.94	1.25	0.53	0.26	1.53%	5.56%

5.5 Discuss: Generalization

In order to evaluate the generalization ability of HL-EncDec, we take some additional small-scale experiments on an English dataset. The original dataset consists of about 377K message-response pairs from Twitter [1]. We only use 215K pairs with moderate lengths (i.e. 4-20 words). For this dataset, we set the embedding vector dimension as 320, the hidden vector size as 240, and batch size as 64. In HL-EncDec's source side, the character embedding vector dimension d_c is set to 160. The beam search $k = 5$ is applied to infer the responses.

We evaluate four models here, EncDec, BpeEncDec, HL-EncDec, and HL-BPE. HL-BPE is a variant of HL-EncDec, which decomposes an unknown word into a sequence of subword units instead of characters in the target side. An English word generally has more characters than a Chinese word, BPE may help HL-EncDec to reduce the sequence length in the target side. For EncDec, BpeEncDec and the target side of HL-EncDec and HL-BPE, their vocabularies are allowed to record 15,000 items. For the source side of HL-EncDec and HL-BPE, their character vocabularies consist of 617 characters, their word vocabularies consist of 14,383 words.

Table 5 shows the experimental results. EncDec and BpeEncDec have similar BLEU scores, but BpeEncDec has the best ROUGE and Distinct scores. Compared with BpeEncDec, HL-EncDec generates responses with higher qualities (i.e. higher ROUGE+BLEU scores), but less informative(i.e. lower Distinct scores). HL-BPE could be seen as a trade-off between HL-EncDec and BpeEncDec. In brief, these experiments show that HL-EncDec works well with other languages, and HL-EncDec is flexible enough to do some adaptive optimization for a specific language, which is very hard for BpeEncDec.

5.6 Discussion: Why hybrid-level?

Currently, while some previous works could alleviate the Preference Issue for word-level EncDec models in some degree (Weng et al., 2017; Wu et al., 2017), the Unknown Words Issue is still a problem with none effective solution. Employing a vocabulary with fixed words is infeasible since words not only numerous but also being created at any time. A word-level model must take extra actions to alleviate the impact of those unknown OOV words, which may involve more challenges. Character-level approaches could address them, but as what we introduced above and the experimental results, these approaches may decrease the performance.

In the hybrid-level model HL-EncDec, our goal is to keep the advantages of word-level models being reserved as much as possible after we addressed mentioned two issues since a word is a smallest and basic unit in linguistics. Hence, we ingeniously absorb the advantages of characters and propose a hybrid-level Encoder and a hybrid-level decoder.

6 Conclusion and Future work

In this paper, we presented a hybrid-level Encoder-Decoder model, HL-EncDec, for response generation task. This hybrid-level approach absorbs the advantages of both word-level models and character-level models and addressed the Unknown Words Issue and Preference Issue. Experimental results show HL-EncDec could deliver more informative, more relevant responses without the appearance of unknown words. In the future, we will mainly investigate how to improve the techniques in the target side, meanwhile, we will also seek for techniques to enhance our hybrid-level representation technique.

[1] this dataset is released by a third-party github user: https://github.com/marsan-ma/chat_corpus (twitter_en.txt.gz)

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation By Jointly Learning To Align and Translate. In *International Conference on Learning Representations*, pages 1–15, sep.

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A Survey on Dialogue Systems: Recent Advances and New Frontiers. *ACM SIGKDD Explorations Newsletter*, 19.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *empirical methods in natural language processing*, pages 1724–1734.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. dec.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A Character-Level Decoder without Explicit Segmentation for Neural Machine Translation. mar.

Marta R Costajussa and Jose A R Fonollosa. 2016. Character-based Neural Machine Translation. *meeting of the association for computational linguistics*, pages 357–361.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv: Neural and Evolutionary Computing*.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On Using Very Large Target Vocabulary for Neural Machine Translation. *Annual Meeting of the Association for Computational Linguistics*, 000:1–10.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. *national conference on artificial intelligence*, pages 2741–2749.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2016. Fully Character-Level Neural Machine Translation without Explicit Segmentation. In *Annual Meeting of the Association for Computational Linguistics*, pages 1693–1703, mar.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A Diversity-Promoting Objective Function for Neural Conversation Models. *north american chapter of the association for computational linguistics*, pages 110–119.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. *empirical methods in natural language processing*, pages 1412–1421.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. *meeting of the association for computational linguistics*, pages 1715–1725.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural Responding Machine for Short-Text Conversation. In *Annual Meeting of the Association for Computational Linguistics*, pages 1577–1586.

Louis Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating High-Quality and Informative Conversation Responses with Sequence-to-Sequence Models. *eprint arXiv:1701.03185*, jan.

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

Zhiliang Tian, Rui Yan, Lili Mou, Yiping Song, Yansong Feng, and Dongyan Zhao. 2017. How to Make Context More Useful? An Empirical Study on Context-Aware Neural Conversational Models. pages 231–236.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *Computer Science*.

Rongxiang Weng, Shujian Huang, Zaixiang Zheng, Xinyu Dai, and Jiajun Chen. 2017. Neural Machine Translation with Word Predictions. aug.

Yu Wu, Wei Wu, Dejian Yang, Can Xu, Zhoujun Li, and Ming Zhou. 2017. Neural Response Generation with Dynamic Vocabularies.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Weiying Ma. 2016. Topic Aware Neural Response Generation. *national conference on artificial intelligence*, pages 3351–3357.

Multi-Perspective Context Aggregation for Semi-supervised Cloze-style Reading Comprehension

Liang Wang[1], **Sujian Li**[2], **Wei Zhao**[1],
Kewei Shen[1], **Meng Sun**[1], **Ruoyu Jia**[1], **Jingming Liu**[1]
[1]Yuanfudao Research, Beijing, China
[2]Key Laboratory of Computational Linguistics, Peking University, MOE, China
{wangliang01,zhaowei01,shenkw,sunmeng,jiary,liujm}@fenbi.com
lisujian@pku.edu.cn

Abstract

Cloze-style reading comprehension has been a popular task for measuring the progress of natural language understanding in recent years. In this paper, we design a novel multi-perspective framework, which can be seen as the joint training of heterogeneous experts and aggregate context information from different perspectives. Each perspective is modeled by a simple aggregation module. The outputs of multiple aggregation modules are fed into a one-timestep pointer network to get the final answer. At the same time, to tackle the problem of insufficient labeled data, we propose an efficient sampling mechanism to automatically generate more training examples by matching the distribution of candidates between labeled and unlabeled data. We conduct our experiments on a recently released cloze-test dataset CLOTH (Xie et al., 2017), which consists of nearly $100k$ questions designed by professional teachers. Results show that our method achieves new state-of-the-art performance over previous strong baselines.

1 Introduction

Reading comprehension is a challenging task which requires the deep understanding of natural language. Cloze test is a particular form of reading comprehension: given a passage with blanks, an examinee is required to fill in the missing word (or phrase) that best fits the context surrounding the blank. Recently, cloze-style reading comprehension has drawn growing interests from NLP research communities, since such a task meets the practical requirements and is relatively easy to design.

The research of cloze-style reading comprehension is first advanced by two large-scale corpora: the CNN/Daily Mail (Hermann et al., 2015) and CBT (Hill et al., 2015) datasets, which are automatically constructed by randomly or periodically deleting a word from original passage. Though the automatically generated datasets usually consist of a large quantity of labeled data and make it possible to train large neural network models, they are in nature far away from real-world language understanding problems and have serious ambiguity issues (Chen et al., 2016). As a result, the state-of-the-art system of cloze test almost reaches the performance ceiling and loses its improvement direction due to the limitation of the corpus (Chen et al., 2016). In such situation, Xie et al. (2017) argues that it is a more reliable means to assess language proficiency with carefully designed questions by professional teachers, and releases a novel corpus CLOTH. The CLOTH dataset brings the new challenge of exploring a comprehensive evaluation of language proficiency and specifically divides the questions into several types including matching, reasoning and grammar etc. Table 1 shows several example questions from CLOTH.

From experiments by Xie et al. (2017), we can see that the *Stanford attention reader* (Chen et al., 2016) of having the near state-of-the-art performance (with an accuracy of about 0.74) on CNN/Daily Mail only gets an accuracy of 0.487 on CLOTH and there exists a huge performance gap between human and popular machine learning models. The main reason is that attention models are mainly good at processing matching questions (e.g., the first example in Table 1 has matching between *"police"* and *"accident"*, *"man died"*), which occupy a less percentage in CLOTH than in CNN/Daily Mail. Xie

Proceedings of the 27th International Conference on Computational Linguistics, pages 857–867
Santa Fe, New Mexico, USA, August 20-26, 2018.

question	type
..... As a Senior student, I have to __ many exams. ..... *A: finish* *B: win* ***C: take*** *D: join*	collocation
I am calling from the __ station ..... " There was an accident, and a man died ." ..... *A: post* *B: bus* ***C: police*** *D: railway*	matching
a student reported that I made an error ..... He was __ and after thanking him for his honesty ..... he said angrily . *A: wise* ***B: right*** *C: rigid* *D: angry*	reasoning
..... They are used to __ messages by computers and smart phones. ..... ***A: sending*** *B: send* *C: sent* *D: sends*	grammar

Table 1: Example questions and their corresponding types from the CLOTH dataset. "....." represents some omitted irrelevant sentences.

et al. (2017) also present the word-predicting potential of language models (LM) which can well tackle lexical collocation (e.g., *"take"* and *"exam"* in the second example), given a large volume of unlabeled data and high computation power. Furthermore, Xie et al. (2017) points out that the most difficult questions belong to the long-term-reasoning type (e.g., the third example question), which constitutes approximately 22.4% in CLOTH and needs more semantics to deal with.

To comprehensively consider the progress and questions in CLOTH, we come up with the idea of modeling multiple perspectives to arrive at the correct answer, given limited computation power. Our multi-perspective network consists of several parallel modules, where each module aggregates context information from a unique perspective. We model long-distance matching with attentive readers, global semantics with iterative dilated convolutions and lexical collocation with both n-gram and neural language model(LM). The outputs of aggregation modules are further integrated and fed into a one-timestep pointer network (Vinyals et al., 2015) to get the final answer.

Next, one challenging problem is how to effectively train our multi-perspective network due to the insufficiency of labeled data. To overcome this problem, Xie et al. (2017) present a representativeness-based weighted loss function. Their approach has two drawbacks: first, it requires to train another model for predicting a candidate's representativeness score; second, it is not a sample-efficient way since each word including uninformative stop words becomes a training example. In this paper, we improve on Xie et al. (2017)'s approach and develop a semi-supervised learning method by matching the distribution of candidates between labeled and unlabeled data. The intuition is to make automatically constructed data as similar as possible to existing labeled data. Stop words, named entities and out-of-vocabulary words should be downsampled while meaningful content words should be kept for training.

Our method is simple, straightforward and shows better performance with only a fraction of training examples. Experiments show that our semi-supervised multi-perspective network is able to outperform state-of-the-art results on the CLOTH dataset by 4.2%.

2 Model

Formally, the task of cloze-style reading comprehension requires choosing the correct answer from $|c|$ candidates $\{c_i\}_{i=1}^{|c|}$ given a sequence of words $\{w_i\}_{i=1}^{n}$ as context. Candidate c_i could be a word or a phrase. For the CLOTH dataset, each question has $|c| = 4$ candidates.

2.1 MPNet: Multi-Perspective Context Aggregation Network

The overall architecture of our proposed MPNet is shown in Figure 1. It consists of an input layer, a multi-perspective aggregation layer and an output layer.

Input Layer Given the passage as a variable-length word sequence $\{w_i\}_{i=1}^{n}$, we embed each word into 300-dimensional word embeddings $\{\mathbf{e}_i\}_{i=1}^{n}$ using GloVe vectors. Then, we apply bidirectional GRU(BiGRU) on $\{\mathbf{e}_i\}_{i=1}^{n}$ to get contextualized word representations $\{\mathbf{h}_i\}_{i=1}^{n}$ (McCann et al., 2017)

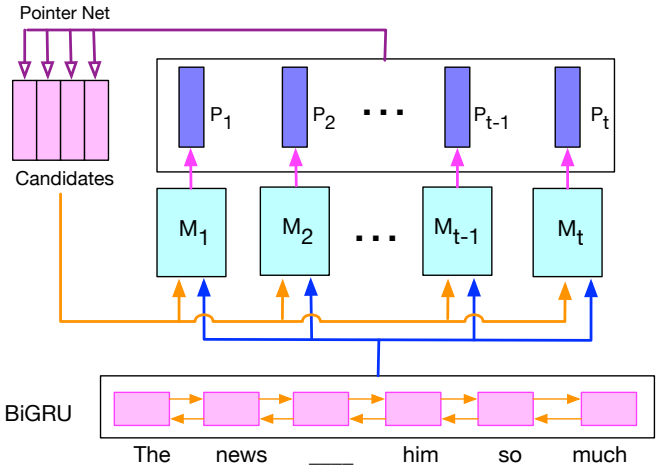


Figure 1: MPNet: **M**ulti-**P**erspective context aggregation network. We only show part of the context *"The news ___ him so much"* and "___" is the blank to fill in.

(Peters et al., 2018), since GRU is computationally more efficient and shows slightly better performance than LSTM.

$$\begin{aligned}\overrightarrow{\mathbf{h}}_i &= \overrightarrow{GRU}(\overrightarrow{\mathbf{h}}_{i-1}, \mathbf{e}_i) \\ \overleftarrow{\mathbf{h}}_i &= \overleftarrow{GRU}(\overleftarrow{\mathbf{h}}_{i+1}, \mathbf{e}_i) \\ \mathbf{h}_i &= [\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i]\end{aligned} \tag{1}$$

We also use another GRU to encode candidates $\{c_i\}_{i=1}^{|c|}$ into fixed-length vectors $\{\mathbf{u}_i\}_{i=1}^{|c|}$, as candidates may be multi-word phrases.

Multi-Perspective Aggregation Layer This layer consists of several independent aggregation modules $\{M_i\}_{i=1}^{t}$. Computation can be easily parallelized since modules are independent. Each module M_i takes contextualized word representations $\{\mathbf{h}_i\}_{i=1}^{n}$ and candidates' encoding $\{\mathbf{u}_i\}_{i=1}^{|c|}$ as input and outputs a vector $\mathbf{p}_i$, which reflects the information from module M_i's perspective. We also assume aggregation modules can have access to $\{w_i\}_{i=1}^{n}$ and $\{c_i\}_{i=1}^{|c|}$. For cloze-style reading comprehension, each module should be able to distill some knowledge which can judge whether a candidate fits a given context from a perspective.

The aggregation modules that we use are listed below:

- **Selective Copying** Assuming the index of the blank is j, this module simply selects the hidden representation of the blank $\mathbf{h}_j$, directly copies it to the output $\mathbf{p}_{sc}$ and ignores everything else. Note that $\mathbf{h}_j$ is the output of BiGRU and already incorporates context information from both forward and backward directions. This resembles a bidirectional language model without softmax output layer. Words near the blank are paid more attention which is consistent with our intuition of filling in the blank.

- **Attentive Reader** A large portion of questions involve matching candidates with related words which may be far away from each other such as the second example in Table 1. *Attentive reader* proposed by Chen et al. (2016) directly attends to the entire context and therefore avoids the difficulty of modeling long-range dependence. Original bilinear attention function (Chen et al., 2016) is slightly modified by introducing $\mathbf{b}_{ar}$ to model attention bias towards the ith word. $\mathbf{u}$ is the vector representation of a candidate, we omit its subscript for simplicity.

$$\begin{aligned}\alpha_i &= \text{softmax}_i(\mathbf{u}^T\mathbf{W}_{ar}\mathbf{h}_i + \mathbf{b}_{ar}^T\mathbf{h}_i), i \in [1, n] \\ \mathbf{p}_{ar} &= \sum_{i=1}^{n} \alpha_i \mathbf{h}_i\end{aligned} \tag{2}$$

- **Iterative Dilated Convolution** Convolutional neural networks have been a successful method for modeling both natural language (Kim, 2014) and images. Multiple layers of CNNs can extract features in a hierarchical way, which shares similarity with the compositional property of natural language. Dilated convolution is a variant of traditional convolution and is more efficient for multi-scale context aggregation (Yu and Koltun, 2015; Strubell et al., 2017). In this work, we use two blocks where each block consists of two dilated convolutions with dilation rate set to 1 and 3 respectively. Max pooling across filters is applied to get the final output $\mathbf{p}_{idc}$.

- **N-gram Statistics** To explicitly incorporate collocation information, we use this module to output logarithmic n-gram counts $\mathbf{p}_{ng}$ from *English Wikipedia* with $n \in [1, 5]$. Logarithmic function avoids the optimization difficulty with extremely large numbers.

Note that the output $\mathbf{p}_{sc}$ from selective copying module and $\mathbf{p}_{idc}$ from iterative dilated convolution module don't depend on the candidates. We therefore get context representation $\mathbf{P}_{ctx} = [\mathbf{p}_{sc}; \mathbf{p}_{idc}]$ by concatenating $\mathbf{p}_{sc}$ and $\mathbf{p}_{idc}$. Similarly, we can get the representation vector $\mathbf{C}_i$ for ith candidate by concatenating the output $\mathbf{p}^i_{ar}$ from attentive reader module, $\mathbf{p}^i_{ng}$ from n-gram statistics and $\mathbf{u}_i$ from the candidate encoder: $\mathbf{C}_i = [\mathbf{u}_i; \mathbf{p}^i_{ar}; \mathbf{p}^i_{ng}], i \in [1, |c|]$.

Output Layer We use a one-timestep pointer network (Vinyals et al., 2015) to choose the correct answer from $|c|$ candidates $\{\mathbf{c}_i\}_{i=1}^{|c|}$. Given context representation $\mathbf{P}_{ctx}$ and candidates representation $\{\mathbf{C}_i\}_{i=1}^{|c|}$, we first refine candidates representation with a gating mechanism:

$$\begin{aligned} \mathbf{g}_i &= \sigma(\mathbf{W}_1\mathbf{P}_{ctx} + \mathbf{W}_2\mathbf{C}_i + \mathbf{b}),\ i \in [1, |c|] \\ \mathbf{C'}_i &= \mathbf{C}_i \odot \mathbf{g}_i,\ i \in [1, |c|] \end{aligned} \tag{3}$$

σ is the sigmoid function and $\odot$ denotes pointwise multiplication. Then we calculate the distribution of being the correct answer over candidates with bilinear function:

$$\hat{y}_i = \mathrm{softmax}_i(\mathbf{C'}_i^T \mathbf{W}_o \mathbf{P}_{ctx} + \mathbf{b}_o^T \mathbf{C'}_i),\ i \in [1, |c|] \tag{4}$$

$\{\hat{y}_i\}_{i=1}^{|c|}$ is a probability distribution and the pointer points to the candidate $\arg\max_i(\hat{y}_i)$.

Model Learning The model is trained by minimizing the standard cross-entropy loss.

Discussion Different aggregation modules summarize context from different perspectives. In order to precisely locate the correct answer, a set of complementary aggregation modules are preferred where one module may only focus on lexical collocation and another module may be sensitive to the global matching. It is worth noting that our MPNet framework can be easily extended by adding other effective aggregation modules.

In addition, the main idea of MPNet is to some extent connected with the mixture of experts (MoE) (Masoudnia and Ebrahimpour, 2014). If each aggregation module can be seen as an expert, then multiple aggregation modules become MoE. One key difference is that aggregation modules in MPNet have heterogeneous network structures while traditional MoE models usually consist of homogeneous experts.

2.2 SemiMPNet: Semi-supervised Learning with Distribution Matching

SemiMPNet is the semi-supervised variant of our proposed MPNet in Section 2.1, with exactly the same network architecture. Though CLOTH consists of nearly $100k$ questions, it is generally not enough to train large neural models. Semi-supervised learning comes to the rescue. We propose to sample from unlabeled text to construct training examples automatically. In order to train effectively, we need to make the automatically generated data similar to labeled data and ensure that candidates should have a similar distribution in original labeled data to that in the generated data. Then, we formulate candidates distribution matching in two datasets as two sampling problems as follows:

How to sample positive candidates? We assume D_u is a collection of unlabeled documents, D_c is the collection of all candidates in the CLOTH dataset and V is the vocabulary which is composed of all the candidates occurring in CLOTH. Each word $w_i \in V$ is associated with an unknown sampling probability $p(w_i)$. To match the distribution of candidates between D_u and D_c, the following constraints about $p(w_i), i \in [1, |V|]$ should be satisfied:

$$\begin{gathered} \frac{p(w_i)\#(w_i, D_u)}{\sum_{j=1}^{|V|} p(w_j)\#(w_j, D_u)} = \frac{\#(w_i, D_c)}{\sum_{j=1}^{|V|} \#(w_j, D_c)},\ i \in [1, |V|] \\ 0 \leq p(w_i) \leq 1, i \in [1, |V|] \\ \max\{p(w_i)|\ i \in [1, |V|]\} = 1 \end{gathered} \tag{5}$$

Function $\#(w_i, D)$ returns the frequency of w_i in corpus D. The second constraint is to make sure $\{p(w_i)\}_{i=1}^{|V|}$ is a valid probability distribution and the third constraint is to make full use of data. There is generally no exact solution to Equation(5) as $\#(w_i, D_u) = 0$ and $\#(w_i, D_c) > 0$ may hold for some i. Instead, we use an approximate solution:

$$p(w_i) = \min(1, \frac{\#(w_i, D_c)}{\#(w_i, D_u)} \times \frac{\gamma \sum_{j=1}^{|V|} \#(w_j, D_u)}{\sum_{j=1}^{|V|} \#(w_j, D_c)}) \tag{6}$$

The coefficient γ can be interpreted as the average probability of sampling a word. We set $\gamma = 0.5$ based on validation data. With this strategy, we sample the positive candidates and use the corresponding passages as their contexts.

How to sample negative candidates? Given a positive candidate w_p, the probability of w_i being sampled as a negative candidate $p(w_i|w_p)$ can be calculated as follows:

$$p(w_i|w_p) = \frac{\lambda}{|V|} + (1 - \lambda)\frac{\#(w_i, w_p)}{\sum_{j=1}^{|V|} \#(w_j, w_p)} \tag{7}$$

$\#(w_i, w_p)$ is the co-occurrence counts of w_i and w_p as candidates in labeled dataset D_c. Intuitively, the co-occurrence probability of w_i and w_p should match between constructed data and labeled data. λ is the probability of randomly selecting a word from the entire vocabulary, similar to the exploration mechanism in reinforcement learning. It makes our model more robust to overfitting and we set $\lambda = 0.1$ throughout the experiments.

In the case that candidates are multi-word phrases, our method is also applicable by simply expanding the vocabulary V to include phrases in D_c.

3 Experiments

3.1 Experimental Setup

Dataset and Evaluation Metrics We use the CLOTH (Xie et al., 2017) dataset for training and evaluation. RACE (Lai et al., 2017) dataset and *English Wikipedia* [1] serve as background text corpora for semi-supervised learning. RACE dataset consists of nearly $28k$ reading comprehension passages from high-school examinations. We delete passages that have a Jaccard similarity over 0.85 with passages in the CLOTH dataset. Furthermore, background text corpora also include training passages from the CLOTH dataset by filling the correct answer back into the corresponding blank.

Accuracy is used as the evaluation metric. To make a fair comparison with Xie et al. (2017), we also report performance on CLOTH-M(middle school questions) and CLOTH-H(high school questions).

Hyperparameters Our model is implemented with Tensorflow (Abadi et al., 2016). Hyperparameters are optimized with random search based on validation data. All our models are run on a single GPU(Tesla P40). NLTK (Bird and Loper, 2004) is used for tokenization. Word embeddings are initialized with

[1] https://dumps.wikimedia.org/enwiki/

300-dimensional GloVe (Pennington et al., 2014) vectors. Only vectors of top 1000 frequent words are fine-tuned during training. Our network is trained with Adam algorithm (Kingma and Ba, 2014). The initial learning rate is set to 10^{-3}. We decrease learning rate to 10^{-4} after $15k$ iterations and further decrease it to 10^{-5} after $50k$ iterations. Both forward and backward GRU have 128 hidden units. For input, we use a context window of 80 words. For 1D dilated convolution, we use 2 blocks, the number of filters is 128 and the convolution width is 3 for all layers. Batch normalization and ReLU are applied on top of convolution. Gradients are clipped to have a maximum L2 norm of 5. Dropout with probability 0.5 is applied to the output of BiGRU.

Model	+ constructed data?	CLOTH	CLOTH-M	CLOTH-H
Random	No	25.0%	25.0%	25.0%
LSTM (Xie et al., 2017)	No	48.4%	51.8%	47.1%
Stanford Attention Reader (Chen et al., 2016)	No	48.7%	52.9%	47.1%
MPNet - ngram	No	**50.1%**	**53.2%**	**49.0%**
Language Model (Xie et al., 2017)	Yes	54.8%	64.6%	50.6%
Representativeness (Xie et al., 2017)	Yes	56.5%	66.5%	52.6%
LSTM + Representativeness (Xie et al., 2017)	Yes	58.3%	67.3%	54.9%
SemiMPNet - ngram	Yes	**60.9%**	**67.6%**	**58.3%**
Human	–	86.0%	89.7%	84.5%

Table 2: Experimental results without using external data. We exclude *n-gram* as *n-gram* is calculated based on external corpus *Wikipedia*. SemiMPNet uses passages from CLOTH for semi-supervised data augmentation. Human performance is from Xie et al. (2017).

3.2 Baselines

LSTM is a baseline model by Xie et al. (2017). First, a BiLSTM layer is applied to context word embeddings. Then it uses the outputs near the blank to calculate the probability of being the correct answer for each candidate.

Stanford Attention Reader is an attention-based neural model for reading comprehension presented by Chen et al. (2016). Experimental results are from Xie et al. (2017).

Language Model To overcome the difficulty of insufficient labeled data. Xie et al. (2017) propose to train a neural language model on passages from the CLOTH dataset. The candidate that results in the highest probability is chosen as the predicted answer. It's fair to say *Language Model* is a simple data augmentation approach that treats every word as a training example with equal weight.

Representativeness is another semi-supervised data augmentation approach by Xie et al. (2017). It assigns different weights to different constructed examples based on *Representativeness score*. *Representativeness* can be interpreted as the probability of a given word being selected as a blank by human. For more technical details, please refer to Xie et al. (2017).

One-billion-word-LM is a state-of-the-art neural language model (Jozefowicz et al., 2016) trained on one-billion-word benchmark (Chelba et al., 2013). It has more than 1 billion parameters and is publicly available[2].

3.3 Main Results

We evaluate our model's performance in two experimental settings: use external data or not. For the setting without external data, we only use passages from CLOTH for training and semi-supervised data augmentation. Though GloVe vectors are trained on external text corpora, it has become a standard practice for NLP to use pretrained embeddings. Therefore, GloVe vectors are used in both settings and so does the work by Xie et al. (2017).

Results w/o External Data Results are shown in Table 2. When trained only on labeled data, both LSTM by Xie et al. (2017) and our proposed MPNet perform poorly, though MPNet slightly outper-

[2] https://github.com/tensorflow/models/tree/master/research/lm_1b

forms LSTM by 1.7% in overall accuracy. The accuracy of middle school questions (CLOTH-M) is consistently higher than high school questions (CLOTH-H) across all of our experiments, since middle school questions are relatively easier.

w	$p(w)$	w	$p(w)$	w	$p(w)$
I	0.04	festivals	0.75	California	0.13
the	0.03	birthday	1.0	thank you	0.26
Frank	0.09	8	0.09	congratulation	1.0

Table 3: Sample probability for some words. $p(w)$ is the probability of sampling w to construct a training example. Stop words, named entities usually have low probability. See Section 2.2 for details.

Table 2 clearly shows that constructed data can significantly boost both models' performance. Xie et al. (2017) explore several different ways for data augmentation: *Language Model* treats every word equally, while *Representativeness* method assigns different weights to different words by training an representativeness prediction network. This mechanism improves the accuracy from 48.4% to 58.3%. Further, our proposed method adopts a new sampling method and requires sampling words with distribution constraints, which makes training more efficient. As shown in Table 3, stop words (e.g., *"I"* and *"the"*), named entities (e.g., *"Frank"* and *"California"*) and common phrases (e.g., *"thank you"*) have low probability of being sampled. Content words such as *"festivals"* and *"birthday"* are more likely to be sampled. One limitation of our sampling method is its inability to handle synonyms. Since synonyms tend to co-occur as candidates in the labeled dataset, this problem is not as severe as it looks like to be. *"SemiMPNet - ngram"* beats all baseline methods and achieves the highest accuracy 60.9%. Human performance is 86.0% which is much higher than *"SemiMPNet - ngram"*. The effectiveness of constructed data indicates that the lack of labeled data has become a bottleneck.

Model	CLOTH	CLOTH-M	CLOTH-H
One-billion-word-LM	70.7%	74.5%	69.3%
MPNet	65.3%	70.0%	63.6%
SemiMPNet	70.4%	75.5%	68.5%
SemiMPNet + One-billion-word-LM	**74.9%**	**79.0%**	**73.3%**
Human	86.0%	89.7%	84.5%

Table 4: Experimental results with external data. SemiMPNet use passages from the RACE dataset for semi-supervised data augmentation.

Results with External Data As shown in Table 4, incorporation of the RACE dataset for semi-supervised learning improves accuracy from 65.3% to 70.4%. However, MPNet and SemiMPNet still underperform a pretrained state-of-the-art neural language model *One-billion-word-LM* (Jozefowicz et al., 2016). It is trained on a large corpus with nearly 1 billion words and achieves an accuracy of 70.7%. In contrast, SemiMPNet is trained on only $\sim$ 10 million words and has a 0.3% gap in accuracy, which is pretty impressive given that the sizes of two corpora differ by two orders of magnitude. Once again it shows the power of transferring knowledge from unlabeled text corpora.

As a further discussion, we'd like to point out that although language model can achieve good results, it is not the most efficient way. Actually, experimental results in Table 2 show that language model underperforms SemiMPNet given the same amount of text. A fair comparison would be training SemiMPNet on the one-billion-word benchmark. Considering the size of the one-billion-word corpus, applying our semi-supervised method directly on the one-billion-word corpus would require a sizable amount of computing power. Here we design an approximate method *"SemiMPNet + One-billion-word-LM"* and combine MPNet and *One-billion-word-LM* by linear interpolation of their output probabilities:

$$p = \beta p_{mp} + (1 - \beta) p_{lm} \tag{8}$$

p_{mp} is the output probability by SmiMPNet and p_{lm} is the normalized probability by *One-billion-word-*

LM[3]. Setting $\beta = 0.5$ yields empirically good results. Hyper-parameter search shows the results are quite robust to a wide range of β values. We can see that our model *"SemiMPNet + One-billion-word-LM"* achieves a new state-of-the-art performance of 74.9%, which improves *One-billion-word-LM* by 4.2%. This also shows the complementarity of SemiMPNet and *One-billion-word-LM*. Two models can learn different aspects of the contexts.

3.4 Ablation Study

Our proposed MPNet consists of four aggregation modules. To examine the effect of each module, we conduct an ablation study. The results are shown in Table 5.

Model	CLOTH
SemiMPNet	**70.4%**
w/o selective copying	69.4% (-1.0)
w/o attentive reader	67.6% (-2.8)
w/o dilated convolution	69.6% (-0.8)
w/o n-gram statistics	63.0% (-7.4)

Table 5: Results for SemiMPNet ablation study. RACE dataset is used for semi-supervised data augmentation.

N-gram statistics turn out to be the single most influential factor. Overall performance decreases by 7.4% without *n-gram statistics*. On one hand, this result further highlights the importance of distilling knowledge from large text corpora. On the other hand, it proves that our background corpus is not large enough for neural models to learn reliable lexical collocation information.

Attentive reader also has a significant impact on overall performance. Attention mechanism is able to locate useful information regardless of its positional distance from the blank. In contrast, RNNs need to preserve such information over a long distance which is nontrivial.

Besides, the results in Table 5 support an important intuition in this paper: different modules capture context information from different perspectives, and removing any one of them would result in decreased performance.

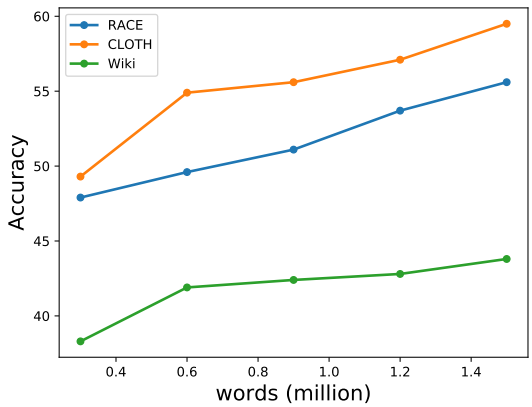


Figure 2: Examining effects of different text corpora. The x-axis is the number of words in background corpus, and the y-axis is the accuracy on test data. To avoid the influence of external data, we report model performance with *"SemiMPNet - ngram"*.

3.5 Examining Effects of Background Corpus

For our semi-supervised learning model SemiMPNet, background corpus is used to construct training examples. The choice of background corpus can make a big difference. In this section, we conduct an

[3]The normalization makes sure the probabilities for all candidates sum to 1.

experiment to examine such effects. Three different corpora are used: passages from the training set of CLOTH, RACE and English Wikipedia.

Results are shown in Figure 2. Unsurprisingly, more data lead to better performance. Moreover, given the same amount of text, CLOTH consistently beats RACE and RACE consistently beats Wikipedia. As we know, CLOTH and RACE consists of passages designed for high school students, while Wikipedia entries are for the general public and therefore have a different word distribution. Thus, how to make use of huge unlabeled data to help training is a key for performance improvement, since training corpora of higher quality are generally smaller in scale.

4 Related Work

Reading Comprension or machine reading is drawing more and more interests among NLP research communities. The CNN/Daily Mail (Hermann et al., 2015) and CBT (Hill et al., 2015) are two automatically generated cloze-style datasets. Though they can be large in scale, the quality of automatically generated questions is generally lower than manually labeled ones. Instead, SQuAD (Rajpurkar et al., 2016) adopts a crowd-sourcing approach to ensure its quality. In SQuAD, each passage accompanies one or more questions and the answer is a text span of the given passage for the convenience of automatic evaluation. Rapid progress has been made with neural network based models (Wang et al., 2016). The performances of state-of-the-art models on SQuAD such as QANet (Yu et al., 2018) and ELMo (Peters et al., 2018) are already very close to human. There are also some datasets focusing on answering questions from real-world scenarios. MS MARCO (Nguyen et al., 2016) and DuReader (He et al., 2017) are two typical examples. Such datasets are usually harder as they require the ability of both comprehension and language generation. BLEU and ROUGE are often used as evaluation metrics. One potential problem is that answers with high BLEU scores may have very different semantic meanings.

Cloze Test is a particular form of reading comprehension task and has been widely adopted as a method for assessing students' language proficiency. Zweig and Burges (2011) presented a challenging dataset for sentence completion but its scale is too small with only 1040 questions. CNN/Daily Mail (Hermann et al., 2015), CBT (Hill et al., 2015), LAMBADA (Paperno et al., 2016) and CLOTH (Xie et al., 2017) are all large-scale cloze-test datasets, with the difference that each question in CLOTH has four candidate options. Recently proposed Story Cloze (Mostafazadeh et al., 2017) is a cloze-test dataset that goes beyond words and phrases, and requires choosing a sentence as the appropriate story ending.

Semi-supervised Methods for reading comprehension are widely studied due to the fact that labeled data is scarce. One major approach is pretraining a model for text representation and reusing the weights during supervised learning. Autoencoders (Hewlett et al., 2017), machine translation (McCann et al., 2017) and language model (Peters et al., 2018) can be used for representation learning. Another approach aims to directly construct training examples from unlabeled text corpora. Weighted loss function (Xie et al., 2017) and reinforcement learning (Yang et al., 2017) can be used to alleviate the discrepancy between human-labeled data and automatically-constructed data.

5 Conclusion

In this paper, we propose a multi-perspective network MPNet for cloze-style reading comprehension. MPNet consists of several parallel context aggregation modules. Each module summarizes the variable-length context and candidates into a fixed-length vector from a unique perspective. We explore four effective implementations of aggregation modules in experiments. The architecture of MPNet is very flexible and can be easily extended by adding more task-specific modules.

To overcome the difficulty of limited labeled data, we turn to semi-supervised learning by automatically constructing training examples from unlabeled text corpora. Experiments on the CLOTH dataset show that our semi-supervised MPNet achieves new state-of-the-art performance. In our future work, we'd like to come up with more effective methods to tackle this challenge.

Acknowledgements

We would like to thank three anonymous reviewers for their insightful comments, and COLING 2018 organizers for their efforts.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Steven Bird and Edward Loper. 2004. NLTK: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2358–2367.

Wei He, Kai Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, Xuan Liu, et al. 2017. DuReader: a Chinese Machine Reading Comprehension Dataset from Real-world Applications. *arXiv preprint arXiv:1711.05073*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Daniel Hewlett, Llion Jones, Alexandre Lacoste, et al. 2017. Accurate supervised and semi-supervised machine reading for long documents. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2001–2010.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Saeed Masoudnia and Reza Ebrahimpour. 2014. Mixture of experts: a literature survey. *Artificial Intelligence Review*, pages 1–19.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. *arXiv preprint arXiv:1708.00107*.

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James F Allen. 2017. LSDSem 2017 Shared Task: The Story Cloze Test. *LSDSem 2017*, page 46.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *NAACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate entity recognition with iterated dilated convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2660–2670.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2016. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211*.

Qizhe Xie, Guokun Lai, Zihang Dai, and Eduard Hovy. 2017. Large-scale cloze test dataset designed by teachers. *arXiv preprint arXiv:1711.03225*.

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William W Cohen. 2017. Semi-Supervised QA with Generative Domain-Adaptive Nets. *arXiv preprint arXiv:1702.02206*.

Fisher Yu and Vladlen Koltun. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *ICLR*.

Geoffrey Zweig and Christopher JC Burges. 2011. The Microsoft Research sentence completion challenge. Technical report, Technical Report MSR-TR-2011-129, Microsoft.

A Lexicon-Based Supervised Attention Model for Neural Sentiment Analysis

Yicheng Zou, Tao Gui, Qi Zhang, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
rowitzou@gmail.com, {tgui16,qz,xjhuang}@fudan.edu.cn

Abstract

Attention mechanisms have been leveraged for sentiment classification tasks because not all words have the same importance. However, most existing attention models did not take full advantage of sentiment lexicons, which provide rich sentiment information and play a critical role in sentiment analysis. To achieve the above target, in this work, we propose a novel lexicon-based supervised attention model (LBSA), which allows a recurrent neural network to focus on the sentiment content, thus generating sentiment-informative representations. Compared with general attention models, our model has better interpretability and less noise. Experimental results on three large-scale sentiment classification datasets showed that the proposed method outperforms previous methods.

1 Introduction

Sentiment analysis has the goal of analyzing people's sentiments or opinions and has been well explored (Turney, 2002; Tang et al., 2014b; Chen et al., 2016). In order to improve sentiment analysis results, large sentiment lexicons have been built (Wilson et al., 2005; Baccianella et al., 2010; Tang et al., 2014a). A sentiment lexicon is a set of words such as *excellent*, *terrible* and *ordinary*, each of which is assigned a fixed positive or negative score that presents its sentiment polarity and strength (Tang et al., 2014a). Such information can serve as sentiment-informative features and significantly boost the classification performance (Agarwal et al., 2013; Teng et al., 2016; Qian et al., 2016).

In sentiment classification tasks, sentiment words such as *great* and *terrible* tend to play a more critical role than other words in texts (Liu, 2010). One attribute of words annotated in sentiment lexicons is their sentiment strength, which is intuitively associated with a word's contribution to the sentiment representation of a sentence. It is similar to the basic idea of the attention mechanism that not all words have the same importance. However, very few studies have focused on a method that combines an attention mechanism with such sentiment information. Yang et al. (2016) and Chen et al. (2016) proposed a hierarchical RNN model to learn attention weights based on the local context using an unsupervised method. However, their method may induce much noise and suffers from a lack of interpretability because it tends to capture some domain-specific words (Mudinas et al., 2012) instead of real sentiment information. For example, in movie reviews, the name of a movie with a good reputation tends to be regarded as positive words, and is thereby assigned higher weights, which does not work in other domains. Long et al. (2017) incorporated the reading time of human beings into the attention mechanism, but their method also has much noise because during the reading process, people tend to spend more time on intricate contents (Goodman, 1988) than on real sentiment words like *good* and *bad*. Other methods employed external information such as users and products to guide attention weights (Ma et al., 2017; Chen et al., 2016), which can boost the classification performance by a large margin. However, most of the time, we have no access to such external information.

In this paper, we propose a novel Lexicon-Based Supervised Attention model (LBSA) that combines sentiment lexicons and an attention mechanism to better extract sentiment information and form

Proceedings of the 27th International Conference on Computational Linguistics, pages 868–877
Santa Fe, New Mexico, USA, August 20-26, 2018.

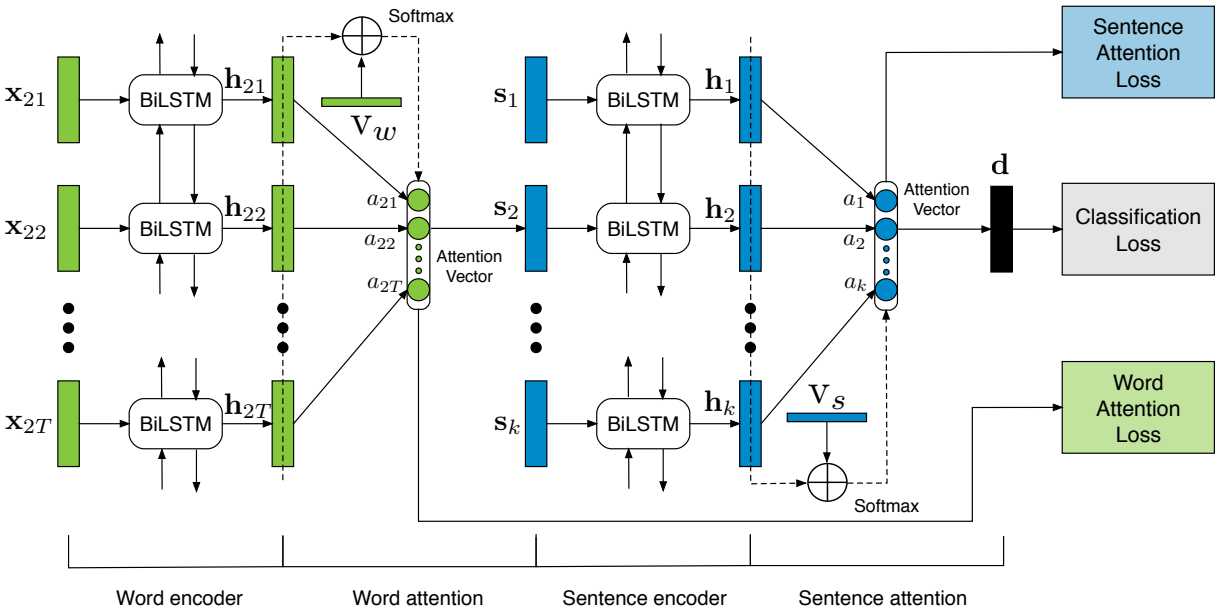


Figure 1: LBSA sentiment classification model.

sentiment-informative representations. We hypothesized that regardless of the polarities, sentiment words should be given more emphasis than other words in texts. For example, in the phrases *great film* and *terrible weather*, we should pay more attention to the sentiment words *great* and *terrible* than the words *film* and *weather*. Moreover, we define the *sentiment degree* to describe the intensity of the word sentiment, which can be attained from existing sentiment lexicons. The sentiment degree can then be used as the criterion for attention weights in a recurrent neural network model. Under the guidance of a supervised mechanism, sentiment words will be assigned higher attention weights, which reduces the influence of domain-specific words and generates sentiment-informative representations. We conducted experiments on three sentiment analysis benchmark datasets (IMDB, Yelp13, Yelp14). The experimental results showed that our model outperformed other state-of-the-art methods.

To summarize, our main contributions are as follows:

- We propose a novel lexicon-based attention model. It combines an attention mechanism with sentiment lexicons based on the *sentiment degree*. As a result, our model can precisely identify the sentiment contents in texts and capture accurate sentiment information.
- Benefiting from the guidance of supervised methods, our attention model has better interpretability and less noise than other attention models. Thus, it is better at summarizing and analyzing the sentiment of texts in an RNN fashion.
- Our model outperforms state-of-the-art methods on three sentiment analysis datasets (IMDB, Yelp13, Yelp14). This work validated the effectiveness of combining the sentiment lexicons and attention mechanism.

2 Approach

The overall architecture of our model is shown in Figure 1. It incorporates a word-level supervised attention layer and sentence-level supervised attention layer into a hierarchical bidirectional LSTM sentiment classification model. Formally, let D be a set of documents, and L be a sentiment lexicon. A document $d_m \in D$ contains k sentences S_1, S_2, ..., S_i, ..., S_k. A sentence S_i is a sequence of words w_{i1} w_{i2}...w_{it}, $t \in [1, T]$, where T denotes the length of S_i. For each word w_{it}, we define $SD^L(w_{it})$ as the *sentiment degree* (see Section 2.2.1) of w_{it} according to lexicon L. The sentiment degree is a positive real number, which can be attained from lexicons and indicates the strength of word sentiment.

As the figure shows, the word w_{it} is represented by its word embedding $\mathbf{x}_{it}$, which is fed into a bidirectional LSTM model (Graves and Schmidhuber, 2005; Graves et al., 2013) to extract sequential information. The word attention layer aggregates the representation of sentiment-informative words to form a sentence representation $\mathbf{s}_i$, with the supervision of the sentiment degree. Similarly, we can obtain the document representation $\mathbf{d}$ through a sentence-level attention layer to finally conduct sentiment classification. The concrete design is introduced in the following subsections.

2.1 Bidirectional LSTM

We adopt a Long Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) for the word-level and sentence-level feature extraction. As an example, consider the word-level case. For each word token w_t in a sentence, the model calculates a hidden state vector $\mathbf{h}_t$. The basic LSTM model has five internal vectors for node t. They control the information flow from the history $\mathbf{x}_1$ ...$\mathbf{x}_t$ and $\mathbf{h}_1$ ...$\mathbf{h}_{t-1}$ to the current state $\mathbf{h}_t$. Formally, $\mathbf{h}_t$ is calculated as follows:

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W_i}\mathbf{x}_t + \mathbf{U_i}\mathbf{h}_{t-1} + \mathbf{V_i}\mathbf{c}_{t-1} + \mathbf{b_i}) \\
\mathbf{f}_t &= \mathbf{1.0} - \mathbf{i}_t \\
\mathbf{c}'_t &= tanh(\mathbf{W_{c'}}\mathbf{x}_t + \mathbf{U_{c'}}\mathbf{h}_{t-1} + \mathbf{b_{c'}}) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{c}'_t \\
\mathbf{o}_t &= \sigma(\mathbf{W_o}\mathbf{x}_t + \mathbf{U_o}\mathbf{h}_{t-1} + \mathbf{V_o}\mathbf{c}_t + \mathbf{b_o}) \\
\mathbf{h}_t &= \mathbf{o}_t \odot tanh(\mathbf{c}_t)
\end{aligned}
$$

Each word is embedded into a low-dimensional semantic space. Namely, each word w_t is mapped to its embedding $\mathbf{x}_t \in \mathbb{R}^n$, where n is the word embedding size. $\odot$ denotes element-wise multiplication, and σ is the sigmoid function. $\mathbf{W_i}$, $\mathbf{U_i}$, $\mathbf{V_i}$, $\mathbf{b_i}$, $\mathbf{W_{c'}}$, $\mathbf{U_{c'}}$, $\mathbf{b_{c'}}$, $\mathbf{W_o}$, $\mathbf{U_o}$, $\mathbf{V_o}$, and $\mathbf{b_o}$ represent LSTM parameters.

We use a bidirectional version of LSTM (BiLSTM) (Graves and Schmidhuber, 2005; Graves et al., 2013) to obtain information from both directions for words. The bidirectional LSTM contains the forward LSTM $\overrightarrow{f}$, which reads the sentence S_i from w_{i1} to w_{it}, and the backward LSTM $\overleftarrow{f}$, which reads from w_{it} to w_{i1}. The BiLSTM model maps each word embedding $\mathbf{x}_{it}$ to a pair of hidden vectors $\overrightarrow{\mathbf{h}_{it}}$ and $\overleftarrow{\mathbf{h}_{it}}$. We use different parameters for the forward LSTM and backward LSTM. These hidden vectors are the composition of sentence embedding $\mathbf{s}_i$, and used as features for calculating attention weights. On the sentence level, we also feed sentence embeddings $[\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_k]$ into BiLSTM to obtain pairs of hidden vectors in a similar way.

2.2 Lexicon-based Supervised Attention

In this subsection, we introduce supervised attention to explicitly use sentiment lexicon information to aggregate sentiment-informative texts. Our basic idea is simple: sentiment words should be given more attention than others. To achieve this goal, we construct gold attention vectors using the lexicon information. These are employed as criteria to supervise the attention weights.

2.2.1 Gold Attention Vectors

Each word in the sentiment lexicon is annotated or assigned a fixed positive or negative score that reflects its sentiment polarity and strength. We map all of them into the interval $[-1, 1]$ as normalized sentiment scores, where a score in $(0, 1]$ represents various scales of *good* and one close to 1 means *very good*, with negations showing the opposite. Intuitively, a word with a strong positive or negative polarity requires more attention because it contains more sentiment information than a neutral word. In order to quantify the intensity of the sentiment, for each word w_{it} in sentence s_i, we define *sentiment degree* $SD^L(w_{it})$ as follows:

$$SD^L(w_{it}) = |score^L(w_{it})|, \tag{1}$$

where $score^L(w_{it})$ is the normalized sentiment score according to lexicon L. $SD^L(w_{it})$ is the absolute value of $score^L(w_{it})$, which means its range is $[0, 1]$. For words not in the lexicon, their sentiment

degrees are assigned values of 0. For sentence S_i, whose length is T, we compute word-level gold attention vectors $\mathbf{a}_i^\star$ by:

$$a_{it}^\star = \frac{exp(\lambda_w SD^L(w_{it}))}{\sum_{t=1}^{T} exp(\lambda_w SD^L(w_{it}))}. \tag{2}$$

λ_w is a positive hyper-parameter that adjusts the variance of the sentiment degree. A larger λ_w means a sharper distinction between sentiment terms and neutral ones in gold attention vectors. $a_{it}^\star$ denotes the t^{th} dimension of $\mathbf{a}_i^\star$.

We likewise construct sentence-level gold attention vectors. Given a document with k sentences, S_i denotes the i^{th} sentence in it. Empirically, a sentence tends to achieve a higher sentiment degree if it contains a higher proportion of sentiment words. Hence, we calculate the sentiment degree of S_i using a simple averaging method:

$$SD^L(S_i) = \frac{\sum_{t=1}^{T} SD^L(w_{it})}{T}. \tag{3}$$

Let λ_s be the sentence-level hyper-parameter. We then obtain sentence-level gold attention vectors $\mathbf{a}^\star$ in a similar way:

$$a_i^\star = \frac{exp(\lambda_s SD^L(S_i))}{\sum_{i=1}^{k} exp(\lambda_s SD^L(S_i))}. \tag{4}$$

2.2.2 Learned Attention Vectors

As described in the previous subsection, we obtain pairs of hidden states $\overrightarrow{\mathbf{h}_{it}}$ and $\overleftarrow{\mathbf{h}_{it}}$ from the word-level BiLSTM model. They can be employed to calculate a weight value π_{it}, which presents the sentiment strength of word w_{it}. Formally, π_{it} is defined as follows:

$$\pi_{it} = tanh(\mathbf{W}_{wf}\overrightarrow{\mathbf{h}_{it}} + \mathbf{W}_{wb}\overleftarrow{\mathbf{h}_{it}} + \mathbf{b}_w) \cdot \mathbf{v}_w^\top, \tag{5}$$

where $\mathbf{W}_{wf}$ and $\mathbf{W}_{wb}$ denote word-level weight matrices, and $\mathbf{b}_w$ is the bias vector. It is worth noting that $\mathbf{v}_w$ is a weight vector that can record historical sentiment information. It is just like the query vector employed by memory networks (Sukhbaatar et al., 2015). It is jointly learned during the training process. $\mathbf{v}_w^\top$ represents the transpose of $\mathbf{v}_w$.

After calculating the weight value π_{it}, we use a softmax function to generate the attention vector $\mathbf{a}_i$. The t^{th} dimension a_{it} denotes the attention weight of the t^{th} word w_{it} in sentence S_i. Specifically, a_{it} is computed as follows:

$$a_{it} = \frac{exp(\pi_{it})}{\sum_{t=1}^{T} exp(\pi_{it})}. \tag{6}$$

The learned attention vector $\mathbf{a}_i$ contains different weights for the corresponding words in sentence S_i. We then compute the sentence vector $\mathbf{s}_i$ as a weighted sum of the hidden vectors according to the weights, where $\mathbf{h}_{it}$ means a concatenation of $\overrightarrow{\mathbf{h}_{it}}$ and $\overleftarrow{\mathbf{h}_{it}}$:

$$\mathbf{s}_i = \sum_{t=1}^{T} a_{it}\mathbf{h}_{it}. \tag{7}$$

After obtaining sentence vectors $\mathbf{s}_i$, $1 \leq i \leq k$, we feed them into the sentence-level BiLSTM model to output hidden vector pairs. Similarly, with the attention mechanism, the hidden vector pairs $\overrightarrow{\mathbf{h}_i}$ and $\overleftarrow{\mathbf{h}_i}$, and their concatenation $\mathbf{h}_i$, document vector $\mathbf{d}$ can be induced as follows:

$$\pi_i = tanh(\mathbf{W}_{sf}\overrightarrow{\mathbf{h}_i} + \mathbf{W}_{sb}\overleftarrow{\mathbf{h}_i} + \mathbf{b}_s) \cdot \mathbf{v}_s^\top, \tag{8}$$

$$a_i = \frac{exp(\pi_i)}{\sum_{i=1}^{k} exp(\pi_i)}, \tag{9}$$

$$\mathbf{d} = \sum_{i=1}^{k} a_i\mathbf{h}_i. \tag{10}$$

Data	Class	Train Size	Dev. Size	Test Size	Sents/doc	Words/sent
IMDB	10	67,426	8,381	9,112	16.08	24.54
Yelp14	5	183,019	22,745	25,399	11.41	17.26
Yelp13	5	62,522	7,773	8,671	10.89	17.38

Table 1: Statistical information for three datasets. Sents/doc is the average number of sentences in a document, while Words/sent denotes the average length of a sentence.

2.3 Jointly Supervised Classification and Attention

The final document vector $\mathbf{d}$ is used to conduct sentiment classification, which requires the probability of labeling a document with sentiment polarity c, $c \in [1, C]$, where C represents the total number of classes. The probability is computed by a softmax function:

$$\mathbf{p} = softmax(\mathbf{W}_c\mathbf{d} + \mathbf{b}_c). \tag{11}$$

To jointly supervise the classification and attention, we introduce a soft constraint method that is employed by other tasks with supervised attention (Liu et al., 2016). Specifically, it is defined as follows:

$$\mathcal{L} = -\sum_{c=1}^{C} \mathbf{g}_c log(\mathbf{p}_c) + \mu_w \cdot \sum_{i=1}^{T} \triangle(\mathbf{a}_i^\star, \mathbf{a}_i) + \mu_s \cdot \triangle(\mathbf{a}^\star, \mathbf{a}), \tag{12}$$

where the classification loss function is cross entropy. $\mathbf{g}_c \in \mathbb{R}^C$ denotes the ground truth of the sentiment classification, presented by a one-hot vector. $\mathbf{p}_c \in \mathbb{R}^C$ is the predicted probability for each class. $\mathbf{a}_i^\star$ and $\mathbf{a}^\star$ are gold attention vectors for the word and sentence levels, respectively. $\mathbf{a}_i$ and $\mathbf{a}$ are learned attention vectors for the word and sentence levels, respectively. $\triangle$ is a loss function that indicates the disagreement between two vectors. Because the learned attention vector is a distribution, we naturally use cross entropy (CE) as the metric:

$$\triangle(\mathbf{a}^\star, \mathbf{a}) = -\sum_i \mathbf{a}_i^\star log(\mathbf{a}_i). \tag{13}$$

μ_w and μ_s are the coefficients for attention loss functions, which can balance the preference between classification and attention disagreements, to alleviate the overfitting problem.

3 Experimental Settings

In this section, we introduce the experimental settings in detail, including the datasets and lexicons, hyper-parameter settings, and baseline models.

3.1 Datasets and Lexicons

We conduct experiments to evaluate the effectiveness of our method on three document-level review datasets: IMDB, Yelp 2013 and Yelp 2014, which are developed by Tang et al. (2015). We split the datasets into training, development and testing sets in the proportion of 8:1:1, using pre-processing in the same way as Tang et al. (2015). Table 1 summarizes the statistics of the datasets. All of these datasets can be publicly accessed[1].

The sentiment lexicon contains four parts. During the process of constructing our lexicon, we only use the sentiment scores for unigrams. The first part comes from SentimentWordNet3.0 (SWN) (Baccianella et al., 2010). The original scores in SWN are the probabilities of positive, negative, or neutral polarities for words. We take the maximum sum of the positive and negative scores as the sentiment degree for different part-of-speech tags. The second part is extracted from MPQA (Wilson et al., 2005), which tags polarity ratings for sentiment words. To a word whose polarity is positive or negative, we assign 1 as its sentiment degree. Otherwise, the word's sentiment degree is 0. The third part consists of the leaf nodes of the Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013). The sentiment scores of

[1] http://ir.hit.edu.cn/dytang/paper/acl2015/dataset.7z

SST are in the range of [0, 1]. We scale it to [-1, 1] and keep the absolute value as the sentiment degree. The last part is the sentiment lexicon of the Hownet Knowledge Database (HKD)[2], which contains a list of positive or negative words. We conduct the pre-processing in the same way as for MPQA. Finally, we combine the four parts and average the sentiment degree for words appearing in two or more lexicon parts. All of the neutral words in different parts are included in our lexicon because of the ignorance of polarity. Hence, the final lexicon is huge, containing 156,242 tokens and covering 96.8% of the words in three review datasets.

The sentiment lexicon is of great significance in our experiments. A lexicon of high quality ensures the effectiveness of the supervised mechanism. Empirically, if the overlapping of the dataset and lexicon is poor, the performance will be limited. The availability of lexicons is also not a given for any domain or language. In such scenarios, several existing methods can be used to construct or extend sentiment lexicons based on large corpora in a semi-supervised way to cover the dataset (Lu et al., 2011; Hamilton et al., 2016). As the above potential problems are not the main concern of our work, we just briefly touch on this and will explore it in future work.

3.2 Hyper-parameter Settings

Following Tang et al. (2015) and Chen et al. (2016), we set the dimension of the word embeddings to 200. We use the word2vec tool (Mikolov et al., 2013) to pre-train the word embeddings. We set the dimensions of the word-level bidirectional LSTM hidden states to 100. All the weight matrices are initialized by a uniform distribution in [-0.1, 0.1]. The hyper-parameter λ_w and λ_s which adjust the variance of the gold attention vectors, are both set to 3.0. The hyper-parameter μ_w which adjusts the proportion of classification and attention loss, is set to 0.001 while μ_s is set to 0.05. We use Adam (Kingma and Ba, 2014) as the optimizer, which adopts a self-adaptive learning rate to optimize parameters, and its initial learning rate is set to 0.001. The batch size is set to 32 for efficiency. The model achieving best results on the developing set is chosen for the final evaluation of the test set.

3.3 Baseline models

We separate baseline models into four groups. In the first group, all of the methods are recently developed and achieve good performances on three review benchmarks. **SSWE + SVM** (Tang et al., 2014b) first generates sentiment-specific word embeddings to compose document presentations and then trains a SVM classifier. **Paragraph + Vector** (Le and Mikolov, 2014) learns distributed representations of a document for classification. **RNTN + RNN** (Socher et al., 2013) represents sentences with the Recursive Neural Tensor Network (RNTN) and feeds them into a recurrent neural networks (RNN) to obtain document representations. **UPNN** (Tang et al., 2015) uses a text preference matrix and vector for each user and product as extra information to train a CNN sentiment classifier. *UPNN(noUP)* only uses CNN without considering user and product information.

The models in Group 2 are based on sentiment lexicons. **BiLSTM** (Xu et al., 2016) is a bidirectional LSTM baseline with a simple average pooling layer. **AveLex** naively averages sentiment scores of words in the document to measure the overall sentiment polarities according to our sentiment lexicon. **BiLSTM + Lex** (Teng et al., 2016) takes the local context into consideration, which leverages a bidirectional LSTM to capture context information and calculates the weighted sum of the sentiment scores. The two lexicon-based methods only work on the word level because sentences do not have a gold sentiment score. For a fair comparison, **BiLSTM + LBSA** is our model, which removes the hierarchical structure.

The models in Group 3 are based on attention mechanisms. **H-LSTM** (Chen et al., 2016) is a baseline utilizing an average pooling layer at both the word and sentence levels. **H-LSTM + LA** (Chen et al., 2016) uses local context to capture semantic information as an attention mechanism. **H-LSTM + CBA** (Long et al., 2017) adds cognitive information from external reading time materials. For a fair comparison, we simplify our method from bidirectional LSTMs to basic ones, which is denoted as **H-LSTM + LBSA**.

[2]http://www.keenage.com/html/e_index.html

Model	IMDB		Yelp 2013		Yelp 2014	
	Acc.	RMSE	Acc.	RMSE	Acc.	RMSE
SSWE + SVM (Tang et al., 2014b)	0.312	1.973	0.549	0.849	0.557	0.851
Paragraph + Vector (Le and Mikolov, 2014)	0.314	1.814	0.554	0.832	0.564	0.802
RNTN + RNN (Socher et al., 2013)	0.401	1.764	0.574	0.804	0.582	0.821
UPNN(noUP) (Tang et al., 2015)	0.405	1.629	0.577	0.812	0.585	0.808
AveLex	0.223	2.388	0.312	1.393	0.334	1.202
BiLSTM (Xu et al., 2016)	0.433	1.494	0.584	0.764	0.592	0.733
BiLSTM + Lex (Teng et al., 2016)	0.441	1.466	0.588	**0.758**	0.598	0.726
BiLSTM + LBSA	**0.443**	**1.457**	**0.590**	0.761	**0.603**	**0.720**
H-LSTM (Chen et al., 2016)	0.443	1.465	0.627	0.701	0.637	0.686
H-LSTM + LA (Chen et al., 2016)	0.487	1.381	0.631	0.706	0.631	0.715
H-LSTM + CBA (Long et al., 2017)	**0.489**	1.365	0.638	0.697	0.641	0.678
H-LSTM + LBSA	0.488	**1.360**	**0.640**	**0.694**	**0.647**	**0.670**
H-BiLSTM	0.484	1.399	0.640	0.700	0.646	0.672
H-BiLSTM + LA	0.492	1.359	0.647	0.698	0.648	**0.665**
H-BiLSTM + LBSA	**0.494**	**1.322**	**0.650**	**0.691**	**0.651**	0.668

Table 2: Sentiment classification performance on different models. Acc. (Accuracy) and RMSE are the evaluation metrics. The best results are in bold in the last three groups, respectively.

In Group 4, **H-BiLSTM** is a hierarchical bidirectional LSTM model with an average pooling layer. **H-BiLSTM + LA** is a BiLSTM version of the local attention model based on Chen et al. (2016).

4 Results

This section compares our model (LBSA) with other baseline methods. Two common performance evaluation metrics are used, including Accuracy (Acc.) and Root Mean Squared Error (RMSE). They are defined as follows:

$$Accuracy = \frac{T}{N}, \tag{14}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}}. \tag{15}$$

Accuracy is a standard metric to measure the overall sentiment classification performance, where N is the number of overall samples, and T denotes the number of predicted outputs that are identical with gold labels. RMSE measures the divergences between the predicted sentiment classes and ground truth ones, where $\hat{y}_i$ and y_i represent the predicted outputs and gold labels, respectively. Table 2 illustrates the comparison results, which are separated into four groups. Some results of the baseline methods are directly taken from Long et al. (2017) and Xu et al. (2016), because we conducted experiments on the same datasets.

We can observe from Group 2 that the performances are limited without a hierarchical structure because the texts are very long in document-level sentiment classification. **AveLex** is the worst method because it naively averages the sentiment scores of words without considering any textual semantics. Compared with **BiLSTM + Lex**, our method achieves a relatively better result. The main reason is that our method ignores the prior polarity of the sentiment words in lexicons. LBSA is able to lead neural networks to concentrate on sentimental contents and learn a more accurate polarity according to historical information. To illustrate the results more specifically, we investigate the sentiment word *long* in our datasets. *Long* has a sentiment degree of 0.42 in our constructed lexicon, which indicates that it plays a relatively critical role in sentiment classification. The sentiment polarity of *long* in MPQA and SST is negative. A random sentence sample *'The several tables were available, so I didn't understand*

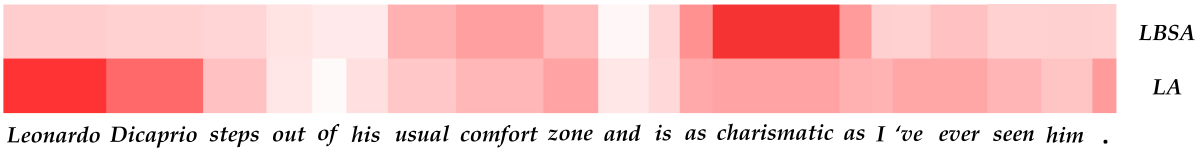


Figure 2: Sentence example from IMDB. Local attention (LA) focuses more on the domain-specific words *Leonardo Dicaprio*, while our model (LBSA) can exactly capture the real sentiment word *charismatic*.

*the **long** wait.'* is taken from a one-star review in Yelp 2013, which meets our expectation. However, in other domains or contexts, *long* is likely to be positive. For instance, the sentence *'Its appeal to me was immediate and **long** lasting.'* is extracted from a 10-star movie review in IMDB. It is obviously a positive sentence, but the prior polarity of *long* in the sentiment lexicon misleads the judgment of general methods that directly use sentiment scores.

In Group 3, we observe that our method significantly outperforms other attention mechanisms with the Yelp 2013 and Yelp 2014 datasets. The evaluation on the RMSE metric is also better than other methods with the IMDB dataset. This demonstrates that LBSA is effective at capturing sentiment information, which can be a crucial factor in sentiment classification. Compared with **H-LSTM + CBA**, our model achieves better results, which means lexicon information is more suitable for attention supervision in sentiment analysis tasks. In the forth group, the bidirectional LSTM improves the basic performance, which indicates that both forward and backward sequences indispensably contribute to information extraction. Compared with local context-based attention, the basic performance of our method is outstanding and has a relatively small improvement when a bidirectional mechanism is added. This indicates that our method lays emphasis on contents containing rich sentiment information, which is robust in a simpler structure. In contrast, local context-based attention relies heavily on contexts in both directions. The characteristic of being dedicated to the contexts may induce domain-specific words and ignore real sentiment information. Here, we use a sentence from a movie review in IMDB as an example: *'Leonardo Dicaprio steps out of his usual comfort zone and is as charismatic as I've ever seen him.'* Figure 2 illustrates the main difference between two attention mechanisms.

The LA model gives higher weights to the name *Leonardo Dicaprio* because it usually appears in a positive movie review. The words are domain-specific and attract attention from real sentiment words. In contrast, our method tends to extract specific sentiment information. As a result, it assigns a very high weight to the word *charismatic*, which indeed plays a decisive role as a positive adjective.

5 Related Work

Sentiment lexicons are widely used in sentiment analysis and opinion mining tasks because they contain rich sentiment information. Turney (2002) employed the sum of the sentiment scores of all the words in the texts that appeared in a sentiment lexicon. This is still the standard practice for specific domains with some carefully constructed domain-specific lexicons. Agarwal et al. (2013) proposed a tree-structured model leveraging non-polarity features such as POS-tags and capitalized words, adding the summation of the prior polarity scores of words, which achieved high performance on the twitter sentiment analysis benchmark. Teng et al. (2016) presented a context-sensitive lexicon-based method that uses a bidirectional LSTM to extract context information. It calculates the weighted sum of the sentiment scores of words to measure the sentiment value of a sentence. Qian et al. (2016) proposed a linguistically regularized LSTM for sentiment analysis. The model combines linguistic resources such as sentiment lexicons, negation words and intensity words with the basic LSTM model. It is able to capture the semantic information and sentiment effects in sentences more accurately. Unlike most previous studies, we incorporate the features of lexicon words into an attention mechanism, which was proven to be effective in our experiments.

Recently, the attention mechanism has been widely studied in sentiment classification. Yang et al. (2016) proposed two attention-based bidirectional GRUs to enforce the neural networks to attend to

the related part of a sentence or document. Apart from local contexts, Chen et al. (2016) incorporated extra information such as users and products in review datasets into a hierarchical attention mechanism. Ma et al. (2017) cascaded multiway of user and product information to enhance the effects of different aspects. Zhou et al. (2016) proposed a LSTM network based on an attention mechanism for cross-lingual sentiment classification at the document level. The model consists of two attention-based hierarchical LSTMs for bilingual representation. Long et al. (2017) took cognition into consideration, leveraging extra resources including the reading time of human beings. They proposed a mutimodel that learns to predict the reading time to construct the cognitive attention. The main difference between our method and others is that we supervise the attention weights using sentiment information. As a result, it is able to capture real sentiment texts and achieves a better result.

6 Conclusion

In this paper, we propose a novel lexicon-based supervised attention model. We combine an attention mechanism and sentiment lexicons to guide the neural network to focus on the sentiment content. It has better interpretability and less noise compared with other attention models. Experiments on three large review datasets validated the effectiveness of our model, showing that it can capture real sentiment information and generate sentiment-informative representations to improve the classification performance.

Acknowledgements

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by National Natural Science Foundation of China (No. 61532011, 61473092, and 61472088) and STCSM (No. 16JC1420401).

References

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. 2013. Sentiment analysis of twitter data. In *The Workshop on Languages in Social Media*, pages 30–38.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC 2010, 17-23 May 2010, Valletta, Malta*, pages 83–90.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1650–1659.

Kenneth Goodman. 1988. The reading process. *Interactive approaches to second language reading*, 6.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

William L Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2016, page 595. NIH Public Access.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. *arXiv preprint arXiv:1609.04186*.

Bing Liu. 2010. Sentiment analysis and subjectivity. *Handbook of natural language processing*, 2:627–666.

Yunfei Long, Lu Qin, Rong Xiang, Minglei Li, and Chu-Ren Huang. 2017. A cognition based attention model for sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 462–471.

Yue Lu, Malu Castellanos, Umeshwar Dayal, and ChengXiang Zhai. 2011. Automatic construction of a context-aware sentiment lexicon: an optimization approach. In *Proceedings of the 20th international conference on World wide web*, pages 347–356. ACM.

Dehong Ma, Sujian Li, Xiaodong Zhang, Houfeng Wang, and Xu Sun. 2017. Cascading multiway attentions for document-level sentiment classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 634–643.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Andrius Mudinas, Dell Zhang, and Mark Levene. 2012. Combining lexicon and learning based approaches for concept-level sentiment analysis. In *Proceedings of the first international workshop on issues of sentiment discovery and opinion mining*, page 5. ACM.

Qiao Qian, Minlie Huang, Jinhao Lei, and Xiaoyan Zhu. 2016. Linguistically regularized lstms for sentiment classification. *arXiv preprint arXiv:1611.03949*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Duyu Tang, Furu Wei, Bing Qin, Ming Zhou, and Ting Liu. 2014a. Building large-scale twitter-specific sentiment lexicon: A representation learning approach. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 172–182.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014b. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1555–1565.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1014–1023.

Zhiyang Teng, Duy Tin Vo, and Yue Zhang. 2016. Context-sensitive lexicon features for neural sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1629–1638.

Peter D. Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Meeting on Association for Computational Linguistics*, pages 417–424.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.

Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuangjing Huang. 2016. Cached long short-term memory neural networks for document-level sentiment classification. *arXiv preprint arXiv:1610.04989*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016. Attention-based lstm network for cross-lingual sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 247–256.

Open-Domain Event Detection using Distant Supervision

Jun Araki and **Teruko Mitamura**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
junaraki@cs.cmu.edu teruko@andrew.cmu.edu

Abstract

This paper introduces *open-domain event detection*, a new event detection paradigm to address issues of prior work on restricted domains and event annotation. The goal is to detect all kinds of events regardless of domains. Given the absence of training data, we propose a distant supervision method that is able to generate high-quality training data. Using a manually annotated event corpus as gold standard, our experiments show that despite no direct supervision, the model outperforms supervised models. This result indicates that the distant supervision enables robust event detection in various domains, while obviating the need for human annotation of events.

1 Introduction

Events are a key semantic component integral to natural language understanding. They are a ubiquitous linguistic phenomenon appearing in various domains. For clarification, we use the term 'domain' to refer to a specific genre of text, such as biology, finance, and so forth. Prior studies on automatic event detection traditionally focus on limited types of events, mainly defined by several research initiatives and shared tasks in a few domains:

- Newswire: TIPSTER (Onyshkevych et al., 1993) and MUC (Grishman and Sundheim, 1996)
- Multi-domain: ACE (Doddington et al., 2004) and TAC KBP (Mitamura et al., 2016).
- Biology: PASBio (Wattarujeekrit et al., 2004), GENIA (Kim et al., 2008), BioNLP (Kim et al., 2009) and ProcessBank (Berant et al., 2014).

Although closed-domain event detection might be of practical use in some domain-specific scenarios, it only contributes to partial understanding of events because it only addresses a subset of events by definition. On the other hand, there is an established consensus that in order to advance natural language applications such as open-domain question answering, we need automatic event identification techniques with larger, wider, and more consistent coverage (Saurí et al., 2005; Pradhan et al., 2007).

There are several pieces of prior work on open-domain events, but they have some limitations with respect to coverage of events. Lexical databases such as WordNet (Miller et al., 1990), FrameNet (Baker et al., 1998) and PropBank (Palmer et al., 2005) can be viewed as a superset of events, and their subtaxonomies seem to provide an extensional definition of events. However, these databases have a narrow coverage of events because they are generally expected to cover basic terminology due to their dictionary nature. For instance, none of WordNet, FrameNet and PropBank covers current terminology and proper nouns such as 'Hurricane Katrina'. OntoNotes (Weischedel et al., 2011) is aimed at covering an unrestricted set of events and entities, but its event annotation is limited to a small number of eventive nouns. TimeML (Pustejovsky et al., 2003) annotates events and times in a domain-agnostic manner, focusing on temporal aspects of events, but it does not deal with multi-word and generic events. ECB+ (Cybulska and Vossen, 2014) augments the extended EventCorefBank corpus (Lee et al., 2012) by re-annotating events and event coreference. Our initial analysis shows that it annotates events only in a portion of each document, not all events in the text. Ritter et al. (2012) address open-domain event detection on Twitter. Their annotation follows TimeML and thus is likely to have similar problems to TimeML. Richer

Proceedings of the 27th International Conference on Computational Linguistics, pages 878–891
Santa Fe, New Mexico, USA, August 20-26, 2018.

Event Description (RED) (Palmer et al., 2016) defines events in a general manner, but its annotation was performed only in the clinical domain (O'Gorman et al., 2016).

To address these issues, we introduce a new paradigm of *open-domain event detection*. Our goal is to detect all kinds of events without a specific event type, while not limiting events to particular domains and syntactic types. As compared to the traditional closed-domain event detection described above, our target is more challenging because there is no unified definition of events with wider and more consistent coverage, and there is no training data. Due to the ubiquity and ambiguities of events, human annotation of events in the open domain is substantially expensive, ending up with a small dataset which does not make supervised models generalize well. In this work, we propose a novel approach to circumvent the data sparsity problem in open-domain event detection. At the core of the approach is distant supervision from WordNet (Miller et al., 1990). Our main hypothesis is that the distant supervision allows us to build models detecting events robustly in various domains, while obviating the need for human annotation of events. We verify this hypothesis in both heuristics-based and classifier-based algorithms.

Our contribution is as follows:

1. We introduce **a new paradigm of open-domain event detection**, whose goal is to detect all kinds of events. Specifically, the differences from prior work are events in unrestricted domains and with wider coverage.
2. Our **distant supervision method** is able to generate high-quality training data (see Section 5.1). The method is not bounded to any particular datasets, offering a versatile solution for event detection. Using a manually annotated corpus as gold standard, our experiments show that despite no direct supervision, the distantly supervised model outperforms supervised models in both in-domain and out-domain settings (see Section 5.3).
3. To facilitate future studies on event detection, we **release the new corpus**[1] of human-annotated events in 10 different domains such as geology and economics. The annotated events comprise verbs, nouns, adjectives, and phrases which are continuous or discontinuous (see Section 2.2). Despite this relatively wide and flexible annotation of events, we achieved high inter-annotator agreement (see Section 4).

2 Definition of Events

We give our definition of events from both semantic and syntactic perspective.

2.1 Semantic Perspective: Eventualities

Events are a highly ambiguous notion, and thus there are many ways to define them. In this work, we use the notion of eventualities by Bach (1986) and define the three classes on the basis of durativity and telicity (Moens and Steedman, 1988; Pulman, 1997):

- **states**: notions that are durative and changeless, e.g, want, own, love, resemble
- **processes**: notions that are durative and atelic, e.g., walking, sleeping, raining
- **actions**[2]: notions that are telic or momentaneous happenings, e.g., build, walk to Boston, recognize

We define expressions as events if they denote an eventuality, i.e., a state, a process, or an action.

2.2 Syntactic Perspective: Event Nuggets

From a syntactic perspective, we use the notion of event nugget (Mitamura et al., 2015), which is a semantically meaningful unit that expresses an event. We give several examples below, where we use boldface to highlight event nuggets and underlines to show units of multi-word ones.

(1) He **opened fire** at the teller in the bank.
(2) I **cried** when my grandpa **kicked** the **bucket**.
(3) Tom was **happy** when he **received** a present.
(4) Susan **turned** the TV **on**.
(5) She **responded** his email **dismissively**.

[1] https://bitbucket.org/junaraki/coling2018-event

[2] Bach (1986) uses the term 'events' to refer to this class. In this work, we use 'actions' instead for clarification.

Event nuggets can be either a single word (verb, noun, or adjective) or a phrase which is continuous or discontinuous. For example, 'turned ... on' in (2) is a discontinuous phrasal event nugget, excluding 'the TV'. For more details, we refer readers to our annotation guidelines.[3]

3 Distantly Supervised Event Detection

This section describes our approach for open-domain event detection. Figure 1 shows a high-level overview of the approach. As shown, the algorithm comprises two phases: training data generation and event detection. At the core of the approach is distant supervision from WordNet[4] in the former phase to address ambiguities on eventiveness and generate high-quality training data automatically.

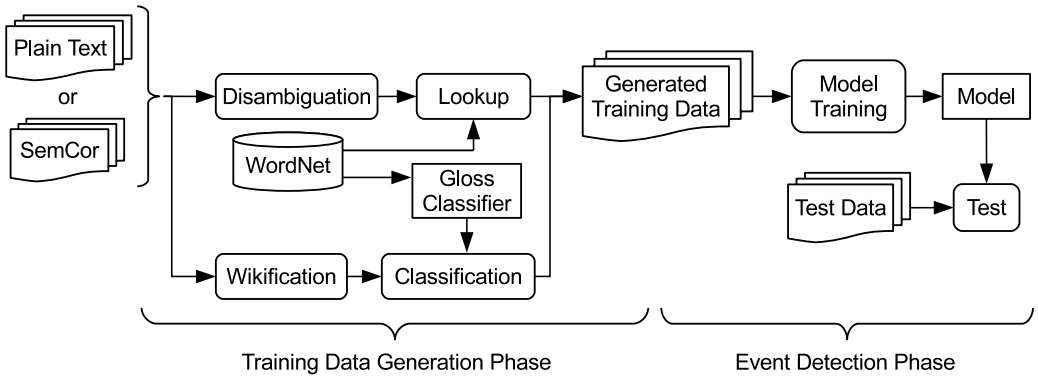


Figure 1: An overview of our distantly-supervised open-domain event detection.

3.1 Training Data Generation

The goal of training data generation is to generate high-quality training data automatically from unannotated text. We first disambiguate text using a WordNet-based word sense disambiguation tool. Given sense-annotated text as input, we implement a rule-based algorithm, which we refer to as **RULE**. It is our basis for generating training data. Looking at our annotation guidelines (see Section 4), we employ several heuristics to identify event nuggets, according to syntactic types:

Verbs. We detect most of main verbs as eventive, excluding be-verbs and auxiliary verbs. However, some exceptions exist. In the examples below, we use italic face to highlight non-eventives.

(6) That is what I **meant**.
(7) 'Enormous' *means* 'very big'.

In (6), 'meant' is eventive because it indicates the action of "intend to say" whereas 'means' in (7) is not because it merely shows equality, playing the almost same role as a be-verb. Thus, we define a set of non-eventive verb senses and filter out verbs if their disambiguated sense is in the set.

Nouns. There are also ambiguous nouns:

(8) His **payment** was late.
(9) His *payment* was $10.
(10) **Force** equals mass times acceleration.

In (8), 'payment' is eventive because it means the action of his paying something while 'payment' in (9) is not because it refers to specific money paid by him, which is $10. These examples also show that eventive nouns cannot be simply approximated by verb nominalizations. 'Force' in (10) can be easily disambiguated to its physical sense, but still deciding its eventiveness is difficult. To address the issue, we make use of distant supervision from WordNet. Let word_n^i denote the i-th sense (synset) of noun word in WordNet. For example, car_n^1 is the first synset of noun 'car'. Looking at textual definitions of synsets called *glosses*, we assume that there is a semantic correspondence between the components of eventualities described in Section 2.1 and the following WordNet synsets:

- state_n^2: the way something is with respect to its main attributes

[3]We release the guidelines at `https://bitbucket.org/junaraki/coling2018-event`.
[4]We access WordNet 3.0 using NLTK (Bird et al., 2009).

- process^{6}_{n}: a sustained phenomenon or one marked by gradual changes through a series of states
- event^{1}_{n}: something that happens at a given place and time

We detect nouns as events if their disambiguated sense is subsumed by the three synsets above through (instance-)hyponym relations.

Adjectives. Adjectives can also be ambiguous:

(11) Mary was **talkative** at the **party**.

(12) Mary is a *talkative* person.

In (11), 'talkative' is eventive because it implies that Mary talked a lot at the party, whereas 'talkative' in (12) is not because it just indicates Mary's personal attribute. As illustrated, major problems with adjectives are to differentiate states from attributes and to figure out if they imply actual occurrences (Palmer et al., 2016). Unlike nouns, no direct supervision is available from WordNet because it does not have hyponym taxonomies for adjectives. Thus, we use simple and conservative heuristics to detect adjectives as events if they are originated from present and past participles of verbs, illustrated as follows:

(13) There is a **man-made** river in the country.

(14) The tower has 20,000 **sparkling** lights.

We convert these adjectives to verbs in the infinitive form using pattern.en (De Smedt and Daelemans, 2012).

Adverbs. We develop heuristics using WordNet. We first convert adverbs to adjectives by looking up pertainyms (relational adjectives) and seeking verbs derivationally related to the adjectives in WordNet. Adverbs connect with their modifying verbs, forming a single event nugget, as illustrated in (5). Thus, we combine eventive adverbs with such verbs to detect resulting verb phrases as events.

Phrases. Following Schneider et al. (2014), we define *phrases* to be lexicalized combinations of two or more words that are exceptional enough to be considered as single units in the lexicon. We assume that this definition is suitable to event detection because the exceptionality of multi-word units in the phrase lexicon translates to the meaningfulness of textual units of (phrasal) event nuggets. From the perspective of open-domain event detection, supervised phrase detection models are likely suboptimal because they might be limited to particular domains or overfitting to small datasets. Therefore, we explore a simple dictionary-lookup approach to detect WordNet phrasal verbs as events, inspired by Yin and Schütze (2015). One enhancement to their approach is that we examine dependencies using Stanford CoreNLP (Manning et al., 2014), illustrated as follows:

(15) Snipers were **picking** them **off**.

(16) He **picked** an apple off the tree.

In (15), 'picking ... off' forms a discontinuous phrasal verb, whereas 'picked' in (16) does not. Dependencies can be of help to resolve these two cases. In the former case, a dependency relation 'picking $\xrightarrow{\texttt{compound:prt}}$ off' is a direct signal of the phrasal verb construction. As for noun phrases, we apply our heuristics for nouns to head words of the phrases.

3.2 Enhancements with Wikipedia

This subsection describes two techniques to enhance our training data generation: heuristics-based enhancement (Section 3.2.1) and classifier-based enhancement (Section 3.2.2). One disadvantage of RULE is the limited coverage of WordNet. As described in Section 1, WordNet does not cover many proper nouns that we generally see in newspaper articles, such as the following:

(17) Property damage by **Hurricane Katrina** around $108 billion.

(18) The **Cultural Revolution** was ...

In order to achieve higher recall, we incorporate Wikipedia to capture proper nouns which are not in WordNet, motivated by the fact that Wikipedia has a much broader coverage of concepts than WordNet synsets.[5] We use the Illinois Wikifier (Ratinov et al., 2011) to extract Wikipedia concepts from text.

[5]WordNet 3.0 has 120K synsets, and English Wikipedia has 5.5M articles as of October 2017, as shown at `https://stats.wikimedia.org/EN/TablesWikipediaEN.htm`.

3.2.1 Heuristics-based Enhancement

For our first enhancement, we make two assumptions: (1) the first sentence of a Wikipedia article provides a high-quality gloss of its corresponding concept, and (2) the syntactic head of a gloss represents a high-level concept carrying significant information to decide eventiveness. The first assumption is supported by Wikipedia's style manual on how to write the first sentence of an article.[6] The manual says "If an article's title is a formal or widely accepted name for the subject, display it ... as early as possible in the first sentence." For instance, the first sentence of entry Electron is:

(19) The electron is a subatomic particle with a negative elementary electric charge.

The gloss of Electron is the underlined text above. Our analysis shows that most Wikipedia articles follow the first-sentence format.

The second assumption is illustrated by the syntactic head of the Electron gloss, which is 'particle'. Based on the assumptions, we develop head-based heuristics, which we call **HeadLookup**. We find the syntactic head of a gloss using dependencies and disambiguate the head using a state-of-the-art word sense disambiguation tool IMS (It Makes Sense) (Zhong and Ng, 2010). We then check if the head's sense is subsumed by the three synsets of state^2_n, $\mathsf{process}^6_n$, and event^1_n. In the case of Electron, the head's sense atom^2_n is not under the synsets. Thus, the model concludes that Electron is non-eventive. Note that HeadLookup itself is a general technique which can be applied to any gloss. Our first enhancement applies it to Wikipedia glosses, and we refer to the enhanced model as **RULE-WP-HL**.

3.2.2 Classifier-based Enhancement

Our second enhancement leverages a binary gloss classifier to decide the eventiveness of proper nouns. We refer to this enhanced model as **RULE-WP-GC**. We use WordNet glosses to train the classifier. Our assumption is that although WordNet and Wikipedia are maintained by different people for different purposes, the classifier trained on WordNet glosses generalizes well against unseen Wikipedia concepts because glosses of the two resources have comparable quality.

Data Collection from WordNet. We collect our gloss datasets automatically from WordNet. The goal of data collection is to create a large amount of dataset $D = D_+ \cup D_-$ where D_+ is a set of eventive (positive) glosses, D_- is a set of non-eventive (negative) ones, and $D_+ \cap D_- = \emptyset$. Since WordNet provides a gloss for each synset, the goal reduces to creating a set of positive synsets S_+ and a set of negative ones S_-. Given root synset s, we collect a subset of synsets S_s (including s) by traversing the WordNet taxonomy under s through hyponym and instance-hyponym relations. Using the three synsets introduced in Section 3.1, we have $S_+ = S_{\mathsf{event}^1_n} \cup S_{\mathsf{state}^2_n} \cup S_{\mathsf{process}^6_n}$. With respect to S_-, we simply take all the WordNet synsets that are not in S_+. Table 1 gives several examples of glosses in D_+ and D_-. As shown, the ambiguous word 'payment' has positive and negative synsets. The sizes of D_+ and D_- are $|D_+| = 13,415$ and $|D_-| = 68,700$.

Synset	Gloss
riding^2_n	travel by being carried on horseback
shower^3_n	a brief period of precipitation
$\mathsf{payment}^2_n$	the act of paying money
pork^1_n	meat from a domestic hog or pig
year^1_n	a period of time containing 365 (or 366) days
$\mathsf{payment}^1_n$	a sum of money paid or a claim discharged

Table 1: Examples of WordNet glosses in D. The examples above and below the dashed line are from D_+ and D_-, respectively.

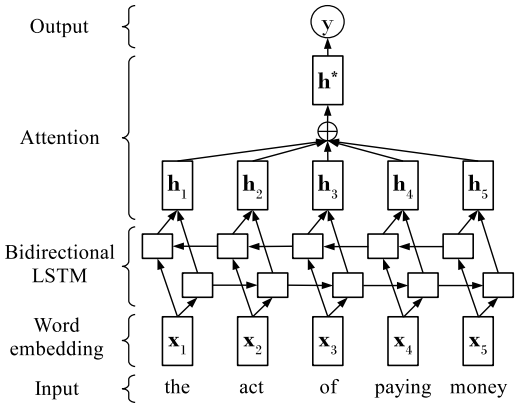


Figure 2: The bidirectional LSTM model with a self-attention mechanism.

[6] https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Lead_section\#First_sentence

Learning. We train the binary classifier on D using a bidirectional long short-term memory (BLSTM) (Graves and Schmidhuber, 2005). We call the classifier **GC-BLSTM**. Given an input vector $\mathbf{x}_t$ at time step t, let us denote the forward hidden state as $\overrightarrow{\mathbf{h}_t}$ and the backward one as $\overleftarrow{\mathbf{h}_t}$. We obtain the hidden state of a BLSTM using concatenation: $\mathbf{h}_t = [\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}]$. We then use a linear projection of $\mathbf{h}_T$ into two classes: $y = 1$ (eventive) and $y = 0$ (non-eventive). Finally, we add a softmax layer on top of the linear projection, and train the model using the binary cross-entropy loss.

Attention. Neural networks with attention mechanisms have achieved great success in a wide variety of natural language tasks. The basic idea is to enable the model to attend to all past hidden vectors and put higher weights on important parts so that the model can encode the sequence information more effectively. This idea intuitively makes sense for gloss classification as well, because syntactic heads are likely more important, as illustrated in Section 3.2.1. As shown in Figure 2, we leverage a self-attention mechanism, following (Zhou et al., 2016; Lin et al., 2017). Let $\mathbf{H} \in \mathbb{R}^{d \times T}$ denote a matrix comprising hidden vectors $[\mathbf{h}_1, \ldots, \mathbf{h}_T]$ where d is the dimensionality of a hidden vector. The self-attention mechanism computes the hidden state as follows:

$$\mathbf{M} = \tanh(\mathbf{H}) \tag{1}$$

$$\alpha = \text{softmax}(\mathbf{w}^T \mathbf{M}) \tag{2}$$

$$\mathbf{r} = \mathbf{H}\alpha^T \tag{3}$$

$$\mathbf{h}^* = \tanh(\mathbf{r}) \tag{4}$$

where a vector $\alpha \in \mathbb{R}^T$ is attention weights and $\mathbf{w} \in \mathbb{R}^d$ is a parameter vector. We refer to the attention-based classifier as **GC-BLSTM-Attn**.

Implementation Details. We randomly sample 1,000 examples from each of D_+ and D_- to create a test set and a validation set, and use the rest of D for a training set. We use the 300-dimensional GloVe vectors[7] from (Pennington et al., 2014) and do not fine-tune them during training. We map all out-of-vocabulary words to a single vector randomly initialized by uniform sampling from $[-0.01, 0.01]$. We use a single hidden layer of 100 dimensions, i.e., $d = 100$. We optimize model parameters using minibatch stochastic gradient descent (SGD) with momentum 0.9. We choose an initial learning rate of $\eta_0 = 1.0\text{e}{-3}$. We use a minibatch of size 1. To mitigate overfitting, we apply the dropout method (Srivastava et al., 2014) to the inputs and outputs of the network. We also employ L2 regularization. We perform a small grid search over combinations of dropout rates $\{0.0, 0.1, 0.2\}$ and L2 regularization penalties $\{0.0, 1.0\text{e}{-3}, 1.0\text{e}{-4}\}$. We use early stopping based on performance on the validation set.

3.3 Learning for Event Detection

After generating training data, we train a supervised event detection model on the synthesized data. As seen in the self-training model by Liao and Grishman (2011), erroneously generated training data worsen system performance. Therefore, we need to generate training data as accurately as possible. On the other hand, our algorithm for generating training data comprises at least three non-trivial (error-prone) submodules: disambiguation, wikification, and gloss classification. To eliminate negative effects of disambiguation errors, we choose the SemCor corpus (Miller et al., 1993) as our base text for training data generation. SemCor has human-annotated WordNet senses on 186 documents in numerous genres. We apply our rule-based event detector to generate training data automatically from SemCor.

We formalize event detection as a sequence labeling problem and employ a BLSTM for sequence modeling. One difference from traditional sequence labeling problems is that our output includes discontinuous phrases. Thus, we leverage an extended version of the standard BIO tagging scheme, inspired by Metke-Jimenez and Karimi (2016). Besides the three tags of B, I and O, we introduce two additional tags: DB and DI. DB means the beginning of a discontinuous concept, and DI the continuation of a discontinuous concept. The BLSTM model computes a hidden representation from each input word and then predicts one of $\{\text{B}, \text{I}, \text{DB}, \text{DI}, \text{O}\}$. For the BLSTM model, we use a fixed concatenation of the GloVe embeddings and 50-dimensional word embeddings from Turian et al. (2010) under the same assumption

[7] https://nlp.stanford.edu/projects/glove/

as (Lee et al., 2017) that different learning objectives of the GloVe and Turian embeddings can provide orthogonal information. We additionally employ 10-dimensional part-of-speech embeddings. We train the model with the objective of minimizing cross-entropy loss. We use early stopping based on the loss on a validation set.

4 Open-Domain Event Corpus

Since our target is unrestricted domains, we need gold standard data to evaluate whether systems work robustly in various domains. However, annotating events manually in all domains would be unrealistic due to annotation cost. To make the corpus creation manageable while retaining the domain diversity, we select 100 articles in Simple English Wikipedia, comprising 10 from each of 10 different domains. The domains are: architecture, chemistry, disasters, diseases, economics, education, geology, history, politics, and transportation. We choose Simple Wikipedia because our annotators are not necessarily experts in these domains, and the simplified sentences could facilitate our annotation of events against text from the wide variety of domains. Three annotators perform annotation independently following our annotation guidelines, and we measure inter-annotator agreement using the pairwise F1 score under two conditions: strict match and partial match. The former checks whether two annotations have exactly the same span. The latter checks whether there is an overlap between annotations, with the restriction that each annotation can only be matched to one annotation by the other annotator. We compute the agreements for all annotator pairs and average them for the overall agreement. As a result, the inter-annotator agreement was 80.7% (strict match) and 90.3% (partial match). Finally, the most experienced annotator finalizes event annotation. We refer to the event corpus as **SW100**.[8]

Table 2(a) shows that event nuggets appear in the 10 domains almost uniformly, ensuring the ubiquity of events. We use Stanford CoreNLP to tokenize text and decide head words of event nuggets with dependencies. Table 2(b) counts multi-word event nuggets by part-of-speech tags of their heads, amounting to 955. 24% of the 955 are discontinuous, and most (97%) of the discontinuous multi-word event nuggets are verb phrases. 'Others' in Table 2(b) include pronouns, demonstrative determiners, and numbers.

Domain	# (%)	Domain	# (%)
Architecture	475 (8.8)	Education	653 (12.1)
Chemistry	576 (10.7)	Geology	483 (8.9)
Disaster	510 (9.4)	History	486 (9.0)
Disease	618 (11.4)	Politics	534 (10.0)
Economics	479 (8.9)	Transportation	583 (10.8)

(a) Event nuggets with respect to domains.

	Single-word	Multi-word	All
Verb	2799 (51.9)	560 (10.4)	3359 (62.2)
Noun	1273 (23.6)	382 (7.1)	1655 (30.6)
Adjective	192 (3.6)	2 (0.0)	194 (3.6)
Others	178 (3.3)	11 (0.2)	189 (3.5)
All	4442 (82.3)	955 (17.7)	5397 (100.0)

(b) Event nuggets with respect to syntactic types.

Table 2: Corpus statistics of SW100. Percentages (%) are shown in parentheses.

5 Experimental Results

This section describes our experimental results.

5.1 Results of Gloss Classification

Gloss classification is a binary classification subtask for training data generation, aimed to improve event detection by capturing proper nouns with Wikipedia knowledge. We use two datasets to evaluate our gloss classifiers described in Section 3.2. One is the test set of WordNet glosses described in Section 3.2.2. However, we actually care about the gloss classification performance against unseen Wikipedia concepts. Thus, we create an additional dataset which comprises Wikipedia concepts that do not appear in WordNet. We collect 100 eventive and 100 non-eventive Wikipedia glosses in 10 domains[9], independently of SW100. We measure the performance of gloss classification using accuracy. For comparison, we consider two baselines. **BoW-LR** is a bag-of-words model trained with logistic

[8] https://bitbucket.org/junaraki/coling2018-event

[9] Economics, history, politics, psychology, architecture, earth science, physics, chemistry, biology, and medicine.

regression, and **DAN** is a deep average network proposed by Iyyer et al. (2015). Since DAN can be seen as a neural bag-of-words model, these two models are word-order insensitive baselines. Table 3 shows that GC-BLSTM-Attn performs best and significantly better than GC-BLSTM on both WordNet and Wikipedia datasets. This result verifies our assumptions that the classifier trained on WordNet glosses generalizes well against unseen Wikipedia concepts and that the attention mechanism is effective for gloss classification.

Model	WordNet	Wikipedia
HeadLookup	77.80	73.50
BoW-LR	79.50	73.00
DAN	83.15	64.00
GC-BLSTM	90.10	80.00
GC-BLSTM-Attn	**91.65****	**85.00***

Table 3: Accuracy of gloss classifiers. The stars indicate statistical significance compared to GC-BLSTM (*: $p < 0.05$; **: $p < 0.005$) based on McNemar's test.

Model	Strict			Partial		
	P	R	F1	P	R	F1
VERB (Baseline)	79.5	51.7	62.7	95.4	62.0	75.2
PRED (Baseline)	55.1	62.4	58.5	67.6	76.6	71.8
RULE	80.1	77.0	78.5	89.0	85.5	87.2
RULE-WP-HL	80.5	77.5	79.0	88.6	85.3	86.9
RULE-WP-GC	80.8	77.7	**79.2**	89.1	85.7	**87.3**

Table 4: Performance of the rule-based event detectors on SW100.

5.2 Results of Training Data Generation

We measure the performance of the rule-based event detectors on SW100 using precision (P), recall (R), and F1 with the two matching options described in Section 4. We use IMS (It Makes Sense) for disambiguation. Table 4 shows the results. **VERB** is a simple baseline that detects all single-word main verbs as events, excluding be-verbs and auxiliary verbs. **PRED** is another baseline that detects all predicates as events by running a state-of-the-art semantic role labeler called PathLSTM[10] (Roth and Lapata, 2016). Since PathLSTM is trained on both PropBank and NomBank, it is able to detect both verbal and nominal predicates. However, semantic role labeling has a different focus on predicate-argument structures. More specifically, the combination of PropBank and NomBank has a narrower coverage of events while having non-event predicates. This difference explains PathLSTM's relatively low performance of 58.5 strict F1, even underperforming the VERB baseline. The performance difference between RULE and VERB mostly comes from nouns, indicating that our WordNet-based heuristics is effective. We found that RULE-WP-GC achieves the best F1. This result shows that the proposed enhancements with Wikipedia can contribute to generating higher-quality training data.

We then applied the best-performing RULE-WP-GC to SemCor and found that the generated data contains 59,796 event nuggets in total. We randomly split this data or its subset to 9:1 with respect to the number of documents, creating training and validation data. We train the neural event detector described in Section 3.3 on the training data and measure its performance on SW100. Figure 3 shows how the amount of training data affects the performance of event detection. As shown, a larger amount of training data enables the model to achieve better performance.

5.3 Comparison with Supervised Models

In order to test the robustness of our distant supervision model, in this experiment we divide SW100 into in-domain and out-domain datasets by randomly sampling 5 domains for each. We further split the in-domain dataset into training (60%), validation (20%) and test subsets (20%) with respect to the number of documents, and then train the BLSTM event detection model described in Section 3.3. We repeat this procedure three times and measure the average of F1 scores with strict match (Table 5). We refer to the model trained on the in-domain data as **BLSTM** and the model trained on the data generated from SemCor as **DS-BLSTM**. In all the three runs, DS-BLSTM outperforms BLSTM in both in-domain and out-domain settings. The performance difference in the out-domain setting is statistical significant at $p < 0.05$ based on a two-tailed t-test. BLSTM ends up overfitting to the training data, and its weak generalization power is more evident in the out-domain setting. In contrast, DS-BLSTM performs robustly in both settings. Table 6 shows more detailed performance of DS-BLSTM.

[10] https://github.com/microth/PathLSTM

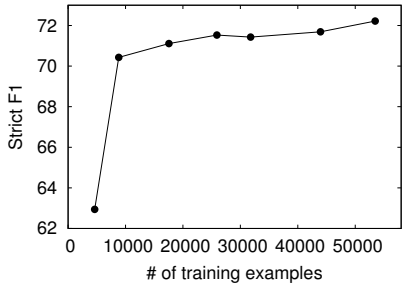


Figure 3: Performance of the event detection model on SW100 with respect to the number of training examples generated from SemCor.

Setting	Model	Strict F1	Partial F1
In-domain	BLSTM	73.8	85.9
	DS-BLSTM	**76.1**	**88.0**
Out-domain	BLSTM	67.9	82.8
	DS-BLSTM	**71.3**	**86.6**

Table 5: Results of event detection.

Domain	Strict			Partial		
	P	R	F1	P	R	F1
Architecture	81.1	71.2	75.8	92.3	81.1	86.3
Chemistry	75.5	71.2	73.3	91.0	85.8	88.3
Disaster	80.7	70.6	75.3	95.7	83.7	89.3
Disease	66.6	53.2	59.2	89.9	71.8	79.9
Economics	73.7	67.8	70.7	93.2	85.8	89.3
Education	71.8	67.4	69.5	86.9	81.6	84.2
Geology	78.6	71.6	75.0	92.3	84.1	88.0
History	77.4	69.8	73.4	92.2	83.1	87.4
Politics	78.2	69.7	73.7	93.5	83.3	88.1
Transportation	81.5	75.6	78.5	91.5	84.9	88.1

(a) With respect to domains.

Syntactic type	Strict			Partial		
	P	R	F1	P	R	F1
Verbs	79.7	83.3	81.5	94.9	99.2	97.0
Nouns	67.4	51.7	58.5	82.5	63.4	71.7
Adjectives	68.9	26.3	38.1	68.9	26.3	38.1

(b) With respect to syntactic types.

Table 6: Detailed performance of DS-BLSTM.

6 Error Analysis and Discussion

This section discusses our error analysis.

Analysis of Gloss Classification. One error pattern is that the gloss of an eventive example is partially or completely overlapped with that of non-eventive one, confusing the classifier:

- broadcast^2_n: a radio or television show
- laugh_track^1_n: prerecorded laughter added to the soundtrack of a radio or television show

The former is eventive and the latter is not. Beside such errors, we found possible inconsistencies in WordNet entries in terms of eventiveness:

- sufficiency^1_n: sufficient resources to provide comfort and meet obligations
- unanimity^1_n: everyone being of one mind
- minority^3_n: any age prior to the legal age

The first three synsets are in the state^2_n taxonomy, but the ways of defining them, especially the syntactic heads of glosses (underlined above) sound like non-eventive entities rather than events.

#	Submodule	Description	Examples
(1)	Phrase detection	A phrase is missing or incorrectly detected.	amount to, stay clear, take one's toll
(2)	Wikification	A proper name is not identified or disambiguated into an incorrect entry in Wikipedia.	Polish Revolution, Battle of The Little Horn
(3)	Wikipedia gloss extraction	A corresponding Wikipedia article does not provide a gloss of an expected form.	Spanish flu, Exxon Valdez oil spill
(4)	Gloss classification	A disambiguated gloss is misclassified.	Anglican parson, Archbishop

Table 7: Noticeable errors of our training data generation.

Analysis of Training Data Generation. Besides gloss classification, our training data generation is subject to errors from the rule-based event detection and wikification. Table 7 shows noticeable errors in training data generation. The cause of error (1) is small coverage of WordNet phrases, particularly verb phrases. Error (2) and (4) can be reduced by more sophisticated wikification and gloss classification, respectively. An example of error (3) is that Wikipedia entry Spanish_flu is empty because it is redirected

to 1918_flu_pandemic. A simple remedy of partial help to error (3) is to resolve such empty concepts using Wikipedia redirect relations.

Analysis of Event Detection. Two major error sources of the DS-BLSTM model are nouns and phrases. Our training data generated from SemCor is 11 times larger than SW100 with respect to the number of event nuggets. Still, many nouns and phrases do not appear in the training data, making correct predictions difficult. As shown in Table 5(a), the most difficult domain is 'Disease' where numerous domain-specific terms, such as migraine and bubonic plague, can appear even in simplified text of Simple Wikipedia, but not in SemCor at all.

7 Related Work

Most prior work has addressed event detection using supervised models based on symbolic features. Some studies employ token-level classifiers (Ahn, 2006; Ji and Grishman, 2008; Liao and Grishman, 2010b; Hong et al., 2011; Berant et al., 2014), while others cast event detection as a sequence labeling problem and apply structured prediction models (McClosky et al., 2011; Lu and Roth, 2012; Li et al., 2013; Li et al., 2014; Araki and Mitamura, 2015; Yang and Mitchell, 2016). Recent work has explored neural network models with distributional features (Ghaeini et al., 2016; Nguyen et al., 2016; Chen et al., 2015; Nguyen and Grishman, 2015; Feng et al., 2016). These models are typically trained on a small amount of human-annotated data in closed domains and thus subject to overfitting.

Semi-supervised and unsupervised approaches are less studied than supervised ones in event detection. Several studies leverage bootstrapping methods to find new patterns for similar events (Liao and Grishman, 2010a; Liu and Strzalkowski, 2012; Huang and Riloff, 2012; Huang and Riloff, 2013). Others explore self-training (Liao and Grishman, 2011), event vector representation (Peng et al., 2016), tensor-based composition (Huang et al., 2016), and distant supervision (Chen et al., 2017). However, their models focus on predicate-argument structures and are validated in a few domains, mostly in ACE. It is unclear how well they scale to the open domain, particularly to phrases and proper nouns.

Several lines of recent work refine the definition of events. Rich ERE (LDC, 2015) defines events under a particular event ontology mainly from the syntactic perspective, without clarifying what semantically constitutes events. The ISO-TimeML specification (Pustejovsky et al., 2010) defines events as "something that can be said to obtain or hold true, to happen or to occur." This definition is consistent with Bach's eventualities and the closest to ours. The studies described above have narrower coverage of events than our work in terms of domains, syntactic types, or multi-word expressions.

8 Conclusion and Future Work

We have introduced open-domain event detection, a new event detection paradigm whose goal is to detect all kinds of events regardless of domains. Due to the ubiquity and ambiguities of events, human annotation of events in the open domain is substantially expensive. We presented a distant supervision method that is able to generate high-quality training data automatically, obviating the need for human annotation. Our distantly supervised model is not bounded to any particular datasets and offers a versatile solution for event detection. Our experiment shows that the model outperforms supervised models in both in-domain and out-domain settings.

There are numerous avenues for future work. One could conduct experiments on normal English text, such as newspaper articles, in various domains. We plan on event coreference resolution to detect eventive pronouns and demonstrative determiners. We recognize that event types and epistemic status are key features for event coreference resolution, and thus extracting them is an important problem.

Acknowledgements

This publication was partly made possible by grant NPRP-08-1337-1-243 from the Qatar National Research Fund (a member of the Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of COLING/ACL Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

Jun Araki and Teruko Mitamura. 2015. Joint event trigger identification and event coreference resolution with structured perceptron. In *Proceedings of EMNLP*, pages 2074–2080.

Emmon Bach. 1986. The algebra of events. *Linguistics and Philosophy*, 9:5–16.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proceedings of COLING*, pages 86–90.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of EMNLP*, pages 1499–1510.

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

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of ACL/IJCNLP*, pages 167–176.

Yubo Chen, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *Proceedings of ACL*.

Agata Cybulska and Piek Vossen. 2014. Using a sledgehammer to crack a nut? Lexical diversity and event coreference resolution. In *Proceedings of LREC*, pages 4545–4552.

Tom De Smedt and Walter Daelemans. 2012. Pattern for python. *Journal of Machine Learning Research*, 13:2063–2067.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program tasks, data, and evaluation. In *Proceedings of LREC*, pages 837–840.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of ACL*, pages 66–71.

Reza Ghaeini, Xiaoli Fern, Liang Huang, and Prasad Tadepalli. 2016. Event nugget detection with forward-backward recurrent neural networks. In *Proceedings of ACL*, pages 369–373.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference-6: A brief history. In *Proceedings of COLING*, pages 466–471.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of ACL-HLT*, pages 1127–1136.

Ruihong Huang and Ellen Riloff. 2012. Bootstrapped training of event extraction classifiers. In *Proceedings of EACL*, pages 286–295.

Ruihong Huang and Ellen Riloff. 2013. Multi-faceted event recognition with bootstrapped dictionaries. In *Proceedings of NAACL-HLT*, pages 41–51.

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare R. Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of ACL*, pages 258–268.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of ACL/IJCNLP*, pages 1681–1691.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL-HLT*, pages 254–262.

Jin-Dong Kim, Tomoko Ohta, and Jun'ichi Tsujii. 2008. Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics*, 9:10.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of BioNLP'09 shared task on event extraction. In *Proceedings of BioNLP-ST Workshop*, pages 1–9.

LDC, 2015. *DEFT Rich ERE Annotation Guidelines: Events*. Linguistic Data Consortium. Version 3.0.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of EMNLP/CoNLL*, pages 489–500.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of EMNLP*, pages 188–197.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of ACL*, pages 73–82.

Qi Li, Heng Ji, Yu Hong, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of EMNLP*, pages 1846–1851.

Shasha Liao and Ralph Grishman. 2010a. Filtered ranking for bootstrapping in event extraction. In *Proceedings of COLING*, pages 680–688.

Shasha Liao and Ralph Grishman. 2010b. Using document level cross-event inference to improve event extraction. In *Proceedings of ACL*, pages 789–797.

Shasha Liao and Ralph Grishman. 2011. Can document selection help semi-supervised learning? A case study on event extraction. In *Proceedings of ACL-HLT*, pages 260–265.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *Proceedings of ICLR*.

Ting Liu and Tomek Strzalkowski. 2012. Bootstrapping events and relations from text. In *Proceedings of EACL*, pages 296–305.

Wei Lu and Dan Roth. 2012. Automatic event extraction with structured preference modeling. In *Proceedings of ACL*, pages 835–844.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings ACL: System Demonstrations*, pages 55–60.

David McClosky, Mihai Surdeanu, and Christopher Manning. 2011. Event extraction as dependency parsing. In *Proceedings of ACL-HLT*, pages 1626–1635.

Alejandro Metke-Jimenez and Sarvnaz Karimi. 2016. Concept identification and normalisation for adverse drug event discovery in medical forums. In *Proceedings of the Workshop on Biomedical Data Integration and Discovery*.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1990. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–244.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Proceedings of the Workshop on Human Language Technology*, pages 303–308.

Teruko Mitamura, Yukari Yamakawa, Susan Holm, Zhiyi Song, Ann Bies, Seth Kulick, and Stephanie Strassel. 2015. Event nugget annotation: Processes and issues. In *Proceedings of NAACL-HLT 2015 Workshop on Events: Definition, Detection, Coreference, and Representation*, pages 66–76.

Teruko Mitamura, Zhengzhong Liu, and Eduard Hovy. 2016. Overview of TAC-KBP 2016 Event Nugget track. In *Proceedings of Text Analysis Conference*.

Marc Moens and Mark Steedman. 1988. Temporal ontology and temporal reference. *Computational Linguistics*, 14(2):15–28.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of ACL/IJCNLP*, pages 365–371.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of NAACL-HLT*, pages 300–309.

Tim O'Gorman, Kristin Wright-Bettner, and Martha Palmer. 2016. Richer Event Description: Integrating event coreference with temporal, causal and bridging annotation. In *Proceedings of the 2nd Workshop on Computing News Storylines*, pages 47–56.

Boyan Onyshkevych, Mary Ellen Okurowski, and Lynn Carlson. 1993. Tasks, domains, and languages for information extraction. In *Proceedings of the TIPSTER Text Program: Phase I*, pages 123–133.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–105.

Martha Palmer, Will Styler, Kevin Crooks, and Tim O'Gorman, 2016. *Richer Event Description (RED) Annotation Guidelines*. University of Colorado at Boulder. Version 1.7, `https://github.com/timjogorman/RicherEventDescription/blob/master/guidelines.md`.

Haoruo Peng, Yangqiu Song, and Dan Roth. 2016. Event detection and co-reference with minimal supervision. In *Proceedings of EMNLP*, pages 392–402.

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

Sameer S. Pradhan, Lance Ramshaw, Ralph Weischedel, Jessica MacBride, and Linnea Micciulla. 2007. Unrestricted coreference: Identifying entities and events in OntoNotes. In *Proceedings of the 2007 International Conference on Semantic Computing*, pages 446–453.

Stephen G. Pulman. 1997. Aspectual shift as type coercion. *Transactions of the Philological Society*, 95(2):279–317.

James Pustejovsky, José M. Castaño, Robert Ingria, Roser Sauri, Robert J. Gaizauskas, Andrea Setzer, and Graham Katz. 2003. TimeML: Robust specification of event and temporal expressions in text. In *Fifth International Workshop on Computational Semantics (IWCS-5)*, pages 28–34.

James Pustejovsky, Kiyong Lee, Harry Bunt, and Laurent Romary. 2010. ISO-TimeML: An international standard for semantic annotation. In *Proceedings of LREC*, pages 394–397.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of ACL*, pages 1375–1384.

Alan Ritter, Mausam, Oren Etzioni, and Sam Clark. 2012. Open domain event extraction from Twitter. In *Proceedings of SIGKDD*, pages 1104–1112.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of ACL*, pages 1192–1202.

Roser Saurí, Robert Knippen, Marc Verhagen, and James Pustejovsky. 2005. Evita: A robust event recognizer for QA systems. In *Proceedings of HLT/EMNLP*, pages 700–707.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A. Smith. 2014. Discriminative lexical semantic segmentation with gaps: Running the MWE gamut. *Transactions of the Association for Computational Linguistics*, 2:193–206.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394.

Tuangthong Wattarujeekrit, Parantu K. Shah, and Nigel Collier. 2004. PASBio: predicate-argument structures for event extraction in molecular biology. *BMC Bioinformatics*, 5:155.

Ralph Weischedel, Eduard Hovy, Mitchell Marcus, Martha Palmer, Robert Belvin, Sameer Pradhan, Lance Ramshaw, and Nianwen Xue. 2011. OntoNotes: A large training corpus for enhanced processing. In *Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitation*, pages 54–63. Springer.

Bishan Yang and Tom Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of NAACL-HLT*, pages 289–299.

Wenpeng Yin and Hinrich Schütze. 2015. Discriminative phrase embedding for paraphrase identification. In *Proceedings of NAACL-HLT*, pages 1368–1373.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of ACL: System Demonstrations*, pages 78–83.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of ACL*, pages 207–212.

Semi-Supervised Lexicon Learning for Wide-Coverage Semantic Parsing

Bo Chen[†‡], **Bo An**[†‡], **Le Sun**[†], **Xianpei Han**[†]
[†]State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences, Beijing, China
[‡]University of Chinese Academy of Sciences, Beijing, China
{chenbo,anbo,sunle,xianpei}@iscas.ac.cn

Abstract

Semantic parsers critically rely on accurate and high-coverage lexicons. However, traditional semantic parsers usually utilize annotated logical forms to learn the lexicon, which often suffer from the lexicon coverage problem. In this paper, we propose a graph-based semi-supervised learning framework that makes use of large text corpora and lexical resources. This framework first constructs a graph with a phrase similarity model learned by utilizing many text corpora and lexical resources. Next, graph propagation algorithm identifies the label distribution of unlabeled phrases from labeled ones. We evaluate our approach on two benchmarks: WEBQUESTIONS and FREE917. The results show that, in both datasets, our method achieves substantial improvement when comparing to the base system that does not utilize the learned lexicon, and gains competitive results when comparing to state-of-the-art systems.

Title and Abstract in Chinese

基于半监督词典学习的语义解析技术研究

语义解析器的性能往往依赖于词典的准确度和覆盖度。传统语义解析器利用标注好的句子-逻辑表达式对来学习词典，这通常会面临词典覆盖度不足的问题。本文提出了一种基于图的半监督学习框架，该框架能够充分利用容易获取的大量文本语料和词典资源来进行词典扩充学习。该词典扩充学习方法首先利用大量文本语料和词典资源来学习词语与词语之间的相似度，并构建用于图传播的图；接着使用图传播算法从少量标注的词汇中学习新的词汇。本文在两个公开数据集上进行了实验，实验结果表明：本文系统相比未使用新词汇的基准系统取得了显著提升，相比当前最好的系统，也取得了具有竞争力的结果。

1 Introduction

Semantic parsing aims to map natural language sentences into formal meaning representations, e.g., Figure 1 shows an example of semantic parsing. Semantic parsing plays an important role in natural language understanding, and has attracted increasing attention in recent years (Zelle and Mooney, 1996; Wong and Mooney, 2007; Lu et al., 2008; Liang et al., 2011; Kwiatkowski et al., 2011; Artzi and Zettlemoyer, 2013; Krishnamurthy and Mitchell, 2014; Li et al., 2015; Chen et al., 2016; Xiao et al., 2016; Jia and Liang, 2016; Reddy et al., 2016; Liang et al., 2017).

The performance of semantic parsers critically depends on the quality of lexicon, including accuracy and coverage. Specifically, in order to construct the logical form from a sentence, we first need to learn a lexicon,[1] which contains the mappings from natural language phrases (e.g., "*born*") to logical predicates (e.g., `PlaceOfBirth`). From the example in Figure 1, we can see that lexicon is the foundation of parsing, and lexicon learning plays an important role in semantic parsing.

Traditional semantic parsers are usually domain-specific, which only contains a limited number of logical predicates (Zettlemoyer and Collins, 2005; Kwiatkowksi et al., 2010; Artzi et al., 2014). In this

[1]We follow the lexicon style defined in Berant et al. (2013).

Proceedings of the 27th International Conference on Computational Linguistics, pages 892–904
Santa Fe, New Mexico, USA, August 20-26, 2018.

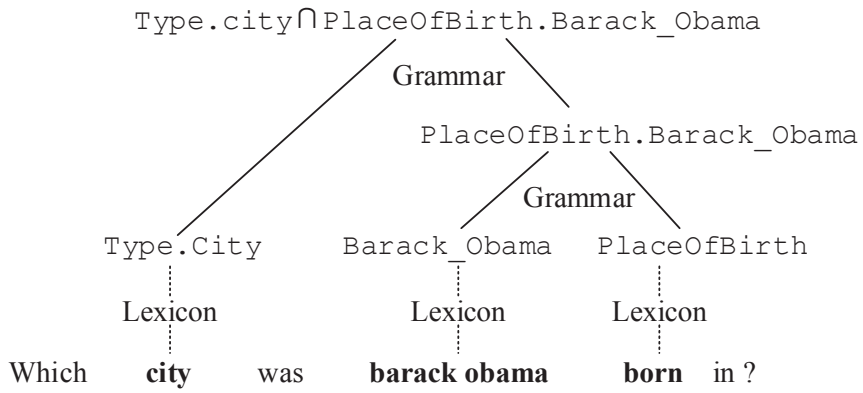


Figure 1: An example of semantic parsing, which uses lexicons to map phrases to predicates, and applies grammars to construct the logical form.

case, the mappings for each predicate can be learned relatively easily from training corpus. Recently, a growing body of research has scaled up semantic parsers to open domain (Cai and Yates, 2013a; Cai and Yates, 2013b; Berant et al., 2013; Krishnamurthy and Mitchell, 2012; Kwiatkowski et al., 2013) , where the number of predicates has increased substantially, making it hard to learn a lexicon with high coverage.

To resolve the lexicon coverage problem, there have been several papers on lexicon learning for semantic parsing. Cai and Yates (2013a) learns lexicons by pattern matching. Berant et al. (2013) learns lexicons by aligning Freebase [2] predicates with relations from ClueWeb [3], and then the alignments are used as lexicons. However, the lexicon coverage of these alignment-based methods highly depends on entity co-occurrences, and they mostly can only learn predicates which indicating relations between entities. It is still hard to cover all expressions and all predicates using alignment-based methods.

In this paper, we propose a semi-supervised lexicon learning algorithm for semantic parsing, which can increase the lexicon coverage by exploiting easily obtained text corpora and lexical resources. [4] The intuition behind our approach is that similar phrases should map to similar predicates, thus the phrase similarity can be used to propagate known predicate mappings to unknown mappings. For example, assuming we have a seed mapping: "*currency*" :: `currency`, and we know "*money*" is strongly related to "*currency*", we then can predict "*money*" should also map to `currency`. To achieve the above goal,we employ a graph-based semi-supervised learning framework, which learns lexicons not only used the alignments between Freebase and text, but also the semantic relatedness between phrases in the text side. Specifically, we use the abundant lexical resources for high coverage lexicon learning (Figure 2 shows the difference). There are three main tasks in this process. (1) we need a seed lexicon; (2) we need to measure the similarity between words; (3) we need to smooth the mappings to unlabeled words. For the similarity measure between words, we learn them from large text resources. This similarity plays two roles in our lexicon learning: (1) it is used for label propagation; (2) the similarity is used as a constraint on smoothing. That is, since we assume similar words will map to similar labels, the similarity can then strengthen the correct mapping and weaken the wrong mapping. Once we have seed lexicons and similarities between words, we smooth the lexicon by using a graph-based semi-supervised learning framework.

[2]https://en.wikipedia.org/wiki/Freebase

[3]https://www.lemurproject.org/clueweb12.php/

[4]We use *text corpora* to refer text resources from web, e.g., Wikipedia and WikiAnswers, and *lexical resources* to refer organized resources related to lexical items, e.g., WordNet and PPDB.

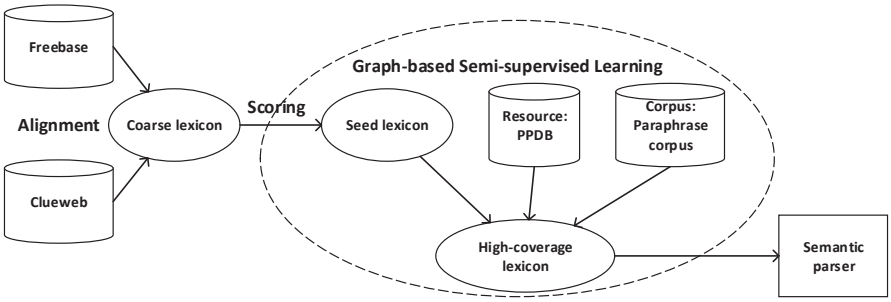


Figure 3: The framework of lexicon learning for semantic parsing in this paper. We can utilize large amount of text corpora and lexical resources to extend the lexicon for wide-open semantic parsing.

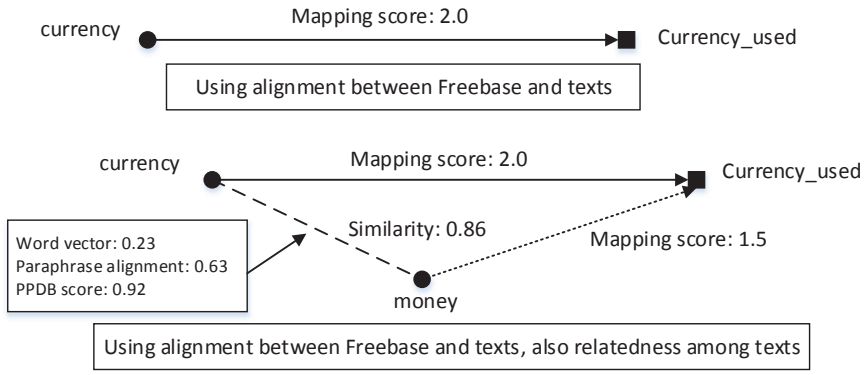


Figure 2: Example of our approach (the below one) and the previous approach (the above one). The previous approach only utilizes the alignments between text corpora and Freebase. By contrast, our approach further makes use of alignments between text and text. In this way, we can learn wide-coverage lexicon from several labeled lexicon.

The framework of our approach is shown in Figure 3. First, we make use of Freebase and ClueWeb to gain a coarse lexicon, and then we score these lexicons to gain the seed lexicon for following smooth. Next, we utilize easily obtained text corpora and text resources to learn the wide-coverage lexicon in a graph-based semi-supervised learning framework. Finally, we use the extended lexicon in the semantic parser to evaluate our approach.

We evaluate our lexicon learning algorithm on two benchmark datasets: WEBQUESTIONS and FREE917. The results show that our method can learn lexicon with higher coverage, and enhance the performance of semantic parsing system, especially its recall.

The contributions of this paper can be summarized as follows:

1. We propose a new semi-supervised learning framework for wide-coverage lexicon learning. Different to previous work, our approach can improve the lexicon coverage by further exploiting the easily obtained text corpora and lexical resources.

2. We design a graph-based learning algorithm to learn a wide-coverage lexicon from a seed lexicon.

3. We evaluate our approach on two benchmark datasets. Our system outperforms baseline systems significantly, and achieves competitive results with state-of-the-art systems.

2 Related Work

Lexicon learning is fundamental for semantic parsing. Traditional semantic parsers usually utilize annotated logical forms to learn the lexicon (Zettlemoyer and Collins, 2005; Kwiatkowksi et al., 2010; Kwiatkowski et al., 2011; Krishnamurthy and Mitchell, 2012; Berant et al., 2013; Krishnamurthy, 2016). Zettlemoyer and Collins (2005) utilizes the alignment between phrases in sentences and predicates in

annotated logical forms, and then assigned a confidence score to each lexical entry. Obviously, these approaches are limited by the annotated data.

Recently, many researchers begin to scale up semantic parsers to open domain. Learning high coverage lexicon in open domain requires large amount of annotated data, which is quite expensive even they only use question-answer pairs for supervision. There are several papers that focus on extending the lexicon for open-domain semantic parsing.

Cai and Yates (2013a) first extend the semantic parser to open domains. They utilize pattern matching to extend the lexicon. Specifically, they define knowledge patterns from knowledge bases, and text patterns from a search engine. Then they learn the lexicon based on the assumption that the phrase between two entities may map to the predicate if these two entities are also found under the predicate in knowledge bases. Berant et al. (2013) learn the lexicon by similar idea, but they use annotated ClueWeb corpus as the text side. Besides, they propose what they call a bridge operation, which is in fact a type-shifting which can bring in a predicate using minimum information. Compared to these approaches, our approach not only utilizes the alignments between knowledge bases and text corpora, but also makes use of text corpora and text resources to get the phrase similarity and phrase co-occurrence. In this way, we can learn more lexicon from little seed lexicon. Krishnamurthy (2016) also learned a lexicon for semantic parsing. However, they aim to extend the predicate side as they think the predicates have limited coverage for new sentences. Our aim is to extend the phrase that can trigger the predicates.

Graph-based semi-supervised learning algorithm has been used to resolve the OOV problem in machine translation (Razmara et al., 2013; Saluja et al., 2014; Zhao et al., 2015; Mehdizadeh Seraj et al., 2015). frame semantic parsing (Das and Smith, 2011), sentiment lexicon induction (Hamilton et al., 2016), and morph-syntactic lexicon induction (Faruqui et al., 2016).

3 Graph-based Lexicon Induction

Lexicon learning aims to learn the mapping from natural language phrases to predicates in knowledge base. There are three types of lexicons, including entity lexicon (e.g., "*city*" :: `Type.City`), unary lexicon (e.g., "*barack obama*" :: `Barack_Obama`) and binary lexicon (e.g., "*born*" :: `PlaceOfBirth`). In most cases entity lexicons are learned using entity linking techniques, therefore we usually only consider unary and binary lexicons in lexicon learning.

In open-domain semantic parsing, it is hard to learn high-coverage lexicon from annotated data for lexicon learning. In this paper, we use the (phrase, predicate) mappings as seeds, then learn new (phrase, predicate) mappings by propagating mapping information through similarities between phrases. Specifically, we propose a graph-based semi-supervised approach to resolve this problem. Our approach makes use of easily obtained text corpora and lexical resources to learn a similarity between words,[5] and then use a graph-based semi-supervised learning framework to smooth the lexicon graph. In this way, we can learn a new lexicon from labeled ones. Our method consists of three main steps:

1. Construct seed lexicon using alignments between Freebase and text corpora.
2. Learn similarities between words using both text corpora and text resources.
3. Learn new lexicon using label propagation.

3.1 Seed Lexicon Construction

We propagate information from seed lexicon to unknown ones. However, as we do not have enough annotated logical forms to learn the lexicon, and the number of predicates is too large, It is impossible to hand-make the lexicon, To obtain high-quality phrases mappings for each predicate, we construct the seed lexicon in two steps.

First, we gain a coarse lexicon by aligning Freebase with text corpora, using the lexicon learning methods as the same as Berant et al. (2013), with a slight difference that we only use unigram. We do this for two reasons: one is that a word (e.g., `born`) in a phrase (e.g., `born in`) can trigger a

[5]We use *phrase* and *word* alternately in this paper, since we use unigram for the lexicon and a *phrase* contains a single word.

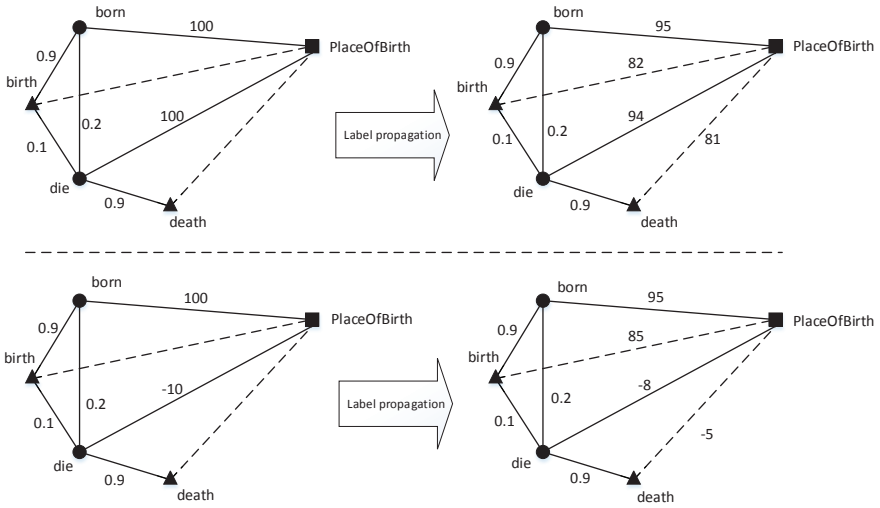


Figure 4: An example of propagation. This example shows that we need to score the seed lexicon first. Otherwise, the label propagation will bring more noise to lexicon learning.

predicate (e.g., *PlaceOfBirth*) if the phrase triggers the predicate. The other is that it is easy to compute the similarity between unigrams. Although this technique will raise more ambiguity, we use an extra feature template to handle this.

Next, we select high quality lexicons by scoring each lexical entry. Specifically, we use the lexicon gained in first step to train a semantic parser, and define several features to measure the quality of each lexical entry. After training, we can compute the score for each lexical entry. Since the higher the score of a lexical entry, the better its quality. We pick top K (K=5) lexical entries for each predicates as our seed lexicon. It is important to assign score to each lexical entry in the seed lexicon. As Figure 4 shows if we don't assign score, as there are many incorrect lexical entries in the seed lexicon, and these lexical entries will bring more noise to the lexicon when doing graph propagation.

3.2 Graph Construction

We construct a graph over all phrases in the seed lexicons and words which occur in the whole data. Besides, we also consider bridge words, which are both near to labeled node and unlabeled node. There are three types of nodes in the graph (As Figure 5 shows): Labeled nodes represent words in the seed lexicon; unlabeled nodes represent words in the whole data; bridge nodes represent the shared nearest neighbor nodes for the labeled nodes and unlabeled nodes.

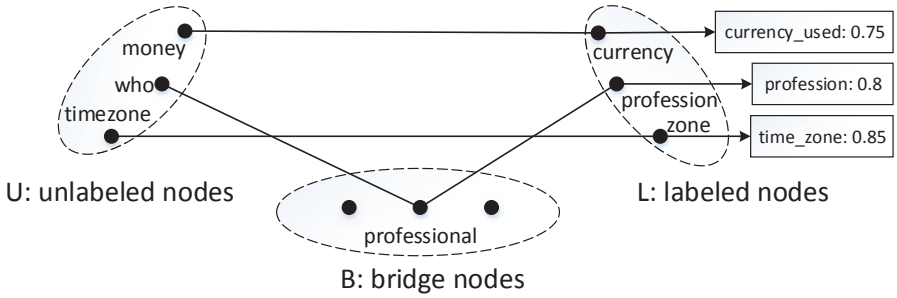


Figure 5: The graph between labeled nodes, unlabeled nodes and bridge nodes. Mapping can propagate either directly from labeled nodes to unlabeled nodes or indirectly via bridge nodes.

We use three resources to compute similarity for graph construction.

First, we use distributional representations for phrases to compute the similarity. Recently, a fair amount of research has showed that the word vector is quite useful for natural language process, especially for tasks related to similarity. Our purpose is to find similar words for the labeled ones, and label

neighbor words	scores
guilder	3.05
coin	3.03
taxa	3.00
les	2.89
exchange	2.85
monetary	2.76
money	2.63

Table 1: Nearest neighbor words for "*currency*" from PPDB-2.0.

them with the same labels. We use the published word vector (Huang et al., 2012) directly, and use cosine distance for similarity.

The second resource we used is paraphrase tables. In this part, we want to utilize paraphrase pairs, like "*money*" and "*currency*". We construct these pairs using the Paralex corpus (Fader et al., 2013). Paralex is a large question paraphrases corpus from WikiAnswers,[6] and each clique questions were tagged as expressing the same meaning by users. Paraphrase pairs in Paralex are word-aligned using standard machine translation methods. We use the word alignments to construct a word table by applying the consistent word pair heuristic to only unigram. This paraphrase tables is suitable for our needs since it focuses on question paraphrases.

The third resource we used is PPDB. Specifically, we use PPDB-2.0 (Pavlick et al., 2015) to calculate the similarity between two phrases by utilizing their scores, which consider many aspects. As we argued before, we only consider single word. So we use the lexical part of PPDB-2.0. Moreover, we pre-process the PPDB dataset by lemming the words. Table 1 shows several nearest neighbor words for the word "*currency*" from PPDB-2.0.

In fact, we can also use lexicon resources like synset from WordNet,[7] and Allen (2014) has used the VerbNet for learning a lexicon for broad-coverage semantic parsing. However, we find that the synsets for words are almost covered by the resources mentioned before. So we don't use these resources here.

In order to limit the graph size, we consider the top 10 nearest labeled nodes and top 5 nearest bridge nodes for each unlabeled one; for the each bridge node, we consider the top 5 nearest labeled nodes. Moreover, we also consider edges between labeled nodes and labeled nodes.

Finally, the overall similarity score between two given phrases w_1 and w_2 is computed as follow:

$$sim(w_1, w_2) = \alpha sim_1(w_1, w_2) + \beta sim_2(w_1, w_2) + (1 - \alpha - \beta) sim_3(w_1, w_2) \quad (1)$$

Before the final computing, we normalize each similarity score which are obtained using three resources separately. The hyperparameters are turned by development training.

3.3 Graph Propagation

Graph propagation is used to propagate the labels from labeled nodes to unlabeled ones by following the graph's structure. This approach is based on the smoothness assumption: similar nodes in the graph have similar labels. This paper utilizes the modified Adsorption algorithm (Talukdar and Crammer, 2009).

$$\min_{\widehat{Y}} \quad \mu_1 \sum_{v \in V_L} p_1 \, ||Y_v - \widehat{Y}_v||_2^2 \; + \mu_2 \sum_{v,u} p_2 \, W_{v,u} \, ||\widehat{Y}_v - \widehat{Y}_u||_2^2 \; + \mu_3 \sum_{v} p_3 \, ||\widehat{Y}_v - R_u||_2^2 \quad (2)$$

There are three parts in Formula (2), the first part enforces the labels of the seed nodes to keep unchanged. The second part enforces the smoothness, making similar nodes have similar labels. The third part enforces an uniform distribution for the unlabeled nodes. We use the Junto label propagation toolkit[8] for label propagation.

[6] http://www.answers.com/Q/

[7] https://wordnet.princeton.edu/

[8] https://github.com/parthatalukdar/junto

4 Semantic Parsing with Extended Lexicon

After graph propagation, each unlabeled phrase is labeled with a distribution over the set of predicates. We use SEMPRE (Berant et al., 2013; Berant and Liang, 2015) as our base semantic parser. In order to use the learned lexicon, we add a feature which indicates the final score for each lexical entry. The semantic parser will train on the training data with the learned lexicon as its initial lexicon. Following Berant and Liang (2015), we also use the feature template that conjoins predicates and content lemmas, and this feature template has been proved very helpful in Berant and Liang (2015).

5 Experiments

We evaluate our method on two benchmark datasets: WEBQUESTIONS and FREE917.

Dataset: WEBQUESTIONS dataset (Berant et al., 2013) contains 5,810 question-answer pairs. These questions are collected by crawling the Google Suggest API, and the answers are obtained using Amazon Mechanical Turk. This dataset covers several popular topics and its questions are commonly asked on the web. In our experiments, we use the standard train-test split (Berant et al., 2013), i.e., 3,778 questions (65%) for training and 2,032 questions (35%) for testing.

The FREE917 dataset (Cai and Yates, 2013a) contains 917 questions, annotated with logical forms. This dataset covers a wide range of domains. One example is "*what fuel does an internal combustion engine use*". Following Cai and Yates (2013a), we use the original split of the questions into 70% questions (641) to train and 30% questions (276) to test.

Setup: In our experiments, we use the Freebase Search API for entity lookup in WEBQUESTIONS dataset, and build a Lucene index over the 41M Freebase entities to map entities in FREE917 dataset. We load Freebase using Virtuoso, and execute logical forms by converting them to SPARQL and querying using Virtuoso. We learn the parameters of our system by making several passes (3 for WEBQUESTIONS and 6 for FREE917) over the training dataset, with the beam size (200 in WEBQUESTIONS and 500 for FREE917).

For the similarity computation, we set $\alpha = 0.05$, $\beta = 0.85$. For the parameters in Junto, we set $\mu_1 = 0.55$, $\mu_2 = 0.44$, $\mu_3 = 0.01$, $\beta = 2$.

Comparing systems: To evaluate our method, we mainly compare our system (Base + lexicon) to the base system (Base) which does not use the learned wide-coverage lexicon, also to system (Base + bridge) which utilize bridge operator to serve as lexicon (Berant et al., 2013). We also compare to several nearly published systems, including semantic parsing based system (Kwiatkowski et al., 2013; Berant and Liang, 2015), information extraction based systems (Yao and Van Durme, 2014; Yao, 2015), machine translation based systems (Bao et al., 2014), embedding based systems (Bordes et al., 2014; Yang et al., 2014), and QA based system (Bast and Haussmann, 2015).

5.1 Experimental Results

Table 2 and Table 3 provide the performances of all baselines[9] and our method in WEBQUESTIONS and FREE917. From Table 2 and Table 3, we can see that:

1. Our method achieves competitive performance: Our system outperforms base system (Base) greatly and gets a better performance when comparing to the base system with a bridge operator (Base + bridge).

2. The learned lexicon has wider coverage than the seed one: Our system obtains higher recall than the Base. By utilizing large amount of text corpora and lexical resources, the extended lexicon improves the semantic parsing system. For FREE917, our system gains the highest recall. This indicates that our lexicon really has wider coverage, especially for dataset with more domains like FREE917.

3. The bridge operation from Berant and Liang (2015) is quite powerful. It can resolve the problem of lexicon coverage to some degree. And our approach, which learns the lexicon directly, can gain a better performance.

[9]We collect the results of other systems from https://nlp.stanford.edu/software/sempre/.

System	Prec.	Rec.	F1 (avg)
Berant et al. (2013)	48.0	41.3	35.7
Yao and Van Durme (2014)	51.7	45.8	33.0
Berant and Liang (2014)	40.5	46.6	39.9
Bao et al. (2014)	–	–	37.5
Bordes et al. (2014)	–	–	39.2
Yang et al. (2014)	–	–	41.3
Bast and Haussmann (2015)	49.8	60.4	49.4
Yao (2015)	52.6	54.5	44.3
Berant and Liang (2015)	50.5	55.7	49.7
Yih et al. (2015)	52.8	60.7	52.5
Base	51.0	47.6	40.5
Base + bridge	50.0	58.5	50.0
Our approach	51.6	59.7	51.2

Table 2: The results of our system and recently published systems on WEBQUESTIONS.

System	Prec.	Rec.	F1
Cai and Yates (2013a)	67.0	59.0	63.0
Kwiatkowski et al. (2013)	76.7	68.0	72.1
Bast and Haussmann (2015)	72.0	67.8	69.8
Base	71.2	59.5	64.8
Base + bridge	69.4	64.4	66.8
Our approach	71.5	67.9	69.6

Table 3: The results of our system and recently published systems on FREE917.

4. Compared to all baselines, our system gets a competitive recall. This result indicates that our parser can parse more sentences when the lexicon has wider coverage. Interestingly, for WEBQUESTIONS, both the two systems with the highest recall (Bast and Haussmann, 2015; Yih et al., 2015) rely on extra-techniques such as entity linking and relation matching.

In Section 3.1, we normalize the seed lexicon using the unigram for the lexeme. By this way, the final graph for label propagation will not be too large, and it is convenient to compute the similarity between word and word using text corpora and lexical resources. We design some experiments to evaluate the new seed lexicon (unigram for lexeme). Table 4 and Table 5 shows the results. We can see that the system using the new seed lexicon has similar performance to the system using the original seed lexicon.

System	Prec.	Rec.	F1 (avg)
Original seed	40.6	47.5	**40.6**
New seed	**51.0**	**47.6**	40.5

Table 4: The results of using different seed lexicons on WEBQUESTIONS dataset.

System	Prec.	Rec.	F1 (avg)
Original seed	69.8	59.0	63.9
New seed	**71.2**	**59.5**	**64.8**

Table 5: The results of using different seed lexicons on FREE917 dataset.

System	Prec.	Rec.	F1 (avg)
Only word vector	41.8	45.9	39.7
Only paraphrase table	50.0	46.3	39.4
Only PPDB-2.0	51.2	58.7	50.6
All	**51.6**	**59.7**	**51.2**

Table 6: The results of using different resources for measuring similarities on WEBQUESTIONS dataset.

To evaluate the text corpora and lexical resources we used, we also conduct several experiments on WEBQUESTIONS. Table 6 shows the results. We can see that:

1. Only using word vector for similarity computation, the final result is not ideal. We think that the word vector consider many aspects in similarity, and in lexicon learning, what we expect for similarity is paraphrasing.

2. The paraphrase table pairs help a little. We think that is due to we use simple alignment for scoring.

3. The PPDB-2.0 serves quite well, even only use PPDB-2.0 score for similarity computation. We think that the PPDB-2.0 was extracted from paraphrase corpus, so the similar lexicon are almost paraphrase to each other.

4. Using word vector, paraphrase align table pairs and PPDB-2.0 score together achieves the best performance.

5.2 Analysis

Our aim is to learn a wide-coverage lexicon for semantic parsing. Our approach utilizes text corpora and lexical resources to extend seed lexicon. Table 7 shows several learned new mappings with the final score from the semantic parser. We can find that after label propagation, we can obtain new lexical entries which can improve the coverage of semantic parser. The results proved our intuition, i.e., the unlabeled phrase maps to the same predicate of its nearest labeled phrases.

Predicate	Seed phrase	Learned phrase	score
`Currency`	*currency*	*money*	4.91
`Education`	*education*	*school*	4.75
`Religion`	*religion*	*believe*	2.30
`Profession`	*professional*	*who*	2.30

Table 7: Several learned lexical entries with un-normalized scores on WEBQUESTIONS dataset.

The learned new lexicon has wide coverage, however, this means the accuracy for the lexicon will be influenced. Berant and Liang (2015) added new lexicalized features (lemmaAndBinary) that connect natural language phrases to binary predicates. For example, given the utterance "*What countries have german as the official language?*", the predicate for phrase "*language*" can be `Language-spoken` and `Offical-language`. The added feature will conjoin binaries with all content word lemmas. After observing enough examples, the phrase "*language*" will map to `Offical-language` when has "*official*" as its content, because the feature, which corresponds to "*offical*" and `Offical-language`, will be up-weighted. Berant and Liang (2015) have proved this feature is really helpful. In our experiments, we also use this feature. Table 8 shows the ablation test results. We can see that, this feature improves our system greatly, especially for precision. We think that as we only use the unigram as our lexeme for lexical entry, the lexicon has more noise. And the alignments between predicates with content words will help the parser to choose the right lexicon during parsing.

System	Prec.	Rec.	F1 (avg)
Our system - feature	48.0	48.9	41.8
Our system	**51.6**	**59.7**	**51.2**

Table 8: The results of ablation test for the lemmaAndBinary feature.

Manual error analysis To better understand our system, we manually inspected the errors our system made. We found that many errors are due to mistakes in labeling. The rest of the errors are mainly complicate cases, like N-ary predicate (event in Freebase), superlative, temporal clause etc. We argue that more attention should be given to these complicated cases.

6 Conclusion

In this paper, we make use of low-cost, easily obtained text corpora and lexical resources in a graph-based semi-supervised learning framework to learn lexicon for semantic parsing. Experiments demonstrate that our method improves the semantic parsing system, especially, when the lexicon is not covered in the training data. Our method can learn wide-coverage lexicon for open-domain semantic parsing.

Traditionally, a semantic parser needs a lexicon first, and then parses the sentence in a bottom-up way. For these parsers, the lexicon is extremely important, and it is hard to learn lexicon with high coverage. Currently, some semantic parsers use the knowledge base in advance, and utilize entity linking and relation matching during parsing, and these methods parse the sentence like a top-down way. As the knowledge base is huge, the searching space is usually quite large. In future work, We want to design parsing algorithm which can take advantages from both sides.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants no. 61433015, 61572477 and 61772505, and the Young Elite Scientists Sponsorship Program no. YESS20160177. Moreover, we sincerely thank the reviewers for their valuable comments.

References

James Allen. 2014. Learning a lexicon for broad-coverage semantic parsing. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 1–6, Baltimore, MD, June. Association for Computational Linguistics.

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

Yoav Artzi, Dipanjan Das, and Slav Petrov. 2014. Learning compact lexicons for ccg semantic parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1273–1283, Doha, Qatar, October. Association for Computational Linguistics.

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 967–976, Baltimore, Maryland, June. Association for Computational Linguistics.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, pages 1431–1440.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, Baltimore, Maryland, June. Association for Computational Linguistics.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October. Association for Computational Linguistics.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620, Doha, Qatar, October. Association for Computational Linguistics.

Qingqing Cai and Alexander Yates. 2013a. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 423–433, Sofia, Bulgaria, August. Association for Computational Linguistics.

Qingqing Cai and Alexander Yates. 2013b. Semantic parsing freebase: Towards open-domain semantic parsing. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 328–338, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Bo Chen, Le Sun, Xianpei Han, and Bo An. 2016. Sentence rewriting for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 766–777, Berlin, Germany, August. Association for Computational Linguistics.

Dipanjan Das and Noah A. Smith. 2011. Semi-supervised frame-semantic parsing for unknown predicates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1435–1444, Portland, Oregon, USA, June. Association for Computational Linguistics.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1608–1618, Sofia, Bulgaria, August. Association for Computational Linguistics.

Manaal Faruqui, Ryan McDonald, and Radu Soricut. 2016. Morpho-syntactic lexicon generation using graph-based semi-supervised learning. *Transactions of the Association for Computational Linguistics*, 4:1–16.

William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 595–605, Austin, Texas, November. Association for Computational Linguistics.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–882, Jeju Island, Korea, July. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany, August. Association for Computational Linguistics.

Jayant Krishnamurthy and Tom Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 754–765, Jeju Island, Korea, July. Association for Computational Linguistics.

Jayant Krishnamurthy and Tom M. Mitchell. 2014. Joint syntactic and semantic parsing with combinatory categorial grammar. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1188–1198, Baltimore, Maryland, June. Association for Computational Linguistics.

Jayant Krishnamurthy. 2016. Probabilistic models for learning a semantic parser lexicon. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 606–616, San Diego, California, June. Association for Computational Linguistics.

Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233, Cambridge, MA, October. Association for Computational Linguistics.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1512–1523, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556, Seattle, Washington, USA, October. Association for Computational Linguistics.

Junhui Li, Muhua Zhu, Wei Lu, and Guodong Zhou. 2015. Improving semantic parsing with enriched synchronous context-free grammar. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1455–1465, Lisbon, Portugal, September. Association for Computational Linguistics.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 590–599, Portland, Oregon, USA, June. Association for Computational Linguistics.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–33, Vancouver, Canada, July. Association for Computational Linguistics.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 783–792, Honolulu, Hawaii, October. Association for Computational Linguistics.

Ramtin Mehdizadeh Seraj, Maryam Siahbani, and Anoop Sarkar. 2015. Improving statistical machine translation with a multilingual paraphrase database. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1379–1390, Lisbon, Portugal, September. Association for Computational Linguistics.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430, Beijing, China, July. Association for Computational Linguistics.

Majid Razmara, Maryam Siahbani, Reza Haffari, and Anoop Sarkar. 2013. Graph propagation for paraphrasing out-of-vocabulary words in statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1115, Sofia, Bulgaria, August. Association for Computational Linguistics.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Avneesh Saluja, Hany Hassan, Kristina Toutanova, and Chris Quirk. 2014. Graph-based semi-supervised learning of translation models from monolingual data. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 676–686, Baltimore, Maryland, June. Association for Computational Linguistics.

Partha Pratim Talukdar and Koby Crammer. 2009. New regularized algorithms for transductive learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 442–457. Springer.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967, Prague, Czech Republic, June. Association for Computational Linguistics.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1341–1350, Berlin, Germany, August. Association for Computational Linguistics.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 645–650, Doha, Qatar, October. Association for Computational Linguistics.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 956–966, Baltimore, Maryland, June. Association for Computational Linguistics.

Xuchen Yao. 2015. Lean question answering over freebase from scratch. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 66–70, Denver, Colorado, June. Association for Computational Linguistics.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China, July. Association for Computational Linguistics.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*, pages 1050–1055, Portland, OR, August. AAAI Press/MIT Press.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 658–666.

Kai Zhao, Hany Hassan, and Michael Auli. 2015. Learning translation models from monolingual continuous representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1527–1536, Denver, Colorado, May–June. Association for Computational Linguistics.

Summarization Evaluation in the Absence of Human Model Summaries Using the Compositionality of Word Embeddings

Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond Wong, Fang Chen
University of New South Wales, Sydney, Australia
Data61 CSIRO, Sydney, Australia
{elahehs,mohammade,wong,fang}@cse.unsw.edu.au

Abstract

We present a new summary evaluation approach that does not require human model summaries. Our approach exploits the compositional capabilities of corpus-based and lexical resource-based word embeddings to develop the features reflecting coverage, diversity, informativeness, and coherence of summaries. The features are then used to train a learning model for predicting the summary content quality in the absence of gold models. We evaluate the proposed metric in replicating the human assigned scores for summarization systems and summaries on data from query-focused and update summarization tasks in TAC 2008 and 2009. The results show that our feature combination provides reliable estimates of summary content quality when model summaries are not available.

1 Introduction

Quantifying the quality of summaries is an important and necessary task in the field of automatic text summarization. Current summary evaluation methods like manual and automated pyramid (Passonneau et al., 2005; Passonneau et al., 2013) and well-established ROUGE scores (Lin, 2004) heavily rely on multiple human-generated model summaries to assess the quality of system-generated summaries. This evaluation paradigm falls short on non-standard test sets where model summaries are not available. According to the quantitative analysis by Louis and Nenkova (2009a); Singh and Jin (2016), evaluating summaries by their comparison with the input obtains good correlations with manual evaluations. Therefore, identifying a suitable input-summary similarity metric will provide a means for model-free evaluation of summaries.

We hypothesize that comparing semantic representations of the input and summary content will lead to more accurate input-summary evaluation. Hence, we explore the effectiveness of compositionality of word embeddings in developing a model-free automatic metric to evaluate summary content quality. In particular, our approach incorporates the word embedding models trained on the Google News corpus and the WordNet lexical resource to compare centroid vectors of the input and summary. To demonstrate the effectiveness of our approach, we have conducted a set of experiments on data from query-focused and update summarization tasks in TAC[1] 2008 and 2009. The reliability of our metric is also studied conducting an error analysis. The experiment results show that quantifying the indicators of content quality by taking advantage of compositional properties of the word and sense embeddings produces summary scores which accurately replicate human assessments. It is noteworthy that our approach complements but is not intended to replace existing model-based evaluation approaches, since their reliability and strength are important for high confidence evaluations.

2 Related Work

Proposals for developing automatic summary evaluation methods (Ellouze et al., 2013; Ng and Abrecht, 2015; ShafieiBavani et al., 2017) have been put forward in the past. However, these methods are not

[1] http://www.nist.gov/tac/

Proceedings of the 27th International Conference on Computational Linguistics, pages 905–914
Santa Fe, New Mexico, USA, August 20-26, 2018.

applicable on non-standard test sets where model summaries are not available. Herein, we try to briefly review the most significant approaches that have addressed this issue. Donaway et al. (2000) proposed an alternative to model-based evaluation where a comparison of the input text with a summary can clarify how good the summary is. A summary that has higher similarity with the input text can be considered better than one with lower similarity. Radev et al. (2003) performed an automated ranking of the test documents using a search engine scenario. Their approach was motivated by the assumption that the distribution of terms in a good summary is similar to the distribution of terms in the input document.

With the same intuition, Louis and Nenkova (2009a; 2013) introduced an evaluation system (SIMetrix) that comprises multiple features to determine the quality of a summary. Their focus was on computing divergences between the probability distributions of words in the input and summary. Jensen Shannon divergence and feature regression turned out to be their best metrics. Louis and Nenkova (2009b) also presented a similar evaluation approach utilizing a collection of large number of system summaries in place of model summaries. Saggion et al. (2010); Cabrera-Diego and Torres-Moreno (2017) proposed follow-up works to SIMetrix to assess the usefulness of divergences for multilingual summarization evaluation, and the applicability of multiple divergences for evaluating summaries.

Alternatively, we assume that the way of representing the input and summary is a key factor in high performance prediction of manual metrics. To this end, we exploit the compositional capabilities of word embeddings to design our features of content quality based on the continuous vector representation of words and senses. We then present a quantitative analysis of our features for characterizing the relation between the input and summary content in the absence of model summaries. We have finally evaluated our approach through two levels of granularity on two years data in TAC for query-focused and update summarization tasks. We have also compared our approach with model-based ROUGE, and model-free SIMetrix as an input-summary evaluation metric. The results demonstrate that the Support Vector Regression (SVR) of our features achieves the best correlation with manual judgments.

3 Data and Evaluation Metrics

We carry out our experiments on the query-focused and update summarization tasks from TAC 2009 with 44 inputs as our test set, and from TAC 2008 with 48 inputs as the development set. These datasets consist of two sets of 10 news documents for each input: (i) set *A* for initial summaries; (ii) set *B* for update summaries. Both *A* and *B* are on the same general topic but *B* contains documents published later than those in *A*. The update summary of set *B* is created assuming that the user is aware of what exists in set *A*. There are also four human-crafted model summaries for each input in each document set. A maximum of 100 words summary that addresses the information required by the given query statement (consisting of a title and narrative) has been produced by each of the 53 and 58 automatic summarizers participated in TAC 2009 and 2008, respectively. An example query statement is shown here:

Title: Barack Obama
Narrative: Track the increase in Barack Obama's popularity, visibility, support, and activities.

Content and linguistic quality are two conventional factors in evaluation of summary quality. Herein, we focus on the problem of automatic evaluation of content quality. Hence, we assess the performance of our metrics in replicating manual correlations of pyramid and responsiveness. It is noteworthy that responsiveness incorporates at least some aspects of linguistic quality.

Pyramid: This evaluation method (Passonneau et al., 2005) is a content assessment measure which compares content units in a system summary to weighted content units in a set of model summaries. It uses multiple human models from which annotators identify semantically defined Summary Content Units (SCU). Each SCU is assigned a weight equal to the number of human model summaries that express that SCU. An ideal maximally informative summary would express a subset of the most highly weighted SCUs, with multiple maximally informative summaries being possible. The pyramid score for a system summary is equal to the ratio between the sum of weights of SCUs expressed in a summary (again identified manually) and the sum of weights of an ideal summary with the same number of SCUs.

Four human summaries provided by NIST for each input and task were used for the pyramid evaluation at TAC.

Responsiveness: This is a measure of overall quality combining both content and linguistic quality. Summaries must present useful content in a structured fashion in order to better satisfy the user's need. Assessors directly assigned scores on a scale of 1 (poor) to 5 (very good) to each summary. These assessments are done without reference to any model summaries.

Linguistic Quality: This measure ranks summaries in a 5-point scale indicating how well a summary satisfied the factors of linguistic quality (i.e., grammaticality, non-redundancy, referential clarity, focus, structure and coherence). In our work, we do not evaluate linguistic quality.

4 Features for Summary Evaluation

We propose five classes of features to assess the quality of summary content in the absence of model summaries: (i) *Distributional Semantic Similarity*; (ii) *Topical Relevance*; (iii) *Query Relevance*; (iv) *Coherence*; and (v) *Novelty*. Before computing the features, all words in input documents, summaries, and queries are converted to lower case and stop-word filtered. We experiment with two variants of word embeddings as the basic building block to design our features:

Corpus-based Word Embeddings: We utilize the 300-dimensional embeddings for 3M words and phrases trained on Google News[2], a corpus of $\sim 10^{11}$ tokens, using word2vec CBOW (Mikolov et al., 2013). Word2vec learns a vector representation for each word using a neural network language model. It also allows to learn complex semantic relationships using simple vectorial operators, such as $vec(king) - vec(man) + vec(woman) \approx vec(queen)$. Stemming is not performed to make the word embeddings discover the linguistic regularities of words with the same root.

Lexical Resource-based Word Embeddings: We use WordNet (Fellbaum, 1998) to measure the lexico-semantic similarity between the input and its summary. Since the constraints of WordNet lexical resource can be formalized as constraints on embeddings, we can use embeddings of non-word data types (i.e., senses). Specifically, we compute the embedding of a word by averaging the embeddings of its senses in WordNet. For example, the vector of the word *suit* is modeled as the average of a vector representing *lawsuit* and a vector representing *business suit*.

We obtain the sense embeddings using the pre-trained model by Rothe and Schütze (2015), that lives in the same vector space as the pre-trained word2vec by Mikolov et al. (2013). Their model is an autoencoder neural-network that takes word embeddings and learns sense embeddings based on the following intuitions: (i) a word's embedding is the sum of the embeddings of its senses; and (ii) the senses related by WordNet relations (e.g., hypernymy, antonymy, similarity) have similar embeddings. Considering WordNet relations also helps to compute embeddings for senses in WordNet which are not in the word2vec vocabulary.

We further assume that the probability of a word sense is in proportion to its frequency in WordNet. Hence, the probability that a sense $\mathcal{S}_{ij}$ is the meaning of the word w_i, is the ratio of the frequency of that sense $freq(\mathcal{S}_{ij})$ to the total frequency of the word. If the frequency of a word sense is 0 in WordNet, we set it to 1. Finally, the embedding of word w_i is computed[3] as a weighted average of its senses $\mathcal{S}_{ij}, 1 \leq j \leq n$, where the weights represent the probability of senses:

$$\vec{w_i} = \frac{\sum_{\mathcal{S}_{ij} \in \mathcal{S}yn(w_i)} freq(\mathcal{S}_{ij}) \times \vec{\mathcal{S}_{ij}}}{n \sum_{\mathcal{S}_{ij} \in \mathcal{S}yn(w_i)} freq(\mathcal{S}_{ij})} \quad (1)$$

4.1 Distributional Semantic Similarity

A good summary must satisfy both *coverage* and *diversity* properties. For clarity, summary sentences should cover a sufficient non-redundant amount of information from the original input text. Diversity

[2]https://code.google.com/p/word2vec/

[3]Words were stemmed before inferring their embeddings.

property is also fundamental especially for multi-document summarization. Moreover, one would expect good summaries to be characterized by low distance between probability distributions of words in the input and summary, and by high similarity with the input. Hence, we design this feature based on the geometric meaning of the centroid vector of a document using the compositional properties of the word embeddings (Mikolov et al., 2013). The main idea is to give a distributed representation of words/senses in the input and its summary, and compare their centroid vectors to realize how much the summary content works as a pseudo-input and condenses the meaningful information of the input.

The centroid embedding $\vec{T}$ of a text $T = \{t_1, t_2, ..., t_n\}$, is the sum of the embeddings of tokens of T divided by the number of tokens n. Based on the problem, we can also assign a weight $\mathcal{W}$ to each token in T (Figure 1). Accordingly, the centroid embedding for each summary sentence $\vec{s_j}$ is computed by averaging the embeddings of all words comprising the sentence (Radev et al., 2004). Similarly, we construct a centroid vector for each document, $\vec{d_i}$, in the input document set. To better assess the *informativeness* of the summary content, we assign higher weights to specialized words in a document by considering the Inverse Document Frequency (IDF) scores of words:

$$\vec{d_i} = \frac{\sum_{w_j \in d_i} \vec{w_j} \times TF(w_j, d_i) \times IDF(w_j)}{n \sum_{w_j \in d_i} TF(w_j, d_i) \times IDF(w_j)} \tag{2}$$

where n is the number of words in document d_i, and $\vec{w_j}$ is the embedding of word w_j. $TF(w_j, d_i)$ stands for the term frequency of w_j in d_i. The IDF scores are computed on the whole document set.

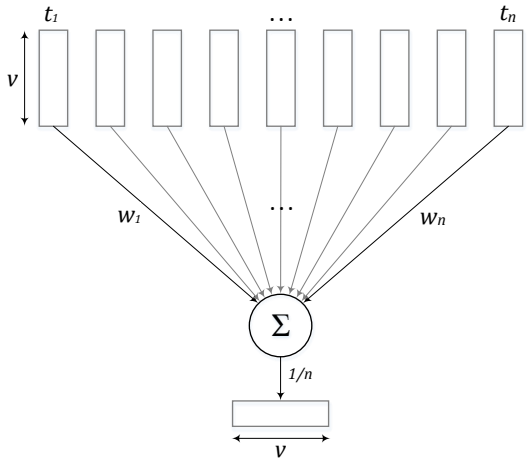


Figure 1: The weighted centroid embedding of text $T = \{t_1, t_2, ..., t_n\}$

Finally, we compare summary sentences and the input documents using the Word Mover's Distance (WMD) algorithm (Kusner et al., 2015). WMD measures the total distance the centroid embeddings of summary sentences and the input documents have to travel to become identical. Accordingly, we measure the dissimilarity degree between two sets of embedding vectors, $D = \{\vec{d_1}, ..., \vec{d_n}\}$ and $S = \{\vec{s}_1, ..., \vec{s}_m\}$, by calculating the minimum amount of summing up individual distances (travel costs) that centroid embeddings of the documents in D need to travel to reach the embeddings of sentences in S:

$$WMD(D, S) = \min_{F \geq 0} \sum_{\vec{d_i} \in D} \sum_{\vec{s_j} \in S} F_{\vec{d_i}\vec{s_j}} \times dist(\vec{d_i}, \vec{s_j}) \tag{3}$$

subject to,

$$\sum_{\vec{d_i} \in D} F_{\vec{d_i}\vec{s_j}} = \frac{1}{|S|}, \forall \vec{s_j} \in S, \sum_{\vec{s_j} \in S} F_{\vec{d_i}\vec{s_j}} = \frac{1}{|D|}, \forall \vec{d_i} \in D$$

where $F \in \mathbb{R}^{V \times V}$ with V as the vocabulary size, is a flow matrix which indicates how much probability mass should flow (or travel) from document centroid embedding $\vec{d_i}$ in set D to sentence embedding $\vec{s_j}$

in set S, and vice versa. $dist(\vec{d_i}, \vec{s_j})$ denotes the individual distance (or travel cost) between $\vec{d_i}$ and $\vec{s_j}$: $dist(\vec{d_i}, \vec{s_j}) = \|\vec{d_i} - \vec{s_j}\|_2$.

4.2 Topical Relevance

Topic features serve as a basis for evaluating topical relevance of a summary to the input documents. Herein, we aim to find the distribution of the most probable topics embodied in the input document set, and their relevance to the summary sentences. To this end, we use Latent Dirichlet Allocation (LDA) algorithm (Blei et al., 2003; Arora and Ravindran, 2008) to determine the topics that characterize every document set. LDA is a generative model for documents to determine topic compositions of words and document mixtures of topics (represented by a probability distribution over topics), by assigning words to topics within documents. Hence, in the context of text modeling, the topic distribution provides an underlying semantic representation of the documents and can be useful in evaluating the summaries. Using weighted topic compositions, we measure the similarity of summary sentences with the most important topics identified in the document set.

We use Gibbs sampling (Griffiths, 2002) for inference in the topic model with concentration parameters $\alpha = 0.1$ and $\beta = 0.01$. We also set the number of topics $K = 10$ for each document set. Formally, each topic is defined as $\mathcal{T}_i = \{p_1, p_2, ..., p_n\}$, where p_j is the probability distribution of word w_j. We consider top $m = 30$ words and their probabilities to build a centroid as the representative of each topic. The embedding vector for word w_j is then multiplied with its normalized probability $\mathcal{P}_j$, and the weighted vectors are averaged to build a topic centroid representation:

$$\vec{\mathcal{T}_i} = \frac{1}{m} \sum_{j=1}^{m} \mathcal{P}_j \vec{w_j}, \quad where\ \mathcal{P}_j = \frac{p_j}{\sum_{i=1}^{m} p_i} \tag{4}$$

Finally, we use WMD to measure the dissimilarity degree between the centroid embeddings of summary sentences and those of the topics for evaluating topical relevance of the summary content.

4.3 Query Relevance

To measure the relevance degree of the summary content to the given query, we calculate the query embedding vector $\vec{Q}$ by averaging the embeddings of all words in the query narrative. Similarly, the centroid embedding vector for each summary $\vec{S}$ is also constructed. We further measure the cosine similarity between these vectors to formulate query relevance:

$$sim(\vec{S}, \vec{Q}) = \frac{\vec{S} \cdot \vec{Q}}{\| \vec{S} \| \| \vec{Q} \|} \tag{5}$$

4.4 Coherence

Coherence measures the degree to which a sequence of summary sentences represents a logical flow of thought. We compute the similarity between embeddings of adjacent summary sentences using cosine similarity. It results in $n-1$ comparisons for a summary of n sentences. While similarity between sentences is beneficial for coherence, very high similarity reflects redundancy in the summary. Given that, we combine the *mean* and *standard deviation* of the cosine similarity scores by training a simple linear regression model on our development set. In this way, we measure the trade-off between continuity and redundancy as the coherence feature.

4.5 Novelty

We would like our evaluation model to move beyond assessing initial summaries by giving a simple feature of Novelty to better evaluate update summaries. This feature rewards the update summary consisting novel words that do not exist in initial document set D_A, but are semantically related to update document set D_B. The relevancy of these words in update summary S_j, to the documents in set *B*, is measured using the cosine similarity between the embeddings of novel words and the centroid embedding of the

whole document set *B*. We use the bag-of-words representation of the summary and the document sets while defining novel words. We finally measure the degree of novelty ($\mathcal{N}$) as:

$$\mathcal{N}(S_j) = \frac{1}{|S_j|} \sum_{w_i \in S_j | w_i \notin D_A} sim(\vec{w_i}, \vec{D_B}) \tag{6}$$

where $|S_j|$ is the total number of unique words in the update summary S_j. For S_j without any novel words, $\mathcal{N}(S_j) = 0$.

5 Feature Combination with SVR

We combine all the above features using a Support Vector Regression (SVR) model to predict the summary quality. We first transform the proposed features into a standard vector notation. Each summary S_i is represented by a feature vector $X = \{x_1, x_2, ..., x_n\}$ where n is the number of features. SVR model aims to learn a function $f : \mathbb{R}^n \mapsto \mathbb{R}$, which will be used to predict the content evaluation score for each summary $y \in \mathbb{R}$ given a feature vector $X \in \mathbb{R}^n$. In particular, given l training instances $(X_1, y_1), ..., (X_l, y_l)$, the SVR model is learnt by solving the following optimization problem (Vapnik, 1999); W is a vector of feature weights; ϕ is a function that maps feature vectors to a new vector space of higher dimensionality to allow non-linear functions to be learnt in the original space; $C > 0$ and $\epsilon > 0$ are given.

$$\min_{W,b,\xi,\xi^*} \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{l} \xi_i + C\sum_{i=1}^{l} \xi^* \tag{7}$$

subject to (for $i = 1, ..., l$):

$$\begin{aligned} W^T \cdot \phi(X_i) + w_0 - y_i &\leq \epsilon + \xi_i \\ y_i - W^T \cdot \phi(X_i) - w_0 &\leq \epsilon + \xi^* \\ \xi_i &\geq 0 \\ \xi^* &\geq 0 \end{aligned}$$

The goal is to learn a linear (in the new space) function, whose prediction (value) $W^T \cdot \phi(X_i) + w_0$ for each training instance X_i will not to be farther than ϵ from the target (correct) value y_i. Since this is not always feasible, two slack variables ξ_i and ξ_i^* are used to measure the prediction's error above or below the target y_i. The objective (7) jointly minimizes the total prediction error and $\|W\|$, to avoid overfitting. The utilized SVR is implemented in Scikit-learn (Pedregosa et al., 2011). We use the default parameter settings, (*kernel='rbf', degree=3, gamma='auto', coef0=0.0, tol=0.001, C=1.0, epsilon=0.1, shrinking=True, cache_size=200, verbose=False, max_iter=-1*) without further optimization.

6 Experiments and Results

Reporting correlations with manual evaluation metrics is the norm for validating automatic metrics. We use Spearman correlation metric to study the predictive power of our automatic features in replicating manual correlations of pyramid and responsiveness. Hence, we compare the rankings of systems against the human scores assigned to systems. The correlations[4] of our metrics are reported at two levels of granularity:

System Level (MACRO): The average score for a system is computed over the entire set of test inputs using both manual and automatic evaluations. The correlations between ranks assigned to systems by these average scores are indicative of the strength of our features to predict overall system rankings on the test set.

Analyzing the macro level results on TAC 2008 (Table 1), we find that the variants of distributional similarity and the topical relevance features produce system rankings very similar to those produced by

[4]Significance values for the correlations are produced using the AS 89 algorithm (Best and Roberts, 1975).

	QUERY - MACRO		QUERY - MICRO		UPDATE - MACRO		UPDATE - MICRO	
Features	**Pyr.**	**Resp.**	**Pyr.**	**Resp.**	**Pyr.**	**Resp.**	**Pyr.**	**Resp.**
Corpus-based Dist. Similarity	-0.887	-0.748	75.0	72.9	-0.833	-0.761	77.1	70.8
LexRes-based Dist. Similarity	-0.871	-0.723	72.9	68.8	-0.828	-0.755	77.1	68.8
Corpus-based Topical Relevance	-0.803	-0.696	72.9	70.8	-0.777	-0.720	75.0	72.9
LexRes-based Topical Relevance	-0.799	-0.705	70.8	70.8	-0.759	-0.735	72.9	70.8
LexRes-based Query Relevance	0.624	0.590	58.3	58.3	0.599	0.576	62.5	56.3
Corpus-based Query Relevance	0.615	0.547	56.3	52.1	0.613	0.576	60.4	56.3
LexRes-based Novelty	-	-	-	-	0.537	0.502	54.2	50.0
Corpus-based Novelty	-	-	-	-	0.530	0.500	58.3	45.8
Corpus-based Coherence	0.361	0.375	37.5	39.6	0.352	0.358	41.7	35.4
LexRes-based Coherence	0.353	0.362	35.4	37.5	0.349	0.358	37.5	37.5
Support Vector Regression	0.895	0.786	79.2	75.0	0.872	0.808	87.5	77.1
SIMetrix JS divergence	-0.880	-0.736	72.9	72.9	-0.827	-0.764	85.4	75.0
SIMetrix regression	0.867	0.705	77.1	66.7	0.789	0.605	81.3	58.3
ROUGE-1 recall (4 models)	0.859	0.806	97.9	95.8	0.912	0.865	97.9	95.8
ROUGE-2 recall (4 models)	0.905	0.873	100	91.7	0.941	0.884	100	91.7

Table 1: Input-summary evaluation on the query focused and update summarization tasks from TAC 2008 data: MACRO level Spearman correlations, all results are significant ($p < 0.05$); MICRO level percentage of inputs with significant correlations ($p < 0.05$).

human. Other features, on the other hand, are less predictive of content quality. Distributional similarities also outperform SIMetrix, which proves the importance of semantic representation of the input and summary for comparison purposes in summary content evaluation. Overall, our feature regression obtains the best correlations with both types of manual scores, and even outperforms ROUGE-1 regarding pyramid for query-focused task. The usefulness of novelty feature is also reflected in high SVR correlation results for the update summarization task.

Overall ROUGE correlation is evidence that the model summaries provide information that is unlikely to ever be approximated by exploring the input alone. However, our features can provide reliable estimates of system quality when averaged over a set of test inputs. We also observe that corpus-based models mostly outperform their corresponding lexical resource-based models. A possible reason is the higher coverage of Google News word2vec model comparing to the WordNet-based sense embedding model. For example, some words like proper nouns (e.g., *'Barak Obama'*) are not covered in WordNet. However, replacing a word's embedding by the sum of the embeddings of its senses could generally improve the quality of embeddings (Rothe and Schütze, 2015). That is why our SVR performs well by leveraging WordNet senses for more precise word embeddings, and involving Google News to complement the WordNet coverage.

Input Level (MICRO): For each individual input, we compare the rankings for the system summaries using manual and automatic evaluations. Micro-level analysis highlights the ability of an evaluation metric to assess the quality of system summaries produced for a specific input. This task is bound to be harder than system level predictions. For clarity, even with wrong prediction of rankings on a few inputs, the average scores (macro-level) for a system might not be affected.

To be in line with SIMetrix, we report the percentage of inputs for which significant correlations were obtained (Table 1). We observe that feature combination with SVR gives the best results overall, similar to our findings for the macro level. The implication is that no single feature can reliably predict good content for a particular input. Moreover, our feature regression outperforms SIMetrix. This is because our approach depends not merely on the distribution of terms in the input, and therefore provides better representation for a set of documents each describing different opinion on a given issue. For example, our topical relevance feature gives a representative vector for every important aspect of the document set. However, superiority of ROUGE performance to the rest of measures shows that model summaries generated for specific input would still give better indication of important information in the input.

6.1 Error Analysis:

In this study, we aim to assess the reliability of our metric for evaluation in the absence of human model summaries, where ROUGE cannot be used. It is noteworthy that we do not intend to directly compare the performance of ROUGE with our metric. Thereupon, we provide an error analysis to understand if our SVR and ROUGE are making errors in ordering the same systems or whether their errors are different. Since at the macro level, the correlations between our regression and pyramid scores is close to those of ROUGE-2, we further analyze their errors. We considered pairs of systems and identified the better system in each pair according to the pyramid scores. Afterwards, we recorded how often ROUGE-2 and the SVR provided the correct judgment for the pairs as indicated by the pyramid evaluation. Table 2 provides the results for all 1,653 pairs of systems at the macro level.

	SVR correct	**SVR incorrect**
ROUGE-2 correct	1,355(82.0%)	97(5.9%)
ROUGE-2 incorrect	100(6.0%)	101(6.1%)

Table 2: Error analysis: Overlap between ROUGE-2 and SVR predictions for the best system in a pair (TAC 2008, 1,653 pairs). The gold-standard judgment for a better system is computed using pyramid.

A large majority (82%) of the same pairs are correctly predicted by both ROUGE and the SVR. Another 6% of the pairs are such that both metrics do not provide the correct judgment. Therefore, ROUGE and our SVR appear to agree on a large majority of the system pairs. There is a small percentage (12%) that is correctly predicted by only one of the metrics.

6.2 Evaluation on the Test Set:

Our SVR was trained on the TAC 2008 data with pyramid scores as the target. Herein, we evaluate this metric using the TAC 2009 data (Table 3). We report the correlations obtained by ROUGE-SU4 as the official baseline measure at TAC 2009 for comparison of automatic evaluation metrics. The results indicate that the correlations are lower than on our development set. This might be caused by the different characteristics of inputs in two year's data (Louis and Nenkova, 2013). However, the SVR is consistently predictive across two years, and outperforms SIMetrix.

	QUERY - MACRO		**QUERY - MICRO**		**UPDATE - MACRO**		**UPDATE - MICRO**	
Metric	**Pyr.**	**Resp.**	**Pyr.**	**Resp.**	**Pyr.**	**Resp.**	**Pyr.**	**Resp.**
Support Vector Regression	0.80	0.75	87.5	77.1	0.77	0.65	79.2	75.0
SIMetrix JS divergence	-0.74	-0.71	84.1	75.0	-0.72	-0.61	77.3	72.7
SIMetrix Regression	0.77	0.67	81.8	65.9	0.71	0.54	75.0	52.3
ROUGE-SU4 (4 models)	0.92	0.79	95.5	81.8	0.85	0.69	100	86.4

Table 3: Input-summary evaluation on the query focused and update summarization tasks from TAC 2009 data: MACRO level Spearman correlations, all results are significant ($p < 0.05$); MICRO level percentage of inputs with significant correlations ($p < 0.05$).

Overall results also show that correlations with pyramid scores are higher than those with responsiveness. The reason is that our features mainly evaluate summary content. Responsiveness judgments, on the other hand, are based on both content and linguistic quality. Nevertheless, our SVR performs better than SIMetrix in replicating responsiveness scores. This might be advantaged by considering coherence as a linguistic quality feature. Hence, a natural extension of our work would be considering more linguistic quality features along with content evaluations.

7 Conclusion and Future Work

We have presented an effective model-free summary content evaluation approach that exploits the compositional properties of word and sense embeddings to develop a variety of features for input-summary comparisons. The results show that the strength of different features varies considerably, and their combination provides reliable estimates of summary content quality when model summaries are not available. This lends further support to our proposal to use semantic representation of the input and summary contents for the model-free summary content evaluation.

Our ongoing work includes considering distributional and relational semantics together (Fried and Duh, 2014; Verga and McCallum, 2016; Rossiello, 2016) for different sentence representations, and using more complex neural language models (Le and Mikolov, 2014; Zhang and LeCun, 2015; Jozefowicz et al., 2016) for the comparison.

Acknowledgements

We thank the anonymous reviewers for their insightful comments and valuable suggestions. The first author was supported by the "Australian Government Research Training Program Scholarship".

References

Rachit Arora and Balaraman Ravindran. 2008. Latent Dirichlet Allocation based multi-document summarization. In *Proceedings of the Second Workshop on Analytics for Noisy Unstructured Text Data*, pages 91–97. ACM.

DJ Best and DE Roberts. 1975. Algorithm AS 89: the upper tail probabilities of Spearman's rho. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 24(3):377–379.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Luis Adrián Cabrera-Diego and Juan-Manuel Torres-Moreno. 2017. Summtriver: A new trivergent model to evaluate summaries automatically without human references. *Data & Knowledge Engineering*.

Robert L Donaway, Kevin W Drummey, and Laura A Mather. 2000. A comparison of rankings produced by summarization evaluation measures. In *Proceedings of the 2000 North American Chapter of the Association for Computational Linguistics - Applied Neural Language Processing Conference: Workshop on Automatic Summarization*, pages 69–78. Association for Computational Linguistics.

Samira Ellouze, Maher Jaoua, and Lamia Hadrich Belguith. 2013. An evaluation summary method based on a combination of content and linguistic metrics. In *Recent Advances in Natural Language Processing*, pages 245–251. Citeseer.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Daniel Fried and Kevin Duh. 2014. Incorporating both distributional and relational semantics in word representations. *arXiv preprint arXiv:1412.4369*.

Dimitrios Galanis, Gerasimos Lampouras, and Ion Androutsopoulos. 2012. Extractive multi-document summarization with integer linear programming and support vector regression. *Proceedings of the International Conference on Computational Linguistics*, pages 911–926.

Tom Griffiths. 2002. Gibbs sampling in the generative model of Latent Dirichlet Allocation.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches out: Proceedings of the Association for Computational Linguistics Workshop*, volume 8.

Annie Louis and Ani Nenkova. 2009a. Automatically evaluating content selection in summarization without human models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1*, pages 306–314. Association for Computational Linguistics.

Annie Louis and Ani Nenkova. 2009b. Predicting summary quality using limited human input. In *Text Analysis Conference*.

Annie Louis and Ani Nenkova. 2013. Automatically assessing machine summary content without a gold standard. *Computational Linguistics*, 39(2):267–300.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Jun-Ping Ng and Viktoria Abrecht. 2015. Better summarization evaluation with word embeddings for ROUGE. *arXiv preprint arXiv:1508.06034*.

Rebecca J Passonneau, Ani Nenkova, Kathleen McKeown, and Sergey Sigelman. 2005. Applying the pyramid method in DUC 2005. In *Proceedings of the Document Understanding Conference*.

Rebecca J Passonneau, Emily Chen, Weiwei Guo, and Dolores Perin. 2013. Automated pyramid scoring of summaries using distributional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: Short Papers*.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Dragomir R Radev, Simone Teufel, Horacio Saggion, Wai Lam, John Blitzer, Hong Qi, Arda Celebi, Danyu Liu, and Elliott Drabek. 2003. Evaluation challenges in large-scale document summarization. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 375–382. Association for Computational Linguistics.

Dragomir R Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing & Management*, 40(6):919–938.

Gaetano Rossiello. 2016. Neural abstractive text summarization. In *Proceedings of the Doctoral Consortium of AI*IA 2016 co-located with the 15th International Conference of the Italian Association for Artificial Intelligence*, pages 70–75.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. *arXiv preprint arXiv:1507.01127*.

Horacio Saggion, Juan-Manuel Torres-Moreno, Iria da Cunha, and Eric SanJuan. 2010. Multilingual summarization evaluation without human models. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1059–1067. Association for Computational Linguistics.

Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond Wong, and Fang Chen. 2017. A semantically motivated approach to compute ROUGE scores. *arXiv preprint arXiv:1710.07441*.

Abhishek Singh and Wei Jin. 2016. Ranking summaries for informativeness and coherence without reference summaries. In *Proceedings of the Twenty-Ninth International Florida Artificial Intelligence Research Society Conference*, pages 104–109.

Vladimir Naumovich Vapnik. 1999. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999.

Patrick Verga and Andrew McCallum. 2016. Row-less universal schema. *arXiv preprint arXiv:1604.06361*.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.

A review of Spanish corpora annotated with negation

Salud María Jiménez-Zafra[1], Roser Morante[2],
María Teresa Martín-Valdivia[1], L. Alfonso Ureña-López[1]
[1] SINAI, Computer Science Department, Advanced Studies Center in ICT (CEATIC)
Universidad de Jaén, Campus Las Lagunillas s/n, E-23071
{sjzafra, maite, laurena}@ujaen.es
[2] CLTL Lab, Computational Linguistics
VU University Amsterdam, De Boelelaan 1105, 1081 HV
r.morantevallejo@vu.nl

Abstract

The availability of corpora annotated with negation information is essential to develop negation processing systems in any language. However, there is a lack of these corpora even for languages like English, and when there are corpora available they are small and the annotations are not always compatible across corpora. In this paper we review the existing corpora annotated with negation in Spanish with the purpose of first, gathering the information to make it available for other researchers and, second, analyzing how compatible are the corpora and how has the linguistic phenomenon of negation been addressed. Our final aim is to develop a supervised negation processing system for Spanish, for which we need training and test data. Our analysis shows that it will not be possible to merge the small corpora existing for Spanish due to lack of compatibility in the annotations.

Title and Abstract in Spanish

Revisión de los corpus españoles anotados con negación

La disponibilidad de corpus anotados con información sobre la negación es esencial para cualquier idioma, ya que son necesarios para poder desarrollar sistemas capaces de procesar este fenómeno lingüístico. Sin embargo, hay una escasez de corpus anotados con negación, incluso para idiomas como el inglés, y cuando hay corpus disponibles, en la mayoría de los casos son pequeños y las anotaciones no siempre son compatibles entre ellos. En este trabajo revisamos los corpus anotados con información sobre la negación en español con el propósito, en primer lugar, de recopilar la información para que esté disponible para otros investigadores y, en segundo lugar, de analizar la compatibilidad de los corpus y las estructuras de negación que se han anotado. Nuestro objetivo final es desarrollar un sistema automático para el procesamiento de la negación en español. Nuestro análisis muestra que no será posible unir los pequeños corpus existentes actualmente en español debido a la falta de compatibilidad en las anotaciones.

1 Introduction

Nowadays, there is a vast amount of information on the Internet. The large number of sources and the high volume of texts make it difficult for users to select information of interest. In order to extract fine-grained information, automatic systems need to be able to process a diversity of linguistic phenomena such as negation, irony or sarcasm that are used to add extra-propositional meaning. In this study, we focus on negation.

Negation is a very relevant linguistic phenomenon for most of Natural Language Procesing (NLP) tasks, such as information extraction, question answering and sentiment analysis, since negation cues act as operators that can change the meaning of the words that are within their scope by changing the truth value of propositions (Horn, 1989). Negation is a main linguistic phenomenon whose computational

Proceedings of the 27th International Conference on Computational Linguistics, pages 915–924
Santa Fe, New Mexico, USA, August 20-26, 2018.

treatment has not been solved yet, not even for English, due to its complexity and the multiple forms in which it can appear (syntactic, lexical and morphological). Because of the low frequency of negations as compared to other phenomena, the impact in the performance of applications is low from a quantitative perspective, but high from a qualitative view, for example, when processing clinical records. Moreover, some systems we use regularly do not treat this phenomenon effectively. Not even Google deals properly with negation in Spanish. The search "películas que no sean de aventuras", returns adventure movies, whereas it should return non adventure movies.

The standard two phases of a negation processing system are: identifying the presence of negation markers and determining their scope. As in NLP tasks, the availability of corpora annotated with information about negation is essential to train algorithms. However, it is not easy to find corpora annotated with negation, especially in languages other than English. In addition, it is not only necessary that corpora exist, but also that corpora are publicly available for the community to use them, that they are well documented, and that they contain annotations of quality. Ideally, they should also be large enough to allow training robust machine learning systems.

There are different catalogs and platforms that provide information about resources and/or access to them, such as LDC catalog[1], ELRA catalog[2], LRE Map[3], META-SHARE[4] and ReTeLe[5] catalog. If we make a query[6] in these repositories for Spanish corpora annotated with negation, we only find 1 resource in ReTeLe catalog, the SFU Review$_{SP}$-NEG corpus (Jiménez-Zafra et al., 2018). Therefore, we decided to perform an exhaustive search of resources. In this way, we facilitate the task for researchers interested in working on this topic by providing a description of the corpora as well as the direct links to get the data when possible.

In this work, we review the existing Spanish corpora annotated with negation. We focus on Spanish because i) there are no negation processing systems available, ii) it is the second language with most native speakers, and iii) it is the third language most used on the Internet. Our final goal is to develop a supervised negation processing system, for which we need to use annotated corpora. The main purposes of our review are to analyze how compatible are the Spanish corpora annotated with negation and to find out whether the annotations account for the complexity of negation in Spanish.

The rest of the paper is organized as follows: in Section 2 we describe the main features of negation in Spanish, in Section 3 we present the corpora annotated with negation, in Section 4 we analyzed them, in Section 5 we propose a solution to the problems found and, finally, we put forward conclusions in Section 6.

2 Negation in Spanish

Processing negation is not as easy as using a list of negation markers and applying look-up methods. They can be used to find out potential negation cues but they are not adequate because the presence of a cue does not imply that it acts as a negation. In the sentence "You bought the car to use it, didn't you?" the cue "not" is not used as a negation but it is used to reinforce the first part of the sentence. Moreover, it is also necessary to identify the scope or part of the sentence affected by the negation and its focus, the part more prominently negated. If we want to advance in the study of this phenomenon, as for most of NLP tasks, the availability of annotated corpora is essential to train algorithms. According to existing resources for English, annotating negation involves the annotation of the following aspects:

- *Negation cue*: lexical item(s) that modify the truth value of the propositions that are within its scope. There are different types of negation according to the type of the negation cue used:

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

[2] http://catalog.elra.info/en-us/

[3] http://lremap.elra.info/

[4] http://www.meta-share.org/

[5] ReTeLe is a network of resources for language technologies that has as goal the compilation and documentation of linguistic resources created in Spain. ReTele catalog contains metadata describing resources in RDF format. ReteLe catalog: http://linguistic.linkeddata.es/retele-share/sparql-editor/

[6] Query performed on March 6th 2018

  - Syntactic negation, if a syntactically independent negation marker is used to express negation (i.e. *no ['no/not']*, *nunca ['never']*).
  - Lexical negation, if a word is used whose meaning has a negative component (i.e. *negar ['deny']*, *desistir ['desist']*).
  - Morphological negation, if a morpheme is used to express the negation (i.e. *i-* in *ilegal ['illegal']*, *in* in *incoherente ['incoherent']*). It is also known as affixal negation.

- *Scope*: the part of the sentence affected by the negation cue (Vincze et al., 2008). The scope can be continuous or discontinuous.

- *Focus*: the part of the scope that is most prominently or explicitly negated (Blanco and Moldovan, 2011).

- *Negated event*: the event that is directly negated by the negation cue, usually a verb, a noun or an adjective (Kim et al., 2008).

This is just a list of the main aspects that have been annotated for negation. However, each language has specific linguistic resources to express negation and specific negation structures, which should also be reflected in the information annotated in corpora. As we will show in Section 4, most existing annotation schemes for Spanish do not account for the complexity of the linguistic structures used to express negation that are present in texts. This happens mainly because of two reasons: first, annotation of negation started with the annotation of clinical reports in English (Chapman et al., 2001; Goldin and Chapman, 2003; Mutalik et al., 2001a), where there is not too much variation of negation structures. Second, corpora have been created for specific purposes, such as extracting negated clinical events, and not with the intention of accounting for all the linguistic complexity of the negation phenomenon.

An exception to this is the SFU Review$_{SP}$-NEG corpus (Jiménez-Zafra et al., 2018; Martí et al., 2016). The guidelines specify a great variety of negation patterns at the syntactic level that we summarize below. Additionally, the guidelines also specify expressions that involve a negation cue but do not express negation.

On the one hand, patterns that express negation can be divided into three categories:

1. *Simple negation markers*, if they are composed of only one single negation marker (i.e. *no ['no/not']*, *nunca ['never']*).
2. *Complex negation markers*, if negation is expressed using two or more negation markers that can be continuous (i.e. *casi no ['almost not']*, *casi nunca ['hardly ever']*) or discontinuous (i.e. *no ... en absoluto ['not ... at all']*, *no ... mucho ['not ... a lot']*). Complex negation markers are usually used to reinforce negation or to modulate the value of negation (increase or diminish the degree of negation).
3. *Expressions that do not contain any negation marker*: lexicalized complex constructions that express negation in specific contexts even though they do not contain any negation marker (i.e. *en mi vida ['in my life']*).

On the other hand, patterns that not express negation can be categorized as follows:

1. *Rhetorical negation markers*, if a negation marker is used with an emphatic or expletive value, that is, if it is used to make a sentence more full, intense or harmonious, although it is not necessary to understand the meaning of the sentence (i.e. *Viniste a verlo, ¿no? ['You came to see him, didn't you?']*).
2. *Idioms containing negation markers*, if an idiom (i.e. *ni corta ni perezosa ['without thinking twice']*) or a lexicalised cue (i.e. *hasta que no ['until']*) contain a negation marker that does not express negation.

3. *Negation markers in contrastive constructions*, if negation markers are used to counterpose different ideas, to correct something, to introduce new information or to express obligation, rather than to express negation (i.e. *No hay más solución que comprar una lavadora ['There is no other solution than to buy a washing machine']*).
4. *Negation markers in comparative constructions*, if negation markers are used to compare some property with something, that is, negation is used to place an entity below or above another entity on a scale (i.e. *No es tan grande como me lo imaginaba ['It is not as big as I imagined']*).

3 Corpora annotated with negation

In this section, the Spanish corpora annotated with negation are presented[7]. To the best of our knowledge, five corpora exist from different domains, although the clinical domain is the predominant one.

3.1 UAM Spanish Treebank

The first Spanish corpus annotated with negation that we are aware of is the UAM Spanish Treebank (Moreno et al., 2003), which was enriched with the annotation of negation cues and their scopes (Sandoval and Salazar, 2013).

The initial UAM Spanish Treebank consisted of 1,500 sentences extracted from newspaper articles (*El País Digital* and *Compra Maestra*) that were annotated syntactically. Trees were encoded in a nested structure, including syntactic category, syntactic and semantic features, and constituent nodes, following the Penn Treebank model. Later, this version of the corpus was extended with the annotation of negation and 10.67% of the sentences were found to contain negations (160 sentences).

In this corpus, syntactic negation was annotated but not lexical nor morphological negation. It was annotated by two experts in corpus linguistics who followed similar guidelines to those of Bioscope corpus (Szarvas et al., 2008; Vincze, 2010). They included negation cues within the scope as in Bioscope and NegDDI-DrugBank (Bokharaeian et al., 2014). All the arguments of the negated events were also included in the scope of negation, including the subject, which was excluded from the scope in active sentences in Bioscope. There is no information about inter-annotator agreement.

The UAM Spanish Treebank corpus is freely available at `http://www.lllf.uam.es/ESP/Treebank.html`. It is in XML format, negation cues are tagged with the label *Type="NEG"* and the scope of negation is tagged with the label *Neg="YES"* in the syntactic constituent on which negation acts.

3.2 IxaMed-GS

The IxaMed-GS corpus (Oronoz et al., 2015) is composed of 75 real electronic health records from the outpatient consultations of the Galdakao-Usansolo Hospital in Biscay (Spain). It was annotated by two experts in pharmacology and pharmacovigilance with entities related to diseases and drugs, and with the relationships between entities indicating adverse drug reaction events. They defined their own annotation guidelines taken into consideration the issues that should be considered for the design of a corpus according to Ananiadou and McNaught (2006).

The objective of this corpus was not the annotation of negation but the identification of entities and events in clinical reports. However, negation and speculation were taken into account in the annotation process. In the corpus, four entity types were annotated: diseases, allergies, drugs and procedures. For diseases and allergies, they distinguished between negated entity, speculated entity and entity (for non-speculative and non-negated entities). 2,362 diseases were annotated, out of which 490 (20.75%) were tagged as negated diseases and 40 (1.69%) as speculated diseases. 404 allergy entities were identified, of which 273 (67.57%) were negated and 13 (3.22%), speculated. The quality of the annotation process was assessed by measuring the inter-annotator agreement, which was 90.53% for entities and 82.86% for events.

The corpus might be acquired via the EXTRECM project[8] by agreeing to some conditions that include a confidentiality agreement.

[7] `https://github.com/sjzafra/spanish_negation_corpora`
[8] `http://ixa.si.ehu.eus/extrecm`

3.3 SFU Review$_{SP}$-NEG

The SFU Review$_{SP}$-NEG[9] (Jiménez-Zafra et al., 2018) is the first Spanish corpus that includes the event in the annotation of negation and that takes into account discontinuous negation markers. Moreover, it is the first corpus where the effect of the negation on the words that are within its scope is annotated, that is, whether there is a change in the polarity or an increment or reduction of its value. It is an extension of the Spanish part of the SFU Review corpus (Taboada et al., 2006) and it could be considered as the counterpart of the SFU Review Corpus with negation and speculation annotations[10] (Konstantinova et al., 2012).

The Spanish SFU Review corpus consists of 400 reviews extracted from the website *Ciao.es* that belong to 8 different domains: cars, hotels, washing machines, books, cell phones, music, computers, and movies. For each domain there are 50 positive and 50 negative reviews, defined as positive or negative based on the number of stars given by the reviewer (1-2=negative; 4-5=positive; 3-star review were not included). Later, it was extended to the SFU Review$_{SP}$-NEG corpus in which each review was automatically annotated at the token level with POS-tags and lemmas, and manually annotated at the sentence level with negation cues and their corresponding scopes and events. It is composed of 9,455 sentences, out of which 3,022 sentences (31.97%) contain at least one negation marker.

In this corpus, syntactic negation was annotated but not lexical nor morphological negation, as in the UAM Spanish Treebank corpus. Unlike this one, annotations on the event and on how negation affects the polarity of the words within its scope were included. The annotations were performed by two senior researchers with in-depth experience in corpus annotation who supervised the whole process and two trained annotators who carried out the annotation task. The Kappa coefficient for inter-annotator agreement was of 0.97 for negation cues, 0.95 for negated events and 0.94 for scopes.[11] A detailed discussion of the main sources of disagreements can be found in (Jiménez-Zafra et al., 2016).

The guidelines of the Bioscope corpus were taken into account but after a thorough analysis of negation in Spanish, a typology of Spanish negation patterns was defined (Martí et al., 2016). As in Bioscope, NegDDI-DrugBank and UAM Spanish Treebank, negation markers were included within the scope. Moreover, the subject was also included within the scope when the word directly affected by negation is the verb of the sentence, as in ConanDoyle-neg corpus (Morante and Daelemans, 2012). The event was also included in the scope of negation as in ConanDoyle-neg corpus.

The SFU Review$_{SP}$-NEG is publicly available and can be downloaded at `http://sinai.ujaen.es/sfu-review-sp-neg-2/`.

3.4 UHU-HUVR

The UHU-HUVR (Cruz Díaz et al., 2017) is the first Spanish corpus in which affixal negation is annotated. It is composed of 604 clinical reports from the Virgen del Rocío Hospital in Seville (Spain). 276 of this clinical documents correspond to radiology reports and 328 to the personal history of anamnesis reports written in free text.

In this corpus, all types of negation were annotated, syntactic, morphological (affixal negation), and lexical. It was annotated with negation markers and the negated events by two domain expert annotators following closely the Thyme corpus guidelines (Styler IV et al., 2014) with some adaptations. In the anamnesis reports, 1,079 sentences (35.20%) were found to contain negations out of 3,065 sentences. On the other hand, 1,219 sentences (22.80%) out of 5,347 sentences were annotated with negations in the radiology reports. The Dice coefficient for inter-annotator agreement was higher than 0.94 for negation markers and higher than 0.72 for negated events. Most of the disagreements were the result of a human error, i.e., the annotators missed a word or included a word that did not belong either to the event or to the marker. However, other cases of disagreement can be explained by the difficulty of the task and the lack of clear guidance. They encountered the same type of disagreements as Jiménez-Zafra et al. (2016) when annotating the SFU Review$_{SP}$-NEG corpus.

[9]First Online: 22 May 2017 `https://doi.org/10.1007/s10579-017-9391-x`

[10]`https://www.sfu.ca/~mtaboada/SFU_Review_Corpus.html`

[11]The inter-annotator agreement values have been corrected with respect to those published in (Jiménez-Zafra et al., 2018) due to the detection of an error in the calculation thereof.

Authors say that the annotated corpus will be made publicly available, but it is not currently available probably because of legal and ethical issues.

3.5 IULA Spanish Clinical Record

The IULA Spanish Clinical Record corpus (Marimon et al., 2017) contains 300 anonymized clinical records from several services of one of the main hospitals in Barcelona (Spain) that was annotated with negation markers and their scopes. It contains 3,194 sentences, out of which 1,093 (34.22%) were annotated with negation cues.

In this corpus, syntactic negation and lexical negation were annotated but not morphological negation. It was annotated with negation cues and their scopes by three computational linguists annotators advised by a clinician. The inter-annotator agreement Kappa rates were 0.85 between annotators 1 and 2, and annotators 1 and 3; and 0.88 between annotators 2 and 3. The authors defined their own annotation guidelines taking into account the currently existing guidelines for corpora in English (Mutalik et al., 2001b; Szarvas et al., 2008; Morante and Daelemans, 2012). Differently from previous work, they did not include the negation cue nor the subject in the scope (except when the subject is located after the verb).

The corpus is publicly available with a CC-BY-SA 3.0 license and it can be downloaded at `http://eines.iula.upf.edu/brat//#/NegationOnCR_IULA/`.

4 Analysis

In order to take an informed decision about which (combination of) corpus can we use to develop a system, we have performed a detailed analysis. An overview of the information annotated can be found in Tables 1 and 2.

Table 1 presents a summary of the negation aspects annotated in each corpus, the domain of the documents, the size in number of sentences and the inter-annotator measure used to estimate the agreement. Table 2 contains the type of negation cues that have been annotated in each corpus and negation types that have been taken into account.

	UAM Spanish Treebank	IxaMed-GS	SFU Review$_{SP}$-NEG	UHU-HUVR	IULA Spanish Clinical Record
Domain	Newspaper articles	Clinical reports	Movies, books, product reviews	Clinical reports	Clinical reports
Total sentences	1,500	NA	9,455	8,412	3,194
Sentences with negation	160 (10.73%)	NA	3,022 (31.97%)	2,298 (27.32%)	1,093 (34.22%)
Negation cue	✓	-	✓	✓	✓
Scope	✓	-	✓	-	✓
Event	-	✓	✓	✓	-
Focus	-	-	-	-	-
IAA measure	NA	%	Kappa	Dice	Kappa

Table 1: Spanish corpora annotated with negation (NA: Non-Available, -: Absent, ✓:Present).

The years of publication of the corpora show the novelty of the task. The first corpus annotated with negation in Spanish appeared in 2013, while the others have been compiled in the last two years. Important aspects of the corpora to be analyzed are the type of documents included, the size, the guidelines applied, the annotation schemes and the inter-annotator agreement.

Three of the five corpora focus on the clinical domain, which reflects the demands for the treatment of negation in this domain. Processing negation in clinical documents is crucial because the health of a patient is at stake, it is not the same to say that *a patient has* or *does not have a disease* or that *he is* or *is not allergic to a compound*. Moreover, we observe that there are other domains of interest, such as product reviews and news. The rating of a film will be totally different if a viewer says *"I liked the*

	UAM Spanish Treebank	SFU Review$_{SP}$-NEG	UHU-HUVR	IULA Spanish Clinical Record
Syntactic	✓	✓	✓	✓
Lexical	-	-	✓	✓
Morphological	-	-	✓	-
Simple	✓	✓	✓	✓
Complex	NA	✓	NA	NA
Expressions not containing negation markers	NA	✓	NA	NA
Rhetorical	NA	✓	NA	NA
Idioms	NA	✓	NA	NA
Contrastive	NA	✓	NA	NA
Comparative	NA	✓	NA	NA

Table 2: Negation types in the Spanish corpora annotated with negation cues (NA: Non-Available, -: Absent, ✓:Present).

movie" or if he says *"I did not like the movie"*. In the case of news, the impact would be totally different if it is said *"A plane crashed"* or if it is said *"Finally the plane did not crash"*.

As for the size, the available corpora are not very large and, although negation is an important phenomenon for NLP tasks, it is relatively unfrequent. In newspaper articles, only 10.73% of the sentences contain negation and, in the case of product reviews and clinical reports this value amounts to 31.97% and 29.22%[12], respectively. These percentages show the need of continuing working on the annotation of negation and its study. Training supervised systems usually relies on the existence of annotated corpora and, consequently, corpus generation is an important part for the development and testing of NLP techniques.

In relation to the guidelines used, it is noteworthy that there is no uniformity. In the first place, there are divergences in the negation aspects being annotated (negation cue, scope, event, focus). None of the corpora contain annotations of the four elements and the focus has been annotated in none of them. Only the SFU Review$_{SP}$-NEG corpus contains annotations of three elements (negation cue, scope and event). The UAM Spanish Treebank and the IULA Spanish Clinical Record corpora have focused on annotating negation cues and their scopes, and the UHU-UVR on the annotation of negation cues and their events. In the second place, these elements have not been annotated in the same way:

- *Negation cue*. As it has been described in Section 2, negation in Spanish is a complex phenomenon. Depending on the negation cue used, it can be syntactic, lexical or morphological. Moreover there are different types of negation patterns that express negation (simple negation markers, complex negation markers and expressions not containing any negation marker) and that do not express negation (rhetorical negation markers, idioms containing negation markers, negation markers in contrastive constructions and negation markers in comparative constructions). Only the UHU-UVR corpus contains annotations about the three types of negation cues (syntactic, lexical and morphological). The UAM Spanish Treebank and the SFU Review$_{SP}$-NEG corpora take only into account syntactic negation, and the IULA Spanish Clinical Record corpus also considered it along with lexical negation. However, in general, it is not specified whether the complexity of negation has been taken into account during the annotation process. An exception to this is the SFU Review$_{SP}$-NEG corpus. The guidelines specify that the different types of negation patterns according to the semantic interpretation have been considered for the annotation of negation at the syntactic level. This

[12]For clinical reports, it has been considered the average corresponding to the sentences annotated in UHU-HUVR and IULA Spanish Clinical Record corpora.

information has been summarized in Table 2.

- *Scope*. In the UAM Spanish Treebank and the SFU Review$_{SP}$-NEG corpora negation cues were included within the scope of negation as in Bioscope (Vincze et al., 2008), but in the IULA Spanish Clinical Record corpus they were not included. On the other hand, in the UAM Spanish Treebank all the arguments of the negated events, including the subject, were included within the scope of negation. However, in the SFU Review$_{SP}$-NEG corpora, the subject was included within the scope of negation when the word directly affected by negation is the verb of the sentence, as in ConanDoyle-neg corpus (Morante and Daelemans, 2012). In the IULA Spanish Clinical Record corpus it was not included, except when the subject is located after the verb.

- *Focus*. This element has not been annotated in the existing corpora.

- *Negated event*. In the SFU Review$_{SP}$-NEG corpus the event is always included within the scope of negation, as in Conan Doyle-neg corpus, and it is usually the head of the phrase in which the negation appears. In the UHU-UVR corpus negated events are annotated if they are clinically relevant, so not all negated events are annotated.

As for the annotation schemes, there is no a standard one. Each project devices the own scheme according to the needs of the project, which has consequences in the compatibility of the annotations across corpora. It is not possible to combine the corpora for machine learning purposes in order to obtain more training data.

Furthermore, the annotated corpora do not use the same coefficient to measure the inter-annotator agreement and there is even a corpus for which this measure is not provided. Providing this measure is very important because it allows to show the reliability of the annotation and the difficulty of the task.

The main purposes of our review was to analyse how compatible are the Spanish corpora annotated with negation and to find out whether the annotations account for the complexity of negation in Spanish. We have found that the existing corpora are not compatible, they have not been annotated with the same purpose and they do not contain annotations for the same negation aspects. Even in those corpora that contain annotations for the same negation aspect, the annotations have been made based on different guidelines. In relation to the complexity of negation, only the SFU Review$_{SP}$-NEG guidelines specify how different negation structures should be annotated. The guidelines of the other corpora do not contain information about the linguistic structures that have been taken into account.

5 A solution

The result of this analysis opens some questions. The existing annotation schemes are not compatible because of differences in genre, annotation guidelines, and the aspects of negation that have been annotated (negation cue, scope, event, focus). Therefore, it would be desirable to define a new scheme that integrates the contents of existing schemes. The scheme should be domain independent and all negation elements should be annotated in the same way. We are currently working on this. We have proposed a workshop (NEGES - Workshop on Negation in Spanish[13]) to advance in the study of this phenomenon and one of the tasks[14] has as goal to reach an agreement on the guidelines to follow. In order to participate in the task, researchers must analyze the existing guidelines and send a document indicating which aspects of the guidelines they agree with and which they do not, all duly justified. This information will be used to discuss the aspects of interest and to try to reach a consensus.

Additionally, we will conduct a study on the feasibility of an automatic conversion of the corpora. It is a pity that after all the work done and the time invested in the annotation of these corpora, it is not possible to merge them. We will explore semi-automatic approaches to re-annotate the most corpora possible.

[13] http://www.sepln.org/workshops/neges/index.php
[14] Task 1 - Annotation guidelines

6 Conclusions

Processing negation is a very important task in NLP because negation can change the truth value of a proposition. Detecting this is crucial in some tasks such as sentiment analysis or question answering. Most existing work on the treatment of negation has been carried out for English, but it is necessary to focus on other languages such as Chinese, Spanish or Arabic.

In this paper, we have presented the existing Spanish corpora annotated with negation information and we have described the main features of each one. Three of them are centered on the clinical domain probably because it is one of the areas where the treatment of negation is crucial in order not to extract false information about clinical conditions. It is worth noting that the first corpus annotated with negation appeared in 2013, whereas the rest have been compiled in the last two years, which shows the novelty of the task.

Our analysis shows that it would not be possible to merge the existing corpora in order to obtain a bigger one to train a machine learning system because of differences in genre, annotation guidelines, and the aspects of negation that have been annotated. The corpora have not been annotated with the same purpose and they do not contain annotations for the same negation aspects. Despite the fact that there have been already several annotation efforts, the community lacks a standard to annotate negation, contrary to what happens with other phenomena such as semantic roles.

As future work we plan to develop annotation standards to annotate negation in Spanish, in such a way that they are applicable to different genres and domains, and to analyze the feasibility of an automatic conversion of the corpora to a common annotation scheme.

Acknowledgments

This work has been partially supported by a grant from the Ministerio de Educación Cultura y Deporte (MECD - scholarship FPU014/00983), Fondo Europeo de Desarrollo Regional (FEDER) and REDES project (TIN2015-65136-C2-1-R) from the Spanish Government. RM is supported by the Netherlands Organization for Scientific Research (NWO) via the Spinoza-prize awarded to Piek Vossen (SPI 30-673, 2014-2019).

References

Sophia Ananiadou and John McNaught. 2006. *Text mining for biology and biomedicine*. Artech House London.

Eduardo Blanco and Dan Moldovan. 2011. Semantic representation of negation using focus detection. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 581–589, Portland, Oregon, USA, June. Association for Computational Linguistics.

Behrouz Bokharaeian, Alberto Diaz, Mariana Neves, and Virginia Francisco. 2014. Exploring negation annotations in the DrugDDI Corpus. In *Fourth Workshop on Building and Evaluating Resources for Health and Biomedical Text Processing (BIOTxtM 2014)*. Citeseer.

W. W. Chapman, W. Bridewell, P. Hanbury, G. F. Cooper, and B.G. Buchanan. 2001. A simple algorithm for identifying negated findings and diseases in discharge summaries. *J Biomed Inform*, 34:301–310.

Noa P Cruz Díaz, Roser Morante Vallejo, Manuel J Maña López, Jacinto Mata Vázquez, and Carlos L Parra Calderón. 2017. Annotating Negation in Spanish Clinical Texts. *SemBEaR 2017*, page 53.

l. M. Goldin and W.W. Chapman. 2003. Learning to detect negation with 'Not' in medical texts. In *Proceedings of ACM-SIGIR 2003*.

Laurence R. Horn. 1989. *A natural history of negation*. CSLI Publications.

Salud María Jiménez-Zafra, M Teresa Martín-Valdivia, L Alfonso Ureña-López, M Antonia Martí, and Mariona Taulé. 2016. Problematic Cases in the Annotation of Negation in Spanish. In *Proceedings of the Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics (ExProM)*, pages 42–48.

Salud María Jiménez-Zafra, Mariona Taulé, M Teresa Martín-Valdivia, L Alfonso Ureña-López, and M Antónia Martí. 2018. SFU ReviewSP-NEG: a Spanish corpus annotated with negation for sentiment analysis. A typology of negation patterns. *Language Resources and Evaluation*, 52(2):533–569.

Jin-Dong Kim, Tomoko Ohta, and Jun'ichi Tsujii. 2008. Corpus annotation for mining biomedical events from literature. *BMC bioinformatics*, 9(1):1.

Natalia Konstantinova, Sheila CM De Sousa, Noa P Díaz Cruz, Manuel J Maña López, Maite Taboada, and Ruslan Mitkov. 2012. A review corpus annotated for negation, speculation and their scope. In *LREC*, pages 3190–3195.

Montserrat Marimon, Jorge Vivaldi, Núria Bel, and Roc Boronat. 2017. Annotation of negation in the IULA Spanish Clinical Record Corpus. *SemBEaR 2017*, 5(36.41):43.

M Antónia Martí, M Teresa Martín Valdivia, Mariona Taulé, Salud María Jiménez Zafra, Montserrat Nofre, and Laia Marsó. 2016. La negación en español: análisis y tipología de patrones de negación. *Procesamiento del Lenguaje Natural*, 57:41–48.

Roser Morante and Walter Daelemans. 2012. ConanDoyle-neg: Annotation of negation in Conan Doyle stories. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, Istanbul*. Citeseer.

Antonio Moreno, Susana López, Fernando Sánchez, and Ralph Grishman. 2003. Developing a syntactic annotation scheme and tools for a Spanish treebank. In *Treebanks*, pages 149–163. Springer.

A.G. Mutalik, A. Deshpande, and P.M. Nadkarni. 2001a. Use of general-purpose negation detection to augment concept indexing of medical documents. a quantitative study using the UMLS. *J Am Med Inform Assoc*, 8(6):598–609.

Pradeep G Mutalik, Aniruddha Deshpande, and Prakash M Nadkarni. 2001b. Use of general-purpose negation detection to augment concept indexing of medical documents: a quantitative study using the UMLS. *Journal of the American Medical Informatics Association*, 8(6):598–609.

Maite Oronoz, Koldo Gojenola, Alicia Pérez, Arantza D'ıaz de Ilarraza, and Arantza Casillas. 2015. On the creation of a clinical gold standard corpus in Spanish: Mining adverse drug reactions. *Journal of biomedical informatics*, 56:318–332.

Antonio Moreno Sandoval and Marta Garrote Salazar. 2013. La anotación de la negación en un corpus escrito etiquetado sintácticamente. Annotation of negation in a written treebank. *Revista Iberoamericana de Linguistica*, 8.

William F Styler IV, Steven Bethard, Sean Finan, Martha Palmer, Sameer Pradhan, Piet C de Groen, Brad Erickson, Timothy Miller, Chen Lin, Guergana Savova, et al. 2014. Temporal annotation in the clinical domain. *Transactions of the Association for Computational Linguistics*, 2:143–154.

György Szarvas, Veronika Vincze, Richárd Farkas, and János Csirik. 2008. The BioScope corpus: annotation for negation, uncertainty and their scope in biomedical texts. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, pages 38–45. Association for Computational Linguistics.

Maite Taboada, Caroline Anthony, and Kimberly Voll. 2006. Methods for creating semantic orientation dictionaries. In *Proceedings of the 5th Conference on Language Resources and Evaluation (LREC'06)*, pages 427–432.

Veronika Vincze, György Szarvas, Richárd Farkas, György Móra, and János Csirik. 2008. The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC bioinformatics*, 9(11):1.

Veronika Vincze. 2010. Speculation and negation annotation in natural language texts: what the case of bioscope might (not) reveal. In *Proceedings of the workshop on negation and speculation in natural language processing*, pages 28–31. Association for Computational Linguistics.

Document-level Multi-aspect Sentiment Classification by Jointly Modeling Users, Aspects, and Overall Ratings

Junjie Li[1,2], **Haitong Yang**[3] **and Chengqing Zong**[1,2,4]
[1] National Laboratory of Pattern Recognition, Institute of Automation, CAS, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] School of Computer, Central China Normal University, Wuhan 430079, China
[4] CAS Center for Excellence in Brain Science and Intelligence Technology
{junjie.li, cqzong}@nlpr.ia.ac.cn, htyang@mail.ccnu.edu.cn

Abstract

Document-level multi-aspect sentiment classification aims to predict user's sentiment polarities for different aspects of a product in a review. Existing approaches mainly focus on text information. However, the authors (i.e. users) and overall ratings of reviews are ignored, both of which are proved to be significant on interpreting the sentiments of different aspects in this paper. Therefore, we propose a model called Hierarchical User Aspect Rating Network (HUARN) to consider user preference and overall ratings jointly. Specifically, HUARN adopts a hierarchical architecture to encode word, sentence, and document level information. Then, user attention and aspect attention are introduced into building sentence and document level representation. The document representation is combined with user and overall rating information to predict aspect ratings of a review. Diverse aspects are treated differently and a multi-task framework is adopted. Empirical results on two real-world datasets show that HUARN achieves state-of-the-art performances.

1 Introduction

The ever-increasing popularity of online consumer review platforms, such as Tripadvisor[1] and Yelp[2], has led to large amounts of online reviews that are often too numerous for users to analyze. Consequently, there is a growing need for systems analyzing reviews automatically. Lots of approaches (Xia et al., 2011; Socher et al., 2013; Tang et al., 2015a; Yang et al., 2016) usually focus on determining the overall sentiment rating of a review. Actually, not only does a review express the general attitude of reviewer, but it also conveys fine-grained sentiments towards different aspects of corresponding products. Figure 1 shows an example where *Bob* posts a review about a hotel and gives scores on overall attitude, *location*, *room*, and *service* respectively. The analysis of these aspect ratings could not only benefit mining interested aspects for users, but also help companies better understand the major pros and cons of the product. However, compared with the overall rating, users are less motivated to give aspect ratings. The reviews without aspect ratings are rampant, which are more than 46% in a simple corpus-based statistics[3]. Accordingly, it is really useful to perform document-level multi-aspect sentiment classification, whose goal is to predict ratings for different aspects in a review (Yin et al., 2017).

Multi-task learning (Caruana, 1997; Collobert et al., 2011; Luong et al., 2016) is a straightforward approach for document-level multi-aspect sentiment classification, which shares the input and hidden layers to obtain a document representation as the input of different aspect-specific classifiers. However, the representation fails to capture the differences between aspects. In fact, when we predict the sentiment rating of *service*, the first two sentences in Figure 1 are most helpful and other sentences are auxiliary or even unnecessary for classifying *service*. Therefore, aspect-specific document representation is vital for this task. To this end, Yin et al. (2017) use iterative attention module to mine aspect-specific words and sentences based on a list of aspect keywords and obtain state-of-the-art results.

[1] https://www.tripadvisor.com/
[2] https://www.yelp.com/
[3] The result is computed on 387,805 reviews crawled from https://www.tripadvisor.com/.

Proceedings of the 27th International Conference on Computational Linguistics, pages 925–936
Santa Fe, New Mexico, USA, August 20-26, 2018.

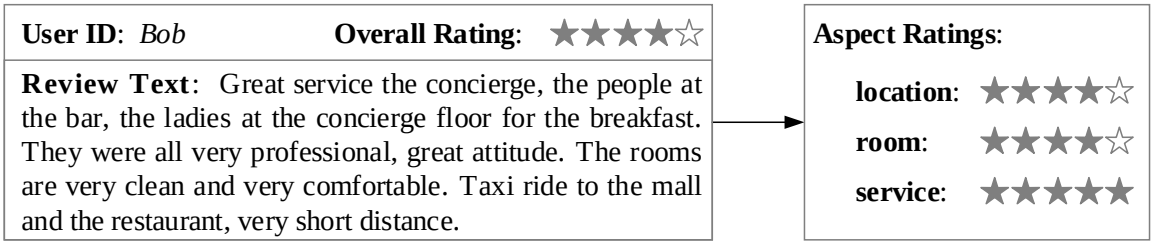


Figure 1: An example of a review. The left part is the review content, the upper right part is the reviewer *Bob* and overall rating of the review and bottom right part is different aspect ratings of the review. We focus on incorporating user preference and overall rating into review content to infer aspect ratings.

Despite the success of methods mentioned above, they typically only use text information. Two kinds of important information are ignored: users and overall ratings of reviews. The results of our statistical analysis are convincing that the two factors have strong correlations with aspect ratings (Section 2). As for users, different users may care about different aspects. When scoring aspects of a hotel, a business traveler may be critical to *service* but lenient with *price* or *room*. Such preference obviously affects the aspect ratings. Actually, many studies (Tang et al., 2015b; Chen et al., 2016; Dou, 2017) have shown that user preference can boost the performance of a related task, document-level sentiment classification that predicts an overall polarity instead of multi-aspect ratings. For our multi-aspect sentiment classification, the overall rating is given, and it can provide prior information to aspect ratings. Usually, the two types of rating are positive correlation. For example in Figure 1, the overall rating is 4 stars and the aspect ratings are all not less than 4 stars.

Inspired by the above analysis, we propose a model called Hierarchical User Aspect Rating Network (HUARN) to consider user preference and overall rating jointly for document-level multi-aspect sentiment classification. Specifically, HUARN utilizes a hierarchical structure to encode word, sentence, to document level information. Then, user and aspect information are embedded as attentions over word-level and sentence-level representation to construct a user-aspect-specific document representation. Based on the document representation, users and overall ratings are combined to express their influences on predicting aspect ratings. Finally, we adopt a multi-task framework to mutually enhance aspect rating prediction between different aspects.

In summary, our main contributions are as follows:

- For document-level multi-aspect sentiment classification, we validate the influences of users and overall ratings in terms of aspect ratings on massive Tripadvisor reviews.
- To the best of our knowledge, this is the first work to incorporate user preference and overall rating into a unified model (HUARN) in this task.
- We conduct experiments on two real-world datasets to verify the effectiveness of HUARN. The experimental results show that HUARN outperforms state-of-the-art methods significantly. The code and data for this paper are available at https://github.com/Junjieli0704/HUARN.

2 Data and Observations

In this section, we first introduce real-world datasets used in our work and present some explorations about the impacts of user preference and overall ratings on aspect ratings.

2.1 Data

We evaluate HUARN on two datasets: TripDMS and TripOUR. They are both crawled from Tripadvisor website and contain seven aspects (*value*, *room*, *location*, *cleanliness*, *check in*, *service*, and *business service*) which are provided by Tripadvisor website. The first dataset is built by Yin et al. (2017). However, there is no available user information in this dataset, thus we create the second one. Statistics

Datasets	#docs	#users	#docs/user	#words/sen	#words/doc
TripOUR	58,632	1,702	34.44	17.80	181.03
TripDMS	29,391	N/A	N/A	18.0	251.7

Table 1: Statistics of our datasets. The rating scale of TripOUR and TripDMS are 1-5.

Datasets	*value*	*room*	*location*	*cleanliness*	*check in*	*service*	*business service*
TripOUR	43,258	41,295	42,354	42,601	1,283	58,449	801
TripDMS	28,778	29,140	23,401	29,184	23,373	28,322	15,939

Table 2: The absolute number of rating of different aspects in TripOUR and TripDMS.

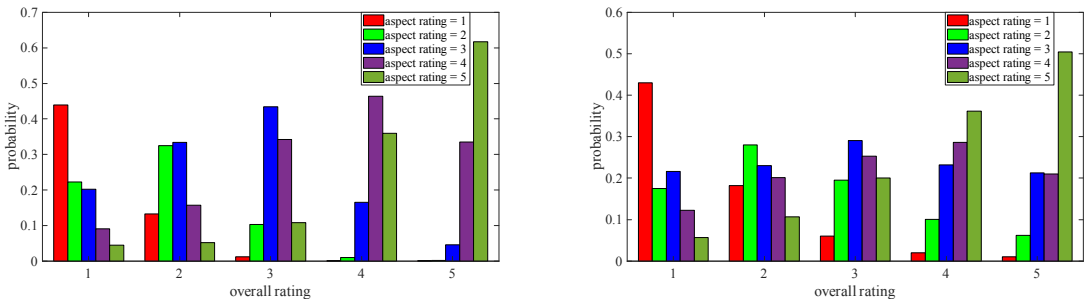
of our datasets are summarized in Table 1. Table 2 presents the absolute number of ratings of these aspects in our datasets.

2.2 Observations

Effects of user preference. Inspired by Tang et al. (2015b), we argue that the influences of users include the following two aspects: (1) *user-rating consistency*: different users have different characteristics in scoring aspects, and aspect ratings from the same user are more consistent than those from different users. (2) *user-text consistency*: different users have different word-using habits to express opinions and texts from the same user are more consistent than those from different users. To verify these consistencies, we conduct hypothesis testing as follows:

First, we construct three vectors $\mathbf{v}_s$, $\mathbf{v}_r$ and $\mathbf{v}_a$ with equal number (l) of elements. $\mathbf{v}_{s_i}$ is obtained by calculating a measurement between two reviews (d_i and d_i^+) posted by the same user, $\mathbf{v}_{r_i}$ is obtained by calculating a measurement between d_i and another random review and $\mathbf{v}_{a_i}$ is a random aspect (such as *service*), where $i \in \{1, 2, ..., l\}$.

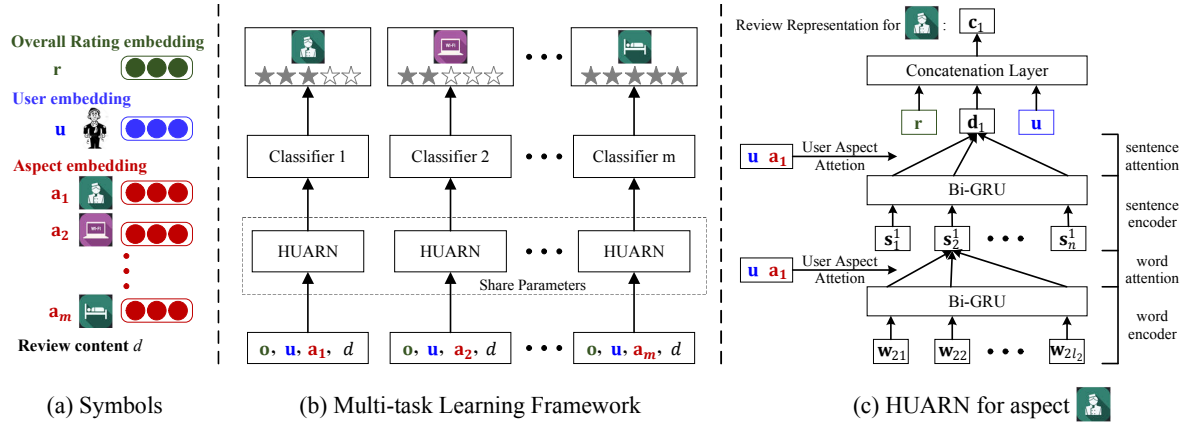
For *user-rating consistency*, the measurement is calculated by $||y - y^+||$ for $\mathbf{v}_s$ or $||y - y^-||$ for $\mathbf{v}_r$, where y, y^+, y^- is aspect rating of review d, d^+, d^- with aspect $\mathbf{v}_{a_i}$ respectively. For *user-text consistency*, the measurement is calculated by the cosine similarity between bag-of-words representation of two reviews. We perform a two-sample t-test on $\mathbf{v}_s$ and $\mathbf{v}_r$. The null hypothesis is that there is no difference between the two vectors, $H_0 : \mathbf{v}_s = \mathbf{v}_r$; the alternative hypothesis is that the difference between reviews with same user is less than with two random reviews , $H_1 : \mathbf{v}_s < \mathbf{v}_r$. The t-test results, p-values, show that there is strong evidence (with the significance level $\alpha = 0.01$) to reject the null hypothesis in *user-rating consistency* test and *user-text consistency* test on TripOUR. In other words, we observe the existence of *user-rating consistency* and *user-text consistency* in TripOUR.



(a) TripOUR (b) TripDMS

Figure 2: Aspect rating distributions for different overall ratings in TripOUR and TripDMS.

Effects of overall ratings. When scoring a product, users may consider multiple aspects of the product. If these aspects could meet the users' requirement, they can give a high overall rating, otherwise, they could give low scores. Therefore, overall rating can partly reflect the user's attitudes to aspects, which



(a) Symbols (b) Multi-task Learning Framework (c) HUARN for aspect

Figure 3: The architecture of HUARN. Left: some embedding symbols used in the figure. Example aspects are *service*, *business service*, and *room*. Middle: Multi-task learning framework for document-level multi-aspect sentiment classification. Right: the architecture of HUARN for aspect *service*.

is called *overall rating prior*. To investigate effects of overall ratings, we compute the aspect rating distributions for different overall ratings from our two datasets and the distributions are shown in Figure 2. We can conclude that high/low overall ratings often result in high/low aspect ratings. For example, when the overall rating is 5 stars, more than 70% aspect ratings are not less than 4 stars in our datasets.

3 Methods

The analysis proves users and overall ratings are significant on interpreting the sentiments of different aspects. Therefore, we introduce these two kinds of information into HUARN and detail the model here. First, we give the formalizations of document-level multi-aspect sentiment classification (Figure 3(a)). Afterwards, we discuss the multi-task learning framework for this task (Figure 3(b)) and how to obtain document semantic representation via Hierarchical Bidirectional Gated Recurrent Unit network. At last, we present user and aspect attention mechanism to construct user-aspect-specific representation and add a concatenating layer to combine user, overall rating and document representation together (Figure 3(c)). The enhanced document representation is used as features for predicting aspect ratings.

3.1 Formalizations

Suppose we have a corpus D about a specific domain (such as "hotel") and m aspects $\{a_1, a_2, ..., a_m\}$ (such as *service* and *room*). Review d is a sample of D with n sentences $\{s_1, s_2, ..., s_n\}$. Sentence s_i consists of l_i words as $\{w_{i1}, w_{i2}, ..., w_{il_i}\}$. The overall rating of review d is r and its author is user u. Document-level multi-aspect sentiment classification aims to predict aspect ratings for these reviews.

3.2 Multi-task Learning Framework

It is natural to model document-level multi-aspect sentiment classification as a multi-task learning. First, we can treat each aspect rating as a classification task. Then, we share document encoder network to obtain document representation and exploits different softmax classifiers to predict ratings of different aspects. The main benefit of the multi-task framework is that it can mutually enhance aspect rating prediction between different aspects.

3.3 Hierarchical Bidirectional Gated Recurrent Network

Since a document is composed of multiple sentences, and a sentence is composed of multiple words, we model the semantics of a document through a hierarchical structure from word-level, sentence-level to document-level. To model the semantic representation of a sentence, we adopt bidirectional GRU (Bi-GRU). Similarly, we also use Bi-GRU to learn document representations.

Given sentence s_i, we embed each word w_{ij} to vector $\mathbf{w}_{ij}$. Then, we use a Bi-GRU to encode contextual information of word w_{ij} into its hidden representation $\mathbf{h}_{ij}$. Hidden states $\{\mathbf{h}_{i1}, \mathbf{h}_{i2}, ...\mathbf{h}_{il_i}\}$ are feed

into an average pooling layer to obtain the sentence representation $\mathbf{s}_i$. In sentence level, we also feed the sentence vectors $\{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_n\}$ into Bi-GRU and then obtain the document representation $\mathbf{d}$ similarly.

3.4 Encoding user, aspect, and overall rating

It is obvious that not all words (sentences) contribute equally to the sentence (document) meaning. To consider *user-text consistency* and build an aspect-specific representation, we introduce user attention and aspect attention. Specifically, we employ word (sentence) level user aspect attention to generate sentence (document) representation.

Word-level Attention. We first embed user u and aspect $\{a_k|k \in 1, 2, ..., m\}$ as continuous and real-valued vector $\mathbf{u}$ and $\mathbf{a}_k$. Then, instead of feeding word-level hidden states ($\mathbf{h}_{ij}$) to an average pooling layer, we adopt a user aspect attention mechanism to extract user-aspect-specific words and obtain the sentence representation as follows:

$$\mathbf{m}_{ij} = \tanh(\mathbf{W}_{wh}\mathbf{h}_{ij} + \mathbf{W}_u\mathbf{u} + \mathbf{W}_a\mathbf{a}_k + \mathbf{b}_w) \tag{1}$$

$$\alpha_{ij} = \frac{\exp(\mathbf{v}_w^T\mathbf{m}_{ij})}{\sum_j \exp(\mathbf{v}_w^T\mathbf{m}_{ij})} \tag{2}$$

$$\mathbf{s}_i^k = \sum_j \alpha_{ij}\mathbf{h}_{ij} \tag{3}$$

where $\mathbf{W}_{wh}, \mathbf{W}_{wu}, \mathbf{W}_{wa}$ and $\mathbf{b}_w$ are parameters in the attention layer. α_{ij} measures the importance of the j-th word for user u and aspect a_k and $\mathbf{s}_i^k$ is the representation of sentence s_i for aspect a_k.

Sentence-level Encoder and Attention. After obtaining sentence representation $\mathbf{s}_i^k$ for aspect a_k, we also use a Bi-GRU to encode the sentences and get hidden representation $\mathbf{h}_i^k$ for $\mathbf{s}_i^k$.

When classifying document based on different aspects, different sentences may have different influences. Different users may also pay attention to different sentences. Therefore, in sentence level, we also apply an attention mechanism with user vector $\mathbf{u}$ and aspect vector $\mathbf{a}_k$ in sentence level to select informative sentences to compose user-aspect-specific document representation. The document representation $\mathbf{d}_k$ for aspect a_k is obtained via:

$$\mathbf{t}_i = \tanh(\mathbf{W}_{sh}\mathbf{h}_i^k + \mathbf{W}_{su}\mathbf{u} + \mathbf{W}_{sa}\mathbf{a}_k + \mathbf{b}_s) \tag{4}$$

$$\beta_i = \frac{\exp(\mathbf{v}_s^T\mathbf{t}_i)}{\sum_i \exp(\mathbf{v}_s^T\mathbf{t}_i)} \tag{5}$$

$$\mathbf{d}_k = \sum_i \beta_i\mathbf{h}_i^k \tag{6}$$

where β_i measures the importance of the i-th sentences for user u and aspect a_k.

Concatenation Layer. To explicitly encode *user-rating consistency* and *overall rating prior*, we add a concatenation layer. First, we embed overall rating r as continuous and real-valued vector $\mathbf{r}$ with g_r dimensions. Then, we generate review content representation $\mathbf{c}_k$ by concatenating user embedding $\mathbf{u}$, rating embedding $\mathbf{r}$ and document vector $\mathbf{d}_k$:

$$\mathbf{c}_k = \mathbf{u} \oplus \mathbf{r} \oplus \mathbf{d}_k \tag{7}$$

3.5 Document-level Multi-aspect Sentiment Classification

For each aspect, we obtain review representation $\{\mathbf{c}_k|k \in 1, 2, ..., m\}$. All these representations are high-level representations of the combination of user, aspect, overall rating and document information. It can be used as features for predicting aspect ratings. For aspect a_k, we can use a softmax layer to project $\mathbf{c}_k$ into sentiment distribution $\mathbf{p}(d, k)$ over L classes:

$$\mathbf{p}(d, k) = \text{softmax}(\mathbf{W}_{lk}\mathbf{c}_k + \mathbf{b}_k) \tag{8}$$

where $p_l(d,k)$ is used to represent the predicted probability of sentiment class l for d based on a_k and $\mathbf{W}_{lk}, \mathbf{b}_k$ are parameters of softmax layer for classifying review $\mathbf{c}_k$. Then we define the cross-entropy error between gold sentiment distribution and our model's sentiment distribution as our loss function :

$$L = -\sum_{d \in D} \sum_{k \in \{1,2,\ldots,m\}} \sum_{l=1}^{L} \mathbb{1}\{g_{d,k} = l\} \cdot \log(p_l(d,k)) \tag{9}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function and $g_{d,k}$ represents the ground truth label for review d for aspect a_k.

4 Experiments

In this section, we present data preprocessing and implementation details, all the comparison methods and the empirical results on the task of document-level multi-aspect sentiment classification.

4.1 Data Preprocessing and Implementation Details

We preprocess our datasets as follows: For TripDMS, we use the same splitting method as (Yin et al., 2017). For TripOUR, we tokenize the dataset, split sentences by Stanford CoreNLP (Manning et al., 2014) and randomly split them into training, development, and testing sets with 80/10/10%.

The model hyper-parameters are tuned based on the development sets. For word embeddings, we use the pre-trained word embeddings provided by (Yin et al., 2017), whose embedding size is 200. For user and overall rating embeddings, we initialize them randomly and set their dimensions to 200. For aspect embeddings, we first get aspect keywords[4] from (Yin et al., 2017) and initial aspect embedding by averaging word embeddings of these keywords belong to the aspect. The dimensions of all hidden vectors are set to 150. To avoid model over-fitting, we use dropout with rate of 0.2. All the parameters are trained using Adam (Kingma and Ba, 2014) with a learning rate of 0.001.

4.2 Comparison Methods

We compare HUARN with the following baselines:

Majority is a heuristic baseline method, which assigns the majority sentiment category in the training set to aspect rating in the test dataset.

OverallRatingSame is also a heuristic baseline method, which assigns the overall rating of a review to its aspect ratings.

MajOverallRating splits the training instances into five clusters (per overall rating) and assigns the most frequent rating for the seven aspects per cluster in the test dataset.

SVM and **NBoW** are SVM classifiers with different features. One with unigrams, bigrams as features and another with the mean of word embeddings in a document as features.

CNN (Kim, 2014) performs a convolution operation over a sentence to extract words neighboring features, then gets a fixed-sized representation by a pooling layer.

HAN (Yang et al., 2016) models review in a hierarchical structure and utilizes an attention mechanism to capture important words and sentences, which is only based on text information and achieves state-of-the-art result in predicting overall rating of document.

MHCNN is an extended model of **CNN** with hierarchical architecture and multi-task framework.

MHAN is an extended model of **HAN** with multi-task framework.

DMSCMC (Yin et al., 2017) use iterative attention modules to build up aspect-specific representation for review, and obtain state-of-the-art results in document-level multi-aspect sentiment classification.

HGRUN is the basic form of **HUARN** without user, aspect and overall rating.

HARN is a variant of **HUARN**, which abandons user information from **HUARN**.

4.3 Results

We use Accuracy and Mean Squared Error (MSE) as the evaluation metrics, and the results are shown in Table 3. For heuristic methods, we can see that **Majority** performs very poor because it does not

[4]Sample keywords for *service* are service, food, breakfast, and buffet.

Models	TripOUR		TripDMS	
	Accuracy↑	MSE↓	Accuracy↑	MSE↓
Majority	0.3850	0.954	0.2389†	2.549†
OverallRatingSame	0.3074	1.705	0.2012	3.446
MajOverallRating	0.3487	1.536	0.2414	3.273
SVM	0.4635	1.025	0.3526†	1.963†
NBoW	0.4865	0.912	0.3909†	1.808†
CNN	0.5054	0.752	0.4335†	1.456†
HAN	0.5123	0.705	0.4468†	1.301†
MHCNN	0.5108	0.712	0.4379†	1.398†
MHAN	0.5419	0.629	0.4494†	1.210†
HGRUN	0.5392	0.635	0.4435	1.303
DMSCMC	0.5549	0.583	0.4656†	1.083†
HARN	0.5815*	0.528	**0.4821***	**0.923**
HUARN	**0.6070***	**0.514**	N/A	N/A

Table 3: Document-level multi-aspect sentiment classification on our datasets. Our full model is HUARN. The best performances in **bold**. "†" indicates that the result is reported from (Yin et al., 2017). "*" indicates that the model significantly outperforms DMSCMC. Statistical significance testing has been performed using paired t-test with $p < 0.05$.

capture any text information. **OverallRatingSame** and **MajOverallRating** are also very poor, even though overall rating has strong correction with aspect ratings, it is not enough to decide aspect ratings only based on it.

Compared with **SVM**, **NBoW** achieves higher accuracy by at least 2.3% in both datasets, which shows that embedding features are more effective than unigram and bigram features on these two datasets. When applying more complex neural networks (such as **CNN** and **HAN**), the model can achieve higher accuracy by at least 1.5% in both datasets compared with **NBoW**. Additionally, we observe that the multi-task learning and hierarchical architecture are beneficial for neural networks. Performance on **MHAN** and **MHCNN** are slightly better than **HAN** and **CNN**. Beyond that, we also find attention mechanism is useful. The only difference between **MHAN** and **HGRUN** is that **MHAN** uses attention mechanism to obtain sentence and document representations while **HGRUN** utilizes an average pooling layer, which results in the performance of **MHAN** is better than **HGRUN**. After obtaining aspect-aware representation for the document, **DMSCMC** achieves best results and outperforms other baselines.

Compared to **DMSCMC**, **HARN** achieves improvements of 2.7% and 1.7% on TripOUR and TripDMS respectively, which shows that the incorporation of overall rating and aspect attention helps build up more discriminative representation. Moreover, when incorporating user information, our full model (**HUARN**) can achieve improvements of 5.3% compared with **DMSCMC** on TripOUR[5], which shows user preference can benefit the document-level multi-aspect sentiment classification task.

5 Discussions

In this section, we first give some discussions about the effects of users, aspects and overall ratings on predicting aspect ratings, and then show case study for attention results and visualize user embeddings.

5.1 Effects of Users, Overall Ratings and Aspects

Users, overall ratings and aspects are three kinds of information in HUARN. We present the effects of users, overall ratings, and aspects on document-level multi-aspect sentiment classification in Table 4. From the table, we can observe that: (1) Compared with user-agnostic models (line 1-4), user-aware models (line 5-8) can achieve improvements of 2.2%, 2.5%, 1.2% and 2.5% in TripOUR, which shows

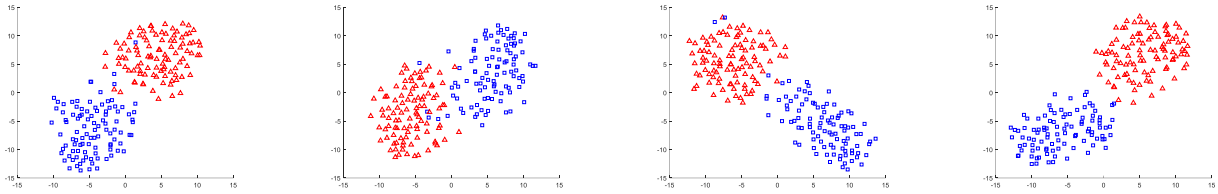
[5]Since there is no user information in TripDMS, we can only compare **DMSCMC** with **HARN**.

No.	Different information			TripOUR		TripDMS	
	User	**OverallRating**	**Aspect**	Accuracy↑	MSE↓	Accuracy↑	MSE↓
1	–	–	–	0.5392	0.635	0.4435	1.303
2	–	–	✓	0.5514	0.599	0.4566	1.256
3	–	✓	–	0.5719	0.555	0.4740	1.093
4	–	✓	✓	0.5815	0.528	0.4821	0.923
5	✓	–	–	0.5640	0.619	N/A	N/A
6	✓	–	✓	0.5764	0.581	N/A	N/A
7	✓	✓	–	0.5839	0.560	N/A	N/A
8	✓	✓	✓	0.6070	0.514	N/A	N/A

Table 4: Effects of user, overall rating and aspect on document-level multi-aspect sentiment classification. Each line represents a variant of HUARN, where "✓" denotes the variant considers the specific information, while "–" denotes not. For example, model in line 1 means HUARN abandons these three kinds of information and degenerates into HGRUN.

that model encoding user information can obtain user-aware document representation and is more suitable for document-level multi-aspect sentiment classification. (2) Compared with models without overall rating information (line 1, 2, 5 and 6), models considering overall ratings (line 3, 4, 7, 8) can obtain 3.2% (3.1%), 3.0% (2.6%), 1.9% (N/A) and 3.1% (N/A) improvements in accuracy in both datasets, which indicates overall rating information is useful for building more discriminative document representation and helpful for predicting aspect ratings. (3) Compared with models without aspect information (line 1, 3, 5 and 7), aspect-based models (line 2, 4, 6, 8) can obtain 1.2% (1.3%), 1.0% (0.8%), 1.2% (N/A) and 2.4% (N/A) improvements in accuracy in both datasets. It shows that aspect information is useful for building aspect-aware document representation and helpful for predicting aspect ratings. (4) After users, overall ratings, and aspects being considered jointly, our model obtains the best performance.

5.2 Visualization of User Embeddings



(a) Service (b) Value (c) Cleanliness (d) Location

Figure 4: t-SNE visualization of user embeddings for different aspects in TripOUR. Blue square and red triangle represent users are "High Score" users and "Low Score" users, respectively.

As different users have different aspect rating preferences and HUARN imports user embedding to consider users, we identify whether such personalized information are encoded in user embedding. To this end, we first rank all users according to their average score in the training set for each aspect. Then the top 100 users are labeled as "High Score" users and bottom 100 users are labeled as "Low Score" users. Due to space limit, here we only show embeddings of users in four aspects (*service*, *value*, *cleanliness*, *location*), which are top frequently scored by all users, in Figure 4. We find "High Score" users and "Low Score" users are separated apparently. The visualization shows that user embedding learned by HUARN can encode personalized traits in scoring different aspects.

5.3 Case Study for Attention Results

To show the ability that HUARN captures user preference and aspect semantic meanings, we take one sentence from TripOUR as example. The content of the sentence is "The food is **good**, but the price is

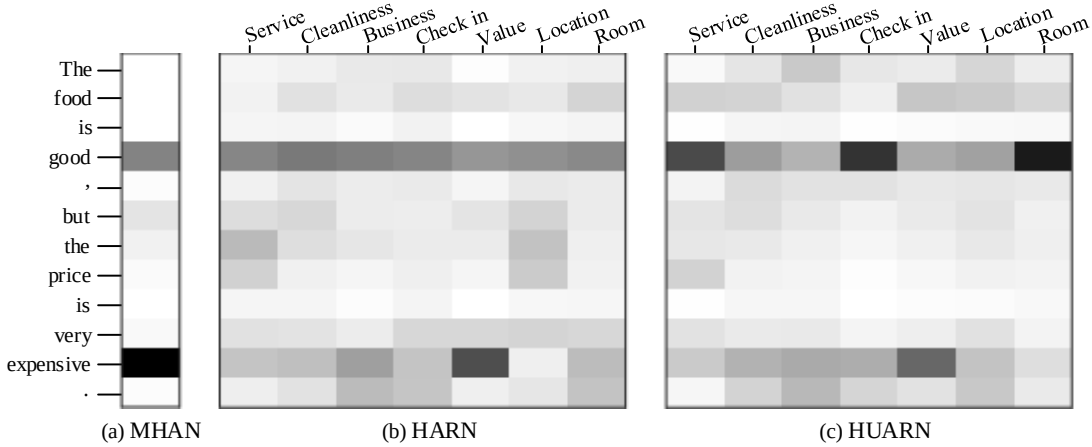


Figure 5: The attention visualization of words. Dark color means higher weight. (a), (b) and (c) show word-level attention weights of **MHAN**, **HARN** and **HUARN**.

very **expensive**", in which "good" is a general sentiment word and can be used to describe many aspects (such as *service*, *room* and so on), while "expensive" is a aspect-specific sentiment word which only applies to describe *value*. We visualize attention weights of the sentence in Figure 5.

From Figure 5(a), we can find that considering word-level attention, **MHAN** distinguishes sentiment words and non-sentiment words, however it is hard to identify the sentiment word is a general one or an aspect-specific one. After adding aspect attention over word-level representations, **HARN** can distinguish these two kinds of sentiment words (Figure 5(b)). The attention weights of "good" for different aspects are very close, while the attention weights of "expensive" for different aspects are different, and the maximum is *value*. When adding user information into word-level attention, **HUARN** can also treat "good" differently. From figure 5(c), we can find attention weights of "good" are different for different aspects, where weights for *service*, *check in* and *room* are higher than weights for other aspects. we check all reviews of the sample review's author and find that he/she often (more than 80%) use "good" to describe *service*, *check in* and *room*.

5.4 Error Analysis

Figure 6: Examples of error cases. GT means ground trouth and P means prediction result of HUARN. Sentences in review text with darker color means higher attention weight for the sentence.

We analyze error cases in the experiments. Some examples of error cases are shown in Figure 6. We can find that HUARN is hard to select important sentences for aspects. For example, sentence "the rooms are large and the bed comfy." and sentence "however the carpet should have been replaced long time." in the first sample in Figure 6 are all important to decide the rating of aspect *room*. However, HUARN pays more attention to the former sentence when predicting rating of *room* and obtains the wrong result.

Based on the literature study, we find that Yin et al. (2017) uses iterative attention models to build up aspect-specific representation for review. It may alleviate this problem. We leave how to encode user and overall rating information into DMSCMC as our future work.

6 Related Work

Multi-aspect sentiment classification is an extensively studied task in sentiment analysis (Pang and Lee, 2008; Liu, 2012). Lu et al. (2011) propose Segmented Topic Model to model document and extract features, then exploit support vector regression to predict aspect ratings based on these features. McAuley et al. (2012) add a dependency term in final multi-class SVM objective to consider the correction between aspects. Many other studies (Titov and McDonald, 2008; Wang et al., 2010; Wang et al., 2011; Diao et al., 2014; Pappas and Popescu-Belis, 2014; Pontiki et al., 2016; Toh and Su, 2016) solve multi-aspect sentiment classification as a subproblem by utilizing heuristic based methods or topic models. However, these approaches often rely on strict assumptions about words and sentences, for example, word syntax has been used to distinguish aspect word or sentiment word, or appending an specific aspect to a sentence. Another related problem is called aspect-level sentiment classification (Pontiki et al., 2014; Dong et al., 2014; Wang et al., 2016; Tang et al., 2016; Schouten and Frasincar, 2016). Wang et al. (2016) and Tang et al. (2016) employ attention-based LSTM and deep memory network for aspect-level sentiment classification, respectively. However, the task is sentence level. Document-level sentiment classification (Li and Zong, 2008; Li et al., 2010; Li et al., 2013; Xia et al., 2015; Yang et al., 2016) is also a related research field because we can treat single aspect sentiment classification as an individual document classification task. However, they did not consider multiple aspects in a document.

In addition to these methods, the work of Yin et al. (2017) is the most related to ours, which focuses on using iterative attention mechanism to build discriminative aspect-aware representation to perform document-level multi-aspect sentiment classification. However, it ignores the influences of users and overall ratings on aspect ratings. Actually, many studies (Tang et al., 2015b; Tang et al., 2015c; Chen et al., 2016; Li et al., 2016; Dou, 2017) have shown considering user preference can boost the performance of Document-level Sentiment Classification. Partially inspired by these approaches, we propose HUARN to consider users, overall ratings and aspects jointly into document-level multi-aspect sentiment classification. Compared with these user-aware approaches, HUARN has some differences: (1) They do not consider aspects. (2) Although Chen et al. (2016) and Dou (2017) embedding user to consider user-text consistency to perform sentiment classification, they ignore user-rating consistency. (3) Tang et al. (2015b; 2015c) embed user in a matrix and build user-specific representation by a convolutional neural network structure. However, it is hard to train with limited reviews for user matrix. Our motivation is that (1) Aspect information is very useful for selecting informative words and sentences and building up aspect-specific representation for document-level multi-aspect sentiment classification, therefore, we add aspect attention into our model. (2) Compared with user-text consistency, user-rating consistency describes the correlation between users and ratings more directly. (3) Embedding user in a vector and using attention mechanism to build user-specific representation is an effective way to consider users. User embedding is enough to encode the relation between user and rating in The most important is that it is easy to train.

7 Conclusion and Future work

In this paper, we present Hierarchical User Aspect Rating Network (HUARN) to incorporate user preference and overall rating into document-level multi-aspect sentiment classification. HUARN encodes different kinds of information (word, sentence and document) into a hierarchical structure. To consider user preference and overall rating, HUARN introduces user information as attention over word-level representation and sentence-level representation, and then generates review representation by combining user, overall rating and document information. Extensive experiments show that our model outperforms state-of-the-art methods significantly. In the future, we will study how to encode user and overall rating information into DMSCMC.

Acknowledgements

We thank Xiaomian Kang and Yang Zhao for valuable discussions. We also thank the anonymous reviewers for their suggestions. The research work descried in this paper has been supported by the Natural Science Foundation of China under Grant No. 61333018 and 61673380.

References

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

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. In *Proceedings of EMNLP*.

Ronan Collobert, Jason Weston, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(1):2493–2537.

Qiming Diao, Minghui Qiu, Chao Yuan Wu, Alexander J. Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proceedings of KDD*, pages 193–202.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of ACL*, pages 49–54.

Zi-Yi Dou. 2017. Capturing user and product information for document level sentiment analysis with deep memory network. In *Proceedings of EMNLP*, pages 532–537, Copenhagen, Denmark, September. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Eprint Arxiv*.

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

Shoushan Li and Chengqing Zong. 2008. Multi-domain sentiment classification. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA, Short Papers*, pages 257–260.

Shoushan Li, Chu-Ren Huang, Guodong Zhou, and Sophia Yat Mei Lee. 2010. Employing personal/impersonal views in supervised and semi-supervised sentiment classification. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 414–423.

Shoushan Li, Yunxia Xue, Zhongqing Wang, and Guodong Zhou. 2013. Active learning for cross-domain sentiment classification. In *Proceedings of IJCAI*, pages 2127–2133.

Junjie Li, Haitong Yang, and Chengqing Zong. 2016. Sentiment classification of social media text considering user attributes. In *Natural Language Understanding and Intelligent Applications - 5th CCF Conference on Natural Language Processing and Chinese Computing, NLPCC 2016, and 24th International Conference on Computer Processing of Oriental Languages, ICCPOL 2016, Kunming, China, December 2-6, 2016, Proceedings*, pages 583–594.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Bin Lu, Myle Ott, Claire Cardie, and Benjamin K Tsou. 2011. Multi-aspect sentiment analysis with topic models. In *Proceedings of ICDM Workshops*, pages 81–88.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proceedings of ICLR*, San Juan, Puerto Rico, May.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Julian Mcauley, Jure Leskovec, and Dan Jurafsky. 2012. Learning attitudes and attributes from multi-aspect reviews. In *Proceedings of ICDM*, pages 1020–1025.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Nikolaos Pappas and Andrei Popescu-Belis. 2014. Explaining the stars: Weighted multiple-instance learning for aspect-based sentiment analysis. In *Proceedings of EMNLP*, pages 455–466. Association for Computational Linguistics.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. *Proceedings of International Workshop on Semantic Evaluation at*, pages 27–35.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphee De Clercq, Veronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeniy Kotelnikov, Núria Bel, Salud María Jiménez-Zafra, and Gülşen Eryiğit. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 19–30. Association for Computational Linguistics.

Kim Schouten and Flavius Frasincar. 2016. Survey on aspect-level sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, 28(3):813–830.

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

Duyu Tang, Bing Qin, and Ting Liu. 2015a. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of EMNLP*, pages 1422–1432, Lisbon, Portugal, September. Association for Computational Linguistics.

Duyu Tang, Bing Qin, and Ting Liu. 2015b. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of ACL*, pages 1014–1023, July.

Duyu Tang, Bing Qin, Yuekui Yang, and Yuekui Yang. 2015c. User modeling with neural network for review rating prediction. In *Proceedings of IJCAI*, pages 1340–1346.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. *arXiv preprint arXiv:1605.08900*.

Ivan Titov and Ryan McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of ACL*, pages 308–316. Association for Computational Linguistics.

Zhiqiang Toh and Jian Su. 2016. Nlangp at semeval-2016 task 5: Improving aspect based sentiment analysis using neural network features. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 282–288. Association for Computational Linguistics.

Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of KDD*, pages 783–792.

Hongning Wang, Yue Lu, and Cheng Xiang Zhai. 2011. Latent aspect rating analysis without aspect keyword supervision. In *Proceedings of KDD*, pages 618–626.

Yequan Wang, Minlie Huang, Li Zhao, and Xiaoyan Zhu. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of EMNLP*.

Rui Xia, Chengqing Zong, and Shoushan Li. 2011. Ensemble of feature sets and classification algorithms for sentiment classification. *Information Sciences*, 181(6):1138–1152.

Rui Xia, Feng Xu, Chengqing Zong, Qianmu Li, Yong Qi, and Tao Li. 2015. Dual sentiment analysis: Considering two sides of one review. *IEEE Transactions on Knowledge and Data Engineering*, 27(8):2120–2133.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. 2016. Hierarchical attention networks for document classification. In *HLT-NAACL*, pages 1480–1489.

Yichun Yin, Yangqiu Song, and Ming Zhang. 2017. Document-level multi-aspect sentiment classification as machine comprehension. In *Proceedings of EMNLP*, pages 2044–2054. Association for Computational Linguistics.

Leveraging Meta-Embeddings for Bilingual Lexicon Extraction from Specialized Comparable Corpora

Amir Hazem **Emmanuel Morin**

LS2N - UMR CNRS 6004, Université de Nantes, France
{amir.hazem,emmanuel.morin}@univ-nantes.fr

Abstract

Recent evaluations on bilingual lexicon extraction from specialized comparable corpora have shown contrasted performance while using word embedding models. This can be partially explained by the lack of large specialized comparable corpora to build efficient representations. Within this context, we try to answer the following questions: First, (i) among the state-of-the-art embedding models, whether trained on specialized corpora or pre-trained on large general data sets, which one is the most appropriate model for bilingual terminology extraction? Second (ii) is it worth it to combine multiple embeddings trained on different data sets? For that purpose, we propose the first systematic evaluation of different word embedding models for bilingual terminology extraction from specialized comparable corpora. We emphasize how the character-based embedding model outperforms other models on the quality of the extracted bilingual lexicons. Further more, we propose a new efficient way to combine different embedding models learned from specialized and general-domain data sets. Our approach leads to higher performance than the best individual embedding model.

1 Introduction

Bilingual lexicons are fundamental resources in multilingual natural language processing tasks such as machine translation (Och and Ney, 2003), cross-language information retrieval (Nie, 2010) or computer-assisted translation (Delpech, 2014). Because a manual compilation of bilingual lexicons requires substantial human efforts, bilingual lexicons are automatically extracted from bilingual corpora. These corpora can be parallel or comparable data sets. Despite good results obtained when compiling bilingual lexicons from parallel corpora, the latter are scarce resources, especially for specialized and technical domains and for language pairs not involving English. In this context, comparable corpora are an interesting and practical alternative to the use of parallel corpora.

Comparable corpora, which gather texts sharing common features such as domain, topic, discourse, etc. without having a parallel source text-target text relationship, allow access to the original vocabulary without falling under the influence of the human translation. Compiling a large comparable corpus is easier, especially for general language (Talvensaari et al., 2007). In contrast, specialized comparable corpora are traditionally of modest size due to the difficulty to obtain many specialized documents in a language other than English. Specialized comparable corpora have a size of around one million words whereas general-domain comparable corpora can gather several million words (Morin and Hazem, 2014).

One way to overcome the small size of specialized comparable corpora is to associate external resources. These resources may be close specialized corpora (e.g. a breast cancer corpus may benefit from contexts derived from a more general oncology corpus), corpora of different types of discourse and gender (e.g. a corpus of popular science discourse supplementing a corpus of scientific discourse), corpora of general language or out-of-domain data. The main challenge is to know how to associate such resources with a comparable specialized corpus.

Proceedings of the 27th International Conference on Computational Linguistics, pages 937–949
Santa Fe, New Mexico, USA, August 20-26, 2018.

According to Jakubina and Langlais (2017), word embeddings are more effective on large comparable corpora than on small comparable corpora. This statement lend support the idea that enriching small specialized comparable corpora may be beneficial to bilingual terminology extraction task. The combination of external resources such as a general-domain comparable corpus with a specialized comparable corpus can be performed using word embedding models. We recently conducted a first attempt in Hazem and Morin (2017) and have shown under which conditions external resources introduced in the form of Skip-gram and CBOW models can be jointly used to improve the performance of bilingual terms extraction. However, our approach was not able to compete with the historical count-based projection approach (Fung and McKeown, 1997; Rapp, 1999). Our current work pursues this direction by contrasting different neural embedding models and by showing how to take advantage of their combination. More specifically, we show that the character-based Skip-gram and CBOW models (Bojanowski et al., 2016) drastically outperform other models including Skip-gram and CBOW. We also propose a new approach based on Ensemble models which combines specialized and general domain embeddings to obtain a unified Meta-Embedding model. Our approach shows significant improvements and obtains the best results on two specialized English/French comparable corpora.

2 Related Work

According to Hermann and Blunsom (2014), methods dealing with bilingual lexicon extraction from comparable corpora can be classified as distributional-based or distributed-based approaches. In the former, words are represented by their context vectors using a distributional count-based approach also known as the standard approach (Fung, 1998; Rapp, 1999). While in the latter, words are embedded into a low-dimensional vector space using neural network models (Bengio et al., 2003; Mikolov et al., 2013b).

The historical standard approach builds a context vector for each word of the source and the target languages, translates the source context vectors to the target language using a bilingual seed lexicon, and compares the translated context vectors to each target context vector using a similarity measure. Different contributions have been proposed in the past few years to improve each step of the standard approach (Gaussier et al., 2004; Gamallo, 2007; Ismail and Manandhar, 2010; Prochasson and Fung, 2011; Hazem and Morin, 2012; Bouamor et al., 2013, among others).

With the advent of neural network techniques, Mikolov et al. (2013a) were the first to propose a method to learn a linear transformation from the source language into the target language to improve the task of lexicon extraction from bilingual corpora. Faruqui and Dyer (2014) introduced canonical correlation analysis (CCA) to project the embeddings in both languages to a shared vector space. More recently, Artetxe et al. (2016) presented an approach for learning bilingual mappings of word embeddings that preserves monolingual invariance using several meaningful and intuitive constraints related to other proposed methods (Faruqui and Dyer, 2014; Xing et al., 2015).

Jakubina and Langlais (2017) made a careful comparison of the approaches of Mikolov et al. (2013a) and Faruqui and Dyer (2014) with the standard approach. They have clearly shown that the two previous approaches outperform the standard approach for very frequent terms to be translated (the number of occurrences is at least 250 from an English-Spanish comparable corpus obtained from the 6th workshop on statistical machine translation gathering 2.55 giga words). On the other hand, when the terms are less frequent (the number of occurrences is less than 25 from a French/English comparable corpora built from the Wikipedia dumps gathering 1.53 giga words), the standard approach slightly outperforms the two previous embedding approaches. More recently, we have shown under which conditions Skip-gram and CBOW models can be jointly used to improve the performance of bilingual terms extraction from specialized comparable corpora without exceeding the results of the standard approach (Hazem and Morin, 2017).

Other works performed bilingual word representation without word-to-word alignments of comparable corpora. Chandar et al. (2014) and Gouws et al. (2014) for instance used multilingual word embeddings based on sentence-aligned parallel data whereas Vulić and Moens (2015) and Vulić and Moens (2016) used document-aligned non-parallel data to produce bilingual word embeddings. Theses works are based

on sentence- or document-aligned of general-domain comparable corpora and are outside the scope of this study. It is unlikely to find this type of alignments in a specialized comparable corpus.

3 Embedding Models

In this section, we briefly recall the main and recent word embedding models that we will investigate in this study.

CBOW and Skip-gram are two distributed representations introduced by Mikolov et al. (2013b) that capture linguistic regularities, namely the Continuous Bag-of-Words (CBOW) model and the Skip-gram model. The principle of the CBOW model is to combine the representations of surrounding words to predict the word in the middle, while the training objective of the Skip-gram model is to learn how to predict the surrounding words based on the representations of the middle word. If these models exhibit similar architectures, CBOW is faster and more suitable for large data sets while Skip-gram gives better word representations when monolingual data is small (Mikolov et al., 2013a).

Glove takes advantage of the main benefits of count data while simultaneously capturing the meaningful linear substructures prevalent in prediction-based methods such as word2vec. It is a global log-linear regression model that makes use of a global factorization model and local context window methods to represent words in a global vector space model (Pennington et al., 2014). This model directly captures the global statistics from the corpus based on co-occurrence word probabilities. Its training objective is to learn word vectors such that their dot product equals the log-probability of word's co-occurrence. Glove has shown good results in word analogy, word similarity, and named entity recognition tasks.

Structured Embeddings are two adaptations of CBOW and Skip-gram models that include ordering information[1] (Ling et al., 2015). While word2vec is insensitive to word order, the structured embedding model includes position information in the context representation of words. Given the embedding of the center word w, the Skip-gram model for instance uses a single output matrix to predict every contextual word. In contrast, the structured Skip-gram adapts the model to the positioning of the surrounding words. It defines an output for each relative position to the center word. The adaptation of CBOW is the continuous window model where the input is the concatenation of the embeddings of context words. While in the standard CBOW, the input model is the sum of the embeddings of the context words.

Character n-gram Embeddings is an enhanced variant of the Skip-gram and CBOW models that enrich word vectors with subwords information. It takes into account the internal structure of words which can be very useful for morphologically rich languages. It also incorporates character n-gram embeddings where each word is represented by a bag-of character n-gram (Bojanowski et al., 2016). More precisely, it uses character embedding and word embedding models jointly performing the vector sum of both to form the final embedding representation of words. We refer to the character Skip-gram model by CharSG and the character CBOW model by CharCBOW.

Other models such as the dependency-based model (Levy and Goldberg, 2014) and generalized-based model (Li et al., 2017) were assessed but not presented in this paper for sake of clarity and because of the very low results obtained on the specialized domains when compared to the above presented models.

Several pre-trained embedding models are publicly available such as CBOW and Skip-gram models (Mikolov et al., 2013b), global word representation-based models (Pennington et al., 2014), character skip-gram-based models (Bojanowski et al., 2016), etc. If it is interesting to study the impact of pre-trained embeddings on bilingual terminology extraction from comparable corpora. The major part of the above cited pre-trained embedding models exist solely in English. We only use the character skip-gram (CharSG) model (Bojanowski et al., 2016) which is available in both French and English.

[1] Word order in context word representation.

4 Approach

The task of bilingual terminology extraction from specialized comparable corpora consists of acquiring for each term of the source language its translation in the target language. The basic idea using embedding models is to first (1) build word embeddings of the source and the target languages, then, (2) to build a mapping matrix (Artetxe et al., 2016) that allows to obtain for each source term, its representation in the target language and finally, (3) to measure the similarity between the target representation of the source term and all the word candidates of the target language to extract the most similar term as the correct translation (Mikolov et al., 2013a; Artetxe et al., 2016). Our approach follows these three steps (Mikolov et al., 2013a) while acting at the word embedding level representation. Our idea is to enrich the word embedding representation of the source and target languages in order to improve the mapping matrix and so, bilingual terminology extraction from specialized comparable corpora. To do so, we present several ways to take advantage of word embedding models and ensemble approaches.

The principle of ensemble approaches is to combine different models in order to capture the strengths of each individual model. The main combination techniques that have shown their effectiveness so far are vectors addition (Garten et al., 2015) and vectors concatenation (Garten et al., 2015; Yin and Schütze, 2016). For vectors addition, given two embedding models, the procedure consists in applying a simple dimension-wise vectors addition[2]. For vectors concatenation, given two embedding models of dimensions $dim1$ and $dim2$, the resulting concatenated embedding vector will be of size $dim1+dim2$. The vectors have to be normalized before concatenation. Usually L2 norm is applied[3]. Yin and Schütze (2016) performed a weighted concatenation of five embedding models. They also experienced the SVD (Singular Value Decomposition) on top of weighted concatenation vectors of dimension 950. This resulted in a reduced model of 200 dimensions.

In the line of the above cited approaches, we first explore the ensemble modeling (additive and concatenation) on a large scale over the multiple word embedding models presented in Section 3. This is done exclusively on the small specialized comparable corpora. We then explore different ways to supply each specialized comparable corpus with external resources based on ensemble approaches. We show new ways to take advantage of external data and embedding models in order to efficiently extract bilingual lexicons from specialized comparable corpora. Our methodology is two-fold. In the subsection 4.1 we first describe ensemble approaches and their application on one type of corpora (here the specialized comparable corpora), then in subsection 4.2, we introduce the adaptation of ensemble approaches while combining the specialized corpus with external resources.

4.1 Specialized Meta-Embeddings

While each embedding model captures some specific context word information, it is natural and straightforward to seek for their complementarity. The specialized meta-embeddings approach consists in combining different embedding models learned from the specialized corpus. We basically use an ensemble approach to represent each word, which means that each word has its own meta-embedding. This is illustrated in the following equation:

$$Ensemble(w) = f(v_w^1, v_w^2, ..., v_w^n) \quad (1)$$

with $Ensemble(w)$ the meta-embedding representation of a given word w and f the ensemble approach used to combine the different embedding models. f can be the additive or the concatenation technique. Finally, v_w^n represents a given embedding model of the word w and n the number of used embedding models. For instance, given the Glove, the CBOW and the Skip-gram models trained on a specialized corpus, the ensemble model of a given word w would be the concatenation (or addition) of its three embedding representations ($f(v_w^{Glove}, v_w^{CBOW}, v_w^{Skip-gram}$).

[2]This technique can not be applied when embeddings are not of the same dimension size (unless using padding).

[3]L2 norm can be applied either at dimension level (as suggested by Glove authors) or at vector length level.

4.2 Mixed Meta-Embeddings

While specialized comparable corpora suffer from the lack of data, one good alternative is to enrich them with external resources. The remaining question is how to take advantage of out-of-domain data to increase specialized corpus size without degrading its specific properties. A basic way to combine word embeddings from two different data sets (here the specialized and the external corpora) is to apply an ensemble approach while fixing the type of the embedding model. For instance, using the CBOW model, each word can be jointly represented by its embedding vector issued from the specialized corpus (noted v_w^s) and its embedding model issued from the general domain corpus (noted v_w^g). This is illustrated in the following equation:

$$Ensemble(w) = f(v_w^s, v_w^g) \tag{2}$$

The Mixed Meta-Embedding representation can be generalized over several embedding models as follows:

$$Ensemble(w) = f'(f(v_w^{s^1}, v_w^{g^1}), f(v_w^{s^2}, v_w^{g^2}), ..., f(v_w^{s^n}, v_w^{g^n})) \tag{3}$$

where f and f' represent the ensemble functions (concatenation or addition). n represents the number of embedding models. One condition while combining embeddings built from different corpora is to ensure that a given word w is present in all the combined corpora. If not, we can choose to discard this word or to replace the missing vector by zeros (padding).

5 Data and Resources

In this section, we describe the different textual resources used for our experiments: the comparable corpora, the bilingual dictionary and the terminology reference lists.

5.1 Comparable Corpora

The specialized comparable corpora were selected in terms of bilingual terminology access in technical domains. For our experiments, we used two specialized comparable corpora:

Breast cancer corpus (BC) is composed of documents collected from scientific and medical portals such as the ScienceDirect[4]. The documents were taken from the medical domain within the sub-domain of "breast cancer". We have selected the documents published between 2001 and 2015 where the title or the keywords contain the term *breast cancer* in English and its translation in French.

Wind energy corpus (WE) has been released in the TTC project[5]. This corpus has been crawled from the Web using *Babouk* crawler (Groc, 2011) based on several keywords such as *wind*, *energy*, *rotor* in English and its translation in French.

In addition, we use three corpora of general language as external resources:

JRC acquis corpus (JRC) is a collection of legislative texts of the European Union[6]. We used the French-English version at OPUS which is based on the paragraph-aligned corpus provided by JRC (Tiedemann, 2012).

Common crawl corpus (CC) is an open repository of data collected over 7 years of web crawling sets of raw web page data and text extracts[7].

[4] www.sciencedirect.com/
[5] www.ttc-project.eu/index.php/releases-publications
[6] opus.lingfil.uu.se/JRC-Acquis.php
[7] commoncrawl.org

Wikipedia corpus (Wiki) The English wikipedia corpus[8] is a dump which was released on 03-Feb-2018 and the French wikipedia corpus [9] was released on 02-Feb-2018.

Even if JRC, CC are parallel corpora, we didn't explicitly exploit their parallel relationship. We considered these external data sets as if they were comparable corpora. The documents were normalized through tokenisation, part-of-speech tagging, and lemmatisation using the TTC TermSuite[10]. Finally, the function words were removed and the hapax were discarded. Table 1 shows the number of documents (# doc.) and the number of content words (# words) for each corpus.

Corpus	# content words		# distinct words	
	FR	EN	FR	EN
BC	521,262	525,934	6,630	8,221
WE	313,954	314,551	5,346	6,378
JRC	70.3M	64.2M	100,004	93,104
CC	91.3M	81.1M	250,999	259,226
WIKI	740.2M	2,669M	1,067,095	2,443,866

Table 1: Characteristics of the corpora.

5.2 Bilingual Dictionary

The bilingual dictionary used in our experiments is the French/English dictionary ELRA-M0033 (243,539 entries) available from the ELRA catalogue[11]. This resource is a general language dictionary which contains only a few terms related to the medical and wind energy domains.

5.3 Gold Standard

To evaluate the quality of bilingual terminology extraction from specialized comparable corpora, a bilingual terminology reference list is required. In the general domain, the reference list is randomly composed of a sub-part of the bilingual dictionary (Gaussier et al., 2004; Jakubina and Langlais, 2017). In the specialized domain, this list is usually composed of few words that reflect the terminology of the specialized comparable corpus. For instance, Chiao and Zweigenbaum (2002) used a list composed of 95 single words, Morin et al. (2007) used 100 single words and Bouamor et al. (2013) used 125 and 79 single words. For breast cancer, the lists are derived from the UMLS[12] meta-thesaurus. Concerning wind energy, the lists are provided with the corpora (see footnote 5). Each word composing a pair of terms of a reference list appears at least 5 times in the comparable corpus. The bilingual terminology reference list is composed of 248 French/English single words for the Breast cancer corpus and 150 French/English single words for the Wind energy corpus.

6 Experiments and Results

We conducted two sets of experiments. The first one aims at providing insights into the behaviour of each state-of-the-art embedding model on the specialized comparable corpora. The second one aims at studying the contribution of ensemble models.

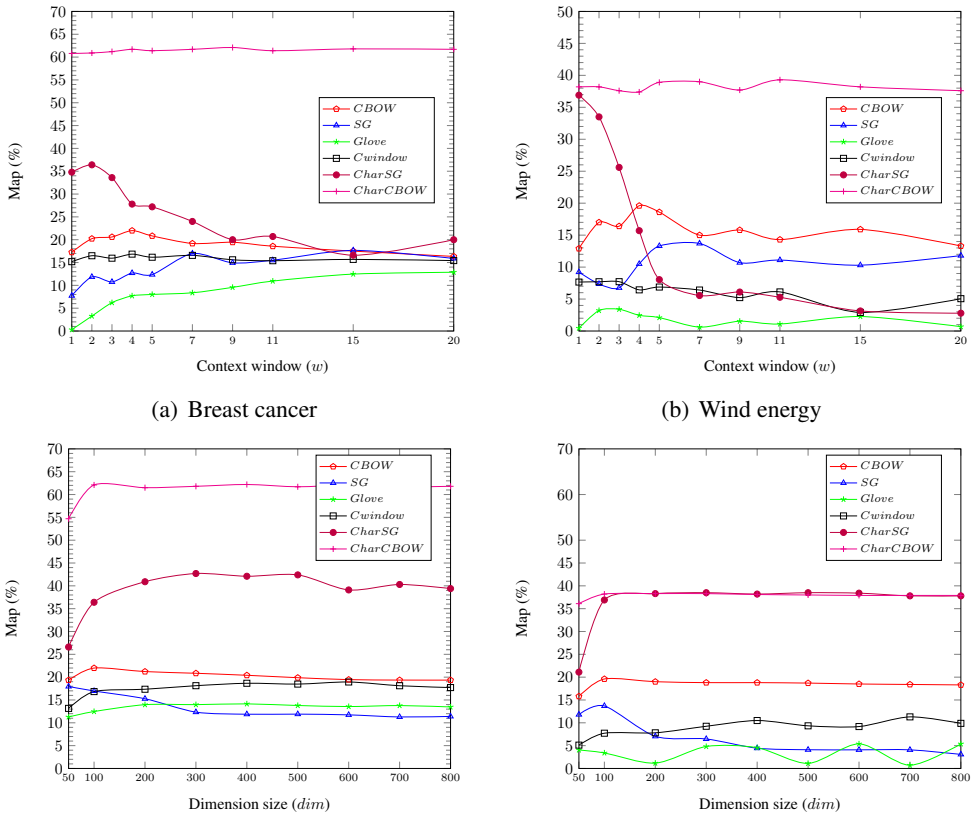
We present the results obtained for the terms belonging to the reference list for English to French direction measured in terms of the Mean Average Precision (MAP) (Manning et al., 2008) as follows:

$$MAP(Ref) = \frac{1}{|Ref|} \sum_{i=1}^{|Ref|} \frac{1}{r_i} \quad (4)$$

[8] dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2
[9] dumps.wikimedia.org/frwiki/latest/frwiki-latest-pages-articles.xml.bz2
[10] code.google.com/p/ttc-project
[11] www.elra.info/
[12] www.nlm.nih.gov/research/umls

where $|Ref|$ is the number of terms of the reference list and r_i the rank of the correct candidate translation i.

Figure 1 shows the results of each embedding model for the task of bilingual terminology extraction from the two specialized comparable corpora. We report the results of the continuous bag-of-word model ($CBOW$), the Skip-gram model (SG), the glove model ($Glove$), the structured continuous window model ($Cwindow$) and the two character n-gram models, namely the character skip-gram model ($CharSG$) and the character CBOW model ($CharCBOW$)[13]. For each specialized comparable corpus, we varied the context window size (w) (see sub-figures 1(a) and 1(b)) and the embeddings dimension size (dim) (see sub-figures 1(c) and 1(d)).



(a) Breast cancer (b) Wind energy

(c) Breast cancer (d) Wind energy

Figure 1: Contrasting several embedding representations on the breast cancer and wind energy corpora.

The first remarkable point is the performance of the character n-gram embedding models which far outperform other models. This is an important finding since to our knowledge, no previous evaluation of character n-gram-based models have been conducted so far for bilingual terminology extraction from specialized corpora. The first competitive model which is $CharCBOW$, is not sensitive to the context window size as well as the embeddings dimension size. The second competitive model which is $CharSG$, is also not sensitive to the embeddings dimension size but sensitive to the context window size. $CBOW$ which is the third competitive model, turned out to be the best word-based model. Nevertheless, its performance is far below the character-based models. $CBOW$ is also not very sensitive to the context window and the embeddings dimension sizes.

Tables 2 and 3 show a comparison of several combinations of word embedding models for the two specialized comparable corpora. Each combination is based on L2 normalization at vector length level (len) or at vector dimension level (dim). We chose the three embedding models that have shown the best performance individually (according to the Figure 1: $CharSG$, $CharCBOW$ and $CBOW$) and

[13]We don't report the results of dependency-based and structured-based models due to the low performance of these approaches.

we combine them with all the models. For each Table, the first line is a quick reminder of the three embedding models and their performance in terms of MAP scores shown between brackets. The following two lines also remind the embedding models used for combination and their MAP scores when used individually. The following four lines present the results of the combination of these models by addition or concatenation for both normalizations. For instance in Table 2, the $CharSG$ and $CBOW$ embedding models used individually have respectively 36.4% and 21.9% of MAP and the combination of the two models gives 22.9% of MAP by addition and 34.9% of MAP by concatenation with L2 normalization at vector length level. Combining embedding models is very effective when using concatenation in most cases. The best combination model is obtained by the concatenation of $CharSG$ and $CharCBOW$ using length L2 Norm (70.3% of MAP in Table 2) and by the concatenation of $CharCBOW$ and $CBOW$ (49.5% of MAP in Table 3) followed by the same model using dimension L2 Norm for breast cancer corpus (68.1% of MAP in Table 2) and using $CharSG$ with SG concatenation for wind energy corpus (45.4% of MAP in Table 3). We observe that both types of L2 normalization are in general useful for concatenation with a better performance when using length normalization.

	CharSG (36.4)					CharCBOW (60.8)				CBOW (21.9)			L2 Norm.
	CBOW	SG	Glove	CharCBOW	Cwindow	SG	Glove	CBOW	Cwindow	SG	Glove	Cwindow	
	21.9	16.9	12.4	60.8	16.1	16.9	12.4	21.9	16.1	16.9	12.4	16.1	
Addition	22.9	15.9	19.7	59.3	18.9	47.6	40.5	57.3	55.8	20.3	14.6	18.4	Len
Concat.	34.9	34.6	29.8	**70.3**†	34.1	65.6†	48.3	**64.3**†	64.7†	**26.2**†	18.2	23.9†	
Addition	21.9	17.2	20.2	56.0	18.8	40.9	40.8	55.1	48.5	17.4	15.5	18.5	Dim
Concat.	37.7†	37.0	31.4	68.1†	33.8	58.8	50.9	62.2†	59.2	23.9†	18.9	**26.9**†	

Table 2: Embeddings combination using the breast cancer corpus (MAP %) (†:t-test significance at 0.05).

	CharSG (36.9)					CharCBOW (38.2)				CBOW (19.6)			L2 Norm.
	CBOW	SG	Glove	CharCBOW	Cwindow	SG	Glove	CBOW	Cwindow	SG	Glove	Cwindow	
	19.6	13.7	4.81	38.2	19.6	13.7	4.81	19.6	19.6	13.7	4.81	19.6	
Addition	35.7	21.1	18.6	33.2	23.7	29.5	23.1	41.9†	25.5	11.8	13.4	10.4	Len
Concat.	**45.4**†	42.8†	24.1	38.8†	30.0	45.6†	28.0	**49.5**†	40.1†	21.1†	15.9	22.0†	
Addition	20.9	25.5	13.5	33.2	12.8	26.5	12.1	25.6	12.6	11.9	13.2	11.1	Dim
Concat.	31.3	43.4†	18.2	39.0†	32.1	44.0†	17.4	32.3	41.6†	20.2	17.1	**23.5**†	

Table 3: Embeddings combination using the wind energy corpus (MAP %)(†:t-test significance at 0.05).

Model	BC	JRC	CC	WIKI	ALL	BC + JRC	BC + CC	BC + WIKI	ALL
CharCBOW	**60.8**	35.3	57.4	60.7	62.7	52.9	73.9	73.1	**74.3**
CharSG	36.4	41.1	61.6	58.8	61.8	65.2	76.5	69.2	70.3
(Bojanowski et al., 2016)	-	-	-	**72.4**	-	-	-	-	-
Model	WE	JRC	CC	WIKI	ALL	WE + JRC	WE + CC	WE + WIKI	ALL
CharCBOW	**38.2**	42.8	60.1	67.8	70.1	43.8	65.9	71.3	71.4
CharSG	36.9	49.9	61.9	71.0	**70.8**	55.1	65.7	72.2	**72.4**
(Bojanowski et al., 2016)	-	-	-	68.2		-	-	-	-

Table 4: Results (MAP %) of different embedding models on the specialize corpora as well as several out-of-domain corpora and their combinations ((Bojanowski et al., 2016) is the pre-trained CharSG model).

Table 4 shows the results of different combinations[14] of specialized corpora with external resources.

[14]Combination means that we first merge different data sets in one single corpus and then, we learn an embedding model

We represent in the 1^{st} column the results for the specialized corpora taken individually (BC and WE) and then, from the 2^{nd} to the 4^{th} column we show the results of the external resources taken individually and their combination represented in the 5^{th} column (ALL). Finally, from the 6^{th} to the 8^{th} column we combine the specialized corpora with each external data and then their entire combination[15] (All).

Overall, we see that combining different resources gives significant improvements over the two specialized corpora. We also notice the usefulness of external data used individually which performs better than small specialized corpora, except for CharCBOW which shows the strength of this model and its usefulness over other types of embedding models. Also, using the pre-trained embedding model of Bojanowski et al. (2016) obtained good results (72.4% for BC and 68.2% for WE) but if we compare it to the same model (CharSG) learned while combining specialized and external data sets, we observe better performance (76.5% for BC and 72.4% for WE) which shows that using jointly specialized and external data is more efficient than the use of external data only. This statement is confirmed by the results obtained by individual external data sets which are always lower than their combination with specialized corpora.

	Individual corpus			Corpus combination (GSA)	
Models	BC	JRC	CC	BC + JRC	BC + CC
SA	27.0	**52.0**	**75.5**	61.7	**80.2**
CBOW	17.1	40.3	60.9	49.9	67.7
SG	12.8	40.5	56.0	46.5	63.2
CharCBOW	**60.8**	35.3	57.4	52.9	73.9
CharSG	36.4	41.1	61.6	**65.2**	76.5
				Vector concatenation	
SCBOW	-	-	-	53.7	70.7
SSG	-	-	-	36.3	40.2
SCBOW+SSG	-	-	-	56.1	70.9
SSA	-	-	-	66.6	**82.3**
Our approaches					
SCharCBOW	-	-	-	64.9	74.9
SCharSG	-	-	-	73.0	77.4
SCharCBOW+SCharSG	-	-	-	67.0	**80.7**
Meta-Emb (Best)	-	-	-	**74.8**	**83.1**

Table 5: Results (MAP %) of the *Standard Approach* (SA), the *Global Standard Approach* (GSA) and the *Selective Standard Approach* (SSA) and our approaches using CharCBOW and CharSG and their combinations (SCharCBOW, SCharSG and Meta-Emb) for the breast cancer corpus (BC) using the different external data (the improvements indicate a significance at the 0.05 level using the Student t-test).

In Table 5, we report the results obtained in Hazem and Morin (2017) (SA, CBOW, SG, SCBOW, SSG ,SCBOW+SSG and SSA) and our results using combination and meta-embeddings of character n-gram models (SCharCBOW, SCharSG, SCharCBOW+SCharSG and Meta-Emb (Best)[16]). The main conclusion in Hazem and Morin (2017) is that the best embedding combination (SCBOW+SSG) couldn't outperform the selective standard approach (SSA). According to our results, character n-gram models and their combination obtained better results than the best CBOW and Skip-gram combination (SCBOW+SSG) and also outperformed the selective standard approach (SSA obtained 66.6% using BC + JRC and 82.3% using BC + CC while our best model obtained **74.8%** and **83.1%** on the same corpora). The results of Table 5 provide strong support for data combination and meta-embeddings using character n-gram models. Also, we highlight the fact that character ngram models and their combination is much faster than CBOW and Skip-gram models. In addition, the dimension size of embedding models

on the merged corpus.

[15] ALL in the 5^{th} column means that we combine JRC, CC and WIKI, while ALL in the 9^{th} column means that we combine JRC, CC and WIKI and the specialized corpora BC or WE.

[16] Meta-Emb (best) stands for the combination of SCharSG on specialized corpus with SCharCBOW+SCharSG on external data.

is very low (around 300) while in the standard approach it corresponds to the vocabulary size which leads to sparse vectors and high computational cost.

7 Discussion

The first important finding of this work is the efficiency of the character n-gram models (CharCBOW and CharSG) which drastically outperform other models, whether on specialized or general domain data sets. Their better performance can be explained by the fact that both models are based on characters to build the embedding models. While CBOW and Skip-gram suffer from the lack of data to build efficient models over specialized corpora, the character-based approaches benefit of much more training examples as they use characters for their models. The second important finding is the performance of external data when applied for extracting bilingual terms. This is not surprising as external data sets such as wikipedia or common crawl for instance, contain several scientific and specialized documents. In addition, we could observe the complementarity of external resources with the specialized domain data sets. More precisely, concatenating embeddings of specialized and external resources significantly improves the results. This can be explained by the nature of the captured information which can be resumed in the concatenated embedding vectors. If a single generic embedding model is difficult to obtain, character n-gram and word-based embedding models can be efficiently combined to improve bilingual terminology extraction from comparable corpora.

We also conducted an error analysis of the different proposed models and we couldn't find a strong relation between the embedding models and the non captured terms. However, we observed that the CharCBOW model with the meta-embedding combination using BC with CC corpora improves overall the rank of the translations obtained for each individual corpus. By looking at the 115 translations of BC and the 80 of BC+CC not found in the first rank, we found 58 terms in common including 44 terms with a better rank with BC+CC corpora. In the same way, from the 75 translations of CC and the 80 of BC+CC not found in the first rank, we found 62 terms in common including 46 terms with a better rank with BC+CC corpora. It might seem surprising that we found more terms outside the first rank with CC corpus than BC+CC (80 versus 75) since the MAP is lower with CC than BC+CC (57.4 versus 73.9). In fact, the CharCBOW model with BC+CC improves overall the rank of all the translations of CC and more particularly the rank of the first hundred translations taken into account in the calculation of the MAP. We also observed that the more frequent terms are the best translated for SA and SSA approaches. For the embedding approaches, this observation is not relevant. For instance, the translations of frequent terms such as *cancer* and *breast* are found in the first ranks and the translations of infrequent terms such as *lumpectomy* and *fibroadenoma* are found in the last ranks with SA and SSA approaches. With embedding approaches, *cancer* and *fibroadenoma* are found in the first ranks and *breast* and *lumpectomy* in the last ranks. We have not been able to better characterize terms in the last ranks with embedding approaches. This is strongly dependent on embedding parameters and also context size and embedding dimensions. Our best system, however obtained 90% of precision on top 5, in the perspective of providing first translation terms for translation aided systems, our proposed approach is certainly more appropriate than the standard approach whether in terms of computational cost or in terms of accuracy.

8 Conclusion

In this paper we have explored a variety of embedding models and there impact on the task of bilingual terminology extraction from specialized comparable corpora. We have also proposed meta-embedding representations and have shown under which conditions they can be jointly used for better performance. If further investigations are probably needed, our findings strengthens the idea that using meta-embeddings based on specialized and general domain data sets improves the performance of mining bilingual specialized lexicons.

Acknowledgments

The research leading to these results has received funding from the French National Research Agency under grant ANR-17-CE23-0001 ADDICTE (Distributional analysis in specialized domain).

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*, pages 2289–2294, Austin, TX, USA.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR*, abs/1607.04606.

Dhouha Bouamor, Nasredine Semmar, and Pierre Zweigenbaum. 2013. Context Vector Disambiguation for Bilingual Lexicon Extraction from Comparable Corpora. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'13)*, pages 759–764, Sofia, Bulgaria.

A. P. Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh M. Khapra, Balaraman Ravindran, Vikas C. Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS'14)*, pages 1853–1861, Montreal, Quebec, Canada.

Yun-Chuang Chiao and Pierre Zweigenbaum. 2002. Looking for candidate translational equivalents in specialized, comparable corpora. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, pages 1208–1212, Tapei, Taiwan.

Estelle Maryline Delpech. 2014. *Comparable Corpora and Computer-assisted Translation*. John Wiley & Sons, Inc.

Manaal Faruqui and Chris Dyer. 2014. Improving Vector Space Word Representations Using Multilingual Correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL'14)*, pages 462–471, Gothenburg, Sweden.

Pascale Fung and Kathleen McKeown. 1997. Finding terminology translations from non-parallel corpora. In *Proceedings of the 5th Annual Workshop on Very Large Corpora (VLC'97)*, pages 192–202, Hong Kong.

Pascale Fung. 1998. A Statistical View on Bilingual Lexicon Extraction: From Parallel Corpora to Non-parallel Corpora. In *Proceedings of the 3rd Conference of the Association for Machine Translation in the Americas on Machine Translation and the Information Soup (AMTA'98)*, pages 1–17, Langhorne, PA, USA.

Pablo Gamallo. 2007. Learning bilingual lexicons from comparable english and spanish corpora. In *Proceedings of the 11th Conference on Machine Translation Summit (MT Summit XI)*, pages 191–198, Copenhagen, Denmark.

Justin Garten, Kenji Sagae, Volkan Ustun, and Morteza Dehghani. 2015. Combining Distributed Vector Representations for Words. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 95–101, Denver, CO, USA.

Eric. Gaussier, Jean-Michel Renders, Irena. Matveeva, Cyril. Goutte, and Hervé Déjean. 2004. A geometric view on bilingual lexicon extraction from comparable corpora. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL'04)*, pages 526–533, Barcelona, Spain.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2014. Bilbowa: Fast bilingual distributed representations without word alignments. *CoRR*, abs/1410.2455.

Clément De Groc. 2011. Babouk : Focused Web Crawling for Corpus Compilation and Automatic Terminology Extraction. In *Proceedings of 10th International Conferences on Web Intelligence (WIC'11)*, pages 497–498, Lyon, France.

Amir Hazem and Emmanuel Morin. 2012. ICA for Bilingual Lexicon Extraction from Comparable Corpora. In *Proceedings of the 5th Workshop on Building and Using Comparable Corpora (BUCC)*, pages 126–133, Istanbul, Turkey.

Amir Hazem and Emmanuel Morin. 2017. Bilingual Word Embeddings for Bilingual Terminology Extraction from Specialized Comparable Corpora. In *Proceedings of the 8th International Joint Conference on Natural Language Processing (IJCNLP'17)*, pages 685–693, Taipei, Taiwan.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL'14)*, pages 58–68, Baltimore, MD, USA.

Azniah Ismail and Suresh Manandhar. 2010. Bilingual lexicon extraction from comparable corpora using in-domain terms. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING'10)*, pages 481–489, Beijing, China.

Laurent Jakubina and Phillippe Langlais. 2017. Reranking translation candidates produced by several bilingual word similarity sources. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL'17)*, pages 605–611, Valencia, Spain.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL'14)*, pages 302–308, Baltimore, MD, USA.

Bofang Li, Tao Liu, Zhe Zhao, Buzhou Tang, Aleksandr Drozd, Anna Rogers, and Xiaoyong Du. 2017. Investigating Different Syntactic Context Types and Context Representations for Learning Word Embeddings. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP'17)*, pages 2421–2431.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/Too Simple Adaptations of Word2Vec for Syntax Problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'15)*, pages 1299–1304, Denver, CO, USA.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Emmanuel Morin and Amir Hazem. 2014. Looking at Unbalanced Specialized Comparable Corpora for Bilingual Lexicon Extraction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL'14)*, pages 1284–1293, Baltimore, MD, USA.

Emmanuel Morin, Béatrice Daille, Koichi Takeuchi, and Kyo Kageura. 2007. Bilingual Terminology Mining – Using Brain, not brawn comparable corpora. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07)*, pages 664–671, Prague, Czech Republic.

Jian-Yun Nie. 2010. Cross-language information retrieval. *Synthesis Lectures on Human Language Technologies*, 3(1):1–125.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

Emmanuel Prochasson and Pascale Fung. 2011. Rare Word Translation Extraction from Aligned Comparable Documents. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL'11)*, pages 1327–1335, Portland, OR, USA.

Reinhard Rapp. 1999. Automatic Identification of Word Translations from Unrelated English and German Corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, pages 519–526, College Park, MD, USA.

Tuomas Talvensaari, Jorma Laurikkala, Kalervo Järvelin, Martti Juhola, and Heikki Keskustalo. 2007. Creating and Exploiting a Comparable Corpus in Cross-language Information Retrieval. *ACM Transactions on Information Systems (TOIS)*, 25(1).

Jörg Tiedemann. 2012. Parallel Data, Tools and Interfaces in OPUS. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul, Turkey.

Ivan Vulić and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL'15)*, pages 719–725, Beijing, China.

Ivan Vulić and Marie-Francine Moens. 2016. Bilingual distributed word representations from document-aligned comparable data. *Journal of Artificial Intelligence Research (JAIR)*, 55(1):953–994.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'15)*, pages 1006–1011, Denver, CO, USA.

Wenpeng Yin and Hinrich Schütze. 2016. Learning Word Meta-Embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16)*, pages 1351–1360, Berlin, Germany.

Learning Emotion-enriched Word Representations

Ameeta Agrawal, Aijun An and **Manos Papagelis**
Department of Electrical Engineering and Computer Science
York University, Toronto, Canada
{ameeta,aan,papaggel}@eecs.yorku.ca

Abstract

Most word representation learning methods are based on the distributional hypothesis in linguistics, according to which words that are used and occur in the same contexts tend to possess similar meanings. As a consequence, emotionally dissimilar words, such as "*happy*" and "*sad*" occurring in similar contexts would purport more similar meaning than emotionally similar words, such as "*happy*" and "*joy*". This complication leads to rather undesirable outcome in predictive tasks that relate to affect (emotional state), such as emotion classification and emotion similarity. In order to address this limitation, we propose a novel method of obtaining emotion-enriched word representations, which projects emotionally similar words into neighboring spaces and emotionally dissimilar ones far apart. The proposed approach leverages distant supervision to automatically obtain a large training dataset of text documents and two recurrent neural network architectures for learning the emotion-enriched representations. Through extensive evaluation on two tasks, including emotion classification and emotion similarity, we demonstrate that the proposed representations outperform several competitive general-purpose and affective word representations.

1 Introduction

Emotion detection from text is the task of identifying emotions from natural language data such as reviews, blogs, news articles, and so on (Alm et al., 2005; Aman and Szpakowicz, 2007). While numerous taxonomies of emotions have been proposed (Ekman, 1992; Plutchik, 1980; Parrott, 2001), most psychologists agree that an emotion is a feeling that characterizes the state of mind such as *happiness*, *sadness*, *anger*, among others. The ability to detect emotions in text is critical for a number of applications and services in diverse domains, including market research, customer relations, gaming, and intelligent tutoring systems, to name a few (Mohammad and Turney, 2013).

Despite its potentially wide-spread use, the automatic detection of emotions remains a challenging multi-class, sometimes multi-label, classification problem due to a number of reasons, including: (i) different emotion models consist of different number and types of emotion categories; (ii) emotions are not only subjective but also fuzzy, with more than one emotion occurring at the same time. As a result, development of emotion related resources, such as training data, has been limited to a few manually annotated datasets or lexicons, a process that requires much time and effort, and is expensive.

To solve the limited training data problem, the recent success of word embeddings has garnered increased attention in the design of emotion classification systems (Bravo-Marquez et al., 2016; Pool and Nissim, 2016; Mohammad and Bravo-Marquez, 2017). Word embeddings are distributed word representations (Collobert et al., 2011; Turian et al., 2010), where each word w in the vocabulary $\mathcal{V}$ is mapped into a dense, low-dimensional, continuous-valued vector $x \in \mathbb{R}^d$, $d \ll |\mathcal{V}|$. The underlying idea is that words that frequently occur together in same contexts get mapped to similar regions of the vector space.

Most embeddings (Mikolov et al., 2013b; Pennington et al., 2014) are typically modeled using the syntactic context of words following the distributional hypothesis, i.e., words which occur in similar

Proceedings of the 27th International Conference on Computational Linguistics, pages 950–961
Santa Fe, New Mexico, USA, August 20-26, 2018.

word pair	GloVe	CBOW
(happy, joy) ↑	0.601	0.355
(happy, sad) ↓	0.643	0.535
(cry, weep) ↑	0.605	0.574
(cry, laugh) ↓	0.657	0.403

Table 1: Cosine similarity between emotionally similar (↑) and emotionally dissimilar (↓) word pairs

contexts tend to be semantically similar. While the property of semantic similarity is beneficial in a number of tasks, modeling emotionally *dissimilar* words with similar contexts into neighboring spaces becomes counterproductive in affective tasks such as emotion classification. To further motivate this limitation, Table 1 presents the cosine similarity between the word vectors of a few word pairs obtained from popular pre-trained word embeddings such as GloVe (Pennington et al., 2014) and CBOW (Mikolov et al., 2013b). According to the similarity scores, both GloVe and CBOW rate the word pair (*happy*, *sad*) as more similar than (*happy*, *joy*).

The effectiveness of word embeddings has been shown to be task-dependent (Labutov and Lipson, 2013; Bansal et al., 2014) and while there is some work on generating task-specific embeddings (Kalchbrenner et al., 2014; Tang et al., 2014; Chen and Manning, 2014; Qu et al., 2015), there is little work specifically exploring the role of task-specific *emotion-enriched* embeddings.

In this paper, we propose learning emotion-enriched word representations[1], which we call Emotion Word Embeddings (EWE), in order to project emotionally similar words into neighboring spaces. Towards that end, first, a method of distant supervision is employed to automatically create a large training dataset with a rich spectrum of emotions. Then, two recurrent neural network architectures are employed to learn emotion-aware word representations by leveraging noisy, but large training data. Specifically, we use Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) to capture the *semantic* information between the words of the text document as well as the *emotion* information provided in the form of the target label obtained through distant supervision. Experimental evaluation demonstrates the effectiveness of learned emotion embeddings in the two tasks of emotion classification and emotion similarity.

The major contributions of this work include: (i) a novel distant supervision method for automatically labeling a large corpus of training data with fine-grained emotions; (ii) two LSTM model architectures for learning emotion-enriched word embeddings from text documents (a single-label model and a multi-label model); (iii) and, an extensive evaluation of the learned word vectors on two tasks: emotion classification over four domains of text (blogs, fairy tales, personal experiences, and tweets) and emotion similarity.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 describes the proposed model, followed by the experimental setup and results in Section 4. Section 5 presents the qualitative analysis and finally, Section 6 concludes the paper.

2 Related Work

There exists a large body of work discussing representation learning. Generic word vector models use unannotated text to learn the embedding vector of each term as a fixed length continuous representation by predicting adjacent terms in the document (Bengio et al., 2003; Collobert and Weston, 2008; Collobert et al., 2011; Mikolov et al., 2013b; Mikolov et al., 2013a; Pennington et al., 2014). Incorporating distributed word embeddings as features has proven effective in a variety of natural language processing tasks, including parsing (Socher et al., 2013), language modeling (Bengio et al., 2003; Mnih and Hinton, 2008) and sentiment analysis (Socher et al., 2011; Labutov and Lipson, 2013; Tang et al., 2014; Tang et al., 2016). However, the effectiveness of generic word embeddings has been shown to be heavily task-dependent (Labutov and Lipson, 2013; Bansal et al., 2014).

[1]Available for download: `https://www.dropbox.com/s/5egqnbktbfxp2im/ewe_uni.txt.zip?dl=0`

To increase the effectiveness of generic word embeddings, therefore, there have been some lines of work in using neural networks for inducing task-specific *affective* embeddings. Socher et al. (2011) learned vector space representations for multi-word phrases using recursive autoencoders for the task of sentiment analysis. Labutov and Lipson (2013) produced task-specific embeddings from existing word embeddings for sentiment analysis. Kalchbrenner et al. (2014) trained their models on a large dataset of tweets, where a tweet was automatically labeled as positive or negative depending on the emoticon that occurs in it. Tang et al. (2014; 2016) also induced embeddings from scratch for sentiment analysis using a dataset of 10M tweets obtained through distant supervision labeled with positive and negative emoticons. More recently, affective word representations have been obtained using a corpus of almost 1B tweets weakly labeled with a set of 64 emojis (Felbo et al., 2017). An alternative to learning task-specific embeddings from scratch or updating existing embeddings using neural networks is post-processing (or fine-tuning) existing embeddings with respect to some external knowledge source such as a lexicon (Faruqui et al., 2015).

All the above-mentioned approaches of learning task-specific affective embeddings (Tang et al., 2014; Tang et al., 2016; Felbo et al., 2017) rely on tweets data obtained from Twitter, automatically labeled using emoticons. However, tweets data do not generalize well to texts from other domains such as blogs, narratives, etc. Instead, we explore a novel domain of text (product reviews) to present a more generalizable approach to obtaining large-scale training data using distant supervision. In addition, while previous embeddings were trained on corpora of sizes ranging from 10M to 1B tweets, our models are able to learn rich representations from a much smaller dataset of about 200K reviews. Furthermore, although a binary spectrum of positive and negative sentiment (Tang et al., 2014) or a large axis of 64 emojis (Felbo et al., 2017) has been previously used to generate representations, we align our embeddings along an emotion model firmly grounded in psychology which remains unexplored yet. Lastly, while the previous approaches used only a single-label setting (i.e., only one affect label per document), we propose modeling a more natural multi-label setting where a document can be associated with more than one emotion label.

3 Emotion-enriched Word Representations

In this section, we first describe two neural network models and their components for learning Emotion Word Embeddings (EWE). Then, we describe the process of automatically obtaining a large training dataset of text documents labeled with emotions through distant supervision.

3.1 Training Word Embeddings using LSTM

Let $\mathcal{V} = \left\{w_1, ..., w_{|\mathcal{V}|}\right\}$ be the vocabulary of word tokens in the annotated dataset. Each word w_i is represented as a n-dimensional continuous vector $\mathbf{x}_i \in \mathbb{R}^n$ and the full embedding matrix is $E \in \mathbb{R}^{n \times |\mathcal{V}|}$. Starting from original embeddings $\mathbf{x}_i^o$ of word w_i (initialized either randomly or through some pre-trained word embeddings), the goal is to learn emotion-enriched embeddings $\mathbf{x}_i^e$ for w_i.

The LSTM (Long Short-Term Memory) model finds a dense low dimensional representation of words by sequentially and recurrently processing each word in a document. Specifically, the inputs of the LSTM are preprocessed text documents that consist of a sequence of words and their corresponding target variable. Let $\mathcal{D} = \{(d_1, y_1), ..., (d_D, y_D)\}$ denote an annotated dataset of documents, where $d = \{w_1, w_2, ..., w_N\}$ denotes a text document consisting of a sequence of N words and y_i is the corresponding emotion label distribution for document d_i. The words of the text document are, first, converted into vector representations, which are then sequentially fed into the LSTM model left-to-right. Then, through back-propagation, the original word vectors get updated during training, producing emotion-enriched embeddings $\mathbf{x}_i^e$ for all $w_i \in \mathcal{V}$.

In this work, we consider two model architectures to capture the *context* information by modeling the long-range dependencies between the words of a text document and *emotion* information provided through the target label to map each word into an affective embedding space. Model 1 (EWE_{UNI}) considers a single emotion label for each document, whereas Model 2 ($\text{EWE}_{\text{MULTI}}$) allows multiple labels for a document. Figure 1 presents an overview of the proposed framework. First, we create a cor-

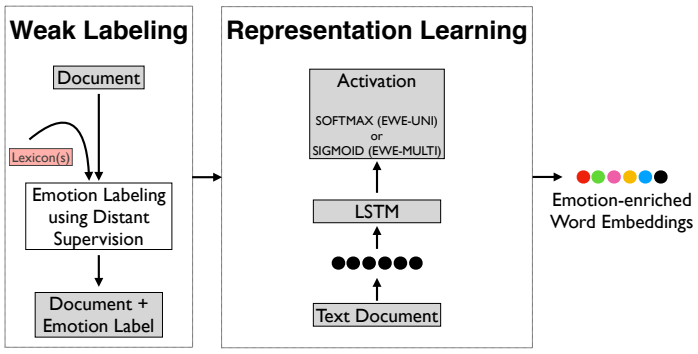


Figure 1: Overview of the framework for obtaining emotion-aware word representations

pus of emotion-labeled documents using emotion lexicons through a distant supervision process (to be described in Section 3.2). Then this corpus is used as training data to learn emotion-enriched word representations using LSTM. In other words, while document-level (entire examples) labeling is used to create the training set, the embeddings get updated at individual word level.

3.1.1 Model 1: EWE_{UNI}

Most words evoke only one emotion depending on the context. As an example, consider two benchmark emotion datasets (Alm et al., 2005; Aman and Szpakowicz, 2007) where each sentence is annotated with a single emotion label. Guided by this intuition, we propose EWE_{UNI} which follows a multi-class setting, where there exists only one valid mutually exclusive emotion label l_i for a text document d_i, and $l_i \in \mathcal{L}$, where $\mathcal{L} = \{l_1, ..., l_k\}$ denotes a discrete, finite set of k emotions.

Given an annotated document with its associated emotion label, the target value y is a one-hot vector, where the values of all the indices but one are 0. For example, if d is labeled with emotion l_i, then it holds that:

$$y_j = \begin{cases} 1, & \text{if } y_j = l_i \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

The neural network consists of one hidden layer, with the embedding matrix E added to the input layer. To predict the emotion label of the input text, an output layer with a softmax activation function which gives a probability distribution over the k classes is added on top of the hidden layer for modeling multi-class probabilities. The softmax function converts the classification result into label probabilities, i.e. $y' \in [0, 1]^k$.

The final training objective is to minimize the multinomial cross-entropy loss of the predicted and true distributions, where the error over a batch of n documents is calculated as:

$$\xi = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} y_{ij} \log \left(y'_{ij} \right) \tag{2}$$

where i denotes the ith training sample, j denotes the jth class, y is the true distribution (one-hot representation of the emotion label), and y' is the predicted probability distribution, $y'_{ij} \in [0, 1]$ and $\sum_j y'_{ij} = 1$.

3.1.2 Model 2: $\text{EWE}_{\text{MULTI}}$

Although modeling emotion classification as a multi-class problem captures the basic emotion connotation of many words, in reality, most words can be associated with more than one emotion. For instance, during the process of creating the NRC EmoLex emotion lexicon (Mohammad and Turney, 2013), it was found that *anger* words tend to be associated with *disgust*, *joy* terms tend to be related with *trust*, and *surprise* terms are largely also associated with *joy*.

In order to capture a word's association with more than one emotion, the $\text{EWE}_{\text{MULTI}}$ models multi-label classification setup where each document can belong to multiple emotion classes at the same time.

Assuming k emotion classes, and more than one valid emotion label for each document, the target variable y is binary represented. In other words, $y_j = 1$ indicates presence of an emotion class, and $y_j = 0$ otherwise. For example, if document d is labeled with a subset of emotion classes, $s_i \subseteq \mathcal{L}$, then:

$$y_j = \begin{cases} 1, & \text{if } y_j \in s_i \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

To predict the emotion label of the input text, an output layer with a sigmoid activation function, which squashes the inputs into a probability range of $[0, 1]$ for every class, is added to the last layer for modeling the probability of each class independently from the other classes.

The loss objective in this case is binomial cross-entropy, computed as follows:

$$\xi = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log\left(y_i'\right) + (1 - y_i)\log\left(1 - y_i'\right)\right] \tag{4}$$

where i denotes the ith training sample, y is the binary representation of true emotion label, and y' is the predicted probability.

3.1.3 Implementation

We use pre-trained word embeddings (GloVe $|\mathcal{V}| = 1.9\text{M}$, $d = 300$ (Pennington et al., 2014)) to initialize E and use random initialization sampled from a zero mean Gaussian distribution: $x \sim \mathcal{N}\left(0, \sigma^2\right)$ for words not found in the pre-trained embeddings. Optimization of the loss function is carried out with the Adam optimizer (Kingma and Ba, 2014), which is known for yielding quicker convergence, with learning rate of 0.001, and mini-batch size set to 1024.

3.2 Labeling Training Data using Distant Supervision

To learn the emotion embeddings, we require a large dataset of text with corresponding emotion labels. Due to the challenges involved in creating large-scale emotion resources (Mohammad and Turney, 2013), however, most existing manually-annotated emotion datasets contain a very limited number of instances and words. For example, two popular emotion datasets created by Alm (2008) and Aman and Szpakowicz (2007) contain around 1200 sentences each and only about 5000 unique words each. At the same time, in order for learned word representation models to be useful, they need to generalize well to diverse domains and applications by including a much larger number of words. For instance, the vocabulary size of most existing word representations is orders of magnitude larger (e.g., 400K to 1.9M words in GloVe (Pennington et al., 2014), 3M words in word2vec (Mikolov et al., 2013a), and so on).

As it is quite challenging to create a large manually annotated emotion dataset due to human time and effort required, we leverage distant supervision (Go et al., 2009) to create a weakly labeled training dataset automatically in order to obtain emotion-enriched word representations for a much larger vocabulary. Distant supervision is the process of labeling instances based on some heuristics or rules, with some of the instances being assigned noisy or imprecise labels.

3.2.1 Distant Supervision for EWE_{UNI}

Let $\mathcal{D} = \{d_1, d_2, ..., d_D\}$ be the set of unlabeled documents. The goal is to generate an annotated dataset $\mathcal{D} = \{(d_1, l_1), ..., (d_D, l_D)\}$, where $l_i \in \mathcal{L}$ is the corresponding emotion label for document d_i and $\mathcal{L} = \{l_1, ..., l_k\}$ is a known finite set of emotion labels.

Let $d = \left\{w_1, w_2, ..., w_{|d|}\right\}$ denote the sequence of words in a document, $w_i \in d$. For each word w_i, we compute an emotion vector $\mathbf{a}(w) = \langle a_1, a_2, ..., a_k\rangle$, where a_j indicates the word-emotion association as derived from a lexicon. Although technically, while any emotion taxonomy can be followed for deriving the word-emotion vector $\mathbf{a}(w)$, in this work, we adopt Ekman's (1992) model of six emotions (*anger*, *disgust*, *fear*, *happiness*, *sadness* and *surprise*), whose origins are firmly grounded and extensively verified in psychology. To this end, we select **WordNetAffect (WNA)** (Strapparava and Valitutti, 2004), which was developed by manually labeling the emotions of a few seed words and extending it to all their WordNet synonyms, and **NRC EmoLex (NRC)** (Mohammad and Turney, 2010; Mohammad

and Turney, 2013), which was created through crowdsourcing by annotating unigrams with one or more of Plutchik's (1980) eight emotions, which in turn is a superset of Ekman's (1992) model. In WNA, each word is associated with only one emotion, therefore $a_j = 1$ if w is associated with that emotion, and $a_j = 0$ otherwise. On the other hand, in NRC, a word can be binary associated with more than one emotion, with 1 indicating an association and 0 denoting no association. For a given word w, we extract its binary association scores corresponding to the six categories of Ekman's model.

The emotion vector $\mathbf{a}(d)$ for document d is then, the sum of the emotion vectors of all its words, $\mathbf{a}(d) = \sum_{i \in d} \mathbf{a}(w_i)$. If the document has an association with at least one emotion, i.e., $\exists j, a_j(d) > 0$, then, $S = \text{argmax}_i\ \mathbf{a}(d)$, where $S \subseteq \mathcal{L}$. In other words, documents assigned zero emotion score are not considered. Finally, in case multiple emotion labels have the maximum value, i.e., $|S| > 1$, we sample uniformly at random one emotion label $l \in S$.

We investigate two strategies of computing the affective knowledge: (i) **one lexicon** - where any one lexicon is used to guide the labeling process; and, (ii) **two or more lexicons** - whereby two or more lexicons are used in order to mitigate some effects of noisy labeling. This variant assigns an emotion label to a document only if the labels output by both the lexicons match.

3.2.2 Distant Supervision for $\text{EWE}_{\text{MULTI}}$

Some words evoke more than one emotion at the same time. For example, out of the 14,000 words annotated with emotions in the NRC lexicon, almost 8,000 words (57%) are associated with more than one emotion. Therefore, we relax the labeling scheme followed in EWE_{UNI} and design $\text{EWE}_{\text{MULTI}}$ to take into consideration a multi-class, multi-label setting, where a document can have more than one emotion label.

Unlike EWE_{UNI}, in $\text{EWE}_{\text{MULTI}}$ the set of all emotions with $a_j(d) > 0$ for document d is used as final emotion labels for d. Thus, the multi-label annotated dataset $\mathcal{D}$ is $\{(d_1, S_1), ..., (d_n, S_n)\}$, where each document d_i is assigned a set of emotion labels, $S_i \subseteq \mathcal{L}$

3.2.3 Training Data

Our large corpus of unlabeled documents is extracted from the Amazon reviews dataset (McAuley et al., 2015) consisting of product reviews, spanning May 1996 - July 2014. Each review (considered as a document) is preprocessed by converting it to lowercase, tokenizing it with the NLTK toolkit (punctuation is preserved as tokens), and filtering out reviews that are too short (less than 5 words). Note that, as the proposed weak labeling is not dependent on any domain-specific indicators of affect such as emoticons or hashtags, it can be easily generalized to any type of text documents.

4 Experiments

4.1 Emotion Classification

The first task validates the effectiveness of the emotion embeddings under the supervised framework of emotion classification, where the learned word vectors are fed as features into classification models for predicting the emotion labels. We train two classifiers: (i) L2-regularized multi-class logistic regression (LR) and (ii) support vector machine (SVM) based on LIBSVM (Chang and Lin, 2011), to predict the fine-grained emotion label at the sentence level. The results of 10-fold cross validation are reported in terms of macro F_1 score, which is the average F_1 score over all the emotion classes. F_1 score is the harmonic mean of precision (p) and recall (r), $F_1 = 2\frac{p \cdot r}{p+r}$.

For the emotion lexicons, we generate a feature vector consisting of the total number of words in the sentence associated with each emotion category. For the word embedding models, we compute the average of the word vectors of all the words in the sentence along each dimension to obtain the sentence representation as the input to the classification algorithm.

4.1.1 Emotion Datasets

The following four benchmark emotion datasets from various genres of text are considered for emotion classification. The statistics of the datasets are summarized in Table 2.

dataset	domain	# emotions	total
Alm	fairy tales	5	1207
Aman	blogs	6	1290
ISEAR	experiences	5	5412
EmoTweet-top8	tweets	8	4664

Table 2: Statistics of emotion datasets

Methods		Alm	Aman	ISEAR
Lexicons	WNA	0.459	0.405	0.384
	NRC	0.387	0.370	0.378
	WNA+NRC	0.521	0.474	0.465
EWE_{UNI}	WNA	0.635	0.604	0.674
	NRC	0.604	0.582	0.666
	WNA+NRC	**0.661**	**0.623**	**0.679**
EWE_{MULTI}	NRC	0.630	0.602	0.666

Table 3: Comparison of using lexicons directly versus using lexicons to guide representation learning.

Alm: Emotions are particularly significant in the literary genre of fairy tales and this dataset contains sentences marked with one of five emotion categories: *angry-disgusted, fearful, happy, sad* and *surprised* (Alm, 2008).
Aman: Consisting of highly informal blog data, this dataset includes sentences annotated with one of six emotions: *anger, disgust, fear, joy, sadness* and *surprise* (Aman and Szpakowicz, 2007).
ISEAR: Developed for studying the relationships among cultures and emotions, this dataset contains personal experiences evoking seven emotions (Wallbott and Scherer, 1986). We use a subset of this dataset marked with one of the five emotions: *anger*, *disgust*, *fear*, *joy* and *sadness*.
EmoTweet-28: While the other annotated datasets are modeled after existing emotion taxonomies, this corpus was created by inductively identifying a set of emotion categories that characterize the emotions expressed in tweets (Liew et al., 2016). For our experiments, we extract a subset (**EmoTweet-top8**) of the eight most frequent emotions in the dataset.

4.1.2 Lexicons versus Representations

As the quality of the emotion embeddings depends on the underlying emotion lexicons adopted to create the training data, we analyze the results of using the source emotion lexicons directly versus using them to initialize EWE in Table 3.

We observe that the configurations using both the lexicons (WNA+NRC) yield better results than using any one lexicon alone. Morevover, all the EWE embeddings demonstrate significant improvements over using the lexicons directly, indicating that the affective word representation model learns useful information in addition to the knowledge available in the base lexicons adopted during distant supervision to guide the representation learning process.

4.1.3 Comparison Against State-of-the-art Representations

Next, we analyze the performance of EWE against state-of-the-art *generic embeddings* and *task-specific affective embeddings* described below, and summarized in Table 4.
Generic Embeddings: **(i) GloVe**: Global vectors[2] for word representations (Pennington et al., 2014) trained on aggregated global word-word co-occurrence statistics from a corpus capture linear substructures of the word vector space. We use the vectors that were trained on: **GloVe 6B**: 6 billion words, uncased, from Wikipedia 2014 and Gigaword v5, of dimension $d = 300$; **GloVe 42B**: 42 billion words,

[2]http://www-nlp.stanford.edu/projects/ glove/

embeddings	corpus	size	$\|\mathcal{V}\|$
GloVe 6B	Wiki + Gigaword	6B tokens	400K
GloVe 42B	Common Crawl	42B tokens	1.9M
word2vec	Google news	100B tokens	3M
SSWE	Twitter tweets	10M tweets	137K
DeepMoji	Twitter tweets	1B tweets	50K
EWE	Amazon reviews	200K reviews	183K

Table 4: Details of compared embeddings

methods		Alm		Aman		ISEAR		EmoTweet-top8	
		LR	SVM	LR	SVM	LR	SVM	LR	SVM
GloVe 6B	d =300	0.548	0.583	0.547	0.555	0.648	0.643	0.574	0.581
GloVe 42B	d =300	<u>0.590</u>	<u>0.624</u>	<u>0.564</u>	<u>0.609</u>	<u>0.675</u>	<u>0.671</u>	<u>0.609</u>	<u>0.614</u>
word2vec	CBOW	0.413	0.433	0.424	0.478	0.655	0.661	0.526	0.568
SSWE	u	0.368	0.371	0.363	0.363	0.495	0.505	0.443	0.444
DeepMoji	d =256	0.300	0.275	0.332	0.336	0.598	0.607	0.533	0.560
Retrofitting	GloVe 42B	0.141	0.110	0.111	0.111	0.553	0.559	0.245	0.220
Retrofitting	word2vec	0.110	0.108	0.100	0.098	0.488	0.472	0.232	0.214
EWE_{UNI}	WNA+NRC	**0.632**	**0.661**	**0.602**	**0.623**	**0.679**	**0.679**	**0.610**	**0.618**

Table 5: Comparison against state-of-the-art word representations (*generic embeddings* in the top half; *affective embeddings* in the bottom half) on emotion classification. The best results are shown in **bold**, and the second best results are <u>underlined</u>. Paired t-tests using the results on all four datasets indicate EWE is significantly better than all the other methods with p-values < 0.02.

uncased, from Common Crawl, of dimension $d = 300$. **(ii) `word2vec`**: These word representations[3] were learned with a continuous bag-of-words model (CBOW) (Mikolov et al., 2013a), where a target word is predicted given its surrounding context words. We use the vectors that were trained on 100 billion words of Google news dataset and are of $d = 300$.

Affective Embeddings: **(i) Sentiment-specific word embeddings (SSWE)**: These embeddings, obtained using a corpus of 10 million tweets, encode the sentiment information (derived using a set of positive and negative emoticons) of the text in the continuous representation of words[4] (Tang et al., 2014). We use embeddings that were trained with the unified model (SSWE_u). **(ii) DeepMoji**: These word representations were obtained from a corpus of almost 1 billion tweets weakly labeled using a set of 64 emojis (Felbo et al., 2017). **(iii) Retrofitting**: Instead of directly training task-specific affective embeddings such as SSWE and DeepMoji, Retrofitting (Faruqui et al., 2015) is a post-processing technique of tuning existing embeddings according to a task-specific lexicon. Using WNA as the source emotion lexicon, where words with same emotions are clustered together, we apply Retrofitting to the generic word vectors (GloVe and word2vec).

The results of the emotion classification are presented in Table 5, with the generic embeddings model in the top half and affective embeddings in the bottom half. In general, we observe that GloVe 42B yields the second best results overall, and in line with other recent studies (Pool and Nissim, 2016), Retroffiting did not improve over any original word embeddings suggesting that post-processing word embeddings with respect to emotion knowledge requires additional considerations.

Secondly, although SSWE and DeepMoji were both trained on tweets data, they perform very differently to each other, most likely due to their extremely different choices of affect spectrum (SSWE was

[3]https://code.google.com/p/word2vec

[4]http://ir.hit.edu.cn/~dytang/paper/sswe/embedding-results.zip

embedding	$n = 10$	$n = 20$	$n = 30$
SSWE_u	32.6	28.8	28.2
word2vec	35.5	33.1	30.2
GloVe	35.1	32.5	30.4
$\text{EWE}_{\text{UNI(WNA+NRC)}}$	**36.7**	**33.2**	**31.3**

Table 6: Accuracy of emotion similarity tested on emotion lexicon DepecheMood

modeled along binary polarities, whereas DeepMoji used an axis of 64 categories), thus highlighting the importance of the emotion model adopted for creating the training dataset. In addition, and rather surprisingly, all the generic embeddings (GloVe and word2vec) outperform all the affective embeddings (SSWE and DeepMoji) on all the four datasets. One possible reason for this could be due to the more generalizable sources of data that were used to induce the generic embeddings, while the affective embeddings were trained on tweets data, thus showing the significance of the choice of the underlying text used to derive the representations.

Lastly, $\text{EWE}_{\text{UNI(WNA+NRC)}}$ statistically significantly outperforms all the other baselines across all the four datasets, indicating the effectiveness of the proposed method.

4.2 Emotion Similarity

The second task measures the *emotion similarity* of the word vectors by comparing against the emotion similarity obtained from an emotion lexicon. In this experiment, the test affective information is derived from **DepecheMood (DM)** (Staiano and Guerini, 2014), an emotion lexicon consisting of 37,000 words and their emotion scores across eight affective dimensions. We consider the emotion label of a word as the emotion category with the maximum affective weight.

Following previous experimental setup for measuring affective consistency (Tang et al., 2014), we compute the accuracy of emotion similarity consistency between each emotion word and its top n nearest neighboring words as follows:

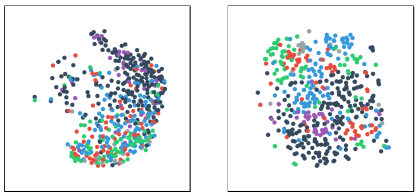
$$Accuracy = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\alpha\left(w_i, c_{ij}\right)}{m \times n} \tag{5}$$

where m is the number of words in the emotion lexicon, w_i is the ith word in the lexicon, c_{ij} is the jth closest word to w_i in terms of their cosine similarity, $\alpha\left(w_i, c_{ij}\right)$ is an indicator function, where $\alpha = 1$ if w_i and c_{ij} belong to the same emotion category and $\alpha = 0$ otherwise. The higher the accuracy, the better the clustering of emotionally similar words in the embedding space.

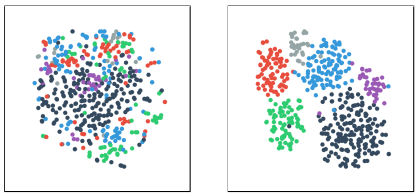
Table 6 presents the results of various embeddings for $n = \{10, 20, 30\}$, where n is the number of nearest neighboring words. For fair comparison, for each word embeddings, only the words that appear in both the vocabularies (i.e., DM and word embeddings) have been used. Again, we observe that generic embeddings such as GloVe and word2vec outperform affective embeddings such as SSWE. The best results are obtained from EWE which have been specifically trained to capture emotion similarity.

5 Qualitative and Error Analysis

To further analyze the learned emotion embedding space, we use t-SNE (van der Maaten and Hinton, 2008) to visualize the word representations of a small subset of words in Figure 2. The plots show that compared to other models, EWE is effective in clustering emotionally similar words into neighboring vector spaces. Figure 3 shows confusion matrix plots providing an overview for some error analysis. In general, for imbalanced datasets such as Alm and Aman, it is observed that most misclassified instances are incorrectly labeled as *happy* class, likely because the *happy* class contains a disproportionately large number of training instances. Moreover, instances belonging to the *surprise* class are more often misclassified than correctly predicted, likely because the *surprise* class is highly underrepresented. Balancing

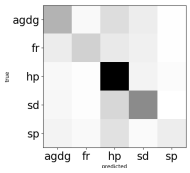


(i) SSWE (ii) word2vec

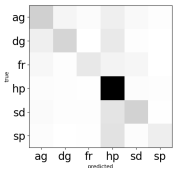


(iii) GloVe (iv) EWE

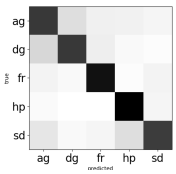
Figure 2: t-SNE visualization of word embeddings



(i) Alm



(ii) Aman



(iii) ISEAR

Figure 3: Confusion matrix error analysis

the datasets might prove helpful. In ISEAR, *anger* and *disgust* classes are found to be confused with each other and *sadness* seems to be challenging.

6 Conclusions

In this paper, we described a novel method of learning emotion-enriched word representations by leveraging distant supervision and neural networks. Significant improvements over baseline representations in two tasks including emotion classification and emotion similarity is obtained. In addition, we presented a qualitative analysis of the learned word vectors. As future work, we plan on considering alternate taxonomies of emotions such as Plutchik's (1980), obtaining emotion-enriched representations at phrase level and exploring sentence compositionality.

Acknowledgments

This work is funded by Natural Sciences and Engineering Research Council of Canada (NSERC) and Big Data Research, Analytics, and Information Network (BRAIN) alliance established by the Ontario Research Fund - Research Excellence Program (ORF-RE).

References

Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. 2005. Emotions from text: Machine learning for text-based emotion prediction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 579–586, Stroudsburg, PA, USA. ACL.

Ebba Cecilia Ovesdotter Alm. 2008. *Affect in Text and Speech*. Ph.D. thesis, University of Illinois at Urbana-Champaign.

Saima Aman and Stan Szpakowicz. 2007. Identifying expressions of emotion in text. In *Proceedings of the 10th International Conference on Text, Speech and Dialogue*.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 809–815.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Felipe Bravo-Marquez, Eibe Frank, Saif M Mohammad, and Bernhard Pfahringer. 2016. Determining word-emotion associations from tweets by multi-label classification. In *Web Intelligence (WI), 2016 IEEE/WIC/ACM International Conference on*, pages 536–539. IEEE.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*. ACL.

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

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, nov.

Paul Ekman. 1992. An argument for basic emotions. *Cognition & Emotion*, 6(3-4).

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1606–1615.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *EMNLP*.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *Processing*, pages 1–6.

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

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June. Association for Computational Linguistics.

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

Igor Labutov and Hod Lipson. 2013. Re-embedding words. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers*, pages 489–493.

Jasy Suet Yan Liew, Howard R. Turtle, and Elizabeth D. Liddy. 2016. Emotweet-28: A fine-grained emotion corpus for sentiment analysis. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.*

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*, pages 43–52. ACM.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Andriy Mnih and Geoffrey E. Hinton. 2008. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 1081–1088.

Saif M. Mohammad and Felipe Bravo-Marquez. 2017. Emotion intensities in tweets. In *Proceedings of the sixth joint conference on lexical and computational semantics (*Sem)*, Vancouver, Canada.

Saif M. Mohammad and Peter D. Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, CAAGET '10, pages 26–34, Stroudsburg, PA, USA. Association for Computational Linguistics.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a word-emotion association lexicon. 29(3):436–465.

W. Parrott, Gerrord. 2001. *Emotions in Social Psychology*. Psychology Press, Philadelphia.

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

Robert Plutchik, 1980. *A general psychoevolutionary theory of emotion*, pages 3–33. Academic press, New York.

Chris Pool and Malvina Nissim. 2016. Distant supervision for emotion detection using facebook reactions. *arXiv preprint arXiv:1611.02988*.

Lizhen Qu, Gabriela Ferraro, Liyuan Zhou, Weiwei Hou, Nathan Schneider, and Timothy Baldwin. 2015. Big data small data, in domain out-of domain, known word unknown word: The impact of word representations on sequence labelling tasks. In *Proceedings of the 19th Conference on Computational Natural Language Learning, CoNLL 2015, Beijing, China, July 30-31, 2015*, pages 83–93.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 151–161.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 455–465.

Jacopo Staiano and Marco Guerini. 2014. Depechemood: a lexicon for emotion analysis from crowd-annotated news. *CoRR*, abs/1405.1605.

Carlo Strapparava and Alessandro Valitutti. 2004. WordNet-Affect: An affective extension of WordNet. In *LREC*, pages 1083–1086.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL*.

Duyu Tang, Furu Wei, Bing Qin, Nan Yang, Ting Liu, and Ming Zhou. 2016. Sentiment embeddings with applications to sentiment analysis. *IEEE Trans. Knowl. Data Eng.*, 28(2):496–509.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics.

Laurens van der Maaten and Geoffrey E. Hinton. 2008. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

Harald G. Wallbott and Klaus R. Scherer. 1986. How universal and specific is emotional experience? evidence from 27 countries on five continents. *Social Science Information*, 25(4):763–795.

Evaluating the text quality, human likeness and tailoring component of PASS: A Dutch data-to-text system for soccer

Chris van der Lee
Tilburg University
c.vdrlee@tilburguniversity.edu

Bart Verduijn
Tilburg University
b.w.verduijn@tilburguniversity.edu

Emiel Krahmer
Tilburg University
e.j.krahmer@tilburguniversity.edu

Sander Wubben
Tilburg University
s.wubben@tilburguniversity.edu

Abstract

We present an evaluation of PASS, a data-to-text system that generates Dutch soccer reports from match statistics which are automatically tailored towards fans of one club or the other. The evaluation in this paper consists of two studies. An intrinsic human-based evaluation of the system's output is described in the first study. In this study it was found that compared to human-written texts, computer-generated texts were rated slightly lower on style-related text components (fluency and clarity) and slightly higher in terms of the correctness of given information. Furthermore, results from the first study showed that tailoring was accurately recognized in most cases, and that participants struggled with correctly identifying whether a text was written by a human or computer. The second study investigated if tailoring affects perceived text quality, for which no results were garnered. This lack of results might be due to negative preconceptions about computer-generated texts which were found in the first study.

1 Introduction

Evaluation of end-to-end Natural Language Generation (NLG) systems is important to assess whether the system has properly expressed certain properties (e.g. quality, speed), or whether the designed properties work as intended (Gkatzia and Mahamood, 2015). Traditional NLG evaluation approaches can typically be assigned to one of two categories: intrinsic or extrinsic (Belz and Reiter, 2006). Intrinsic approaches seek to evaluate properties of the system itself. This can be done using automatic measures such as BLEU, NIST, ROUGE, etc. or by asking human participants to rate the systems output with e.g. Likert or rating scales. Extrinsic approaches aim to assess the impact of the system, by measuring if the system can fulfill its purpose or what the user gains from the systems output.

While scholars have been positive about the effort the NLG community has put into their evaluations (Gatt and Belz, 2010, for instance), it is often the case that an extensive evaluation does not take much priority after a system is built. The usage of automatic measures is gaining traction due to its quickness and low costs, but they are still considered controversial by many (Reiter and Belz, 2009; Novikova et al., 2017, for instance). Furthermore, the intrinsic approaches with human ratings are often limited in scope, using a relatively small sample of participants, using relatively short questionnaires that only shine light on a small aspect of text quality, and/or comparing the computer-generated texts against non-representative human texts. Similarly, the amount of extrinsic evaluations that have been performed up until now is low. While they are considered the most useful type of evaluation by some (Reiter and Belz, 2009), they are not carried out as often as other types of evaluation.

In many cases, an extrinsic evaluation is also difficult because of the system's set-up. A system that is designed to produce texts that stay close to the facts and which purpose is to merely inform readers, should mainly be evaluated on the factual correctness of its output, which is covered by doing an intrinsic evaluation. A different kind of system was chosen as the object of evaluation in the current study: the Personalized Automated Soccer texts System (hereafter: PASS). This system generates summaries of soccer matches based on match statistics (van der Lee et al., 2017). A unique feature of PASS is that

Proceedings of the 27th International Conference on Computational Linguistics, pages 962–972
Santa Fe, New Mexico, USA, August 20-26, 2018.

the system produces texts tailored towards fans of one club or the other. Previously, scholars have emphasized that one of the potential strengths of data-to-text systems is their potential to produce multiple variants of texts based on the same data (Gatt and Krahmer, 2018). This tailoring has been suggested to improve attitudinal and behavioral outcomes in the context of human-written persuasive texts (Noar et al., 2007), but if these positive outcomes can also be found for computer-generated texts and/or texts with aims other than persuasion warrants further research.

PASS is relatively unique in the NLG landscape, because of its aim to not only inform but also to produce a text that is 'enjoyable' to read by its target audience, which it tries to accomplish by tailoring texts towards the wishes of an audience. These goals make a brief check for readability of its texts inadequate, and make a more extensive evaluation project necessary. The current study examined whether the PASS system succeeds in the aim to produce texts of significant linguistic quality and if the tailoring component plays a role in this aim, similar to how tailoring improves attitudinal and behavioral responses for human-written persuasive texts. To examine this, the goals of this paper were twofold. The first goal was to assess how the quality of PASS-generated texts fares against similar human texts, and the second goal was to assess whether the intended tailoring was clear and if it had effects on attitude towards the text. Besides these two main points, differences in preconceptions about human-written and computer-generated texts were also investigated, because these preconceptions could moderate the effectiveness of tailoring.

2 Background

2.1 Evaluation in NLG

Evaluation has become an increasingly important topic within the NLP domain as a whole, but also within the NLG domain more specifically. While the NLG domain has a strong evaluation tradition (Gatt and Belz, 2010), there is an ongoing discussion regarding the type of evaluation that should be used. One popular evaluation method is the use of metrics that can be computed automatically, such as BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), and NIST (Doddington, 2002). However, previous literature suggests that the use of such metrics should be done with caution. These metrics are attractive because they are quick, fast and repeatable (Reiter and Belz, 2009), and have in some cases been found to correlate with human judgments (Gkatzia and Mahamood, 2015). However, most if not all of the current automated metrics that are used in the NLG domain are based on overlap with a certain reference text. Therefore, it can be derived that such metrics are only useful if an aligned NLG output-reference text corpus can be constructed and if it can be assumed that qualitatively good output has a strong overlap with this standard. This will not be the case in many situations. Furthermore, these metrics also rely on the assumption that the reference text is a good representation of the 'best case scenario, which it often is not (Reiter and Sripada, 2002). Finally, it has been argued that such metrics do not provide a good assessment of many linguistic properties such as content selection, information structure, appropriateness, etcetera (Scott and Moore, 2007).

While use of automatic metrics is increasing (Gkatzia and Mahamood, 2015), the above concerns might contribute to the fact that evaluation using human input is still the most popular evaluation method in the NLG domain. Most of these evaluations have been quantitative (Sambaraju et al., 2011), but there can be sizable differences between these quantitative evaluations. A main distinction can be made between intrinsic and extrinsic methods (Hastie and Belz, 2014). Intrinsic methods evaluate the output of the system itself, for example by having humans read and rate text output of the NLG system and comparing these ratings against human written texts on metrics such as fluency, correctness, understandability, etcetera. Extrinsic measures aim to evaluate the impact of the system, for instance by measuring whether a system can fulfill the purpose it was built for (Hastie and Belz, 2014). A corpus analysis by Gkatzia and Mahamood (2015) shows that intrinsic human-based measures are used to a much greater degree compared to extrinsic human-based evaluations, although increasingly more effort regarding the latter evaluation can be observed in the past few years (Gkatzia et al., 2017; Goldstein et al., 2017; Ramos-Soto et al., 2017, for instance)

These extrinsic task-focused evaluations have traditionally been regarded as the type of evaluation

that provides the most meaningful results. However, it can be an expensive and timely undertaking to execute such an evaluation (Reiter and Belz, 2009). Another reason why extrinsic evaluations are not used as often is because in many cases, the purpose of the system itself makes such an evaluation unfit to use. An intrinsic evaluation measuring correctness and grammaticality would often be sufficient for systems that are designed to provide a general audience information in a straightforward manner. However, one of the strengths of data-to-text systems is the fact that they can tailor texts towards specific audiences relatively effortlessly. A notable example of this is the BabyTalk project (Gatt et al., 2009), where separate reports about babies in a Neonatal Intensive Care Unit (NICU) are generated for physicians, nurses and the baby's family based on data in the baby's electronic medical record. Recently, an increasing trend might also be observed in methods that can produce a text without too much human input (e.g. sequence-to-sequence models), which would make the development of systems with this feature much easier. Implementation of a tailoring feature, however, also means that the purpose of the system extends beyond merely informing a general audience. Tailoring is often implemented to increase behavior or attitude (Noar et al., 2007, for an overview), which would warrant a more extensive evaluation to measure the outcomes of the computer-generated texts. Furthermore, the effects of tailoring have mainly been investigated using human-written texts. It might be possible that people respond differently to computer-generated texts, for instance because they have different expectations of texts written by computers compared to human-written texts (Graefe et al., 2016).

2.2 PASS and 'Affective' Natural Language Generation

While an increase in NLG systems with a tailoring component might be expected, one of the few systems in this category is PASS (van der Lee et al., 2017). PASS generates soccer match summaries aimed at fans of one of the teams that participated in the match. Thus, the system can be seen as part of the 'Affective' NLG (ANLG) tradition, which aims to produce texts tailored towards the emotional aspects of the intended reader (Mahamood and Reiter, 2011; Ghosh et al., 2017, for instance). The input data is scraped from Goal.com and contains various types of data related to a soccer match (e.g. date played, goals scored, information about the players). Based on this data, a short Dutch match summary of about 50 to 150 words is produced, which is inspired by the reports of the GoalGetter system (Theune et al., 2001). However, although inspired by previous work, texts by PASS are novel in the sense that the templates have been directly derived from sentences in the MeMo FC corpus (Braun et al., 2016). This corpus contains match reports directly taken from the clubs that participated in the match. This often means that the tone of voice in these reports is emotional, while still maintaining a relatively professional style. Using these reports makes it possible to produce tailored match reports with PASS. This means that the tone of a generated report should appear to be more disappointed or frustrated in case the team of the target audience lost, and more upbeat in case of a win for the team of the target audience (van der Lee et al., 2017, for examples).

Empirical evaluation of ANLG systems is considered challenging (Belz and others, 2003; Mahamood and Reiter, 2011) and very few ANLG systems have been tested properly at this point in time. PASS is no exception to that. The lack of extensive empirical testing of that system makes it difficult to determine how well the system performs in terms of text quality and the effectiveness and importance of its 'emotional' tailoring component. Therefore, a more extensive evaluation is necessary in order to assess the perception of the generated texts and the consequences of tailoring.

2.3 Current challenges

Part of what makes evaluation of ANLG systems challenging is finding the right material to compare the output to. One way to compare the output quality of PASS to human texts would be to use texts in the MeMo FC corpus about the same soccer matches, written for the same target audience as reference texts. However, questions arise whether the texts in the MeMo FC corpus are a suitable equivalent for the texts produced by PASS. The texts in the MeMo FC corpus are usually match reports written by people that watched the full soccer match. Thus, it can be assumed that the writers of MeMo FC corpus texts had much more input data to base their reports on. Furthermore, there are no standard guidelines (e.g. text length, text structure) that all writers for soccer team websites adhere to, meaning that the style and

quality of texts in the MeMo FC corpus can be vastly different depending on the writer and the website it was written for.

The use of automated metrics was also deemed to be a non-viable option for the current study if texts from the MeMo FC corpus serve as reference texts, since sentences from the MeMo FC corpus are used as the basis for the templates in PASS. This would make it likely that much overlap occurs between the generated texts and the reference texts, resulting in high scores on metrics such as BLEU and METEOR. However, these scores would say little about how well these templates blend together to form an enjoyable and coherent text that displays the information in a sensible way (Scott and Moore, 2007).

Finding the right reference texts is not the only challenge when evaluating ANLG systems. The effects of tailoring the generated text towards the emotional needs of the intended audience has not received a lot of attention yet. Most research has focused on the effects of tailoring in the context of persuasive messages. A meta-analysis by Noar et al. (2007) shows that tailoring can improve the attitude towards a message and behavioral outcomes, although a relatively small mean effect size for tailoring interventions was found. In the context of texts that aim to achieve other goals rather than persuade, much less is known about the effects of tailoring. Especially in the NLG domain. One of the notable studies in this domain that attempts to study the effects of tailoring is by Reiter et al. (2003) who studied the effects on smoking behavior that tailored smoking cessation letters had. They did not find significant behavioral differences between people that received a tailored vs. non-tailored letter. However, it might be argued that such behavioral changes are difficult to achieve with only a letter. Furthermore, the tailoring aspect was elaborated by tailoring the arguments to the specific challenges that the reader said to face, which is a different kind of tailoring compared to the tailoring of PASS where the style and diction was tailored to fit with the emotions that the reader experiences. This difference in tailoring could also result in different effects.

A study similar to the current study is that of Wann and Branscombe (1992), who investigated mood changes after reading a (human-written) tailored basketball summary. They found that if participants identified strongly with a team involved in the match and if the text was tailored towards fans of that team, the emotions were the most extreme: participants reported the most positive mood if the team won and the most negative if the team lost. Tailoring did not have a significant effect on mood if the participants did not identify strongly with the team. Mahamood and Reiter (2011) found that affective texts were also preferred to non-affective texts and that emotional appropriateness of texts also correlated with understanding ratings when investigating generated BabyTalk texts. These results support the basic premise of PASS that tailoring of sports reports results in behavioral and attitudinal outcomes that non-tailored texts do not achieve. The current study investigated this more deeply, while also investigating the overall perceived text quality of the system. Furthermore, possible moderators of the effectiveness of computer-generated texts on attitudinal and behavioral responses were investigated by looking into possible differences in preconceptions between human-written and computer-generated texts.

3 Evaluation

As previously noted, the aim of the current paper was to study several components of PASS via two evaluation studies. The first study was a human evaluation of the systems output compared to human-written texts. The second study assessed the impact of PASSs tailoring feature. Both evaluation methods will be described more thoroughly in this section.

3.1 Evaluation of text-related attributes and preconceived notions

In this study, texts generated by PASS were compared to human-written texts regarding the perceived text quality, the evidentness of tailoring, and whether participants can accurately identify the writer to be a human or a computer. Furthermore, preconceived notions about human-written and computer-generated texts were investigated.

3.1.1 Participants

A total of 60 people participated in the study (35 women). All these participants were native Dutch college students and had an average age of 21 years and 9 months.

3.1.2 Design

Participants were asked to rate a total of 5 text-pairs (5 human texts and 5 texts generated by PASS). The human texts were written by 14 journalism and communication students, that together, wrote a total of 22 texts. These writers were given the same match statistics as PASS uses to generate a text and they were instructed to write a soccer match summary of a similar size as PASSs reports based on this data. Furthermore, the writers were asked to imagine that they were writing the reports for the club website of one of the two teams that participated in the match, thus writing one soccer report per match. No writers were involved in the rating task. All these instructions were implemented to get the foundation of the human-written text to be fairly equal to the PASS-generated text. The written soccer matches were about soccer matches played in the Dutch second league in the 2015/2016 season. The teams in this division are generally more obscure teams, which would minimize the chance that people's ratings are affected by their love or hatred for one of the teams involved in the match. Furthermore, the computer-generated texts were about the same Dutch second division matches from 2015/2016 and written for the same target audience as the human-written texts. The participants were randomly assigned to one of two versions where each version contained 5 different text-pairs. Counterbalancing was also applied to reduce order bias: so half of the participants in each version received the text-pairs in opposite order from the other half.

Not only were participants provided with a match report, they were also given the match data so that they were able to rate the completeness and accurateness of the information discussed in the report.

3.1.3 Procedure

The study was conducted using *Qualtrics*: an online platform to design surveys. The procedure for all participants was the same. The experiment started with a written instruction and consent form, after which the experiment started. On every page the participants were provided with match statistics and an accompanied text written by either a human or by PASS. The match data was shown so that participants were able to rate the correctness of the information discussed in the report.

After viewing the match data and reading these match reports, participants were asked to indicate whether they thought the text was written for fans of the home team or the away team, whether they thought the text was written by a human or generated by a computer, and why they thought the text was written by a human or generated by a computer. This last question asked for free-text comments to obtain information about preconceived notions participants have in regards to differences between human-written and computer-generated texts. Previous research has shown that these free-text comments often provide valuable insights about generated texts (Reiter and Belz, 2009). These comments were structured based on the text components of (Callaway and Lester, 2002): style (comments on the overall writing style), grammaticality (comments on the syntactic quality of the text), flow (comments on the fluency of the sentences), diction (comments on word choices), readability (comments on how easy to read the text is), logicality (comments on the aptness of the text structure and if information is correctly represented), detail (comments on the amount of detail in the text), and other (uncategorized comments). Furthermore positive, negative, and preconceptions with unclear valence were distinguished for every category.

The participants judged the quality of each text using seven-point Likert-scales on clarity: how clear and understandable the report is ('While reading, I immediately understood the text), fluency: how fluent and easy to read the report is ('This text is written in proper Dutch, 'This text is easily readable), and correctness: how well the information the report is based on is represented in the report itself ('This report does not include extraneous or incorrect information, 'This report does not omit important information).

3.2 Evaluation of tailoring effects

In this study, the effects of tailoring in a soccer match report on perceived text quality was investigated.

3.2.1 Participants

Native Dutch fans of the three most popular and successful soccer teams in the Netherlands: *Ajax*, *Feyenoord*, and *PSV*, were recruited via *Crowdflower*, these clubs were chosen because previous research has found that success increases fan identification (Wann et al., 1994). This resulted in a total of 171 participants (118 male, average age of 29 years and six months). Most of them were Ajax-fans (99), followed by PSV (55) and Feyenoord (47). Supporters of all three teams identified themselves with the club to a similar degree ($F(2, 168) = 1.77$, $p = .17$). Participants also had different educational backgrounds (72 participants college or university; 99 lower education level).

3.2.2 Design

Participants in this study were asked to read and rate a total of four match reports generated by PASS. The match reports that were presented to the participants was based on the club that they supported. The team they supported was involved in all four instances. A between-subjects design with two conditions was used (text tailored towards the team the participant identifies with, or text tailored towards the team the participant does not identify with).

Participants in all conditions got to see similar match summaries. The reports were based on actual matches played by Ajax, Feyenoord and PSV in the 2015-2016 season. By presenting matches based on actual matches, the aim was to present summaries that are realistic to the participants, but not recent enough for participants to remember the match. Perspective was manipulated by generating two reports with PASS, tailored towards fans of each respective team that participated in the match. The matches shown to participants were chosen randomly from the matches that Ajax, Feyenoord and PSV played in the 2015-2016 season, with the exception of any matches that the team they identified with played against rivaled teams, since these matches could result in more extreme ratings (Cialdini et al., 1995). Similarly, no matches of clubs that have been relegated since the 2015-2016 season were shown to avoid that participants view these matches as unrealistic.

3.2.3 Procedure

Participation of this study was done via an online *Qualtrics* survey. After receiving instructions and filling out a consent form, the participants were asked via multiple choice to indicate whether they had a preference for Ajax, Feyenoord, or PSV. After indicating their preference, fans were asked to read and rate four texts. The texts they received were based on their team preference: all match reports shown involved the preferred team.

After reading a match report, participants were asked to rate the text on 10 7-point semantic differentials based on Maes et al. (1996). Five differentials covered an aesthetic judgment on the text ('uninteresting/interesting', 'detached/appealing', 'distant/inviting', 'boring/engaging', 'impersonal/personal', 'monotonous/varied') and five covered clarity ('difficult/easy', 'complicated/simple', 'unclear/clear', 'complex/clear', 'illogically structured/logically structured', 'cumbrous/concise').

4 Results

4.1 Evaluation of text-related attributes and preconceived notions

4.1.1 Text quality

A two-way repeated-measures ANOVA was executed to investigate an effect of computer-generated vs. human-written texts on perceptions of clarity, fluency and correctness. The results showed main effects for all three text quality components (clarity: $F(1, 59) = 9.448$, $p = .003$; fluency: $F(1, 59) = 8.656$, $p = .005$; correctness: $F(1, 59) = 8.302$, $p = .006$). The participants rated the human-written texts as more clear and fluent compared to its computer-generated counterparts. Conversely, the computer-generated counterparts gave a more precise overview of the information it is trying to convey although scores in both categories were low. These low scores might be due to the fact that most human-written as well as most computer-generated texts did not express all the match data that was shown to raters. While the differences were significant, it should also be noted that the differences were relatively small: for all

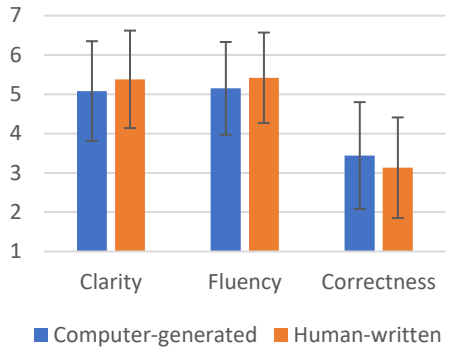


Figure 1: Average scores on clarity, fluency and correctness for the human-written and computer-generated texts on a 7-point Likert-scale. Error bars represent standard deviations.

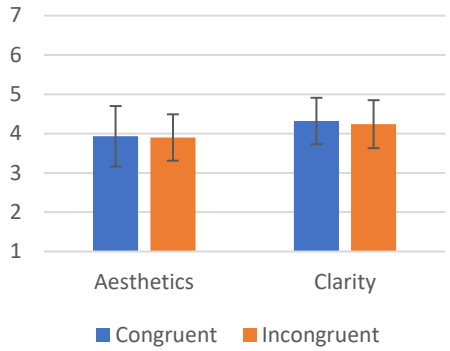


Figure 2: Average scores on aesthetics and clarity for the texts congruent and incongruent with team preference of participants on a 7-point Likert scale. Error bars represent standard deviations.

three text components, the difference between the computer-generated and human-written text was only around 0.3 on a 7-point scale; cf. Figure 1. This would suggest that PASS-generated texts are not that far off the text quality of human texts.

Text type	Correct perception	Incorrect perception
Computer	246	55
Human	228	72

Table 1: Cross-tab comparing the participants' correct and incorrect tailoring perceptions for computer-generated and human-written texts

Text type	Computer (perceived)	Human (perceived)
Computer	121	180
Human	94	206

Table 2: Cross-tab comparing the participants' perceived type of text versus the actual text type (computer-generated or human-written)

4.1.2 Perceived tailoring

In general, participants were able to correctly identify the target audience the text was tailored to. In almost 79% of all cases, the participants' perceived target audience was congruent with the intended target audience. An additional chi-square test did not show a significant difference in the evidentness of tailoring between the computer-generated and human-written texts ($\chi^2(1) = 2.96$, $p = .09$). The tailoring was similarly clear for computer-generated texts as for human-written texts; cf. Table 1.

4.1.3 Perceived text type

Results of a chi-square test showed a correlation between the perceived type of writer and the actual type of writer ($\chi^2(1) = 5.14$, $p = .02$). The origin of human-written texts were more often perceived as human and less often as computer compared to computer-generated texts. However, subsequent analysis made it clear that participants had trouble with this task. Mostly for the computer-generated texts: in less than half of the cases were people able to correctly identify the computer-generated text as such (in 40% of cases). Similar to 4.1.1, these results suggest that the texts generated by PASS contains human-like qualities, which was also further investigated in 4.1.4; cf. Table 2.

4.1.4 Free text comments

The free text comments - structured using the aforementioned categories based on (Callaway and Lester, 2002) - revealed clear differences in preconceived notions, most evidently between human-written and computer-generated texts. This was corroborated by a chi-square test for positive and negative notions ($\chi^2(1) = 391.09$, $p < .001$): participants generally had a much more positive stance towards human-written texts compared to computer-generated texts for nearly every text component. People tend to think of human-written texts as more emotional, dynamic, and well-written and computer-generated texts as more static, boring and poorly-written. This was expressed in the style, flow, and readability category. Interestingly, these judgments were not always justified, as 3 and 4 show. Human-written texts

		Positive	Negative	Unclear	Total
	Style	47	0	5	52
	Grammaticality	9	3	2	14
	Flow	8	0	0	8
Incorrectly	Diction	55	0	12	67
Perceived	Readability	10	0	0	10
Human	Logicality	11	9	3	23
	Detail	0	0	6	6
	Other	0	0	5	5
	Total	140	12	33	185
	Style	1	35	6	42
	Grammaticality	0	6	0	6
	Flow	1	3	0	4
Incorrectly	Diction	0	15	7	22
Perceived	Readability	0	2	0	2
Computer	Logicality	4	6	2	12
	Detail	0	10	2	12
	Other	0	0	3	3
	Total	6	77	20	103

Table 3: Frequency of the type of comments in support of participants' incorrectly perceived text type

		Positive	Negative	Unclear	Total
	Style	59	0	5	64
	Grammaticality	8	8	1	17
	Flow	13	0	0	13
Correctly	Diction	62	0	11	73
Perceived	Readability	13	0	0	13
Human	Logicality	16	5	5	26
	Detail	4	2	7	13
	Other	0	0	7	7
	Total	175	15	36	226
	Style	1	53	7	61
	Grammaticality	1	6	1	8
	Flow	0	6	0	6
Correctly	Diction	0	12	5	17
Perceived	Readability	2	4	1	7
Computer	Logicality	4	31	0	35
	Detail	0	4	0	4
	Other	0	0	3	3
	Total	8	116	17	141

Table 4: Frequency of the type of comments in support of participants' correctly perceived text type

were often misjudged as computer-generated, with substantiations such as *less personal*, *boring* and *no emotion*. Similarly, computer-generated texts were mistaken for human-written because they were *vividly written*, *enthusiastically written* and contained *emotions*. Thus, the intended design of PASS, which was to produce texts that are similar to human in style seems to be successful according by these comments.

Further support for this was found in the comments on diction. Participants expected human texts to contain more varied, figurative language and more adjectives (e.g. *use of proverbs*, *lots of variation in word choice*, *use of adjectives*), while computer-generated language was expected to be factual, simple, and sometimes illogical (e.g. *contains lots of numbers*, *illogical word choices*, *simple language use*). Examples of these comments were also found in correct and incorrect attributions of the text type, further suggesting that the computer-generated texts used in this study had similar qualities as human-written texts.

Participants also expected computers to be wrong more often in terms of syntax and information presented. Examples like *contains many errors*, and *grammatically incorrect* were used to explain why the text was perceived as a computer-generated text. These results are especially interesting, because these preconceptions contradict the findings in 4.1.1 where correctness for computer-generated texts was rated (slightly) higher compared to human-written texts.

4.2 Evaluation of tailoring effects

An independent-samples MANOVA was performed on the average scores of the four rated texts. A distinction between aesthetics and clarity was made in the test. For both these components, the MANOVA did not show a significant effect of tailoring congruity (aesthetics: $F(1, 167) = 0.11$, $p = .74$; clarity: $F(1, 167) = 0.61$, $p = .45$). Participants did not find the text to be aesthetically different if the tailoring was congruent with their preference or if it was not. Similarly, participants did not perceive the text as more clear if they were part of the audience tailored to or if they were not; cf. Figure 2. Thus, while the tailoring component was clear, no effects of tailoring were found on attitude towards the text.

5 Discussion

Various aspects of PASS, a data-to-text system that converts soccer match data into a textual soccer match summary, have been evaluated in this paper (van der Lee et al., 2017). PASS is relatively unique in the NLG landscape in the sense that it tailors texts towards specific subgroups. In the case of PASS: supporters of one team or the other involved in a soccer match. This tailoring component is enabled by using texts from the MeMo FC corpus, which contains soccer reports for club websites written in a more emotional tone-of-voice compared to the more neutral newspaper reports. By incorporating a tailoring component, it can be argued that the purpose of PASS is not to merely inform readers of a soccer match, but also to entertain, which required further investigation. Tailoring has been found to increase behavioral and attitudinal outcomes (Noar et al., 2007), and tailored sports reports have been

found to increase emotions for people who identify with a participating team (Wann and Branscombe, 1992). However, if tailoring-related effects could be found for computer-generated texts and if tailoring also affects perceived text quality warranted further investigation. In sum, the aim of the paper was to perform a more extensive evaluation of PASS by comparing its output to similar human-written texts, and to investigate if the attempted tailoring was clear and if it had any attitudinal effects. Furthermore, differences in preconceived notions between human-written and computer-generated texts - a possible moderating factor in the effectiveness of tailoring - was explored. These aims were investigated using two evaluation studies.

The first evaluation study showed differences in perceived quality between human-written and computer-generated soccer match summaries from PASS. The results showed that the language use of the human-written texts was found to be more fluent and easy to read, as well as more clear and understandable. However, the computer-generated texts gave a better overview of the match-data it was based on. The differences in perceived text quality between the human-written and computer-generated texts might be due to the fact that PASS texts are inspired by texts from GoalGetter - a data-to-text system that attempted to produce a neutral and factual match summary (Theune et al., 2001) - in terms of its text structure and the types of data that are incorporated into a text. Perhaps, if the aim of PASS is to entertain rather than inform, it might be fruitful to stray further away from the structure and data use of GoalGetter texts. Interesting, however, was the fact that despite differences in information representation and language use, participants had trouble identifying the computer-generated texts as such: these texts were incorrectly marked as a human text in 60% of cases, which does suggest that the current state of PASS-generated texts is of good quality. Another reason to think that the PASS-generated texts are effective in their goal is the fact that the intended tailoring was correctly identified by participants during evaluations, to a similar degree as the tailoring in human-texts.

However, the second study did not show any effects of tailoring on perceived text quality. Texts that were tailored towards the preferences of the participant were neither seen as more aesthetically pleasing nor as more understandable. Thus, no support was found for tailoring to have an effect on the attitude towards the text. A reason for this might be the findings in the free text comments of the first study. These comments showed that people have negative preconceived notions about computer-generated texts. They are generally expected to be written with boring and unemotional language in a predictable and static style. Knowing that the reports were computer-generated might have biased the participants in the second study to think that the reports were lacking emotions, thus making it harder to achieve tailoring effects. However, it should be noted that this lack of result in the second study does not necessarily mean that there is no effect of tailoring in PASS texts. Previous research has shown that tailored human-written texts about sports matches did increase emotional experiences for participants the texts were tailored towards (Wann and Branscombe, 1992), which could suggest that the PASS texts affect emotional experiences rather than opinions on the text itself. Furthermore, (Mahamood and Reiter, 2011) found that 'emotional appropriateness' correlated with understandability and that 'tailored' texts scored higher on all aspects of text quality compared to 'neutral' texts, which are aspects that are still worth investigating in the context of PASS. There is also a lack of research looking into the relationship between extrinsic evaluation and intrinsic evaluation (Gatt and Belz, 2010), which could be an interesting avenue for further research. Furthermore, the current quantitative human-based approaches could be combined with a qualitative analysis of the output texts similar to McKinlay et al. (2010) and Sambaraju et al. (2011) in order to get a better sense which components of PASS should be improved and how they should be improved.

With the study described in this article, a more extensive evaluation of a data-to-text system was executed than is conventional in the NLG domain, with a few notable exceptions (Reiter and Belz, 2009; Reiter et al., 2003, for instance). We feel that similar evaluations and further research into evaluation methods should be greatly encouraged. Over the last few years, interest in NLG systems has been increasing in domains outside academia as well, such as the journalism domain, and many people expect that computer-generated texts will become more visible in everyday life. In light of these developments, increasing the quality and quantity of evaluations is necessary for a better understanding of the state of these NLG-systems and the role they should play alongside human writers.

Acknowledgements

We received support from RAAK-PRO SIA (2014-01-51PRO) and The Netherlands Organization for Scientific Research (NWO 360-89-050), which is gratefully acknowledged. Thanks are due to Martijn Goudbeek for help in setting up in the experiments, Nadine Braun for providing us with the corpus, Monique Hamers for helping us find young journalists and the three reviewers for their comments.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72.

Anja Belz et al. 2003. And now with feeling: Developments in emotional language generation. *Information Technology Research Institute Technical Report Series*.

Anja Belz and Ehud Reiter. 2006. Comparing automatic and human evaluation of NLG systems. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 313–320. Association for Computational Linguistics.

Nadine Braun, Martijn Goudbeek, and Emiel Krahmer. 2016. The Multilingual Affective Soccer Corpus (MASC): Compiling a biased parallel corpus on soccer reportage in English, German and Dutch. In *Proceedings of the 9th International Natural Language Generation Conference*, pages 74–78.

Charles B Callaway and James C Lester. 2002. Narrative prose generation. *Artificial Intelligence*, 139(2):213–252.

Robert B Cialdini, Melanie R Trost, and Jason T Newsom. 1995. Preference for consistency: The development of a valid measure and the discovery of surprising behavioral implications. *Journal of Personality and Social Psychology*, 69(2):318.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, pages 138–145. Morgan Kaufmann Publishers Inc.

Albert Gatt and Anja Belz. 2010. Introducing shared tasks to NLG: The TUNA shared task evaluation challenges. In *Empirical Methods in Natural Language Generation*, pages 264–293. Springer.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Albert Gatt, Francois Portet, Ehud Reiter, Jim Hunter, Saad Mahamood, Wendy Moncur, and Somayajulu Sripada. 2009. From data to text in the neonatal intensive care unit: Using NLG technology for decision support and information management. *Ai Communications*, 22(3):153–186.

Sayan Ghosh, Mathieu Chollet, Eugene Laksana, Louis-Philippe Morency, and Stefan Scherer. 2017. Affect-lm: A neural language model for customizable affective text generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 634–642.

Dimitra Gkatzia and Saad Mahamood. 2015. A snapshot of NLG evaluation practices 2005-2014. In *Proceedings of the 15th European Workshop on Natural Language Generation (ENLG)*, pages 57–60. Association for Computational Linguistics.

Dimitra Gkatzia, Oliver Lemon, and Verena Rieser. 2017. Data-to-text generation improves decision-making under uncertainty. *IEEE Computational Intelligence Magazine*, 12(3):10–17.

Ayelet Goldstein, Yuval Shahar, Efrat Orenbuch, and Matan J Cohen. 2017. Evaluation of an automated knowledge-based textual summarization system for longitudinal clinical data, in the intensive care domain. *Artificial Intelligence in Medicine*, 82:20–33.

Andreas Graefe, Mario Haim, Bastian Haarmann, and Hans-Bernd Brosius. 2016. Readers perception of computer-generated news: Credibility, expertise, and readability. *Journalism*, pages 1–16.

Helen Hastie and Anja Belz. 2014. A comparative evaluation methodology for NLG in interactive systems. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Alfons Maes, Mathilde Maria Nicoline Ummelen, and Hans Hoeken. 1996. *Instructieve teksten. Analyse, ontwerp en evaluatie*. Uitgeverij Coutinho.

Saad Mahamood and Ehud Reiter. 2011. Generating affective natural language for parents of neonatal infants. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 12–21. Association for Computational Linguistics.

Andy McKinlay, Chris McVittie, Ehud Reiter, Yvonne Freer, Cindy Sykes, and Robert Logie. 2010. Design issues for socially intelligent user interfaces. *Methods of Information in Medicine*, 49(04):379–387.

Seth M Noar, Christina N Benac, and Melissa S Harris. 2007. Does tailoring matter? Meta-analytic review of tailored print health behavior change interventions. *Psychological bulletin*, 133(4):673.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

A Ramos-Soto, J Janeiro, JM Alonso, A Bugarin, and D Berea-Cabaleiro. 2017. Using fuzzy sets in a data-to-text system for business service intelligence. In *Advances in Fuzzy Logic and Technology 2017*, pages 220–231. Springer.

Ehud Reiter and Anja Belz. 2009. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4):529–558.

Ehud Reiter and Somayajulu Sripada. 2002. Should corpora texts be gold standards for NLG? In *Proceedings of the International Natural Language Generation Conference*, pages 97–104.

Ehud Reiter, Roma Robertson, and Liesl M Osman. 2003. Lessons from a failure: Generating tailored smoking cessation letters. *Artificial Intelligence*, 144(1-2):41–58.

Rahul Sambaraju, Ehud Reiter, Robert Logie, Andy McKinlay, Chris McVittie, Albert Gatt, and Cindy Sykes. 2011. What is in a text and what does it do: Qualitative evaluations of an NLG system –the BT-Nurse– using content analysis and discourse analysis. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 22–31. Association for Computational Linguistics.

Donia Scott and Johanna Moore. 2007. An NLG evaluation competition? Eight reasons to be cautious. In *Proceedings of the Workshop on Shared Tasks and Comparative Evaluation in Natural Language Generation*, pages 22–23.

Mariët Theune, Esther Klabbers, Jan-Roelof de Pijper, Emiel Krahmer, and Jan Odijk. 2001. From data to speech: A general approach. *Natural Language Engineering*, 7(1):47–86.

Chris van der Lee, Emiel Krahmer, and Sander Wubben. 2017. PASS: A Dutch data-to-text system for soccer, targeted towards specific audiences. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 95–104. Association for Computational Linguistics.

Daniel L Wann and Nyla R Branscombe. 1992. Emotional responses to the sports page. *Journal of Sport and Social Issues*, 16(1):49–64.

Daniel L Wann, Thomas J Dolan, Kimberly K MeGeorge, and Julie A Allison. 1994. Relationships between spectator identification and spectators' perceptions of influence, spectators' emotions, and competition outcome. *Journal of Sport and Exercise Psychology*, 16(4):347–364.

Answerable or Not: Devising a Dataset for Extending Machine Reading Comprehension

Mao Nakanishi **Tetsunori Kobayashi** **Yoshihiko Hayashi**
School of Science and Engineering, Waseda University
Waseda-machi 27, Shinjuku, Tokyo 1690042, Japan
nakanishi@pcl.cs.waseda.ac.jp koba@waseda.jp yshk.hayashi@aoni.waseda.jp

Abstract

Machine reading comprehension (MRC) has recently attracted attention in the fields of natural language processing and machine learning. One of the problematic presumptions with current MRC technologies is that each question is assumed to be answerable by looking at a given text passage. However, to realize human-like language comprehension ability, a machine should also be able to distinguish not-answerable questions (NAQs) from answerable questions. To develop this functionality, a dataset incorporating hard-to-detect NAQs is vital; however, its manual construction would be expensive. This paper proposes a dataset creation method that alters an existing MRC dataset, the Stanford Question Answering Dataset, and describes the resulting dataset. The value of this dataset is likely to increase if each NAQ in the dataset is properly classified with the difficulty of identifying it as an NAQ. This difficulty level would allow researchers to evaluate a machine's NAQ detection performance more precisely. Therefore, we propose a method for automatically assigning difficulty level labels, which basically measures the similarity between a question and the target text passage. Our NAQ detection experiments demonstrate that the resulting dataset, having difficulty level annotations, is valid and potentially useful in the development of advanced MRC models.

1 Introduction

Language understanding is one of the ultimate goals of artificial intelligence. Because the machine understanding of human language is hard to define, one often presumes that if a machine can answer a set of questions under given circumstances, then it understands language. Machine reading comprehension (MRC) developed along this direction and has recently attracted a great deal of attention in natural language processing (NLP) and machine learning.

The tasks of MRC are defined in various ways (Richardson et al., 2013; Mostafazadeh et al., 2017; Weston et al., 2015; Chen et al., 2016; Rajpurkar et al., 2016). Some tasks (Chen et al., 2016; Hill et al., 2016) provide a set of fill-in-the-blank style questions. The most prominent tasks (Rajpurkar et al., 2016), however, allow a machine to locate an answer span in given text passages. It is a highly efficient scheme, because the task can be formulated by the recognition of the start and end positions of the passage. This model facilitates the development of machine learning approaches and the automatic evaluation of performances. However, as criticized by (Jia and Liang, 2017), a machine can succeed in a task of this kind by remembering and recalling linguistic patterns that are prominent in a target dataset. This means that machine comprehension of this type would be an insufficient measure for assessing the degree of machine understanding of language.

To tackle this issue, Jia and Liang (2017) developed a dataset, in which passages were altered to include adversarial sentences that could lead to wrong answers. Contrary to their approach, which modified text passages, this paper proposes to extend MRC by incorporating questions for which relevant answers cannot be found in the given text passages. These questions are called *not-answerable questions (NAQs)*.

Proceedings of the 27th International Conference on Computational Linguistics, pages 973–983
Santa Fe, New Mexico, USA, August 20-26, 2018.

This partly meets our expectation, which can be stated as, "a good MRC system, given text passages, should properly detect and reject NAQs."

To develop an appropriate functionality, a dataset that accommodates NAQs and answerable questions is mandatory. Furthermore, each NAQ in the dataset should not be easily detected. However, the manual construction of such a dataset would be very costly. Therefore, this paper proposes an automatic dataset creation method for incorporating NAQs by making use of an existing MRC dataset. We created such a dataset by modifying a popular MRC dataset, the Stanford Question Answer Dataset (SQuAD) (Rajpurkar et al., 2016).

The value of such a dataset would increase if each NAQ in the dataset were to be graded according to the difficulty level used to determine question as not-answerable. These difficulty level annotations would allow us to choose a subset where the difficulty levels of NAQs could be adjusted as required. Therefore, we propose a method for automatically assigning difficulty level labels to NAQs. The proposed method employs a set of similarity features that are highly effective in determining the not-answerability of a question. NAQ detection experiments, employing an answerable/not-answerable binary classifier and a "null-answer detector" are conducted, showing that the resulting dataset with its respective difficulty level labels can be effectively used in the study of the genuine machine understanding of a language.

2 Related Work

Recently, several MRC datasets have been developed. The characteristics of each MRC dataset varies, depending on the purpose. For example, the Childrens Book Test dataset (Hill et al., 2016) was built from books for children, where the 21st sentence that follows the preceeding 20 sentences is employed as a "question." Additionally, the CNN/Daily dataset (Chen et al., 2016) collects news articles with bullet-pointed summaries, in which each summary is converted into a question. Among the many MRC datasets, this section introduces two datasets, WIKIQA (Yang et al., 2015) and SQuAD (Rajpurkar et al., 2016). Of these, the latter is used in this work. Additionally, we address potential problems with SQuAD.

2.1 Related MRC Datasets

WIKIQA: WIKIQA is one of the few MRC datasets that contains questions without answers, which is a main concern of this paper. WIKIQA was created to capture the characteristics of natural queries asked by people in the real world. Each question in this dataset was taken from a search engine's actual query log, and the corresponding Wikipedia summary paragraph is employed as a text passage, where each of the contained sentences is treated as an answer candidate. This means that a machine is only required to infer the positive or negative status of each sentence. This problem can be often solved by only looking at the questions; the ability of reading comprehension is not necessarily required. WIKIQA maintains 3,047 questions, of which about two thirds are NAQs, as they never have a positive sentence in the corresponding paragraphs. Moreover, 20.3% of the answers share no content words with the questions, contributing to the elevated difficulty level of the dataset. Unfortunately, this dataset is relatively small, and the amount of training data is inevitably limited.

SQuAD: SQuAD is an MRC dataset accommodating 107,785 question-and-answer (QA) pairs on 536 Wikipedia articles. A passage is associated with, at most, five QA pairs, and each question has the corresponding answer, forming a span in the associated passage. The evaluation metrics are exact match (EM) and F1. An EM score measures the percentage of predictions that match any one of the ground truth answers exactly. An F1 score is a metric that measures the average overlap between the prediction and the ground truth answer. Figure 1, adopted from (Rajpurkar et al., 2016), exemplifies three QA pairs taken from a paragraph in a Wikipedia article, whose topic is precipitation. Note that each of the three questions refers to the same passage, and the corresponding answer can be found as a span within the passage. As detailed in the next section, we create an MRC dataset with NAQs by modifying an existing dataset. If a good source dataset is properly chosen, this strategy could prevent the shortage of QAs and enjoy the advantages of the existing dataset. Among the many MRC datasets, we choose SQuAD as the source, because it is a rather large-scaled dataset of the answer-in-the-text style, which functions as a good starting point to pursue the study of genuine machine understanding of a language.

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

Figure 1: Example of QA pairs on a text passage in SQuAD.

2.2 Potential Problems with SQuAD

Jia and Liang (2017) make a point that the extent to which MRC systems truly understand language remains unclear. To reward systems that have real language understanding abilities, they proposed an adversarial evaluation method for SQuAD. Their method challenges the MRC system by altering the passages to include adversarial sentences that could lead to wrong answers. The results of their experiments proved that current MRC systems are easily affected by inserted adversarial sentences, concluding that the machines need to understand language more precisely. Their approach can be contrasted with ours, in that they modify text passages, whereas we propose to incorporate NAQs. However, both share the same objective of extending the conventional MRC framework.

Sugawara et al. (2017) evaluated a published MRC dataset for ascertaining its quality. They adopted two metrics: prerequisite skills and readability. Prerequisite skills include abilities, such as object tracking, coreference resolution, and logical reasoning. Readability was evaluated based on NLP metrics, such as average sentence length in words and adverb variation. Their evaluation analyses revealed that the prerequisite skills correlated more with dataset quality than readability. They concluded that SQuAD passages were not very readable, whereas the questions were relatively easy to answer. Although this insight does not have any direct connection with the present work, it should be noted, because it provided considerable analyses of SQuAD.

3 Dataset Creation

We propose an automatic method for creating a dataset that is useful in the development of an NAQ detection mechanism. The proposed method modifies an existing popular MRC dataset (i.e., SQuAD), which avoids the manual construction of a dataset from scratch. This section first describes the dataset creation method and then proposes a method for grading the difficulty level of an NAQ. Here the "difficulty level" signifies how difficult a question is for a machine to identify as an NAQ[1]. The description of the resulting dataset concludes this section.

[1] A set of Python scripts for creating an NAQ dataset from SQuAD is available at `https://github.com/nknsh0000/createNAQs`.

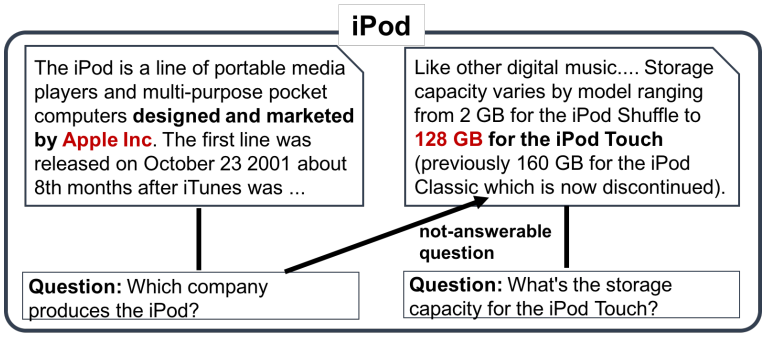


Figure 2: Creation of NAQ with our proposed creation strategy.

3.1 Devising NAQs

3.1.1 The Strategy

The priority in the creation of an applicable dataset is to incorporate hard-to-detect NAQs. It seems rather easy to devise an NAQ. Randomly selected or generated questions can cater to the most fundamental requirements. However, the questions gathered by this manner could be easily detected by a machine. Thus, they are an inadequate device for assessing the degree of machine language comprehension.

The similarity between a question and the corresponding text passage plays a key role in the creation of NAQs. More specifically, an NAQ is created by simply taking a question that was originally associated with another text passage that is highly similar to the text passage of the current target. This practical method can be applied to any MRC datasets, where the answer to a question is restricted to a span in the corresponding text passage.

3.1.2 Creation of NAQs using SQuAD

Adjacent text passages in SQuAD are taken from Wikipedia articles, meaning that these passages are very likely to share a topic and thus be similar. This pattern allows us to simply select an adjacent passage as the target text passage of an NAQ. Figure 2 illustrates the procedure for NAQ creation from two pairs of a text passage and a question. The question, "which company produces the iPod?," originally associated with the left passage, is taken as an NAQ of the right passage. Notice here that the entity, "iPod," resides in the passage, potentially confusing a machine. Questions that accidentally have the corresponding answers in a shifted passage are thus removed[2].

3.2 Grading NAQ Difficulties

Although the difficulty levels of created NAQs may vary, their proper annotations can be effectively utilized. We can extract a subset of the entire dataset by choosing the difficulty levels and by enabling more detailed evaluation of a machine's NAQ detection performance. Difficulty levels should be automatically determined by referring to mechanically-extractable features, rather than relying on human assessments.

3.2.1 Feature Detection

The selection of a feature to well-estimate the difficulty level of an NAQ is not trivial. To accomplish this sub-task, we first collect potentially useful features and train an answerable/not-answerable binary classifier. We then conduct ablation tests to find the most effective feature among the candidates.

Potentially useful features can be divided into two feature groups: Individual features and Similarity features.

Individual features: To represent a passage and a question individually, an averaged word embedding vector and its TF-IDF weighted version are respectively computed for each of the passages and

[2]We removed 1,825 questions.

each of the questions. We employed pre-trained 100-dimensional GloVe (Pennington et al., 2014) word embedding vectors[3].

Similarity features: A largely irrelevant NAQ that is easy-to-detect is expected to have lower similarity to the given passage. We thus compute five types of similarity features as listed below.

- (a) max_word_sim: the maximum similarity between a word in the question $q = (\boldsymbol{w_1^q}, \ldots, \boldsymbol{w_m^q}, \ldots, \boldsymbol{w_M^q})$ and a word in the passage $p = (\boldsymbol{w_1^p}, \ldots, \boldsymbol{w_n^p}, \ldots, \boldsymbol{w_N^p})$. Thus, this similarity feature is formulated as: $max\_word\_sim = \max_{m,n} \cos(\boldsymbol{v_m^q}, \boldsymbol{v_n^p})$;
- (b) ave_word_sim: the averaged similarity between a word in the query and its most similar word in the passage: $ave\_word\_sim = \frac{1}{M} \sum_{m=1}^{M} \max_n \cos(\boldsymbol{v_m^q}, \boldsymbol{v_n^p})$;
- (c) the BLEU score (Papineni et al., 2002);
- (d) t_{cos}: the cosine similarity between each of the TF-IDF bag-of-words vectors; and
- (e) gt_{cos}: the cosine similarity between the averaged TF-IDF weighted word embedding vectors.

In the calculation of all features, we treat all out-of-vocabulary words as "unknown", which means that these words are simply skipped.

Binary classification: We conducted answerable/not-answerable binary classification experiments by employing the above-mentioned feature groups. In the experiments, we compared four classification algorithms (i.e., random forest (RF), linear regression (LR), support vector machine (SVM), and AdaBoost). Each were trained with 45,000 NAQs and the same number of answerable questions, randomly chosen from the created dataset. A 10-fold cross validation was applied in each run. As Table 1 clearly shows, similarity features could be utilized as a relatively good indicator.

Classifier	RF	LR	SVM	AdaBoost
Individual Features	0.597	0.563	0.565	0.554
Similarity Features	0.847	**0.852**	0.851	**0.852**

Table 1: Accuracy results of the binary classification.

Ablation tests: Based on the experimental results, we employed similarity features with the LR classifier. We then conducted ablation tests, where each similarity feature was ablated under the same experimental settings. Table 2 summarizes the results. As indicated by the greatest degradation, the feature of average similarity between words in the question and the passage, ave_word_sim, contributed the most. Given these results, we simply exploited the ave_word_sim feature as the indicator of difficulty level.

Ablated feature	max_word_sim	ave_word_sim	BLEU	t_{cos}	gt_{cos}
Accuracy	0.851	**0.812**	0.830	0.852	0.851

Table 2: Accuracy results from the ablation tests (comparing accuracy: 0.852).

3.3 Created Dataset

We finally completed our dataset by integrating the created NAQs with the original answerable questions from SQuAD. Table3 measures the number of questions in the created dataset.

Furthermore, we assigned a difficulty level label to each NAQ in the dataset by using the similarity feature, ave_word_sim, with the following criteria. Table 4 classifies the distribution of assigned difficulty levels. Note that the NAQs creation and the difficulty level assignment were individually conducted for train set and dev set.

[3] `http://nlp.stanford.edu/data/glove.6B.zip`

	Train		Dev	
	#passages	#questions	#passages	#questions
Answerable questions	18,896	87,599	2,067	10,570
NAQs	17,071	75,155	2,065	9,762

Table 3: The number of questions in the created dataset.

- LEVEL1 (easy): $0.0 \leq ave\_word\_sim < 0.5$
- LEVEL2 (moderate): $0.5 \leq ave\_word\_sim < 0.7$
- LEVEL3 (difficult): $0.7 \leq ave\_word\_sim \leq 1.0$

	Ranges of ave_word_sim	Train	Dev
ALL	$0.0 \leq ave\_word\_sim \leq 1.0$	75,155	9,762
LEVEL1	$0.0 \leq ave\_word\_sim < 0.5$	9,686	823
LEVEL2	$0.5 \leq ave\_word\_sim < 0.7$	57,730	4,636
LEVEL3	$0.7 \leq ave\_word\_sim \leq 1.0$	7,739	4,303

Table 4: Distribution of NAQ difficulty levels.

For reference, Figure 3 and Figure 4 respectively exemplify a LEVEL3 (difficult) and a LEVEL1 (easy) questions.

Passage:
On September 6 2006 Sony announced that PAL region PlayStation 3 launch would be delayed until March 2007 because of a shortage of materials used in the Blu-ray drive. At the Tokyo Game Show on September 22 2006 Sony announced that it would include an **HDMI port** on the 20 GB system but a chrome trim flash card readers silver logo and Wi-Fi would not be included. Also the launch price of the Japanese 20 GB model was reduced by over 20% and the 60 GB model was announced for an open pricing scheme in Japan. During the event Sony showed 27 playable PS3 games running on final hardware.

Question:
How many **USB ports** did the original PS3 prototype have?

Figure 3: An example of difficult NAQ.

Passage:
PlayStation Home is a virtual 3D social networking service for the PlayStation Network. Home allows users to create a custom avatar which can be groomed realistically. Users can edit and decorate their personal apartments avatars or club houses with free premium or won content. Users can shop for new items or win prizes from PS3 games or Home activities. Users interact and connect with friends and customise content in a virtual world. Home also acts as a meeting place for users that want to play multiplayer games with others.

Question:
What section of What's New can't show links to websites?

Figure 4: An example of an easy NAQ.

4 NAQ Detection Experiments and the Results

We conducted two types of NAQ detection experiments: primary experiments for comparing model architectures, and additional experiments for confirming the validity of the difficulty levels.

4.1 Comparing Model Architectures

In Section 3.2, answerable/not-answerable binary classification experiments were described. They were conducted to explore efficient features for the difficulty-level grading. However, the model architecture for NAQ detection was not necessarily limited to the described one. Rather, we adopted an existing MRC answering model for the present purpose. Namely, we exploited BiDAF (Min et al., 2016), a popular neural network model for MRC/QA in the present NAQ detection experiments.

Figure 5 overviews two BiDAF-based architectures for NAQ detection: (a) binary classification model with answer span confidences, and (b) "null-answer" question detection model.

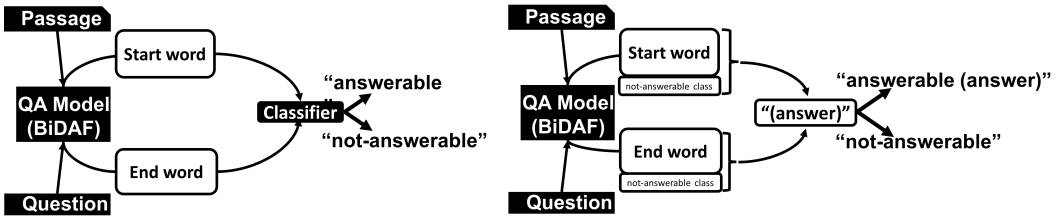


Figure 5: BiDAF-based model architectures for NAQ detection experiments: (left) binary classification model; (right) "null-answer" detection model.

(a) Binary classification model with answer span confidences: Like many existing MRC models, BiDAF estimates both start and end words of an answer candidate in the given text passage, allowing us to utilize two confidences associated with these two words as features. In this paper, we refer to the confidence of a start-word vector as $\boldsymbol{s}$ and that of an end word as $\boldsymbol{e}$. The numbers of dimensions of $\boldsymbol{s}$ and $\boldsymbol{e}$ were aligned with the number of the words in the longest text passage. This resulted in 673 dimensions. In this experiment, 75,155 NAQs and the same number of answerable questions were randomly chosen for training, and 9,762 NAQs and the same number of answerable questions were randomly chosen for testing.

(b) "null-answer" question detection model: The BiDAF model can be leveraged another way. This model explicitly adds a "null-answer" class by properly representing an empty span associated with a null-answer. Through a preliminary experiment, we expressed it by locating both the start and end positions at the very last word of the text passage. The same set of answerable questions and NAQs was also used in this experiment.

4.1.1 The Results

Table 5 summarizes the classification accuracy with the binary classification model, where four types of classifiers (RF, LR, SVM, and AdaBoost) were particularly compared. As seen in Table 5, the highest accuracy of 0.813 was achieved with the RF classifier that employs both features, (yp_{start} and yp_{end}). These results were worse than those of the classifiers with similarity features utilized in the difficulty level grading, presented in Section 3.2.

Table 6, on the other hand, presents the classification results of accuracy with the null-answer detection model, where the accuracy was as high as 0.895, which greatly outperformed other models, including the one described in Section 3.2. Moreover, the recall rate for recovering answerable questions was as high as 0.936, whereas that for recovering NAQ was as low as 0.854, indicating that the detection of NAQs is more difficult. If we only consider the results with answerable questions, the EM score was 0.607, which was moderately worse than 0.680, achieved with the model trained without NAQs. Thus, the percentage

Classifier		RF	LR	SVM	AdaBoost
Features	yp_{start}	0.807	0.774	0.760	0.808
	yp_{end}	0.785	0.756	0.751	0.792
	yp_{start}, yp_{end}	**0.813**	0.780	0.774	0.813

Table 5: Accuracy results with the binary classification model.

	All	Answerable questions	NAQs
Accuracy of NAQ detection	**0.895**	0.936	0.854
EM scores	-	0.607	-

Table 6: Accuracy results with the null-answer detection model.

of misclassifying answerable questions as not-answerable was only 6.36% (1-0.936), whereas that of misclassifying NAQs as answerable was 14.6% (1-0.854).

With the null-answer model, it is expected that correct answers will be recovered. The error rate of answer extraction for the correctly identified answerable questions was 0.33, which is comparable with 0.32, achieved with the model trained without NAQs. This small difference supports the assumption that the model would have misclassified many answerable questions as NAQs.

4.2 Validating the Difficulty Level Grading

We can extract a subset of the entire dataset by choosing the difficulty levels, which enables more detailed evaluation of the machine's NAQ detection performance. If the detection accuracies degrade with the increase of difficulty levels, then the assigned difficulty levels are valid. Moreover, if the NAQ detection accuracies are low enough, the created dataset is useful for the dev of the functionality to distinguish NAQs from answerable questions.

For each class of the difficulty level, we randomly extracted the same number of answerable questions from the SQuAD dataset as the NAQs from our dataset, and conducted the binary classification for the dev set. Again, we compared the binary classification model (with the RF classifier) with the null-answer detection model.

4.2.1 The Results

The results in Table 7 show that the accuracies degrade with the increase of the difficulty level, indicating that the difficulty grading is valid. Additionally, we confirmed that the null-answer model was superior to the present binary classification model. Moreover, the results for the most difficult level NAQs gave significantly lower accuracy figures, signaling that the models for detecting NAQs still have room for improvement. Nevertheless, we can construct a sub-dataset with designated difficulty levels from the entire dataset, which could be effectively exploited in the development of MRC models that deal with NAQs.

4.3 Error Analysis

Errors are divided into two cases, where the machine: (a) judged an NAQ as answerable (false answerable), or (b) judged an answerable question as an NAQ (false not-answerable).

Model	Binary classification	Null-answer detection
LEVEL1	0.860	0.953
LEVEL2	0.855	0.948
LEVEL3	0.748	0.734

Table 7: Accuracy results of NAQ detection experiments for each difficulty level.

4.3.1 False answerable

Figures 6, 7 and 8 respectively display a false answerable error case, where the machine wrongly predicted a question as answerable in each difficulty level.

In the Level-1 question exemplified in Fig. 6, the machine wrongly answered the shape of thylakoid rather than that of pyrenoids. This error could be attributed to the existence of the keyword "shape" appeared in the question. The passage actually had nothing to do with pyrenoids. In the example shown in Fig. 7, the passage contains the sentence, "In 1893, Tesla returned to his birthtown, Smijian." The machine's prediction would have been strongly affected by the existence of not only "Tesla" but "go": "go" in the question and "return" in the passage exhibit a level of semantic similarity. In this example, the machine should have recognize that "Smiljan" and "Karlovac" are totally different placements, whose similarity would not be taken into account. Finally, Fig. 8 displays a Level-3 question. Similar to the example given in Fig. 7, the machine was again not able to recognize the difference in nouns; in this case the focused nouns are compounds, each of them designates a specific event.

These examples suggest that the machine estimated the NAQs as answerable questions especially when some words overlapped between passage and question. In these cases, the machine tends to simply predict a span whose estimated semantic type is matched with the type of the interrogative.

4.3.2 False not-answerable

In false not-answerable cases, we observed differences in the mean length of questions and that of passage as counted in Table 8. This table suggests that the machine tended to recognize answerable questions as NAQs when the passage length is short or when the ground truth answer length is long.

Passage:
Chloroplast (Level 1)
In the helical thylakoid model, grana consist of a stack of flattened **circular** granal thylakoids that resemble pancakes. Each granum can contain anywhere from two to a hundred thylakoids, though grana with 10–20 thylakoids are most common. ...

Question:
What **shape** are pyrenoids?
Wrong Answer: circular

Figure 6: A Level-1 NAQ that the machine misjudged as an answerable question.

Passage:
Nikola_Tesla (Level 2)
In **1873**, **Tesla returned** to his birthtown, Smiljan. Shortly after he arrived, Tesla contracted cholera; he was bedridden for nine months and was near death multiple times. Tesla's father, in a moment of despair, promised to send him to the best engineering school if he recovered from the illness (his father had originally wanted him to enter the priesthood).

Question:
When did **Tesla go** to Karlovac?
Wrong Answer: 1873

Figure 7: A Level-2 NAQ that the machine misjudged as an answerable question.

5 Concluding Remarks

Machine understanding of language is difficult to define and accomplish. However, we must approach this issue by developing a computational mechanism, along with the relevant resources. As a step along this direction, in this paper, an automatic dataset creation method with which NAQs were incorporated was proposed. We created a dataset by altering an existing QA dataset, SQuAD.

Passage:
Super_Bowl_50 (Level 3)
QuickBooks sponsored a "Small Business Big Game" **contest**, in which Death Wish Coffee had a 30-second commercial aired free of charge courtesy of QuickBooks. Death Wish Coffee beat out nine other contenders from across the United States for the free advertisement.

Question:
What brand **sponsored** the "Crash the Super Bowl" **contest**?
Wrong Answer: QuickBooks

Figure 8: A Level-3 NAQ that the machine mistook as an answerable question.

	Answerable questions	
	Correctly predicted as answerable	Incorrectly predicted as NAQ
Passage	777.39	722.01
Answer	2.82	3.25
Answer/Passage Ratio [%]	0.42	0.53

Table 8: The lengths of passages and answers contained answerable questions.

One of the key requirements for such a dataset would be the acquisition of NAQs that are difficult to identify as not-answerable. Given this sub-goal, a method for assigning the difficulty levels to NAQs was also described. The results of NAQ detection experiments demonstrated that the assigned difficulty levels were valid, because the detection accuracies were degraded with the increase in difficulty level. This means that we acquired a way to examine and evaluate NAQ detection methods and models more precisely. Moreover, the lower accuracy results confirmed that there is still scope for improvement in the NAQ detection models. Therefore, the created dataset could be effectively utilized in developing the NAQ detection functionality.

For future work, we should develop a more difficult MRC dataset by further investigating into the notion of the difficulty in NAQ detection, which would inherently require us to incorporate various aspects of human language understanding. Along with this direction, we will devise a set of questions, for which confusing items reside in the target text passages.

References

Chen, D., Bolton, J., and Manning, C. D. (2016). A thorough examination of the CNN/Daily Mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2358–2367, Berlin, Germany.

Hill, F., Bordes, A., Chopra, S., and Weston, J. (2016). The goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of the 4th International Conference on Learning Representations*, San Juan, Puerto Rico.

Jia, R. and Liang, P. (2017). Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark.

Min, S. J., Aniruddha, K., Ali, F., and Hannaneh, H. (2016). Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

Mostafazadeh, N., Roth, M., Louis, A., Chambers, N., and Allen, J. (2017). Lsdsem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 46–51, Valencia, Spain. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: A method for automatic evaluation of

machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Stroudsburg, PA, USA.

Pennington, J., Socher, R., and Manning, D. C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, USA.

Richardson, M., Burges, J. C. C., and Renshaw, E. (2013). Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Emprical Methods in Natural Language Processing*, pages 193–203, USA.

Sugawara, S., Kido, Y., Yokono, H., and Aizawa, A. (2017). Evaluation metrics for machine reading comprehension: Prerequisite skills and readability. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 806–817, Vancouver, Canada.

Weston, J., Bordes, A., Chopra, S., and Mikolov, T. (2015). Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698.

Yang, Y., Yih, S. W., and Meek, C. (2015). Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal.

Style Obfuscation by Invariance

Chris Emmery[*]
Tilburg University
University of Antwerp
cmry@pm.me

Enrique Manjavacas[*]
University of Antwerp
enrique.manjavacas
@uantwerpen.be

Grzegorz Chrupała
Tilburg University
g.chrupala@uvt.nl

Abstract

The task of obfuscating writing style using sequence models has previously been investigated under the framework of obfuscation-by-transfer, where the input text is explicitly rewritten in another style. A side effect of this framework are the frequent major alterations to the semantic content of the input. In this work, we propose obfuscation-by-invariance, and investigate to what extent models trained to be explicitly style-invariant preserve semantics. We evaluate our architectures in parallel and non-parallel settings, and compare automatic and human evaluations on the obfuscated sentences. Our experiments show that the performance of a style classifier can be reduced to chance level, while the output is evaluated to be of equal quality to models applying style-transfer. Additionally, human evaluation indicates a trade-off between the level of obfuscation and the observed quality of the output in terms of meaning preservation and grammaticality.

1 Introduction

The fact that writing style uniquely characterizes a person, and can be leveraged for automatic author identification (Holmes, 1998; Stamatatos et al., 2000), has been well-studied in the field of (computational) stylometry (Neal et al., 2017). Similarly, work on author profiling (Koppel et al., 2002) has demonstrated that such stylometric features can be used to accurately infer an extensive set of personal information, such as age, gender, education, socio-economic status, and mental health issues (Eisenstein et al., 2011; Alowibdi et al., 2013; Volkova et al., 2014; Plank and Hovy, 2015; Volkova and Bachrach, 2016). Traditionally, these techniques relied on expensive human-labelled examples; however, more recent work has demonstrated near equal accuracy when only relying on self-reports as a distant supervision signal (Beller et al., 2014; Emmery et al., 2017; Yates et al., 2017). While these efforts have been greatly beneficial to various research fields such as computational sociolinguistics (Daelemans, 2013), they potentially expose users of such media to attacks where this information can be abused unbeknownst to them. This is particularly harmful to individuals in a vulnerable position regarding race, political affiliation, mental health, or any other personal information made explicitly unavailable.

Adversarial stylometry, or style obfuscation, is one of the proposed methods aimed at protecting users against such attacks. Its objective is to rewrite an input text such that a classifier (the adversary) trained on detecting a particular variable (such as a demographic attribute) is fooled — effectively protecting this variable. The main challenge is to preserve the original meaning of the input, whilst hiding the act of obfuscation (Potthast et al., 2016). Recent work on automatic obfuscation (Shetty et al., 2017; Karadzhov et al., 2017) shows promising results in minimizing performance of the adversary; however, these models (also noted by the authors) struggle with correctly maintaining the semantic content of the input. To illustrate, while rewriting *school* to *wedding*[1] effectively fools an age classifier into thinking the text is written by an adult rather than a teen, the original meaning is not preserved in the output.

[*]Equal contribution.

[1]Example taken from Shetty et al. (2017).

Proceedings of the 27th International Conference on Computational Linguistics, pages 984–996
Santa Fe, New Mexico, USA, August 20-26, 2018.

We propose that this observed shift in meaning is to some extent a by-product of the formulation of the obfuscation task. Content words that are strongly related to a particular attribute often play a significant role in the accuracy of a potential adversary. There is ample evidence for this phenomenon in age and gender classification work (Koppel et al., 2002; Rao et al., 2010; Burger et al., 2011; Sap et al., 2014, inter alia). Taking examples from Sap et al. (2014) specifically, features with strong coefficient weights for gender include e.g. *boxers, shaved, girlfriend, beard, fightin* for males, and *purse, blueberry, pedicure, hubby, earrings* for females. It is therefore not a surprising result that models explicitly tasked to *minimize* the performance of such classifiers perform what we will refer to as obfuscation-by-transfer. To illustrate, the adversary is easily fooled when a sentence looks strongly female even though it was written by a male. As such, the easiest route to obfuscation from this perspective is a form of style-transfer: swapping a few strongly target-associated content words for their contrastive variant (*wife* to *husband*, *school* to *wedding*). When such variants are also close in semantic spaces that sequence models make use of, any reconstruction metrics—such as BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), embedding distances, etc.—might become an inaccurate indication of the change in meaning.

Our contributions In this work we propose a different approach to automatic obfuscation that we hypothesize partly overcomes the limitations to preserving meaning of the input: obfuscation-by-invariance. Here, the objective shifts towards maximizing adversary's *uncertainty*, implying its accuracy on the protected variable should be as close to chance level as possible. Fixing the adversary's performance around chance involves making the input text devoid of stylistic features that strongly correlate with any of the protected variables, thus producing language that is neutral with respect to these style differences. We test our hypothesis in several experimental conditions.[2] The main component in our encoder-decoder architecture to achieve a style-invariant encoding of the input is a Gradient Reversal Layer (GRL) (Ganin and Lempitsky, 2015) inserted between the input sentence embedding and the style classifier. We investigate the effect of this module in isolation, as well as in a style-invariant soft transfer setting by using a conditioned decoder. First, to gauge if this architecture can successfully decode from the style-invariant encoding—and what the effect on adversary performance would be—we train sequence-to-sequence models on a parallel corpus of English Bible styles. Secondly, given that in a realistic obfuscation setting there is no access to such parallel sources, we drop the target pairs to create an autoencoder setting. In our experiments, we demonstrate a trade-off around chance-level performance: obfuscation-by-transfer in a parallel setting works well using a many-to-many translation model, but scores worse in the human evaluation than our style-invariant model. As such, we pose that there is potential in an style-invariant approach to obfuscation, and it deservers further investigation.

2 Related Work

2.1 Adversarial Stylometry

The idea that computational stylometry might be used to compromise anonymity was first explored by Rao et al. (2000). They saw potential in concealing style information using machine translation (MT), but noted that it was not powerful enough at the time. Kacmarcik and Gamon (2006) continued the proposed line of work by informing users regarding characteristic features and deeper linguistic cues in their writing style. Recent related studies can be found in (Caliskan-Islam et al., 2015; Le et al., 2015). Brennan et al. (2012) explicitly frame obfuscation as an adversarial task and use MT (round-trip translation), similar to (Caliskan and Greenstadt, 2012). Rule-based perturbations (Juola and Vescovi, 2011) and mixtures of both (Karadzhov et al., 2017) have also been applied for fully automatic obfuscation. Closest to our approach is recent work by Shetty et al. (2017), who pursue the task of learning obfuscation-by-transfer using a Generative Adversarial Network architecture. In their setup, a generator is trained to produce sentences that maximize the probability assigned by a discriminator that is, in turn, trained to distinguish real from generated sentences. While they incorporate different semantic losses, and demonstrate successful obfuscation on age and gender annotated micro-blog data and political

[2] All code and data to fully replicate the experiments is available at `github.com/cmry/style-obfuscation`.

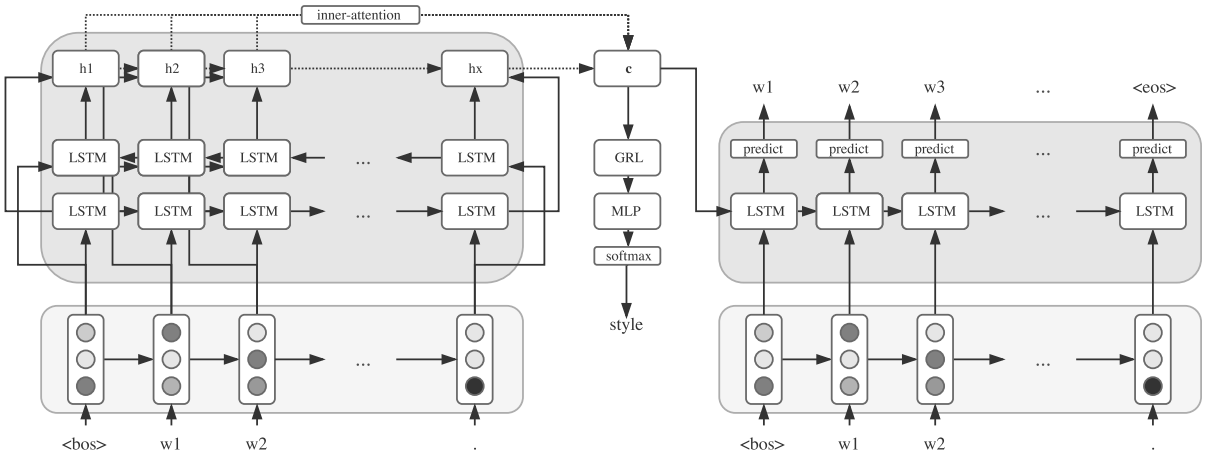


Figure 1: Model architecture with the Gradient Reversal Layer (GRL) module: Left the encoder part, middle the GRL taking the input sentence embedding (also called context vector) **c** as input, and right the decoder part of the architecture. Note that via the GRL working on the context vector, the encoder part of the model tries to minimize the style classification performance of the MLP, whilst still having to produce a useful representation for the decoder.

speeches, their output suffers from the lack of semantic preservation we described before. Finally, a style-transfer approach with potential application to obfuscation is work by Carlson et al. (2017), who investigate textual zero-shot style-transfer using a sequence-to-sequence MT model inspired by zero-shot translation (Johnson et al., 2016). Their work demonstrates successful translation between many different versions of the English Bible.

2.2 Gradient Reversal

The use of a Gradient Reversal Layer (GRL) for learning domain invariant feature representations was proposed by Ganin and Lempitsky (2015), who demonstrated its viability for learning lightning-condition invariance in computer vision. Since then, it has been applied to several language tasks: e.g. textual feature extraction (Pryzant et al., 2017), POS tagging (Kim et al., 2017; Gui et al., 2017), image captioning (Chen et al., 2017), and document classification (Liu et al., 2017; Xu and Yang, 2017). Most importantly, Xie et al. (2017) demonstrate the GRL module can be used to implement an adversarial setting, and to improve performance for a number of language tasks, including generation. These results bode well for its application to obfuscation-by-invariance.

3 Models

Our base architecture is a neural encoder-decoder (Sutskever et al., 2014) model similar to that of Wu et al. (2016), implemented in PyTorch (Paszke et al., 2017). Given an input sequence of one-hot encoded words, the encoder first embeds the words into dense vectors which are then processed by one or more bidirectional (Schuster and Paliwal, 1997) layers with LSTM cells (Hochreiter and Schmidhuber, 1997). The resulting sequence of processed vectors is then merged into a single dense representation using an inner-attention mechanism that will be described below. After encoding, a neural language-model is trained to decode the output sequence conditioned on the sentence embedding (a so called context vector) resulting from the encoder. Training is accomplished by minimizing the locally-normalized per-word cross-entropy of the target sequence.

In an autoencoder setting, the goal of the network is to simply reconstruct the original sentence based on the encoded context vector (Lauly et al., ; Li et al., 2015). This set-up can be combined with the GRL to encourage the encoder to produce attribute-invariant context vectors. The target is the input itself in the case of an autoencoder (AE) architecture, or a paired sentence in the case of a sequence-to-sequence (S2S) architecture. See Figure 1 for a visual representations of the base architecture.

3.1 Architecture Components

In addition to the architecture described above, we introduce a few extra components:

Gradient Reversal Layer (GRL) The GRL (Sutskever et al., 2014) is applied on top of the context vector. During the forward pass the GRL computes the identity function and feeds its input to a shallow Multi-Layer Perceptron (MLP) style classifier. However, during back-propagation the gradient of the style classifier is flipped in sign. The idea is that the encoder parameters are optimized to generate sentence embeddings that do not contain any stylistic information so that the style of the input cannot be recovered.

Conditional Decoder (C) Previous research on neural language modelling has demonstrated the effectiveness of conditioning a language model on sentential and contextual variables (such as tense, modality or voice). In our experiments, we evaluate a conditional autoencoder in which the decoder is conditioned on the input style label. We implement conditioning following the approach by Ficler et al. (2017), which simply concatenates the corresponding attribute embedding vector (in our case, the corresponding style embedding) to each of the word embeddings input to the decoder. Each style is therefore associated with an embedding vector $\mathbf{c}$, which is fed into the architecture at each step. In contrast to the simple autoencoder, the conditional autoencoder allows to choose a desired style at test time. We suspect that by encouraging the decoder to target a certain style, the output would be have more linguistic consistency without fully recovering the targeted style.

Token Transfer (TT) It can be argued that an MT system relying on a parallel corpus of styles (be it attributes or authors) would perform obfuscation-by-transfer. Moreover, it would likely preserve the original meaning as translation is largely a meaning-preserving operation (Ide et al., 2002; Dyvik, 2004). However, such parallel corpora are generally not available and have very high associated compilation costs, as it would require large amounts of identical information (ideally on sentence-level) to be written by e.g. teens and adults. Textual style transfer by MT is therefore not a plausible use-case for obfuscation. However, it does provide a good indication of the performance of an obfuscation model under the framework of obfuscation-by-transfer. For this, we apply a sequence-to-sequence translation model trained on style as discussed by (Carlson et al., 2017). Following the work of (Johnson et al., 2016), we use a target style token, allowing for a model trained on many-to-many translation that can be used to rewrite an input sentence in a different style simply by prepending a target token. This is the only configuration that uses an attention mechanism as it is not tested in combination with the GRL.

3.2 Architecture Details

All our models use 300-dimensional embeddings. Both the encoder and the decoder are implemented with a single layer of 1000 LSTM cells. The sequence of hidden states from the encoder are merged into a single representation using a feature-wise inner-attention mechanism. Let w_t and h_t denote respectively the input word embedding and hidden state of the LSTM for step t for a total of n total steps. The i^{th} feature of the sentence embedding c is computed by a weighted sum over the i^{th} feature of the hidden states:

$$c_i = \sum_{t=1}^{n} a_i^t h_i^t \tag{1}$$

where each weight a_i^t is computed by:

$$a_i^t = \frac{\exp([W^T z^t]_i)}{\sum_{s=1}^{n} \exp([W^T z^s]_i)} \tag{2}$$

In Equation 2, z_t is the concatenation of w_t and h_t, $W \in \mathbb{R}^{(D+H)\times H}$ is an additional projection matrix to be learned, and D and H denote the dimensionalities of the word embedding matrix and the LSTM layer respectively. In comparison to traditional merging models (such as max or mean pooling), the additional parameters help the model to learn what input words and what features are more important for

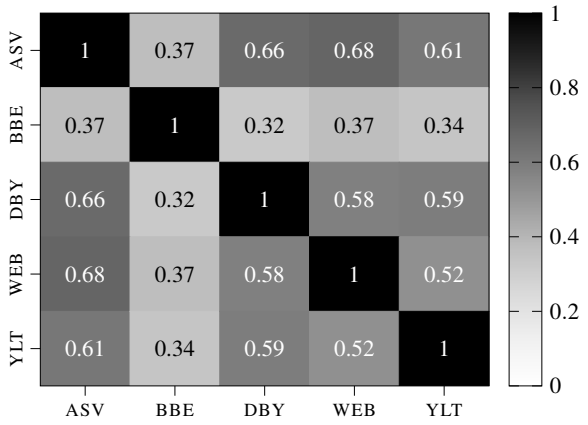


Figure 2: Jaccard index between vocabularies of the different Bible styles.

style	types	tokens	unique	punct
ASV	12,5K	3,7M	602	0.13
BBE	6,0K	3,9M	569	0.11
DBY	13,8K	3,7M	1729	0.14
WEB	12,8K	3,6M	1663	0.14
YLT	12,2K	3,9M	1682	0.17

Table 1: (Normalized) corpus descriptives, showing type and token frequencies, amount of unique types, and the punctuation ratio for each Bible style respectively.

the task. This is similar to conventional attention over hidden activation vectors with the main difference being that weighting is done feature-wise and all information flow from the encoder to the decoder is passed through the bottleneck of the single output encoding vector. Note that a traditional alignment-style attention mechanism is not compatible with the application of the GRL, since the latter must have scope over all information being passed from the encoder to the decoder.

The decoder is conditioned on the context vector by concatenating the latter to the input of the decoder at each step. This facilitates learning by increasing the gradient signal to the encoder. All model parameters are optimized with Adam (Kingma and Ba, 2014) in mini-batches of 50 examples and a learning rate of 0.001 which is decreased by 0.75 after each epoch. To avoid overfitting, dropout (Srivastava et al., 2014) is applied to the input of each LSTM layer with a dropping probability of 0.25 and we stop training when loss stops decreasing for 3 epochs. During test time, we approximate the global best output sequence applying beam search with a beam of size 5. GRL is implemented with a single-layer MLP with same dimensionality as the encoder.

4 Experimental Set-up

Our main goal is investigating the effectiveness of obfuscation-by-invariance, and more specifically to what extent style-invariant representations preserve sentential semantics of the input. To this end, we perform three experiments for different corpora and obfuscation settings, using the components described above. In each of these settings, there is an adversary trained to detect the variable to be obfuscated. We describe our experiments and evaluations below.

4.1 Data

We use a parallel corpus of five different versions of the English Bible (retrieved from GitHub[3], originally collected from `openbible.info`[4]). These versions are semantically consistent, but vary stylistically across different aspects; BBE is written in simplified English, YLT follows the syntactic structures of the original Greek and Hebrew, and other versions (DBY, ASV) correspond to older editions reflecting diachronic variations. The sentence-level verse coding (book + chapter + verse ID) is used for almost perfect pairing between the different versions (some missing pairs were removed), forming style quintuplets, which were tokenized using `spaCy`[5] (Honnibal and Montani, 2017). The style tuples are divided amongst train (80%), dev (10%), and test (10%) sets—ensuring all sentences in the development and test splits are unseen, regardless of the style combination that they occur in—and all style combinations

[3] `https://github.com/scrollmapper/bible_databases`
[4] `http://www.openbible.info/labs/cross-references/`
[5] `https://spacy.io/`

(excluding a same-style combination) are used to form 620,752 pairs. Further corpus descriptives for the different styles can be found in Figure 2 and Table 1.

4.2 Adversary

The adversary is a sentence-level classifier in the form of `fastText`[6] (Joulin et al., 2017); a simple linear model with one hidden embedding layer that learns sentence representations using bag of words or n-gram input, producing a probability distribution over the given styles using the softmax function. The classifier is trained on the source side of the training split, as these are the instances we intend to obfuscate. It is run for 20 epochs using 100 dimensional embeddings, uni and bi-grams, a learning rate of 0.01, and a bucket size of 1M. It achieves an accuracy of 86.6%, and chance level performance for the adversary is 20% given 5 classes with an even distribution.

4.3 Evaluation

To automatically evaluate the reconstruction and semantic preservation of our generated sentences, we use MT metrics, as well as distance in semantic embedding space. Obfuscation is measured by the difference of the adversary's accuracy compared to chance level.

MT Metrics (BLEU, METEOR) We calculate BLEU-4 and METEOR (Papineni et al., 2002; Banerjee and Lavie, 2005) using `nlg-eval`[7]. Given that this is not a standard MT task, we provide these scores between the generated sentence and the source sentence ($\leftarrow$), as well as the generated sentence and the target sentence ($\rightarrow$). For the sequence-to-sequence models, $\rightarrow$ is the primary indicator of successful obfuscation-by-transfer. However, $\leftarrow$ gives some indication how much the output is still related to the original. Introducing the GRL to this models should decrease scores for both $\leftarrow$ and $\rightarrow$. For the autoencoder, evaluation can only be conducted on $\leftarrow$, for which scores should generally strongly decrease when adding the GRL.

Word Mover's Distance (WMD) To assess the word embedding distance of the obfuscated sentence to the original, we take the Word Mover's Distance (WMD) (Kusner et al., 2015), based on the English `fastText` embeddings for Wikipedia (Bojanowski et al., 2016). WMD takes the distance between two sentences in a weighted point cloud of embedded words as the minimum cumulative distance that this cloud for sentence A needs to travel to align with that of sentence B, and is shown to capture some semantic relations.

Adversary Impact (Δ acc) We compare the accuracy of the adversary on the generated sentence to that of the original to assess obfuscation strength. However, as our goal is to keep the adversary's performance level close to chance, we define Δ accuracy $=$ accuracy $- p$ where p is majority baseline. Therefore, if Δ accuracy is negative, this means the adversary's performance has dropped below chance, and the task is closer to obfuscation-by-transfer rather than by-invariance. Subsequently, a positive score indicates the extent to which obfuscation fails to match both cases.

Gaussian Noise (σ) To enforce a significant change in the decoded output, one can simply add a Gaussian noise mask to the context vector during generation time. We generate this mask as a random vector from a Gaussian Distribution $N(0, \sigma)$, where $\sigma = \{0.01, 0.05, 0.10, 0.15, 0.20\}$ and add this to the values of the context vector. This noise can be utilized to increase obfuscation (due to more random decoding behaviour) at the cost of quality of the output.

4.4 Experiments

Using all components discussed above, we define three experimental settings to measure the effect of applying a GRL and conditional decoder to achieve obfuscation-by-invariance. (**1**) We train our architecture on the style pairs from the English Bible corpus in a many-to-many sequence-to-sequence setting. By introducing a GRL here, words that are highly indicative of the target style are not captured by the

[6] `https://github.com/facebookresearch/fastText`
[7] `https://github.com/Maluuba/nlg-eval`

model	C	GRL	TT	PPL	BLEU ←	MTR ←	BLEU →	MTR →	WMD	Δ ACC
S2S				9.08	22.0	25.3	17.8	23.6	1.50	1.6
S2S		x		9.27	21.8	25.2	16.9	22.9	1.50	6.9
S2S			x	7.38	34.9	30.5	39.2	29.9	1.24	-12.0
AE				1.51	79.5	52.9	-	-	0.25	64.4
AE		x		1.99	60.0	41.2	-	-	0.65	50.8
AE	x	x		1.87	51.9	38.1	-	-	0.79	18.3
source					100.0	100.0	36.0	34.5	0.00	66.6

Table 2: Results for the Bible experiments. The first column (model) indicates the setting of our base architecture: either sequence-to-sequence (S2S), or autoencoder (AE). The second column specifies which modules were incorporated: C for the conditioned decoder, GRL for the Gradient Reversal Layer, and `tt` for the prepended style token. The results show perplexity on the dev set (PPL), BLEU-4 and METEOR between source (←) / target (→) and the obfuscated sentence, the Word Mover's Distance score between source and the obfuscated sentence (WMD) and the extent to which the obfuscated sentence pushes the adversary to chance level performance (Δ ACC). In the last row, we note a 'source' baseline, that is achieved by simply copying the source. This shows how overall how much source and target overlap, and the actual above-chance performance of the adversary.

encoder. To achieve an effective many-to-many MT system (and thus style-transfer) setting we prepend the <2{stylename}> token. (**2**) We train an autoencoder on disconnected pairs. Here we introduce both the GRL, plus the conditioned decoder. We hypothesize that the conditioned decoder allows for soft style-transfer from a neutral encoding, implying it would preserve semantic structure better than the MT model. (**3**) We use Gaussian noise to make the sequence-to-sequence and autoencoder models equivalent in obfuscation performance to allow for direct comparison.

5 Results

5.1 Experimental Results

All results and automatic evaluations are shown in Table 2. As we hypothesized, using style-transfer for obfuscation works well overall, performing either at, or below chance level. Looking at the target BLEU and METEOR, the sequence-to-sequence model without the target token generates sentences that are closer to source than they are to the target; and achieves low scores overall, with the sentences being quite far off based on WMD. However, note that this is many-to-many translation without any signal regarding the target, given languages with largely the same vocabulary. As such, TT is a more realistic reflection of style-transfer success. In terms of translation quality it barely improves over the original baseline, but it does successfully perform obfuscation-by-transfer, as indicated by the 12% accuracy below chance. Assessing the performance of the GRL in this setting, it does not seem improve translation, nor obfuscation — which was largely in line with our expectations.

The autoencoder setting provides a clearer look into the performance of the GRL, in particular in terms of obfuscation-by-invariance. The plain autoencoder to some extent successfully reproduces the target; looking at BLEU, the adversary performance, and WMD, it is still closely related to the input. Introducing the GRL does impact the relation to the source sentence, but gains little in comparison on obfuscation performance. However, when the conditioned decoder is added to the architecture, obfuscation performance is visibly impacted more than the decrease in BLEU and METEOR. Lastly, the effect of adding a Gaussian noise mask on the decoder can be found in Table 3. Around 0.15, the metrics seem to be largely comparable in terms of BLEU, METEOR and Δ accuracy. We further investigate the three most suitable models (S2S + TT, AE + GRL + C, and AE + GRL + C + $\sigma = 0.15$) in a human evaluation.

	AE			AE + GRL			AE + C + GRL		
σ	BLEU ←	MTR ←	Δ ACC	BLEU ←	MTR ←	Δ ACC	BLEU ←	MTR ←	Δ ACC
-	79.5	52.9	64.4	60.0	41.2	50.8	51.9	38.1	18.3
0.01	78.7	52.2	64.1	59.7	41.4	50.5	52.0	38.5	18.5
0.05	52.9	38.9	57.2	54.8	39.1	47.6	49.1	37.2	16.5
0.10	14.7	21.6	36.1	32.8	32.7	36.9	40.7	33.6	11.6
0.15	4.2	14.7	23.3	25.3	26.2	24.3	30.0	29.1	4.6
0.20	1.5	11.3	16.4	15.3	21.5	15.5	21.1	24.9	0.0

Table 3: Effect of adding Gaussian Noise (σ) to the autoencoder.

	SEMANTICS	GRAMMATICALITY	CHANGES
1	Semantics are broken; sentence does not mean the same.	Part(s) of, or the complete sentence is garbled.	Change in special characters or flipping a single word.
2	Slight semantic change, but not intrusive.	Slight word order change that is ungrammatical.	Multiple words were changed, but they align with the original.
3	Semantics are intact, changes do not alter the meaning.	Grammaticality has not been affected.	New parts have been introduced / rewritten in the sentence.

Table 4: Explanations per rating for the three dimensions in the human evaluation study.

	SEMANTICS	GRAMMATICALITY	CHANGES
S2S + TT	1.88	2.21	2.43
AE + GRL + C	2.12	2.32	1.99
AE + GRL + C + $\sigma = 0.15$	1.35	1.66	2.65

Table 5: Average scores per model for the three dimensions in the human evaluation.

5.2 Human Evaluation

For the human evaluation, 20 pairs (original, obfuscated) were sampled from the output of the three models of interest (making 60 pairs in total). Each pair was rated by four participants (all with a linguistics background), on three dimensions using a three point scale. These dimensions included semantic consistency between the original and the obfuscated sentence, the syntactical coherence of the latter, and the amount of changes in the output. The participants were made aware of which sentence of the pairs was the original, and were explicitly asked to rate the dimensions with the original as reference. The participants were not aware that there were multiple models, and the pairs were shuffled so that comparing between pairs with the same original was impossible. To simplify the comparison to the original, we only sampled from BBE (Basic English Bible). See Table 5 for a sample of the instructions given to the participants, and Table 4 for the results.

According to the evaluation results, AE + GRL + C has the overall preference of the raters in all three dimensions. Specifically, given that we are interested in semantic preservation, this model is evaluated better than a style-transfer model that has some access to semantic relations between source and target on the SEMANTICS dimension. Note that based on the CHANGES dimension, the AE + GRL + C is the most conservative, which is in line with the BLEU and METEOR scores in Table 2, and likely propagates into the GRAMMATICALITY dimension.

5.3 Qualitative Analysis

In this section we perform a comparison of the behaviour, and the text generated by the three models that were evaluated by our raters in the previous section. Accordingly, we will identify the strengths and weaknesses of our experimental approach, and propose possible lines of future work.

ORIGINAL	For the strong town is without men, an unpeopled living - place; and she has become a waste land: there the young ox will take his rest, and its branches will be food for him.
S2S + TT	For the strong city is powerless, an astonishment living and she is become a corrupt land: a young ox shall rest, and its branches shall be for him.'
AE + GRL + C	For the city is without living men, there is without a waste land; and she makes an dry land: an ox shall take his work, and their branches shall be food for him.
AE + GRL + C + $\sigma = 0.15$	A man dwelleth without an dry land; and, wandering she - place; men shall there become a waste a land: an ox - goat shall take his horses, and his branches shall be prepare for him .

Table 6: Example 1 — Isaiah 27:10

ORIGINAL	Then they made the beryl stones, fixed in twisted frames of gold and cut like the cutting of a stamp, with the names of the children of Israel.'
S2S + TT	Then they made the onyx stones as a hundred stones, burning in engraved stones of gold, and cut as the marks of a seal, with the names of the children of Israel.
AE + GRL + C	Then they wrote the stones, whereupon they bound in the skillfully woven red frames of gold and made like the cutting of a stamp, with the names of the children of Israel.'
AE + GRL + C + $\sigma = 0.15$	Then they presented the pillars that belonged in Henadad. The bottom of fine gold and made like the jewels of a stamp of them, at the dial of the children of Israel.'

Table 7: Example 2 — Exodus 39:6

One of the issues we identified with obfuscation-by-transfer was that of small, localized changes in the input, specifically focussing on words that are most relevant to the adversary. When looking at longer sentences such as Table 6, some (semi-)correct variants can be found in e.g. town → city, waste land → dry land, and in Table 7, beryl → onyx, stamp → seal, but incorrect ones also remain: rest → work. Overall, the longer the sequence, the more variation can be observed.

A more interesting observation is that some parts of sentences are added to by the models, e.g. *there is without a waste land ; and she makes an dry land* — while incorrectly inserting '*without*', the rest can be considered is a correct expansion. The same holds for Table 7, where *fixed in twisted frames of gold* is expanded to *whereupon they bound in the skillfully woven red frames of gold* , partly erroneously, similar to *living men* in Table 6. Contrastively, the autoencoder + GRL in particular also seems to favour somewhat compressed phrases, removing adjectives such as *young* in *young ox*, *strong* in *strong city*, *beryl* in *beryl stones* and not incorporating *an unpeopled living* altogether.

When directly comparing the sequence-to-sequence and autoencoder + GRL examples, it can be inferred that the transfer approach seems (at least in these examples) quite conservative, sticking to an almost exact alignment, and only making small changes. This however also causes the autoencoder to replace words with unrelated variants or insert not directly related ones; the same behaviour can also be observed in the sequence-to-sequence model, however.

The output of the autoencoder shows some evidence that minor *rewrites* of the sentence are employed, which could potentially be an interesting path to further pursue. Including different variables in the conditioned decoder such as sentence length, also demonstrated by Fickler et al. (2017), would make experiments in this direction feasible. Rewrites are not restricted to the autoencoder only, as is demonstrated in Figure 3. We previously noted in Section 5.2 that the AE + GRL + C architecture seems more conservative in changing the input, which can also clearly be induced from the generally equal or higher amount of newly introduced tokens by the S2 + TT model, and its many more outliers (some even ex-

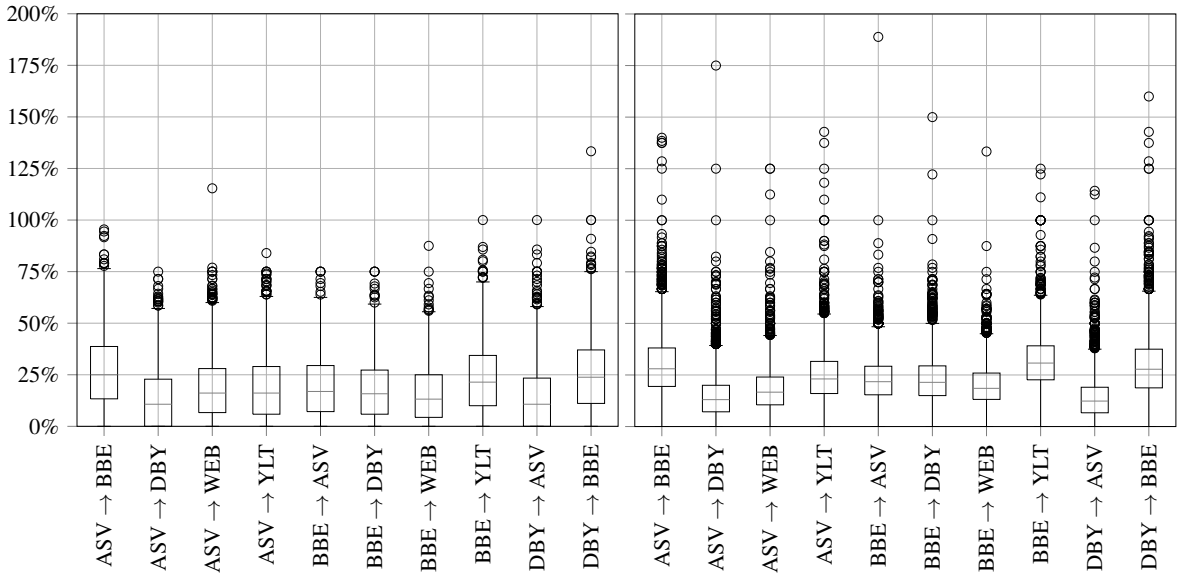


Figure 3: Distributions of the percentage of newly introduced tokens with respect to the original amount of tokens — shown for both the AE + GRL + C obfuscation-by-invariance architecture (left), and the S2S + TT obfuscation-by-transfer model (right). Three out of five source styles are shown.

ceeding the original amount of tokens). This plot also reveals a tendency to add more tokens for BBE and YLT, which is in line with their larger amount of tokens shown in Table 1. The extreme outliers are largely attributable to artefacts of recurrence (*and Benaiah , the son of Zadok , and Eber , and Eliphelet*).

Finally, it must be noted that evidence of style-neutral rewrites is difficult to find in generated output concerning the Bible; not only due to possible archaic constructions (which we attempted to minimize by using BBE), but more so due to the fact that it requires a level of expertise to recognize style shifts. However, most importantly, applying the autoencoder to data that is non-parallel has some viable ground given the current results, and can therefore be extended to other domains in future work.

6 Conclusion

We presented an alternative framing of the task of automatic style obfuscation—obfuscation-by-invariance—and tested several components in a neural encoder-decoder architecture that were hypothesized to achieve style-invariant rewrites of the input text. We tested the effect of a Gradient Reversal Layer and a conditioned decoder for obfuscation in parallel and non-parallel settings. Although strong evidence for style-neutral text was difficult to find for the Bible corpus, we demonstrated through human evaluation that our autoencoder architecture trained on non-parallel data obtained a better evaluation than a model trained on parallel data with partial access to semantic relations between source and target. In our qualitative analysis we found evidence for semantically correct local changes of the input, as well as partial rewrites that fit the context of the verses. These results bode well for extending this architecture to other non-parallel corpora to test obfuscation in a practical use-case, e.g. author attributes such as age and gender.

References

Jalal S Alowibdi, Ugo A Buy, and Philip Yu. 2013. Empirical evaluation of profile characteristics for gender classification on twitter. In *Machine Learning and Applications (ICMLA), 2013 12th International Conference on*, volume 1, pages 365–369. IEEE.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Charley Beller, Rebecca Knowles, Craig Harman, Shane Bergsma, Margaret Mitchell, and Benjamin Van Durme. 2014. Ima belieber: Social roles via self-identification and conceptual attributes. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 181–186.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Michael Brennan, Sadia Afroz, and Rachel Greenstadt. 2012. Adversarial stylometry: Circumventing authorship recognition to preserve privacy and anonymity. *ACM Transactions on Information and System Security (TISSEC)*, 15(3):12.

John D Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Discriminating gender on twitter. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1301–1309. Association for Computational Linguistics.

Aylin Caliskan and Rachel Greenstadt. 2012. Translate once, translate twice, translate thrice and attribute: Identifying authors and machine translation tools in translated text. In *Semantic Computing (ICSC), 2012 IEEE Sixth International Conference on*, pages 121–125. IEEE.

Aylin Caliskan-Islam, Fabian Yamaguchi, Edwin Dauber, Richard Harang, Konrad Rieck, Rachel Greenstadt, and Arvind Narayanan. 2015. When coding style survives compilation: De-anonymizing programmers from executable binaries. *arXiv preprint arXiv:1512.08546*.

Keith Carlson, Allen Riddell, and Daniel Rockmore. 2017. Zero-shot style transfer in text using recurrent neural networks. *arXiv preprint arXiv:1711.04731*.

Tseng-Hung Chen, Yuan-Hong Liao, Ching-Yao Chuang, Wan-Ting Hsu, Jianlong Fu, and Min Sun. 2017. Show, adapt and tell: Adversarial training of cross-domain image captioner. In *The IEEE International Conference on Computer Vision (ICCV)*, volume 2.

Walter Daelemans. 2013. Explanation in computational stylometry. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 451–462. Springer.

Helge Dyvik. 2004. Translations as semantic mirrors: from parallel corpus to wordnet. *Language and computers*, 49:311–326.

Jacob Eisenstein, Noah A Smith, and Eric P Xing. 2011. Discovering sociolinguistic associations with structured sparsity. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1365–1374. Association for Computational Linguistics.

Chris Emmery, Grzegorz Chrupała, and Walter Daelemans. 2017. Simple queries as distant labels for predicting gender on twitter. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 50–55.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. *arXiv preprint arXiv:1707.02633*.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189.

Tao Gui, Qi Zhang, Haoran Huang, Minlong Peng, and Xuanjing Huang. 2017. Part-of-speech tagging for twitter with adversarial neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2411–2420.

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

David I Holmes. 1998. The evolution of stylometry in humanities scholarship. *Literary and linguistic computing*, 13(3):111–117.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Nancy Ide, Tomaz Erjavec, and Dan Tufis. 2002. Sense discrimination with parallel corpora. In *Proceedings of the ACL-02 workshop on Word sense disambiguation: recent successes and future directions-Volume 8*, pages 61–66. Association for Computational Linguistics.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics, April.

Patrick Juola and Darren Vescovi. 2011. Analyzing stylometric approaches to author obfuscation. In *IFIP International Conference on Digital Forensics*, pages 115–125. Springer.

Gary Kacmarcik and Michael Gamon. 2006. Obfuscating document stylometry to preserve author anonymity. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 444–451. Association for Computational Linguistics.

Georgi Karadzhov, Tsvetomila Mihaylova, Yasen Kiprov, Georgi Georgiev, Ivan Koychev, and Preslav Nakov. 2017. The case for being average: A mediocrity approach to style masking and author obfuscation. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 173–185. Springer.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for pos tagging without cross-lingual resources. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2832–2838.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Moshe Koppel, Shlomo Argamon, and Anat Rachel Shimoni. 2002. Automatically categorizing written texts by author gender. *Literary and linguistic computing*, 17(4):401–412.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966.

Stanislas Lauly, Hugo Larochelle, Mitesh M Khapra, Balaraman Ravindran, Vikas Raykar, Amrita Saha, et al. An autoencoder approach to learning bilingual word representations.

Hoi Le, Reihaneh Safavi-Naini, and Asadullah Galib. 2015. Secure obfuscation of authoring style. In *IFIP International Conference on Information Security Theory and Practice*, pages 88–103. Springer.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742*.

Tempestt Neal, Kalaivani Sundararajan, Aneez Fatima, Yiming Yan, Yingfei Xiang, and Damon Woodard. 2017. Surveying stylometry techniques and applications. *ACM Computing Surveys (CSUR)*, 50(6):86.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Barbara Plank and Dirk Hovy. 2015. Personality traits on twitterorhow to get 1,500 personality tests in a week. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 92–98.

Martin Potthast, Matthias Hagen, and Benno Stein. 2016. Author obfuscation: Attacking the state of the art in authorship verification. In *CLEF (Working Notes)*, pages 716–749.

Reid Pryzant, Young-joo Chung, and Dan Jurafsky. 2017. Predicting sales from the language of product descriptions.

Josyula R Rao, Pankaj Rohatgi, et al. 2000. Can pseudonymity really guarantee privacy? In *USENIX Security Symposium*, pages 85–96.

Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. 2010. Classifying latent user attributes in twitter. In *Proceedings of the 2nd international workshop on Search and mining user-generated contents*, pages 37–44. ACM.

Maarten Sap, Gregory Park, Johannes Eichstaedt, Margaret Kern, David Stillwell, Michal Kosinski, Lyle Ungar, and Hansen Andrew Schwartz. 2014. Developing age and gender predictive lexica over social media. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1146–1151.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Rakshith Shetty, Bernt Schiele, and Mario Fritz. 2017. A^4NT: Author attribute anonymity by adversarial training of neural machine translation. *arXiv preprint arXiv:1711.01921*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Efstathios Stamatatos, Nikos Fakotakis, and George Kokkinakis. 2000. Automatic text categorization in terms of genre and author. *Computational linguistics*, 26(4):471–495.

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

Svitlana Volkova and Yoram Bachrach. 2016. Inferring perceived demographics from user emotional tone and user-environment emotional contrast. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL*.

Svitlana Volkova, Glen Coppersmith, and Benjamin Van Durme. 2014. Inferring user political preferences from streaming communications. In *ACL (1)*, pages 186–196.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Qizhe Xie, Zihang Dai, Yulun Du, Eduard Hovy, and Graham Neubig. 2017. Controllable invariance through adversarial feature learning. In *Advances in Neural Information Processing Systems*, pages 585–596.

Ruochen Xu and Yiming Yang. 2017. Cross-lingual distillation for text classification. *arXiv preprint arXiv:1705.02073*.

Andrew Yates, Arman Cohan, and Nazli Goharian. 2017. Depression and self-harm risk assessment in online forums. *arXiv preprint arXiv:1709.01848*.

Encoding Sentiment Information into Word Vectors for Sentiment Analysis

Zhe Ye[♠] **Fang Li**[♠] **Timothy Baldwin**[♡]

♠ Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai, 200240, China

♡ School of Computing and Information Systems,
The University of Melbourne, Victoria, 3010, Australia

{yezhejack, fli}@sjtu.edu.cn, tb@ldwin.net

Abstract

General-purpose pre-trained word embeddings have become a mainstay of natural language processing, and more recently, methods have been proposed to encode external knowledge into word embeddings to benefit specific downstream tasks. The goal of this paper is to encode sentiment knowledge into pre-trained word vectors to improve the performance of sentiment analysis. Our proposed method is based on a convolutional neural network and an external sentiment lexicon. Experiments on four popular sentiment analysis datasets show that this method improves the accuracy of sentiment analysis compared to a number of benchmark methods.

1 Introduction

Sentiment analysis plays an important role in many real-world applications. The objective of sentiment classification is to classify a sentence, message or document according to sentiment, often in the form of ordinal regression (e.g. positive vs. neutral vs. negative). In recent years, deep neural networks, such as convolutional neural networks ("CNNs"), have been widely used for sentiment classification. A simple CNN trained over pre-trained word vectors has been shown to achieve highly competitive results (Kim, 2014). Learning task-specific vectors through fine-tuning may offer further gains in performance, and this is the primary focus of this paper.

Separately, there has been recent work on methods for learning word embeddings based not just on textual contexts, but also external knowledge bases (Wieting et al., 2015; Yu and Dredze, 2014; Xu et al., 2014; Bian et al., 2014; Faruqui et al., 2015; Mrkšić et al., 2016; Mrkšić et al., 2017; Vulić et al., 2017). This has also been applied to sentiment classification (Rouvier and Favre, 2016; Yu et al., 2017), with empirical results indicating that explicitly embedding sentiment resources can improve the performance of sentiment analysis.

Existing methods for encoding external knowledge into word vectors are generally trained independently of the downstream task. In order to leverage sentiment lexicons for sentiment analysis, we propose a novel method to combine a feedforward neural network (denoted "SentiNet") with a CNN classifier to encode sentiment knowledge into word vectors during training. The method tunes word vectors through the CNN and SentiNet, based on independent information from supervised training data and sentiment lexicons. Our hypothesis is that joint training of sentiment-targeted word embeddings should improve the overall accuracy of the resulting sentiment analyzer. We conduct several experiments to verify this hypothesis, and compare our method with competitor methods that use antonymy/synonymy lexicons and paraphrase databases.

The major contributions of this paper are as follows: (1) the sentiment lexicon is encoded into word vectors by a feedforward neural network instead of an objective function based on a fixed metric such as cosine similarity or Euclidean distance, and in doing so are able to dynamically learn how to encode the lexicon; (2) word vectors are fine-tuned based on supervised training data and the sentiment lexicon during the training of the CNN sentiment classifier; and (3) we achieve state-of-the-art accuracy over a range of benchmark sentiment analysis datasets.

Proceedings of the 27th International Conference on Computational Linguistics, pages 997–1007
Santa Fe, New Mexico, USA, August 20-26, 2018.

2 Related Work

2.1 Encoding External Knowledge into Word Vectors

Existing approaches to leveraging external knowledge for word embedding learning for natural language processing fall into two categories: (1) encoding external knowledge during the word vector learning stage; and (2) encoding external knowledge into pre-trained word vectors. Both styles of approach make use of similar linguistic resources such as WordNet (Miller, 1995), FrameNet (Baker et al., 1998), the Paraphrase Database ("PPDB": Ganitkevitch et al. (2013)), or BabelNet (Navigli and Ponzetto, 2012).

Methods in the first category usually change the objective function of the language model or add regularization terms into the original objective function. Yu and Dredze (2014) combine CBOW (Mikolov et al., 2013) with word relations extracted from WordNet and PPDB. Xu et al. (2014) regard relational knowledge and categorical knowledge as learning regularizers, and combine them with the skip-gram objective function. Bian et al. (2014) also combine the objective function of CBOW with external syntactic and semantic knowledge to improve word vectors for extrinsic tasks. These methods all need large unlabeled corpora to learn word vectors from scratch.

In contrast, methods in the second category are lightweight because they adapt pre-trained word vectors via post-processing. This means these methods are compatible with different kinds of word vectors. Wieting et al. (2015) learn PARAGRAM word vectors by fine-tuning over paraphrase data from PPDB. The resultant embeddings outperform the baseline skip-gram embeddings over an extrinsic sentiment analysis task for low-dimensionality word embeddings. Faruqui et al. (2015) use synonym relations extracted from WordNet and other resources to construct an undirected graph. They then retrofit the undirected graph to pre-trained word vectors to obtain new word vectors, under the constraint that the resulting vectors should be close to the vectors of their neighbours in the semantic graph. Antonyms are generally close in vector space, presenting a problem when learning general-purpose word vectors (as in most scenarios, it is undesirable for antonyms to be closely related) (Mnih and Hinton, 2008; Collobert et al., 2011; Mikolov et al., 2013; Levy and Goldberg, 2014; Pennington et al., 2014). In order to solve this problem, antonym lexicons have been used to fine-tune pre-trained word vectors. Mrkšić et al. (2016) present a method called counter-fitting to inject antonymy and synonymy constraints into word vectors trained with GloVe (Pennington et al., 2014) and PARAGRAM (Wieting et al., 2015). The adapted word vectors trained with PARAGRAM achieve the second-highest SimLex-999 (Hill et al., 2015) score. Mrkšić et al. (2017) extend this previous work using negative sampling, to force synonym pairs to be closer to each other than to their negative examples, and forcing antonyms pairs to be further away from each other than from their negative examples.

Encoding external knowledge into word vectors has shown to be effective for improving pre-trained word vectors for intrinsic evaluation such as WordSim-353 (Finkelstein et al., 2002) and SimLex-999. It has also shown to be effective for improving extrinsic tasks such as dialogue state tracking (Mrkšić et al., 2016; Mrkšić et al., 2017; Vulić et al., 2017), sentiment analysis (Faruqui et al., 2015; Wieting et al., 2015; Yu et al., 2017), document classification (Kiela et al., 2015), and word sense disambiguation (Rothe and Schütze, 2015).

The above two kinds of methods both encode external knowledge into word vector space before applying word vectors in downstream tasks. Our method encodes a sentiment lexicon into word vectors when fine-tuning the word vectors in a downstream task.

2.2 Adapting Word Vectors for Sentiment Analysis

There are many methods of adapting word vectors for sentiment analysis. Maas et al. (2011) combine two components — a probabilistic document model and a sentiment component — to jointly learn word vectors. The probabilistic document model does not require labelled data. The sentiment component uses document-level sentiment annotations to constrain words expressing similar sentiment to have similar representations. Tang et al. (2014) changed the objective function of the C&W (Collobert et al., 2011) model to produce sentiment-specific word vectors for Twitter sentiment analysis, by leveraging large volumes of distant-supervised tweets. In order to capture morphological and shape information from words, dos Santos and Gatti (2014) concatenate character-level embeddings and word-level embeddings to form

a combined word representation for sentiment analysis. Severyn and Moschitti (2015) refine pre-trained word vectors through a CNN based on distant-supervised data. Zhou et al. (2016) introduced three kinds of word vectors as features into their sentiment classifier. One is general purpose, and the the other two are trained by leveraging sentiment information. However, their feature selection experiments indicate that the task-specific trained word vectors do not improve the performance of their system substantially. Ren et al. (2016) use a recursive autoencoder to learn topic-enhanced word vectors, based on the assumption that the same word can vary in sentiment according to topic. Rouvier and Favre (2016) proposed three kinds of word embeddings — lexical embeddings, part-of-speech embeddings, and sentiment embeddings — in order to train three CNN-based sentiment classifiers. The three classifiers are combined into a fusion model to make the final prediction.

Other methods regard sentiment information as a kind of external knowledge. They encode sentiment information into word vectors by using customized objective functions or introducing regularization terms into objective function of the language model. Yu et al. (2017) utilize a sentiment lexicon to re-rank the nearest neighbors in order to capture sentiment information. The refinement model is based on an objective function which calculates the distance among vectors, in order to adapt word vectors for sentiment analysis. Tang et al. (2016) build on their earlier method (Tang et al., 2014) for Twitter sentiment classification, by leveraging a sentiment lexicon instead of large volumes of distant-supervised Twitter data. Our method also leverages a sentiment lexicon to encode sentiment information into word vectors. The difference between our method and the methods proposed by Yu et al. (2017) and Tang et al. (2014) is that our method applies to word vectors directly via the word embedding layer of the neural network in the context of training a sentiment analyzer, rather than over pre-trained word vectors without explicit task-based training. Our method uses a feedforward neural network to encode sentiment information, as distinct from prior work, which has used cosine similarity or Euclidean distance in the objective function to model word embedding (dis)similarity.

3 Methods

Sentiment lexicons are considered to be a critical component of sentiment analysis. Such external knowledge resources — such as SentiWordNet (Baccianella et al., 2010) and the extended version of Affective Norms of English Words (E-ANEW: Warriner et al. (2013)) — are often used to provide more accurate information about the polarity of a word. Encoding sentiment knowledge into word vectors has been proven to be an effective way to enhance the performance of sentiment analysis.

3.1 Encoding Method

A high-level overview of methods for encoding external knowledge into word vectors for CNN classifiers is presented in Figure 1. Word vectors are used to initialize word embeddings in a CNN classifier. The parameters of the CNN and components for fine-tuning the word embeddings are learned based on supervised training data. The differences in approaches to encoding external knowledge into word vectors are indicated with dotted rectangles in Figure 1. There are three rectangles, representing the two categories of existing approaches described in Section 2.1, and our proposed method: (a) the first class of approach, where word vectors are trained based on external knowledge and unsupervised corpora from scratch; (b) the second class of approach, where pre-trained word vectors are fine-tuned based on external knowledge, and the fine-tuned word vectors are then used to initialize the word embeddings for a CNN classifier; and (c) our method, where we combine external knowledge with pre-trained word vectors during joint parameter training. The parameters of the "Embedding" component are therefore trained not only based on supervised training data, but also based on external knowledge.

Figure 2(a) illustrates the architecture of the CNN classifier proposed by Kim (2014). The input to the Embedding component is a document.[1] The Embedding component of the model outputs a real-valued matrix consisting of the word vectors representing the document. The CNN takes the matrix as input and predicts the sentiment class distribution of the document.

[1] In practice, a sentence in the original paper.

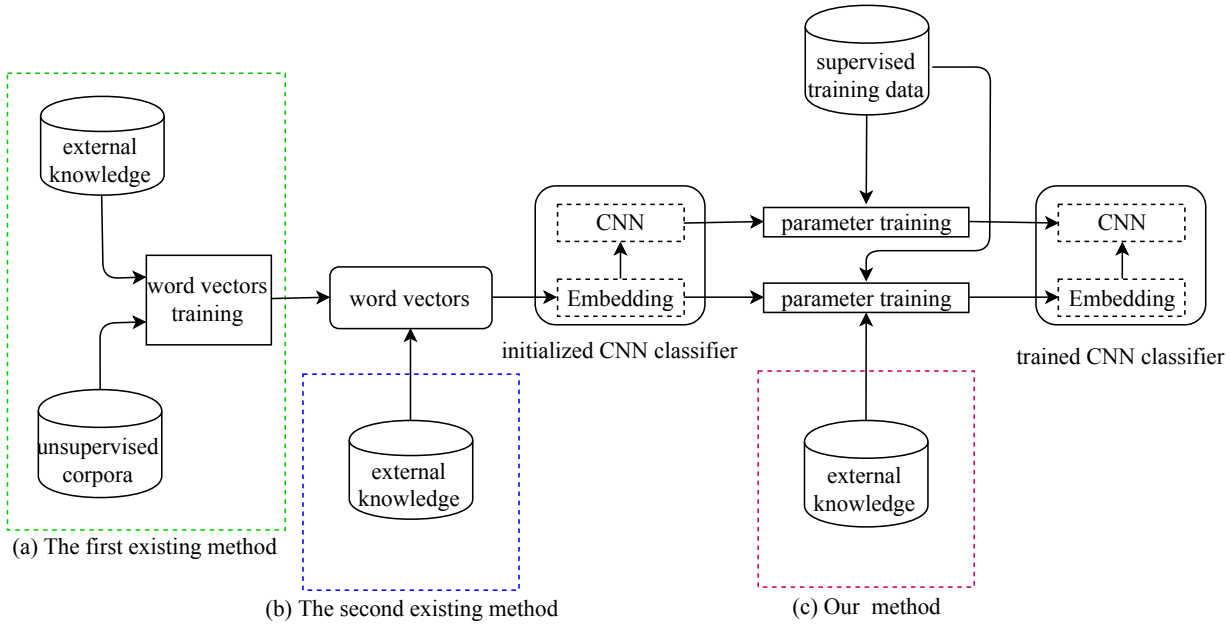
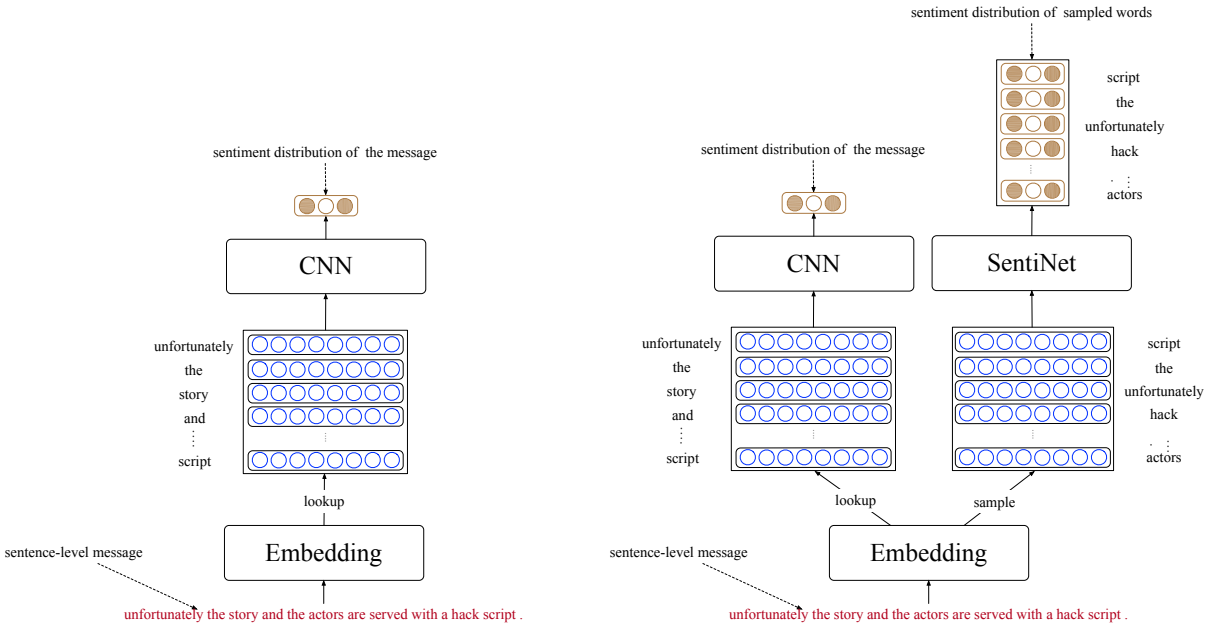


Figure 1: High-level overview of existing methods and our method of encoding external knowledge into word vectors.



(a) CNN-non-static (b) Our framework with SentiNet based on CNN-non-static

Figure 2: The framework of CNN-non-static (Kim, 2014) vs. our method for encoding a sentiment lexicon into word vectors by SentiNet.

Figure 2(b) shows the architecture of our method. The Embedding component is initialized with pre-trained word vectors to train the CNN classifier, and a feedforward neural network called "SentiNet" is used to encode sentiment information from SentiWordNet into the word vectors. In Figure 2(b), the input to Embedding is a vector of words contained in the document, denoted by $[w_1, w_2, \ldots, w_N]$. Embedding outputs a real-valued matrix, $\boldsymbol{W} = [\boldsymbol{w}_1; \boldsymbol{w}_2; \ldots; \boldsymbol{w}_N]^\intercal$, which consists of the word vectors to represent the document. The CNN sentiment classifier takes the matrix as input and predicts the sentiment distribution of the document. Let $\boldsymbol{f}_m^h(\boldsymbol{W})$ and $\boldsymbol{f}_m^g(\boldsymbol{W})$ be the prediction and gold-standard

distribution of the document. The loss function of the CNN is:

$$\mathcal{L}_{\text{CNN}} = CE(\boldsymbol{f}_m^h(\boldsymbol{W}), \boldsymbol{f}_m^g(\boldsymbol{W}))\,, \tag{1}$$

where $CE(\cdot)$ is a scalar value representing the categorical cross-entropy between the prediction and the gold-standard sentiment distribution. The parameters of the CNN will be updated according to $\mathcal{L}_{\text{CNN}}$. M words are sampled from $[w_1, w_2, \ldots, w_N]$, denoted as $[w_{s_1}, w_{s_2}, \ldots, w_{s_M}]$ where $s_k \in [1, N]$ and $k \in [1, M]$. The word vectors of the sampled words are denoted as $[\boldsymbol{w}_{s_1}, \boldsymbol{w}_{s_2}, \ldots, \boldsymbol{w}_{s_M}]$. SentiNet uses the sampled word vectors to predict the word-level sentiment distribution based on SentiWordNet. Let $\boldsymbol{f}_w^h(\boldsymbol{w}_{s_k})$ and $\boldsymbol{f}_w^g(\boldsymbol{w}_{s_k})$ be the prediction and gold-standard sentiment distribution of the word w_{s_k}. The loss function for SentiNet is:

$$\mathcal{L}_{\text{SentiNet}} = \sum_{k=1}^{M} CE(\boldsymbol{f}_w^h(\boldsymbol{w}_{s_k}), \boldsymbol{f}_w^g(\boldsymbol{w}_{s_k}))\,. \tag{2}$$

The parameters of SentiNet are updated according to $\mathcal{L}_{\text{SentiNet}}$, and the parameters of Embedding are updated according to the combined loss $\mathcal{L}$:

$$\mathcal{L} = \mathcal{L}_{\text{CNN}} + \mathcal{L}_{\text{SentiNet}} \tag{3}$$

Instead of encoding external knowledge into word vectors and training the CNN sentiment classifier separately, the word vectors in our method are fine-tuned with not only the CNN, but also with SentiNet, where SentiNet is regarded as an indicator of word-level sentiment information. If the training of SentiNet converges, the word vectors are considered to have sentiment information.

3.2 SentiNet

Our SentiNet method uses a feedforward neural network with one hidden layer, distinct from previous work which has tended to use an objective function based on cosine similarity or Euclidean distance to capture word embedding similarity. SentiNet takes a word vector $\boldsymbol{w}_{s_k}$ as input, and outputs the sentiment distribution of the word, denoted as $\boldsymbol{f}_w^h(\boldsymbol{w}_{s_k})$. The calculation of the sentiment distribution of the word is based on:

$$\boldsymbol{f}_w^h(\boldsymbol{w}_{s_k}) = \text{softmax}(\boldsymbol{\theta}_2(\sigma(\boldsymbol{\theta}_1 \boldsymbol{w}_{s_k} + \boldsymbol{b}_1)) + \boldsymbol{b}_2)\,, \tag{4}$$

where $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$, $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are trainable parameters, and σ is the sigmoid function. The reason for using a feedforward neural network is that it has the same structure as standard word embedding training models such as CBOW, skip-gram and C&W (see Section 2.1).[2]

4 Experiments

4.1 Experimental Setup

Experiments are conducted over four popular, publicly-available sentence-level sentiment classification datasets:

- SemEval2016: The dataset of the SemEval-2016 Message Polarity Classification task. The goal is to classify a given Twitter message according to positive, negative or neutral sentiment (Nakov et al., 2016).
- SemEval2017: The dataset of the SemEval-2017 Message Polarity Classification task. The task formulation is exactly the same as SemEval2016 and the same training data is used, with new test data (Rosenthal et al., 2017).

[2]When training SentiNet with a mini-batch sentences, the words of the mini-batch are classified into positive, neutral and negative according to their sentiment score in the sentiment lexicon. It takes two uniform sample steps to sample a word. First, the word kind (positive, neutral and negative) is sampled uniformly. Then a word is sampled uniformly from the words of the sampled word kind.

	Train			Dev			Test		
	Positive	**Neutral**	**Negative**	**Positive**	**Neutral**	**Negative**	**Positive**	**Neutral**	**Negative**
SemEval2016	9169	8098	4115	—	—	—	7059	10342	3231
SemEval2017	19902	22591	7840	—	—	—	2375	5937	3972
MR	5331	—	5331	—	—	—	—	—	—
SST-2	3610	—	3310	444	—	428	909	—	912

Table 1: Statistical breakdown of the four sentiment classification datasets used in this research.

- MR: Single-sentence movie reviews, labeled as having positive or negative sentiment (Pang and Lee, 2005).
- SST-2: A relabelled version of MR, where the overall review is labeled as having positive or negative sentiment (Socher et al., 2013).

Statistics for the four datasets are shown in Table 1. SemEval2016, SemEval2017 and SST-2 have standard training–test splits. MR does not have such a standard split, so we use 10-fold cross validation, consistent with other work on the dataset. We hold out 10% of the training data for SemEval2016, SemEval2017 and MR for development purposes (e.g. for early stopping), whereas SST-2 has a standard development partition. Consistent with standard practice for the respectivce datasets, we evaluate SemEval2016 and SemEval2017 based on macro-averaged F-score ("F1"), and MR and SST-2 based on classification accuracy.

The CNN sentiment classifier is based on the model proposed by Kim (2014). The differences are that: (1) the penultimate layer of our model is not regularized; and (2) we use the Adam optimizer (Kingma and Ba, 2014), whereas Adadelta (Zeiler, 2012) was used in the original. Hyperparameter tuning was performed over the SemEval2016 development data, based on which we use rectified linear units (ReLU), and a dropout rate of 0.5 on the penultimate layer. Three filter window sizes of 3, 4 and 5 are used, each of which contains 100 feature maps. These values are consistent with the hyperparameter settings of Kim (2014).[3]

4.2 Training Variations

We propose three settings for learning the parameters of Embedding, CNN and SentiNet. Learning all parameters together will lead to co-adaption of the parameters, degrading performance. The number of epochs is set to 100. We use the test result of the epoch with highest performance in dev data. The procedure for training the sentiment classifier is divided into two stages, each made up of 50 epochs. There are three variations according to the training time of Embedding and CNN, named BEFORE, DURING and AFTER, which are optionally integrated with the training of SentiNet:

- **BEFORE**: The parameters of Embedding are updated in the first stage (the first 50 epochs), and the parameters of CNN are updated in the second stage (the second 50 epochs).
- **BEFORE+SentiNet**: The parameters of Embedding and SentiNet are updated in the first stage, and the parameters of CNN are updated in the second stage.
- **DURING**: The parameters of Embedding and CNN are updated synchronously across both stages.
- **DURING+SentiNet**: The parameters of Embedding, SentiNet and CNN are updated synchronously across both stages.
- **AFTER**: The parameters of Embedding are updated in the second stage and the parameters of CNN are updated in the first stage.
- **AFTER+SentiNet**: The parameters of Embedding are updated in the second stage. The parameters of CNN and SentiNet are updated in the first stage.

[3]Our code is available at `https://github.com/yezhejack/SentiNet`.

	SemEval2016	SemEval2017	MR	SST-2
Baseline (CNN + Embedding)	0.606	0.640	0.782	0.825
BEFORE	**0.607**	0.640	**0.786**	**0.831**
BEFORE+SentiNet	**0.609**	0.636	**0.788**	**0.834**
DURING	**0.608**	**0.646**	**0.789**	0.825
DURING+SentiNet	**0.610**	0.637	**0.794**	**0.840**
AFTER	**0.614**	**0.650**	**0.787**	**0.834**
AFTER+SentiNet	**0.616**	**0.651**	**0.794**	**0.837**

Table 2: Results of three variations with and without SentiNet. The evaluation metric in the second and third columns is Macro-averaged F-score, while the measurement in the fourth and fifth columns is accuracy. Above-baseline results are indicated in **bold**, and the best result over each dataset is underlined.

4.3 Results

Baseline system: The baseline system consists of Embedding and CNN. The parameters of Embedding are static during training.

Experimental results on the four sentiment classification datasets are reported in Table 2.[4] BEFORE+SentiNet performs better than BEFORE on SemEval2016, MR and SST-2 datasets. DURING+SentiNet performs better than DURING on SemEval2016, MR and SST-2 datasets. AFTER+SentiNet performs better than AFTER over all four datasets. These results show that SentiNet improves the performance of the CNN sentiment classifier. AFTER+SentiNet achieves the best performance on SemEval2016 and SemEval2017, while DURING+SentiNet achieves the best performance on MR and SST-2. AFTER+SentiNet performs better than DURING+SentiNet, as fine-tuning Embedding after CNN and SentiNet can avoid the co-adaption of parameters. BEFORE+SentiNet is the worst because CNN has not been trained when Embedding is tuned, meaning Embedding does not benefit from the task-specific supervised training data.

4.4 Comparison with Other Methods

Experimental setup: We compare our method with three existing methods for encoding external knowledge resources into word embeddings: (1) attract-repel (Mrkšić et al., 2017); (2) PARAGRAM (Wieting et al., 2015); and (3) counter-fitting (Mrkšić et al., 2016). We experiment with word embeddings generated by these three methods to initialize Embedding in our model. These embeddings encode different external knowledges. Embedding and CNN are jointly trained over the supervised training data. For our proposed method, we compare against AFTER+SentiNet and DURING+SentiNet, based on the results from Section 4.3.

The benchmark methods we compare our method against are as follows:

- **word2vec**: The 300 dimensional pre-trained word vectors based on Google News data, and distributed by Google.[5]

- **attract-repel-pos-neg**: The attract-repel[6] method of Mrkšić et al. (2017), which is used to encode the sentiment lexicon into word2vec word embeddings. The sentiment lexicon is the same as the one used in our method.

- **attract-repel-ant-syn**: The attract-repel method of Mrkšić et al. (2017) applied to word2vec word embeddings, based on antonym/synonym lexicons.

[4] Based on McNemar's test, the difference between AFTER-SentiNet and AFTER is significant for SemEval2016 ($p < 0.05$) and SST-2 ($p < 0.01$), but not SemEval2017 or MR ($p > 0.05$).

[5] https://code.google.com/archive/p/word2vec/

[6] https://github.com/nmrksic/attract-repel

	SemEval2016	SemEval2017	MR	SST-2
word2vec (Mikolov et al., 2013)	0.608	0.647	0.792	0.837
attract-repel-pos-neg (Mrkšić et al., 2017)	0.608	0.641	0.792	0.836
attract-repel-ant-syn (Mrkšić et al., 2017)	0.608	0.626	0.785	0.817
PARAGRAM (Wieting et al., 2015)	0.599	0.632	0.781	**0.839**
counter-fitting (Mrkšić et al., 2016)	0.603	0.627	0.780	**0.838**
DURING+SentiNet	0.601	0.641	0.790	**0.842**
AFTER+SentiNet	**0.613**	**0.648**	**0.796**	**0.844**
SwissCheese (Deriu et al., 2016)	0.633	—	—	—
SENSEI-LIF (Rouvier and Favre, 2016)	0.630	—	—	—
UNIMELB (Xu et al., 2016)	0.617	—	—	—
BB_twtr (Cliche, 2017)	—	0.685	—	—
CNN-non-static (Kim, 2014)	—	0.685	—	—
Re(word2vec) (Yu et al., 2017)	—	—	—	0.879

Table 3: Experiment results of word vectors on the four Sentiment Classification Datasets. The measurement of the second and third columns is Macro F1. The measurement of the fourth and fifth columns is Accuracy.

- **PARAGRAM**: The word vectors of PARAGRAM, generated by encoding PPDB into word2vec word embeddings, based on the pre-trained PARAGRAM word embedding set.[7]
- **counter-fitting**: The word vectors of counter-fitting, generated by encoding an antonym/synonym lexicon into PARAGRAM word vectors, based on the pre-trained word embedding set.[8]

Experimental Analysis: The results of the experiment are shown in Table 3. DURING+SentiNet and AFTER+SentiNet are the results of our method, run on the same sentiment lexicons as other methods.[9]

AFTER+SentiNet performs better than the other methods over all four datasets. The reason is that our word vectors encode not only sentiment information through SentiNet, but also propagate supervision signal from the sentiment analysis task. Comparing the same method with different external knowledge sources, the sentiment lexicon has greater utility than the antonym/synonym lexicon for sentiment classification. Counter-fitting and PARAGRAM outperform word2vec only on SST-2, and not the other three datasets.

The last 6 lines in Table 3 detail state-of-the-art results for the four datasets. SwissCheese (Deriu et al., 2016) leverages large amounts of distant-supervised data to train an ensemble of CNNs. SENSEI-LIF (Rouvier and Favre, 2016) trains three kinds of word embeddings: lexical embeddings, part-of-speech embeddings and sentiment embeddings, in order to train three CNN sentiment classifiers. The three classifiers are combined into a fusion model to make the final prediction. UNIMELB (Xu et al., 2016) consists of a soft-voting ensemble of a language model adapted to classification, a CNN, and a long-short term memory network (LSTM). BB_twtr (Cliche, 2017) also uses a large amount of unlabelled data to pre-train word vectors. It also ensembles CNNs and LSTMs to boost performance. All of these methods are based on ensembling classifiers to improve results, while our method uses a single classifier. Additionally, our CNN model is tuned based on the development data of SemEval2016, and not tuned to the respective datasets. In that sense, the results are highly encouraging, and there is every expectation that ensemble methods would benefit from the incorporation of the outputs of our model.

[7] https://github.com/nmrksic/counter-fitting/blob/master/word_vectors/paragram.txt.zip

[8] https://github.com/nmrksic/counter-fitting/blob/master/word_vectors/counter-fitted-vectors.txt.zip

[9] In Table 2, the whole vocabulary of pre-trained word2vec is used, whereas in Table 3, we use the intersection of the vocabulary for word2vec, attract-repel-pos-neg, attract-repel-ant-syn, PARAGRAM and counter-fitting. As such, the results are not directly comparable.

5 Conclusion

In this paper, we have proposed a novel method for encoding a sentiment lexicon into word embeddings for sentiment analysis, based on a method we call SentiNet. SentiNet uses a feedforward neural network to lexically encode sentiment information, instead of using an objective function based on cosine similarity or Euclidean distance. Three variations — BEFORE+SentiNet, DURING+SentiNet and AFTER+SentiNet — were proposed to combine SentiNet with Embedding and CNN. Experiments on sentiment classification datasets show that DURING+SentiNet and AFTER+SentiNet are the most effective ways of combining the training of SentiNet with the training of Embedding and the CNN. Our method can combine sentiment information from the training data and the sentiment lexicon to fine-tune word vectors for sentiment classification. For sentiment classification, we have shown sentiment lexicons to have greater utility than antonym/synonym lexicons and paraphrase databases.

In future work, we plan to experiment with encoding other external lexical knowledge beyond sentiment lexicons, and to explore frameworks for encoding external knowledge into deep neural networks.

6 Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant No. 61673266).

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2010)*, Valletta, Malta.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL 1998)*, pages 86–90, Montréal, Canada.

Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Knowledge-powered deep learning for word embedding. In *Proceedings of the 2014 European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2014)*, pages 132–148, Nancy, France.

Mathieu Cliche. 2017. Bb_twtr at SemEval-2017 task 4: Twitter sentiment analysis with CNNs and LSTMs. In *Proceedings of the 11th International Workshop on Semantic Evaluation, (SemEval 2017)*, pages 573–580, Vancouver, Canada.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Jan Deriu, Maurice Gonzenbach, Fatih Uzdilli, Aurélien Lucchi, Valeria De Luca, and Martin Jaggi. 2016. SwissCheese at SemEval-2016 task 4: Sentiment classification using an ensemble of convolutional neural networks with distant supervision. In *Proceedings of the 10th International Workshop on Semantic Evaluation, (SemEval 2016)*, pages 1124–1128, San Diego, USA.

Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 69–78, Dublin, Ireland.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2015)*, pages 1606–1615, Denver, USA.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: the concept revisited. *ACM Trans. Inf. Syst.*, 20(1):116–131.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: the paraphrase database. In *Proceedings of Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics (HLT NAACL 2013)*, pages 758–764, Atlanta, USA.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Douwe Kiela, Felix Hill, and Stephen Clark. 2015. Specializing word embeddings for similarity or relatedness. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 2044–2048, Lisbon, Portugal.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1746–1751, Doha, Qatar.

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

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), Volume 2: Short Papers*, pages 302–308, Baltimore, USA.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL 2011)*, pages 142–150, Portland, USA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Annual Conference on Advances in Neural Information Processing Systems (NIPS 2013)*, pages 3111–3119, Lake Tahoe, USA.

George A. Miller. 1995. WordNet: A lexical database for English. *Commun. ACM*, 38(11):39–41.

Andriy Mnih and Geoffrey E. Hinton. 2008. A scalable hierarchical distributed language model. In *Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems (NIPS 2008)*, pages 1081–1088, Vancouver, Canada.

Nikola Mrkšić, Séaghdha Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2016)*, pages 142–148, San Diego, USA.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gai, Anna Korhonen, and Steve Young. 2017. Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the Association for Computational Linguistics*, 5:309–324.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. SemEval-2016 task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation, (SemEval 2016)*, pages 1–18, San Diego, USA.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artif. Intell.*, 193:217–250.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 115–124, Ann Arbor, USA.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543, Doha, Qatar.

Yafeng Ren, Ruimin Wang, and Donghong Ji. 2016. A topic-enhanced word embedding for Twitter sentiment classification. *Inf. Sci.*, 369:188–198.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation, (SemEval 2017)*, pages 502–518, Vancouver, Canada.

Sascha Rothe and Hinrich Schütze. 2015. AutoExtend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2015)*, pages 1793–1803, Beijing, China.

Mickael Rouvier and Benoît Favre. 2016. SENSEI-LIF at SemEval-2016 task 4: Polarity embedding fusion for robust sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation, (SemEval 2016)*, pages 202–208, San Diego, USA.

Aliaksei Severyn and Alessandro Moschitti. 2015. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2015)*, pages 959–962, Santiago, Chile.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, USA.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1555–1565.

Duyu Tang, Furu Wei, Bing Qin, Nan Yang, Ting Liu, and Ming Zhou. 2016. Sentiment embeddings with applications to sentiment analysis. *IEEE Trans. Knowl. Data Eng.*, 28(2):496–509.

Ivan Vulić, Nikola Mrkšić, Roi Reichart, Diarmuid Ó Séaghdha, Steve J. Young, and Anna Korhonen. 2017. Morph-fitting: Fine-tuning word vector spaces with simple language-specific rules. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 56–68, Vancouver, Canada.

Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 English lemmas. *Behavior Research Methods*, 45(4):1191–1207.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 3:345–358.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. RC-NET: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM 2014)*, pages 1219–1228, Shanghai, China.

Steven Xu, Huizhi Liang, and Timothy Baldwin. 2016. UNIMELB at SemEval-2016 tasks 4a and 4b: An ensemble of neural networks and a word2vec based model for sentiment classification. In *Proceedings of the 10th International Workshop on Semantic Evaluation, (SemEval 2016)*, pages 183–189, San Diego, USA.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), Volume 2: Short Papers*, pages 545–550, Baltimore, USA.

Liang-Chih Yu, Jin Wang, K. Robert Lai, and Xue-Jie Zhang. 2017. Refining word embeddings for sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 534–539, Copenhagen, Denmark.

Matthew D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. *CoRR*, abs/1212.5701.

Yunxiao Zhou, Zhihua Zhang, and Man Lan. 2016. ECNU at SemEval-2016 task 4: An empirical investigation of traditional NLP features and word embedding features for sentence-level and topic-level sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation, (SemEval 2016)*, pages 256–261, San Diego, USA.

Multi-Task Neural Models for Translating Between Styles Within and Across Languages

Xing Niu
University of Maryland
xingniu@cs.umd.edu

Sudha Rao
University of Maryland
raosudha@cs.umd.edu

Marine Carpuat
University of Maryland
marine@cs.umd.edu

Abstract

Generating natural language requires conveying content in an appropriate style. We explore two related tasks on generating text of varying formality: monolingual formality transfer and formality-sensitive machine translation. We propose to solve these tasks jointly using multi-task learning, and show that our models achieve state-of-the-art performance for formality transfer and are able to perform formality-sensitive translation without being explicitly trained on style-annotated translation examples.

1 Introduction

Generating language in the appropriate style is a requirement for applications that generate natural language, as the style of a text conveys important information beyond its literal meaning (Hovy, 1987). Heylighen and Dewaele (1999) and Biber (2014) have argued that the formal-informal dimension is a core dimension of stylistic variation. In this work, we focus on the problem of generating text for a desired formality level. It has been recently studied in two distinct settings: (1) Rao and Tetreault (2018) addressed the task of *Formality Transfer* (FT) where given an informal sentence in English, systems are asked to output a formal equivalent, or vice-versa; (2) Niu et al. (2017) introduced the task of *Formality-Sensitive Machine Translation* (FSMT), where given a sentence in French and a desired formality level (approximating the intended audience of the translation), systems are asked to produce an English translation of the desired formality level. While FT and FSMT can both be framed as Machine Translation (MT), appropriate training examples are much harder to obtain than for traditional machine translation tasks. FT requires sentence pairs that express the same meaning in two different styles, which rarely occur naturally and are therefore only available in small quantities. FSMT can draw from existing parallel corpora in diverse styles, but would ideally require not only sentence pairs, but e.g., sentence triplets that contain a French input, its formal English translation, and its informal English translation.

We hypothesize that FT and FSMT can benefit from being addressed jointly, by sharing information from two distinct types of supervision: sentence pairs in the same language that capture style difference, and translation pairs drawn from corpora of various styles. Inspired by the benefits of multi-task learning (Caruana, 1997) for natural language processing tasks in general (Collobert and Weston, 2008; Liu et al., 2015; Luong et al., 2016), and for multilingual MT in particular (Johnson et al., 2017), we introduce a model based on Neural Machine Translation (NMT) that jointly learns to perform both monolingual FT and bilingual FSMT. As can be seen in Figure 1, given an English sentence and a tag (formal or informal), our model paraphrases the input sentence into the desired formality. The same model can also take in a French sentence, and produce a formal or an informal English translation as desired.

Designing this model requires addressing several questions: Can we build a single model that performs formality transfer in both directions? How to best combine monolingual examples of formality transfer and bilingual examples of translation? What kind of bilingual examples are most useful for the joint task? Can our joint model learn to perform FSMT without being explicitly trained on style-annotated translation examples? We explore these questions by conducting an empirical study on English FT and

Proceedings of the 27th International Conference on Computational Linguistics, pages 1008–1021
Santa Fe, New Mexico, USA, August 20-26, 2018.

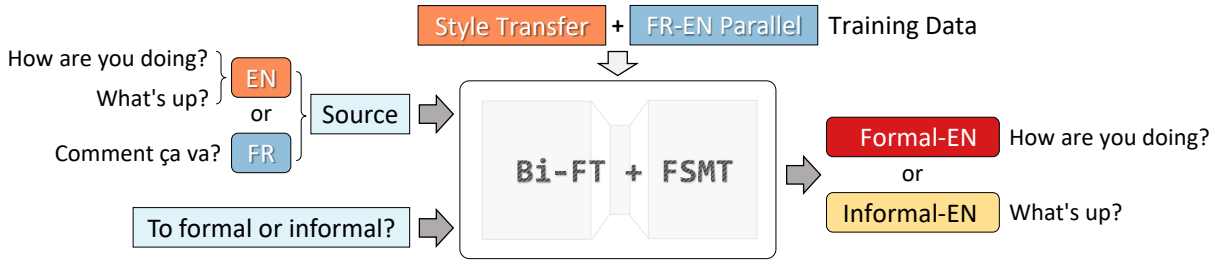


Figure 1: System overview: our multi-task learning model can perform both bi-directional English formality transfer and translate French to English with desired formality. It is trained jointly on monolingual formality transfer data and bilingual translation data.

French-English FSMT, using both automatic and human evaluation. Our results show the benefits of the multi-task learning approach, improving the state-of-the-art on the FT task, and yielding competitive performance on FSMT without style-annotated translation examples. Along the way, we also improve over prior results on FT using a single NMT model that can transfer between styles in both directions.

2 Background

Style Transfer can naturally be framed as a sequence to sequence translation problem given sentence pairs that are paraphrases in two distinct styles. These parallel style corpora are constructed by creatively collecting existing texts of varying styles, and are therefore rare and much smaller than machine translation parallel corpora. For instance, Xu et al. (2012) scrape modern translations of Shakespeare's plays and use a phrase-based MT (PBMT) system to paraphrase Shakespearean English into/from modern English. Jhamtani et al. (2017) improve performance on this dataset using neural translation model with pointers to enable copy actions. The availability of parallel standard and simple Wikipedia (and sometimes additional human rewrites) makes text simplification a popular style transfer task, typically addressed using machine translation models ranging from syntax-based MT (Zhu et al., 2010; Xu et al., 2016), phrase-based MT (Coster and Kauchak, 2011; Wubben et al., 2012) to neural MT (Wang et al., 2016) trained via reinforcement learning (Zhang and Lapata, 2017).

Naturally occurring examples of parallel formal-informal sentences are harder to find. Prior work relied on synthetic examples generated based on lists of words of known formality (Sheikha and Inkpen, 2011). This state of affairs recently changed, with the introduction of the first large scale parallel corpus for formality transfer, GYAFC (Grammarly's Yahoo Answers Formality Corpus). 110K informal sentences were collected from Yahoo Answers and they were rewritten in a formal style via crowd-sourcing, which made it possible to benchmark style transfer systems based on both PBMT and NMT models (Rao and Tetreault, 2018). In this work, we leverage this corpus to enable multi-task FT and FSMT.

Recent work also explores how to perform style transfer without parallel data. However, this line of work considers transformations that alter the original meaning (e.g., changes in sentiment or topic), while we view style transfer as meaning-preserving. An auto-encoder is used to encode a sequence to a latent representation which is then decoded to get the style transferred output sequence (Mueller et al., 2017; Hu et al., 2017; Shen et al., 2017; Fu et al., 2018; Prabhumoye et al., 2018).

Style in Machine Translation has received little attention in recent MT architectures. Mima et al. (1997) improve rule-based MT by using extra-linguistic information such as speaker's role and gender. Lewis et al. (2015) and Niu and Carpuat (2016) equate style with domain, and train conversational MT systems by selecting in-domain (i.e. conversation-like) training data. Similarly, Wintner et al. (2017) and Michel and Neubig (2018) take an adaptation approach to personalize MT with gender-specific or speaker-specific data. Other work has focused on specific realizations of stylistic variations, such as T-V pronoun selection for translation into German (Sennrich et al., 2016a) or controlling voice (Yamagishi et al., 2016). In contrast, we adopt the broader range of style variations considered in our prior work, which introduced the FSMT task (Niu et al., 2017): in FSMT, the MT system takes a desired formality level as

an additional input, to represent the target audience of a translation, which human translators implicitly take into account. This task was addressed via n-best re-ranking in phrase-based MT — translation hypotheses whose formality are closer to desired formality are promoted.

By contrast, in this work we use neural MT which is based on the **Attentional Recurrent Encoder-Decoder** model (Bahdanau et al., 2015; Luong et al., 2016). The input is encoded into a sequence of vector representations while the decoder adaptively computes a weighted sum of these vectors as the context vector for each decoding step.

In the joint model, we employ **Side Constraints** as the formality input to restrict the generation of the output sentence (Figure 1). Prior work has successfully implemented side constraints as a special token added to each source sentence. These tokens are embedded into the source sentence representation and control target sequence generation via the attention mechanism. Sennrich et al. (2016a) append <T> or <V> (i.e. T-V pronoun distinction) to the source text to indicate which pronoun is preferred in the German output. Johnson et al. (2017) and Niu et al. (2018) concatenate parallel data of various language directions and mark the source with the desired output language to perform multilingual or bi-directional NMT. Kobus et al. (2017) and Chu et al. (2017) add domain tags for domain adaptation in NMT.

3 Approach

We describe our unified model for performing FT in both directions (Section 3.1), our FSMT model with side constraints (Section 3.2) and finally our multi-task learning model that jointly learns to perform FT and FSMT (Section 3.3). All models rely on the same NMT architecture: attentional recurrent sequence-to-sequence models.

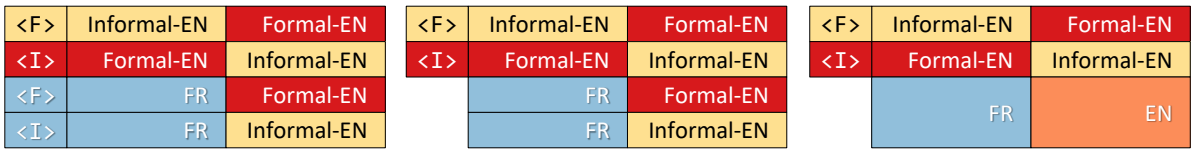
3.1 Bi-Directional Formality Transfer

Rao and Tetreault (2018) used independent neural machine translation models for each formality transfer direction (`informal`→`formal` and `formal`→`informal`). Inspired by the bi-directional NMT for low-resource languages (Niu et al., 2018), we propose a unified model that can handle either direction — we concatenate the parallel data from the two directions of formality transfer and attach a tag to the beginning of each source sentence denoting the desired target formality level i.e. <F> for transferring to formal and <I> for transferring to informal. This enables our FT model to learn to transfer to the correct style via attending to the tag in the source embedding. We train an NMT model on this combined dataset. Since both the source and target sentences come from the same language, we encourage their representations to lie in the same distributional vector space by (1) building a shared Byte-Pair Encoding (BPE) model on source and target data (Sennrich et al., 2016b) and (2) tying source and target word embeddings (Press and Wolf, 2017).

3.2 Formality-Sensitive Machine Translation with Side Constraints

Inspired by Sennrich et al. (2016a), we use side constraints on parallel translation examples to control output formality. At training time, this requires a tag that captures the formality of the target sentence for every sentence pair. Given the vast range of text variations that influence style, we cannot obtain tags using rules as for T-V pronoun distinctions (Sennrich et al., 2016a). Instead, we categorize French-English parallel data into formal vs. informal categories by comparing them to the informal and formal English from the GYAFC corpus.

We adopt a data selection technique, Cross-Entropy Difference (CED) (Moore and Lewis, 2010), to rank English sentences in the bilingual corpus by their relative distance to each style. First, we consider formal English as the target style and define $CED(s) = H_{formal}(s) - H_{informal}(s)$, where $H_{formal}(s)$ is the cross-entropy between a sentence s and the formal language model. Smaller CED indicates an English sentence that is more similar to the formal English corpus and less similar to the informal English corpus. We rank English sentences by their CED scores and select the top N sentences (choice of N discussed in Section 6). Pairing these N English sentences with their parallel French source, we get the formal sample of our bilingual data. Similarly, we construct the informal sample using informal English as the target style. Finally, we combine the formal and the informal samples, attach the <F> and <I>



(a) Formality tags on bilingual data + 2-Style selection

(b) No formality tags on bilingual data + 2-Style selection

(c) No formality tags on bilingual data + Random selection

Figure 2: The training data used for multi-task learning models. The bi-directional formality transfer data and the bilingual data (e.g. FR-EN) of equivalent size are always concatenated.

tags to corresponding source French sentences (i.e. the bottom two rows of data in Figure 2a) and train an NMT model for our FSMT task.

3.3 Multi-Task Learning

We propose a multi-task learning model to jointly perform FT and FSMT using a many-to-one (i.e. multi-language to English) sequence to sequence model (Luong et al., 2016). Following Johnson et al. (2017), we implement this approach using shared encoders and decoders. This approach can use existing NMT architectures without modifications. We design three models to investigate how to best incorporate side constraints at training time, and the benefits of sharing representations for style and language.

MultiTask-tag-style is a straightforward combination of the transfer and translation models above. We hypothesize that using the bilingual parallel data where English is the target could enhance English FT in terms of target language modeling, especially when the bilingual data has similar topics and styles. We therefore combine equal sizes of formality tagged training data (selected as described in Section 3.2) from our FT and FSMT tasks in this configuration (Figure 2a).

MultiTask-style is designed to test whether formality tags for bilingual examples are necessary. We hypothesize that the knowledge of controlling target formality for the FSMT task can be learned from the FT data since the source embeddings of formality tags are shared between the FT and the FSMT tasks. We therefore combine the formality tagged FT data with the MT data without their tags (Figure 2b).

MultiTask-random investigates the impact of the similarity between formality transfer and bilingual examples. Selecting bilingual data which is similar to the GYAFC corpus is not necessarily beneficial for the FSMT task especially when French-English bilingual examples are drawn from a domain distant from the GYAFC corpus. In this configuration, we test how well our model performs FSMT if bilingual examples are randomly selected instead (Figure 2c).

4 Experimental Setup

FT data: We use the GYAFC corpus introduced by Rao and Tetreault (2018) as our FT data. This corpus consists of 110K informal sentences from two domains of Yahoo Answers (*Entertainment and Music (E&M)* and *Family and Relationships (F&R)*) paired with their formal rewrites by humans. The train split consists of 100K informal-formal sentence pairs whereas the dev/test sets consist of roughly 5K source-style sentences paired with four reference target-style human rewrites for both transfer directions.

FSMT data: We evaluate the FSMT models on a large-scale French to English (FR-EN) translation task. Examples are drawn from OpenSubtitles2016 (Lison and Tiedemann, 2016) which consists of movie and television subtitles and is thus more similar to the GYAFC corpus compared to news or parliament proceedings. This is a noisy dataset where aligned French and English sentences often do not have the same meaning, so we use a bilingual semantic similarity detector to select 20,005,000 least divergent examples from ∼27.5M deduplicated sentence pairs in the original set (Vyas et al., 2018). Selected examples are then randomly split into a 20M training pool, a 2.5K dev set and a 2.5K test set.

Preprocessing: We apply four pre-processing steps to both FT and MT data: normalization, tokenization, true-casing, and joint source-target BPE with 32,000 operations for NMT (Sennrich et al., 2016b).

NMT Configuration: We use the standard attentional encoder-decoder architecture implemented in the Sockeye toolkit (Hieber et al., 2017). Our translation model uses a bi-directional encoder with a single

LSTM layer (Bahdanau et al., 2015) of size 512, multilayer perceptron attention with a layer size of 512, and word representations of size 512. We apply layer normalization and tie the source and target embeddings as well as the output layer's weight matrix. We add dropout to embeddings and RNNs of the encoder and decoder with probability 0.2. We train using the Adam optimizer with a batch size of 64 sentences and checkpoint the model every 1000 updates (Kingma and Ba, 2015). Training stops after 8 checkpoints without improvement of validation perplexity. We decode with a beam size of 5. We train four randomly seeded models for each experiment and combine them in a linear ensemble for decoding.[1]

5 Evaluation Protocol

5.1 Automatic Evaluation

We evaluate both FT and FSMT tasks using BLEU (Papineni et al., 2002), which compares the model output with four reference target-style rewrites for FT and a single reference translation for FSMT. We report case-sensitive BLEU with standard WMT tokenization.[2] For FT, Rao and Tetreault (2018) show that BLEU correlates well with the overall system ranking assigned by humans. For FSMT, BLEU is an imperfect metric as it conflates mismatches due to translation errors and due to correct style variations. We therefore turn to human evaluation to isolate formality differences from translation quality.

5.2 Human Evaluation

Following Rao and Tetreault (2018), we assess model outputs on three criteria: *formality*, *fluency* and *meaning preservation*. Since the goal of our evaluation is to compare models, our evaluation scheme asks workers to compare sentence pairs on these three criteria instead of rating each sentence in isolation. We collect human judgments using CrowdFlower on 300 samples of each model outputs. For FT, we compare the top performing NMT benchmark model in Rao and Tetreault (2018) with our best FT model. For FSMT, we compare outputs from three representative models: NMT-constraint, MultiTask-random and PBMT-random.[3]

Formality. For FT, we want to measure the amount of style variation introduced by a model. Hence, we ask workers to compare the source-style sentence with its target-style model output. For FSMT, we want to measure the amount of style variation between two different translations by the same model. Hence, we ask workers to compare the "informal" English translation and the "formal" English translation of the same source sentence in French.[4] We design a five point scale for comparing the formality of two sentences ranging from one being much more formal than the other to the other being much more formal than the first, giving us a value between 0 and 2 for each sentence pair.[5]

Fluency. For both FT and FSMT tasks, we want to understand how fluent are the different model outputs. Hence, we ask workers to compare the fluency of two model outputs of the same target style. Similar to formality evaluation, we design a five point scale for comparing the fluency of two sentences, giving us a value between 0 and 2 for each sentence pair.

Meaning Preservation. For FT, we want to measure the amount of meaning preserved during formality transfer. Hence, we ask workers to compare the source-style sentence and the target-style model output. For FSMT, we want to measure the amount of meaning preserved between two different translations by the same model. Hence, we ask workers to compare the "informal" English translation and the "formal" English translation of the same source sentence in French. We design a four point scale to compare the meaning of two sentences ranging from the two being completely equivalent to the two being not equivalent, giving us a value between 0 and 3 for each sentence pair.

[1] Data and scripts available at `https://github.com/xingniu/multitask-ft-fsmt`.

[2] `https://github.com/EdinburghNLP/nematus/blob/master/data/multi-bleu-detok.perl`

[3] Note that we do not compare with the English reference translation. A more detailed description of the human annotation protocol can be found in the appendix.

[4] Evaluating which systems produces the most (in)formal output is an orthogonal question that we leave to future work.

[5] Details on the conversion from a five point scale to a value between 0 and 2 is in the appendix.

Model	Informal→Formal		Formal→Informal	
	E&M	F&R	E&M	F&R
PBMT (Rao and Tetreault, 2018)	68.22	72.94	33.54	32.64
NMT Baseline (Rao and Tetreault, 2018)	58.80	68.28	30.57	36.71
NMT Combined (Rao and Tetreault, 2018)	68.41	74.22	33.56	35.03
NMT Baseline	65.34	71.28	32.36	36.23
Bi-directional FT	66.30	71.97	34.00	36.33
+ training on E&M + F&R	69.20	73.52	35.44	37.72
+ ensemble decoding ($\times$4)	71.36	74.49	36.18	38.34
+ multi-task learning (MultiTask-tag-style)	**72.13**	**75.37**	**38.04**	**39.09**

Table 1: Automatic evaluation of Formality Transfer with BLEU scores. The bi-directional model with three stacked improvements achieves the best overall performance. The improvement over the second best system is statistically significant at $p < 0.05$ using bootstrap resampling (Koehn, 2004).

6 Formality Transfer Experiments

6.1 Baseline Models from Rao and Tetreault (2018)

PBMT is a phrase-based machine translation model trained on the GYAFC corpus using a training regime consisting of self-training, data sub-selection and a large language model.
NMT Baseline uses OpenNMT-py (Klein et al., 2017). Rao and Tetreault (2018) use a pre-processing step to make source informal sentences more formal and source formal sentences more informal by rules such as re-casing. Word embeddings pre-trained on Yahoo Answers are also used.
NMT Combined is Rao and Tetreault's best performing NMT model trained on the rule-processed GYAFC corpus, with additional forward and backward translations produced by the PBMT model.

6.2 Our Models

NMT Baseline: Our NMT baseline uses Sockeye instead of OpenNMT-py and is trained on raw datasets of two domains and two transfer directions.
Bi-directional FT: Our initial bi-directional model is trained on bi-directional data from both domains with formality tags. It is incrementally augmented with three modifications to get the final multi-task model (i.e. MultiTask-tag-style as described in Section 3.3): (1) We combine training sets of two domains (E&M+F&R) together and train a single model on it. (2) We use ensemble decoding by training four randomly seeded models on the combined data. (3) We add formality-tagged bilingual data and train the model using multi-task learning to jointly learn FT and FSMT. Suppose the amount of original bi-directional FT data is n, we always select kn bilingual data where k is an integer. We also duplicate FT data to make it match the size of selected bilingual data.

6.3 Results

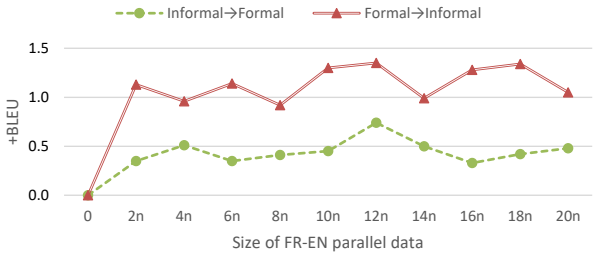
Automatic Evaluation. As shown in Table 1, our NMT baselines yield surprisingly better BLEU scores than those of Rao and Tetreault (2018), even without using rule-processed source training data and pre-trained word embeddings. We attribute the difference to the more optimized NMT toolkit we use.

Initial bi-directional models outperforms uni-directional models. This matches the behavior of bi-directional NMT in low-resource settings studied by Niu et al. (2018) — we work with a relatively small amount of training data ($\sim$50K), and FT models benefit from doubling the size of training data without being confused by mixing two transfer directions. For the same reason, increasing the training data by combining two domains together improves performance further. Ensemble decoding is a consistently effective technique used by NMT and it enhances our NMT-based FT models as expected.

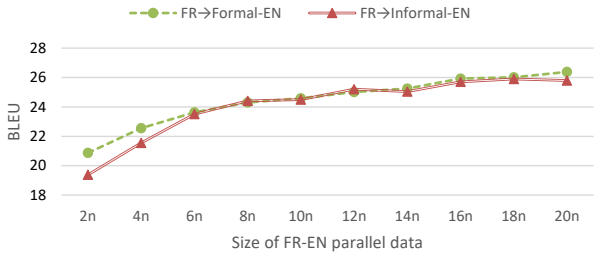
Incorporating the bilingual parallel data by multi-task learning yields further improvement. The target side of bilingual data is selected based on the closeness to the GYAFC corpus, so we hypothesize that the higher quality comes from better target language modeling by training on more English text.

	Model A	Model B	Formality Diff Range = [0,2]		Meaning Preservation Range = [0,3]
			I→F	F→I	
FT	Source	NMT Combined	0.54	0.45	2.94
	Source	MultiTask-tag-style	0.59	**0.64**	2.92
FSMT	NMT-constraint I	NMT-constraint F	**0.35**		2.95
	NMT MultiTask-random I	NMT MultiTask-random F	**0.32**		2.90
	PBMT-random I	PBMT-random F	0.05		2.97

Table 2: Human evaluation of formality difference and meaning preservation. MultiTask-tag-style generates significantly more informal (F→I) English than NMT Combined (p<0.05 using the t-test, see Section 6.3). PBMT-random does not control formality effectively when comparing its informal (I) and formal (F) output (Section 7.2). Formality scores are relatively low because workers rarely choose "much more (in)formal". All models preserve meaning equally well.



(a) BLEU improvements for formality transfer.



(b) BLEU scores for machine translation.

Figure 3: BLEU improvements or scores for four transfer/translation directions vs. the size of FR-EN parallel data. n in x-axis equals to the original size of bi-directional style transfer training data. Formality transfer improves with bilingual data and the performance converges quickly. The translation quality increases monotonously with the size of training data.

Human Evaluation. The superior performance of the best FT model (i.e. MultiTask-tag-style) is also reflected in our human evaluation (see Table 2). It generates slightly more formal English (0.59 vs 0.54) and significantly more informal English (0.64 vs 0.45) than NMT Combined. This is consistent with BLEU differences in Table 1 which show that MultiTask-tag-style yields bigger improvements when transferring formal language to informal. Both models have good quality with respect to meaning preservation (2.94 vs 2.92) and workers can hardly find any fluency difference between outputs of these two models by assigning 0.03 in average in the fluency test (0 means no difference).

Impact of Bilingual Data Size. We evaluate the impact of selected bilingual data size on the combination of development sets from two domains in GYAFC and show the results in Figure 3. The quality of formality transfer improves instantly when using bilingual data and it soon converges when more data is used. Meanwhile, the translation quality increases monotonously with the size of training data. The optimal point is a hyper-parameter that can be determined on the development set. We empirically choose $n = 12$ since it works best for formality transfer and yields reasonable translation quality.

6.4 Qualitative Analysis

We manually inspect 100 randomly selected samples from our evaluation set and compare the target-style output of our best model (MultiTask-tag-style) with that of the best baseline model (NMT-Combined) from Rao and Tetreault (2018). Table 3 shows some samples representative of the trends we find for `informal`→`formal` (3a) and `formal`→`informal` (3b) tasks.

In majority of the cases, the two models produce similar outputs as can be expected since they use similar NMT architectures. In cases where the two outputs differ, in the I→F task, we find that our model produces a more formal output by introducing phrasal level changes (first sample in 3a) or by moving

3a: informal→formal		
1	Original I	chill out sweetie everything will be just fine eventually
	NMT-Combined F	Can you chill out sweetie everything will be just fine eventually.
	MultiTask-tag-style F	Calm down, sweetie, everything will be fine eventually.
2	Original I	Dakota Fanning.....I know that she is only 12 but she is really famous.
	NMT-Combined F	Dakota Fanning.i know that she is only twelve, but she is famous.
	MultiTask-tag-style F	I know that Dakota Fanning is only twelve, but she is really famous.
3	Original I	depends....usully they are about ur personailty but not wat ur gonna do iwith ur life.
	NMT-Combined F	Depends.usully they are about your personailty, but not what your going to do iwith your life.
	MultiTask-tag-style F	It depends. They are about your personality, but not what you are going to do with your life.
4	Original I	THAT DEPENDS...ARE YOU A HOTTIE W/A BODY?
	NMT-Combined F	That depends, are you a hottie with a body?
	MultiTask-tag-style F	That depends. Are you a HOTTIE W / A BODY?
3b: formal→informal		
1	Original F	Therefore I would say that they do succeed but not frequently. I hope this is helpful.
	NMT-Combined I	So I would say that they do failing but not frequently, I hope this is helps.
	MultiTask-tag-style I	so i would say they do it but not all the time, hope this helps.
2	Original F	I am simply inquiring because people behave as though they are no longer interested in them.
	NMT-Combined I	I am just asking because people act as though they are no longer interested in them.
	MultiTask-tag-style I	I'm just asking because people act like they don't like them anymore.
3	Original F	Hello, I am interested in visiting your country.
	NMT-Combined I	Hi, I'm interested in visiting your country.
	MultiTask-tag-style I	hi, I'm going to go to your country.

Table 3: Sample model outputs for the Formality Transfer (FT) task.

phrases around (second sample in 3a), both of which happens frequently during machine translation thus showcasing the benefit of our multi-task approach. Our model very often makes the output sentence more complete (and thereby more formal) by inserting pronouns like 'it', 'they' at the start of the sentence or by removing conjunctions like 'usually', 'and', 'but', 'however' from the beginning of a sentence (sample three in 3a). Likewise, in the F→I task, our model produces more informal sentences compared to the baseline by introducing more phrasal level changes (first and second sample in 3b).

Error analysis: In the I→F task, our model performs worse than the baseline when the original informal sentence consists of all uppercased words (fourth sample in 3a). This is primarily because the baseline model pre-lowercases them using rules. Whereas, we rely on the model to learn this transformation and so it fails to do so for less frequent words. In the F→I task, in trying to produce more informal outputs, our model sometimes fails to preserve the original meaning of the sentence (third sample in 3b). In both tasks, very often our model fails to make transformations for some pairs like ('girls','women'), which the baseline model is very good at. We hypothesize that this could be because for these pairs, human rewriters do not always agree on one of the words in the pair being more informal/formal. This makes our model more conservative in making changes because our bi-directional model combines FT data from both directions and when the original data contains instances where these words are not changed, we double that and learn to copy the word more often than change it.

7 Formality-Sensitive Machine Translation Experiments

7.1 Models

NMT-constraint: We first evaluate the standard NMT model with side constraints introduced in Section 3.2 and then compare it with three variants of FSMT models using multi-task learning as described in Section 3.3 (i.e. **MultiTask-tag-style**, **MultiTask-style** and **MultiTask-random**). The best performing system for FT is MultiTask-tag-style with $12n$ ($\sim$2.5M) bilingual data. For fair comparison, we select this size of bilingual data for all FSMT models either by data selection or randomly.

PBMT-random: We also compare our models with the PBMT-based FSMT system proposed by Niu et al. (2017). Instead of tagging sentences in a binary fashion, this system scores each sentence using a lexical formality model. It requests a desired formality score for translation output and re-ranks n-best

Model	+Tag?	Random?	FR→Formal-EN	FR→Informal-EN
NMT-constraint	✓		27.15	26.70
NMT MultiTask-tag-style	✓		25.02	25.20
NMT MultiTask-style			23.25	23.41
NMT MultiTask-random		✓	25.24	25.14
PBMT-random (Niu et al.)		✓	29.12	29.02

Table 4: BLEU scores of various FSMT models. "+Tag" indicates using formality tags for bilingual data while "Random" indicates using randomly selected bilingual data.

translation hypotheses by their closeness to the desired formality level. We adapt this system to our evaluation scenario — we calculate median scores for informal and formal data (i.e. -0.41 and -0.27 respectively) in GYAFC respectively by a PCA-LSA-based formality model (Niu and Carpuat, 2017; Niu et al., 2017) and use them as desired formality levels.[6] The bilingual training data is randomly selected.

7.2 Results

Automatic Evaluation. We compute BLEU scores on the held out test set for all models as a sanity check on translation quality. Because there is only one reference translation of unknown style for each input sentence, these BLEU scores conflate translation errors and stylistic mismatch, and are therefore not sufficient to evaluate FSMT performance. We include them for completeness here, as indicators of general translation quality, and will rely on human evaluation as primary evaluation method. As can be seen in Table 4, changing the formality level for a given system yields only small differences in BLEU. Based on BLEU scores, we select MultiTask-random as the representative of multi-task FSMT and compare it with NMT-constraint and PBMT-random during our human evaluation.

Human Evaluation. Table 2 shows that neural models control formality significantly better than PBMT-random (0.35/0.32 vs. 0.05). They also introduce more changes in translation: with NMT models, $\sim$80% of outputs change when only the input formality changes, while that is only the case for $\sim$30% of outputs with PBMT-random. Among neural models, MultiTask-random and NMT-constraint have similar quality in controlling output formality (0.32 vs. 0.35) and preserving meaning (2.90 vs. 2.95). They are also equally fluent as judged by humans. Interestingly, multi-task learning helps MultiTask-random perform as well as NMT-constraint with simpler examples that do not require the additional step of data selection to generate formality tags.

7.3 Qualitative Analysis

We randomly sample 100 examples from our test set and manually compare the formal and the informal translations of the French source by MultiTask-random, NMT-constraint and PBMT-random. Table 5 shows representative examples of the observed trends.

We find that in most cases, the difference between the formal and informal style translations is very minor in PBMT-random model, better in NMT-constraint model and the best in our MultiTask-random model (first sample in the table). In general, our MultiTask-random model does a good job of making very large changes while transferring the style, especially into informal (second sample in the table). We hypothesize that this is because our joint model is trained on the GYAFC corpus which consists of parallel sentences that differ heavily in style.

Error analysis: All FSMT models perform well in terms of meaning preservation, yet the human scores are not perfect (Table 2). They occasionally change not only the style but also the meaning of the input (e.g. the third sample of MultiTask-random in Table 5). This motivates future work that penalizes meaning changes more explicitly during training. In general, none of the models do a good job of changing the style when the source sentence is not skewed in one style. For example, consider the French sentence "Combien de fois vous l'ai-je dit?" and its English reference translation "How many

[6]The PCA-LSA-based formality model achieves lowest root-mean-square error on a scoring task of sentential formality as listed on https://github.com/xingniu/computational-stylistic-variations.

1	French Source	Impossible d'avoir accès à internet ici.
	English Reference	I don't know if you've tried yet, but it's impossible to get on the internet up here.
	MultiTask-random Formal	It is impossible to have access to the internet here.
	NMT-constraint Formal	It's impossible to have access to the Internet here.
	PBMT-random Formal	I can't access to the internet here.
	MultiTask-random Informal	Impossible to get to the internet here.
	NMT-constraint Informal	Couldn't have accessed the internet here.
	PBMT-random Informal	I can't access to the internet here.
2	French Source	Abstenez-vous de tout commentaire et répondez à la question, chef Toohey.
	English Reference	Refrain from the commentary and respond to the question, Chief Toohey.
	MultiTask-random Formal	You need to be quiet and answer the question, Chief Toohey.
	NMT-constraint Formal	Please refrain from any comment and answer the question, Chief Toohey.
	PBMT-random Formal	Please refrain from comment and just answer the question, the Tooheys's boss.
	MultiTask-random Informal	Shut up and answer the question, Chief Toohey.
	NMT-constraint Informal	Please refrain from comment and answer the question, chief Toohey.
	PBMT-random Informal	Please refrain from comment and answer my question, Tooheys's boss.
3	French Source	Essaie de présenter des requêtes supplémentaires dès que tu peux.
	English Reference	Try to file any additional motions as soon as you can.
	MultiTask-random Formal	You should try to introduce the sharks as soon as you can.
	NMT-constraint Formal	Try to present additional requests as soon as you can.
	PBMT-random Formal	Try to introduce any additional requests as soon as you can.
	MultiTask-random Informal	Try to introduce sharks as soon as you can.
	NMT-constraint Informal	Try to introduce extra requests as soon as you can.
	PBMT-random Informal	Try to introduce any additional requests as soon as you can.

Table 5: Sample model outputs for the Formality-Sensitive Machine Translation (FSMT) task.

times have I told you, right?". All models produce the same translation "How many times did I tell you?". In such cases, changing style requires heavier editing or paraphrasing of the source sentence that our current models are unable to produce.

8 Conclusion

We explored the use of multi-task learning to jointly perform monolingual FT and bilingual FSMT. Using French-English translation and English style transfer data, we showed that the joint model is able to learn from both style transfer parallel examples and translation parallel examples. On the FT task, the joint model significantly improves the quality of transfer between formal and informal styles in both directions, compared to prior work (Rao and Tetreault, 2018). The joint model interestingly also learns to perform FSMT without being explicitly trained on style-annotated translation examples. On the FSMT task, our model outperforms previously proposed phrase-based MT model (Niu et al., 2017), and performs on par with a neural model with side-constraints which requires more involved data selection.

These results show the promise of multi-task learning for controlling style in language generation applications. In future work, we plan to investigate other multi-task architectures and objective functions that better capture the desired output properties, in order to help address current weaknesses such as meaning errors revealed by manual analysis.

Acknowledgments

We thank the three anonymous reviewers for their helpful comments and suggestions. We thank Joel Tetreault for useful discussions and for making the GYAFC corpus available, as well as members of the Computational Linguistics and Information Processing (CLIP) lab at University of Maryland for helpful discussions. This work is supported by the Clare Boothe Luce Foundation and by the NSF grant IIS-1618193. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsors.

References

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

Douglas Biber. 2014. Using multi-dimensional analysis to explore cross-linguistic universals of register variation. *Languages in contrast*, 14(1):7–34.

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

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of domain adaptation methods for neural machine translation. In *ACL*, pages 385–391.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*, volume 307 of *ACM International Conference Proceeding Series*, pages 160–167.

William Coster and David Kauchak. 2011. Learning to simplify sentences using wikipedia. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 1–9.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *AAAI*.

Francis Heylighen and Jean-Marc Dewaele. 1999. Formality of language: definition, measurement and behavioral determinants. *Interner Bericht, Center Leo Apostel, Vrije Universiteit Brüssel*.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *CoRR*, abs/1712.05690.

Eduard Hovy. 1987. Generating natural language under pragmatic constraints. *Journal of Pragmatics*, 11(6):689–719.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Eric Nyberg. 2017. Shakespearizing modern language using copy-enriched sequence to sequence models. In *Proceedings of the Workshop on Stylistic Variation*, pages 10–19.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *TACL*, 5:339–351.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *ACL (System Demonstrations)*, pages 67–72.

Catherine Kobus, Josep Maria Crego, and Jean Senellart. 2017. Domain control for neural machine translation. In *RANLP*, pages 372–378.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*, pages 388–395.

Will Lewis, Christian Federmann, and Ying Xin. 2015. Applying cross-entropy difference for selecting parallel training data from publicly available sources for conversational machine translation. In *IWSLT*.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *LREC*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *HLT-NAACL*, pages 912–921.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *ICLR*.

Paul Michel and Graham Neubig. 2018. Extreme adaptation for personalized neural machine translation. In *ACL*.

Hideki Mima, Osamu Furuse, and Hitoshi Iida. 1997. Improving performance of transfer-driven machine translation with extra-linguistic informatioon from context, situation and environment. In *IJCAI (2)*, pages 983–989.

Robert C. Moore and William D. Lewis. 2010. Intelligent selection of language model training data. In *ACL (Short Papers)*, pages 220–224.

Jonas Mueller, David K. Gifford, and Tommi S. Jaakkola. 2017. Sequence to better sequence: Continuous revision of combinatorial structures. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 2536–2544.

Xing Niu and Marine Carpuat. 2016. The UMD Machine Translation Systems at IWSLT 2016: English-to-French Translation of Speech Transcripts. In *IWSLT*.

Xing Niu and Marine Carpuat. 2017. Discovering stylistic variations in distributional vector space models via lexical paraphrases. In *Proceedings of the Workshop on Stylistic Variation*, pages 20–27.

Xing Niu, Marianna J. Martindale, and Marine Carpuat. 2017. A study of style in machine translation: Controlling the formality of machine translation output. In *EMNLP*, pages 2814–2819.

Xing Niu, Michael Denkowski, and Marine Carpuat. 2018. Bi-directional neural machine translation with synthetic parallel data. In *NMT@ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. In *ACL*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *EACL*, pages 157–163.

Sudha Rao and Joel R. Tetreault. 2018. Dear sir or madam, may I introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality style transfer. In *NAACL-HLT*, pages 129–140.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Controlling politeness in neural machine translation via side constraints. In *HLT-NAACL*, pages 35–40.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *ACL*.

Fadi Abu Sheikha and Diana Inkpen. 2011. Generation of formal and informal sentences. In *ENLG*, pages 187–193.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *NIPS*, pages 6833–6844.

Yogarshi Vyas, Xing Niu, and Marine Carpuat. 2018. Identifying semantic divergences in parallel text without annotations. In *NAACL-HLT*, pages 1503–1515.

Tong Wang, Ping Chen, John Rochford, and Jipeng Qiang. 2016. Text simplification using neural machine translation. In *AAAI*, pages 4270–4271.

Shuly Wintner, Shachar Mirkin, Lucia Specia, Ella Rabinovich, and Raj Nath Patel. 2017. Personalized machine translation: Preserving original author traits. In *EACL*, pages 1074–1084.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *ACL*, pages 1015–1024.

Wei Xu, Alan Ritter, Bill Dolan, Ralph Grishman, and Colin Cherry. 2012. Paraphrasing for style. In *COLING*, pages 2899–2914.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *TACL*, 4:401–415.

Hayahide Yamagishi, Shin Kanouchi, Takayuki Sato, and Mamoru Komachi. 2016. Controlling the voice of a sentence in japanese-to-english neural machine translation. In *WAT@COLING*, pages 203–210.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *EMNLP*, pages 584–594.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *COLING*, pages 1353–1361.

Appendix A. Details of Human-Based Evaluations

As described in the main paper, we assess the model outputs on three criteria of formality, fluency and meaning preservation. We collect these judgments using CrowdFlower.

Since we want native English speakers to perform this task, we restrict our set of annotators only to these three native English speaking countries: United States, United Kingdom and Australia. We create a sample of 51 gold questions for each of the three tasks (criteria). Annotators have to continually maintain an accuracy of above 70% to be able to contribute to the task.

We collect judgments on 300 samples of each model output and we collect three judgments per sample (i.e. sentence pair). Given the three judgments per sample, we calculate the aggregate score using the weighted average:

$$\frac{\sum_{i=1}^{3} score_i \times trust_i}{\sum_{i=1}^{3} trust_i}$$

where $score_i$ is the score given by an annotator and $trust_i$ is our trust on that annotator. This trust is the accuracy of the annotator on the gold questions.

Formality: Given two sentences, we ask workers to compare their formality using one of the following categories, regardless of fluency and meaning. We do not enumerate specific rules (e.g. typos or contractions) and encourage workers to use their own judgment.

Score	Category
2	*Sentence 1 is much more formal than Sentence 2*
1	*Sentence 1 is more formal than Sentence 2*
0	*No difference or hard to say*
-1	*Sentence 2 is more formal than Sentence 1*
-2	*Sentence 2 is much more formal than Sentence 1*

Fluency: Given two sentences, we ask workers to compare their fluency using one of the following categories, regardless of style and meaning. We define fluency as follows: *A sentence is fluent if it has a meaning and is coherent and grammatical well-formed.*

Score	Category
2	*Sentence 1 is much more fluent than Sentence 2*
1	*Sentence 1 is more fluent than Sentence 2*
0	*No difference or hard to say*
-1	*Sentence 2 is more fluent than Sentence 1*
-2	*Sentence 2 is much more fluent than Sentence 1*

Meaning preservation: Given two sentences, we ask workers to answer "how much of the first sentence's meaning is preserved in the second sentence", regardless of style.

Score	Category
3	*Equivalent since they convey the same key idea*
2	*Mostly equivalent since they convey the same key idea but differ in some unimportant details*
1	*Roughly equivalent since they share some ideas but differ in important details*
0	*Not equivalent since they convey different ideas*

While collecting formality and fluency annotations for sentence pairs, to avoid system-level bias, we randomly swap the two items in the pair and collect annotations on a symmetric range of [-2,2]. But while aggregating these scores, we recover the order and hence the final scores in Table 2 are only in the range of [0,2].

Towards a Language for Natural Language Treebank Transductions

Carlos A. Prolo
Department of Informatics and Applied Mathematics - DIMAp
Federal University of Rio Grande do Norte - UFRN
Natal, RN, 59078-970, Brazil
prolo@dimap.ufrn.br

Abstract

This paper describes a transduction language suitable for natural language treebank transformations and motivates its application to tasks that have been used and described in the literature. The language, which is the basis for a tree transduction tool allows for clean, precise and concise description of what has been very confusingly, ambiguously, and incompletely textually described in the literature also allowing easy non-hard-coded implementation. We also aim at getting feedback from the NLP community to eventually converge to a *de facto* standard for such transduction language.

1 Introduction

Linguists have always liked to use trees to theorize about the structure of natural language sentences (Marcus et al., 1983). In the 1990's, computer scientists also started to grow excited with the possibility of getting those proposed structures reasonably accurately from a computer. The advent of large corpora with annotated syntactic structures, the treebanks, among which the Penn Treebank (PTB) (Marcus et al., 1993; Marcus et al., 1994) is a notable representative, made it possible to build statistical parsers with an accuracy that was not feasible before. From the early work of (Magerman, 1994; Magerman, 1995; Charniak, 1997; Collins, 1997), state-of-the-art accuracy has been progressively raising, achieving now scores above 92% in the English Penn Treebank, as reported, for instance, in (Shindo et al., 2012).

As for dependency representations (Nivre et al., 2016; Nivre and Fang, 2017), although they capture syntactic structure in a different way, they are still generally represented as trees, even if not necessarily caring for the order among the children of a node.

Of course, not everybody is expected to agree on any given tree-structure style. On the linguistic side, each practitioner has their own theoretical conceptions on what is an adequate syntactic structure for a sentence, which may in turn differ from that of a specific treebank. Often all the information that is needed to promote the intended change is already present in the tree, either with some structure different than the desired, or "latent" in the annotation (Chiang and Bikel, 2002), in the form of secondary attributes on the nodes. When the change to the desired structure can be mechanically accomplished by applying to the trees a precisely defined rule, we call it a transduction. In order to perform more complex transformations we may instead define sequences of simpler transductions to be applied as a pipeline.

A few tree transducers have been proposed in the literature, among them the ones described in (Chiang and Bikel, 2002), based on context-free style rules, (Blahetta, 2003), based on Richard Pitto's **TGrep** (see (Rohde, 2005)), **Tsurgeon** (Levy and Andrew, 2006) and **TTT** (Purtee and Schubert, 2012). There are also transducers for other purposes, perhaps the currently most well known of them being W3C's **XPath** (World Wide Web Consortium, 2017), but those do not meet the needs of natural language and computational linguists practitioners (see for instance (Purtee and Schubert, 2012) for an initial discussion.) Although also built from Pitto-Rohde **TGrep** tools, the transduction language we propose in this paper is substantially diverse from that described in Blahetta's thesis (which, according to (Levy and Andrew, 2006) is no longer available).

Proceedings of the 27th International Conference on Computational Linguistics, pages 1022–1032
Santa Fe, New Mexico, USA, August 20-26, 2018.

Our main, initial motivation for designing the language and the tool we describe here came from perceiving the impact tree transduction can have in natural language parsing technology. There has been many really interesting proposals for parsers over these past 20 to 30 years, but what is even more striking is that they arose side by side with an intensive practice of what is often referred to as *hacking the trees*, which is now known to have played an important role in their success. For example, since his early work, Collins realized that it would be important, as a statistical parameter for deciding the correct tree structure of a sentence, to learn about the boundaries of *non-recursive* noun phrases, or *base NPs*, in the sentence.[1] However, his statistical parsing models would not be able to learn that particular parameter directly from the treebank, because base NPs were not explicitly marked as constituents in the tree, even if all necessary information was there: a human or a computer can easily follow some definition rules for base NPs, with no other external knowledge, and tell where they are in the annotated sentences. So, given a Penn Treebank tree like in Figure 1 the base non-recursive NPs, according to Collins, would be the following spans:

- (NP (DT A) (NNP SEC) (NN proposal) (S ...))
- (NP (NN reporting) (NNS requirements))
- (NP (DT some) (NN company) (NNS executives))
- (NP (DT the) (NN usefulness))
- (NP (NN information))
- (NP (NN insider) (NNS trades))
- (NP (DT a) (JJ stock-picking) (NN tool))

Collins realized that, inserting a node with a new label he called NPB to delimit the span, those constituents would help his training process so as to increase by a statistically significant amount his parser accuracy. After the transductions, the trees would be as following. Notice that whenever the entire NP span is itself immediately under some other NP, instead of adding a fresh NPB, the lower NP that dominates the spam is replaced with an NPB. That is the case in the first, third and the last two NPs below.

- (NP (NPB (DT A) (NNP SEC) (NN proposal) (S ...)))
- (NPB (NN reporting) (NNS requirements))
- (NP (NPB (DT some) (NN company) (NNS executives)))
- (NPB (DT the) (NN usefulness))
- (NPB (NN information))
- (NP (NPB (NN insider) (NNS trades)))
- (NP (NPB (DT a) (JJ stock-picking) (NN tool)))

The particular *tricks* used by Collins were reported in (Bikel, 2004). In fact, what once could have been perceived by some as a trick, is increasingly being acknowledged as an opportunity to recover relevant latent information, in principled ways, and making it available to the parser's training model.

Having this in mind we designed a general transduction language, based on Richard Pitto's **TGrep**.[2] We named it **tsed** after Unix **sed**, the string transduction tool counterpart to **grep**.

The obvious test suite was the set of Collins transformations as very precisely described (although in plain English, not in a formal language) in (Bikel, 2004), and also in Section 3.

Most tree transformations used in parsing have been hard-coded into the training and parsing algorithms. In particular, that was the case in the Collins implementation and even in Bikel's version. Although the latter claims to be highly parameterizable, which is indeed true to a great extent, one still has to hard-code in a programming language all the tree transformation processes.

In the following sections we describe the transduction language, and validate its expressive power replicating Collins's transformations as a test case. Of course the tool can be used for many other purposes as already mentioned. We conclude and discuss what is still planned for the future.

2 The Language

We describe here the transduction language. Appendix A contains a concise grammar for the language.

[1] See, for instance, (Collins, 2003) for his particular precise definition of the term.

[2] **TGrep** stands for *Tree Grep*, for the fact that it searches trees in a away which is parallel to the way Unix **grep** searches strings, and is currently available through Douglas Rohde's improved **TGrep2** version (Rohde, 2005).

```
(S (NP-SBJ (DT A) (NNP SEC) (NN proposal)
     (S
       (NP-SBJ (-NONE- *) )
       (VP (TO to)
           (VP (VB ease)
               (NP
                 (NP (NN reporting) (NNS requirements) )
                 (PP (IN for)
                     (NP (DT some)(NN company)
                         (NNS executives))))))))
   (VP (MD would)
       (VP (VB undermine)
           (NP
             (NP (DT the) (NN usefulness) )
             (PP (IN of)
                 (NP
                   (NP (NN information) )
                   (PP (IN on)
                       (NP (NN insider) (NNS trades) ))))
           (PP (IN as)
               (NP (DT a)(JJ stock-picking)(NN tool))))))
   (. .)  )
```

Figure 1: A Penn Treebank tree

2.1 Overview

A transduction is a pair (s, r) (or $s ==> r$ in the syntax of the language) where s is a *search expression* and r is a *replacement expression*. Together they define how to transform a treebank input tree into a new one.[3] The search expression describes what we look for in the input trees, and the replacement expression defines how to build the output tree based on the patterns found during the search phase.

The search expression is based on restrictions to be met by the input tree in order for a match to succeed. There are two kinds of patterns in the search expression: the *node specifications*, intended to match with nodes of the input tree that comply with the pattern; and the *operators* that relate the node patterns defining a layer of restrictions that has to be respected by the nodes. So, a search expression such as $NP < PP$ will look for a node with label "NP", that immediately dominates another node, labeled "PP". This syntax is strongly based on **Tgrep2**.

However, while **Tgrep** is designed only to find patterns, **tsed** has to handle transformations, so we added the concept of *placeholders*, in a somewhat similar way as Unix **grep** was extended into **sed**. When a transduction is applied to an input tree, if the search is successful we say that there is a *match*, and as a byproduct each placeholder will be assigned a subtree of the input. Of course, there can generally be several ways a match can succeed, leading to different assignments of placeholders into subtrees. For instance, in the tree of Figure 1 we can find three NPs that immediately dominate a PP and also the second of this NPs dominates the third one. Indeed it is a very important issue the precise semantics that govern the specifics of search order and disambiguation. This is the subject of a later section of this paper.

The *main placeholder*, marked by enclosing the node specifier with square brackets, also defines the point where the transformation will take place. So a transduction such as $[NP] < PP ==> dog$ will succeed in a tree that has an NP immediately dominating a PP. The match will assign to the main placeholder [] the subtree for the NP found. And the transformation phase will replace the whole subtree of the NP, by the single node "dog". We will see ahead how to express more complex replacement patterns. For now, we just want to note that if the pattern was $NP < [PP] ==> dog$, then the substitution would take place at the PP subtree and the NP would be preserved in the tree as well as all other trees sibling to the PP.

[3]The transducer tool assumes boot input and output trees in the Penn Treebank format - label bracketed, constituent trees representation format, - and this format is used in the examples throughout the paper, though, of course, this format is not a relevant conceptual issue.

2.2 Search Expression

The search expression is in fact a *primary node specification* and an optional Boolean layer of restrictions. This is very much imported from **TGrep2**, with minor changes. So, in the search expression $[NP](< NN| < NNP)\&\$.PP$, "|" means disjunction (logical "or"), "&" means conjunction (logical "and"), "<" stands for immediate domination, and "$." stands for the first node having as an immediate sibling the node to the right. Then the expression instantiates the placeholder [] to a subtree labeled NP that either immediately dominates (<) an NN or that immediately dominates an NNP, and, additionally, the NP has to have as an immediate sibling of its parent a node labeled PP.

The search expression allows embedded restrictions, so $[NP](< NN| < NNP)\&\$.(PP < (IN < of))$, forces that additionally to the restrictions above, the PP (not the NP) should immediately dominate a node labeled IN (the preposition in the English Treebank), and that preposition has to be "of". The reader familiar with **TGrep2** might notice that we used the parenthesis both to disambiguate precedence among the restrictions and for embedded restrictions.[4]

2.3 Place Holders

There are two types of placeholder: *cut* and *copy*. Cut placeholders are marked by enclosing the node specifier with square brackets in the search expression. Once they match and are assigned to a certain subtree, that subtree will be excised from its parent during the transformation, and will not be at its original place anymore in the output tree, unless explicitly reinserted according to the replacement expression. The copy placeholder uses curly brackets ($\{\}$) instead of square brackets. The subtree assigned to it can be copied elsewhere in the output tree, as specified by the replacement expression, but will not be removed from its original place. Copy and cut placeholders are numbered. The main or primary placeholder presented in the previous sections is assigned number 0 by default.

Currently **tsed** is centered in the replacement of the hole which is left in the original tree when the the subtree assigned to the main placeholder is excised. The other cut placeholders are not replaced. An example is given right below.

Advancing a little in the details of the replacement expression, the transduction $[NP] < [1 : PUNCT] \Rightarrow (NP\ [NP]\ [1 : PUNCT])$ helps us exemplify the concept of the cut placeholder. It assigns to [] (same as [0 :], the default). a subtree labeled NP that dominates a node PUNCT. This node PUNCT is assigned to the the other cut placeholder, with number 1 ([1 :]). The output tree will be formed replacing the whole NP tree (placeholder [] or [0 :]), by a new tree defined by the pattern in the replacement expression as a fresh NP, that has two children: the NP subtree matched to $[NP]$, and the subtree matched to $[1 : PUNCT]$. Moreover, the PUNCT subtree will be removed from below the old NP. The reader may have recognized this as a very simple mechanism for raising a node in the tree. If the expression were $[NP] < \{1 : PUNCT\} \Rightarrow (NP\ [NP]\ \{1 : PUNCT\}\}$, then the PUNCT subtree would not be excised from its original place and the effect would be to have two copies of it in the output tree.

At this point the reader should be wondering that the reference to the placeholders could be shorted, and indeed the above expressions are equivalent to $[NP] < [1 : PUNCT] \Rightarrow (NP\ []\ [1 :])$ and $[NP] < \{1 : PUNCT\} \Rightarrow (NP\ []\ \{1 :\})$. We call this shortened representations such as [], $\{1 :\}$ *back references*.

2.4 Replacement Expression

The replacement expression is a sequence of trees. Each tree has the same syntax as the PTB representation of a tree, except that we can use the placeholders instead of labels. In this representation, a leaf node is represented just by its label, such as "NN". A subtree which is not a leaf is represented, recursively by the sequence, enclosed in parenthesis, formed by the label of the subtree root followed by the representation of the subtrees of the root node from left to right. Thus a replacement expression such as

[4] In **TGrep2** embedded restrictions use square brackets instead. This change makes things more uniform, preserves the brackets for placeholders, and does not lead to ambiguity in interpretation. We also force explicit inclusion of & for conjunction which makes the search expression more readable, though, well, for the sake of compatibility with TGrep we allow dropping &.

$(PP\ (IN\ from)\ (NP\ (NNP\ Santa)(NNP\ Fe)))$ represents the usual English PTB representation for the prepositional phrase "from Santa Fe." Now, the reader can go back to the transductions of the previous subsection to recognize that this interpretation has been applied there, for instance to $(NP\ []\ [1:])$ as a subtree rooted at an NP, with two children subtrees. That is the main application for the placeholders and the back references.

The possibility of having a sequence of trees instead of just one, fits nicely the rest of the language. So if one wants to flatten a little more the English Penn Treebank, transductions such as $[NP] < (\{1 : NP\}\ \$.\ \{2 : PP\}) \Rightarrow \{1:\}\ \{2:\}$ replaces the NP subtrees that stand for an NP modified by a PP (the lower NP has an immediate right sibling PP) by a flattened one with the higher NP eliminated.[5] [6]

2.5 Restrictions

We have imported exactly the same very rich set of restriction operators defined in **TGrep2**. These are called "links"in the **Tgrep2** documentation. As for Boolean operators, we took the hard decision of only allowing negation as part of the restriction operators. So $! <$ means "do not immediately dominate, as in **TGrep2**. But using the negation higher in the search expression would be pretty confusing and non-intuitive to interpret in the transduction. [7]

2.6 Node Specification

Node specification in the transduction was one of the hardest issues in the design of **tsed**. Due the characteristics of treebank labels, we felt it would be desirable to have both pattern matching internal to the node label and some amount of editing capability.

In the examples so far we defined a node in the search expression as a constant string possibly enclosed in brackets or braces. This is very limited in many senses and we now extend the syntax in a few ways.

First, the part inside the brackets/braces is allowed to be edited in the replacement expression. Moreover fixed context patterns can be provided outside the brackets/braces, both to the left and to the right.

A node specification is then a triple (l, m, r) where m stands for middle and is the part enclosed in the brackets or braces; l is the left context and r is the right context. So the transduction. $[NP] - TMP \Rightarrow [NPT]$ looks for a node with label NP-TMP, and changes the NP portion of the label to [NPT], keeping the left and right context, therefore resulting in a label $NPT - TMP$. Notice that the left context in the example is empty, so the match will only occur for labels that start with the NP sequence of characters. Moreover, the matched label for the example can not have anything beyond the TMP suffix. In this sense, up to this point, label matching has to be exact given the pattern.

In each of the three parts a "*" matches with any character sequence and "?" is a wild card, like in file specifications in Linux. This is a pretty easy and powerful resource. Then $[NP] * -TMP* \Rightarrow [NPT]$ will replace all NPs that contain the feature TMP to NPT (maintaining the context). So an $NP - SBJ - TMP$ would be replaced to $NPT - SBJ - TMP$.

Escaped characters are allowed as well as substrings enclosed either in single or double quotes.

Finally, POSIX regular expressions are allowed protected by "//" as in **sed**. This provides a very great flexibility to node specification for the experienced user. We only should enforce as we said in the last paragraph, that that behavior is not like in **grep** or **TGrep2**, that looks for the pattern anywhere in the node. Instead it requires full match. So, a node specifier to match any node with the TMP feature should be expressed as something like $*/ - TMP/*$ or using only POSIX regular expression syntax $/.*-TMP.*/$, but not just $/-TMP/$. The interesting point is that all the node-specifier expressions is ultimately converted to a POSIX regular expression, the semantics of which is widely known and very well documented. In the conversion, the separation of the left, middle and right part of the labels are provided by inserting parenthesis, supported by POSIX regex functions that then automatically locate them in the labels of the input tree.

[5] In fact a slight more complex expression should be used since this one is ignoring that extra material on the right of the PP or on the left of the lower NP would be lost.

[6] It is just an example, we are not advocating flattening the PTB!

[7] This was one of the few points where we found important to restrict the language in favor of usability and avoiding unpredictable behavior.

2.7 A Precise Semantics for the Transduction Driver

Avoiding non-deterministic behavior is a crucial issue in the project of artificial languages, often overlooked in its importance by people unfamiliar with language design and implementation. Indeed, it is an essential part of a language definition to provide clear rules specifying exactly what will be the output given any context of application of a valid transduction specification to a valid input tree.

For transduction languages based on patter matching it is essential to state clearly how the driver that executes the transformation works. For instance if we ask to eliminate a subtree labeled A that dominates a B, suppose that there happens to be in the corpus a tree with a path where an A immediately dominates a lower A, and the lower A immediately dominates a B. The question is whether the lower or higher subtree labeled A will be removed. Another issue is that of what to do after a match is performed: should it move on to the next tree, or should it try to reapply the transduction again to the same tree? And in the latter case, should it start it at the beginning of the output tree of the previous application? This issues could be treated as reasonable divergence among designers with regards to preference of behavior. However, it turns out that different behaviors are required depending on each particular situation. This section is intended to make precise the possible behaviors of the driver made available to the user by setting some execution options in the tool.

First, the search expression is seen as if parsed into an abstract, syntax tree like structure (see for example (Aho and Ullman, 1972)) in which operators are internal nodes and operands are their children. The leftmost leaf is the primary node specifier. Then the search is applied against input trees as if reading the syntax tree in an in-order traversal, that is, looking for a node, then looking for the restriction operator, and then to the subexpression to the right of the operator.

The Boolean layer is interpreted as in modern implementation of programming languages where the ”OR” is a ”conditional or” and the ”AND” is a ”conditional and”. This is essential to understand how the placeholders are assigned to subtrees. For instance once a full match succeeds on a tree, through a certain branch of an ”OR”, it will not try other paths. Then it will proceed according to the following algorithm:

1. Let *start* be the root of the input tree.

2. Assign *start* to *current node*

3. Search the input tree from the *current node* in a preorder left-to-right traversal order looking for a node that matches the primary node of the search expression (leftmost leaf of the syntax tree).

4. Proceed according to the syntax tree operators. Each operator dictates its one intuitive behavior. In a nutshell it searches from the *current node* ”outwards”in the tree. For example: ”dominates”search the next node in the same way as the primary node; ”is dominated”searches upwards till a match is found or it tries to move above the root; search for siblings move from the current node to the endpoints (left or right) In fact each operator has its own semantics which we cannot exhaust here, but the principles stated so far should be enough to understand most if not all of the behavior.

5. Whenever a restriction fails, backtrack. Failing branches in an OR layer remove internal assignments made to placeholders on this branch.

6. If the match fails, output the tree as it is at that moment.

7. If the match succeeds,

   (a) Apply the replacement expression, modifying the current tree.

   (b) Assign to *current node* the node that would follow, in a preorder left-to-right traversal, the one that replaced the primary placeholder after the transformation just concluded. Usually this is the leftmost child of the root of the subtree inserted there, unless these subtree is just a leaf.

   (c) Resume at step 3

The way we defined at 7.(b) the reentrant behavior to resume the process after a successful match transformation seems to be the most desirable from our modeling experience so far. It is a compromise to an alternative behavior of restarting over again from the root, which causes many non-termination problems and unpredictable behavior.

However two alternatives still being considered as options at the command line are: to resume from the leftmost child of the node matching the primary node specifier (which usually is the primary placeholder, though not always); and also resuming from the node which would be the next, in a preorder left-to-right traversal, after the whole subtree substituted at primary placeholder was visited. This latter behavior is extremely safe in that it would completely prevent non-termination. And it is certainly faster. But it generally does not correspond to the user's desired behavior.

2.8 Back references

Back references are essential at the replacement expression, but they also have an important use in the search expression. Because we require back references to be previously declared and have strict rules of scope of visibility enforcing the well-formedness of the search expression we do not experience the strange behaviors reported in (Rohde, 2005) by the expression "crossing link".

2.9 Ranges of subtrees and endmarkers

There are situations where being able to express tree-range fillers may be very useful. Tree-range fillers are represented by "..." enclosed as placeholders. Thus, the transduction $[NP] < (\{1 : NP\}(\$, \{3 : ...\}\&\$.(\{2 : PP\}\$.\{4 : ...\}))) \Rightarrow \{3 :\}\{1 :\}\{2 :\}\{4 :\}$ is a more robust version of the flattening example above. $\{3 :\}$ matches with the sequence of siblings to the left of the lower NP, and $\{4 :\}$ matches with the sequence of siblings to the right of the PP, thus the extra material below the higher NP is preserved. Of course there are other ways of doing this. However this greatly simplifies the transductions that have to eliminate intermediate nodes.

Another feature is the endmarker which stands for the lack of nodes. So $\# < SBAR$ means an SBAR at the root of the tree. Similarly the endmarker can be used to capture the notions of leaf node and right- and leftmost child.

2.10 "Semantic"consistency of the Boolean layer with respect to placeholders

Every artificial language has those issues related to allowing only well-formed expressions. For instance in **tsed**, we enforce that a placeholder cannot be defined twice in the search expression, and that a back reference can only appear if the corresponding full placeholder has been defined earlier in the expression. We are not going to exhaust these aspects here. However it is important to notice that the definition and use of placeholders interact very dangerously with the logical operators. For instance, a back reference in a restriction in a branch of a logical OR cannot refer to a placeholder defined in an earlier branch. This is not a problem if the restrictions are in the branches of an AND operator. Natural language semanticists will quickly perceive this asymmetry which is generally not an issue in programming language implementation. Another issue is that a placeholder is visible outside of an OR layer of restrictions only if it has been declared in all branches. We have considered very intuitive and indeed interesting rules of scope and visibility for the placeholders but will refrain from entering in a morass of detail that would be required to explain them here.

3 A Case Study

We present in this section examples of transductions that implement Collins pre-processing steps as described in in (Bikel, 2004). All the quotations in this section are from his article, with references to figures removed to avoid confusion.

3.1 Base NP node insertion

This transformation was exemplified earlier in the previous section. Bikel describes it as follows:

An NP is basal when it does not itself dominate an NP; such NP nodes are relabeled NPB. More accurately, an NP is basal when it dominates no other NPs except possessive NPs, where a possessive NP is an NP that dominates POS, the preterminal possessive marker for the Penn Treebank. These possessive NPs are almost always themselves base NPs and are therefore (almost always) relabeled NPB. For consistency's sake, when an NP has been relabeled as NPB, a normal NP node is often inserted as a parent nonterminal. This insertion ensures that NPB nodes are always dominated by NP nodes. The conditions for inserting this "extra" NP level are slightly more detailed than is described in Collins' thesis, however. The extra NP level is added if one of the following conditions holds:

- *The parent of the NPB is not an NP.*
- *The parent of the NPB is an NP but constitutes a coordinated phrase.*
- *The parent of the NPB is an NP but*
  - *the parent's head-child is not the NPB, and*
  - *the parent has not already been relabeled as an NPB.*[8]

The transformation is accomplished by the following **tsed** transduction sequence. We well apply the sequence to the following PTB annotated sentence:

```
(S (NP-SBJ (NP (NP (DT The) (NNP SEC))
               (POS 's))
           (NNP Mr.) (NNP Lane))
   (VP (ADVP-MNR (RB vehemently))
       (VF disputed)
       (NP (DT those) (NN estimates)))
   (. .))
```

1. We first replace NP labels that dominate some POS with a new fresh label POSNP: We find some NP nodes which dominates (operator <<, not necessarily immediate domination) a POS node as stated in the search string. The $*$ means that the labels, either POS or NP, may be followed by any character sequence, so it can match, say, an NP-SBJ. The brackets [] are used as placeholders once a match is found. That means that (1) the subtree rooted at the NP node is the one going to be replaced; (2) the brackets have been matched to that subtree in case we want to refer to that tree in the replacement string; and (3) we have the chance to replace the part of the string inside the brackets in case we want to use it. In this case it selects only the NP part of the label, not the additional suffix. That is because the "*"is outside the brackets. The replacement string tells us then that the same subtree will be kept there (because it puts the brackets back there). But the NP part of the label is replaced with POSNP, so, if that was an NP-SBJ, it will become POSNP-SBJ.

   search string: '[NP]* <<POS*' **replacement string:** '[POSNP]'
   output:

```
(S ( POSNP-SBJ ( POSNP (NP (DT The) (NNP SEC))
                       (POS 's))
               (NNP Mr.) (NNP Lane))
   (VP (ADVP-MNR (RB vehemently))
       (VF disputed)
       (NP (DT those) (NN estimates)))
   (. .))
```

   The search is made in a preorder traversal and finds all matching occurrences, each one being subject to the replacement.

2. In this rule we replace with NPB the labels of NPs that do not dominate another NP (those are the base NPs). The exclamation mark means negation as in **Tgrep**. Notice that we have already preclude the possibility of targeting NPs dominating POS, since we changed their names to POSNP.

   search string: '[NP]* !<<NP*' **replacement string:** '[NPB]'
   output:

```
(S (POSNP-SBJ (POSNP ( NPB (DT The) (NNP SEC))
                       (POS 's))
               (NNP Mr.) (NNP Lane))
   (VP (ADVP-MNR (RB vehemently))
       (VF disputed)
       ( NPB (DT those) (NN estimates)))
   (. .))
```

[8] Only applicable if relabeling of NPs is performed using a preorder traversal.

3. Then we switch the POSNP labels back to NP.

search string: '[POSNP]*' **replacement string** '[NP]'

output:

```
(S ( NP-SBJ ( NP (NPB (DT The) (NNP SEC))
                 (POS 's))
             (NNP Mr.) (NNP Lane))
   (VP (ADVP-MNR (RB vehemently))
       (VF disputed)
       (NPB (DT those) (NN estimates)))
   (. .))
```

4. And finally we add an extra NP node on top of the NPB, in case it is not immediately dominated by some other NP (again following **TGrep**, a single > means that it has to be **immediately** dominated in order to match.

search string '[NPB]* !>NP*' **replacement string** 'NP <[NPB]'

output:

```
(S (NP-SBJ (NP (NPB (DT The) (NNP SEC))
                 (POS 's))
             (NNP Mr.) (NNP Lane))
   (VP (ADVP-MNR (RB vehemently))
       (VF disputed)
       ( NP (NPB (DT those) (NN estimates))))
   (. .))
```

The reverse transformation is described by Bikel as:

> *I*n post-processing, when an NPB is an only child of an NP node, the extra NP level is removed by merging the two nodes into a single NP node, and all remaining NPB nodes are relabeled NP.

This is easily accomplished by a single transduction rule

search string '[NPB]* !>NP*' **replacement string** 'NP <[NPB]'

We have implemented also other transductions from (Bikel, 2004) and the reverse versions whenever well defined. Some of them depend on location the head node on a constituent. That could be accomplished through a pre-processing tool that could mark the head nodes with a "-HEAD"tag. We have considered replicating the head finding rules but we have not done it yet.

4 Conclusion and Future Work

Of course the appeal of tree transductions is not only its use for parsing pre- and post-processing tasks. We used this application in this paper to show that **tsed** language is sufficiently powerful so that it can be used to replace transformations originally made by programming language code pieces, that were never subject to a restriction in power of expression. That is, Collins defined and coded the transformations that he thought as useful and that he could mechanically express having all the Turing machine power of a general programming language at his disposition. Those transformations can be elegantly and succinctly captured by a special purpose transducer language, stripping them out of the main learning algorithm.

By inspection, we extensively observed that the application of all the transductions generated the new trees as we think they should. In fact there is no easy way to directly check against what Collins would generate, because the latter are never explicitly realized. They are internal to the parsing process. We have considered instead, comparing the parsing accuracy of the original parser with that of the parser with the transformations made by **tsed**. Unfortunately this requires removing the corresponding pieces of code in either Collins' or Bikel's implementation, and we have not done it yet.

The examples in the test case have been successfully implemented in an existing implementation that have some limitations in expressive power. Still it could model the rules described in (Bikel, 2004).

Tsed will be used in projects to experiment on parsing accuracy enhancement through sound corpora transformations.

We hope to get feedback from the NLP community joining efforts to eventually converge to a *de facto* standard for a transduction language.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. The Theory of Parsing, Translation, and Compiling, volume I: Parsing. Prentice-Hall, Englewood Cliffs, NJ, USA.

Daniel Bikel. 2004. Intricacies of Collins' parsing model. Computational Linguistics, 30:479–511.

Don Blahetta. 2003. Function Tagging. Ph.D. thesis, Brown University.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In Proceedings of the Fourteenth National Conference on Artificial Intelligence, pages 598–603, Menlo Park, CA, USA.

David Chiang and Daniel M. Bikel. 2002. Recovering latent information in treebanks. In Proceedings of the 19th International Conference on Computational Linguistics (COLING'2002), pages 183–189, Taipei, Taiwan.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, pages 16–23, Madrid, Spain.

Michael Collins. 2003. Head-driven statiscal models for natural language processing. Computational Linguistics, 29(4):589–637.

Roger Levy and Galen Andrew. 2006. Tregex and tsurgeon: tools for querying and manipulating tree data structures. In Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC), Genoa, Italy.

David M. Magerman. 1994. Natural Language Parsing as Statistical Pattern Recognition. Ph.D. thesis, Stanford University.

David M. Magerman. 1995. Statistical decision-tree models for parsing. In Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics, pages 276–283, Cambridge, MA, USA.

Mitchell P. Marcus, Donald Hindle, and Margaret M. Flack. 1983. D-theory: Talking about talking about trees. In Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics, pages 129–136, Cambridge, Massachusetts, USA, June. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. Computational Linguistics, 19(2):313–330.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In Proceedings of the 1994 Human Language Technology Workshop.

Joakim Nivre and Chiao-Ting Fang. 2017. Universal dependency evaluation. In Proceedings of the NoDaLiDa Workshop on Universal Dependencies, UDW@NoDaLiDa 2017, Gothenburg, Sweden, May 22, 2017, pages 86–95.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan T. McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.

Adam Purtee and Lenhart Schubert. 2012. Ttt: A tree transduction language for syntactic and semantic processing. In Proceedings of the EACL 2012 Workshop on Tree Automata Techniques in Natural Language Processing, Avignon, France.

Douglas L. T. Rohde, 2005. TGrep2 User Manual.

Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. 2012. Bayesian symbol-refined tree substitution grammars for syntactic parsing. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 440–448, Jeju, Korea.

World Wide Web Consortium. 2017. Xml path language (xpath) 3.1. `http://www.w3.org/TR/xpath-31/`, March.

Appendix A. A reference grammar for tsed language

A concise Yacc-style grammar can be very helpful as a reference to the language even without any semantic restriction or explanation (a grammar is worth a thousand words!) Also shown are the main lexical items in Lex-style.

```
start:  search_expression  turnsto  replacement_expression

search_expression:  search_primary   opt_restrictions  ;
opt_restrictions: restrictions |  ;
restrictions: restrictions terminal_or restrictions_and
            | restrictions_and   ;
restrictions_and: restrictions_and terminal_and restrictions_not
        | restrictions_and restrictions_not /* same as AND */
        | restrictions_not ;
restrictions_not: terminal_not restrictions_not /* "not" is under scrutiny */
        | lpar restrictions rpar
        | restriction ;  /* break precedence, [] on tgrep*/
restriction:    operator search_second  ;
search_second: lpar search_expression rpar  |  search_primary ;
search_primary: node_specifier | node_range_filler ;
node_specifier: nodename | end_marker ;

replacement_expression: tree_seq;
tree_seq: tree_seq tree  |  tree   ;
tree: node |  // folha
      LPAR node tree_seq RPAR ;
node: new_node |                        /* similar to nodename */
      primary_placeholder_instance |
      auxiliary_placeholder_instance |      ;

/* nodename specification in lex */
  {NODEBLOCK}
  {NODEBLOCK}?"["[[:blank:]]*"]"{NODEBLOCK}?
  {NODEBLOCK}?"{"({NUMBER}\:)?[[:blank:]]*"}"{NODEBLOCK}?
  {NODEBLOCK}?"["({NUMBER}\:)?[[:blank:]]*"]"{NODEBLOCK}?
  {NODEBLOCK}?"["{NODEBLOCK}"]"{NODEBLOCK}?
  {NODEBLOCK}?"{"({NUMBER}\:)?{NODEBLOCK}"}"{NODEBLOCK}?
  {NODEBLOCK}?"{"({NUMBER}\:)?[[:blank:]]*\.\.\.[[:blank:]]*"}"{NODEBLOCK}?
  {NODEBLOCK}?"["({NUMBER}\:)?[[:blank:]]*\.\.\.[[:blank:]]*"]"{NODEBLOCK}?

/* Helpers */
ALFANUM                 [[:alnum:]-]
WILDCARD                \?
ANYSEQUENCE             \*
ESCAPED                 \\.
DQSTRING                \"([^"\\]|\\.)*\"
SQSTRING                \'([^'\\]|\\.)*\'
REGEX                   \/([^/\\]|\\+[^\\])+\/
NODEBLOCK          ({ALFANUM}|{WILDCARD}|{ANYSEQUENCE}|{ESCAPED}|{DQSTRING}|{SQS
```

Generating Reasonable and Diversified Story Ending Using Sequence to Sequence Model with Adversarial Training

Zhongyang Li, Xiao Ding and Ting Liu*
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{zyli,xding,tliu}@ir.hit.edu.cn

Abstract

Story generation is a challenging problem in artificial intelligence (AI) and has received a lot of interests in the natural language processing (NLP) community. Most previous work tried to solve this problem using *Sequence to Sequence* (Seq2Seq) model trained with *Maximum Likelihood Estimation* (MLE). However, the pure MLE training objective much limits the power of Seq2Seq model in generating high-quality stories. In this paper, we propose using adversarial training augmented Seq2Seq model to generate reasonable and diversified story endings given a story context. Our model includes a generator that defines the policy of generating a story ending, and a discriminator that labels story endings as human-generated or machine-generated. Carefully designed human and automatic evaluation metrics demonstrate that our adversarial training augmented Seq2Seq model can generate more reasonable and diversified story endings compared to purely MLE-trained Seq2Seq model. Moreover, our model achieves better performance on the task of Story Cloze Test with an accuracy of 62.6% compared with state-of-the-art baseline methods.

1 Introduction

The task of story generation was first proposed in the field of artificial intelligence (AI) (Schank and Abelson, 1977). Recently, benefiting from deep learning technology, story generation again received increasing interests in the natural language processing (NLP) community. In this paper, we propose using adversarial training augmented Seq2Seq model to generate story ending given a story context. For example, given the story context "Sara had lost her cat. She was so sad! She put up signs all over the neighborhood. Then a wonderful thing happened.", our goal is to generate a possibly reasonable story ending "Somebody found her cat".

Much previous work in story generation or prediction focused on learning statistical models of event sequences from large-scale text corpora. Chambers and Jurafsky (2008) proposed unsupervised induction of *narrative event chains* from raw newswire texts, with *narrative cloze* as the evaluation metric. However, they utilized a very impoverished representation of events as the form of (*event, dependency*). To overcome the drawback of this event representation, Pichotta and Mooney (2014) presented a script learning approach that employed events with multiple arguments. Pichotta and Mooney (2016a) showed that the LSTM-based event sequence model outperformed previous co-occurrence-based methods for event prediction. However, this line of work build their models based on discrete verbs and tokens, which is far from being a complete sentence or a story.

There have been a number of recent work for story generation focusing on complete sentences. Kiros et al. (2015) described an approach for unsupervised learning of a generic, distributed sentence representations called *skip-thought* vectors. Such vectors can be used to predict neighboring sentences in the context. Pichotta and Mooney (2016b) used Seq2Seq framework directly operating on raw tokens to predict sentences, finding it is roughly comparable with systems operating on structured verb-argument

*Corresponding author: tliu@ir.hit.edu.cn

Proceedings of the 27th International Conference on Computational Linguistics, pages 1033–1043
Santa Fe, New Mexico, USA, August 20-26, 2018.

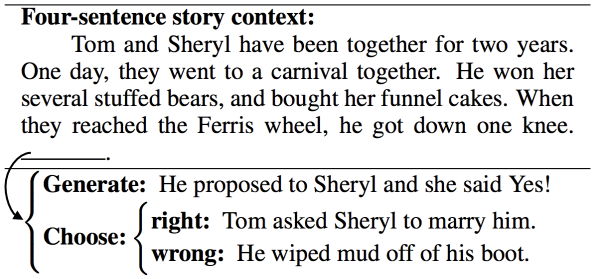


Figure 1: The task of generating or choosing the right story ending.

events in terms of predicting missing events in documents. Hu et al. (2017) also developed an end-to-end model for future subevent prediction. However, they use large-scale news corpus as training data, which is quite noisy and far from being reasonable stories. Moreover, traditional Seq2Seq model are usually trained purely by *Maximum Likelihood Estimation* (MLE). This makes sense in some tasks, such as, machine translation, where the gold-standard target sequence is unique, and the goal is to produce a target sequence as much like the gold-standard one as possible. However, this training objective is not enough for the task of story ending generation. Because this task essentially has no gold-standard answers, and any reasonable story ending can be the right one. On the other hand, purely MLE trained Seq2Seq model tends to generate frequent words or phrases in the test stage, which is a well known intractable obstacle. Hence, it is crucial to explore new ways to enhance Seq2Seq model in the task of story ending generation.

As shown in Figure 1, given a four-sentence story context, our target is to generate a possibly reasonable story ending. To this end, Story ending generation is modeled as a sequence to sequence generation process. In order to understand the context better and generate more reasonable and diversified story endings, we adopt the idea of adversarial training from *Generative Adversarial Nets* (Goodfellow et al., 2014) in recent image generation advances, which includes a generator that defines the probability of generating a story ending, and a discriminator that labels story endings as human-generated or machine-generated. In adversarial training augmented Seq2Seq model, the generator is encouraged to generate story endings that are indistinguishable from human generated story endings, and the discriminator gives a score of judging whether the current story ending is generated by the human or the machine, which can be used as a reward for the generator. Then the generator is trained to maximize the expected reward of the generated story ending using REINFORCE algorithm (Williams, 1992).

We conducted extensive experiments on the ROCStories corpus (Mostafazadeh et al., 2016). Carefully designed human evaluation demonstrates that adversarial training augmented Seq2Seq model can generate logically better story endings than conventional Seq2Seq model. Automatic evaluation metrics also suggest that adversarial training can improve the diversity of the generated story endings. Furthermore, we evaluate the effectiveness of our approach on the task of Story Cloze Test, which requires to choose the right story ending from two candidates given a story context. Our model chooses the right ending based on average word embedding similarities between our generated ending and the two candidates, and achieves the best performance on the task of Story Cloze Test compared to state-of-the-art baseline methods.

2 Sequence to Sequence Learning with Adversarial Training

As shown in Figure 2, our adversarial training augmented Seq2Seq model consists of three components. First, the generator $\mathcal{G}$ defines the policy that generates the story ending Y given the story context X. Second, the discriminator $\mathcal{D}$ is a binary classifier that takes as input a complete story $\{X, Y\}$ and outputs a label indicating whether the input is generated by human-beings or machines. The third component is an adversarial training process between the former two components.

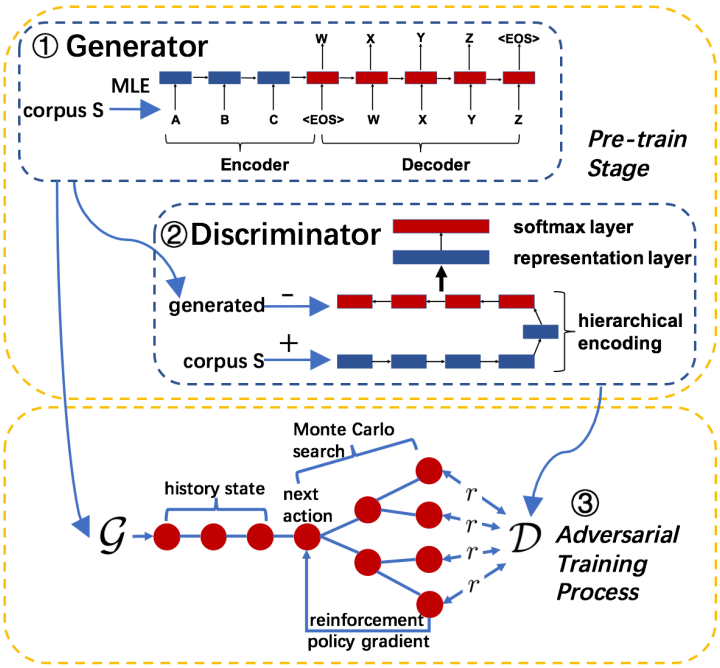


Figure 2: Framework of adversarial training augmented Seq2Seq model.

2.1 Sequence to Sequence Model as the Generator

In our task, the input is a four-sentence story context, and the output is a reasonable story ending. Hence, it is natural to use Seq2Seq model (Sutskever et al., 2014; Cho et al., 2014) as the generator (see ① in Figure 2). In Seq2Seq model, the story context is mapped to a vector representation using a recurrent neural network (Elman, 1990), and then the model computes the probability of generating each token in the target sequence (story ending) using a softmax function.

In recent work on Seq2Seq model, long-short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) (Cho et al., 2014) have shown their power on a wide variety of NLP tasks. Here, we use LSTM as the computation unit in our Seq2Seq generator, and sigmoid is used as the nonlinear activation function. Given a sequence of inputs $X = \{x_1, x_2, ..., x_N\}$, the generator defines a distribution over outputs and sequentially predicts tokens using a softmax function:

$$\begin{aligned} p(Y|X) &= \prod_{t=1}^{T} p(y_t|x_1, x_2, ..., x_N, y_1, y_2, ..., y_{t-1}) \\ &= \prod_{t=1}^{T} \frac{exp(f(h_{t-1}, e_{y_t}))}{\sum_{y'} exp(f(h_{t-1}, e_{y'}))} \end{aligned} \tag{1}$$

where $f(h_{t-1}, e_{y_t})$ denotes the activation function between h_{t-1} and e_{y_t} , and h_{t-1} is the representation output from the LSTM at time $t-1$. Each sentence concludes with a special end-of-sentence symbol *EOS*. Commonly, input and output use different LSTMs with separate compositional parameters to capture different compositional patterns.

In the decoding procedure, the algorithm terminates when an *EOS* token is predicted. Beam search is adopted for next word prediction, the embedding of which is then combined with preceding output for next step token prediction.

2.2 Discriminator

In the idea of adversarial training, we want to produce more natural, diversified and logically reasonable story endings. To this end, we introduce a discriminator (see ② in Figure 2) that judges whether the input is generated by human-beings or machines, and transfers this signal to the Seq2Seq generator, to help adjust its parameters for generating our expected story endings.

The discriminator $\mathcal{D}$ is a binary classifier. We use a hierarchical sequence encoder (Li et al., 2015) to map a complete story $\{X, Y\}$ into a vector representation, which is then fed to a 2-class softmax function, returning the probability of the story being a machine-generated story ending (denoted by $\mathcal{R}_-(\{X, Y\})$) or a human-generated one (denoted by $\mathcal{R}_+(\{X, Y\})$).

2.3 Adversarial Training Process

In the adversarial training augmented Seq2Seq model, the generator is encouraged to generate story endings that are indistinguishable from human generated story endings. We use *policy gradient* methods to achieve such goal (see ③ in Figure 2). The score $\mathcal{R}_+(\{X, Y\})$ of the current generated story ending is assigned by the discriminator. And then it is used as a reward for the generator. At last, the generator (we use θ to denote the parameters of the generator) is trained to maximize the expected reward of generated story ending using the REINFORCE algorithm (Williams, 1992):

$$J(\theta) = \mathbb{E}_{Y \sim p(Y|X)}(\mathcal{R}_+(\{X, Y\})|\theta). \tag{2}$$

We consider the story ending generation procedure as a sequential decision making process. Given the input story context X, the generator generates a story ending Y by sampling from the policy. The concatenation of the generated ending Y and the input X is fed to the discriminator.

Following Yu et al. (2016), we use reward for every intermediate token to update the generator. In our model, Monte Carlo (MC) search is used to assign the reward score for every intermediate token. Given a partially decoded sentence S, the model keeps sampling tokens from the generator until the decoding finishes. Such a process is repeated M times and the M generated sequences will share a common prefix S. These M sequences are fed to the discriminator, the average score of which is used as a reward for S.

For each machine-generated story ending Y, the discriminator gives a classification score $\mathcal{R}(X, Y)$. The gradient of eq. (2) is approximated using the likelihood ratio trick (Glynn, 1990; Williams, 1992), which is used to update the generator:

$$\begin{aligned} \nabla J(\theta) \approx & [\mathcal{R}_+(\{X, Y\}) - b(\{X, Y\})] \\ & \nabla \sum_t \log p(y_t|X, y_{1:t-1}), \end{aligned} \tag{3}$$

where $b(\{X, Y\})$ denotes the baseline value to reduce the variance of the estimate while keeping it unbiased. The discriminator is simultaneously updated with the human generated story ending including story context $\{X, Y\}$ as a positive example, and the machine-generated ending along with story context $\{X, \hat{Y}\}$ as a negative example.

To train the Seq2Seq model in adversarial manner, we need to first pre-train a Seq2Seq model on training corpus $\mathcal{S}$ with MLE. Then pre-train the discriminator use the instance $\{X, Y\}$ in training corpus $\mathcal{S}$ as positive examples, and generated ending along with story context $\{X, \hat{Y}\}$ as negative examples. Subsequently, the generator and discriminator are trained alternatively. During this process, discriminator $\mathcal{D}$ is first trained for *d-steps* with a similar way like pre-train stage. Then generator $\mathcal{G}$ is trained for *g-steps* using the REINFORCE algorithm. At last, we get the adversarially-trained generator $\mathcal{G}$.

3 Experiments

The performance of our adversarially-trained Seq2Seq model is compared with state-of-the-art baselines by evaluating the quality of generated sequences on two tasks: story ending generation and Story Cloze Test (choosing the right story ending from two candidates).

3.1 Dataset

The dataset used in this paper is ROCStories corpus, which is released by (Mostafazadeh et al., 2016). This corpus is a collection of five-sentence commonsense stories by crowd sourcing. Each story has the following major characteristics: (1) is realistic and non-fictional; (2) has a clear beginning and ending where something happens in between; (3) does not include anything irrelevant to the core story. These

	Training	Development	Test
#stories	98,161	1,871	1,871
#words	4,887,555	107,336	107,402

Table 1: Statistics of datasets.

stories are full of stereotypical causal and temporal relations between events, making them a great resource for commonsense reasoning and script knowledge learning. A two-step quality control step makes sure that there are no vague or boundary cases in the test dataset, making human performance of 100% accuracy possible.

The dataset is split into training, development and test datasets. Each instance in the training dataset is a five-sentence story. But in the development and test datasets, each instance is a four-sentence story and two candidate endings (one is the right ending and the other is the wrong ending). Specifically, the wrong endings are purposely designed to fit to the story context but logically wrong. Hence, sampling negative wrong endings from other stories is unreasonable. Detailed statistics of training, development and test datasets are shown in Table 1. All methods are evaluated on the test dataset, and only the development dataset could be used for tuning purposes.

3.2 Evaluation Metrics

For the task of Story Cloze Test, we use accuracy to evaluate the effectiveness of our approach. For the task of story ending generation, we ask human annotators to evaluate the performance of our approach, as it is difficult to find a generally accepted automatic metric for this task. Following Shang et al. (2015), we carefully designed the human evaluation procedure to evaluate the ability of our proposed model on generating story endings. We randomly pick 100 story endings generated by the baseline method and our approach on the test dataset respectively, and distribute them to three annotators. The annotators read the story context, and judge whether the generated story ending is appropriate and reasonable according to the story context. Four levels are assigned to each ending with scores from 0 to 3:

- **Bad (0)**: The ending doesn't make sense and unrelated to the story context.
- **Relevant (+1)**: The ending is partially related to the story context.
- **Good (+2)**: The ending is highly related to the story context.
- **Perfect (+3)**: The ending is high-quality, context-related and logically correct to the story context.

We give the following three aspects judgement criteria for the annotators: (a) **Grammar and Fluency:** Endings should be natural language and free of fluency and grammar errors. (b) **Context Relevance:** Person names, pronouns and phrases in the endings should be relevant to the story context. (c) **Logic Consistency:** Endings should be logically consistent with the story context.

Furthermore, we ask the annotators to directly compare the story endings that generated by the baseline method and our approach, and choose the better one.

We did not use perplexity or BLEU (Papineni et al., 2002) as evaluation metric, as neither of them is likely to be an effective evaluation metric in our scenario. Because our proposed model is designed to steer away from the standard Seq2Seq model, in order to generate more reasonable and diversified story endings. Following Li et al. (2016), we report the degree of diversity by calculating the number of distinct unigrams and bigrams in generated story endings. In order to avoid favoring long sentences, the value is scaled by total number of generated tokens.

3.3 Baselines and Proposed Models

For the evulation of story ending generation quality, the compared models are listed below:

- **Seq2Seq-MLE**: Seq2Seq model purely trained with MLE.
- **Seq2Seq-Adversarial**: Seq2Seq model pre-traind with MLE and then augmented with adversarial training.

	Models	
	Seq2Seq-MLE	**Seq2Seq-Adversarial**
Bad (0)	26.7%	17.7%
Relevant (+1)	23.3%	21.0%
Good (+2)	26.3%	27.0%
Perfect (+3)	23.7%	34.3%
Mean Score	1.47	**1.78 (+21.1%)**
Agreement	0.339	0.344

Table 2: Human evaluation results of story ending generation. Mean score is the average evaluation scores over three annotators.

Both good	6.7%
Both bad	11.3%
Seq2Seq-MLE is better	33.0%
Seq2Seq-Adversarial is better	**49.0**%
Agreement	0.443

Table 3: Pairwise model comparison of story ending generation.

We also evaluate the generated endings in the Story Cloze Test task, compare with the baseline methods used in (Mostafazadeh et al., 2016).

- **Word2Vec**: Choose the ending with closer average word2vec (Mikolov et al., 2013) to the average word2vec of four-sentence context.
- **Skip-thoughts**: (Kiros et al., 2015)'s Sentence2Vec encoder which models the semantic space of sentences, according to which we can choose the ending having a closer embedding to the four-sentence context.
- **Deep Structured Semantic Model (DSSM)**: MSR Sentence2Vector model (Huang et al., 2013), according to which we can choose the ending that has a closer embedding to the context.
- **Conditional Generative Adversarial Networks (CGAN)**: The Conditional GAN model proposed in (Wang et al., 2017), in which the discriminator is used to choose the correct story ending.

We choose the right story ending based on average embedding similarities between endings generated by **Seq2Seq-MLE** and **Seq2Seq-Adversarial** models, and the two candidates.

3.4 Results and Analysis

Human evaluation results of story ending generation are shown in Table 2 and Table 3. From Table 2, we find that Seq2Seq-Adversarial achieves better results than Seq2Seq-MLE model. Seq2Seq-Adversarial achieves a mean score of 1.78, which is 21.1% improvement over Seq2Seq-MLE (1.47). For the **Perfect** and **Good** levels, Seq2Seq-Adversarial outperforms Seq2Seq-MLE. While for the **Relevant** and **Bad** levels, Seq2Seq-Adversarial gets smaller percentage scores. The annotation agreements for Seq2Seq-MLE and Seq2Seq-Adversarial are evaluated by Fleiss' kappa (Fleiss, 1971), as a statistical measure of inter-rater consistency, which are 0.339 and 0.344, respectively. Considering the complex judgement criteria, they can demonstrate high agreement between three annotators (Shang et al., 2015).

As shown in Table 3, we directly compare the performance of Seq2Seq-MLE and Seq2Seq-Adversarial model. There are 49% story endings generated from Seq2Seq-Adversarial model better than that from Seq2Seq-MLE model. In turns, Seq2Seq-MLE model can only achieve better performance than

	UnigramDiv	**BigramDiv**
Seq2Seq-MLE	0.038	0.104
Seq2Seq-Adversarial	0.064 (+68.4%)	0.236 (+126.9%)

Table 4: Diversity evaluation of the generated endings on test dataset. **UnigramDiv** and **BigramDiv** are respectively the number of distinct unigrams and bigrams divided by total number of generated words.

Methods	Accuracy
Word2Vec (Mikolov et al., 2013)	53.9%
Skip-thoughts (Kiros et al., 2015)	55.2%
DSSM (Huang et al., 2013)	58.5%
Seq2Seq-MLE (Cho et al., 2014)	58.6%
CGAN (Wang et al., 2017)	60.9%
Seq2Seq-Adversarial (ours)	**62.6%**

Table 5: Experimental results of Story Cloze Test on test dataset.

Seq2Seq-Adversarial model on 33% story endings. And we get a higher Fleiss' kappa value for pairwise model comparison, which is up to 0.443.

Diversity evaluation results are shown in Table 4. We find that by integrating adversarial training, Seq2Seq model can generate more diversified story endings. Seq2Seq-Adversarial model achieves 68.4% higher unigram diversity and 126.9% higher bigram diversity score than Seq2Seq-MLE model respectively.

The main reasons for these results are as follows. First, MLE tends to generate constant phrases regardless of the story context. Especially when it reads an unfamiliar story context as input, it will generate some frequent phrases appearing in the training dataset, but unrelated to the story context. Second, adversarial training always tries to understand the whole semantics of story context and generates endings as human-beings. Hence, is will generate some context-related or even semantic and logically related endings. Third, adversarial training is dependent on MLE pre-trained Seq2Seq model. Hence, it inherits the advantages of MLE training. During the adversarial training process, it will update its parameters under another criteria (the reward scores given by the discriminator). Hence, adversarial training augmented Seq2Seq model is indeed an ensemble of two different training objectives. MLE training guarantees fluency of the generated endings, and adversarial training adds diversity to them by exploring more words in MC search process.

Results of Story Cloze Test on the ROCStories test dataset are shown in Table 5. The best baseline model is CGAN and it achieves an accuracy of 60.9%. This indicates the difficulty of this task. Seq2Seq-MLE (accuracy: 58.6%) achieves comparable result with the DSSM model (accuracy: 58.5%), and Seq2Seq-Adversarial gets the highest accuracy of 62.6% on the task of Story Cloze Test.

Indeed, all baseline models cannot generate story endings. They either try to directly compare the similarity between story context and two candidate endings or try to map the story context into a hidden vector representation, and then choose a similar ending. While our model try to address the problem of Story Cloze Test from a generation perspective. We first generate real story endings, and then choose the right ending based on the average embedding similarities between them and the candidates. This is easier to understand for readers. Because human readers can read both the generated endings and the candidates, and understand why we can choose the right ending.

Word2Vec method tends to choose the candidate ending whose average word embedding is more similar to the average word embedding of four-sentence context. Comparing with it, our model achieves a huge boost of accuracy (from 53.9% to 62.6%). This is mainly because our model can understand the global semantic of the story context, other than the local meaning of words. Our model achieves better performance than DSSM model, mainly because DSSM model needs to sample negative examples from other stories in its training process, but there are no negative examples in the training dataset. Our model also outperforms the CGAN model, which doesn't generate real story ending but use the discriminator to choose the correct one. This demonstrates the effectiveness of our model, which solve the problem based on real genrated endings.

3.5 Case Study

Three typical cases are shown in Table 6. In case one, both Seq2Seq-MLE and Seq2Seq-Adversarial models generate perfect story endings, that are fluent, grammatical and logically correct. In case two, Seq2Seq-MLE model generate relatively poor ending, which is not relevant to the story context. We argue that in the MLE-training process, it cannot see a similar story context in the training dataset, so

Case one	**Context**	Eric was overly excited for lunch today.The cafeteria was serving his absolute favorite meal.He loved the school's pizza and french fries.He sprinted to the lunch line but it was too late.
	Right	Eric had to settle for cold peanut butter and jelly for lunch.
	Wrong	Eric got a second plate of pizza and french fries.
	Seq2Seq-MLE	eric had to go to the store and buy a pizza instead.
	Seq2Seq-Adversarial	eric decided to go to the store for lunch instead.
Case two	**Context**	Aya wanted to paint a picture.She bought canvas and paints.Then she sat down by a window for inspiration.She began to paint an image of the landscape.
	Right	Aya became famous for her landscape pictures.
	Wrong	Aya put the finishing strokes on a picture of a skyscraper.
	Seq2Seq-MLE	aya was thrilled with her purchase.
	Seq2Seq-Adversarial	by the end of the day she had a picture.
Case three	**Context**	Ivy was scared to go to summer camp.But she steeled herself and got on the bus.When she got there, she went to talk to the other campers.Soon she had made a few new friends.
	Right	Ivy ended up loving summer camp.
	Wrong	But she wanted to go home.
	Seq2Seq-MLE	ivy had a great time with her friends.
	Seq2Seq-Adversarial	ivy decided to go to church instead.

Table 6: Example story endings generated by our models, along with the original right and wrong story endings in the test dataset.

it tends to generate some frequent phrases. While Seq2Seq-Adversarial model generates a perfect story ending, which really amazed us: it is even better than the original right story ending. Moreover, this ending is far from being similar to the gold-standard right ending. But it fits to the story context perfectly. Hence, we believe that Seq2Seq-Adversarial model can generate more diversiform knowledge for this world. In case three, Seq2Seq-MLE generates a better story ending than Seq2Seq-Adversarial. This is mainly because in the adversarial training process, the MC search sampled the *go to church* phrase and it got a high reward score from the discriminator. This can be improved by carefully controlling the MC search process and filtering the meaningless words or phrases.

4 Related Work

4.1 Script Learning

The use of scripts in AI dates back to the 1970s (Minksy, 1975; Schank and Abelson, 1977; Mooney and DeJong, 1985). In this conception, *scripts* are composed of complex events without probabilistic semantics. In recent years, a growing body of research has investigated learning probabilistic co-occurrence-based models with simpler events. Chambers and Jurafsky (2008) proposed unsupervised induction of *narrative event chains* from raw newswire text, with *narrative cloze* as the evaluation metric, and pioneered the recent line of work on statistical script learning (Jans et al., 2012; Pichotta and Mooney, 2014; Pichotta and Mooney, 2016b; Granroth-Wilding and Clark, 2016). However, they utilized a very impoverished representation of events as the form of (*event, dependency*). To overcome the drawback of this event representation, Pichotta and Mooney (2014) presented a script learning approach that employed events with multiple arguments.

There have been a number of recent neural models for script learning. Pichotta and Mooney (2016a) showed that the LSTM-based event sequence model outperformed previous co-occurrence-based methods for event prediction. Pichotta and Mooney (2016b) used a Seq2Seq model directly operating on raw tokens to predict sentences, finding it is roughly comparable with systems operating on structured verb-argument events. Granroth-Wilding and Clark (2016) described a feedforward neural network which composed verbs and arguments into low-dimensional vectors, evaluating on a multiple-choice version of the Narrative Cloze task. However, most of this line of work build their models based on discrete verbs and tokens, which is far from being a complete sentence or story segment as used in this paper.

A line of works has studied the problem of Story Cloze Test on ROCStories corpus (Mostafazadeh et al., 2016). Chaturvedi et al. (2017) explored three distinct semantic aspects including sequence of

events, emotional trajectory and plot consistency, and used a hidden variable coherence model to join these aspects together. Lin et al. (2017) adopted a similar method, and also exploited heterogeneous knowledge for this task. Wang et al. (2017) applied adversarial networks on this task, which is most similar to our work. However, all these studies put their focuses on choosing the correct story ending through discriminative approaches. We mainly aim to generate reasonable and diversified story endings.

4.2 Sequence to Sequence Learning

Sequence to sequence learning (Seq2Seq) aims to directly model the conditional probability $p(Y|X)$ of mapping an input sequence $X = \{x_1, ..., x_N\}$, into an output sequence $Y = \{y_1, ..., y_T\}$. It accomplishes such goal through the encoder-decoder framework proposed by (Sutskever et al., 2014) and (Cho et al., 2014). The encoder computes a representation s for each input sequence. Based on the input representation, the decoder generates an output sequence one word at a time.

A natural model for sequential data is the recurrent neural network (RNN) (Elman, 1990), which is used by most of the recent Seq2Seq work. These work, however, differ in terms of: (a) architecture - from unidirectional, to bidirectional, and deep multi-layer RNNs, (b) RNN type - which is long-short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) or gated recurrent unit (GRU) (Cho et al., 2014).

Another important difference among Seq2Seq work lies in what constitutes the input representations. The early Seq2Seq work (Sutskever et al., 2014; Cho et al., 2014) used only the last encoder state to initialize the decoder state. While, Bahdanau et al. (2015) proposed an attention mechanism, a way to provide Seq2Seq models with a random access memory, to handle long input sequences. Recent work such as (Xu et al., 2015; Yin et al., 2017; Ayana et al., 2017) has found that it was crucial to empower Seq2Seq models with the attention mechanism. Despite its success, many issues emerge due to its over-simplified training objective. In this paper, we propose training Seq2Seq model with adversarial strategy.

4.3 Generative Adversarial Networks

The idea of generative adversarial nets is proposed by (Goodfellow et al., 2014), which has achieved great success in computer vision (Mirza and Osindero, 2014; Radford et al., 2015). Training is formalized as a game in which the generator is trained to generate outputs fooling the discriminator.

However, this idea has not achieved comparable success in the NLP field. This is mainly due to the fact that unlike in image generation, the discrete property of text generation makes the error computed by the discriminator hard to backpropagate to the generator. Some recent work try to address this issue: (Lamb et al., 2016) proposed providing the discriminator with intermediate hidden vectors of the generator, which makes the system differentiable and achieves promising results in language modeling and handwriting generation tasks. Yu et al. (2016) used policy gradient reinforcement learning to back-propagate the error from the discriminator, showing improvement in multiple generation tasks such as poem, speech language and music generation. Li et al. (2017) used a similar strategy to boost the dialogue generation quality, which achieved good experimental results.

Not limited to the task of sequence generation, Chen et al. (2016) applied the idea of adversarial training to sentiment analysis, and (Zhang et al., 2017) investigated the problem of domain adaptation based on adversarial networks. To our knowledge, this is the first paper to study adversarial training augmented Seq2Seq model on the task of generating story endings.

5 Conclusion

Story generation is a challenging problem in AI. In this paper, we explore new ways to generate story ending given a four-sentence story context. In order to generate high-quality story endings, we adopt the idea of adversarial training from *Generative Adversarial Nets* and propose using adversarial training augmented Seq2Seq model to generate reasonable and diversified story endings. Carefully designed human evaluation shows that the adversarial training augmented Seq2Seq model can generate higher quality story endings than purely MLE-trained Seq2Seq model, with 21.1% mean human annotation score improvement and 126.9% bigram diversity improvement. Furthermore, our model can choose the

right ending based on the generated story ending, and achieves the best performance. This verifies the potential of solving the Story Cloze Test problem from a generation perspective.

Acknowledgements

This work is supported by the National Key Basic Research Program of China via grant 2014CB340503 and the National Natural Science Foundation of China (NSFC) via grants 61472107 and 61702137. The authors would like to thank the anonymous reviewers for their insightful comments.

References

Ayana, Shi-Qi Shen, Yan-Kai Lin, Cun-Chao Tu, Yu Zhao, Zhi-Yuan Liu, Mao-Song Sun, et al. 2017. Recent advances on neural headline generation. *Journal of Computer Science and Technology*, 32(4):768–784.

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

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. In *ACL*, volume 94305, pages 789–797.

Snigdha Chaturvedi, Haoruo Peng, and Dan Roth. 2017. Story comprehension for predicting what happens next. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1604–1615, Copenhagen, Denmark, September. Association for Computational Linguistics.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *EMNLP*, pages 1724–1734.

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

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Peter W Glynn. 1990. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84.

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

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *AAAI*.

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

Linmei Hu, Juanzi Li, Liqiang Nie, Xiao-Li Li, and Chao Shao. 2017. What happens next? future subevent prediction using contextual hierarchical lstm. In *AAAI*, pages 3450–3456.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*, pages 2333–2338. ACM.

Bram Jans, Steven Bethard, Ivan Vulić, and Marie Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *EACL*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*, pages 3294–3302.

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *NIPS*, pages 4601–4609.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *ACL*, pages 1106–1115.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. *NAACL*, pages 110–119.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *EMNLP*, pages 2147–2159.

Hongyu Lin, Le Sun, and Xianpei Han. 2017. Reasoning with heterogeneous knowledge for commonsense machine comprehension. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2022–2033, Copenhagen, Denmark, September. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Marvin Minksy. 1975. A framework for representing knowledge. *The psychology of computer vision*, 73:211–277.

Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *Computer Science*, pages 2672–2680.

Raymond Mooney and Gerald DeJong. 1985. Learning schemata for natural language processing. *Urbana*, 51:61801.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. *NAACL*, pages 740–750.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Karl Pichotta and Raymond J Mooney. 2014. Statistical script learning with multi-argument events. In *EACL*, volume 14, pages 220–229.

Karl Pichotta and Raymond J Mooney. 2016a. Learning statistical scripts with lstm recurrent neural networks. In *AAAI*.

Karl Pichotta and Raymond J Mooney. 2016b. Using sentence-level lstm language models for script inference. *ACL*, pages 279–289.

Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Roger C Schank and Robert P Abelson. 1977. Scripts, plans, goals, and understanding: An inquiry into human knowledge structures (artificial intelligence series).

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *ACL*, pages 1577–1586.

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

Bingning Wang, Kang Liu, and Jun Zhao. 2017. Conditional generative adversarial networks for commonsense machine comprehension. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4123–4129.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, volume 14, pages 77–81.

Jun Yin, Wayne Xin Zhao, and Xiao-Ming Li. 2017. Type-aware question answering over knowledge base with attention-based tree-structured neural networks. *Journal of Computer Science and Technology*, 32(4):805–813.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2016. Seqgan: sequence generative adversarial nets with policy gradient. *AAAI*.

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *TACL*.

Point Precisely: Towards Ensuring the Precision of Data in Generated Texts Using Delayed Copy Mechanism

Liunian Li **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{liliunian, wanxiaojun}@pku.edu.cn

Abstract

The task of data-to-text generation aims to generate descriptive texts conditioned on a number of database records, and recent neural models have shown significant progress on this task. The attention based encoder-decoder models with copy mechanism have achieved state-of-the-art results on a few data-to-text datasets. However, such models still face the problem of putting incorrect data records in the generated texts, especially on some more challenging datasets like ROTOWIRE. In this paper, we propose a two-stage approach with a delayed copy mechanism to improve the precision of data records in the generated texts. Our approach first adopts an encoder-decoder model to generate a template text with data slots to be filled and then leverages a proposed delayed copy mechanism to fill in the slots with proper data records. Our delayed copy mechanism can take into account all the information of the input data records and the full generated template text by using double attention, position-aware attention and a pairwise ranking loss. The two models in the two stages are trained separately. Evaluation results on the ROTOWIRE dataset verify the efficacy of our proposed approach to generate better templates and copy data records more precisely.

1 Introduction

One important task in the natural language generation (NLG) area is to generate a natural language description for some structured data (i.e., a number of database records), a.k.a., the data-to-text generation task. Applicable to wide areas such as weather forecasting, financial reporting, game broadcasting, data-to-text generation systems have been studied for decades (Gatt and Krahmer, 2018). Traditionally this task is addressed by using templates or statistically learned shallow models with hand-crafted features (van Deemter et al., 2005; Belz, 2008; Konstas and Lapata, 2012). Recently, neural generation systems have shown significant progress on this task. The attention based encoder-decoder models with the copy mechanism have achieved state-of-the-art results on a few data-to-text datasets (Wiseman et al., 2017; Sha et al., 2017).

In this paper, we try to address the problem of faithfully describing the data in the generated texts, i.e., how to generate a descriptive text that contains correct entities and numeric values. Intuitively speaking, when a human writer writes weather forecasts or game summaries, he may not consider the exact entities or numbers right away. Rather, he may have finished the whole sentence (without the actual entities or numbers) in his mind, then look up the data table and fill in those exact entities and numbers at last.

Likewise, we don't have to train our models to construct sentences and fill in the data at the same time. If we take out the entities and numeric values from the text, what is left can be considered some kind of templates with data slots waiting to be filled in. Therefore, the process of generating descriptions can be naturally divided into two stages: template generation and slot filling. An encoder-decoder based generator can be in charge of generating the templates with entities and numeric values left blank. A delayed copy network can be in charge of filling in those data slots with proper data records. Compared to traditional copy mechanisms, our approach has the following advantages:

Proceedings of the 27th International Conference on Computational Linguistics, pages 1044–1055
Santa Fe, New Mexico, USA, August 20-26, 2018.

- Traditional copy mechanisms generate and copy words at the same time, which involves merging a score distribution over the input words and a score distribution over the words in the vocabulary. It remains unclear exactly how different merging strategies affect performance. We avoid this uncertainty by leaving the task of generating words to the template generator, the task of deciding which word to copy to the delayed copy network. The two-stage approach also allows us to use a more effective pairwise ranking loss function.

- Relieved of the burden of having to both generate and copy words, the template generator can now concentrate on generating better templates. This can be addressed by using a typical encoder-decoder model, which has proven to be easy to train and tune.

- The delayed copy mechanism can now concentrate on slot filling, and it can utilize all the information in the input data records and the full generated template text by using double attention and position-aware attention, making the delayed copy mechanism point more precisely.

We evaluate our approach on the public ROTOWIRE dataset and evaluation results verify the efficacy of our approach, which makes a boost on the data precision of generated texts and brings a noticeable increase on BLEU score.

We organize the paper as follows. We introduce some background knowledge in Section 2 and describe our approach in Section 3. In Section 4 we present the experiments and have discussion. In Section 5 we note some additional related works. In Section 6 we conclude this paper.

2 Background

The Data-to-text Task Following the notations in Liang et al. (2009) and Wiseman et al. (2017), let $S = \{r_j\}$ be a set of records, where for each $r \in S$, $r.type$, $r.entity$ and $r.value$ are the record's type, entity and value, respectively. For example, there could be a record r in a basketball dataset such that $r.type$ = POINTS FIRST QUARTER, $r.entity$ = LEBRON JAMES, and $r.value = 10$, which means LeBron James scored 10 points in the first quarter. From these records, we are interested in generating an descriptive summary $\hat{\boldsymbol{y}} = \hat{y}_1, \ldots, \hat{y}_T$ of T words. A data-to-text dataset consists of pairs like $(S, \boldsymbol{y})$, where $\boldsymbol{y}$ is a gold (human generated) summary for the record set S. In this paper, we develop our system on the ROTOWIRE dataset (Wiseman et al., 2017), which consists of documents summarizing NBA basketball games and the corresponding game data. For every entity that appears in S, we add an additional record r^* to S, with $r^*.entity$ and the $r^*.value$ both set to the entity's name. For example, if LEBRON JAMES appears in the game, a record r^* with $r^*.entity$ = LEBRON JAMES, $r^*.value$ = LEBRON JAMES, $r^*.type$ = PLAYER NAME is added to S. This aims to copy entity in a neat and uniform way, which will be discussed later.

If a record is mentioned in the text, we consider the text as "faithfully describing the data" if the sentence in which $r.entity$ and $r.value$ are both present correctly reflects $r.type$. For example, for the fore-mentioned record r, if the sentence is "LeBron James got 10 rebounds in the second quarter", it is considered unfaithful to the record. Instead, a sentence correctly reflecting the record should be "LeBron James scored 10 points in the first quarter." Generating texts conditioned on the records is a non-trivial problem because a sentence could likely contain several records. However, there have been few works explicitly addressing this issue due to the lack of challenging datasets. Previous datasets (Liang et al., 2009; Chen and Mooney, 2008; Murakami et al., 2017) generally use relatively simple language and record structure, where some simple approaches suffice. For instance, some models (Mei et al., 2016) generate entities and numeric values as other normal words while some works (van Deemter et al., 2005; Liang et al., 2009; Murakami et al., 2017) employ a *"tag-then-replace"* method, where they train their models to generate special tags which will be replaced with true entities or numeric values using hand-written rules. However, these approaches may fail under more complicated situations. For example, in the ROTOWIRE dataset, there are over 40 $type$s of records and the entity names that appear in the $entity$ portion of records change from game to game, making it hard, if not impossible, to devise hand-written rules for the *"tag-then-replace"* method.

Copy Mechanism Copy mechanism in encoder-decoder models (Vinyals et al., 2015; Gu et al., 2016; Gülçehre et al., 2016; Yang et al., 2017) provides a way to directly copy words from the input. At each time step of decoding, an attention-like function is used to produce an unnormalized score distribution over the inputs, denoted as P^*_{copy}. The decoder also produces an unnormalized distribution P^*_{word} over the words in the vocabulary. These two unnormalized score distributions are then merged into a normalized distribution P_{joint}. The loss function is defined as the negative log likelihood function of vanilla encoder-decoder models: $L = \sum_t -\log P_{joint}(w^*_t|w_{1:t-1}, S)$, where w^*_t is the target word at time step t, $w_{1:t-1}$ are previous words, and S is the inputs. We now illustrate how copy mechanism is used in this task. A copy means the $value$ portion (could be a numeric value or an entity name) of a record r is copied into the output text. To train a copy network, one has to specify which token in the output text is copied and from which input record it is copied. For some token y_t in the gold text, we assume y_t to be copied from some input record r only if the following constraint is met: $y_t = r.value$ and $r.entity$ is present in the same sentence with y_t. Therefore, for every token y_t, if we can find a non-empty set of records C, where every $r \in C$ meets the constraint, y_t is considered to be copied, and C is the set of valid records that is the source of y_t.

3 Our Approach

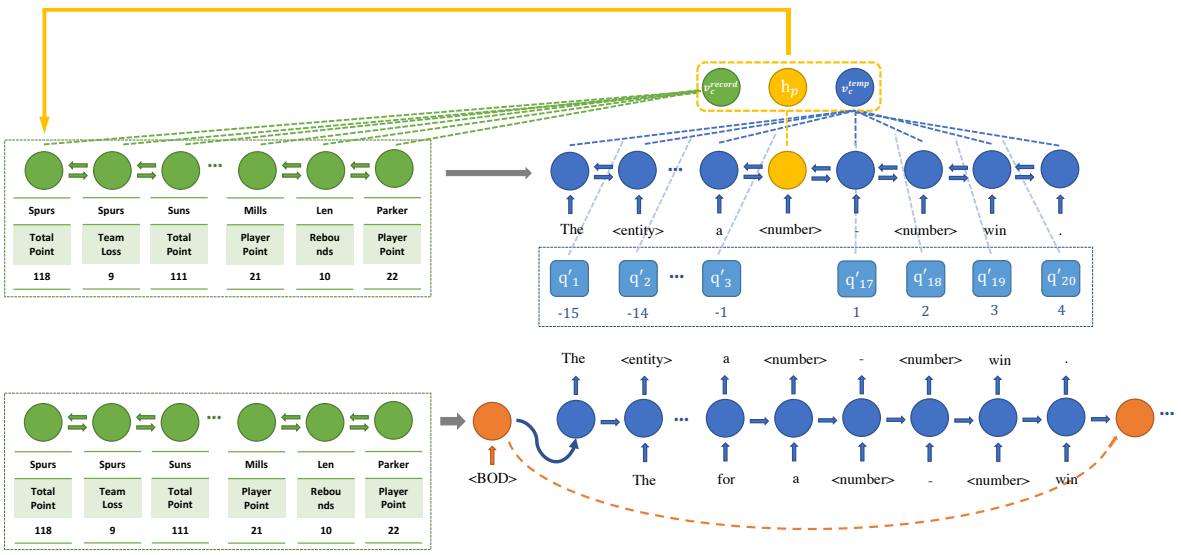


Figure 1: Our Framework: The template generator (below) generates templates and then the delayed copy network (above) fills in the slots.

Our system consists of two models, an encoder-decoder based template generator that constructs the whole sentence with data slots left blank and a delayed copy network that fills in those data slots informed by both the input and the generated sentence. From a computational perspective, this two-stage method has many practical advantages. The template generator is freed from the burden of having to both generate and copy words while the copy network could benefit from utilizing the information from the full templates. This is somewhat similar to Xia et al. (2017), where they observed performing a second decoding using information from the first decoding could boost performance. In addition, detaching the copy network from the text decoder allows room for alternative loss functions, opening up a new possibility for our model. We divided the task in a neat and intuitive way, avoiding the uncertainty in the merging procedure in traditional copy mechanisms. At last, we can reduce training time by training the template generator and the copy network concurrently.

Original Text: The San Antonio Spurs (47 - 9) held off the Phoenix Suns (14 - 42) for a 118 - 111 win. Spurs 's three - point shooting was the key , as they went 10 - for - 16 from long distance in the victory .
Target Template: The <entity> (<number> - <number>) held off the <entity> (<number> - <number>) for a <number> - <number> win. <entity> 's three - point shooting was the key , as they went <number> - for - <number> from long distance in the victory .

Table 1: An example of the original text and the corresponding target template.

3.1 Template Generation with Encoder-Decoder Model

Overview The template generator generates the major part of the text while leaving the task of actually copying to the delayed copy network. It is essentially an encoder-decoder model and any well-performing model would suffice. In this work, we use a hierarchical model similar to the one in Li et al. (2015).

While the template generator does not decide which input record to copy from, it is responsible for deciding whether to copy a token from the input records. At each time step t, if the template generator finds it necessary to copy a token, it outputs a special placeholder (a data slot), which will be replaced with actual entities and numeric values by the copy network. The special placeholder could be "<data>". However in this specific task, we make a distinction between entities and numeric values, and define two placeholders, "<entity>" and "<number>". We denote the templates as $\boldsymbol{y'}$. Now instead of learning to generate the original text $\boldsymbol{y}$, the template generator learns to generate $\boldsymbol{y'}$. An example is shown in Table 1.

The placeholders are added to the vocabulary of the template generator. The template generator outputs the placeholders just like other words. In this way, the process of deciding whether to copy a token is incorporated naturally into the text decoding process, cleverly avoiding the merging operation in traditional copy mechanisms. As mentioned in Section 2, traditional copy mechanisms have to merge P^*_{copy} and P^*_{word} into a normalized distribution P_{joint}. This was designed to allow the input records to "compete" with the words in the vocabulary but there is no clear conclusion about how different merging operations affect performance. In this task, we find that the template generator is fully capable of deciding whether to copy or not.

Structure An input encoder Enc_{input} first encodes every record $r \in S$ by embedding $r.type$, $r.entity$ and $r.value$ and applying a 1-layer MLP followed by an LSTM over the embedding vectors, i.e. $\{\tilde{r}_j\} = Enc_{\text{input}}(\{r_j\})$. We also note that in this paper, all embedding vectors and hidden states are in $\mathbb{R}^D$ unless stated otherwise, where D is the dimension. The input encoder is followed by a hierarchical decoder consisting of a sentence decoder Dec_{sent} and a word decoder Dec_{word}. The sentence decoder outputs sentence representations while the word decoder predicts words in a sentence sequentially. The hidden states of the word decoder Dec_{word} are denoted as $\{h^{word}_t\}$. At test time, beam search can be applied to the word decoder of the template generator. For more details, please refer to Li et al. (2015).

3.2 Slot Filling with Delayed Copy Network

Overview Our copy network fills in the data slots (placeholders) in the generated template. For every placeholder in the template $\boldsymbol{y'}$, the delayed copy network takes in the following inputs: an input vector h_p corresponding to the placeholder; a series of input-record-based vectors $\{h^{\text{record}}_j\}$, each corresponding to an input record r; another series of template-text-based vectors $\{h^{\text{temp}}_t\}$, each corresponding to a word in the template sentence that the placeholder is in. After applying the attention mechanism to both the input records and the template text, it generates a score distribution over the input records and selects the record with the highest score to copy from.

Let us assume the placeholder in question is the m-th word in the template sentence. We always let $h_p = h^{\text{temp}}_m$. We now discuss how we can obtain $\{h^{\text{record}}_j\}$ and $\{h^{\text{temp}}_t\}$. There are actually two ways to do this. We can reuse the hidden states from the template generator to get $\{h^{\text{record}}_j\}$ and $\{h^{\text{temp}}_t\}$[1]. Or we

[1]Concretely, the input-record-based vectors are from the encoder, i.e. $\{h^{\text{record}}_j\} = \{\tilde{r}_j\}$; the template-text-based vectors are from the word decoder, i.e. $\{h^{\text{temp}}_t\} = \{h^{word}_t\}$

can fully detach the delayed copy network from the template generator by using a template text encoder Enc_{temp} to get $\{h_t^{\text{temp}}\}$, a separate input encoder Enc^*_{input} to get $\{h_j^{\text{record}}\}$[2].

Although in the first method, models can be trained in an end-to-end fashion, the burden of actually copying tokens still falls back to the template generator as its hidden states are reused in the copy network. In the second method, which we call a "two-stage" method, the template generator and the copy network are trained separately. This allows the template generator to focus on generating templates. Moreover, since the copy happens after template generation, a bi-directional RNN (Schuster and Paliwal, 1997) can be used in Enc_{temp}, which enables the hidden state of each word to summarize not only the preceding words but also the following words.

In this specific task, we always fill in the <entity> placeholders first and when we do that, we only consider the additional records r^* mentioned in Section 2. When we fill in the <number> placeholders, we only consider the records whose $entity$ portions have been copied into the sentence. In short, when we replace a certain placeholder, only a subset of S, which we denote as S^*, are considered. This small S^* trick ensures the copy network does not copy numeric values whose $entity$ portion is not present in the same sentence. This trick is not applicable in traditional copy networks where the order in which we copy tokens can not be adjusted.

Double Attention Attention mechanism (Bahdanau et al., 2014; Luong et al., 2015) can be described as a function mapping a query vector v_q and a series of source vectors $\{v_s\}$ to a context vector v_c, i.e. $v_c = Atten(v_q, \{v_s\})$. More concretely,

$$v_c = \sum a_s v_s \tag{1}$$

where

$$a_s = \frac{\exp(u_s)}{\sum_{s'} \exp(u'_s)}, \quad u_s = v_q^T \cdot W \cdot v_s \tag{2}$$

and W is a parameter matrix in $\mathbb{R}^{D \times D}$.

To point more precisely, attention mechanism is employed to the input records as well as to the generated template. Given the input vector h_p, the input-record-based vectors $\{h_j^{\text{record}}\}$ and the template-text-based vectors $\{h_t^{\text{temp}}\}$, an input-record-based context vector $v_c^{\text{record}} = Atten(h_p, \{h_j^{\text{record}}\})$ is computed to utilize information from the input records. Likewise, a template-text-based context vector $v_c^{\text{temp}} = Atten(h_p, \{h_t^{\text{temp}}\})$ is computed to utilize information from the template. In contrast, the traditional copy network only applies attention mechanism to the input records.

v_c^{record}, v_c^{temp} and the input vector h_p are then combined to produce a score for a i-th input record r_i:

$$Score(r_i) = [v_c^{\text{record}}; v_c^{\text{temp}}; h_p]^T \cdot W_1 \cdot \tilde{r}_i \tag{3}$$

where W_1 is a parameter matrix in $\mathbb{R}^{3D \times D}$.

Position-aware Attention In the task of relationship classification, which involves determining the relationship of a pair of entities, it has been observed that the relative distances from other words in the sentence to the entities could be informative (Zeng et al., 2014; dos Santos et al., 2015). Inspired by Zhang et al. (2017), we employ a modified position-aware attention mechanism to the template text. When replacing some placeholder using the copy network, the distances between every word in the generated text and the placeholder are incorporated into the calculation of the template-text-based context vector v_c^{temp}. Compared to simply using $Atten(h_p, \{h_t^{\text{temp}}\})$ to calculate v_c^{temp}, this method allows the network to pay attention to words according to their relative locations to the placeholder.

Formally, let's assume $\{y'_1, y'_2, ..., y'_n\}$ is the generated template sentence and the placeholder we are replacing is the m-th token. For every word in the sentence other than the placeholder itself, we calculate its distance to the placeholder to obtain a position sequence $\{q_1, q_2, ..., q_{m-1}, q_{m+1}, ..., q_n\}$,

[2]Concretely, $\{h_t^{\text{temp}}\} = Enc_{\text{temp}}(\{y'_t\})$, where $\{y'_t\}$ is the sentence the placeholder is in; the separate input encoder has identical settings with the input encoder in template generator, so $\{h_j^{\text{record}}\} = Enc^*_{\text{input}}(\{r_j\})$; the hidden state of the Enc_{temp} is initialized using the hidden state of Enc^*_{input}.

where $q_t = t - m$. Using a position embedding matrix Q, we can then obtain a positional embedding sequence$\{q'_1, q'_2, ..., q'_{m-1}, q'_{m+1}, ..., q'_n\}$. Combining $\{q'_t\}$ and $\{h_t^{\text{temp}}\}$, we can calculate the template-text-based context vector v_c^{temp} as:

$$v_c^{\text{temp}} = \sum_{t \in [1,n]; t \neq m} a_t h_t^{\text{temp}} \tag{4}$$

where

$$a_t = \frac{\exp(u_t)}{\sum_{t'} \exp(u'_t)}, \quad u_t = h_p^T \cdot W_2 \cdot [h_t^{\text{temp}}; q'_t] \tag{5}$$

and W_2 is a parameter matrix in $\mathbb{R}^{D \times 2D}$.

Pairwise Ranking Loss As mentioned in Section 2, in traditional copy mechanisms, the unnormalized distribution over input records P^*_{copy} and the unnormalized distribution over words in the vocabulary P^*_{word} are merged into a normalized distribution P_{joint}, which involves somehow applying a softmax function to P^*_{copy}[3]. This method limits the possibility of alternative loss functions for the copy network. In fact, choosing the right input record to copy can also be viewed as ranking the records according to their relative appropriateness, which justifies the use of a ranking loss function. It has been observed in the task of relationship classification that a pairwise ranking loss function is better than a log-likelihood loss function following the softmax function (dos Santos et al., 2015). Following the notation in Section 2, for any y_t that is considered to be copied from the input, C is the set of valid input records that is the source of y_t while S^* is the record set worth considering. A score can be calculated for every record in S^* using Equation 3. We reward the scores of the valid input records and penalize the highest score among all invalid input records by minimizing a pairwise ranking loss function, defined as:

$$L = log(1 + exp(\gamma(m^+ - \sum_{r_i \in C} Score(r_i))) + log(1 + exp(\gamma(m^- + \max_{r_i \in S^*; r_i \notin C} Score(r_i))) \tag{6}$$

where m^+ and m^- are margins and γ is a scaling factor that magnifies the difference between the score and the margin, and helps to penalize more on the prediction errors.

4 Experiments

4.1 Setup

Dataset We experiment our method on the ROTOWIRE dataset (Wiseman et al., 2017). The average text length is 337.1, while the average text lengths of previous datasets (Chen and Mooney, 2008; Liang et al., 2009; Lebret et al., 2016) do not exceed 30. The average number of input records is 628, while those of other datasets do not exceed 200. The text in this dataset also contains a decent percentage of entities and numeric values. This is indeed a challenging enough dataset to test our model on.

Implementation We perform an additional step in preprocessing called "relexicalization"[4]. We used PyTorch (Paszke et al., 2017) for implementation. For encoders and decoders, we use two layers of LSTM. The hidden states and the embedding vectors are all in $\mathbb{R}^{600}$. General-style attention and input-feeding (Luong et al., 2015) are employed. The models are trained using Adam (Kingma and Ba, 2014). The hyper parameters for pairwise ranking loss are set to those provided in dos Santos et al. (2015) without further tuning.

[3] In some models (Gu et al., 2016) P^*_{copy} is not directly normalized. It is mixed with P^*_{word} and then normalized jointly. But generally, we can say that P^*_{copy} is somehow "normalized" using a softmax function.

[4] An entity may have different aliases. For example, "Phoenix Suns" can be referred to as "Suns", "Phoenix" or "Phoenix Suns"; a player can be referred to using his first name or second name or full name. In this work, we didn't make our model learn this naming strategy for processing simplicity. We replace all different alias of the same entity with a uniform name. However, there might be need for diverse naming strategies in a real world situations and we leave that to future work.

	Development					
	RG		CS		CO	BLEU
Model	$P\%$	$\#$	$P\%$	$R\%$	$DLD\%$	
Gold	94.39	16.72	100	100	100	100
Joint Copy with Rec & TVD	62.83	16.76	26.99	**39.62**	11.96	12.50
Conditional Copy	74.62	16.53	30.74	38.23	14.69	14.47
Our model	**85.81**	19.39	31.02	39.45	**16.93**	**16.44**
-double attention	83.04	18.22	**31.20**	38.83	16.70	16.23
-pos attention	84.80	19.17	30.88	39.32	16.73	16.18
-ranking loss	77.62	17.93	30.29	39.36	15.71	15.79
-S^* trick	83.17	**19.69**	29.25	39.06	16.19	16.30
	Test					
Gold	94.50	16.99	100	100	100	100
Joint Copy with Rec & TVD	60.82	16.31	26.78	**39.90**	14.15	13.47
Conditional Copy	74.70	16.19	**32.32**	38.96	15.24	14.16
Our Model	**84.86**	**19.31**	30.81	38.79	**16.34**	**16.19**

Table 2: Comparison results. All results are tested after "relexicalization".

Evaluation Metrics For a long time, the automatic evaluation metrics for the data-to-text task have been limited to ones like BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004). However, Wiseman et al. (2017) proposed to use relationship classification techniques for evaluation. Given a sentence and a pair of an entity and a numeric value in the sentence, a relationship classification system can predict the relationship between the entity and the numeric value. Therefore, such a system can be used to assess the quality of automatic generations by extract correctly described records from the generated text. We follow Wiseman et al. (2017) and evaluate our models on these three automatic metrics:

Relation Generation (RG): precision ($P\%$) and number ($\#$) of unique records correctly reflected in the generated text.

Content Selection (CS): precision ($P\%$) and recall ($R\%$) of unique records correctly reflected in the generated text that are also reflected in the gold text.

Content Ordering (CO): normalized Damerau-Levenshtein Distance ($DLD\%$) (Brill and Moore, 2000) between the sequences of records extracted from the generated text and those from the gold text.

Among the three metrics, we argue that the RG metrics is the most crucial one as it evaluates the data fidelity of the system. In this task, it could be considered okay if the system decides to report the game from a different angle and describe some data that are not present in the gold text or describe them in a different order, as the saying goes, "There are a thousand Hamlets in a thousand people's eyes." However, it would be intolerable if it got the scores wrong. We also note that these automatic metrics provided in Wiseman et al. (2017) can achieve over 94% accuracy on held-out data, making it practical for evaluation.

4.2 Comparison with the State-of-the-art Models

Model	Dev Set BLEU	Test Set BLEU
Joint Copy with Rec & TVD	11.53	11.83
Conditional Copy	13.08	12.59
Our Model	**14.66**	**14.33**

Table 3: Template quality

Here we present the comparison results against models with state-of-the-art traditional copy mechanisms[5] in Table 2, including the Joint Copy with Rec & TVD model, which is an augmented version of

[5]We also tried to train a Conditional Copy model with a hierarchical structure, however the results are not satisfying. We argue that this could be because the hierarchical structure is interfering with the copy network, let alone the fact that this Conditional Copy model has been observed to require extra consideration in training procedure.

the Joint Copy model (Gu et al., 2016), and the Conditional Copy model (Gülçehre et al., 2016). Here we used trained models provided by Wiseman et al. (2017) and performed "relexicalization" on the results to keep things fair.

It is clear that our model outperformed other models by a large margin on data fidelity. The precision of record generation (RG) boosted by more than 10%, achieving 85%. The number of correctly generated records also went up by more than 2.5.

We also observed an increase in BLEU score by about two points. We argue that it is not only because the entities and numbers are filled in more precisely, but also because the templates are generated better. In fact, we compared the template quality of different models in Table 3. We take out all the entities and numeric values in the generated text as well as the gold text and calculate their BLEU score. Results show that our model indeed generates better templates.

With the precision up, we also observed a decent increase on the content ordering (CO) metrics. It is worth noting that an increase on RG does not guarantee improvement on content selection (CS). This is perhaps because models that learn to generate records more accurately may not learn to generate records more like human writers. In fact, all models achieved close scores on the recall of CS. It is worth noting that the Joint Copy model, worst on RG, achieved the highest recall on CS.

4.3 Model Validation

We conduct experiments to show how the model's performance is affected by removing some components in the delayed copy network. We used the same template generator for all these models. Results (Table 3) reveal some interesting characteristics about our models.

The biggest boost comes from the pairwise ranking loss we employed. Replacing the ranking loss with a log-likelihood loss in traditional copy mechanisms ("*-ranking loss*") would result in a drop in RG. The BLEU suffered less, which we argue is due to the well-performing template generator. The "*-ranking loss*" model still outperforms the Conditional Copy model, with higher precision of record generation, more correct records and higher BLEU score. Removing the template-text-based attention ("*-double attention*") or replacing the template-text-based position-aware attention with a vanilla attention ("*-pos attention*") also results in a decline on mostly RG and BLEU, showing the efficacy the double attention and position-aware attention. We also tested a model without the S^* trick ("*-S^* trick*") mentioned in Section 2. It can be seen that this trick makes a small improvement to our model yet it is not a defining factor.

4.4 Case Study

Our Model: The Magic (25 - 53) defeated the Bulls (46 - 32) 105 - 103 on Wednesday at the Amway Center in Orlando . The Magic got off to a quick start in this one , out - scoring the Bulls 29 - 21 in the first quarter alone . The Magic were the superior shooters , going 46 percent from the field and 35 percent from the three - point line , while the Bulls went 43 percent from the floor and just 46 percent from beyond the arc . The Magic were also able to force the Bulls into 25 turnovers , while committing only 15 of their own , which may have been the difference in this one . The Magic were led by Tobias Harris , who posted 8 points (2 - 13 FG , 4 - 4 FT) , 3 rebounds and three steals ...
Conditional Copy: The Magic (25 - 53) defeated the Magic (25 - 53) 105 - 105 on Wednesday at the Amway Center in Orlando . The Bulls got off to a quick start in this one , out - scoring the Bulls 29 - 21 right away in the first quarter . The Magic were the superior shooters in this game , going 46 percent from the field and 46 percent from the three - point line , while the Magic went just 43 percent from the floor and 35 percent from deep . The Bulls were also able to force the Magic into 16 turnovers , while committing only 16 of their own . The Bulls were led by the duo of Nikola Vucevic and Nikola Vucevic . Nikola Vucevic went 9 - for - 16 from the field and 3 - for - 4 from the three - point line to score a game - high of 22 points , while also adding two rebounds and two assists ...
Gold: The Magic (25 - 53) defeated the Bulls (46 - 32) 105 - 103 on Wednesday at the Amway Center in Orlando . Down two with just over six seconds left in the game , it was Pau Gasol who was able to force his way to the free throw line and hit a pair of free throws to tie the game up . Victor Oladipo then came up clutch for the Magic , driving for a layup with just a second left in the game , therefore giving them a two point lead and eventually the victory . With the loss , the Bulls move into a tie with the Toronto Raptors for the third projected playoff seed in the Eastern Conference . With just four games left in the regular season , it will be a battle for future playoff positioning from here on out . The Magic were superior shooters in this one , going 46 percent from the field , while the Bulls finished at 43 percent ...

Table 4: Example texts generated by different models. Only a part of the texts are showed due to space limitation. The erroneous text has been marked red. Unmarked text correctly reflects records.

We show an example of the text generated by our model, the Conditional Copy model and the corresponding gold text in Table 4. It is clear that our model made fewer mistakes when it comes to accurately describing the data. One interesting finding is that the Conditional Copy model suffers from a common issue found in language-model based models, repetition. The repetition is not shown in repeated sentence

Model	Correct	Wrong	Repeated
Conditional Copy	3.34	1.58	0.35
Our Model	**3.78**	**1.03**	**0.10**

Table 5: Average number of correct , wrong and repeated records in the generated text per sentence.

patterns like in language-model based models but in the repeatedly copied entities and numbers. There are more than 7 repetitions in this short text. However, in our model, the delayed copy network avoided such problem since it is actually not based on language models. We conducted human evaluations to support our assumption. We manually checked 100 sentences randomly selected from summaries for 8 randomly selected games. We checked for correct records, wrong records and repeated records in the text. A record is considered repeated if it is mentioned twice in a short segment. Results (Table 5) show that our model generates records more precisely and makes much fewer repetitions.

We also inspected the attention heat map of the decoding-side attention. The attention weights seem to concentrate heavily on a single word, rather than spread across words. We argue that this is probably because the attention mechanism in our delayed copy network works somehow differently from the traditional attention mechanism. In machine translation (Bahdanau et al., 2014; Luong et al., 2015), the attention mechanism is applied between a language-model based decoder and an encoder. However, in our model, the template-text-based attention mechanism is applied within a bi-directional text encoder. So the bi-directional encoder in our model could have learned to condense the information into the hidden state of a single word.

5 Related Work

Traditionally, data-to-text tasks are decomposed into two subproblems (Kukich, 1983; Goldberg et al., 1994): *content selection*, which involves choosing a subset of relevant records to talk about, and *surface realization*, which is concerned with generating natural language descriptions for this subset. There is also an alternative decomposition in Reiter and Dale (1997), where they break the problem down to three modules: *content selection*, *micro planning*, and *surface realization*. There is also a corresponding work (Mellish et al., 2006) that challenges this decomposition. In early stages, *surface realization* is often realized using templates (van Deemter et al., 2005) or statistically learned models with hand-crafted features (Belz, 2008; Konstas and Lapata, 2012). Machine learning approaches have also been applied to *content selection*. Barzilay and Lapata (2005) models it as a classification problem, whereas Liang et al. (2009) uses a generative semi-Markov model.

With the rise of neural networks, some recent work has focused on generating text from data using neural networks, for example, generating weather forecasts as well as game descriptions (Mei et al., 2016), generating short biographies from Wikipedia Tables (Lebret et al., 2016; Hachey et al., 2017; Sha et al., 2017; Liu et al., 2017), and generating market comments from stock prices (Murakami et al., 2017). With these models achieving excellent results on traditional evaluation metrics such as BLEU (Papineni et al., 2002) or ROUGE (Lin, 2004), Wiseman et al. (2017) proposed to evaluate data-to-text systems from three aspects using relationship classification systems: data fidelity, content selection and content ordering. While there has been some work (Liu et al., 2017; Sha et al., 2017) related to the last two aspects, we find little work explicitly tackling the data fidelity problem.

Here we point out a parallel and independent work (Lu et al., 2018) on image caption that also employs a similar framework. They also generate slotted templates first and then fill in those slots with a separate model, though the exact models used in their work is quite different from ours.

6 Conclusion and Future Work

In this paper, we present a novel two-stage approach with a delayed copy mechanism to improve the precision of data in generated texts. Experiments show that our model points more precisely than other state-of-the-art models and also generates better templates. There is lots of future work we can do. The content selection ability of the model has the potential to be improved and we would like to further

improve the precision of copy mechanism. Another appealing direction is to adopt our delayed copy mechanism to other NLG domains.

Acknowledgments

This work was supported by National Natural Science Foundation of China (61772036, 61331011) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We thank Jiwei Tan, Hongyu Zang and the anonymous reviewers for their helpful comments. We also thank the authors of Wiseman et al. (2017) for releasing their code. Xiaojun Wan is the corresponding author.

References

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

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*, pages 331–338.

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(4):431–455.

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling correction. In *38th Annual Meeting of the Association for Computational Linguistics, Hong Kong, China, October 1-8, 2000.*

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 128–135.

Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 626–634.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Eli Goldberg, Norbert Driedger, and Richard I Kittredge. 1994. Using natural-language processing to produce weather forecasts. *IEEE Expert*, 9(2):45–53.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Çaglar Gülçehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Ben Hachey, Will Radford, and Andrew Chisholm. 2017. Learning to generate one-sentence biographies from wikidata. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 633–642.

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

Ioannis Konstas and Mirella Lapata. 2012. Unsupervised concept-to-text generation with hypergraphs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 752–761. Association for Computational Linguistics.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *Proceedings of the 21st annual meeting on Association for Computational Linguistics*, pages 145–150. Association for Computational Linguistics.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1203–1213.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1106–1115.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 91–99.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2017. Table-to-text generation by structure-aware seq2seq learning. *CoRR*, abs/1711.09724.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2018. Neural baby talk. *CoRR*, abs/1803.09845.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 720–730.

Chris Mellish, Donia Scott, Lynne J. Cahill, Daniel S. Paiva, Roger Evans, and Mike Reape. 2006. A reference architecture for natural language generation systems. *Natural Language Engineering*, 12(1):1–34.

Soichiro Murakami, Akihiko Watanabe, Akira Miyazawa, Keiichi Goshima, Toshihiko Yanase, Hiroya Takamura, and Yusuke Miyao. 2017. Learning to generate market comments from stock prices. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1374–1384.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 311–318.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. 2017. Order-planning neural text generation from structured data. *CoRR*, abs/1709.00155.

Kees van Deemter, Mariët Theune, and Emiel Krahmer. 2005. Real versus template-based natural language generation: A false opposition? *Computational Linguistics*, 31(1):15–24.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2692–2700.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2253–2263.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 1782–1792.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1850–1859.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 2335–2344.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 35–45.

Enhancing General Sentiment Lexicons for Domain-Specific Use

Tim Kreutz **Walter Daelemans**
CLiPS - Computational Linguistics Group
Department of Linguistics
University of Antwerp
{tim.kreutz,walter.daelemans}@uantwerpen.be

Abstract

Lexicon based methods for sentiment analysis rely on high quality polarity lexicons. In recent years, automatic methods for inducing lexicons have increased the viability of lexicon based methods for polarity classification. SentProp is a framework for inducing domain-specific polarities from word embeddings. We elaborate on SentProp by evaluating its use for enhancing DuOMan, a general-purpose lexicon, for use in the political domain. By adding only top sentiment bearing words from the vocabulary and applying small polarity shifts in the general-purpose lexicon, we increase accuracy in an in-domain classification task. The enhanced lexicon performs worse than the original lexicon in an out-domain task, showing that the words we added and the polarity shifts we applied are domain-specific and do not translate well to an out-domain setting.

1 Introduction

Work in sentiment analysis can roughly be divided into two approaches: corpus based techniques and lexicon based techniques (Taboada et al., 2011). Corpus based techniques use supervised learning on large target domain corpora to learn classification models. Currently, implementations using deep learning make up the state of the art in binary polarity classification on Twitter (Rosenthal et al., 2017).

Lexicon based techniques rely on high quality polarity lexicons. Document-level polarity can be predicted by looking at occurrences of lexical units in the document and doing some form of averaging on their respective polarity values. Despite not yielding the state of the art results of corpus based methods, lexicon based methods still see frequent use since they do not rely on big (labelled) data and allow for more interpretability. In recent years, polarity lexicons have been improved by adding new phrases from web data (Velikovich et al., 2010) or by automatically learning domain-specific polarities (Hamilton et al., 2016).

Sentprop is a framework for inducing domain-specific polarities (Hamilton et al., 2016). SentProp has been shown to accurately reproduce domain-specific lexicons from in-domain word embeddings. Additionally, it has been used to examine interesting historical and community-specific polarity shifts.

We elaborate on SentProp by evaluating its use for domain adaptation of a general-purpose lexicon. Starting from a Dutch general polarity lexicon and a corpus of political forum posts, we automatically expand and enhance the general lexicon entries. We show that the enhanced lexicon outperforms the general-purpose lexicon on an in-domain classification task and we further add to the existing body of work by evaluating the enhanced lexicon on an out-domain task. In the out-domain task the general purpose-lexicon yields better accuracy, showing that our proposed method learns domain-specific polarities that do not translate well to other domains.

The paper is organized as follows. Section 2 will embed SentProp in a larger context of semi-supervised lexicon expansion research. In section 3 we describe the corpus used for inducing polarities, our extensions to the SentProp framework and the setup for the classification experiments. In section 4 we describe findings from a qualitative viewpoint and discuss the results of our experiments. We conclude with some final remarks and possible future directions for the research.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1056–1064
Santa Fe, New Mexico, USA, August 20-26, 2018.

2 Related work

In this section we discuss related work to better embed our selected methods for lexicon induction. We first give a broad overview of work done in predicting word polarity. We then address the work that incorporates generated lexicons in polarity classification experiments. The last subsection will discuss research into lexicon adaptation in a domain-specific setting.

2.1 Polarity prediction

Early work in polarity prediction used linguistic rules to extract word relations. Hatzivassiloglou and McKeown (1997) extract conjunctions of adjectives to infer semantic oriëntation (SO; i.e. polarity). The resulting graph modelled similarity links ("and" conjunctions) and dissimilarity links ("but" conjunctions) and clustering was performed to further separate the positive from the negative polarity words. This approach yielded high accuracy (90%) in predicting word polarity, even when using only seven links per adjective.

Turney (2003) used statistical association between seed words and unseen words to induce SO. Specifically, the point-wise mutual information (PMI) of word co-occurrence statistics was used to find similarities of a word to a seed set. The evaluation setup compared induced SO to existing lexicons at different confidence intervals. The highest accuracy (98.20 percent) was found when comparing the top 25 percent most confident SO words induced from the largest corpus (10^{11} tokens). Using lower confidence words and smaller corpora harmed performance significantly.

Another vein in the lexicon expansion literature uses lexical graphs to draw word associations. Rao and Ravichandran (2009) use WordNet to leverage its rich relational structure. They construct a lexical graph by linking words through their synonymy and hypernymy relations. Polarities are propagated over the lexical graph using the label propagation algorithm from Zhu and Ghahramani (2002). Even when reducing the seed set to ten percent of its original size, this yields around 80 percent F-score for nouns and verbs. Adjectives suffer somewhat more from using fewer seed words.

2.2 Classification using induced polarity lexicons

Moving away from the intrinsic evaluation in the papers mentioned above, task-based evaluation is more relevant to our current work.

Velikovich et. al (2010) combine the statistical approach (Turney and Littman, 2003) with simplified label propagation (Zhu and Ghahramani, 2002; Rao and Ravichandran, 2009) to derive a web-based polarity lexicon. Co-occurrence statistics from a large web corpus ($4*10^9$ tokens) were used to construct a lexical graph. Polarities were then propagated over the graph starting from a positive and a negative seed set of respectively 187 and 192 words. The break with predefined lexical resources such as WordNet allows for finding new lexical entries in spelling variations, slang and multi-word expressions (they consider n-grams up to length 10). As such, the purpose of the work is not to compare induced polarities with existing lexicons, but rather to derive a lexicon with high coverage for sentiment analyses.

This results in a 178,104 phrase polarity lexicon which can be applied to a diversity of sentiment analysis subtasks. The web-derived lexicon outperforms two other automatically generated polarity lexicons, including one based on label-propagating WordNet, in a polarity classification and polarity ranking task. Velikovich et. al (2010) show that statistical similarity measures work even when the obtained corpus is very noisy. They also demonstrate the practical usefulness of a web-derived polarity lexicon.

One drawback that remains however, is the neglect of domain-specific information in these general-purpose lexicons. The next subsection will discuss work that has been done in generating lexicons for domain-specific purposes.

2.3 Domain adaptation

There is a large variety of recent work in obtaining domain-specific polarity lexicons. Most are concerned with generating a new lexicon for a target domain (Bross and Ehrig, 2013; Tai and Kao, 2013), while others aim to augment existing lexicons for domain-specific use with a variation of polarity clues.

Choi and Cardie (2009) use linear programming to find the most likely polarity label for words based on their occurrence in opinion expressions. Two types of clues; word-to-word relations within each expression and word-to-expression relations, are exploited to adapt a general-purpose lexicon. The adapted lexicon significantly improves expression classification with regards to the original lexicon.

Du et. al (2010) adapt the information bottleneck (IB) method to take domain-specific knowledge into account. This slightly changes the problem of adapting a general-purpose lexicon to fit a specific domain, because IB is used to adapt an in-domain lexicon to fit out-domain properties. To infer polarity values for words in the unseen domain, the authors use three clues: the relationships between out-of-domain words and out-of-domain documents, the relationships between out-of-domain words and in-domain words and the relationships between out-of-domain words and in-domain documents. To evaluate the domain adaptations, three domain-specific lexicons are manually composed and used as a basis for adaptation. When using IB to adapt an in-domain lexicon for out-domain purposes, polarity classification accuracy of the adapted lexicon is higher than most baselines that use only word occurrence statistics to infer in-domain polarities.

Lu et. al (2011) present an optimization framework that takes a range of signals such as sentiment scores from general lexicons, domain-specific document level sentiment, WordNet and linguistic clues to produce a unified domain-specific lexicon. The framework is tested on hotel reviews and printer reviews and significantly outperforms other lexicon-based methods in a polarity classification task.

2.4 SentProp

The work on domain adaptation for polarity lexicons mirrors the polarity lexicon induction literature in that there are a wide range of possible lexical resources to use as polarity clues and a wide range of induction methods that are all similarly feasible. SentProp implements word-embeddings with label propagation to offer a uniform approach for domain-specific lexicon generation (Hamilton et al., 2016). It further promises accurate performance even when using smaller corpora ((10^7 tokens). Hamilton et. al (2016) demonstrate their method by reproducing existing domain-specific lexicons. They outperform other induction methods including PMI (Turney and Littman, 2003) and graph propagation (Velikovich et al., 2010) in the domain-specific configuration and for standard English with a 1000x reduction of the corpus.

3 Methods

We elaborate on the SentProp method to evaluate it in domain-specific polarity classification tasks. Instead of pursuing state of the art performance in a target domain task, we seek to provide insight into the domain-specificity of polarities derived from the Sentprop method and the applicability of a generated domain-specific lexicon to an in-domain and an out-domain task.

3.1 Word embeddings

The data for the domain-specific input to our lexicon is derived from discussion boards on politics. We use two sources: politics.be and 9lives.be.

Politics.be features a dedicated political discussion board that has seen active use since 2002. It has more than 60,000 registered members and hosts close to 250,000 discussions.

9lives.be is a website originally dedicated to video game news, but also features a discussion board with general topics. We targeted the politics and actuality sub-forum for our data set. The sub-forum hosts close to 2,000 discussions on news and politics since 2004.

Using the Scrapy package for Python, we were able to download close to 20,000 discussion threads with $2x10^6$ posts. Specifications of the collection are reflected in Table 1.

In the choice for a word embedding method we followed Hamilton et. al. (2016), who find that vectors containing positive point-wise mutual information (PPMI) with singular-value decomposition (SVD) outperformed SGNS and GloVe embeddings in lexicon reproduction. This is further supported by Levy et. al. (2015) who also provide a useful overview of hyper parameters in each of the discussed methods. In particular this led us to run 500-dimensional SVD-embeddings with a context-window of

Source	Documents	Types	Tokens
Politics.be	1,778,101	824,978	68,596,562
9lives.be	434,261	315,266	28,006,269
Total	2,212,362	935,292	96,602,831

Table 1: Corpus specifications.

size six (Velikovich et al., 2010) with context-distribution smoothing and eigenvalue weighting (Levy et al., 2015).

3.2 General-purpose lexicon

We used the DuOMAn subjectivity lexicon as our general-purpose lexicon (Jijkoun and Hofmann, 2009). With 8,782 entries, it is the largest polarity lexicon for Dutch. Polarity scores range from -2 (very negative) to 2 (very positive) and contains 3,631 zero (neutral) values.

3.3 Seed sets

SentProp's default pipeline requires the word embeddings and a positive and negative seed word set. We extracted 10 positive and 10 negative seed words by sorting candidates from DuOMan by polarity first and by occurrences in our corpus second. This provided us with a selection of the most positive and negative words that are also used relatively frequently. We then manually filtered words that have any sort of semantic ambiguity. It is important that the seed words are unambiguously positive or negative because they serve as anchors in our graph (Hamilton et al., 2016).

Positive seeds (occurrences)		**Negative seeds** (occurrences)	
perfect (1.926)	leuk (1.368)	probleem (7.542)	slecht (2.834)
gelukkig (1.306)	effectief (1.304)	onzin (1.805)	onmogelijk (1.600)
correct (1.288)	winnen (887)	jammer (1.321)	dom (1.175)
top (789)	positief (771)	spijtig (919)	hypocriet (771)
succes (667)	prachtig (263)	kwaad (718)	discriminatie (709)

Table 2: Seed words as first sorted by polarity in the DuOMAn lexicon, then sorted by their occurrences in the discussion-board corpus. Some semantically ambiguous words were manually filtered like 're-delijk' which translates to 'reasonable' as an adjective but to 'fairly' as an adverb. All positive seed words have a polarity of 2.0 in the lexicon and all negative seed words have a polarity of -2.0 in the lexicon.

3.4 Lexicon expansion

We ran SentProp with default parameters. Starting from the seed words, positive and negative polarity labels are propagated from a word to their nearest neighbor using random walks. The nearest neighbors for any given word are determined by calculating the cosine-similarities between its embedding vector and the vectors of all other words. This theoretically composes a ranking of words with the most similar meanings to the given word. For our purposes we further imply that words with similar meanings have similar polarities. Although this is likely not plausible for all words, on an aggregate level we hope to at least rely on regularities tying positive words to positive words and negative words to other negative words.

The induced polarity score reflects the probability of a random walk reaching the word from the positive set versus reaching it from the negative set. A score closer to 1 reflects a high likelihood for the polarity to be positive and a score closer to 0 reflects the high likelihood for it be negative (Hamilton et al., 2016).

We considered a vocabulary that is much larger than the size of our general-purpose lexicon. In theory, all words in the vocabulary and their induced polarities could be added to DuOMAn, but we

No.	Configuration rule
#1	Apply all full polarity shifts
#2	Apply all weighted polarity shifts
#3	Apply full polarity shifts for x% largest shifts
#4	Apply full polarity shifts for x% smallest shifts
#5	Apply weighted polarity shifts for x% largest shifts
#6	Apply weighted polarity shifts for x% smallest shifts

Table 3: Polarity shift configuration rules. Each of the percentage-based configurations is applied with 5-percent increments to find the best settings. Full polarity shifts entail applying the induced polarity for any word, weighted polarity shifts take the average of the induced polarity and the polarity from the general-purpose domain and applying the average.

expect words with predominantly neutral polarities to only add noise to the lexicon. To only consider sentiment-bearing words and be able to plot them to the scale the general-purpose domain adheres to, we determined the limits to the amount of positive and negative words with two parameters:

$$num\_neg(X, \beta, \gamma) = \gamma\beta|X| \tag{1}$$

$$num\_pos(X, \beta, \gamma) = (1 - \gamma)\beta|X| \tag{2}$$

Where β is the fraction of vocabulary words that will be treated as sentiment-bearing, γ is the proportion of negative polarities to positive polarities in the vocabulary, and $|.|$ is the cardinality operator.

Our algorithm distributes the negative and positive words evenly over the granularity of the general-purpose lexicon. The words with the highest likelihood of being negative hence end up in the -2 group. There is no theoretical rule of thumb for determining the distribution of words over polarity labels and we expect this to be highly domain-dependent. We used a uniform distribution over the labels to make minimal assumptions.

3.5 Enhancing general entries

The algorithm also produces new labels for words already present in the general-purpose lexicon. We do not expect each of these induced polarities to better reflect word polarities than the values in the pre-existing lexicon. Correct label assignment depends on quality of the word embeddings and can change due to the simplified approach to label assignment we described above. We experimented with a few possible configuration rules that combine the information from the general-purpose lexicon and the induced polarities to decide on the size of polarity shifts. We considered applying weighted polarity shifts that average the polarity in the general-purpose lexicon and the induced polarity. We also considered only shifting polarities when the difference between the existing entry and the induced polarity is relatively large or when it is relatively small (Table 3).

3.6 Classification setup

For evaluation, we designed an in-domain and an out-domain binary polarity classification task. The in-domain data consists of over 4,000 polarity annotated tweets that mention the name of a Flemish politician. The out-domain data consists of product reviews from two online web shops: Bol.com and Coolblue.be. The products under review are mostly books, dvd's, games and some consumer electronics. The reviews have a rating that ranges from one to five stars. To match the in-domain binary polarity, we grouped one and two star reviews and four and five star reviews and assigned them negative and positive labels respectively.

The in-domain documents contain 2,126 positive and 2,126 negative tweets. The out-domain documents contain 3,959 positive reviews and the same number of negative reviews. We developed our in-domain lexicon on 90 percent of the twitter data and tested on the remaining 10 percent. We also tested the lexicon on the reviews to see if the enhanced lexicon improved classification in general, or mainly in the domain-specific setting.

Positive words			Negative words		
Word	Type	Translation	Word	Type	Translation
Geniaal	Regular addition	Genius	Verzuurd	Regular addition	Soured
Hoera	Regular addition	Hooray	Brainwashen	Regular addition	Brainwashing
Boeiend	Regular addition	Interesting	Oogkleppen	Regular addition	(horse) blinkers
Goei	Spelling variation	Good	Aggressief	Spelling variation	Aggressive
Perfekt	Spelling variation	Perfect	Groffe	Spelling variation	Rough
Intersant	Spelling variation	Interesting	Associaal	Spelling variation	Anti-social
Chapeau	Non-Dutch	Hats off	Debiel	Vulgarity	Imbecile
Style	Non-Dutch	Style	Dement	Vulgarity	Dementing
Zeitgeist	Non-Dutch	Zeitgeist	Pipo	Vulgarity	Idiot

Table 4: Example additions to the enhanced lexicon. The positive additions contain regular word additions, spelling variations and non-dutch words. The negative additions contain a lot of vulgarities.

For classification we implemented a majority-vote algorithm. Since we respect the granularity of the polarity labels in the original lexicon, words carry different weights. Instead of using binary values to vote on a document's polarity, we took the average of the occurring polarity values. If the average is a positive value we judged the document to have a positive label, and vice versa.

4 Results and discussion

We will now discuss our findings with regards to the best configuration for our proposed method, and the eventual classification accuracy this yielded on the proposed tasks. Moreover, in our qualitative evaluation we look at which words were added and which polarities adapted, and in which regard they reflect the domain-specificity of the generated lexicon.

4.1 Lexicon induction

We induced polarities for words with at least 100 occurrences in the corpus (n $\approx$ 1 per million). This resulted in a vocabulary of 26,563 words. We performed grid search for the parameters in equations (1) and (2) and the possible configurations from Table 3, comparing accuracy for each of the settings. The optimal β-value we found was 0.06, which signifies that 6 percent of all words in the vocabulary, which amounts to 1,303 words that were not already present in DuOMAn, were added to the general-purpose lexicon. The optimal γ-value we found was 0.58 which signifies that 58 percent of the added words are negative polarity words. Applying the average of the induced polarity and the lexicon polarity value for words that shift less than 25 percent works best for enhancing general entries (configuration #6 in Table 3). This resulted in 1,165 polarity shifts in the general lexicon. The resulting enhanced lexicon will be used for our qualitative inspection as well as the classification experiments on test data.

4.2 Qualitative evaluation

The 1,303 additions in the enhanced lexicon are a mixture of new words, spelling variations and words borrowed from other languages (Table 4). Some words are not inherently positive or negative but can be expected to occur mostly in a positive or negative context. The negative additions contain a lot of insults and other vulgarities. Generally speaking the additions seem more applicable in the domain of political discussion than in other domains.

The polarity shifts that were applied to further enhance the general-purpose lexicon had a positive effect on the classification performance for the development data, but since we only applied the subtle polarity shifts we cannot easily discern why the shifts improve results. Of the 3,292 entries in the DuOMAn lexicon that also appeared in the vocabulary, 993 had no polarity difference from the induced polarities, 1,165 had only a small polarity difference ($<$25%) and 1,134 had big polarity differences ($>$25%).

	Political tweets					Product Reviews				
	Positive		Negative		Overall	Positive		Negative		Overall
Lexicon	P	R	P	R	Acc.	P	R	P	R	Acc.
Majority baseline	50.0	0.0	50.0	100.0	50.0	50.0	0.0	50.0	100.0	50.0
DuOMan	65.0	54.0	60.6	**70.9**	62.4	**72.9**	79.7	76.5	**66.2**	**72.9**
Enhanced lexicon	**66.0**	**65.7**	**65.9**	66.2	**66.0**	62.6	**89.7**	**81.8**	46.4	68.0

Table 5: Precision, recall and classification accuracy for the original and the enhanced lexicon. The added out-domain task shows that the general-purpose domain was solely enhanced for the in-domain task.

In 1,078 of the 1,165 applied shifts, the polarity of a word is reduced. This leads us to believe the induced polarities had a lot of neutral values for words that were sentiment-bearing in the general-purpose lexicon. This weakening of sentiment in part of the lexicon did have a positive effect on classification.

4.3 Quantitative evaluation

We tested the enhanced lexicon on the held out tweets and the review data. The results are presented in Table 5. The improvement of the enhanced lexicon over DuOMan is restricted to the political tweets. In fact, with the additions and shifts applied for the political domain, classification on product reviews performs a lot worse overall. Each polarity class shows a precision-recall trade-off in the results. For classification of the political tweets, the additions and shifts in the lexicon yield a more optimal balance between the two classes, while for classification of the product reviews the balance is upset, incorrectly predicting far more positive labels than DuOMan.

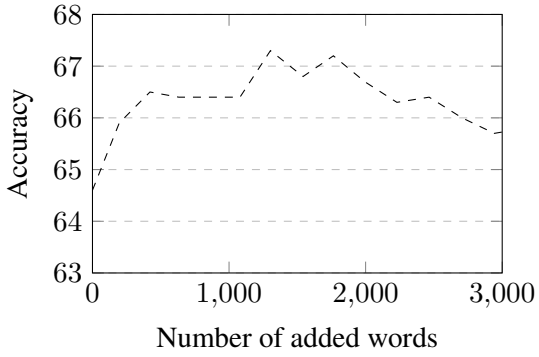


Figure 1: Adding more words from the vocabulary to the enhanced lexicon improves accuracy on the development data initially, but as more neutral words are added, the effect is diminished.

Our interpretation is that for the political domain specifically, the negative sentiment additions contain human traits rather than words that may describe a product. The positive additions seem more general and reusable, which explains the increased tendency of the enhanced lexicon to predict positive labels in the product reviews. When solely using a polarity lexicon to classify documents, the balance between the generality of positive versus negative words is very important.

4.4 Error analysis

Figure 2 shows some of the typical corrections that were made when applying the domain-specific lexicon as opposed to the general purpose lexicon. The first sentence gets the default negative label because none of its words are present in DuOMAn. By expanding the general purpose lexicon with more words, the correct positive label is assigned. This first type of correction accounted for 21 out of 28 total corrections on the test data (75%). The second sentence demonstrates a similar correction in which words were found in DuOMAn but the addition of new words corrected the label. This type of correction was present in 5 out of 28 corrections on the test data (about 18 %). The last type of correction is uncommon, but is caused by a reduction of polarity in the original lexicon where the default label is actually the correct label. This explained the remaining 2 corrections.

We look at errors caused by the adapted lexicon on the out-domain test set because they were far more numerous than the errors on the in-domain data. Our interpretation in section 4.3 is right: in 88 percent of the 656 additional misclassifications the (correct) negative label was flipped because of new positive

D:	boeiend	nieuwtje	gehoord	op	lezing	kris	peeters	:	#mer	zou	behoorlijk	wat	procedurefouten	bevatten				
	-	-	-	-	-	-	-	-	-	-	-	-	-	-				
A:	boeiend	nieuwtje	gehoord	op	lezing	kris	peeters	:	#mer	zou	behoorlijk	wat	procedurefouten	bevatten				
	+2.0	-	-	-	+1.0	-	-	-	-	-	-	-	-	-				
D:	zeker	niet	tegenover	bart	de	wever	,	die	het	overigens	schitterend	doet	op	televisie	.			
	+1.0	-1.0	-	-	-	-	-	-	-	-	-	-	-	-	-			
A:	zeker	niet	tegenover	bart	de	wever	,	die	het	overigens	schitterend	doet	op	televisie	.			
	+1.0	-1.0	-0.5	-	-	-	-	-	-	-	+1.5	-	-	-	-			
D:	als	rik	torfs	zonder	werk	valt	,	kan	ie	nog	steeds	proberen	te	solliciteren	bij	bond	zonder	naam	.
	-	-	-	-	0.0	-	-	0.0	-	-	-	0.0	-	-	-	-	-	1.0	-
A:	als	rik	torfs	zonder	werk	valt	,	kan	ie	nog	steeds	proberen	te	solliciteren	bij	bond	zonder	naam	.
	-	-	-	-	0.0	-	-	0.0	-	-	-	0.0	-	-	-	-	-	0.0	-

Figure 2: Classifications from the DuOMAn lexicon (D) and respective corrections made by the adapted lexicon (A) on the political tweets test set.

polarity word in the review. Similarly, 79 percent of the 268 corrections on the review data was made this way.

There were specific additions to the lexicon that caused many misclassifications, for example *'hoofdstuk'* (chapter), *literatuur* (literature) and *'uitgelezen'* (finished reading; but can also mean excellent) all have positive polarities in the domain-specific lexicon but cause many book reviews to be wrongfully classified as positive. Personal traits such as *'asociale'* (anti-social), *egocentrische* (selfish) and *'houding'* (attitude) caused problems when book or movie plots were described in the review. The same is true for *mannetje* (little man), *goal* and *pass* but for video game reviews.

With regards to domain-specificity these specific word examples show us that there are many (sub)domains within the product review data to be considered. The lexicon adaptation method we propose in this paper can generate a lexicon for use on any level, given there is enough in-domain textual data available. Beyond polarity classification, generated lexicons may be used to study sentiment differences between domains.

5 Conclusion

We have shown that enhancing a lexicon with SentProp for domain-specific use improves its accuracy in an in-domain classification task. Not all induced polarities should be considered when using majority voting for polarity classification. The words with the most extreme polarities make for the best additions to the lexicon (see Figure 1).

When applying polarity shifts, only applying small shifts that average the general-purpose lexicon polarity with the induced polarity had a positive effect on classification accuracy. A large difference between an induced polarity and the general-purpose domain is likely due to some very specific context in the word embeddings. We ignored these large differences and used the general-purpose polarity instead.

We added 1,303 new words to the general-purpose lexicon and applied 1,165 small polarity shifts to its entries. This gave us the best enhanced lexicon for classification on political tweets. We also tested the enhanced lexicon on out-domain product reviews and found that this yielded worse accuracy than the general-purpose domain, showing that the additions and adaptations were domain-specific and do not translate well to other domains.

In future work, we would like to compare lexicons enhanced on different corpora. This should provide more insight into the word additions and shifts that are particular for a domain. We also expect the balance between positive and negative words to shift depending on the domain. Another important addition to the current work would be a comparison of SentProp to other polarity induction methods for domain-specific classification.

SentProp has been shown to accurately reproduce domain-specific lexicon (Hamilton et al., 2016). We have elaborated on the existing body of work by demonstrating its capability to enhance a general-purpose domain for domain-specific classification.

References

Juergen Bross and Heiko Ehrig. 2013. Automatic construction of domain and aspect specific sentiment lexicons for customer review mining. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1077–1086. ACM.

Yejin Choi and Claire Cardie. 2009. Adapting a polarity lexicon using integer linear programming for domain-specific sentiment classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 590–598. Association for Computational Linguistics.

Weifu Du, Songbo Tan, Xueqi Cheng, and Xiaochun Yun. 2010. Adapting information bottleneck method for automatic construction of domain-oriented sentiment lexicon. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 111–120. ACM.

William L Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2016, page 595. NIH Public Access.

Vasileios Hatzivassiloglou and Kathleen R McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the association for computational linguistics*, pages 174–181. Association for Computational Linguistics.

Valentin Jijkoun and Katja Hofmann. 2009. Generating a non-english subjectivity lexicon: Relations that matter. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 398–405. Association for Computational Linguistics.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Yue Lu, Malu Castellanos, Umeshwar Dayal, and ChengXiang Zhai. 2011. Automatic construction of a context-aware sentiment lexicon: an optimization approach. In *Proceedings of the 20th international conference on World wide web*, pages 347–356. ACM.

Delip Rao and Deepak Ravichandran. 2009. Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 675–682. Association for Computational Linguistics.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307.

Yen-Jen Tai and Hung-Yu Kao. 2013. Automatic domain-specific sentiment lexicon generation with label propagation. In *Proceedings of International Conference on Information Integration and Web-based Applications & Services*, page 53. ACM.

Peter D Turney and Michael L Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems (TOIS)*, 21(4):315–346.

Leonid Velikovich, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan McDonald. 2010. The viability of web-derived polarity lexicons. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 777–785. Association for Computational Linguistics.

Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. *CMU-CALD-02-107*.

An Operation Network for Abstractive Sentence Compression

Naitong Yu Jie Zhang* Minlie Huang† Xiaoyan Zhu
State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Dept. of Computer Science and Technology, Tsinghua University, Beijing, PR China
*Microsoft Search Technology Center
ynt12@mails.tsinghua.edu.cn zhanjie@microsoft.com
aihuang@tsinghua.edu.cn zxy-dcs@tsinghua.edu.cn

Abstract

Sentence compression condenses a sentence while preserving its most important contents. Delete-based models have the strong ability to delete undesired words, while generate-based models are able to reorder or rephrase the words, which are more coherent to human sentence compression. In this paper, we propose Operation Network, a neural network approach for abstractive sentence compression, which combines the advantages of both delete-based and generate-based sentence compression models. The central idea of Operation Network is to model the sentence compression process as an editing procedure. First, unnecessary words are *deleted* from the source sentence, then new words are either *generated* from a large vocabulary or *copied* directly from the source sentence. A compressed sentence can be obtained by a series of such edit operations (*delete*, *copy* and *generate*). Experiments show that Operation Network outperforms state-of-the-art baselines.

1 Introduction

Sentence compression is the natural language generation (NLG) task of condensing a sentence while preserving its most important contents. It can also be viewed as a sentence-level summarization task. With the rapid growth of the web contents in recent years, summarization techniques such as sentence compression are becoming more and more important since these techniques can greatly reduce the information overload on the web. Sentence compression can benefit a wide range of applications, especially those on mobile devices which have restricted screen spaces. Sentence compression models can be broadly classified into two categories: delete-based models and abstractive models. Delete-based approaches remove unimportant words from the source sentence and generate a shorter sentence by stitching the remaining fragments together. On the contrary, abstractive models consider operations beyond word deletion, such as reordering, substitution and insertion. Abstractive sentence compression models produce a reform of the source sentence from scratch, thus the results produced by abstractive sentence compression models are more expressive. Obviously, abstractive sentence compression is much harder than delete-based sentence compression, since it needs deeper understanding of the source sentence.

Delete-based sentence compression treats the task as a word deletion problem: given an input *source* sentence x $= x_1, x_2, ..., x_n$ (where x_i stands for the ith word in the sentence x), the goal is to produce a *target* sentence by removing any subset of words in the source sentence x (Knight and Marcu, 2002). Delete-based sentence compression has been widely explored across different modeling paradigms, such as noisy-channel model (Knight and Marcu, 2002; Turner and Charniak, 2005), large-margin learning (McDonald, 2006; Cohn and Lapata, 2007), integer linear programming (Clarke and Lapata, 2008) and variational auto-encoder (Miao and Blunsom, 2016). In delete-based sentence compression models, only *delete* operations are allowed, thus the order of the remaining words can not be changed. These constraints make delete-based sentence compression a relatively easier task. However, in spite of the strong ability of deleting undesired words, delete-based models are not able to rephrase the words, which is far

†Corresponding author: Minlie Huang (aihuang@tsinghua.edu.cn)

Proceedings of the 27th International Conference on Computational Linguistics, pages 1065–1076
Santa Fe, New Mexico, USA, August 20-26, 2018.

from human sentence compression. For example, in human sentence compression, the word "remove" can replace the phrase "get rid of" under some particular circumstance, but this substitution can not be accomplished by delete-only models.

Abstractive sentence compression has the ability to reorder, substitute words or rephrase, thus it is more coherent to human sentence compression. Due to the difficulty of abstractive sentence compression, there was only a limited number of work on the task (Cohn and Lapata, 2008; Cohn and Lapata, 2013; Galanis and Androutsopoulos, 2011; Coster and Kauchak, 2011a). However, with the recent success of the sequence-to-sequence (Seq2Seq) model, the task of abstractive sentence compression has become viable. Seq2Seq has an encoder-decoder architecture where the encoder encodes the input sequence into hidden states, and the decoder then generates the output sequence from the hidden states. The attention mechanism (Bahdanau et al., 2014), which can align the output sequence with the input sequence automatically, boosts the performance of Seq2Seq significantly. A number of abstractive sentence compression work has been built upon the Seq2Seq architecture with attention mechanism, such as (Chopra et al., 2016; Wubben et al., 2016; Nallapati et al., 2016; See et al., 2017). These abstractive models (which will be termed *generate-based* models hereafter) have the ability to reorder words or rephrase. However, none of these models consider explicit word deletion. As Coster and Kauchak (2011b) pointed out, deletion is a frequently occurring phenomena in sentence compression dataset. Coster and Kauchak (2011a) imposed delete operation on their sentence compression model and improved the performance significantly. Thus, deletion is also very important for abstractive sentence compression task.

Inspired by previous work, we propose an Operation Network for abstractive sentence compression, which combines the advantages of both delete-based and generated-based sentence compression. The central idea is to model the sentence compression process as an editing procedure. We not only enable the model with a strong ability of word deletion inspired from delete-based models, but also endow the model with the ability of word reordering and word rephrasing inspired from generate-based models.

With a series of editing operations, the source text can be transformed into a condensed version of itself. There are three kinds of editing operations in our model: *delete*, *copy* and *generate*. Given a source text as input, first the *delete* operations remove unnecessary words from the source text yet retain the important content. Then, the summary is constructed by the *copy* and *generate* operations. *Copy* operations duplicate words directly from the selected source text, while *generate* operations produce words which are not in the source texts. Our model is built upon the Seq2Seq framework, and can be trained in an end-to-end fashion.

To summarize, the contributions of this paper are as follows:

- We propose Operation Network, a neural framework for abstractive sentence compression, which models the sentence compression task as a series of editing operations.
- Our Operation Network combines the advantages of both delete-based and generate-based sentence compression. The model is equipped with a strong ability of not only word deletion, but also word reordering and word rephrasing. Experiments show that our model outperforms the baselines.

2 Task Definition

We formulate the problem of abstractive sentence compression discussed in this paper as follows: Given a source sentence $X = (x_1, x_2, ..., x_n)$ as input, the goal is to generate a target sentence $Y = (y_1, y_2, ..., y_m)$, $m < n$, which is a condensed version of X with good readability and retains the most important information in X. Essentially, the model estimates the conditional probability as follows:

$$P(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{t=1}^{m} P(y_t|y_{<t}, \boldsymbol{X}). \tag{1}$$

3 Background: Seq2Seq Model

Our model is built upon the general Seq2Seq model (Sutskever et al., 2014), which is also called the encoder-decoder model. The model consists of an encoder and a decoder. The encoder takes as input

a source sequence $\boldsymbol{X} = (x_1, x_2, ..., x_n)$, and outputs a sequence of hidden states $\boldsymbol{h} = (h_1, h_2, ..., h_n)$. The decoder takes $\boldsymbol{h}$ as input and outputs the target sequence $\boldsymbol{Y} = (y_1, y_2, ..., y_m)$.

In both encoder and decoder, we use gated recurrent unit (GRU) (Cho et al., 2014; Chung et al., 2014) as the basic unit. We use a bi-directional RNN (Schuster and Paliwal, 1997) as encoder. A bi-directional RNN contains two distinct RNNs, a forward RNN and a backward RNN. Given the input $\boldsymbol{X} = (x_1, x_2, ..., x_n)$ and the embedding lookup table e, the forward RNN reads the input in the left-to-right direction, resulting a sequence of forward hidden states $\overrightarrow{\boldsymbol{h}} = (\overrightarrow{h}_1, \overrightarrow{h}_2, ..., \overrightarrow{h}_n)$, where $\overrightarrow{h}_t = \text{GRU}(\overrightarrow{h}_{t-1}, e(x_t))$. Similarly, the backward RNN reads the input in the reversed direction and outputs $\overleftarrow{\boldsymbol{h}} = (\overleftarrow{h}_1, \overleftarrow{h}_2, ..., \overleftarrow{h}_n)$. At each time step, we concatenate the hidden states of the corresponding forward and backward RNNs and obtain the encoder hidden states $\boldsymbol{h} = (h_1, h_2, ..., h_n)$, where $h_t = [\overrightarrow{h}_t; \overleftarrow{h}_t]$, $t = 1, ..., n$ and $[A; B]$ denotes vector concatenation.

In the decoder for general Seq2Seq model, another GRU is used for updating the decoder hidden states. Given a context vector c_t and the previously decoded word y_{t-1}, the decoder hidden states are updated as follows:

$$s_t = \text{GRU}(s_{t-1}, [c_t; e(y_{t-1})]) \tag{2}$$

The context vector c_t is designed to dynamically attend on key information of the input sentence during the decoding procedure (Bahdanau et al., 2014). c_t is calculated as a weighted sum of the encoder hidden states:

$$c_t = \sum_{k=1}^{n} \alpha_{tk} h_k \tag{3}$$

$\boldsymbol{\alpha}_t = (\alpha_{t1}, \alpha_{t2}, ..., \alpha_{tn})$ is called the *attention distribution*. It can be viewed as a probability distribution over the source words, which tells the decoder where to attend to produce the next word. The attention distribution is calculated as follows:

$$\alpha_{tk} = \text{softmax}(e_{tk}) = \frac{\exp(e_{tk})}{\sum_{j=1}^{n} \exp(e_{tj})} \tag{4}$$

$$e_{tk} = v_a^\top \tanh(W_a s_{t-1} + U_a h_k + b_a) \tag{5}$$

where v_a, W_a, U_a and b_a are trainable parameters.

The vocabulary distribution P_t^{voc} at time step t is calculated as follows:

$$P_t^{voc} = \text{softmax}(W_o'(W_o[s_t; c_t] + b_o) + b_o') \tag{6}$$

where W_o', W_o, b_o and b_o' are trainable parameters. P_t^{voc} is a probability distribution over all words in the vocabulary V, and the output word y_t at time step t is calculated as follows:

$$y_t = \arg\max_w P_t^{voc}(w), w \in V \tag{7}$$

During training, the loss for time step t is the negative log likelihood of the target word y_t^*: $\text{loss}_t = -\log P_t^{voc}(y_t^*)$. The overall loss function for the whole word sequence is: $\text{loss} = \frac{1}{T_Y} \sum_{t=1}^{T_Y} \text{loss}_t$, where T_Y is the length of the target word sequence $\boldsymbol{Y}$.

4 Operation Network

In this section, we introduce our model, called as the Operation Network, to address the task of abstractive sentence compression. In Operation Network, we consider the transformation from the source sentence to the target sentence as a series of operations. We use three kinds of different operations: delete, copy and generate. To enable the model to perform these three types of edit operations, we use two distinct decoders, one for delete and the other for copy and generate. Specifically, we use the same encoder as in the general Seq2Seq model, but employ a delete decoder and a copy-generate decoder. After the input

sentence is encoded by the encoder, the delete decoder will first delete unnecessary words from the input sentence, then the copy-generate decoder will produce the output sentence either by copying from the choices of the delete decoder, or generating from a fixed vocabulary. The overall architecture of our model is shown in Figure 1.

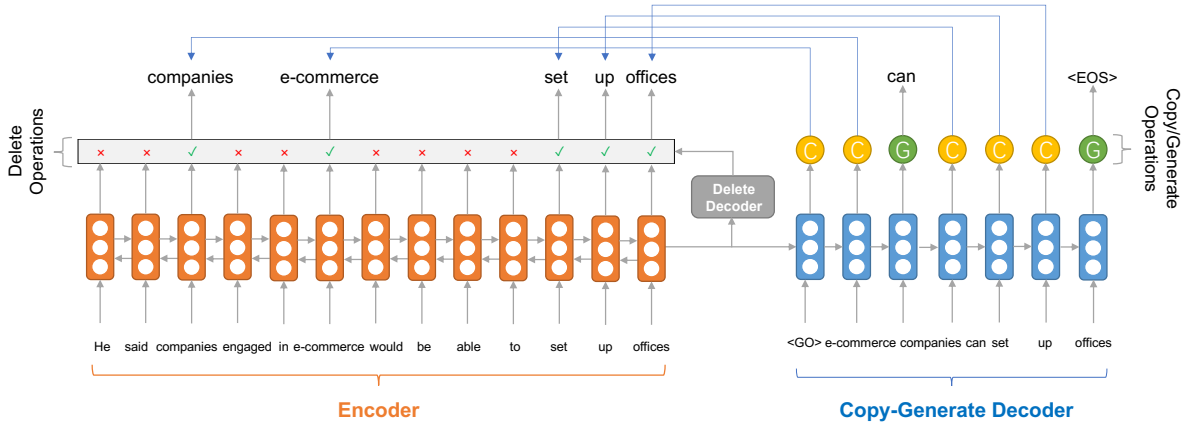


Figure 1: Operation Network, which consists of an encoder, a delete decoder and a copy-generate decoder.

4.1 Delete Decoder

The delete decoder "deletes" all unimportant words from the sentence. It takes as input the sequence of the encoder hidden states $\boldsymbol{h} = (h_1, h_2, ..., h_n)$, the text sequence $\boldsymbol{X} = (x_1, x_2, ..., x_n)$, and outputs a sequence $\boldsymbol{d} = (d_1, d_2, ..., d_n)$, which has the same length n. $d_i \in [0, 1], i \in 1, 2, ..., n$. If d_i is close to 0, then the corresponding word x_i tends to be deleted. Otherwise if d_i is close to 1, then the corresponding word x_i tends to be kept. The output sequence $\boldsymbol{d}$ will be used together with the copy-generate decoder which we will discuss in Section 4.2. The architecture of the delete decoder is shown in Figure 2.

For decoding step t in the delete decoder, we feed the decoder with the embedding of word x_t, the previous decoder output d_{t-1} and the context vector c_t. The delete decoder states are updated as follows:

$$s_t = \text{GRU}(s_{t-1}, [c_t; e(x_t); d_{t-1}]) \tag{8}$$

$$d_t = \sigma(W_d s_t^d + b_d) \tag{9}$$

where σ is the sigmoid function, c_t is the context vector calculated in the same way as Equation 3, e denotes the word embedding table, and W_d and b_d are trainable parameters.

The output of the delete decoder, $\boldsymbol{d} = (d_1, d_2, ..., d_n)$, is fed into the copy-generate decoder for further calculation.

4.2 Copy-Generate Decoder

The Copy-Generate decoder produces output the compressed sentence word by word, and the output words are either copied from the input words which are filtered by the delete decoder, or generated with a fixed vocabulary. In other words, our model integrates copy, generate, delete operations together to produce the output sequence.

We implement the Copy-Generate decoder as a hybrid network between the basic Seq2Seq network and a pointer network (Vinyals et al., 2015). The structure of the Copy-Generate decoder is close to CopyNet (Gu et al., 2016), Pointer Softmax (Gulcehre et al., 2016) and Pointer-Generator (See et al., 2017). However, the Copy-Generate decoder model has some unique characteristic designed for abstractive sentence compression task. The major difference is that our Copy-Generate decoder incorporates the result of the delete decoder, to make sure that unnecessary words will not be copied back by accident. Another difference from the other models is that our model generates explicit operations. During the

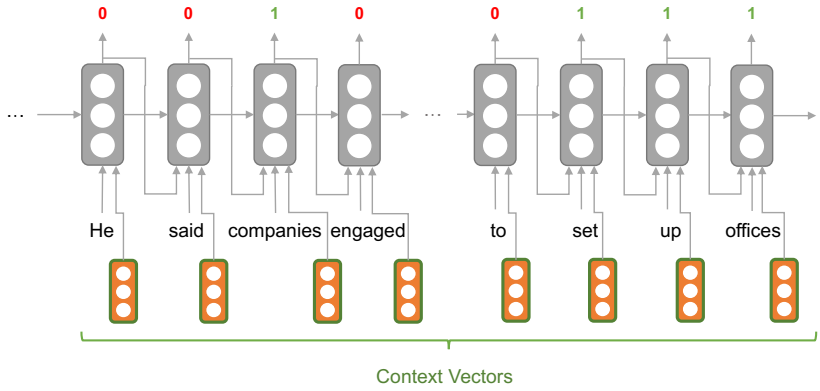


Figure 2: Delete Decoder.

training procedure, we also add supervision on these operations (see Section 4.3). We find that explicit operation supervision can lead to better results. Last, unlike Gu et al. (2016) and Gulcehre et al. (2016), where copy operations are only used for handling UNK tokens (special tokens which stand for out-of-vocabulary words) or named entities, in our model, copy operations are triggered much frequently as long as the output word can be copied from the input sentence. This is useful especially when the training dataset is relatively small.

The Copy-Generate decoder takes as input the encoder hidden states $\boldsymbol{h} = (h_1, h_2, ..., h_n)$ and the attention distributions $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_n)$, and outputs the target sentence as a sequence of words $\boldsymbol{Y} = (y_1, y_2, ..., y_m)$. At each step of decoding, a switch network is used to decide whether to use copy mode or generate mode. If copy mode is chosen, the word is copied directly from the source words (filtered by the delete decoder). We use the attention distributions to sample the source words. The output of the delete decoder is used to modify the attention distributions to filter out unwanted words. Otherwise, if generate mode is chosen, the word is generated from the fixed vocabulary. Specifically, for words that are neither in the source sentence nor in the vocabulary, the switch network will choose generate mode and UNK tokens will be produced from the decoder.

The Copy-Generate decoder use another distinct GRU to update its decoder hidden states as follows:

$$s'_t = \text{GRU}(s'_{t-1}, [c_t; e(y_{t-1})]) \tag{10}$$

where s'_t is the decoder hidden state at time step t, c_t is the context vector, e denotes the word embedding table, and y_{t-1} is the previous decoded output.

At each time step t, the switch network calculates *generate probability* as follows:

$$g_t^{gen} = \sigma(W_z[s'_t; c_t] + b_z) \tag{11}$$

g_t^{gen} acts as a soft switch to choose between generate mode and copy mode.

In generate mode, the target word is sampled from the vocabulary distribution P_t^{voc} (see Equation 6).

In copy mode, first we use the outputs of the delete decoder $\boldsymbol{d}$ (see Section 4.1) to mask and re-normalize the attention distribution $\boldsymbol{\alpha_t}$:

$$\boldsymbol{\alpha'_t} = \frac{1}{\sum_{i=1}^{n} d_i \alpha_{ti}} (d_1\alpha_{t1}, d_2\alpha_{t2}, ..., d_n\alpha_{tn}) = (\alpha'_{t1}, \alpha'_{t2}, ..., \alpha'_{tn}). \tag{12}$$

Then the target word is sampled from the modified attention distribution $\boldsymbol{\alpha'_t}$.

For each sentence, let the *extended vocabulary* V' denotes the union of the vocabulary V and all words appearing in the source sentence, then the final probability distribution over the extended vocabulary V' is:

$$P_t(w) = g_t^{gen} P_t^{voc}(w) + (1 - g_t^{gen}) \sum_{k:w=x_k} \alpha'_{tk}, w \in V' \tag{13}$$

where x_k denotes the kth words in the source sequence $\boldsymbol{X}$. The output word y_t at time step t is sampled from the final distribution P_t:

$$y_t = \arg\max_w P_t(w), w \in V' \tag{14}$$

4.3 Loss Function

The total loss consists of two parts: the loss from delete decoder and the loss from copy-generate decoder.

4.3.1 Delete Decoder Loss

The delete decoder loss measures the difference between the target deletion sequence and the actual deletion sequence predicted by the delete decoder. The delete decoder loss is calculated as follows:

$$delete\_loss = \frac{1}{T_X}\sum_{t=1}^{T_X}\Big(-\hat{d}_t \log d_t - (1-\hat{d}_t)\log(1-d_t)\Big) \tag{15}$$

where T_X is the length of the source word sequence X, and $\hat{\boldsymbol{d}}$ is the target deletion sequence, $\hat{d}_t = 0$ if word x_t should be deleted, otherwise, if word x_t should be kept, $\hat{d}_t = 1$.

4.3.2 Copy-Generate Decoder Loss

At each time step t, the copy-generate decoder loss can be divided into three parts: the switch loss, the copy loss and the generate loss. The switch loss measures the difference between the target switch and the predicted switch by the model. The copy loss measures the difference between the target attention distribution and the actual attention distribution predicted by the model. And the generate loss measures the loss between the target vocabulary distribution and the generated vocabulary distribution at that time step. The losses at time step t is calculated as follows:

$$\begin{aligned} switch\_loss_t = &- \hat{g}_t^{gen} \log g_t^{gen} \\ &- (1-\hat{g}_t^{gen}) \log(1-g_t^{gen}) \end{aligned} \tag{16}$$

$$copy\_loss_t = - (1-\hat{g}_t^{gen}) \log \sum_{k:y_t^*=x_k} \alpha'_{tk} \tag{17}$$

$$gen\_loss_t = - \hat{g}_t^{gen} \log P_t^{voc}(y_t^*) \tag{18}$$

where $\hat{g}_t^{gen} = 1$ if target is in *generate mode* at time step t, otherwise, if target is in *copy mode*, $\hat{g}_t^{gen} = 0$. y_t^* denotes the target word at time step t.

The overall copy-generate loss function for the target compressed sentence is as follows:

$$copy\_gen\_loss = \frac{1}{T_Y}\sum_{t=1}^{T_Y}(switch\_loss_t + copy\_loss_t + gen\_loss_t) \tag{19}$$

where T_Y is the length of the target word sequence $\boldsymbol{Y}$.

5 Experiments

In this section, we introduce the experiments for abstractive sentence compression with our proposed Operation Network. First, we present a brief description of the dataset, and the pre-processing procedure in our experiments. Then, we introduce baselines that are compared with our model. Next, we introduce parameters of our models in the experiments. At last we present and analyze the experiment results.

5.1 Dataset

We adopt the dataset provided by Toutanova et al. (2016) for our experiments. It's a manually-created, multi-reference dataset for sentence and short paragraph compression. It contains 6,169 source texts with multiple compressions (26,423 pairs of source and compressed texts), consisting of business letters,

news journals, and technical documents sampled from the Open American National Corpus (OANC[1]). Of all the source texts, 3,769 are single sentences and the rest are 2-sentence short paragraphs. Each pair of the source and compressed text is aligned by the state-of-the-art monolingual aligner Jacana (Yao et al., 2013). The dataset is split into a training set (21,145 pairs), a validation set (1,908 pairs) and a test set (3,370 pairs). The dataset is available online[2].

The alignment information together with the pair of source text and compressed text are used for generating operation sequences for our experiments. Specifically, for each pair of the source and compressed text, a delete operation sequence and a copy/generate sequence are generated. The delete operation sequence is a sequence of *delete*/*retain* tokens which has the same length with the source text. For each word in the source text, if the word exists in the compressed text or is aligned by the aligner, the corresponding token in the delete operation sequence is *retain*. On the other hand, if the word neither exists in the compressed text nor aligned by the aligner, the corresponding token in the delete operation sequence should be *delete*. The copy/generate operation sequence is a sequence of *copy*/*generate* tokens which has the same length with the compressed text. For each word in the compressed text, if the word is aligned by the aligner and is the same as its counterpart, then the corresponding token in the sequence is *copy*, which means this word is copied from the source text. If the word is not aligned or not the same as its counterpart, then the corresponding token in the sequence is set to *generate*, which means this word is generated from the vocabulary. The reason we didn't put *delete*, *copy* and *generate* operations in a single sequence is that *delete* operations are source-text-based operations while *copy* and *generate* operations are target-text-based operations. It is consistent with that the delete operation sequence has the same length as the source text and the copy/generate operation sequence has the same length as the compressed text.

5.2 Baselines

We compared the abstractive sentence compression results generated by our model (Operation Network) with those by two baselines. These baselines consist of a generate-only model (Seq2Seq) and a generate + copy model (Pointer-Generator). The details of the baselines are described as follows:

- **Seq2Seq:** Seq2Seq is an generate-only model similar to the model described by Nallapati et al. (2016). It uses the same bi-directional RNN encoder and attention mechanisms as our model. The decoder of Seq2Seq model can only generate words from the fixed vocabulary.

- **Pointer-Generator:** Pointer-Generator is a model proposed by See et al. (2017). It uses a hybrid pointer-generator network that can copy words from source text or generate words directly from a large vocabulary. This model also use coverage to keep track of what has been summarized, which discourage repetition. Refer to See et al. (2017) for more details.

5.3 Experiment Parameters

For all experiments, we use 300-dimensional word embeddings. We also use the pre-trained *GoogleNews* vectors to initialize the embeddings. For words do not exist in *GoogleNews* vectors, we initialize them randomly. The vocabulary size is set to 20,000. We also use the large vocabulary tricks described by Jean et al. (2014). The sampled vocabulary size is set to 4,096. The hidden state of a single GRU is 256-dimensional. To overcome overfitting, we also imposed dropout on the input, output and state vector of the GRUs. The dropout probability is set to 0.5.

We trained the models on a single Nvidia Titan X GPU with a batch size of 32. During training, we used Stochastic Gradient Descent with an initial learning rate of 0.15, and applied the exponential decay to the learning rate as the training process proceeds. At test time, the output sentences are decoded using beam search with beam size 4.

[1] `http://www.anc.org/data/oanc`

[2] The dataset can be downloaded from the project's website
`https://www.microsoft.com/en-us/research/project/intelligent-editing/`

5.4 Experiment Metric

In the experiments, we compare our model and the baselines with the following metrics.

Compression Ratio: The common assumption in compression research is that the system can make the determination of the optimal compression length. Thus, compression ratios can vary drastically across systems. Different systems can be compared only when they are compressing at similar ratios (Napoles et al., 2011). Compression ratio is defined as:

$$\text{CompRatio} = \frac{\text{\# of tokens in compressed text}}{\text{\# of tokens in source text}} \tag{20}$$

ROUGE: We evaluated our models with the standard ROUGE metric proposed by Lin (2004). ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. It is commonly used for measuring the quality of the summary by comparing computer-generated summaries to reference summaries generated by humans. The basic idea of ROUGE is to count the number of overlapping units such as n-grams, word sequences, and word pairs between computer-generated summaries and the reference summaries. In our experiments, we considered ROUGE-1, ROUGE-2 and ROUGE-L (which respectively measures the word-overlap, bigram-overlap, and longest common sequence between the reference summary and the summary to be evaluated).

BLEU: We also report BLEU scores of the baseline models and Operation Network on the test dataset. BLEU is proposed by Papineni et al. (2002), and is usually used for automatic evaluation of statistical machine translation systems. However, it can also be used for evaluating sentence compression task (Napoles et al., 2011). We use the multi-bleu script[3] for BLEU score calculation.

5.5 Result Analysis

We present the average ratios of the operations in Operation Network's outputs and the human-written references on the test dataset in Figure 3. Note that the delete ratio is calculated as the number of delete operations divide by the number of tokens in *source* text, and the copy and generate ratio is calculated as the number of copy and generate operations divided by the number of tokens in *compressed* text, respectively. From the figure we can see that Operation Network can utilize the operation information to help accomplish the sentence compression task. The average delete ratio in Operation Network is lower than in human reference, since there are multiple references for one source text in the dataset, and the Operation Network learns to delete a token only it is deleted in most of the references. The average copy ratio is higher in Operation Network than in human reference, this may suggest that Operation Network tends to copy words directly from the source text whenever it is possible.

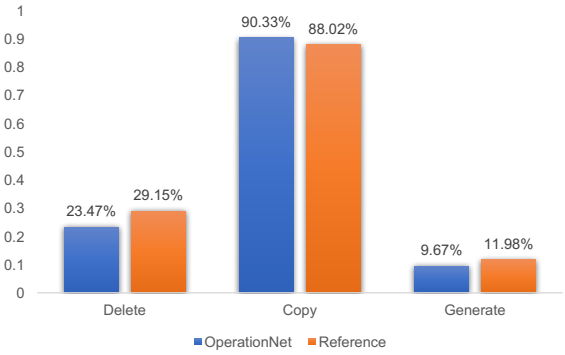


Figure 3: The average ratios of *delete*, *copy* and *generate* operations of Operation Network and human-written references on the test dataset.

Table 1 shows the average compression ratios (%), ROUGE and BLEU scores of the baseline models and Operation Network on the test dataset. The second column shows the average compression ratios.

[3] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

The average compression ratio is calculated as the average of the compression ratios of all the compressed text outputs in the test dataset. We can see that the average compression ratios of both our model and the other two baselines are similar. Thus, the comparison between our model and the baseline models is fair. Since we have employed the delete decoder in Operation Network, the average compression ratio of our model is lower than the other baselines.

	CompRatio (%)	ROUGE-1	ROUGE-2	ROUGE-L	BLEU
Seq2Seq	69.39	30.05	10.42	26.87	18.32
Pointer-Generator	66.59	35.34	16.57	32.56	25.09
OperationNet	65.53	**36.21**	**17.43**	**33.72**	**26.30**

Table 1: Average compression ratios (%), ROUGE and BLEU scores of the models on the test dataset. Statistically significant improvements ($p < 0.01$) over the baselines are demonstrated by bold fonts.

From Table 1 we can see that the generate-only model (Seq2Seq) performs poorly compared to its counterparts. This is due to the fact that the Seq2Seq model can only generate words from a fixed vocabulary. When the training dataset is relatively small (about 21k pairs of source and compressed text), the model may not be trained adequately. We tried the same models with random-initialized word embeddings and with *GoogleNews*-vector-initialized word embeddings and it turned out that random-initialized word embeddings lead to worse performance. The copy-and-generate model (Pointer-Generator) performs much better than the generate-only model. This is because the copy operations allow to copy words directly from the source texts and the attention distribution is much smaller than the vocabulary distribution and easier to train. With the addition of delete operation, our model (Operation Network) outperforms the other baselines in terms of all three ROUGE metrics and the BLEU metric. Results show that the delete operations can effectively filter out unnecessary words from the source texts and lead to better performance. This can also show that the Operation Network is suitable for the abstractive sentence compression task.

In order to evaluate the quality of the summaries produced by our model and the other baselines, we ask annotators to do a score evaluation on some aspects of the summaries. Specifically, we ask them to score the grammaticality and non-redundancy of the summaries. Annotators are instructed to read the summary carefully and rate each aspect with scores matching the quality of the corresponding aspect. Each aspect is rated with a score from 0 (bad) to 5 (excellent). We randomly sample 100 samples from the test set for evaluation. Each summary generated by our model or baselines is rated by at least 5 annotators. The summaries are randomly reordered and model information is anonymous to the annotators. The evaluation result is shown in Table 2.

	Grammaticality	Non-Redundancy
Seq2Seq	2.53	2.24
Pointer-Generator	**3.46**	3.58
OperationNet	3.23	**3.64**

Table 2: Results of aspect evaluation.

The results in Table 2 show that combined with the delete decoder, our model can effectively delete unimportant contents without losing much grammar quality of the text. This exactly matches what we expect from our model. Pointer-Generator model performs better on grammaticality aspect than our model, however, our model outperforms the other baselines on non-redundancy aspect. The basic Seq2Seq model performs poorly on both grammaticality aspect and non-redundancy aspect.

6 Related Work

Delete-based sentence compression. A large number of work is devoted to delete-based sentence compression. Jing (2000) presented a system that used multiple sources of knowledge to decide which phrases in a sentence can be removed. Knight and Marcu (2000) proposed statistical approaches to mimic the sentence compression process, they used both noisy-channel and decision-tree to solve the problem. McDonald (2006) presented a discriminative large-margin learning framework coupled with a feature set and syntactic representations for sentence compression. Clarke and Lapata (2006) compared different models for sentence compression across domains and assessed a number of automatic evaluation measures. Clarke and Lapata (2008) used integer linear programming to infer globally optimal compression with linguistically motivated constraints. Berg-Kirkpatrick et al. (2011) proposed a joint model of sentence extraction and compression for multi-document summarization. Filippova and Altun (2013) presented a method for automatically building delete-based sentence compression corpus and proposed an compression method which used structured prediction.

Abstractive sentence compression. Abstractive sentence compression extends delete-based compression methods with additional operations, such as substitution, reordering and insertion. Cohn and Lapata (2008) proposed a discriminative tree-to-tree transduction model which incorporated a grammar extraction method and used a language model for coherent output. Galanis and Androutsopoulos (2011) presented a dataset for extractive and abstractive sentence compression and proposed a SVR based abstractive sentence compressor which utilized additional PMI-based and LDA-based features. Shafieibavani et al. (2016) proposed a word graph-based model which can improve both informativeness and grammaticality of the sentence at the same time.

Neural sentence compression. Filippova et al. (2015) proposed a delete-based sentence compression system which took as input a sentence and output a binary sequence corresponding to word deletion decisions in the sentence. The model was trained on a set of 2 millions sentence pairs which was constructed by the same approach used in Filippova and Altun (2013). There are also some neural approaches for abstractive sentence compression. Rush et al. (2015) proposed a fully data-driven approach which utilized neural language models for abstractive sentence compression. They tried different kinds of encoders to encode the input sentence into vector representation of fixed dimensions. Chopra et al. (2016) further improved the model with Recurrent Neural Networks. However, both works used vocabularies of fixed size for target sentence generation. Wubben et al. (2016) used a Seq2Seq model with bi-directional LSTMs for abstractive compression of captions. Toutanova et al. (2016) manually created a multi-reference dataset for sentence and short paragraph compression and studied the correlations between several automatic evaluation metrics and human judgment.

7 Conclusion

In this paper, we propose Operation Network, a neural approach for abstractive sentence compression, which combines the advantages of both delete-based and generate-based abstractive sentence compression. The central idea of Operation Network is to model the sentence compression process as an editing procedure. With 3 kinds of operations (*delete*, *copy* and *generate*), Operation Network can transform the source sentence into the condensed target. Operation Network is implemented based on the neural Seq2Seq network and pointer network, which consists of a delete decoder and a copy-generate decoder. Given a source sentence as input, first the delete decoder *deletes* unnecessary words from the source sentence, then new words are either *generated* from a large vocabulary or *copied* from the source sentence with the copy-generate decoder. The model is equipped with a strong ability of not only word deletion, but also word reordering and word rephrasing. Experiments show that the model outperforms the baselines.

Acknowledgements

This work was partly supported by the National Science Foundation of China under grant No.61272227/61332007 and the National Basic Research Program (973 Program) under grant No. 2013CB329403.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 481–490. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734. Association for Computational Linguistics.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. *Proceedings of NAACL-HLT16*, pages 93–98.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*. Presented at the Deep Learning workshop at NIPS2014.

James Clarke and Mirella Lapata. 2006. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 377–384. Association for Computational Linguistics.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Trevor Cohn and Mirella Lapata. 2007. Large margin synchronous generation and its application to sentence compression. In *EMNLP-CoNLL*, pages 73–82.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 137–144. Association for Computational Linguistics.

Trevor Cohn and Mirella Lapata. 2013. An abstractive approach to sentence compression. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(3):41.

William Coster and David Kauchak. 2011a. Learning to simplify sentences using wikipedia. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, MTTG '11, pages 1–9, Stroudsburg, PA, USA. Association for Computational Linguistics.

William Coster and David Kauchak. 2011b. Simple english wikipedia: a new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 665–669. Association for Computational Linguistics.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1481–1491. Association for Computational Linguistics.

Katja Filippova, Enrique Alfonseca, Carlos Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Dimitrios Galanis and Ion Androutsopoulos. 2011. A new sentence compression dataset and its use in an abstractive generate-and-rank sentence compressor. In *Proceedings of the UCNLG+Eval: Language Generation and Evaluation Workshop*, pages 1–11. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1631–1640. Association for Computational Linguistics.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 140–149. Association for Computational Linguistics.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*.

Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the sixth conference on Applied natural language processing*, pages 310–315. Association for Computational Linguistics.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 703–710. AAAI Press.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 297–304.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, pages 280–290. Association for Computational Linguistics.

Courtney Napoles, Benjamin Van Durme, and Chris Callison-Burch. 2011. Evaluating sentence compression: Pitfalls and suggested remedies. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 91–97. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389. Association for Computational Linguistics.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Abigail See, Christopher Manning, and Peter Liu. 2017. Get to the point: Summarization with pointer-generator networks. In *Association for Computational Linguistics*.

Elaheh Shafieibavani, Mohammad Ebrahimi, Raymond K Wong, Fang Chen, Robert J Durrant, and Kee-Eung Kim. 2016. An efficient approach for multi-sentence compression. In *Proceedings of the 8th Asian Conference on Machine Learning*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Kristina Toutanova, Chris Brockett, Ke M. Tran, and Saleema Amershi. 2016. A dataset and evaluation metrics for abstractive compression of sentences and short paragraphs. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 340–350. Association for Computational Linguistics.

Jenine Turner and Eugene Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 290–297, Stroudsburg, PA, USA. Association for Computational Linguistics.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Neural Information Processing Systems*.

Sander Wubben, Emiel Krahmer, Antal van den Bosch, and Suzan Verberne. 2016. Abstractive compression of captions with attentive recurrent neural networks. In *Proceedings of the 9th International Natural Language Generation Conference*. Association for Computational Linguistics.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. A lightweight and high performance monolingual word aligner. In *ACL*, pages 702–707.

Enhanced Aspect Level Sentiment Classification with Auxiliary Memory

Peisong Zhu, Tieyun Qian *
School of Computer Science, Wuhan University, China
{zhups24, qty}@whu.edu.cn
*Contact author

Abstract

In aspect level sentiment classification, there are two common tasks: to identify the sentiment of an aspect (category) or a term. As specific instances of aspects, terms explicitly occur in sentences. It is beneficial for models to focus on nearby context words. In contrast, as high level semantic concepts of terms, aspects usually have more generalizable representations. However, conventional methods cannot utilize the information of aspects and terms at the same time, because few datasets are annotated with both aspects and terms. In this paper, we propose a novel deep memory network with auxiliary memory to address this problem. In our model, a main memory is used to capture the important context words for sentiment classification. In addition, we build an auxiliary memory to implicitly convert aspects and terms to each other, and feed both of them to the main memory. With the interaction between two memories, the features of aspects and terms can be learnt simultaneously. We compare our model with the state-of-the-art methods on four datasets from different domains. The experimental results demonstrate the effectiveness of our model.

1 Introduction

Sentiment analysis (Pang and Lee, 2008; Liu, 2012) is a fundamental task in the field of natural language processing. As one of the subtasks of sentiment analysis, the goal of aspect level sentiment classification is to infer the sentiment polarity (e.g. positive, negative, neutral) of the aspect (e.g. food, service). Aspect level sentiment classification is arousing more and more researchers' attention, as it can provide more all-round and deeper analysis than document or sentence level sentiment classification.

There are two common scenes in aspect level sentiment classification. The first one aims to identify the sentiment of a predefined aspect, we call it ASC (aspect sentiment classification) for short. The other aims to identify the sentiment of an explicit term which occurs in the sentence, we call it TSC (term sentiment classification). We take the review "*A mix of students and area residents crowd into this narrow, barely there space for its quick, tasty treats at dirt-cheap prices.*" as an example. For ASC task, we need to infer the sentiment polarities for a set of predefined aspects including "*food*", "*price*", and "*ambience*". Here, the sentiment polarities for "*food*" and "*price*" are positive while that for "*ambience*" is negative. For TSC task, we are given three terms "*space*", "*treats*", "*prices*" in the review and need to infer the sentiment polarities for these terms. Here, the sentiment polarities for terms "*treats*" and "*prices*" are positive while that for "*space*" is negative.

In most cases, aspect level sentiment classification works in a supervised manner. Researchers usually extract features and use machine learning algorithms to train the sentiment classifier. Typical methods are usually based on the manually extracted features. Such methods need either laborious feature engineering works or massive linguistic resources. In recent years, neural networks have achieved significant performance in various NLP tasks like question answering (Sukhbaatar et al., 2015), machine translation (Bahdanau et al., 2014), text summarization (Rush et al., 2015), and text classification (Kim, 2014; Kalchbrenner et al., 2014; Yang et al., 2016). The key advantage of these methods is that they

Proceedings of the 27th International Conference on Computational Linguistics, pages 1077–1087
Santa Fe, New Mexico, USA, August 20-26, 2018.

can automatically learn text features or representations from data. There are several pioneering studies introducing neural networks into the field of sentiment classification, including long short term memory (Tang et al., 2016a; Wang et al., 2016) and deep memory network (Tang et al., 2016b; Tay et al., 2017). The main intuition behind these studies is to capture the important context words related to the given aspects.

The terms and aspects in ASC or TSC are closely related to each other. For example, the term "*sandwich*" is surely about the aspect "*food*". However, existing neural network based methods fail to utilize the relevance between the aspects and the terms as very few datasets are annotated with both aspects and terms. On one hand, the number of terms is extremely large and they normally exhibit a long-tailed distribution. For example, in the review "*Not only did they have amazing, sandwiches, soup, pizza etc, but their homemade sorbets are out of this world!*", "*sandwiches*", "*soup*", "*pizza*", and "*sorbets*" are different words though they are all about the aspect of "food", and many of them rarely occur in the corpus. It is hard for neural networks to train high quality embeddings for the terms with low frequency. On the other hand, as the high level semantic concepts of terms, the number of aspects is small and the aspects are usually predefined. This ensures that the embeddings of aspects are of high quality and generalization ability. However, aspects do not explicitly occur in sentences, thus it is hard for a model to capture aspect related context words from the sentence.

In this paper, we propose a novel deep memory network with an auxiliary memory (**DAuM**) to solve the above problems. Specifically, we first adopts a basic memory to capture the important part of context words for sentiment classification. We then present an auxiliary memory to make the connections between aspects and terms. Due to the lack of supervised information, we cannot get term words for each sentence in ASC task, or get aspect words for each sentence in TSC task. Following the idea of (Lau et al., 2017) and (He et al., 2017), we implicitly generate aspects for terms or extract terms for aspects via the auxiliary memory. Finally, the original and generated aspects/terms are fed into the main memory to capture the sentiment words related to the aspects/terms. With the interaction between two memories, the output vectors will combine features of both aspects and terms. We compare our model with the state-of-the-art methods on four datasets. The experimental results show that our model performs well for different domains of data, and consistently outperforms all baselines.

2 Related Work

Aspect level sentiment classification is a challenging task which aims to identify the sentiment polarity of an aspect or a term in the sentence. Recent years have witnessed the boom of deep learning methods in in aspect level sentiment classification tasks. Tang et al. (2016a) proposed target-dependent LSTM to capture the aspect information when modeling sentences. A forward LSTM and a backward LSTM towards term words are used to capture the information before and after the aspect term. Obviously, this method is not suitable to ASC task as aspect words do not always explicitly occur in sentences.

Attention mechanism (Bahdanau et al., 2014) is an effective mechanism which enforces the model to focus on the important part of context words by computing a weight distribution for context words. Wang et al. (2016) proposed an attention-based LSTM method for the ASC task by concentrating on different parts of a sentence to different aspects. In many cases, given terms have multiple words which makes it difficult to build the semantic representations. Ma et al. (2017) proposed an interactive attention network to address this problem.

Tang et al. (2016b) introduced an end-to-end memory network for aspect level sentiment classification, which employs an attention mechanism over an external memory to capture the importance of each context word. Such importance degree and text representation are calculated with multiple computational layers, each of which is a neural attention model over an external memory. Tay et al. (2017) designed a dyadic memory network that models dyadic interactions between aspect and context with neural tensor compositions or holographic compositions for memory selection operation.

We distinguish our work with the memory networks (Tang et al., 2016b; Tay et al., 2017) in that our goal in to capture the relatedness between term and aspect to improve the sentiment classification performance and we design an auxiliary memory to this end. In contrast, existing studies only focus

on the relatedness between context words the aspect/term using a sentiment memory, which is a single component in our model. Our study is inspired by the recent advances in neural networks, including the unsupervised aspect extraction method in (He et al., 2017), the deep multi-task learning framework in (Li and Lam, 2017) for aspect and opinion extraction tasks, the end-to-end deep memory network for attitude identification and polarity classification task in (Li et al., 2017), and the neural attention networks for stance classification task in (Du et al., 2017).

3 Deep Memory Network with Auxiliary Memory

In this section, we describe our proposed **DAuM** model for aspect level sentiment classification. We mainly introduce the model for TSC task, which has similar architecture for ASC task. We first give the overview of the model and then describe the single layer case. Finally, we introduce how to extend model to stack multiple layers.

3.1 An Overview

An illustration of 3-layer model for TSC is given in Figure 1.

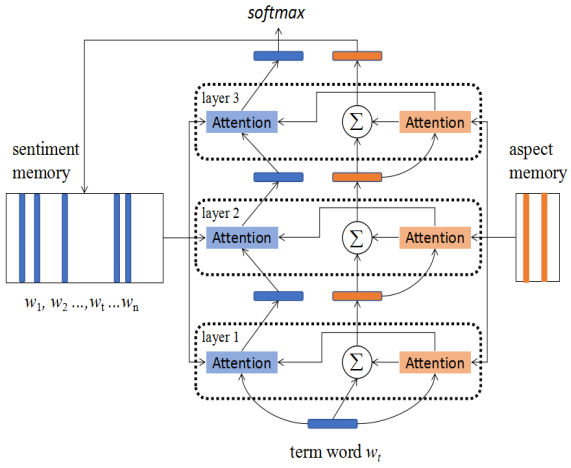


Figure 1: Architecture of **DAuM**

The left part in blue is the main memory for sentiment classification, and we call it sentiment memory. The right part in red is the auxiliary memory. In TSC, auxiliary memory takes a term as input and generates corresponding aspect representation. Therefore we call it aspect memory. Both the original term representation and generated aspect representation, which incorporate term and aspect information at the same time, are fed into sentiment memory to capture the important part of context words. The middle part consists in multiple computational layers. The output vector of a layer is taken as the input of the next layer.

3.2 Single Layer

Input Embedding: To apply deep-learning method to text data, we first map each word into a low-dimensional continuous vector, which is also known as word embedding (Mikolov et al., 2013; Pennington et al., 2014). Specifically, let $\mathbb{L} \in \mathbb{R}^{d \times |V|}$ be an embedding matrix made up of all the word embeddings, where d is the dimensionality of word embedding and $|V|$ is vocabulary size. Given a sentence $s = \{w_1, w_2, ..., w_n\}$ consisting n words, we first use a lookup layer to get the embedding $x_i \in \mathbb{R}^d$ of word w_i, which is a column in the embedding matrix $\mathbb{L}$.

Since terms occur in sentences and a term may contain one single word or multiple words as a phrase, we use word embedding matrix $\mathbb{L}$ to get the representations for terms. In case that a term contains only one word, the term representation will be the embedding of the word. For a phrase-form term, its representation is the average of all constituting word embeddings. We use $v_t \in \mathbb{R}^d$ to denote the term representation.

Aspect Memory: In order to implicitly infer the corresponding aspect for a given term, we define an auxiliary memory to stack the aspect information. The memory takes a term as input, and generates an output representation, which is combined with aspect information. Formally, we define the aspect memory as $M_a \in \mathbb{R}^{k \times d}$, where k is the memory size or the number of defined aspects, and m_i represents the *i*-th piece of the memory.

Taking the term embedding $v_t \in \mathbb{R}^d$ as input, the aspect memory generates a continuous output vector $o_a \in \mathbb{R}^d$. The vector o_a is computed as a weighted sum of each piece of memory m_i via attention mechanism.

$$o_a = \sum_{i=1}^{k} m_i \alpha_i, \tag{1}$$

where α_i is the weight for a piece of memory m_i. The weight α_i is computed based on the semantic relatedness between the term and each aspect.

$$u_i = m_i \cdot W_a \cdot v_t \tag{2}$$

$$\alpha_i = \frac{exp(u_i)}{\sum_{j=1}^{k} exp(u_j)} \tag{3}$$

Eq.2 is a bilinear function used to compute the semantic relatedness. The output of the bilinear function is then fed into a softmax function to generate the final weight α_i in Eq.3.

Sentiment Memory: The goal of TSC is to identify the sentimental polarity of given terms. To this end, we build another external memory, called sentiment memory, to capture the important context words which may be helpful to sentiment classification. Different from the memory network in (Tang et al., 2016b), the sentiment memory in our model takes both term representation and generated aspect representation as the input. Hence our input contains richer information for sentiment classification than that in (Tang et al., 2016b).

We build sentiment memory $M_s \in \mathbb{R}^{n \times d}$ via word embeddings of a sentence , where n is the memory size or the length of the sentence. To differentiate a piece of sentiment memory from that of aspect memory, we use x_i to represent the *i*-th piece in sentiment memory, which is indeed the embedding of the *i*-th word. Similar to aspect memory, the output vector of sentiment memory is also computed as a weighted sum of each piece of memory x_i via attention mechanism. When the term representation $v_t \in \mathbb{R}^d$ is fed into sentiment memory, a feed forward neural network is applied to computing the semantic relatedness between the term and each memory piece.

$$\beta_i^t = softmax(\tanh(W_s^t[x_i; v_t] + b_s^t)), \tag{4}$$

where $W_s^t \in \mathbb{R}^{1 \times 2d}$ and $b_s^t \in \mathbb{R}^{1 \times 1}$ are transformation parameters. To make full use of the information of the term and implicit aspect, sentiment memory further takes the generated aspect representation o_a as input, and generates a weight distribution via attention mechanism.

$$\beta_i^a = softmax(\tanh(W_s^a[x_i; o_a] + b_s^a)), \tag{5}$$

where $W_s^a \in \mathbb{R}^{1 \times 2d}$ and $b_s^a \in \mathbb{R}^{1 \times 1}$ are transformation parameters.

The final weight and output of sentiment memory is computed based on term level weight β^t and aspect level weight β^a. Formally,

$$\beta = (1 - \lambda)\beta^t + \lambda\beta^a, \tag{6}$$

$$o_s = \sum_{i=1}^{n} x_i \beta_i, \tag{7}$$

where β is the final weight combining both term and implicit aspect information, $0 < \lambda < 1$ is the hyperparameter controlling the importance of the aspect or the term, and $o_s \in \mathbb{R}^d$ is the output vector of sentiment memory.

3.3 Multiple Layers

Previous studies (LeCun et al., 2015) show that multiple processing layers can learn higher level representations. Hence we extend our model to stack multiple layers. In the first computational layer, we take the term vector as the input of aspect memory to generate corresponding aspect vector based on

each memory piece in aspect memory through an attention layer. Both the original term vector and the generated aspect vector are then fed into the first layer of sentiment memory to capture the important part of context words through another attention layer. The model is then stacked as follows. In aspect memory, the output of the attention layer and the aspect vector in N_{th} layer are summed as the input of $N+1_{th}$ layer. Then the newly updated aspect vector in $N+1_{th}$ layer of aspect memory and the output of N_{th} attention layer of sentiment memory are taken as the input of $N+1_{th}$ layer of sentiment memory.

3.4 Output and Model Training

After the computation in multiple layers, we get the final output vector o_s^f of sentiment memory and the final output vector o_a^f of aspect memory. Here o_s^f is the sentiment feature, which takes both term and aspect information into consideration. o_a^f is the high level abstraction of a given term. We apply them to different tasks to train the model.

Term Sentiment Classification: Given a term w_t, a sentence s, and the sentiment category set C, the TSC task aims to identify the sentiment polarity of the term in its context. In our model, we regard it as a probability problem, i.e., estimating the conditional probability of w_t in s to a category $c \in C$. We take o_s^f as the final sentiment feature of a sentence, and use a linear layer to transform o_s^f into a real-valued vector which has the same length as the category number $|C|$. We then apply a softmax layer to calculating the conditional probability distribution.

$$P_c(s, w_t) = softmax(W_c o_s^f + b_c), \tag{8}$$

where (s, w_t) denotes a sentence-term pair, and $P_c(s, w_t)$ is the probability of predicting category c given the sentence s and the term w_t. The model is trained in an end-to-end supervised way by minimizing the cross entropy between predicted and real probability distribution. The loss function is written as:

$$L_{cla} = -\sum_{(s,w_t)\in T}\sum_{c\in C} y_c(s, w_t) \cdot log P_c(s, w_t) \tag{9}$$

where T is the set of training data, C is the collection of sentiment categories, and $y_c(s, w_t)$ is the ground truth for (s, w_t) pair.

Term Prediction: o_a^f is the high level aspect representation of a given term. Inspired by (Lau et al., 2017), we believe that the aspect representation generated via a term should be able to predict the term in turn. In other words, the term embedding is more similar to the aspect representation than the embeddings of other words in the sentence. We include this into our objective and define a hinge loss function which maximizes the semantic relatedness between o_a^f and the term while simultaneously minimizing the semantic relatedness between o_a^f and other words in the sentence.

$$L_{pre} = \sum_{(s,w_t)\in T}\sum_{i\in\{1\ldots n\}/\{t\}} max(0, 1 - o_a^f \cdot W_p \cdot v_t + o_a^f \cdot W_p \cdot x_i) \tag{10}$$

Aspect Regularization: We aim to learn the aspect embeddings which can represent different aspects in real-life data. However, the aspect embeddings may suffer from the redundancy problem (He et al., 2017) during the training process. We add a regularization term to ensure the diversity of aspect embeddings.

$$L_{reg} = \|M_a M_a^T - I\|, \tag{11}$$

where I is the identity matrix with the non-diagonal element a_{ij} $(i \neq j)$. $M_a M_a^T$ corresponds to the dot product of two different aspect embeddings, which reflects the relevance between two aspects. L_{reg} reaches its minimum value when the dot product between any two different aspect embeddings is zero. Thus the regularization term encourages orthogonality among the rows of the aspect embedding matrix and penalizes redundant representation. Our final loss function is the linear combination of L_{cla}, L_{pre}, L_{reg}:

$$L_{final} = L_{cla} + \gamma_1 L_{pre} + \gamma_2 L_{reg}, \tag{12}$$

where γ_1, γ_2 are hyperparameters that control the weight of different parts.

3.5 Applied to ASC

The model for ASC task has similar architecture as the model introduced above. With a certain number of aspects, we don't need hyperparameter k for aspect embedding. In addition, auxiliary memory here is used to stack implicit term information, and is initialized with embeddings of words in the sentence. The auxiliary memory takes an aspect as input and generates corresponding term representation with the same method.

4 Experiment

We conduct both the TSC and ASC experiments to check whether our proposed model improve the performance of aspect-level sentiment classification.

4.1 Settings

4.1.1 Dataset

we conduct experiments on four datasets for ASC task and TSC task. In case that the same dataset is used for both tasks, we use a suffix (ASC)/(TSC) to distinguish them.

- **Restaurants(ASC)** is a dataset obtained from SemEval 2014 (Pontiki et al., 2014). This dataset contains customer reviews from the restaurant domain. Each review is annotated a sentiment polarity towards a given aspect. Sentiment polarity set includes "*Positive*", "*Negative*" and "*Neutral*". Predefined aspect set includes "*food*", "*price*", "*service*", "*ambience*" and "*anecdotes/miscellaneous*".
- **Restaurants(TSC)** is the same dataset as Restaurants(ASC) except that terms are given in Restaurants(TSC) while aspects are not annotated.
- **Laptops** is a dataset obtained from SemEval 2014. This dataset contains customer reviews pertaining to the computers and laptops domain. The sentiment polarity set is the same as that in Restaurants(TSC). As terms are given in Laptops, they are used for TSC task.
- **TweetNews** is a dataset obtained from Semeval 2016 Task 6 (Mohammad et al., 2016). It contains English tweets which are annotated for the stance towards one of five topics "*Atheis*", "*Climate Change is a Real Concern*", "*Feminist Movement*", "*Hillary Clinton*", and "*Legalization of Abortio*". The stance set includes "*Favor*", "*Against*" and "*None*" (We treat them as "*Positive*", "*Negative*" and "*Neutral*" for simplification). This dataset is used for ASC task.

We use the official training/testing split in our experiments. To prevent the model from overfitting, we randomly sample one-sixth samples from the training set as the development set. The statistics of four datasets are showed in Table 1.

Table 1: Statistics of four datasets.

Dataset	Set	Total	Pos.	Neg.	Neu.
Restaurants(ASC)	Train	2927	1806	697	424
	Test	973	657	222	94
	Dev	591	372	142	77
Restaurants(TSC)	Train	3017	1806	669	542
	Test	1120	728	196	196
	Dev	591	358	138	95
Laptops	Train	1934	823	730	381
	Test	638	341	128	169
	Dev	394	171	140	63
TweetNews	Train	2416	627	1163	626
	Test	1249	303	716	230
	Dev	498	125	233	140

4.1.2 Metrics

In aspect level sentiment classification task, each dataset has multiple sentiment polarities, and each sample is classified into one of polarities. So we adopt *Accuracy*, *Precision*, *Recall*, as the evaluation

metrics for multi-class classification. We also adopt the macro-averaged *F-score* to show a more thorough result following previous studies (Tay et al., 2017; Tang et al., 2016a).

4.1.3 Experimental Setting

In our experiments, we use 300-dimension word vectors pre-trained by GloVe (Pennington et al., 2014). The dimensionalities of word embedding, aspect embedding are all set to 300, which are same as those in the baselines AE-LSTM and ATAE-LSTM (Wang et al., 2016). λ, γ_1, γ_2 are set to 0.4, 1.0, and 0.5, respectively. We randomize other parameters with uniform distribution $U(-0.01, 0.01)$. We train the model with Stochastic Gradient Descent optimization algorithm and set learning rate as 0.01. We use grid search to get hyper-parameters settings for our method.

4.2 Baselines

In our experiments, we compare our model with the following baseline methods on each dataset.

- **ContextAVG** is a simple baseline which averages the vectors of all context words as the representation of a sentence. The result is then concatenated with the aspect vector and finally fed into softmax function.
- **LSTM** is an implementation of standard LSTM where aspect information cannot be captured in this model.
- **TD-LSTM** (Tang et al., 2016a) takes aspect information into consideration. It uses a forward LSTM and a backward LSTM towards term words to capture the information before and after the term. We run it for only TSC task, as aspect words do not explicitly occur in sentences in ASC task.
- **AE-LSTM** (Wang et al., 2016) takes into account aspect information by embedding aspects into another vector space. The embedding vectors of aspects are learnt during the process of training.
- **ATAE-LSTM** (Wang et al., 2016) is an extension of AE-LSTM. It uses attention mechanism to capture the most important information in response to a given aspect. In addition, ATAE-LSTM can capture the important and different parts of a sentence when different aspects are provides.
- **TAN** (Du et al., 2017) is a model proposed for stance classification which is similar to ASC task. Here we use it as one of baselines to evaluate the effectiveness our model on the task and dataset of other domain.
- **MemNN** (Tang et al., 2016b) is an end-to-end deep memory network which employs an attention mechanism with an external memory to capture the importance of each context word. Such importance degree and text representation are calculated with multiple computational layers.

We re-implemented all baseline methods to make their results as similar as possible to those in the original papers. Each competitor was optimized independently. For the parameters like dimensions, learning rate, and optimization method, we follow their settings in the original paper of the baselines.

4.3 Main Results

In this section, we discuss the experimental results on both ASC task and TSC task. Table 2 reports the results for ASC task on Restaurants (ASC) and TweetNews dataset. Table 3 reports the results for TSC task on Restaurants (TSC) and Laptops dataset.

Table 2: Experimental results for TSC

	Restaurants(TSC)				Laptops			
Method	Accuracy	Precision	Recall	F-score	Accuracy	Precision	Recall	F-score
ContextAVG	75.09	68.93	62.91	63.96	67.24	65.56	60.96	61.86
LSTM	74.28	68.72	61.89	62.21	66.46	65.04	60.54	61.72
TD-LSTM	75.63	69.18	63.05	64.16	68.18	66.86	61.15	62.28
AE-LSTM	76.25	69.76	63.21	64.32	68.97	67.12	61.32	62.50
ATAE-LSTM	77.23	70.83	63.95	64.95	68.65	66.98	61.18	62.45
MemNN	80.09	72.10	65.68	67.82	72.21	70.82	65.03	66.75
DAuM	**82.32**	**74.68**	**70.18**	**71.45**	**74.45**	**72.96**	**69.21**	**70.16**

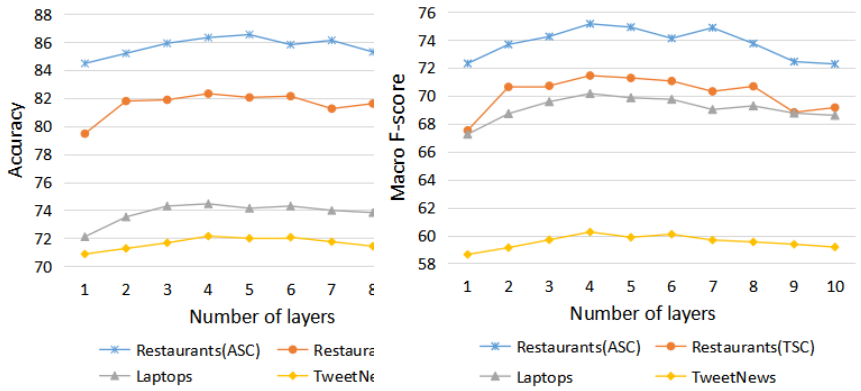
Table 3: Experimental results for ASC

	Restaurants(ASC)				TweetNews			
Method	Accuracy	Precision	Recall	F-score	Accuracy	Precision	Recall	F-score
ContextAVG	80.37	72.86	67.58	68.72	65.25	57.52	51.96	52.88
LSTM	82.01	74.20	69.16	70.20	66.86	58.76	53.12	54.32
AE-LSTM	82.53	74.56	69.36	70.48	69.42	60.96	55.18	56.28
ATAE-LSTM	83.98	75.92	70.88	71.76	69.58	61.10	55.45	56.72
TAN	82.53	74.48	69.50	70.55	68.78	60.42	55.06	56.25
MemNN	84.28	76.06	70.24	72.38	70.14	62.21	57.05	58.62
DAuM	**86.33**	**77.54**	**73.82**	**75.16**	**72.14**	**64.52**	**58.96**	**60.24**

As shown in Table 2 and Table 3, our method consistently outperforms all baselines on four datasets for both ASC task and TSC task. ContextAVG performs poorly. Since averaging context pay equivalent attention to all words, it is unable to capture the sentiment words which are particularly important to sentiment classification. LSTM does not perform well either. The reason may be that it ignores aspect information which plays key role in aspect level sentiment classification. ATAE-LSTM and MemNN perform relatively better as they not only utilize aspect information but also capture important words related to aspect via an attention mechanism. Our model DAuM outperforms ATAE-LSTM and MemNN, as implicit aspect/term representations are generated in our model, which provide more abundant features for sentiment classification. In addition, our model performs better than TAN on TweetNews dataset. This further proves that our model is suitable to some tasks from other domain like stance classification.

4.4 Effects of Multiple Layers

The number of multiple layers has an important effect to the performance of our model. We evaluate our model with 1 to 10 layers with $k = 5$. The results are shown in Figure 2. In general, multiple layers can help compute high level representation and improve the performance. Our model with 4 or 5 layers already works pretty well. However, the performance does not always enhance with the increasing number of layers. It is because too many layers make the training of the model difficult. In addition, the model becomes less generalizable with more parameters introduced.

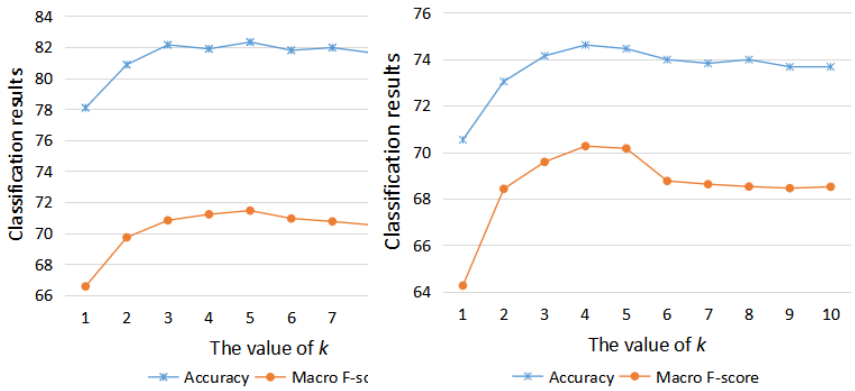


(a) Accuracy (b) Macro F-score

Figure 2: Effects of multiple layers

4.5 Effects of Aspect Number

The aspect number k is a main parameter in our model. To evaluate the effects of aspect number, we fix the number of layers to 4 and change k from 1 to 10. The results are shown in Figure 3. In general, our model performs well in Restaurants(TSC) and Laptops when $k = 5$ and $k = 4$ respectively. When $k < 3$, the result is not as good as when k is large in both datasets. The reason may be that terms are converted into similar aspect representations when k is small. This leads to similar predicted results even



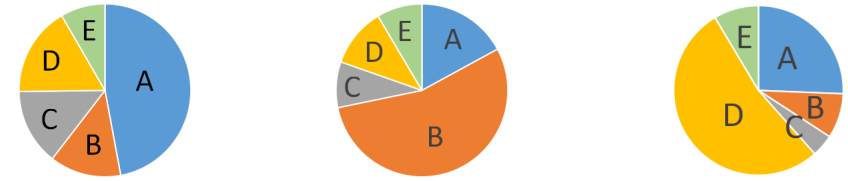
(a) Results on Restaurants(TSC) (b) Results on Laptops

Figure 3: Effects of aspect number k

though different terms are given. It proves that, a limited number of aspects is helpful to our tasks as they have more generalizable representations, but if the number of aspects is too small, it will lose the capability of representing the semantics of different aspects in real-life datasets.

4.6 Case Study

In our model, auxiliary memory takes term/aspect as input and generates corresponding aspect/term representation. To make it more intuitive, we visualize the aspect attention when different terms are fed into aspect memory. The experiment is conducted on Restaurants(TSC) with $k = 5$. "A", "B", "C", "D" and "E" represent the five aspects. Three terms "*pasta*", "*waitress*", "*price*", which belong to aspect "*food*", "*service*", "*price*" respectively, are taken as an example. The pie charts are drawn based on the attention weight for each aspect. The results are shown in Figure 4.



(a) Given "*pasta*" (b) Given "*waitress*" (c) Given "*price*"

Figure 4: Aspect attention weight for different terms

As can be seen, given a term"*pasta*", the aspect memory generates an attention distribution assigning the largest weight to aspect "A". Similarly, the generated attention distribution gives the largest weight to aspect "B" and "D" respectively when other two terms '*waitress*" and "*price*" are fed into aspect memory. To a certain degree, these three cases prove that the aspect memory can effectively compute the semantic relatedness between the given term and each aspect, and consequently generate the most related high level representation.

5 Conclusion

In this paper, we propose a novel deep memory network with auxiliary memory to handle with aspect level sentiment classification tasks. To address the problem that aspect and term information cannot be captured at the same time, we utilize an auxiliary memory to generate aspect or term representation implicitly. The original and generated aspects/terms are all fed into the main memory to capture sentiment words related the aspects/terms. With the interaction between two memories, aspects provide more generalizable representations for terms. And vise versa, terms provide clues for aspects to focus on nearby context words. We demonstrated the effectiveness of our model on four datasets. The results show that

it can significantly outperform the state-of-the-art methods. Besides the improvements of classification performance, our method has a key advantage that it does not need extra annotated aspects or terms. Furthermore, our model can be easily extended to solve other related problems in this domain, or modified with other advanced basic models.

Acknowledgments

The work described in this paper has been supported in part by the NSFC projects (61572376, 91646206), and the 111 project(B07037).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Jiachen Du, Ruifeng Xu, Yulan He, and Lin Gui. 2017. Stance classification with target-specific neural attention. In *Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 3988–3994.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017. An unsupervised neural attention model for aspect extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 388–397.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Jey Han Lau, Timothy Baldwin, and Trevor Cohn. 2017. Topically driven neural language model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 355–365.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436.

Xin Li and Wai Lam. 2017. Deep multi-task learning for aspect term extraction with memory interaction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2886–2892.

Cheng Li, Xiaoxiao Guo, and Qiaozhu Mei. 2017. Deep memory networks for attitude identification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 671–680. ACM.

Bing Liu. 2012. Sentiment analysis and opinion mining. In *Synthesis Lectures on Human Language Technologies*, pages 152–153.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 4068–4074.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *International Workshop on Semantic Evaluation*, pages 31–41.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1):459–526.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. *Proceedings of International Workshop on Semantic Evaluation at*, pages 27–35.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING 2016*, pages 3298–3307.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 214–224.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017. Dyadic memory networks for aspect-based sentiment analysis. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 107–116. ACM.

Yequan Wang, Minlie Huang, Li Zhao, et al. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Author Profiling for Abuse Detection

Pushkar Mishra
Dept. of CS and Technology
University of Cambridge
United Kingdom
pm576@cl.cam.ac.uk

Marco Del Tredici
ILLC
University of Amsterdam
The Netherlands
m.deltredici@uva.nl

Helen Yannakoudakis
Dept. of CS and Technology
The ALTA Institute
University of Cambridge
United Kingdom
hy260@cl.cam.ac.uk

Ekaterina Shutova
ILLC
University of Amsterdam
The Netherlands
e.shutova@uva.nl

Abstract

The rapid growth of social media in recent years has fed into some highly undesirable phenomena such as proliferation of hateful and offensive language on the Internet. Previous research suggests that such abusive content tends to come from users who share a set of common stereotypes and form communities around them. The current state-of-the-art approaches to abuse detection are oblivious to user and community information and rely entirely on textual (i.e., lexical and semantic) cues. In this paper, we propose a novel approach to this problem that incorporates community-based profiling features of Twitter users. Experimenting with a dataset of $16k$ tweets, we show that our methods significantly outperform the current state of the art in abuse detection. Further, we conduct a qualitative analysis of model characteristics. We release our code, pre-trained models and all the resources used in the public domain.

1 Introduction

Abuse, a term used to collectively refer to offensive language, hate speech, sexist remarks, etc., is omnipresent in social media. Users on social media platforms are at risk of being exposed to content that may not only be degrading but also harmful to their mental health in the long term. *Pew Research Center* highlighted the gravity of the situation via a recently released report (Duggan, 2014). As per the report, 40% of adult Internet users have personally experienced harassment online, and 60% have witnessed the use of offensive names and expletives. Expectedly, the majority (66%) of those who have personally faced harassment have had their most recent incident occur on a social networking website or app. While most of these websites and apps provide ways of flagging offensive and hateful content, only 8.8% of the victims have actually considered using such provisions. These statistics suggest that passive or manual techniques for curbing propagation of abusive content (such as flagging) are neither effective nor easily scalable (Pavlopoulos et al., 2017). Consequently, the efforts to automate the detection and moderation of such content have been gaining popularity in natural language processing (NLP) (Waseem and Hovy, 2016; Wulczyn et al., 2017).

Several approaches to abuse detection demonstrate the effectiveness of character-level bag-of-words features in a supervised classification setting (Djuric et al., 2015; Nobata et al., 2016; Davidson et al., 2017). More recent approaches, and currently the best performing ones, utilize recurrent neural networks (RNNs) to transform content into dense low-dimensional semantic representations that are then used for classification (Pavlopoulos et al., 2017; Badjatiya et al., 2017). All of these approaches rely solely on lexical and semantic features of the text they are applied to. Waseem and Hovy (2016) adopted a

Proceedings of the 27th International Conference on Computational Linguistics, pages 1088–1098
Santa Fe, New Mexico, USA, August 20-26, 2018.

more user-centric approach based on the idea that perpetrators of abuse are usually segregated into small demographic groups; they went on to show that gender information of *authors* (i.e., users who have posted content) is a helpful indicator. However, Waseem and Hovy focused only on coarse demographic features of the users, disregarding information about their communication with others. But previous research suggests that users who subscribe to particular stereotypes that promote abuse tend to form communities online. For example, Zook (2012) mapped the locations of racist tweets in response to President Obama's re-election to show that such tweets were not uniformly distributed across the United States but formed clusters instead. In this paper, we present the first approach to abuse detection that leverages author profiling information based on properties of the authors' social network and investigate its effectiveness.

Author profiling has emerged as a powerful tool for NLP applications, leading to substantial performance improvements in several downstream tasks, such as text classification, sentiment analysis and author attribute identification (Hovy, 2015; Eisenstein, 2015; Yang and Eisenstein, 2017). The relevance of information gained from it is best explained by the idea of *homophily*, i.e., the phenomenon that people, both in real life as well as on the Internet, tend to associate more with those who appear similar. Here, similarity can be defined along various axes, e.g., location, age, language, etc. The strength of author profiling lies in that if we have information about members of a community c defined by some similarity criterion, and we know that the person p belongs to c, we can infer information about p. This concept has a straightforward application to our task: knowing that members of a particular community are prone to creating abusive content, and knowing that the author *p* is connected to this community, we can leverage information beyond linguistic cues and more accurately predict the use of abusive/non-abusive language from p. The questions that we seek to address here are: are some authors, and the respective communities that they belong to, more abusive than the others? And can such information be effectively utilized to improve the performance of automated abusive language detection methods?

In this paper, we answer these questions and develop novel methods that take into account community-based profiling features of authors when examining their tweets for abuse. Experimenting with a dataset of $16k$ tweets, we show that the addition of such profiling features to the current state-of-the-art methods for abuse detection significantly enhances their performance. We also release our code (including code that replicates previous work), pre-trained models and the resources we used in the public domain.[1]

2 Related Work

2.1 Abuse detection

Amongst the first ones to apply supervised learning to the task of abuse detection were Yin et al. (2009) who used a linear SVM classifier to identify posts containing harassment based on local (e.g., n-grams), contextual (e.g., similarity of a post to its neighboring posts) and sentiment-based (e.g., presence of expletives) features. Their best results were with all of these features combined.

Djuric et al. (2015) experimented with comments extracted from the Yahoo Finance portal and showed that distributional representations of comments learned using *paragraph2vec* (Le and Mikolov, 2014) outperform simpler bag-of-words (BOW) representations in a supervised classification setting for hate speech detection. Nobata et al. (2016) improved upon the results of Djuric et al. by training their classifier on a combination of features drawn from four different categories: linguistic (e.g., count of insult words), syntactic (e.g., POS tags), distributional semantic (e.g., word and comment embeddings) and BOW-based (word and characters n-grams). They reported that while the best results were obtained with all features combined, character n-grams contributed more to performance than all the other features.

Waseem and Hovy (2016) created and experimented with a dataset of racist, sexist and clean tweets. Utilizing a logistic regression (LR) classifier to distinguish amongst them, they found that character n-grams coupled with gender information of users formed the optimal feature set; on the other hand, geographic and word-length distribution features provided little to no improvement. Working with the same dataset, Badjatiya et al. (2017) improved on their results by training a gradient-boosted decision

[1] https://github.com/pushkarmishra/AuthorProfilingAbuseDetection

tree (GBDT) classifier on averaged word embeddings learnt using a long short-term memory (LSTM) network that they initialized with random embeddings.

Waseem (2016) sampled $7k$ more tweets in the same manner as Waseem and Hovy (2016). They recruited expert and amateur annotators to annotate the tweets as *racism*, *sexism*, *both* or *neither* in order to study the influence of annotator knowledge on the task of hate speech detection. Combining this dataset with that of Waseem and Hovy (2016), Park et al. (2017) explored the merits of a two-step classification process. They first used a LR classifier to separate abusive and non-abusive tweets, followed by another LR classifier to distinguish between racist and sexist ones. They showed that this setup had comparable performance to a one-step classification setup built with convolutional neural networks.

Davidson et al. (2017) created a dataset of about $25k$ tweets wherein each tweet was annotated as being *racist*, *offensive* or *neither of the two*. They tested several multi-class classifiers with the aim of distinguishing clean tweets from racist and offensive tweets while simultaneously being able to separate the racist and offensive ones. Their best model was a LR classifier trained using TF-IDF and POS n-gram features, as well as the count of hash tags and number of words.

Wulczyn et al. (2017) prepared three different datasets of comments collected from the English Wikipedia Talk page; one was annotated for personal attacks, another for toxicity and the third one for aggression. Their best performing model was a multi-layered perceptron (MLP) classifier trained on character n-gram features. Experimenting with the personal attack and toxicity datasets, Pavlopoulos et al. (2017) improved the results of Wulczyn et al. by using a gated recurrent unit (GRU) model to encode the comments into dense low-dimensional representations, followed by a LR layer to classify the comments based on those representations.

2.2 Author profiling

Author profiling has been leveraged in several ways for a variety of purposes in NLP. For instance, many studies have relied on demographic information of the authors. Amongst these are Hovy et al. (2015) and Ebrahimi et al. (2016) who extracted age and gender-related information to achieve superior performance in a text classification task. Pavalanathan and Eisenstein (2015), in their work, further showed the relevance of the same information to automatic text-based geo-location. Researching along the same lines, Johannsen et al. (2015) and Mirkin et al. (2015) utilized demographic factors to improve syntactic parsing and machine translation respectively.

While demographic information has proved to be relevant for a number of tasks, it presents a significant drawback: since this information is not always available for all authors in a social network, it is not particularly reliable. Consequently, of late, a new line of research has focused on creating representations of users in a social network by leveraging the information derived from the connections that they have with other users. In this case, node representations (where nodes represent the authors in the social network) are typically induced using neural architectures. Given the graph representing the social network, such methods create low-dimensional representations for each node, which are optimized to predict the nodes close to it in the network. This approach has the advantage of overcoming the absence of information that the previous approaches face. Among those that implement this idea are Yang et al. (2016), who used representations derived from a social graph to achieve better performance in entity linking tasks, and Chen and Ku (2016), who used them for stance classification.

A considerable amount of literature has also been devoted to sentiment analysis with representations built from demographic factors (Yang and Eisenstein, 2017; Chen et al., 2016). Other tasks that have benefited from social representations are sarcasm detection (Amir et al., 2016) and political opinion prediction (Tălmăcel and Leon, 2017).

3 Dataset

We experiment with the dataset of Waseem and Hovy (2016), containing tweets manually annotated for abuse. The authors retrieved around $136k$ tweets over a period of two months. They bootstrapped their collection process with a search for commonly used slurs and expletives related to religious, sexual, gender and ethnic minorities. From the results, they identified terms and references to entities that

frequently showed up in abusive tweets. Based on this sample, they used a public Twitter API to collect the entire corpus of ca. $136k$ tweets. After having manually annotated a randomly sampled subset of $16,914$ tweets under the categories *racism*, *sexism* or *none* themselves, they asked an expert to review their annotations in order to mitigate against any biases. The inter-annotator agreement was reported at $\kappa = 0.84$, with a further insight that 85% of all the disagreements occurred in the *sexism* class.

The dataset was released as a list of $16,907$ tweet IDs along with their corresponding annotations[2]. Using python's *Tweepy* library, we could only retrieve $16,202$ of the tweets since some of them have now been deleted or their visibility limited. Of the ones retrieved, 1,939 (12%) are labelled as *racism*, 3,148 (19.4%) as *sexism*, and the remaining 11,115 (68.6%) as *none*; this distribution follows the original dataset very closely (11.7%, 20.0%, 68.3%).

We were able to extract community-based information for 1,836 out of the 1,875 unique authors who posted the $16,202$ tweets, covering a cumulative of 16,124 of them; the remaining 39 authors have either deactivated their accounts or are facing suspension. Tweets in the *racism* class are from 5 of the 1,875 authors, while those in the *sexism* class are from 527 of them.

4 Methodology

4.1 Representing authors

In order to leverage community-based information for the authors whose tweets form our dataset, we create an undirected unlabeled community graph wherein nodes are the authors and edges are the connections between them. An edge is instantiated between two authors u and v if u follows v on Twitter or vice versa. There are a total of 1,836 nodes and 7,561 edges. Approximately 400 of the nodes have no edges, indicating *solitary* authors who neither follow any other author nor are followed by any. Other nodes have an average degree[3] of 8, with close to 600 of them having a degree of at least 5. The graph is overall sparse with a density of 0.0075.

From this community graph, we obtain a vector representation, i.e., an embedding that we refer to as *author profile*, for each author using the *node2vec* framework (Grover and Leskovec, 2016). *Node2vec* applies the skip-gram model of Mikolov et al. (2013) to a graph in order to create a representation for each of its nodes based on their positions and their neighbors. Specifically, given a graph with nodes $V = \{v_1, v_2, \ldots, v_n\}$, *node2vec* seeks to maximize the following log probability:

$$\sum_{v \in V} \log Pr\left(N_s(v) \mid v\right)$$

where $N_s(v)$ denotes the *network neighborhood* of node v generated through sampling strategy s.

In doing so, the framework learns low-dimensional embeddings for nodes in the graph. These embeddings can emphasize either their structural role or the local community they are a part of. This depends on the sampling strategies used to generate the neighborhood: if breadth-first sampling (BFS) is adopted, the model focuses on the immediate neighbors of a node; when depth-first sampling (DFS) is used, the model explores farther regions in the network, which results in embeddings that encode more information about the nodes' structural role (e.g., hub in a cluster, or peripheral node). The balance between these two ways of sampling the neighbors is directly controlled by two *node2vec* parameters, namely p and q. The default value for these is 1, which ensures a node representation that gives equal weight to both structural and community-oriented information. In our work, we use the default value for both p and q. Additionally, since *node2vec* does not produce embeddings for *solitary* authors, we map these to a single zero embedding.

Figure 1 shows example snippets from the community graph. Some authors belong to densely-connected communities (left figure), while others are part of more sparse ones (right figure). In either case, *node2vec* generates embeddings that capture the authors' neighborhood.

[2] https://github.com/ZeerakW/hatespeech/blob/master/NAACL_SRW_2016.csv

[3] The degree of a node is equal to the number of its direct connections to other nodes.

(a) Densely-connected authors

(b) Sparsely-connected authors

Figure 1: Snippets from the community graph for our Twitter data.

4.2 Classifying content

We experiment with seven different methods for classifying tweets as one of *racism*, *sexism*, or *none*. We first re-implement three established and currently best-performing abuse detection methods — based on character n-grams and recurrent neural networks — as our baselines. We then test whether incorporating author profiling features improves their performance.

Char n-grams (LR). As our first baseline, we adopt the method used by Waseem and Hovy (2016) wherein they train a logistic regression (LR) classifier on the Twitter dataset using character n-gram counts. We use uni-grams, bi-grams, tri-grams and four-grams, and L_2-normalize their counts. Character n-grams have been shown to be effective for the task of abuse detection (Nobata et al., 2016).

Hidden-state (HS). As our second baseline, we take the "RNN" method of Pavlopoulos et al. (2017) which achieves state-of-the-art results on the Wikipedia datasets released by Wulczyn et al. (2017). The method comprises a 1-layer gated recurrent unit (GRU) that takes a sequence $w_1, \ldots, w_n$ of words represented as d-dimensional embeddings and encodes them into hidden states $h_1, \ldots, h_n$. This is followed by an LR layer that uses the last hidden state h_n to classify the tweet. We make two minor modifications to the authors' original architecture: we deepen the 1-layer GRU to a 2-layer GRU and use softmax instead of sigmoid in the LR layer.[4] Like Pavlopoulos et al., we initialize the word embeddings to GLoVe vectors (Pennington et al., 2014). In all our methods, words not available in the GLoVe set are randomly initialized in the range ± 0.05, indicating the lack of semantic information. By not mapping these words to a single random embedding, we mitigate against the errors that may arise due to their conflation (Madhyastha et al., 2015). A special OOV (out of vocabulary) token is also initialized in the same range. All the embeddings are updated during training, allowing some of the randomly-initialized ones to get task-tuned; the ones that do not get tuned lie closely clustered around the OOV token, to which unseen words in the test set are mapped.

Word-sum (WS). As a third baseline, we adopt the "LSTM+GLoVe+GBDT" method of Badjatiya et al. (2017), which achieves state-of-the-art results on the Twitter dataset we are using. The authors first utilize an LSTM to task-tune GLoVe-initialized word embeddings by propagating the error back from an LR layer. They then train a gradient boosted decision tree (GBDT) classifier to classify texts based on the average of the embeddings of constituent words. We make two minor modifications to this method: we use a 2-layer GRU[5] instead of the LSTM to tune the embeddings, and we train the GBDT classifier on the L_2-normalized sum of the embeddings instead of their average.[6] Although the authors achieved

[4] We also experimented with 1-layer GRU/LSTM and 1/2-layer bi-directional GRUs/LSTMs but performance only worsened or showed no gains; using sigmoid instead of softmax did not have any noteworthy effects on the results either.

[5] We note the deeper 2-layer GRU slightly improves performance.

[6] Although GBDT, as a tree based model, is not affected by the choice of monotonic function, the L_2-normalized sum ensures uniformity of range across the feature set in all our methods.

state-of-the-art results on Twitter by initializing embeddings randomly rather than with GLoVe (which is what we do here), we found the opposite when performing a 10-fold stratified cross-validation (CV). A possible explanation of this lies in the authors' decision to not use stratification, which for such a highly imbalanced dataset can lead to unexpected outcomes (Forman and Scholz, 2010). Furthermore, the authors train their LSTM on the entire dataset (including the test set) without any early stopping criterion, which leads to over-fitting of the randomly-initialized embeddings.

Author profile (AUTH). In order to test whether community-based information of authors is in itself sufficient to correctly classify the content produced by them, we utilize just the author profiles we generated to train a GBDT classifier.

Char n-grams + author profile (LR + AUTH). This method builds upon the LR baseline by appending author profile vectors on to the character n-gram count vectors for training the LR classifier.

Hidden-state + author profile (HS + AUTH) and **Word-sum + author profile (WS + AUTH).** These methods are identical to the *char n-grams + author profile* method except that here we append the author profiling features on to features derived from the *hidden-state* and *word-sum* baselines respectively and feed them to a GBDT classifier.

5 Experiments and Results

5.1 Experimental setup

We normalize the input by lowercasing all words and removing stop words. For the GRU architecture, we use exactly the same hyper-parameters as Pavlopoulos et al. (2017),[7] i.e., 128 hidden units, Glorot initialization, cross-entropy loss, and the Adam optimizer (Kingma and Ba, 2015). Badjatiya et al. (2017) also use the same settings except they have fewer hidden units. In all our models, besides dropout regularization (Srivastava et al., 2014), we hold out a small part of the training set as validation data to prevent over-fitting. We implement the models in *Keras* (Chollet and others, 2015) with *Theano* backend and use 200-dimensional pre-trained GLoVe word embeddings.[8] We employ *Lightgbm* (Ke et al., 2017) as our GDBT classifier and tune its hyper-parameters using 5-fold grid search. For the *node2vec* framework, we use the same parameters as in the original paper (Grover and Leskovec, 2016) except we set the dimensionality of node embeddings to 200 and increase the number of iterations to 25 for better convergence.

5.2 Results

We perform 10-fold stratified cross validation (CV), as suggested by Forman and Scholz (2010), to evaluate all seven methods described in the previous section. Following previous research (Badjatiya et al., 2017; Park and Fung, 2017), we report the average weighted precision, recall, and F_1 scores for all the methods. The average weighted precision is calculated as:

$$\frac{\sum_{i=1}^{10} (w_r \cdot \mathrm{P}_r^i + w_s \cdot \mathrm{P}_s^i + w_n \cdot \mathrm{P}_n^i)}{10}$$

where $\mathrm{P}_r^i, \mathrm{P}_s^i, \mathrm{P}_n^i$ are precision scores on the *racism*, *sexism*, and *none* classes from the i^{th} fold of the CV. The values w_r, w_s, and w_n are the proportions of the *racism*, *sexism*, and *none* classes in the dataset respectively; since we use stratification, these proportions are constant ($w_r = 0.12$, $w_s = 0.19$, $w_n = 0.69$) across all folds. Average weighted recall and F_1 are calculated in the same manner.

The results are presented in Table 1. For all three baseline methods (LR, WS, and HS), the addition of author profiling features significantly improves performance ($p < 0.05$ under 10-fold CV paired t-test). The LR + AUTH method yields the highest performance of $F_1 = 87.57$, exceeding its respective baseline by nearly 4 points. A similar trend can be observed for the other methods as well. These results point to

[7] The authors have not released their models, and we therefore replicate their approach based on the details in their paper.

[8] http://nlp.stanford.edu/data/glove.twitter.27B.zip

the importance of community-based information and author profiling in abuse detection and demonstrate that our approach can further improve the performance of existing state-of-the-art methods.

	Method	P	R	F_1
Baselines	LR	84.07	84.31	83.81
	HS	83.50	83.71	83.54
	WS	82.86	83.10	82.37
Our methods	AUTH	72.13	76.05	71.26
	LR + AUTH	**87.57**	**87.66**	**87.57**
	HS + AUTH	87.29	87.32	87.29
	WS + AUTH	87.11	87.20	87.08

Table 1: Average weighted precision, recall and F_1 scores of the different methods on the Twitter datasest. All improvements are significant ($p < 0.05$) under 10-fold CV paired t-test.

Method	P	R	F_1
LR	**77.29**	67.92	72.28
HS	74.15	72.46	73.24
WS	76.43	67.77	71.78
AUTH	43.33	0.31	0.61
LR + AUTH	76.10	**74.16**	**75.09**
HS + AUTH	74.42	73.54	73.91
WS + AUTH	75.12	72.46	73.72

(a) *Racism* class

Method	P	R	F_1
LR	82.66	63.98	72.09
HS	76.04	68.84	72.24
WS	81.75	57.37	67.38
AUTH	66.85	75.44	70.88
LR + AUTH	86.22	79.07	82.47
HS + AUTH	84.15	**81.32**	**82.75**
WS + AUTH	**86.37**	77.92	81.91

(b) *Sexism* class

Table 2: Performance of the methods on the *racism* and *sexism* classes separately. All improvements are significant ($p < 0.05$) under 10-fold CV paired t-test.

In Table 2, we further compare the performance of the different methods on the *racism* and *sexism* classes individually. As in the previous experiments, the scores are averaged over 10 folds of CV. Of particular interest are the scores for the *sexism* class where the F_1 increases by over 10 points upon the addition of author profiling features. Upon analysis, we find that such a substantial increase in performance stems from the fact that many of the 527 unique authors of the sexist tweets are closely connected in the community graph. This allows for their penchant for sexism to be expressed in their respective author profiles.

The author profiling features on their own (AUTH) achieve impressive results overall and in particular on the *sexism* class, where their performance is typical of a community-based generalization, i.e., low precision but high recall. For the *racism* class on the other hand, the performance of AUTH on its own is quite poor. This contrast can be explained by the fact that tweets in the *racism* class come from only 5 unique authors who: (i) are isolated in the community graph, or (ii) have also authored several tweets in the *sexism* class, or (iii) are densely connected to authors from the *sexism* and *none* classes which possibly camouflages their racist nature.

We believe that the gains in performance will be more pronounced as the underlying community graph grows since there will be less solitary authors and more edges worth harnessing information from.[9] Even when the data is skewed and there is an imbalance of abusive vs. non-abusive authors, we do expect our approach to still be able to identify clusters of authors with similar views.

6 Analysis and discussion

We conduct a qualitative analysis of system errors and the cases where author profiling leads to the correct classification of previously misclassified examples. Table 3 shows examples of abusive tweets from the dataset that are misclassified by the LR method, but are correctly classified upon the addition of author profiling features, i.e., by the LR + AUTH method. It is worth noting that some of the wins scored by the latter are on tweets that are part of a larger abusive discourse or contain links to abusive

[9]Regarding the scalability of our approach, we quote the authors of *node2vec*: "The major phases of *node2vec* are trivially parallelizable, and it can scale to large networks with millions of nodes in a few hours".

content while not explicitly having textual cues that are indicative of abuse per se. The addition of author profiling features may then be viewed as a proxy for wider discourse information, thus allowing us to correctly resolve the cases where lexical and semantic features alone are insufficient.[10]

Tweet	Predicted label	
	LR	LR + AUTH
@Mich_McConnell Just "her body" right?	*none*	*sexism*
@Starius: #GamerGate https://t.co/xuFwsIgxFK WE WIN! ahahahaha	*none*	*sexism*
#Islam dominates our crime, prison & welfare system & national security. Why are we still importing it? @PeterDutton_MP #amagenda #auspol	*none*	*racism*
@Wateronatrain: @MT8_9 You might like this #patriarchy http://t.co/c9m2pFmFJ3	*none*	*sexism*
It seems that Allah sits around all day obsessing about women's hands and faces showing. I guess idiots need a god on their level. #Islam	*none*	*racism*
@SalemP08: @MT8_9 @LiljaOB @midnitebacon @Superjutah @Transic_nyc her response is pretty terrifying.	*none*	*sexism*
@JosephIsVegan @SumbelinaZ @IronmanL1 @Hatewatch Why would you profile white people. Blacks murder at 6 times the rate as whites.	*none*	*racism*

Table 3: Examples of improved classification upon the addition of author profiling features (AUTH).

However, a number of abusive tweets still remain misclassified despite the addition of author profiling features. According to our analysis, many of these tend to contain URLs to abusive content, e.g., "*@salmonfarmer1: Logic in the world of Islam http://t.co/6nALv2HPc3*" and "*@juliarforster Yes. http://t.co/ixbt0uc7HN*". Since Twitter shortens all URLs into a standard format, there is no indication of what they refer to. One way to deal with this limitation could be to additionally maintain a blacklist of links. Another source of system errors is the deliberate obfuscation of words by authors in order to evade detection, e.g., "*Kat, a massive c*nt. The biggest ever on #mkr #cuntandandre*". Current abuse detection methods, including ours, do not directly attempt to address this issue. While this is a challenge for bag-of-word based methods such as LR, we hypothesize that neural networks operating at the character level may be helpful in recognizing obfuscated words.

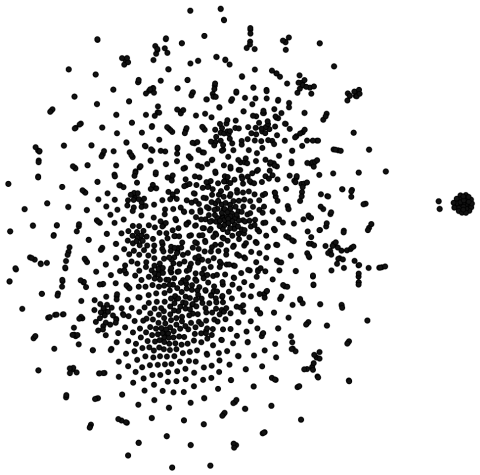


Figure 2: Visualization of author embeddings in 2-dimensional space.

We further conducted an analysis of the author embeddings generated by *node2vec*, in order to validate that they capture the relevant aspects of the community graph. We visualized the author embeddings in

[10]We note that the annotators of the dataset took discourse into account when annotating the tweets. However, the dataset was released as a list of tweet ID and corresponding annotation (racism/sexism/none) pairs; there is no annotation available regarding which tweets are related to which other ones.

(a) *None* class (b) *Sexism* class

Figure 3: Visualization of authors from different classes.

2-dimensional space using *t*-SNE (van der Maaten and Hinton, 2008), as shown in Figure 2. We observe that, as in the community graph, there are a few densely populated regions in the visualization that represent authors in closely knit groups who exhibit similar characteristics. The other regions are largely sparse with smaller clusters. Note that we exclude *solitary* users from this visualization since we have to use a single zero embedding to represent them.

Figure 3 further provides visualizations for authors from the *sexism* and *none* classes separately. While the authors from the *none* class are spread out in the embedding space, the ones from the *sexism* class are more tightly clustered. Note that we do not visualize the 5 authors from the *racism* class since 4 of them are already covered in the *sexism* class.

7 Conclusions

In this paper, we explored the effectiveness of community-based information about authors for the purpose of identifying abuse. Working with a dataset of $16k$ tweets annotated for *racism* and *sexism*, we first comprehensively replicated three established and currently best-performing abuse detection methods based on character n-grams and recurrent neural networks as our baselines. We then constructed a graph of all the authors of tweets in our dataset and extracted community-based information in the form of dense low-dimensional embeddings for each of them using *node2vec*. We showed that the inclusion of author embeddings significantly improves system performance over the baselines and advances the state of the art in this task. Users prone to abuse do tend to form social groups online, and this stresses the importance of utilizing community-based information for automatic abusive language detection.

In the future, we wish to explore the effectiveness of community-based author profiling in other tasks such as stereotype identification and metaphor detection.

References

Silvio Amir, Byron C Wallace, Hao Lyu, and Paula Carvalho Mário J Silva. 2016. Modelling context with user embeddings for sarcasm detection in social media. *arXiv preprint arXiv:1607.00976*.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, pages 759–760, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Wei-Fan Chen and Lun-Wei Ku. 2016. Utcnn: a deep learning model of stance classification on social media text. *arXiv preprint arXiv:1611.03599*.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1650–1659.

François Chollet et al. 2015. Keras.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 29–30, New York, NY, USA. ACM.

Maeve Duggan. 2014. Online harassment.

Javid Ebrahimi and Dejing Dou. 2016. Personalized semantic word vectors. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1925–1928. ACM.

Jacob Eisenstein. 2015. Written dialect variation in online social media. *Charles Boberg, John Nerbonne, and Dom Watt, editors, Handbook of Dialectology. Wiley*.

George Forman and Martin Scholz. 2010. Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement. *SIGKDD Explor. Newsl.*, 12(1):49–57, November.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 752–762.

Anders Johannsen, Dirk Hovy, and Anders Søgaard. 2015. Cross-lingual syntactic variation over age and gender. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 103–112.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3149–3157. Curran Associates, Inc.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, ICLR '15.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053.

Pranava Swaroop Madhyastha, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Mapping unseen words to task-trained embedding spaces. *CoRR*, abs/1510.02387.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Shachar Mirkin, Scott Nowson, Caroline Brun, and Julien Perez. 2015. Motivating personality-aware machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1102–1108.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 145–153, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Ji Ho Park and Pascale Fung. 2017. One-step and two-step classification for abusive language detection on twitter. In *Proceedings of the First Workshop on Abusive Language Online*, pages 41–45. Association for Computational Linguistics.

Umashanthi Pavalanathan and Jacob Eisenstein. 2015. Confounds and consequences in geotagged twitter data. *arXiv preprint arXiv:1506.02275*.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. Deep learning for user comment moderation. In *Proceedings of the First Workshop on Abusive Language Online*, pages 25–35. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Ciprian Tălmăcel and Florin Leon. 2017. Predicting political opinions in social networks with user embeddings. In *Proceedings of the 2017 13th IEEE International Conference on Intelligent Computer Communication and Processing*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California, June. Association for Computational Linguistics.

Zeerak Waseem. 2016. Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 138–142. Association for Computational Linguistics.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 1391–1399, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Yi Yang and Jacob Eisenstein. 2017. Overcoming language variation in sentiment analysis with social attention. *Transactions of the Association for Computational Linguistics*.

Yi Yang, Ming-Wei Chang, and Jacob Eisenstein. 2016. Toward socially-infused information extraction: Embedding authors, mentions, and entities. *arXiv preprint arXiv:1609.08084*.

Dawei Yin, Brian D. Davison, Zhenzhen Xue, Liangjie Hong, April Kontostathis, and Lynne Edwards. 2009. Detection of harassment on web 2.0. In *Processings of the Content Analysis in the WEB 2.0*, 2:1-7.

Matthew Zook. 2012. Mapping racist tweets in response to president obama's re-election. [Online; accessed 15 March 2018].

Automated Scoring: Beyond Natural Language Processing

Nitin Madnani **Aoife Cahill**

Educational Testing Service
Princeton, NJ, 08541 USA
{nmadnani,acahill}@ets.org

Abstract

In this position paper, we argue that building operational automated scoring systems is a task that has disciplinary complexity above and beyond standard competitive shared tasks which usually involve applying the latest machine learning techniques to publicly available data in order to obtain the best accuracy. Automated scoring systems warrant significant cross-discipline collaboration of which natural language processing and machine learning are just two of *many* important components. Such systems have multiple stakeholders with different but valid perspectives that can often times be at odds with each other. Our position is that it is essential for us as NLP researchers to understand and incorporate these perspectives into our research and work towards a mutually satisfactory solution in order to build automated scoring systems that are accurate, fair, unbiased, and useful.

1 What is Automated Scoring?

Automated scoring is an NLP application usually deployed in the educational domain. It involves automatically analyzing a student's response to a question and generating either (a) a score in order to assess the student's knowledge and/or other skills and/or (b) some actionable feedback on how the student can improve the response (Page, 1966; Burstein et al., 1998; Burstein et al., 2004; Zechner et al., 2009; Bernstein et al., 2010). It is considered an NLP application since typically the core technology behind the automated analysis of the student response enlists NLP techniques. The student responses can include essays, short answers, or spoken responses and the two most common kinds of automated scoring are the automated evaluation of writing quality and content knowledge. Both the scores and feedback are usually based on linguistic characteristics of the responses including but not limited to:

(i) Lower-level errors in the response (e.g., pronunciation errors in spoken responses and grammatical/spelling errors in written responses),

(ii) The discourse structure and/or organization of the response,

(iii) Relevance of the response to the question that was asked.

2 Motivation

Over the last few years, there has been a significant increase in the number of NLP conference and workshop publications on the task of automated scoring of student responses (Burrows et al., 2015; Zesch et al., 2015; Sultan et al., 2016). Much of this increase in interest stems from the public availability of fairly large datasets containing scored human responses as part of shared tasks and public contests (the ASAP[1] and ASAP2[2] Kaggle shared tasks, the Powergrading dataset (Basu et al., 2013), and the SemEval

[1] https://www.kaggle.com/c/asap-aes

[2] https://www.kaggle.com/c/asap-sas

Proceedings of the 27th International Conference on Computational Linguistics, pages 1099–1109
Santa Fe, New Mexico, USA, August 20-26, 2018.

2013 Shared Task (Dzikovska et al., 2013)). Although this increase in interest is well-motivated and has led to useful technical advances, it has also propagated the impression that the task of automated scoring can be a self-contained, moderately simple machine learning task requiring "only" sophisticated feature engineering (Somasundaran et al., 2015; Ghosh et al., 2016) or, more recently, complex neural network architectures (Taghipour and Ng, 2016; Dong et al., 2017; Riordan et al., 2017; Tay et al., 2017). Our motivation for writing this paper is to highlight the differences between automated scoring as a shared task and automated scoring as an area of NLP research that serves the end goal of building and deploying accurate and unbiased scoring models for use in actual assessments or classrooms. For the latter, we argue that it is essential for NLP researchers to interact with and, indeed, collaborate with additional parties who are impacted by and contribute to automated scoring.[3]

To be clear, the issues we highlight are not simply a result of "operationalizing" NLP research. Instead, we claim that in order to make meaningful contributions in the area of automated scoring, these issues must be thought about and discussed as part of the NLP research and development process from the start. Our aim is not to strike a prescriptive tone. We acknowledge that despite the best of intentions, practical and business considerations can sometimes impact the extent to which such desiderata can be incorporated into the research process. However, our position is that these are extremely important problems to solve and even limited engagement with and progress towards solving these problems constitutes an important step. As more and more educational institutions and start-ups opt into automated scoring of educational assessments — which has the potential to impact the lives and livelihoods of members of the public — it is incumbent upon us as NLP researchers to strive towards fair, accountable, and transparent utilization of our research (Barocas et al., 2017).

3 Related Work

Many of the issues we refer to in the previous section are commonly discussed in the related fields of psychometric measurement and educational research. For example, Table 1 lists some of the standards in the 2014 Standards for Educational and Psychological Testing which are relevant for automated scoring. These standards help ensure fair and valid tests and provide guidelines for all aspects of assessment development. In addition, these communities have also been working towards best practices for the use of automated scoring (Clauser et al., 2002; Yang et al., 2002; Williamson et al., 2012; Bridgeman et al., 2012; Bejar et al., 2016). However, since there are limited opportunities for that community to interact with the NLP community, our goal is to bring them to the attention of the mainstream NLP audience as a first step towards the cross-discipline collaboration that we argue for in this paper.

In addition, there have also been efforts in the NLP community itself to highlight how NLP research on automated scoring can be situated in cross-disciplinary contexts (e.g. the workshops on Innovative Use of NLP for Building Educational Applications (BEA) and the workshops on NLP Techniques for Educational Applications (NLPTEA)) (Napoles and Callison-Burch, 2015; Wilson and Martin, 2015; Burstein et al., 2016; Beigman Klebanov et al., 2016; Zalmout et al., 2016; Lugini and Litman, 2017; Pado, 2017; Yaneva et al., 2017). However, the topics of these workshops tend not to always make it into the mainstream NLP conference discussions. Yet, at the same time there has been an increasing interest in automated scoring techniques at main NLP conferences. It is important not to lose the link between developing new and improved NLP techniques for automated scoring and the contexts in which they are generally applied.

4 Perspectives on Automated Scoring

Before we describe our position, we describe the various entities who are likely to be affected by automated scoring systems, and their corresponding perspectives on the use of automated scoring. All of these entities have a stake in making sure that the automated scoring system performs in line with their own expectations and, therefore, we refer to them as "stakeholders" from this point on.

[3]Of course, these kinds of challenges are not unique to the use of automated scoring in the educational domain. Other complex, real-world applications involving NLP also have several stakeholders. For example, in the health care industry, NLP is being used to support clinical decision-making where stakeholders can include patients, medical professionals, caregivers, and healthcare institutions.

Standard 3.8	"When tests require the scoring of constructed responses, test developers and/or users should collect and report evidence of the validity of score interpretations for relevant subgroups in the intended population of test takers for the intended uses of the test scores."
Standard 4.19	"When automated algorithms are to be used to score complex examinee responses, characteristics of responses at each score level should be documented along with the theoretical and empirical basis for the use of the algorithms."
Standard 6.8	"Those responsible for test scoring should establish scoring protocols …When scoring of complex responses is done by computer, the accuracy of the algorithm and processes should be documented."
Standard 6.9	"Those responsible for test scoring should establish and document quality control processes and criteria. Adequate training should be provided. The quality of scoring should be monitored and documented. Any systematic source of scoring errors should be documented and corrected."

Table 1: Some standards in the 2014 Standards for Educational and Psychological Testing that are relevant for automated scoring.

- **Score Users**. The people to whom any assessment is administered (test takers) are important stakeholders since any decisions made about the scoring of the assessment affect them directly. This is particularly true in cases where the assessment is likely to have a significant impact on the test-takers' futures, e.g., by contributing to a decision about whether to admit them into a college or graduate school (IELTS, GRE, TOEFL, SAT, ACT, etc.), or a decision whether to grant them a license to engage in a professional activity such as teaching (PRAXIS). If an automated scoring system is being employed to score their responses to such critically positioned assessments, test-takers want to ensure that such a system measures the same set of characteristics and skills that a well-trained human scorer would[4] and that such measurements are calculated accurately based on their responses. In addition, the test-takers also place emphasis on receiving their scores quickly and on a *score report* providing useful information on why they received the particular scores that they did and what, if anything, they can do to improve their scores if they decide to retake the assessment. Another important consumer of test scores are institutions who use the test scores to make decisions. For example, universities may use test scores to make placement decisions; immigration authorities may use test scores to make visa decisions; school districts and states may use test scores to make funding and policy decisions. Such institutions want to ensure that the scores from the test are valid and reliable and that any automated scoring component does not introduce any biases towards any particular subgroup of test takers.

- **Teachers**. Teachers in classrooms are also likely to be affected by the decision to use automated scoring systems. It is important to them as stakeholders that scores from automated systems are not used inappropriately. For example, if the assessment — and the automated scoring system — are only designed to measure the students' writing proficiency, the scores assigned by the system should not be used for a different purpose, e.g., to assess the teacher's teaching abilities. Teachers are also impacted if automated systems are used directly in the classroom, e.g., to provide feedback to students in the context of *formative* assessments (informal assessments conducted by teachers in the classroom to improve how students learn). For such cases, the teachers want to ensure that the feedback provided by the system is reasonably accurate, does not lead the students astray from the actual learning goal, and encourages engagement with the material being taught.

- **Subject-matter Experts**. Subject-matter experts, also known as assessment developers, write the questions that are included in the assessments. In addition, they also assemble specific questions

[4] An automated system cannot "read" the response in the same way that a human can but it can use features that are reasonable approximations for factors that human scorers consider in their evaluation.

into assessments while taking into account that the chosen questions should cover a wide range of skills that are to be measured by the assessment (usually known as the *construct*) and that different questions try to measure complementary aspects of such skills. As part of the assessment design, they also create what is called a *scoring rubric* - a document that attempts to describe specific and consistent criteria for how human scorers should score responses to each question in the assessment. Rubrics tend to be complex and subjective, particularly for assessments that contain relatively open-ended writing or content-based questions. Such experts would like to ensure that any automated scoring system deployed to score the assessments they have designed pays attention to the scoring rubric and that only construct-relevant information is used by the system during the scoring process. For example, although the length of a response to a question designed to measure the test-taker's writing proficiency might correlate very highly with its score, it is not actually relevant to the mental construct of writing proficiently and, therefore, should not be used as a feature in the automated scoring system (Bejar, 2017).

- **Business Units**. It takes significant resources to develop, administer, and score well-designed assessments. Therefore, the institutions undertaking this process are more likely to be educational technology companies with dedicated staff for assessment development, psychometric analysis, and natural language processing. Therefore, there is a business aspect to educational assessments in addition to research. For example, business units at such companies might comprise of people who try to procure state and federal contracts for developing and scoring K-12 assessments. When it comes to automated scoring systems, such units place emphasis on more practical business aspects of system development, such as building systems that are fast (e.g., with a short turn-around time between the submission of the response and the production of the score), cost-effective to deploy and maintain in an operational setting, and have little to no measurable impact on the overall validity and reliability of the assessment.

- **NLP Researchers & Developers**. NLP researchers and developers such as the authors (and, most likely, the readers) of this paper tend to have a different perspective when it comes to building automated scoring systems. The automated scoring system should perform accurately where accuracy is generally defined as agreement with human scores. Secondly, the system should not only build on top of state-of-the-art ideas and tools from the field but, if possible, should also advance the field forward by sharing and disseminating tools, lessons, and ideas at conferences and workshops. Another important consideration is the modularity of the system that not only allows replacement of NLP components (e.g., taggers and parsers) with newer and better-performing versions but also allows new scoring features to easily be incorporated into the system. Finally, the system should be easy to maintain and well-documented to make it easy for new developers and researchers to become familiar with the system.

Since each set of stakeholders is trying to optimize the automated scoring system for a different set of criteria, it is only natural that many of the above perspectives can be at odds with each other. Below we specifically outline the conflicts between the perspectives of the **NLP researchers** and the other four sets of stakeholders.

(a) **vs. Business Units**. Business units might sometimes place a greater emphasis on getting systems to market faster and with a limited investment depending on the budget available. However, in order to build a system that has a reasonably high agreement with human scores and is more likely to generalize to new and unexpected responses in the field, NLP researchers might require additional time as well as investment. For example, annotation of additional responses may be required for use as training data, or an existing research technique from the literature may need to be adapted for the domain before deployment.

(b) **vs. Score Users**. As described above, one of the most important considerations for test-takers is a reasonably clear explanation of why they received the particular scores that they did. The NLP

researchers optimizing for agreement with human scores might lean towards using more sophisticated machine learning models such as SVMs with non-linear kernels and deep neural networks. However, such models do not really lend themselves to *post-hoc interpretability* (Lipton, 2016). Although interpretability is an active area of research in the machine learning literature (Ribeiro et al., 2016; Koh and Liang, 2017; Doshi-Velez and Kim, 2017), it currently lags far behind the research on machine learning methods. Ensuring that there are no biases in automated scores – important for institutions using test scores to make decisions – is a topic that sees little discussion in the NLP literature. This is partly driven by a lack of demographic data available in publicly available datasets, as well as perhaps a focus on empirical accuracy.

(c) **vs. Subject-matter Experts**. Subject-matter experts or assessment developers want to ensure that all the hard work that has been done on their end to develop a valid (actually measuring what it is supposed to measure) and reliable assessment (scores are comparable across repeated administrations) is not undone by an automated scoring system that is either using construct-irrelevant factors as features or using a set of features that, taken together, have low *construct coverage*, i.e., they only measure part of the skill being assessed. It is difficult for NLP researchers to convert the salient aspects of a complicated – and subjective – document like the scoring rubric into features that are reasonably efficient to compute. Although the conversion can be aided by asking human scorers how they mentally translate the rubric into specific scoring decisions, humans are not as interpretable as one might think (Lipton, 2016).

(d) **vs. Teachers**. To make sure that automated scoring (or feedback) systems behave as expected if deployed for in-classroom use, NLP researchers would like to conduct research studies with such systems in real classrooms in order to collect useful data, e.g., written or spoken responses, student behavior, and indicators of engagement which can then be used to improve the system further (Burstein et al., 2016; Burstein and Sabatini, 2016; Madnani et al., 2016). However, teachers want to ensure that such systems are sufficiently nuanced — and not too primitive — to handle interactions with students and do not lead to students being distracted instead of learning. Furthermore, it takes time to build up a level of trust between the teachers and NLP researchers as a system is being fine-tuned and developed.

It is evident from the above discussion that trying to cater to everyone is akin to solving a difficult constraint satisfaction problem. For example, if NLP researchers want to build a more interpretable automated scoring system that can provide more useful feedback to test-takers, it requires investing more money and time which might need to be negotiated with the business units. Or, if subject-matter experts design an assessment with more intricate, open-ended questions in order to accurately assess the required set of skills, the corresponding automated scoring system would likely require more time and resources to build and potentially be less transparent.

5 Our Position

While NLP and machine learning techniques form a core part of automated scoring systems, there are several other non-NLP considerations that need to be taken into account when designing and developing such techniques. It is important to remember the perspectives of the other stakeholders — as outlined in the previous section — when making decisions during the NLP component development, and not simply focus on having the most accurate, or fastest automated scoring systems. While it is admirable to take advantage of the recent availability of relevant data and develop novel and more sophisticated NLP techniques to advance the field, sometimes it is necessary to take a step back and ensure that the NLP advances are also aligned with the interests of other stakeholders in automated scoring.

For example, the NLP researchers may find through ablation studies that certain computationally complex features are not contributing much to the overall accuracy of the model. They may choose to simplify the processing pipeline and resulting models by excluding those features, at very little cost to overall accuracy (on some held-out evaluation data). However, if those features are measuring important aspects

of the construct, removing them weakens the validity and construct coverage of the system, since now the system omits measuring an important aspect of what is being measured by humans, as defined in the scoring rubric. This is very important for both subject-matter experts (who will assume that any automated system is at least trying to measure the same thing that humans are) and business units (who require a valid test).

6 Case Studies

To explain our position more concretely, we describe three hypothetical case studies involving automated scoring or feedback systems. We focus on some of the stakeholder interactions that we believe are critical for successful deployment of automated scoring systems and provide suggestions for best practices. Note that many of our suggestions are adapted from the best practices that are already recommended by research practitioners in the educational measurement community.

6.1 Adding Automated Scoring to an Existing Assessment

The decision to add automated scoring to an existing assessment can be initiated by the business units as a potential cost-saving measure. It can also be initiated by the NLP researchers who believe they have developed a system that can accurately score a particular type of spoken or written response. In either case, a number of considerations need to be taken into account before automated scoring can be added to an existing assessment.

If a business unit wants to add automated scoring for an existing assessment, they should typically first approach the NLP researchers to estimate the effort involved in developing automated scoring capabilities for the specific question types contained in the assessment (e.g. essays that measure writing quality or free-text responses that measure knowledge of some content area). NLP researchers will assess the feasibility of automated scoring for the requested item types. At this point, it is critical to engage the subject-matter experts in order to fully understand the construct being measured, the scoring rubric and any supplemental scoring guidelines, as well as to get access to any training materials used to train human scorers. Without engaging the subject-matter experts it is all too easy to make assumptions about the assessment based on observations made from a limited amount of scored data. This can lead to automated scoring systems that measure the construct inaccurately or in a limited fashion, that handle aberrant responses incorrectly, and that could ultimately lead to unfairly scored assessments yielding significant consequences for the test-takers.

If NLP researchers initiate the request to add automated scoring to an assessment, they should have already connected with the subject-matter experts to ensure that they have built a system that adequately measures the correct construct. It is important that they also consider other aspects of the assessment and communicate with all relevant stakeholders. For example, if an assessment has a low number of test-takers each year, then the amount of data available to monitor[5] the automated system may be too low. If the number of responses available for monitoring is low, then the risks are that a sample small enough for the monitoring to be cost effective will not provide statistically meaningful monitoring metrics. Conversely, a sample large enough to provide statistically meaningful metrics would effectively offset any potential cost-savings obtained from automated scoring. This consideration would be very important to the business units who have to fund both the setup costs of integrating automated scoring into an existing assessment, as well as the ongoing maintenance costs.

Finally, for any automated scoring system that is proposed, NLP researchers need to take into account ethical considerations regarding fairness and validity and evaluate the system on dimensions other than just the agreement with human scores. For example, it is critical to evaluate whether the system is biased towards certain sub-populations of test-takers. Aggregated metrics of agreement with human scores (such as Pearson correlation or Cohen's kappa) will not be able to capture such biases or fairness issues. Madnani et al. (2017) proposed an open-source tool, RSMTool[6], as an initial step in this direction. In

[5] Monitoring is an important aspect of high-stakes automated scoring systems, wherein a random sample of the responses scored by the automated system are also scored by an expert human scorer to ensure that the system is performing as expected.

[6] https://github.com/EducationalTestingService/rsmtool

addition to the agreement metrics with human scorers, RSMTool also evaluates how well the automated scoring system performs for each sub-population represented in the data. Of course, one key assumption underlying this type of evaluation is that the demographic information about the candidates is available and permitted to be used as part of the evaluation pipeline.

6.2 Creating a New Assessment that Includes Automated Scoring

This particular scenario offers many more opportunities for a collaborative and multi-perspective development process to be adopted right from the start as compared to the previous case study. As a business unit puts together the plans for a new assessment, automated scoring is often a desired component (usually for perceived cost-saving measures). It is important at this juncture for the subject-matter experts and the NLP researchers to collaborate in order to understand the types of questions that will be included in the new assessment, and which ones might actually be suitable for being scored automatically. It is important for the NLP researchers to understand the specific constructs being targeted by the new assessment and advise the subject-matter experts where construct-irrelevant differences might impact automated scoring.

For example, a question might use a passage about a young student with a name that is not likely to be familiar to the population of students taking the test. This means that the test-takers may guess the gender of this student, resulting in a range of pronouns appearing in the sample responses from which automated scoring models might be built. This unnecessarily "dilutes" the vocabulary of the response pool with construct-irrelevant variation. On the other hand, giving the hypothetical student a name that has a widely acknowledged gender associated with it will help limit this kind of variation without any impact on the construct coverage or validity of the question.

At the same time, while subject-matter experts are developing the questions, it is important for the NLP researchers to assess the feasibility of automated scoring for the question types being considered for automated scoring. If a question would require NLP techniques that are in very early stages of research or have not yet been fully tested, this information needs to be shared as quickly as possible with the business units so that they can build contingency plans into their budgets and timetables.

Finally, a comprehensive evaluation of the sort described in the previous case study is still warranted in this scenario since ensuring the accuracy and fairness of an automated scoring system does not depend on whether it is being deployed for a entirely new assessment or for an existing one.

6.3 Including Automated Scoring in a Classroom Setting

It is crucial that NLP researchers engage with both teachers as well as students when developing tools to be deployed in an authentic classroom setting, e.g., tools that can provide feedback on students' writing or content knowledge. An NLP researcher designing and implementing a classroom tool in isolation is not likely to be successful, no matter the accuracy or novelty of the underlying NLP techniques. Before developing any tools, the NLP researchers need to fully understand the problem they are trying to solve by engaging teachers to find out how NLP technology can be integrated in a supportive fashion to their teaching curricula. Ideally, classroom tools should be built in an iterative fashion, by learning what features or techniques improve student engagement and learning and which ones do not. An example study that evaluates the effect of a new tool in an authentic classroom setting has many non-trivial (and non-NLP-specific) components:

- Identify a representative sample of schools/classes/students for the study
- Conduct surveys of teachers and control groups
- Conduct teacher training
- Analyze teachers' daily/weekly logs
- Conduct classroom observations
- Conduct pre- and post-study evaluations

- Analyze tool usage logs

This requires a lot of time and effort both from the teachers as well as the NLP researchers, but at the end of the day is more likely to lead to truly useful tools that can have a positive impact on the classroom learning process and improve students' understanding of the material being taught by the teacher.

7 Conclusion

Our goal in this paper is to bring attention to the inherent complexity of automated scoring of student responses, given the large number of stakeholders, often times with different priorities. We take the position that in order to build fair and accurate automated scoring systems — especially for use in high-stakes assessments where the consequences of being unfair and inaccurate can be severe for the test-takers — NLP researchers must incorporate the perspectives of other stakeholders into the research and development process and avoid working in a bubble surrounded only by scored data and machine learning algorithms. We feel encouraged that our position is increasingly shared by many educational NLP researchers, as is evident by publications in more focused educational NLP workshops, and hope that all NLP researchers will keep this in mind as they develop new and exciting techniques for automated scoring that could be deployed for operational use. We hope to see more and more publications at NLP venues that describe these new advances while at the same time taking into account the perspectives of many of the stakeholders described in this paper.

In closing, we would like to make some concrete suggestions – informed by our position on automated scoring – that can actually apply more broadly to the field:

1. Conference and workshop organizers should strive to include industry tracks with published proceedings so that issues of the sort presented in this paper can be discussed and appropriate best practices can be developed. In addition, this would create a resource that the NLP community at large can go to when considering issues that affect the transition of NLP technology from theory into practice. More panels at conferences with both industry practitioners and academic researchers as panelists will also be helpful.

2. Shared task organizers should provide more context around the data and the specific task. As an example, for automated scoring, additional metadata can be included, if possible, to encourage more comprehensive evaluations of the sort described in §6.1 and §6.2.

3. Companies that have experience with transitioning research into practice should share more of their experiences publicly. Such companies should also offer internship positions for students to help expose them to the challenges that accompany such a transition.

4. Faculty members should encourage students to attend industry panels at conferences and to actively pursue internships in the industry in order to learn about what it takes to for academic research to be deployed in practice. This would likely lead to better dissertations and publications with improved evaluations and nuanced discussions.

Acknowledgments

We would like to thank Jill Burstein, Yoko Futagi, Beata Beigman Klebanov, and the three anonymous reviewers for useful insights and suggestions on how to improve the paper.

References

American Educational Research Association, American Psychological Association, National Council on Measurement in Education, et al. 2014. *Standards for Educational and Psychological Testing*. American Educational Research Association.

Solon Barocas, Sorelle Friedler, Joshua Kroll, Berk Ustun, Suresh Venkatasubramanian, and Hanna Wallach, editors. 2017. *Proceedings of the ACM SIGKDD Workshop on Fairness, Accountability, and Transparency in Machine Learning*.

Sumit Basu, Chuck Jacobs, and Lucy Vanderwende. 2013. Powergrading: a Clustering Approach to Amplify Human Effort for Short Answer Grading. *Transactions of the Association for Computational Linguistics (TACL)*, 1:391–402.

Beata Beigman Klebanov, Jill Burstein, Judith Harackiewicz, Stacy Priniski, and Matthew Mulholland. 2016. Enhancing stem motivation through personal and communal values: Nlp for assessment of utility value in student writing. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 199–205, San Diego, CA, June. Association for Computational Linguistics.

Isaac I Bejar, Robert J Mislevy, and Mo Zhang. 2016. Automated Scoring with Validity in Mind. *The Wiley Handbook of Cognition and Assessment: Frameworks, Methodologies, and Applications*, page 226.

Isaac I Bejar. 2017. Threats to Score Meaning in Automated Scoring. *Validation of Score Meaning for the Next Generation of Assessments: The Use of Response Processes*, page 75.

Jared Bernstein, A. Van Moere, and Jian Cheng. 2010. Validating Automated Speaking Tests. *Language Testing*, 27(3):355–377.

Brent Bridgeman, Catherine Trapani, and Yigal Attali. 2012. Comparison of Human and Machine Scoring of Essays: Differences by Gender, Ethnicity, and Country. *Applied Measurement in Education*, 25(1):27–40.

Steven Burrows, Iryna Gurevych, and Benno Stein. 2015. The Eras and Trends of Automatic Short Answer Grading. *International Journal of Artificial Intelligence in Education*, 25(1):60–117.

Jill Burstein and John Sabatini. 2016. The Language Muse Activity Palette: Technology for Promoting Improved Content Comprehension for English Language Learners. In *Adaptive Educational Technologies for Literacy Instruction*, chapter 17, pages 275–280. Taylor & Francis, Routledge: NY.

Jill Burstein, Karen Kukich, Susanne Wolff, Chi Lu, Martin Chodorow, Lisa Braden-Harder, and Mary Dee Harris. 1998. Automated Scoring Using A Hybrid Feature Identification Technique. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 206–210, Montreal, Quebec, Canada, August. Association for Computational Linguistics.

Jill Burstein, Martin Chodorow, and Claudia Leacock. 2004. Automated Essay Evaluation: The Criterion Online Writing Service. *AI Magazine*, 25(3):27.

Jill Burstein, Norbert Elliot, and Hillary Molloy. 2016. Informing Automated Writing Evaluation using the Lens of Genre: Two studies. *CALICO journal*, 33(1):117.

Brian E. Clauser, Michael T. Kane, and David B. Swanson. 2002. Validity Issues for Performance-Based Tests Scored With Computer-Automated Scoring Systems. *Applied Measurement in Education*, 15(4):413–432.

Fei Dong, Yue Zhang, and Jie Yang. 2017. Attention-based Recurrent Convolutional Neural Network for Automatic Essay Scoring. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 153–162, Vancouver, Canada, August. Association for Computational Linguistics.

Finale Doshi-Velez and Been Kim. 2017. Towards A Rigorous Science of Interpretable Machine Learning. *CoRR*, abs/1702.08608.

Myroslava O. Dzikovska, Rodney Nielsen, Chris Brew, Claudia Leacock, Danilo Giampiccolo, Luisa Bentivogli, Peter Clark, Ido Dagan, and Hoa Trang Dang. 2013. SemEval-2013 Task 7: The Joint Student Response Analysis and 8th Recognizing Textual Entailment Challenge. **SEM 2013: The First Joint Conference on Lexical and Computational Semantics*.

Debanjan Ghosh, Aquila Khanam, Yubo Han, and Smaranda Muresan. 2016. Coarse-grained Argumentation Features for Scoring Persuasive Essays. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 549–554, Berlin, Germany, August. Association for Computational Linguistics.

Pang Wei Koh and Percy Liang. 2017. Understanding Black-box Predictions via Influence Functions. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894, International Convention Centre, Sydney, Australia, 06–11 Aug. PMLR.

Zachary C. Lipton. 2016. The Mythos of Model Interpretability. In *Proceedings of the ICML Workshop on Human Interpretability in Machine Learning*.

Luca Lugini and Diane Litman. 2017. Predicting specificity in classroom discussion. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–61, Copenhagen, Denmark, September. Association for Computational Linguistics.

Nitin Madnani, Jill Burstein, John Sabatini, Kietha Biggers, and Slava Andreyev. 2016. Language muse: Automated linguistic activity generation for english language learners. In *Proceedings of ACL-2016 System Demonstrations*, pages 79–84, Berlin, Germany, August. Association for Computational Linguistics.

Nitin Madnani, Anastassia Loukina, Alina von Davier, Jill Burstein, and Aoife Cahill. 2017. Building Better Open-Source Tools to Support Fairness in Automated Scoring. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 41–52, Valencia, Spain, April. Association for Computational Linguistics.

Courtney Napoles and Chris Callison-Burch. 2015. Automatically scoring freshman writing: A preliminary investigation. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 254–263, Denver, Colorado, June. Association for Computational Linguistics.

Ulrike Pado. 2017. Question difficulty – how to estimate without norming, how to use for automated grading. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–10, Copenhagen, Denmark, September. Association for Computational Linguistics.

Ellis B. Page. 1966. The Imminence of ... Grading Essays by Computer. *The Phi Delta Kappan*, 47(5):238–243.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144.

Brian Riordan, Andrea Horbach, Aoife Cahill, Torsten Zesch, and Chong Min Lee. 2017. Investigating Neural Architectures for Short Answer Scoring. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 159–168, Copenhagen, Denmark, September. Association for Computational Linguistics.

Swapna Somasundaran, Chong Min Lee, Martin Chodorow, and Xinhao Wang. 2015. Automated Scoring of Picture-based Story Narration. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 42–48, Denver, Colorado, June. Association for Computational Linguistics.

Md Arafat Sultan, Cristobal Salazar, and Tamara Sumner. 2016. Fast and Easy Short Answer Grading with High Accuracy. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1070–1075, San Diego, California, June. Association for Computational Linguistics.

Kaveh Taghipour and Hwee Tou Ng. 2016. A Neural Approach to Automated Essay Scoring. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1882–1891, Austin, Texas, November. Association for Computational Linguistics.

Yi Tay, Minh C. Phan, Luu Anh Tuan, and Siu Cheung Hui. 2017. SkipFlow: Incorporating Neural Coherence Features for End-to-End Automatic Text Scoring. *CoRR*, abs/1711.04981.

David M. Williamson, Xiaoming Xi, and F. Jay Breyer. 2012. A Framework for Evaluation and Use of Automated Scoring. *Educational Measurement: Issues and Practice*, 31(1):2–13.

Joshua Wilson and Trish Martin. 2015. Using pegwriting® to support the writing motivation and writing quality of eighth-grade students: A quasi-experimental study. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 179–189, Denver, Colorado, June. Association for Computational Linguistics.

Victoria Yaneva, Constantin Orasan, Richard Evans, and Omid Rohanian. 2017. Combining multiple corpora for readability assessment for people with cognitive disabilities. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 121–132, Copenhagen, Denmark, September. Association for Computational Linguistics.

Yongwei Yang, Chad W. Buckendahl, Piotr J. Juszkiewicz, and Dennison S. Bhola. 2002. A Review of Strategies for Validating Computer-Automated Scoring. *Applied Measurement in Education*, 15(4):391–412, oct.

Nasser Zalmout, Hind Saddiki, and Nizar Habash. 2016. Analysis of foreign language teaching methods: An automatic readability approach. In *Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA2016)*, pages 122–130, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Klaus Zechner, Derrick Higgins, Xiaoming Xi, and David M. Williamson. 2009. Automatic Scoring of Non-native Spontaneous Speech in Tests of Spoken English. *Speech Communication*, 51(10):883–895.

Torsten Zesch, Michael Wojatzki, and Dirk Scholten-Akoun. 2015. Task-Independent Features for Automated Essay Grading. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 224–232, Denver, Colorado, June. Association for Computational Linguistics.

Aspect and Sentiment Aware Abstractive Review Summarization

Min Yang[1], Qiang Qu[1]*, Ying Shen[2], Qiao Liu[3], Wei Zhao[1], Jia Zhu[4]
[1] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[2] Peking University Shenzhen Graduate School
[3] University of Electronic Science and Technology of China
[4] South China Normal University
{min.yang,qiang,wei.zhao}@siat.ac.cn, shenying@pkusz.edu.cn
qliu@uestc.edu.cn, jzhu@m.scnu.edu.cn

Abstract

Review text has been widely studied in traditional tasks such as sentiment analysis and aspect extraction. However, to date, no work is towards the end-to-end abstractive review summarization that is essential for business organizations and individual consumers to make informed decisions. This work takes the lead to study the aspect/sentiment-aware abstractive review summarization in an end-to-end manner without hand-crafted features and templates by exploring the encoder-decoder framework and multi-factor attentions. Specifically, we propose a mutual attention mechanism to interactively learns the representations of context words, sentiment words and aspect words within the reviews, acted as an encoder. The learned sentiment and aspect representations are incorporated into the decoder to generate aspect/sentiment-aware review summaries via an attention fusion network. In addition, the abstractive summarizer is jointly trained with the text categorization task, which helps learn a category-specific text encoder, locating salient aspect information and exploring the variations of style and wording of content with respect to different text categories. The experimental results on a real-life dataset demonstrate that our model achieves impressive results compared to other strong competitors.

1 Introduction

User generated reviews on products are expanding rapidly with the emergence and advancement of e-commerce. These reviews are valuable to business organizations for improving their products and to individual consumers for making informed decisions. Unfortunately, reading though all the product reviews is hard, especially for the reviews that are lengthy and have low readability. It is therefore essential to provide coherent and concise summaries of user generated reviews. In this paper, we focus on generating abstractive summaries of product reviews. Abstractive text summarization is the task of generating a short and concise summary that captures the salient ideas of the source text. The generated summaries potentially contain new phrases and sentences that may not appear in the source text. Inspired by recent success of sequence-to-sequence (seq2seq) model in statistical machine translation, most abstractive summarization systems employ seq2seq framework to generate summaries (Nallapati et al., 2016; See et al., 2017; Paulus et al., 2017). In general, the seq2seq model firstly uses an encoder to convert the input text as a vector representation, and then it feeds this representation into a decoder to generate summary.

Despite the remarkable progress of previous studies, generating aspect/sentiment-aware summaries of product reviews remains a challenge in real-world for two reasons. (i) First, neural sequence-to-sequence models tend to generate trivial and generic summary, often involving high-frequency phrases. These summaries cannot capture the aspect and sentiment information from the product reviews which play a vital role in helping customers to make quick and informed decisions on certain products (Ly et al., 2011). (ii) According to what we observe, summary styles and words in different categories can significantly vary. However, existing methods apply a uniform model to generate text summaries for

* Corresponding author.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1110–1120
Santa Fe, New Mexico, USA, August 20-26, 2018.

the source documents in different categories, which easily miss or under represent salient aspects of the documents.

To alleviate the aforementioned limitations, we design a Multi-factor attention fusion network for aspect/sentiment-aware Abstractive Review Summarization (MARS). Our model exploits the recent success of the encoder-decoder framework to generate aspect/sentiment-aware review summaries. Specifically, a mutual attention mechanism is proposed to capture the correlation of context words, sentiment words and aspect words, which interactively learns attentions in the three kinds of words and generates the representations for contexts, sentiments and aspects separately. The text encoder is regularized with the co-training to perform an additional task of text categorization. The purpose of co-training is not to achieve the best performance on the text categorization (auxiliary task), but rather to compensate for the missing regularization requirement of abstractive summarization in the standard framework, learning a category-specific text encoder and improving the the quality of locating salient aspect information of the review. In addition, we explore three kinds of attentions (i.e., semantic attention, sentiment attention and aspect attention) to selectively attend to the context information when decoding summaries. Finally, we employ a reinforcement learning technique to maximize long-term rewards and address the exposure bias issue (Ranzato et al., 2015).

We summarize our main contributions as follows:

- We leverage text categorization task to learn better category-specific review representation for summarization. The variation of style and wording of summaries with respect to different text categorization are explored.
- We exploit the coordination of context words, sentiment words and aspect words via the mutual attention mechanism to learn aspect/sentiment-aware review representation. With this design, our model can also well represent the collocative sentiment and aspect words, which are helpful to learn sentiment and aspect attentions during decoding.
- We explore three kinds of attentions (i.e., semantic attention, sentiment attention and aspect attention) to selectively attend to the context information when decoding summaries.
- We employ the reinforcement learning technique (i.e., policy gradient) to directly optimize the model with respect to the non-differentiable ROUGE scores, moderating the exposure bias issue.
- The experimental results show that our model outperforms the competitors from both quantitative and qualitative perspectives.

2 Related work

In general, existing text summarization approaches can be categorized as extractive and abstractive. The extractive summarization copies representative sentences from the input (Zhang et al., 2012), while the abstractive summarization generates new phrases, possibly rephrasing or using words that are not in the original text (Rush et al., 2015). In this paper, we focus on abstractive text summarization systems.

Inspired by the recent success of the encoder-decoder framework in statistical machine translation, there has been increasing interest in generalizing the neural language model to the field of abstractive summarization (Rush et al., 2015; Chopra et al., 2016; Chen et al., 2016; Nallapati et al., 2016; See et al., 2017). For example, Rush et al. (2015) were the first to apply attention-based encoder-decoder model to abstractive text summarization, achieving state-of-the-art performance two sentence-level summarization datasets. Nallapati et al. (2016) proposed off-the-shelf attention encoder-decoder RNN that captured hierarchical document structure and identified the key sentences and keywords in the document. See et al. (2017) proposed a hybrid pointer-generator network that allowed both copying words from the source text via pointing, and generating words from a fixed vocabulary.

Several recent studies attempted to integrate the encoder-decoder RNN and reinforcement learning paradigms for abstractive summarization, taking advantages of both (Paulus et al., 2017; Liu et al., 2018). For example, Paulus et al. (2017) combined the maximum-likelihood cross-entropy loss with

rewards from policy gradient reinforcement learning to reduce exposure bias. Liu et al. (2018) proposed an adversarial process for abstractive text summarization, in which the generator is built as an agent of reinforcement learning.

There are also some studies working on summarization of product reviews (Li et al., 2010; Gerani et al., 2014; Di Fabbrizio et al., 2014; Gerani et al., 2016; Yu et al., 2016; Mason et al., 2016). For example, Gerani et al. (2014) proposed an abstractive summarization system to product reviews by applying a template-based NLG framework and taking advantage of the discourse structure of reviews. Yu et al. (2016) proposed a phrase-based approach which leveraged phrase properties to choose a subset of optimal phrases for generating the final summary.

Different from the previous work, this study focuses on generating sentiment/aspect-aware abstractive review summarization that may better fit users' needs by using encoder decoder framework with sentiment/aspect attentions.

3 Methodology

This section defines the key notations and briefly formulates the problem of this study. We suppose that a review x consists of k words $x^c = [w_1^c, w_2^c, ..., w_k^c]$, n aspect words $x^t = [w_1^t, w_2^t, ..., w_n^t]$, and m sentiment words $x^s = [w_1^s, w_2^s, ..., w_m^s]$. To prevent conceptual confusion, we use superscripts "s", "t" and "c" to indicate the variables that are related to sentiment words, aspect words and content, respectively. Each review x in the corpus has a category label y and a corresponding reference summary $Z = [z_1, z_2, \ldots, z_T]$, where T is the length of the reference summary.

Our model MARS consists of two tasks: the abstractive review summarization task and the text categorization task, both working on a shared document encoding layer. In this section, we elaborate the main components of MARS in detail.

3.1 LSTM Encoder

This section introduces our Mutual Attention Network (MAN) to learn better sentiment, aspect and context representation via interactive learning. MAN utilizes the attention mechanism associated with the sentiment and aspect words to capture important information from the input review and learn the sentiment/aspect-aware review representation. Further, MAN makes use of the interactive information from the input review to supervise the modeling of the sentiment and aspect words which are helpful to capture important information in summary generation.

3.1.1 Initial Document Representation

Each word w in the review is mapped to a low-dimensional embedding $e \in \mathbb{R}^d$ through a word embedding layer, where d denotes the embedding dimensionality. Then, we employ three independent LSTM networks to obtain the hidden states of context words, the sentiment words, and the aspect words. Formally, given the input word embedding e_t at time step t, the hidden state h_t can be updated with the previous hidden state h_{t-1}, which is computed by

$$i_t = \sigma(W_i e_t + U_i h_{t-1} + b_i) \tag{1}$$
$$f_t = \sigma(W_f e_t + U_f h_{t-1} + b_f) \tag{2}$$
$$o_t = \sigma(W_o e_t + U_o h_{t-1} + b_o) \tag{3}$$
$$c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \tag{4}$$
$$\tilde{c} = \tanh(W_c e_t + U_c h_{t-1} + b_c) \tag{5}$$
$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

where i_t, f_t, o_t, c_t are input gate, forget gate, output gate and memory cell, respectively. W and U denote weight matrices to be learned. b represents biases. σ is a sigmoid function and $\odot$ stands for element-wise multiplication. Hence, we can use the LSTM networks to obtain the hidden states $H^c = [h_1^c, h_2^c, ..., h_k^c] \in R^{k \times u}$ for context words, the hidden states $H^s = [h_1^s, h_2^s, ..., h_m^s] \in R^{m \times u}$ for sentiment words in the review, and the hidden states $H^t = [h_1^t, h_2^t, ..., h_n^t] \in R^{n \times u}$ for aspect words in the review, where u is

the size of hidden states for each LSTM unit. Then, we feed these hidden states to a mean-pooling layer to obtain the initial representation of context words, sentiment words, and aspect words in the review, respectively.

$$v^c = \sum_{i=1}^{k} h_i^s/k; \quad v^s = \sum_{i=1}^{m} h_i^c/m; \quad v^t = \sum_{i=1}^{n} h_i^t/n \tag{7}$$

3.1.2 Aspect/sentiment-aware Document Representation

Attention mechanism plays an important role in text modeling. Inspired by (Bahdanau et al., 2014; Ma et al., 2017), this section introduces the proposed mutual attention network (MAN) to learn a better sentiment-aware and aspect-specific document representation. In addition, the MAN model can also well represent the sentiment and aspect word representations. Formally, given the context word representations $[h_1^c, h_2^c, ..., h_k^c]$, the initial representations of sentiment and aspect words (i.e., v^s and v^t, respectively), the mutual attention mechanism generates the attention weight C_i of the context by

$$C_i = \frac{\exp(\rho([h_i^c; v^s; v^t]))}{\sum_{j=1}^{k} \exp(\rho([h_j^c; v^s; v^t]))} \tag{8}$$

where C_i indicates the importance of the i-th word in the context, and ρ is the attention function that calculates the importance of h_i^c in the context:

$$\rho([h_j^c; v^s; v^t]) = {U_c}^T \text{tanh}(W_c[h_j^c; v^s; v^t] + b_c) \tag{9}$$

where U_c and W_c are projection parameters to be learned, and b_c is the bias.

Only using the attention vector C cannot capture the interactive information of the context words and the aspect words (sentiment words), and lacks the ability of discriminating the importance of the words in the context. To make use of the interactive information between the context words and the aspect words (sentiment words), we also use the context words as attention source to attend to the aspect words (sentiment words). Similar to Eq. (8), we can calculate the attention vectors T and S for the aspect words and sentiment words as:

$$T_i = \frac{\exp(\rho([h_i^t; v^c; v^s]))}{\sum_{i=1}^{n} \exp(\rho([h_i^t; v^c; v^s]))} \tag{10}$$

$$S_i = \frac{\exp(\rho([h_i^s; v^c; v^t]))}{\sum_{i=1}^{n} \exp(\rho([h_i^s; v^c; v^t]))} \tag{11}$$

where ρ is the same as in Equation 9.

After computing the mutual attention vectors for the context words, aspect words and sentiment words, we can get the final context, aspect, sentiment representations emb^c, emb^s and emb^t based on the mutual attention vectors C, S and T by:

$$emb_x^c = \sum_{i=1}^{k} (C_i h_i^c), \quad emb_x^s = \sum_{i=1}^{m} (T_i h_i^s), \quad emb_x^t = \sum_{i=1}^{n} (S_i h_i^t) \tag{12}$$

Finally, we concatenate the context, aspect, sentiment representations to form the aspect/sentiment-aware review representation emb_x for review x:

$$emb_x = [emb_x^c, emb_x^t, emb_x^s] \tag{13}$$

3.2 Text Categorization

We feed the final document representation emb_x into a task-specific fully connected layer and a softmax classifier to predict the category distribution of the input document x:

$$\hat{y} = softmax(V_2 \cdot F_x), \; F_x = \text{tanh}(V_1 \cdot emb_x) \tag{14}$$

where V_1 and V_2 are projection parameters to be learned. We train this model by minimizing the cross-entropy between the predicted distribution $\hat{y}$ and the ground truth distribution y for each review in the training data:

$$J_{ML}^{\text{categ.}}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{D}\mathbf{I}(y_i = j)\log(\hat{y}_i) \tag{15}$$

where θ is the set of parameters of our model, D is the number of categories, N is the number of reviews in the training set, $\mathbf{I}(\cdot)$ is an indicator such that $\mathbf{I}(\text{true}) = 1$ and $\mathbf{I}(\text{false}) = 0$.

3.3 Abstractive Review Summarization

The abstractive review summarization subtask shares the same review representation module (encoder) with the text categorization subtask. The generation of summary Z is performed by a LSTM decoder.

3.3.1 Category-specific Review Representation

To generate category-specific summaries, the review representation emb_x are transformed to category-specific review embedding which is expected to capture category characteristics. Inspired by (Dong et al., 2014; Cao et al., 2017), we develop a category-specific transformation process to make the transformed review embedding hold the category characteristics information. Formally, our model transforms the review embedding emb_x to a category-specific review embedding $cemb_x$ by

$$cemb_x = \tanh(\mathbf{W}_\mu \times emb_x) \tag{16}$$

where $\mathbf{W}_\mu \in \mathbb{R}^d$ is the transformation matrix, d is the dimensionality of the category-specific document embedding. Note that we define the same dimensionality for both the document embedding and the category-specific document embedding.

To make the transformed embedding capture category-specific information, we develop the category-specific transformation matrix $\mathbf{W}_\mu$ according to the predicted product category. We introduce $|C|$ sub-matrices $(\mathbf{W}_\mu^1, \cdots, \mathbf{W}_\mu^{|C|})$, with each directly corresponding to one product category. Based on the predicted category derived from Eq. 14, the category-specific transformation matrix $\mathbf{W}_\mu$ is computed as the weighted sum of these sub-matrices: $\mathbf{W}_\mu = \sum_{i=1}^{|C|} \hat{y}\mathbf{W}_\mu^i$. In this way, $\mathbf{W}_\mu$ is automatically biased to the sub-matrix of the predicted category.

3.3.2 LSTM Decoder

Inspired by (See et al., 2017), the pointer-generator network is adopted as the decoder to generate summaries. The pointer-generator network allows both copying words from input text via pointing (P_{vocab}), and generating words from a fixed vocabulary (P_{gen}). Thus, the pointer-generator has the ability to produce out-of-vocabulary (OOV) words.

The category-specific review representation $cemb_x$ is used to initialize the hidden states s_0 of LSTM decoder. On each step t of decoding, the decoder receives the word embedding of the previous word w_{t-1} (while training, this is the previous word of the reference summary; at test time it is the previous word emitted by the decoder) and update its hidden state s_t:

$$s_t = \text{LSTM}(s_{t-1}, c_t, w_{t-1}) \tag{17}$$

The attention mechanism is used to calculate the attention weights a_t and context vector c_t. Attention mechaism is expected to take both context-sentiment and context-aspect correlations into consideration. The enhanced context vector c_t is aggregated by the representation of those informative words (see Eq. 21). In this paper, we explore three kinds of attention: semantic attention, sentiment attention and aspect attention. Details of these three kinds of attention are described as follows.

Semantic Attention Semantic attention simply applies the context representation itself as attention source. Following (Shimaoka et al., 2017), we apply a multi-layer perceptron (MLP) to compute semantic attention weights as follows:

$$a_{t,i}^{semantic} = \tanh(W_{a_1} h_i^c + U_{a_1} s_t + b_{a_1}) \tag{18}$$

where W_{a_1} and U_{a_1} are parameter matrices, b_{a_1} is bias parameter. The attention computed for context words are independent of the aspect/sentiment words. Hence, it is difficult for semantic attention to focus on those context words that are highly related to the aspects and sentiments.

Sentiment Attention In order to capture the correlation between sentiment words and the context, we take sentiment word representation emb_x^s as attention source to compute sentiment attention weights:

$$a_{t,i}^{sentiment} = \tanh(emb_x^s W_{a_2} h_i^c + U_{a_2} s_t + b_{a_2}) \tag{19}$$

where W_{a_2} is a bi-linear parameter matrix, U_{a_2} is parameter matrix, b_{a_2} is bias parameter.

Aspect Attention Aspect attention applies aspect word representation emb_x^t as attention query, which is expected to capture the correlations between aspect words and context words.

$$a_{t,i}^{aspect} = \tanh(emb_x^t W_{a_3} h_i^c + U_{a_3} s_t + b_{a_3}) \tag{20}$$

where W_{a_3} is a bi-linear parameter matrix, U_{a_3} is parameter matrix, b_{a_3} is bias parameter.

Attention Fusion We define the attention fusion of the semantic attention, sentiment attention and aspect attention at timestep t as:

$$a_{t,i} = \text{softmax}(\lambda_1 a_{t,i}^{semantic} + \lambda_2 a_{t,i}^{sentiment} + \lambda_3 a_{t,i}^{aspect}), \quad c_t = \sum_{i=1}^{k} a_{t,i} h_i^c \tag{21}$$

where λ_1, λ_2 and λ_3 are hyper-parameters that determines the weights of the three kinds of attentions. We set λ_1 = 0.5, λ_2 = λ_3 = 0.25. Note that for the documents that do not contain sentiment words and aspect words, we use only the semantic attention to distinguish the important information.

The context vector c_t is then concatenated with the decoder state s_t and fed through a linear layer and a *softmax* layer to compute the output probability distribution over a vocabulary of words at the current state:

$$P_{vocab}(w_t) = \text{softmax}(V_{d_2}(V_{d_1}[s_t, c_t] + b_{d_1}) + b_{d_2}) \tag{22}$$

where V_{d_1}, V_{d_2}, b_{d_1} and b_{d_2} are learnable parameters. The number of rows in V_{d_2} represents the number of words in the vocabulary.

On top of the LSTM decoder, we adopt the copy mechanism (See et al., 2017) to integrate the attention attribution into the final vocabulary distribution which is defined as the interpolation between two probability distributions:

$$P(w_t) = p_{gen} P_{vocab}(w_t) + (1 - p_{gen}) \sum_{i:w_i = w_t} a_{t,i} \tag{23}$$

where $p_{gen} \in [0, 1]$ is the switch variable for controlling generating a word from the vocabulary or directly copying it from the original review. If w is an out-of-vocabulary (OOV) word, then $P_{vocab}(w)$ is zero; if w does not appear in the source review, then $\sum_{i:w_i=w} a_{t,i}$ is zero. p_{gen} can be defined as:

$$p_{gen} = \text{sigmoid}(U_{d_1}^T c_t + U_{d_2}^T s_t + U_{d_3}^T w_{t-1} + b_{gen}) \tag{24}$$

where vectors U_{d_1}, U_{d_2}, U_{d_3} and scalar b_{gen} are learnable parameters.

A common way of training a summary generation model is to estimate the parameters by minimizing the negative log-likelihood of the training data:

$$J_{\text{ML}}^{\text{sum}}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \log P(w_t) \tag{25}$$

3.4 Training of MARS Model

Overall, MARS consists of two subtasks, each has training objective. To make the document embedding sensitive to the category knowledge, we train these two related task simultaneously. The joint multi-task objective function is minimized by:

$$J_{\text{ML}}(\theta) = \gamma_1 J_{ML}^{\text{categ.}}(\theta) + \gamma_2 J_{\text{ML}}^{\text{sum}}(\theta) \tag{26}$$

where γ_1 and γ_2 are hyper-parameters that determines the weights of L_1 and L_2. Here, we set $\gamma_1 = 0.2$, $\gamma_2 = 0.8$.

3.4.1 Policy Gradient Reinforcement Learning for Summary Generation

However, the maximum likelihood estimation (MLE) method suffers from two main issues. First, the evaluation metric is different from the training loss. For example, in summarization generation systems, the encoder-decoder models are trained using the cross-entropy loss but they are typically evaluated at test time using discrete and non-differentiable metrics such as BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004). Second, the input of the decoder at each time step is often the previous ground-truth word during training. Nevertheless, when generating summaries in the testing phase, the input of the next time step is the previous word generated by the decoder. This exposure bias (Ranzato et al., 2015) leads to error accumulation at the testing phase. Once the model generates a "bad" word, the error will propagate and accumulate with the length of the sequence.

To alleviate the aforementioned issues when generating summaries, we also optimize directly for ROUGE-1 since it achieves best results among the alternatives such as METEOR (Lavie and Agarwal, 2007) and BLEU (Papineni et al., 2002), by using policy gradient algorithm, and minimize the negative expected rewards:

$$J_{\text{RL}}^{\text{sum}}(\theta) = (r(\hat{z}) - r(z^s)) \sum_{t}^{T} \log p(z_t^s | Z_{1:t-1}^s; X) \tag{27}$$

where $r(\hat{z})$ is the reward of a greedy decoding generated sequence $\hat{z}$, and $r(z^s)$ is the reward of sequence z^s generated by sampling among the vocabulary at each step.

After pre-training the proposed model by minimizing the joint ML objective (see Eq. 26), we switch the model to further minimize a mixed training objective, integrating the reinforcement learning objective $J_{\text{RL}}^{\text{sum}}(\theta)$ with the original multi-task loss $J_{\text{ml}}(\theta)$:

$$J_{\text{mixed}}(\theta) = \beta J_{\text{ML}}(\theta) + (1 - \beta) J_{\text{RL}}^{\text{sum}}(\theta) \tag{28}$$

where β is a hyper-parameter, and we set β=0.1.

4 Experimental Setup

4.1 Datasets Description

We evaluate our model on Amazon reviews dataset from Stanford Network Analysis Project (SNAP) (McAuley and Leskovec, 2013). The raw dataset consists of 34,686,770 Amazon reviews from 6,643,669 users spanning different kinds of products such as books, video games, food, music. Each review mainly contains product ID, user information, ratings, a plaintext review and a review summary. There are 82 tokens on average of reviews. Since the dataset is too large to process all at once on our local computer, we randomly choose 50,000 reviews from each of the books, food, electronics, movies, music and clothing categories. For each of the six product categories, we use 80% instances as the training data, 10% instances as the validation data, and the remaining are used for testing.

Sentiment Words Collection The sentiment lexicon used in this paper is the combination of three popular sentiment lexicons: HowNet (Dong and Dong, 2006), MPQA (Wilson et al., 2005) and Liu's Lexicon (Hu and Liu, 2004), which consists of 11,017 words in total.

Aspect Words Collection The aspect dictionary contains about 800 words provided by the experts of each product categories, including product names and the aspect words of the products. Following (Yang et al., 2014), we apply topic model to extend the aspect dictionary by using a minimal set of seed words as prior knowledge to discover a much richer lexicon.

4.2 Baseline Methods

In the experiments, we compare our model with several strong baseline methods:

SummaRuNNer A simple recurrent network based sequence classifier (Nallapati et al., 2017) which facilitates interpretable visualization of its decisions. It is an extractive summarization model.

ABS Attentional encoder decoder recurrent neural networks for abstractive text summarization proposed in (Nallapati et al., 2016).

PGC The pointer-generator coverage networks proposed in (See et al., 2017) which copies words from the source text via pointing, while retaining the ability to produce novel words through the generator.

DeepRL The deep reinforced model (ML+RL version) proposed in (Paulus et al., 2017), which introduce a new objective function by combining the maximum-likelihood cross-entropy loss with rewards from policy gradient reinforcement learning to reduce exposure bias.

GANsum The generative adversarial network for abstractive summarization (Liu et al., 2018).

4.3 Implementation Details

We use 100-dimensional word2vec (Mikolov et al., 2013) vectors pre-trained on English Wikipedia data to initialize the word embeddings, and all out-of-vocabulary words are initialized by sampling from the uniform distribution U(-0.25,0.25) We initialize the recurrent weight matrices of LSTMs as random orthogonal matrices, and all the bias vectors are initialized to zero. The hidden state size of each LSTM is 300. We conduct mini-batch (with size 64) training using Adadelta optimization algorithm. Other hyperparameters include: learning rate 0.01, L2 regularization 0.001, dropout 0.2. For the pointer-generator models, we use a vocabulary of 50k words for both training and text data.

5 Experimental Results

In this section, we compare our model with baseline methods from quantitative and qualitative perspectives.

5.1 Quantitative Evaluation

Following the same evaluation as in (Nallapati et al., 2016), we compare our model with baseline methods in terms of ROUGE-1, ROUGE-2, ROUGE-L and Human evaluation.

ROUGE-N is a widely used evaluation metric for summarization tasks. It measures the consistency between n-gram occurrences in the generated and reference summaries. ROUGE-L compares the longest common sequence between the reference summary and the generated summary. We summarize the ROUGE scores of our model and the baseline methods in Table 1. PGC consistently perform better than ABS. This may be because that the copy mechanism used in PGC can handle the out-of-vocabulary words. DeepRL and GANsum are better than PGC, because they utilize reinforcement learning to alleviate the exposure bias problem and optimize directly the evaluation metrics. **Our model** performs even better than the strong competitors by leveraging text categorization task and aspect/sentiment attentions to improve the summarization results. This verifies the effectiveness of our model for abstractive review summarization.

We also perform human evaluation to evaluate the readability and quality of the generated summaries. We randomly select 200 test examples from the dataset. For each review, three human evaluators are invited to rank each summary generated by all 6 models based on their readability, where 1 indicates the lowest level of readability while 6 indicates the highest level. The experimental results based on human annotations are summarized in Table 1 (fifth column). MARS achieves the best results. Specifically, MARS improves 14.9% on the human evaluation score over the best result of baseline methods (i.e., PGC) on the test data.

Method	ROUGE-1	ROUGE-2	ROUGE-L	Human Evaluation
SummaRuNNer	80.53	62.23	82.64	2.72
ABS	78.53	60.92	79.21	1.68
PGC	81.84	64.15	83.18	2.94
DeepRL	82.12	65.09	84.31	3.22
GANsum	82.64	66.12	84.31	3.42
MARS	**84.13**	**68.28**	**86.15**	**3.93**

Table 1: Quantitative evaluation results. All our ROUGE scores have a 95% confidence interval of at most ± 0.25 as reported by the official ROUGE script.

Method	ROUGE-1	ROUGE-2	ROUGE-L	Human Evaluation
MARS	**84.13**	**68.28**	**86.15**	**3.93**
w/o categorization	81.33	65.25	82.35	2.76
w/o sentiment attention	82.44	66.23	84.14	3.24
w/o aspect attention	83.05	66.09	84.85	3.35
w/o semantic attention	82.12	65.84	83.42	3.06

Table 2: Ablation test results.

5.2 Ablation Study

To investigate the effect of each component of the MARS model, we also perform the ablation test of MARS in terms of discarding text categorization, aspect attention, sentiment attention, and semantic attention. The results are reported in Table 2. For human evaluation, three human evaluators are invited to rank each summary generated by all 5 models based on their readability, where 1 indicates the lowest level of readability while 5 indicates the highest level. Generally, all three factors contribute, and text categorization contribute most. This is within our expectation since the the text categorization helps learn better category-specific review representations. In addition, it also helps the learning of aspect and sentiment representations. The aspect and sentiment attention also makes great contribution to abstractive review summarization, verifying that the aspect and sentiment information plays a vital role in review summaries.

Input summary: Our pup has experienced allergies in forms of hotspots and itching from other dog foods. The cheap 'you can buy it anywhere' food not only have crazy preservatives in them but can cause health problems for your pets. This food works wonders on reducing allergies and our dog loves the food. **Ground-truth summary:** Great allergy sensitive dog food, dogs love it. **Summary by GANsum:** Great foods loves. **Summary by MARS:** Great dog food for reducing allergies.
Input summary: The NOOKColor is awesome- it plays music, is expandable, displays gorgeous color, allows you to surf the internet (not optimally, but the ability is there), and is mostly easy to navigate. From me, it gets a solid 4 stars. The problem is that as an eReader, it's already overpriced, and the Amazon sellers who offer it are asking ridiculous amounts. It's cheaper (and ships free) from Barnes and Noble. Amazon is usually No1 at everything, including price, and I already own two Kindles (I would buy the Kindle in color if it came that way) but the price being charged for the NOOKColor on Amazon is absurd. **Ground-truth summary:** Awesome eReader, Ridiculously Priced from this seller. **Summary by GANsum:** NOOKColor is awesome music gorgeous color Amazon. **Summary by MCARS:** Awesome NOOKColor, absurd Amazon price.

Table 3: Examples summaries.

5.3 Qualitative Evaluation

To evaluate the proposed model qualitatively, we reported some generated summaries by different models. Due to the limitation of space, we randomly choose two generated summaries by GANsum and our model from test data for comparison. The results are reported in Table 3. We observe that MARS tends to generate more specific and meaningful summaries in response to the given reviews. For example, our model successfully catches the sentiment word "Awesome" for the product *eReader* and the sentiment

word "absurd" for the *price* aspect of the eReader. The advantage of our model comes from its capability of integrating category, sentiment and aspect information into the attention encoder-decoder model.

6 Conclusion

We proposed MARS to improve the performance of aspect/sentiment-aware abstractive review summarization. A mutual attention mechanism was employed to integrated the sentiment and aspect information into the encoder-decoder abstractive summarizer. In addition, MARS leveraged text categorization to improve the performance of summarization by learning a category-specific review representation. We also explored three kinds of attentions (i.e., semantic attention, sentiment attention and aspect attention) to selectively attend to the context information when decoding summaries. The experimental results on a real-life dataset showed that our model substantially outperformed the strong competitive methods.

Acknowledgement

The work was partially supported by the CAS Pioneer Hundred Talents Program, the National Science Foundation of China (No.61772117), the National Science Foundation of China (No.61750110516).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2017. Improving multi-document summarization via text classification. In *AAAI*, pages 3053–3059.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *The 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

Giuseppe Di Fabbrizio, Amanda Stent, and Robert Gaizauskas. 2014. A hybrid approach to multi-document summarization of opinions in reviews. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, pages 54–63.

Zhendong Dong and Qiang Dong. 2006. *HowNet and the Computation of Meaning*. World Scientific.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *The Annual Meeting of the Association for Computational Linguistics*, pages 49–54.

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1602–1613.

Shima Gerani, Giuseppe Carenini, and Raymond T Ng. 2016. Modeling content and structure for abstractive review summarization. *Computer Speech & Language*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231. Association for Computational Linguistics.

Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Ying-Ju Xia, Shu Zhang, and Hao Yu. 2010. Structure-aware review mining and summarization. In *Proceedings of the 23rd international conference on computational linguistics*, pages 653–661. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *ACL Workshop on Text Summarization Branches Out*, volume 8.

Linqing Liu, Yao Lu, Min Yang, Qiang Qu, Jia Zhu, and Hongyan Li. 2018. Generative adversarial network for abstractive text summarization. In *Association for the Advancement of Artificial Intelligence*.

Duy Khang Ly, Kazunari Sugiyama, Ziheng Lin, and Min-Yen Kan. 2011. Product review summarization from a deeper perspective. In *Annual international ACM/IEEE joint conference on Digital libraries*, pages 311–314. ACM.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 4068–4074.

Rebecca Mason, Benjamin Gaska, Benjamin Van Durme, Pallavi Choudhury, Ted Hart, Bill Dolan, Kristina Toutanova, and Margaret Mitchell. 2016. Microsummarization of online reviews: An experimental study. In *AAAI*, pages 3015–3021.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *ACM conference on Recommender systems*, pages 165–172. ACM.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290. Association for Computational Linguistics.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Association for the Advancement of Artificial Intelligence*, pages 3075–3081.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *The 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

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

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389. Association for Computational Linguistics.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1073–1083. Association for Computational Linguistics.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural architectures for fine-grained entity type classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.

Min Yang, Dingju Zhu, M Rashed, and Kam-Pui Chow. 2014. Learning domain-specific sentiment lexicon with supervised sentiment-aware lda. *Frontiers in Artificial Intelligence and Applications*.

Naitong Yu, Minlie Huang, Yuanyuan Shi, et al. 2016. Product review summarization by exploiting phrase properties. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 1113–1124.

Lanbo Zhang, Yi Zhang, and Yunfei Chen. 2012. Summarizing highly structured documents for effective search interaction. In *SIGIR*. ACM.

Effective Attention Modeling for Aspect-Level Sentiment Classification

Ruidan He[†‡], **Wee Sun Lee**[†], **Hwee Tou Ng**[†], and **Daniel Dahlmeier**[‡]
[†]Department of Computer Science, National University of Singapore
[‡]SAP Innovation Center Singapore
[†]{ruidanhe,leews,nght}@comp.nus.edu.sg
[‡]d.dahlmeier@sap.com

Abstract

Aspect-level sentiment classification aims to determine the sentiment polarity of a review sentence towards an opinion target. A sentence could contain multiple sentiment-target pairs; thus the main challenge of this task is to separate different opinion contexts for different targets. To this end, *attention mechanism* has played an important role in previous state-of-the-art neural models. The mechanism is able to capture the importance of each context word towards a target by modeling their semantic associations. We build upon this line of research and propose two novel approaches for improving the effectiveness of attention. First, we propose a method for target representation that better captures the semantic meaning of the opinion target. Second, we introduce an attention model that incorporates syntactic information into the attention mechanism. We experiment on attention-based LSTM (Long Short-Term Memory) models using the datasets from SemEval 2014, 2015, and 2016. The experimental results show that the conventional attention-based LSTM can be substantially improved by incorporating the two approaches.

1 Introduction

Aspect-level sentiment classification is an important task in fine-grained sentiment analysis (Pang and Lee, 2008). Given a sentence and an opinion target (also called aspect expression) occurring in the sentence, the task aims to determine the sentiment polarity of the sentence towards the opinion target. An opinion target or target for short refers to a word or a phrase (a sequence of words) describing an aspect of an entity. For example, in the sentence "*This little place has a cute interior decor and affordable prices*", the targets are *interior decor* and *prices*, and they belong to the aspects *ambience* and *price* respectively.

Compared to document-level or sentence-level sentiment classification, the main challenge of aspect-level sentiment classification is to differentiate sentiments towards different targets when there are multiple targets in a sentence. For instance, the sentence "*The appetizers are ok, but the service is slow.*" expresses a neutral sentiment on the target *appetizers* and a negative sentiment on the target *service*. To this end, attention mechanism has played an important role in state-of-the-art neural models for this task. It assigns a positive weight att_i for each context word w_i, which can be interpreted as the probability that w_i is the right word to focus on when inferring the sentiment polarity of the given target. The weight att_i is generally computed as a function of the hidden representation $\mathbf{h_i}$ of w_i and the target representation $\mathbf{t}$ as follows:

$$att_i \propto f_{score}(\mathbf{h_i}, \mathbf{t}) \tag{1}$$

It has been shown that adding an attention model substantially improves the accuracy of aspect-level sentiment classification (Tang et al., 2016b; Wang et al., 2016; Liu and Zhang, 2017).

Our work builds upon this line of research. We propose two novel approaches for improving the effectiveness of attention models. The first approach is a new way of encoding a target which better captures the aspect semantics of the target expression. The target representation is crucial since attention weights are computed based on it as shown in Eq. 1. In representing the target, we are mapping a word or

Proceedings of the 27th International Conference on Computational Linguistics, pages 1121–1131
Santa Fe, New Mexico, USA, August 20-26, 2018.

a phrase into a vector in $\mathbb{R}^d$. Ideally, targets that are semantically similar should be mapped to vectors that are close together in $\mathbb{R}^d$. However, previous neural attention models simply map a target by averaging its component word vectors. This may work fine for targets that only contain one word but may fail to capture the semantics of more complex expressions, as also mentioned by Tang et al. (2016b). For example, we cannot obtain a good representation for "*hot dog*" by averaging the word vectors of "*hot*" and "*dog*". Hot would be close to words like warm or cold and dog would be close to animals like cat. The average would not be close to other food like burgers or spaghetti. Another example is "*hong kong style food*". As it consists of many words, the averaged word vector could be far away from "*food*" in vector space.

To address this problem, inspired by He et al. (2017), we instead model each target as a mixture of K aspect embeddings where we would like each embedded aspect to represent a cluster of closely related targets. We use an *autoencoder* structure to learn both the aspect embeddings as well as the representation of the target as a weighted combination of the aspect embeddings. The weight vector represents the probability distribution over aspects for the given target. The autoencoder structure is jointly trained with a neural attention-based sentiment classifier to provide a good target representation as well as a high accuracy on the predicted sentiment. We found the learned embeddings to be semantically meaningful, i.e., embeddings of words that are semantically related appear close to the same aspect embedding. For example, embeddings of the words *service*, *servers*, *staff*, and *courteous* appear close to the same aspect embedding, which we interpret to represent the aspect *service*.

Our second approach exploits syntactic information to construct a syntax-based attention model. The attention models used in previous works give equal importance to all context words. In that case, the computed attention weights rely entirely on the semantic associations between context words and the target. However, this may not be sufficient for differentiating opinions words for different targets. Instead, our syntax-based attention mechanism selectively focuses on a small subset of context words that are close to the target on the syntactic path which is obtained by applying a dependency parser on the review sentence.

We conducted experiments on attention-based LSTM models using the SemEval 2014, 2015, and 2016 datasets. The results show that attention-based LSTM can be substantially improved by incorporating our two proposed methods, and that the resulting model outperforms all baseline methods on aspect-level sentiment classification.

2 Related Work

Under supervised learning conditions, aspect-level sentiment classification is typically considered as a classification problem. Early works (Boiy and Moens, 2009; Jiang et al., 2011; Kiritchenko et al., 2014; Wagner et al., 2014) mainly used manually designed features such as sentiment lexicon, n-grams, and dependency information. However, these methods highly depend on the quality of the designed features, which is labor-intensive. With the advances of deep learning methods, various neural models (Dong et al., 2014; Nguyen and Shirai, 2015; Vo and Zhang, 2015; Tang et al., 2016a; Tang et al., 2016b; Wang et al., 2016; Zhang et al., 2016; Liu and Zhang, 2017; Chen et al., 2017; He et al., 2018) have been proposed for automatically learning target-dependent sentence representations for classification. The main idea behind these works is to develop neural architectures that are capable of learning continuous features without feature engineering and at the same time capturing the intricate relatedness between a target and context words.

Among these works, attention-based neural models have attracted growing interest due to their ability to explicitly capture the importance of context words. Tang et al. (2016b) have shown that a better sentence representation could be obtained by stacking multiple layers of attention. In the work of Wang et al (2016), a variant of attention-based LSTM was proposed. Chen et al. (2017) also adopts multiple layers of attention and aggregates the attention outputs through a recurrent neural network.

As aspect information is very beneficial, in some works (Wang et al., 2016; Cheng et al., 2017; Ma et al., 2018), an aspect embedding is directly used to capture the importance of context words through an attention mechanism, where the authors assume that the aspect label is provided as an input. Unlike them,

we do not assume that the aspects of each sentence are given. Instead, we propose to learn the probability distribution over aspects for the given target, and use the weighted summation of aspect embeddings for target representation. The probability distribution and the aspect embeddings are learned via an unsupervised objective, which is jointly trained with the neural attention-based sentiment classifier.

3 Model Description

We propose two approaches to improve the effectiveness of the attention mechanism. The approaches may be applied more generally but we use them on attention-based LSTM as it has been widely used in previous works for sentiment analysis (Chen et al., 2016; Wang et al., 2016; Chen et al., 2017; Liu and Zhang, 2017; Ma et al., 2018). We first give the task definition in (§3.1). Then, we briefly describe the architecture of attention-based LSTM (§3.2) and introduce the two proposed approaches (§3.3 & §3.4). Finally, we describe the overall architecture of our model for aspect-level sentiment classification and the training objective (§3.5).

3.1 Task Definition and Notation

Given a review sentence $s = (w_1, w_2, ..., w_n)$ consisting of n words, and an opinion target occurring in the sentence $a = (a_1, a_2, ..., a_m)$ consisting of a subsequence of m continuous words from s, aspect-level sentiment classification aims to determine the sentiment polarity of sentence s towards the opinion target a. When dealing with a text corpus, we begin by associating each word w with a continuous feature vector $\mathbf{e}_w \in \mathbb{R}^d$, also known as word embedding (Mikolov et al., 2013), where d denotes the embedding dimension. The vectors associated with the words correspond to the rows of a word embedding matrix $\mathbf{E} \in \mathbb{R}^{V \times d}$, where V is the vocabulary size.

3.2 Attention-based LSTM

We briefly describe a conventional attention-based LSTM in this subsection. Given a sequence of word embeddings $\{\mathbf{e}_{w_1}, \mathbf{e}_{w_2}, ..., \mathbf{e}_{w_n}\}$ of a sentence s, LSTM with trainable parameters θ_{lstm} makes use of three gates to discard or pass the information through time (Hochreiter and Schmidhuber, 1997), and outputs a sequence of hidden vectors $h = \{\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n\}$. The sentence representation $\mathbf{z}_s$ used for sentiment classification is then computed as the weighted summation of hidden vectors.

$$\mathbf{z}_s = \sum_{i=1}^{n} p_i \mathbf{h}_i \tag{2}$$

A positive weight p_i is computed for each $\mathbf{h}_i$, which can be interpreted as the probability that w_i is the right word to focus on when inferring the sentiment polarity of the opinion target a. The value p_i is computed by an attention model, which conditions on the hidden vector $\mathbf{h}_i$ as well as the target representation. In previous works (Tang et al., 2016b; Wang et al., 2016; Liu and Zhang, 2017), the attention process is usually described with the following equations:

$$p_i = \frac{\exp(d_i)}{\sum_{j=1}^{n} \exp(d_j)} \tag{3}$$

$$d_i = f_{score}(\mathbf{h}_i, \mathbf{t}_s) \tag{4}$$

$$\mathbf{t}_s = \frac{1}{m} \sum_{i=1}^{m} \mathbf{e}_{a_i} \tag{5}$$

where f_{score} is a function that computes a score for word w_i according to the semantic association between $\mathbf{h}_i$ and $\mathbf{t}_s$, and $\mathbf{t}_s$ is the vector representation of the given target.

3.3 Target Representation

Most previous work (Tang et al., 2016b; Liu and Zhang, 2017; Chen et al., 2017) represent a target by averaging its component word or hidden vectors as shown in Equation 5. Simple averaging may not

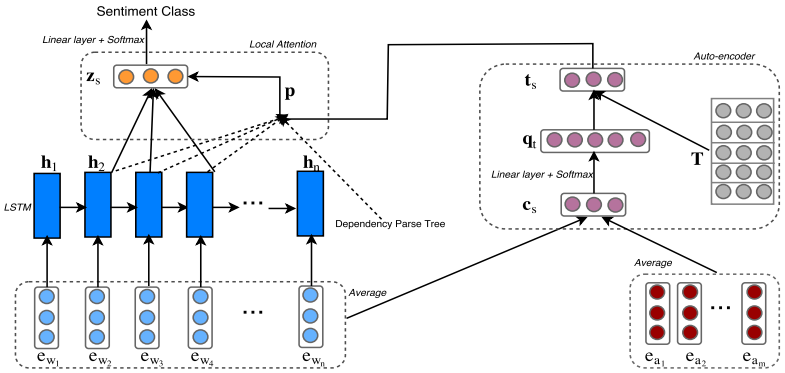


Figure 1: The overall architecture of the integrated model.

capture the real semantics of the target well. Inspired by He et al. (2017), we represent the target as a weighted summation of aspect embeddings, as illustrated in Fig. 1. An aspect embedding matrix is represented by $\mathbf{T} \in \mathbb{R}^{K \times d}$, where K, the number of aspects defined by the user, is much smaller than V. The process is formalized as follows:

$$\mathbf{t}_s = \mathbf{T}^\top \cdot \mathbf{q}_t \tag{6}$$

$$\mathbf{q}_t = \mathit{softmax}(\mathbf{W}_t \cdot \mathbf{c}_s + \mathbf{b}_t) \tag{7}$$

$$\mathbf{c}_s = \mathit{Average}(\frac{1}{m}\sum_{i=1}^{m}\mathbf{e}_{a_i}, \frac{1}{n}\sum_{j=1}^{n}\mathbf{e}_{w_j}) \tag{8}$$

where *Average* returns the mean of the input vectors. $\mathbf{c}_s$ captures both target information and context information. $\mathbf{q}_t$ is the weight vector over K aspect embeddings, where each weight represents the probability that the target belongs to the related aspect. $\mathbf{W}_t$ and $\mathbf{b}_t$ are a weight matrix and a bias vector respectively.

We would like the learned aspect embeddings to be meaningful and semantically coherent. This would allow us to interpret an aspect by looking at its nearby words in vector space. However, the aspect embedding matrix $\mathbf{T}$ is randomly initialized. It is difficult to obtain coherent aspect embeddings if we only rely on the training of the sentiment classifier. Therefore, we add an unsupervised objective function to ensure the quality of the aspect embeddings, which is jointly trained with the attention-based LSTM. Indeed, we can understand the process shown by Eq. (6) (7) (8) as an autoencoder, where we first reduce $\mathbf{c}_s$ from d dimensions to K dimensions with softmax non-linearity. Only the dimensions that are relevant to the aspects are retained in $\mathbf{q}_t$, whereas the other dimensions are removed. Then we reconstruct $\mathbf{c}_s$ from $\mathbf{q}_t$ through linear combination of aspect embeddings. The unsupervised objective is thus to minimize the reconstruction error as shown below:

$$U(\theta) = -\sum_{(s,a)\in D} \log(\min(\epsilon, \mathit{CosSim}(\mathbf{t}_s, \mathbf{c}_s))) \tag{9}$$

where cosine similarity *CosSim()* is used as the similarity measure. ϵ denotes a very small positive number. We set it to 10^{-7} in all experiments. D denotes all training samples, (s, a) denotes a sentence-target pair, and $\theta = \{\mathbf{E}, \mathbf{T}, \mathbf{W}_t, \mathbf{b}_t\}$ is the set of trainable parameters.

The learning process can also be viewed as multi-task learning where the unsupervised objective shown as Eq. 9 is an auxiliary task. Multi-task learning can help to reduce the amount of data required for learning and to improve the model generalization ability.

3.4 Syntax-based Attention Mechanism

The attention mechanism used in previous works (Tang et al., 2016b; Wang et al., 2016; Liu and Zhang, 2017) gives equal importance to all context words, where the attention weight is merely a measure of

semantic association between the target and the context word. But intuitively not all words are equally important for determining the polarity of a target. Words that appear near the target or have a modifier relation to the target, for example, are more important and should receive higher weight. This is particularly true for opinion words that express sentiment and when there are multiple targets and multiple opinion words in one sentence. To address this issue, we propose an attention mechanism that also encodes the syntactic structure of a sentence, where syntactic information is obtained from a dependency parser. Fig. 2 shows the dependency tree of an example sentence. The opinion words that are closer to the target in the dependency tree are more relevant for determining its sentiment. In our model, we define the location l of a context word as its distance to the target[1] along the dependency path. The attention model selectively attends to a small window of context words based on their location. We use ws to denote the attention window size. In our experiment, we ignore context words whose location is larger than ws and for context words within the window, different weights are applied so that words closer to the target receive more attention. The details of the proposed syntax-based attention model are described as follows:

$$p_i = \frac{d_i}{\sum_j d_j} \tag{10}$$

$$d_i = \begin{cases} \frac{1}{2^{(l_i-1)}} \cdot exp(f_{score}(\mathbf{h}_i, \mathbf{t}_s))), & \text{if } l_i \in [1, ws] \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

where t_s is the target representation constructed using the method described in §3.3. We adopt a simple score function as follows:

$$f_{score}(\mathbf{h}_i, \mathbf{t}_s) = tanh(\mathbf{h}_i^T \cdot \mathbf{W}_a \cdot \mathbf{t}_s) \tag{12}$$

where $\mathbf{W}_a \in \mathbb{R}^{d \times d}$ is a trainable weight matrix.

3.5 Overall Architecture and Training Objective

After incorporating the two proposed approaches into the attention-based LSTM, our final model is illustrated in Fig. 1. The attention-based LSTM component is associated with the categorical cross entropy loss of sentiment classification. The loss function is given below:

$$J(\theta) = -\sum_{(s,a) \in D} \sum_{c \in C} P^g_{(s,a)}(c) \log(P_{(s,a)}(c)) \tag{13}$$

where C is the collection of sentiment classes, $P^g_{(s,a)}(c)$ is either 1 or 0, indicating whether the gold label is c for (s, a), and $P_{(s,a)}(c)$ is the predicted probability that (s, a) has sentiment class c. $\theta = \{\mathbf{E}, \mathbf{T}, \mathbf{W}_t, \mathbf{b}_t, \mathbf{W}_a, \theta_{lstm}\}$ is the set of trainable parameters.

The aspect embeddings in $\mathbf{T}$ may become similar to each other during training. To ensure diversity, we employ a regularization term to enforce the uniqueness of each aspect embedding:

$$R(\theta) = \|(\mathbf{T}_{norm} \cdot \mathbf{T}_{norm}^\top - \mathbf{I})^2\| \tag{14}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{T}_{norm}$ is the L_2 normalization of $\mathbf{T}$, and $\|\|$ denotes the sum of all entries in the matrix. R reaches the minimum when the dot product between any two different aspect embeddings is zero. Thus, the regularization term aims to enforce orthogonality among the rows of $\mathbf{T}$, which punishes redundancy between aspect embeddings.

The final objective function of our model is defined as:

$$L(\theta) = J(\theta) + \lambda_u U(\theta) + \lambda_r R(\theta) \tag{15}$$

where λ_u and λ_r are hyperparameters that control the weights of the unsupervised objective described in §3.3 and the regularization term respectively.

[1]For a target containing multiple words, the distance between a context word and each word in the target is computed, and the minimal value is used to define the location of the context word.

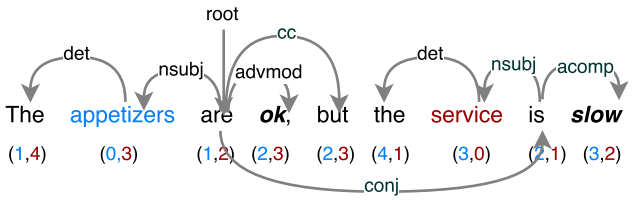


Figure 2: A dependency tree example. The numbers indicate the distances from the word to the two targets respectively along the syntactic path.

	Dataset	Pos	Neg	Neu
D1	Restaurant14-Train	2164	807	637
	Restaurant14-Test	728	196	196
D2	Laptop14-Train	994	870	464
	Laptop14-Test	341	128	169
D3	Restaurant15-Train	1178	382	50
	Restaurant15-Test	439	328	35
D4	Restaurant16-Train	1620	709	88
	Restaurant16-Test	597	190	38

Table 1: Dataset description.

4 Experiments

4.1 Datasets

We evaluate our proposed model on four benchmark datasets, taken from SemEval 2014 task 4 (Pontiki et al., 2014), SemEval 2015 task 12 (Pontiki et al., 2015), and SemEval 2016 task 5[2] (Pontiki et al., 2016). Each training and test sample in the 2014 datasets consists of the review sentence, the opinion target, and the sentiment polarity towards the opinion target. Following previous works (Tang et al., 2016b; Wang et al., 2016), we remove samples with *conflicting* polarity in the 2014 datasets – the number of samples in that class is very small and incorporating it will make the training dataset extremely unbalanced. The data format in the 2015 and 2016 datasets is a bit different, where each opinion target is also associated with one or multiple aspects and thus can have multiple sentiment polarities. Below is an example:

The food was delicious but expensive.
(target="food", aspect=food#quality, polarity=Pos)
(target="food", aspect=food#prices, polarity=Neg)

Since our model only takes a sentence and an opinion target as input, without using the aspect information, we remove a sample in both training and test sets if the opinion target has different polarities as the example above. This removes about 5% and 4% of test samples from the 2015 and 2016 datasets respectively. Statistics of the resulting datasets are presented in Table 1.

We initialize word embeddings using the 300-dimension GloVe vectors supplied by Pennington et al. (2014) and we use the dependency parser from spaCy[3] to obtain dependency paths of review sentences. We randomly select 20% of the original training data as the development set and only use the remaining 80% for training. Values for the hyperparameters are obtained empirically on the development set of one task and are fixed for all other experiments. The dimension of the LSTM hidden vectors is set to 300, the objective weights λ_u and λ_r are set to 1 and 0.1 respectively, the attention window size ws is set to 5 and the number of aspects K is set to 8.

We use RMSProp with base learning rate set to 0.001 and decay rate set to 0.9 for network training. The minibatch size is set to 32. As a regularizer, we apply dropout (Srivastava et al., 2014) with probability 0.5 to the LSTM layer and the output layer. We train the network for a fix number of epochs and select the best model according to the performance on the development set, and evaluate it on the test set.

4.2 Model Comparisons

We compare our model with the following baselines:

(1) **Feature-based SVM** (Kiritchenko et al., 2014): We compare with the reported results of a top system in SemEval 2014. We could not directly compare with the reported results from SemEval 2015 and 2016 as their model inputs are different from ours (aspect is also one of their inputs).

(2) **LSTM**: An LSTM network is built on top of word embeddings. The mean over hidden vectors is used as the sentence representation.

[2]Although there is another English dataset in the laptop domain from SemEval 2015 and 2016, it does not contain opinion targets, thus it cannot be used directly in our work.

[3]https://spacy.io

Methods	D1		D2		D3		D4	
	Acc.	Macro-F1	Acc.	Macro-F1	Acc.	Macro-F1	Acc.	Macro-F1
Feature-based SVM	80.16	NA	70.49	NA	NA	NA	NA	NA
LSTM	75.23	64.21	66.79	64.02	75.28	54.10	81.94	58.11
LSTM+ATT	76.83	66.48	68.07	65.27	77.38	60.52	82.73	59.12
TDLSTM	75.37	64.51	68.25	65.96	76.39	58.70	82.16	54.21
TDLSTM+ATT	75.66	65.23	67.82	64.37	77.10	59.46	83.11	57.53
ATAE-LSTM	78.60	67.02	68.88	65.93	78.48	62.84	83.77	61.71
MM	76.87	66.40	68.91	63.95	77.89	59.52	83.04	57.91
RAM	78.48	68.54	72.08	68.43	79.98	60.57	83.88	62.14
Ours: LSTM+ATT+TarRep	78.95	68.67	70.69	66.59	80.05	**68.73**	84.24	**68.62**
Ours: LSTM+SynATT	80.45	71.26	**72.57**	69.13	80.28	65.46	83.39	66.83
Ours: LSTM+SynATT+TarRep	**80.63***	**71.32***	71.94	**69.23**	**81.67***	66.05*	**84.61***	67.45*

Table 2: Average accuracies and Macro-F1 scores over 5 runs with random initializations. The best results are in bold. * indicates that our full model (LSTM+SynATT+TarRep) is significantly better than LSTM, LSTM+ATT, TDLSTM, TDLSTM+ATT, ATAE-LSTM, MM and RAM with $p < 0.05$ based on one-tailed unpaired t-test.

(3) **LSTM+ATT**: The model described in section 3.2.

(4) **TDLSTM** (Tang et al., 2016a): It uses a forward LSTM and a backward LSTM to encode the information before and after the target.

(5) **TDLSTM+ATT**: It extends TDLSTM by incorporating an attention mechanism.

(6) **ATAE-LSTM** (Wang et al., 2016): It is a variant of the attention-based LSTM model.

(7) **MM** (Tang et al., 2016b): It uses multi-hops of attention layers for sentence representation.

(8) **RAM** (Chen et al., 2017): It uses multi-hops of attention layers and combines the multiple attention outputs with a recurrent neural network for sentence representation.

We produce the results of TDLSTM, ATAE-LSTM, and MM with the source codes released by their authors. We re-implement RAM following the instructions in its paper as the code is not available. The comparison results are shown in Table 2. Both accuracy and macro-F1 are used for evaluation as the label distributions are unbalanced. The reported numbers are obtained as the average value over 5 different runs with random initializations for each method. Significant test results are included for testing the robustness of methods under random parameter initializations. We also show the effect of each proposed approach: **LSTM+ATT+TarRep** denotes the model where the proposed target representation is used while the attention model remains the same as LSTM+ATT; **LSTM+SynATT** denotes the model where only the conventional attention is replaced by our syntax-based attention; and **LSTM+SynATT+TarRep** denotes the full model with both approaches integrated as shown in Fig. 1.

We make the following observations: 1) Feature-based SVM is still a strong baseline, our best model achieves competitive results on D1 and D2 without relying on so many manually-designed features and external resources. 2) Compared with all other neural baselines, our full model achieves statistically significant improvements ($p < 0.05$) on both accuracies and macro-F1 scores for D1, D3, D4. 3) Compared with LSTM+ATT, all three settings of our model are able to achieve statistically significant improvements ($p < 0.05$) on all datasets. This demonstrates that both proposed approaches are effective. 4) The integrated full model overall achieves the best performance compared to using only one of the two proposed approaches. This indicates that the two proposed approaches are complementary, thus further improvements could be obtained when combining them. 5) The proposed target representation is more helpful on restaurant domain (D1, D3, and D4) than laptop domain (D2). A plausible reason is that restaurant domain has clearer aspects for opinion targets, while it is much harder to determine the aspects for many opinion targets in the laptop domain. Since our model represents the target as weighted summation of aspect embeddings, domains with clear aspects may benefit more from the model.

4.3 Model Analysis

We conduct more detailed analysis of the proposed approaches quantitatively and qualitatively. By examining the final test outputs of the relevant models, we try to investigate what kind of errors made by the baseline can be more effectively treated by our proposed approaches.

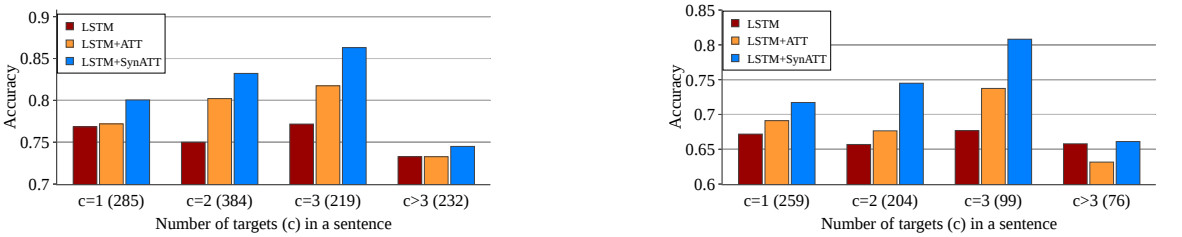
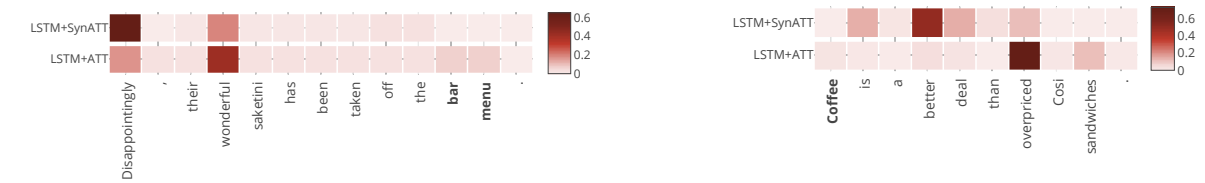
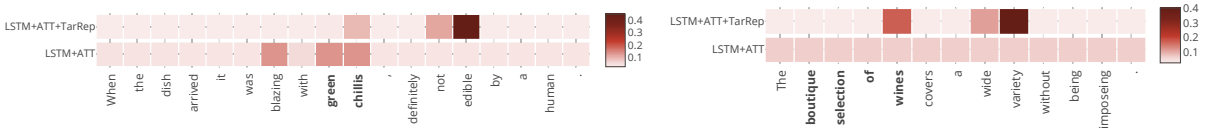


Figure 3: Classification accuracies on groups with different number of opinion targets. The number in brackets indicates the number of test instances in that group. Restaurant14 (left), Laptop14 (right).



(a) LSTM+ATT vs. LSTM+SynATT



(b) LSTM+ATT vs. LSTM+ATT+TarRep

Figure 4: Attention visualization on example sentence-target pairs. The opinion target is in bold.

Impact of Syntax-based Attention Syntax-based attention is supposed to better differentiate opinion contexts for different targets when there are multiple targets appearing in the sentence. To verify this, we compare LSTM+SynATT with LSTM and LSTM+ATT on sentences grouped by their number of targets. Fig. 3 shows the accuracies on the test sets of Restaurant14 (D1) and Laptop14 (D2).

LSTM+SynATT performs the best on all groups. In particular, it performs substantially better on groups with two or three targets. By analyzing a number of examples from these groups, we find that the proposed syntax-based attention is more effective in capturing the relevant opinion context for a given target when there are multiple targets in the sentence. Two examples are given in Fig. 4a, where our syntax-based attention successfully captures the correct opinion word towards the target of interest, whereas since conventional attention only relies on semantic association between words and the target, it fails by mis-attending to the opinion word towards other target which has similar aspect semantics.

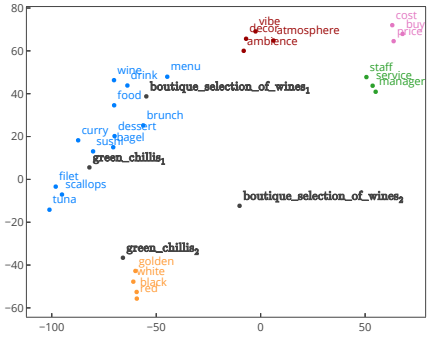


Figure 5: t-SNE visualization of embedding space. Subscripts $_1$ and $_2$ are used to denotes the target representation learned by our method and the method of averaging word vectors used in previous works respectively.

In addition, we observe that all models perform poorly on the group with more than three targets. By analyzing the errors, we find two main causes. First, those sentences are relatively long, involving more complex opinion expressions and sentence structures. Second, the proportion of neutral samples

service	table	atmosphere	price	champagne	curry	bagels	scallops
servers	tables	ambience	prices	wine	thai	bagel	fillet
staff	reservation	ambiance	buy	drink	dumplings	dessert	mignon
courteous	reservations	decor	buying	bottle	sushi	pastries	salmon
waitstaff	waiting	surroundings	cost	wines	samosa	pies	tuna

Table 3: Top 5 representative words of the eight discovered aspects on Restaurant14

contained in this group is much higher than in other groups. Since the number of neutral samples in the training set is small, the trained classifier has difficulties in predicting neutral samples in the test set.

Impact of Target Representation To investigate how the proposed target representation helps to improve performance, we extract test examples from Restaurant14 which are mis-classified by LSTM+ATT but are correctly classified by LSTM+ATT+TarRep. Among these examples, 56% are associated with opinion targets consisting of more than one words. Two examples are shown in Fig. 4b where the targets are "green chillis" and "boutique selection of wines" respectively. Fig. 5 uses t-SNE visualization to show the comparison of the learned target representations between our method and the method of averaging word vectors used in previous works on these two examples. In Fig. 5, we can observe that simply averaging the component word vectors fails to capture the correct semantics of both targets, as the target representations are far away from the food-related words in the embedding space. Due to the inaccurate representation of target, as shown in Fig. 4b, LSTM+ATT fails to attend to the right opinion context in both examples. Our proposed target representation is able to capture the correct aspect semantics for both targets and as a result, the attention mechanism can capture the correct opinion context.

Furthermore, the proposed target representation also outputs aspect embeddings after the training process. Each aspect can be interpreted by its nearby words in vector space. Table 3 presents top representative words of the eight discovered aspects on Restaurant14. The words are ranked based on their cosine similarities with the aspect embedding. As shown, each aspect is semantically coherent and our model is able to discover the typical aspects of a restaurant such as food, ambience, service, and price. Since q_t in Equation (8) represents the probability distribution over aspects for the input target, our model could additionally be used to map the input target to an aspect. We did not conduct further experiments on this since it is not our main focus in this work, but it could be an interesting direction to explore in future.

4.4 Remaining Error Analysis

We additionally conduct a careful analysis of a subset of errors made by our full model, in order to better understand its limitations. To do that, we randomly sample 100 examples with classification errors on the test set of Restaurant14, and classify them into several error categories. Table 4 shows the top three error categories, the corresponding proportions, and some representative examples for each category. The top category is *Neutral*, which denotes examples where the gold sentiment label is neutral. There are two main groups of errors under this category: (1) The polarity of the target is affected by other sentiment words in the sentence. As shown in example 1), the sentence holds a positive sentiment on atmosphere, but expresses no specific opinion on *drinks*. However, affected by the word *perfect*, the predicted sentiment towards *drinks* is positive. Although the proposed attention mechanism aims to address this type of errors, it still fails on complex examples; (2) The sentence is objective, with no opinion expression such as example 2). Since there are many more positive training examples, the predictions on such neutral examples are often biased towards positive sentiment.

The second most common error category is *Complex*, which includes examples with implicit opinion expressions (example 3) or those that require deep comprehension to be understood (example 4). This type of errors is difficult to handle with current techniques, especially when trying to build an end-to-end neural network. The diversity and low frequency of those expressions make it hard for a statistical approach to capture their patterns. For errors made on examples with negation words, we believe this is due to the insufficient training data such that LSTM cannot effectively capture certain sequential patterns.

No.	Category	(%)	Examples
1	Neutral	43	1) *A beautiful atmosphere, perfect for [drinks]$_{neu}$* 2) *We started with the [scallops]$_{neu}$ and [asparagus]$_{neu}$ and also had the [soft shell crab]$_{neu}$.*
2	Complex	28	3) *The [banana pudding]$_{neg}$ they serve has never seen as oven ...* 4) I can understand the [prices]$_{neg}$ if it served better food.
3	Negation words (sentiment shifter)	9	5) *I thought the [food]$_{neg}$ is not cheap at all compared to Chinatown.* 6) *The [dinner]$_{pos}$ here is never disappointing.*

Table 4: Top three error categories.

5 Conclusion

We propose two novel approaches to improve the effectiveness of attention mechanism for aspect-level sentiment classification. In our experiments, we show quantitatively and qualitatively that both methods help to improve the performance of a conventional attention-based LSTM. The integrated model achieves the best results over baseline methods.

As future work, we can consider improving the accuracy on neutral examples. Possible methods include data augmentation on neutral examples and integration of linguistic knowledge to better determine target-relevant opinion expressions.

References

Erik Boiy and Marie-Francine Moens. 2009. A machine learning approach to sentiment analysis in multilingual web texts. *Information Retrieval*, 12:526–558.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*.

Jiajun Cheng, Shenglin Zhao, Jiani Zhang, Irwin King, Xin Zhang, and Hui Wang. 2017. Aspect-level sentiment classification with heat (hierarchical attention) network. In *International Conference on Information and Knowledge Management (CIKM 2017)*.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent Twitter sentiment classification. In *Annual Meeting of the Association for Computational Linguistics (ACL 2014)*.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2017. An unsupervised neural attention model for aspect extraction. In *Annual Meeting of the Association for Computational Linguistics (ACL 2017)*.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Exploiting document knowledge for aspect-level sentiment classification. In *Annual Meeting of the Association for Computational Linguistics (ACL 2018)*.

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

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent Twitter sentiment classification. In *Annual Meeting of the Association for Computational Linguistics (ACL 2011)*.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif M. Mohammad. 2014. NRC-Canada-2014: Detecting aspects and sentiment in customer reviews. In *International Workshop on Semantic Evaluation (SemEval 2014)*.

Jiangming Liu and Yue Zhang. 2017. Attention modeling for targeted sentiment. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*.

Yukun Ma, Haiyun Peng, and Erik Cambria. 2018. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm. In *AAAI Conference on Artificial Intelligence (AAAI 2018)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Neural Information Processing Systems (NIPS, 2013)*.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. PhraseRNN: Phrase recursive neural network for aspect-based sentiment analysis. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*.

Maria Pontiki, Dimitrios Galanis, John Pavlopoulos, Haris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. SemEval-2014 task 4: Aspect based sentiment analysis. In *International Workshop on Semantic Evaluation (SemEval 2014)*.

Maria Pontiki, Dimitrios Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. SemEval-2015 task 12: Aspect based sentiment analysis. In *International Workshop on Semantic Evaluation (SemEval 2015)*.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammed AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, Veronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeniy Kotelnikov, Núria Bel, Salud Maria Jiménez-Zafra, and Gülşen Eryiğit. 2016. SemEval-2016 task 5: Aspect based sentiment analysis. In *International Workshop on Semantic Evaluation (SemEval 2016)*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective LSTMs for target-dependent sentiment classification. In *International Conference on Computational Linguistics (COLING 2016)*.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent Twitter sentiment classification with rich automatic features. In *International Joint Conference on Artificial Intelligence (IJCAI 2015)*.

Joachim Wagner, Piyush Arora, Santiago Cortes, Utsab Barman, Dasha Bogdanova, Jennifer Foster, and Lamia Tounsi. 2014. DCU: Aspect-based polarity classification for SemEval task 4. In *International Workshop on Semantic Evaluation (SemEval 2014)*.

Yequan Wang, Minlie Huang, Li Zhao, and Xiaoyan Zhu. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated neural networks for targeted sentiment analysis. In *AAAI Conference on Artificial Intelligence (AAAI 2016)*.

Bringing replication and reproduction together with generalisability in NLP: Three reproduction studies for Target Dependent Sentiment Analysis

Andrew Moore and Paul Rayson
School of Computing and Communications, Lancaster University, Lancaster, UK
initial.surname@lancaster.ac.uk

Abstract

Lack of repeatability and generalisability are two significant threats to continuing scientific development in Natural Language Processing. Language models and learning methods are so complex that scientific conference papers no longer contain enough space for the technical depth required for replication or reproduction. Taking Target Dependent Sentiment Analysis as a case study, we show how recent work in the field has not consistently released code, or described settings for learning methods in enough detail, and lacks comparability and generalisability in train, test or validation data. To investigate generalisability and to enable state of the art comparative evaluations, we carry out the first reproduction studies of three groups of complementary methods and perform the first large-scale mass evaluation on six different English datasets. Reflecting on our experiences, we recommend that future replication or reproduction experiments should always consider a variety of datasets alongside documenting and releasing their methods and published code in order to minimise the barriers to both repeatability and generalisability. We have released our code with a model zoo on GitHub with Jupyter Notebooks to aid understanding and full documentation, and we recommend that others do the same with their papers at submission time through an anonymised GitHub account.

1 Introduction

Repeatable (replicable and/or reproducible[1]) experimentation is a core tenet of the scientific endeavour. In Natural Language Processing (NLP) research as in other areas, this requires three crucial components: (a) published methods described in sufficient detail (b) a working code base and (c) open dataset(s) to permit training, testing and validation to be reproduced and generalised. In the cognate sub-discipline of corpus linguistics, releasing textual datasets has been a defining feature of the community for many years, enabling multiple comparative experiments to be conducted on a stable basis since the core underlying corpora are community resources. In NLP, with methods becoming increasingly complex with the use of machine learning and deep learning approaches, it is often difficult to describe all settings and configurations in enough detail without releasing code. The work described in this paper emerged from recent efforts at our research centre to reimplement other's work across a number of topics (e.g. text reuse, identity resolution and sentiment analysis) where previously published methods were not easily repeatable because of missing or broken code or dependencies, and/or where methods were not sufficiently well described to enable reproduction. We focus on one sub-area of sentiment analysis to illustrate the extent of these problems, along with our initial recommendations and contributions to address the issues.

The area of Target Dependent Sentiment Analysis (TDSA) and NLP in general has been growing rapidly in the last few years due to new neural network methods that require no feature engineering. However it is difficult to keep track of the state of the art as new models are tested on different datasets, thus preventing true comparative evaluations. This is best shown by table 1 where many approaches

[1]We follow the definitions in Antske Fokkens' guest blog post "replication (obtaining the same results using the same experiment) as well as reproduction (reach the same conclusion through different means)" from http://coling2018.org/slowly-growing-offspring-zigglebottom-anno-2017-guest-post/

Proceedings of the 27th International Conference on Computational Linguistics, pages 1132–1144
Santa Fe, New Mexico, USA, August 20-26, 2018.

are evaluated on the SemEval dataset (Pontiki et al., 2014) but not all. Datasets can vary by domain (e.g. product), type (social media, review), or medium (written or spoken), and to date there has been no comparative evaluation of methods from these multiple classes. Our primary and secondary contributions therefore, are to carry out the first study that reports results across all three different dataset classes, and to release a open source code framework implementing three complementary groups of TDSA methods.

In terms of reproducibility via code release, recent TDSA papers have generally been very good with regards to publishing code alongside their papers (Mitchell et al., 2013; Zhang et al., 2015; Zhang et al., 2016; Liu and Zhang, 2017; Marrese-Taylor et al., 2017; Wang et al., 2017) but other papers have not released code (Wang et al., 2016; Tay et al., 2017). In some cases, the code was initially made available, then removed, and is now back online (Tang et al., 2016a). Unfortunately, in some cases even when code has been published, different results have been obtained relative to the original paper. This can be seen when Chen et al. (2017) used the code and embeddings in Tang et al. (2016b) they observe different results. Similarly, when others (Tay et al., 2017; Chen et al., 2017) attempt to replicate the experiments of Tang et al. (2016a) they also produce different results to the original authors. Our observations within this one sub-field motivates the need to investigate further and understand how such problems can be avoided in the future. In some cases, when code has been released, it is difficult to use which could explain why the results were not reproduced. Of course, we would not expect researchers to produce industrial strength code, or provide continuing free ongoing support for multiple years after publication, but the situation is clearly problematic for the development of the new field in general.

In this paper, we therefore reproduce three papers chosen as they employ widely differing methods: Neural Pooling (NP) (Vo and Zhang, 2015), NP with dependency parsing (Wang et al., 2017), and RNN (Tang et al., 2016a), as well as having been applied largely to different datasets. At the end of the paper, we reflect on bringing together elements of repeatability and generalisability which we find are crucial to NLP and data science based disciplines more widely to enable others to make use of the science created.

	Datasets					
Methods	1	2	3	4	5	6
Mitchell et al. (2013)			✓			
Kiritchenko et al. (2014)				✓		
Dong et al. (2014)	✓					
Vo and Zhang (2015)	✓	✓	✓			
Zhang et al. (2015)			✓			
Zhang et al. (2016)	✓	✓	✓			
Tang et al. (2016a)	✓			✓		
Tang et al. (2016b)				✓		
Wang et al. (2016)				✓		
Chen et al. (2017)	✓			✓		
Liu and Zhang (2017)	✓	✓	✓			
Wang et al. (2017)	✓				✓	
Marrese-Taylor et al. (2017)				✓		✓
1=Dong et al. (2014), 2=Wilson (2008), 3=Mitchell et al. (2013), 4=Pontiki et al. (2014), 5=Wang et al. (2017), 6=Marrese-Taylor et al. (2017)						

Table 1: Methods and Datasets

2 Related work

Reproducibility and replicability have long been key elements of the scientific method, but have been gaining renewed prominence recently across a number of disciplines with attention being given to a 're-producibility crisis'. For example, in pharmaceutical research, as little as 20-25% of papers were found to be replicable (Prinz et al., 2011). The problem has also been recognised in computer science in general (Collberg and Proebsting, 2016). Reproducibility and replicability have been researched for sometime

in Information Retrieval (IR) since the Grid@CLEF pilot track (Ferro and Harman, 2009). The aim was to create a 'grid of points' where a point defined the performance of a particular IR system using certain pre-processing techniques on a defined dataset. Louridas and Gousios (2012) looked at reproducibility in Software Engineering after trying to replicate another authors results and concluded with a list of requirements for papers to be reproducible: (a) All data related to the paper, (b) All code required to reproduce the paper and (c) Documentation for the code and data. Fokkens et al. (2013) looked at reproducibility in WordNet similarity and Named Entity Recognition finding five key aspects that cause experimental variation and therefore need to be clearly stated: (a) pre-processing, (b) experimental setup, (c) versioning, (d) system output, (e) system variation. In Twitter sentiment analysis, Sygkounas et al. (2016) stated the need for using the same library versions and datasets when replicating work.

Different methods of releasing datasets and code have been suggested. Ferro and Harman (2009) defined a framework (CIRCO) that enforces a pre-processing pipeline where data can be extracted at each stage therefore facilitating a validation step. They stated a mechanism for storing results, dataset and pre-processed data[2]. Louridas and Gousios (2012) suggested the use of a virtual machine alongside papers to bundle the data and code together, while most state the advantages of releasing source code (Fokkens et al., 2013; Potthast et al., 2016; Sygkounas et al., 2016). The act of reproducing or replicating results is not just for validating research but to also show how it can be improved. Ferro and Silvello (2016) followed up their initial research and were able to analyse which pre-processing techniques were important on a French monolingual dataset and how the different techniques affected each other given an IR system. Fokkens et al. (2013) showed how changes in the five key aspects affected results.

The closest related work to our reproducibility study is that of Marrese-Taylor and Matsuo (2017) which they replicate three different syntactic based aspect extraction methods. They found that parameter tuning was very important however using different pre-processing pipelines such as Stanford's CoreNLP did not have a consistent effect on the results. They found that the methods stated in the original papers are not detailed enough to replicate the study as evidenced by their large results differential.

Dashtipour et al. (2016) undertook a replication study in sentiment prediction, however this was at the document level and on different datasets and languages to the originals. In other areas of (aspect-based) sentiment analysis, releasing code for published systems has not been a high priority, e.g. in SemEval 2016 task 5 (Pontiki et al., 2016) only 1 out of 21 papers released their source code. In IR, specific reproducible research tracks have been created[3] and we are pleased to see the same happening at COLING 2018[4].

Turning now to the focus of our investigations, Target Dependent sentiment analysis (TDSA) research (Nasukawa and Yi, 2003) arose as an extension to the coarse grained analysis of document level sentiment analysis (Pang et al., 2002; Turney, 2002). Since its inception, papers have applied different methods such as feature based (Kiritchenko et al., 2014), Recursive Neural Networks (RecNN) (Dong et al., 2014), Recurrent Neural Networks (RNN) (Tang et al., 2016a), attention applied to RNN (Wang et al., 2016; Chen et al., 2017; Tay et al., 2017), Neural Pooling (NP) (Vo and Zhang, 2015; Wang et al., 2017), RNN combined with NP (Zhang et al., 2016), and attention based neural networks (Tang et al., 2016b). Others have tackled TDSA as a joint task with target extraction, thus treating it as a sequence labelling problem. Mitchell et al. (2013) carried out this task using Conditional Random Fields (CRF), and this work was then extended using a neural CRF (Zhang et al., 2015). Both approaches found that combining the two tasks did not improve results compared to treating the two tasks separately, apart from when considering POS and NEG when the joint task performs better. Finally, Marrese-Taylor et al. (2017) created an attention RNN for this task which was evaluated on two very different datasets containing written and spoken (video-based) reviews where the domain adaptation between the two shows some promise. Overall, within the field of sentiment analysis there are other granularities such as sentence level (Socher et al., 2013), topic (Augenstein et al., 2018), and aspect (Wang et al., 2016; Tay et al., 2017). Aspect-level sentiment analysis relates to identifying the sentiment of (potentially multiple) topics in the

[2] http://direct.dei.unipd.it/

[3] http://ecir2016.dei.unipd.it/call_for_papers.html

[4] http://coling2018.org/

same text although this can be seen as a similar task to TDSA. However the clear distinction between aspect and TDSA is that TDSA requires the target to be mentioned in the text itself while aspect-level employs a conceptual category with potentially multiple related instantiations in the text.

Tang et al. (2016a) created a Target Dependent LSTM (TDLSTM) which encompassed two LSTMs either side of the target word, then improved the model by concatenating the target vector to the input embeddings to create a Target Connected LSTM (TCLSTM). Adding attention has become very popular recently. Tang et al. (2016b) showed the speed and accuracy improvements of using multiple attention layers only over LSTM based methods, however they found that it could not model complex sentences e.g. negations. Liu and Zhang (2017) showed that adding attention to a Bi-directional LSTM (BLSTM) improves the results as it takes the importance of each word into account with respect to the target. Chen et al. (2017) also combined a BLSTM and attention, however they used multiple attention layers and combined the results using a Gated Recurrent Unit (GRU) which they called Recurrent Attention on Memory (RAM), and they found this method to allow models to better understand more complex sentiment for each comparison. Vo and Zhang (2015) used neural pooling features e.g. max, min, etc of the word embeddings of the left and right context of the target word, the target itself, and the whole Tweet. They inputted the features into a linear SVM, and showed the importance of using the left and right context for the first time. They found in their study that using a combination of Word2Vec embeddings and sentiment embeddings (Tang et al., 2014) performed best alongside using sentiment lexicons to filter the embedding space. Other studies have adopted more linguistic approaches. Wang et al. (2017) extended the work of Vo and Zhang (2015) by using the dependency linked words from the target. Dong et al. (2014) used the dependency tree to create a Recursive Neural Network (RecNN) inspired by Socher et al. (2013) but compared to Socher et al. (2013) they also utilised the dependency tags to create an Adaptive RecNN (ARecNN).

Critically, the methods reported above have not been applied to the same datasets, therefore a true comparative evaluation between the different methods is somewhat difficult. This has serious implications for generalisability of methods. We correct that limitation in our study. There are two papers taking a similar approach to our work in terms of generalisability although they do not combine them with the reproduction issues that we highlight. First, Chen et al. (2017) compared results across SemEval's laptop and restaurant reviews in English (Pontiki et al., 2014), a Twitter dataset (Dong et al., 2014) and their own Chinese news comments dataset. They did perform a comparison across different languages, domains, corpora types, and different methods; SVM with features (Kiritchenko et al., 2014), Rec-NN (Dong et al., 2014), TDLSTM (Tang et al., 2016a), Memory Neural Network (MNet) (Tang et al., 2016b) and their own attention method. However, the Chinese dataset was not released, and the methods were not compared across all datasets. By contrast, we compare all methods across all datasets, using techniques that are not just from the Recurrent Neural Network (RNN) family. A second paper, by Barnes et al. (2017) compares seven approaches to (document and sentence level) sentiment analysis on six benchmark datasets, but does not systematically explore reproduction issues as we do in our paper.

3 Datasets used in our experiments

We are evaluating our models over six different English datasets deliberately chosen to represent a range of domains, types and mediums. As highlighted above, previous papers tend to only carry out evaluations on one or two datasets which limits the generalisability of their results. In this paper, we do not consider the quality or inter-annotator agreement levels of these datasets but it has been noted that some datasets may have issues here. For example, Pavlopoulos and Androutsopoulos (2014) point out that the Hu and Liu (2004) dataset does not state their inter-annotator agreement scores nor do they have aspect terms that express neutral opinion.

We only use a subset of the English datasets available. For two reasons. First, the time it takes to write parsers and run the models. Second, we only used datasets that contain three distinct sentiments (Wilson (2008) only has two). From the datasets we have used, we have only had issue with parsing Wang et al. (2017) where the annotations for the first set of the data contains the target span but the second set does not. Thus making it impossible to use the second set of annotation and forcing us to

only use a subset of the dataset. An as example of this: "Got rid of bureaucrats 'and we put that money, into 9000 more doctors and nurses'... to turn the doctors into bureaucrats#BattleForNumber10" in that Tweet 'bureaucrats' was annotated as negative but it does not state if it was the first or second instance of 'bureaucrats' since it does not use target spans. As we can see from table 2, generally the social media datasets (Twitter and YouTube) contain more targets per sentence with the exception of Dong et al. (2014) and Mitchell et al. (2013). The only dataset that has a small difference between the number of unique sentiments per sentence is the Wang et al. (2017) election dataset.

Lastly we create training and test splits for the YouTuBean (Marrese-Taylor et al., 2017) and Mitchell (Mitchell et al., 2013) datasets as they were both evaluated originally using cross validation. These splits are reproducible using the code that we are open sourcing.

Dataset	DO	T	Size	M	ATS	Uniq	AVG Len	S1	S2	S3
SemEval 14 L	L	RE	2951	W	1.58	1295	18.57	81.09	17.62	1.29
SemEval 14 R	R	RE	4722	W	1.83	1630	17.25	75.26	22.94	1.80
Mitchel	G	S	3288	W	1.22	2507	18.02	90.48	9.43	0.09
Dong Twitter	G	S	6940	W	1.00	145	17.37	100.00	0.00	0.00
Election Twitter	P	S	11899	W	2.94	2190	21.68	44.50	46.72	8.78
YouTuBean	MP	RE/S	798	SP	2.07	522	22.53	81.45	18.17	0.38

L=Laptop, R=Restaurant, P=Politics, MP=Mobile Phones, G=General, T=Type, RE=Review, S=Social Media, ATS=Average targets per sentence, Uniq=No. unique targets, AVG len=Average sentence length per target, S1=1 distinct sentiment per sentence, S2=2 distinct sentiments per sentence, S3=3 distinct sentiments per sentence, DO=Domain, M=Medium, W=Written, SP=Spoken

Table 2: Dataset Statistics

4 Reproduction studies

In the following subsections, we present the three different methods that we are reproducing and how their results differ from the original analysis. In all of the experiments below, we lower case all text and tokenise using Twokenizer (Gimpel et al., 2011). This was done as the datasets originate from Twitter and this pre-processing method was to some extent stated in Vo and Zhang (2015) and assumed to be used across the others as they do not explicitly state how they pre-process in the papers.

4.1 Reproduction of Vo and Zhang (2015)

Vo and Zhang (2015) created the first NP method for TDSA. It takes the word vectors of the left, right, target word, and full tweet/sentence/text contexts and performs max, min, average, standard deviation, and product pooling over these contexts to create a feature vector as input to the Support Vector Machine (SVM) classifier. This feature vector is in affect an automatic feature extractor. They created four different methods: 1. **Target-ind** uses only the full tweet context, 2. **Target-dep-** uses left, right, and target contexts, 3. **Target-dep** Uses both features of **Target-ind** and **Target-dep-**, and 4. **Target-dep+** Uses the features of **Target-dep** and adds two additional contexts left and right sentiment (LS & RS) contexts where only the words within a specified lexicon are kept and the rest of the words are zero vectors. All of their experiments are performed on Dong et al. (2014) Twitter data set. For each of the experiments below we used the following configurations unless otherwise stated: we performed 5 fold stratified cross validation, features are scaled using Max Min scaling before inputting into the SVM, and used the respective C-Values for the SVM stated in the paper for each of the models.

One major difficulty with the description of the method in the paper and re-implementation is handling the same target multiple appearances issue as originally raised by Wang et al. (2017). As the method requires context with regards to the target word, if there is more than one appearance of the target word then the method does not specify which to use. We therefore took the approach of Wang et al. (2017) and found all of the features for each appearance and performed median pooling over features. This change could explain the subtle differences between the results we report and those of the original paper.

4.1.1 Sentiment Lexicons

Vo and Zhang (2015) used three different sentiment lexicons: MPQA[5] (Wilson et al., 2005), NRC[6] (Mohammad and Turney, 2010), and HL[7] (Hu and Liu, 2004). We found a small difference in word counts between their reported statistics for the MPQA lexicons and those we performed ourselves, as can be seen in the bold numbers in table 3. Originally, we assumed that a word can only occur in one sentiment class within the same lexicon, and this resulted in differing counts for all lexicons. This distinction is not clearly documented in the paper or code. However, our assumption turned out to be incorrect, giving a further illustration of why detailed descriptions and documentation of all decisions is important. We ran the same experiment as Vo and Zhang (2015) to show the effectiveness of sentiment lexicons the results can be seen in table 4. We can clearly see there are some difference not just with the accuracy scores but the rank of the sentiment lexicons. We found just using HL was best and MPQA does help performance compared to the **Target-dep** baseline which differs to Vo and Zhang (2015) findings. Since we found that using just HL performed best, the rest of the results will apply the **Target-dep+** method using HL and using HL & MPQA to show the affect of using the lexicon that both we and Vo and Zhang (2015) found best.

	Word Counts					
	Original		Reproduction			
Lexicons	Positive	Negative	Positive	Positive Lowered	Negative	Negative Lowered
MPQA	2289	4114	**2298**	**2298**	**4148**	**4148**
HL	2003	4780	2003	2003	4780	4780
NRC	2231	3243	2231	2231	3243	3243
MPQA & HL	2706	5069	**2725**	**2725**	**5080**	**5076**
All three	3940	6490	**4016**	**4016**	**6530**	**6526**

Table 3: Sentiment lexicon statistics comparison

	Results (Accuracy %)	
Sentiment Lexicon	Original	Reproduction
Target-dep	65.72	66.81
Target-dep+: NRC	66.05	67.13
Target-dep+: HL	67.24	**68.61**
Target-dep+: MPQA	65.56	66.81
Target-dep+: MPQA & HL	**67.40**	68.37
Target-dep+: All three	67.30	68.23

Table 4: Effectiveness of Sentiment Lexicons

4.1.2 Using different word vectors

The original authors tested their methods using three different word vectors: 1. Word2Vec trained by Vo and Zhang (2015) on 5 million Tweets containing emoticons (W2V), 2. Sentiment Specific Word

[5] http://mpqa.cs.pitt.edu/lexicons/subj_lexicon/
[6] http://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm
[7] https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon

Embedding (SSWE) from Tang et al. (2014), and 3. W2V and SSWE combined. Neither of these word embeddings are available from the original authors as Vo and Zhang (2015) never released the embeddings and the link to Tang et al. (2014) embeddings no longer works[8]. However, the embeddings were released through Wang et al. (2017) code base[9] following requesting of the code from Vo and Zhang (2015). Figure 1 shows the results of the different word embeddings across the different methods. The main finding we see is that SSWE by themselves are not as informative as W2V vectors which is different to the findings of Vo and Zhang (2015). However we agree that combining the two vectors is beneficial and that the rank of methods is the same in our observations.

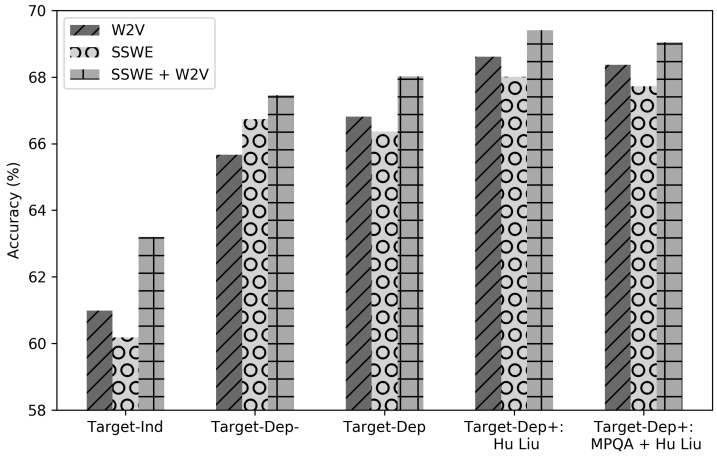


Figure 1: Effectiveness of word embedding

4.1.3 Scaling and Final Model comparison

We test all of the methods on the test data set of Dong et al. (2014) and show the difference between the original and reproduced models in figure 2. Finally, we show the effect of scaling using Max Min and not scaling the data.

As stated before, we have been using Max Min scaling on the NP features, however Vo and Zhang (2015) did not mention scaling in their paper. The library they were using, LibLinear (Fan et al., 2008), suggests in its practical guide (Hsu et al., 2003) to scale each feature to [0, 1] but this was not re-iterated by Vo and Zhang (2015). We are using scikit-learn's (Pedregosa et al., 2011) LinearSVC which is a wrapper of LibLinear, hence making it appropriate to use here. As can be seen in figure 2, not scaling can affect the results by around one-third.

4.2 Reproduction of Wang et al. (2017)

Wang et al. (2017) extended the NP work of Vo and Zhang (2015) and instead of using the full tweet/sentence/text contexts they used the full dependency graph of the target word. Thus, they created three different methods: 1. **TDParse-** uses only the full dependency graph context, 2. **TDParse** the feature of **TDParse-** and the left and right contexts, and 3. **TDParse+** the features of **TDParse** and LS and RS contexts. The experiments are performed on the Dong et al. (2014) and Wang et al. (2017) Twitter datasets where we train and test on the previously specified train and test splits. We also scale our features using Max Min scaling before inputting into the SVM. We used all three sentiment lexicons as in the original paper, and we found the C-Value by performing 5 fold stratified cross validation on the training datasets. The results of these experiments can be seen in figure 3[10]. As found with the results of Vo and Zhang (2015) replication, scaling is very important but is typically overlooked when reporting.

[8] `http://ir.hit.edu.cn/~dytang/`

[9] `https://github.com/bluemonk482/tdparse`

[10] For the Election Twitter dataset TDParse+ result were never reported in the original paper.

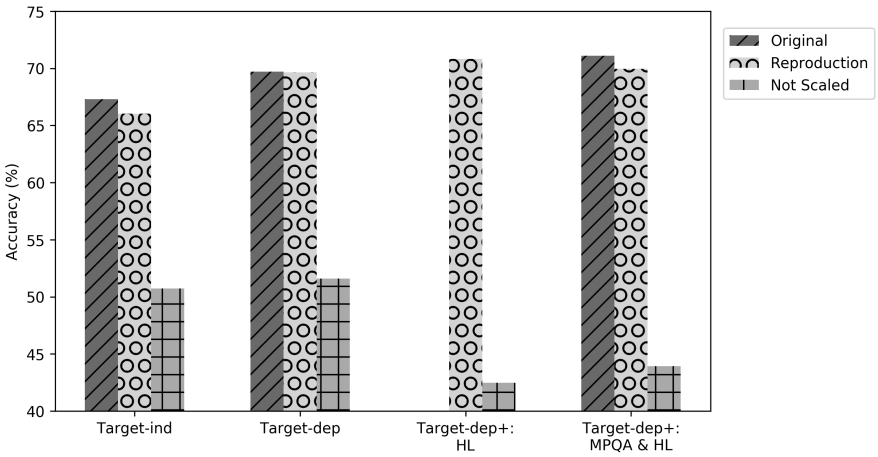


Figure 2: Target Dependent Final Results

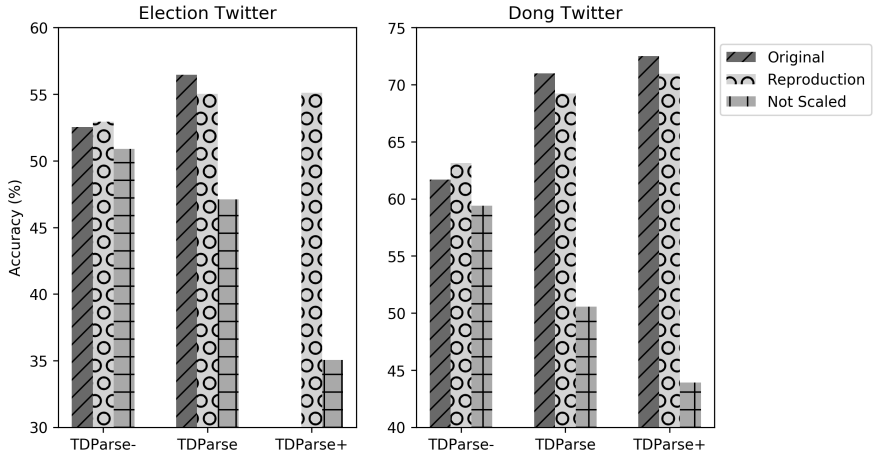


Figure 3: TDParse Final Results

4.3 Reproduction of Tang et al. (2016a)

Tang et al. (2016a) was the first to use LSTMs specifically for TDSA. They created three different models: 1. **LSTM** a standard LSTM that runs over the length of the sentence and takes no target information into account, 2. **TDLSTM** runs two LSTMs, one over the left and the other over the right context of the target word and concatenates the output of the two, and 3. **TCLSTM** same as the **TDLSTM** method but each input word vector is concatenated with vector of the target word. All of the methods outputs are fed into a softmax activation function. The experiments are performed on the Dong et al. (2014) dataset where we train and test on the specified splits. For the LSTMs we initialised the weights using uniform distribution U(0.003, 0.003), used Stochastic Gradient Descent (SGD) a learning rate of 0.01, cross entropy loss, padded and truncated sequence to the length of the maximum sequence in the training dataset as stated in the original paper, and we did not "set the clipping threshold of softmax layer as 200" (Tang et al., 2016a) as we were unsure what this meant. With regards to the number of epochs trained, we used early stopping with a patience of 10 and allowed 300 epochs. Within their experiments they used SSWE (Tang et al., 2014) and Glove Twitter vectors[11] (Pennington et al., 2014).

As the paper being reproduced does not define the number of epochs they trained for, we use early stopping. Thus for early stopping we require to split the training data into train and validation sets to know when to stop. As it has been shown by Reimers and Gurevych (2017) that the random seed statistically significantly changes the results of experiments we ran each model over each word embedding thirty times, using a different seed value but keeping the same stratified train and validation split, and

[11] https://nlp.stanford.edu/projects/glove/

	Macro F1		
Methods	O	R (Max)	R (Mean)
LSTM	64.70	64.34	60.69
TDLSTM	69.00	67.04	65.63
TCLSTM	69.50	67.66	65.23
O=Original, R=Reproduction			

Table 5: LSTM Final Results

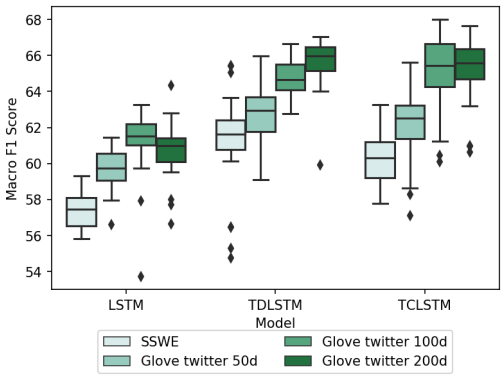


Figure 4: Distribution of the LSTM results

reported the results on the same test data as the original paper. As can be seen in Figure 4, the initial seed value makes a large difference more so for the smaller embeddings. In table 5, we show the difference between our mean and maximum result and the original result for each model using the 200 dimension Glove Twitter vectors. Even though the mean result is quite different from the original the maximum is much closer. Our results generally agree with their results on the ranking of the word vectors and the embeddings.

Overall, we were able to reproduce the results of all three papers. However for the neural network/deep learning approach of Tang et al. (2016a) we agree with Reimers and Gurevych (2017) that reporting multiple runs of the system over different seed values is required as the single performance scores can be misleading, which could explain why previous papers obtained different results to the original for the **TDLSTM** method (Chen et al., 2017; Tay et al., 2017).

5 Mass Evaluation

For all of the methods we pre-processed the text by lower casing and tokenising using Twokenizer (Gimpel et al., 2011), and we used all three sentiment lexicons where applicable. We found the best word vectors from SSWE and the common crawl 42B 300 dimension Glove vectors by five fold stratified cross validation for the NP methods and the highest accuracy on the validation set for the LSTM methods. We chose these word vectors as they have very different sizes (50 and 300), also they have been shown to perform well in different text types; SSWE for social media (Tang et al., 2016a) and Glove for reviews (Chen et al., 2017). To make the experiments quicker and computationally less expensive, we filtered out all words from the word vectors that did not appear in the train and test datasets, and this is equivalent with respect to word coverage as using all words. Finally we only reported results for the LSTM methods with one seed value and not multiple due to time constraints.

The results of the methods using the best found word vectors on the test sets can be seen in table 6. We find that the **TDParse** methods generally perform best but only clearly outperforms the other non-dependency parser methods on the YouTuBean dataset. We hypothesise that this is due to the dataset containing, on average, a deeper constituency tree depth which could be seen as on average more complex sentences. This could be due to it being from the spoken medium compared to the rest of the datasets which are written. Also that using a sentiment lexicon is almost always beneficial, but only by a small amount. Within the LSTM based methods the **TDLSTM** method generally performs the best indicating that the extra target information that the **TCLSTM** method contains is not needed, but we believe this needs further analysis.

We can conclude that the simpler NP models perform well across domain, type and medium and that even without language specific tools and lexicons they are competitive to the more complex LSTM based methods.

Dataset	Target-Dep F1	Target-Dep+ F1	TDParse F1	TDParse+ F1	LSTM F1	TDLSTM F1	TCLSTM F1
Dong Twitter	65.70	65.70	66.00	**68.10**	63.60	66.09	67.14
Election Twitter	45.50	45.90	**46.20**	44.60	38.70	43.60	42.08
Mitchell	40.80	42.90	40.50	50.00	47.17	**51.16**	41.03
SemEval 14 L	60.00	63.70	59.60	**64.50**	47.84	57.91	46.80
SemEval 14 R	56.20	57.70	59.40	**61.00**	46.36	57.68	55.38
YouTuBean	53.10	55.60	**71.70**	68.00	45.93	45.47	38.07
Mean	53.55	55.25	57.23	**59.37**	48.27	53.65	48.42

Table 6: Mass Evaluation Results

6 Discussion and conclusion

The fast developing subfield of TDSA has so far lacked a large-scale comparative mass evaluation of approaches using different models and datasets. In this paper, we address this generalisability limitation and perform the first direct comparison and reproduction of three different approaches for TDSA. While carrying out these reproductions, we have noted and described above, the many emerging issues in previous research related to incomplete descriptions of methods and settings, patchy release of code, and lack of comparative evaluations. This is natural in a developing field, but it is crucial for ongoing development within NLP in general that improved repeatability practices are adopted. The practices adopted in our case studies are to reproduce the methods in open source code, adopt only open data, provide format conversion tools to ingest the different data formats, and describe and document all settings via the code and Jupyter Notebooks (released initially in anonymous form at submission time)[12]. We therefore argue that papers should not consider repeatability (replication or reproduction) or generalisability alone, but these two key tenets of scientific practice should be brought together.

In future work, we aim to extend our reproduction framework further, and extend the comparative evaluation to languages other than English. This will necessitate changes in the framework since we expect that dependency parsers and sentiment lexicons will be unavailable for specific languages. Also we will explore through error analysis in which situations different neural network architectures perform best.

Acknowledgements

This research is funded at Lancaster University by an EPSRC Doctoral Training Grant.

References

Isabelle Augenstein, Sebastian Ruder, and Anders Søgaard. 2018. Multi-task learning of pairwise sequence classification tasks over disparate label spaces. *arXiv preprint arXiv:1802.09913*.

Jeremy Barnes, Roman Klinger, and Sabine Schulte im Walde. 2017. Assessing state-of-the-art sentiment models on state-of-the-art sentiment datasets. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 2–12. Association for Computational Linguistics.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 463–472. Association for Computational Linguistics.

Christian Collberg and Todd A. Proebsting. 2016. Repeatability in computer systems research. *Commun. ACM*, 59(3):62–69, February.

Kia Dashtipour, Soujanya Poria, Amir Hussain, Erik Cambria, Ahmad YA Hawalah, Alexander Gelbukh, and Qiang Zhou. 2016. Multilingual sentiment analysis: state of the art and independent comparison of techniques. *Cognitive Computation*, 8(4):757–771.

[12] `https://github.com/apmoore1/Bella`

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 49–54. Association for Computational Linguistics.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

Nicola Ferro and Donna Harman. 2009. Clef 2009: Grid@ clef pilot track overview. In *Workshop of the Cross-Language Evaluation Forum for European Languages*, pages 552–565. Springer.

Nicola Ferro and Gianmaria Silvello. 2016. The clef monolingual grid of points. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 16–27. Springer.

Antske Fokkens, Marieke Van Erp, Marten Postma, Ted Pedersen, Piek Vossen, and Nuno Freire. 2013. Offspring from reproduction problems: What replication failure teaches us. In *ACL (1)*, pages 1691–1701.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 42–47. Association for Computational Linguistics.

Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. 2003. A practical guide to support vector classification.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. 2014. Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 437–442. Association for Computational Linguistics.

Jiangming Liu and Yue Zhang. 2017. Attention modeling for targeted sentiment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 572–577. Association for Computational Linguistics.

Panos Louridas and Georgios Gousios. 2012. A note on rigour and replicability. *ACM SIGSOFT Software Engineering Notes*, 37(5):1–4.

Edison Marrese-Taylor and Yutaka Matsuo. 2017. Replication issues in syntax-based aspect extraction for opinion mining. In *Proceedings of the Student Research Workshop at the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 23–32. Association for Computational Linguistics.

Edison Marrese-Taylor, Jorge Balazs, and Yutaka Matsuo. 2017. Mining fine-grained opinions on closed captions of youtube videos with an attention-rnn.

Margaret Mitchell, Jacqui Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1654. Association for Computational Linguistics.

Saif Mohammad and Peter Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 26–34. Association for Computational Linguistics.

Tetsuya Nasukawa and Jeonghee Yi. 2003. Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of the 2nd international conference on Knowledge capture*, pages 70–77. ACM.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*.

John Pavlopoulos and Ion Androutsopoulos. 2014. Aspect term extraction for sentiment analysis: New datasets, new evaluation measures and an improved unsupervised method. In *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)*, pages 44–52. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35. Association for Computational Linguistics.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphee De Clercq, Veronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeniy Kotelnikov, Núria Bel, Salud María Jiménez-Zafra, and Gülşen Eryiğit. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 19–30. Association for Computational Linguistics.

Martin Potthast, Sarah Braun, Tolga Buz, Fabian Duffhauss, Florian Friedrich, Jörg Marvin Gülzow, Jakob Köhler, Winfried Lötzsch, Fabian Müller, Maike Elisa Müller, et al. 2016. Who wrote the web? revisiting influential author identification research applicable to information retrieval. In *European Conference on Information Retrieval*, pages 393–407. Springer.

Florian Prinz, Thomas Schlange, and Khusru Asadullah. 2011. Believe it or not: how much can we rely on published data on potential drug targets? *Nature Reviews Drug Discovery*, 10:712.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics.

Efstratios Sygkounas, Giuseppe Rizzo, and Raphaël Troncy. 2016. A replication study of the top performing systems in semeval twitter sentiment analysis. In *International Semantic Web Conference*, pages 204–219. Springer.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1555–1565. Association for Computational Linguistics.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3298–3307. The COLING 2016 Organizing Committee.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 214–224.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2017. Learning to attend via word-aspect associative fusion for aspect-based sentiment analysis. *arXiv preprint arXiv:1712.05403*.

Peter Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.

Duy-Tin Vo and Yue Zhang. 2015. Target-dependent twitter sentiment classification with rich automatic features. In *IJCAI*, pages 1347–1353.

Yequan Wang, Minlie Huang, xiaoyan zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615. Association for Computational Linguistics.

Bo Wang, Maria Liakata, Arkaitz Zubiaga, and Rob Procter. 2017. Tdparse: Multi-target-specific sentiment recognition on twitter. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 483–493. Association for Computational Linguistics.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

Theresa Ann Wilson. 2008. *Fine-grained subjectivity and sentiment analysis: recognizing the intensity, polarity, and attitudes of private states*. University of Pittsburgh.

Meishan Zhang, Yue Zhang, and Duy Tin Vo. 2015. Neural networks for open domain targeted sentiment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 612–621.

Meishan Zhang, Yue Zhang, and Duy-Tin Vo. 2016. Gated neural networks for targeted sentiment analysis. In *AAAI*, pages 3087–3093.

Multilevel Heuristics for Rationale-Based Entity Relation Classification in Sentences

Shiou Tian Hsu, Mandar Chaudhary and Nagiza F. Samatova
North Carolina State University
shsu3@ncsu.edu, mschaudh@ncsu.edu, samatova@csc.ncsu.edu

Abstract

Rationale-based models provide a unique way to provide justifiable results for relation classification models by identifying rationales (key words and phrases that a person can use to justify the relation in the sentence) during the process. However, existing generative networks used to extract rationales come with a trade-off between extracting diversified rationales and achieving good classification results. In this paper, we propose a multilevel heuristic approach to regulate rationale extraction to avoid extracting monotonous rationales without compromising classification performance. In our model, rationale selection is regularized by a semi-supervised process and features from different levels: word, syntax, sentence, and corpus. We evaluate our approach on the SemEval 2010 dataset that includes 19 relation classes and the quality of extracted rationales with our manually-labeled rationales. Experiments show a significant improvement in classification performance and a 20% gain in rationale interpretability compared to state-of-the-art approaches.

1 Introduction

The goal of sentence-level entity relation classification is to infer how two target entities are semantically associated in a sentence. It is a core NLP function that supports many high level tasks such as information extraction and knowledge graph population (Hendrickx et al., 2009; Niu et al., 2012). Recent advances in generative models facilitate improved classification performance along with justifiable results which improves model interpretability. A generative model will first extract the most representative fragments in the sentence that expresses the relation first, and augments the classifier with the rationales (Hsu et al., 2018) . These representative fragments are called rationales as per (Zhang et al., 2016; Lei et al., 2016; Hsu et al., 2018), and can be used to justify the results. We illustrate an example of rationale-based models using an example sentence which expresses a Instrument-Agency(e2,e1) relationship between the given target entities e_1 = "attacker" and e_2 = "instrument":

Rationale based models: *She had struggled violently with her [attacker]*e_1*, who* ***killed*** *her* ***with*** *a blunt [instrument]*e_2.
(words marked in bold and the entities are the major sources to infer the relation)

The most commonly used structure in generative models consists of two components: a Generator and a Discriminator. The Generator extracts rationales from an input sentence in an unsupervised fashion; the Discriminator, which is supervised, predicts the relation class utilizing the rationales. However, training a good Generator can be essentially hard due to the mode collapse problem (Chen et al., 2016; Che et al., 2017; Duhyeon and Hyunjung, 2018). Mode collapse is a commonly observed paradox where the Generator attempts to extract varied rationales, but converges to select monotonous rationales because the Discriminator failed to model diversified rationales. A collapsed Generator is not capable of providing

Proceedings of the 27th International Conference on Computational Linguistics, pages 1145–1155
Santa Fe, New Mexico, USA, August 20-26, 2018.

meaningful rationales since it extracts the same rationales repeatedly regardless of the context. However, as shown in Figure 1, a collapsed Generator does not always lead to poor classification performance when compared to a finely converged Generator.

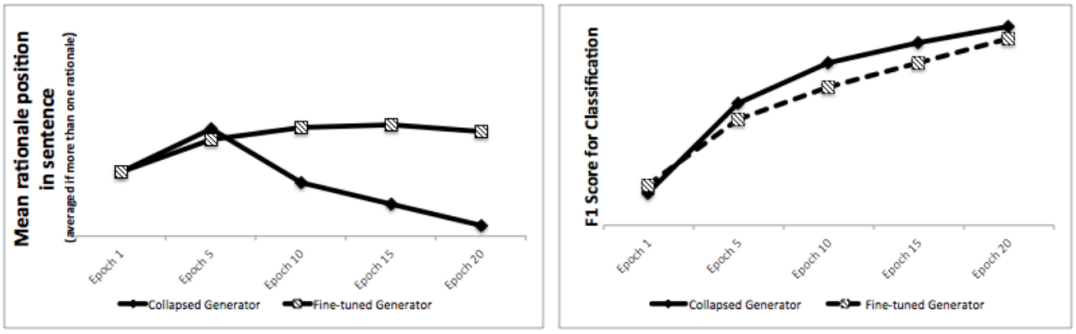


Figure 1: A Generator can converge at collapsed rationales but still lead to good classification results. Figure on the left shows the rationale distribution of a collapsed and a well-converged Generator that uses a recurrent neural net . The well-converged Generator evenly selects rationales from a sentence, while the collapsed Generator selects rationales close to the beginning of the sentence. This is because the best scenario for Discriminator is that some rationales are repetitively selected and optimized in every training iteration, which is difficult for the Generator. Eventually, the Generator learns to exploit position features since it is less diversified than words. Figure on the right shows that a collapsed Generator does not necessarily deteriorate the classification performance.

One of the major causes of mode collapse is the lack of regularizing terms to smooth out the Generator(Chen et al., 2016; Arjovsky and Bottou, 2017; Salimans et al., 2016). In this paper, we aim to improve the rationale-based models for relation classification by introducing semi-supervised capacity and addition regularizing force to the Generator to prevent mode collapsing. In our Generator, we develop a multilevel heuristic to replace the context-focused recurrent neural net used by existing approach(Hsu et al., 2018). Our Generator jointly considers features from different levels of the dataset such as contextual words, target entities, sentence syntax features, rationale-relation closeness, and global word distribution in corpus. The details of our improved Generator are shown in Figure 2.

Our Generator jointly considers several aspects such as contextual words, target entities, word distribution in corpus, sentence grammatical features, and selected rationale-relation closeness to regularize rationale selection.

We evaluate our model on the SemEval 2010 Task 8 dataset which includes 19 relation classes. Specifically, we use F1 score to measure the quality of extracted rationales and the classification performance. Given the absence of ground-truth of rationales in the dataset, we manually label the test set and make it publicly available[1]. We empirically demonstrate an improvement in interpretability of the rationales by 20% compared to Hsu et al. (2018), and also an improvement in the F1 score of 90.7 compared to 89.5 in the state-of-the-art model. We summarize the contributions of our approach as follows:

- We propose a multilevel heuristics approach to avoid mode collapse in rationale-based relation classification models.
- We empirically demonstrate our model produces improved relation classification performance and more interpretable rationales compared to other models
- We provide a labeled set of rationales that can be used for evaluation in future rationale research.

2 Related Work

Research for improving generative neural networks have gained much interest in the research community due to the effectiveness and versatility of these models. These improvements can be divided into two categories: architecture-oriented and divergence-oriented.

[1] https://github.ncsu.edu/shsu3/rationale-improvement

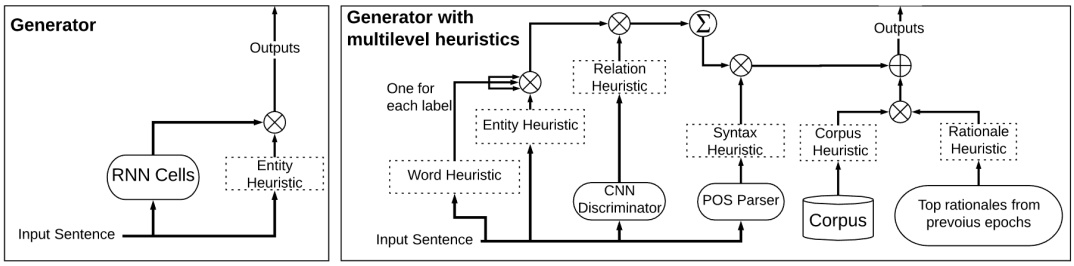


Figure 2: Figure on the left and the right shows the Generator in Hsu et al. (2018) and our proposed multilevel heuristics Generator respectively. The Word heuristic measures the relative closeness of each word in the sentence to every relation class. The Entity heuristic uses the target entities to regulate the Word heuristic. The Relation heuristic contains predictions of the relation from a standalone convolutional neural net model and is also used used to regulate the Word heuristic. The Word heuristic is then summed and weighted by the Syntax heuristic, which considers both, the word distance to entities, and Part-of-Speech labels. The Corpus heuristic is obtained by word-relation distribution in the corpus. The Rationale heuristic is the semi-supervised component which is based on rationales from the previous training iteration.

The Generative Adversarial Network (GAN) (Goodfellow et al., 2014) is the one of most representative model in the architecture-oriented group. Goodfellow et al. (2014) proposed a modified generative model using an adversarial minimax-game formulation. The objective of GAN is to jointly train a Generator and Discriminator, where the loss for Discriminator and the reward for Generator comes from failed attempts to determine ground-truths from the emulated results created by Generator. However, the adversary process in GAN is the major source of mode collapse, hence further extensions of GAN aimed towards addressing this issue. For example, InfoGan (Chen et al., 2016) proposed to maximize the mutual information between a subset of the noise variables with the observations to produce disentangled unsupervised representations. Salimans et al. (2016) proposed to use semi-supervised training and label-smoothing to help GAN converge to Nash Equilibrium between the Generator and the Discriminator. Duhyeon and Hyunjung (2018) proposed RFGAN to implicitly regularize the discriminator using representative features extracted from the data distribution. In VEEGAN (Srivastava et al., 2017), the authors suggested to jointly train an extra Reconstructor with the Generator which is an approximate reverse action of the Generator to encode noise.

Another group of methods have aimed to reduce mode collapse in generative models by modifying the joint loss functions used between the Generator and Discriminator. In Arjovsky et al. (2017), the authors theoretically showed that the KullbackLeibler (KL) divergence often used in GAN was one of the source of instability. Based on this discovery, Arjovsky and Bottou (2017) proposed Wasserstein distance to measure the similarity between the fake and real data to make stabler Generator. Gulrajani et al. (2017) introduce the gradient penalty for regularizing the divergence as an alternative to gradient clipping used in Arjovsky and Bottou (2017) to capture stronger movements in gradients . In DRAGAN (Kodali et al., 2017), they proposed to regulate sharp gradients in the Discriminator, which often happens at some undesired *local equilibria* between Generator and Discriminator. Roth et al. (2017) proposed another type of regularization that breaks down the divergence into different Discriminator output to achieve stability.

We position our work closer to the first category since we can view the multilevel heuristics used in our model close to a combination of models in the first category. The Word and Corpus heuristic are derived from data distribution similar to RFGAN, and the Relation and Rationale heuristic are substantially close to the semi-supervised learning techniques used in Salimans et al. (2016). However, our work is not exclusive to the second category as the Word heuristic included idea close to Roth et al. (2017) where the divergence in the Word heuristic maps to label level in Discriminator instead of treating the Discriminator as a whole.

On a final note, we would like to mention that this work shares some resemblance with (Giuliano et al., 2006) and can be potentially considered as its extension. In Giuliano et al. (2006), the authors utilize several shallow linguistic information for relation classification, such as word-relation frequency or POS to avoid dependent syntactic information. In this work, the fundamental difference is that all token-based computations are replaced by embeddings which are more generalizable when considering words of semantic relatedness. Additionally, the training phase is carried out by the adversarial process.

3 Rationale Generation Framework

We describe the framework for rationale-based approach in the following. Consider an input sentence $\boldsymbol{x} = \{x_t\}_{t=1}^{T}$, where e_1 and e_2 are given target entities of interest and e_1 and $e_2 \subset \{x_1, x_2, ..., x_T\}$. The goal of relation classification is thus to use ($\boldsymbol{x}$,e_1,e_2) to predict the relationship y between e_1 and e_2. We call this prediction process as encoding and denote it as $\boldsymbol{enc}(\boldsymbol{x},e_1,e_2)$.

Following Hsu et al. (2018), we consider this rationale based model as an adversarial problem the goal of the Generator is to generate rationales $\boldsymbol{r}$ that $\boldsymbol{enc}(\boldsymbol{r},\boldsymbol{x},e_1,e_2)$ can outperform all other sets of rationales in $\boldsymbol{enc}(\tilde{\boldsymbol{r}},\boldsymbol{x},e_1,e_2)$. In Hsu et al. (2018), they split the Generator into two steps: the Generator and the Selector. The Generator first generates candidates $\boldsymbol{gen}(\boldsymbol{x},e_1,e_2)$, which samples $\boldsymbol{c} = \{c_t\}_{t=1}^{T}$ from the input where $c_t \in [0, 1]$ and it represents whether x_t should be considered as a rationale ; the Selector then samples rationales $\boldsymbol{r}$ from $\boldsymbol{c}$ and is represented as $\boldsymbol{sel}(\boldsymbol{c},e_1,e_2)$. We illustrate the framework in Figure 3.

A difference against other generative works is worth noting: in Lei et al. (2016), rationales are defined as a sequence of words in a customer review that are directly related to different rating aspects. This is because in Lei et al. (2016), the assumption is that the reviews contain convoluted sentiments, and the goal of a rationale is to disentangle the sentiments. In effect, the goal of $\boldsymbol{r}$ in Lei et al. (2016) is to serve as a proxy of $\boldsymbol{x}$ that makes $\boldsymbol{enc}(\boldsymbol{r})$ approximate to $\boldsymbol{enc}(\boldsymbol{x})$. while our goal is to find $\boldsymbol{r}$ that outperform all possible short enumerations of $\boldsymbol{x}$ in $\boldsymbol{enc}(\boldsymbol{r},\boldsymbol{x},e_1,e_2)$.

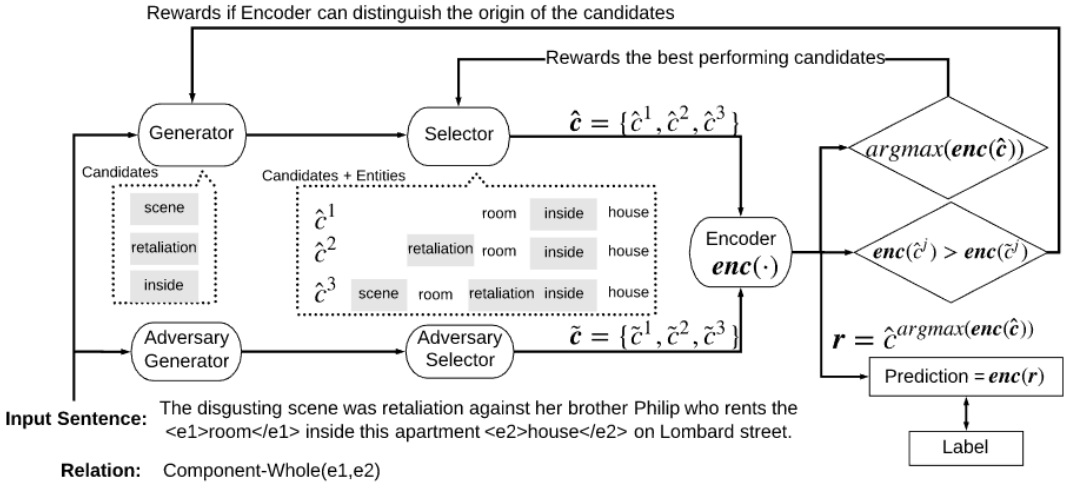


Figure 3: The model structure from Hsu et al. (2018) with a sample input sentence and the given entities. The Generator selects candidate rationales, and the Selector enumerates all possible combinations of candidates with entities and selects one 'best performing candidate set'. An adversary Generator and Selector carry out the same process using a randomized approach. Candidate sets are then transformed into a vector representation by the Encoder and are evaluated against the ground-truth. Rewards are fed back to the Generator if the Encoder is able to identify candidate sets that are generated randomly, while the Selector is rewarded if the best performing rationale candidate set outperforms the one in the adversary. The best performing candidate set from Selector is finally considered as the extracted rationales, and then used in the Encoder to predict entity relation.

4 Model

In this section, we describe each component of the framework. We begin by explaining the Encoder to formalize the goal of relation classification.

4.1 Encoder

Given an input training tuple ($\boldsymbol{x}$,e_1,e_2,y), where $\boldsymbol{x}$ = $\{x_t\}_{t=1}^{T}$, and y is the one-hot L dimensional vector representing the relation class. The goal of the Encoder $\boldsymbol{enc}(\boldsymbol{r},\boldsymbol{x},e_1,e_2)$ is to produce a probability distribution $\hat{y}$ that approximates to y as formulated as follows (N is the size of the training set):

$$loss = \sum_{i=1}^{|N|} -y_i \cdot log(\hat{y}_i) \tag{1}$$

The Encoder produces $\hat{y}$ by encoding ($\boldsymbol{r}$,$\boldsymbol{x}$,e_1,e_2) as a vector representation and then projects it to the y space. This vector representation can further break down to a sentence vector V_x and a rationale vector V_r. We produce V_x through CNN using the same model as Kim (2014) which encodes the sentence through a continuous window approach, where K are the set of sizes for the windows. To produce V_r, we implement a RNN model using GRU cells that reads the rationales and target entities. Since RNN is order sensitive, the rationales and the entities are sorted based on their position in the sentence before applying the RNN model.

$$\begin{aligned} V_x &= CNN(\boldsymbol{x}) & \hat{y} &= \boldsymbol{enc}(\boldsymbol{r}, \boldsymbol{x}, e_1, e_2) \\ V_r &= RNN([\boldsymbol{r}, e_1, e_2]) & &= softmax(W_{enc} \cdot [V_r, V_x, e_1, e_2] + b_{enc}) \end{aligned} \tag{2}$$

4.2 Generator

The goal of the Generator $\boldsymbol{gen}(\boldsymbol{x})$ is to derive a set of binary variables $\boldsymbol{c} = \{c_t\}_{t=1}^{T}$ by sampling from probability scores $\boldsymbol{s} = \{s_t\}_{s=1}^{T}$, where $c_t \in [0, 1]$ indicates whether x_t is chosen as a candidate rationale and s_t is computed by the multilevel heuristics. We represent it as $\boldsymbol{c} \sim \boldsymbol{gen}(\boldsymbol{x})$.

In contrast to Hsu et al. (2018) which uses an RNN model for Generator, we adopted a set of logistic regression models with several feed-forward networks to facilitate the Generator. The reason being, an RNN model can be difficult to train and might be susceptible to mode collapse if not carefully tuned. In the Generator, the number of logistic regression models is equal to L which is the number of classes in y. Each word in the sentence is first moderated by the Entity heuristic he, which is obtained from the target entities and fed through the logistic regression models to obtain the Word heuristic $hw = \{\{hw_t^l\}_{l=1}^{L}\}_{t=1}^{T}$. Each hw_t^l can be considered as a probability of x_t being chosen as a rationale if the given sentence is of relation class l. The second step is to compute the Relation heuristic $hr = \{hr^l\}_{l=1}^{L}$ and the Syntax heuristic hs_t that are applied to the Word heuristic. The Relation heuristic initializes the relation with a random guess using the whole sentence. The heuristic is then trained through a CNN model which similar to the CNN model we used to compute Vx. The Syntax heuristic hs_t is obtained from a feed forward neural network that takes as input (1) word distance to target entities, and (2) Part of Speech (POS) tags. The output, s_t^{local}, is obtained by combining the Word, Entity, Syntax and Relation heuristics, and it represents local sentence information score.

The final score s_t is then computed by applying the Corpus and Rationale heuristics to s_t^{local}. The Corpus heuristic $hc = \{hc^l\}_{l=1}^{L}$ contains binary values computed by sorting and selecting the frequent K words in each relation class in the training set. The Rationale heuristic $hra = \{hra^l\}_{l=1}^{L}$ also contains binary values which are based on taking the top portion of the Word heuristic for each relation class, but the Word heuristic considered here is not moderated by entities when computing the Rationale heuristic. The Corpus and Rationale heuristics capture the association between the words and relations are in the dataset. To leverage this information, we construct a global heuristic, s_t^{global}, by computing an element-wise multiplication of hc and hra. Note that s_t^{global} can be seen as a self-learning resource for semi-supervised learning because the Rationale heuristic is based on the results from previous iterations.

Accordingly, s_t^{global} will not be used in testing and will only be included after several training iterations have completed. Finally, before applying s_t^{global} to the final score s_t we apply a dynamically computed weight factor CR to $s_t^{local}$and s_t^{global}. We formulate all the heuristics in the following equation,

$$
\begin{aligned}
&he = sigmoid(W_{he} \cdot [e1, e2] + b_h e) \\
&hw_t = sigmoid(W_{hw} \cdot (x_t \odot he) + b_h e) \\
&hr = CNN(\boldsymbol{x}) \\
&hs_t = sigmoid(W_{hs} \cdot [POS_t, position_t]) \\
&CR = sigomid(W_{CR} \cdot hr' + b_{CR})
\end{aligned}
\qquad
\begin{aligned}
&s_t^{local} = hw_t \cdot \mathcal{T}(hr) \cdot hs \\
&\quad (\mathcal{T} \text{stands for transpose function}) \\
&s_t^{global} = hc \odot hra \\
&s_t = s_t \cdot (1 - CR) + s_t^{global} CR
\end{aligned}
\tag{3}
$$

In the most simplistic Generator, the probability that x_t is chosen is conditionally independent of all other $\boldsymbol{x}$. In other words, we can sample candidate $\boldsymbol{c}$ from a uniform distribution using the probability scores, $\boldsymbol{s}$. However, we observe that the target rationales in the dataset often consist of few words - for example "*room **inside** apartment*". Therefore, we added the following Equation (4) to limit the number of rationales selected, $\boldsymbol{c}$, to be at most J.

$$
\begin{aligned}
&c_t = \begin{cases} 1, sort(\{(s_t > rand(0,1)) \cdot s_t\}_{t=1}^{T})[1:J] \\ 0, otherwise \end{cases} \\
&\boldsymbol{c} = \{c_t\}_{t=1}^{T}
\end{aligned}
\tag{4}
$$

4.3 Selector

The last component of the model is the Selector $\boldsymbol{sel}(\boldsymbol{c})$, where the goal of the Selector is to decide the number of rationales best suited for the prediction. The Selector is facilitated by a simple feed-forward network that outputs a number from $1 \sim J$ by scoring rationale sets of length ranging from $1 \sim J$ based on the top ranked $s_t * c_t$. During training, all rationale sets will be forwarded to the Encoder, and the training label for Selector will be the size of the set that obtains the least training loss in Encoder. This Selector is identical to the one in Hsu et al. (2018) and is defined as follows:

$$
\begin{aligned}
&\hat{c}_t^j = \begin{cases} 1, sort(\{c_t \cdot s_t\}_{t=1}^{T})[1:j], j \in [1:J] \\ 0, otherwise \end{cases} \\
&\hat{\boldsymbol{c}}^{\boldsymbol{j}} = \{\hat{c}_t^j\}_{t=1}^{T}, \text{where } \hat{c}_t^j = 1 \\
&score(\hat{\boldsymbol{c}}^{\boldsymbol{j}}) = W_c \cdot \boldsymbol{enc}(\hat{\boldsymbol{c}}^{\boldsymbol{j}}, \boldsymbol{x}, e1, e2)
\end{aligned}
\qquad
\begin{aligned}
&scores^j = softmax(\{score(\hat{\boldsymbol{c}}^{\boldsymbol{j}})\}_{j=1}^{J})^j \\
&\hat{scores}^j = \begin{cases} 1 & , j = argmax(scores) \\ 0 & , otherwise \end{cases} \\
&\boldsymbol{r} = \{x_t\}, \text{where } \hat{c}_t^j = 1,\ j = argmax(scores)
\end{aligned}
\tag{5}
$$

4.4 Joint Objective and Adversarial Training

We formulate the joint loss function in the following to bind the components.

As described earlier, the goal of the Generator and the Selector is not to emulate $\boldsymbol{x}$ with $\boldsymbol{r}$ due to the nature of this research. In effect, the Generator and the Selector are not competing against the Encoder, but instead with an adversary Generator, for which we use a randomized approach. The adversary Generator will sample $\tilde{\boldsymbol{c}}$ randomly from $\boldsymbol{x}$. The adversary rationales $\tilde{\boldsymbol{r}}$ will then be selected using $\tilde{\boldsymbol{c}}$ as per Equation (4). $\tilde{\boldsymbol{r}}$ will be passed to the Encoder to generate a projection $\tilde{y}$ onto the target dimension. We compare $\tilde{y}$ with $\hat{y}$ in terms of their similarity to y and is denoted by $\mathcal{D} \in \{0, 1\}$. $\mathcal{D} = 1$ when $\hat{y}$ is closer to y compared to $\tilde{y}$, and is noted as a 'model success case', meanwhile $1 - \mathcal{D} = 1$ for the opposite situation and is noted as an 'adversary success case'. The objective for the Generator and Encoder reacts differently to the two different cases, where the Encoder optimize whichever rationales that performs better, and the Generator rewards the selected rationales in model success case and penalize in the other case. We further split $\hat{y}$, $\tilde{y}$ and $\mathcal{D}$ into $\hat{y}^j$, $\tilde{y}^j$, $\mathcal{D}^j$ and $j \in [1:J]$, where $\mathcal{D}^j$ represents that $\hat{y}^j$ outperforms $\tilde{y}^j$ when using j rationales. Finally, we introduce a penalizing factor P^j when $j \neq argmax(scores)$ to penalize the gradients from poor performing cases. We summarize as follows:

$$
\begin{aligned}
f(s_t^{'jl}) &= -log(s_t^{'jl}) \cdot c_t^j \cdot y_i + \\
&\quad - log(1 - s_t^{'jl}) \cdot c_t^j \cdot (1 - y_i) \\
f(c_t^j) &= -log(s_t) \cdot c_t^j - log(1 - s_t) \cdot (1 - c_t^j) \\
f_y(\hat{y}_i^j) &= -y_i \cdot log(\hat{y}_i^j)
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{L}_{enc} &= \sum_{j=1}^{J} P^j \left[\mathcal{D}^j f_y(\hat{y}^j) + (1 - \mathcal{D}^j) f_y(\tilde{y}^j) \right] \\
\mathcal{L}_{sel} &= \sum_{j=1}^{J} -\mathcal{D}^j * \hat{scores}^j * log(scores^j) \\
\mathcal{L}_{gen} &= \sum_{j=1}^{J} \sum_{t=1}^{T} \left[\mathcal{D}^j f(\hat{c}_t^j) - (1 - \mathcal{D}^j) f(\tilde{c}_t^j) \right] + \\
&\quad \sum_{j=1}^{J} \sum_{t=1}^{T} \sum_{l=1}^{L} \left[\mathcal{D}^j f(\hat{s}_t^{jl}) - (1 - \mathcal{D}^j) f(\tilde{s}_t^{jl}) \right]
\end{aligned}
\tag{6}
$$

The final joint objective and the expected cost is defined as:

$$
\mathcal{L}(\boldsymbol{r}, \boldsymbol{x}, e_1, e_2, y) = \mathcal{L}_{gen} + \mathcal{L}_{sel} + \mathcal{L}_{enc}
$$

$$
\min_{\theta e \theta g \theta s} \sum_{(\boldsymbol{x}, e_1, e_2, y) \in N} \mathbb{E}_{\boldsymbol{r} \sim sel(\boldsymbol{c}), \boldsymbol{c} \sim gen(\boldsymbol{x})} \mathcal{L}(\boldsymbol{r}, \boldsymbol{x}, e_1, e_2, y) \tag{7}
$$

where θe, θg, θs are the parameters used in the Encoder, Generator and Selector respectively.

5 Experiments

5.1 Dataset

We evaluate our approach by performing experiments on the SemEval 2010 Task 8 dataset. A collection of sentences is provided with marked target entities and ground-truth for the relation between the entities.

The dataset contains 10,717 sentences, where 8000 entries are used in training. There are 9 types of relationships plus an "Other" class (Table 1). Relationship types (except for "Other") are expressed bi-directionally, making 19 classes in total. Following SemEval 2010's protocol, we used the macro-F1 metric in the experiment, excluding all cases of the "Other" class.

Besides the original SemEval 2010 Task 8 data, we asked 3 human annotators to label rationales used in the test set. Annotators were asked to choose 0-3 rationales for each test sentence based on their intuition. Each annotator is asked to roughly annotate one-third of the test set. Distribution of rationales are as follows: no-rationales:2%, 1-rationale:57%, 2-rationales:38% and 3-rationales:2%. Note that we did not annotated the training set.

Cause-Effect	Component-Whole	Content-Container
Entity-Destination	Entity-Origin	Instrument-Agency
Member-Collection	Message-Topic	Product-Producer

Table 1: Relation classes in SemEval 2010 task 8

5.2 Experimental Setup

We now describe in detail the parameters used in our experiments. We initialize the word vectors with the GoogleNews[2] corpus. The CNN filter size was initialized to 50. We use at most 3 rationales and set the penalizing factor P_j to 0.1 based on our analysis of the training set. Due to the lack of sufficient samples in certain relation classes, we ignored the relationship directionality while computing the Relation heuristic. Finally, the dimensionality of the POS tag vectors and the position vectors is set to 25, and the Generator is set to ignore nouns, articles and adjectives.

[2]https://code.google.com/p/word2vec

Finally, after training for 10 iterations, the Generator includes the output from the Corpus and Rationale heuristics. The K frequently used words in the Corpus heuristic are computed for each relation class, where the threshold for K_l is 10% of number of sentences of the given l relation class. The Rationale heuristic selects the top 25% of words, hw^l, output by the Word heuristic.

We used the Adagrad optimizer in our experiments and the learning rate is initialized to 0.01, and reduces by 10% after every iteration. During each training iteration, we sample 25 times from the Generator, and re-sample 25 times when training the Selector and the Encoder.

5.3 Results

We present a comparison of relation classification between our model and the state-of-the-art models in Table 2. Dependency tree models are neural network models that work with word dependency tree parsed by a grammar parser. Each model utilizes the dependency tree differently. For instance MVRNN (Socher et al., 2012) uses the whole dependency tree, while SPTree (Miwa and Bansal, 2016) experiments using the entire tree and the nodes on the path that connect the target entities. We refer to the models that do not use a dependency tree as independent models.

Classifier	**F1**
Manually Engineered Models	
SVM(Rink and Harabagiu, 2010)	82.2
Dependency Tree Models	
RNN (Socher et al., 2012)	77.6
MVRNN (Socher et al., 2012)	82.4
FCM (Yu et al., 2014)	83.0
Hybrid FCM (Gormley et al., 2015)	83.4
DepNN (Liu et al., 2015)	83.6
DRNNs (Xu et al., 2015)	85.6
SPTree (Miwa and Bansal, 2016)	84.5

Classifier	**F1**
Independent Models	
CNN & softmax (Zeng et al., 2014)	82.7
Stackforward(Nguyen and Grishman, 2015)	83.4
Vote-bidirect (Nguyen and Grishman, 2015)	84.1
Vote-backward (Nguyen and Grishman, 2015)	84.1
ATT-Input-CNN (Wang et al., 2016)	87.5
ATT-Pooling-CNN (Wang et al., 2016)	88.0
Rationale Models	
Proxy-Rationale-1(Hsu et al., 2018)	84.5
Proxy-Rationale-2(Hsu et al., 2018)	85.3
Proxy-Rationale-3(Hsu et al., 2018)	87.8
GAN(Hsu et al., 2018)	89.5
Our Model	**90.7**

Table 2: Comparisons with benchmarking models

Table 3, presents a detailed comparison between the different variations of our model and other rationale-based models. We evaluate the rationale quality using macro-F1, where precision and recall are computed by taking the intersection between the generated and the manually labeled rationales.

Rationale Models	**Relation F1**	**Rationale F1**
Proxy-Rationale-1(Hsu et al., 2018)	84.5	6.3
Proxy-Rationale-2(Hsu et al., 2018)	85.3	22.2
Proxy-Rationale-3(Hsu et al., 2018)	87.8	26.1
GAN(Hsu et al., 2018)	89.5	33.2
Our Model prior global heuristics s^{global}	89.5	37.1
Our Model after global heuristics s^{global}	**90.7**	**53.2**

Table 3: Comparisons between variations and with rationale models

Finally, we summarize a few samples of rationales chosen from the test set in Table 4. We observe that the extracted rationales are similar to the rationales chosen manually and they can be used to infer the relationship between entities.

Relation Class	Selected rationales with target entities
Cause-Effect	dips$_{e1}$ caused by ... y-rays$_{e2}$, liver$_{e1}$ ... cause ... hypertension$_{e2}$ plaintiffs$_{e1}$ ...resulting from ... explosion$_{e2}$, storm$_{e1}$ generated by ... cold$_{e2}$
Component-Whole	site$_{e1}$ is part of ... network$_{e2}$, cabinet$_{e1}$ encloses ... woofer$_{e2}$, ladder$_{e1}$ comprises ... steps$_{e2}$, crocodile$_{e1}$ has ... snout $_{e2}$
Content-Container	guns$_{e1}$ are locked in safe$_{e2}$, grenade$_{e1}$ hidden inside canister$_{e2}$ spider$_{e1}$ was contained in ... box$_{e2}$, suitcase$_{e1}$ full ... books$_{e2}$
Entity-Destination	weapons$_{e1}$ ... delivered to ... navy$_{e2}$, money$_{e1}$ ... into ... custody$_{e1}$ children$_{e1}$ were handed over to relatives$_{e2}$, water$_{e1}$ ... been poured into ... river$_{e2}$
Entity-Origin	squirrel$_{e1}$ popped out of ... shirt$_{e2}$, robbers$_{e1}$ ... away from scene$_{e2}$ gases$_{e1}$ emanated from ... sources$_{e2}$, starch$_{e1}$ is source of sugars$_{e2}$
Instrument-Agency	mechanic$_{e1}$ tightens ... with spanner$_{e2}$, intellect$_{e1}$ wields pen$_{e2}$ project$_{e1}$ uses art$_{e2}$ as instrument, gives ... best practices$_{e1}$ for programmers$_{e2}$
Member-Collection	bloat$_{e1}$ of hippopotamuses$_{e2}$, formation$_{e1}$ comprised of ... dragoons$_{e2}$ surgeon$_{e1}$ is part of the team$_{e2}$, sergeant$_{e1}$ in army$_{e2}$
Message-Topic	caveats$_{e1}$ outlined ... e-mail$_{e2}$, speech$_{e1}$ ... about conversation$_{e2}$ speech$_{e1}$ was summary of ... problems$_{e2}$, parameters$_{e1}$ described text$_{e2}$
Product-Producer	production materials$_{e1}$ by industries$_{e2}$, products$_{e1}$ grown by ... defendant$_{e2}$ engineers$_{e1}$ ... devised ... method$_{e2}$, investment firm$_{e1}$ co-founded by ... head$_{e2}$

Table 4: List of words/phrases that are selected as rationales in test set by our model. Additional words are denoted as ... if they are located between entities and rationales but are not selected as rationales.

6 Discussion and Future Goals

6.1 Logistic Regressions and RNN

The major difference between this work and the previous rationale-based research is that we designed the Generator without a sequential RNN model. Specifically, we developed a layer by layer process that considers different levels of text information for three reasons. First, in relation classification, the excess sequential modeling power from RNN is not required since rationales are often an extremely small span of words. Also, rationales are often strongly-related to POS tags which can be used to replace the sequence information. Second, the purpose of rationales in relation classification is to augment features and illuminate how the model derives the answer. This is different from other tasks like aspect-level sentiment classification in Lei et al. (2016) where the rationales are treated as a compression of the input. Although more empirical experiments are required, we hypothesize that this simpler multilevel heuristic approach suits better for tasks with simpler rationales. Finally, dropping RNN can help the model parallelize and also less prone to mode collapse and gradient vanishing.

6.2 Axillary Information

The improvement in rationale interpretability comes from regularizing the Generator through a wide set of knowledge sources, like the Relation, the Corpus and the Rationale heuristic, and we believe this can be further improved by including more knowledge sources. For example, the Entity heuristic can be improved by annotating entities through a large open source data such Wikipedia, or the Corpus heuristic can be more diverse with distant supervision tricks. However, the amount of axillary information to include and to what degree is a challenge in itself.

Another major difference is that the Generator is directly related to the relation-labels through the Relation and Word heuristics, which is different from Hsu et al. (2018) where the Generator is not aware of the ground-truth. This can be a potential challenge when the diversity of relation vastly increases. To be more specific, the number of relations in some knowledge graph population tasks can be more than a few thousands, which can lead to severe label imbalance issues. Also, we notice several relation classes share the same rationales, like 'of' is commonly seen in Member-Collection and Component-

Whole relations. This can be unfavorable for the Word heuristic since our approach assumes rationales are more heterogeneous than homogeneous between different relations. A potential solution is to cluster or introduce hierarchical structures to the relations thereby reducing diversity.

6.3 Potential Applications: Noise-Reduction in Distant Supervision

A potential application of this work, besides classifying relations and providing interpretable results, is to help reduce noise in distant supervision for relation classification. Distant supervision is a commonly used approach in relation classification where producing ground-truth data is expensive. Distant supervision exploits a simple assumption: if a sentence s contains an entity-relation pair $(e1, relation, e2)$ that exists in a trustworthy knowledge source, then s can be a training data for the $relation$. As a result, distant supervision often contains many noisy false-negative training samples and degrades the result. To reduce the noise, one can penalize sentences that do not contain commonly used rationales for the relation.

7 Conclusion

In this paper, we have proposed an improved rationale-based model for entity relation classification. In our model, besides context word information, we also moderate rationale generation with multiple heuristics computed from different text level features. With multilevel heuristics, we successfully reduce the variability in the Generator to achieve meaningful rationales. Quantitative analysis demonstrates that our model improves both classification performance and the rationale quality. Finally, we provide an annotated test set for rationales, which can be used in future related research to evaluate rationales.

Acknowledgments

We thank the COLING reviewers for their detailed and insightful feedback, and also thank the annotators for their intensive efforts for developing the annotated set. This material is based upon work supported in whole or in part with funding from LAS. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the LAS and/or any agency or entity of the United States Government.

References

Martin Arjovsky and Leon. Bottou. 2017. Towards principled methods for training generative adversarial networks. In *arXiv preprint arXiv:1701.04862*.

Martin Arjovsky, Soumith Chintala, and Leon. Bottou. 2017. Wasserstein gan. In *arXiv preprint arXiv:1701.07875, 2017*.

Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie. Li. 2017. Mode regularized generative adversarial networks. In *ICLR*.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*.

Bang Duhyeon and Shim Hyunjung. 2018. Improved training of generative adversarial networks using representative features. In *arxiv: preprint arXiv:1801.09195*.

Claudio Giuliano, Alberto Lavelli, and Lorenza Romano. 2006. Exploiting shallow linguistic information for relation extraction from biomedical literature. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

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

Matthew R Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. In *EMNLP*, pages 1774–1784.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron. Courville. 2017. Improved training of wasserstein gans. In *arXiv preprint arXiv:1704.00028*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *NAACL HLT*, pages 94–99.

Shiou Tian Hsu, Changsung Moon, Paul Jones, and Nagiza Samatova. 2018. An interpretable generative adversarial approach to classification of latent entity relations from unstructured sentences. In *AAAI*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt. Kira. 2017. On convergence and stability of gans. In *arXiv preprint arXiv:1705.07215*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *EMNLP*, pages 107–117.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. 2015. A dependency-based neural network for relation classification. In *ACL*, pages 285–290.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *ACL*, pages 1105–1116.

Thien Huu Nguyen and Ralph Grishman. 2015. Combining neural networks and log-linear models to improve relation extraction. In *IJCAI Workshop on Deep Learning for Artificial Intelligence (DLAI)*.

Feng Niu, Ce Zhang, Christopher Ré, and Jude W Shavlik. 2012. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. *VLDS*, 12:25–28.

Bryan Rink and Sanda Harabagiu. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. In *SemEval*, pages 256–259.

Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. 2017. Stabilizing training of generative adversarial networks through regularization. In *NIPS*.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2234–2242.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP*, pages 1201–1211.

Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles Sutton. 2017. Veegan: Reducing mode collapse in gans using implicit variational learning. In *NIPS*.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *ACL*, pages 1298–1307.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015. Semantic relation classification via convolutional neural networks with simple negative sampling. In *EMNLP*, pages 536–540.

Mo Yu, Matthew Gormley, and Mark Dredze. 2014. Factor-based compositional embedding models. In *NIPS*, pages 95–101.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344.

Ye Zhang, Iain Marshall, and Byron C Wallace. 2016. Rationale-augmented convolutional neural networks for text classification. In *EMNLP*, pages 795–804.

Adversarial Multi-lingual Neural Relation Extraction

Xiaozhi Wang[1*], **Xu Han**[1*], **Yankai Lin**[1], **Zhiyuan Liu**[1†], **Maosong Sun**[1,2]
[1]Department of Computer Science and Technology,
State Key Lab on Intelligent Technology and Systems,
Beijing National Research Center for Information Science and Technology,
Tsinghua University, Beijing, China
[2]Beijing Advanced Innovation Center for Imaging Technology,
Capital Normal University, Beijing, China

Abstract

Multi-lingual relation extraction aims to find unknown relational facts from text in various languages. Existing models cannot well capture the consistency and diversity of relation patterns in different languages. To address these issues, we propose an adversarial multi-lingual neural relation extraction (AMNRE) model, which builds both consistent and individual representations for each sentence to consider the consistency and diversity among languages. Further, we adopt an adversarial training strategy to ensure those consistent sentence representations could effectively extract the language-consistent relation patterns. The experimental results on real-world datasets demonstrate that our AMNRE model significantly outperforms the state-of-the-art models. The source code of this paper can be obtained from `https://github.com/thunlp/AMNRE`.

1 Introduction

Relation extraction (RE) is a crucial task in NLP, which aims to extract semantic relations between entity pairs from the sentences containing them. For example, given an entity pair (*Bill Gates*, *Microsoft*) and a sentence "*Bill Gates* is the co-founder and CEO of *Microsoft*", we want to figure out the relation `Founder` between the two entities. RE can potentially benefit many applications, such as knowledge base construction (Zhong et al., 2015; Han et al., 2018) and question answering (Xiang et al., 2017).

Recently, neural models have shown their great abilities in RE. Zeng et al. (2014) introduce a convolutional neural network (CNN) to extract relational facts with automatically learning features from text. To address the issue of lack of data, Zeng et al. (2015) incorporate multi-instance learning with a piece-wise convolutional neural network (PCNN) to extract relations in distantly supervised data. Because distant supervision suffer from wrong labeling problems, Lin et al. (2016) further employ a sentence-level selective attention to filter out those noisy sentences in distantly supervised data and achieve state-of-the-art performance. All these neural relation extraction (NRE) models merely focus on extracting relational facts from mono-lingual data, ignoring the rich information in multi-lingual data.

Lin et al. (2017) propose a multi-lingual attention-based neural relation extraction (MNRE) model, which considers the consistency and complementarity in multi-lingual data. MNRE builds a sentence representation for each sentence in various languages and employs a multi-lingual attention to capture the pattern consistency and complementarity among languages.

Although MNRE achieves great success in multi-lingual RE, it still has some problems. MNRE learns a single representation for each sentence in various languages, which cannot well capture both the consistency and diversity of relation patterns in different languages. Moreover, MNRE simply utilizes a multi-lingual attention mechanism and a global relation predictor to capture the consistent relation patterns among multiple languages. From the experimental data, we find that the sentence representations in different languages are still far from each other and linearly separable. Therefore, it is hard for the multi-

[*] indicates equal contribution
[†] Corresponding author: Zhiyuan Liu (liuzy@tsinghua.edu.cn).

Proceedings of the 27th International Conference on Computational Linguistics, pages 1156–1166
Santa Fe, New Mexico, USA, August 20-26, 2018.

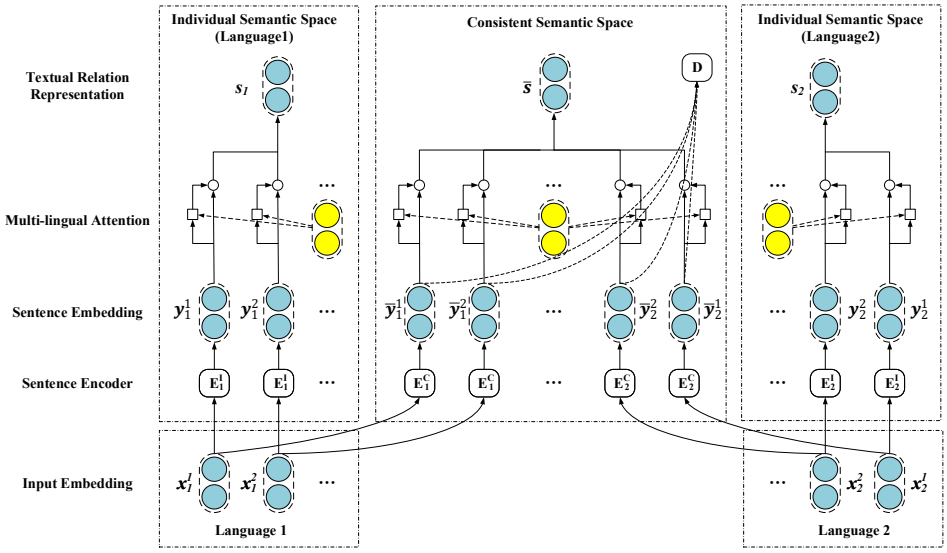


Figure 1: Overall architecture of our adversarial multi-lingual neural relation extraction (AMNRE) which contains two languages.

lingual attention mechanism and global relation predictor to extract relation consistency from distinct sentence representations.

To address these issues, we propose an adversarial multi-lingual NRE (AMNRE) model. As shown in Figure 1, for an entity pair, we encode its corresponding sentences in various languages through neural sentence encoders. For each sentence, we build an individual representation to grasp its individual language features and a consistent representation to encode its substantially consistent features among languages. Further, we adopt an adversarial training strategy to ensure AMNRE can extract the language-consistent relation patterns from the consistent representations. Orthogonality constraints are also adopted to enhance differences between individual representations and consistent representations for each language.

In experiments, we take Chinese and English to show the effectiveness of AMNRE. The experimental results show that AMNRE outperforms all baseline models significantly by explicitly encoding the consistency and diversity among languages. And we further give a case study and an ablation study to demonstrate the adversarial training strategy could help AMNRE to capture language-consistent relation patterns.

2 Related Works

2.1 Relation Extraction

Traditional supervised RE models (Zelenko et al., 2003; Socher et al., 2012; Santos et al., 2015) heavily rely on abundant amounts of high-quality annotated data. Hence, Mintz et al. (2009) propose a distantly supervised model for RE. Distant supervision aligns knowledge bases (KBs) and text to automatically annotate data, and thus distantly supervised models inevitably suffer from wrong labeling problems.

To alleviate the noise issue, Riedel et al. (2010) and Hoffmann et al. (2011) propose multi-instance learning (MIL) mechanisms for single-label and multi-label problems respectively. Then, Zeng et al. (2015) attempt to integrate neural models into distant supervision. Lin et al. (2016) further propose a sentence-level attention to jointly consider all sentences containing same entity pairs for RE. The attention-based neural relation extraction (NRE) model has become a foundation for some recent works (Ji et al., 2017; Zeng et al., 2017; Liu et al., 2017b; Wu et al., 2017; Feng et al., 2018; Zeng et al., 2018).

Most existing RE models are devoted to extracting relations from mono-lingual data and ignore information lying in text of multiple languages. Faruqui and Kumar (2015) and Verga et al. (2016) first attempt to adopt multi-lingual transfer learning for RE. However, both of these works learn predictive

models on a new language for existing KBs, without fully leveraging semantic information in text. Then, Lin et al. (2017) construct a multi-lingual NRE (MNRE) model to jointly represent text of multiple languages to enhance RE. In this paper, we propose a novel multi-lingual NRE framework to explicitly encode language consistency and diversity into different semantic spaces, which can achieve more effective representations for RE.

2.2 Adversarial Training

Goodfellow et al. (2015) propose adversarial training for image classification tasks. Afterwards, Goodfellow et al. (2014) propose a mature adversarial training framework and use the framework to train generative models. Adversarial networks have recently been used as methods to narrow probability distributions and proven effective in some tasks. In domain adaptation, Ganin et al. (2016) and Bousmalis et al. (2016) adopt adverarial training strategies to transfer the features of one source domain to its corresponding target domain.

Inspired by Ganin et al. (2016), adversarial training has also been explored in some typical NLP tasks for multi-feature fusion. Park and Im (2016) propose a multi-modal representation learning model based on adversarial training. Then, Liu et al. (2017a) employ adversarial training to construct a multi-task learning model for text classification by extending the original binary adversarial training to the multi-class version. And a similar adversarial framework is also adapted by Chen et al. (2017) to learn features from different datasets for chinese word segmentation. In this paper, we adopt adversarial training to boost feature fusion to grasp the consistency among different languages.

3 Methodology

In this section, we introduce the overall framework of our proposed AMNRE in detail. As shown in Figure 1, for each entity pair, AMNRE encodes its corresponding sentences in different languages into several semantic spaces to grasp their individual language patterns. Meanwhile, a unified space is also set up to encode consistent features among languages. By explicitly encoding the consistency and diversity among languages, AMNRE can achieve better extraction results in the multi-lingual scenario.

For each given entity pair, we define its corresponding sentences in n different languages as $\mathcal{T} = \{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$, where $\mathcal{S}_j = \{x_j^1, \ldots, x_j^{|\mathcal{S}_j|}\}$ denotes the sentence set in the j-th language. All these sentences are labeled with the relation $r \in \mathcal{R}$ by heuristical labeling algorithms in distant supervision (Mintz et al., 2009). Our model aims to learn a relation extractor by maximizing the conditional probability $p(r|\mathcal{T})$ with the following three components:

Sentence Encoder. Given a sentence and its target entity pair, we employ neural networks to encode the sentence into a embedding. In this paper, we implement the sentence encoder with both convolutional (CNN) and recurrent (RNN) architectures. Specifically, we set the encoders E_j^I and E_j^C to encode each sentence in the j-th language into its individual and consistent embeddings respectively, and expect these embeddings to capture the diversity and consistency among languages.

Multi-lingual Attention. Since not all sentences are labeled correctly in distant supervision, we adopt multi-lingual attention mechanisms to capture those informative sentences. In practice, we apply language-individual and language-consistent attentions to compute local and global textual relation representations respectively for final prediction.

Adversarial Training. Under the framework of AMNRE, we encode the sentences in various languages into a unified consistent semantic space. We further adopt adversarial training to ensure these sentences are well fused in the unified space after encoding so that our model can effectively extract the language-consistent relation patterns.

We will introduce the three components in detail as follows.

3.1 Sentence Encoder

Given a sentence $x = \{w_1, w_2, \ldots\}$ containing two entities, we apply neural architectures including both CNN and RNN to encode the sentence into a continuous low-dimensional space to capture its implicit semantics.

3.1.1 Input Layer

The input layer transforms all input words in the sentence into corresponding input embeddings by concatenating their word embeddings and position embeddings. The word embeddings are pre-trained by Skip-Gram (Mikolov et al., 2013). The position embeddings are a widely-used technique in RE proposed by Zeng et al. (2014), representing each word's relative distances to the two entities into two k_p-dimensional vectors. The input layer represents the input sentence as a k_i-dimensional embedding sequence $\mathbf{x} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots\}$, where $k_i = k_w + k_p \times 2$, k_w and k_p are the dimensions of word embeddings and position embeddings respectively.

3.1.2 Encoding Layer

After representing the input sentence as a k_i-dimensional embedding sequence, we select both CNN (Zeng et al., 2014) and RNN (Zhang and Wang, 2015) to encode the input embedding sequence $\mathbf{x} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots\}$ to its sentence embedding.

CNN slides a convolution kernel with the window size m to extract the k_h-dimensional local features,

$$\mathbf{h}_i = \text{CNN}\big(\mathbf{w}_{i-\frac{m-1}{2}}, \ldots, \mathbf{w}_{i+\frac{m-1}{2}}\big). \tag{1}$$

A max-pooling is then adopted to obtain the final sentence embedding $\mathbf{y}$ as follows,

$$[\mathbf{y}]_j = \max\{[\mathbf{h}_1]_j, \ldots, [\mathbf{h}_n]_j\}. \tag{2}$$

RNN is mainly designed for modeling sequential data. In this paper, we adopt bidirectional RNN (Bi-RNN) to incorporate information from both sides of the sentence sequence as follows,

$$\overrightarrow{\mathbf{h}}_i = \text{RNN}_f(\mathbf{x}_i, \overrightarrow{\mathbf{h}}_{i-1}), \quad \overleftarrow{\mathbf{h}}_i = \text{RNN}_b(\mathbf{x}_i, \overleftarrow{\mathbf{h}}_{i+1}), \tag{3}$$

where $\overrightarrow{\mathbf{h}}_i$ and $\overleftarrow{\mathbf{h}}_i$ are the k_h-dimensional hidden states at the position i of the forward and backward RNN respectively. $\text{RNN}(\cdot)$ is the recurrent unit and we select gated recurrent unit (GRU) (Cho et al., 2014) as the recurrent unit in this paper. We concatenate both the forward and backward hidden states as the sentence embedding $\mathbf{y}$,

$$\mathbf{y} = [\overrightarrow{\mathbf{h}}_n; \overleftarrow{\mathbf{h}}_1]. \tag{4}$$

For simplicity, we denote such a sentence encoding operation as the following equation,

$$\mathbf{y} = \text{E}(x). \tag{5}$$

For each sentence $x_j^i \in \mathcal{S}_j$, we adopt the individual sentence encoder E_j^I and the consistent sentence encoder E_j^C to embed the sentence into its individual and consistent representations respectively,

$$\{\mathbf{y}_j^1, \mathbf{y}_j^2, \ldots\} = \{\text{E}_j^I(x_j^1), \text{E}_j^I(x_j^2), \ldots\}, \quad \{\bar{\mathbf{y}}_j^1, \bar{\mathbf{y}}_j^2, \ldots\} = \{\text{E}_j^C(x_j^1), \text{E}_j^C(x_j^2), \ldots\}. \tag{6}$$

3.2 Multi-lingual Selective Attention

For each given entity pair, AMNRE adopts multi-lingual selective attention mechanisms to exploit informative sentences in $\mathcal{T}$. We explicitly encode languages' consistency and diversity into individual and consistent representations, thus our attentions are more simple than those proposed in Lin et al. (2017).

3.2.1 Language-individual Attention

Since it is intuitive that each language has its own characteristic, we set language-individual attention mechanisms for different languages. In the individual semantic space of the j-th language, we assign a query vector $\mathbf{r}_j$ to each relation $r \in \mathcal{R}$. The attention score for each sentence in $\mathcal{S}_j = \{x_j^1, x_j^2, \ldots\}$ is defined as follows,

$$\alpha_j^i = \frac{\exp(\mathbf{r}_j^\top \mathbf{y}_j^i)}{\sum_{k=1}^{|\mathcal{S}_j|} \exp(\mathbf{r}_j^\top \mathbf{y}_j^k)}. \tag{7}$$

The attention scores can be used to compute language-individual textual relation representations,

$$\mathbf{s}_j = \sum_{k=1}^{|\mathcal{S}_j|} \alpha_j^k \mathbf{y}_j^k. \tag{8}$$

3.2.2 Language-consistent Attention

Besides language-individual attention mechanisms, we also adopt a language-consistent attention to take all sentences in all languages into consideration. In the consistent semantic space, we also assign a query vector $\bar{\mathbf{r}}$ to each relation $r \in \mathcal{R}$ and the attention score for each sentence is defined as follows,

$$\beta_j^i = \frac{\exp(\bar{\mathbf{r}}^\top \bar{\mathbf{y}}_j^i)}{\sum_{l=1}^{n} \sum_{k=1}^{|\mathcal{S}_l|} \exp(\bar{\mathbf{r}}^\top \bar{\mathbf{y}}_l^k)}. \tag{9}$$

The attention scores can be used to compute language-consistent textual relation representations,

$$\bar{\mathbf{s}} = \sum_{l=1}^{n} \sum_{k=1}^{|\mathcal{S}_l|} \beta_l^k \bar{\mathbf{y}}_l^k. \tag{10}$$

3.3 Relation Prediction

With the language-individual textual relation representations $\{\mathbf{s}_1, \mathbf{s}_2, \ldots\}$ and the language-consistent textual relation representation $\bar{\mathbf{s}}$, we can estimate the probability $p(r|\mathcal{T})$ over each relation $r \in \mathcal{R}$,

$$p(r|\mathcal{T}) = p(r|\bar{\mathbf{s}}) \prod_{j=1}^{n} p(r|\mathbf{s}_j). \tag{11}$$

$p(r|\bar{\mathbf{s}})$ and $p(r|\mathbf{s}_j)$ can be defined as follows,

$$p(r|\mathbf{s}_j) = \text{softmax}[\mathbf{R}_j \mathbf{s}_j + \mathbf{d}_j], \quad p(r|\bar{\mathbf{s}}) = \text{softmax}[\bar{\mathbf{R}}\bar{\mathbf{s}} + \bar{\mathbf{d}}], \tag{12}$$

where $\mathbf{d}_j$ and $\bar{\mathbf{d}}$ are bias vectors, $\mathbf{R}_j$ is the specific relation matrix of the j-th language, and $\bar{\mathbf{R}}$ is the consistent relation matrix. We define the objective function to train the relation extractor as follows,

$$\min_{\theta} \mathcal{L}_{nre}(\theta) = -\sum_{l} \log p(r_l|\mathcal{T}_l), \tag{13}$$

where θ is all parameters in the framework. In the training phase, $p(r|\mathcal{T})$ is computed using the labeled relations as the attention queries. In the test phase, we need to use each possible relation as attention queries to compute $p(r|\mathcal{T})$ for relation prediction since the relations are unknown in advance.

3.4 Adversarial Training

In our framework, we encode sentences of various languages into a consistent semantic space to grasp the consistency among languages. One possible situation is that sentences of different languages are aggregated in different places of the space and linearly separable. In this case, our purpose of mining substantially consistent relation patterns in different languages is difficult to be reached. Inspired by Ganin et al. (2016), we adopt adversarial training into our framework to address this problem.

In the adversarial training, we define a discriminator to estimate which kind of languages the sentences from. The probability distributions over these sentences are formalized as follows,

$$\text{D}(\bar{\mathbf{s}}_j^i) = \text{softmax}(\text{MLP}(\bar{\mathbf{s}}_j^i)), \tag{14}$$

where MLP is a two-layer multilayer perceptron network.

Contrary to the discriminator, the consistent sentence encoders are expected to produce sentence embeddings that cannot be reliably predicted by the discriminator. Hence, the adversarial training process is a min-max game and can be formalized as follows,

$$\min_{\theta_{\text{E}}^C} \max_{\theta_{\text{D}}} \sum_{j=1}^{n} \sum_{i=1}^{|\mathcal{S}_j|} \log[\text{D}(\text{E}_j^C(x_j^i))]_j, \tag{15}$$

where $[\cdot]_j$ is the j-th value of the vector.

The formula means that given a sentence of any language, the corresponding sentence encoder of its language generates the sentence embedding to confuse the discriminator. Meanwhile, the discriminator

tries its best to predict the language of the sentence according to the sentence embedding. After sufficient training, the encoders and the discriminator reach a balance, and sentences of different languages containing similar semantic information can be well encoded into adjacent places of the space. In training, we optimize the following loss functions instead of Eq. 15,

$$\min_{\theta_{\mathrm{E}}^C} \mathcal{L}_{adv}^E(\theta_{\mathrm{E}}^C) = \sum_l \sum_{\mathcal{S}_j \in \mathcal{T}_l} \sum_{x_j^i \in \mathcal{S}_j} \log[\mathrm{D}(\mathrm{E}_j^C(x_j^i))]_j, \quad \min_{\theta_{\mathrm{D}}} \mathcal{L}_{adv}^D(\theta_{\mathrm{D}}) = -\sum_l \sum_{\mathcal{S}_j \in \mathcal{T}_l} \sum_{x_j^i \in \mathcal{S}_j} \log[\mathrm{D}(\mathrm{E}_j^C(x_j^i))]_j, \tag{16}$$

where θ_{E}^C and θ_{D} are all parameters of the consistent sentence encoders and the discriminator.

We notice that language-individual semantics could be wrongly encoded into the consistent semantic space, and may have negative effects on extracting language-consistent features. Inspired by Bousmalis et al. (2016), we adopt orthogonality constraints to alleviate this issue. We minimize the following penalty function:

$$\min_{\theta_{\mathrm{E}}} \mathcal{L}_{penalty}(\theta_{\mathrm{E}}) = \sum_{j=1}^{n} \left\| \mathbf{I}_j^T \mathbf{C}_j \right\|_F, \tag{17}$$

where $\mathbf{I}_j$ and $\mathbf{C}_j$ are two matrices whose row vectors are the embeddings of sentences in the j-th language encoded by E_j^I and E_j^C respectively. θ_{E} is parameters of the all encoders. And $\|\cdot\|_F$ is the squared Frobenius norm.

3.5 Implementation Details

During training process, we combine the extraction and adversarial objective functions as follows,

$$\mathcal{L} = \mathcal{L}_{nre}(\theta) + \lambda_1 \mathcal{L}_{adv}^D(\theta_{\mathrm{D}}) + \lambda_2 \mathcal{L}_{adv}^E(\theta_{\mathrm{E}}^C) + \lambda_3 \mathcal{L}_{penalty}(\theta_{\mathrm{E}}), \tag{18}$$

where λ_1, λ_2, and λ_3 are harmonic factors. All models are optimized using stochastic gradient descent (SGD). In practice, we integrate λ_1 and λ_2 into the alternating ratio among the loss functions, and we calibrate a 1:1:5 ratio among $\mathcal{L}_{nre}(\theta) + \lambda_3 \mathcal{L}_{penalty}(\theta_{\mathrm{E}})$, $\mathcal{L}_{adv}^D(\theta_{\mathrm{D}})$ and $\mathcal{L}_{adv}^E(\theta_{\mathrm{E}}^C)$. λ_3 is set as 0.02.

4 Experiments

4.1 Datasets and Evaluation

We evaluate our models on a multi-lingual relation extraction dataset developed by Lin et al. (2017). The dataset consists of English and Chinese data, and has 176 relations including a special relation `NA` indicating that there is no relation between entities. The whole dataset is divided into three parts for training, validation and test. The statistics of the dataset are listed in Table 1.

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

Table 1: Statistics of the dataset

We evaluate all models by the held-out evaluation following previous works (Mintz et al., 2009; Lin et al., 2017). In experiments, we report precision-recall curves of recall under 0.3 since we focus more on the performance of those top-ranked results. To give a complete view of the performance, we also report the area under the curve (AUC).

4.2 Experiment Settings

Following the settings of previous works, we use the pre-trained word embeddings learned by Skip-Gram as the initial word embeddings. We implement the MNRE framework proposed by Lin et al. (2017) by ourselves. For fair comparision, we set most of the hyperparameters following Lin et al. (2017). We list the best setting of hyperparameters in Table 2.

Batch Size B	160	Convolution Kernel Size m	3
Learning Rate α	0.002	Dropout Probability p for CNN and RNN	0.5
Hidden Layer Dimension k_h for CNN	230	Dropout Probability p_d for the Discriminator	0.1
Hidden Layer Dimension k_h for RNN	200	Word Dimension k_w	50
Hidden Layer Dimension k_d for the Discriminator	2048	Position Dimension k_p	5

Table 2: Parameter settings.

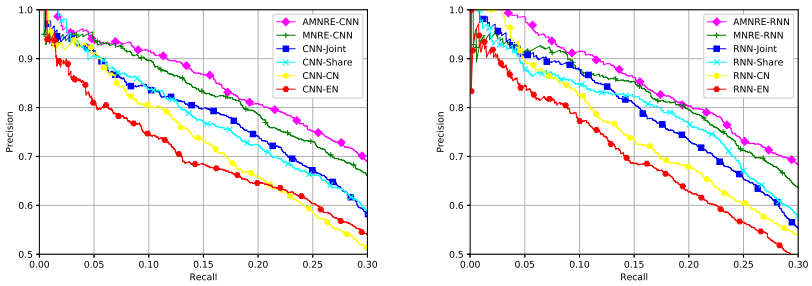


Figure 2: The aggregated precision-recall curves for proposed models and various baseline models. Left: models with CNN as sentence encoders. Right: models with RNN as sentence encoders.

4.3 Overall Evaluation Results

To evaluate the effectiveness of our proposed models **AMNRE-CNN** and **AMNRE-RNN**, we compare the proposed models with various neural methods: **MNRE-CNN** and **MNRE-RNN** are multi-lingual attention-based NRE models with CNN and RNN sentence encoders respectively (Lin et al., 2017); **CNN-EN** and **RNN-EN** are vanilla selective-attention NRE models trained with English data, which are the state-of-the-art models in mono-lingual RE (Lin et al., 2016); **CNN-CN** and **RNN-CN** are trained with Chinese data; **CNN-Joint** and **RNN-Joint** are naive joint models which predict relations by directly summing up ranking scores of both English and Chinese; **CNN-Share** and **RNN-Share** are another naive joint models which train English and Chinese models with shared relation embeddings. The results of precision-recall curves are shown in Figure 2 and the results of AUC are shown in Table 3.

From the results, we have the following observations:

(1) Both for CNN and RNN, the models jointly utilizing English and Chinese sentences outperform the models only using mono-lingual sentences. This demonstrates that the rich information in multi-lingual data is useful and can significantly enhance existing NRE models.

(2) The **-Joint** models achieve similar performance with the **-Share** models, and both of them underperform the **MNRE** and **AMNRE** models. They all benefit from the multi-lingual information, but the models with multi-lingual attentions can better take advantage of multi-lingual data. It indicates that designing targeted schemes to extract rich multi-lingual information is crucial.

(3) **AMNRE** achieves the best results among all the baseline models over the entire range of recall in Figure 2, even as compared with **MNRE**. **AMNRE** also outperforms **MNRE** with 3 percentage points increasing in the AUC results. It indicates our proposed framework which explicitly encodes language-consistent and language-individual semantics better extract multi-lingual information, and therefore lead to the significant improvement in RE performance.

Models	CNN-EN	CNN-CN	CNN-Joint	CNN-Share	MNRE-CNN	AMNRE-CNN
AUC	36.6	33.2	37.1	37.0	43.4	**46.2**
Models	RNN-EN	RNN-CN	RNN-Joint	RNN-Share	MNRE-RNN	AMNRE-RNN
AUC	34.5	34.4	36.5	37.6	44.2	**47.3**

Table 3: The AUC results of different models (%).

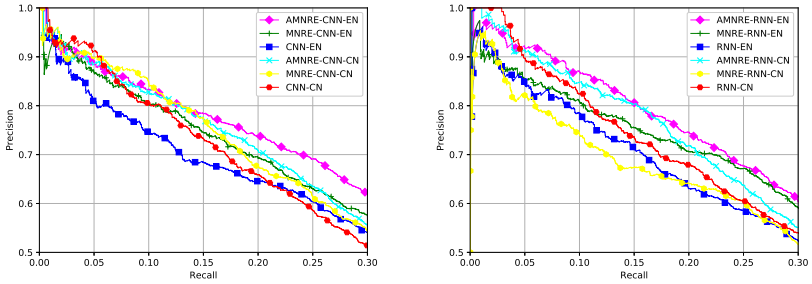


Figure 3: The aggregated precision-recall curves for proposed models and various baseline models in the mono-lingual scenario. Left: models with CNN as sentence encoders. Right: models with RNN as sentence encoders.

Models	CNN-EN	MNRE-EN	AMNRE-EN	RNN-EN	MNRE-EN	AMNRE-EN
AUC	36.6	39.6	**42.7**	34.5	42.2	**43.2**
Models	CNN-CN	MNRE-CN	AMNRE-CN	RNN-CN	MNRE-CN	AMNRE-CN
AUC	33.2	34.6	**37.9**	33.5	34.8	**36.4**

Table 4: The AUC results of different models in the mono-lingual scenario (%).

4.4 Mono-lingual Evaluation Results

To further verify that every mono-lingual RE models can benefit from our proposed framework, which explicitly consider language-consistent relation patterns, we train models with multi-lingual data and evaluate the performance of these models in the mono-lingual RE scenario. To show the results clearly, we report the precision-recall curves in Figure 3 and the AUC results in Table 4.

From the results, we can observe that:

(1) As compared with the models directly learned with the mono-lingual data, the models exploiting the multi-lingual information perform better in the mono-lingual scenario. This demonstrates that there is latent consistency among languages, and grasping this consistency from multi-lingual data can provide additional information for models in each language to enhance their results in the mono-lingual scenario.

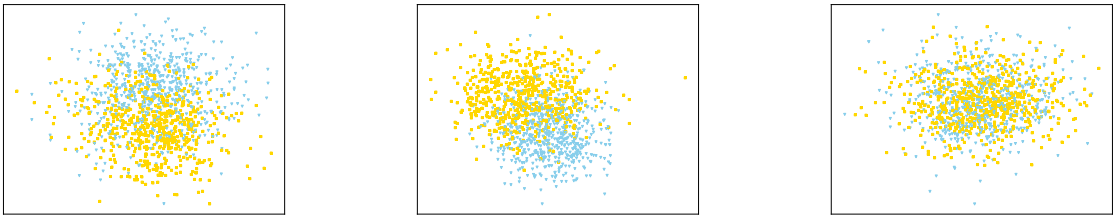
(2) Our proposed models achieve the best precision over the entire range of recall and also significantly improve the AUC results as compared with both MNRE and mono-lingual RE models. It indicates that due to the consistent semantic space in our framework, language-consistent information lying in the multi-lingual data is better mined and serve the mono-lingual scenario.

4.5 Effectiveness of Adversarial Training and Orthogonality Constraints

We adopt an adversarial training strategy to fuse the features from different languages to extract consistent relation patterns. Orthogonality constraints are also adopted to separate the consistent and individual feature spaces. To measure the effectiveness of them, we conduct an ablation study which compares the proposed models with the similar models but without adversarial training strategy (**AMNRE-noA**), without orthogonality constraints (**AMNRE-noO**), and without both of them (**AMNRE-noBoth**). The AUC results are shown in Table 5.

We can observe that both the adversarial training strategy and orthogonality constraints have significant influence on the performance of our proposed model. This demonstrates the effectiveness of adversarial training strategy and orthogonality constraints for multi-lingual RE. To give a more intuitive picture of the effect of these two mechanisms, we visualize the distribution of sentence feature embeddings encoded by the individual and consistent encoders using t-SNE (Maaten and Hinton, 2008). The results are shown in Figure 4.

Figure 4(a) shows that there are obvious differences between the feature embeddings encoded from the same sentences by individual and consistent encoders. It indicates the orthogonality constraints are effective to separate the individual and consistent latent spaces. From the comparison between Figure



(a) The same English sentences encoded by the consistent encoder (yellow) and individual encoder (blue).

(b) The English sentences (yellow) and the Chinese sentences (blue) encoded by their own consistent encoders without adversarial training.

(c) The English sentences (yellow) and the Chinese sentences (blue) encoded by their own consistent encoders with adversarial training.

Figure 4: The visualization of sentence feature embeddings with different mechanisms.

Models	AMNRE-CNN	AMNRE-CNN-noA	AMNRE-CNN-noO	AMNRE-CNN-noBoth
AUC	**46.2**	44.1	43.9	41.3
Models	AMNRE-RNN	AMNRE-RNN-noA	AMNRE-RNN-noO	AMNRE-RNN-noBoth
AUC	**47.3**	43.5	43.5	42.2

Table 5: The AUC results of the proposed models and ablated models.(%)

4(b) and Figure 4(c), we can observe that the feature embeddings from different languages are well-mixed due to the adversarial training strategy. We can more easily to grasp latent consistency among languages after multi-feature fusion.

5 Case Study

To further show the effectiveness of our proposed model to extract the language-consistent semantic information, we give an example in Table 6. We adopt the cosine similarity to measure the similarity between sentence embeddings encoded by consistent encoders. The first sentence in the middle column is the standard Chinese translation of the left sentence, thus they share the same semantic information. We observe that in our proposed model, the feature embedding similarity between these two sentences are significantly higher than the other English sentences sharing entity pair and relational fact but differing in semantics. It indicates that sentences in different languages containing similar semantics can be indeed encoded into adjacent places of the consistent space in our framework.

Relation : `Located in`		Cosine Similarity
There are eighteen small glaciers in the **North Island** on **Mount Ruapehu**.	北岛的**鲁阿佩胡山**上有十八个小冰川。	**0.584**
	...the bottom of the **North Island** of New Zealand up to the area of **Mount Ruapehu**.	0.3538
	It is located on the south-eastern **North Island** volcanic plateau, ...south-east of **Mount Ruapehu**.	0.342

Table 6: The example highlighting entities for the case study by measuring the cosine similarities between the sentence in the left column and each sentence in the middle column.

6 Conclusion and Future Work

In this paper, we introduce a novel adversarial multi-lingual neural relation extraction model (AMNRE). AMNRE builds both individual and consistent representations for each sentence to consider the consistency and diversity of relation patterns among languages. It also employs an adversarial training strategy and orthogonality constraints to ensure the consistent representations could extract the language-consistent features to extract relations. The experimental results on real-world datasets demonstrate that

our AMNRE could effectively encode the consistency and diversity among languages, and achieves state-of-the-art performance in relation extraction.

We will explore the following directions as our future work: (1) AMNRE can be also implemented in the scenario of multiple languages, and this paper shows the effectiveness of AMNRE on the dataset with two languages (English and Chinese). In the future, we will explore AMNRE in much more other languages such as French, Spanish, and so on. (2) AMNRE simply aligns the sentences with similar semantics in different languages with an adversarial training strategy. In fact, machine translation is a typical approach to align sentences in various languages. In the future, we will combine machine translation with our model to further improve the extraction performance.

Acknowledgments

We thank Jiacheng Zhang for his help. This work is supported by the National Natural Science Foundation of China (NSFC No. 61621136008, 61772302) and Tsinghua University Initiative Scientific Research Program (20151080406). This research is part of the NExT++ project, supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@Singapore Funding Initiative.

References

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Proceedings of NIPS*, pages 343–351.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for chinese word segmentation. In *Proceedings of ACL*, pages 1193–1203.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: encoder-decoder approaches. In *Proceedings of SSST-8*.

Manaal Faruqui and Shankar Kumar. 2015. Multilingual open relation extraction using cross-lingual projection. In *Proceedings of NAACL*, pages 1351–1356.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *Proceedings of AAAI*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *JMLR*, 17(1):2096–2030.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of NIPS*, pages 2672–2680.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of ICML*, pages 1–10.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *Proceedings of AAAI*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL*, pages 541–550.

Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Proceedings of AAAI*, pages 3060–3066.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of ACL*, pages 2124–2133.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. Neural relation extraction with multi-lingual attention. In *Proceedings of ACL*, pages 34–43.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017a. Adversarial multi-task learning for text classification. In *Proceedings of ACL*, pages 1–10.

Tianyu Liu, Kexiang Wang, Baobao Chang, and Zhifang Sui. 2017b. A soft-label method for noise-tolerant distantly supervised relation extraction. In *Proceedings of EMNLP*, pages 1790–1795.

Laurens Van Der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *JMLR*, 9(12):2579–2605.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*, pages 1–12.

Mintz, Mike, Steven, Jurafsky, and Dan. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL*, pages 1003–1011.

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

Sebastian Riedel, Limin Yao, and Andrew Mccallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of ECML-PKDD*, pages 148–163.

Cicero Nogueira Dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of ACL*, pages 626–634.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, pages 1201–1211.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew Mccallum. 2016. Multilingual relation extraction using compositional universal schema. In *NAACL*, pages 886–896.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *Proceedings of EMNLP*, pages 1778–1783.

Yang Xiang, Qingcai Chen, Xiaolong Wang, and Yang Qin. 2017. Answer selection in community question answering via attentive neural networks. *IEEE SPL*, 24(4):505–509.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *JMLR*, 3(2):1083–1106.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, pages 2335–2344.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of EMNLP*, pages 1753–1762.

Wenyuan Zeng, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. Incorporating relation paths in neural relation extraction. In *Proceedings of EMNLP*.

Xiangrong Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. Large scaled relation extraction with reinforcement learning. In *Proceedings of AAAI*.

Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network. *arXiv preprint arXiv:1508.01006*.

Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. 2015. Aligning knowledge and text embeddings by entity descriptions. In *Proceedings of EMNLP*, pages 267–272.

Neural Relation Classification with Text Descriptions

Feiliang Ren, Di Zhou, Zhihui Liu, Yongcheng Li, Rongsheng Zhao, Yongkang Liu, Xiaobo Liang
School of Computer Science and Engineering, Northeastern University, Shenyang, 110819, China
renfeiliang@cse.neu.edu.cn

Abstract

Relation classification is an important task in natural language processing fields. State-of-the-art methods usually concentrate on building deep neural networks based classification models on the training data in which the relations of the labeled entity pairs are given. However, these methods usually suffer from the data sparsity issue greatly. On the other hand, we notice that it is very easy to obtain some concise text descriptions for almost all of the entities in a relation classification task. The text descriptions can provide helpful supplementary information for relation classification. But they are ignored by most of existing methods. In this paper, we propose DesRC, a new neural relation classification method which integrates entities text descriptions into deep neural networks models. We design a two-level attention mechanism to select the most useful information from the *"intra-sentence"* aspect and the *"cross-sentence"* aspect. Besides, the adversarial training method is also used to further improve the classification performance. Finally, we evaluate the proposed method on the SemEval 2010 dataset. Extensive experiments show that our method achieves much better experimental results than other state-of-the-art relation classification methods.

1 Introduction

The aim of relation classification is that given a sentence in which two target entities are labeled, to select a proper relation for these two entities from a predefined relation set. For example, given a sentence "*The system as described above has its greatest application in an arrayed <e1>configuration </e1> of antenna <e2>elements</e2>*", a relation classification system aims to identify that there is a "*Component-Whole*" relation from *e2* to *e1*. Obviously, accurate relation classification results would benefit lots of natural language processing tasks, such as sentence interpretations, Q&A, knowledge graph construction, ontology learning, and so on. Thus, lots of researchers have devoted to this research field.

For relation classification, deep neural networks (DNN) based methods have been widely explored and have achieved state-of-the-art experimental results. However, when evaluating state-of-the-art relation classification methods on some standard datasets, experimental results show that there are usually huge performance gaps between different relations. Besides, when using the trained relation classification models to predict relations on new data, the prediction performance is usually far lower than expected. This is mainly because of the data sparsity issue: first, there are always some relations that have far less training data than others; second, the available training data is not sufficient enough to train a robust relation classification model.

On the other hand, we notice that for almost all of the entities in a relation classification task, there are usually available text descriptions for them on some Encyclopedia websites like Wikipedia, DBpedia, Wikidata, etc. For example, from the Wikipedia website, we can extract the following text descriptions as shown in Figure 1, where the example sentence is taken from the SemEval 2010 dataset.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1167–1177
Santa Fe, New Mexico, USA, August 20-26, 2018.

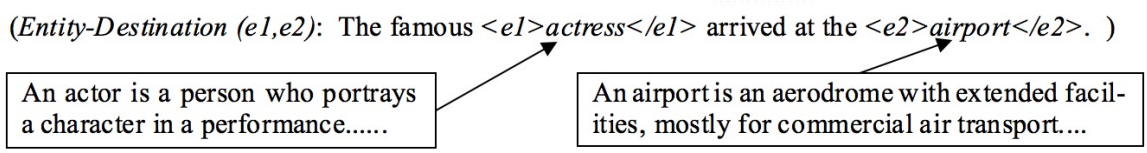


Figure 1: Example of text descriptions extracted from Wikipedia.

2 Related Work

Up to now, lots of novel relation classification methods have been proposed. Early research mainly focuses on features based methods. Usually, these methods firstly select some syntactic and semantic features from the given sentences. Then the selected features are fed into some classification models like support vector machines, maximum entropy, etc.

Recently, DNN based methods have been widely explored and have achieved state-of-the-art experimental results. The core of these methods is to embed features into real-valued vectors, and then feed these vectors into some DNN based learning frameworks. Generally, there are three widely used DNN frameworks for relation classification: convolutional neural networks (CNN), recurrent neural networks (RNN), and their combination. In most recent years, inspired by both the success of DNN methods and the broad consensus that syntactic tree structures are of great help for relation classification, more and more research attention is being paid to the methods that integrate syntactic tree features into DNN based learning frameworks. Among the syntactic tree features, the *Shortest Dependent Path* (SDP) is one of the most frequently used. In Table 1, we summarize some representative state-of-the-art DNN based relation classification methods.

<table>
<tr><th colspan="2">Learning Frameworks</th><th>Representative Methods</th><th>External resources</th><th>Loss function</th><th>Optimization method</th></tr>
<tr><td colspan="2" rowspan="2">CNN</td><td>Zeng et al., 2014</td><td>WordNet</td><td>Cross entropy</td><td rowspan="5">SGD</td></tr>
<tr><td>Dos Santos et al., 2015</td><td>No</td><td>Ranking loss</td></tr>
<tr><td rowspan="3">DNN+SDP</td><td>RNN+SDP</td><td>Xu et al., 2016</td><td rowspan="4">WordNet</td><td rowspan="4">Cross entropy</td></tr>
<tr><td>LSTM+SDP</td><td>Xu et al., 2015a</td></tr>
<tr><td>CNN+SDP</td><td>Xu et al., 2015b</td></tr>
<tr><td rowspan="3">Combination</td><td>SDP+ CNN+RNN</td><td>Cai et al., 2016</td><td>AdaDelta</td></tr>
<tr><td></td><td>Liu et al., 2015</td><td></td><td></td><td>NoReported</td></tr>
<tr><td>CNN+RNN</td><td>Vu et al., 2016</td><td rowspan="3">No</td><td>Ranking loss</td><td rowspan="2">SGD</td></tr>
<tr><td rowspan="2">DNN+Attention</td><td>CNN+ Attention</td><td>Wang et al., 2016</td><td>Distance based loss</td></tr>
<tr><td>LSTM+ Attention</td><td>Zhou et al., 2016</td><td rowspan="2">Cross entropy</td><td>AdaDelta</td></tr>
<tr><td colspan="2">End-to-End Joint Learning</td><td>Miwa et al., 2016</td><td>WordNet</td><td>Adam</td></tr>
</table>

Table1: A summarization of representative state-of-the-art relation classification methods.

From Table 1 we can see that there are many similarities among the state-of-the-art relation classification methods. For example, most of them use a cross entropy loss function, use WordNet, and use the stochastic gradient descent (SGD) method for optimization, etc. The main differences among them mainly lie in the learning frameworks.

CNN is a very popular learning framework for relation classification and lots of methods are based on it. For example, Zeng et al. (2014) proposed a CNN based approach for relation classification. In their method, sentence level features are learned through a CNN model that takes word embedding features and position embedding features as input. In parallel, lexical level features are extracted from some context windows that are around the labeled entities. Then the sentence level features and the lexical level features are concatenated into a single vector. This vector is then fed into a *softmax* classifier for relation prediction. Another representative CNN based relation classification method is CR-CNN

(Dos Santos et al., 2015), which tackles the relation classification task with a CNN model that performs classification by ranking. They proposed a new pairwise ranking loss function that is easy to reduce the impact of the artificial relation "*Other*". Their method is also the unique one that takes a specific processing strategy for the "*Other*" relation.

Xu et al. (2016) pointed out that compared with a raw word sequence or a whole parse tree, the SDP between two entities has two main advantages. First, it reduces irrelevant information; second, grammatical relations between words focus on the action and agents in a sentence and are naturally suitable for a relation classification task. Thus many researchers integrate SDP into DNN based learning frameworks for relation classification. For example, based on SDP, Xu et al. (2016) proposed deep recurrent neural networks (DRNNs) for relation classification. Their method can be roughly regarded as a "*RNN + SDP*" relation classification method. Xu et al. (2015a) proposed a neural relation classification architecture that picks up heterogeneous information along the left and right sub-path of the *SDP* respectively, leveraging RNN with multichannel *long short term memory* (LSTM) units. And their method can be roughly regarded as a "*LSTM + SDP*" relation classification method. Other similar work, Xu et al. (2015b) proposed to learn more robust relation representations from SDP through a CNN model; Liu et al. (2015) proposed augmented dependency path (ADP), which is a variant of SDP. Both of these two methods can be roughly regarded as a "*CNN + SDP*" relation classification method.

Some researchers combine CNN and RNN together for relation classification. For example, Vu et al. (2016) investigated CNN and RNN as well as their combination for relation classification. They proposed extended middle context, a new context representation for the CNN architecture. The extended middle context uses all parts of the sentence (the relation arguments, left/right and between of the relation arguments) and pays special attention to the middle part. Meanwhile, they proposed a connectionist bidirectional RNN model and introduced a ranking loss function for the RNN model. Finally, CNN and RNN are combined with a simple voting scheme. Cai et al. (2016) proposed a bidirectional neural network BRCNN, which consists of two RCNNs that can learn features along SDP inversely at the same time. Specifically, information of words and dependency relations is extracted by a two-channel RNN model with LSTM units. The features of dependency units in a SDP are extracted by a convolution layer. Liu et al. (2015) used a RNN model to learn the features of the sub-trees, and used a CNN model to capture the most important features on a SDP.

Recently, the attention method is achieving more and more research attention. Some researchers also add the attention method in their relation classification models. For example, Wang et al. (2016) proposed a multi-level attention CNN model for relation classification. In their method, two levels of attentions are used in order to better discern patterns in heterogeneous contexts. Zhou et al. (2016) proposed an attention-based bidirectional LSTM model for relation classification.

Another research line explores a kind of end-to-end method for relation classification. For example, Miwa et al. (2016) proposed a novel end-to-end neural model to extract entities and the relations between them. Their model captures both word sequence and dependency tree substructure information by stacking bidirectional tree-structured LSTM-RNNs on bidirectional sequential LSTM-RNNs, which allows the model to jointly represent both entities and relations with shared parameters in a single model.

3 Our Model

Figure2 demonstrates the architecture of our method. For each original training/test sentence S_i, it will be augmented to a new triplet format like $<S_i$, $Des_i(e_1)$, $Des_i(e_2)>$, where $Des_i(e_1)$ and $Des_i(e_2)$ are the text descriptions of the labeled entities e_1 and e_2 in S_i. Our model takes the augmented training/test sentences as input.

From Figure2 we can see that in our model, there are three parallel DNN-based encoders that are used to learn the real-valued vector representations for S_i, $Des_i(e_1)$, and $Des_i(e_2)$ respectively. Then with a "*cross-sentence*" attention method, the three learned vector representations are combined into one global real-valued vector representation. Finally, the classification decision is made based on the global vector representation.

For the description representation learning, we use a CNN-based method. And for the original training/test sentence representation learning, we use two methods: one is the CNN-based method that is the

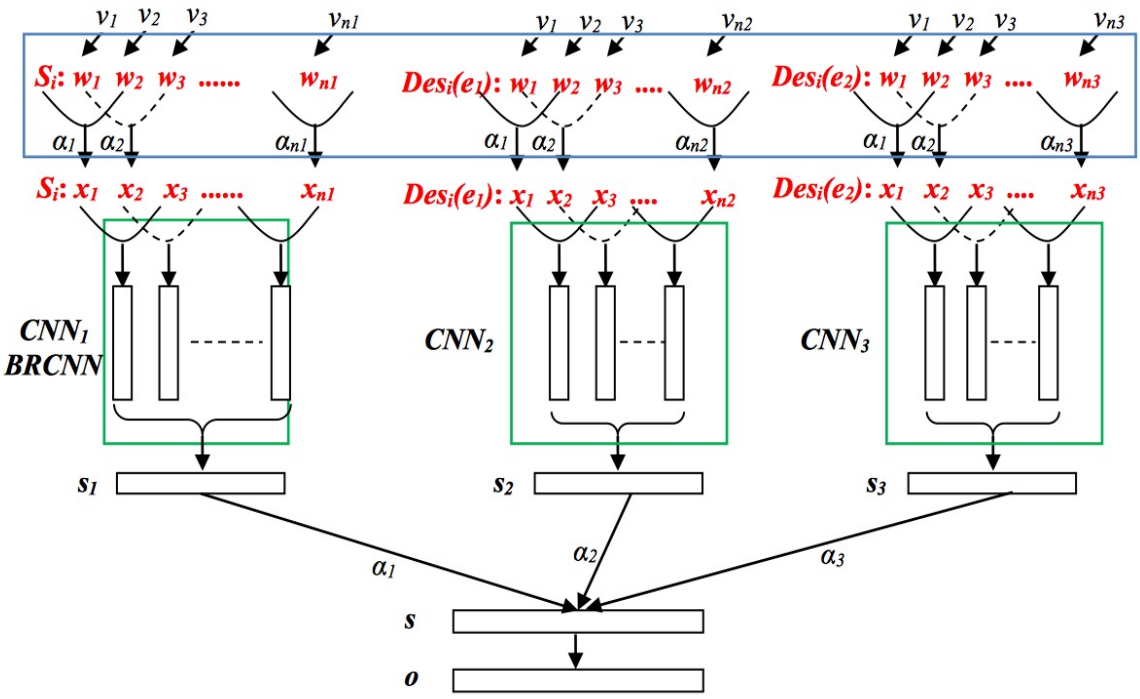


Figure 2: Architecture of DesRC

same as the one used in the description representation learning, and the other is a BRCNN-based method that is similar to the one proposed by Cai et al. (2016). In the three representation learning encoders, an "*intra-sentence*" attention method is used to select the most useful word information inside an input sentence. Besides, the adversarial training technique is also used on the word embedding level during training (in Figure2, v_i denotes the adversarial perturbation w.r.t. a word embedding w_i).

3.1 CNN-Based Encoder

Given a sentence, this encoder aims to transform it into a distributed representation via a CNN model. There is a same encoding process for both S_i and its two text descriptions $Des_i(e_1)$ and $Des_i(e_2)$. Here we take S_i as an example to demonstrate the whole CNN-based endoding process. First, each word in S_i is transformed into a real-valued vector representation. Then, a convolutional operation, a max-pooling operation, and a non-linear transformation operation are performed in turn. Finally, the encoder outputs a distributed representation for S_i.

Word Representation Given a sentence $S_i = (w_1, w_2, ..., w_n)$, we transform each of its word w_i into the concatenation of two kinds of embedding representations: 1) a word embedding that captures syntactic and semantic meaning of this word; and 2) a position embedding that specifies which input words are the labeled entities (or entity) or how close an input word to the labeled entities (or entity). Finally, the sentence S_i can be represented as a vector sequence $\boldsymbol{S}_i = (\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_n)$, where $\boldsymbol{w}_i \in R^d$ and $d = d_a + 2*d_b$ (for original training/test sentences) or $d = d_a + d_b$ (for descriptions). d_a and d_b are the dimension of word embeddings and position embeddings respectively.

***Intra-sentence* Attention** In a sentence, not all of its words are equally useful for the final classification decision. Some of them may be important, some of them may not. Thus, we design an "*intra-sentence*" attention method to automatically identify which words in a sentence are more important for the final classification decision. Following Feng et al. (2017), for each word $\boldsymbol{w}_i$ in $\boldsymbol{S}_i$, we take the embedding of its m context words and the corresponding entities' embeddings as input, the "*intra-sentence*" attention model outputs a new representation $\boldsymbol{x}_i$ for $\boldsymbol{w}_i$ with the following formula.

$$x_i = \sum_{l=1}^{w} \alpha_l * \boldsymbol{m}_l \tag{1}$$

where $\boldsymbol{m}_l \in R^d$ is the embedding of a considered context word, and the attention score α_l is defined as:

$$\alpha_l = \frac{exp(f_i)}{\sum_{j=1}^{n} exp(f_j)} \tag{2}$$

f_i is a function to evaluate the semantic relatedness of a text fragment with the given entity (for text descriptions) or the given entity pair (for the original training/test sentences) and the linked relation. It is calculated with the following formulas.

$$f_i = tanh(\boldsymbol{M} * [\boldsymbol{m_i}; \boldsymbol{w_{e1}}; \boldsymbol{w_{e2}}] + U * \boldsymbol{r}) \tag{3}$$

$$or \quad f_i = tanh(\boldsymbol{M_{1/2}} * [\boldsymbol{m_i}; \boldsymbol{w_{e1}/w_{e2}}] + U * \boldsymbol{r}) \tag{3'}$$

where $\boldsymbol{M} \in R^{d*3}$, $\boldsymbol{M}_{1/2} \in R^{d*2}$, $\boldsymbol{U} \in R^{dr}$, d_r is the dimension of relation, and $\boldsymbol{w}_{e1/e2}$ and $\boldsymbol{r}$ are the word embeddings of the labeled entities and the relation linked with them. The "[]" symbol denotes the concatenation operation. Similar to Lin et al. (2017), in the testing phase, since $\boldsymbol{r}$ is not known, we will take each possible relation into consideration. Finally, the "*intra-sentence*" attention model outputs a new representation for $\boldsymbol{S}_i$ that can be denoted as $\boldsymbol{S}_i = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)$.

Convolution Transformation After the above operation, a convolutional layer is used to extract the local features of an input sentence. This layer slides a word window of length *w* over the input sentence and performs a convolution operation within each sliding window. Its output for the *i-th* window is computed with the following formula.

$$\boldsymbol{C_i} = \boldsymbol{M_3} * \boldsymbol{cx_i} + \boldsymbol{b_1} \tag{4}$$

where $\boldsymbol{M}_3 \in R^{h1 * (d*w)}$, h_1 (a hyper-parameter) is the size of hidden units in the convolutional layer, $\boldsymbol{cx}_i$ is the embedding concatenation of the *w* word within the *i-th* window, and $\boldsymbol{b}_1$ is a bias term.

Max-pooling After the convolutional transformation, a max-pooling operation is applied to capture the most useful local features produced by the convolutional operation. This process can be written as the following formula.

$$\boldsymbol{p_i} = max_n \boldsymbol{C(i, n)}; \qquad 0 \leq i \leq h_1 \tag{5}$$

The result of max-pooling is an h_1-element real-valued vector whose size is no longer related to the length of the input sentence.

Non-linear Operation After the max-pooling operation, its output vectors $\boldsymbol{p}$ is fed into a non-linear transformation layer to generate the final representation for S_i with the following formula.

$$\boldsymbol{s_i} = tanh(\boldsymbol{M_4} * \boldsymbol{p} + \boldsymbol{b_2}) \tag{6}$$

where $\boldsymbol{M}_4 \in R^{h2*h1}$, h_2 (a hyper-parameter) is the size of hidden units in this layer, and $\boldsymbol{b}_2$ is a bias term.

3.2 BRCNN-Based Encoder

BRCNN is a relation classification model that is first proposed by Cai et al. (2016). It builds a relation classification model based on SDP, and can take full advantages of both the CNN model and the RNN model. Specifically, given a sentence and its dependency tree, the BRCNN model first extracts a SDP from the dependency tree. Then, along the SDP, two RNN models with LSTM units are used to learn hidden representations of words and dependency relations respectively. A convolution layer is applied to capture local features from hidden representations of every two neighbor words and the dependency relations between them. A max-pooling layer thereafter gathers information from local features of the SDP and the inverse SDP. Finally, a linear operation is performed to generate the final representation of the given sentence. More detail information can be referred to the work of Cai et al. (2016).

Here we take the BRCNN model as a new encoder to learn the representations of the original training/test sentences mainly for the following two reasons. First, BRCNN is one of the current-best relation classification models, and we want to explore the maximal potential of our method. Second, we want to explore the effectiveness of text descriptions in different relation classification frameworks.

However, it should be noted that the BRCNN-based encoder cannot be used in the description representation learning. This is because that the BRCNN model is based on SDP, which is extracted from the shorted dependency path between the **TWO** labeled entities, but a text description only focuses on **ONE** single entity. Thus it is impossible to extract a SDP from an entity's text description. Accordingly, the BRCNN based encoder cannot be utilized.

3.3 Representation Combination

After the CNN/BRCNN based representation learning, there will be three generated representations that are for the original training/test sentence and its two text descriptions respectively. We use a "*cross-sentence*" attention method to combine them into **ONE** global real-valued representation. With this attention method, we can capture the most useful information among different representations. The global representation $\boldsymbol{s}$ is computed as a weighted sum of the single representation vectors.

$$\boldsymbol{s} = \sum_i \alpha_i * \boldsymbol{s}_i \tag{7}$$

The attention score α_i is defined as:

$$\alpha_i = \frac{exp(g_i)}{\sum_{j=1}^{k} exp(g_j)} \tag{8}$$

g_i is a function to evaluate how well a representation vector reflects its corresponding entities (or entity) and the relation linked by the two labeled entities. It is calculated with the following formula.

$$g_i = \boldsymbol{s_i} * \boldsymbol{r} \tag{9}$$

where $\boldsymbol{r}$ is the embedding of the corresponding relation. In the testing phase, we will also take each possible relation into consideration

3.4 Classification Prediction

After the representation combination, the generated representation $\boldsymbol{s}$ is fed into a linear output layer to compute the confidence score for each possible relation. A *softmax* classifier is further used to get the probability distribution $\boldsymbol{y}$ over all relations. This process is written as formula10.

$$\boldsymbol{y} = softmax(\boldsymbol{M_5} * \boldsymbol{s} + \boldsymbol{b_3}) \tag{10}$$

where $\boldsymbol{M}_5 \in R^{h3*h2}$ and h_3 is the number of all possible relations.

3.5 Dropout Operation

Over-fitting is an issue that cannot be ignored in DNN models. Hinton et al. (2012) proposed the dropout method that has been proved to be effective for alleviating this issue. This method randomly sets a proportion (called drop rate, a hyper-parameter) of features to zero during training. It is expected to obtain less interdependent network units, thus the over-fitting issue is expected to be alleviated. In our method, we take dropout operations on $\boldsymbol{w}_i$ (see the word representation section) and $\boldsymbol{s}$ (see formula 7). The drop rates for them are denoted as $dp_{1\sim2}$ respectively.

3.6 Training Procedure

All the parameters in our method can be denoted as θ=($\boldsymbol{E}^w$,$\boldsymbol{E}^p$,$\boldsymbol{M}$,$\boldsymbol{M}_1$,$\boldsymbol{M}_2$,$\boldsymbol{M}_3$,$\boldsymbol{M}_4$,$\boldsymbol{M}_5$,$\boldsymbol{U}$,$\boldsymbol{b}_1$,$\boldsymbol{b}_2$,$\boldsymbol{b}_3$,$\boldsymbol{r}$),where $\boldsymbol{E}^w$ and $\boldsymbol{E}^p$ represent the embedding matrices of word and position respectively. In this paper, we use the *word2vecc* toolkit (Mikolov et al., 2013) to train the word embedding matrix $\boldsymbol{E}^w$ on the English Wikipedia data from May 2014, which is similar to Vu et al. (2016). $\boldsymbol{E}^p$, other transformation matrices, and the bias terms are randomly initialized. All the parameters are tuned using the back propagation method. SGD optimization technique is used for training. Formally, we try to maximize the following loss function.

$$L(\theta) = \sum_{i=1}^{N} log(y_i) \tag{11}$$

where N is the total number of training samples. During training, each input sample is considered independently. And each parameter is updated by applying the following update rule, where η is the learning rate.

$$\theta = \theta + \eta * \partial log y_i / \partial \theta \tag{12}$$

3.7 Adversarial Training

Adversarial training (Goodfellow et al., 2014) is a method that adds some random perturbations to the training data, whose aim is to improve the robustness of a classifier. Researchers (Wu et al., 2017, Miyato et al., 2016, etc) show that when some adversarial noise is added at the level of word embedding by computing the gradient direction of a loss function w.r.t. the data, better experimental results are obtained. Following these previous methods, we also add adversarial noise at the word embedding level during training. This process is written as the following formulas.

$$\boldsymbol{w_i} = \boldsymbol{w_i} + \boldsymbol{v_i} \tag{13}$$

$$\boldsymbol{v_i} = \epsilon * g/||g|| \qquad where \quad g = \partial L/\partial w_i \tag{14}$$

where L is the loss function, and $||g||$ is the norm of gradients over all the words in a given training sentence (or the text descriptions).

4 Experimental Results and Analysis

Dataset The SemEval-2010 Task 8 dataset is used to evaluate our method. In this dataset, there are 8000 training sentences and 2717 test sentences. For each training/test sentence, two entities that are expected to be predicted a relation are labeled. In this dataset, there are 9 relations whose directions need to be considered and an extra artificial relation "*Other*" that does not need to consider the direction. Thus totally there are 19 relations in this dataset. Some statistics of this dataset are reported in Table 2. In this paper, macro-averaged *F1* score (excluding "*Other*"), the official evaluation metric, is used. And the direction of a relation is considered. In experiments, we download the needed text descriptions for all the labeled entities from *Wikipedia*[1]. All the dependency parsing trees used in the BRCNN model are generated by the Stanford Parser (Klein and Manning, 2003).

Relations	# in training dataset	# in test dataset
Cause-Effect(e1,e2)	151/5.56%	344/4.3%
Cause-Effect(e2,e1)	207/7.62%	659/8.24%
Component-Whole(e1,e2)	162/5.92%	470/5.88%
Component-Whole(e2,e1)	153/5.63%	471/5.89%
Content-Container(e1,e2)	179/6.59%	374/4.68%
Content-Container(e2,e1)	37/1.36%	166/2.08%
Entity-Destination(e1,e2)	316/11.63%	844/10.55%
Entity-Destination(e2,e1)	0/0%	1/0.01%
Entity-Origin(e1,e2)	223/8.21%	568/7.1%
Entity-Origin(e2,e1)	43/1.58%	148/1.85%
Instrument-Agency(e1,e2)	25/0.92%	97/1.21%
Instrument-Agency(e2,e1)	139/5.12%	407/5.09%
Member-Collection(e1,e2)	42/1.54%	78/0.98%
Member-Collection(e2,e1)	235/8.65%	612/7.65%
Message-Topic(e1,e2)	245/9.02%	490/6.13%
Message-Topic(e2,e1)	62/2.28%	144/1.8%
Product-Producer(e1,e2)	120/4.42%	323/4.04%
Product-Producer(e2,e1)	131/4.82%	394/4.93%
Other	247/9.09%	1410/17.63%

Table 2: Statistics for the Experimental Dataset

In experiments, we apply a cross-validation procedure on the training data to select suitable hyper-parameters. Finally, the best configurations are: the dimensions of word embeddings (d_a) and relation embeddings (d_r) are both set to 300, the dimension of position embeddings (d_b) is set to 15, learning rate η is set to 0.001, $h_{1\sim2}$ are set to 200 and 300, $dp_{1\sim2}$ are set to 0.35 and 0.3, both m (see formula 1) and w (see formula 4) are set to 3, the adversarial training parameter ϵ (see formula 14) is set to 0.01.

[1]https://www.wikipedia.org/

Effectiveness of Different Model Components In the first part of our experiments, we conduct experiments to evaluate: (1) the effectiveness of text descriptions; (2) the contributions of two proposed attention methods; and (3) the contribution of the adversarial training. To this end, we implement two classification models: one is a "*CNN-CNN-CNN*" model that uses the CNN based method for the original training/test sentence representation learning; and the other is a "*BRCNN-CNN-CNN*" model that uses the BRCNN based method for the original training/test sentence representation learning. They are denoted as *DesRC*(CNN) and *DesRC*(BRCNN) respectively. In experiments, we first implement a basic CNN model and a basic BRCNN model. Based on them, we incrementally add text descriptions, attention methods and the adversarial training. The experimental results are reported in Table 3.

Model	F1	Model	F1
CNN	83.6	BRCNN	84.5
+ *adv*	83.9	+ *adv*	84.7
+ "*intra*" *att*	84.7	+ "*intra*" *att*	84.7
+ *des*(e_1) + "*cross*" *att*	84.5	+ *des*(e_1) + "*cross*" *att*	85.1
+ *des*(e_2) + "*cross*" *att*	84.6	+ *des*(e_2) + "*cross*" *att*	84.8
+ *des*(e_1+ e_2) + "*cross*" *att*	85.3	+ *des*(e_1+ e_2) + "*cross*" *att*	85.7
+ *des*(e_1) + "*intra+cross*" *att*	84.7	+ *des*(e_1) + "*intra+cross*" *att*	85.0
+ *des*(e_2) + "*intra+cross*" *att*	84.6	+ *des*(e_2) + "*intra+cross*" *att*	84.9
+ *des*(e_1+ e_2) + "intra+cross" *att*	**85.6**	**+ *des*(e_1+e_2) + "intra+cross" *att***	**86.3**
+ "*intra*" *att* + *adv*	84.8	+ "*intra*" *att* + *adv*	85.2
+ *des*(e_1) + "*cross*" *att*+ *adv*	84.7	+ *des*(e_1) + "*cross*" *att*+ *adv*	85.0
+ *des*(e_2) + "*cross*" *att*+ *adv*	84.7	+ *des*(e_2) + "*cross*" *att*+ *adv*	86.1
+ *des*(e_1+ e_2) + "*cross*" *att*+ *adv*	85.4	+ *des*(e_1+ e_2) + "*cross*" *att*+ *adv*	86.4
+ *des*(e_1) + "*intra+cross*" *att*+ *adv*	84.9	+ *des*(e_1) + "*intra+cross*" *att*+ *adv*	85.4
+ *des*(e_2) + "*intra+cross*" *att*+ *adv*	84.7	+ *des*(e_2) + "*intra+cross*" *att*+ *adv*	85.2
+ *des*(e_1+ e_2) + "intra+cross" *att*+ *adv*	**86.1**	**+ *des*(e_1+ e_2) + "intra+cross" *att*+ *adv***	**86.7**
+ *des*(e_1+ e_2)+"intra+cross"*att+adv+WN*	**86.6**	**+ *des*(e_1+ e_2) + "intra+cross"*att+adv+WN***	**87.4**

Table 3: Performance of *DesRC*(CNN)/*DesRC*(BRCNN) with different features, WN means WordNet.

From Table 3 we can draw the following conclusions.

First, there is substantial performance improvement for both the basic CNN model and the basic BRCNN model when text descriptions are added. With the "*intra-sentence*" and "*cross-sentence*" attention methods, the F1 score for the basic CNN model increases about 2, and the F1 score for the basic BRCNN increases about 1.8. The experimental results indicate that text descriptions could provide much useful decision information for relation classification. Accordingly, the data sparsity issue is alleviated greatly when text descriptions are utilized. It is worth noting that when only part of the descriptions used (for example, only the descriptions of *e1* or *e2* are used), the F1 score still improves for both the basic CNN model and the basic BRCNN model. This is in line with our original hypothesis: as long as the text description could provide some property information for a given entity, the possible relations linked by this entity will be limited into a small candidate set. Thus, the classification decisions are more easy to be made. Accordingly, the F1 score increases.

Second, a well-designed attention method can improve the performance of relation classification greatly. For example, when using only *des(e1)* or *des(e2)* with an "*cross-sentence*" attention method, the maximal F1 improvement for the basic CNN model and the basic BRCNN model are about 1 and 0.6 respectively. We notice that the "*intra-sentence*" attention contributes much more performance improvement for the basic CNN model (F1 increases 1.1) than for the basic BRCNN model (F1 increases only 0.2). This is because that the basic BRCNN model is based on SDP whose average word number is very small (about 4, Cai et al., 2016). Thus as long as a word is in SDP, it must be much important. In other words, the role of SDP and the attention method overlaps to a certain extent. As a result, the role of the

"*intra-sentence*" attention for a basic BRCNN model is not as obvious as for a basic CNN model.

Third, the adversarial training method can further improve the performance of relation classification. When adding the adversarial training method in the basic CNN model and the basic BRCNN model, their F1 scores increase about 0.3 and 0.2 respectively. When both the description information and two attention methods are used, the F1 contributions of the adversarial training method are 0.5 and 0.4 for the basic CNN model and the basic BRCNN model respectively.

Fourth, when WordNet is added, the F1 scores for both *DesRC*(CNN) and *DesRC*(BRCNN) further increase about 0.5 and 0.7, which is a little less than the results reported in other state-of-the-art DNN based methods. In their methods, the F1 score often increases about 1 when WordNet is used. We think this is because that part of the information provided by WordNet can also be provided by text descriptions. Thus, the role of WordNet decreases when text descriptions are used. In other words, in a relation classification task, text descriptions can replace WordNet to a certain extent.

Comparisons with other State-of-the-art Relation Classification Methods In the second part of our experiments, we compare our method with several state-of-the-art DNN based relation classification methods. The comparison results are shown in Table 4. From Table 4 we can see that even without

Methods	Extra resources used	F1
CNN(Zeng et al., 2014)	WordNet	82.7
CRCNN(Dos Santos et al., 2015)	No	84.1
RNN + SDP(Xu et al., 2016)	WordNet	86.1
LSTM + SDP(Xu et al., 2015a)	WordNet	83.7
CNN + SDP(Xu et al., 2015b)	WordNet	85.6
CNN + RNN + SDP(Cai et al., 2016)	WordNet	**86.3**
CNN + SDP(Liu et al., 2015)	WordNet	83.6
CNN + RNN(Vu et al., 2016)	No	84.9
LSTM + Attention(Zhou et al., 2016)	No	84
End-to-End(Miwa et al., 2016)	WordNet	85.5
DesRC(CNN)	Text descriptions	***86.1***
DesRC(BRCNN)	Text descriptions	***86.7***
DesRC(CNN)	Text descriptions + WordNet	***86.6***
DesRC(BRCNN)	Text descriptions + WordNet	***87.4***

Table 4: Comparisons with other state-of-the-art DNN based methods.

WordNet, our method still achieves much better results than the baselines. For *DesRC*(CNN), its F1 score is close to the current-best F1 score very much. And for *DesRC*(BRCNN), it achieves the best F1 score. When WordNet is used, both *DesRC*(CNN) and *DesRC*(BRCNN) outperform the current-best method. These experimental results indicate that text descriptions are of great adaptive. They can benefit both the CNN based learning framework and the RNN based learning frameworks (BRCNN can be seen as a combination of CNN and RNN). What's more, compared with WordNet, text descriptions are more easily obtained. Thus, our method is easier to be transplanted to a new language's relation classification task.

Detailed Results In the third part of our experiments, we compare the classification performance of different relations. Here the used model is *DesRC*(BRCNN) and WordNet is used. The comparison results are reported in Table 5.

From Table 5 we can see that there are still huge performance gaps between different relations in our method. For example, the best F1 score (for example, "*cause-effect*" and "*entity-destination*") is almost 10 higher than the worst F1 score (for example, "*product-producer*" and "*content-container*"). But compared with the huge imbalance among the number of their training samples, the performance gaps are far smaller. These experimental results also show the effectiveness of text descriptions for alleviating the data sparsity issue.

Relations	P	R	F
Cause-Effect	96.43	90.25	93.24
Component-Whole	86.51	82.05	84.22
Content-Container	84.28	80.26	82.22
Entity-Destination	95.40	88.04	91.57
Entity-Origin	89.78	87.64	88.70
Instrument-Agency	87.97	81.61	84.67
Member-Collection	91.71	82.23	86.71
Message-Topic	94.05	83.28	88.34
Product-Producer	85.68	80.88	83.21

Table 5: Classification results of different relations

5 Conclusions and Future Work

In this paper, we propose a new relation classification method that uses text descriptions as a kind of supplement information. The main contributions of our method are listed as follows.

First, to the best of our knowledge, this is the first relation classification method that uses description information. In our method, two well-designed attention methods are used to combine the classification features that come from the original input sentences and their corresponding descriptions. Besides, the adversarial training method is also used to further improve the performance of our method.

Second, we conduct extensive experiments to evaluate the proposed method. Experimental results show that our method is much effective for relation classification, and it outperforms all the compared state-of-the-art baselines.

In the future, we will further explore the following two research directions. First, we will explore more kinds of descriptions that come from different websites, even to explore multi-lingual descriptions, as Lin et al. (2017) do in their work. Second, we will explore whether there are more effective learning framework that can take personalized classification strategies for different relations.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (NSFC No. 61572120, 61672138 and 61432013).

References

Nguyen Bach and Sameer Badaska. 2007. A review of relation extraction. Literature review for Language and Stastics H.

Ronan Collobert, Jason Weston, Lon Bottou, Michael Karlen, KorayKavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, pages 12:2493–2537.

Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics*, pages 626–634.

Xiaocheng Feng, Jiang Guo, Bing Qin, Ting Liu, and Yongjie Liu. 2017. Effective deep memory networks for distant supervised relation extraction. In *Proceedings of IJCAI-17*, pages 4002–4008.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Kazuma Hashimoto, Makoto Miwa, YoshimasaTsuruoka, and Takashi Chikayama. 2013. Simple customization of recursive neural networks for semantic relation classification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1372–1376.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. In *proceedings of 2003 ACL*, page arXiv: 1207.0580.

Dan Klein and Christopher D.Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of 2003 ACL*, pages 432–430.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2124–2133.

Yang Liu, Furu Wei, Sujian Li, HengJi, Ming Zhou, and Houfeng Wang. 2015. A dependency-based neural network for relation classification. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics*, pages 285–290.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the Workshop at ICLR*.

T. Miyato, A. M. Dai, and I. Goodfellow. 2016. Adversarial Training Methods for Semi-Supervised Text Classification. *ArXiv e-prints*, May.

RuiCai, Xiaodong Zhang, and Houfeng Wang. 2016. Bidirectional recurrent convolutional neural network for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 756–765.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y.Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on EMNLP and Computational Natural Language Learnin*, pages 1201–1211.

Richard Socher, Alex Perelygin, Jean Y.Wu, Christopher D.Manning Jason Chuang, Andrew Y.Ng, and Christopher Potts. 2013a. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Richard Socher, John Bauer, Christopher D.Manning, and Andrew Y.Ng. 2013b. Parsing with compositional vector grammars. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*, pages 455–465.

S.Petrov and D.Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of NAACL HLT 2007*, pages 404–411.

S.Petrov and D.Klein. 2017. Neural relation extraction with multi-lingual attention. In *Proceedings of the 55th ACL*, pages 34–43.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and HinrichSchutze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of NAACL-HLT 2016*, pages 534–539.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1298–1307.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1779–1784.

Kun Xu, Yangsong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative samplin. In *Proceedings of 2015 Conference on Empirical Methods in Natural Language Processing*, pages 536–540.

Yan Xu, LiliMou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2015. Relation classification via convolutional deep neural network. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 2335–2344.

Zhisong Zhang, Hai Zhao, and Lianhui Qin. 2016. Probabilistic graph-based dependency parsing with convolutional neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1382–1392.

Abstract Meaning Representation for Multi-Document Summarization

Kexin Liao Logan Lebanoff Fei Liu
Computer Science Department
University of Central Florida, Orlando, FL 32816, USA
{ericaryo,loganlebanoff}@knights.ucf.edu feiliu@cs.ucf.edu

Abstract

Generating an abstract from a collection of documents is a desirable capability for many real-world applications. However, abstractive approaches to multi-document summarization have not been thoroughly investigated. This paper studies the feasibility of using Abstract Meaning Representation (AMR), a semantic representation of natural language grounded in linguistic theory, as a form of content representation. Our approach condenses source documents to a set of summary graphs following the AMR formalism. The summary graphs are then transformed to a set of summary sentences in a surface realization step. The framework is fully data-driven and flexible. Each component can be optimized independently using small-scale, in-domain training data. We perform experiments on benchmark summarization datasets and report promising results. We also describe opportunities and challenges for advancing this line of research.

1 Introduction

Abstractive summarization seeks to generate concise and grammatical summaries that preserve the meaning of the original; further, they shall abstract away from the source syntactic forms. The task often involves high-level text transformations such as sentence fusion, generalization, and paraphrasing (Jing and McKeown, 1999). Recent neural abstractive summarization studies focus primarily on single-document summarization (Paulus et al., 2017; See et al., 2017). These approaches are limited by the availability of training data, and large datasets for multi-document summarization can be costly to obtain. Generating abstractive summaries for sets of source documents thus remains a challenging task.

Traditional approaches to abstractive summarization often condense the source documents to a set of "semantic units," then reconstruct abstractive summaries from these semantic units. Previous work has investigated various forms of content representation. Examples include noun/verb phrases (Genest and Lapalme, 2011; Bing et al., 2015), word-occurrence graphs (Ganesan et al., 2010), syntactic parse trees (Cheung and Penn, 2014; Gerani et al., 2014), and domain-specific templates (Pighin et al., 2014). Nonetheless, generating summary text from these heuristic forms of representation can be difficult. There is an increasing need to exploit a semantic formalism so that condensing source documents to this form and generating summary sentences from it can both be carried out in a principled way.

This paper explores Abstract Meaning Representation (AMR, Banarescu et al., 2013) as a form of content representation. AMR is a semantic formalism based on propositional logic and the neo-Davidsonian event representation (Parsons, 1990; Schein, 1993). It represents the meaning of a sentence using a rooted, directed, and acyclic graph, where nodes are concepts and edges are semantic relations. Figure 1 shows an example AMR graph. A concept node can be a PropBank frameset ("state-01"), an English word ("warhead"), a special keyword ("date-entity"), or a string literal ("Japan"). A relation can be either a core argument ("ARG0," "ARG1") or a modification relationship ("mod," "time"). The AMR representation abstracts away from surface word strings and syntactic structure, producing a language-neutral representation of meaning. The graph representation is flexible and not specifically designed for a particular domain. It is thus conceptually appealing to explore AMR for abstractive summarization.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1178–1190
Santa Fe, New Mexico, USA, August 20-26, 2018.

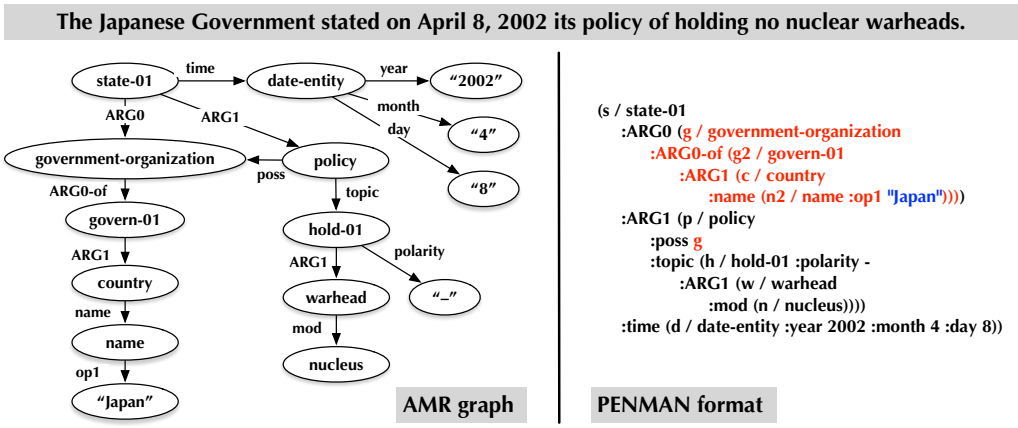


Figure 1: A example sentence, its goldstandard AMR graph, and the corresponding PENMAN format.

Our goal in this work is to generate a text abstract containing multiple sentences from a cluster of news articles discussing a given topic. The system framework includes three major components: *source sentence selection* takes a set of news articles as input and selects sets of similar sentences covering different aspects of the topic; *content planning* consumes a set of similar sentences and derives a summary graph from them; *surface realization* transforms a summary graph to a natural language sentence. This framework allows each component (source sentence selection, content planning, surface realization) to be individually optimized using small-scale, in-domain training data, reducing the need for large-scale parallel training data. Our research contributions are summarized as follows:

- we investigate AMR, a linguistically-grounded semantic formalism, as a new form of content representation for multi-document summarization. Liu et al. (2015) conducted a pilot study using AMR for single-document summarization. This paper exploits the structured prediction framework but presents a full pipeline for generating abstractive summaries from multiple source documents;
- we study to what extent the AMR parser and generator, used for mapping text to and from AMR, can impact the summarization performance. We also compare multiple source sentence selection strategies to group source sentences into clusters covering various aspects of the topic;
- we conduct extensive experiments on benchmark summarization datasets, and contrast our work with state-of-the-art baselines, including the pointer-generator networks (See et al., 2017). Results show that leveraging the AMR representation for summarization is promising. Our framework is flexible, allowing different components to be optimized independently using small-scale, in-domain datasets. We finally describe opportunities and challenges for advancing this line of research.

2 Related Work

Neural abstractive summarization has sparked great interest in recent years. These approaches focus primarily on short text summarization and single-document summarization (Rush et al., 2015; Nallapati et al., 2016). Variants of the neural encoder-decoder architecture have been exploited to reduce out-of-vocabulary tokens and word repetitions (See et al., 2017; Suzuki and Nagata, 2017), improve the attention mechanism (Chen et al., 2016; Zhou et al., 2017; Tan et al., 2017), control the summary length (Kikuchi et al., 2016), improve the learning objective and search (Ranzato et al., 2016; Huang et al., 2017), and generate summaries that are true to the original inputs (Cao et al., 2018; Song et al., 2018). Training neural models requires large amounts of data; they are often acquired by pairing news articles with titles or human-written highlights. Nonetheless, obtaining parallel data for multi-document summarization is often costly. There is thus a need to investigate alternative approaches that are less data-thirsty.

Abstractive summarization via natural language generation (NLG, Reiter and Dale, 2000; Gatt and Krahmer, 2018) is a promising line of work. The approaches often identify salient text units from source documents, arrange them in a compact form, such as domain-specific templates, and subsequently synthesize them into natural language texts (Barzilay et al., 1999; Genest and Lapalme, 2011; Oya et al., 2014; Gerani et al., 2014; Fabbrizio et al., 2014). A challenge faced by these approaches is that there

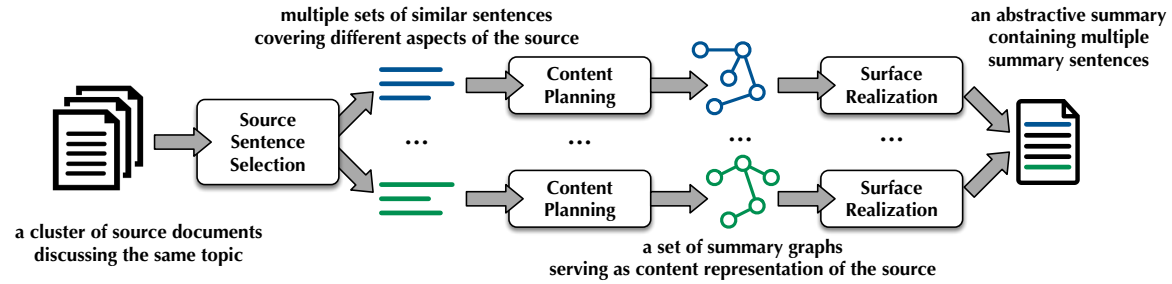


Figure 2: Our system framework, consisting of three major components.

lacks a principled means of content representation. This paper studies the feasibility of using AMR, a semantic formalism grounded in linguistic theory, for content representation. Within this framework, condensing source documents to summary AMR graphs and generating natural language sentences from summary graphs are both data-driven and not specifically designed for any domain.

The AMR formalism has demonstrated great potential on a number of downstream applications, including machine translation (Tamchyna et al., 2015), entity linking (Pan et al., 2015), summarization (Liu et al., 2015; Takase et al., 2016), question answering (Jurczyk and Choi, 2015), and machine comprehension (Sachan and Xing, 2016). Moreover, significant research efforts are dedicated to map English sentences to AMR graphs (Flanigan et al., 2014; Wang et al., 2015a; Wang et al., 2015b; Ballesteros and Al-Onaizan, 2017; Buys and Blunsom, 2017; Damonte et al., 2017; Szubert et al., 2018), and generating sentences from AMR (Flanigan et al., 2016; Song et al., 2016; Pourdamghani et al., 2016; Song et al., 2017; Konstas et al., 2017). These studies pave the way for further research exploiting AMR for multi-document summarization.

3 Our Approach

We describe our major system components in this section. In particular, content planning (§3.1) takes as input a set of similar sentences. It maps each sentence to an AMR graph, merges all AMR graphs to a connected *source graph*, then extracts a *summary graph* from the source graph via structured prediction. Surface realization (§3.2) converts a summary graph to its PENMAN representation (Banarescu et al., 2013) and generates a natural language sentence from it. Source sentence extraction (§3.3) selects sets of similar sentences from source documents discussing different aspects of the topic. The three components form a pipeline to generate an abstractive summary from a collection of documents. Figure 2 illustrates the system framework. In the following sections we describe details of the components.

3.1 Content Planning

The meaning of a source sentence is represented by a rooted, directed, and acyclic AMR graph (Banarescu et al., 2013), where nodes are concepts and edges are semantic relations. A sentence AMR graph can be obtained by applying an AMR parser to a natural language sentence. In this work we investigate two AMR parsers to understand to what extent the performance of AMR parsing may impact the summarization task. JAMR (Flanigan et al., 2014) presents the first open-source AMR parser. It introduces a two-part algorithm that first identifies concepts from the sentence and then determines the relations between them by searching for the maximum spanning connected subgraph (MSCG) from a complete graph representing all possible relations between the identified concepts. CAMR (Wang et al., 2015b) approaches AMR parsing from a different perspective. It describes a transition-based AMR parsing algorithm that transforms from a dependency parse tree to an AMR graph. We choose JAMR and CAMR because these parsers have been made open-source and both of them reported encouraging results in the recent SemEval evaluations (May, 2016; May and Priyadarshi, 2017).

Source Graph Construction. Given a set of source sentences and their AMR graphs, source graph construction attempts to consolidate all sentence AMR graphs to a connected *source graph*. This is accomplished by performing *concept merging*. Graph nodes representing the same concept, determined by the surface word form, are merged to a single node in the source graph. Importantly, we perform

coreference resolution on the source documents to identify clusters of mentions of the same entity or event. Graph nodes representing these mentions are also merged. A special treatment to date entity (see "date-entity" in Figure 1) and named entity ("country") includes collapsing the subtree to a "mega-node" whose surface form is the concatenation of the consisting concepts and relations (e.g., "date-entity_:year_2002_:month_1_:day_5"). These mega-nodes can then only be merged with other identical fragments. Finally, a 'ROOT' node is introduced; it is connected to the root of each sentence AMR graph, yielding a connected source graph.

Summary Graph Extraction. We hypothesize that a summary graph, containing the salient content of source texts, can be identified from the source graph via a trainable, feature-rich structured prediction framework. The framework iteratively performs *graph decoding* and *parameter update*. The former identifies an optimal summary graph using integer linear programming, while the latter performs parameter update by minimizing a loss function that measures the difference between the system-decoded summary graph and the goldstandard summary graph.

We use $G = (V, E)$ to represent the source graph. Let v_i and $e_{i,j}$ be a set of binary variables where $v_i = 1$ (or $e_{i,j} = 1$) indicates the corresponding source graph node (or edge) is selected to be included in the summary graph. The node and edge saliency are characterized by a set of features, represented using $\mathbf{f}(i)$ and $\mathbf{g}(i, j)$, respectively. $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are the feature weights. Eq. (1) presents a scoring function for any graph G. It can be factorized into a sum of scores for selected nodes and edges. In particular, $[\boldsymbol{\theta}^\top \mathbf{f}(i)]_{v_i=1}$ denotes the node score (if v_i is selected) and $[\boldsymbol{\phi}^\top \mathbf{g}(i, j)]_{e_{i,j}=1}$ denotes the edge score.

$$score(G; \boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{i=1}^{N} v_i \underbrace{[\boldsymbol{\theta}^\top \mathbf{f}(i)]_{v_i}}_{\text{node score}} + \sum_{(i,j)\in E} e_{i,j} \underbrace{[\boldsymbol{\phi}^\top \mathbf{g}(i, j)]_{e_{i,j}}}_{\text{edge score}} \quad (1)$$

Features characterizing the graph nodes and edges are adopted from (Liu et al., 2015). They include concept/relation labels and their frequencies in the documents, average depth of the concept in sentence AMR graphs, position of sentences containing the concept/relation, whether the concept is a named entity/date entity, and the average length of concept word spans. We additionally include the concept TF-IDF score and if the concept occurs in a major news event (Wiki, 2018). All features are binarized.

The *graph decoding* process searches for the summary graph that maximizes the scoring function: $\hat{G} = \arg\max_G score(G)$. We can formulate graph decoding as an integer linear programming problem. Each summary graph corresponds to a set of values assigned to the binary variables v_i and $e_{i,j}$. We implement a set of linear constraints to ensure the decoded graph is connected, forms a tree structure, and limits to L graph nodes (Liu et al., 2015).

The *parameter update* process adjusts $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ to minimize a loss function capturing the difference between the system-decoded summary graph ($\hat{G}$) and the goldstandard summary graph (G^*). Eq. (2) presents the structured perceptron loss. Minimizing this loss function with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ is straightforward. However, the structured perceptron loss has problems when the goldstandard summary graph G^* is unreachable via the graph decoding process. In that case, there remains a gap between $score(\hat{G})$ and $score(G^*)$ and the loss cannot be further minimized. The structured ramp loss (Eq. (3)) addresses this problem by performing cost-augmented decoding. It introduces a cost function $cost(G; G^*)$ that can also be factored over graph nodes and edges. A cost of 1 is incurred if the system graph and the goldstandard graph disagree on whether a node (or edge) should be included. As a result, the first component of the loss function $\max_G(score(G) + cost(G; G^*))$ yields a decoded system graph that is slightly *worse* than $\hat{G} = \arg\max_G score(G)$; and the second component $\max_G(score(G) - cost(G; G^*))$ yields a graph that is slightly *better* than $\hat{G}$. The scoring difference between the two system graphs becomes the structured ramp loss we wish to minimize. This formulation is similar to the objective of the structured support vector machines (SSVMs, Tsochantaridis et al., 2005). Becase decent results have been reported by (Liu et al., 2015), we adopt structured ramp loss in all experiments.

$$\mathcal{L}_{\text{perc}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \max_G score(G) - score(G^*) = score(\hat{G}) - score(G^*) \quad (2)$$

$$\mathcal{L}_{\text{ramp}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \max_G(score(G) + cost(G; G^*)) - \max_G(score(G) - cost(G; G^*)) \quad (3)$$

3.2 Surface Realization

The surface realization component converts each summary graph to a natural language sentence. This is a nontrivial task, because AMR abstracts away from the source syntactic forms and an AMR graph may correspond to a number of valid sentence realizations. In this section we perform two subtasks that have not been investigated in previous studies. We first convert the summary AMR graphs to the PENMAN format (Figure 1), which is a representation that can be understood by humans. It is also the required input form for an AMR-to-text generator. We then leverage the AMR-to-text generator to generate English sentences from summary graphs.

Algorithm 1 presents our algorithm for transforming a summary graph to the PENMAN format. Because the summary graph is rooted and acyclic, we can extract all paths from the ROOT node to all leaves. These paths are sorted by the concept indices on the path, and the paths sharing the same ancestors will be processed in order. The core of Algorithm 1 is the *while* loop (line 9–35); it writes out one concept per line. An AMR concept can have three forms: a regular form ("c / country"), string literal ("Japan"), or a re-entrance ("g"). The last two are treated as "special forms." In these cases, a relation is first written out, followed by the special concept ($\mathcal{R}_{m',n'}$::$\mathcal{C}_{n'}$). "::" denotes a whitespace. A regular concept will be wrapped by a left bracket ($\mathcal{R}_{m',n'}$::"("::$\mathcal{C}_{n'}$, line 23/25) and a right bracket (($k-k'$)*")"::EOS, line 41). Finally, a proper number of closing brackets is postpended to each path (line 37–42).

To transform the PENMAN string to a summary sentence we employ the JAMR AMR-to-text generator (Flanigan et al., 2016). JAMR is the first full-fledged AMR-to-text generator. It is trained on approximately 10K sentences and achieves about 22% BLEU score on the test set. The system first transforms the input graph to a spanning tree, and then decodes it into a string using a tree-to-string transducer and a language model. The final output sentence is the highest-scoring sentence according to a feature-rich discriminatively trained linear model. We choose the JAMR AMR-to-text generator because of its competitive performance in the recent SemiEval evaluations (May and Priyadarshi, 2017).

Algorithm 1 An algorithm for transforming a summary graph to the PENMAN format.

Input: Triples: (src_concept, relation, tgt_concept).

1: $r \leftarrow$ index of the ROOT concept
2: $\mathcal{P} \leftarrow$ all paths from ROOT to leaves, sorted by the concept indices on the paths
3: ▶ Set all concepts and relations as unvisited
4: $visited[\mathcal{C}_m] \leftarrow$ FALSE, $\forall m$
5: $visited[\mathcal{R}_{m,n}] \leftarrow$ FALSE, $\forall m, n$
6: **for** $i = 1, \ldots, |\mathcal{P}|$ **do**
7: $flag\_special\_concept \leftarrow$ FALSE.
8: $k \leftarrow 0$
9: **while** $k < |p_i|$ **do**
10: $k \leftarrow k + 1$
11: $n' \leftarrow$ index of the k-th concept on the path
12: $m' \leftarrow$ index of the previous concept
13: **if** $visited[\mathcal{C}_{n'}]$ = FALSE **then**
14: ▶ Concept unvisited
15: $visited[\mathcal{C}_{n'}] \leftarrow$ TRUE
16: $visited[\mathcal{R}_{m',n'}] \leftarrow$ TRUE
17: $output$ += $(k-1)$*TAB
18: **if** $\mathcal{C}_{n'}$ is a string literal **then**
19: $output$ += $\mathcal{R}_{m',n'}$::$\mathcal{C}_{n'}$
20: $flag\_special\_concept \leftarrow$ TRUE
21: BREAK
22: **else if** $k < |p_i|$ **then**
23: $output$ += $\mathcal{R}_{m',n'}$::"("::$\mathcal{C}_{n'}$::EOS
24: **else**
25: $output$ += $\mathcal{R}_{m',n'}$::"("::$\mathcal{C}_{n'}$
26: **end if**
27: **else if** $visited[\mathcal{R}_{m',n'}]$ = FALSE **then**
28: ▶ Concept reentrance
29: $visited[\mathcal{R}_{m',n'}] \leftarrow$ TRUE
30: $output$ += $(k-1)$*TAB
31: $output$ += $\mathcal{R}_{m',n'}$::$\mathcal{C}_{n'}$
32: $flag\_special\_concept \leftarrow$ TRUE
33: BREAK
34: **end if**
35: **end while**
36: ▶ Output path ending brackets and EOS.
37: $k' \leftarrow$ tracing the path backwards to find position of the closest ancestor who has an unvisited child; if none exists, $k' \leftarrow 0$
38: **if** $flag\_special\_concept$ = TRUE **then**
39: $output$ += $(k-k'-1)$*")"::EOS
40: **else**
41: $output$ += $(k-k')$*")"::EOS
42: **end if**
43: **end for**

3.3 Source Sentence Selection

We seek to generate an abstractive summary containing multiple sentences from a cluster of documents discussing a single topic (e.g., health and safety). Each summary sentence will cover a topic aspect; it is generated by fusing a set of relevant source sentences. We thus perform clustering on all source sentences to find salient topic aspects and their corresponding sets of similar sentences. Spectral clustering has been shown to perform strongly on different clustering problems (Ng et al., 2002; Yogatama and Tanaka-Ishii, 2009). The approach constructs an affinity matrix by applying a pairwise similarity function to all source sentences. It then calculates the eigenvalues of the matrix and performs clustering in the low-dimensional space spanned by the largest eigenvectors. A large cluster indicates a salient topic aspect. We focus on the M largest clusters and extract N sentences from each cluster.[1] These sentences have the highest similarity scores with other sentences in the cluster. The selected sets of relevant sentences are later fed to the content planning component to generate summary AMR graphs.

Training the content planning component, however, requires sets of source sentences paired with their goldstandard summary graphs. Manually selecting sets of sentences and annotating summary graphs is costly and time-consuming. Instead, we leverage human reference summaries to create training instances. We obtain summary graphs by AMR-parsing sentences of human reference summaries. For every reference sentence, we further extract a set of source sentences. They are judged similar to the reference sentence via a similarity metric. The summary AMR graphs and sets of source sentences thus form the training data for content planning. We gauge how best to select source sentences by exploring different similarity metrics. In particular, (i) **LCS** calculates the longest common subsequence between a candidate source sentence and the reference sentence; (ii) **VSM** represents sentences using the vector space model and calculates the cosine similarity between the two sentence vectors; (iii) **Smatch** (Cai and Knight, 2013) calculates the F-score of AMR concepts between the candidate and reference sentences; (iv) **Concept Coverage** selects source sentences to maximize the coverage of AMR concepts of the reference sentence. We experiment with these source sentence selection strategies and compare their effectiveness in Section §5.1.

4 Datasets and Baselines

We perform experiments on standard multi-document summarization datasets[2], prepared by the NIST researchers for DUC/TAC competitions and later exploited by various summarization studies (Nenkova and McKeown, 2011; Hong et al., 2014; Yogatama et al., 2015). A summarization instance includes generating a text summary containing 100 words or less from a cluster of 10 source documents discussing a single topic. 4 human reference summaries are provided for each cluster of documents; they are created by NIST assessors. We use the datasets from DUC-03, DUC-04, TAC-09, TAC-10, and TAC-11 in this study, containing 30/50/44/46/44 clusters of documents respectively.

We compare our AMR summarization framework with a number of extractive (*ext*-$*$) and abstractive (*abs*-$*$) summarization systems, including the most recent neural encoder-decoder architecture (See et al., 2017). They are described as follows.

- *ext*-**LexRank** (Erkan and Radev, 2004) is a graph-based approach that computes sentence importance based on the concept of eigenvector centrality in a graph representation of source sentences;
- *ext*-**SumBasic** (Vanderwende et al., 2007) is an extractive approach that assumes words occurring frequently in a document cluster have a higher chance of being included in the summary;
- *ext*-**KL-Sum** (Haghighi and Vanderwende, 2009) describes a method that greedily adds sentences to the summary so long as it decreases the KL divergence;
- *abs*-**Opinosis** (Ganesan et al., 2010) generates abstractive summaries by searching for salient paths on a word co-occurrence graph created from source documents;

[1] We use N=M=5 in our experiments. This setting fuses 5 source sentences to a summary sentence. It then produces 5 summary sentences for each topic, corresponding to the average number of sentences in human summaries.

[2] https://duc.nist.gov/data.html https://tac.nist.gov/data/index.html

Approach	Nodes			(Oracle) Nodes			Edges			(Oracle) Edges		
	P	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
LCS	16.7	26.7	19.9	31.5	49.5	37.6	6.7	8.0	6.9	16.1	18.7	16.6
Smatch	20.9	33.2	24.9	33.2	52.0	39.6	9.3	10.7	9.4	17.2	20.1	17.8
Concept Cov.	**25.0**	**40.3**	**30.1**	**48.8**	**77.5**	**58.7**	7.3	10.0	8.0	18.9	**25.3**	20.8
VSM	24.0	38.6	28.8	40.8	64.3	48.9	**9.6**	**11.3**	**9.8**	**21.1**	25.1	**22.1**

Table 1: Summary graph prediction results on the DUC-04 dataset. The scores measure how well the predicted summary graphs match reference summary graphs on nodes and edges. Reference summary graphs are created by parsing reference summary sentences using the CAMR parser. "Oracle" results are obtained by performing only cost-based decoding. They establish an upper bound for the respective approaches.

- *abs*-**Pointer-Generator** (See et al., 2017) describes a neural encoder-decoder architecture. It encourages the system to copy words from the source text via pointing, while retaining the ability to produce novel words through the generator. It also includes a coverage mechanism to keep track of what has been summarized, thus reducing word repetition. The pointer-generator networks have not been tested for multi-document summarization. In this study we evaluate their performance on the DUC/TAC datasets.

5 Experimental Results

In this section we evaluate our AMR summarization framework. We are interested in knowing how well the system performs on predicting summary graphs from sets of relevant source sentences (§5.1). We also investigate the system's performance on generating abstractive summaries and its comparison with various baselines (§5.2). Finally, we provide an analysis on system summaries and outline challenges and opportunities for advancing this line of work (§5.3).

5.1 Results on Summary Graph Prediction

Graph prediction results on the DUC-04 dataset (trained on DUC-03) are presented in Table 1. We report how well the decoded summary graphs match goldstandard summary graphs on nodes (concepts) and edges (relations). We compare several strategies to select sets of source sentences. The goldstandard summary graphs are created by parsing the reference summary sentences via the CAMR parser (Wang et al., 2015b). Note that we cannot obtain goldstandard summary graphs for sets of source sentences selected by spectral clustering. This approach therefore is not evaluated for graph prediction. The system-decoded summary graphs are limited to 15 graph nodes, corresponding to the average number of words in reference summary sentences (stopwords excluded). We additionally report "Oracle" decoding results, obtained by performing only cost-based decoding $\hat{G} = \arg\max_G(-cost(G; G^*))$ on the source graph, where G^* is the goldstandard summary graph. The oracle results establish an upper bound for the respective approaches.

We observe that node prediction generates better results than edge prediction. Using 'Concept Cov,' the system-decoded summary graphs successfully preserve 40.3% of the goldstandard summary concepts, and this number increases to 77.5% when using oracle decoding, indicating the content planning component is effective at identifying important source concepts and preserving them in summary graphs. 'VSM' performs best on edge prediction. It achieves an F-score of 9.8% and the oracle decoding further boosts the performance to 22.1%. We observe that only 42% of goldstandard summary bigrams appear in the source documents, serving as a cap for edge prediction. The results suggest that 'VSM' is effective at selecting sets of source sentences containing salient source relations. The high performance on summary node prediction but low on edge prediction suggests that future work may consider increasing the source graph connectivity by introducing edges between concepts so that salient summary edges can be effectively preserved.

5.2 Results on Summarization

In Table 2 we report the summarization results evaluated by ROUGE (Lin, 2004). In particular, R-1, R-2, and R-SU4 respectively measure the overlap of unigrams, bigrams, and skip bigrams (up to 4 words)

	System	ROUGE-1			ROUGE-2			ROUGE-SU4		
		P	R	F	P	R	F	P	R	F
DUC 2004	*ext*-SumBasic	37.5	24.9	29.5	5.3	3.6	4.3	11.1	7.3	8.6
	ext-KL-Sum	31.1	31.1	31.0	6.0	6.1	6.0	10.2	10.3	10.2
	ext-LexRank	34.3	34.6	34.4	7.1	7.2	7.1	11.1	11.2	11.2
	abs-Opinosis	36.5	23.7	27.5	7.2	4.3	5.1	11.7	7.4	8.6
	abs-Pointer-Gen-all	37.5	20.9	26.5	8.0	4.4	**5.6**	12.3	6.7	8.5
	abs-Pointer-Gen	33.2	21.5	25.6	5.8	3.8	4.5	10.3	6.6	7.9
	abs-AMRSumm-Clst	29.9	30.5	**30.2**	4.1	4.2	4.1	8.7	8.9	**8.8**
	abs-AMRSumm-VSM	36.7	39.0	**37.8**	6.5	6.9	**6.6**	11.4	12.2	**11.8**
TAC 2011	*ext*-SumBasic	37.3	28.2	31.6	6.9	5.5	6.1	11.8	9.0	10.1
	ext-KL-Sum	31.2	31.4	31.2	7.1	7.1	7.1	10.5	10.6	10.6
	ext-LexRank	32.9	33.3	33.1	7.4	7.6	7.5	11.1	11.2	11.1
	abs-Opinosis	38.0	20.4	25.2	8.6	4.0	5.1	12.9	6.5	8.1
	abs-Pointer-Gen-all	37.3	22.2	27.6	7.8	4.6	**5.8**	12.2	7.1	8.9
	abs-Pointer-Gen	34.4	21.6	26.2	6.9	4.4	5.3	10.9	6.8	8.2
	abs-AMRSumm-Clst	32.2	31.7	**31.9**	4.7	4.7	4.7	9.8	9.7	**9.7**
	abs-AMRSumm-VSM	40.1	42.3	**41.1**	8.1	**8.5**	**8.3**	13.1	13.9	**13.5**

Table 2: Summarization results on DUC-04 and TAC-11 datasets. We compare the AMR summarization framework (AMRSumm-*) with both extractive (*ext*-*) and abstractive (*abs*-*) summarization systems.

between system and reference summaries. Our AMR summarization framework outperforms all other abstractive systems with respect to R-1 and R-SU4 F-scores on both DUC-04 (trained on DUC-03) and TAC-11 (trained on TAC-09,10) datasets. "AMRSumm-VSM" further produces the highest R-2 F-scores. We conjecture that the R-2 scores of AMRSumm-* are related to the performance of the AMR-to-text generator (Flanigan et al., 2016). When it transforms a summary graph to text, there can be multiple acceptable realizations (e.g., "I hit my hand on the table" or "My hand hit the table") and the one chosen by the AMR generator may not always be the same as the source text. Because abstractive systems are expected to produce novel words, they may yield slightly inferior results to the best extractive system (LexRank). Similar findings are also reported by Nallapati et al. (2017) and See et al. (See et al., 2017).

We experiment with two variants of the pointer-generator networks: "Pointer-Generator-all" uses all source sentences of the document set and "Pointer-Generator" uses the source sentences selected by spectral clustering, hence the same input as "AMRSumm-Clst." We observe that "AMRSumm-Clst" performs stronger than "Pointer-Generator" at preserving salient summary concepts, yielding R-1 F-scores of 30.2% vs. 25.6% and 31.9% vs. 26.2% on DUC-04 and TAC-11 datasets. Further, we found that the summaries produced by the pointer-generator networks are more extractive than abstractive. We report the percentages of summary n-grams contained in the source documents in Figure 3. "Pointer-Generator-all" has 99.6% of unigrams, 95.2% bigrams, and 87.2% trigrams contained in the source documents (DUC-04). In contrast, the ratios for human summaries are 85.2%, 41.6% and 17.1%, and for "AMRSumm-Clst" the ratios are 84.6%, 31.3% and 8.4% respectively. Both human summaries and "AMRSumm-Clst" summaries tend to be more abstractive, with fewer bigrams/trigrams appeared in the source. These results suggest that future abstractive systems for multi-document summarization may need to carefully balance between copying words from the source text with producing new words/phrases in order to generate summaries that resemble human abstracts.

5.3 Result Analysis

Table 3 shows results of the AMR-Summ framework with different system configurations. We use CAMR (Wang et al., 2015b) to parse source sentences into AMR graphs during training, and apply either JAMR (Flanigan et al., 2014) or CAMR to parse sentences at test time. We observe that the quality of AMR parsers has an impact on summarization performance. In particular, JAMR reports 58% F-score and CAMR reports 63% F-score for parsing sentences. The inter-annotator agreement places

	Approach	JAMR P	JAMR R	JAMR F	CAMR P	CAMR R	CAMR F	(Oracle) CAMR P	(Oracle) CAMR R	(Oracle) CAMR F
ROUGE-1	AMRSumm-Clst	29.0	29.8	29.4	29.9	30.5	30.2	36.4	37.8	37.1
	AMRSumm-Concept Cov	36.3	37.8	**36.9**	36.9	39.3	**38.1**	46.9	49.8	**48.3**
	AMRSumm-VSM	35.9	37.2	36.5	36.7	39.0	37.8	43.3	46.1	44.6
ROUGE-2	AMRSumm-Clst	3.2	3.3	3.2	4.1	4.2	4.1	6.0	6.3	6.1
	AMRSumm-Concept Cov	4.8	5.0	**4.9**	5.7	6.0	5.8	9.8	10.4	**10.1**
	AMRSumm-VSM	4.6	4.8	4.7	6.5	6.9	**6.6**	9.7	10.3	10.0

Table 3: Summarization results of the AMR-Summ framework on the DUC-04 dataset. "JAMR" uses the JAMR parser (Flanigan et al., 2014) to produce AMR graphs for source sentences, while "CAMR" uses the CAMR parser (Wang et al., 2015b). "Oracle" results are obtained by performing only cost-based decoding.

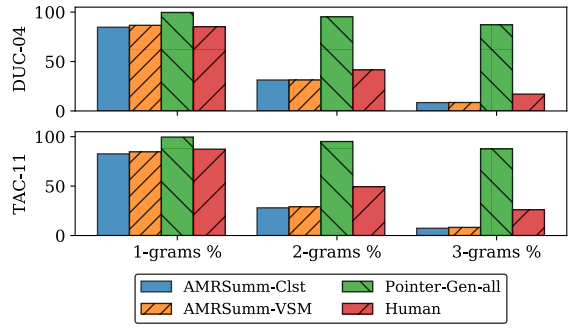


Figure 3: Percentages of summary n-grams contained in the source documents. Both human summaries and AMRSumm summaries are highly abstractive, with few bigrams and trigrams contained in the source. Pointer-Generator summaries appear be more extractive than abstractive.

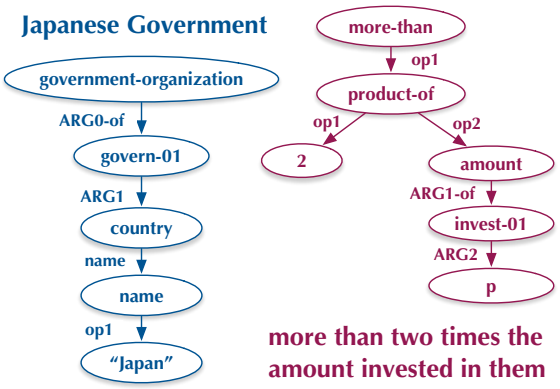


Figure 4: Example AMR for text segments.

an upper bound of 83% F-score on expected parser performance (Wang et al., 2015b). There remains a significant gap between the current parser performance and the best it can achieve. Consequently, we notice that there is a gap of 1∼2 points in terms of ROUGE scores when comparing summaries produced using the two parsers. We notice that source sentence selection strategies making use of reference summary sentences produces better results than 'AMRSumm-Clst.' Using oracle decoding further boosts the summarization performance by 7-10% for R-1 F-score and 2-5% for R-2 F-score.

When examining the source and summary graphs, we notice that a simplified AMR representation could be helpful to summarization. As a meaning representation grounded in linguistic theory, AMR strives to be comprehensive and accurate. However, a summarization system may benefit from a reduced graph representation to increase the algorithm robustness. For example, the large 'semantic content units' could be collapsed to "mega-nodes" in some cases. Figure 4 shows two examples. In the first example ("Japanese Government"), the human annotator chooses to decompose derivational morphology given that a relative clause paraphrase is possible (Schneider et al., 2015). It produces 5 concept nodes, representing "government organization that governs the country of Japan." In the second example, "more than two times the amount invested in them" also has fine-grained annotation. For the purpose of summarization, these graph fragments could potentially be collapsed to "mega-nodes" and future AMR parsers may consider working on reduced AMR graphs.

6 Conclusion

In this paper we investigated the feasibility of utilizing the AMR formalism for multi-document summarization. We described a full-fledged approach for generating abstractive summaries from multiple source documents. We further conducted experiments on benchmark summarization datasets and showed that the AMR summarization framework performs competitively with state-of-the-art abstractive approaches. Our findings suggest that the abstract meaning representation is a powerful semantic formalism that holds potential for the task of abstractive summarization.

Acknowledgements

We are grateful to the anonymous reviewers for their insightful comments. The authors thank Chuan Wang, Jeffrey Flanigan, and Nathan Schneider for useful discussions.

References

Miguel Ballesteros and Yaser Al-Onaizan. 2017. Amr parsing using stack-LSTMs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of Linguistic Annotation Workshop*.

Regina Barzilay, Kathleen R. McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca J. Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. In *Proceedings of ACL*.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for document summarization. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*.

Jackie Chi Kit Cheung and Gerald Penn. 2014. Unsupervised sentence enhancement for automatic summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of EACL*.

Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*.

Giuseppe Di Fabbrizio, Amanda J. Stent, and Robert Gaizauskas. 2014. A hybrid approach to multi-document summarization of opinions in reviews. *Proceedings of the 8th International Natural Language Generation Conference (INLG)*.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from abstract meaning representation using tree transducers. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: A graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Pierre-Etienne Genest and Guy Lapalme. 2011. Framework for abstractive summarization using text-to-text generation. In *Proceedings of ACL Workshop on Monolingual Text-To-Text Generation*.

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T. Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Kai Hong, John M Conroy, Benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A repository of state of the art and competitive baseline summaries for generic news summarization. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*.

Liang Huang, Kai Zhao, and Mingbo Ma. 2017. When to finish? Optimal beam search for neural text generation (modulo beam size). In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Hongyan Jing and Kathleen McKeown. 1999. The decomposition of human-written summary sentences. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

Tomasz Jurczyk and Jinho D. Choi. 2015. Semantics-based graph approach to complex question-answering. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of EMNLP*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Chin-Yew Lin. 2004. ROUGE: a package for automatic evaluation of summaries. In *Proceedings of ACL Workshop on Text Summarization Branches Out*.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Jonathan May and Jay Priyadarshi. 2017. SemEval-2017 task 9: Abstract meaning representation parsing and generation. In *Proceedings of SemEval*.

Jonathan May. 2016. SemEval-2016 task 8: Meaning representation parsing. In *Proceedings of SemEval*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2002. On spectral clustering: Analysis and an algorithm. In *Proceedings of the 14th International Conference on Neural Information Processing Systems (NIPS)*.

Tatsuro Oya, Yashar Mehdad, Giuseppe Carenini, and Raymond Ng. 2014. A template-based abstractive meeting summarization: Leveraging summary and source text relationships. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*.

Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised entity linking with abstract meaning representation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Terence Parsons. 1990. *Events in the semantics of English*. MIT Press.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Daniele Pighin, Marco Cornolti, Enrique Alfonseca, and Katja Filippova. 2014. Modelling events through memory-based, open-ie patterns for abstractive summarization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nima Pourdamghani, Kevin Knight, and Ulf Hermjakob. 2016. Generating english from abstract meaning representations. In *Proceedings of the 9th International Natural Language Generation conference (INLG)*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press New York, NY, USA.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for sentence summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mrinmaya Sachan and Eric P. Xing. 2016. Machine comprehension using rich semantic representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Barry Schein. 1993. *Plurals and Events*. MIT Press.

Nathan Schneider, Jeffrey Flanigan, and Tim O'Gorman. 2015. The logic of amr: Practical, unified, graph-based sentence semantics for nlp. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorial Abstracts (NAACL)*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Linfeng Song, Yue Zhang, Xiaochang Peng, Zhiguo Wang, and Daniel Gildea. 2016. Amr-to-text generation as a traveling salesman problem. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Linfeng Song, Xiaochang Peng, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2017. Amr-to-text generation with synchronous node replacement grammar. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kaiqiang Song, Lin Zhao, and Fei Liu. 2018. Structure-infused copy mechanisms for abstractive summarization. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Jun Suzuki and Masaaki Nagata. 2017. Cutting-off redundant repeating generations for neural abstractive summarization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Ida Szubert, Adam Lopez, and Nathan Schneider. 2018. A structured syntax-semantics interface for english-amr alignment. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ales Tamchyna, Chris Quirk, and Michel Galley. 2015. A discriminative model for semantics-to-string translation. In *Proceedings of the ACL Workshop on Semantics-Driven Statistical Machine Translation (S2MT)*.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484.

Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond SumBasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing and Management*, 43(6):1606–1618.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based amr parsing with refined actions and auxiliary analyzers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for amr parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Wiki. 2018. Portal:Current events. https://en.wikipedia.org/wiki/Portal:Current_events.

Dani Yogatama and Kumiko Tanaka-Ishii. 2009. Multilingual spectral clustering using document similarity propagation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Dani Yogatama, Fei Liu, and Noah A. Smith. 2015. Extractive summarization by maximizing semantic volume. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Abstractive Unsupervised Multi-Document Summarization using Paraphrastic Sentence Fusion

Mir Tafseer Nayeem
University of Lethbridge
Lethbridge, AB, Canada
mir.nayeem@uleth.ca

Tanvir Ahmed Fuad
University of Lethbridge
Lethbridge, AB, Canada
t.fuad@uleth.ca

Yllias Chali
University of Lethbridge
Lethbridge, AB, Canada
chali@cs.uleth.ca

Abstract

In this work, we aim at developing an unsupervised abstractive summarization system in the multi-document setting. We design a paraphrastic sentence fusion model which jointly performs sentence fusion and paraphrasing using skip-gram word embedding model at the sentence level. Our model improves the information coverage and at the same time abstractiveness of the generated sentences. We conduct our experiments on the human-generated multi-sentence compression datasets and evaluate our system on several newly proposed Machine Translation (**MT**) evaluation metrics. Furthermore, we apply our sentence level model to implement an abstractive multi-document summarization system where documents usually contain a related set of sentences. We also propose an optimal solution for the classical summary length limit problem which was not addressed in the past research. For the document level summary, we conduct experiments on the datasets of two different domains (e.g., news article and user reviews) which are well suited for multi-document abstractive summarization. Our experiments demonstrate that the methods bring significant improvements over the state-of-the-art methods.

1 Introduction

The task of automatic document summarization aims at finding the most relevant informations in a text and presenting them in a condensed form. A good summary should retain the most important contents of the original document or a cluster of related documents, while being coherent, non-redundant and grammatically readable. There are two types of summarizations: abstractive summarization and extractive summarization. Abstractive methods need extensive natural language generation to rewrite the sentences (Chali et al., 2017). Therefore, research community is focusing more on extractive summaries, which selects salient (important) sentences from the source document without any modification to create a summary. The abstractive techniques which are traditionally used are sentence compression, syntactic reorganization and lexical paraphrasing. Summarization is classified as single-document or multi-document based upon the number of source document. The information overlap between the documents from the same topic makes the multi-document summarization more challenging than the task of summarizing single documents. However, in case of multi-document summarization where source documents usually contain similar information, the extractive methods would produce redundant summary or biased towards specific source document (Nayeem and Chali, 2017a).

Multi-sentence compression (**MSC**) can be a useful solution for the above problem. It usually takes a group of related sentences and produces an output sentence through merging the sentences about the same topic, retaining the most important information and still maintain the grammaticality of the generated sentence. MSC is a text-to-text generation process in which a novel sentence is produced as a result of summarizing a set of similar sentences originally called sentence fusion (Barzilay and McKeown, 2005). On the other hand, lexical paraphrasing aims at replacing some selected words with other similar words while preserving the meaning of the original text. A good lexical substitution for a target word needs to be semantically similar to the target word and compatible with the given context (Melamud et al., 2015).

Proceedings of the 27th International Conference on Computational Linguistics, pages 1191–1204
Santa Fe, New Mexico, USA, August 20-26, 2018.

For example, the sentence "Jack composed these verses in 1995" could be lexically paraphrased into "Jack wrote these lines in 1995" without altering the sense of the initial sentence. The contributions of this paper are as follows:

- We design a novel abstractive sentence generation model which jointly performs sentence fusion and paraphrasing using skipgram word embedding model.
- We apply our sentence level model to design a full abstractive multi-document summarization system and achieved the state-of-the-art results on two different datasets. Different from the recent neural abstractive models, our model is completely unsupervised, full abstractive (not limited to deletion based compressions) and applied to multi-document summarization.
- We also propose an optimal solution for the classical summary length limit problem in multi-document setting.

2 Related Work

Abstractive summarization is generally much more difficult which involves sophisticated techniques for meaning representation, content organization, sentence compression, sentence fusion, paraphrasing etc. There has been a huge interest on compressive document summarization that tries to compress original sentences to form a summary (Clarke and Lapata, 2006; Clarke and Lapata, 2008; Filippova, 2010) as a first intermediate step towards abstractive summarization. Compressive summarization techniques include sentences which are compressed from original sentences without further modifications other than word deletion. Sentence compression involving two or more sentences is called **MSC** (Multi-Sentence Compression). Most of the previous MSC approaches rely on the syntactic parsing to build the dependency tree for each related sentence in a cluster for producing grammatical compressions (Filippova and Strube, 2008). Unfortunately, syntactic parsers are not available for all the languages. As an alternative, word graph-based approaches that only require a POS tagger and a list of stopwords have been proposed first by (Filippova, 2010). A directed word graph is constructed in which nodes represent words and edges represent the adjacency between words in a sentence. Hence, compressed sentences are generated by finding k-shortest paths in the word graph. (Boudin and Morin, 2013) improved Filippova's approach by re-ranking the fusion candidate paths according to keyphrases to generate more informative sentences. However, grammaticality is sacrificed to improve informativity in these works (Nayeem and Chali, 2017b).

(Banerjee et al., 2015) proposed an abstractive multi-document summarization system using sentence fusion approach of (Filippova, 2010) combined with Integer Linear Programming (**ILP**) sentence selection. Following (Banerjee et al., 2015) work, several recent approaches have been proposed with slight modifications. Multiword Expressions (**MWE**) was exploited in (ShafieiBavani et al., 2016) to produce more informative compressions. Recently, (Tuan et al., 2017) include syntax factor along with (Banerjee et al., 2015) to improve performance. However, all the above mentioned systems try to produce compressions by copying the source sentence words, no paraphrasing is involved in the process.

Recently end-to-end training with encoder-decoder neural networks have achieved huge success in case of abstractive summarization. These systems have adopted techniques such as encoder-decoder with attention (Bahdanau et al., 2015; Luong et al., 2015) neural network models from the field of machine translation to model the sentence summarization task. (Rush et al., 2015) was the first to use neural sequence-to-sequence learning in headline generation task from a single document. Unfortunately, this line of research under the term sentence summarization (Rush et al., 2015), which can generate only a single sentence, somewhat misleadingly called text summarization in some follow-up research works (Nallapati et al., 2016; Chopra et al., 2016; Suzuki and Nagata, 2017; Zhou et al., 2017; Ma et al., 2017). There are some limitations to the above mentioned models, one is that the the produced output is also very short (about 75 characters). Same as the headlines, their model produces ungrammatical sentences during generation. However, there are some recent attempts which uses **CNN/DailyMail** corpus (Hermann et al., 2015) as a supervised training data to generate multi-sentence summary from a single document (See

et al., 2017; Li et al., 2017b; Paulus et al., 2017; Narayan et al., 2018a; Narayan et al., 2018b). The recent abstractive summarization models actually produce compressive summaries by deleting the words form a single source document, no direct paraphrasing was involved in the process. Hence, no new words were generated which are different form the source document words (other than morphological variation), which is pointed out by their own experimental results. Very recently, some researchers employ neural network based framework to tackle the summarization problem in multi-document setting (Yasunaga et al., 2017; Li et al., 2017a). (Yasunaga et al., 2017) is limited to extractive summmarization. On the other hand, (Li et al., 2017a) is limited to compressive summary generation using an ILP based model, and there is no explicit redundancy control in the summary side. Unfortunately, full abstractive summarization in multi-document setting still lacks satisfactory solutions due to the lack of large multi-document summarization datasets needed to train the computationally expensive sequence-to-sequence models. In this paper, we tackle this issue in an unsupervised way using deep representation learning.

3 Paraphrastic Sentence Fusion Model

Most of the previous works rely only on deletion based compressions, either sentence compression or fusion for abstracting sentences. Instead, in this paper we take the first step towards finding a joint representation for sentence abstraction using sentence fusion and lexical paraphrase rather than treating these two independently.

3.1 Word Graph Construction for Sentence Fusion

Given a cluster of related sentences we construct a word-graph following (Filippova, 2010; Boudin and Morin, 2013). Let $S = \{s_1, s_2, ..., s_n\}$ be a set of related sentences, we construct a graph $G = (V, E)$ by iteratively adding sentences to it. The vertices are the words along with the parts-of-speech (POS) tags and directed edges are formed by simply connecting the adjacent words in the sentences. Once the first sentence is added, words from the other related sentences are mapped onto a node in the graph provided that they have exactly the same lower cased word form and the same POS tag. Each sentence is connected to dummy start and end nodes to mark the beginning and ending of the sentences. Figure 1 illustrates an example word-graph for the following two sentences,

S1: In Asia Japan Nikkei lost 9.6% while Hong Kongs Hang Seng index fell 8.3%.

S2: Elsewhere in Asia Hong Kongs Hang Seng index fell 8.3% to 12,618.

As we can see, the two input sentences contain similar information, but differs in sentence length, syntax, and the detail of information. The solid directed arrows connect the words in the first sentence S1, while the dotted arrows join the words in the second sentence S2. After constructing the word-graph using (Filippova, 2010; Boudin and Morin, 2013) as described above, we can generate K-shortest paths from dummy start node to end node in the word graph (see Figure 1). For example, we can generate these paths:

Ex1: In Asia Hong Kongs Hang Seng index fell 8.3%.

Ex2: Elsewhere in Asia Hong Kongs Hang Seng index fell 8.3%.

Ex3: Elsewhere in Asia Japan Nikkei lost 9.6% while Hong Kongs Hang Seng index fell 8.3%.

The above examples are sampled from the K-shortest paths generated from the word-graph G (K is usually ranges from 50 to 200 according to the literature (Filippova, 2010; Boudin and Morin, 2013)). The main challenge is to rank these K fused sentences according to the information they contain. Hence, we design a candidate ranking strategy to sort the generated K-shortest paths based on the information coverage.

3.2 Candidate Ranking

We rank the fused candidates by applying **TextRank** algorithm (Mihalcea and Tarau, 2004) which involves constructing an undirected graph where candidates are vertices, and weighted edges are formed by connecting candidate sentences by a similarity metric. Original TextRank algorithm determines the similarity based on the lexical overlap. However, this algorithm has a serious drawback: If two sentences

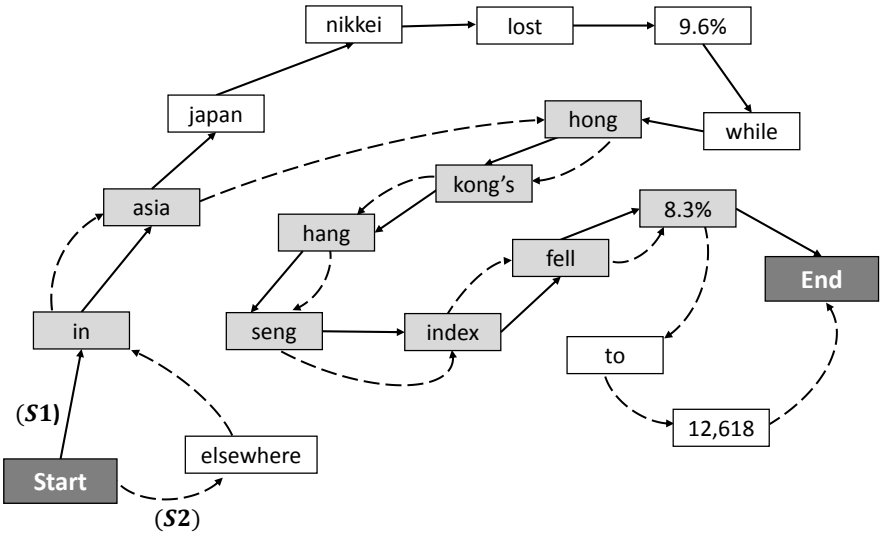


Figure 1: Constructed Word graph and a possible compression path (light gray nodes)

are talking about the same topic without using any overlapped words, there will be no edge between them. Instead, we apply the following representation of a sentence to capture the semantic information.

3.2.1 Sentence Embedding

A sentence is a sequence of words $\mathbf{S} = (w_1, w_2, ..., w_L)$, where L is the length of the sentence. We encode a sentence using bi-directional GRUs (Cho et al., 2014). In the simplest uni-directional case, while reading input symbols from left to right, a GRU learns the hidden annotations h_t at time t with

$$h_t = \mathbf{GRU}(h_{t-1}, e(w_t)) \tag{1}$$

where, the $h_t \in \mathbb{R}^n$ encodes all content seen so far at time t which is computed from h_{t-1} and $e(w_t)$, where $e(w_t) \in \mathbb{R}^m$ is the m-dimensional embedding of the current word w_t. We use 300-dimensional pre-trained word2vec embeddings[1](Mikolov et al., 2013) for each word as input to GRU.

As shown in Figure 2, **Bi-GRU** processes the input sentence in both forward and backward direction with two separate hidden layers calculated with GRUs, obtains the forward hidden states $(\overrightarrow{h_1}, \ldots, \overrightarrow{h_L})$ and the backward hidden states $(\overleftarrow{h_1}, \ldots, \overleftarrow{h_L})$. For each position t, we simply concatenate both forward and backward states into the final hidden state:

$$h_t = \overrightarrow{h_t} \oplus \overleftarrow{h_t} \tag{2}$$

in which operator $\oplus$ indicates concatenation. $\overrightarrow{h_t}$ is calculated using Eq. (1) and $\overleftarrow{h_t}$ is calculated using the following equation.

$$\overleftarrow{h_t} = \mathbf{GRU}(\overleftarrow{h_{t+1}}, e(w_t)) \tag{3}$$

$\overrightarrow{h_0}$ is initialized as zero vector, and the output sentence embedding x_i for the sentence S_i is the last hidden state:

$$x_i = h_L \tag{4}$$

We start by constructing an undirected graph where fused sentence candidates are vertices, and weighted edges are formed by measuring the cosine distance between the candidate sentence embeddings obtained from equation (4). After we have our graph, we run the TextRank (Mihalcea and Tarau,

[1]https://code.google.com/archive/p/word2vec/

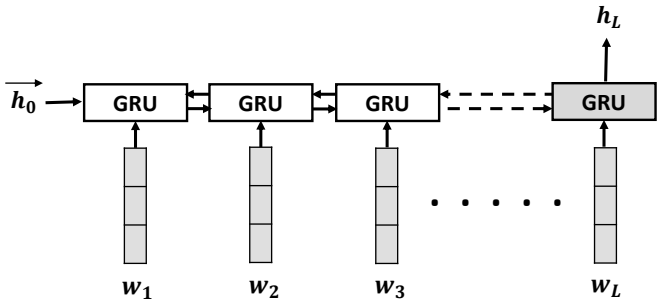


Figure 2: Sentence Embedding

2004) algorithm on it. This involves initializing a score of 1 for each vertex, and repeatedly applying the TextRank update rule until convergence. After reaching convergence, we extract the fused candidate sentences along with TextRank scores. For instance, $Rank(S_i)$ indicates the importance score assigned to sentence S_i.

3.3 Context Sensitive Lexical Substitution

3.3.1 Target Word Identification for Substitution

After constructing the word-graph as presented in section 3.1, we take only the nouns and verbs for possible substitution candidates from the word-graph G. We didn't consider the named entities, where, $NE \in \{PER; LOC; ORG; MISC\}$ for the substitution.

3.3.2 Substitution Selection

The **PPDB 2.0** (Pavlick et al., 2015) provides millions of lexical, phrasal and syntactic paraphrases which come into packages of different sizes (going from S to XXXL). For our model, we use the lexical XXL. For instance, we can gather lexical substitution set S = {*gliding, sailing, diving, travelling*} for the target word ($t = flying$) from PPDB 2.0. We hardcoded the model to select substitutes with the same POS tag and that are not a morphological variant (*e.g., fly, flew, flown*).

3.3.3 Substitution Ranking

Word embeddings are low-dimensional vector representations of words such as **word2vec** (Mikolov et al., 2013) that recently gained much attention in various semantic tasks. **Word2vecf** (Levy and Goldberg, 2014) is an extension of word2vec to produce syntax-based word embeddings. They show that these embeddings tend to capture functional word similarity (as in *manage* $\rightarrow$ *supervise*) rather than topical similarity (as in *manage* $\rightarrow$ *manager*). We use the word and context vectors released by (Melamud et al., 2015) which was shown to perform strongly on lexical substitution task. These embeddings contain 600d (600 dimension) vectors for 173k words and about 1M syntactic contexts processed using the dependency based word2vecf model (Levy and Goldberg, 2014). Their measure $addCos$ for estimating the appropriateness of a substitute s from the substitution set S, for the target word t in the set of the target word's context elements $C = \{c_1, c_2, ..., c_n\}$, which is defined as follows,

$$addCos(s|t, C) = \frac{cos(s, t) + \sum_{c \in C} cos(s, c)}{|C| + 1}$$

Finally, we select the best substitution $\mathbf{s}$ according to maximum $\mathbf{addCos}$ scores over $\mathbf{0.7}$ and attach it with the target word vertex $\mathbf{t}$ in the word-graph G along with the $\mathbf{addCos}$ score. For the other vertices which don't have substitution alternatives, we assign an $\mathbf{addCos}$ score of zero in the word-graph G.

3.3.4 Confidence Score

Once the substitutions are placed into the word-graph G, in order to maintain the grammaticality and the reliability of the final generated sentences, we use a 3-gram language model, which assigns probabilities to sequence of words in a generated candidate. Suppose that a candidate contains a sequence of m

words $\{w_1, w_2, w_3, \ldots, w_m\}$. The score **CS** (Confidence Score) assigned to each candidate is defined as follows:

$$CS(w_1, ..., w_m) = \frac{1}{1 - Score_{LM}(w_1, ..., w_m)}$$

In our experiment, we used a language model that is trained on the English Gigaword corpus[2]. In the word-graph G, for each substitution candidate we calculate the confidence score with the adjacent vertices (in our case words) and calculate their average. We also assign this confidence score with the substitution vertex in the word-graph G.

Finally, we rank the K candidate fusions and find the N-best paraphrastic sentence fusion which balances the information coverage and the abstractiveness. The score of a candidate sentence fusion c is given by the following linear combination between the candidate rank score and the abstractiveness score (where, we set $\alpha = 0.5$ to give equal importance and scaling the score to [0,1]).

$$score(c) = \alpha \cdot Rank(c) + (1 - \alpha) \cdot \sum_{V_i = V_{start}}^{V_{end}} addCos(\mathbf{V_i}) + CS(\mathbf{N}(\mathbf{V_i})) \quad (5)$$

where, $addCos(\mathbf{V_i})$ is the *addCos* score of the vertex V_i and $N(V_i)$ is the neighbours of the vertex V_i from the word-graph G.

4 Multi-Document Abstractive Summarization

In this section, we apply our sentence level paraphrastic fusion model to generate multi-document level abstractive summary under a certain length limit (L). Our system takes a set of related documents as input and preprocesses them which includes tokenization, Part-Of-Speech (POS) tagging, removal of stopwords, filtering punctuation marks and Lemmatization. We use **NLTK** toolkit[3] to preprocess each sentence to obtain a more accurate representation of the information. In the following, we successively describe each of the steps involved in the document summarization process.

4.1 Sentence Clustering

The sentence clustering step allows us to group similar sentences. We use a hierarchical agglomerative clustering (Murtagh and Legendre, 2014) with a complete linkage criteria. This method proceeds incrementally, starting with each sentence considered as a cluster, and merging the pair of similar clusters after each step using bottom up approach. The complete linkage criteria determines the metric used for the merge strategy, which means largest distance between a sentence in one cluster and a sentence in the other candidate cluster. In building the clusters, we use the cosine similarity between the sentence embeddings obtained from equation (5). We set a similarity threshold ($\tau = 0.5$) to stop the clustering process by using a hold out dataset **SICK**[4] of SemEval-2014 (Marelli et al., 2014) for getting optimal performance. If we cannot find any cluster pair with a similarity above the threshold ($\tau = 0.5$), the process stops, and the clusters are released. The clusters may be small, but are highly coherent as each sentence they contain must be similar to every other sentence in the same cluster. This sentence clustering step is very important due to two main reasons,

- Selecting at most one sentence from each cluster of related sentences will decrease redundancy from the summary side.
- Selecting sentences from the diverse set of clusters will increase the information coverage from the document side as well.

[2]Available : http://www.keithv.com/software/giga/ (We used the 64K NVP vocabulary version)
[3]http://www.nltk.org/
[4]http://clic.cimec.unitn.it/composes/sick.html

For each cluster of related sentences, we generate 10-best (N = 10) abstractive fused sentences using our model described in section 3, the generated sentences differ in lengths as well as information. However, for the clusters containing only one sentence, we use our context sensitive lexical substitution model presented in section 3.3 to generate just the abstractive version of the source sentence.

4.2 Abstractive Sentence Selection

In our work, we use the concept-based ILP framework introduced in (Gillick and Favre, 2009) with some suitable changes to select the best subset of sentences. This approach aims to extract sentences that cover as many important concepts as possible, while ensuring the summary length is within a given budgeted constraint. Unlike (Gillick and Favre, 2009) which uses bigrams as concepts, we use keyphrases as concepts. Keyphrases are the words or phrases that represent the main topics of a document. Sentences containing the most relevant keyphrases are important for the summary generation. We extract the keyphrases from the document cluster using **RAKE**[5] (Rose et al., 2010). We assign a weight to each keyphrase using the score returned by RAKE.

Let $\bar{w}_i$ be the weight of keyphrase i and k_i a binary variable that indicates the presence of keyphrase i in the selected parafused sentences. Let l_j be the number of words or characters in sentence j, s_j a binary variable that indicates the presence of sentence j in the selected parafused sentence set and L the length limit for the set. Let Occ_{ij} indicate the occurrence of keyphrase i in sentence j, the ILP formulation is,

$$max : (\sum_i \bar{w}_i k_i + \sum_j (score(s_j) + \frac{l_j}{L}) \cdot s_j) \tag{6}$$

$$Subject\ to : \sum_j l_j s_j \leq L \tag{7}$$

$$s_j Occ_{ij} \leq k_i, \quad \forall i, j \tag{8}$$

$$\sum_j s_j Occ_{ij} \geq k_i, \quad \forall i \tag{9}$$

$$\sum_{j \in g_c} s_j \leq 1, \quad \forall g_c \tag{10}$$

$$k_i \in \{0, 1\} \quad \forall i \tag{11}$$

$$s_j \in \{0, 1\} \quad \forall j \tag{12}$$

We try to maximize the weight of the keyphrases and our **ParaFuse** model's score (6), while avoiding repetition of those keyphrases (8, 9) and staying under the maximum number of words or characters allowed for the selected parafused sentences (7). In order to ensure at most one sentence per parafused cluster in the summary, we add an extra constraint (10), this will ensure non-redundancy from the summary side. In this process, we select the optimal combination of abstractive sentences that maximize information coverage while minimizing redundancy.

[5]https://github.com/aneesha/RAKE

4.3 Summary Length Limit Problem

One of the essential properties of the text summarization systems is the ability to generate a summary with a fixed length (**DUC 2004**, Task-2 (Multi-Document): Length limit = 100 Words). According to (Hong et al., 2014) all the multi-document summarizer from the previous research either truncated the summary to 100^{th} word, or removed the last sentence from the summary set. However, the first option produces a certain ungrammatical sentence, the later one can lose a lot of information in the worst case, if the sentences are long. Recently, (Kikuchi et al., 2016) propose four methods in order to tackle this issue, two of them are based on different decoding procedures without model architecture modification and the other two are learning-based, i.e., the models take the desired length information as input and encode it into the model architecture. However, their model is limited to headline generation task, where models generate a single sentence headline of a document. In this work, we tackle this issue in multi-document setting by generating N-best paraphrastic fusion length variations of a cluster of related sentences. Our model can effectively produce different length variations because of the shortest path strategy from start node to end node (see section 3.1 for the examples). In our ILP formulation for the document level summary generation, we try to maximize the total summary length in the objective function (equation (6)) to optimally solve the length limit problem. Under any circumstances, our model can choose a shorter variation of a sentence automatically to be included in the summary.

4.4 Experiments

In this section, we present our experimental details for assessing the performance of the sentence level paraphrastic fusion model and multi-document level abstractive summarization system as described above. We give details on the datasets we used, evaluation metrics, and the baseline systems used for comparison with our approach.

4.4.1 Sentence Level Experiments

We generate 50 shortest paths from start to end node for each cluster of related sentences using our paraphrastic sentence fusion model. The paths shorter than eight words or that do not contain a verb are filtered. To ensure pure abstractive compression generation, we remove the paths that have $cosineSimilarity \geq 0.9$ to any of the original sentence in the cluster. We then select 3-best candidates from K paths using the scoring function in equation (5). For fair evaluation, we also select the 3-best candidates for the baseline systems that we compare with our model.

Dataset: We conduct experiments on the human generated sentence fusion dataset released by (McKeown et al., 2010). This dataset consists of 300 English sentence pairs taken from newswire clusters accompanied by human-produced sentence fusions rewrites collected via Amazon's Mechanical Turk service[6]. We filtered the sentences which have no main verbs. The resulting set contains 296 pairs of sentences.

Evaluation Metric: We evaluate our system automatically using various automatic metrics. **BLEU** (Papineni et al., 2002) is the most commonly used metric for Machine Translation evaluation. BLEU relies on exact matching of n-grams and has no concept of synonymy or paraphrasing. **SARI** (Xu et al., 2016) a recently proposed metric which compares **S**ystem output **A**gainst **R**eferences and against the **I**nput sentence. SARI computes the arithmetic average of n-gram precision and recall of three rewrite operations: addition, copying, and deletion which correlates well with human references. **METEOR-E**[7] (Servan et al., 2016) is an augmented version of METEOR (Denkowski and Lavie, 2014) using distributed representations which can easily measure the abstractiveness. **Compression Ratio** is a measure of how terse a compression. A compression ratio of zero implies that the source sentence is fully uncompressed. We define **Copy Rate** as how many tokens are copied to the abstract sentence from the source sentence without paraphrasing in the following equation (13). Lower copy rate score means more paraphrasing is involved in the abstract sentence. Copy rate of 100% means no paraphrasing.

[6] http://www.mturk.com

[7] https://github.com/cservan/METEOR-E

Input Sentences	Bush, who initially nominated Roberts to replace retiring Justice Sandra Day O'Connor, tapped him to lead the court the day after Rehnquist's death. President Bush initially nominated Roberts in July to succeed retiring Justice Sandra Day O'Connor.
(Filippova, 2010)	president bush initially nominated roberts to replace retiring justice sandra day o'connor .
(Boudin and Morin, 2013)	bush , who initially nominated roberts in july to succeed retiring justice sandra day o'connor , tapped him to lead the court the day after rehnquist 's death .
(Banerjee et al., 2015)	bush , who initially nominated roberts to replace retiring justice sandra day o'connor , tapped him to lead the court the day after rehnquist 's death .
Paraphrastic Fusion (*ours*)	president bush initially **recommended** roberts in july to **substitute** retiring justice sandra day o'connor , tapped him to **run** the court the day after rehnquist 's death .

Table 1: The output generated by the baseline and our system (the paraphrased words are marked bold)

Model	BLEU	SARI	METEOR-E	Compression Ratio	Copy Rate
(Filippova, 2010)	40.6	34.6	0.31	**0.57**	99.8
(Boudin and Morin, 2013)	**44.0**	37.2	0.36	0.42	99.9
(Banerjee et al., 2015)	42.3	36.5	0.34	0.45	99.8
Paraphrastic Fusion (*ours*)	42.5	**37.4**	**0.43**	0.41	**76.2**

Table 2: Comparison with baselines and our **Paraphrastic Fusion** model across different automatic evaluation metrics (the scores are averaged)

$$Copy\ Rate = \frac{|S_{orig} \cap S_{abs}|}{|S_{abs}|} \tag{13}$$

4.5 Baseline Systems and Results

We compare our system with (Filippova, 2010), (Boudin and Morin, 2013)[8] and (Banerjee et al., 2015)[9]. Table 1 shows the output generated by the baseline and our system. We report our system's performance compared with the baselines in terms of different evaluation metrics in Table 2. Our model balances the information coverage (**BLUE**, **SARI**) and complete abstractiveness (**METEOR-E**, **Copy Rate**) instead of over compressing the generated sentences (**Compression Ratio**). We get slightly higher score in **SARI** because of the multiple human abstractive rewrites along with input sentence. The **Copy Rate** score of other baseline systems clearly indicates the fact that they are doing completely deletion based compression, no paraphrasing is involved. Moreover, we also get higher score in **METEOR-E** metric because of the lexical substitution operation. As expected, we get little lower **BLEU** score compared to (Boudin and Morin, 2013) for two main reasons (1) We tried to balance between information coverage and abstractiveness (2) **BLEU** works well on surface level lexical overlap.

4.5.1 Multi-Document Level Experiments

Dataset: We consider the generic multi-document summarization dataset provided at Document Understanding Conference (**DUC 2004**)[10] which is one of the main benchmark dataset in the multi-document summarization field. It contains 50 document clusters and each is composed of 10 news wire articles about a given topic from the Associated Press and The New York Times that are published between 1998 to 2000. The dataset also contains multiple human-written summaries which are used for the evaluation of system-generated summaries. The **Opinosis** (Ganesan et al., 2010) is another dataset consists of short user reviews in 51 different topics collected from TripAdvisor, Amazon, and Edmunds. The dataset is well suited for multi-document summarization which includes 5 different golden summaries for each topic created by human authors.

[8]https://github.com/boudinfl/takahe

[9]https://github.com/StevenLOL/AbTextSumm

[10]http://duc.nist.gov/duc2004/

Dataset	Models	R-1	R-2	R-WE-1	R-WE-2
DUC 2004	LexRank (Erkan and Radev, 2004)	35.95	7.47	36.91	7.91
	Submodular (Lin and Bilmes, 2011)	39.18	9.35	40.03	9.92
	RegSum (Hong and Nenkova, 2014)	38.57	9.75	39.12	10.33
	ILPSumm (Banerjee et al., 2015)	39.24	11.99	40.21	12.08
	PDG* (Yasunaga et al., 2017)	38.45	9.48	39.07	10.24
	ParaFuse_doc (*ours*)	**40.07**	**12.04**	**42.31**	**12.96**
Opinosis 1.0	TextRank (Mihalcea and Tarau, 2004)	27.56	6.12	28.20	6.45
	Opinosis (Ganesan et al., 2010)	32.35	9.13	33.54	9.41
	Biclique (Muhammad et al., 2016)	33.03	8.96	33.91	9.25
	ParaFuse_doc (*ours*)	**33.86**	**9.74**	**34.46**	**10.09**

Table 3: Results on DUC 2004 (Task-2) and Opinosis 1.0

Evaluation Metric: We evaluate our summarization system using **ROUGE**[11] (Lin, 2004) on **DUC 2004** (Task-2, Length limit (L) = 100 Words) and **Opinosis 1.0** (L = 15 Words). However, ROUGE scores are unfairly biased towards lexical overlap at surface level. Taking this into account, we also evaluate our system using **ROUGE-WE** (Ng and Abrecht, 2015), which considers word embeddings to compute the semantic similarity of the words. We report limited length recall performance for both the metrics, as our system generated summaries are forced to be concise through some constraints (such as length limit constraint). Therefore, we consider using just the recall score since precision is of less concern in this scenario.

4.5.2 Baseline Systems & Results

The summaries generated by the baseline **LexRank** (Erkan and Radev, 2004) and the state-of-the-art summarizers (**Submodular** (Lin and Bilmes, 2011) and **RegSum** (Hong and Nenkova, 2014)) on the DUC 2004 dataset were collected from (Hong et al., 2014). In case of **ILPSumm**[12] (Banerjee et al., 2015) and **PDG*** (Yasunaga et al., 2017), we use the author provided implementation to generate summary from their model. For Opinosis 1.0 dataset, we use an open source implementation of **TextRank** (Mihalcea and Tarau, 2004)[13]. Moreover, we use the author provided implementation for the **Opinosis** (Ganesan et al., 2010) and **Biclique** (Muhammad et al., 2016) to generate summaries. According to the Table 3, our multi-document level model **ParaFuse_doc** achieves the best summarization performance on all the ROUGE metrics for both the datasets. The slight increase in terms of **R-WE** metric clearly justifies the fact of abstractiveness proposed in this work which highly correlates with human references.

5 Conclusion

In this paper, we designed a new abstractive fusion generation model at the sentence level which jointly performs sentence fusion and paraphrasing. Our sentence level model is very well suited for full abstractive multi-document summarization which was justified by the experimental results on two benchmark datasets of different domains. Furthermore, we designed an optimal solution for the classical summary length limit problem in multi-document setting.

Acknowledgements

We would like to thank the anonymous reviewers for their useful comments. The research reported in this paper was conducted at the University of Lethbridge and supported by the Natural Sciences and Engineering Research Council (**NSERC**) of Canada discovery grant and the University of Lethbridge.

[11] ROUGE-1.5.5 with options: -n 2 -m -u -c 95 -x -r 1000 -f A -p 0.5 -t 0

[12] https://github.com/StevenLOL/AbTextSumm

[13] https://github.com/davidadamojr/TextRank

References

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

Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document abstractive summarization using ilp based multi-sentence compression. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1208–1214. AAAI Press.

Regina Barzilay and Kathleen R. McKeown. 2005. Sentence fusion for multidocument news summarization. *Comput. Linguist.*, 31(3):297–328, September.

Florian Boudin and Emmanuel Morin. 2013. Keyphrase extraction for n-best reranking in multi-sentence compression. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 298–305, Atlanta, Georgia, June. Association for Computational Linguistics.

Yllias Chali, Moin Tanvee, and Mir Tafseer Nayeem. 2017. Towards abstractive multi-document summarization using submodular function-based framework, sentence compression and merging. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing, IJCNLP 2017, Taipei, Taiwan, November 27 - December 1, 2017, Volume 2: Short Papers*, pages 418–424.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. pages 103–111, October.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, California, June. Association for Computational Linguistics.

James Clarke and Mirella Lapata. 2006. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 377–384, Stroudsburg, PA, USA. Association for Computational Linguistics.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA, June. Association for Computational Linguistics.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479, December.

Katja Filippova and Michael Strube. 2008. Sentence fusion via dependency graph compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 177–185, Stroudsburg, PA, USA. Association for Computational Linguistics.

Katja Filippova. 2010. Multi-sentence compression: Finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 322–330, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: A graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 340–348, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, ILP '09, pages 10–18, Stroudsburg, PA, USA. Association for Computational Linguistics.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1693–1701, Cambridge, MA, USA. MIT Press.

Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multi-document summarization. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 712–721, Gothenburg, Sweden, April. Association for Computational Linguistics.

Kai Hong, John Conroy, Benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A repository of state of the art and competitive baseline summaries for generic news summarization. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1608–1616, Reykjavik, Iceland, May. European Language Resources Association (ELRA). ACL Anthology Identifier: L14-1070.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1328–1338, Austin, Texas, November. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland, June. Association for Computational Linguistics.

Piji Li, Wai Lam, Lidong Bing, Weiwei Guo, and Hang Li. 2017a. Cascaded attention based unsupervised information distillation for compressive summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2081–2090, Copenhagen, Denmark, September. Association for Computational Linguistics.

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017b. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2091–2100, Copenhagen, Denmark, September. Association for Computational Linguistics.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 510–520, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July. Association for Computational Linguistics.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421, Lisbon, Portugal, September. Association for Computational Linguistics.

Shuming Ma, Xu Sun, Jingjing Xu, Houfeng Wang, Wenjie Li, and Qi Su. 2017. Improving semantic relevance for sequence-to-sequence learning of chinese social media text summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 635–640, Vancouver, Canada, July. Association for Computational Linguistics.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may. European Language Resources Association (ELRA).

Kathleen McKeown, Sara Rosenthal, Kapil Thadani, and Coleman Moore. 2010. Time-efficient creation of an accurate sentence fusion corpus. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–320, Los Angeles, California, June. Association for Computational Linguistics.

Oren Melamud, Omer Levy, and Ido Dagan. 2015. A simple word embedding model for lexical substitution. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 1–7, Denver, Colorado, June. Association for Computational Linguistics.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 404–411, Barcelona, Spain, July. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Azam Sheikh Muhammad, Peter Damaschke, and Olof Mogren. 2016. Summarizing online user reviews using bicliques. In *Proceedings of the 42Nd International Conference on SOFSEM 2016: Theory and Practice of Computer Science - Volume 9587*, pages 569–579, New York, NY, USA. Springer-Verlag New York, Inc.

Fionn Murtagh and Pierre Legendre. 2014. Ward's hierarchical agglomerative clustering method: Which algorithms implement ward's criterion? *J. Classif.*, 31(3):274–295, October.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Ça glar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *CoNLL 2016*, page 280.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018a. Ranking Sentences for Extractive Summarization with Reinforcement Learning. In *Proceedings of the NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics*.

Shashi Narayan, Nikos Papasarantopoulos, Shay B. Cohen, and Mirella Lapata. 2018b. Neural extractive summarization with side information. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, February.

Mir Tafseer Nayeem and Yllias Chali. 2017a. Extract with order for coherent multi-document summarization. In *Proceedings of TextGraphs@ACL 2017: the 11th Workshop on Graph-based Methods for Natural Language Processing, Vancouver, Canada, August 3, 2017*, pages 51–56.

Mir Tafseer Nayeem and Yllias Chali. 2017b. Paraphrastic fusion for abstractive multi-sentence compression generation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 2223–2226.

Jun-Ping Ng and Viktoria Abrecht. 2015. Better summarization evaluation with word embeddings for rouge. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1925–1930, Lisbon, Portugal, September. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430, Beijing, China, July. Association for Computational Linguistics.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. In Michael W. Berry and Jacob Kogan, editors, *Text Mining. Applications and Theory*, pages 1–20. John Wiley and Sons, Ltd.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal, September. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada, July. Association for Computational Linguistics.

Christophe Servan, Alexandre Berard, zied elloumi, Hervé Blanchon, and Laurent Besacier. 2016. Word2vec vs dbnary: Augmenting meteor using vector representations or lexical resources? In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1159–1168, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Elaheh ShafieiBavani, Mohammad Ebrahimi, Raymond K. Wong, and Fang Chen. 2016. An efficient approach for multi-sentence compression. In *Proceedings of The 8th Asian Conference on Machine Learning*, volume 63 of *Proceedings of Machine Learning Research*, pages 414–429, The University of Waikato, Hamilton, New Zealand, 16–18 Nov. PMLR.

Jun Suzuki and Masaaki Nagata. 2017. Cutting-off redundant repeating generations for neural abstractive summarization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 291–297, Valencia, Spain, April. Association for Computational Linguistics.

Dung Tran Tuan, Nam Van Chi, and Minh-Quoc Nghiem, 2017. *Multi-sentence Compression Using Word Graph and Integer Linear Programming*, pages 367–377. Springer International Publishing, Cham.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 452–462, Vancouver, Canada, August. Association for Computational Linguistics.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1095–1104, Vancouver, Canada, July. Association for Computational Linguistics.

Adversarial Domain Adaptation for Variational Neural Language Generation in Dialogue Systems

Van-Khanh Tran[1,2] and Le-Minh Nguyen[1]
[1]Japan Advanced Institute of Science and Technology, JAIST
1-1 Asahidai, Nomi, Ishikawa, 923-1292, Japan
{tvkhanh, nguyenml}@jaist.ac.jp
[2]University of Information and Communication Technology, ICTU
Thai Nguyen University, Vietnam
tvkhanh@ictu.edu.vn

Abstract

Domain Adaptation arises when we aim at learning from source domain a model that can perform acceptably well on a different target domain. It is especially crucial for Natural Language Generation (NLG) in Spoken Dialogue Systems when there are sufficient annotated data in the source domain, but there is a limited labeled data in the target domain. How to effectively utilize as much of existing abilities from source domains is a crucial issue in domain adaptation. In this paper, we propose an adversarial training procedure to train a Variational encoder-decoder based language generator via multiple adaptation steps. In this procedure, a model is first trained on a source domain data and then fine-tuned on a small set of target domain utterances under the guidance of two proposed critics. Experimental results show that the proposed method can effectively leverage the existing knowledge in the source domain to adapt to another related domain by using only a small amount of in-domain data.

1 Introduction

Traditionally, Spoken Dialogue Systems are typically developed for various specific domains, including: finding a hotel, searching a restaurant (Wen et al., 2015a), or buying a tv, laptop (Wen et al., 2015b), flight reservations (Levin et al., 2000), etc. Such system are often requiring a well-defined ontology, which is essentially a data structured representation that the dialogue system can converse about. Statistical approaches to multi-domain in SDS system have shown promising results in how to reuse data in a domain-scalable framework efficiently (Young et al., 2013). Mrkšić et al. (2015) addressed the question of multi-domain in the SDS belief tracking by training a general model and adapting it to each domain.

Recently, Recurrent Neural Networks (RNNs) based methods have shown improving results in tackling the domain adaptation issue (Chen et al., 2015; Shi et al., 2015; Wen et al., 2016a; Wen et al., 2016b). Such generators have also achieved promising results when providing such adequate annotated datasets (Wen et al., 2015b; Wen et al., 2015a; Tran et al., 2017; Tran and Nguyen, 2017a; Tran and Nguyen, 2017b). More recently, the development of the variational autoencoder (VAE) framework (Kingma and Welling, 2013; Rezende and Mohamed, 2015) has paved the way for learning large-scale, directed latent variable models. This has brought considerable benefits to significant progress in natural language processing (Bowman et al., 2015; Miao et al., 2016; Purushotham et al., 2017; Mnih and Gregor, 2014), dialogue system (Wen et al., 2017; Serban et al., 2017).

This paper presents an adversarial training procedure to train a variational neural language generator via multiple adaptation steps, which enables the generator to learn more efficiently when in-domain data is in short supply. In summary, we make the following contributions: (1) We propose a variational approach for an NLG problem which benefits the generator to adapt faster to new, unseen domain irrespective of scarce target resources; (2) We propose two critics in an adversarial training procedure, which can guide the generator to generate outputs that resemble the sentences drawn from the target domain; (3) We propose a unifying variational domain adaptation architecture which performs acceptably well

Proceedings of the 27th International Conference on Computational Linguistics, pages 1205–1217
Santa Fe, New Mexico, USA, August 20-26, 2018.

in a new, unseen domain by using a limited amount of in-domain data; (4) We investigate the effectiveness of the proposed method in different scenarios, including ablation, domain adaptation, scratch, and unsupervised training with various amount of data.

2 Related Work

Generally, Domain Adaptation involves two different types of datasets, one from a source domain and the other from a target domain. The source domain typically contains a sufficient amount of annotated data such that a model can be efficiently built, while there is often little or no labeled data in the target domain. Domain adaptation for NLG have been less studied despite its important role in developing multi-domain SDS. Walker et al. (2001) proposed a SPoT-based generator to address domain adaptation problems. Subsequently, a system focused on tailoring user preferences (Walker et al., 2007), and controlling user perceptions of linguistic style (Mairesse and Walker, 2011). Moreover, a phrase-based statistical generator (Mairesse et al., 2010) using graphical models and active learning, and a multi-domain procedure (Wen et al., 2016a) via data counterfeiting and discriminative training.

Neural variational framework for generative models of text have been studied longitudinally. Chung et al. (2015) proposed a recurrent latent variable model VRNN for sequential data by integrating latent random variables into hidden state of a RNN model. A hierarchical multi scale recurrent neural networks was proposed to learn both hierarchical and temporal representation (Chung et al., 2016). Zhang et al. (2016) introduced a variational neural machine translation that incorporated a continuous latent variable to model underlying semantics of sentence pairs. Bowman et al. (2015) presented a variational autoencoder for unsupervised generative language model.

Adversarial adaptation methods have shown promising improvement in many machine learning applications despite the presence of domain shift or dataset bias, which reduce the difference between the training and test domain distributions, and thus improve generalization performance. Tzeng et al. (2017) proposed an improved unsupervised domain adaptation method to learn a discriminative mapping of target images to the source feature space by fooling a domain discriminator that tries to differentiate the encoded target images from source examples. We borrowed the idea of (Ganin et al., 2016), where a domain-adversarial neural network are proposed to learn features that are discriminative for the main learning task on the source domain, and indiscriminate with respect to the shift between domains.

3 Variational Domain-Adaptation Neural Language Generator

Drawing inspiration from Variational autoencoder (Kingma and Welling, 2013) with assumption that there exists a continuous latent variable z from a underlying semantic space of Dialogue Act (DA) and utterance pairs $(\mathbf{d}, \mathbf{y})$, we explicitly model the space together with variable $\mathbf{d}$ to guide the generation process, *i.e.*, $p(\mathbf{y}|z, \mathbf{d})$. With this assumption, the original conditional probability evolves to reformulate as follows:

$$p(\mathbf{y}|\mathbf{d}) = \int_z p(\mathbf{y}, z|\mathbf{d})\mathbf{d}_z = \int_z p(\mathbf{y}|z, \mathbf{d})p(z|\mathbf{d})\mathbf{d}_z \tag{1}$$

This latent variable enables us to model the underlying semantic space as a global signal for generation, in which the variational lower bound of variational generator can be formulated as follows:

$$\mathcal{L}_{VAE}(\theta, \phi, \mathbf{d}, \mathbf{y}) = -KL(q_\phi(z|\mathbf{d}, \mathbf{y})||p_\theta(z|\mathbf{d})) + \mathbb{E}_{q_\phi(z|\mathbf{d},\mathbf{y})}[\log p_\theta(\mathbf{y}|z, \mathbf{d})] \tag{2}$$

where: $p_\theta(z|\mathbf{d})$ is the prior model, $q_\phi(z|\mathbf{d}, \mathbf{y})$ is the posterior approximator, and $p_\theta(\mathbf{y}|z, \mathbf{d})$ is the decoder with the guidance from global signal z, $KL(Q||P)$ is the Kullback-Leibler divergence between Q and P.

3.1 Variational Neural Encoder

The variational neural encoder aims at encoding a given input sequence $w_1, w_2, .., w_L$ into continuous vectors. In this work, we use a 1-layer, Bidirectional LSTM (BiLSTM) to encode the sequence embedding. The BiLSTM consists of forward and backward LSTMs, which read the sequence from left-to-right and right-to-left to produce both forward and backward sequence of hidden states $(\overrightarrow{\mathbf{h}_1}, .., \overrightarrow{\mathbf{h}_L})$, and

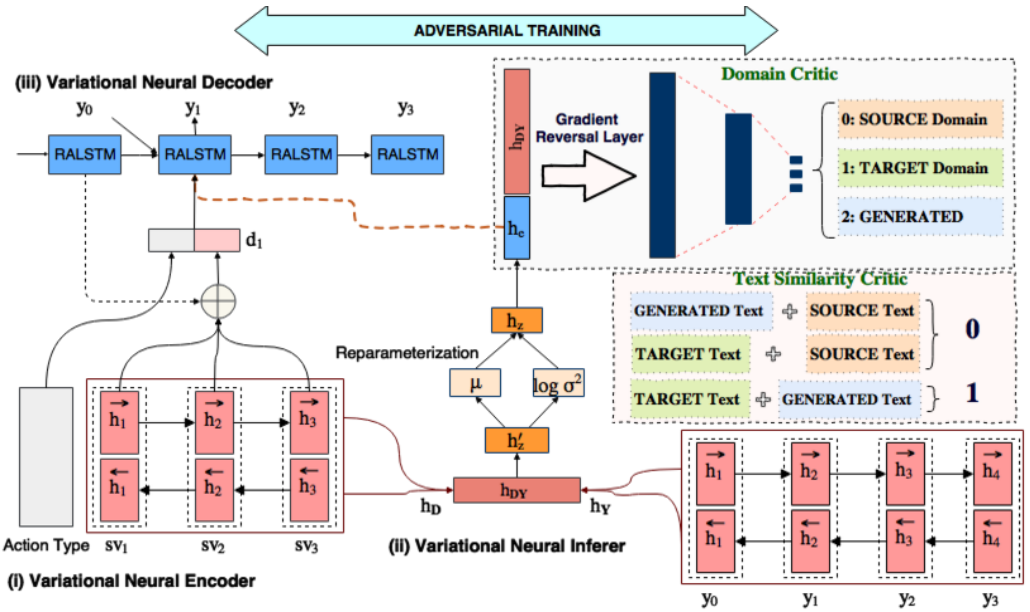


Figure 1: The VDANLG architecture which consists of two main components: the VRALSTM to generate the sentence and two Critics with an adversarial training procedure to guide the model in domain adaptation.

$(\overleftarrow{\mathbf{h}}_1, .., \overleftarrow{\mathbf{h}}_L)$, respectively. We then obtain the sequence of encoded hidden states $\mathbf{h_E} = (\mathbf{h}_1, \mathbf{h}_2, .., \mathbf{h}_L)$ where: $\mathbf{h}_i = \overrightarrow{\mathbf{h}_i} + \overleftarrow{\mathbf{h}_i}$. We utilize this encoder to represent both the sequence of slot-value pairs $\{\mathbf{sv}_i\}_{i=1}^{T_{DA}}$ in a given Dialogue Act, and the corresponding utterance $\{\mathbf{y}_i\}_{i=1}^{T_Y}$ (see the red parts in Figure 1). We finally operate the mean-pooling over the BiLSTM hidden vectors to obtain the representation: $\mathbf{h}_D = \frac{1}{T_{DA}} \sum_i^{T_{DA}} \mathbf{h}_i, \mathbf{h}_Y = \frac{1}{T_Y} \sum_i^{T_Y} \mathbf{h}'_i$. The encoder, accordingly, produces both the DA representation vector which flows into the inferer and decoder, and the utterance representation which streams to the posterior approximator.

3.2 Variational Neural Inferer

In this section, we describe our approach to model both the prior $p_\theta(z|\mathbf{d})$ and the posterior $q_\phi(z|\mathbf{d}, \mathbf{y})$ by utilizing neural networks.

Neural Posterior Approximator

Modeling the true posterior $p(z|\mathbf{d}, \mathbf{y})$ is usually intractable. Traditional approach fails to capture the true posterior distribution of z due to its oversimplified assumption when using the mean-field approaches. Following the work of (Kingma and Welling, 2013), in this paper we employ neural network to approximate the posterior distribution of z to simplify the posterior inference. We assume the approximation has the following form:

$$q_\phi(z|\mathbf{d}, \mathbf{y}) = \mathcal{N}(z; \mu(f(\mathbf{h}_D, \mathbf{h}_Y)), \sigma^2(f(\mathbf{h}_D, \mathbf{h}_Y))\boldsymbol{I}) \tag{3}$$

where: the mean μ and standard variance σ are the outputs of the neural network based on the representations of $\mathbf{h}_D$ and $\mathbf{h}_Y$. The function f is a non-linear transformation that project the both DA and utterance representations onto the latent space:

$$\mathbf{h}'_z = f(\mathbf{h}_D, \mathbf{h}_Y) = g(\mathbf{W}_z[\mathbf{h}_D; \mathbf{h}_Y] + b_z) \tag{4}$$

where: $\mathbf{W}_z \in \mathbb{R}^{d_z \times (d_{\mathbf{h}_D} + d_{\mathbf{h}_Y})}$, $b_z \in \mathbb{R}^{d_z}$ are matrix and bias parameters respectively, d_z is the dimensionality of the latent space, $g(.)$ is an elements-wise activation function which we set to be $Relu$ in our experiments. In this latent space, we obtain the diagonal Gaussian distribution parameter μ and $\log \sigma^2$ through linear regression:

$$\mu = \mathbf{W}_\mu \mathbf{h}'_z + b_\mu, \log \sigma^2 = \mathbf{W}_\sigma \mathbf{h}'_z + b_\sigma \tag{5}$$

where: μ, $\log \sigma^2$ are both d_z dimension vectors.

Neural Prior Model

We model the prior as follows:

$$p_\theta(z|\mathbf{d}) = \mathcal{N}(z; \mu'(\mathbf{d}), \sigma'(\mathbf{d})^2 \boldsymbol{I}) \tag{6}$$

where: μ' and σ' of the prior are neural models based on DA representation only, which are the same as those of the posterior $q_\phi(z|\mathbf{d}, \mathbf{y})$ in Eq. 4 and Eq. 5, except for the absence of $\mathbf{h}_Y$. To acquire a representation of the latent variable z, we utilize the same technique as proposed in VAE (Kingma and Welling, 2013) and re-parameterize it as follows:

$$\mathbf{h}_z = \mu + \sigma \odot \epsilon, \epsilon \sim \mathcal{N}(0, \boldsymbol{I}) \tag{7}$$

In addition, we set $\mathbf{h}_z$ to be the mean of the prior $p_\theta(z|\mathbf{d})$, *i.e.*, μ', during decoding due to the absence of the utterance y. Intuitively, by parameterizing the hidden distribution this way, we can back-propagate the gradient to the parameters of the encoder and train the whole network with stochastic gradient descent. Note that the parameters for the prior and the posterior are independent of each other.

In order to integrate the latent variable $\mathbf{h}_z$ into the decoder, we use a non-linear transformation to project it onto the output space for generation:

$$\mathbf{h}_e = g(\mathbf{W}_e \mathbf{h}_z + b_e) \tag{8}$$

where: $\mathbf{h}_e \in \mathbb{R}^{d_e}$. It is important to notice that due to the sample noise ϵ, the representation of $\mathbf{h}_e$ is not fixed for the same input DA and model parameters. This benefits the model to learn to quickly adapt to a new domain (see Table 1-(a) and Table 3, *sec.* 3).

3.3 Variational Neural Decoder

Given a DA $\mathbf{d}$ and the latent variable z, the decoder calculates the probability over the generation $\mathbf{y}$ as a joint probability of ordered conditionals:

$$p(\mathbf{y}|z, \mathbf{d}) = \prod_{j=1}^{T_Y} p(\mathbf{y}_t|\mathbf{y}_{<t}, z, \mathbf{d}) \tag{9}$$

where: $p(\mathbf{y}_t|\mathbf{y}_{<t}, z, \mathbf{d}) = g'(RNN(\mathbf{y}_t, \mathbf{h}_{t-1}, \mathbf{d}_t)$ In this paper, we borrow the $\mathbf{d}_t$ calculation and the computational RNN cell from (Tran and Nguyen, 2017a) where RNN(.)=RALSTM(.) with a slightly modification in order to integrate the representation of latent variable, *i.e.*, $\mathbf{h}_e$, into the RALSTM cell, which is denoted by the bold dashed orange arrow in Figure 1-(iii). We modify the cell calculation as follows:

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \hat{\mathbf{c}}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{W}_{4d_h, 4d_h} \begin{pmatrix} \mathbf{h}_e \\ \mathbf{d}_t \\ \mathbf{h}_{t-1} \\ \mathbf{y}_t \end{pmatrix} \tag{10}$$

where: $\mathbf{i}_i$, $\mathbf{f}_t$, $\mathbf{o}_t$ are input, forget and output gates respectively, d_h is hidden layer size, $\mathbf{W}_{4d_h, 4d_h}$ is model parameter.

The resulting Variational RALSTM (VRALSTM) model is demonstrated in Figure 1-(i), (ii), (iii), in which the latent variable can affect the hidden representation through the gates. This allows the model can indirectly take advantage of the underlying semantic information from the latent variable z. In addition, when the model learns to adapt to a new domain with unseen dialogue act, the semantic representation $\mathbf{h}_e$ can help to guide the generation process (see *sec.* 6.3 for details).

3.4 Critics

In this section, we introduce a *text-similarity critic* and a *domain critic* to guarantee, as much as possible, that the generated sentences resemble the sentences drawn from the target domain.

Text similarity critic

To check the relevance between sentence pair in two domains and to encourage the model generating sentences in the style which is highly *similar* to those in the target domain, we propose a Text Similarity Critic (SC) to classify $(\mathbf{y}_{(1)}, \mathbf{y}_{(2)})$ as 1-similar or 0-unsimilar text style. The model SC consists of two parts: a shared BiLSTM $\mathbf{h}_Y$ with the Variational Neural Encoder to represent the $\mathbf{y}_{(1)}$ sentence, and a second BiLSTM to encode the $\mathbf{y}_{(2)}$ sentence. The SC model takes input as a pair $(\mathbf{y}_{(1)}, \mathbf{y}_{(2)})$ of ([*target*], *source*), ([*target*], *generated*), and ([*generated*], *source*). Note that we give priority to encoding the $\mathbf{y}_{(1)}$ sentence in [.] using the shared BiLSTM, which guides the model to learn the sentence style from the target domain, and also contributes the target domain information into the global latent variables. We further utilize Siamese recurrent architectures (Neculoiu et al., 2016) for learning sentence similarity, in which the architecture allows us to learn useful representations with limited supervision.

Domain critic

In consideration of the shift between domains, we introduce a Domain Critic (DC) to classify sentence as *source*, *target*, or *generated* domain, respectively. Drawing inspiration from work of (Ganin et al., 2016), we model DC with a gradient reversal layer and two standard feed-forward layers. It is important to notice that our DC model shares parameters with the Variational Neural Encoder and the Variational Neural Inferer. The DC model takes input as a pair of given DA and corresponding utterance to produce a concatenation of both its representation and its latent variable in the output space, which is then passed through a feed-forward layer and a 3-labels classifier. In addition, the gradient reversal layer, which multiplies the gradient by a specific negative value during back-propagation training, ensures that the feature distributions over the two domains are made similar, as indistinguishable as possible for the domain critic, hence resulting in the domain-invariant features.

4 Training Domain Adaptation Model

Given a training instance represented by a pair of DA and sentence $(\mathbf{d}^{(i)}, \mathbf{y}^{(i)})$ from the rich source domain $\mathcal{S}$ and the limited target domain $\mathcal{T}$, the task aims at finding a set of parameters $\Theta_{\mathcal{T}}$ that can perform acceptably well on the target domain.

4.1 Training Critics

We provide as following the training objective of SC and DC. For SC, the goal is to classify a sentence pair into 1-*similar* or 0-*unsimilar* textual style. This procedure can be formulated as a supervised classification training objective function:

$$\mathcal{L}_s(\psi) = -\sum_{n=1}^{N} \log C_s(l_s^n | \mathbf{y}_{(1)}^n, \mathbf{y}_{(2)}^n, \psi), l_s^n = \begin{cases} 1 - similar & \text{if } (\mathbf{y}_{(1)}^n, \mathbf{y}_{(2)}^n) \in \mathcal{P}_{sim}, \\ 0 - unsimilar & \text{if } (\mathbf{y}_{(1)}^n, \mathbf{y}_{(2)}^n) \in \mathcal{P}_{unsim}, \end{cases} \quad (11)$$
$$\mathcal{Y}_{\mathcal{G}} = \{\mathbf{y} | \mathbf{y} \sim \mathcal{G}(.|\mathbf{d}_{\mathcal{T}}, .)\}, \mathcal{P}_{sim} = \{\mathbf{y}_{\mathcal{T}}^n, \mathbf{y}_{\mathcal{Y}_{\mathcal{G}}}^n\}, \mathcal{P}_{unsim} = (\{\mathbf{y}_{\mathcal{T}}^n, \mathbf{y}_{\mathcal{S}}^n\}, \{\mathbf{y}_{\mathcal{Y}_{\mathcal{G}}}^n, \mathbf{y}_{\mathcal{S}}^n\})$$

where: N is number of sentences, ψ is the model parameters of SC, $\mathcal{Y}_{\mathcal{G}}$ denotes sentences generated from the current generator $\mathcal{G}$ given target domain dialogue act $\mathbf{d}_{\mathcal{T}}$. The scalar probability $C_s(1|\mathbf{y}_{\mathcal{T}}^n, \mathbf{y}_{\mathcal{Y}_{\mathcal{G}}}^n)$ indicates how a generated sentence $\mathbf{y}_{\mathcal{Y}_{\mathcal{G}}}^n$ is relevant to a target sentence $\mathbf{y}_{\mathcal{T}}^n$.

The DC critic aims at classifying a pair of DA-utterance into *source*, *target*, or *generated* domain. This can also be formulated as a supervised classification training objective as follows:

$$\mathcal{L}_d(\varphi) = -\sum_{n=1}^{N} \log C_d(l_d^n | \mathbf{d}^n, \mathbf{y}^n, \varphi), l_d^n = \begin{cases} source & \text{if } (\mathbf{d}^n, \mathbf{y}^n) \in (\mathcal{D}_{\mathcal{S}}, \mathcal{Y}_{\mathcal{S}}), \\ target & \text{if } (\mathbf{d}^n, \mathbf{y}^n) \in (\mathcal{D}_{\mathcal{T}}, \mathcal{Y}_{\mathcal{T}}), \\ generated & \text{if } (\mathbf{d}^n, \mathbf{y}^n) \in (\mathcal{D}_{\mathcal{T}}, \mathcal{Y}_{\mathcal{G}}), \end{cases} \quad (12)$$

where: φ is the model parameters of DC, $(\mathcal{D}_S, \mathcal{Y}_S)$, $(\mathcal{D}_T, \mathcal{Y}_T)$ are the DA-utterance pairs from source, target domain, respectively. Note also that the scalar probability $C_d(target|\mathbf{d}^n, \mathbf{y}^n)$ indicates how likely the DA-utterance pair $(\mathbf{d}^n, \mathbf{y}^n)$ is from the target domain.

4.2 Training Variational Generator

We utilize the Monte Carlo method to approximate the expectation over the posterior in Eq. 2, *i.e.*, $\mathbb{E}_{q_\phi(z|\mathbf{d},\mathbf{y})}[.] \simeq \frac{1}{M}\sum_{m=1}^{M} \log p_\theta(\mathbf{y}|\mathbf{d}, \mathbf{h}_z^{(m)})$ where: M is the number of samples. In this study, the joint training objective for a training instance $(\mathbf{d}, \mathbf{y})$ is formulated as follows:

$$\mathcal{L}(\theta, \phi) \simeq -KL(q_\phi(z|\mathbf{d}, \mathbf{y})||p_\theta(z|\mathbf{d})) + \frac{1}{M}\sum_{m=1}^{M}\sum_{t=1}^{T_y} \log p_\theta(\mathbf{y}_t|\mathbf{y}_{<t}, \mathbf{d}, \mathbf{h}_z^{(m)}) \tag{13}$$

where: $\mathbf{h}_z^{(m)} = \mu + \sigma \odot \epsilon^{(m)}$, and $\epsilon^{(m)} \sim \mathcal{N}(0, \mathbf{I})$. The first term is the KL divergence between two Gaussian distribution, and the second term is the approximation expectation. We simply set $M = 1$ which degenerates the second term to the objective of conventional generator. Since the objective function in Eq. 13 is differentiable, we can jointly optimize the parameter θ and variational parameter ϕ using standard gradient ascent techniques.

4.3 Adversarial Training

Our domain adaptation architecture is demonstrated in Figure 1, in which both generator $\mathcal{G}$ and critics C_s, and C_d jointly train by pursuing competing goals as follows. Given a dialogue act $\mathbf{d}_\mathcal{T}$ in the target domain, the generator generates K sentences $\mathbf{y}$'s. It would prefer a "*good*" generated sentence $\mathbf{y}$ if the values of $C_d(target|\mathbf{d}_\mathcal{T}, \mathbf{y})$ and $C_s(1|\mathbf{y}_\mathcal{T}, \mathbf{y})$ are large. In contrast, the critics would prefer large values of $C_d(generated|\mathbf{d}_\mathcal{T}, \mathbf{y})$ and $C_s(1|\mathbf{y}, \mathbf{y}_\mathcal{S})$, which imply the small values of $C_d(target|\mathbf{d}_\mathcal{T}, \mathbf{y})$ and $C_s(1|\mathbf{y}_\mathcal{T}, \mathbf{y})$. We propose a domain-adversarial training procedure in order to iteratively updating the generator and critics as described in Algorithm 1. While the parameters of generator are optimized to minimize their loss in the training set, the parameters of the critics are optimized to minimize the error of text similarity, and to maximize the loss of domain classifier.

Algorithm 1: Adversarial Training Procedure

Require: generator $\mathcal{G}$, domain critic C_d, text similarity critic C_s, generated sentence $\mathcal{Y}_\mathcal{G} = \emptyset$;
Input: DA-utterance pairs of source $(\mathcal{D}_\mathcal{S}, \mathcal{Y}_\mathcal{S})$, target $(\mathcal{D}_\mathcal{T}, \mathcal{Y}_\mathcal{T})$;
1 Pretrain $\mathcal{G}$ on $(\mathcal{D}_\mathcal{S}, \mathcal{Y}_\mathcal{S})$ using VRALSTM;
2 **while** Θ *has not converged* **do**
3 **for** *i = 0, .., $N_\mathcal{T}$* **do**
4 Sample $(\mathbf{d}_\mathcal{S}, \mathbf{y}_\mathcal{S})$ from source domain;
5 (D_1)-Compute $g_d = \nabla_\varphi \mathcal{L}_d(\varphi)$ using Eq. 12 for $(\mathbf{d}_\mathcal{S}, \mathbf{y}_\mathcal{S})$ and $(\mathbf{d}_\mathcal{T}, \mathbf{y}_\mathcal{T})$;
6 (D_2)-Adam update of φ for C_d using g_d;
7 (G_1)-Compute $g_\mathcal{G} = \{\nabla_\theta \mathcal{L}(\theta, \phi), \nabla_\phi \mathcal{L}(\theta, \phi)\}$ using Eq. 13
8 (G_2)-Adam update of θ, ϕ for $\mathcal{G}$ using $g_\mathcal{G}$
9 (S_1)-Compute $g_s = \nabla_\psi \mathcal{L}_s(\psi)$ using Eq. 11 for $(\mathbf{y}_\mathcal{T}, \mathbf{y}_\mathcal{S})$;
10 (S_2)-Adam update of ψ for C_s using g_s;
11 $\mathcal{Y}_\mathcal{G} \leftarrow \{\mathbf{y}_{\bar{k}}\}_{\bar{k}=1}^{K}$, where $\mathbf{y}_{\bar{k}} \sim \mathcal{G}(.|\mathbf{d}_\mathcal{T}^{(i)}, .)$;
12 Choose top ***k*** best sentences of $\mathcal{Y}_\mathcal{G}$;
13 **for** *j = 1,..,**k*** **do**
14 (D_1), (D_2) steps for C_d with $(\mathbf{d}_\mathcal{T}, \mathbf{y}_\mathcal{G}^{(j)})$;
15 (S_1), (S_2) steps for C_s with $(\mathbf{y}_\mathcal{G}^{(j)}, \mathbf{y}_\mathcal{S})$ and $(\mathbf{y}_\mathcal{T}, \mathbf{y}_\mathcal{G}^{(j)})$;
16 **end**
17 **end**
18 **end**

Generally, the current generator $\mathcal{G}$ for each training iteration i takes a *target* dialogue act $\mathbf{d}_\mathcal{T}^{(i)}$ as input to over-generate a set $\mathcal{Y}_\mathcal{G}$ of K candidate sentences (step 11). We then choose top **k** best sentences in the $\mathcal{Y}_\mathcal{G}$ set (step 12) after re-ranking to measure how "*good*" the generated sentences are by using the critics (steps 14-15). These "*good*" signals from the critics can guide the generator step by step to generate the outputs which resemble the sentences drawn from the target domain. Note that the re-ranking step is important for separating the "*correct*" sentences from the current generated outputs $\mathcal{Y}_\mathcal{G}$ by penalizing the generated sentences which have redundant or missing slots.

5 Experiments

We conducted experiments on the proposed models in different scenarios: ***Adaptation***, ***Scratch***, and ***All*** using several model architectures, evaluation metrics, datasets (Wen et al., 2016a), and configurations (see Appendix A).

KL cost annealing strategy (Bowman et al., 2015) encourages the model to encode meaningful representations into the latent vector z, in which we gradually anneal the KL term from 0 to 1. This helps our model to achieve solutions with non-zero KL term.

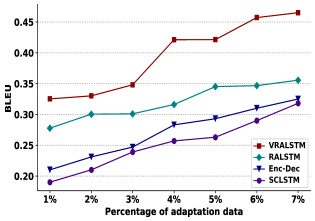
Gradient reversal layer (Ganin et al., 2016) leaves the input unchanged during forward propagation and reverses the gradient by multiplying it with a negative scalar $-\lambda_p$ during the backpropagation-based training. We set the domain adaptation parameter λ_p which gradually increases, starting from 0 to 1, by using the following schedule for each training step i: $p = float(i)/num\_steps$, and $\lambda_p = \frac{2}{1+exp(-10*p)} - 1$ where: num_steps is a constant which is set to be 8600, p is training progress. This strategy allows the Domain critic to be less sensitive to noisy signal at the early training stages.

6 Results and Analysis

6.1 Integrating Variational Inference

We compare the original model RALSTM with its modification by integrating Variational Inference (VRALSTM) as demonstrated in Table 2 and Table 1-(a). It clearly shows that the VRALSTM not only preserves the power of the original RALSTM on generation task since its performances are very competitive to those of RALSTM, but also provides a compelling evidence on adapting to a new, unseen domain when the target domain data is scarce, *i.e.*, from 1% to 7%. Table 3, *sec.* 3 further shows the necessity of the integrating in which the VRALSTM achieved a significant improvement over the RALSTM in ***Scratch*** scenario, and of the adversarial domain adaptation algorithm in which although both the RALSTM and VRALSTM model can perform well when providing sufficient in-domain training data (Table 2), the performances are extremely impaired when training from ***Scratch*** with only a limited data. These indicate that the proposed variational method can learn the underlying semantic of DA-utterance pairs in the source domain via the representation of the latent variable z, from which when adapting to another domain, the models can leverage the existing knowledge to guide the generation process.

Table 1: Results when adapting models trained on (a) union, and (b) counterfeting dataset.



(a) Result on **Laptop** when adapting models trained on [Restaurant+Hotel] data.

Source \ **Target**(*Test*)	**R2H**(*Hotel*)		**H2R**(*Restaurant*)		**L2T**(*Tv*)		**T2L**(*Laptop*)	
	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR
Hotel$^\sharp$	-	-	0.5931	12.50%	0.4183	2.38%	0.3426	13.02%
Restaurant$^\sharp$	0.6224	1.99%	-	-	0.4211	2.74%	0.3540	13.13%
Tv$^\sharp$	0.6153	4.30%	0.5835	14.49%	-	-	0.3630	7.44%
Laptop$^\sharp$	0.6042	5.22%	0.5598	15.61%	0.4268	1.05%	-	-

(b) Results evaluated on (*Test*) domains by ***Unsupervised*** adapting VDANLG from Source domains using only **10%** of the **Target** domain **Counterfeit X2Y**. {**X**,**Y**}=**R** : Restaurant, **H** : Hotel, **T** : Tv, **L** : Laptop.

Table 2: Results evaluated on **Target** domains by training models from scratch with ***All*** in-domain data.

Model \ **Target**	**Hotel**		**Restaurant**		**Tv**		**Laptop**	
	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR
HLSTM (Wen et al., 2015a)	0.8488	2.79%	0.7436	0.85%	0.5240	2.65%	0.5130	1.15%
SCLSTM (Wen et al., 2015b)	0.8469	3.12%	0.7543	0.57%	0.5235	2.41%	0.5109	0.89%
Enc-Dec (Wen et al., 2016b)	0.8537	4.78%	0.7358	2.98%	0.5142	3.38%	0.5101	4.24%
RALSTM (Tran and Nguyen, 2017a)	0.8911	0.48%	0.7739	0.19%	0.5376	0.65%	0.5222	0.49%
VRALSTM (Ours)	0.8851	0.57%	0.7709	0.36%	0.5356	0.73%	0.5210	0.59%

Table 3: Ablation studies' results evaluated on **Target** domains by *adaptation* training proposed models from Source domains using only *10%* amount of the **Target** domain data (*sec.* 1, 2, 4, 5). The results were averaged over 5 randomly initialized networks.

	Target / Source	Hotel		Restaurant		Tv		Laptop	
		BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR
no Critics	Hotel	-	-	0.6814	11.62%	0.4968	12.19%	0.4915	3.26%
	Restaurant	0.7983	8.59%	-	-	0.4805	13.70%	0.4829	9.58%
	Tv	0.7925	12.76%	0.6840	8.16%	-	-	0.4997	4.79%
	Laptop	0.7870	15.17%	0.6859	7.55%	0.4953	18.60%	-	-
	[R+H]	-	-	-	-	0.5019	7.43%	0.4977	5.96%
	[L+T]	0.7935	11.71%	0.6927	6.49%	-	-	-	-
+ DC + SC	Hotel	-	-	0.7131	2.53%	0.5164	3.25%	0.5007	1.68%
	Restaurant	0.8217	3.95%	-	-	0.5043	2.99%	0.4931	2.77%
	Tv	0.8251	4.89%	0.6971	4.62%	-	-	0.5009	2.10%
	Laptop	0.8218	2.89%	0.6926	2.87%	0.5243	1.52%	-	-
	[R+H]	-	-	-	-	0.5197	2.58%	0.5009	1.61%
	[L+T]	0.8252	2.87%	0.7066	3.73%	-	-	-	-
scr10	RALSTM	0.6855	22.53%	0.6003	17.65%	0.4009	22.37%	0.4475	24.47%
	VRALSTM	0.7378	15.43%	0.6417	15.69%	0.4392	17.45%	0.4851	10.06%
+ DC only	Hotel	-	-	0.6823	4.97%	0.4322	27.65%	0.4389	26.31%
	Restaurant	0.8031	6.71%	-	-	0.4169	34.74%	0.4245	26.71%
	Tv	0.7494	14.62%	0.6430	14.89%	-	-	0.5001	15.40%
	Laptop	0.7418	19.38%	0.6763	9.15%	0.5114	10.07%	-	-
	[R+H]	-	-	-	-	0.4257	31.02%	0.4331	31.26%
	[L+T]	0.7658	8.96%	0.6831	11.45%	-	-	-	-
+ SC only	Hotel	-	-	0.6976	5.00%	0.4896	9.50%	0.4919	9.20%
	Restaurant	0.7960	4.24%	-	-	0.4874	12.26%	0.4958	5.61%
	Tv	0.7779	10.75%	0.7134	5.59%	-	-	0.4913	13.07%
	Laptop	0.7882	8.08%	0.6903	11.56%	0.4963	7.71%	-	-
	[R+H]	-	-	-	-	0.4950	8.96%	0.5002	5.56%
	[L+T]	0.7588	9.53%	0.6940	10.52%	-	-	-	-

sec. 3: Training RALSTM and VRALSTM models from *scratch* using 10% of **Target** domain data;

6.2 Ablation Studies

The ablation studies (Table 3, *sec.* 1, 2) demonstrate the contribution of two Critics, in which the models were assessed with either no Critics or both or only one. It clearly sees that combining both Critics makes a substantial contribution to increasing the BLEU score and decreasing the slot error rate by a large margin in every dataset pairs. A comparison of model adapting from source Laptop domain between VRALSTM without Critics (Laptop$^\flat$) and VDANLG (Laptop$^\sharp$) evaluated on the target **Hotel** domain shows that the VDANLG not only has better performance with much higher the BLEU score, 82.18 in comparison to 78.70, but also significantly reduce the ERR, from 15.17% down to 2.89%. The trend is consistent across all the other domain pairs. These stipulate the necessary Critics in effective learning to adapt to a new domain.

Table 3, *sec.* 4 further demonstrates that using DC only (*sec.* 4) brings a benefit of effectively utilizing similar slot-value pairs seen in the training data to *closer* domain pairs such as: Hotel→Restaurant (68.23 BLEU, 4.97 ERR), Restaurant→Hotel (80.31 BLEU, 6.71 ERR), Laptop→Tv (51.14 BLEU, 10.07 ERR), and Tv→Laptop (50.01 BLEU, 15.40 ERR) pairs. Whereas it is inefficient for the *longer* domain pairs since their performances are worse than those without Critics, or in some cases even worse than the VRALSTM in ***scr10*** scenario, such as Restaurant→Tv (41.69 BLEU, 34.74 ERR), and the cases where Laptop to be a Target domain. On the other hand, using only SC (*sec.* 5) helps the models achieve better results since it is aware of the sentence style when adapting to the target domain.

6.3 Distance of Dataset Pairs

To better understand the effectiveness of the methods, we analyze the learning behavior of the proposed model between different dataset pairs. The datasets' order of difficulty was, from easiest to hardest:

Hotel↔Restaurant↔Tv↔Laptop. On the one hand, it might be said that the *longer* datasets' distance is, the more difficult of domain adaptation task becomes. This clearly shows in Table 3, *sec.* 1, at **Hotel** column where the adaptation ability gets worse regarding decreasing the BLEU score and increasing the ERR score alongside the order of Restaurant→Tv→Laptop datasets. On the other hand, the *closer* the dataset pair is, the faster model can adapt. It can be expected that the model can better adapt to the target **Tv/Laptop** domain from source Laptop/Tv than those from source Restaurant, Hotel, and vice versa, the model can easier adapt to the target **Restaurant/Hotel** domain from source Hotel/Restaurant than those from Laptop, Tv. However, the above-mentioned is not always true that the proposed method can perform acceptably well from *easy* source domains (Hotel, Restaurant) to the more *difficult* target domains (Tv, Laptop) and vice versa (Table 3, *sec.* 1, 2).

Table 3, *sec.* 2 further shows that the proposed method is able to leverage the out of domain knowledge since the adaptation models trained on union source dataset, such as [R+H] or [L+T], show better performances than those trained on individual source domain data. A specific example in Table 3, *sec.* 2 shows that the adaptation VDANLG model trained on the source union dataset of Laptop and Tv ([L+T]$^\sharp$) has better performance, at 82.52 BLEU and 2.87 ERR, than those models trained on the individual source dataset, such as Laptop$^\sharp$ (82.18 BLEU, 2.89 ERR), and Tv$^\sharp$ (82.51 BLEU, 4.89 ERR). Another example in Table 3, *sec.* 2 also shows that the adaptation VDANLG model trained on the source union dataset of Restaurant and Hotel ([R+H]$^\sharp$) has better results, at 51.97 BLEU and 2.58 ERR, than those models trained on the separate source dataset, such as Restaurant$^\sharp$ (50.43 BLEU, 2.99 ERR), and Hotel$^\sharp$ (51.64 BLEU, 3.25 ERR). The trend is mostly consistent across all other domain comparisons in different training scenarios. All these demonstrate that the proposed model can learn global semantics that can be efficiently transferred into new domains.

6.4 *Adaptation* vs. *All* Training Scenario

It is interesting to compare ***Adaptation*** (Table 3, *sec.* 2) with ***All*** training scenario (Table 2). The VDANLG model shows its considerable ability to shift to another domain with a limited of in-domain labels whose results are competitive to or in some cases better than the previous models trained on full labels of the **Target** domain. A specific comparison evaluated on the **Tv** domain where the VDANLG model trained on the source Laptop$^\sharp$ achieved better performance, at 52.43 BLEU and 1.52 ERR, than HLSTM (52.40, 2.65), SCLSTM (52.35, 2.41), and Enc-Dec (51.42, 3.38). The VDANLG models, in many cases, also have lower of slot error rate ERR scores than the Enc-Dec model. These indicate the stable strength of the VDANLG models in adapting to a new domain when the target domain data is scarce.

6.5 Unsupervised Domain Adaptation

We further examine the effectiveness of the proposed methods by training the VDANLG models on target ***Counterfeit*** datasets (Wen et al., 2016a). The promising results are shown in Table 1-(b), despite the fact that the models were instead adaptation trained on the **Counterfeit** datasets, or in other words, were indirectly trained on the (*Test*) domains. However, the proposed models still showed positive signs in remarkably reducing the slot error rate ERR in the cases of *Hotel* and *Tv* be the (*Test*) domains. Surprisingly, even the source domains (Hotel$^\sharp$/Restaurant$^\sharp$) are far from the (*Test*) domain *Tv*, and the **Target** domain **Counterfeit L2T** is also very different to the source domains, the model can still acceptably adapt well since its BLEU scores on (*Test*) Tv domain reached to (41.83/42.11), and it also produced a very low of ERR scores (2.38/2.74). This phenomenon will be further investigated in the unsupervised scenario in the future work.

6.6 Comparison on Generated Outputs

On the one hand, the VRALSTM models (trained from ***Scratch*** or trained adapting model from Source domains) produce the outputs with a diverse range of error types, including missing, misplaced, redundant, wrong slots, or even spelling mistake information, leading to a very high of the slot error rate ERR score. Specifically, the VRALSTM from ***Scratch*** tends to make repeated slots and also many of the missing slots in the generated outputs since the training data may inadequate for the model to generally

Table 4: Comparison of top **Laptop** responses generated for different scenarios by adaptation training VRALSTM (denoted by ♭) and VDANLG (denoted by ♯) models from Source domains, and by training VRALSTM from ***Scratch***. Errors are marked in colors ([missing], misplaced, redundant, wrong, spelling mistake information). [OK] denotes successful generation. VDANLG$^{\sharp}$ = VRALSTM$^{\flat}$+SC+DC.

Model	Generated Responses from Laptop Domain
DA 1	compare(name='*tecra erebus 20*'; memory='*4 gb*'; isforbusinesscomputing='*true*'; name='*satellite heracles 45*'; memory='*2 gb*'; isforbusinesscomputing='*false*')
Reference 1	compared to *tecra erebus 20* which has a *4 gb* memory and *is for business computing* , *satellite heracles 45* has a *2 gb* memory and *is not for business computing* . which one do you prefer
VRALSTM	which would be the *tecra erebus 20 is a business computing* laptop with *4 gb* of memory and is the SLOT_NAME , and is not for business computing . [satellite heracles 45][2 gb]
Hotel$^{\flat}$	*the tecra erebus 20 is used for business computing* . the *satellite heracles 45* has 4 gb of memory and a SLOT_BATTERY battery life for business computing . which one do you want
Restaurant$^{\flat}$	the *tecra erebus 20 is for business computing* . the *satellite heracles 45* which has 4 gb of memory and *is not for business computing* . which one do you want [2 gb]
Tv$^{\flat}$	*the tecra erebus 20* has *4 gb* of memory and is not for business computing . which one do you prefer [is for business computing][satellite heracles 45][2 gb]
[R+H]$^{\flat}$	the *tecra erebus 20* is not for business computing . which one do you want a business computing . which one do you prefer [4 gb][is for business computing][satellite heracles 45][2 gb]
Hotel$^{\sharp}$	the *tecra erebus 20* has a *4 gb* memory , that *is for business computing* . the *satellite heracles 45* with *2 gb* of memory and *is not for business computing* . which one do you want [OK]
Restaurant$^{\sharp}$	the *tecra erebus 20* has a *4 gb* memory , and *is for business computing* . the *satellite heracles 45 is not for business computing* . which one do you want to know more [2 gb]
Tv$^{\sharp}$	the *tecra erebus 20 is a business computing* . the *satellite heracles 45* has a 4 gb memory and *is not for business computing* . which one do you prefer [2 gb]
[R+H]$^{\sharp}$	the *tecra erebus 20 is for business computing* , has a 2 gb of memory. the *satellite heracles 45* has 4 gb of memory , *is not for business computing*. which one do you want

handle the unseen dialog acts. Whereas the VRALSTM models without Critics adapting trained from Source domains (denoted by ♭ in Table 4 and Appendix B. Table 5) tend to generate the outputs with fewer error types than the model from ***Scratch*** because the VRALSTM$^{\flat}$ models may capture the overlap slots of both source and target domain during adaptation training.

On the other hand, under the guidance of the Critics (SC and DC) in an adversarial training procedure, the VDANLG model (denoted by ♯) can effectively leverage the existing knowledge of the source domains to better adapt to the target domains. The VDANLG models can generate the outputs in style of the target domain with much fewer the error types compared with the two above models. Moreover, the VDANLG models seem to produce satisfactory utterances with more correct generated slots. For example, a sample outputted by the [R+H]$^{\sharp}$ in Table 4-example 1 contains all the required slots with only a misplaced information of two slots *2 gb* and *4 gb*, while the generated output produced by Hotel$^{\sharp}$ is a successful generation. Another samples in Appendix B. Table 5 generated by the Hotel$^{\sharp}$, Tv$^{\sharp}$, [R+H]$^{\sharp}$ (in DA 2) and Laptop$^{\sharp}$ (DA 3) models are all fulfilled responses. An analysis of the generated responses in Table 5-example 2 illustrates that the VDANLG models seem to generate a concise response since the models show a tendency to form some potential slots into a concise phrase, *i.e.*, "SLOT_NAME SLOT_TYPE". For example, the VDANLG models tend to concisely response as "the *portege phosphorus 43 laptop* ..." instead of "the *portege phosphorus 43* is a *laptop* ...". All these above demonstrate that the VDANLG models have ability to produce better results with a much lower of the slot error rate ERR score.

7 Conclusion and Future Work

We have presented an integrating of a variational generator and two Critics in an adversarial training algorithm to examine the model ability in domain adaptation task. Experiments show that the proposed models can perform acceptably well in a new, unseen domain by using a limited amount of in-domain data. The ablation studies also demonstrate that the variational generator contributes to effectively learn the underlying semantic of DA-utterance pairs, while the Critics show its important role of guiding the model to adapt to a new domain. The proposed models further show a positive sign in unsupervised domain adaptation, which would be a worthwhile study in the future.

Acknowledgements

This work was supported by the JST CREST Grant Number JPMJCR1513, the JSPS KAKENHI Grant number 15K16048 and the SIS project.

References

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Xie Chen, Tian Tan, Xunying Liu, Pierre Lanchantin, Moquan Wan, Mark JF Gales, and Philip C Woodland. 2015. Recurrent neural network language model adaptation for multi-genre broadcast speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2016. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Esther Levin, Shrikanth Narayanan, Roberto Pieraccini, Konstantin Biatov, Enrico Bocchieri, Giuseppe Di Fabbrizio, Wieland Eckert, Sungbok Lee, A Pokrovsky, Mazin Rahim, et al. 2000. The at&t-darpa communicator mixed-initiative spoken dialog system. In *Sixth International Conference on Spoken Language Processing*.

François Mairesse and Marilyn A. Walker. 2011. Controlling user perceptions of linguistic style: Trainable generation of personality traits. *Comput. Linguist.*, 37(3):455–488, September.

François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. 2010. Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1552–1561, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International Conference on Machine Learning*, pages 1727–1736.

Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. *arXiv preprint arXiv:1506.07190*.

Paul Neculoiu, Maarten Versteegh, Mihai Rotaru, and Textkernel BV Amsterdam. 2016. Learning text similarity with siamese recurrent networks. *ACL 2016*, page 148.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th ACL*, pages 311–318. Association for Computational Linguistics.

Sanjay Purushotham, Wilka Carvalho, Tanachat Nilanon, and Yan Liu. 2017. Variational adversarial deep domain adaptation for health care time series analysis.

Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues.

Yangyang Shi, Martha Larson, and Catholijn M Jonker. 2015. Recurrent neural network language model adaptation with curriculum learning. *Computer Speech & Language*, 33(1):136–154.

Van-Khanh Tran and Le-Minh Nguyen. 2017a. Natural language generation for spoken dialogue system using rnn encoder-decoder networks. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 442–451, Vancouver, Canada, August. Association for Computational Linguistics.

Van-Khanh Tran and Le-Minh Nguyen. 2017b. Semantic refinement gru-based neural language generation for spoken dialogue systems. *arXiv preprint arXiv:1706.00134*.

Van-Khanh Tran, Le-Minh Nguyen, and Satoshi Tojo. 2017. Neural-based natural language generation in dialogue using rnn encoder-decoder with semantic aggregation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 231–240, Saarbrcken, Germany, August. Association for Computational Linguistics.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. *arXiv preprint arXiv:1702.05464*.

Marilyn A Walker, Owen Rambow, and Monica Rogati. 2001. Spot: A trainable sentence planner. In *Proceedings of the 2nd NAACL*, pages 1–8. Association for Computational Linguistics.

Marilyn A Walker, Amanda Stent, François Mairesse, and Rashmi Prasad. 2007. Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research*, 30:413–456.

Tsung-Hsien Wen, Milica Gašić, Dongho Kim, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015a. Stochastic Language Generation in Dialogue using Recurrent Neural Networks with Convolutional Sentence Reranking. In *Proceedings SIGDIAL*. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015b. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016a. Multi-domain neural network language generation for spoken dialogue systems. *arXiv preprint arXiv:1603.01232*.

Tsung-Hsien Wen, Milica Gašic, Nikola Mrkšic, Lina M Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016b. Toward multi-domain language generation using recurrent neural networks.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017. Latent intention dialogue models. *arXiv preprint arXiv:1705.10229*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

B. Zhang, D. Xiong, J. Su, H. Duan, and M. Zhang. 2016. Variational Neural Machine Translation. *ArXiv e-prints*, May.

Appendix A. Experimental Setups

We followed the configurations for the RALSTM model from work of (Tran and Nguyen, 2017a), in which the hidden layer size and beam width were set to be 80 and 10, respectively, and the generators were trained with a 70% of keep dropout rate. We performed 5 runs with different random initialization of the network, and the training process is terminated by using early stopping. We then selected a model that yields the highest BLEU score (Papineni et al., 2002) on the validation set. For the variational inference, we set the latent variable size to be 16. We used Adam optimizer with the learning rate is initially set to be 0.001, and after 3 epochs for the Union dataset and 5 epochs for the single dataset the learning rate is decayed every epoch using an exponential rate of 0.95. The decoding phase is mostly similar to those in work of (Tran and Nguyen, 2017a).

Appendix B. Generated Utterances

Table 5 presents top responses generated for different scenarios from Laptop and TV domains.

Table 5: Comparison of top responses generated for different scenarios by adaptation training VRALSTM (denoted by ♭) and VDANLG (denoted by ♯) models from Source domains, and by training VRALSTM from ***Scratch***. Errors are marked in colors ([missing], misplaced, redundant, wrong, spelling mistake information). [OK] denotes successful generation. VDANLG$^{\sharp}$ = VRALSTM$^{\flat}$+SC+DC.

Model	**Generated Responses from Laptop Domain**
DA 2	inform(name='*portege phosphorus 43*'; type='*laptop*'; design='*black magnesium chassis with brushed metallic finish , matt black keyboard*'; isforbusinesscomputing='*false*'; drive='*320 gb*')
Reference 2	the *portege phosphorus 43 laptop* has a *320 gb* drive , *is not for business computing* and *has a black magnesium chassis with brushed metallic finish , matt black keyboard*
VRALSTM	the *portege phosphorus 43* is a *laptop* with a *320 gb* drive and *has a black magnesium chassis with brushed metallic finish , matt black keyboard* . [is not for business computing]
Hotel$^{\flat}$	the *portege phosphorus 43* is a *laptop* has a *320 gb* drive , *is not for business computing* . it is not for business computing , it has a design of *black magnesium chassis with brushed metallic finish , matt black keyboard*
Restaurant$^{\flat}$	the *portege phosphorus 43* is a *laptop* with a *320 gb* drive , has a design of *black magnesium chassis with brushed metallic finish , matt black keyboard* . [is not for business computing]
Tv$^{\flat}$	the *portege phosphorus 43* is a *laptop* with a *black magnesium chassis with brushed metallic finish , matt black keyboard* . it is not for business computing [320 gb]
[R+H]$^{\flat}$	the *portege phosphorus 43* is a *laptop* with a *black magnesium chassis with brushed metallic finish , matt black keyboard* [is not used for business computing] [320 gb]
Hotel$^{\sharp}$	the *portege phosphorus 43 laptop* has a *320 gb* drive , has a *black magnesium chassis with brushed metallic finish , matt black keyboard* design and *is not for business computing* [OK]
Restaurant$^{\sharp}$	*the portege phosphorus 43 laptop* has a *320 gb* drive , it is for business computing , it has a design of *black magnesium chassis with brushed metallic finish , matt black keyboard*
Tv$^{\sharp}$	*the portege phosphorus 43 laptop* has a *320 gb* drive and a design of *black magnesium chassis with brushed metallic finish , matt black keyboard* . it *is not for business computing* [OK]
[R+H]$^{\sharp}$	the *portege phosphorus 43 laptop* has a *320 gb* drive , and *is not for business computing* . it has a *black magnesium chassis with brushed metallic finish , matt black keyboard* [OK]

Model	**Generated Responses from TV Domain**
DA 3	compare(name='*crios 69*'; ecorating='*a++*'; powerconsumption='*44 watt*'; name='*dinlas 61*'; ecorating='*a+*'; powerconsumption='*62 watt*')
Reference 3	compared to *crios 69* which is in the *a++* eco rating and has *44 watt* power consumption , *dinlas 61* is in the *a+* eco rating and has *62 watt* power consumption . which one do you prefer ?
VRALSTM	the *crios 69* is the dinlas 61 is the SLOT_NAME is the SLOT_NAME is the SLOT_NAME is the SLOT_NAME is the SLOT_NAME is the SLOT_NAME . it has an a++ eco rating [44 watt][a+][62 watt]
Hotel$^{\flat}$	the *crios 69* has a *44 watt* power consumption , whereas the *dinlas 61* has *62 watt* power consumption , whereas the SLOT_NAME has SLOT_POWERCONSUMPTION power consumption and has an a++ eco rating [a+]
Restaurant$^{\flat}$	the *crios 69* has a *a++* eco rating , *44 watt* power consumption , and an a+ eco rating and 62 watt power consumption [dinlas 61]
Laptop$^{\flat}$	the *crios 69* has SLOT_HDMIPORT hdmi port -s , the *dinlas 61* has a++ eco rating and 44 watt power consumption [62 watt][a+]
[R+H]$^{\flat}$	the *crios 69* is in the SLOT_FAMILY product family with a++ eco rating ? [44 watt][dinlas 61][62 watt][a+]
Hotel$^{\sharp}$	the *crios 69* has an *a++* eco rating and *44 watt* power consumption and a 62 watt power consumption [dinlas 61][a+]
Restaurant$^{\sharp}$	the *crios 69* has *44 watt* power consumption of *a++* and has an a+ eco rating and 62 watt power consumption [dinlas 61]
Laptop$^{\sharp}$	the *crios 69* has an *a++* eco rating and *44 watt* power consumption , whereas the *dinlas 61* has *62 watt* power consumption and *a+* eco rating . [OK]
[R+H]$^{\sharp}$	the *crios 69* has *44 watt* power consumption , and an *a++* eco rating and the *dinlas 61* has a *62 watt* power consumption . [a+]

Ask No More: Deciding when to guess in referential visual dialogue

Ravi Shekhar[†], **Tim Baumgärtner**[*], **Aashish Venkatesh**[*],
Elia Bruni[*], **Raffaella Bernardi**[†] and **Raquel Fernandez**[*]
[*]University of Amsterdam, [†]University of Trento
raquel.fernandez@uva.nl raffaella.bernardi@unitn.it

Abstract

Our goal is to explore how the abilities brought in by a dialogue manager can be included in end-to-end visually grounded conversational agents. We make initial steps towards this general goal by augmenting a task-oriented visual dialogue model with a decision-making component that decides whether to ask a follow-up question to identify a target referent in an image, or to stop the conversation to make a guess. Our analyses show that adding a decision making component produces dialogues that are less repetitive and that include fewer unnecessary questions, thus potentially leading to more efficient and less unnatural interactions.

1 Introduction

The field of interactive conversational agents, also called dialogue systems, is receiving renewed attention not only within Computational Linguistics (CL) and Natural Language Processing (NLP) – its original and probably most natural locus – but also within the Machine Learning (ML) and the Computer Vision (CV) communities. The overarching challenge, in line with the long-term aims of Artificial Intelligence, is to develop data-driven agents that are capable of perceiving (and possibly acting upon) the external world and that we can collaborate with through natural language dialogue to achieve common goals.

Within the ML and CV communities, recent research on conversational agents combined with Deep Learning techniques has yielded interesting results on visually grounded tasks (Das et al., 2017a; de Vries et al., 2017; Mostafazadeh et al., 2017). In this line of research, the focus is mostly on improving model performance by investigating new machine learning paradigms (like reinforcement learning or adversarial learning) in end-to-end settings, where the model learns directly from raw data without symbolic annotations (Strub et al., 2017; Lu et al., 2017; Wu et al., 2017). Task accuracy, however, is not the only criterion by which a conversational agent should be judged. Crucially, the dialogue should be coherent, with no unnatural repetitions nor unnecessary questions — unlike the 5-turn dialogue shown in Figure 1. To achieve this, a conversational agent needs to learn a strategy to decide how to respond given the current context and the task at hand. These abilities are typically considered part of *dialogue management* and have been the focus of attention in dialogue systems research within the CL/NLP community (Larsson and Traum, 2000; Williams et al., 2008; Bohus and Rudnicky, 2009; Young et al., 2013).

Questioner	**Answerer**
1. Is it a person?	No
2. Is it the dog?	Yes
⇝ *success by our model*	
3. The dog?	Yes
4. Is it in the foreground?	No
5. Is it the whole dog?	Yes
⇝ *success by baseline model*	

Figure 1: Dialogue that leads to task success in the *GuessWhat?!* game by our model, which decides when to stop asking questions, and by the baseline model in de Vries et al. (2017), which does not.

In this paper, we thus take a step back: instead of focusing on learning paradigms, we focus on the system architecture. We argue that the time is ripe for exploring how the abilities brought in by a dia-

Data and code are available at https://vista-unitn-uva.github.io

Proceedings of the 27th International Conference on Computational Linguistics, pages 1218–1233
Santa Fe, New Mexico, USA, August 20-26, 2018.

logue manager can be included in end-to-end conversational agents within the Deep Learning paradigm mostly put forward by the ML and CV communities. We make initial steps towards this general goal by augmenting the task-oriented visual dialogue model proposed by de Vries et al. (2017), not yet with a full-fledged dialogue manager, but with a decision-making component that decides whether to ask a follow-up question to identify a target referent in an image, or to stop the conversation to make a guess (see Figure 1). Our focus is on providing a thorough analysis of the resulting dialogues. Our results show that the presence of a decision making component leads to dialogues that are less repetitive and that include fewer unnecessary questions.

2 Related Work

Our system operates on both linguistic and visual information. Visually-grounded dialogue has experienced a boost in recent years, in part thanks to the construction of large visual human-human dialogue datasets built by the Computer Vision community (Mostafazadeh et al., 2017; Das et al., 2017a; de Vries et al., 2017). These datasets include two participants, a Questioner and an Answerer, who ask and answer questions about an image. For example, in the *GuessWhat?!* dataset developed by de Vries et al. (2017), which we exploit in the present work, a Questioner agent needs to guess a target object in a visual scene by asking yes-no questions (more details are provided in the next section).

Research on visually-grounded dialogue within the Computer Vision community exploits encoder-decoder architectures (Sutskever et al., 2014) — which have shown some promise for modelling chatbot-style dialogue (Vinyals and Le, 2015; Sordoni et al., 2015; Serban et al., 2016; Li et al., 2016a; Li et al., 2016b) — augmented with visual features. This community has mostly focused on model learning paradigms. Initial models, proposed by de Vries et al. (2017) and Das et al. (2017a), use supervised learning (SL): the Questioner and the Answerer are trained to generate utterances (by word sampling) that are similar to the human gold standard. To account for the intuition that dialogues require some form of planning, subsequent work by Das et al. (2017b) and Strub et al. (2017) makes use of reinforcement learning (RL). In all these approaches but Strub et al. (2017), however, the Questioner performs a non-linguistic action (i.e., selects an image or object within an image) after a fixed number of question-answer rounds. Thus, there is no decision making on whether further questions are or are not needed to identify a visual target. To address this limitation, Strub et al. (2017) put forward a more flexible approach: They let the Questioner ask at most 8 questions, but introduce an extra token (`stop`) within the vocabulary, which the question generation model has to learn. This strategy, however, is suboptimal: The question generator needs to generate probabilities for items that do not lie on the same distribution (the distribution of natural language words vs. the distribution of binary decisions *ask*/*guess*).[1] Our work addresses this limitation in a more principled way, by including a new decision-making module within the encoder-decoder architecture and analysing its impact on the resulting dialogues.

We build on work by the dialogue systems community. In traditional dialogue systems, the basic system architecture includes several components – mainly, a language interpreter, a dialogue manager, and a response generator – as discrete modules that operate in a pipeline (Jurafsky and Martin, 2009; Jokinen and McTear, 2009) or in a cascading incremental manner (Schlangen and Skantze, 2009; Dethlefs et al., 2012). The *dialogue manager* is the core component of a dialogue agent: it integrates the semantic content produced by the interpretation module into the agent's representation of the context (the *dialogue state*) and determines the next action to be performed by the agent, which is transformed into linguistic output by the generation module. Conceptually, a dialogue manager thus includes both (i) a *dialogue state tracker*, which acts as a context model that ideally keeps track of aspects such as current goals, commitments made in the dialogue, entities mentioned, and the level of shared understanding among the participants (Clark, 1996); and (ii) an *action selection policy*, which makes decisions on how to act next, given the current dialogue state. In the present work, we focus on incorporating a *decision-making module* akin to an action selection policy into a visually-grounded encoder-decoder architecture and leave

[1] We also note that on the *GuessWhat?!* GitHub page at `https://github.com/GuessWhatGame/guesswhat` it is mentioned that in the updated version of the system by Strub et al. (2017) "qgen [the question generator] stop learning to stop" (GitHub accessed on 16/03/2018).

the integration of other more advanced dialogue management aspects for future work.

In particular, work on incremental dialogue processing, where a system needs to decide not only *what* to respond but also *when* to act (Rieser and Schlangen, 2011), has some similarities with the problem we address in the present paper, namely, when to stop asking questions to guess a target.[2] Researchers within the dialogue systems community have applied different approaches to design incremental dialogue policies for how and when to act. Two common approaches are the use of rules parametrised by thresholds that are optimised with human-human data (Buß et al., 2010; Ghigi et al., 2014; Paetzel et al., 2015; Kennington and Schlangen, 2016) and the use of reinforcement learning (Kim et al., 2014; Khouzaimi et al., 2015; Manuvinakurike et al., 2017). For example, Paetzel et al. (2015) implement an agent that aims to identify a target image out of a set of images given descriptive content by its dialogue partner. Decision making is handled by means of a parametrised rule-based policy: the agent keeps waiting for additional descriptive input until either her confidence on a possible referent exceeds a given threshold or a maximum-time threshold is reached (in which case the agent gives up). The thresholds are set up by optimising points per second on a corpus of human-human dialogues (pairs of participants score a point for each correct guess). In a follow-up paper by Manuvinakurike et al. (2017), the agent's policy is learned with reinforcement learning, achieving higher performance.

We develop a decision-making module that determines, after each question-answer pair in the visually grounded dialogue, whether to ask a further question or to pick a referent in a visual scene. We are interested in investigating the impact of such a module in an architecture that can be trained end-to-end directly from raw data, without specific annotations commonly used in dialogue systems, such as dialogue acts (Paetzel et al., 2015; Manuvinakurike et al., 2017; Kennington and Schlangen, 2016), segment labels (Manuvinakurike et al., 2016), dialogue state features (Williams et al., 2013; Young et al., 2013; Kim et al., 2014), or logical formulas (Yu et al., 2016).

3 Dataset

To develop our model and perform our analyses, we use the *GuessWhat?!* dataset,[3] a dataset of approximately 155k human-human dialogues created via Amazon Mechanical Turk (de Vries et al., 2017). *GuessWhat?!* is a cooperative two-player game: both players see an image with several objects; one player (the Oracle) is assigned a target object in the image and the other player (the Questioner) has to guess it. To do so, the Questioner has to ask Yes/No questions to the Oracle. When the Questioner thinks he/she can guess the object, the list of objects is provided and if the Questioner picks the right one the game is considered successful. No time limit is given, but the Questioner can leave the game incomplete (viz. not try to guess). The set of images and target objects has been built from the training and validation sections of the MS-COCO dataset (Lin et al., 2014) by only keeping images that contain at least three and at most twenty objects and by only considering target objects whose area is big enough to be located well by humans ($area > 500px^2$). Further details are provided in Appendix A.

4 Models

de Vries et al. (2017) develop models of the Questioner and Oracle roles in *GuessWhat?!*. We first describe their models, which we consider as our baseline, and then describe our modified Questioner model. As explained below, de Vries et al. (2017) model the Questioner role by means of two disconnected modules: a Question Generator (QGen) and a Guesser, that are trained independently. After a fixed number of questions by QGen, the Guesser selects a candidate object. We propose and evaluate a model of the Questioner role that incorporates a decision-making component that connects the tasks of asking and guessing (which we take to be part of the planning capabilities of a single agent) and offers more flexibility regarding the number of questions asked to solve the game.

[2] Our system is not word-by-word incremental at this point, but given the incremental nature of encoder-decoder architectures, an extension in this direction should be possible. We leave this for future work.

[3] Available at: `https://guesswhat.ai/`

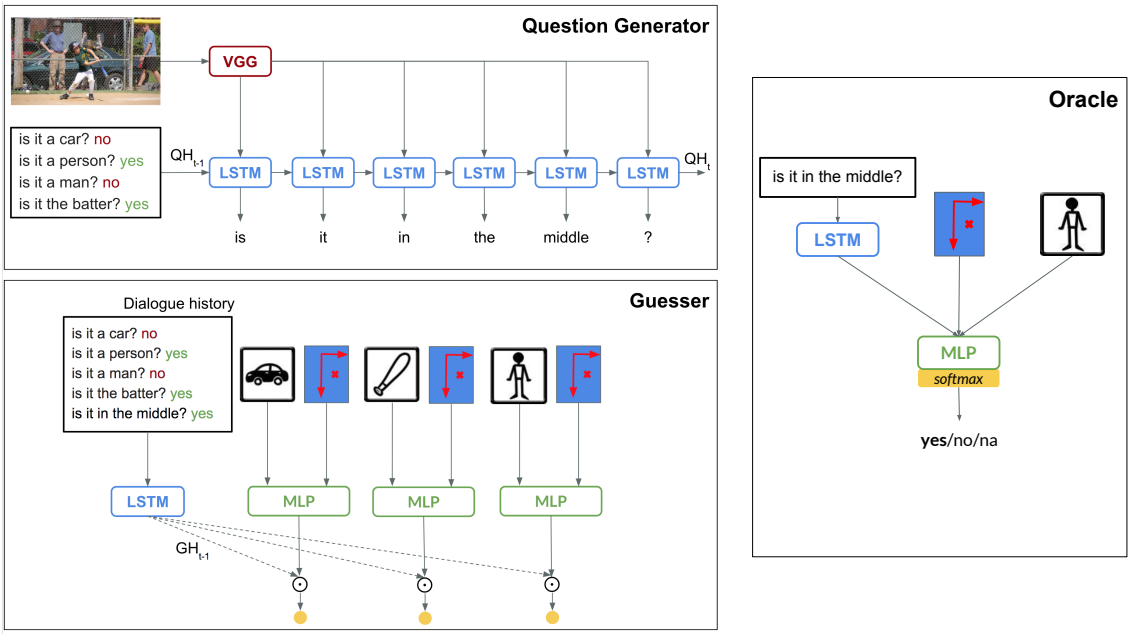


Figure 2: Baseline models. Lefthand side: Independent modules for the Questioner model: Question Generator (top) and Guesser (bottom). Righthand side: Oracle model.

4.1 Baseline Models

We provide a brief description of the models by de Vries et al. (2017), which we re-implement for our study. Further details on the implementation of each module are available in Appendix A.

Question Generator (QGen) This module is implemented as a Recurrent Neural Network (RNN) with a transition function handled with Long-Short-Term Memory (LSTM), on which a probabilistic sequence model is built with a Softmax classifier. Given the overall image (encoded by extracting its VGG features) and the current dialogue history (i.e. the previous sequence of questions and answers), QGen produces a representation of the visually grounded dialogue (the RNN's hidden state QH_{t-1} at time $t-1$ in the dialogue) that encodes information useful to generate the next question q_t. See the sketch on the top-right part of Figure 2.

Guesser The best performing model of the Guesser module by de Vries et al. (2017) represents candidate objects by their object category and spatial coordinates. These features are passed through a Multi-Layer Perceptron (MLP) to get an embedding for each object. The Guesser also takes as input the dialogue history processed by an LSTM, whose hidden state GH_{t-1} is of the same size as the MLP output. A dot product between both returns a score for each candidate object in the image. A diagram of the architecture is given on the bottom-right section of Figure 2.

Oracle The Oracle is aware of the target object and answers each question by QGen with Yes, No, or Not Applicable. The best performing model of the Oracle module by de Vries et al. (2017) takes as input embeddings of the target object category, its spatial coordinates, and the current question. These embeddings are concatenated into a single vector and fed to an MLP that outputs the answer, as illustrated in Figure 2, righthand side.

4.2 Our Questioner Model

We extend the Questioner model of de Vries et al. (2017) with a third module, a decision making component (DM) that determines, after each question/answer pair, whether QGen should *ask* another question or whether the Guesser should *guess* the target object. We treat this decision problem as a binary classification task, for which we use an MLP followed by a Softmax function that outputs probabilities for the two classes of interest: *ask* and *guess*. The Argmax function then determines the class of the next action.

With this approach, we bypass the need to specify any decision thresholds and instead let the model learn whether enough evidence has been accumulated during the dialogue so far to let the Guesser pick up a referent.[4]

We experimented with two versions of the DM component, DM1 and DM2, which differ with respect to the encoding of the linguistic input they have access to. Both decision makers have access to the image and implicitly to the linguistic dialogue history: DM1 exploits the dialogue encoding learned by QGen's LSTM, which is trained to record information relevant for generating a follow-up question. In contrast, DM2 leverages the dialogue encoding learned by the Guesser's LSTM, which is trained to capture the properties of the linguistic input that are relevant to make a guess. The two versions are illustrated in Figure 3. The DM takes as input the concatenation (s_t) of the visual features and of the dialogue history representation it gets either via QGen's hidden state QH_{t-1} (DM1) or the Guesser's hidden state GH_{t-1} (DM2). The resulting vector is passed through a MLP. The output is then scaled between 0 and 1 by the Softmax function and treated as a probability distribution.

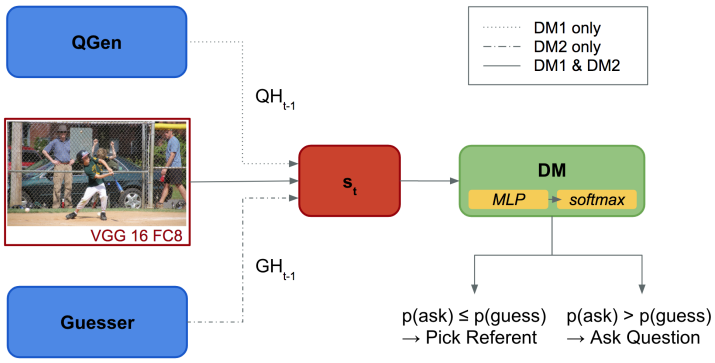


Figure 3: Our Questioner Model with two versions of the Decision Making module, DM1 and DM2.

We also implemented a hybrid DM module that receives as input both QH_{t-1} and GH_{t-1}, but we obtained worse performance. Instead of insisting on the hybrid architecture (left for future work), we maintain the two versions, DM1 and DM2, separated here so that we can focus on investigating their differences and their complementary contributions.[5]

5 Experiments and Results

Next, we present our experimental setup and accuracy results. In Section 6 we then provide an in-depth analysis of the games and dialogues.

5.1 Experimental setup

The modules in the system by de Vries et al. (2017) (QGen, Guesser, and Oracle) are trained independently with supervised learning. To allow for direct comparison with their model, we follow the same setup for training the three original modules and also our new decision making module. Details on hyperparameter settings are provided in Appendix A. We use the same train, validation, and test sets as de Vries et al. (2017).

Both DM1 and DM2 are trained with Cross Entropy loss in a supervised manner, which requires decision labels for the dialogue state after each question/answer pair. We use two different approaches to obtain decision labels from the *GuessWhat?!* ground truth dialogues. In the first approach, the question/answer pairs are labelled based on the human decision to *ask* or *guess*, obtained by checking whether there is a follow-up question in the human-human dialogues. We refer to this paradigm as *gt-label* (*gt* for *ground truth*). In the second approach, we label the question/answer pairs based on whether the Guesser module is able to correctly predict the target object given the current dialogue fragment. If the Guesser

[4]Note also that, since we have separate states for *ask* and *guess* decisions, we could potentially extend the approach to decide among multiple action types beyond the binary case.

[5]In principle, a decision-making module could also leverage the entropy of the scores for each candidate object produced by the Guesser. However, given the setup of the baseline Guesser (which we keep untouched for comparability), this would lead to implicitly using the symbolic object categories and spatial coordinates of each candidate object. This information, however, is not available to the humans at the time of deciding whether to guess or to continue asking: the list of candidate targets only becomes available once a participant decides to guess. Our DM modules only use the raw visual features, and thus are in a similar position to humans performing the task.

MaxQ	5	8	10	12	20	25	30
Baseline	41.18	40.7	39.8	38.96	36.02	35.918	35.88
DM1	40.45 (4.62)	40.12 (6.17)	40.02 (6.70)	40.00 (6.97)	39.87 (7.14)	39.68 (7.14)	39.68 (7.14)
DM2	42.19 (4.53)	41.30 (7.22)	41.12 (8.71)	39.73 (10.72)	37.75 (13.39)	36.86 (13.47)	36.83(13.51)

Table 1: Accuracy on the entire test set (all games, viz. including successful, unsuccessful, decided and undecided ones) for task success by varying the maximum number of questions allowed (MaxQ). Within brackets, the average number of questions asked for each setting. The baseline model always asks the maximum number of questions allowed.

module is able to make a correct prediction after a given question/answer pair, we label that dialogue state with *guess* and otherwise with *ask*. This is referred to as *guess-label*. DM1 can only be trained with *gt-label* (since it does not have access to information coming from the Guesser). For DM2, we treat the choice between these two labelling approaches as a hyperparameter to be tuned on the validation set. DM2 achieves better results when trained with *guess-label*. Experiments on the test set are then conducted with the optimal settings.

5.2 Accuracy Results

We first report results on task accuracy, i.e., the percentage of games where the task is successfully accomplished. Human accuracy is 90.8%. Humans can ask as many questions as they like and, on average, they guess after having asked 5.12 questions. Table 1 gives an overview of the accuracy results obtained by the baseline model, which always asks a fixed number of questions, and by our extended model with the two different versions of the DM, which decides when to ask or guess. We report the accuracy of our re-implementation of the baseline system, which is slightly better than the one reported by de Vries et al. (2017) for 5 questions.[6] To highlight the impact of including a DM module, we report the accuracies the models achieve when changing the maximum number of questions allowed (MaxQ). When MaxQ = 5, the accuracies of the model enriched with a DM module are very similar to the baseline model (slightly lower for DM1: 40.45%, and slightly higher for DM2: 42.19%), and the average number of questions asked by the DM models is also comparable to the 5 questions asked by the baseline, namely 4.62 (DM1) and 4.53 (DM2). However, if we observe how the model accuracy varies when increasing MaxQ, we see that the models equipped with a DM module tend to perform better than the baseline model and that they do so by asking fewer questions than the baseline on average. Furthermore, of the two models enriched with a decision maker, DM1 is more stable across the various settings both in terms of accuracy and of number of questions asked.

In the following analysis, unless indicated otherwise, we consider the baseline model with 5 questions, because it yields the highest baseline accuracy, and the DM models with MaxQ = 10, because more than 90% of the games solved by humans contain up to 10 questions.

6 Analysis

To better understand the results reported above and to gain insight on how the inclusion of a decision making component affects the resulting dialogues, we carry out an analysis of the behaviour of our models. We first examine how the complexity of the game (as determined by visual properties of the image) affects performance, and then analyse the quality of the linguistic interaction from the perspective of the Questioner role.

6.1 Complexity of the Game

Intuitively, the more complex the image involved in a round of the game, the harder it is to guess the target object. As a proxy for image complexity, we consider the following measures: (i) the number of objects in the image, (ii) the number of objects with the same category as the target object, and (iii) the

[6] de Vries et al. (2017) report an accuracy of 34% with a fixed number of 5 questions, while 40.8% is reported on the first author's GitHub page. Our re-implementation of the Oracle and Guesser obtain an accuracy of 78.47% (78.5) and 61.26% (61.3), respectively (in brackets, the original accuracies reported by de Vries et al. (2017)).

	successful vs. *unsuccessful*					*decided* vs. *undecided*	
	all games			decided games		all games	
	Baseline	DM1	DM2	DM1	DM2	DM1	DM2
# objects	-0.213094	-0.212920	-0.217468	-0.220929	-0.23967	-0.05292	0.144233
# objects same cat. as target	-0.150294	-0.144740	-0.150090	-0.148251	-0.165415	-0.058392	0.087068
% target object's area	4.88720	4.254	3.82114	4.15606	7.0496	1.59634	-2.38053

Table 2: Estimated regression coefficients for the logistic regression models distinguishing between *successful* vs. *unsuccessful* and *decided* vs. *undecided* games. A positive/negative coefficient indicates a positive/negative correlation between a predictor and the probability of the *successful* or *decided* class. The contribution of all predictors is statistically significant in all models ($p < 0.0001$).

size of the target object, which we compute in terms of the proportion of the cropped target object area with respect to the whole image. The distribution of games in the whole dataset is fairly balanced across these factors. See Appendix B for full details.

We fit a linear logistic regression model for the task of predicting whether a game will be successful or unsuccessful (i.e., whether the right target object will be selected), using the three measures above as independent predictor variables. It should be noted that in some cases our Questioner model may reach the maximum number of 10 questions without ever taking the action to guess. We refer to these games as *undecided*, whereas we call *decided* games those games where the DM lets the Guesser pick a referent within the maximum number of questions allowed. Out of the whole test set, the amount of decided games is 77.67% and 15.58% for DM1 and DM2, respectively.[7] It is critical to take this into account, since cases where the model is simply forced to make a guess are less informative about the performance of the DM module. Therefore, we fit two additional logistic regression models using the same three predictors: one where we consider only decided games and predict whether they are *successful* or *unsuccessful*, and one where the dependent variable to be predicted are the *decided* vs. *undecided* status of a game.

In Table 2, we report the regression coefficients for the different logistic regression models, estimated with iteratively reweighted least squares.[8] Plots of the predictor variables are available in Appendix B. Regarding the distinction between successful and unsuccessful games, we observe that the three image complexity measures we consider play a significant role. The three models (baseline, DM1, and DM2) are more likely to succeed in games that are intuitively easier — i.e., when the image contains fewer objects overall and fewer objects of the same category as the target (negative coefficients), and when the relative size of the target object is larger (positive coefficients). Interestingly, when we look into the distinction between decided and undecided games, we observe different behaviour for the two versions of the DM module. While DM1 tends to make a decision to stop asking questions and guess in easier games, surprisingly DM2 is more likely to make a guessing decision when the image complexity is higher (note the contrasting tendency of the coefficients in the last column of Table 2 for DM2). However, similarly to DM1, once DM2 decides to guess (decided games in Table 2), the simpler the image the more likely the model is to succeed in picking up the right target object.

6.2 Quality of the Dialogues

As stated in the introduction, we believe that a good visual dialogue model should not only be measured in terms of its task success but also with respect to the quality of its dialogues. In particular unnatural repetitions and unnecessary questions should be avoided. Hence, here we look into the dialogues produced by the different Questioner models comparing them with respect to these two criteria.

[7] To understand the rather big difference between the number of decided games in DM1 and DM2, we also evaluated the models using ground truth data for the QGen and Oracle modules. When human-human dialogues are used as input, the percentage of decided games is high and virtually identical for the two versions of the DM module (81.31% for DM1 and 81.30 for DM2.) This shows that DM2 is affected much more than DM1 by the errors produced by other modules. We leave an analysis of this aspect to future work. Throughout the present paper, all results and analyses reported are not based on ground truth data, but on the representations automatically generated by other modules.

[8] We use the R implementation of the logistic regression algorithm.

	All games						Decided games			
	Overall			Objects			Overall		Objects	
	Baseline	DM1	DM2	Baseline	DM1	DM2	DM1	DM2	DM1	DM2
across-games	98.07%	74.66%	84.05%	44.88%	32.94%	40.10%	69.18%	7.90%	67.05%	7.79%
within-game	45.74%	23.34%	39.27%	18.38%	11.61%	16.42%	31.28%	12.52%	30.06%	12.36%

Table 3: Percentages of repeated questions in all games and in decided games. Overall: all types of questions; Objects: only questions mentioning a candidate target object. All differences between the baseline and our models are statistically significant (Welch t-test with $p < 0.0001$).

Repeated questions Qualitative examination of the dialogues shows that the Questioner models often ask the same question over and over again. This results in unnatural linguistic interactions that come across as incoherent — see, for example, the dialogue in Figure 4 (bottom part). We analyse the dialogues produced by the models in terms of the amount of repetitions they contain. For the sake of simplicity, we only consider repeated questions that are exact string matchings (i.e., verbatim repetitions of entire questions). In this case, we consider the baseline model that asks 10 questions, as this makes for a fairer comparison with our models, where MaxQ = 10 (see end of Section 5).

Table 3 reports the percentage of dialogues that contain at least one repeated question (across-games) and the percentage of repeated questions within a dialogue averaged across games (within-game). We check the percentages with respect to both all the games and only decided games. When considering all games, we find that the baseline model produces many more dialogues with repeated questions (98.07% vs. 74.66% for DM1 and 84.05% for DM2) and many more repeated questions per dialogue (45.74% vs. 23.34% for DM1 and 39.27% for DM2) than our DM models. Among our models, the dialogues by DM1 are less repetitive than those by DM2. However, the difference between the two DM models is reversed when zooming into the decided games.

Our method for quantifying repeated questions is clearly simplistic: questions that have an identical surface form may not count as mere repetitions if they contain pronouns that have different antecedents. For example, a question such as "Is it the one on the left?" could be asked twice within the same dialogue with different antecedents for the anaphoric phrase "the one". In contrast, several instances of a question that includes a noun referring to a candidate object (such as "Is it a dog?") most probably are true repetitions that should be avoided. Therefore, as a sanity check, we perform our analysis taking into account only questions that mention a candidate object.[9] As shown in Table 3, this yields the same patterns observed when considering all types of questions.

Unnecessary questions Another feature of the dialogues revealed by qualitative examination is the presence of questions that are not repetitions of earlier questions but that, in principle, are not needed to successfully solve the game given the information gathered so far. For example, the last three questions in the dialogue in Figure 1 in the Introduction are not necessary to solve the game, given the evidence provided by the first two questions. By including a decision making component, our models may be able to alleviate this problem. In this analysis, we compare the baseline system that asks 5 questions to our models with MaxQ = 10 and look into cases where our models ask either fewer or more questions than the baseline.

Table 4 provides an overview of the results. When considering all the games, we see that the DM models ask many more questions (64.43% DM1 and 85.14% DM2) than the baseline. This is not surprising, since many games are undecided (see Section 6.1) and hence contain more questions than the baseline (10 vs. 5). Zooming into decided games thus allows for a more appropriate comparison. Table 4 also includes information on whether asking fewer or more questions helps (+ Change), hurts (– Change) or does not have an impact on task success (No Change) with respect to the baseline results. We observe that DM2 dramatically decreases the number of questions: in 95.17% of decided games, it asks fewer questions than the baseline; interestingly, in only 13.98% of cases where it asks fewer questions its performance is worse than the baseline — in all the other cases, either it achieves the same success

[9] A list of objects is provided in Appendix C.

DM	Decided games								All games	
	+ Change		– Change		No Change		Total		Total	
	Fewer	More	Fewer	More	Fewer	More	Fewer	More	Fewer	More
DM1	1.77	3.46	2.64	3.79	22.58	50.35	26.99	57.6	22.63	64.43
DM2	25.01	0.16	13.98	0.81	56.18	3.67	95.17	4.64	14.83	85.14

Table 4: Games played by DM with MaxQ=10, and the baseline with 5 fixed questions. Percentages of games (among all games and only decided games) where the DM models ask either fewer or more questions than the baseline. For the decided games, percentages of games where asking fewer/more questions helps (+ Change), hurts (– Change) or does not have an impact on task success w.r.t. the baseline result (No Change).

is it a person? No
is it an elephant? Yes
is it in the middle? Yes (DM2) (status: success)
is it the one on the left? Yes (DM1) (status: failure)
is it the one on the right? Yes (baseline) (status: failure)

is it a person? No
is it on the table? Yes
is it in the front? Yes
is it in the front? Yes
is it in the left? Yes (baseline) (status: failure)
is it in the 1st one? Yes
is it in the 1st one? Yes (DM1) (status: success)

Figure 4: Examples where our model achieves task success by asking fewer or more questions than the baseline. Answers in red highlight Oracle errors. QGen often produces repeated or incoherent questions.

(56.18%) or even improves on the baseline results (25.01%). The latter shows that DM2 is able to reduce the number of unnecessary questions, as illustrated in Figure 1. On the other hand, DM1 does not seem to reduce the number of unnecessary questions in a significant way.

6.3 Discussion

Our analyses show that using a decision making component produces dialogues with fewer repeated questions and can reduce the number of unnecessary questions, thus potentially leading to more efficient and less unnatural interactions. Indeed, for some games not correctly resolved by the baseline system, our model is able to guess the right target object by asking fewer questions. DM2 is substantially better at this than DM1 (25.01% vs. 1.77% of decided games; see Table 4). By being restricted to a fixed number of questions, the baseline system often introduces noise or apparently forgets about important information that was obtained with the initial questions. Thanks to the DM component, our model can decide to stop the dialogue once there is enough information and make a guess at an earlier time, thus avoiding possible noise introduced by Oracle errors, as illustrated in Figure 4 (left). Qualitative error analysis, however, also shows cases where the DM makes a premature decision to stop asking questions before obtaining enough information. Yet in other occasions, the DM seems to have made a sensible decision, but the inaccuracy of the Oracle or the Guesser components lead to task failure. Further examples are available in Appendix D.

In some games with complex images, the information obtained with 5 questions (as asked by the baseline) is not enough to resolve the target. The flexibility introduced by the DM allows our model to reach task success by asking additional questions. Figure 4 (right) gives an example. Furthermore, if noise has been introduced at earlier stages of the dialogue, asking further questions can increase the chance to recover relevant information. However, we also observe that in some games correctly guessed

by the baseline system, asking more questions leads to failure since it opens the door to getting wrong information from the Oracle.

In the current analyses, we have not studied the behaviour of our DM models when using ground truth data instead of the noisy automatic output produced by the other modules. In part, this is motivated by our long-term goal of developing fully data-driven multimodal conversational agents that can be trained end-to-end. We leave for future work carrying out a proper ablation study that analyses the impact of using ground truth vs. automatic data on the DM component.

7 Conclusion

Research on dialogue systems within the Computational Linguistics community has shown the importance of equipping such systems with dialogue management capabilities. Computer Vision researchers have launched the intriguing Visual Dialogue challenge mostly focusing on comparing strong machine learning paradigms on task accuracy, and largely ignoring the aforementioned line of research on dialogue systems. Our goal is to explore how data-driven conversational agents, modelled by neural networks without additional annotations usually exploited by traditional dialogue systems, can profit from a dialogue management module. The present work is a first step towards this long-standing goal. We have taken the *GuessWhat?!* task as our testbed, since it provides a simple setting with elementary question-answer sequences and is task-oriented, which opens the door to using an unsupervised approach in the future. We have focused on augmenting the Questioner agent of the *GuessWhat?!* baseline, which consists of a Question Generator and a Guesser module, with a decision making component (DM) that determines after each question-answer pair whether the Question Generator should ask another question or whether the Guesser should guess the target object. The solution we propose is technically simple, and we believe promising and more cognitive principled than, for example, including a `stop` token as Strub et al. (2017). We show that incorporating a decision making component does lead to less unnatural dialogues. It remains to be seen whether a hybrid DM module that exploits the dialogue encodings of both the Question Generator and the Guesser modules could bring further qualitative and quantitative improvements.

Acknowledgements

We kindly acknowledge the European Network on Integrating Vision and Language (iV&L Net) ICT COST Action IC1307. The Amsterdam team is partially funded by the Netherlands Organisation for Scientific Research (NWO) under VIDI grant nr. 276-89-008, *Asymmetry in Conversation*. In addition, we gratefully acknowledge the support of NVIDIA Corporation with the donation to the University of Trento of the GPUs used in our research.

References

Dan Bohus and Alexander I Rudnicky. 2009. The ravenclaw dialog management framework: Architecture and systems. *Computer Speech & Language*, 23(3):332–361.

Okko Buß, Timo Baumann, and David Schlangen. 2010. Collaborating on utterances with a spoken dialogue system using an isu-based approach to incremental dialogue management. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 233–236. Association for Computational Linguistics.

Herbert H Clark. 1996. *Using Language*. Cambridge University Press.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M.F. Moura, Devi Parikh, and Dhruv Batra. 2017a. Visual Dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Abhishek Das, Satwik Kottur, José M.F. Moura, Stefan Lee, and Dhruv Batra. 2017b. Learning cooperative visual dialog agents with deep reinforcement learning. In *International Conference on Computer Vision (ICCV)*.

Harm de Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron C. Courville. 2017. Guesswhat?! Visual object discovery through multi-modal dialogue. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Nina Dethlefs, Helen Hastie, Verena Rieser, and Oliver Lemon. 2012. Optimising incremental dialogue decisions using information density for interactive systems. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 82–93. Association for Computational Linguistics.

Fabrizio Ghigi, Maxine Eskenazi, M Ines Torres, and Sungjin Lee. 2014. Incremental dialog processing in a task-oriented dialog. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Kristiina Jokinen and Michael McTear. 2009. Spoken dialogue systems. *Synthesis Lectures on Human Language Technologies*, 2(1):1–151.

Daniel Jurafsky and James H. Martin. 2009. Dialogue and conversational agents. In *Speech and Language Processing*, chapter 24. Pearson Prentice Hall.

Casey Kennington and David Schlangen. 2016. Supporting spoken assistant systems with a graphical user interface that signals incremental understanding and prediction state. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 242–251.

Hatim Khouzaimi, Romain Laroche, and Fabrice Lefevre. 2015. Optimising turn-taking strategies with reinforcement learning. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 315–324.

Dongho Kim, Catherine Breslin, Pirros Tsiakoulis, M Gašić, Matthew Henderson, and Steve Young. 2014. Inverse reinforcement learning for micro-turn management. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Staffan Larsson and David R Traum. 2000. Information state and dialogue management in the trindi dialogue move engine toolkit. *Natural language engineering*, 6(3-4):323–340.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California, June. Association for Computational Linguistics.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016b. Deep reinforcement learning for dialogue generation. In *Proceedings of EMNLP*.

T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, Dollar, P., and C. L. Zitnick. 2014. Microsoft COCO: Common objects in context. In *Proceedings of ECCV (European Conference on Computer Vision)*.

Jiasen Lu, Anitha Kannan, Jianwei Yang, Devi Parikh, and Dhruv Batra. 2017. Best of both worlds: Transferring knowledge from discriminative learning to a generative visual dialog model. In *Neural Inforamtion Processing Systems (NIPS) 2017*.

Ramesh Manuvinakurike, Casey Kennington, David DeVault, and David Schlangen. 2016. Real-time understanding of complex discriminative scene descriptions. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 232–241.

Ramesh Manuvinakurike, David DeVault, and Kallirroi Georgila. 2017. Using reinforcement learning to model incrementality in a fast-paced dialogue game. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 331–341.

Nasrin Mostafazadeh, Chris Brockett, Bill Dolan, Michel Galley, Jianfeng Gao, Georgios P. Spithourakis, and Lucy Vanderwende. 2017. Image-grounded conversations: Multimodal context for natural question and response generation. *CoRR*, abs/1701.08251.

Maike Paetzel, Ramesh R. Manuvinakurike, and David DeVault. 2015. "so, which one is it?" the effect of alternative incremental architectures in a high-performance game-playing agent. In *SIGDIAL Conference*, pages 77–86.

Hannes Rieser and David Schlangen. 2011. Introduction to the special issue on incremental processing in dialogue. *Dialogue & Discourse*, 1:1–10.

David Schlangen and Gabriel Skantze. 2009. A general, abstract model of incremental dialogue processing. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 710–718. Association for Computational Linguistics.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of NAACL-HLT*, pages 196–205.

Florian Strub, Harm de Vries, Jeremie Mary, Bilal Piot, Aaron Courville, and Olivier Pietquin. 2017. End-to-end optimization of goal-driven and visually grounded dialogue systems. In *Joint Conference on Artificial Intelligence*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. In *ICML Deep Learning Workshop*.

Jason D Williams, Pascal Poupart, and Steve Young. 2008. Partially observable markov decision processes with continuous observations for dialogue management. In *Recent Trends in Discourse and Dialogue*, pages 191–217. Springer.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413.

Qi Wu, Peng Wang, Chunhua She, Ian Reid, and Anton van den Hengel. 2017. Are you talking to me? reasoned visual dialog generation through adversarial learning. arXiv:1711.07613.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5).

Yanchao Yu, Arash Eshghi, and Oliver Lemon. 2016. Training an adaptive dialogue policy for interactive learning of visually grounded word meanings. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 339.

Appendix A: Details of *GuessWhat?!* Dataset and Experimental Setup

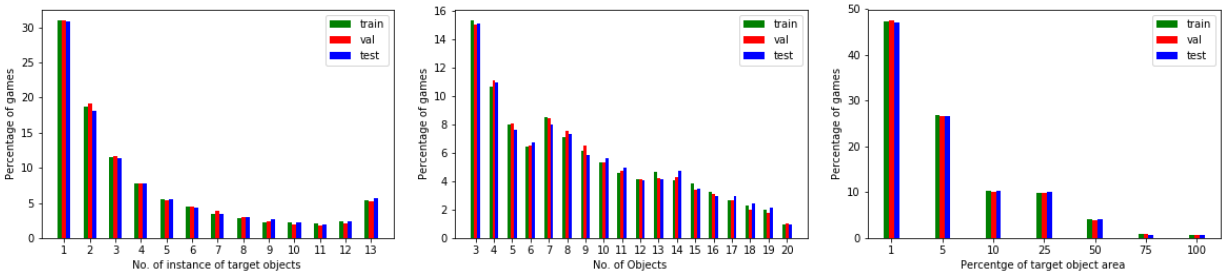
Dataset. The GuessWhat?! dataset contains 77,973 images with 609,543 objects and around 155K human-human dialogues. The dialogues contain around 821K question/answer pairs composed out of 4900 words (counting only words that occur at least 3) on 66,537 unique images and 134,073 target objects. Answers are Yes (52.2%), No (45.6%) and NA (not applicable, 2.2%); dialogues contain on average 5.2 questions and there are on average 2.3 dialogues per image. There are successful (84.6%), unsuccessful (8.4%) and not completed (7.0%) dialogues.

Games and Experimental Setting. In the baseline model, games consist of 5 turns each consisting of question/answer pairs asked by QGen and answered by the Oracle. The QGen module stops asking questions after having received the answer to the 5th question. It is then the turn of the Guesser.

All the modules are trained independently using Ground Truth data. For the visual features, 'fc8' of the VGG-16 network is used. Before visual feature calculation, all images are resized to 224X224. For each object, the module receives the representation of the object category, viz., a dense category embedding obtained from its one-hot class vector using a learned look-up table, and its spatial representation, viz., an 8-dimensional vector. The dialogue is encoded using variable length LSTM with 512 hidden size for Guesser and Oracle and 1024 hidden size for QGen. The LSTM, object category/word look-up tables and MLP parameters are optimized while training by minimizing the negative log-likelihood of the correct answer using ADAM optimizer with learning rate 0.001 for Guesser and Oracle. For QGen, the conditional log-likelihood is maximized based on the next question given the image and dialogue history. All the parameters are tuned on the validation set, training is stopped when there is no improvement in the validation loss for 5 consecutive epochs and best epoch is taken.

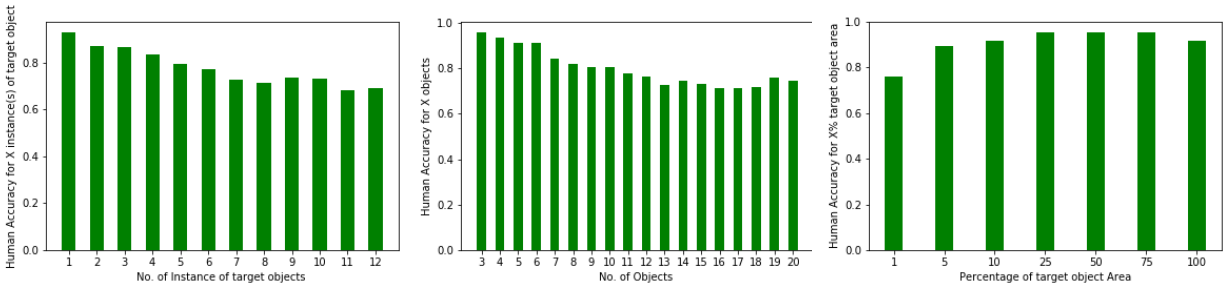
Appendix B: Analysis Regarding Image Complexity

Distribution of image complexity measures. Figure 5 shows the image distribution across the train, validation and test sets with respect to the image complexity measures, namely (a) the number of instances of the target object, (b) the number of objects, and (c) the percentage of target object area with respect to the overall image. We can see that the distribution with respect to these measures is very similar in the three sets. Figure 6 provides human performance based on the different image complexity measures. Human performance is similar to the model performance. While human accuracy is comparatively high, it also decreases when the image complexity increases.



(a) # instances of target object. (b) # of objects. (c) % area covered by target object.

Figure 5: Image distribution with respect to the image complexity measures in the different dataset splits.

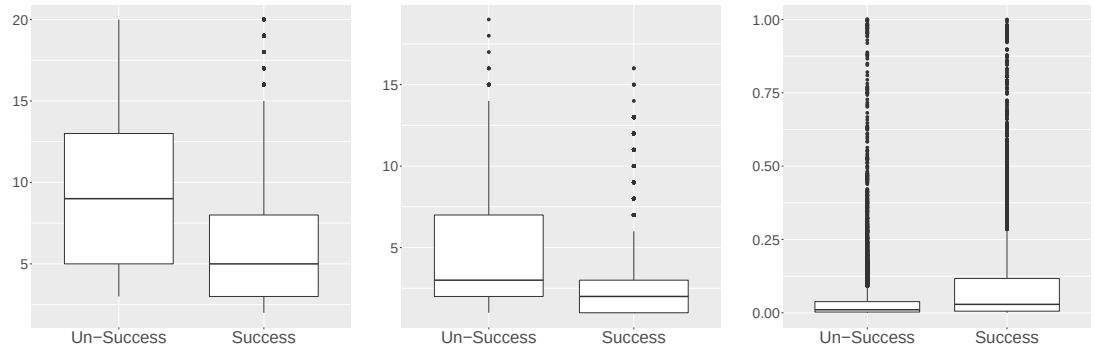


(a) no. of instances of target object. (b) no. of objects. (c) % area covered by target object.

Figure 6: Human accuracy distribution with respect to the image complexity features.

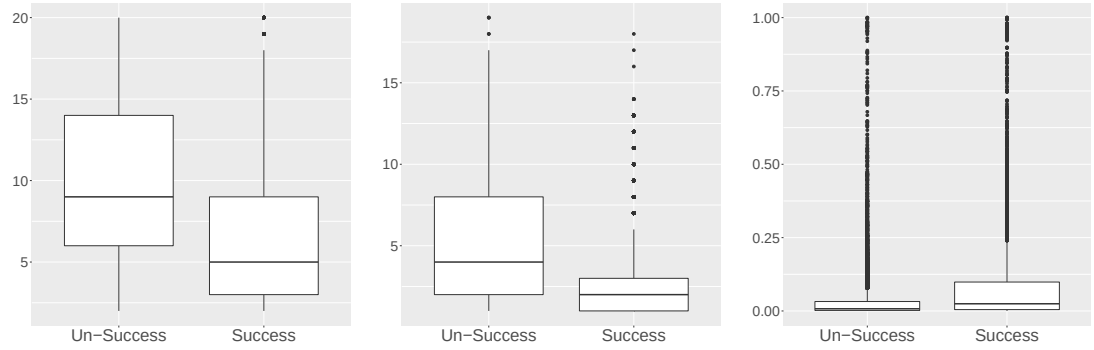
Image complexity measures as predictors in the logistic regression models. Figures 7 and 8 show plots of the image complexity measures in successful vs. unsuccessful games, for all games played by DM1 and DM2. As already noted in Section 6.1, fewer instances of the target object, fewer objects, and larger area of the target object correspond to higher chance of the game being successful, similarly to what we had noticed for humans. We observe the same trend when we restrict the analysis to decided games only, as shown in Figures 9 and 10 for DM1 and DM2, respectively

Figures 11 and 12 compare decided vs. undecided games played by the two DMs. In this case, we observe a difference: DM1 seems to exploit the image complexity measures in a way similar to humans, as noted earlier. DM2, however, decides more often when there are more instances of the target object and when the number of objects in the image is higher. Why this is the case remains unclear.



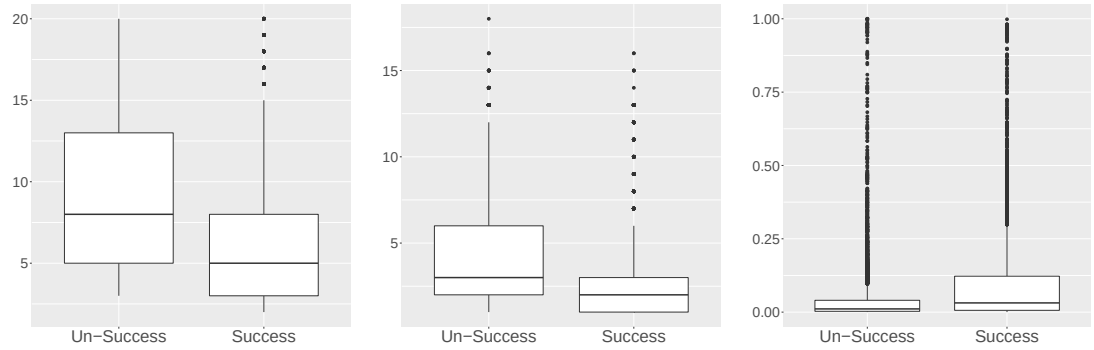
(a) no. of instances of target object. (b) no. of objects. (c) % area covered by target object.

Figure 7: Effect of image complexity measures on successful vs. unsuccessful games played by DM1.



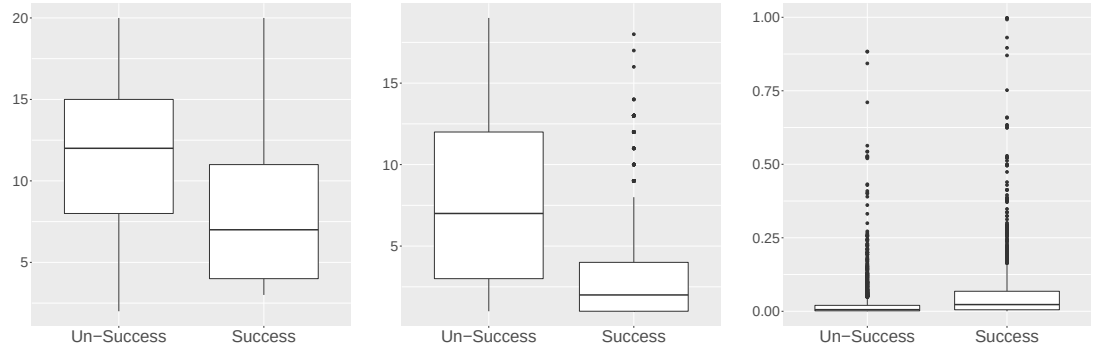
(a) # of instances of target object. (b) # of objects. (c) % area covered by target object.

Figure 8: Effect of image complexity measures on successful vs. unsuccessful games played by DM2.



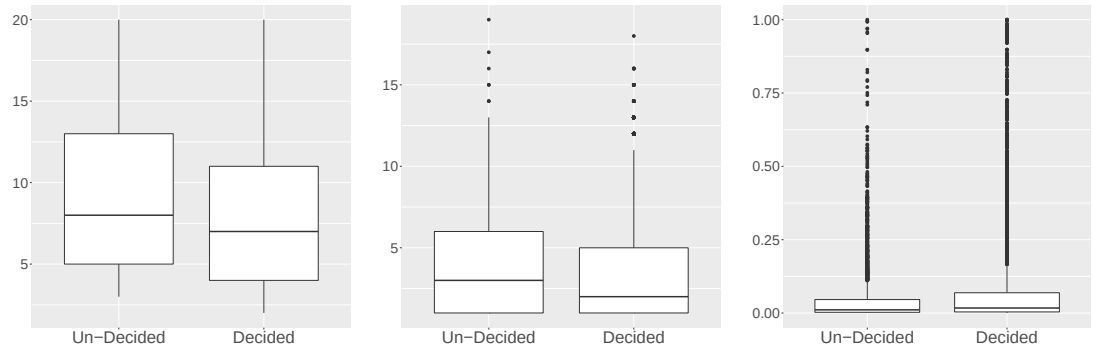
(a) # of instances of target object. (b) # of objects. (c) % area covered by target object.

Figure 9: Effect of image complexity measures on successful vs. unsuccessful *decided games* by DM1.



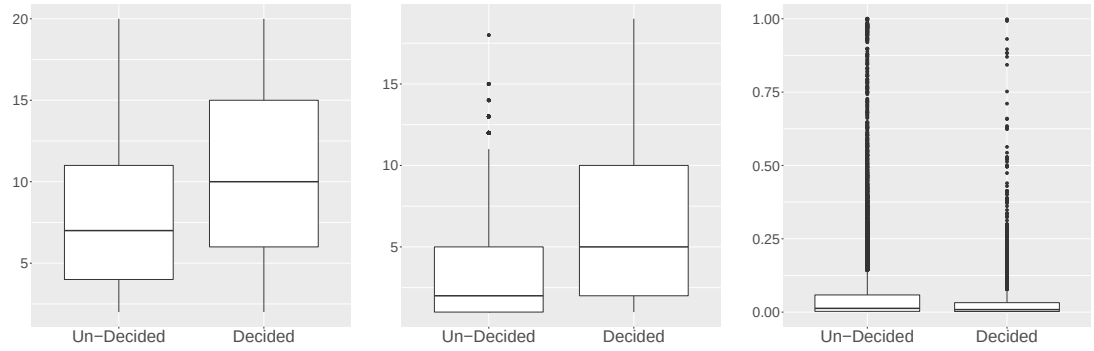
(a) # of instances of target object. (b) # of objects. (c) % area covered by target object.

Figure 10: Effect of image complexity measures on successful vs. unsuccessful *decided games* by DM2.



(a) no. of instances of target object. (b) no. of objects. (c) % area covered by target object.

Figure 11: Effect of image complexity measures on decided vs. undecided games played by DM1.



(a) no. of instances of target object. (b) no. of objects. (c) % area covered by target object.

Figure 12: Effect of image complexity measures on decided vs. undecided games played by DM2.

Appendix C: Repeated Questions

To automatically find the repeated questions in a generated game (see Section 6.2), we have used full string matching, i.e., a question is considered a repetition only when there is another question in the same game having exactly the same words. We have considered repetitions for three types of questions, namely, questions about Object type, Attributes, and Spatial location. Keyword matching is used to decide the type of question that is repetition. For instance, a repeated question is of Object type if it contains keywords such as 'dog', 'cat', etc. These keywords are created using MS-COCO object categories and super-categories, plus the following manually curated list: [man, woman, girl, boy, table, meter, bear, cell, phone, wine, glass, racket, baseball, glove, hydrant, drier, kite].

Appendix D: Example Dialogues

We provide some more examples of successful and unsuccessful games.

(1.) Is it a person? Yes
(2.) Is it the girl? Yes (DM2) (Success)
(3.) Is she in the background? No
(4.) Is she in the background? No
(5.) Is she wearing a white shirt? No (Baseline)(Fail)

(1.) Is it a bike? No
(2.) Is it a car? No
(3.) Is it a bike? No
(4.) Is it a person? Yes
(5.) Is it in the background? Yes (Baseline) (Fail)
(6.) Is it in the left side? Yes (DM1) (Success)

(1.) Is it a person? Yes
(2.) Is it in the foreground? No
(3.) the road? No
(4.) Is it in the foreground? No
(5.) Is it in the background? Yes (Baseline) (Fail)
(6.) the left side? Yes (DM2) (Success)

Figure 13: Successful games where the DM decides to ask more/fewer questions compared to the Baseline. Some of the target objects are very small and by allowing to agent to ask extra questions about the spatial location, the Guesser manages to correctly guess the target object.

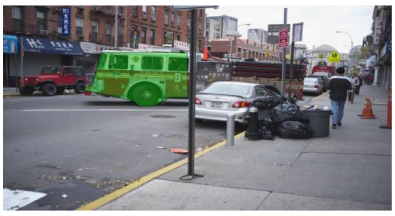
(1.) is it a person? Yes
(2.) in the foreground ? Yes (Oracle Error)
(3.) is it the person in the foreground? No (DM1) (Fail)
(4.) is it the person in the foreground? No
(Repetitive Question)
(5.) is it the person in the background? Yes (Success)

(1.) is it a person? Yes
(2.) is it a male? Yes
(3.) is he wearing a blue shirt? Yes (Oracle Error)
(4.) is he wearing a blue shirt? Yes DM1 (Fail)
(Oracle Error)(Repetitive Question)
(5.) is he wearing a black shirt? Yes (Baseline)
(Success)

(1.) is it a person? No
(2.) is it on the table? Yes
(3.) is it on the table? Yes (DM1) (Fail)
(Repetitive Question)
(4.) Is it the cake? Yes
(5.) Is it the cake? No (Baseline) (Success)
(Repetitive Question)

Figure 14: Unsuccessful games played by DM1, where the system decides to stop asking earlier than the baseline. These failures seem to occur due to the problems by QGen (which generates repeated questions) or the Oracle (which provides a wrong answer).

(1.) is it a person? Yes
(2.) in the foreground? Yes
(3.) the picture? Yes (DM2) (Fail)
(Oracle Error)
(4.) is it the left? No
(5.) Is it the right? Yes (Baseline) (Success)



(1.) is it a person? No
(2.) is it a vehicle? Yes
(3.) on the right? Yes (DM2) (Fail)
(Oracle Error)
(4.) is it the bus? Yes
(5.) is it the fire truck? No (Baseline) (Success)

Figure 15: Unsuccessful games played by DM2, where the system decides to stop asking earlier than the baseline. DM2 seems to be overconfident. Again, failures are often affected by Oracle errors.

Sequence-to-Sequence Data Augmentation for Dialogue Language Understanding

Yutai Hou, Yijia Liu, Wanxiang Che*, Ting Liu
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{ythou, yjliu, car, tliu}@ir.hit.edu.cn

Abstract

In this paper, we study the problem of data augmentation for language understanding in task-oriented dialogue system. In contrast to previous work which augments an utterance without considering its relation with other utterances, we propose a sequence-to-sequence generation based data augmentation framework that leverages one utterance's same semantic alternatives in the training data. A novel diversity rank is incorporated into the utterance representation to make the model produce diverse utterances and these diversely augmented utterances help to improve the language understanding module. Experimental results on the Airline Travel Information System dataset and a newly created semantic frame annotation on Stanford Multi-turn, Multi-domain Dialogue Dataset show that our framework achieves significant improvements of 6.38 and 10.04 F-scores respectively when only a training set of hundreds utterances is represented. Case studies also confirm that our method generates diverse utterances.

Title and Abstract in Chinese

对话语义理解的序列到序列数据增强

在本文中，我们研究了面向任务的对话系统中语言理解模块的数据增强问题。相比之前的工作在生成新语句时不考虑语句间关系，我们利用训练数据中与一个语句具有相同语义的其他句子，提出了基于序列到序列生成的数据增强框架。我们创新地将多样性等级结合到话语表示中以使模型产生多样化的语句数据，而这些多样化的新语句有助于改善语言理解模块。在航空旅行信息系统数据集以及一个新标注的斯坦福多轮多域对话数据集上的实验结果表明，当训练集仅包含数百句语料时，我们的框架在F值上分别实现了6.38和10.04的显着提升。案例研究也证实我们的方法能够产生多样化的话语。

1 Introduction

Language understanding (LU) is the initial and essential component in the task-oriented dialogue system pipeline (Young et al., 2013). One challenge in building robust LU is to handle myriad ways in which users express demands. This challenge becomes more serious when switching to a new domain whose large-scale labeled data is usually unreachable. Insufficiency in training data makes LU vulnerable to unseen utterances which are syntactically different but semantically related to the existing training data, and further harms the whole task-oriented dialogue system pipeline.

Data augmentation, which enlarges the size of training data in machine learning systems, is an effective solution to the data insufficiency problem. Success has been achieved with data augmentation on a wide range of problems including computer vision (Krizhevsky et al., 2012), speech recognition (Hannun et al., 2014), text classification (Zhang et al., 2015), and question answering (Fader et al., 2013). However, its application in the task-oriented dialogue system is less studied. Kurata et al. (2016a) presented the only work we know that tried to augment data for LU. In their paper, an encoder-decoder is

* Email corresponding.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1234–1245
Santa Fe, New Mexico, USA, August 20-26, 2018.

learned to reconstruct the utterances in the training data. During the augmenting process, the encoder's output hidden states are randomly perturbed to yield different utterances.

The work of Kurata et al. (2016a) augments one single utterance by adding noise without considering its relation with other utterances. Besides theirs, there are also works which explicitly consider the paraphrasing relations between instances that share the same output. These works achieve improvements on tasks like text classification and question answering. Paraphrasing techniques including word-level substitution (Zhang et al., 2015; Wang and Yang, 2015), hand-crafted rules generation (Fader et al., 2013; Jia and Liang, 2016), and grammar-tree generation (Narayan et al., 2016) have been explored. Compared with these work, Kurata et al. (2016a) has the advantage of fully data-driven method and can easily switch to new domain without too much domain-specific knowledge, but doesn't make use of the relations between instances within the training data.

In this paper, we study the problem of data augmentation for LU and propose a novel data-driven framework that models relations between utterances of the same semantic frame in the training data. A sequence-to-sequence (seq2seq, Sutskever et al. 2014) model lies in the core of our framework which takes a delexicalised utterance and generates its lexical and syntactical alternatives. To further encourage diverse generation, we incorporate a novel *diversity rank* into the utterance representation. When training the seq2seq model, the diversity rank is also used to filter the over-alike pairs of alternatives. These approaches lead to diversely augmented data that significantly improves the LU performance in the domains that labeled data is scarce.

We conduct experiments on the Airline Travel Information System dataset (ATIS, Price 1990) along with a newly annotated layer of slot filling over the Stanford Multi-turn, Multi-domain Dialogue Dataset (Eric and Manning, 2017).[1] On the small proportion of ATIS which contains 129 utterances, our method outperforms the baseline by a 6.38 F-score on slot filling. On the medium proportion, this improvement is 2.87. Similar trends are witnessed on our LU annotation over Stanford dialogue dataset which the average improvement on three new domains is 10.04 on 100 utterances and 0.47 on 500 utterances.

The major contributions of this paper include:

- We propose a data augmentation framework for LU (§2) using the seq2seq model. A novel diversity rank (§3) is used to encourage our seq2seq model to generate diverse utterances both in the augmentation and training (§4).

- We conduct experiments on the ATIS and Stanford dialogue dataset (§5). Experimental results show our augmentation can effectively enlarge the training data and improve LU performance by a large margin when only a small size of training data is presented. Case studies also confirm that our method generates diverse utterances compared to the results from previous work.

We release our code at:
`https://github.com/AtmaHou/Seq2SeqDataAugmentationForLU`.

2 Overview of the Approach

Notion and Problem Description. In this paper, we study the data augmentation for language understanding (LU), which maps a natural language utterance into its semantic frames. We focus on *slot filling* and follow previous works (Pieraccini et al., 1992) by treating it as a sequence classification in which semantic class labels (*slot types*) are assigned to contiguous sequences of words indicating these sequences are corresponding *slot values*. In this paper, we use the bidirectional long short term memory (BiLSTM) for slot labeling (tagging) as previous works did (Mesnil et al., 2013; Yao et al., 2014; Kurata et al., 2016b).

We formalize the data augmentation for LU as given a natural language utterance $\mathbf{u}$ and its semantic frame $\mathbf{s}$, we generate a set of new utterances with corresponding semantic frames. During the augmenting process, we go through the whole training data $D = \{(\mathbf{u}_i, \mathbf{s}_i)\}_{i=1}^{N}$. For each training instance $(\mathbf{u}_i, \mathbf{s}_i)$,

[1] abbreviated as *Stanford dialogue dataset* henceforth.

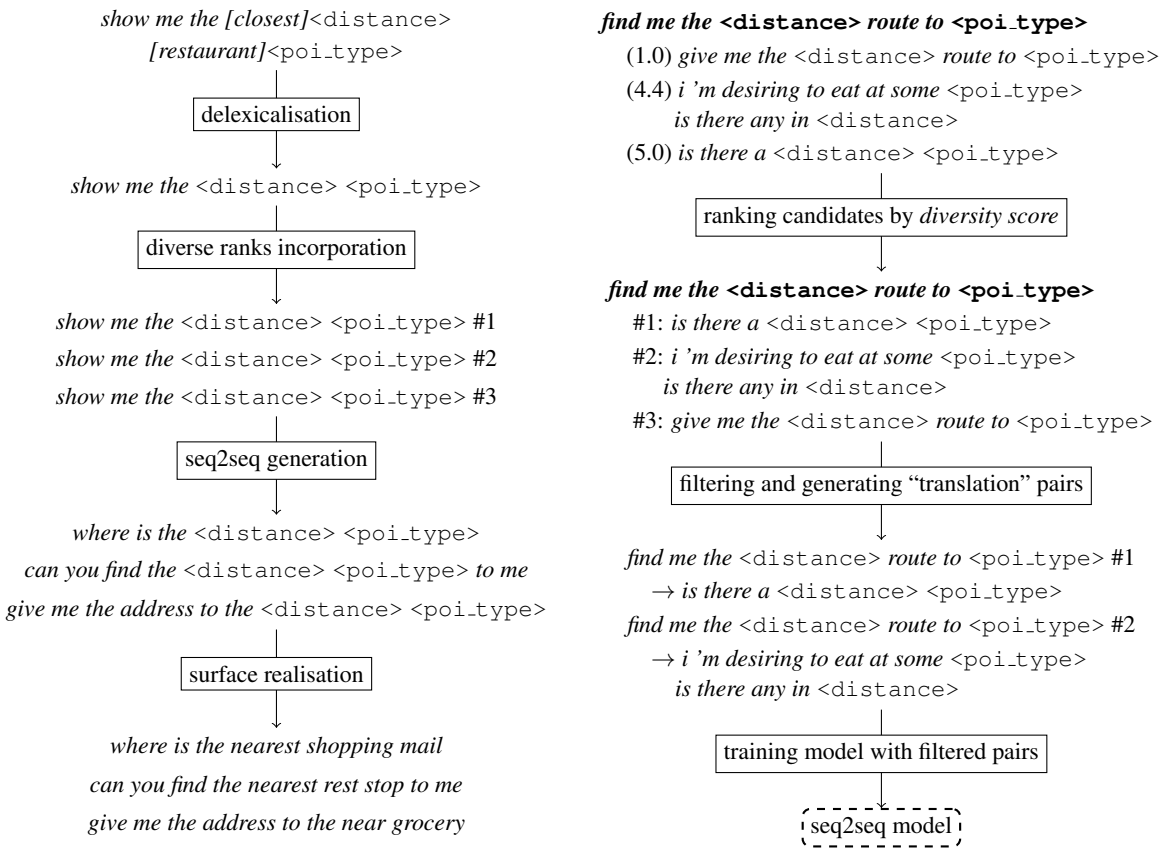


Figure 1: The workflow of our framework. The left part shows the augmenting process and the right part shows the training instance generation process for our seq2seq model. $\mathbf{u} \rightarrow \mathbf{u}'$ marks that $\mathbf{u}$ can be augmented into $\mathbf{u}'$.

we expand it to a set of instances $\{(\mathbf{u}_i^k, \mathbf{s}_i^k)\}_k$ and use the union of the expanded instances as new data to train the LU module.

In the training phase, we define the cluster of semantic frame $\mathbf{s}$ as $C_\mathbf{s} = \{(\mathbf{u}', \mathbf{s}') \mid (\mathbf{u}', \mathbf{s}') \in D \wedge \mathbf{s}' = \mathbf{s}\}$. For one utterance $\mathbf{u}$ and its semantic frame $\mathbf{s}$, each utterance $\mathbf{u}' \in C_\mathbf{s}/\{(\mathbf{u}, \mathbf{s})\}$ is considered as the alternative expression and augmentation of $\mathbf{u}$. We use $\mathbf{u} \rightarrow \mathbf{u}'$ to mark this relation.

To achieve the goal of generating variant utterances under the same semantic frames, We break down the problem into first converting the input utterance $\mathbf{u}$ into its delexicalised form $\mathbf{d}$, and then generating the delexicalised variances of $\mathbf{d}$ with a seq2seq model. Finally, surface realization is carried out to convert the delexicalised form into the raw utterance. The left part of Figure 1 shows the workflow of our augmenting process.

Delexicalisation. When given the raw utterance and its semantic frames associated with certain segments of the utterance, we can easily delexicalise the utterance by replacing the corresponding segments with the semantic frame label. For example, when given the 4th word in "*show me the closest restaurant*" as a `<distance>` slot type and 5th word as `<poi_type>` slot type, its delexicalized form "*show me the* `<distance>` `<poi_type>`" is straight-forward to achieve.

In the task-oriented dialogue system, slot values usually consist of various entity names and are very sparse. Delexicalisation reduces the size of vocabulary and makes the model focus more on generating variant ways of expressing demands. What's more, the semantic frames can be directly derived from the delexicalised generation and used for training the LU module.

Incorporating Diversity Ranks into Utterance Representations. Considering the example in the right part of Figure 1, "*is there a* `<distance>` `<poi_type>`" is more diverse than "*give me the* `<distance>` *route to* `<poi_type>`" when compared with "*find me the* `<distance>` *route to*

<poi_type>". This example shows that for utterance $\mathbf{u}$ with semantic frame $\mathbf{s}$, its alternatives expressions can have different ranks in diversity. To consider the ranking information, we compile the *diversity rank* as an additional information into the utterance representation. By setting it to a higher rank, we aim to generate input utterance's diverse augmentation, and by setting it to lower, a similar utterance should be generated. We will discuss the details of how to compute the ranks during training and how to decide the effective numbers of ranks during testing in Section 3.

Data Augmentation as Seq2Seq Generation. When given the delexicalised input utterance $\mathbf{d}$ and the specified diverse rank k, we use the standard seq2seq model to generate the alternative delexicalised utterance $\mathbf{d}'$. In our seq2seq model, we append $\#k$ to the end of the input utterances and the model is formalized as

$$p(\mathbf{d}' \mid \mathbf{d}, k) = \prod_t p(d'_t \mid d_1, ..., d_n, \#k, d'_1, ..., d'_{t-1})$$

where n is the number of words for the input utterance $\mathbf{d}$.

In this paper, we follow the seq2seq model for neural machine translation and use the *input-feeding* network in (Luong et al., 2015) with attention as our seq2seq model. During testing, we use beam search with beam size of 10 to yield more than one translation following Gimpel et al. (2013) and Vijayakumar et al. (2016).

To train the seq2seq model, our basic assumption is that if $\mathbf{d}$ and $\mathbf{d}'$ contain the same semantic frames, they can be generated from each other. Generally, we assume each pair of delexicalised utterances in the cluster $C_\mathbf{s}$ makes a pair of generation. However, it's nontrivial to assign *diverse ranks* to training data. What's more, to prevent the model from just producing produce lexical paraphrases (like "*show me*" to "*give me*"), we propose to also consider the diversities when generating training translations for the seq2seq model. We will talk about the details in Section 4.

Surface Realisation. Till now, we have achieved the lexically and syntactically different utterances in their delexicalized forms. We would like to bridge these utterances to their lexicalized forms and surface realisation is employed as the final step of our approach.

In this paper, the surface realisation is performed by replacing the slot type in the delexicalised form with its slot value. The mapping from slot type to its set of slot values (e.g. from <poi_type> to {*hospital*, *restaurant*}) is collected on the training data. Somehow, it's nontrivial to just do the replacement because one slot value doesn't fit its slot type in any context. Taking the utterance in Figure 1 for example, in the delexicalised utterance "*i 'm desiring to eat at some* <poi_type> *is there any in* <distance>", 'hospital' doesn't fit in the <poi_type> because 'hospital' isn't the place intended for a meal. To make the surface realisation more reasonable, we build the mapping with consideration of the context and use slot type along with its surrounding 5 words as the key in the mapping.

During surface realisation for an utterance, we first extract the slot type and its context. Then we use this to get all its slot values. If the slot type under certain context is not presented in the mapping, we use the one with the most similar context in the sense of *edit distance*. If more than one slot values present, we randomly pick a slot value.

3 *Diversity Ranks* in Utterance Representations

The major motivation of this paper is to encourage diverse generation. To accomplish this motivation, we propose a criterion named *diversity rank* to model the diversities. During augmenting the data, for an instance $(\mathbf{u}, \mathbf{s})$ we generate the delexicalised utterance at rank from 1 to $N_\mathbf{s}$, where $N_\mathbf{s}$ is a number governed by the semantic frame $\mathbf{s}$ and calculated as $||C_\mathbf{s}||/2$, which is the half size of the instances in D that have the semantic frame $\mathbf{s}$.

During training the seq2seq model with diversity rank, for one instance $(\mathbf{u}, \mathbf{s})$, we first collect $C_\mathbf{s}$, then rank each instance $(\mathbf{u}', \mathbf{s}) \in C_\mathbf{s}/\{(\mathbf{u}, \mathbf{s})\}$ by its *diversity score* against $\mathbf{u}$. In this paper, the diversity score of an utterance pair $(\mathbf{u}, \mathbf{u}')$ is calculated by both considering the *edit distance* and a *length difference penalty* (LDP) as:

$$\text{SCORE}(\mathbf{u}, \mathbf{u}') = \text{EDITDISTANCE}(\mathbf{u}, \mathbf{u}') \times \text{LDP}(\mathbf{u}, \mathbf{u}') \quad (1)$$

	Navigation	Scheduling	Weather
# of training utterances	500	500	500
# of devel. utterances	321	201	262
# of test utterances	337	212	271
Kappa	0.68	0.92	0.90
Agreement	85.05	90.75	95.99

Table 1: Statistics for our annotation.

where LDP is defined as $\text{LDP}(\mathbf{u}, \mathbf{u}') = e^{-\frac{|||\mathbf{u}||-||\mathbf{u}'|||}{||\mathbf{u}||}}$. After obtaining the ranks over the utterances $\mathbf{u}'$, we directly incorporate the rank value as an additional last token for the seq2seq model.

We note that using the LDP reduces the impact of differences in length and makes the score paying more attention to the lexical and syntactical difference. For example, the first block of right part of Figure 1 shows the diversity scores of three different utterances. Although the utterance "*i 'm desiring to eat at some* <poi_type> *is there any in* <distance>" presents larger *edit distance* (12 in this case) than that of "*is there a* <distance> <poi_type>" (5 in this case), the final score is penalized to 4.4 because the length difference.

In our method, the diversity rank can be treated as an utterance-independent controller for the diversity of target generation.

4 Filtering the Alike Instances

To learn the seq2seq model, it's straight-forward to use each pair of utterances in $C_{\mathbf{s}}$ as training data for the model. However, the goal of our paper is to generate diverse augmented data and the usefulness of less diverse pair (like *give me the* <distance> *route to* <poi_type> and *find me the* <distance> *route to* <poi_type> in Figure 1) is arguable.

In this paper, we propose to filter the less diverse pairs when training the seq2seq model. Again, we make use of the ranks derived by the diversity scores and for an utterance $\mathbf{u}$ only the most diverse half of the translations $\mathbf{u} \rightarrow \mathbf{u}'$ are used to train the seq2seq model and the training data can be formalized as

$$D_{\text{seq2seq}} = \bigcup_{(\mathbf{u},\mathbf{s}) \in D} \{\mathbf{u}, \textsc{Rank}(\mathbf{u}, \mathbf{u}') \rightarrow \mathbf{u}' \mid \mathbf{u}' \in C_{\mathbf{s}}, \textsc{Rank}(\mathbf{u}, \mathbf{u}') \geq ||C_{\mathbf{s}}||/2\}$$

After filtering the less diverse pairs, we use D_{seq2seq} to train the seq2seq model.

In this section, we revisit the role of our diversity ranks in the learning perspective. Since we consider the utterance in cluster $C_{\mathbf{s}}$ as translation to each other, without the RANK value, one utterance can simultaneously translate to different utterances in the training data. It increases the ambiguities in learning the seq2seq model and even makes it intractable. With the RANK value, such ambiguities are resolved because each pair of the training data is expanded with a unique value.

5 Experiments

5.1 Settings

Dataset. In this paper, we conduct our experiments on the ATIS dataset which is extensively used for LU (Mesnil et al., 2013; Mesnil et al., 2015; Chen et al., 2016a). The ATIS dataset contains 4978 training utterances from Class A training data in the ATIS-2 and ATIS-3 corpus, while the test contains 893 utterances from the ATIS-3 Nov93 and Dec94 datasets. The size of the training data is relatively large for LU in a single domain. To simulate the data insufficient situations, we follow Chen et al. (2016a), and also evaluate our model on two small proportions of the training data which is *small* (1/40 of the original training set with 129 instances) proportion and *medium* (1/10 of the original training set with 515 instances). In all the experiments, a development set of 500 instances is used.

To test our model on new domains beyond ATIS, we also create a new LU annotation over the Stanford dialogue dataset (Eric and Manning, 2017). We use the same data split as Eric and Manning (2017) and

Model	small	medium	full
	129	515	4,478
Baseline	67.33**	85.85**	94.93*
Ours	73.71	88.72	94.82
Re-implementation of Kurata et al. (2016a)	67.93**	87.34**	94.61**
Model-1 Additive (Kurata et al., 2016a)	-	-	95.08
K-SAN syntax (Chen et al., 2016a)	74.35	88.40	95.00
Model-III (Zhai et al., 2017)	-	-	95.86

Table 2: The results on the ATIS dataset. The first block shows the results from our implementation and the second block is drawn from the papers of previous works. Here we use * to indicate that the difference between the model and Ours is statistically significant under t-test (** for p-value threshold as 0.05 and * for threshold as 0.1) .

annotate the full test sets for the three domains (*navigation*, *scheduling*, and *weather*) along with a small training set of 500 utterances. The Stanford dialogue dataset provides semantic frames (*slot*) for each utterance but doesn't associate the semantic class of the slot with corresponding segment in the utterance. Our annotation focus on assigning the slot to its corresponding segment. During the annotation, each dialogue was processed by two annotators. Data statistics, Kappa value (Snow et al., 2008), and inner annotator agreement measured by F-score on the three domains are shown in Table 1.

Evaluation. We evaluate our data augmentation's effect on LU with F-score. `conlleval` is used in the same way with previous works (Mesnil et al., 2013; Mesnil et al., 2015; Chen et al., 2016a).

Implementation. We use OpenNMT (Klein et al., 2017) as the implementation of our seq2seq model. We set the number of layers in LSTM as 2 and the size of hidden states as 500. Utterances that are longer than 50 are truncated. We adopt the same training setting as Luong et al. (2015) and use Adam (Kingma and Ba, 2014) to train the seq2seq model. Learning rate is halved when perplexity on the development set doesn't decrease. During generation, we replace the model-yielded unknown token (*unk*) with the source word that has the highest attention score.

For the slot tagging model, we set both the dimension for word embedding and the size of hidden state to 100. We also vary dropout rate in $\{0, 0.1, 0.2\}$ considering its regularization power on small size of data. The batch size is set to 16 in all the experiments. Best hyperparameter settings are determined on the development set. GloVe embedding (Pennington et al., 2014) is used to initialize the word embedding in the model. Adam with the suggested settings in Kingma and Ba (2014) is used to train the parameters.

Reimers and Gurevych (2017) pointed out that neural network training is nondeterministic and depends on the seed for the random number generator. We witness dramatic changes of the slot tagging performance using different random seeds. To control for this effect, we take their suggestions and report the average of 5 differently-seeded runs.

5.2 Results on ATIS

Table 2 shows the slot tagging results on the ATIS dataset. Our baseline model is the vanilla BiLSTM slot tagger and our augmented slot tagger use the same architecture but is trained with the augmented data generated by our method. Compared with the vanilla tagger baseline, our augmentation method significantly improves the LU performance by a 6.38 F-score on the *small* proportion and a 2.02 F-score on the *medium* proportion. The improvements show the effectiveness of our augmentation method in the data-insufficient scenario. On the full data, our augmentation slightly lags the baseline. We address this to the fact that full ATIS is large enough for LU on a single domain and our augmentation introduce some noise.

To compare with the previous augmentation work from Kurata et al. (2016a), we re-implemented their *model-1 additive* model using the suggested settings in their paper. The results on the *small*, *medium*, and *full* proportions are shown in the third row of Table 2. On all the proportions, our augmentation

# utterances	Model	Navigation	Scheduling	Weather
100	Baseline	59.93	68.29	82.43
	Ours	72.91	77.30	90.55
500	Baseline	78.99	86.05	93.68
	Ours	78.46	87.67	94.01

Table 3: The results on Stanford dialogue dataset.

Model	F-score	# new	max. ED
Ours	88.72	301	3.18
- seq2seq generation	-0.84**	0	0
- diversity ranks	-0.40*	163	2.42
- filtering	-0.38	870	2.86

Table 4: The result of the ablation test. *# new* marks the number of newly generated delexicalised utterances. *max. ED* marks the averaged maximum edit distances. Here we use * to indicate that the result is statistically significant under t-test (** for p-value threshold as 0.05 and * for threshold as 0.1) By removing the seq2seq generation from our method, no delexicalised utterance will be generated so the *max. ED* cell is 0.

method outperforms theirs and the differences are significant on *small* and *medium*. Since their model relies on learning a seq2seq model to reconstruct the input utterances, it's usually difficult to train a reasonable model on very small data due to sparsity. Our method mitigates this by both generating on the delexicalised utterances and learning the generation model from pairs of utterances that share same semantic frame which enlarge the size of data for us to train the model. We also compare our model with the syntax version of K-SAN (Chen et al., 2016a) without joint training from intent annotation. We see that our augmented tagger lags their syntax-parsing-enhanced model by a 0.64 F-score on small proportion and outperforms theirs by a 0.32 F-score on medium proportion. But considering the training data is sampled with different random seeds between our work and theirs, these results are not directly comparable. At last, we show the (Zhai et al., 2017) as state-of-art results on ATIS dataset, which views slot filling task as sequence chunking problem. As we focus data augmentation for sequence labeling task rather than chunking, this result is not directly comparable to ours. Besides, K-SAN (Chen et al., 2016a) and (Zhai et al., 2017) are not data augmentation methods, we included their results to show that our augmentation method is reasonably good The basic trend shows that our augmentation can be used as an alternative to the LU model leveraging rich syntactic information.

5.3 Results on Stanford Dialogue Dataset

The results for Stanford dialogue dataset are shown in Table 3. Similar trend as the ATIS experiments is witnessed in which the augmentation improves the LU performance. The average improvement on the training data with 100 utterances is 10.04, and the number is 0.47 for that with 500 utterances. Considering that only fewer than 350 utterances present in the test set in all these domains, these improvements are reasonable. Besides, similar to the ATIS results, the margin of improvements is larger for the smaller training set.

An advantage of our method is that it's purely data-driven. Only a mapping from slot type context to slot values is required and it can be constructed from the training data. It's easy for our method to switch to new domains and our results on the Stanford dialogue dataset confirms this.

5.4 Analysis

Ablation. To get further understanding of each component in our method, we conduct ablation on the *medium* proportion, Each of the three parts of our method is removed respectively, including the *seq2seq generation*, *diversity ranks*, and *filtering*. In addition to evaluate the model's performance with F-score, we also examine the augmented data by the number of newly generated delexicalised utterances and the

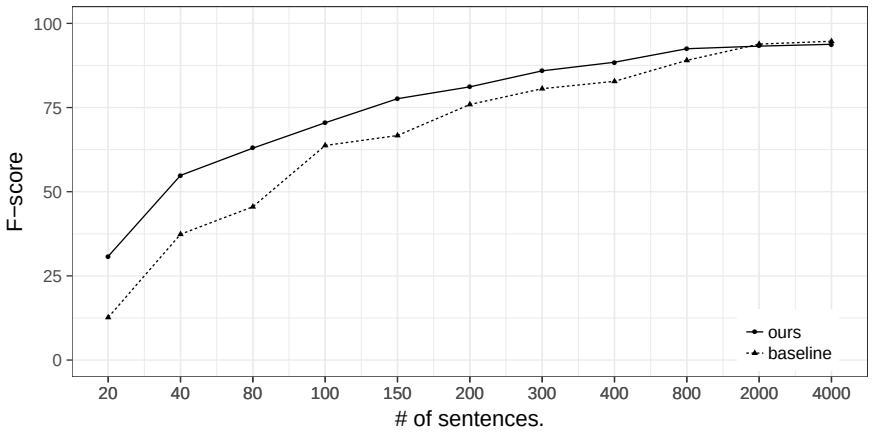


Figure 2: Our method's performances on the ATIS training data of different sizes.

maximum edit distances against the rest of instances.[2] The results are shown in Table 4.

For our method without *seq2seq generation*, we only conduct surface realisation on the delexicalised utterance and a 0.84 F-score drop is witnessed. Since surface realisation only substitutes slot type with different slot values without changing the utterances syntactically, this ablation shows it's more beneficial to generate syntactic alternatives using our seq2seq model.

For our method without *diversity ranks*, we remove diversity ranks from the utterance representation and this lead a drop of 0.40 F-score. We address the drop of performance to the fact that removing either these components will lead to less diverse generation. The second and third column in Table 4 confirm this by showing less newly and diversely generated delexicalised utterances.

If we don't filter the alike instances when training the seq2seq model, the drop of performance is a 0.65 F-score. However, larger number of new utterances with smaller edit distances are yielded which indicates that more noise is introduced when the training data of the seq2seq model is not properly filtered.

This ablation also shows correlation between the maximum edit distance and the final F-score, which indicates generating diverse augmentation helps the performance.

Effect of Training Data Size. The results on ATIS and Stanford dialogue dataset witness the trend that smaller training data benefits more from our augmentation method. A natural question that arises is what's boundary of our augmentation in the sense of improving the baseline. In this section, we study this by varying training data size on the ATIS data. Figure 2 shows the results. For the ATIS data, improvements can be achieved in all our settings with training size smaller than one thousand. These results indicate that our augmentation is applicable when we only access to a LU training data of hundreds instances.

Case Study. In this paragraph, we perform case study on our method to verify its capability of generating diversely augmented data. Table 5 shows two cases of our augmentation. Each case includes the original sentence and its delexicalised form (in *italic* font), the diversity rank (starts with # mark), the training utterance under this rank, our augmentation along with surface realization, and the augmentation produced by Kurata et al. (2016a).

By comparing our augmentation with the delexicalised form of source utterance, two observations can be drawn: 1) our method yields syntactically different alternatives meanwhile keeps the original semantic frame as the source utterance; 2) the lengths of the generated utterances are in the same scale with the source utterance thanks to the effect of length penalty in Equation 1.

By comparing our augmentation with the target training utterance under the same rank, our seq2seq model yields different utterance instead of repeating the training utterance. We address this diversity

[2]This number is normalized by the total number of utterances.

show me all flights from atlanta to washington with prices		
(delex.) *show me all flights from* <from_city> *to* <to_city> *with prices*		
#1	train	let 's look at <from_city> to <to_city> again
	ours	what are all the flights between <from_city> and <to_city>
		(realized) what are all the flights between indianapolis and tampa
#100	train	list types of aircraft that fly between <from_city> and <to_city>
	ours	i 'm looking for a flight from <from_city> to <to_city>
		(realized) i 'm looking for a flight from milwaukee to los angeles
	Kurata16	show me all flights from [atlanta]<from_city> to [washington]<to_city> with airports
is there a flight between san francisco and boston with a stopover at dallas fort worth		
(delex.) *is there a flight between* <from_city> *and* <to_city> *with a stopover at* <stop_city>		
#1	train	which airlines fly from <from_city> to <to_city> and have a stopover in <stop_city>
	ours	is there a flight from <from_city> to<to_city> with a stop in <stop_city>
		(realized) is there a flight from washington to miami with a stop in dallas fort worth
#30	train	do you have any airlines that would stop at <stop_city> on the way from <from_city> to <to_city>
	ours	i 'd like to fly from <from_city> to <to_city> with a stop in <stop_city>
		(realized) i 'd like to fly from memphis to boston with a stop in minneapolis
	Kurata16	is there a flight between [san francisco]<from_city> and [boston]<to_city> with a stopover at [dallas fort worth]<to_city>

Table 5: Case study of our augmented data against the training data and the results of Kurata et al. (2016a) (marked as Kurata16). *train* marks the target utterance in the training data. **(delex.)** marks the delexicalised form of the input utterance. **(realized)** marks the utterance after surface realisation.

to the fact that our diversity rank has some universal effect on modeling the diversity degree across different instances. When contrasting to the augmentation of Kurata et al. (2016a), our method clearly shows diverse augmentation against the source utterance while theirs are basically repeating the source utterances. In the sense of generating diverse alternatives for expressing the same semantics, our method has the advantage.

6 Related work

Data augmentation is an effective way of improving the model's performance and it has been extensively explored on the computer vision community. Single transformation approaches like randomly copying, flipping, and changing the intensity of RGB are the common practice in the top-performed vision systems (Krizhevsky et al., 2012). Beyond these classic approaches, adding noise to the image, randomly interpolating a pair of images (Zhang et al., 2018) are also proposed in previous works. However, these signal transformation approaches are not directly applicable to language because order of words in language may form rigorous syntactic and semantic meaning (Zhang et al., 2015). Therefore, the best way of data augmentation in language usually involves generating the alternative expressions.

Paraphrasing is the most studied techniques in natural language processing for generating alternative expressions (Barzilay and McKeown, 2001; Bannard and Callison-Burch, 2005; Callison-Burch, 2008). However, generic paraphrasing technique has been reported not helpful for specific problem (Narayan et al., 2016). Most of the successful work that applying paraphrasing for data augmentation requires special tailored paraphrasing techniques. For example, Wang and Yang (2015) performed word-level paraphrasing to extend their corpus on twitter that contains annoying behaviors. Fader et al. (2013) derived question templates from seed paraphrases and bootstrap the templates to achieve the enlarged open-domain QA dataset. Narayan et al. (2016) constructed latent variable PCFG for questions and augment the training data by sampling from the grammar. All these works assume the same output (i.e.

class in text classification, answer in question answering) for input paraphrases. Our method resembles theirs in the assumption for input paraphrases, but differs on using the seq2seq generation which is purely data-driven and doesn't rely on special tailored domain knowledge. Besides these methods, works that introduce errors to language understanding have also been proposed (Schatzmann et al., 2007b; Sagae et al., 2012).

Language understanding, as an important component in the task-oriented dialogue system pipeline, has drawn a lot of research attention in recent year, especially when enhanced by the rich representation power of the neural network, like recurrent neural network, LSTM (Yao et al., 2013; Yao et al., 2014; Mesnil et al., 2013; Mesnil et al., 2015) and memory network (Chen et al., 2016b). Rich linguistic features (Chen et al., 2016a) and representation in broader scope on sentence-level (Kurata et al., 2016c) and dialogue history-level (Chen et al., 2016b) have also been studied. Our augmentation method is orthogonal to these works and it's hopeful to achieve more improvements with their works.

Dialogue management is also a key component of task-oriented dialogue system, which mainly focuses on dialogue policy. However, optimal dialogue policy is hard to obtain from a static corpus due to the vast space of conversation process. A solution is to transform the static corpus into user simulator (Kreyssig et al., 2018), and most user simulators work on user semantics level. (Eckert et al., ; Schatzmann et al., 2007a; Asri et al., 2016; Scheffler and Young, 2000; Scheffler and Young, 2001; Pietquin and Dutoit, 2006; Georgila et al., 2005; Cuayáhuitl et al., 2005). Recent work starts to generate user utterance directly to reduce data annotation(Kreyssig et al., 2018).

In recent years, Generative Adversarial Network (GAN, Goodfellow et al. 2014) draws a lot of research attention. Its ability of generating adversarial examples is attractive for data augmentation. However, it hasn't been tried in data augmentation beyond computer vision (Antoniou et al., 2018). How to apply GAN to language understanding is still an open question.

7 Conclusion

In this paper, we study the problem of data augmentation for LU. We propose a data-driven framework to augment training data. In our framework, one utterance's alternative expressions of the same semantic are leveraged to train seq2seq model. We also propose a novel diversity rank to encourage diverse generation and filter alike instances. In the experiments, our model achieves significant improvements of 6.38 and 10.04 F-scores respectively when only a training set of hundreds utterances is represented. Careful case study also shows the capability of our framework to generate diverse alternative expressions.

Acknowledgements

We thank Xiaoming Shi for the LU annotation over the Stanford dialogue dataset. We are grateful for helpful comments and suggestions from the anonymous reviewers. This work was supported by the National Key Basic Research Program of China via grant 2014CB340503 and the National Natural Science Foundation of China (NSFC) via grant 61632011 and 61772153.

References

Anthreas Antoniou, Amos Storkey, and Harrison Edwards. 2018. Data augmentation generative adversarial networks.

Layla El Asri, Jing He, and Kaheer Suleman. 2016. A sequence-to-sequence model for user simulation in spoken dialogue systems. *arXiv preprint arXiv:1607.00070*.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proc. of ACL*.

Regina Barzilay and Kathleen R. McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proc. of ACL*.

Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proc. of EMNLP*.

Yun-Nung Chen, Dilek Hakanni-Tür, Gokhan Tur, Asli Celikyilmaz, Jianfeng Guo, and Li Deng. 2016a. Syntax or semantics? knowledge-guided joint semantic frame parsing. In *SLT*, pages 348–355.

Yun-Nung Vivian Chen, Dilek Hakkani-Tür, Gokhan Tur, Jianfeng Gao, and Li Deng. 2016b. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *INTERSPEECH*.

Heriberto Cuayáhuitl, Steve Renals, Oliver Lemon, and Hiroshi Shimodaira. 2005. Human-computer dialogue simulation using hidden markov models. In *ASRU*, pages 290–295. IEEE.

W. Eckert, E. Levin, and R. Pieraccini.

Mihail Eric and Christopher D Manning. 2017. Key-value retrieval networks for task-oriented dialogue. *arXiv preprint arXiv:1705.05414*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proc. of ACL*.

Kallirroi Georgila, James Henderson, and Oliver Lemon. 2005. Learning user simulations for information state update dialogue systems. In *Ninth European Conference on Speech Communication and Technology*.

Kevin Gimpel, Dhruv Batra, Chris Dyer, and Gregory Shakhnarovich. 2013. A systematic exploration of diversity in machine translation. In *Proc. of EMNLP*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. 2014. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proc. of ACL*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proc. of ACL 2017, System Demonstrations*.

Florian Kreyssig, Inigo Casanueva, Pawel Budzianowski, and Milica Gasic. 2018. Neural user simulation for corpus-based policy optimisation for spoken dialogue systems. *arXiv preprint arXiv:1805.06966*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Gakuto Kurata, Bing Xiang, and Bowen Zhou. 2016a. Labeled data generation with encoder-decoder LSTM for semantic slot filling. In *INTERSPEECH 2016*, pages 725–729.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016b. Leveraging sentence-level information with encoder lstm for semantic slot filling. In *Proc. of EMNLP*.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016c. Leveraging sentence-level information with encoder lstm for semantic slot filling. *arXiv preprint arXiv:1601.01530*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *INTERSPEECH 2013*.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Z. Hakkani-Tür, Xiaodong He, Larry P. Heck, Gökhan Tür, Dong Yu, and Geoffrey Zweig. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM TASLP*, 23(3):530–539.

Shashi Narayan, Siva Reddy, and Shay B. Cohen. 2016. Paraphrase generation from latent-variable pcfgs for semantic parsing. In *Proc. of INLG*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

R. Pieraccini, E. Tzoukermann, Z. Gorelov, J. L. Gauvain, E. Levin, C. H. Lee, and J. G. Wilpon. 1992. A speech understanding system based on statistical representation of semantics. In *Proc. of ICASSP*, Mar.

Olivier Pietquin and Thierry Dutoit. 2006. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(2):589–599.

P. J. Price. 1990. Evaluation of spoken language systems: The atis domain. In *Proc. of the Workshop on Speech and Natural Language*, HLT '90.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proc. of EMNLP*.

Kenji Sagae, Maider Lehr, E Prud'hommeaux, Puyang Xu, Nathan Glenn, Damianos Karakos, Sanjeev Khudanpur, Brian Roark, Murat Saraclar, Izhak Shafran, et al. 2012. Hallucinated n-best lists for discriminative language modeling. In *ICASSP*, pages 5001–5004. IEEE.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007a. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *NAACL*, pages 149–152. Association for Computational Linguistics.

Jost Schatzmann, Blaise Thomson, and Steve Young. 2007b. Error simulation for training statistical dialogue systems. In *ASRU*, pages 526–531. IEEE.

Konrad Scheffler and Steve Young. 2000. Probabilistic simulation of human-machine dialogues. In *ICASSP*, volume 2, pages II1217–II1220. IEEE.

Konrad Scheffler and Steve Young. 2001. Corpus-based dialogue simulation for automatic strategy learning and evaluation. In *NAACL*, pages 64–70.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proc. of EMNLP*, pages 254–263.

Ashwin K. Vijayakumar, Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David J. Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *CoRR*, abs/1610.02424.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using# petpeeve tweets. In *Proc. of EMNLP*, pages 2557–2563.

Kaisheng Yao, Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu. 2013. Recurrent neural networks for language understanding. In *Interspeech*, pages 2524–2528.

K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi. 2014. Spoken language understanding using long short-term memory neural networks. In *2014 IEEE SLT*, pages 189–194, Dec.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proc. of the IEEE*, 101(5):1160–1179.

Feifei Zhai, Saloni Potdar, Bing Xiang, and Bowen Zhou. 2017. Neural models for sequence chunking. In *AAAI*, pages 3365–3371.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

Dialogue-act-driven Conversation Model: An Experimental Study

Harshit Kumar
IBM Research,
Delhi, India
harshitk@in.ibm.com

Arvind Agarwal
IBM Research,
Delhi, India
arvagarw@in.ibm.com

Sachindra Joshi
IBM Research,
Delhi, India
jsachind@in.ibm.com

Abstract

The utility of additional semantic information for the task of next utterance selection in an automated dialogue system is the focus of study in this paper. In particular, we show that additional information available in the form of dialogue acts –when used along with context given in the form of dialogue history– improves the performance irrespective of the underlying model being generative or discriminative. In order to show the model agnostic behavior of dialogue acts, we experiment with several well-known models such as sequence-to-sequence encoder-decoder model, hierarchical encoder-decoder model, and Siamese-based models with and without hierarchy; and show that in all models, incorporating dialogue acts improves the performance by a significant margin. We, furthermore, propose a novel way of encoding dialogue act information, and use it along with hierarchical encoder to build a model that can use the *sequential* dialogue act information in a natural way. Our proposed model achieves an MRR of about 84.8% for the task of next utterance selection on a newly introduced DailyDialog dataset, and outperform the baseline models. We also provide a detailed analysis of results including key insights that explain the improvement in MRR because of dialogue act information.

1 Introduction

In the last decade, natural language processing and machine learning –in particular deep learning– have come a long way towards building an automated dialogue system. In a fully automated dialogue system, the goal is to predict an appropriate response given the dialogue history. This problem of response prediction can be formulated in two ways. One is purely generative, where the task is to *generate* a text response, i.e. generating a sentence or utterance from scratch, whereas the other is Next Utterance Selection, where the task is to *select* an appropriate response from a set of given candidates. Despite significant research in text generation, a pure generative model capable of generating syntactically and semantically correct text still remains a distant reality. There have been several efforts such as (Vinyals and Le, 2015; Serban et al., 2016a; Serban et al., 2016b; Serban et al., 2017b) for the task of dialogue generation, however these models still do not seem to work in practice (Liu et al., 2016). This is particularly true for open domain dialogue systems. Dialogue generation in a task-oriented oriented dialogue system, such as flight-booking and troubleshooting, is much easier than in a non-task oriented dialogue system. This level of difficulty arises because a non-task-oriented dialogue system has no predefined goal (or domain), and the vocabulary and possibilities of the dialogues could be endless. Given these challenges, researchers have defined a simpler problem for conversation modeling based on retrieval, i.e. next utterance selection. In this paper we use this second formulation of the problem, and show that using additional information available in the form of dialogue acts help in improving the performance of the underlying model.

Dialogue acts (DA) are higher level semantic abstractions assigned to utterances in a conversation. An example of a dialogue act for an utterance *i'll give you a call tonight* is *Inform* since speaker is providing information. In a traditional dialogue system, where dialogues are formulated by first sentence planning

Proceedings of the 27th International Conference on Computational Linguistics, pages 1246–1256
Santa Fe, New Mexico, USA, August 20-26, 2018.

and then by surface realization, the first step is to understand the dialogue act of the utterance that needs to be generated, and then plan and realize the dialogue accordingly. To better understand the importance of dialogue acts, consider an example of a simple conversation, where if the previous utterance is of type *Question* then the next utterance is most likely going to be of the type, i.e. *Inform*, providing information to that question. Knowing that the next utterance is of type *Inform*, a conversation system with support of dialogue act information can filter a set of candidate responses, and select the most appropriate one. Driven by this intuition, we hypothesize that understanding dialogue acts and using them in the task of next utterance selection should improve the performance irrespective of the underlying model.

Driven by this intuition, we hypothesize that understanding dialogue acts and using them in the task of next utterance selection should improve the performance irrespective of the underlying model.

Most of the existing literature for the task of next utterance selection can be classified into two categories. First is based on Sequence-to-sequence models (generative models) (Serban et al., 2016a; Serban et al., 2017a; Vinyals and Le, 2015), where a model is trained to *generate* a response given context; and the other is Siamese models (discriminative models) (Lowe et al., 2017), where a model is trained to *discriminate* between positive and negative responses for a similar context. In both types of models, at test time, a set of candidate responses is provided consisting of one correct response and several incorrect responses, and the model is evaluated on its ability to assign a higher rank to the true response.

In this paper, through the experimentation with both generative and discriminative types of models, we validate the hypothesis that additional information available in the form of dialogue act significantly improves the performance irrespective of the underlying model. In addition to showing the utility of dialogue acts, we propose a novel model that can use the sequential dialogue act information in a natural way. More specifically, we propose a dialogue-act-driven hierarchical Siamese model. Hierarchical models have shown to perform better than non-hierarchical models for the task of dialogue generation, whereas Siamese models have been shown to outperform the encoder-decoder based models for the task of next utterance selection. In this paper, we combine both of these models, and further enhance them with a dialogue act encoder. The proposed model has a hierarchical encoder which encodes the past utterances, and combine them with the representation of additional contextual information, obtained from the dialogue acts associated with the past utterances, to discriminate the correct response from the incorrect ones. Our proposed model provides us the best of both worlds and outperforms the baseline models by a significant margin. Among others, a key contribution of this paper is that we do a deeper analysis of the reasons for the performance improvement due to inclusion of dialogue act and draw several important key insights such as, dialogue acts induce uniformity in the data, they aid in learning the right patterns. We believe that these insights would inspire new research in this field and push the boundary even further. The main contributions of this paper are as follows:

1. For the task of next utterance selection, we validate the hypothesis that additional information available in the form of dialogue acts improves the performance irrespective of the underlying models.

2. We propose a novel model that combines the strength of Siamese network with strengths of hierarchical structure inherent in the conversations and dialogue act information. The model gives us the best of all, and outperforms the baseline models by a significant margin on the DailyDialog Dataset.

3. We perform a deeper analysis of the utility of the dialogue act information and draw three key insights: models learn dominant dialogue act patterns; dialogue acts induce uniformity; dialogue acts reinforce correct dialogue act patterns.

4. We modify the DailyDialog (Li et al., 2017b) dataset for the task of next utterance selection, and release it publicly along with the code-base of the proposed model[1]. We believe that this dataset will work as a benchmark dataset for further research on this problem. Similar benchmark datasets have been released earlier(Lowe et al., 2015; Serban et al., 2015), however they do not come with dialogue act information.

[1] https://github.com/hk-bmi/ddialog-da-generation

2 Approach

In this section, we provide details of several existing models that we will use to validate our hypothesis. These models include generative models (such as encoder-decoder model and its hierarchical version i.e., hierarchical encoder-decoder) and discriminative model (Siamese-based model). Next, we provide details of the proposed model that adds the hierarchical structure to the Siamese model along with the dialogue act information. To set the notations, we are given a set $\mathcal{D}$ of N conversations, i.e. $\mathcal{D} = (C^1, C^2, \ldots C^N)$, with each conversation C^i being a sequence of R_i utterances, $C^i = (u_1, u_2, \ldots u_{R_i})$. Each utterance u_j in turn is itself a sequence of S_j words, i.e. $u_j = (w_1, w_2, \ldots w_{S_j})$.

2.1 Generative Model

Generative models are the most widely used models for conversation modeling. These models include encoder-decoder model and hierarchical encoder-decoder model.

2.1.1 Encoder-decoder Model

An encoder-decoder is a *generative* model that works on the idea of obtaining a representation of an input and use it for generating an output. It has two main components, encoder and decoder. The encoder encodes the first K utterances, and the decoder uses that encoding to generate the next $K + 1^{th}$ utterance. In a conversation, all words in first K utterances can be stringed together to form a single long chain and passed to an RNN encoder as following:

$$\begin{aligned} e_k &= f^1_{embed}(w_k) \quad \forall k \in 1, 2, \ldots \\ h^e_k &= f^1_{rnn}(h^e_{k-1}, e_k) \quad \forall k \in 1, 2, \ldots \end{aligned} \tag{1}$$

where, f^1_{embed} represents the embedding layer, whereas f^1_{rnn} is the encoder (RNN). Let v be the final output of the encoder which is considered as a representation of the entire context, and used to initialize the decoder (another RNN). Mathematically, the sequence of operations at the decoder are as follows:

$$\begin{aligned} h^d_0 &= v \\ h^d_k &= f^2_{rnn}(h^d_{k-1}, f^2_{embed}(w_k)) \quad \forall k \in 1, 2, \ldots n-1 \\ P_k &= Logistic(h^d_k). \end{aligned} \tag{2}$$

Here, f^2_{embed} represents the embedding layer. $Logistic$ is the final layer, which outputs the probability distribution over the vocabulary. Encoder-decoder models are trained to maximize the likelihood of generating the next utterance, however, for the task of next utterance selection, they are tested based on the probability of generating the candidate utterances.

2.1.2 Hierarchical Encoder-decoder Model

A simple encoder-decoder treats the first K utterances as a single long chain of words, and therefore fails to leverage the hierarchical structure, which is an inherent part of a conversation. Hierarchy is important for conversation modeling since it captures the natural dependency among utterances. Several researchers (Sordoni et al., 2015; Serban et al., 2016b; Serban et al., 2017b; Dehghani et al., 2017; Kumar et al., 2017) have shown that hierarchical models outperform standard non-hierarchical models. Hierarchical models use two encoders to capture the hierarchical structure. The first encoder, referred as utterance encoder, operates at the utterance level, encoding each word in each utterance. The second encoder, referred as conversation encoder, operates at the conversation level, encoding each utterance in the conversation, based on the representations of the previous encoder. These two encoders make sure that the output of the conversation encoder captures the dependencies among utterances. For a given conversation, each word w_k of each utterance u_j is processed by an embedding layer, followed by an RNN which serves as the utterance encoder. Similar to the encoder in equation (1), an utterance encoder gives us a sequence of representations $v_1, v_2, \ldots v_K$, corresponding to the first K utterances $u_1, u_2, \ldots u_K$ in a conversation. These representations are passed on to the conversation encoder, another RNN, which transforms v_j to another representation g_j. The representation obtained from the last time-step of the

conversation-level encoder i.e. g_K is considered as the representation of the entire conversation and used to initialize the decoder which works in the same way as Equation 2.

2.2 Discriminative Model

A decoder in the encoder-decoder model generates the next word given the context, and though it has several valid and reasonable choices, it is burdened with the task of generating exactly a particular choice that matches the ground truth. For example, for a context *I am enjoying the day, it is warm and sunny*, if decoder generates *yes, it is.* and the ground truth dictates *yes, indeed, it is a lovely day*, the decoder has failed, though it is a valid response. Due to these challenges with generative models, discriminative models are trained directly to discriminate between positive and negative utterances. A typical discriminative model, or in particular Siamese model, consists of two encoders, one encoder encoding the context, while another encoding the candidate utterance, i.e utterance $K+1$. These two representations are passed to a final layer that computes the probability of candidate being a valid response given the context. Let $h^{(1)}$ and $h^{(2)}$ be the representations obtained from the first encoder and second encoder, respectively, then the probability of their association can be computed using the following expression.

$$p(s|h^{(1)}, h^{(2)}) = \sigma\big({h^{(1)}}^T A h^{(2)} + b\big) \tag{3}$$

where, the bias b and matrix A are learned model parameters.

2.3 Dialog-act-driven Models

Dialogue acts are higher level abstractions assigned to utterances. In our problem setting, we are given a list of dialogue acts $da_1, da_2, \ldots da_K$, corresponding to first K utterances in the conversation. These dialogue acts are treated as an additional sequence of signals that can aid in the learning process, and are passed through an encoder, denoted as Dialog-Act encoder (DA-encoder). The DA-encoder works on the same principle as the utterance encoder. It builds a dialogue act vocabulary and uses that to learn dialogue act embeddings. Similar to the utterance encoder, the input to the DA-encoder are one hot encodings of the dialogue acts, which are then passed through an embedding layer to learn DA embeddings. These DA embeddings are sent to an RNN to learn dialogue act representations. The sequence of operations for the DA-encoder are as follows:

$$\begin{aligned} e_{da_k} &= f^3_{embed}(da_k) \quad \forall k \in 1, 2, \ldots K \\ h_{da_k} &= f^4_{rnn}(h_{da_{k-1}}, e_{da_k}) \quad \forall k \in 1, 2, \ldots K. \\ q_K &= h_{da_K} \end{aligned} \tag{4}$$

The output of the DA-encoder at the last time step (q_K) gives us the representation of the entire DA sequence which is then used in the further modeling process in generative and discriminative models. In generative models, it is used in the decoder by concatenating g_K and q_K, whereas in discriminative models, it is used along with encoder's output by combining g_K with q_K through a linear combination.

2.4 Dialog-act-driven Hierarchical Siamese - Proposed Model

Our proposed model, i.e. Dialog-act-driven Hierarchical Siamese Model (HSiamese-DA), uses the following three components: a hierarchical encoder to obtain a representation that captures the dependencies among K utterances; an utterance encoder to obtain a representation of the candidate response, $(K+1)^{th}$ utterance; a DA-encoder (Equation 4) that captures the dependencies among the dialogue acts of the first K utterances. Let the representation obtained from the hierarchical encoder, DA-encoder and utterance encoder be g_K, q_K and v_{K+1}, respectively. The two representations, g_K and q_K, are linearly combined to obtained a compositional representation of the context, which is then used along with candidate representation to compute the probability of associating the candidate response with the context using following expression:

$$\begin{aligned} d_K &= \alpha * g_K + (1-\alpha) * q_K \\ p(s|d_K, v_{K+1}) &= \sigma(d_K^T \cdot A \cdot v_{K+1} + b) \end{aligned} \tag{5}$$

The model is trained by minimizing the cross-entropy of all labeled conversations including positive and negative examples. At the test time, each conversation has K utterances followed by a set of 10 candidates responses. The system is tested in its ability to assign a higher rank to the true response.

3 Experiments

In this section, we describe the details of the experiments, i.e. dataset and its preparation, baseline models, experimental setup, and analysis of results.

3.1 Dataset

In our problem setting, we require a dataset that is of reasonable size[2] and has utterances annotated with the corresponding dialogue acts. Although there are several available datasets(Serban et al., 2015), such as SwDA (Switchboard Dialogue Act Corpus (Jurafsky, 1997)), MRDA (Meeting Recorder Dialogue Act corpus (Janin et al., 2003)), Ubuntu(Lowe et al., 2015), OpenSubtitles(Tiedemann, 2009), etc., they are not really suitable for our problem setting. Most of these datasets do not come with dialogue acts, and the ones which do (i.e. SWDA and MRDA) are small in size. Note that the SwDA and MRDA datasets contain 1003 and 51 conversations, respectively. To the best of our knowledge, a recently released dataset, DailyDialog(Li et al., 2017b), is the only dataset that has utterances annotated with dialogue acts and is large enough for conversation modeling methods to work. Furthermore, in this dataset, conversations are non-task oriented, and each conversation focuses on one topic. Each utterance is annotated with four dialogue acts as described in Table 1. The dataset has train, validation, and test splits of 11118, 1000, and 1000 conversations, respectively. We evaluate and report our results on the DailyDialog dataset.

In this paper, we hypothesize that dialogue acts improve conversation modeling. However, it is not always possible that such dialogue acts are available in practice, and it would be ideal to predict dialogue acts first (Kumar et al., 2017), and then use them for next utterance generation/retrieval; having a model where both tasks, i.e. prediction and generation, are performed simultaneously may not be ideal for validating the hypothesis. Note that the error from the dialogue act prediction may propagate to the next utterance generation/retrieval. Therefore, we intentionally did not use the predicted dialogue acts (rather used the available dialogue acts) to make sure that the insights about the usefulness of the dialogue acts are not corrupted due to the error in the upstream prediction model.

Dialogue Act	Description
Inform	A speaker is providing information by means of a question or statement
Question	A speaker intends to obtain information by asking a question
Directive	A speaker is requesting, accept/reject offer, or making a suggestion
Comissive	A speaker accept/reject a request or suggestion

Table 1: Dialogue Acts and their description available in the DailyDialog Dataset.

3.1.1 Dataset Preparation for Next Utterance Selection Task

The DailyDialog dataset in its original form is not directly useful for the task of next utterance selection, and hence requires preparation. The dataset has the dialogues from both the speakers. Owing to the different conversational style of human and conversation agent, our objective is to build a model that is specific to the agent, i.e. bot. Therefore, we need to modify the dataset in such a way that we only consider those turns where we need to predict the bot's utterance. To clarify further, consider the example conversation given in Table 2. The conversation has 8 utterances, and each utterances is marked with the speaker, i.e. human (H) and bot (B). Since we are only interested in building bot-specific model, we only pick those subsequences from this conversation where the last utterance is "B". This gives us three subsequences: 1,2,3,4; 3,4,5,6; 5,6,7,8 for a context of size 3. In each of these sub-conversations,

[2]Generative models such as sequence-to-sequence or discriminative models such as Siamese only perform better when there is large amount of data for training.

the first three utterances constitute the context, while the last utterance is the true response. Our training data consists of such subsequences made up of 4 utterances. In the test data, each subsequence, in addition to these 4 utterances, has 9 more utterances selected randomly from the test pool, therefore a total of 13 utterances. These 9 utterances along with the 4^{th} response (i.e., true response) utterance constitute the candidate pool. With this data preparation exercise, the total number of conversations in train, test, and valid are 30515, 2849, and 2695, respectively. This version of dataset is used for training and testing generative models. For discriminative models, data required is a bit different. The training of discriminative models require positive examples and an equal number of negative examples. Note that training data of the generative models did not have any negative examples. In-order to prepare negative examples, we replicate each conversation by replacing the last utterance with a random utterance from the test data. The test and valid dataset remain as in the generative models. Thus, with this data preparation exercise, the total number of conversations in train, test, and valid are 61030, 2849, and 2695, respectively.

Id		Utterance	DA
1	H	Hello, this is Mike, Kara .	I
2	B	Mike! Good to hear from you. How are you?	Q
3	H	Everything is fine, and how are you?	Q
4	B	Things are going well with me.	I
5	H	Kara, I had fun the other night at the movies and was wondering if you would like to go out again this Friday.	D
6	B	Mike, I don't think that it's a good idea to go out again.	C
7	H	Maybe we could just meet for coffee or something.	D
8	B	I can't really deal with any distractions right now, but I appreciate the nice evening we spent together.	C

Table 2: A snippet of a conversation showing few dialogues between a Human (H) and Bot (B).

3.2 Baseline models and Proposed Model

Here we list the baseline models, their modified version enhanced with dialogue act information, and the proposed model.

Generative Models:

- **ED** - It is a vanilla sequence to sequence model that uses an utterance encoder to obtain a representation of first K utterances which is then used in a decoder to generate next utterance.
- **HRED** - An extension of sequence to sequence model that uses a hierarchical encoder to obtain a representation of first K utterances, which is then used in decoder to generate next utterance.
- **ED-DA** - An extension of the ED model which uses dialogue act information. It has a conditional decoder, that conditions the generation of each word on the dialogue acts representation.
- **HRED-DA** - An extension of the HRED model which uses dialogue act information. Similar to ED-DA, it also has a conditional decoder that conditions the generation of each word on the dialogue acts representation.

Discriminative Models

- **Siamese** - Also known as Dual-Encoder, it uses two encoders (both utterance encoders) with shared weights, to produce the representation for the K utterances and the $(K+1)$ utterance.
- **HSiamese** - A Hierarchical version of the Siamese model that uses a hierarchical encoder to produce a representation for the K utterances, and a plain encoder (utterance encoder) to produce a representation for the $(K+1)$ utterance.

- **Siamese-DA** - An extension of Siamese model that uses the additional dialogue act information obtained through DA-encoder. The representation obtained from the DA-encoder is linearly combined with the representation of the K utterances obtained from an utterance encoder.
- **HSiamese-DA** - The proposed model uses a Hierarchical Encoder and a DA-Encoder. The representation obtained from the DA-Encoder is linearly combined with the representation obtained from the hierarchical encoder.

3.3 Hyper-parameter Tuning

In our experiments, the parameters are tuned on validation set while the results are reported on test set. Each utterance in a mini-batch was padded to the maximum length for that batch. The maximum batch size allowed was 32. The word vectors were initialized with the 300-dimensional Glove embeddings (Pennington et al., 2014), and were also updated during training. For the generative models, the utterance encoder, conversation encoder, DA-Encoder and decoder are all GRUs with rnn_size set to 1000 (optimized over 100 to 1200 in steps of 100). For the discriminative model, the utterance encoder, conversation encoder, and DA-Encoder are all GRUs with rnn_size set to 300 (optimized over 100 to 500 in steps of 100). Dropout of 0.1 (optimized over 0.0 to 0.7 in steps of 0.1) was applied to embeddings obtained from the output of conversation encoder. Note that, dropout was not used in the discriminative model and its variations. Models were trained to minimize cross entropy using Adam optimizer with learning rate of 0.0003 (optimized over 0.0001, 0.0003, 0.0005, 0.0007, 0.001). We found that a higher learning rate up-to 0.0005 helps the model to learn quickly, whereas learning rate greater than 0.0005 leads to oscillations.

3.4 Results and Discussion

In this section, we present results of our experimental study, followed by its analysis.

3.4.1 Performance Evaluation

Since our problem formulation is retrieval based, we use standard IR metrics such as Mean Reciprocal Rank (MRR) and Recall@k as our evaluation metrics. MRR is calculated as the mean of the reciprocal rank of the true candidate response among other candidate responses. Recall@k measures whether the true candidate response appears in a ranked list of k responses.

In this work, our hypothesis is that additional information about utterances available in the form of dialogue acts helps irrespective of the underlying model, i.e. generative or discriminative. Results in Table 3 support our hypothesis. These results clearly indicate that the MRR of the true candidate response improves when dialogue acts of previous utterances are provided. From these tables we see that for all underlying models, the dialogue act version performs better than non dialogue act version. These results furthermore indicate that hierarchical version performs better than non-hierarchical version for both generative and discriminative models. In the generative case, the plain ED has an MRR of 0.474, whereas the same model, when conditioned with DA-Encoder, has an MRR of 0.54, an improvement of 13.9%. The hierarchical encoder-decoder HRED and HRED-DA has an MRR of 0.523 and 0.583, respectively, an improvement of 11.4%. Generative models are sequence-to-sequence models and rather complex in nature, so it is interesting to note that even a much simpler discriminative model, i.e. plain Siamese model, without any dialogue act information, has an MRR of 0.8 compared to 0.58 of the best performing generative model, i.e. HRED-DA. This observation demonstrates the strength of the discriminative models, and therefore is a motivation behind the proposed model. The proposed model improves these baseline numbers by incorporating hierarchy and dialogue act information, and pushes the MRR to 0.848.

3.4.2 Performance Analysis

While we have shown that using dialogue act information does help in the next utterance selection task, in this section, we dig deeper and understand reasons for it. In order to do that, we analyze the dialogue act distribution of the test data and model outputs. Although all K dialogue acts corresponding to K utterances in the context might play a role in ranking candidate utterances, the following analysis only

	MRR	R@1	R@2	R@5
ED	0.474	0.327	0.405	0.639
ED-DA	0.54	0.407	0.478	0.690
HRED	0.523	0.384	0.471	0.676
HRED-DA	0.583	0.448	0.542	0.742

(a) Generative Models

	MRR	R@1	R@2	R@5
Siamese	0.800	0.711	0.785	**0.949**
Siamese-DA	0.844	0.784	**0.824**	0.944
HSiamese	0.817	0.743	0.792	0.948
HSiamese-DA	**0.848**	**0.795**	0.821	0.932

(b) Discriminative Models

Table 3: Comparison of different models with and without dialogue acts

uses the pairs of dialogue acts, i.e. dialogue acts of K^{th} and $(K+1)^{th}$ utterances. Tables 4(a), 4(b) and 4(c) show the distribution of such dialogue act pairs for test data, HSiamese, and HSiamese-DA models respectively. Here, rows indicate the dialogue act of K^{th} utterance, whereas columns indicate the dialogue act of $(K+1)^{th}$ utterance. A cell value indicates the count of utterance pairs with the respective dialogue act combinations where $(K+1)^{th}$ utterance was ranked 1. Note that in the test data, $(K+1)^{th}$ utterance is the true candidate response and always have the rank 1. For instance, there are 742 utterance pairs in the test data, where K^{th} and $(K+1)^{th}$ utterances have dialogue acts Q and I, respectively, however, out of those 742 instances, HSiamese ranked only 605 as 1 while HSiamese-DA ranked 638 as 1. From these tables, we draw following observations.

	I	Q	D	C
I	600	336	176	16
Q	742	73	87	2
D	26	75	56	305
C	69	50	76	6

(a) Ground-truth test data

	I	Q	D	C	R@1
I	403	208	115	9	0.65
Q	605	34	64	2	0.78
D	16	41	42	192	0.63
C	48	33	56	4	0.70

(b) HSiamese

	I	Q	D	C	R@1
I	446	253	121	12	0.74
Q	638	43	61	2	0.82
D	17	58	42	243	0.78
C	52	39	68	5	0.82

(c) HSiamese-DA

Table 4: Number of rank-1 conversations and theie DAs for K^{th} and $(K+1)^{th}$ utterances.

Models Learn Dominant Patterns: The first is that there are certain dominant communication patterns that we observe in both, test data and model outputs (See Table 4), suggesting that models are able to learn these patterns and retain them in their outputs . We observe that a *Question* is often followed by an *Information*, whereas an *Information* can be followed by another *Information* or a *Question*. A *Directive* tends to be followed by *Commissive*. These communication patterns not only make sense intuitively but they are also in agreement with previous studies (Li et al., 2017b; Ribeiro et al., 2015).

Dialogue Acts Bring Uniformity: The second and a rather more important observation is that dialogue acts help the most for the dialogue act class (DA-class) when the utterances belonging to that class are non-uniform in their linguistic construct. In order to better exlpain this, we first compute the break-up of recall@1 according to the dialogue act classes. A DA-class of a conversation in the test data is defined based on the dialogue act of the last utterance (K^{th} utterance) in the context. These numbers are shown in the last column of Tables 4(b) and 4(c) for the respective models. In Table 4(b), first row in recall@1 column is 0.65, which indicates that out of the total number of test conversations where dialogue act of the last utterance of context was I, 65% of true candidate responses were ranked 1 by the HSiamese model. Such a DA-class wise breakup of the recall@1 numbers helps us do an analysis with respect to individual DA-classes. From this break-up, it is clear that for the HSiamese model, *Question* DA-class has the best performance of 78% whereas *Directive* has the worst performance of 63%. This difference can be attributed to the fact that all utterances with dialogue act as *Question* have rather uniform construct. Some examples of *Question* utterances are, '*Q: Do you have a fever?*' and '*Q: Why do you want to work for our company?*', while the examples of *Directive* utterances are, '*D:when we have the final results, we will call you.*' and '*D:we will take the trip. could you give us a pamphlet?*'. From these examples, we observe that utterances belonging to DA-class *Question* have rather uniform construct in terms of linguistic features, whereas utterances belonging to DA-class *Directive* are ambiguous –some of the utterances of type *Directive* can be easily confused for *Question*. This uniformity makes the learning task easier for *Question* class, and thereby giving us better results in the next utterance selection

task, even for the model that does not use the dialogue act information. This performance difference reduces when we provide the dialogue act information along with the textual content (See Table 4(c)). Inclusion of dialogue act information within a non-uniform class such as *Directive* brings in uniformity, and therefore, results in significant performance improvement. In the case of *Directive*, we see as much as 15 percentage point improvement. Similar improvement pattern has been observed for DA-class *Information* and *Comissive*. Similar to *Directive*, utterances belonging to *Comissive* have rather non-uniform construct, and with the availability of dialogue acts, this DA-class is able to gain much more than the *Information* DA-class.

	I	Q	D	C	Total
I	43	45	6	3	97
Q	33	9	-3	0	39
D	1	17	0	51	69
C	4	6	12	1	23
Total	81	77	15	55	228

Table 5: Difference of number of rank-1 conversations between HSiamese-DA and HSiamese.

Dialogue Acts Help Model Learn the Right Patterns: In Table 5, we show the relative improvement of HSiamese-DA model over HSiamese. From this table, we observe that there are a total of 228 conversations where the proposed model was able to improve the ranking of true candidate response to 1. We further observe that the biggest improvement is in $I \rightarrow I$, $I \rightarrow Q$, $Q \rightarrow I$, and $D \rightarrow C$, which make sense intuitively. These are dominant patterns observed in the training data which should be preserved in the model output as well, however these patterns will only be preserved when model is able to capture the *correct* dialogue act information. Since in many cases D and Q have similar construct, without explicit dialogue act information, a model may get confused and may learn patterns not observed in the training data. For example, $Q \rightarrow I$ and $D \rightarrow C$ are the dominant and right patterns in the training data, however in the absence of explicit dialogue act information, the model may get confused between D and Q and may learn $D \rightarrow I$ and $Q \rightarrow C$ instead of the dominant patterns i.e. $Q \rightarrow I$ and $D \rightarrow C$. With the explicit dialogue act information, this ambiguity is alleviated and model learns the right patterns as demonstrated by Table 5. Similar observations are true for other two constructs, i.e. *Information* and *Commisive*. Both are rather similar in construct, '*I: No, thank you*', '*I: It doesn't matter. it happens to everyone .*' and '*C: I knew you'd see it my way .*', '*C: Ok , i am ready to think of other things .*', and there is no obvious distinguishing factor. However, providing explicit DA information helps in disambiguation, and learn the patterns that are observed in the training data such as $I \rightarrow I$, $I \rightarrow Q$.

4 Related Work

In conversation modeling, the most basic problem is to generate a response given a context. Several efforts have been made towards solving the problem of dialogue generation(Vinyals and Le, 2015; Liu et al., 2016; Li et al., 2015), however, due to the inherent difficulty of the problem, these efforts have only had limited success and are known to have issues like generating repetitive and generalized responses such as *I don't know* or *Ok*.

For the task of Next Utterance Selection, which is a relatively simpler problem than generation, though existing generative models can be easily adopted, their counterpart discriminative models have shown to have better performance. In generative models, the most notable work is from (Vinyals and Le, 2015), however this work considers the context as a flat long string of words and ignores the hierarchical structure. Researchers have proposed hierarchical model (Serban et al., 2016b) and their variations (Serban et al., 2017b; Serban et al., 2017a; Li et al., 2017a) but none of these models take into account the dialogue act information. In Discriminative models, such as Siamese, a very notable work by (Kannan et al., 2016), *smart reply*, retrieves the most likely response from a set of candidate response clusters. (Lowe et al., 2017) has used a retrieval based Siamese model and shown its results on the Ubuntu corpus. Our

proposed model builds upon the strengths of generative and discriminative models, and uses hierarchy along with the dialogue act information to achieve the best performance. A recent work by (Zhao et al., 2017) has used dialogue acts for the task of dialogue generation. Our work complements their findings, and further show that dialogue acts improve the model performance across the board irrespective of underlying model (i.e. generative or discriminative models) and for the task of next utterance selection.

5 Conclusion

For the task of next utterance selection, we show that dialogue acts helps achieve better performance irrespective of the underlying model, be it generative or discriminative. We also propose a novel discriminative model that leverages the hierarchical structure in a conversation and dialogue act information to produce much improved results, an MRR of 0.848. Our results not only show the improvement in performance, but we also present key reasons for it by doing a detailed analysis and drawing key insights that the inclusion of dialogue act information induces uniformity and removes ambiguity.

References

Mostafa Dehghani, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. 2017. Learning to attend, copy, and generate for session based query suggestion. In *CIKM*.

Adam Janin, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, et al. 2003. The icsi meeting corpus. In *ICASSP*.

Dan Jurafsky. 1997. Switchboard swbd-damsl shallow-discourse-function annotation coders manual. *www. dcs. shef. ac. uk/nlp/amities/files/bib/ics-tr-97-02. pdf*.

Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, László Lukács, Marina Ganea, Peter Young, et al. 2016. Smart reply: Automated response suggestion for email. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 955–964. ACM.

Harshit Kumar, Arvind Agarwal, Riddhiman Dasgupta, Sachindra Joshi, and Arun Kumar. 2017. Dialogue act sequence labeling using hierarchical encoder with crf. *arXiv preprint arXiv:1709.04250*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. In *NAACL*.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017a. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017b. Dailydialog: A manually labelled multi-turn dialogue dataset. *arXiv preprint arXiv:1710.03957*.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Ryan Lowe, Nissan Pow, Iulian V Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 285.

Ryan Thomas Lowe, Nissan Pow, Iulian Vlad Serban, Laurent Charlin, Chia-Wei Liu, and Joelle Pineau. 2017. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue & Discourse*, 8(1):31–65.

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

Eugénio Ribeiro, Ricardo Ribeiro, and David Martins de Matos. 2015. The influence of context on dialogue act recognition. *arXiv preprint arXiv:1506.00839*.

Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. 2015. A survey of available corpora for building data-driven dialogue systems. *arXiv preprint arXiv:1512.05742*.

Iulian Vlad Serban, Ryan Lowe, Laurent Charlin, and Joelle Pineau. 2016a. Generative deep neural networks for dialogue: A short review. *arXiv preprint arXiv:1611.06216*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016b. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*.

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron C Courville. 2017a. Multiresolution recurrent neural networks: An application to dialogue response generation. In *AAAI*, pages 3288–3294.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017b. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *CIKM*.

Jörg Tiedemann. 2009. News from opus-a collection of multilingual parallel corpora with tools and interfaces. In *Recent advances in natural language processing*, volume 5, pages 237–248.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 654–664.

Structured Dialogue Policy with Graph Neural Networks

Lu Chen*, **Bowen Tan***, **Sishan Long and Kai Yu**
Key Lab. of Shanghai Education Commission for Intelligent Interaction and Cognitive Eng.
SpeechLab, Department of Computer Science and Engineering
Brain Science and Technology Research Center
Shanghai Jiao Tong University, Shanghai, China
{chenlusz, tanbowen, longsishan, kai.yu}@sjtu.edu.cn

Abstract

Recently, deep reinforcement learning (DRL) has been used for dialogue policy optimization. However, many DRL-based policies are not sample-efficient. Most recent advances focus on improving DRL optimization algorithms to address this issue. Here, we take an *alternative* route of designing neural network structure that is better suited for DRL-based dialogue management. The proposed *structured deep reinforcement learning* is based on graph neural networks (GNN), which consists of some sub-networks, each one for a node on a directed graph. The graph is defined according to the domain ontology and each node can be considered as a sub-agent. During decision making, these sub-agents have internal message exchange between neighbors on the graph. We also propose an approach to jointly optimize the graph structure as well as the parameters of GNN. Experiments show that structured DRL significantly outperforms previous state-of-the-art approaches in almost all of the 18 tasks of the PyDial benchmark.

1 Introduction

A task-oriented spoken dialogue system (SDS) is a system that can continuously interact with a human to accomplish a predefined task, e.g. finding a restaurant or booking a flight. Dialogue management (DM) is the core of an SDS. It has two missions: one is to maintain the dialogue state, and another is to decide how to respond according to a dialogue policy. In this paper, we focus on the dialogue policy.

A dialogue policy can be viewed simply as a set of hand-crafted mapping rules from dialogue states to dialogue actions. This is referred to as a rule-based policy. However, in real-world scenarios, unpredictable user behavior, inevitable automatic speech recognition, and spoken language understanding errors make it difficult to maintain the true dialogue state and make the decision. Hence, in recent years, there is a research trend towards statistical dialogue management. A well-founded theory for this is the partially observable Markov decision process (POMDP) (Kaelbling et al., 1998).

Under the POMDP-based framework, a distribution of possible states – belief state $\mathbf{b}$ is maintained in every dialogue turn. Then reinforcement learning (RL) methods automatically optimize the policy π, i.e. a mapping function from belief state $\mathbf{b}$ to dialogue action $a = \pi(\mathbf{b})$ (Young et al., 2013). Initially, linear RL-based models are adopted, e.g. least-squares policy iteration (LSPI) and natural actor-critic (NAC). However, these linear models have a poor ability of expression and suffer from slow training. Recently, nonparametric algorithms, e.g. Gaussian process reinforcement learning (GPRL), which can optimize policies from a minimal number of dialogues have been proposed. However, the computation cost of GPRL increases with the increase of the number of data. It is therefore questionable as to whether GPRL can scale to support commercial wide-domain SDS.

More recently, deep reinforcement learning (DRL) methods are adopted for dialogue policies (Cuayáhuitl et al., 2015; Fatemi et al., 2016; Zhao and Eskenazi, 2016; Williams et al., 2017; Chen et al., 2017b; Chang et al., 2017). These policies are often represented by fully connected deep neural networks including deep Q-networks and actor-critic networks. DRL-based models are often more

* Both authors contributed equally to this work.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1257–1268
Santa Fe, New Mexico, USA, August 20-26, 2018.

expressive and computational effective. However, these deep models are not sample-efficient and not robust to errors from input modules of SDS. So the focus of these recent advances has been on designing improved RL algorithms to improve the sample-efficiency, e.g. eNAC (Su et al., 2017), BBQ (Lipton et al., 2018).

In this paper, we take an *alternative* but *complementary* approach of focusing primarily on innovating a *structured* neural network architecture that is better suited for dialogue policy. This approach has the benefit that the new network can be easily combined with most existing methods for DRL, which results in *structured deep reinforcement learning*.

The proposed structured DRL is based on graph neural networks (GNN) (Scarselli et al., 2009), which consists of some sub-networks, each one corresponding to a node in the directed graph. The graph has two types of nodes: *slot-dependent* node and *slot-independent* node. The same types of nodes partially share parameters. This can speed up the learning process. In order to model the interaction between sub-networks, they have internal message exchanges between neighbors when doing forward computation.

The main contribution of this paper is three-fold: (1) We proposed a unified *structured* dialogue policy based on GNN. To the best of our knowledge, this is the first work to introduce the graph-based approach for dialogue policy optimization. (2) We introduced a novel method to optimize the graph structure along with the parameters of GNN. (3) Our proposed framework achieves state of the art in PyDial benchmark.

2 Related Work

Recent mainstream studies on statistical dialogue management have been modeling dialogue as a partially observable Markov decision process (POMDP), where reinforcement learning (RL) can be applied to realize automatic dialogue policy optimization (Young et al., 2013).

To achieve efficient policy learning, (Gašić et al., 2010) proposed Gaussian process reinforcement learning (GPRL), where the prior correlations of the objective function given different belief states are defined by the kernel function (Gašić et al., 2010; Gašić and Young, 2014). Recently, deep reinforcement learning (DRL) (Mnih et al., 2015) has been applied in dialogue policy optimization, including value function approximation methods, like deep Q-network (DQN) (Cuayáhuitl et al., 2015; Zhao and Eskenazi, 2016; Fatemi et al., 2016; Chen et al., 2017a; Chang et al., 2017; Peng et al., 2017), and policy gradient methods, e.g. REINFORCE (Williams et al., 2017), advantage actor-critic (A2C) (Fatemi et al., 2016). However, compared with GPRL, most of these models are not sample-efficient. More recently, some methods are proposed to speed up the learning process based on improved DRL algorithms, e.g. eNAC (Su et al., 2017), BBQ (Lipton et al., 2018). In contrast, our proposed method is based on a structured neural architecture, which is complementary to various DRL algorithms.

The most related work is the recently proposed multi-agent dialogue policy (MADP) (Chen et al., 2018). In MADP, the dialogue policy is decomposed into some *slot-dependent* sub-policies and a *slot-independent* sub-policy. While MADP is proposed under the view of multi-agent reinforcement learning, it's can be interpreted as the special instance of our proposed graph-based policy with a fully connected graph structure. That is, our proposed framework is more general. Moreover, the graph structure can be jointly optimized with the parameters of neural networks in our methods.

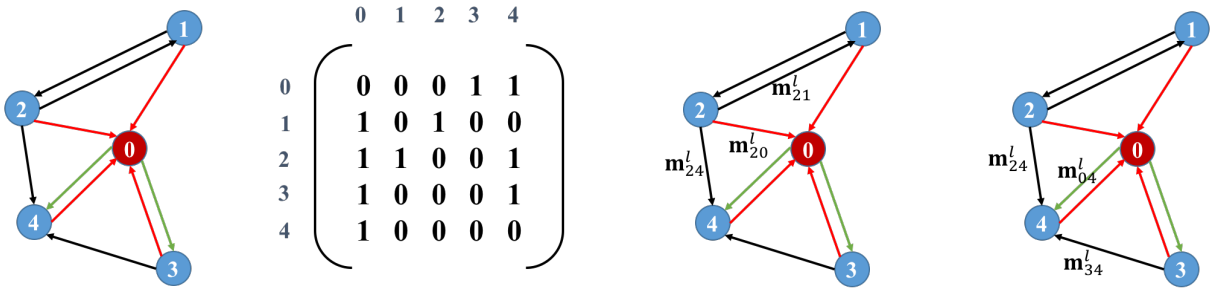
Graph-based approaches are previously adopted for dialogue state tracking, such as the Bayesian update of dialogue state (BUDS) model (Thomson and Young, 2010), structured discriminative model (Lee, 2013). However, they have not been explored for dialogue policy optimization.

3 Proposed Framework

3.1 DRL-based Dialogue Policy

Task-oriented dialogue systems are typically designed according to a structured *ontology* which consists of some concepts (or slots) that a user might wish to use to frame a query. Each slot possesses two attributes: whether it is *requestable* and *informable*. A slot is requestable if the user can request the value of it. An informable slot is one that the user can provide a value for to use as a constraint on their search.

At each dialogue turn, the dialogue state tracker maintains a belief state for each informable slot. Generally, the belief state is a distribution of candidate values for the slot. After the update of belief state,



(a) (b) (c) (d)

Figure 1: (a) An example of directed graph G with 5 nodes and 10 edges. There are two types of nodes: nodes 1∼4 (blue) are *slot-dependent* nodes (S-nodes) and node 0 (red) (I-node) is *slot-independent* node. Accordingly, there are 3 type of edges. (b) The adjacency matrix of G. (c) An example of sending messages. S-node 2 sends messages to its out-going neighbors, i.e. S-nodes 1, 4 and I-node 0. (d) An example of aggregating messages. S-node 4 aggregates messages from its in-coming neighbors, i.e. S-nodes 2, 3 and I-node 0.

the values with the largest belief for each informable slot are used as a constraint to search the database. The matched entities in the database as well as the belief state for slots are concatenated as whole belief dialogue state, which is the input of dialogue policy. Therefore, the dialogue state $\mathbf{b}$ usually can be decomposed into some *slot-dependent* states and a *slot-independent* state, i.e. $\mathbf{b} = \mathbf{b}_1 \oplus \cdots \oplus \mathbf{b}_n \oplus \mathbf{b}_0$. $\mathbf{b}_j (1 \leq j \leq n)$ is the belief state corresponding to j-th informable slot, and $\mathbf{b}_0$ represents the slot-independent state, e.g. database search results.

The output of dialogue policy is a summary action. Similarly, the summary actions can be divided into $n+1$ sets including n slot-dependent action sets $\mathbb{A}_j (1 \leq j \leq n)$, e.g. *request_slot*$_j$, *confirm_slot*$_j$, *select_slot*$_j$, and one slot-independent action set $\mathbb{A}_0$, e.g. *repeat, inform, reqmore, bye, restart*.

Most of previous DRL-based policies don't take advantage of the structure of belief dialogue state and summary actions. The focus in these recent advances has been on designing improved RL algorithms, e.g. eNAC, BBQ. Here, we take an *alternative* but *complementary* approach of focusing primarily on innovating a *structured* neural network architecture that is better suited for dialogue policy. This approach has the benefit that the new network can be easily combined with existing and future algorithms for RL. That is, this paper advances a new network, but uses already published algorithms. In this paper, we adopt deep-Q-networks (DQN). In the next two sections, we will first give the background of DQN and then introduce the proposed structured policy based on graph neural networks (GNN).

3.2 Deep-Q-Networks (DQN)

A Deep Q-Network (DQN) is a multi-layer neural network which maps a belief state $\mathbf{b}$ to the values of the possible actions a at that state, $Q(\mathbf{b}, a; \theta)$, where θ is the weight vector of the neural network. Neural networks for the approximation of value functions have long been investigated (Lin, 1993). However, these methods were previously quite unstable (Mnih et al., 2013). In DQN, two techniques were proposed to overcome this instability, namely experience replay and the use of a target network (Mnih et al., 2013; Mnih et al., 2015).

At every turn, the transition τ including the previous state $\mathbf{b}$, previous action a, corresponding reward r and current state $\mathbf{b}'$ is put in a finite pool $\mathcal{D}$. Once the pool has reached its maximum size, the oldest transition will be deleted. During training, a mini-batch of transitions is uniformly sampled from the pool, i.e. $\tau \sim \mathrm{U}(D)$. This method removes the instability arising from strong correlation between the subsequent transitions of a dialogue. Additionally, a target network with weight vector $\hat{\theta}$ is used. This target network is similar to the Q-network except that its weights are only copied every K steps from the Q-network, and remain fixed during all the other steps. The loss function for the Q-network at each iteration takes the following form:

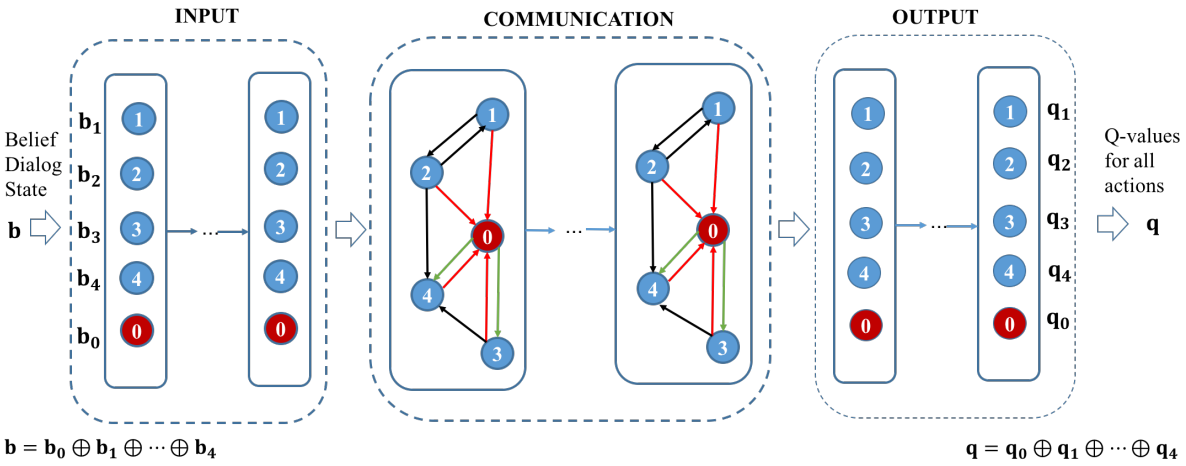


Figure 2: A Q-network with graph neural network (GNN). GNN consists of three modules: input, communication, and output. $\oplus$ denotes concatenation of vectors.

$$\mathcal{L}_{TD}(\theta) = \mathbb{E}_{\tau \sim \mathrm{U}(\mathcal{D})} \left[\left(y - Q_\theta(\mathbf{b}, a) \right)^2 \right], \tag{1}$$

where $y = r + \gamma \max_{a'} Q_{\hat{\theta}}(\mathbf{b}', a')$ and $\gamma \in [0, 1]$ is the discount factor.

Most of the previous work adopts *fully* connected deep neural networks for Q-networks. Here we propose a new network architecture for Q-networks based on *structured* graph neural networks.

3.3 Graph Neural Networks (GNN)

Before delving into details of GNN, we first introduce our notation. We denote the graph structure as $G = (V, E)$ with nodes $v_i (0 \le i \le n) \in V$ and directed edges $e_{ij} \in E$. $\mathcal{N}_{in}(v_i)$ denotes the in-coming neighbors of node v_i, and $\mathcal{N}_{out}(v_i)$ denotes the out-going neighbors of node v_i. $\mathbf{Z}$ is the adjacency matrix of G. The element z_{ij} of $\mathbf{Z}$ is 1 only and only if there is a directed edge from i-th node v_i to j-th node v_j, otherwise z_{ij} is 0. Each node v_i has an associated node type c_i. Similarly, each edge e_{ij} has an edge type u_e, which is determined by node type c_i and node type c_j, i.e. two edges have the same type if and only if their starting node type and their ending node type both are the same.

For dialogue policy, it has two types of nodes: a *slot-independent* node (I-node) and n *slot-dependent* nodes (S-nodes). Accordingly, there are 3 types of edges. Each S-node corresponds to an informable slot in the ontology, while I-node is responsible for slot-independent aspects. Fig.1(a) shows an example of directed graph G with 5 nodes and 10 edges. Nodes 1$\sim$4 (blue) are S-nodes for 4 slots and node 0 (red) is I-node. Fig.1(b) is the corresponding adjacency matrix.

GNN is a deep neural network defined on the graph G. As shown in Fig.3, it consists of three modules: input module, communication module, and output module.

3.3.1 Input Module

At each dialogue turn, each node $v_i (0 \le i \le n)$ will receive the belief dialogue state $\mathbf{b}_i$, which goes through an input module to obtain a state vector $\mathbf{h}_i^0$ as follows:

$$\mathbf{h}_i^0 = f_{c_i}(\mathbf{b}_i), \tag{2}$$

where f_{c_i} is a function for node type c_i, which may be a multilayer perceptron (MLP).

Usually, different slots have different number of candidate values, therefore the dimensions of the inputs of two S-nodes are different. However, in practice, for dialogue policy the probabilities of top K values of each slot play a more important role. Therefore, the whole belief state of each slot is often approximated by the probabilities of *sorted* top K values (Gašić and Young, 2014), which can be implemented by sorted K-max pooling (Kalchbrenner et al., 2014).

3.3.2 Communication Module

The communication module takes $\mathbf{h}_i^0$ as the initial state for node v_i, then update state from one step to the next with following operations.

Sending Messages At l-th step, each node v_i will send messages $\mathbf{m}_{ij}^l$ to node v_j if there is a directed edge from v_i to v_j,

$$\mathbf{m}_{ij}^l = m_{u_e}^l(\mathbf{h}_i^{l-1}), \tag{3}$$

where $m_{u_e}^l$ is a function for edge type u_e at l-th step. For simplicity, we only consider $m_{u_e}^l$ in the form of a linear embedding: $m_{u_e}^l(\mathbf{h}_i^{l-1}) = \mathbf{W}_{u_e}^l \mathbf{h}_i^{l-1}$, where $\mathbf{W}_{u_e}^l$ is a weight matrix to be learned. Fig.1(c) is an example of sending messages for S-node 2. It sends messages to its out-going neighbors, i.e. S-nodes 1, 4 and I-node 0.

Aggregating Messages After sending messages, each node v_j will aggregate messages from its in-coming neighbors,

$$\mathbf{a}_j^l = \sum_{v_i \in \mathcal{N}_{in}(v_j)} \hat{k}_{u_e}^l(\mathbf{h}_i^{l-1}, \mathbf{h}_j^{l-1}) \mathbf{m}_{ij}^l = \sum_{1 \le i \le n} z_{ij} \hat{k}_{u_e}^l(\mathbf{h}_i^{l-1}, \mathbf{h}_j^{l-1}) \mathbf{m}_{ij}^l, \tag{4}$$

where the function $\hat{k}_{u_e}^l$ computes a normalized scalar representing relationship between node v_i and node v_j. There are various versions of $\hat{k}_{u_e}^l$. Here we introduce two instantiations:

- **Mean-Comm**: All messages from in-coming neighbors are treated equally, i.e. $\hat{k}_{u_e}^l(\mathbf{h}_i^{l-1}, \mathbf{h}_j^{l-1}) = \frac{1}{|\mathcal{N}_{in}(v_j)|}$.
- **Attention-Comm**: In practice, some messages are more important than others. Inspired by *self-attention* model for machine translation (Vaswani et al., 2017), here we first compute the similarity of two states in an unified space, i.e.

  $$k_{u_e}^l(\mathbf{h}_i^{l-1}, \mathbf{h}_j^{l-1}) = (\mathbf{W}_{1u_e}^l \mathbf{h}_i^{l-1})^\mathsf{T} (\mathbf{W}_{2u_e}^l \mathbf{h}_j^{l-1}), \tag{5}$$

  then normalize it with softmax:

  $$\hat{k}_{u_e}^l(\mathbf{h}_i^{l-1}, \mathbf{h}_j^{l-1}) = \frac{e^{(\mathbf{W}_{1u_e}^l \mathbf{h}_i^{l-1})^\mathsf{T} (\mathbf{W}_{2u_e}^l \mathbf{h}_j^{l-1})}}{\sum_i e^{(\mathbf{W}_{1u_e}^l \mathbf{h}_i^{l-1})^\mathsf{T} (\mathbf{W}_{2u_e}^l \mathbf{h}_j^{l-1})}}. \tag{6}$$

Fig.1(d) is an example of aggregating messages for S-node 4. It aggregates messages from its in-coming neighbors, i.e. S-nodes 2, 3 and I-node 0.

Updating State After aggregating messages from neighbors, every node v_i will update its state from $\mathbf{h}_i^{l-1}$ to $\mathbf{h}_i^l$,

$$\mathbf{h}_i^l = g_{c_i}^l(\mathbf{h}_i^{l-1}, \mathbf{a}_i^l), \tag{7}$$

where $g_{c_i}^l$ is the update function for node type c_i at l-th step, which may be a non-linear layer in practice, i.e.

$$\mathbf{h}_i^l = \sigma(\mathbf{W}_{c_i}^l \mathbf{h}_i^{l-1} + \mathbf{a}_i^l), \tag{8}$$

where σ is an activation function, e.g. RELU, and $\mathbf{W}_{c_i}^l$ is the transition matrix.

3.3.3 Output Module

After updating state L steps, based on the last state $\mathbf{h}_i^L$ each node v_i will compute the Q-values $\mathbf{q}_i$ for the actions corresponding to v_i:

$$\mathbf{q}_i = o_i(\mathbf{h}_i^L), \tag{9}$$

where o_i is a function for node i, which may be a MLP in practice. Note that, in input module and communication module, the same types of nodes share parameters, which may speed up the learning

process. However, in output module, in oder to capture the *specific characteristics* of each node, they don't share parameters.

When making decision, the outputs of all nodes are first concatenated, i.e. $\mathbf{q} = \mathbf{q}_0 \oplus \mathbf{q}_1 \oplus \cdots \oplus \mathbf{q}_n$, then the action is chosen according to $\mathbf{q}$ as done in vanilla DQN.

3.4 Inference of Graph Structure

In the previous section, we assume that the structure of graph G, i.e. the adjacency matrix $\mathbf{Z}$, is known. However, usually the graph is not known in practice, and the hypothetical structure is not guaranteed to be optimal. Therefore, it's better to optimize the structure along with the parameters of GNN. We assume that $\mathbf{Z}$ is a *latent* variable, and follows a factorized Bernoulli distribution, i.e. each element $z_{ij} \sim \mathrm{Bern}(\phi_{ij})$. The exact posterior distribution of $\mathbf{Z}$ is hard to inference and we can obtain the approximate posterior $q_\phi(\mathbf{Z})$ via variational inference. The loss function in equation (1) will be reformulated as follows:

$$\begin{aligned}\mathcal{L}(\theta, \phi) =& \mathbb{E}_{\tau \sim \mathrm{U}(\mathcal{D}), \mathbf{Z} \sim q_\phi(\mathbf{Z})} \left[\left(y - Q_\theta(\mathbf{b}, a; \mathbf{Z}) \right)^2 \right] + \lambda \mathrm{KL}(q_\phi(\mathbf{Z}) || p(\mathbf{Z})) \\ =& \mathcal{L}_E(\theta, \phi) + \lambda \mathcal{L}_C(\phi),\end{aligned} \tag{10}$$

where $p(\mathbf{Z})$ is the prior, and also follows a factorized Bernoulli distribution.

As shown in the equation, the loss function $\mathcal{L}(\theta, \phi)$ consists of two terms $\mathcal{L}_E(\theta, \phi)$ and $\mathcal{L}_C(\phi)$. $\mathcal{L}_E(\theta, \phi)$ corresponds to the error loss that measures how well the model is fitting the current dataset, whereas $\mathcal{L}_C(\phi)$ refers to the complexity loss that measures the flexibility of the model. For a uniform Bernoulli prior distribution, $\mathcal{L}_C(\phi)$ can be written following:

$$\mathcal{L}_C(\phi) = \sum_{i,j} \mathrm{KL}(q_{\phi_{ij}}(z_{ij}) || p(z_{ij}))) = \sum_{i,j} \left[-\mathrm{log}0.5 + \phi_{ij} \mathrm{log} \phi_{ij} + (1 - \phi_{ij}) \mathrm{log}(1 - \phi_{ij}) \right]. \tag{11}$$

Minimising the KL divergence between $q_{\phi_{ij}}(z_{ij})$ and the prior is equivalent to maximising the entropy of the Bernoulli random variable with probability ϕ_{ij}. This pushes the probability towards 0.5.

While $\mathcal{L}_C(\phi)$ is straightforward to minimize, $\mathcal{L}_E(\theta, \phi)$ does not allow for efficient gradient based optimization due to the discrete nature of $\mathbf{Z}$. To obtain the Q-values $Q_\theta(\mathbf{b}, a; \mathbf{Z})$, we need first to sample the discrete graph structure $\mathbf{Z}$ according to the factorized Bernoulli distribution with parameters ϕ. Therefore, the loss function $\mathcal{L}_E(\theta, \phi)$ is non-differential w.r.t. ϕ. While in principle a score function estimator such as the REINFORCE (Williams, 1992) could be employed, it suffers from high variance in practice. An alternative is to replace the discrete Bernoulli distribution with its continuous relaxation, i.e. the Concrete distribution as done at Concrete dropout (Gal et al., 2017). Instead of sampling the random variable from the discrete Bernoulli distribution we sample realisations from the following Concrete distribution with some temperature t:

$$\tilde{z}_{ij} = \mathrm{sigmoid} \left(\frac{1}{t} \left(\mathrm{log} \phi_{ij} - \mathrm{log}(1 - \phi_{ij}) + \mathrm{log} \epsilon_{ij} - \mathrm{log}(1 - \epsilon_{ij}) \right) \right), \tag{12}$$

where ϵ_{ij} is the noise sampled from a uniform distribution, i.e. $\epsilon_{ij} \sim \mathrm{U}(0, 1)$. The Concrete distribution concentrates most mass on the boundaries of the interval 0 and 1. With the continue relaxation $\tilde{z}_{ij}$ of the Bernoulli random variable z_{ij}, the loss function will be reparameterized as follows:

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{\tau \sim \mathrm{U}(\mathcal{D}), \epsilon \sim \mathrm{U}(0,1)} \left[\left(y - Q_\theta(\mathbf{b}, a; \tilde{\mathbf{Z}}) \right)^2 \right] + \lambda KL(q_\phi(\mathbf{Z}) || p(\mathbf{Z})), \tag{13}$$

where $\epsilon \sim \mathrm{U}(0, 1)$ denotes that every ϵ_{ij} is sampled from the uniform distribution independently. Every element z_{ij} of $\tilde{\mathbf{Z}}$ is calculated with equation (12). Note that whereas the sampling still exists, the sampled noise ϵ is independent of the parameters ϕ. Therefore the loss function $\mathcal{L}(\theta, \phi)$ is differential w.r.t. ϕ and it can be straightforward to minimize.

Note that the procedure of sampling according to equation (12) is approximately equivalent to sampling the neighbors of every node v_i according to the probability ϕ_{ij}. It is a variant of dropout and adopted in self-attention models (Tan et al., 2018) and graph convolutional networks (Kipf and Welling, 2017).

task	Env.1	Env.2	Env.3	Env.4	Env.5	Env.6
SER	0%	0%	15%	15%	15%	30%
Masks	On	Off	On	Off	On	On
User	Standard	Standard	Standard	Standard	Unfriendly	Standard

Table 1: The set of benchmarketing tasks

4 Experiments

4.1 PyDial Benchmark

Dialogue management research is typically evaluated on a small set of environments. Fortunately, a set of extensive simulated dialogue management environments is published in (Casanueva et al., 2017), which can test the capability of models in different environments. These environments are implemented in an open domain toolkit: PyDial. With providing domain-independent implementations of all the dialogue system modules, simulated users and simulated error models, PyDial has the potential to create a set of benchmark environments to compare different models in the same conditions. In this paper, we evaluate our proposed approaches on these environments, which will be briefly introduced next and are summarized in Table 1.

Firstly, there are three different **domains**: information seeking tasks for restaurants in Cambridge (CR) and San Francisco (SFR) and a generic shopping task for laptops (LAP). They are slot-based, which means the dialogue state is factorized into slots.

The second different dimension of variability is the **semantic error rate (SER)**, which simulates different noise levels in the speech understanding input channel.

In addition, env.5's **user model** is defined to be an *Unfriendly* distribution, where the users barely provide any extra information to the system, while others' are all *Standard*.

Finally, in order to test the learning capability of the models, the action **masking mechanism** is disabled in two of the tasks: env2 and env4.

The metrics in the next section has presented the average success rate and average reward for each applied model. The success rate is defined as the percentage of dialogues which are completed successfully.The reward is defined as $20 \times \mathbb{1}(\mathcal{D}) - T$, here $\mathbb{1}(\mathcal{D})$ is the success indicator and T is the dialogue length in turns.

4.2 Results

We evaluate four variants of our proposed structured DRL-based dialogue policy:

- **GNN-M**: GNN-based dialogue policy with *fully* connected graph. The communication method between nodes is *Mean-Comm* described in section 3.3.2.
- **GNN-M-C**: It is similar to GNN-M except that the graph structure is jointly optimized with the parameters of GNN as described in section 3.4. The hyperparameter λ in equation (13) is 4×10^{-4}.
- **GNN-A**: GNN-based dialogue policy with *fully* connected graph. The communication method between nodes is *Attention-Comm* described in section 3.3.2.
- **GNN-A-C**: It is similar to GNN-A except that the graph structure is jointly optimized with the parameters of GNN as described in section 3.4. The hyperparameter λ in equation (13) is 4×10^{-4}.

These models are compared with three baselines[1] presented in (Casanueva et al., 2017): GP-Sarsa, DQN, and eNAC. The results in the 18 tasks after 1000/4000 training dialogues are shown in Table 2. For each task, the result is the mean over 10 different random seeds.

[1]Due to the limitation of space, the results of A2C is omitted. The performance of A2C is worst among these baselines.

		Baselines						Structured DRL							
		GP-Sarsa		DQN		eNAC		GNN-M		GNN-A		GNN-M-C		GNN-A-C	
	Task	Suc.	Rew.	Suc.	Rew.	Suc.	Rew.	Suc.	Rew.	Suc.	Rew.	Suc.	Rew.	Suc.	Rew.
after 1000 training dialogues															
Env.1	CR	98.0	13.0	88.6	11.6	93.0	12.2	95.2	13.0	97.7	13.5	96.3	13.0	***98.8***	**13.7**
	SFR	91.9	10.0	48.0	2.7	85.8	9.9	70.1	8.6	66.6	6.3	**89.0**	**10.2**	85.8	9.6
	LAP	78.9	6.7	61.9	5.5	84.2	8.8	71.0	7.2	79.0	8.7	**92.3**	**10.8**	87.0	9.8
Env.2	CR	91.1	10.8	67.6	6.4	78.8	6.6	94.0	12.4	94.7	12.3	**95.8**	**12.6**	94.5	12.0
	SFR	82.1	7.4	64.2	5.4	67.5	2.7	**85.4**	**10.9**	76.2	7.5	83.8	8.4	77.8	6.7
	LAP	68.4	3.1	70.8	5.8	57.8	-0.5	78.2	8.0	80.7	**8.5**	83.4	6.6	**83.4**	7.1
Env.3	CR	91.9	10.4	79.5	9.2	85.7	10.0	88.4	10.8	**97.1**	**12.5**	95.9	12.2	95.8	12.2
	SFR	76.6	5.5	42.4	1.0	73.6	6.2	78.6	7.5	80.4	**7.8**	80.9	7.1	**81.5**	7.0
	LAP	65.0	2.8	51.9	3.1	71.0	5.5	68.2	5.5	71.1	6.1	**80.0**	**7.1**	79.2	6.2
Env.4	CR	88.2	9.3	73.5	6.9	73.6	4.4	**91.3**	**10.8**	88.9	10.2	91.2	10.4	86.6	9.3
	SFR	73.6	4.9	65.9	4.5	60.4	0.8	**81.3**	**8.2**	79.0	7.4	71.5	3.9	80.9	7.1
	LAP	61.3	0.3	53.2	2.7	46.9	-2.9	66.1	**4.5**	65.8	4.1	65.8	2.1	**67.6**	1.7
Env.5	CR	90.2	9.0	60.1	4.1	81.2	8.1	94.8	**11.2**	95.1	11.1	92.9	10.3	**95.2**	11.0
	SFR	65.3	1.3	32.5	-2.0	54.0	0.9	53.8	1.3	74.3	**5.0**	74.5	3.6	**77.6**	4.3
	LAP	44.9	-2.8	31.4	-1.8	61.3	1.7	51.4	1.3	67.1	2.8	70.7	2.7	**76.6**	**3.1**
Env.6	CR	84.9	8.3	72.3	6.9	73.6	6.7	87.7	9.9	**88.8**	**10.1**	86.9	9.0	87.3	9.2
	SFR	59.7	0.7	35.6	-1.2	55.2	1.4	51.2	1.4	65.6	**3.6**	67.2	2.8	**71.1**	3.1
	LAP	52.0	-1.5	47.5	1.4	56.3	1.9	57.0	2.7	60.1	2.8	**71.4**	**3.3**	65.4	1.6
Mean	CR	90.7	10.1	73.6	7.5	81.0	8.0	91.9	11.4	**93.7**	**11.6**	93.2	11.3	93.0	11.2
	SFR	74.9	5.0	48.1	1.7	66.1	3.6	70.1	6.3	73.7	6.3	77.8	6.0	**79.1**	**6.3**
	LAP	61.7	1.4	52.8	2.8	62.9	2.4	65.3	4.9	70.7	**5.5**	**77.3**	5.4	76.5	4.9
	ALL	75.8	5.5	58.2	4.0	70.0	4.7	75.8	7.5	79.4	**7.8**	82.8	7.6	**82.9**	7.5
after 4000 training dialogues															
Env.1	CR	**99.4**	13.5	93.9	12.7	94.8	12.4	96.1	13.3	99.2	**14.0**	98.6	13.7	99.1	13.9
	SFR	**96.1**	11.4	65.0	5.9	94.0	**11.7**	86.8	10.2	88.7	10.7	93.5	10.8	94.5	11.4
	LAP	89.1	9.4	70.1	6.9	91.4	10.5	85.4	9.9	86.6	9.8	92.2	10.6	**93.7**	**10.8**
Env.2	CR	96.8	12.2	91.9	12.0	83.6	9.0	**98.5**	**13.6**	96.4	12.7	96.6	12.8	90.5	11.3
	SFR	91.9	9.6	84.3	9.2	65.6	3.7	**93.6**	**11.6**	87.4	9.5	91.8	10.3	89.2	10.3
	LAP	82.3	7.3	74.5	6.6	55.1	1.5	83.3	8.8	**92.2**	**11.3**	89.1	9.0	83.9	7.5
Env.3	CR	95.1	11.0	93.4	11.9	90.8	11.2	**96.4**	**12.6**	95.3	12.1	95.6	12.1	96.3	12.3
	SFR	81.6	6.9	60.9	4.0	84.6	8.6	84.1	8.3	84.2	**8.5**	**88.2**	8.2	83.0	7.1
	LAP	68.3	4.5	61.1	4.3	76.6	6.7	77.7	6.9	78.5	**7.0**	**85.6**	6.8	79.6	5.6
Env.4	CR	91.5	9.9	90.0	10.7	85.3	9.0	93.5	**11.5**	**93.9**	11.4	93.4	11.0	92.0	10.9
	SFR	81.6	7.2	77.8	7.7	61.7	2.0	77.7	7.2	79.8	7.5	80.9	7.2	**85.8**	**8.4**
	LAP	72.7	5.3	68.7	5.5	52.8	-0.8	**85.3**	**8.8**	77.5	7.7	72.9	3.5	75.6	4.0
Env.5	CR	93.8	9.8	90.7	10.3	91.6	10.5	**95.1**	**11.2**	94.7	10.9	95.0	10.8	93.4	10.2
	SFR	74.7	3.6	62.8	2.9	74.4	4.5	**88.8**	**7.9**	81.6	6.3	83.9	5.4	79.9	4.3
	LAP	39.5	-1.6	45.5	0.0	75.8	4.1	73.0	**4.2**	64.7	1.7	75.2	1.5	**76.9**	2.6
Env.6	CR	89.6	8.8	87.8	10.0	79.6	8.0	89.0	**10.0**	89.3	9.8	**90.4**	9.9	89.4	9.8
	SFR	64.2	2.7	47.2	0.4	66.7	3.9	72.8	**4.9**	69.3	4.0	67.3	1.7	**74.4**	3.7
	LAP	44.9	-0.2	46.1	1.0	64.6	3.6	64.1	3.6	69.4	**4.1**	**72.0**	1.8	69.8	2.0
Mean	CR	94.4	10.9	91.3	11.3	87.6	10.0	94.8	**12.0**	94.8	11.8	**94.9**	11.7	93.5	11.4
	SFR	81.7	6.9	66.3	5.0	74.5	5.7	84.0	**8.4**	81.8	7.8	84.3	7.3	**84.5**	7.5
	LAP	66.1	4.1	61.0	4.1	69.4	4.3	78.1	**7.0**	78.2	6.9	**81.2**	5.5	79.9	5.4
	ALL	80.7	7.3	72.9	6.8	77.2	6.7	85.6	**9.1**	84.9	8.8	**86.8**	8.2	86.0	8.1

Table 2: Reward and success rates after 1000/4000 training dialogues. The results in bold blue are the best success rates, and the results in bold black are the best rewards.

We can find that our models can achieve the best performance in almost all of the tasks. In the two larger domains (SFR, LAP), the performance boost is more clear. For our four models, there is no significant difference in their performance, which indicates our proposed framework is not sensitive to these variants. However, through further analysis of these results, some interesting phenomena are observed.

After 1000 training dialogues, on average, two models (GNN-A and GNN-A-C) with *Attention-Comm* achieve the best reward and the best success rate respectively. It shows that *Attention-Comm* may benefit the sample-efficiency in the early learning stage. The potential reason is that models with *Attention-Comm* can quickly adjust their parameters to pay more attention on most important messages.

It is interesting to observe that two models with optimization of graph structure obtain better success rate than the other two (GNN-M-C vs. GNN-M, GNN-A-C vs. GNN-A). As described in section 3.4,

our proposed approach for structure optimization is a variant of dropout, i.e dropout neighbors. Dropout introduces some noise in the neural networks and can improve exploration in DRL (Gal and Ghahramani, 2016; Gal et al., 2017). Here, GNN-A-C and GNN-M-C also do more exploration, resulting in long dialogues which have more potential for success.

4.3 Analysis and Discussion

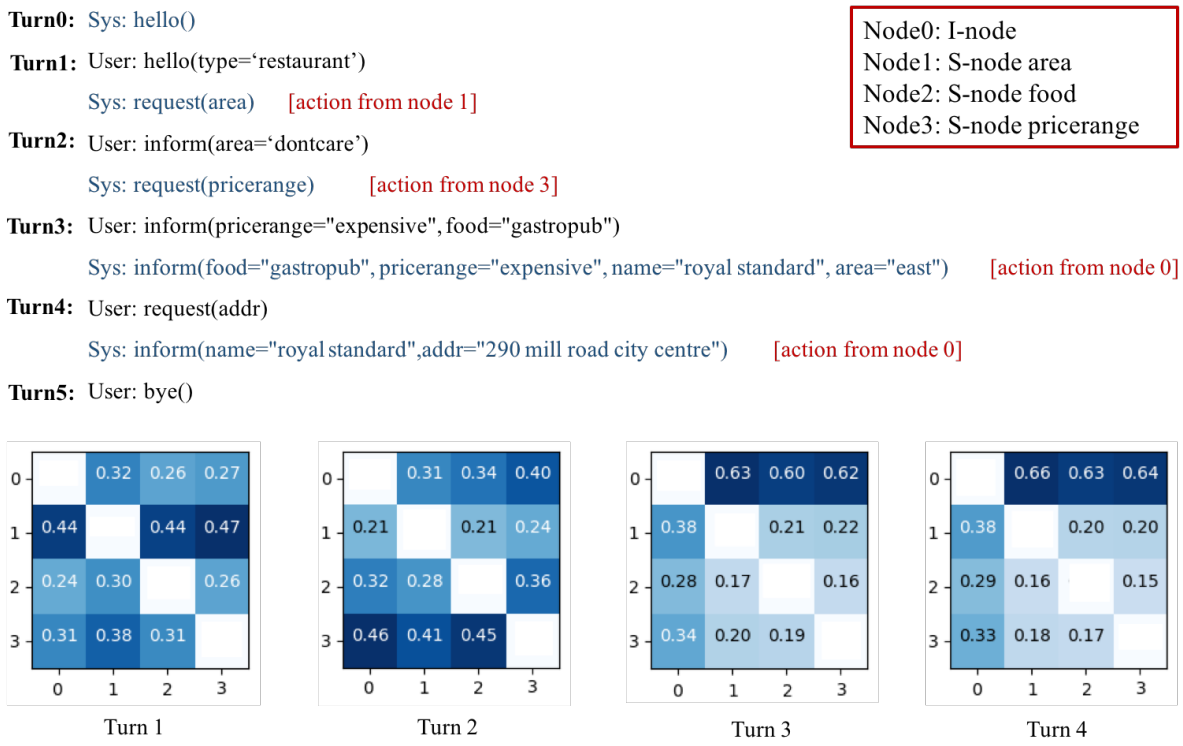


Figure 3: An example of dialogue on semantic level with a policy of GNN-A in CR domain (Env.1).

Fig.3 is an example of dialogue with a policy of GNN-A in CR domain (Env.1). The confusion matrices show the attention weights in the first communication step computed by equation (6). The elements in the i-th row represent the weights node v_i sending messages to other nodes. And each column is normalized by *softmax*. We can find that if the final action comes from the action sets $\mathbb{A}_i$, which is corresponding to node v_i, then the weights in the i-th row will be larger, as if to say:*" Hey, it's my duty to making decision at this turn, and you should make the Q-values for your action sets smaller!"*.

The confusion matrix in Fig.4 in Appendix A is the optimized graph structure in LAP domain (Env.3), i.e. the parameter ϕ_{ij} represents the probability of the existence of edge from node v_i to node v_j. We can find that all probabilities are in the range [0.7,0.9]. However, the probabilities associated with I-node is much larger than others, which means the communication between I-node and S-node is more important than the communication between S-node and S-node.

5 Conclusion

In this paper, we propose a structured dialogue policy, which is represented by a graph neural network (GNN). It consists of some sub-networks, each one for a node in a directed graph. The graph has two types of nodes: a *slot-independent* (I-node) node and n *slot-dependent* nodes (S-nodes). Each S-node corresponds to a slot. The same types of nodes partially share parameters. In order to model the interaction between sub-networks, they have internal message exchanges between neighbors in the graph when doing forward computation. We evaluate our framework on PyDial benchmark. Our models achieve state of the art in almost all of 18 tasks.

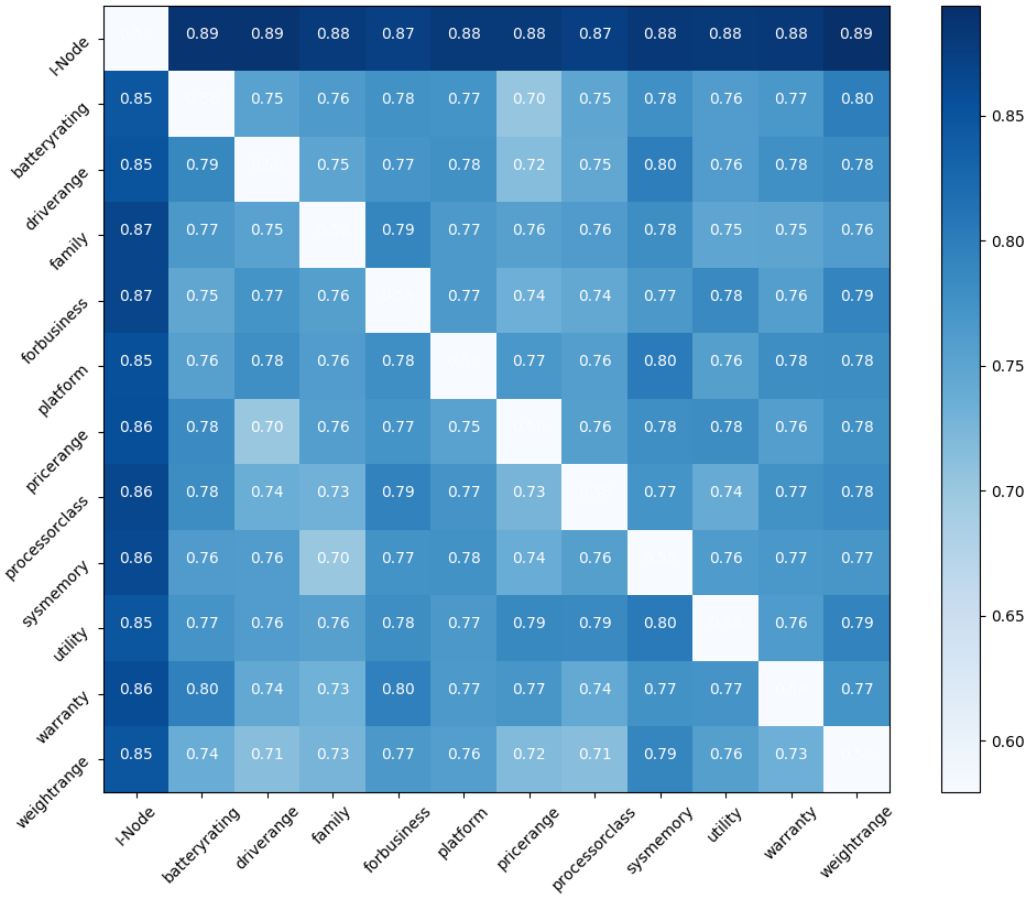


Figure 4: The confusion matrix of graph structure in LAP domain (Env.3). Each element denotes the probability that there is a edge from node v_i to v_j.

Acknowledgements

The corresponding author is Kai Yu. This work has been supported by the National Key Research and Development Program of China under Grant No.2017YFB1002102, Shanghai International Science and Technology Cooperation Fund (No. 16550720300) and the Major Program of Science and Technology Commission of Shanghai Municipality (STCSM) (No. 17JC1404104). Experiments have been carried out on the PI supercomputer at Shanghai Jiao Tong University.

References

Iñigo Casanueva, Paweł Budzianowski, Pei-Hao Su, Nikola Mrkšić, Tsung-Hsien Wen, Stefan Ultes, Lina Rojas-Barahona, Steve Young, and Milica Gašić. 2017. A benchmarking environment for reinforcement learning based task oriented dialogue management. *arXiv preprint arXiv:1711.11023*.

Cheng Chang, Runzhe Yang, Lu Chen, Xiang Zhou, and Kai Yu. 2017. Affordable on-line dialogue policy learning. In *Proceedings of EMNLP*, pages 2190–2199.

Lu Chen, Runzhe Yang, Cheng Chang, Zihao Ye, Xiang Zhou, and Kai Yu. 2017a. On-line dialogue policy learning with companion teaching. *Proceedings of EACL*, pages 198–204.

Lu Chen, Xiang Zhou, Cheng Chang, Runzhe Yang, and Kai Yu. 2017b. Agent-aware dropout dqn for safe and efficient on-line dialogue policy learning. In *Proceedings of EMNLP*, pages 2444–2454.

Lu Chen, Cheng Chang, Zhi Chen, Bowen Tan, Milica Gašić, and Kai Yu. 2018. Policy adaptation for deep reinforcement learning-based dialogue management. In *Proceedings of ICASSP*, pages 6074–6078.

Heriberto Cuayáhuitl, Simon Keizer, and Oliver Lemon. 2015. Strategic dialogue management via deep reinforcement learning. *NIPS DRL Workshop*.

Mehdi Fatemi, Layla El Asri, Hannes Schulz, Jing He, and Kaheer Suleman. 2016. Policy networks with two-stage training for dialogue systems. In *Proceedings of SIGDIAL*, pages 101–110.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of ICML*, pages 1050–1059.

Yarin Gal, Jiri Hron, and Alex Kendall. 2017. Concrete dropout. In *Proceedings of NIPS*, pages 3584–3593.

Milica Gašić, Filip Jurčíček, Simon Keizer, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. 2010. Gaussian processes for fast policy optimisation of POMDP-based dialogue managers. In *Proceedings of SIGDIAL*, pages 201–204.

Milica Gašić and Steve Young. 2014. Gaussian processes for pomdp-based dialogue manager optimization. *IEEE/ACM TASLP*, 22(1):28–40.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*, pages 655–665.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Sungjin Lee. 2013. Structured discriminative model for dialog state tracking. In *Proceedings of SIGDIAL*, pages 442–451.

Long-Ji Lin. 1993. *Reinforcement learning for robots using neural networks*. Ph.D. thesis, Fujitsu Laboratories Ltd.

Zachary C. Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. 2018. Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking. In *Proceedings of AAAI*, pages 5237–5244.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of EMNLP*, pages 2231–2240.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Pei-Hao Su, Pawel Budzianowski, Stefan Ultes, Milica Gasic, and Steve Young. 2017. Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. In *Proceedings of SIGDIAL*, pages 147–157.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proceedings of AAAI*, pages 4929–4936.

Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*, pages 6000–6010.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of ACL*, pages 665–677.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *Proceedings of SIGDIAL*, pages 1–10.

JTAV: Jointly Learning Social Media Content Representation by Fusing Textual, Acoustic, and Visual Features

Hongru Liang[1], Haozheng Wang[1], Jun Wang[2], Shaodi You[3], Zhe Sun[4], Jin-Mao Wei[1], Zhenglu Yang[1]*
[1] CCCE, Nankai University, Tianjin, China
[2] College of Mathematics and Statistics Science, Ludong University, China
[3] Data61-CSIRO, Australian National University, Canberra, Australia
[4] RIKEN Head Office for Information Systems and Cybersecurity Computational Engineering Applications Unit, Japan
{lianghr, hzwang, junwang}@mail.nankai.edu.cn, shaodi.you@data61.csiro.au, zhe.sun.vk@riken.jp, {weijm, yangzl}@nankai.edu.cn

Abstract

Learning social media content is the basis of many real-world applications, including information retrieval and recommendation systems, among others. In contrast with previous works that focus mainly on single modal or bi-modal learning, we propose to learn social media content by fusing jointly textual, acoustic, and visual information (JTAV). Effective strategies are proposed to extract fine-grained features of each modality, that is, attBiGRU and DCRNN. We also introduce cross-modal fusion and attentive pooling techniques to integrate multi-modal information comprehensively. Extensive experimental evaluation conducted on real-world datasets demonstrates our proposed model outperforms the state-of-the-art approaches by a large margin.

1 Introduction

The popularity of the social media (e.g., Twitter, Weibo) over the last two decades has led to increasing demands for learning the content of social media, which may be beneficial to many real-world applications, such as sentiment analysis (Wang et al., 2015), information retrieval (Li et al., 2017), and recommendation systems (Wu et al., 2017). However, the task is non-trivial and challenging, because social media content is commonly multi-modal and involves several types of data, including text, audio, and image (an example is illustrated in Fig. 1). Each representation of the individual modality encodes specific knowledge and complementary, which can be explored to facilitate understanding of the entire meaning of the content. Therefore, gaining comprehensive knowledge from learning arising out of social media requires deliberate exploration of single-modal information and joint learning of the intrinsic correlation among various-modalities.

Figure 1: An example of multi-modal social media content, organized by a textual passage, a clip of the song "Muhammad Ali", and a picture of Muhammad Ali's famous victory.

*Corresponding author

Proceedings of the 27th International Conference on Computational Linguistics, pages 1269–1280
Santa Fe, New Mexico, USA, August 20-26, 2018.

Previous works on learning social media content have focused mainly on single modality, e.g., textual information (Wu et al., 2017) and image information (Garimella et al., 2016). To our best knowledge, no studies have focused on mere acoustic information analysis for social media. These previous works cope with data in the early stage of social media, because users are limited to posting single-modal content. In contrast, today's social media content usually includes textual, acoustic, and visual ingredients simultaneously, and requires multi-modal learning strategies to integrate rich information from all modalities.

Many researchers have studied the bi-modal utilization to address the issue of incomplete information exploration. This utilization can be classified into two kinds of approaches. The first is bi-modal retrieval, which utilizes one modality to find similar content in the other modality, such as finding the most relevant texts to a given image (Li et al., 2017). This kind of approach is similar to machine translation tasks to some extent, for instance, the input is fine-grained features of one modality, the objective of which is to minimize the distance between the output vectors with fine-grained features of the target modality. The second is bi-modal unification (Park and Im, 2016), which seeks to integrate individual information of the two modalities together. The process of this kind of approach can be divided into two stages: (a) feature extraction, in which the two modalities are encoded into their corresponding vectors or matrices via embedding approaches; and (b) bi-modal feature aggregation, in which the extracted features are mapped into a shared latent space through multilayer perceptrons or directly into the target spaces of tasks through classifiers (Chen et al., 2017).

Learning representations of multi-modal data, which contains features of more than two modalities, is a more challenging task. The main reason is that the fine-grained features of the modalities can be difficult to obtain because of the lack of annotated labels for every modality. Previous studies lack the appropriate strategies to aggregate various kinds of information effectively. It's easy to imagine that bi-modal aggregation models might be extended to multi-modal learning with linear operations, e.g., the concatenation operation. However, this technique may generate inappropriate integral representation of the multiple features and lacks further exploration in the previous work.

We address the abovementioned issues by introducing effective strategies to represent the individual features for the three modalities. A general unified framework is proposed to integrate multi-modalities seamlessly. Specifically, we begin by encoding single modal parts into dense vectors. For textual content, we design an attention-based network (i.e., attBiGRU), to incorporate various textual information. For acoustic content, we introduce an effective DRCNN approach to embed temporal audio clips locally and globally. For visual content, we fine-tune a state-of-the-art general framework, DenseNet (Huang et al., 2017). We propose a novel fusion framework, which involves the cross-modal fusion and the attentive pooling strategies, to aggregate various modal information. Extensive experimental evaluations conducted on real-world datasets demonstrate that our proposed model outperforms state-of-the-art approaches by a large margin. The contributions of this paper are presented as follows:

- We introduce effective strategies to extract representative features of the three modalities, including the following. (1) For textual information, we design an attention based network, named as attBiGRU, to integrate various textual parts. The independent textual parts are considered as two separate roles, namely, the protagonist and the supporting players. The supporting players are encoded into an attention weight vector on the protagonist. (2) For acoustic information, we propose the DCRNN strategy based on the densely connected convolutional and recurrent networks. The densely connected convolutional networks are used to learn the acoustic features both locally and globally. The recurrent networks are designed to learn the temporal information. (3) For visual information, we introduce state-of-the-art strategies from (Simonyan and Zisserman, 2015; He et al., 2016; Huang et al., 2017).

- We propose a general multi-modal feature aggregation framework, JTAV, to learn information jointly. The framework considers the textual, acoustic, and visual contents to generate a unified representation of social media content with the help of the optimized feature fusion network, CMF-AP, which can learn inner and outer cross-modal information simultaneously.

- We conduct comprehensive experiments on two kinds of tasks, sentiment analysis and information

retrieval, to evaluate the effectiveness of the proposed JTAV framework. The experiments demonstrate that JTAV outperforms state-of-the-art approaches remarkably, that is, about 2% higher than the state-of-the-art approaches on all metrics in the sentiment analysis experiment and more than ten times better than the baseline on main metrics in music information retrieval experiment.

2 Related Work

Over the past decades, social media learning has gained considerable attention in related research literature. Social media content is utilized as raw material for recommendation (Wang et al., 2013), information retrieval (Agichtein et al., 2008), and so forth.

The dominance of the single-modal content in social media in the early stages, however, has caused most previous works to position themselves in single-modal exploration, e.g., natural language processing and image processing. (Han et al., 2013) focused on the short text messages in social media, and normalized lexical variants to their canonical forms. In (Hutto and Gilbert, 2014), combined lexical features of social media text were proposed for expressing and emphasizing sentiment intensity. An example of image processing was (Garimella et al., 2016), social media images, particularly geo-tagged images, were studied in predicting county-level health statistics.

In recent years, users have come to prefer presenting more attractive ingredients, such as image and audio in addition to natural words. Therefore, literature on social media analysis has been shifting its focus from single-modal to multi-modal learning. For example, (Wang et al., 2015) assume that visual and textual content have a shared sentiment label space, and the optimization problem can be converted to the minimization between the derivatives of both modalities. In (Li et al., 2017), the images and lyrics are mapped into a shared tag space. The lyrics, which have the smallest mean squared error with a given image, emerged as the matching object. (Chen et al., 2017) present a new end-to-end framework for visual and textual sentiment analysis; they began with the co-appearing image and text pairs that provide semantic information for each other, and use the concatenation of the vectors generated from the original image and text, in the shared sentiment space. However, the drawback of these works is that they utilized only partial information.

An intuitive idea to gain a comprehensive learning of the entire meaning of social media is to integrate more modalities effectively. The reason is that each representation of the individual modality encodes specific knowledge and is complementary, an aspect that can be explored to facilitate understanding on the entire meaning of the content. However, this task could be extremely challenging because we need to explore single-modal information deliberately and jointly learn the intrinsic correlation among various modalities. This work is the first study that focuses on the seamless integration of the three modalities (i.e., text, audio, and image), into a general framework to jointly learn the social media content.

3 Model Description

The architecture of our proposed framework is illustrated in Fig. 2. It has two main modules, the feature extraction and the feature aggregation. Let $\boldsymbol{t}$, $\boldsymbol{a}$, and $\boldsymbol{v}$ denote the textual, acoustic, and visual features, which are encoded from the feature extraction module, respectively. Let $\boldsymbol{u}$ denote the target unified representation of the multi-modal content. The feature aggregation module is designed to generate $\boldsymbol{u}$ from $\boldsymbol{t}$, $\boldsymbol{a}$, and $\boldsymbol{v}$ effectively and comprehensively. The example in Fig. 1 is utilized to ground our discussion and to illustrate the intuitive idea of this work.

3.1 Feature Extraction

Fine-grained features, which are related to the qualities of the input of the aggregation processing directly, are the foundation of multi-modal representation learning. The proposed strategies for generating appropriate features for text, audio, and image will be presented in the following subsections.

3.1.1 Text modeling

The purpose of text modeling is to encode raw textual parts into a latent representation $\boldsymbol{t}$. We design a bidirectional gated recurrent neural network (BiGRU) based attentive network (i.e., attBiGRU), to extract sufficient textual features carrying the long-range dependencies, as illustrated in Fig. 3.

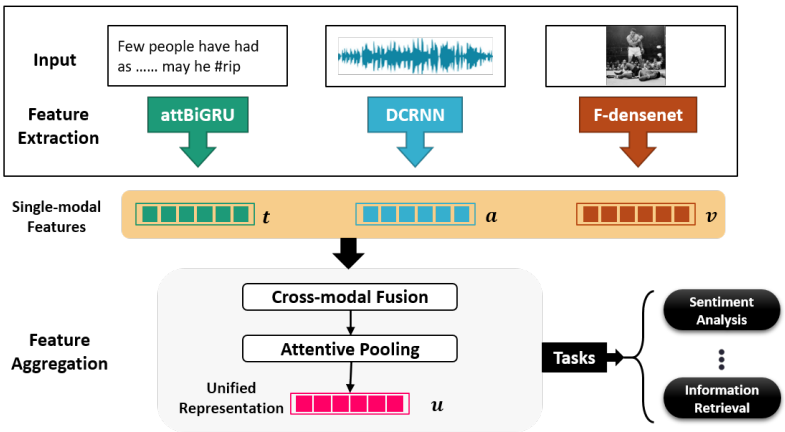


Figure 2: The architecture of the proposed JTAV framework

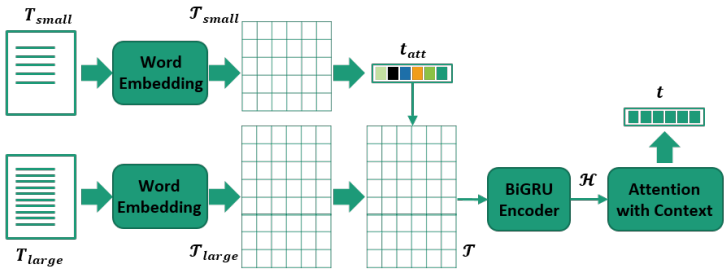


Figure 3: The attBiGRU strategy for modeling textual content

Generally, social media content contains more than one textual part, such as the lyrics and reviews of a song. It is reasonable to assume that a protagonist and supplying players exist among these textual parts. Intuitively and for sake of simplicity, we can regard the part with the maximum number of words as the protagonist and the other parts as the supplying players. Let $\boldsymbol{T_{large}}$ denote the protagonist and $\boldsymbol{T_{small}}$ denote the supplying roles. Inspired by (Li et al., 2017) to reduce the gap between image and text, we use a pre-trained word embedding model, e.g., FastText (Bojanowski et al., 2017), to encode $\boldsymbol{T_{small}}$ into a word matrix (i.e., $\boldsymbol{\mathcal{T}_{small}}$). $\boldsymbol{\mathcal{T}_{small}}$ is computed as $\boldsymbol{w_{small}} W_e$, where $\boldsymbol{w_{small}}$ is the words in $\boldsymbol{T_{small}}$ and W_e is a pre-trained embedding matrix. $\boldsymbol{t_{att}}$ is an attention vector generated from $\boldsymbol{\mathcal{T}_{small}}$ with an average pooling layer, and works on every word vector of $\boldsymbol{\mathcal{T}_{large}}$ through the element-wise product. We obtain an attented word matrix, denoted as $\boldsymbol{\mathcal{T}}$.

The BiGRU encoder is utilized to encode every word vector in the sequence into a latent representation and receive the aid of the other textual parts. Subsequently, we feed the attended word matrix $\boldsymbol{\mathcal{T}}$ into a BiGRU network. The output of the BiGRU encoder is a $2M \times N$ matrix, where M is the number of hidden units of a layer, and N is the number of words in $\boldsymbol{\mathcal{T}_{large}}$. We denote the output matrix as $\boldsymbol{\mathcal{H}}$.

The attention with context part is a simplified version of (Yang et al., 2016) without the sentence attention part. It is introduced to balance the importance among all words and is defined as follows:

$$
\begin{aligned}
\hat{\boldsymbol{h}}_{\boldsymbol{i}} &= tanh(\boldsymbol{W_u h_i} + \boldsymbol{b_u}), \\
\boldsymbol{\alpha_i} &= \frac{exp\left(\hat{\boldsymbol{h}}_{\boldsymbol{i}}{}^{\top} \hat{\boldsymbol{h}}_{\boldsymbol{c}}\right)}{\sum_{j=0}^{N-1} exp\left(\hat{\boldsymbol{h}}_{\boldsymbol{j}}{}^{\top} \hat{\boldsymbol{h}}_{\boldsymbol{c}}\right)}, \\
\boldsymbol{t} &= \sum_{j=0}^{N-1} \boldsymbol{\alpha_j h_j},
\end{aligned}
\tag{1}
$$

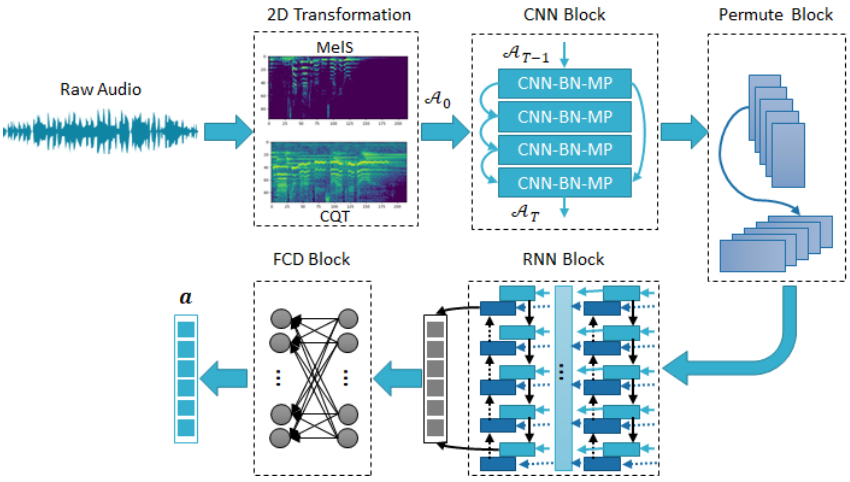


Figure 4: The DRCNN strategy for modeling acoustic content

where $\boldsymbol{h_i}(0 \leq i \leq N-1)$ is the i^{th} column of $\boldsymbol{\mathcal{H}}$, $\hat{\boldsymbol{h}}_{\boldsymbol{i}}$ is a latent vector computed from $\boldsymbol{h}_i$ using the tanh function, $\boldsymbol{\alpha_i}$ is the attention weight scalar computed from $\hat{\boldsymbol{h}}_{\boldsymbol{i}}$ and a random initialized context vector $\hat{\boldsymbol{h}}_{\boldsymbol{c}}$, $\boldsymbol{W_u}$ and $\boldsymbol{b_u}$ are parameters learned from training. The final textual representation $\boldsymbol{t}$ is a weighted sum of $\boldsymbol{h_j}(0 \leq j \leq N-1)$ on $\boldsymbol{\alpha_j}(0 \leq j \leq N-1)$.

3.1.2 Audio modeling

In this paper, we focus on polyphonic audio, such as music clips. To capture useful acoustic features, we design a densely connected convolutional recurrent neural network called DCRNN, which takes raw audio as input, and generates a distributed representation $\boldsymbol{a}$ to effectively represent the acoustic information. DCRNN consists of five phases, as illustrated in Fig. 4.

The first phase is named as 2D transformation, which transfers raw audio into two-dimensional vectors. Following the majorities of the deep learning approaches in music information retrieval (Choi et al., 2017b), we use two dimensional spectrograms instead of the discrete audio signals. We choose two kinds of spectrograms (i.e., mel-spectrogram (Boashash, 1996) and constant-Q transform (Brown and Puckette, 1992)). A mel-spectrogram (MelS) is a visual representation of the spectrum of frequencies of audio and is optimized for human auditory perception. A constant-Q transform (CQT) is computed with the logarithmic-scale of the central frequencies. The CQT is well suited with the frequency distribution of music pitch. MelS and CQT have been used frequently in various acoustic tasks (Zhu et al., 2017; Jansen et al., 2017), and are utilized as coarse-grained features in the current paper. We use $\boldsymbol{\mathcal{A}_0}$ to indicate the 2D spectrograms.

The second phase is a convolutional neural network (CNN) based block (called CNN block). $\boldsymbol{\mathcal{A}_0}$ is used as the input of the block. According to (Choi et al., 2017a), early layers of CNN have the ability to capture local information such as pitch and harmony, whereas deeper layers can capture global information such as melody. We introduce the densely connected CNN, which has been proved powerful at learning both local and global features of images (Huang et al., 2017). The CNN block is composed of several densely connected "CNN-BN-MP" sub-blocks, where "BN" represents batch normalization layers, and "MP" indicates max pooling layers. Suppose there are N densely connected "CNN-BN-MP" sub-blocks in the CNN block, each can be defined as:

$$
\begin{aligned}
\boldsymbol{\mathcal{A}_h^1} &= CNN - BN - MP(\boldsymbol{\mathcal{A}_{T-1}}), \\
\boldsymbol{\mathcal{A}_h^2} &= CNN - BN - MP(\boldsymbol{\mathcal{A}_h^1}), \\
\boldsymbol{\mathcal{A}_h^3} &= CNN - BN - MP(\boldsymbol{\mathcal{A}_h^2}), \\
\boldsymbol{\mathcal{A}_T} &= CNN - BN - MP(\boldsymbol{\mathcal{A}_h^1} \oplus \boldsymbol{\mathcal{A}_h^3}),
\end{aligned}
\tag{2}
$$

where $\boldsymbol{T} \in (1, N)$, $\boldsymbol{\mathcal{A}_{T-1}}$ represents the output of the $(\boldsymbol{T}-1)^{th}$ densely connected "CNN-BN-MP" sub-block, $\boldsymbol{\mathcal{A}_h^*}$ indicate the latent variables learned by the CNN block, $\oplus$ denotes the concatenation operation, and $\boldsymbol{\mathcal{A}_T}$ is the output of the $\boldsymbol{T}^{th}$ sub-block.

The permute block is adopted from the technique introduced in (Panwar et al., 2017). The output of the CNN block is permuted to time major form, and fed into the following recurrent neural network.

The recurrent neural network (RNN) block is built based on bidirectional RNN. The stacked RNN layers are used to learn the long-short term temporal context information. We take the concatenation of the last hidden state (in vector format) in the forward direction and the first hidden state in the backward direction as the output of this phase.

Several fully connected dense layers constitutes the fully connected dense (i.e., FCD) block. We use the output of the last dense layer as the acoustic representation $\boldsymbol{a}$. Note that all the parameters presented in this section, e.g., number of layers, are optimally tuned and will be clarified in the experiment section.

3.1.3 Image modeling

Extracting rewarding features from images is a well explored problem; hence, following the tradition of other image-related tasks (Li et al., 2017; Chen et al., 2017), we use pre-trained approaches on external visual tasks.

VGG (Simonyan and Zisserman, 2015) is a deep convolutional network with nineteen weight layers, ResNet (He et al., 2016) is a residual learning framework with 101 weight layers, and DenseNet is a deep densely connected convolutional network with 121 weight layer. All these models are pre-trained on the ImageNet dataset [1], and are fine-tuned on the experimental datasets by modifying the final densely-connected layers. The last dense layer is used as the visual representation $\boldsymbol{v}$.

3.2 Feature Aggregation

In this section, we propose a feature aggregation network, which consists of the cross-modal fusion (CMF) and the attentive pooling (AP) strategies to generate a unified representation of multi-modal information, as shown in the bottom part of Fig. 2. We call the feature aggregation approach as CMF-AP.

The first phase is cross-modal fusion. We introduce the concepts of the inner and outer information. The inner information refers to information remaining in single modality, like the visual information of an image or the acoustic information of a song clip. The outer mutual information cross modalities refers to the relevant information between two modalities. In addition to the inner information generated from the feature extraction module, we compute the matrix product ($\times$) pairwise on all modal vectors to calculate the cross-modal aware matrices. The cross-modal aware matrices contain outer mutual information cross modalities and the inner information.

However, handling the cross-modal aware matrices directly is not feasible, and thus, we reconstruct $\boldsymbol{t}$, $\boldsymbol{a}$, and $\boldsymbol{v}$ from the cross-modal aware matrices. The attentive pooling strategy is adapted from Eq. 1 to conduct row pooling and column pooling.

After the above procedure, two reconstructed $\boldsymbol{t}$, two reconstructed $\boldsymbol{a}$, and two reconstructed $\boldsymbol{v}$ are obtained. For instance, from the text-image matrix, we can obtain a text-enhanced visual vector and an image-enhanced textual vector. The reconstructed vectors carry both inner and outer information. Densely connected layers are deployed for dimensionality reduction purpose. The final unified representation $\boldsymbol{u}$ is the concatenation of the reconstructed vectors.

The feature aggregation module is formulated as follows:

[1] http://www.image-net.org.

$$
\begin{aligned}
&\boldsymbol{C_{mn}} = \boldsymbol{m} \times \boldsymbol{n}^{\top};\\
&\begin{cases}
\boldsymbol{C_m} = tanh(\boldsymbol{W_m C_{mn}} + \boldsymbol{b_m}),\\
\boldsymbol{\alpha_m} = softmax(\boldsymbol{C_m^{\top}} \boldsymbol{u_m}),\\
\hat{\boldsymbol{m}} = \sigma(\hat{\boldsymbol{W}}_{\boldsymbol{m}} \boldsymbol{C_{mn}} \boldsymbol{\alpha_m} + \hat{\boldsymbol{b}}_{\boldsymbol{m}});
\end{cases}\\
&\begin{cases}
\boldsymbol{C_n} = tanh(\boldsymbol{W_n C_{mn}^{\top}} + \boldsymbol{b_n}),\\
\boldsymbol{\alpha_n} = softmax(\boldsymbol{C_n^{\top}} \boldsymbol{u_n}),\\
\hat{\boldsymbol{n}} = \sigma(\hat{\boldsymbol{W}}_{\boldsymbol{n}} \boldsymbol{C_{mn}^{\top}} \boldsymbol{\alpha_n} + \hat{\boldsymbol{b}}_{\boldsymbol{n}});
\end{cases}\\
&\boldsymbol{u_{mn}} = \hat{\boldsymbol{m}} \oplus \hat{\boldsymbol{n}};
\end{aligned}
\tag{3}
$$

where $\boldsymbol{m}, \boldsymbol{n} \in \{\boldsymbol{t}, \boldsymbol{a}, \boldsymbol{v}\}$ and $\boldsymbol{m} \neq \boldsymbol{n}$, σ represents an activation function, $\hat{\boldsymbol{m}}$ and $\hat{\boldsymbol{n}}$ refer to the reconstructed vectors of $\boldsymbol{m}$ and $\boldsymbol{n}$, respectively, and the remaining variables stand for latent parameters learned from training. The final unified representation $\boldsymbol{u}$ is defined as $\boldsymbol{u} = \boldsymbol{u_{tv}} \oplus \boldsymbol{u_{ta}} \oplus \boldsymbol{u_{av}}$.

3.3 Training and Optimization

The Adam algorithm (Kingma and Ba, 2015) is used as optimizer to minimize the binary cross entropy function. The binary cross entropy is defined as follows:

$$
\mathcal{L} = -\frac{1}{N} \sum_{n=1}^{N} [y_n \log \hat{y}_n + (1 - y_n) \log(1 - \hat{y}_n)], \tag{4}
$$

where N denotes the number of target values, y_n denotes the $n^{th}(1 \leq n \leq N)$ true value, and $\hat{y}_n$ denotes the corresponding predicted value. For the information retrieval task, given a query, we randomly generate a fake candidate along with the correct candidate from the candidate corpus for effective training.

4 Experimental Evaluation

Evaluation is conducted on two kinds of tasks, sentiment analysis and music information retrieval, as presented in Sections 4.1 and 4.2, respectively[2].

4.1 Sentiment Analysis

Task Description. Sentiment analysis is a task that deals with the computational detection and extraction of opinions, beliefs, and emotions in given content (Paltoglou and Thelwall, 2017). In this paper, we analyze posts from a social media platform ShutterSong. Given a post, which includes the visual (image), textual (song lyrics and user defined caption), and acoustic (song clip) parts, the goal is to tag a "positive" or "negative" label for the multi-modal content. We use "1" to indicate the positive label, and "0" to indicate the negative label. The sentiment analysis task can be regarded as a binary classification task.

Dataset and Parameter Settings. We sort a sub dataset from the ShutterSong dataset[3] to conduct the sentiment analysis task. The user-defined mood tags are used as labels, and 3260 items have available moods. After removing duplications, we obtain 272 mood tags. We divide these mood tags into "positive" and "negative" on the basis of the related meaning manually. We obtain 2297 positive and 963 negative samples, with each including a user posted image, a song clip, and its corresponding lyrics, part of the samples have available captions. We call this dataset as MoodShutter, and separate it into train (80%), validation (10%), and test (10%) sets.

The song lyrics are truncated at 100 words, and captions are cleaned into five words according to a caption corpus. We use a 300 dimensional embedding matrix, which is pre-trained on the English

[2]The source code and more details on the experimental settings are available at `http://github.com/mengshor/JTAV`

[3]`https://drive.google.com/file/d/0B2N8XiDRrEgISXFJSXBEMWpUMDA/view`.

Wikipedia dataset[4] through FastText, as W_e. The number of the hidden units of the BiGRU encoder is set as 50. Raw song clips are transformed into spectrograms with the *librosa* tool (McFee et al., 2017). We cut all the audio files into 10-second-segments. Audio sample rates are set as 22050Hz, hop lengths are 1024, and the numbers of frequency bins are 96. The parameter N of DCRNN is set as $N = 2$. The images are sized to $224 \times 224 \times 3$.

Baseline approaches. We compare our JTAV framework with:

- doc2vec (Le and Mikolov, 2014), a paragraph embedding approach;
- lyricsHAN (Alexandros, 2017), a hierarchical attention network which encodes songs into vectors;
- BiGRU, a bidirectional GRU network which utilizes the song lyrics as input;
- attBiGRU, our novel text modeling approach which adds the available captions as the supplying role based on BiGRU;
- MFCC (McFee et al., 2017), a type of cepstral representation of the audio clip generated by *librosa*;
- convnet (Choi et al., 2017a), an acoustic feature learning method based on transfer learning;
- DCRNN-MelS, our novel audio modeling approach which takes MelSs as input;
- DCRNN-CQT, our novel audio modeling approach which takes CQTs as input;
- F-VGG (Simonyan and Zisserman, 2015), a VGG network pre-trained on the ImageNet dataset and fine tuned on the MoodShutter dataset;
- F-ResNet (He et al., 2016), a ResNet pre-trained on the ImageNet dataset and fine tuned on the MoodShutter dataset;
- F-DenseNet (Huang et al., 2017), a DenseNet pre-trained on the ImageNet dataset and fine tuned on the MoodShutter dataset; and
- early fusion, a method which combines features learned by our methods before classification or regression tasks, and is very popularly used (Chen et al., 2017; Oramas et al., 2017).

Evaluation Metrics. The positive and negative samples are unbalanced and the ratio is about $7 : 3$. The weighted average area under the curve (AUC) score, F1 score, and precision score are chosen as evaluation metrics[5].

Results and Discussion. Table 1 presents the results between JTAV and the baseline approaches. For fair comparison, all feature vectors are obtained through optimized training; the classic logistic regression approach is utilized as the classier; and the reported results are the average values of 10 runs.The values in boldface represent the best results among all the approaches. The proposed JTAV performed the best on all metrics, with 0.623 AUC score, 0.691 F1 score, 68.8% precision score.

Observations on single modality. For textual content, attBiGRU obtained the best results, and was superior to doc2vec with about 7 percentage points promotion of AUC score, 11 percentage points promotion of F1 score, and 6 percentage points promotion of precision score. These results demonstrate the proposed BiGRU approach extracts more powerful features than doc2vec and lyricsHAN. And taking the omni textual component into consideration, the attBiGRU approach can generate better results. For acoustic content, DCRNN outperforms MFCC and convnet because it can learn acoustic information both locally and globally. Long-term temporal context dependency is also retained.We also observe that CQT works better than MelS. DenseNet is the best choice for encoding images into vectors as compared with VGG and ResNet.

[4] `https://en.wikipedia.org/wiki/Wikipedia:Database_download#English-language_Wikipedia`.

[5] We utilize the implementation in sklearn, `http://scikit-learn.org/`.

Table 1: Results of the sentiment analysis task

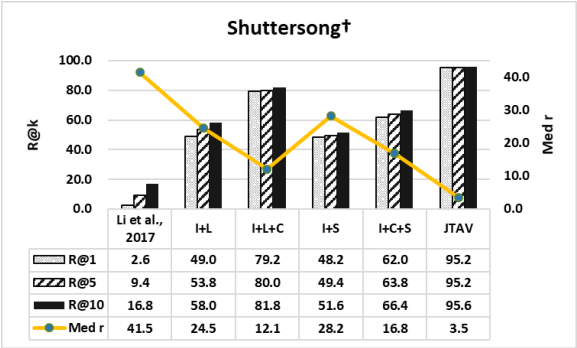
modal	materials	approaches	AUC score	F1 score	precision score
text	lyrics	doc2vec (Le and Mikolov, 2014)	0.513	0.545	0.593
		lyricsHAN (Alexandros, 2017)	0.518	0.575	0.596
		BiGRU	0.572	0.602	0.640
	lyrics+caption	attBiGRU	0.581	0.652	0.650
audio	song clip	MFCC (McFee et al., 2017)	0.505	0.549	0.586
		convnet (Choi et al., 2017a)	0.518	0.614	0.621
		DCRNN-MelS	0.536	0.635	0.634
		DCRNN-CQT	0.559	0.651	0.643
image	image	F-VGG (Simonyan and Zisserman, 2015)	0.546	0.594	0.619
		F-ResNet (He et al., 2016)	0.578	0.618	0.645
		F-DenseNet (Huang et al., 2017)	0.588	0.627	0.653
text+audio	lyrics+caption+ song clip	early fusion	0.586	0.660	0.656
		CMF-AP	0.597	0.673	0.668
text+image	lyrics+caption+ image	early fusion	0.593	0.663	0.661
		CMF-AP	0.611	0.688	0.683
audio+image	image+song clip	early fusion	0.589	0.624	0.653
		CMF-AP	0.603	0.654	0.665
text+audio+image	lyrics+caption+ image+song clip	early fusion	0.602	0.671	0.669
		CMF-AP	**0.623**	**0.691**	**0.688**
JTAV=attBiGRU+(F-DenseNet)+(DCRNN-CQT)+(CMF-AP)					

Observations on multiple modalities. When more modalities emerge, more information is included, and better results are obtained. For example, the performance of combining textual and acoustic information is much better than that of using texts or audio solely. Our CMF-AP approach works steadily better than the baseline approach, because it not only extracts the inner features inside single modality, but also the the outer information cross modalities. Moreover, the best results are presented when we utilize all modalities and all available materials (i.e., JTAV). The AUC score is 0.623, representing an improvement of 2.1 percentage points to the early fusion approach. The F1 score is 0.691, performing an improvement of 2 percentage points over the best baseline approach. The precision score is 0.688, representing about 1.9 percentage points improvement to the best baseline approach. It seems like that the promotions of the proposed JTAV over the best baseline approach is not that huge. As a matter of fact, the early fusion approach utilizes the effective features generated by our proposed work (i.e., attBiGRU, DCRNN-CQT, and F-DenseNet). We believe that, if compared to the original version of early fusion method which only uses the previous features, the margin would be larger. For the sake of space limitation, in Table 1 the early fusion approach on previous features are omitted.

4.2 Music Information Retrieval

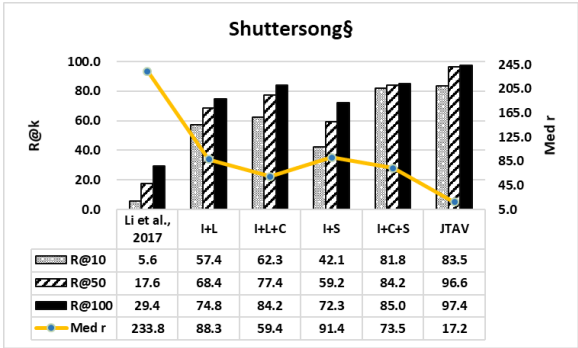
We conduct an extended experiment, which is can be explained as given an image query and named as image2song (Li et al., 2017), to verify the general effectiveness of JTAV. Image2song is a music information retrieval task and aims to find the most relevant song.

We perform the image2song task on two benchmark datasets, they are the Shuttersong† dataset and the Shuttersong§ dataset (Li et al., 2017). Both datasets include no-repeat 620 songs with 3100 images, that is, each song is related to five images. Part of the images are attached with user-defined captions, and each song has an acoustic song clip and corresponding song lyrics. The difference lies in the partition strategy of the train and test dataset. In Shuttersong†, 100 songs and related images are selected randomly for testing, and the rest for training. While in Shuttersong§, the train and test set share the whole song



	Li et al., 2017	I+L	I+L+C	I+S	I+C+S	JTAV
R@1	2.6	49.0	79.2	48.2	62.0	95.2
R@5	9.4	53.8	80.0	49.4	63.8	95.2
R@10	16.8	58.0	81.8	51.6	66.4	95.6
Med r	41.5	24.5	12.1	28.2	16.8	3.5

Figure 5: Results of image2song on Shuttersong†



	Li et al., 2017	I+L	I+L+C	I+S	I+C+S	JTAV
R@10	5.6	57.4	62.3	42.1	81.8	83.5
R@50	17.6	68.4	77.4	59.2	84.2	96.6
R@100	29.4	74.8	84.2	72.3	85.0	97.4
Med r	233.8	88.3	59.4	91.4	73.5	17.2

Figure 6: Results of image2song on Shuttersong§

set, while one of the five images of each song is chosen randomly for testing. The preprocessing settings of images, captions, song lyrics and song clips is the same as that of the sentiment analysis task.

We utilize the rank-based evaluation metrics to compare the results. For fair comparison, we adopt the evaluate metrics in (Li et al., 2017). Med r represents the medium rank of the ground truth retrieved song, and lower values indicate better performance. Recall@k (R@k for short), is the percentage of a ground truth song retrieved in the top-k ranked items, and higher values indicate better performance. In dataset†, $k = \{1, 5, 10\}$, and $1 \leq$ Med r ≤ 100; whereas in Shuttersong§, $k = \{10, 50, 100\}$, and $1 \leq$ Med r ≤ 620.

We compare the proposed JTAV with the state-of-the-art approach (Li et al., 2017). For a more comprehensive comparison, we design several baseline approaches based on our proposed strategies: I+L uses images as queries and song lyrics as candidates with a suit of BiGRU, F-DenseNet and CMF-AP; I+C+L uses images and available captions as queries while song lyrics as candidates with a suit of attBiGRU, F-DenseNet and CMF-AP; I+S uses images as queries and song clips as candidates with a suit of DCRNN-CQT, F-DenseNet and CMF-AP; and I+C+S uses images and available captions as queries while song clips as candidates with a suit of DCRNN-CQT, F-DenseNet and CMF-AP. In JTAV, the images and available captions form the queries, the song clips and lyrics form the candidates.

Fig. 5 and Fig. 6 display the results on Shuttersong† and Shuttersong§, respectively. Both prove JTAV is the most powerful one among all testing approaches. JTAV turns the image2song task from finding the most similar item to finding the most relevant item, which is nearer the real situation in cross-modal retrieval tasks. Notably, JTAV exceeds the state-of-the-art approaches soundly. In Shuttersong†, the Med r is no more than 5, and the R@1 score is more than 95%, which is more than 90% better than that obatined in (Li et al., 2017). In Shuttersong§, the Med r is no more than 20, which is less a tenth of that obtained in (Li et al., 2017), the R@10 score is more than 83%, and the R@100 score is more than 97%.

5 Conclusion

In this paper, we aim to address the issue of learning social media content from the multi-modal view. We have designed effective approaches (i.e., attBiGRU) for textual content, F-DenseNet for visual content, and DCRNN for acoustic content, to extract fine-grained features. We have introduced CMF-AP to generate cross-modal aware matrices and reconstructed modal vectors, which are used to produce an unified representation of the multi-modal content. The proposed framework has been intensively evaluated, and the experimental results have demonstrated the general validity of JTAV by comparing with the state-of-the-art approaches.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant No.U1636116, 11431006, 61772288, the Research Fund for International Young Scientists under Grant No. 61650110510 and 61750110530, and the Ministry of education of Humanities and Social Science project under grant 16YJC790123.

References

Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. 2008. Finding high-quality content in social media. In *Proceedings of the international conference on web search and data mining*, pages 183–194. ACM.

Tsaptsinos Alexandros. 2017. Lyrics-based music genre classification using a hierarchical attention network. In *Proceedings of the 18th International Society of Music Information Retrieval Conference*, pages 694–701. ISMIR.

Boualem Boashash. 1996. Time frequency signal analysis: Past, present and future trends. In *Control and Dynamic Systems*, volume 78, pages 1–69. Elsevier.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Judith C Brown and Miller S Puckette. 1992. An efficient algorithm for the calculation of a constant Q transform. *The Journal of the Acoustical Society of America*, 92(5):2698–2701.

Xingyue Chen, Yunhong Wang, and Qingjie Liu. 2017. Visual and textual sentiment analysis using deep fusion convolutional neural networks. In *Proceedings of the 2017 IEEE International Conference on Image Processing*, pages 296–300. IEEE.

Keunwoo Choi, George Fazekas, Mark Sandler, and Kyunghyun Cho. 2017a. Transfer learning for music classification and regression tasks. In *Proceedings of the International Society of Music Information Retrieval Conference*. ISMIR.

Keunwoo Choi, György Fazekas, Kyunghyun Cho, and Mark B. Sandler. 2017b. A tutorial on deep learning for music information retrieval. *CoRR*, abs/1709.04396.

Venkata Rama Kiran Garimella, Abdulrahman Alfayad, and Ingmar Weber. 2016. Social media image analysis for public health. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 5543–5547. ACM.

Bo Han, Paul Cook, and Timothy Baldwin. 2013. Lexical normalization for social media text. *ACM Transactions on Intelligent Systems and Technology*, 4(1):5.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778. IEEE.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708.

Clayton J Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the 8th international AAAI conference on weblogs and social media*.

Aren Jansen, Manoj Plakal, Ratheet Pandya, Dan Ellis, Shawn Hershey, Jiayang Liu, Channing Moore, and Rif A. Saurous. 2017. Towards learning semantic audio representations from unlabeled data. In *Proceedings of Workshop on Machine Learning for Audio Signal Processing at the 31st Conference on Neural Information Processing Systems*. NIPS.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3th International Conference on Learning Representations*. ICLR.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of International Conference on Machine Learning*, pages 1188–1196.

Xuelong Li, Di Hu, and Xiaoqiang Lu. 2017. Image2song: Song retrieval via bridging image content and lyric words. In *Proceedings of IEEE International Conference on Computer Vision*, pages 5650–5659.

Brian McFee, Matt McVicar, Oriol Nieto, Stefan Balke, Carl Thome, Dawen Liang, Eric Battenberg, Josh Moore, Rachel Bittner, Ryuichi Yamamoto, et al. 2017. librosa 0.5.0.

Sergio Oramas, Oriol Nieto, Francesco Barbieri, and Xavier Serra. 2017. Multi-label music genre classification from audio, text, and images using deep features. In *Proceedings of the 18th International Society of Music Information Retrieval Conference*. ISMIR.

Georgios Paltoglou and Mike Thelwall. 2017. Sensing social media: a range of approaches for sentiment analysis. In *Cyberemotions*, pages 97–117. Springer.

Sharaj Panwar, Arun Das, Mehdi Roopaei, and Paul Rad. 2017. A deep learning approach for mapping music genres. In *Proceedings of the 12th System of Systems Engineering Conference*, pages 1–5. IEEE.

Gwangbeen Park and Woobin Im. 2016. Image-text multi-modal representation learning by adversarial backpropagation. *CoRR*, abs/1612.08354.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 3th International Conference on Learning Representations*.

Zhi Wang, Wenwu Zhu, Peng Cui, Lifeng Sun, and Shiqiang Yang. 2013. Social media recommendation. In *Social Media Retrieval*, pages 23–42. Springer.

Yilin Wang, Suhang Wang, Jiliang Tang, Huan Liu, and Baoxin Li. 2015. Unsupervised sentiment analysis for social media images. In *Proceedings of the International Conference on Artificial Intelligence*, pages 2378–2379. AAAI Press.

Chao Wu, Yaoxue Zhang, Jia Jia, and Wenwu Zhu. 2017. Mobile contextual recommender system for online social media. *IEEE Transactions on Mobile Computing*, 16(12):3403–3416.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 14th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489. NAACL.

Bilei Zhu, Fuzhang Wu, Ke Li, Yongjian Wu, Feiyue Huang, and Yunsheng Wu. 2017. Fusing transcription results from polyphonic and monophonic audio for singing melody transcription in polyphonic music. In *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE.

MEMD: A Diversity-Promoting Learning Framework for Short-Text Conversation

Meng Zou **Xihan Li** **Haokun Liu** **Zhihong Deng***
Key Laboratory of Machine Perception (Ministry of Education)
School of Electronics Engineering and Computer Science
Peking University, Beijing 100871, China
{mengzou, xihanli, 1300012769, zhdeng}@pku.edu.cn

Abstract

Neural encoder-decoder models have been widely applied to conversational response generation, which is a research hot spot in recent years. However, conventional neural encoder-decoder models tend to generate commonplace responses like *"I don't know"* regardless of what the input is. In this paper, we analyze this problem from a new perspective: latent vectors. Based on it, we propose an easy-to-extend learning framework named MEMD (Multi-Encoder to Multi-Decoder), in which an auxiliary encoder and an auxiliary decoder are introduced to provide necessary training guidance without resorting to extra data or complicating network's inner structure. Experimental results demonstrate that our method effectively improve the quality of generated responses according to automatic metrics and human evaluations, yielding more diverse and smooth replies.

1 Introduction

Human-computer conversation is attracting particular attention recently. Research in this field falls into two categories: the retrieval-based method (Ji et al., 2014; Yan et al., 2017; Wu et al., 2017) and the generative method (Shang et al., 2015; Serban et al., 2016). While the retrieval-based method can guarantee completeness of output sentences, it fails to customized for particular posts from users. By contrast, the generative method may produce sentences with grammatical errors, however, it shows great promise in flexibility, which gives rise to a research hot spot.

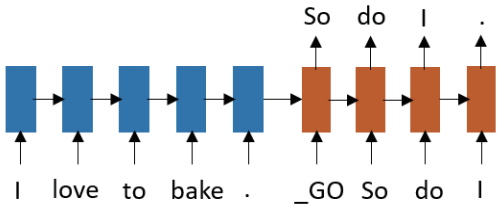


Figure 1: Conventional encoder-decoder model. Blue part represents the encoder, while red part represents the decoder.

Of the generative method, neural encoder-decoder based model (Sutskever et al., 2014) has become the mainstream (Figure 1). (Shang et al., 2015) firstly formalized the generation of response as a decoding process based on the latent representation of the input text, and both encoding and decoding are realized with recurrent neural networks (RNN). Such formalization is widely adopted by later work. However, the conventional encoder-decoder model's performance is far from satisfactory, for it tends to generate meaningless and generic responses like *"I don't know"*.

To tackle the issue of response diversity, lots of models have been proposed, and they can be broadly divided into three categories: (1) introducing external priori knowledge into the procedure of encoding/decoding (Mou et al., 2016; Xing et al., 2017), which usually requires pretreatment on extra large

*Zhihong Deng is the corresponding author.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1281–1291
Santa Fe, New Mexico, USA, August 20-26, 2018.

data. (2) complicating network's structure to enhance model's capacity for encoding/decoding (Zhou et al., 2017; Serban et al., 2017a; Serban et al., 2017b; Zhao et al., 2017; Clark and Cao, 2017; Shen et al., 2017). (3) directly modifying the objective function to penalize generation probability of trivial responses (Li et al., 2016). However, these models merely discuss the latent vector, which is the vital link connecting the encoder and the decoder together.

Different from previous work, we analyse the problem of general response generation from the perspective of latent vectors. We notice that in the basic encoder-decoder model, what the decoder needs to generate a response is only a hidden vector. In other words, for the same decoder, latent vectors directly decide what will be generated. If latent vectors are clustered, the decoder is likely to generate similar responses. If latent vectors are dispersed, the decoder is likely to generate diverse responses. Based on the conjecture that dispersion of latent vectors has positive correlation with diversity of generated responses, encoder-decoder model's poor performance can be discussed in the two following situations:

- Suppose that the decoder is capable to generate different sentences given different latent vectors, model's unsatisfying performance of generating meaningless sentences should be imputed to the encoder—it tends to map whatever inputs to similar vectors. In this situation, to promote diversity in generation, the encoder should be encouraged to generate different latent vectors given different inputs.

- Suppose that the encoder is capable to generate different latent vectors with specific and concrete semantics given different inputs, model's unsatisfying performance of generating meaningless sentences should be imputed to the decoder—it is insensitive to latent vectors and tends to generate similar sentences given whatever latent vectors. In this situation, to promote diversity in generation, decoder's capacity for generating sentences should be enhanced.

During training, however, since there is no constraint on the latent vector's specificity, neither encoder's capacity for latent vector generation nor decoder's capacity for sentence generation can be guaranteed, which leads to poor performance in conversation generation.

For the above motivation, we propose a learning framework named MEMD (Multi-Encoder to Multi-Decoder). In proposed framework, an auxiliary encoder, which aims at guiding the major encoder to generate diverse latent vectors, and an auxiliary decoder, which aims at providing latent vectors with specific and concrete semantics for the major decoder to "practice" decoding, are introduced. During training, parameters of these two encoders and two decoders are updated. While in test, only the major encoder and the major decoder are employed.

In summary, our contributions are as follows:

- We present a new angle to tackle the problem of response diversity—the latent vectors generated by the encoder.

- We propose an easy-to-extend learning framework: MEMD, which introduces necessary training guidance for both encoder and decoder without resorting to extra data or complicating inner structure of networks.

- Experimental results demonstrate that MEMD effectively improves the quality of generated responses according to automatic metrics and human evaluations, yielding more diverse and smooth replies.

2 Technical Background

2.1 Gated Recurrent Unit (GRU)

GRU (Cho et al., 2014) is a special kind of RNN, which is widely used for learning long-term dependencies. It is defined as follows: given a sequence of inputs $(w_1, w_2, \ldots, w_N)$, GRU iterates each timestep with an update gate z_n and a reset gate r_n. Let h_n denote the vector of hidden layer computed by GRU at time n, σ denote the sigmoid function and $\odot$ denote the element-wise product. The vector

representation of hidden layer for each timestep n is given by:

$$z_n = \sigma(W_{zw}w_n + W_{zh}h_{n-1}) \quad (1)$$
$$r_n = \sigma(W_{rw}w_n + W_{rh}h_{n-1}) \quad (2)$$
$$\widetilde{h}_n = tanh(W_{hw}w_n + W_{hh}(r_t \odot h_{t-1})) \quad (3)$$
$$h_n = (1 - z_n)h_{n-1} + z_n\widetilde{h}_n \quad (4)$$

where W_{*w} is the transformation matrix from the input to GRU states, W_{*h} is the recurrent transformation matrix between the recurrent states h_n.

2.2 Encoder-decoder Models

In an encoder-decoder model, given a source sequence message $X = (x_1, x_2, \ldots, x_M)$ and a target sequence response $Y = (y_1, y_2, \ldots, y_N)$, the model would maximizes the generation probability of Y conditioned on X. While the encoder reads X word by word and represents it as a latent vector h_X through a recurrent neural network (RNN), the decoder estimates the generation probability of Y with h_X as initial state. The objective function of the model is as follows:

$$p(y_1, \ldots, y_N|x_1, \ldots, x_M) = p(y_1|X)\prod_{t=2}^{N} p(y_t|y_1, \ldots, y_{t-1}, X), \quad (5)$$

The latent vector h_X is calculated by

$$h_t = f(x_t, h_{t-1}) \quad (6)$$
$$h_X = h_M \quad (7)$$

where h_t is the hidden state at time t and f is a non-linear transformation which can be a gated recurrent unit (GRU). The decoder is a standard RNN language model except the addition of the context vector c. The probability distribution p_t of candidate words at each timestep t is calculated as

$$s_0 = h_X \quad (8)$$
$$s_t = f(y_{t-1}, s_{t-1}) \quad (9)$$
$$p_t = softmax(s_t) \quad (10)$$

where s_t is the hidden state of the decoder RNN at timestep t.

2.3 Attention Mechanism

The traditional sequence-to-sequence model assumes that each word is generated from the same context vector. However, in practice, different words in Y might be related to different words or phrases in X. In order to solve this problem, attention mechanism (Bahdanau et al., 2015; Cho et al., 2014) is introduced into this model. With attention, the context vector c_i corresponded to each y_i in Y is a weighted average of all hidden states of the encoder. Formally, c_i is defined as

$$c_i = \sum_{j=1}^{M} \alpha_{ij}h_j \quad (11)$$
$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{M} exp(e_{ik})} \quad (12)$$
$$e_{ij} = \eta(s_{i-1}, h_j) \quad (13)$$

where η is a multi-layer perceptron (MLP).

3 Proposed Framework

The basic idea of MEMD is simple: we want the encoder to be able to generate different latent vectors given different inputs and the decoder to be able to generate sentence given a latent vector. We prevent the encoder from generating similar vectors by requiring that posts themselves should be reconstructed from the latent vectors, since if vectors are similar, they cannot be interpreted into diverse sentences by the same decoder. And for decoder, we strengthen its decoding ability by requiring it to reconstruct responses given vectors that really encode responses, since latent vectors' effectiveness cannot be guarantee when the encoder takes as input not responses but posts.

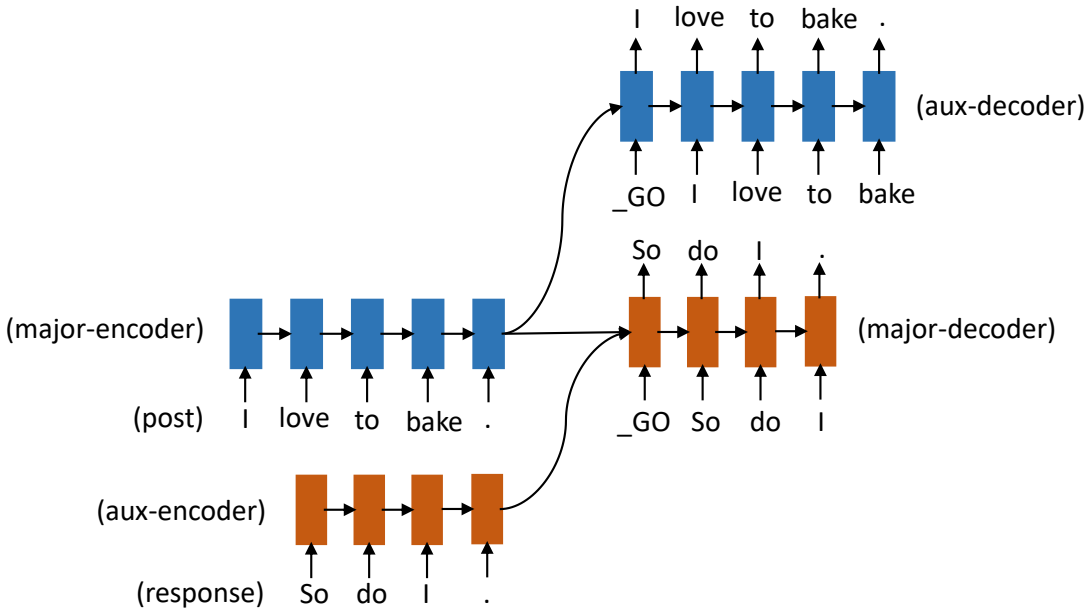


Figure 2: Architecture of MEMD. This framework contains four main components: 1) a major encoder, 2) a major decoder, 3) an auxiliary encoder, and 4) an auxiliary decoder. Three training paths are constructed: 1) from major-encoder to aux-decoder, 2) from major-encoder to major-decoder, and 3) from aux-encoder to major-decoder. Blue parts' inputs/targets are posts, while red parts' inputs/targets are responses.

3.1 Model Architecture

Suppose that short-text conversation consists of a post and a response, we denote a post of length M as $X = [x_1, \ldots, x_M]$ and a response of length N as $Y = [\_GO, y_1, \ldots, y_N]$ where $\_GO$ is a special symbol indicating the begin of a response. Each training example is a (post, response) pair, namely, (X, Y).

Figure 2 illustrates the architecture of MEMD, in which there are two encoders—major encoder (major-encoder) and auxiliary encoder (aux-encoder), and two decoders—major decoder (major-decoder) and auxiliary decoder (aux-decoder). These four components constitute three training paths: path 1 is from major-encoder to aux-decoder, path 2 is from major-encoder to major-decoder, and path 3 is from aux-encoder to major-decoder. Note that the major-encoder produces only one latent vector when fed one post, and the major-decoder receives only one latent vector per decoding procedure. In other words, the major-encoder is shared between path 1 and path 2, and the major-decoder is shared between the path 2 and path 3.

The major-encoder takes as input a post X, and returns its hidden state vector at the last step, i.e., h_M, as output. The aux-encoder has the same structure as the major-encoder, but takes as input the response Y. The output of the aux-encoder is also it's hidden state vector at the last step, i.e., $\hat{h}_N$. The major-decoder takes as input h_M or $\hat{h}_N$, and it's generation target is Y. When it takes h_M as input, it constitutes

a conventional encoder-decoder model with the major-encoder. When it takes $\hat{h}_N$ as input, it constitutes an auto-encoder with the aux-encoder. The aux-decoder has same structure as the major-decoder. It takes as input h_M, and its generation target is X.

3.2 Training Procedure

We first present the training objective along each path, and then give the whole training algorithm.

We denote the parameters of the major-encoder as θ_{ME}, and the parameters of aux-decoder as θ_{AD}. The training objective of path 1 (from the major-encoder to the aux-decoder) is:

$$\min_{\theta_{ME},\theta_{AD}} L_{ME-AD} = -\log p_1(X|X) \tag{14}$$

The parameters of the major-decoder are denoted as θ_{MD}. The training objective of path 2 (from the major-encoder to the major-decoder) is:

$$\min_{\theta_{ME},\theta_{MD}} L_{ME-MD} = -\log p_2(Y|X) \tag{15}$$

The parameters of the aux-encoder are denoted as θ_{AE}. The training objective of path 3 (from the aux-encoder to the major-decoder) is:

$$\min_{\theta_{AE},\theta_{MD}} L_{AE-MD} = -\log p_3(Y|Y) \tag{16}$$

The overall training objective is:

$$\min_{\theta_{ME},\theta_{AE},\theta_{MD},\theta_{AD}} L = L_{ME-AD} + L_{ME-MD} + L_{AE-MD} \tag{17}$$

In actual implementation, for the sake of flexibility and extendibility, we don't directly optimize Eq.(17) but interleave the optimization of Eq.(14), Eq.(15) and Eq.(16) at each iteration, which is inspired by the alternating training approach (Dong et al., 2015). Algorithm 1 summarizes the training procedure.

Algorithm 1 MEMD for short-text conversation

Input: Training data $\{(X, Y)_n\}$
Output: major-encoder, aux-encoder, major-decoder, aux-decoder

1: Initialize $\theta_{ME}, \theta_{AE}, \theta_{MD}, \theta_{AD}$
2: **repeat**
3: Train through path 1 by Eq.(14)
4: Train through path 2 by Eq.(15)
5: Train through path 3 by Eq.(16)
6: **until** convergence

3.3 Discussion

Our proposed framework MEMD is seemingly similar to the many-to-many setting in multi-task sequence-sequence learning (Luong et al., 2015). However, there are obvious distinctions between MEMD and multi-task sequence-sequence learning. MEMD aims at introducing constrains for encoder and decoder from the perspective of latent vectors, and does not require extra data for training. These introduced constrains are designed based on the characteristics of conversational response generation task. The many-to-many setting in multi-task sequence-sequence learning, however, aims at improving the generalization performance of the central task by resorting to training data of other related tasks.

4 Experiments

4.1 Dataset

We carry out experiment on an open-domain dialogue dataset: STC-weibo corpus developed by (Shang et al., 2015). STC-weibo corpus consists of millions of post-response pairs crawled from Weibo[1], which is popular Twitter-like microblogging service in China and has length limit of 140 Chinese characters on both posts and responses. We filter post-response pairs that include "alink" which represents a hyperlink, since we find that sentences are low-quality when "alink" appears. Besides, each post corresponds to 28 different responses at average. To minimize noise, we selected the response that contains the maximum number of frequent bigram in the whole corpus. After data cleaning, we finally get 199384 post-response pairs, and conduct the train/dev/test split of 197424/1000/960.

4.2 Baselines

We use the following models that needn't resort to extra data as our baselines for fair comparison:

Enc-Dec: the standard encoder-decoder model.

Enc-Dec-A: the standard encoder-decoder model with attention.

MMI: the best performing model in (Li et al., 2016).

For each baseline, there is a corresponding version of MEMD, whose major-encoder and major-decoder are the same to the baseline's encoder and decoder respectively. In other words, the baseline and its corresponding MEMD are structurally identical in test. Under this controlled setting we can validate the effectiveness of the proposed learning framework.

4.3 Implementation Details

We implement models in TensorFlow[2] and train them using Adam. The encoder is implemented as bidirectional GRU, and the decoder is implemented as multi-layer GRU (3 layers in Enc-Dec and Enc-Dec-A, 2 layers in MMI). The dimensions of hidden state are set to be 512 in Enc-Dec and Enc-Dec-A, and 256 in MMI. We use 100-dimension word embedding, and keep the size of vocabulary to be 60000. The word embedding is pretrained on the training set and updated during training. We set the learning rate to be 2×10^{-3} for path 2 and 3×10^{-4} for path 1 and path 3. And the batch size is set to be 48. We test the model on development data every 1000 mini-batches. When the model's performance on object function doesn't improve within 4 successive tests on development data, we view it convergent and stop training.

4.4 Evaluation metrics

Distinct-1 & distinct-2: Follow (Xing et al., 2017), we counted numbers of distinct unigrams and bigrams in the generated responses, and divide them by the total number of unigram and bigram respectively. The higher these two metrics are, the more informative and diverse the generated responses are.

Distinct-B & distinct-S: To measure the diversity of sentence pattern, we counted the number of distinct four words at the beginning of sentences, and divide them by the total number of generated sentences. We denote this metric as distinct-B. Moreover, we count the number of distinct sentence, and also calculate the ratio of distinct sentence to the total number of generated sentences. We denote this metric as distinct-S. The higher these two metrics are , the more diverse the generated responses are.

Sentence-level BLEU: Inspired by metrics used for evaluating machine translation, we use BLEU (Chen and Cherry, 2014) to evaluate the responses generated by different models.

Human annotation: Since automatic metrics may not consistently agree with human perception (Stent et al., 2005), we conduct human evaluation on 50 randomly sampled generated sentences. Three labelers with rich Weibo experience were invited to do evaluation. Responses generated by different models were pooled and randomly shuffled for each labeler. we adopt the criteria used in (Xing et al., 2017):

+2:The response is not only relevant and natural, but also informative and interesting.

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

[2]https://www.tensorflow.org/

+1: The response can be used as a reply to the message, but it is too universal like “Yes, I see”, “Me too” and “I don’t know”.

0: The response cannot be used as a reply to the message. It is either semantically irrelevant or disfluent (e.g., with grammatical errors).

4.5 Results and Analysis

We investigate three strategies to initialize parameters of the major-encoder and the major-decoder. From Table 1 we can find that the way of initialization have a great influence on model's performance. When major-encoder and the major-decoder are pretained through path 1 and path 3 (advance-1&3), which in fact constitute two independent auto-encoders, the model tends to use the same words, and lacks variety on sentence pattern according to distinct-1, distinct-2, distinct-B and distinct-S. What's more, results of human annotation suggest that generated sentences are low-quality. When no pretraining is adopted, the model's diversity and generation quality are improved. When pretraining is conducted on path 2 (advance-2), which means the major-encoder and the major-decoder are initialized with a convergent Enc-Dec, the model gets significant improvement on diversity and generation quality. We notice that BLEU scores are somewhat incompatible to human annotation, here we put more priority on human annotation, and use BLEU scores as reference.

pretrain strategy	distinct-1	distinct-2	distinct-B	distinct-S	BLEU	+2	+1	0
advance-1&3	102/.010	308/.038	69/.072	388/.404	**0.569**	16%	54%	30%
no pretrain	172/.017	478/.059	154/.160	456/.475	0.536	26%	52%	22%
advance-2	**882/.086**	**2294/.276**	**547/.570**	**910/.948**	0.559	**40%**	36%	24%

Table 1: Results of MEMD on evaluation metrics. The first four columns are in the format of "the total number/proportion". Before employing algorithm 1, three strategies are adopted to initialize major-encoder's and major-decoder's parameters: 1) train them on path 1 and path 3 in advance. 2)no pretrain. 3)train them on path 2 in advance.

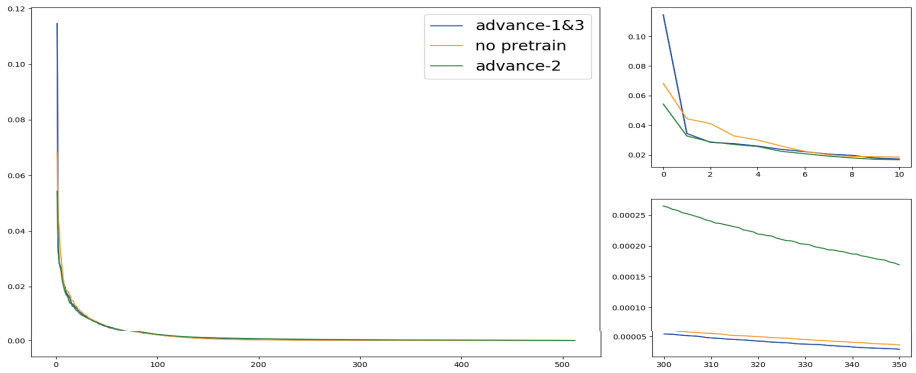


Figure 3: The left subfigure shows models' all 512 sorted PV under different pretraining strategies. The upper-right subfigure shows PV in the interval of 1 to 10, and the lower-right subfigure shows PV in the interval of 300 to 350.

Besides of evaluation metrics, we carry out analysis on latent vectors' distribution under different pretraining strategies. Since latent vectors under different pretraining strategies aren't in the same semantic space, direct comparison is infeasible. To tackle this problem, we utilize PCA (Principal Component Analysis) to get variances along each axes, which are represented by eigenvalues of covariance matrix subtracted the mean of each latent vectors. By dividing the eigenvalue by the sum of all eigenvalues, we get the percentage of variance (PV) explained by corresponding axis. If the variance of PV is large, which indicates variances are mainly distributed along a few axes, latent vectors are clustered. If the variance of

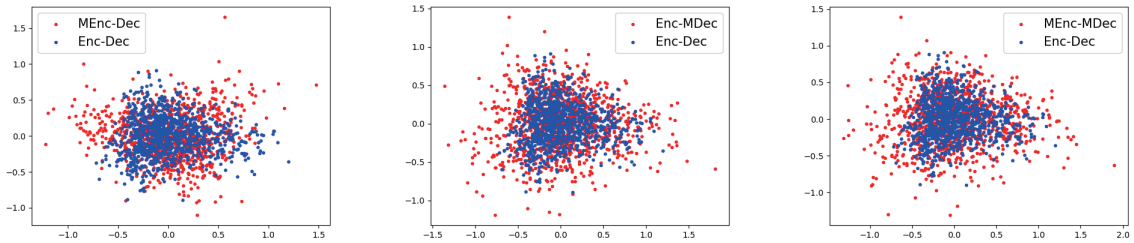
PV is small, which indicates they may have high variances along many axes, latent vectors are scattered. We use the variance of PV to indicate the dispersion of latent vectors. Models' sorted PV along all 512 axes are showed in Figure 3.

From Figure 3 we can clearly observe that the blue line has the largest value among three lines at the very beginning, then it drops down most sharply, and keeps small value later. This indicates that its most part of variances are occur on only a few axes, which means the latent vectors are clustered in the semantic space. The green line has the smallest value among three lines at the beginning and drops down most smoothly. This indicates its variances are distributed more evenly on all axes, so the latent vectors are more scattered in the whole semantic space. The PV variances of advance-1&3, no pretrain, and advance-2 are 4.251×10^{-5}, 3.142×10^{-5}, and 2.138×10^{-5} respectively, indicating that latent vectors of them are more scattered accordingly. The sorted PV distribution and variances of PV echo the results from Table 1.

To validate the effectiveness of the aux-encoder and the aux-encoder in the proposed MEMD, we designed two variations: 1) MEnc-Dec, which only has aux-encoder. 2)Enc-MDec, which only has aux-decoder. Together with MEMD, we get three models and train them with pretraining method of advance-2. The evaluation results are shown in Table 2. Both MEnc-Dec and Enc-MDec get higher scores on distinct-1, distinct-2, distinct-B, distinct-S, and human annotation comparing with Enc-Dec, which can prove that both aux-encoder and the aux-encoder do help promote diversity in response generation. The effectiveness of aux-decoder and aux-encoder support the two situations discussed in introduction. When compare Enc-MDec with MEnc-Dec, we find the latter brings greater promotion to the baseline on human annotation, which indicates that the aux-encoder plays important role in digesting the major decoder's ability to generate smooth sentences. When both aux-encoder and aux-decoder are adopted, MEMD's performance is further improved and gets best results on distinct-1, distinct-2, distinct-B， distinct-S, and human annotation.

	distinct-1	distinct-2	distinct-B	distinct-S	BLEU	+2	+1	0
Enc-Dec	148/.016	412/.055	145/.151	383/.399	0.555	18%	46%	36%
MEnc-Dec	796/.079	2125/.259	512/.533	870/.906	**0.568**	32%	42%	26%
Enc-MDec	730/.068	2133/.241	536/.558	877/.914	0.554	22%	50%	28%
MEMD	**882/.086**	**2294/.276**	**547/.570**	**910/.948**	0.559	**40%**	36%	24%

Table 2: Results of MEMD's variations on evaluation metrics. The first four columns are in the format of "the total number/proportion". Enc-Dec is the baseline, MEnc-Dec represents Enc-Dec with aux-encoder, and Enc-MDec represents Enc-Dec with aux-decoder.



(a) (b) (c)

Figure 4: Visualization of all latent vectors generated by Enc-Dec, Menc-Dec, Enc-MDec and MEnc-MDec in test.

We visualize two sets of all 960 latent vectors which are generated by Enc-Dec, MEnc-Dec, Dec-MDec and MEMD respectively in Figure 4. Note that the major-encoders and the major-decoders of MEnc-Dec, Dec-MDec and MEMD are initialized using Enc-Dec's parameters. We can clearly observe that after training under the MEMD framework, latent vectors become much scattered. Taking Figure 4

and Table 2 together, we find that dispersion of latent vectors has positive correlation with diversity of generated responses, which supports our conjecture.

	distinct-1	distinct-2	distinct-B	distinct-S	BLEU	+2	+1	0
Enc-Dec	148/.016	412/.055	145/.151	383/.399	0.555	18%	46%	36%
MEMD	**882/.086**	**2294/.276**	547/.570	**910/.948**	0.559	40%	36%	24%
Enc-Dec-A	310/.032	683/.088	162/.168	447/.466	**0.571**	20%	42%	38%
MEMD-A	822/.074	2157/.236	**550/.573**	873/.909	0.562	42%	36%	22%
MMI	345/.056	661/.157	317/.330	500/.521	0.440	32%	48%	20 %
MEMD-M	478/.073	820/.178	344/.358	572/.596	0.463	**46%**	32%	22%

Table 3: Results of baselines and their corresponding MEMD on evaluation metrics. The first four columns are in the format of "the total number/proportion".

To reveal the expansibility of the proposed learning framework, we apply MEMD to the other two baselines—Enc-Dec-A and MMI. From Table 3 we can see that MEMD-A has superior performance than Enc-Dec-A on distinct-1, distinct-2, distinct-B, distinct-S and human annotation, which indicates better diversity and quality of generated responses. MMI can produce smoother and more specific sentences than Enc-Dec and Enc-Dec-A, and MEMD-M gets better performance on distinct-1, distinct-2, distinct-B, distinct-S and human annotation than MMI. We also find that better performance on distinct-1, distinct-2, distinct-B, and distinct-S cannot guarantee better performance on human annotation, which can be explained as sentences contain varied words but have poor quality or are irrelevant to posts. Table 4 exhibits some examples. From Table 4 we can see that responses generated by Enc-Dec are general and lack pertinency to posts, while MEMD can catch some keywords in posts like "Hangzhou", "snow", and "talk". For Enc-Dec-A and MMI, they can sometimes generate natural responses, and based on them, MEMD-A and MEMD-M generate more informative and relevant responses.

Post	**Enc-Dec**	**MEMD**	**Enc-Dec-A**	**MEMD-A**	**MMI**	**MEMD-M**
杭州又下雪了，雪中西湖是最美的 It snows again in Hangzhou, and West Lake in the snow is the most beautiful.	这是哪里啊？我也想去 Where is this? I want to go, too.	杭州下雪了，我也在杭州 It snows in Hangzhou, and I am in Hangzhou, too.	我也想去杭州了。 I want to go to Hangzhou, too.	是杭州吗？杭州下雪了？ Is this Hangzhou? Is it snowing in Hangzhou?	... ...	西湖下雪了，西湖下雪了。 It snows in West Lake, it snows in West Lake.
已过零点，跟自己说一声：生日快乐。 The time has passed twelve o'clock at night, I talked to myself: happy birthday.	我也是 Me too.	你这是在说什么啊？ What are you talking about?	这是一个UNK This is a UNK	生日快乐，我的生日快乐 Happy birthday, happy birthday to me	生日快乐 Happy birthday	祝生日快乐，祝生日快乐 Happy birthday to you, happy birthday to you

Table 4: Case study.

5 Conclusion

In this paper, we view the generation diversity from a new angle: the latent vectors. Base on it, we propose an easy but effective learning framework named MEMD. In experiment, we found that the initialization method for the major-encoder and the major-decoder have great impact on the model's performance. Besides, we verify the effectiveness of the aux-encoder and the aux-decoder, and transfer the basic MEMD to MEMD-A and MEMD-M. We analyze the distribution of latent vectors, and find it consistent with evaluation metrics, which supports our conjecture that dispersion of latent vectors has positive correlation with diversity of generated responses.

Acknowledgements

This work is partially supported by the National High Technology Research and Development Program of China (Grant No. 2015AA015403) and the National Natural Science Foundation of China (Grant No. 61170091). We would also like to thank the anonymous reviewers for their helpful comments.

References

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

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level BLEU. In *Proceedings of the Ninth Workshop on Statistical Machine Translation, WMT@ACL 2014, June 26-27, 2014, Baltimore, Maryland, USA*, pages 362--367.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724--1734.

Stephen Clark and Kris Cao. 2017. Latent variable dialogue models and their diversity. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 182--187.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1723--1732.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *CoRR*, abs/1408.6988.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 110--119.

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

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 3349--3358.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 3776--3784.

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron C. Courville. 2017a. Multiresolution recurrent neural networks: An application to dialogue response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3288--3294.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017b. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3295--3301.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1577--1586.

Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. 2017. A conditional variational framework for dialog generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 504--509.

Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *Computational Linguistics and Intelligent Text Processing, 6th International Conference, CICLing 2005, Mexico City, Mexico, February 13-19, 2005, Proceedings*, pages 341--351.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104--3112.

Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 496--505.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3351--3357.

Rui Yan, Dongyan Zhao, and Weinan E. 2017. Joint learning of response ranking and next utterance suggestion in human-computer conversation system. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 685--694.

Tiancheng Zhao, Ran Zhao, and Maxine Eskénazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 654--664.

Ganbin Zhou, Ping Luo, Rongyu Cao, Fen Lin, Bo Chen, and Qing He. 2017. Mechanism-aware neural machine for dialogue response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3400--3407.

Refining Source Representations with Relation Networks for Neural Machine Translation

Wen Zhang[1,2] **Jiawei Hu**[1,2] **Yang Feng**[1,2*] **Qun Liu**[3,1]

[1]Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, CAS
[2]University of Chinese Academy of Sciences
{zhangwen,hujiawei,fengyang}@ict.ac.cn
[3]ADAPT Centre, School of Computing, Dublin City University
qun.liu@dcu.ie

Abstract

Although neural machine translation with the encoder-decoder framework has achieved great success recently, it still suffers drawbacks of forgetting distant information, which is an inherent disadvantage of recurrent neural network structure, and disregarding relationship between source words during encoding step. Whereas in practice, the former information and relationship are often useful in current step. We target on solving these problems and thus introduce relation networks to learn better representations of the source. The relation networks are able to facilitate memorization capability of recurrent neural network via associating source words with each other, this would also help retain their relationships. Then the source representations and all the relations are fed into the attention component together while decoding, with the main encoder-decoder framework unchanged. Experiments on several datasets show that our method can improve the translation performance significantly over the conventional encoder-decoder model and even outperform the approach involving supervised syntactic knowledge.

1 Introduction

In recent years, Neural Machine Translation (NMT) (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015) has achieved great success in some language pairs, rivalling the state-of-the-art Statistical Machine Translation (SMT). The Recurrent Neural Network (RNN) encoder-decoder architecture is widely used framework for NMT, the principle behind which is that: encoding the meaning of the input bidirectionally into a concept space via RNNs and decoding into target words with RNNs based on this encoding (Sutskever et al., 2014; Bahdanau et al., 2015). This means that encoding principle leads to a deeper understanding and learning of the translation rules, and hence better translation than conventional SMT that considers only surface forms, e.g., words and phrases.

The RNNs with gating, such as Gated Recurrent Unit (GRU) (Cho et al., 2014) or Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), are designed to memorize useful history information and meanwhile forget irrelative information. Together with attention technique which makes the decoding process only focus on the most related source words, the RNN encoder-decoder framework is expected to be able to handle long sequences and consider the globally related information. However, the practical situation is that RNNs tend to forget old history information, especially the far older one. Sometimes the older information is indispensable for generating proper translation, e.g., for the source sentence "take the heavy box away", when translating "away", "take" should be considered together. In addition, it has been proven that using phrases rather than words in SMT (Koehn et al., 2003) brings performance improvement, while in NMT the attention is only modeled in the unit of words. In the same sense, improvement is expected if attention is operated on more words rather than one.

Moreover, NMT produces the representation for the source by running through the source words sequentially with a bidirectional RNN (Schuster and Paliwal, 1997), so it only employs word order information and ignores the relation between words. Although some researchers have demonstrated that

*Corresponding author.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1292–1303
Santa Fe, New Mexico, USA, August 20-26, 2018.

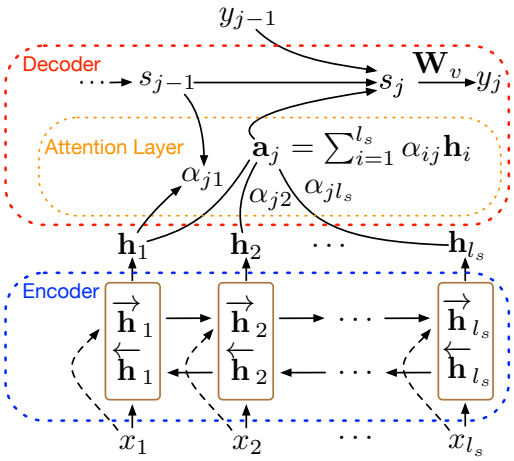


Figure 1: The architecture of attention-based NMT

NMT is able to capture certain syntactic phenomena (e.g. subject-verb agreement) without external syntactic information (Linzen et al., 2016; Shi et al., 2016), there are some other works which has shown their superior performance by modeling word relationship explicitly. However, these works usually need to introduce external syntactic knowledge or connect words according to their relations in the syntactic structure (Sennrich et al., 2016; Bastings et al., 2017; Aharoni and Goldberg, 2017; Li et al., 2017).

In this paper, we present a method to refine the NMT based on the above two points. The main idea is to learn relationship between the source word pairs. Corresponding to the first point, our method employs Convolutional Neural Networks (CNNs) to collect local information around one word and relates each word with its neighbors, which ensures the subsequent operations are performed in the unit of multiple words. As for the second point, Relation Network (RN) (Santoro et al., 2017) is introduced to establish pairwise relationship between words, meanwhile, there's no need to attain external input of syntactic knowledge. In this way, our model can memorize all words ahead and behind via additional connection between words no matter how distant they are. In the RNs, the representations of the source words produced by RNNs are taken as objects and the relationships between them are reasoned.

Specifically, our method introduces a RN component between the encoder and the attention layer in the RNN encoder-decoder framework (Sutskever et al., 2014; Bahdanau et al., 2015). The RN component is composed of three layers: first, the CNN layer slides window along the output of the encoder to capture information among multiple words around one word, then the graph propagation layer constructs a fully connected graph with the information of one window as one node and transfers messages along the edges, so that each node can collect the information from all other nodes, and last the multi-layer perceptron layer transforms the information of each node to the form which is suitable for the attention component to use. We performed experiments on several datasets and got significant improvements over vanilla NMT and SMT systems. Besides, our model significantly outperforms two other models, which introduced latent variables to capture the implicit semantics and employed explicitly external syntactic knowledge respectively.

2 Background

As the main idea of our method is to introduce relation networks into the attention-based NMT (Bahdanau et al., 2015) to learn word relationship and keep all source words in memory, in this section we will briefly describe the baseline model – the attention-based NMT first and the technique used in this paper – relation networks.

2.1 Attention-based NMT

The attention-based NMT follows the encoder-decoder framework, with an additional attention module. It works on the assumption that the source sentence and the target translation share a common continuous

space. It first encodes the source sentence into a continuous space and then performs decoding based on this space, meanwhile, employing attention to indicate the relevance of each source word to the current translation. Figure 1 shows the architecture of the attention-based NMT (Bahdanau et al., 2015), which is composed of three components: the encoder, the attention layer and the decoder.

The Encoder The encoder uses a pair of GRUs to run through source words bidirectionally to get two sequences of hidden states, which are concatenated to produce corresponding hidden state for the i-th source word

$$\overrightarrow{\mathbf{h}}_i = \overrightarrow{\mathbf{GRU}}\left(x_i, \overrightarrow{\mathbf{h}}_{i-1}\right); \qquad \overleftarrow{\mathbf{h}}_i = \overleftarrow{\mathbf{GRU}}\left(x_i, \overleftarrow{\mathbf{h}}_{i+1}\right); \qquad \mathbf{h}_i = \left[\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i\right] \tag{1}$$

The Attention Layer The attention layer aims to extract the source information (called attention) which is highly related to the generation of the current target word. To get the attention of the j-th decoding step, the correlation degree between current target word y_j and $\mathbf{h}_i$ is first evaluated as

$$e_{ij} = \mathbf{v}_a^T \tanh(\mathbf{W}_a \boldsymbol{s}_{j-1} + \mathbf{U}_a \mathbf{h}_i) \tag{2}$$

Then, for the j-th decoding step, the correlation degree is normalized over the whole source sequence, all source hidden states are added weightedly according to the normalized correlation degree to obtain the attention $\mathbf{a}_j$

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{i'=1}^{l_s} \exp\left(e_{i'j}\right)}; \qquad \mathbf{a}_j = \sum\nolimits_{i=1}^{l_s} \alpha_{ij} \mathbf{h}_i \tag{3}$$

The Decoder The decoder first employs a variant of GRU to roll the target information according to previous target word y_{j-1}, previous hidden state $\boldsymbol{s}_{j-1}$ and the attention $\mathbf{a}_j$. The details are described in Bahdanau et al. (2015). The current target hidden state $\boldsymbol{s}_j$ is calculated by

$$\boldsymbol{s}_j = g(y_{j-1}, \boldsymbol{s}_{j-1}, \mathbf{a}_j) \tag{4}$$

After that, the decoder gives a probability distribution over all the words in the target vocabulary and selects the target word with the highest probability as the output of the current step

$$p(y_j | \mathbf{y}_{<j}, \mathbf{x}) \propto \exp(f(\boldsymbol{s}_j, y_{j-1}, \mathbf{a}_j) \cdot \mathbf{W}_{\boldsymbol{v}}) \tag{5}$$

where f stands for a linear transformation and $\mathbf{W}_{\boldsymbol{v}}$ is a weight matrix.

2.2 Relation Networks

A relation network (RN) is a neural network with a structure integrated for relational reasoning. The RN is designed to constrain the functional form of a neural network so that it can capture the core common properties of relational reasoning. Hence its capability of computing relations is inherent without needing to be learned specially.

Formally, given a set of input "objects" denoted as $\mathrm{O} = \{o_1, o_2, \cdots, o_n\}$, RN can be formed as a composition function of objects (Santoro et al., 2017), represented as

$$\mathrm{RN}(\mathrm{O}) = f_\phi\left(\sum\nolimits_{i,j} g_\theta\left(o_i, o_j\right)\right) \tag{6}$$

where o_i is the i-th object, and f_ϕ and g_θ are functions used to calculate relations. Multi-layer perceptrons are often used for f_ϕ and g_θ, as their parameters are learnable synaptic weights, making RNs end-to-end differentiable. Here the role of g_θ is to infer how two objects are related, or whether they are related, and hence the output of g_θ can be treated as "relations".

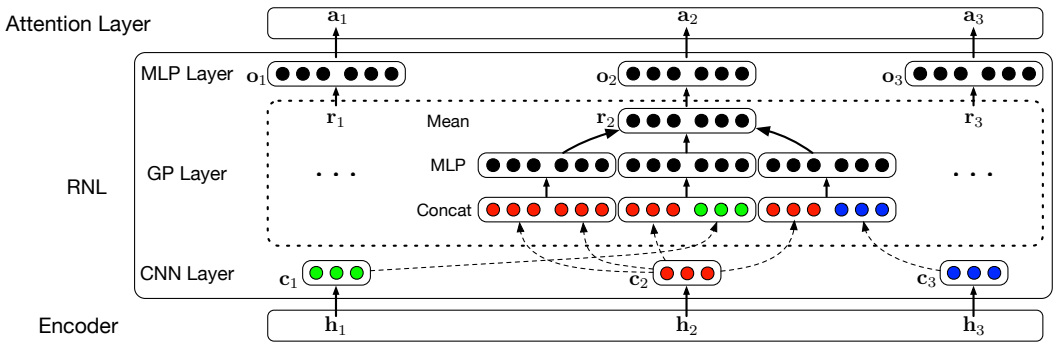


Figure 2: NMT with one RNL. Residual connection is not embodied here. The kernel width of the CNN layer is 3. We take the second word colored in red as an example to show the operations in the RNL, where three colors of green, red and blue indicate the information from the CNN layer of the first, second and third word, respectively.

3 NMT with Relation Networks

In this paper, we introduce a Relation Network Layer (denoted as RNL) on the basis of the attention-based NMT (Bahdanau et al., 2015) and frame it between the encoder and the attention layer. The RNL first employs CNNs to collect information in the unit of multi-words rather than one single word, then takes the outputs of CNNs as objects and makes them fully connected to build a graph propagation layer and associate with each other, finally transforms the acquired representations with word relations via MLP into the form suitable for the attention layer to use. Next, the outputs of the RNL are directly fed into the attention layer, so the RNL can still fit the encoder-decoder framework well. The architecture of our RNL is shown in Figure 2. Briefly, the RNL is composed of three components: the CNN layer, the Graph Propagation (GP) layer and the Multi-layer Perceptron (MLP) layer.

The CNN Layer CNNs are used to collect local information around one word. In this way, not only the information of a single word but their neighbors are considered. The number of neighbors to be considered depends on the kernel width k but can also vary by stacking several convolution layers, e.g., stacking 2 convolution layers with the kernel width $k = 3$ can collect information from 5 words at the same time.

In the CNN layer, the input is the hidden states produced by the bidirectional GRUs (Bi-GRUs), denoted as $\mathbf{h} = \{\mathbf{h}_1, ..., \mathbf{h}_i, ..., \mathbf{h}_{l_s}\}$, so each source word is represented by its hidden state. A filter is applied to convolute over a window of k words to get the convolutional representation. Given the i-th source word and its hidden state $\mathbf{h}_i \in \mathbb{R}^d$, the hidden states covered by the window with the width of k are concatenated and then are fed to the filter where we denote the concatenated vector as $\mathbf{h}_i^k = \left[\mathbf{h}_{i-\lfloor(k-1)/2\rfloor}; \cdots; \mathbf{h}_i; \cdots; \mathbf{h}_{i+\lfloor(k-1)/2\rfloor}\right]$. For the first and last $\lfloor(k-1)/2\rfloor$ words of a sentence, the hidden state $\mathbf{h}_i$ with $i < 1$ or $i > l_s$ are set to zeros (padding). Then the filtering process mentioned above can be formed as

$$\mathbf{c}_i = f\left(\mathbf{W}_{cnn}\mathbf{h}_i^k + \mathbf{b}_{cnn}\right) \tag{7}$$

$\mathbf{W}_{cnn} \in \mathbb{R}^{k\times d}$ is the convolution weights and $\mathbf{b}_{cnn}$ is the bias, where the two together define a linear operation. f is the leaky RELU with the coefficient 0.1 to control the angle of the negative slope. In the RNL, leaky RELU is used as all the nonlinear activation functions. The output of the CNN layer is $\mathbf{c} = \{\mathbf{c}_1, ..., \mathbf{c}_i, ..., \mathbf{c}_{l_s}\}$.

The GP Layer The GP layer is used to learn the relationships between source words. It adopts the outputs of the CNN layer $\mathbf{c} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_{l_s}\}$ as input and formulates the relationships between them into a graph. Here $\mathbf{c}_i$ can be thought as the object mentioned in Section 2.2. In this graph, each input $\mathbf{c}_i$ is taken as a node and has edges connected to all other nodes. Then information flows along the edges and each node receives messages from all its direct neighbors. We call this process graph propagation.

After graph propagation process, another sequence of vectors $\{\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_{l_s}\}$ is produced. The generation of $\mathbf{r}_i$ can be decomposed into three steps:

- Each input vector $\mathbf{c}_i$ in $\mathbf{c}$ is concatenated with all vectors in $\mathbf{c}$ (including itself) to get a set of vectors $\mathbf{C}_i = \{\mathbf{c}_{i1}, \cdots \mathbf{c}_{ij}, \cdots, \mathbf{c}_{il_s}\}$ where $\mathbf{c}_{ij} = [\mathbf{c}_i; \mathbf{c}_j]$.
- Each $\mathbf{c}_{ij}$ is converted into vector $\mathbf{r}_{ij}$ by a 4-hidden-layers MLP. The conversion with 1-hidden-layer MLP can be represented as
$$\mathbf{r}_{ij} = f\left(\mathbf{W}_{gp}\mathbf{c}_{ij} + \mathbf{b}_{gp}\right) \tag{8}$$
- Average over all the outputs above to get the final representation for the i-th source word
$$\mathbf{r}_i = \frac{1}{l_s}\sum\nolimits_{j=1}^{l_s} \mathbf{r}_{ij} \tag{9}$$

The MLP Layer There are several nonlinear transformations which map the inputs into different vector spaces in the GP layer. In order to reduce computation complexity, the output features size of the nonlinear transformations is set to small. Hence we use another MLP layer to map the feature back into the original space, usually the same as that of $\mathbf{h}_i$ to have more powerful representation. The final state $\mathbf{o}_i$ for the i-th source word after the entire RN layer can be got by another 2-hidden-layers MLP, 1-hidden-layer MLP can be written as

$$\mathbf{o}_i = f(\mathbf{W}_{mlp}\mathbf{r}_i + \mathbf{b}_{mlp}) \tag{10}$$

Residual Stacking technique is used in our method. Concretely, we stack multiple layers inside the encoder and meanwhile apply residual connection for two adjacent layers. Assume h^l_{in} and h^l_{out} are the input and the output of the l-th layer, respectively, then residual connection is conducted to get the final output of the l-th layer in the following two steps. First, the input and the output of the l-th layer are added together:

$$\mathrm{h}^l = \mathrm{h}^l_{in} + \mathrm{h}^l_{out} \tag{11}$$

Next, dense concatenation (Huang et al., 2017) is employed to receives features from all previous layers and the final output of the l-th layer is produced by

$$\mathrm{h}^l_{dc} = \mathbf{W}_{dc}\left[\mathrm{h}^1; \mathrm{h}^2; \cdots; \mathrm{h}^l\right] + \mathbf{b}_{dc} \tag{12}$$

where weight matrix $\mathbf{W}_{dc}$ and bias $\mathbf{b}_{dc}$ are adjusted to map the dense-concatenated vectors into the same feature space as the input. Then h^l_{dc} is fed to the next layer which means $\mathrm{h}^{l+1}_{in} = \mathrm{h}^l_{dc}$.

4 Related Work

Many researchers have worked on learning the relationships of the source words to improve translation performance. One line is to refine source presentations by adding relationships between source words or between source and target words, with the main architecture remaining the RNN encoder-decoder framework. Sennrich et al. (2016) enriched source representations with POS tags, dependency labels and other linguistic features. Bastings et al. (2017) employed graph convolutional networks to model relations of words in dependency trees for the source embeddings to include these relations. These two models both require extra supervised syntax input while our method does not need external knowledge and learn the relationship by its own.

Another line is to change the structure of the neural network. Gehring et al. (2017a) and Gehring et al. (2017b) proposed to substitute the conventional RNN encoder with the CNN encoder in order to train faster. They employed stacked CNNs to capture relationships between source words which can be calculated simultaneously, not like RNNs, the computation of which is constrained by temporal dependencies. The attention scores are also computed based on the output of the CNNs and the decoder is still the RNN-based decoder. Vaswani et al. (2017) is another work to eschew the recurrence. It instead

relied entirely on the attention mechanism to draw the global dependencies between input and output. Su et al. (2018) introduced latent random variables into the decoder of NMT and generated these variables recurrently to capture the global semantic contexts and model strong and complex dependencies among target words at different timesteps.

Our method still follows the RNN encoder-decoder framework, giving full play to the advantages of RNNs, which transfers information through words bidirectionally. In addition, we also employs RNs in our method to connect the source words explicitly, further captures relationships between source words without any external knowledge injection, which enables the model to learn the relationships itself and facilitates easy application.

5 Experiments

In the experiment section, we first compare our system with two baseline systems on a Chinese-English (Zh-En) dataset and the WMT17 English-German (En-De) dataset, then compare our method with a related approach on the WMT16 En-De dataset. Finally, we give some analyses about our method in different aspects.

5.1 Data Preparation

We performed experiments on three datasets:

NIST The training data consisted of 1.25M Zh-En parallel sentence pairs with 25M Chinese tokens and 27M English tokens[1]. We used NIST 2002 test dataset (878 sentences) as the validation set, and another four NIST test datasets as the test datasets: NIST 2003 (MT03), NIST 2004 (MT04), NIST 2005 (MT05) and NIST 2006 (MT06), which contain 919, 1788, 1082 and 1357 sentences respectively.

WMT17 The training data was composed of 5.6M En-De preprocessed parallel sentence pairs [2] with 141M English tokens and 194M German tokens. The test dataset of newstest2014 (3003 sentences) was used as the validation set and the following test datasets were used as the test datasets: newstest2015 (2169 sentences), newstest2016 (2999 sentences) and newstest2017 (3004 sentences). Besides, $8k$ merging operations were performed to learn byte-pair encodings (BPE) (Sennrich et al., 2016) on the target side of the parallel training data.

WMT16 We conducted experiments on WMT16 dataset, the same dataset as the work of Bastings et al. (2017) for comparison. We kept the same settings as those in Bastings et al. (2017): The original dataset consists of 4500966 sentence pairs, with 4173550 left after filtering pairs which contains more than 50 tokens on either side after tokenization. newstest2015 and newstest2016 were used as the validation set and test dataset, respectively. $16k$ BPE merging operations were conducted on the target side of the bilingual training data.

For WMT16 dataset, case-sensitive 4-gram BLEU score (Papineni et al., 2002) was reported by using the *multi-bleu.pl* script. The results on the other two datasets were evaluated with case-insensitive 4-gram BLEU score.

5.2 Systems

Results of five systems on different datasets were reported:

RNNsearch We implemented the attention-based NMT of Bahdanau et al. (2015) by PyTorch framework[3] with the following settings: the length of the sentences on both sides was limited up to 50 tokens with 30K vocabulary, and the source and target word embedding sizes were both set to 512, the size of all hidden units in both encoder and decoder RNNs was also set to 512, and all parameters were initialized by using uniform distribution over $[-0.1, 0.1]$. The mini-batch stochastic gradient descent (SGD) algorithm was employed to train the model with batch size of 80. In addition, the learning rate was adjusted by Adadelta optimizer (Zeiler, 2012) with $\rho = 0.95$ and $\epsilon = 1e$-6. Dropout was applied on the output layer with dropout rate of 0.5. The beam size was set to 10.

[1]We chose LDC2002E18, LDC2003E07, LDC2003E14, Hansard's portion of LDC2004T07, LDC2004T08 and LDC2005T06 from the LDC corpora. There were 1.11M sentence pairs left after filtering.

[2]`http://data.statmt.org/wmt17/translation-task/preprocessed`

[3]`http://pytorch.org`

Systems	MT03	MT04	MT05	MT06	Average
RNNsearch	33.70	36.15	31.81	32.71	33.59
RNNsearch*	37.93	40.53	36.65	35.80	37.73
VRNMT	38.08	41.07	36.82	36.72	38.17
RNMT	**39.24***	**42.01***	**37.79***	**37.81***	**39.21**

Table 1: Performance comparison on NIST datasets. $*$ is used to indicate the improvement over RNNsearch* is statistically significant (Collins et al., 2005) ($p < 0.01$).

Systems	test15	test16	test17	Avg.
RNNsearch	17.3	20.9	16.6	18.3
RNNsearch*	21.4	25.6	20.1	22.4
RNMT	**22.7***	**27.8***	**21.8***	**24.1**

Table 2: Performance comparison on WMT17 En-De datasets.

Systems	test16
BiRNN+GCN	23.9
RNMT	**25.4**

Table 3: Performance comparison with the related work on the WMT16 En-De dataset.

RNNsearch* This system is an improved version of RNNsearch where the decoder employs a conditional GRU layer with attention module, consisting of two GRUs and an attention module for each step[4]. Specifically, Equation 4 is substituted with the following two equations:

$$\tilde{\boldsymbol{s}}_j = \mathbf{GRU}_1(y_{j-1}, \boldsymbol{s}_{j-1}); \qquad \boldsymbol{s}_j = \mathbf{GRU}_2(\mathbf{a}_j, \tilde{\boldsymbol{s}}_j) \tag{13}$$

Besides, for the calculation of attention in Equation 2, $\boldsymbol{s}_{j-1}$ is replaced with $\tilde{\boldsymbol{s}}_{j-1}$. The other components of the system keep the same as RNNsearch. We used the same settings for RNNsearch and RNNsearch*.

VRNMT A novel Variational Recurrent NMT (VRNMT) model, proposed by Su et al. (2018), captures more semantic context and complex dependencies among target words by generating latent random variables recurrently in the NMT decoder.

BiRNN+GCN This is the model presented by Bastings et al. (2017). They incorporated dependency syntactic structure into the bidirectional RNN (BiRNN) encoder of NMT and modeled the relation among the source words by using graph convolutional networks (GCNs).

RNMT Our system was implemented by embedding the RNLs into the Bi-GRUs of RNNsearch*. The overall structure used alternatively stacked GRUs and RNLs, in which the two GRU layers are in opposite direction. Inside the RNL, the GP layer employed a 4-hidden-layers MLP (shown in Equation 8) and the MLP layer contained 2 hidden layers (as in Equation 10). For the Zh-En translation task, two convolution layers with kernel width of 1 and 3 were stacked, the output channel sizes of CNN were 128 and 256 respectively, followed by batch normalization (BN) (Ioffe and Szegedy, 2015) with learnable parameters, and MLP contained 256 units. For the En-De translation task, only one convolution layer was used with kernel width of 3, the output channel size was 96, 128 was adopted as the hidden size of MLP. All of the other settings were the same with those of RNNsearch*.

5.3 Performance Comparison

We compared our system RNMT with the two baseline systems RNNsearch and RNNsearch* both on the NIST Zh-En and the WMT17 En-De translation tasks. As RNMT was implemented on the basis of RNNsearch*, in the strict sense, RNNsearch* is the baseline. From the results shown in Table 1, we can see that RNMT significantly improves translation quality on all test datasets and outperforms RNNsearch* by 1.48 BLEU points averagely on the Zh-En dataset. Besides, comparison between our model to VRNMT shows that proposed simple model stably produces better performance on all test datasets and outperforms VRNMT 1.04 BLEU score on average.

[4] https://github.com/nyu-dl/dl4mt-tutorial/blob/master/docs/cgru.pdf

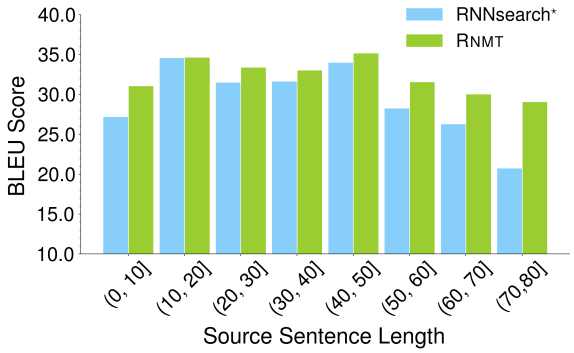


Figure 3: Results on different bins contain sentences of length within corresponding spans.

On the WMT17 En-De dataset, as shown in Table 2, RNMT shows superiority on three test datasets stably, and averagely achieves the gains of 1.7 BLEU points over RNNsearch*, with only 4.1M parameters more. Given the above results, we can conclude that RN can indeed learn the relationships between the source words and these relationships are useful and bring improvement on the translation performance.

We also compared our method with the work of Bastings et al. (2017) which requires the injection of external syntactic knowledge, to see whether the relationships produced by RNs can lead to better translation than the syntax from supervised learning. The results in Table 3 show that our system can achieve an improvement of 1.5 BLEU scores. We believe that the relationships of the source words derived from RNs do not necessarily conform to human cognition, but it can be simultaneously tailored with the other parts of the translation system. In this way, RNs can generate the relationships more suitable for the NMT.

5.4 Impact of Input Length

One motivation of adding RNs is that RNNs tend to forget the distant history which RNs memorize it by explicitly introducing relations between pairs of words. Therefore, we assume that our method suppose to bring greater improvement on relative long sentences, which contains more distant history information than shorter ones that usually forgotten by RNNs. Based on this sense, we split the source sentences in the MT03 test dataset into different bins according to their length and evaluated BLEU scores of the translations from RNNsearch* and RNMT on the different bins, respectively.

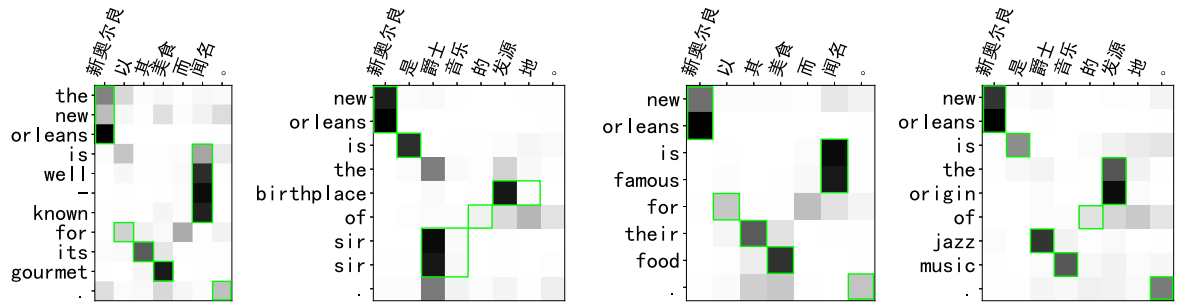
The results are shown in Figure 3. In the bins holding sentences no longer than 50, the BLEU scores of the two systems are close to each other. When the sentence length surpasses 50, RNMT shows its superiority over RNNsearch*. As the sentence length grows, the difference becomes increasingly large. This verifies the deduce that our method can not only memorize history information but capture the relationship between words, both of which are beneficial to translate long sentences.

5.5 Word Alignment

Systems	BLEU	AER
RNNsearch*	22.40	46.76
RNMT	**24.12**	**45.66**

Table 4: Comparison of alignment quality on NIST Zh-En translation task.

In this section, we will verify the translation performance of our model from another perspective. Intuitively, the better translation should have better alignment to the source sentence, so we evaluated the quality of the alignments derived from the attention module of the NMT using Alignment Error Rate (AER) (Och, 2003). We did this experiment on the artificially aligned dataset from Liu and Sun (2015) which contains 900 Zh-En sentence pairs. The alignments were got in this way for both RNNsearch*



(a) RNNsearch* (b) RNMT

Figure 4: Word alignment comparison. The green boxes show the manual golden alignments.

system and our system. When one target word was generated, we retained the alignment link with the highest probability α_{ij} in Equation 3.

The comparison results are shown in Table 4. It illustrates that our system RNMT can produce better translations than the baseline RNNsearch*, a difference of 1.72 BLEU points. Besides, the AER score is 1.1 points lower than the baseline model. Note that the smaller the AER score, the better the alignment quality.

Along with the translation results, we also produce the word alignment matrix based on each target word's attention probability distribution over the whole source sentence. Two source sentences are randomly sampled from websites, both comparisons between baseline alignment and improved alignment generated by RNNsearch* and RNMT are shown in Figure 4.

For the first example, from the view of source side, it is obviously unreasonable that the Chinese word *yi* is contributed to generate three discontinuous English words *the*, *is* and *for*, grammatical knowledge show that the word *yi* should be only aligned to the English word *for*, just like the result of our model. Besides, on the target's ground, if one Chinese word is translated into an English phrase, all words in the phrase should be aligned to the Chinese word, RNNsearch* model improperly aligns *new* and *is* to some other irrelevant words besides the correct one. When generating word *is*, almost the whole source sentence should be considered, our model gets more centralized alignment for it.

In the second case, unlike the baseline model, our model produces correct translation *jazz music* for *jueshi yinyue* and alignment. *the* together with *origin* is aligned to the source word *fayuan*, while RNNsearch* mistakenly aligns *the* to two source words almost with equal probability.

5.6 Translation Examples

As shown in Table 5, we give two example translations generated from baseline model and proposed model. Comparing the translation results between two systems, we can observe that RNNsearch* often miss some information of the source sentence, especially for the long sentence. Both of the sentences are complex sentences with long dependent adversative relation, for the first example, the baseline model forgets the information of the long distance clause about *women jinnian yizhi* $\cdots$ *toumingdu* and ignores to translate the second clause. It similarly happens that, when producing the target text for the second sample, RNNsearch* loses the information after *chengnuo dongaohui* and fails to capture the latter clause with adversative relation. In addition, another phenomenon observed is that the longer the source sentence is, it is easier to ignore important information for RNNsearch*. However, as can be seen from the boldfaced sections marked in results generated with RNMT, proposed model with CNN could captures more source information successfully.

Specifically, RNNs are skilled in modeling the order information of a sequence, while CNNs mainly focus on local features around some specific word. Both of them are weak to capture the long-distance dependency information, However, facts prove that proposed relation layer succeeds in alleviating the deficiencies of the two by integrating CNNs with bidirectional RNNs subtly.

Source	我们 近年 一直 倡导 “ 诚信 ” , 要 “ 打造 阳光 政府 ” , 要 尊重 公众 的 “ 知情权 ” , 要 提高 行政 “ 透明 度 ” , 然而 , 事实 距离 理想 还 有 很 大 差距 。
Reference	in recent years, we have been advocating ”integrity” and we want to ”forge a government-in-sunshine”, improve the ”transparency” of government administration, and respect the public's ”right to know”. however, the reality is still very far from ideal.
RNNsearch*	in recent years , we have always advocated ” honesty ” and ” build a sunshine government , ” and we must respect the public 's ” right to understand ” and to enhance the ” transparency ” of the ” transparency ” of the public .
RNMT	in recent years , we have advocated ” integrity ” and ” build up the sun . ” we should respect public ” right to know ” and improve the **” transparency ” of the public . however , there is still a big gap between reality and ideals .**
Source	经过 国际 奥委会 的 不懈 努力 , 意大利 方面 在 冬奥会 开幕 前 四 天 作出 让步 , 承诺 冬奥会 期间 警 方 不 会 进入 奥运村 搜查 运动员 驻地 , 但是 , 药检 呈 阳性 的 运动员 仍将 接受 意大利 检察 机关 的 调查 。
Reference	through the untiring efforts of the ioc, the italian side made concession four days before the winter olympics opened, promising that police would not enter the olympic village to raid athletes' quarters during the winter olympics, but athletes tested positive for drugs are still subject to investigations of italian prosecutors.
RNNsearch*	through the unremitting efforts of the ioc , the italian side made a concession four days prior to the opening of the international olympic committee .
RNMT	with the unremitting efforts of the international olympic committee , the italian side made a concession in four days before the opening of the ⟨UNK⟩ **and promised that the police would not be able to search for the athlete 's place during the opening period .**

Table 5: Translation examples.

6 Conclusion

As RNNs are not good at remembering the old history and cannot consider word relationship either, sometimes conventional NMT cannot get enough source information and hence emphasizes too much on the fluency of the target. As a result, it suffers from meaning-drift and generates “inaccurate” translation. Even so, NMT can still benefit from the recurrence of RNNs. In this paper, we propose to incorporate RNLs into the attentional NMT. The RNs employs CNNs to collect information around one word and explicitly connect each word with all the other words. In this way, it provides the opportunities for NMT to capture relationship between source words and hence leads to a better source representation. Our method can get better translation on the NIST Zh-En dataset and the WMT En-De dataset and can even outperform the system with supervised syntactic knowledge.

Acknowledgements

We highly appreciate the anonymous reviewers for their precious comments. This work was supported in part by National Natural Science Foundation of China (Nos. 61472428 and 61662077).

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 132–140, Vancouver, Canada, July. Association for Computational Linguistics.

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

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Simaan. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, Copenhagen, Denmark, September. Association for Computational Linguistics.

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

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 531–540, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017a. A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 123–135, Vancouver, Canada, July. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017b. Convolutional sequence to sequence learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252, International Convention Centre, Sydney, Australia, 06–11 Aug. PMLR.

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

G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger. 2017. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, July.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul. PMLR.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA, October. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 688–697, Vancouver, Canada, July. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Tim Lillicrap. 2017. A simple neural network module for relational reasoning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4974–4983. Curran Associates, Inc.

M. Schuster and K. K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, Nov.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91, Berlin, Germany, August. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August. Association for Computational Linguistics.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas, November. Association for Computational Linguistics.

Jinsong Su, Shan Wu, Deyi Xiong, Yaojie Lu, Xianpei Han, and Biao Zhang. 2018. Variational recurrent neural machine translation. *arXiv preprint arXiv:1801.05119*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6000–6010. Curran Associates, Inc.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July. Association for Computational Linguistics.

Yang Liu and Maosong Sun. 2015. Contrastive unsupervised word alignment with non-local features. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2295–2301. AAAI Press.

A Survey of Domain Adaptation for Neural Machine Translation

Chenhui Chu
Institute for Datability Science
Osaka University
chu@ids.osaka-u.ac.jp

Rui Wang
National Institute of Information
and Communications Technology
wangrui@nict.go.jp

Abstract

Neural machine translation (NMT) is a deep learning based approach for machine translation, which yields the state-of-the-art translation performance in scenarios where large-scale parallel corpora are available. Although the high-quality and domain-specific translation is crucial in the real world, domain-specific corpora are usually scarce or nonexistent, and thus vanilla NMT performs poorly in such scenarios. Domain adaptation that leverages both out-of-domain parallel corpora as well as monolingual corpora for in-domain translation, is very important for domain-specific translation. In this paper, we give a comprehensive survey of the state-of-the-art domain adaptation techniques for NMT.

1 Introduction

Neural machine translation (NMT) (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015) allows for end-to-end training of a translation system without the need to deal with word alignments, translation rules and complicated decoding algorithms, which are characteristics of statistical machine translation (SMT) systems (Koehn et al., 2007). NMT yields the state-of-the-art translation performance in resource rich scenarios (Bojar et al., 2017; Nakazawa et al., 2017). However, currently, high quality parallel corpora of sufficient size are only available for a few language pairs such as languages paired with English and several European language pairs. Furthermore, for each language pair the sizes of the domain specific corpora and the number of domains available are limited. As such, for the majority of language pairs and domains, only few or no parallel corpora are available. It has been known that both vanilla SMT and NMT perform poorly for domain specific translation in low resource scenarios (Duh et al., 2013; Sennrich et al., 2013; Zoph et al., 2016; Koehn and Knowles, 2017).

High quality domain specific machine translation (MT) systems are in high demand whereas general purpose MT has limited applications. In addition, general purpose translation systems usually perform poorly and hence it is important to develop translation systems for specific domains (Koehn and Knowles, 2017). Leveraging out-of-domain parallel corpora and in-domain monolingual corpora to improve in-domain translation is known as domain adaptation for MT (Wang et al., 2016; Chu et al., 2018). For example, the Chinese-English patent domain parallel corpus has 1M sentence pairs (Goto et al., 2013), but for the spoken language domain parallel corpus there are only 200k sentences available (Cettolo et al., 2015). MT typically performs poorly in a resource poor or domain mismatching scenario and thus it is important to leverage the spoken language domain data with the patent domain data (Chu et al., 2017). Furthermore, there are monolingual corpora containing millions of sentences for the spoken language domain, which can also be leveraged (Sennrich et al., 2016b).

There are many studies of domain adaptation for SMT, which can be mainly divided into two categories: data centric and model centric. Data centric methods focus on either selecting training data from out-of-domain parallel corpora based a language model (LM) (Moore and Lewis, 2010; Axelrod et al., 2011; Duh et al., 2013; Hoang and Sima'an, 2014; Durrani et al., 2015; Chen et al., 2016) or generating pseudo parallel data (Utiyama and Isahara, 2003; Wang et al., 2014; Chu, 2015; Wang et al.,

Proceedings of the 27th International Conference on Computational Linguistics, pages 1304–1319
Santa Fe, New Mexico, USA, August 20-26, 2018.

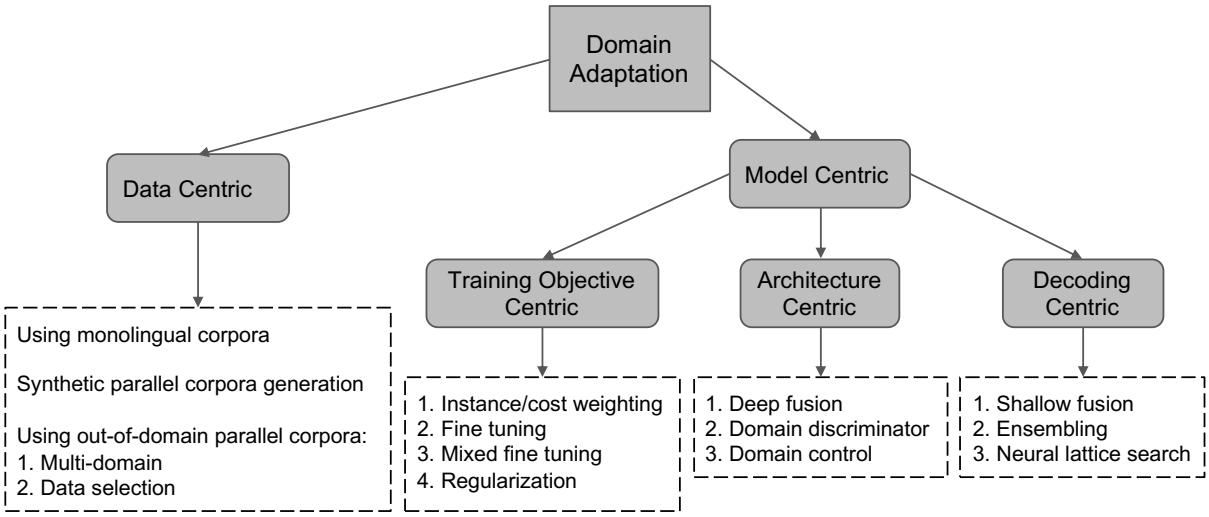


Figure 1: Overview of domain adaptation for NMT.

2016; Marie and Fujita, 2017). Model centric methods interpolate in-domain and out-of-domain models in either a model level (Sennrich et al., 2013; Durrani et al., 2015; Imamura and Sumita, 2016) or an instance level (Matsoukas et al., 2009; Foster et al., 2010; Shah et al., 2010; Rousseau et al., 2011; Zhou et al., 2015). However, due to the different characteristics of SMT and NMT, many methods developed for SMT cannot be applied to NMT directly.

Domain adaptation for NMT is rather new and has attracted plenty of attention in the research community. In the past two years, NMT has become the most popular MT approach and many domain adaptation techniques have been proposed and evaluated for NMT. These studies either borrow ideas from previous SMT studies and apply these ideas for NMT, or develop unique methods for NMT. Despite the rapid development in domain adaptation for NMT, there is no single compilation that summarizes and categorizes all approaches. As such a study will greatly benefit the community, we present in this paper a survey of all prominent domain adaptation techniques for NMT. There are survey papers for NMT (Neubig, 2017; Koehn, 2017); however, they focus on general NMT and more diverse topics. Domain adaptation surveys have been done in the perspective of computer vision (Csurka, 2017) and machine learning (Pan and Yang, 2010; Weiss et al., 2016). However, such survey has not been done for NMT. To the best of our knowledge, this is the first comprehensive survey of domain adaptation for NMT.

In this paper, similar to SMT, we categorize domain adaptation for NMT into two main categories: data centric and model centric. The data centric category focuses on the data being used rather than specialized models for domain adaptation. The data used can be either in-domain monolingual corpora (Zhang and Zong, 2016b; Cheng et al., 2016; Currey et al., 2017; Domhan and Hieber, 2017), synthetic corpora (Sennrich et al., 2016b; Zhang and Zong, 2016b; Park et al., 2017), or parallel copora (Chu et al., 2017; Sajjad et al., 2017; Britz et al., 2017; Wang et al., 2017a; van der Wees et al., 2017). On the other hand, the model centric category focuses on NMT models that are specialized for domain adaptation, which can be either the training objective (Luong and Manning, 2015; Sennrich et al., 2016b; Servan et al., 2016; Freitag and Al-Onaizan, 2016; Wang et al., 2017b; Chen et al., 2017a; Varga, 2017; Dakwale and Monz, 2017; Chu et al., 2017; Miceli Barone et al., 2017), the NMT architecture (Kobus et al., 2016; Gülçehre et al., 2015; Britz et al., 2017) or the decoding algorithm (Gülçehre et al., 2015; Dakwale and Monz, 2017; Khayrallah et al., 2017). An overview of these two categories is shown in Figure 1. Note that as model centric methods also use either monolingual or parallel corpora, there are overlaps between these two categories.

The remainder of this paper is structured as follows: We first give a brief introduction of NMT, and describe the reason for the difficulty of low resource domains and languages in NMT (Section 2); Next, we briefly review the historical domain adaptation techniques being developed for SMT (Section 3); Under these background knowledge, we then present and compare the domain adaptation methods for

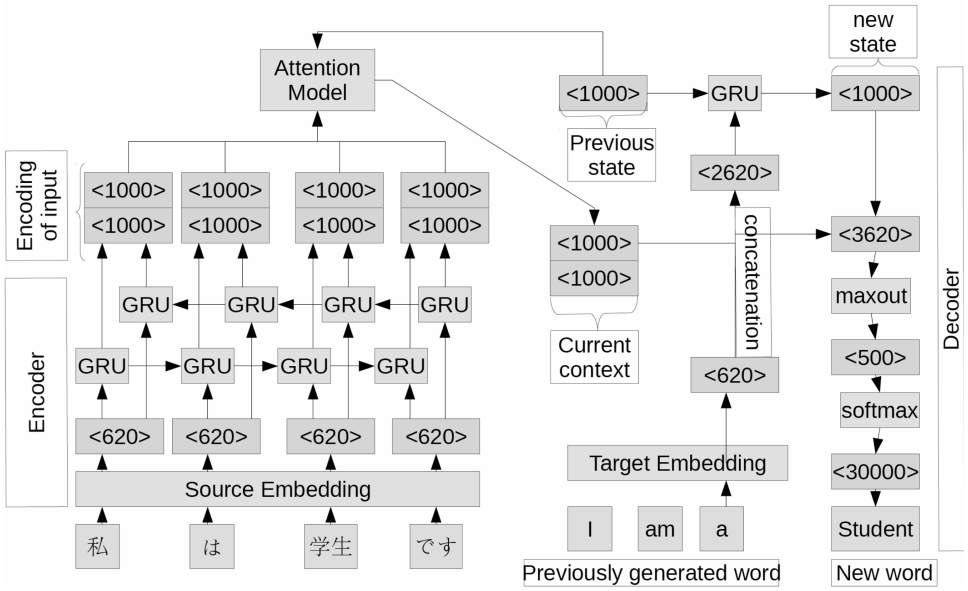


Figure 2: The architecture of the NMT system with attention, as described in (Bahdanau et al., 2015). The notation "<1000>" means a vector of size 1000. The vector sizes shown here are the ones suggested in the original paper.

NMT in detail (Section 4); After that, we introduce domain adaptation for NMT in real word scenarios, which is crucial for the practical use of MT (Section 5); Finally, we give our opinions of future research directions in this field (Section 6) and conclude this paper (Section 7).

2 Neural Machine Translation

NMT is an end-to-end approach for translating from one language to another, which relies on deep learning to train a translation model (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015). The encoder-decoder model with attention (Bahdanau et al., 2015) is the most commonly used NMT architecture. This model is also known as RNNsearch. Figure 2 describes the RNNsearch model (Bahdanau et al., 2015), which takes in an input sentence $\mathbf{x} = \{x_1, ..., x_n\}$ and its translation $\mathbf{y} = \{y_1, ..., y_m\}$. The translation is generated as:

$$p(\mathbf{y}|\mathbf{x};\theta) = \prod_{j=1}^{m} p(y_j|y_{<j}, \mathbf{x};\theta), \tag{1}$$

where θ is a set of parameters, m is the entire number of words in $\mathbf{y}$, y_j is the current predicted word, and $y_{<j}$ are the previously predicted words. Suppose we have a parallel corpus C consisting of a set of parallel sentence pairs $(\mathbf{x}, \mathbf{y})$. The training object is to minimize the cross-entropy loss L w.r.t θ:

$$L_\theta = \sum_{(\mathbf{x},\mathbf{y})\in\mathbf{C}} -\log p(\mathbf{y}|\mathbf{x};\theta). \tag{2}$$

The model consists of three main parts, namely, the encoder, decoder and attention model. The encoder uses an embedding mechanism to convert words into their continuous space representations. These embeddings by themselves do not contain information about relationships between words and their positions in the sentence. Using a recurrent neural network (RNN) layer, gated recurrent unit (GRU) in this case, this can be accomplished. An RNN maintains a hidden state (also called a memory or history), which allows it to generate a continuous space representation for a word given all past words that have been seen. There are two GRU layers which encode forward and backward information. Each word x_i is represented by concatenating the forward hidden state $\overrightarrow{h_i}$ and the backward one $\overleftarrow{h_i}$ as $h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]$. In this way, the source sentence $\mathbf{x} = \{x_1, ..., x_n\}$ can be represented as $\mathbf{h} = \{h_1, ..., h_n\}$. By using both

forward and backward recurrent information, one obtains a continuous space representation for a word given all words before as well as after it.

The decoder is conceptually an RNN language model (RNNLM) with its own embedding mechanism, a GRU layer to remember previously generated words and a softmax layer to predict a target word. The encoder and decoder are coupled by using an attention mechanism, which computes a weighted average of the recurrent representations generated by the encoder thereby acting as a soft alignment mechanism. This weighted averaged vector, also known as the context or attention vector, is fed to the decoder GRU along with the previously predicted word to produce a representation that is passed to the softmax layer to predict the next word. In equation, an RNN hidden state s_j for time j of the decoder is computed by:

$$s_j = f(s_{j-1}, y_{j-1}, c_j), \tag{3}$$

where f is an activation function of GRU, s_{j-1} is the prvious RNN hidden state, y_{j-1} is the previous word, c_j is the context vector. c_j is computed as a weighted sum of the encoder hidden states $\mathbf{h} = \{h_1, ..., h_n\}$, by using alignment weight a_{ji}:

$$c_j = \sum_{j=1}^{n} a_{ji} h_i, \quad a_{ji} = \frac{\exp(e_{ji})}{\sum_{k=1}^{m} \exp(e_{ki})}, \quad e_{ji} = a(s_{j-1}, h_i), \tag{4}$$

where a is an alignment model that scores the match level of the inputs around position i and the output at position j. The softmax layer contains a maxout layer which is a feedforward layer with max pooling. The maxout layer takes the recurrent hidden state generated by the decoder GRU, the previous word and the context vector to compute a final representation, which is fed to a simple softmax layer:

$$P(y_j | y_{<j}, \mathbf{x}) = \text{softmax}(\text{maxout}(s_j, y_{j-1}, c_j)). \tag{5}$$

An abundance of parallel corpora are required to train an NMT system to avoid overfitting, due to the large amounts of parameters in the encoder, decoder, and attention model. This is the main bottleneck of NMT for low resource domains and languages.

3 Domain Adaptation for SMT

In SMT, many domain adaptation methods have been proposed to overcome the problem of the lack of substantial data in specific domains and languages. Most SMT domain adaptation methods can be broken down broadly into two main categories:

3.1 Data Centric

This category focuses on selecting or generating the domain-related data using existing in-domain data.

i) When there are sufficient parallel corpora from other domains, the main idea is to score the out-domain data using models trained from the in-domain and out-of-domain data and select training data from the out-of-domain data using a cut-off threshold on the resulting scores. LMs (Moore and Lewis, 2010; Axelrod et al., 2011; Duh et al., 2013), as well as joint models (Hoang and Sima'an, 2014; Durrani et al., 2015), and more recently convolutional neural network (CNN) models (Chen et al., 2016) can be used to score sentences.

ii) When there are not enough parallel corpora, there are also studies that generate pseudo-parallel sentences using information retrieval (Utiyama and Isahara, 2003), self-enhancing (Lambert et al., 2011) or parallel word embeddings (Marie and Fujita, 2017). Besides sentence generation, there are also studies that generate monolingual n-grams (Wang et al., 2014) and parallel phrase pairs (Chu, 2015; Wang et al., 2016).

Most of the data centric-based methods in SMT can be directly applied to NMT. However, most of these methods adopt the criteria of data selection or generation that are not related to NMT. Therefore, these methods can only achieve modest improvements in NMT (Wang et al., 2017a).

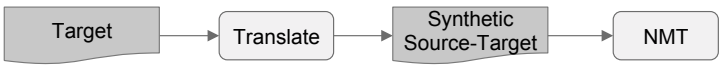


Figure 3: Synthetic data generation for NMT (Sennrich et al., 2016b).

3.2 Model Centric

This category focuses on interpolating the models from different domains.

i) Model level interpolation. Several SMT models, such as LMs, translation models, and reordering models, individually corresponding to each corpus, are trained. These models are then combined to achieve the best performance (Foster and Kuhn, 2007; Bisazza et al., 2011; Niehues and Waibel, 2012; Sennrich et al., 2013; Durrani et al., 2015; Imamura and Sumita, 2016).

ii) Instance level interpolation. Instance weighting has been applied to several natural language processing (NLP) domain adaptation tasks (Jiang and Zhai, 2007), especially SMT (Matsoukas et al., 2009; Foster et al., 2010; Shah et al., 2012; Mansour and Ney, 2012; Zhou et al., 2015). They firstly score each instance/domain by using rules or statistical methods as a weight, and then train SMT models by giving each instance/domain the weight. An alternative way is to weight the corpora by data re-sampling (Shah et al., 2010; Rousseau et al., 2011).

For NMT, several methods have been proposed to interpolate model/data like SMT does. For model-level interpolation, the most related NMT technique is model ensemble (Jean et al., 2015). For instance-level interpolation, the most related method is to assign a weight in NMT objective function (Chen et al., 2017a; Wang et al., 2017b). However, the model structures of SMT and NMT are quite different. SMT is a combination of several independent models; in comparison, NMT is an integral model itself. Therefore, most of these methods cannot be directly applied to NMT.

4 Domain Adaptation for NMT

4.1 Data Centric

4.1.1 Using Monolingual Corpora

Unlike SMT, in-domain monolingual data cannot be used as an LM for conventional NMT directly, and many studies have been conducted for this. Gülçehre et al. (2015) train an RNNLM on monolingual data, and fuse the RNNLM and NMT models. Currey et al. (2017) copy the target monolingual data to the source side and use the copied data for training NMT. Domhan and Hieber (2017) propose using target monolingual data for the decoder with LM and NMT multitask learning. Zhang and Zong (2016b) use source side monolingual data to strengthen the NMT encoder via multitask learning for predicting both translation and reordered source sentences. Cheng et al. (2016) use both source and target monolingual data for NMT through reconstructing the monolingual data by using NMT as an autoencoder.

4.1.2 Synthetic Parallel Corpora Generation

As NMT itself has the ability of learning LMs, target monolingual data also can be used for the NMT system to strengthen the decoder after back translating target sentences to generate a synthetic parallel corpus (Sennrich et al., 2016b). Figure 3 shows the flowchart of this method. It has also been shown that synthetic data generation is very effective for domain adaptation using either the target side monolingual data (Sennrich et al., 2016c), the source side monolingual data (Zhang and Zong, 2016b), or both (Park et al., 2017).

4.1.3 Using Out-of-Domain Parallel Corpora

With both in-domain and out-of-domain parallel corpora, it is ideal to train a mixed domain MT system that can improve in-domain translation while do not decrease the quality of out-of-domain translation. We categorize these efforts as *multi-domain* methods, which have been successfully developed for NMT. In addition, the idea of data selection from SMT also have been developed for NMT.

Multi-Domain The *multi-domain* method in Chu et al. (2017) is originally motivated by Sennrich et al. (2016a), which uses tags to control the politeness of NMT. The overview of this method is shown in

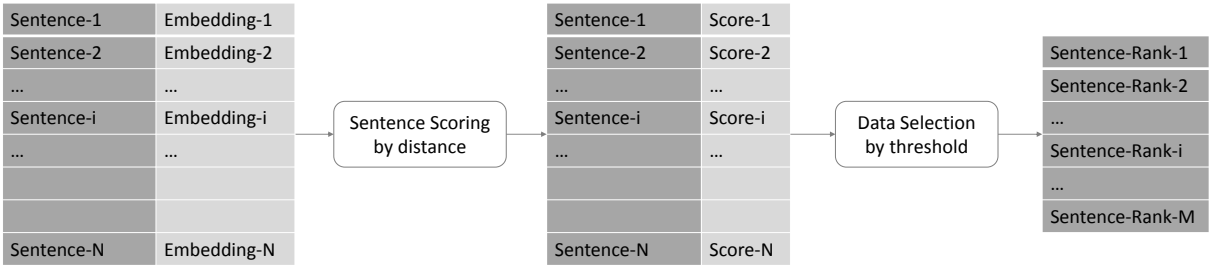


Figure 4: Data selection for NMT (Wang et al., 2017a).

the dotted section in Figure 6. In this method, the corpora of multiple domains are concatenated with two small modifications:

- Appending the domain tag "<2domain>" to the source sentences of the respective corpora. This primes the NMT decoder to generate sentences for the specific domain.
- Oversampling the smaller corpus so that the training procedure pays equal attention to each domain.

Sajjad et al. (2017) further compare different methods for training a multi-domain system. In particular, they compare *concatenation* that simply concatenates the multi-domain corpora, *staking* that iteratively trains the NMT system on each domain corpus, *selection* that selects a set of out-of-domain data which is close to the in-domain data, and *ensemble* that ensembles the multiple NMT models trained independently. They find that fine tuning the concatenation system on in-domain data shows the best performance. Britz et al. (2017) compare the *multi-domain* method with a discriminative method (see Section 4.2.2 for details). They show that the discriminative method performs better than the *multi-domain* method.

Data Selection As mentioned in the SMT section (Section 3.1), the data selection methods in SMT can improve NMT performance modestly, because their criteria of data selection are not very related to NMT (Wang et al., 2017a). To address this problem, Wang et al. (2017a) exploit the internal embedding of the source sentence in NMT, and use the sentence embedding similarity to select the sentences that are close to in-domain data from out-of-domain data (Figure 4). Van der Wees et al. (2017) propose a dynamic data selection method, in which they change the selected subset of training data among different training epochs for NMT. They show that gradually decreasing the training data based on the in-domain similarity gives the best performance.

Although all the data centric methods for NMT are complementary to each other in principle, there are no studies that try to combine these methods, which is considered to be one future direction.

4.2 Model Centric

4.2.1 Training Objective Centric

The methods in this section change the training functions or procedures for obtaining an optimal in-domain training objective.

Instance/Cost Weighting The main challenge for instance weighting in NMT is that NMT is not a linear model or a combination of linear models, which means the instance weight cannot be integrated into NMT directly. There is only one work concerning instance weighting in NMT (Wang et al., 2017b). They set a weight for the objective function, and this weight is learned from the cross-entropy by an in-domain LM and an out-of-domain LM (Axelrod et al., 2011) (Figure 5). Instead of instance weighting, Chen et al. (2017a) modify the NMT cost function with a domain classifier. The output probability of the domain classifier is transferred into the domain weight. This classifier is trained using development data. Recently, Wang et al. (2018) proposed a joint framework of sentence selection and weighting for NMT.

Fine Tuning *Fine tuning* is the conventional way for domain adaptation (Luong and Manning, 2015; Sennrich et al., 2016b; Servan et al., 2016; Freitag and Al-Onaizan, 2016). In this method, an NMT

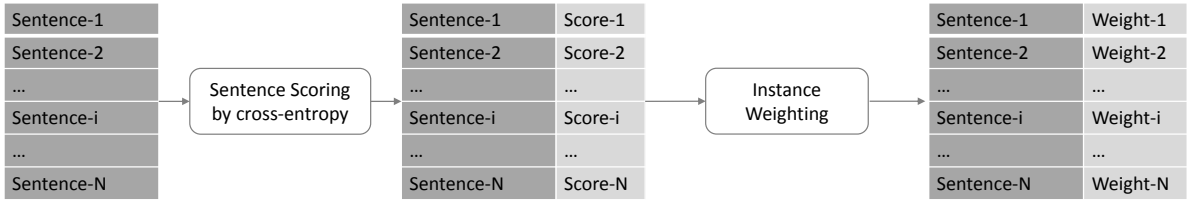


Figure 5: Instance weighting for NMT (Wang et al., 2017b).

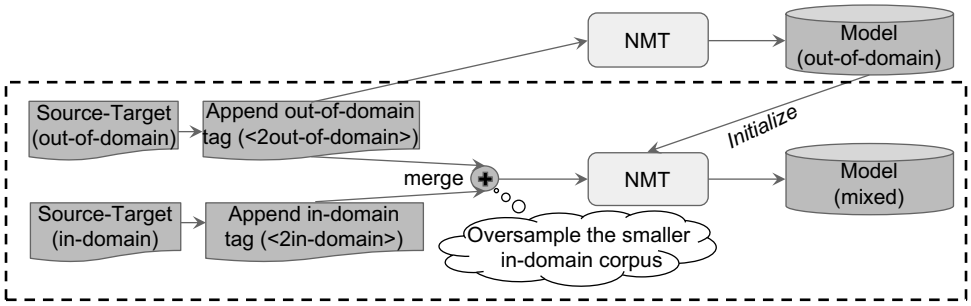


Figure 6: Mixed fine tuning with domain tags for domain adaptation (Chu et al., 2017). The section in the dotted rectangle denotes the *multi-domain* method .

system on a resource rich out-of-domain corpus is trained until convergence, and then its parameters are fine tuned on a resource poor in-domain corpus. Conventionally, fine tuning is applied on in-domain parallel corpora. Varga et al. (2017) apply it on parallel sentences extracted from comparable corpora. Comparable corpora have been widely used for SMT by extracting parallel data from them (Chu, 2015). To prevent degradation of out-of-domain translation after fine tuning on in-domain data, Dakwale and Monz (2017) propose an extension of fine tuning that keeps the distribution of the out-of-domain model based on knowledge distillation (Hinton et al., 2015).

Mixed Fine Tuning This method is a combination of the *multi-domain* and *fine tuning* methods (Figure 6). The training procedure is as follows:

1. Train an NMT model on out-of-domain data until convergence.
2. Resume training the NMT model from step 1 on a mix of in-domain and out-of-domain data (by oversampling the in-domain data) until convergence.

Mixed fine tuning addresses the overfitting problem of fine tuning due to the small size of the in-domain data. It is easier to train a good model with out-of-domain data, compared to training a multi-domain model. Once we obtained good model parameters, we can use these parameters for fine tuning on the mixed domain data to obtain better performance for the in-domain model. In addition, mixed fine tuning is faster than multi-domain because training an out-of-domain model convergences faster than training a multi-domain model, which also convergences very fast in fine tuning on the mixed domain data. Chu et al. (2017) show that mixed fine tuning works better than both *multi-domain* and *fine tuning*. In addition, mixed fine tuning has the similar effect as the ensembling method in Dakw and Monz (2017), which does not decrease the out-of-domain translation performance.

Regularization Barone et al. (2017) also realize the overfitting problem during fine tuning. Their strategy to address this problem is to explore regularization techniques such as dropout and L2-regularization. In addition, they also propose *tuneout* that is a variant of dropout for regularization. We think that mixed fine tuning and regularization techniques are complementary to each other.

4.2.2 Architecture Centric

The methods in this section change the NMT architecture for domain adaptation.

Deep Fusion One technique of adaptation with in-domain monolingual data is to train an in-domain RNNLM for the NMT decoder and combine it (also known as fusion) with an NMT model (Gülçehre et

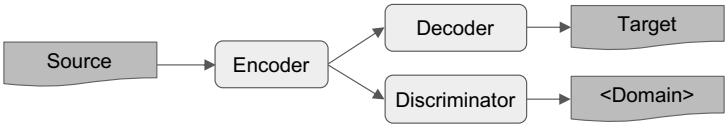


Figure 7: Domain discriminator (Britz et al., 2017).

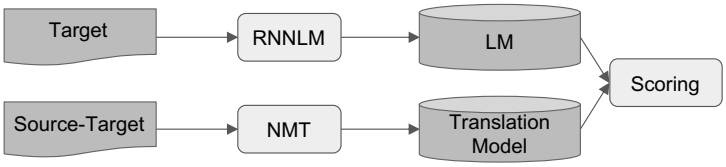


Figure 8: LM shallow fusion (Gülçehre et al., 2015).

al., 2015). Fusion can either be shallow or deep. Formally, deep fusion indicates that the LM and NMT are integrated as a single decoder (i.e., integrating the RNNLM into the NMT architecture). Shallow fusion indicates that the scores of the LM and NMT are considered together (i.e., rescoring the NMT model with the RNNLM model).

In deep fusion, the RNNLM and the decoder of the NMT are integrated by concatenating their hidden states. When computing the output probability of the next word, the model is fine tuned to use the hidden states of both the RNNLM and NMT models. Domhan and Hieber (2017) propose a method similar to the deep fusion method (Gülçehre et al., 2015). However, unlike training the RNNLM and NMT model separately (Gülçehre et al., 2015), Domhan and Hieber (2017) train RNNLM and NMT models jointly.

Domain Discriminator To leverage the diversity of information in multi-domain corpora, Britz et al. (2017) propose a discriminative method. In their discriminative method, they add a feed-forward network (FFNN) as a discriminator on top of the encoder that uses the attention to predict the domain of the source sentence. The discriminator is optimized jointly with the NMT network. Figure 7 shows an overview of this method.

Domain Control Besides using domain tokens to control the domains, Kobus et al. (2016) propose to append word-level features to the embedding layer of NMT to control the domains. In particular, they append a domain tag to each word. They also propose a term frequency - inverse document frequency (tf-idf) based method to predict the domain tag for input sentences.

4.2.3 Decoding Centric

Decoding centric methods focus on the decoding algorithm for domain adaptation, which are essentially complementary to the other model centric methods.

Shallow Fusion Shallow fusion is an approach where LMs are trained on large monolingual corpora, following which they are combined with a previously trained NMT model (Gülçehre et al., 2015). In the shallow fusion (Gülçehre et al., 2015), the next word hypotheses generated by an NMT model is rescored by the weighted sum of the NMT and RNNLM probabilities (Figure 8).

Ensembling Freitag and Al-Onaizan (2016) propose to ensemble the out-of-domain domain and the fine tuned in-domain models. Their motivation is exactly the same as the work of Dakwale and Monz (2017), which is preventing degradation of out-of-domain translation after fine tuning on in-domain data.

Neural Lattice Search Khayrallah et al. (2017) propose a stack-based decoding algorithm over word lattices, while the lattices are generated by SMT (Dyer et al., 2008). In their domain adaptation experiments, they show that stack-based decoding is better than conventional decoding.

5 Domain Adaptation in Real-World Scenarios

A domain adaptation method should be adopted according to the certain scenarios. For example, when there are some pseudo parallel in-domain data in the out-of-domain data, sentence selection is preferred; when only additional monolingual data is available, LM and NMT fusion can be adopted. In many cases, both out-of-domain parallel data and monolingual in-domain data are available, making the combination

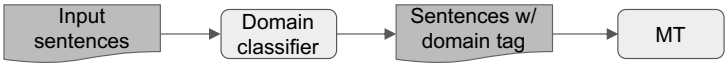


Figure 9: Domain adaptation in an input domain unknown scenario.

of different methods possible. Chu et al. (2018) conduct a study that applys mixed fine tuning (Chu et al., 2017) on synthetic parallel data (Sennrich et al., 2016b), which shows better performance than either method. Therefore, we do not recommend any particular techniques in this paper but recommend readers to choose the best method for their own scenarios.

Most of the above domain adaptation studies assume that the domain of the data is given. However, in a practical view such as an online translation engine, the domain of the sentences input by the users are not given. For such scenario, predicting the domains of the input sentences is crucial for good translation. To address this problem, a common method in SMT is to firstly classify the domains and then translate input sentences in classified domains using corresponding models (Huck et al., 2015). Xu et al. (2007) perform domain classification for a Chinese-English translation task. The classifiers operate on whole documents rather than on individual sentences, using LM interpolation and vocabulary similarities. Huck et al. (2015) extend the work of Xu et al. (2007) on the sentence level. They use LMs and maximum entropy classifiers to predict the target domain. Banerjee et al. (2010) build a support vector machine classifier using tf-idf features over bigrams of stemmed content words. Classification is carried out on the level of individual sentences. Wang et al. (2012) rely on averaged perceptron classifiers with various phrase-based features.

For NMT, Kobus et al. (2016) propose an NMT domain control method, by appending either domain tags or features to the word embedding layer of NMT. They adopt an in-house classifier to distinguish the domain information. Li et al. (2016) propose to search similar sentences in the training data using the test sentence as a query, and then fine tune the NMT model using the retrieved training sentences for translating the test sentence. Farajian et al. (2017) follow the strategy of Li et al. (2016), but propose to dynamically set the hyperparameters (i.e., learning rate and number of epochs) of the learning algorithm based on the similarity of the input sentence and the retrieved sentences for updating the NMT model. Figure 9 shows an overview of domain adaptation for MT in the input domain unknown scenario.

6 Future Directions

6.1 Domain Adaptation for State-of-the-art NMT Architectures

Since the success of RNN based NMT (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015), other architectures of NMT have been developed. One representative architecture is CNN based NMT (Gehring et al., 2017). Compared to RNN based models, CNN based models can be computed fully parallel during training and are much easier to optimize. Another representative architecture is the *Transformer*, which is based on attention only (Vaswani et al., 2017). It has been shown that CNN based NMT and the *Transformer* significantly outperform the state-of-the-art RNN based NMT model of Wu et al. (2016) in both the translation quality and speed perspectives. However, currently, most of the domain adaptation studies for NMT are based on the RNN based model (Bahdanau et al., 2015). The research of domain adaptation techniques for these latest state-of-the-art NMT models is obviously an important future direction.

6.2 Domain Specific Dictionary Incorporation

How to use external knowledge such as dictionaries and knowledge bases for NMT remains a big research question. In domain adaptation, the use of domain specific dictionaries is a very crucial problem. In the practical perspective, many translation companies have created domain specific dictionaries but not domain specific corpora. If we can study a good way to use domain specific dictionaries, it will significantly promote the practical use of MT. There are some studies that try to use dictionaries for NMT, but the usage is limited to help low frequent or rare word translation (Arthur et al., 2016; Zhang and Zong, 2016a). Arcan and Buitelaar (2017) use a domain specific dictionary for terminology translation,

but they simply apply the unknown word replacement method proposed by Luong et al. (2015), which suffers from noisy attention.

6.3 Multilingual and Multi-Domain Adaptation

It may not always be possible to use an out-of-domain parallel corpus in the same language pair and thus it is important to use data from other languages (Johnson et al., 2016). This approach is known as cross-lingual transfer learning, which transfers NMT model parameters among multiple languages. It is known that a multilingual model, which relies on parameter sharing, helps in improving the translation quality for low resource languages especially when the target language is the same (Zoph et al., 2016). There are studies where either multilingual (Firat et al., 2016; Johnson et al., 2017) or multi-domain models (Sajjad et al., 2017) are trained, but none that attempt to package multiple language pairs and multiple domains into a single translation system. Even if out-of-domain data in the same language pair exists, it is possible that using both multilingual and multi-domain data can boost the translation performance. Therefore, we think that multilingual and multi-domain adaptation for NMT can be another future direction. Chu and Dabre (2018) conduct a preliminary study for this topic.

6.4 Adversarial Domain Adaptation and Domain Generation

Generative adversarial networks are a class of artificial intelligence algorithms used in unsupervised machine learning, which are introduced by (Goodfellow et al., 2014). Adversarial methods have become popular in domain adaptation (Ganin et al., 2016), which minimize an approximate domain discrepancy distance through an adversarial objective with respect to a domain discriminator (Tzeng et al., 2017). They have been applied to domain adaptation tasks in computer vision and machine learning (Tzeng et al., 2017; Motiian et al., 2017; Volpi et al., 2017; Zhao et al., 2017; Pei et al., 2018). Recently, some of the adversarial methods began to be introduced into some NLP tasks (Liu et al., 2017; Chen et al., 2017b) and NMT (Britz et al., 2017).

Most of the existing methods focus on adapting from a general domain into a specific domain. In the real scenario, training data and test data have different distributions and the target domains are sometimes unseen. Irvine et al. (2013) analyze the translation errors in such scenarios. Domain generalization aims to apply knowledge gained from labeled source domains to unseen target domains (Li et al., 2018). It provides a way to match the distribution of training data and test data in real-world MT, which may be a future trend of domain adaptation for NMT.

7 Conclusion

Domain adaptation for NMT is a rather new but very important research topic to promote MT for practical use. In this paper, we gave the first comprehensive review of the techniques mainly being developed in the last two years. We compared domain adaptation techniques for NMT with the techniques being studied in SMT, which has been the main research area in the last two decades. In addition, we outlooked the future research directions. Connecting domain adaptation techniques in NMT to the techniques in general NLP, computer vision and machine learning is our future work. We hope that this survey paper could significantly promote the research in domain adaptation for NMT.

Acknowledgement

This work was supported by Grant-in-Aid for Research Activity Start-up #17H06822, JSPS. We are very appreciated to Dr. Raj Dabre for the deep discussion of the structure for this paper. We also thank the anonymous reviewers for their insightful comments.

References

Mihael Arcan and Paul Buitelaar. 2017. Translating domain-specific expressions in knowledge bases with neural machine translation. *CoRR*, abs/1709.02184.

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567, Austin, Texas, November. Association for Computational Linguistics.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Edinburgh, Scotland, U.K.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*, San Diego, USA, May. International Conference on Learning Representations.

Pratyush Banerjee, Jinhua Du, Baoli Li, Sudip Naskar, Andy Way, and Josef Genabith. 2010. Combining multi-domain statistical machine translation models using automatic classifiers. In *The Ninth Conference of the Association for Machine Translation in the Americas*, Denver, Colorado.

Arianna Bisazza, Nick Ruiz, and Marcello Federico. 2011. Fill-up versus interpolation methods for phrase-based SMT adaptation. In *IWSLT*, pages 136–143. ISCA.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark, September. Association for Computational Linguistics.

Denny Britz, Quoc Le, and Reid Pryzant. 2017. Effective domain mixing for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 118–126, Copenhagen, Denmark, September. Association for Computational Linguistics.

M Cettolo, J Niehues, S Stüker, L Bentivogli, R Cattoni, and M Federico. 2015. The iwslt 2015 evaluation campaign. In *Proceedings of the Twelfth International Workshop on Spoken Language Translation (IWSLT)*.

Boxing Chen, Roland Kuhn, George Foster, Colin Cherry, and Fei Huang. 2016. Bilingual methods for adaptive training data selection for machine translation. In *The Twelfth Conference of The Association for Machine Translation in the Americas*, pages 93–106, Austin, Texas.

Boxing Chen, Colin Cherry, George Foster, and Samuel Larkin. 2017a. Cost weighting for neural machine translation domain adaptation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 40–46, Vancouver.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017b. Adversarial multi-criteria learning for chinese word segmentation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1193–1203, Vancouver, Canada. Association for Computational Linguistics.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1965–1974, Berlin, Germany, August. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

Chenhui Chu and Raj Dabre. 2018. Multilingual and multi-domain adaptation for neural machine translation. In *Proceedings of the 24st Annual Meeting of the Association for Natural Language Processing (NLP 2018)*, pages 909–912, Okayama, Japan, Match.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of domain adaptation methods for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, July. Association for Computational Linguistics.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2018. A comprehensive empirical comparison of domain adaptation methods for neural machine translation. *Journal of Information Processing (JIP)*, 26(1):1–10.

Chenhui Chu. 2015. Integrated parallel data extraction from comparable corpora for statistical machine translation. *Doctoral Thesis, Kyoto University*.

Gabriela Csurka. 2017. Domain adaptation for visual applications: A comprehensive survey. *CoRR*, abs/1702.05374.

Anna Currey, Antonio Valerio Miceli Barone, and Kenneth Heafield. 2017. Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 148–156, Copenhagen, Denmark, September. Association for Computational Linguistics.

Praveen Dakwale and Christof Monz. 2017. Fine-tuning for neural machine translation with limited degradation across in- and out-of-domain data. In *Proceedings of the 16th Machine Translation Summit (MT-Summit 2017)*, pages 156–169.

Tobias Domhan and Felix Hieber. 2017. Using target-side monolingual data for neural machine translation through multi-task learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1500–1505, Copenhagen, Denmark, September. Association for Computational Linguistics.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 678–683, Sofia, Bulgaria, August.

Nadir Durrani, Hassan Sajjad, Shafiq Joty, Ahmed Abdelali, and Stephan Vogel. 2015. Using joint models for domain adaptation in statistical machine translation. In *Proceedings of MT Summit XV*, pages 117–130, Miami, FL, USA.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proceedings of ACL-08: HLT*, pages 1012–1020, Columbus, Ohio, June. Association for Computational Linguistics.

M. Amin Farajian, Marco Turchi, Matteo Negri, and Marcello Federico. 2017. Multi-domain neural machine translation through unsupervised adaptation. In *Proceedings of the Second Conference on Machine Translation*, pages 127–137, Copenhagen, Denmark, September. Association for Computational Linguistics.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 866–875.

George Foster and Roland Kuhn. 2007. Mixture-model adaptation for smt. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 128–135, Stroudsburg, PA, USA. Association for Computational Linguistics.

George Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 451–459, Cambridge, MA.

Markus Freitag and Yaser Al-Onaizan. 2016. Fast domain adaptation for neural machine translation. *arXiv preprint arXiv:1612.06897*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122.

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

Isao Goto, Ka-Po Chow, Bin Lu, Eiichiro Sumita, and Benjamin K. Tsou. 2013. Overview of the patent machine translation task at the ntcir-10 workshop. In *Proceedings of the 10th NTCIR Conference*, pages 260–286, Tokyo, Japan, June. National Institute of Informatics (NII).

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *CoRR*, abs/1503.03535.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Cuong Hoang and Khalil Sima'an. 2014. Latent domain translation models in mix-of-domains haystack. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1928–1939, Dublin, Ireland.

Matthias Huck, Alexandra Birch, and Barry Haddow. 2015. Mixed-domain vs. multi-domain statistical machine translation. *Proceedings of MT Summit XV*, 1:240–255.

Kenji Imamura and Eiichiro Sumita. 2016. Multi-domain adaptation for statistical machine translation based on feature augmentation. In *Proceedings of the 12th Conference of the Association for Machine Translation in the Americas*, Austin, Texas, USA.

Ann Irvine, John Morgan, Marine Carpuat, Hal Daume III, and Dragos Munteanu. 2013. Measuring machine translation errors in new domains. *Transactions of the Association for Computational Linguistics*, 1:429–440.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China, July. Association for Computational Linguistics.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271, Prague, Czech Republic.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR*, abs/1611.04558.

Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernand a Vigas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Huda Khayrallah, Gaurav Kumar, Kevin Duh, Matt Post, and Philipp Koehn. 2017. Neural lattice search for domain adaptation in machine translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 20–25, Taipei, Taiwan, November. Asian Federation of Natural Language Processing.

Catherine Kobus, Josep Crego, and Jean Senellart. 2016. Domain control for neural machine translation. *arXiv preprint arXiv:1612.06140*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver, August. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

Philipp Koehn. 2017. Neural machine translation. *CoRR*, abs/1709.07809.

Patrik Lambert, Holger Schwenk, Christophe Servan, and Sadaf Abdul-Rauf. 2011. Investigations on translation model adaptation using monolingual data. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 284–293, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xiaoqing Li, Jiajun Zhang, and Chengqing Zong. 2016. One sentence one model for neural machine translation. *CoRR*, abs/1609.06490.

Ya Li, Mingming Gong, Xinmei Tian, Tongliang Liu, and Dacheng Tao. 2018. Domain generalization via conditional invariant representations. In *The Thirty-Second AAAI Conference on Artificial Intelligence*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10, Vancouver, Canada. Association for Computational Linguistics.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the 12th International Workshop on Spoken Language Translation*, pages 76–79, Da Nang, Vietnam, December.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China, July. Association for Computational Linguistics.

Saab Mansour and Hermann Ney. 2012. A simple and effective weighted phrase extraction for machine translation adaptation. In *The 9th International Workshop on Spoken Language Translation*, Hong Kong.

Benjamin Marie and Atsushi Fujita. 2017. Efficient extraction of pseudo-parallel sentences from raw monolingual data using word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 392–398, Vancouver, Canada, July. Association for Computational Linguistics.

Spyros Matsoukas, Antti-Veikko I. Rosti, and Bing Zhang. 2009. Discriminative corpus weight estimation for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 708–717, Singapore.

Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann, and Rico Sennrich. 2017. Regularization techniques for fine-tuning in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1489–1494, Copenhagen, Denmark, September. Association for Computational Linguistics.

Robert C Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 220–224, Uppsala, Sweden.

Saeid Motiian, Quinn Jones, Seyed Mehdi Iranmanesh, and Gianfranco Doretto. 2017. Few-shot adversarial domain adaptation. *CoRR*, abs/1711.02536.

Toshiaki Nakazawa, Shohei Higashiyama, Chenchen Ding, Hideya Mino, Isao Goto, Hideto Kazawa, Yusuke Oda, Graham Neubig, and Sadao Kurohashi. 2017. Overview of the 4th workshop on asian translation. In *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*, pages 1–54, Taipei, Taiwan, November. Asian Federation of Natural Language Processing.

Graham Neubig. 2017. Neural machine translation and sequence-to-sequence models: A tutorial. *CoRR*, abs/1703.01619.

Jan Niehues and Alex H. Waibel. 2012. Detailed analysis of different strategies for phrase table adaptation in smt. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*, San Diego, US-CA.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October.

Jaehong Park, Jongyoon Song, and Sungroh Yoon. 2017. Building a neural machine translation system using only synthetic parallel data. *CoRR*, abs/1704.00253.

Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. 2018. Multi-adversarial domain adaptation.

Anthony Rousseau, Fethi Bougares, Paul Deléglise, Holger Schwenk, and Yannick Estève. 2011. Lium systems for the iwslt 2011 speech translation tasks. In *International Workshop on Spoken Language Translation*, San Francisco, USA.

Hassan Sajjad, Nadir Durrani, Fahim Dalvi, Yonatan Belinkov, and Stephan Vogel. 2017. Neural machine translation training in a multi-domain scenario. In *Proceedings of the Twelfth International Workshop on Spoken Language Translation (IWSLT)*, Tokyo, Japan.

Rico Sennrich, Holger Schwenk, and Walid Aransa. 2013. A multi-domain translation model framework for statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 832–840, Sofia, Bulgaria.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–40, San Diego, California, June. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany, August. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016c. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August. Association for Computational Linguistics.

Christophe Servan, Josep Crego, and Jean Senellart. 2016. Domain specialization: a post-training domain adaptation for neural machine translation. *arXiv preprint arXiv:1612.06141*.

Kashif Shah, Loïc Barrault, and Holger Schwenk. 2010. Translation model adaptation by resampling. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 392–399.

Kashif Shah, Loïc Barrault, and Holger Schwenk. 2012. A general framework to weight heterogeneous parallel data for model adaptation in statistical machine translation. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*, San Diego, US-CA.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, pages 3104–3112, Cambridge, MA, USA. MIT Press.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. *CoRR*, abs/1702.05464.

Masao Utiyama and Hitoshi Isahara. 2003. Reliable measures for aligning japanese-english news articles and sentences. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 72–79, Sapporo, Japan, July. Association for Computational Linguistics.

Marlies van der Wees, Arianna Bisazza, and Christof Monz. 2017. Dynamic data selection for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1410, Copenhagen, Denmark, September. Association for Computational Linguistics.

Adam Csaba Varga. 2017. Domain adaptation for multilingual neural machine translation. *Master Thesis, Saarlandes University*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Riccardo Volpi, Pietro Morerio, Silvio Savarese, and Vittorio Murino. 2017. Adversarial feature augmentation for unsupervised domain adaptation. *CoRR*, abs/1711.08561.

Wei Wang, Klaus Macherey, Wolfgang Macherey, Franz Och, and Peng Xu. 2012. Improved domain adaptation for statistical machine translation. In *Proceedings of AMTA*, San Diego, California, USA.

Rui Wang, Hai Zhao, Bao-Liang Lu, Masao Utiyama, and Eiichiro Sumita. 2014. Neural network based bilingual language model growing for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 189–195, Doha, Qatar, October. Association for Computational Linguistics.

Rui Wang, Hai Zhao, Bao-Liang Lu, Masao Utiyama, and Eiichiro Sumita. 2016. Connecting phrase based statistical machine translation adaptation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3135–3145, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Rui Wang, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. 2017a. Sentence embedding for neural machine translation domain adaptation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 560–566, Vancouver, Canada, July. Association for Computational Linguistics.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. 2017b. Instance weighting for neural machine translation domain adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1482–1488, Copenhagen, Denmark.

Rui Wang, Masao Utiyama, Andrew Finch, Lemao Liu, Kehai Chen, and Eiichiro Sumita. 2018. Sentence selection and weighting for neural machine translation domain adaptation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. 2016. A survey of transfer learning. *Journal of Big Data*, 3(1):9, May.

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

Jia Xu, Yonggang Deng, Yuqing Gao, and Hermann Ney. 2007. Domain dependent statistical machine translation. In *MT Summit*, Copenhagen, Denmark.

Jiajun Zhang and Chengqing Zong. 2016a. Bridging neural machine translation and bilingual dictionaries. *CoRR*, abs/1610.07272.

Jiajun Zhang and Chengqing Zong. 2016b. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Austin, Texas, November. Association for Computational Linguistics.

Han Zhao, Shanghang Zhang, Guanhang Wu, João P. Costeira, José M. F. Moura, and Geoffrey J. Gordon. 2017. Multiple source domain adaptation with adversarial training of neural networks. *CoRR*, abs/1705.09684.

Xinpeng Zhou, Hailong Cao, and Tiejun Zhao. 2015. Domain adaptation for SMT using sentence weight. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 153–163, Guangzhou, China.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1568–1575.

An Evaluation of Neural Machine Translation Models on Historical Spelling Normalization

Gongbo Tang, Fabienne Cap, Eva Pettersson, and Joakim Nivre
Uppsala University
Uppsala, Sweden
firstname.lastname@lingfil.uu.se

Abstract

In this paper, we apply different NMT models to the problem of historical spelling normalization for five languages: English, German, Hungarian, Icelandic, and Swedish. The NMT models are at different levels, have different attention mechanisms, and different neural network architectures. Our results show that NMT models are much better than SMT models in terms of character error rate. The vanilla RNNs are competitive to GRUs/LSTMs in historical spelling normalization. Transformer models perform better only when provided with more training data. We also find that subword-level models with a small subword vocabulary are better than character-level models. In addition, we propose a hybrid method which further improves the performance of historical spelling normalization.

1 Introduction

With increasing access to digital historical text, the processing of these historical texts is attracting more and more interest. However, in contrast to modern text, historical text processing faces more challenges. First, for historical text, there is little annotated data for training a model, which leads to data sparsity issues when using statistical methods, similar to the situation for low-resource languages. Second, there are a lot of variations in historical texts from different time periods, not only in spelling but also in lexical semantics and syntax. Therefore, the NLP tools developed for modern text cannot be used for these historical texts directly. Spelling normalization is the task of mapping a historical spelling to its modern spelling. It is usually used as a preprocessing step before feeding the historical text into modern NLP tools (Pettersson et al., 2013b; Bollmann, 2013; Sánchez-Martínez et al., 2013), which leads to much better results compared to analyzing unnormalized historical texts.

There are some papers in which neural machine translation (NMT) models are employed for the spelling normalization task. Korchagina (2017) utilizes a character-level NMT model for medieval German texts. Bollmann et al. (2017) apply an attention-based NMT model to historical German texts. The evidence so far is too incomplete to draw any general conclusions about the utility of different NMT models for historical spelling normalization. We are interested in exploring how different properties of NMT models interact with different aspects of the spelling normalization problem and find some generalizations about the use of NMT models for this task.

In this paper, we apply different NMT models to the spelling normalization task for historical stages of five languages, English, German, Hungarian, Icelandic, and Swedish. We compare our result to those of Pettersson et al. (2014), which are obtained with statistical machine translation (SMT) models. We investigate whether NMT models outperform SMT models in general, and explore which properties of NMT models are suitable for spelling normalization. Compared to the conventional machine translation (MT) tasks, we train models on token pairs instead of sentence pairs. Token length is usually shorter than sentence length. After reviewing related work in Section 2, we give our hypotheses about utilizing NMT models for the spelling normalization task and select different NMT models based on our hypotheses in Section 3. The selected NMT models are at different levels (character-level, subword-level), have

Proceedings of the 27th International Conference on Computational Linguistics, pages 1320–1331
Santa Fe, New Mexico, USA, August 20-26, 2018.

different attention mechanisms (no attention, soft-attention, multi-head-attention), and different neural network architectures (vanilla recurrent neural networks (RNNs), gated recurrent units (GRUs), long short-term memory units (LSTMs), and self-attention). In Section 4, we describe the datasets and our detailed experimental settings. In Section 5, we give our results and analyze the performance of different NMT models. Our conclusions and future work are in Section 6.

To conclude, our main contributions can be summarized as follows:

- We evaluate different NMT models on historical spelling normalization in a multilingual setting.
- We find that NMT models are better than SMT models considering character error rate (CER).
- We show that vanilla RNNs are competitive to GRUs/LSTMs.
- We demonstrate that transformer models perform better when provided with more training data.
- We reveal that models with a small subword vocabulary are better than character-level models.

2 Related Work

2.1 Historical Spelling Normalization

Various methods have been employed for historical spelling normalization. Rayson et al. (2005) use a dictionary to map tokens to their modernized spellings, and many different edit-distance-based methods have been proposed to deal with spelling normalization (Bollmann et al., 2011; Pettersson et al., 2013a). In addition, character-level SMT models have been applied to spelling normalization, where models are trained on token pairs instead of sentence pairs (Pettersson et al., 2013b; Scherrer and Erjavec, 2013; Sánchez-Martínez et al., 2013). Each character of a token is viewed as a word of a sentence. The language models are trained on character N-grams instead of word N-grams. Pettersson et al. (2014) evaluate dictionary-based methods, edit distance-based methods, and SMT methods on five different historical languages. The results show that the character-level SMT model performs best on four out of five historical languages.

With the development of deep learning, various neural networks have been applied to many tasks. In recent years, NMT models (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015) have outperformed SMT models (Koehn et al., 2003) distinctly in various translation tasks. We hypothesize that NMT models also perform better than SMT models for the historical spelling normalization task. Bollmann and Søgaard (2016) view the spelling normalization as a character-level sequence labeling task, and utilize a bi-directional LSTM for this task, which is better than a conditional random field (CRF) model. They also use additional data with similar but not the same historical spelling for a multi-task learning model, and gain further improvement. Korchagina (2017) applies a character-level NMT model to medieval German text, and finds that the NMT models can only outperform the SMT models with a larger training set. Bollmann et al. (2017) test attention-based NMT models, and multi-task learning models which learn to normalize and pronounce with a grapheme-to-phoneme dictionary, on spelling normalization. Both of them achieve good performance. They hypothesize that the reason why the combination of these two models does not gain more improvement is that the multi-task learning has already learned the attention patterns.

2.2 Neural Machine Translation

In vanilla NMT models, the source sentence is encoded into a fixed-size vector by the encoder. Then, this vector is fed into a decoder. The decoder generates the target sentence word by word conditioned on the fixed-size vector and the generated target words (Kalchbrenner and Blunsom, 2013). Various RNN architectures are usually used as encoders and decoders. Cho et al. (2014) find that the vanilla RNN-based NMT models perform poorly in translating long sentences, which means that vanilla RNNs have problems with long-distance dependencies. To deal with these problems, Cho et al. (2014) propose GRUs, while Sutskever et al. (2014) use LSTMs (Hochreiter and Schmidhuber, 1997) to replace the vanilla RNNs. However, any two tokens in RNNs still have a linear distance. Thus, Vaswani et al. (2017) replace RNNs with self-attention networks which connect any two tokens in a sentence directly.

Due to the expensive computation of NMT models, the vocabulary size is usually very limited, which causes a lot of out-of-vocabulary (OOV) words. Character-level models (Ling et al., 2015; Costa-jussà

and Fonollosa, 2016; Chung et al., 2016) and subword-level models (Sennrich et al., 2016; Wu et al., 2016) are widely used to deal with OOV problems. These two kinds of models need additional segmentation compared to word-level models. For character-level models, we just need to separate each character by space. But we need more complicated segmentation methods for subword-level models. Sennrich et al. (2016) utilize character n-grams and a byte pair encoding (BPE) algorithm (Gage, 1994) for segmentation. Wu et al. (2016) apply the wordpiece model (Schuster and Nakajima, 2012) to segmentation. Based on their experiments, subword-level models outperform word-level and character-level models.

Attention-based NMT models have outperformed all the other architectures in NMT in recent years. Many improved attention-based models have been proposed. Bahdanau et al. (2015) propose an attention-based model which can automatically search for source words that their hidden states are relevant to predicting a target word during decoding. Source words which have a higher correlation with the predicting target word will be assigned a higher weight. Most of the attention-based models use this kind of attention, which is called soft-attention in Xu et al. (2015). Vaswani et al. (2017) propose a model named *Transformer*, with multi-layer and multi-head attention mechanism which is more fine-grained.

3 NMT Models

When we apply NMT models to the historical spelling normalization task, the first research question is which NMT model is most suitable for this task. In this section, we first give four hypotheses about NMT models for spelling normalization, based on the data features of historical spellings and the features of NMT models. Then, we list 8 different NMT models to consider for the spelling normalization task.

3.1 Hypotheses

Hypothesis 1 The performance gap between vanilla RNNs and GRUs/LSTMs is small. In contrast to conventional NMT models, the historical and modern token pairs are our training data instead of parallel sentence pairs. In our experiments, the average token length varies from 4 to 6, which means that we build the model on much shorter sequences. The long-distance problem will be alleviated. It should be noted that we compare the gap to the gap in NMT (Bahdanau et al., 2015)[1].

Hypothesis 2 The gap between NMT models with attention and without attention is also small. Since the average token length is only around five, additionally paying attention to all the tokens in the source sentence may be unnecessary. Thus, we hypothesize that the decoder in the vanilla Encoder-Decoder model can predict most of the targets correctly with only one fixed-size vector from the encoder, even without any attention mechanisms. It should be mentioned that we compare the gap to the gap in NMT (Britz et al., 2017)[2].

Hypothesis 3 Transformer models perform better than soft-attention-based models. Transformer models have more advanced self-attention networks and more fine-grained multi-head attention mechanisms compared to RNN-based models with soft-attention. Thus, transformer models have better performance in conventional translation tasks. We hypothesize that it is the same in the spelling normalization task.

Hypothesis 4 Subword-level NMT models perform better than character-level NMT models. Character-level and subword-level models are proposed to deal with the problem of out-of-vocabulary words mainly, and subword-level NMT models usually outperform character-level models. As we only have small sets of token pairs, it is better to use character-level or subword-level NMT models rather than word-level models.

3.2 Models

To test our hypotheses proposed in the previous section, we will explore 8 different NMT models for the spelling normalization task. The NMT models vary in attention mechanism, neural network architecture, and token granularity. Table 1 gives a more detailed overview.

[1]The BLEU (Papineni et al., 2002) scores are 15.73, and 21.83.

[2]The BLEU scores are 17.82, and 26.75.

Name	Level	Attention	Architecture
NoAtt-RNN	character	no	RNN
NoAtt-GRU			GRU
NoAtt-LSTM			LSTM
Att-RNN		soft	RNN
Att-GRU			GRU
Att-LSTM			LSTM
Transformer		multi-head	Self-attention
BPE-Soft	subword	soft	LSTM

Table 1: NMT models for the spelling normalization task. *RNN* means vanilla RNNs.

Since we have hypothesized that different RNN architectures have slight differences, all the subword-level models with soft-attention are trained on LSTMs.

4 Experimental Setup

4.1 Data

All the datasets[3] are exactly the same as the parallel datasets for the SMT models in Pettersson et al. (2014), which are described in Table 2. Data details are shown in Table 3. The datasets consist of a list of token pairs, which have one historical spelling and the corresponding modernized spelling. Note that the same modern spelling may occur with different historical spellings. Moreover, some historical words may be extinct, and people have to use a spelling with a similar meaning but different lexemes as its normalization. Some illustrative English examples are given in Table 4. If a historical spelling is identical to its modern spellings, we call it an *unchanged spelling*. Otherwise, it is called a *changed spelling*. In different languages, the number of unchanged spellings is different.

Language	Time period	Origin
English	1386–1698	*Innsbruck Corpus of English Letters*, a subset of the *Innsbruck Computer Archive if Machine-Readable English Texts* (Markus, 1999)
German	1659–1780	*GerManC* corpus (Scheible et al., 2011)
Hungarian	1440–1541	*Hungarian Generative Diachronic Syntax* project (Simon, 2014)
Icelandic	1150–2008	*Icelandic Parsed Historical Corpus* (Rögnvaldsson et al., 2012)
Swedish	1527–1812	*Gender and Work corpus* (GaW) (Fiebranz et al., 2011)

Table 2: Origin and time periods of the datasets.

Language	Training	Development	Test	Unchanged	Token	Char	Max	Avg
English	148,852	16,461	17,791	75.8	22,302	102	22	4.16
German	39,887	5,418	5,005	84.4	11,521	100	27	4.74
Hungarian	137,669	17,181	17,214	17.1	69,624	128	27	5.91
Icelandic	52,440	6,443	6,384	50.5	14,845	89	16	4.14
Swedish	28,327	2,590	33,544	64.6	11,129	92	36	4.55

Table 3: Statistics of the datasets. The figures in *Training*, *Development*, and *Test* are the numbers of token pairs. The *Unchanged* (%) means the rate of unchanged spellings in the test set. *Token* and *Char* show the token and the character vocabulary sizes in the training set. *Max* and *Avg* show the max length and average length of token in the training set. All counts are based on case-sensitive data.

[3] http://stp.lingfil.uu.se/histcorp/tools.html

Historical	citee	gyve	gyf	late
Modern	city	give	give	late

Table 4: Token pair examples in English.

From Table 3, we can see that English and Hungarian have more training data, around 140,000 token pairs, while Swedish only has about 28,000 token pairs. Swedish has the largest test set with more than 33,000 token pairs. The unchanged rate also differs a lot. There are 84.4% and 75.8% historical spellings that are identical to their modern spellings in German and English, respectively. However, the unchanged rate is only 17.1% in Hungarian. In addition, Hungarian has the largest token vocabulary and character vocabulary[4]. The longest token in Icelandic is only 16, but the longest token in Swedish is 36. The average token length of Hungarian is 5.91, which is the longest in all five languages. This is because Hungarian is an agglutinative language.

4.2 Experimental Settings

Different architectures are hard to compare fairly because many factors affect performance. We aim to create a level playing field for the comparison by training with the same toolkit, Marian (Junczys-Dowmunt et al., 2018). Since there is no implementation of models without attention in Marian, we modify the decoder part to enable Marian to train models without attention.[5] We assume that the case of letters is useful for predicting the modern spellings. Thus, the letters in the training set and the tuning set are case-sensitive. The historical spellings in the test set which are the inputs of the model are also case-sensitive. However, to keep consistency with the baseline, we lowercase all the predicted modern spellings during evaluation.

For character-level models, all the characters are added into the vocabulary, even if they only appear once. For subword-level models, we utilize the BPE method in Sennrich et al. (2016) to generate subword units. We try different BPE vocabulary sizes, varying between 100, 200, 300, 500, 1,000 and 5,000.

The vanilla RNN chooses the "tanh" RNN cell. We enable "mini-bach-fit" which automatically choose the mini-batch size for the given "workspace" size, and the "workspace" is set to 7500. We use *Adam* (Kingma and Ba, 2015) as the optimizer. The learning rate is set to 0.0003, but we set the warmup steps to 16,000, which means that the learning rate increases linearly before 16,000 steps. A model checkpoint is saved every 500 updates. The evaluation metrics on the development set are cross-entropy and perplexity. We set the early stopping patience to 8 checkpoints. All the neural networks have 6 layers. The size of embeddings is 512. We tie the target embeddings and the output embeddings in the output layer. We use the checkpoint that achieves the best perplexity to generate the normalizations. We set the beam size to 5 during decoding.

5 Results

The baseline from Pettersson et al. (2014) has very high word accuracy and low CER scores in all five languages. The results in the baseline are obtained using character-level SMT models except for Icelandic, where the combination of a Levenshtein-based method and a dictionary-based method achieved the best results. We use word accuracy and CER to evaluate the predictions. For the historical spelling normalization task, word accuracy is a very important evaluation metric. Moreover, word accuracy is the only evaluation metric in Bollmann and Søgaard (2016) and Bollmann et al. (2017). However, CER is a good supplement to word accuracy. It is more fine-grained and evaluates the character-level normalizations. In our experiments, we use Levenshtein distance to compute CER. Table 5 gives the detailed results of different models in five languages.

[4]The character vocabulary includes both alphabetic and non-alphabetic characters.

[5]The modification, the NMT model settings, and the code are available in `https://github.com/tanggongbo/normalization-NMT`

	English		German		Hungarian		Icelandic		Swedish	
	Acc	CER	Acc	CER	Acc	CER	Acc	CER	Acc	CER
Baseline	94.3	0.07	96.6	0.04	80.1	0.21	84.6	0.19	**92.9**	0.07
NoAtt-RNN	94.73	0.02	94.89	0.02	90.99	0.03	86.73	0.05	91.44	0.03
NoAtt-GRU	94.79	0.02	94.85	0.02	91.03	0.03	86.98	0.05	91.34	0.03
NoAtt-LSTM	94.61	0.02	95.78	0.02	90.91	0.03	86.61	0.05	91.29	0.03
Att-RNN	94.69	0.02	94.23	0.02	91.69	0.02	**87.59**	0.04	91.56	0.03
Att-GRU	94.80	0.02	94.83	0.02	91.68	0.02	87.17	0.05	91.68	0.03
Att-LSTM	94.85	0.02	96.00	0.02	91.57	0.03	86.83	0.05	91.72	0.03
Transformer	**95.16**	0.02	95.22	0.02	**92.14**	0.02	86.45	0.05	88.99	0.05
BPE-Soft	95.02	0.02	**96.64**	0.01	91.96	0.03	87.19	0.03	91.21	0.03

Table 5: Evaluation results in word accuracy (Acc, %) and CER. The best results in each language have background color. Many identical values in CER are different, but the difference is irrelevant in Chi-square test.

5.1 Word Accuracy

Table 5 shows that NMT models outperform SMT models in four out of five languages, except for Swedish, when we use word accuracy as the evaluation metric. Compared to the other four languages, we get a huge absolute improvement of 12.04% in Hungarian, improving the word accuracy from 80.1% to 92.14%. We get 0.04%, 0.86%, and 2.99% absolute improvement in German, English, and Icelandic, respectively. Our best NMT result in Swedish is still a little lower than the baseline in word accuracy. We attribute the reason to the dataset size, because Swedish has the smallest training set.

We divide the incorrectly normalized spellings into three groups by checking the normalizations of the test set automatically:

1. *Change*: modern spelling is identical to historical spelling, but the model normalized the historical spelling to another spelling.
2. *Copy*: modern spelling is different from historical spelling, but the model copied the historical spelling as the normalization.
3. *Other*: other types of error.

	English	German	Hungarian	Icelandic	Swedish
Change	22.3	28.5	6.1	33.8	25.0
Copy	22.7	41.7	6.1	20.8	23.6
Other	55	29.8	87.8	45.4	51.4

Table 6: Error distributions (%).

Table 6 gives us the error distributions of the best model in each language. The *Change* and *Copy* errors only account for 12.2% in Hungarian which is reasonable, because the changed rate in Hungarian is only 17.1%. The other four languages still have a lot of *Change* and *Copy* errors. This finding reveals that it is a little bit difficult for the NMT model trained on the data that mixed with changed and unchanged spellings to normalize unchanged spellings. However, there are only very few unchanged translations in the MT task.

Therefore, we explore a hybrid method, combining the NMT-based method and the dictionary-based method. More specifically, we first extract a list of unchanged spellings from the training set. During the evaluation, if a word is in this list, we simply copy it as its normalization. If it is not in the list, we feed it to the NMT models. The results in Table 7 show that this hybrid method improves the accuracy further. In particular, the improvements on Icelandic are around 5%.

	English		German		Hungarian		Icelandic		Swedish	
	Acc	△	Acc	△	Acc	△	Acc	△	Acc	△
NoAtt-RNN	95.92	1.19	95.78	0.90	91.81	0.82	91.92	5.18	91.83	0.39
NoAtt-GRU	95.93	1.14	95.44	0.60	91.87	0.84	91.70	4.71	91.73	0.39
NoAtt-LSTM	95.81	1.20	96.42	0.64	91.75	0.83	91.78	5.17	91.69	0.41
Att-RNN	95.90	1.21	94.93	0.70	92.47	0.77	92.54	4.95	91.94	0.38
Att-GRU	95.99	1.19	95.48	0.66	92.49	0.82	92.25	5.08	92.04	0.36
Att-LSTM	96.02	1.17	96.44	0.44	92.36	0.78	91.76	4.93	92.08	0.36
Transformer	96.33	1.17	95.70	0.48	92.94	0.80	91.60	5.15	89.48	0.49
BPE-Soft	96.19	1.18	96.96	0.32	92.74	0.78	92.14	4.95	91.56	0.35

Table 7: The results of combining the NMT-based method and the dictionary-based method. "△" denotes the absolute improvement on accuracy (%) compared to the NMT-based method.

5.2 CER

With the CER measure, we calculate the number of correctly normalized characters, without considering the word level. CER is similar to the BLEU score in MT, and we evaluate at sub-sequence-level rather than the overall accuracy. When we use CER as the evaluation metric, NMT models get the best results for all five languages, even though some models achieve lower accuracy than the baseline. This result is different from the result of Korchagina (2017). In her paper, if the SMT models are better than the NMT models in word accuracy, these SMT models are better than the NMT models in CER as well. We assume that this may be due to different neural network architectures: they use CNNs while we use RNNs and self-attention networks.

	English	German	Hungarian	Icelandic	Swedish
Changed	1.45	1.07	2.58	1.41	1.32
Incorrect	1.81	1.64	1.78	1.64	1.54

Table 8: The average edit distance of the changed spellings in test set and the average edit distance of the incorrectly normalized changed spellings.

Table 8 shows the edit distance of spellings. For the incorrectly normalized changed spellings, the average edit distance is smaller than 2. In other words, we just need less than two edits to translate an incorrectly normalized spelling into the correct one. In the incorrect normalizations, Swedish has the shortest average edit distance 1.54, and English has the longest average edit distance 1.81.

Intuitively, if a spelling has smaller edit distance, it is easier for the model to normalize this spelling correctly. That is to say, the average edit distance of incorrectly normalized spellings will be larger compared to the average edit distance before normalization. However, Hungarian is the exception in Table 8, which indicates that spellings with longer edit distance are more likely to be normalized close to modern spellings in Hungarian. For example, the edit distance between “mōdanac” and “mondák” is 6, yet the model can normalize it correctly. Although the model normalized “mėgbètėgeitnc̣” into “megbetegíteniúk”, which is not identical to the modern spelling “megbetegítenék”, the edit distance nevertheless decreased from 9 to 2. We hypothesize that this could be due to the fact that Hungarian belongs to a different language family than the other four languages.

Table 9 gives some incorrectly normalized examples from the development set. Most of the edit distances of spellings are longer than 1. In addition to *Change* and *Copy* errors, some historical spellings are quite different from their modern spelling, such as “wett” in English. For the historical word “wett”, it is extinct, people just mapped a semantic related modern word to it. “know” has no relations with “wett” in spelling and pronunciation. Characters with different accents also cause mistakes easily. For example, “vetém” in Hungarian and “sér” in Icelandic.

	English	German	Hungarian	Icelandic	Swedish
Historical	alys	julius	vètē	uopn	sielffuer
Normalized	alis	jiues	vetem	opnu	själver
Modern	alice	julius	vetém	vopn	själv
Historical	wett	cohærentz	haila	sier	herrskaper
Normalized	wit	cohaerenz	hajola	sjer	herrskaper
Modern	know	kohärenz	hajla	sér	herrskapen

Table 9: Some incorrectly normalized examples from the development set.

5.3 NMT versus SMT

In the conventional MT tasks, NMT models usually outperform SMT models. The first reason is that the dense embeddings in NMT are powerful representations. The second reason is that NMT models usually consider a larger context compared to SMT models. This is the same in historical spelling normalization. In our experiments, the most obvious example is Hungarian. The absolute improvement is 12.04% in word accuracy. Compared to other languages, Hungarian has the largest token and character vocabularies and the highest changed rate. It also has the longest average token length. Thus, NMT models can represent these larger vocabularies better than SMT models. NMT models are also better at capturing the context information when generating the normalization. For example, the NMT models can normalize a 14-character spelling "aldozatt'oknak" into "áldoza**tuk**nak" correctly, while the SMT models normalize it into "áldoza**tok**nak". In the training set, 'tok' is much more frequent than 'tuk'. Since SMT models are more focused on a local context, the SMT models choose 'tok' rather than 'tuk'.

However, in terms of accuracy, it is still hard for NMT models to exceed SMT models in Swedish. We also find that the performance of NMT models is quite close to the baseline in German which has the second smallest training dataset. We hypothesize that the size of training data is crucial for NMT models to exceed SMT models.

As there is much more test data in Swedish compared to other languages, we test our hypothesis by moving some token pairs from test sets to training sets and development sets. More specifically, we create two new datasets, in which 27,000 and 30,000 token pairs are moved from the beginning of the test set to the training set and the development set, respectively. Both the datasets and results are described in Table 10.

Training	Development	Test	Att-RNN	Att-GRU	Att-LSTM	Transformer
28,327	2,590	33,544	91.56	91.68	91.72	88.99
51,237	6,590	6,544	95.45	94.79	94.97	95.18
57,637	3,190	3,544	96.02	95.77	95.65	95.77

Table 10: The accuracy of different models in Swedish with different dataset settings.

Table 10 shows that the NMT models achieve much higher accuracy with more training data. This result indicates that the performance of NMT models is highly related to the size of training set.

5.4 Different NMT Models

Hypothesis 1 is that the performance gap between vanilla RNNs and GRUs/LSTMs will not be huge. The results in Table 5 reveal that the vanilla RNNs are competitive to the GRUs/LSTMs in this task. *Att-RNN* even performs better than *Att-GRU/LSTM* in Icelandic. However, *Att-RNN* is clearly worse than *Att-LSTM* in German. These results support our Hypothesis 1 well.

Hypothesis 2 states that NMT models with and without attention will not differ a lot. The models with attention are slightly better than models without attention in our experiments, which is in line with the results in Bollmann et al. (2017). However, the gap is quite small. Thus, it fits our Hypothesis 2.

Hypothesis 3 is that transformer models are better than soft-attention-based models. From Table 5,

we can see that *Transformer*, with self-attention and multi-head attention, achieves higher word accuracy in English and Hungarian compared to soft-attention-based models. It is interesting that English and Hungarian have much more training data compared to the other three languages. This result reveals that transformer models need more data to exceed RNN-based models.

Hypothesis 4, finally, states that subword-level models are better than character-level models. Our experimental results for four languages (except Swedish) show that subword-level models are superior to character-level models when the BPE vocabulary is small. In subword-level models, the vocabulary includes all the characters and learned subword units. We try different BPE vocabulary sizes. All the subword-level models are trained on LSTMs. Table 11 gives the results with different BPE sizes. Many historical spellings only have several instances in the training set. The NMT model cannot translate the token well at the token level. Moreover, there is also a data sparsity problem for the subwords when we set a larger BPE vocabulary. We assume that BPE maybe cannot learn rare subword units very well, because of the data sparsity. That is why subword-level models perform better in the conventional MT tasks, which have a much larger training set. We find that the subword-level models perform worse than character-level models when the BPE vocabulary is larger than 300 in all five languages.

BPE-size	English	German	Hungarian	Icelandic	Swedish
0	94.85	96	91.57	86.83	**91.72**
100	**95.02**	**96.64**	91.87	**87.19**	91.21
200	94.91	96.28	91.81	86.89	91.13
300	94.69	96.5	**91.96**	86.76	90.84
500	94.54	96.42	91.52	86.51	90.57
1,000	94.52	96.18	91.44	86.29	89.67
5,000	93.71	95.06	89.43	84.87	85.47

Table 11: Accuracy (%) with different BPE vocabulary sizes. "0" represents the character-level models.

Historical languages are considered as low-resource languages. Hence the result of Hypothesis 4 can be interpreted to mean that subword-level models with a small subword vocabulary can further improve the performance compared to character-level models in low-resource languages.

6 Conclusions and Future Work

In this paper, we explore different NMT models for the historical spelling normalization task in five languages, English, German, Hungarian, Icelandic, and Swedish. We propose four hypotheses on NMT models, which are the general questions to ask when applying NMT models to the historical spelling normalization task.

We find that the performance gap between vanilla RNNs and GRUs/LSTMs is very small, vanilla RNNs are even competitive to GRUs/LSTMs in Hungarian and Icelandic. We demonstrate that the gap between NMT models with or without attention is also slight. We show that the subword-level models with a small subword vocabulary are better than character-level models. However, subword-level models with a larger vocabulary suffer from data sparsity.

When we use word accuracy as the evaluation metric, NMT models can get better results for four languages compared to SMT models. However, all NMT models perform better than SMT models for all five languages when we use CER as the evaluation metric. In addition, the size of the training set is crucial to NMT models. Particularly, transformer models are superior to RNN-based models only when provided with more training data. These findings could contribute to the development of general NMT systems, especially for low-resource languages. Since NMT models are more likely to generate incorrect normalizations of unchanged spellings, we propose a hybrid method using both NMT-based methods and dictionary-based method which improves the performance further.

In the future, we could 1) explore some hard-attention-based models, 2) introduce phoneme knowledge into NMT models, and 3) use sentence pairs for spelling normalization. Compared to soft attention,

hard attention (Xu et al., 2015) only pays attention to one or several specified source word annotations. Aharoni and Goldberg (2017) employ hard monotonic attention for a morphological inflection generation task. The variation between historical spelling and modern spelling is usually monotonic, which is similar to morphological inflection. Thus, hard attention should work well in historical spelling normalization as well.

Many words have changed their spellings, but they keep the same pronunciation. Thus, Bollmann et al. (2017) use an additional grapheme-to-phoneme dictionary in a multi-task learning setting. We can add the phonetic dictionaries as additional training data to improve the performance.

In addition to token-pair-based normalization, Ljubešić et al. (2016) use segment pairs with context information to do spelling normalization. NMT models are powerful in using context information. Thus, training the NMT models on sentence pairs is likely to improve the spelling normalization task further, which introduces more context information.

Acknowledgments

We thank all the anonymous reviews who give a lot of valuable and insightful comments. We acknowledge the computational resources provided by CSC in Helsinki and Sigma2 in Oslo through NeIC-NLPL (www.nlpl.eu). We also thank the machine translation group at the University of Edinburgh for providing computational resources. Gongbo Tang is funded by Chinese Scholarship Council (NO. 201607110016).

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2004–2015, Vancouver, Canada. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California, USA.

Marcel Bollmann and Anders Søgaard. 2016. Improving historical spelling normalization with bi-directional lstms and multi-task learning. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 131–139, Osaka, Japan. The COLING 2016 Organizing Committee.

Marcel Bollmann, Florian Petran, and Stefanie Dipper. 2011. Rule-based normalization of historical texts. In *Proceedings of the Workshop on Language Technologies for Digital Humanities and Cultural Heritage*, pages 34–42, Hissar, Bulgaria. Association for Computational Linguistics.

Marcel Bollmann, Joachim Bingel, and Anders Søgaard. 2017. Learning attention for historical text normalization by learning to pronounce. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 332–344, Vancouver, Canada. Association for Computational Linguistics.

Marcel Bollmann. 2013. Pos tagging for historical texts with sparse training data. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 11–18, Sofia, Bulgaria. Association for Computational Linguistics.

Denny Britz, Anna Goldie, Thang Luong, and Quoc Le. 2017. Massive exploration of neural machine translation architectures. *arXiv preprint arXiv:1703.03906*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1693–1703, Berlin, Germany. Association for Computational Linguistics.

Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 357–361, Berlin, Germany. Association for Computational Linguistics.

Rosemarie Fiebranz, Erik Lindberg, Jonas Lindström, and Maria Ågren. 2011. Making verbs count: the research project 'gender and work'and its methodology. *Scandinavian Economic History Review*, 59(3):273–293.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

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

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. *arXiv preprint arXiv:1804.00344*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California, USA.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Natalia Korchagina. 2017. Normalizing medieval german texts: from rules to deep learning. In *Proceedings of the NoDaLiDa 2017 Workshop on Processing Historical Language*, number 133, pages 12–17, Gothenburg, Sweden. Linköping University Electronic Press.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Nikola Ljubešić, Katja Zupan, Darja Fišer, and Tomaž Erjavec. 2016. Normalising Slovene data: historical texts vs. user-generated content. In *Proceedings of the 13th Conference on Natural Language Processing*, pages 146–155, Varanasi, India. Association for Computational Linguistics.

Manfred Markus. 1999. *Manual of ICAMET (Innsbruck Computer Archive of Machine-Readable English Texts)*. Leopold-Franzens-Universitat Innsbruck.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Eva Pettersson, Beáta Megyesi, and Joakim Nivre. 2013a. Normalisation of historical text using context-sensitive weighted levenshtein distance and compound splitting. In *Proceedings of the 19th Nordic Conference of Computational Linguistics*, pages 163–179, Oslo, Norway. Association for Computational Linguistics.

Eva Pettersson, Beáta Megyesi, and Jörg Tiedemann. 2013b. An SMT approach to automatic annotation of historical text. In *Proceedings of the workshop on computational historical linguistics at NODALIDA 2013*, pages 54–69, Oslo, Norway. Association for Computational Linguistics.

Eva Pettersson, Beáta Megyesi, and Joakim Nivre. 2014. A multilingual evaluation of three spelling normalisation methods for historical text. In *Proceedings of the 8th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH)*, pages 32–41, Gothenburg, Sweden. Association for Computational Linguistics.

Paul Rayson, Dawn Archer, and Nicholas Smith. 2005. Vard versus word: A comparison of the UCREL variant detector and modern spell checkers on english historical corpora. In *Proceedings of the Corpus Linguistics 2005*, Birmingham, UK.

Eiríkur Rögnvaldsson, Anton Karl Ingason, Einar Freyr Sigurðsson, and Joel Wallenberg. 2012. The icelandic parsed historical corpus (icepahc). In *Proceedings of the 8th International Conference on Language Resources and Evaluations*, pages 1977–1984, Istanbul, Turkey, May. European Language Resources Association.

Felipe Sánchez-Martínez, Isabel Martínez-Sempere, Xavier Ivars-Ribes, and Rafael C Carrasco. 2013. An open diachronic corpus of historical Spanish: annotation criteria and automatic modernisation of spelling. *arXiv preprint arXiv:1306.3692*.

Silke Scheible, Richard J. Whitt, Martin Durrell, and Paul Bennett. 2011. A gold standard corpus of early modern german. In *Proceedings of the 5th Linguistic Annotation Workshop*, pages 124–128, Portland, Oregon, USA, June. Association for Computational Linguistics.

Yves Scherrer and Tomaž Erjavec. 2013. Modernizing historical slovene words with character-based smt. In *Proceedings of the 4th Biennial International Workshop on Balto-Slavic Natural Language Processing*, pages 58–62, Sofia, Bulgaria. Association for Computational Linguistics.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152, Kyoto, Japan. IEEE.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Eszter Simon. 2014. Corpus building from old hungarian codices. In *The Evolution of Functional Left Peripheries in Hungarian Syntax*, pages 224–236. Oxford University Press.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., Montréal, Canada.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 6000–6010. Curran Associates, Inc.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning*, pages 2048–2057, Lille, France. PMLR.

Fine-Grained Arabic Dialect Identification

Mohammad Salameh[†] **Houda Bouamor**[†] **Nizar Habash**[‡]

[†]Carnegie Mellon University in Qatar
{msalameh,hbouamor}@qatar.cmu.edu

[‡]New York University Abu Dhabi
nizar.habash@nyu.edu

Abstract

Previous work on the problem of Arabic Dialect Identification typically targeted coarse-grained five dialect classes plus Standard Arabic (6-way classification). This paper presents the first results on a fine-grained dialect classification task covering 25 specific cities from across the Arab World, in addition to Standard Arabic – a very challenging task. We build several classification systems and explore a large space of features. Our results show that we can identify the exact city of a speaker at an accuracy of 67.9% for sentences with an average length of 7 words (a 9% relative error reduction over the state-of-the-art technique for Arabic dialect identification) and reach more than 90% when we consider 16 words. We also report on additional insights from a data analysis of similarity and difference across Arabic dialects.

Title and Abstract in Arabic

التصنيف الدقيق في تحديد اللّهجات العربية

اعتمدت الأبحاث السابقة في مسألة تحديد اللهجات العربية على تصنيف عام يتضمن المناطق العربية (خليجي، عراقي، شامي، مصري، مغاربي) بالإضافة للغة العربية الفصحى. خلافا للتصنيف السابق، يعرض هذا البحث العلمي النتائج الأولى في تحديد اللهجات باستخدام تصنيفات دقيقة تشمل ٢٥ مدينة من جميع أنحاء العالم العربي بالإضافة للعربية الفصحى، مما يزيد المسألة صعوبة. في هذا السياق، نبني عدة أنظمة للتصنيف بين اللهجات من خلال الإستطلاع على الخصائص اللغوية المستخرجة من الجمل. تظهر النتائج التي توصلنا إليها انّه بامكاننا تحديد لهجة المدينة للمتحدث بدقة ٦٧،٩ % من خلال نص كتابي يحتوي على معدل ٧ كلمات، وبدقة ٩٠ % من خلال تحليل ١٦ كلمة. بالإضافة، يتضمن البحث تقريرا مبنيا على تحليل الجمل يبرز مدى التشابه والاختلاف بين اللهجات العربية.

1 Introduction

Dialect identification (DID) is the task of automatically identifying the dialect of a particular segment of speech or text of any size (i.e., word, sentence, or document). This task has attracted increasing attention in recent years. For instance, several evaluation campaigns were dedicated to discriminating between language varieties (Malmasi et al., 2016; Zampieri et al., 2017). This is not surprising considering the importance of automatic DID for several NLP tasks, where prior knowledge about the dialect of an input text can be helpful, such as machine translation (Salloum et al., 2014), sentiment analysis (Al-Twairesh et al., 2016), or author profiling (Sadat et al., 2014).

For Arabic DID, previous work typically targeted coarse-grained five dialect classes plus Standard Arabic at most (6-way classification) (Zaidan and Callison-Burch, 2014; Elfardy and Diab, 2013; Darwish et al., 2014). In this paper, we tackle a finer-grained dialect classification task, covering 25 cities from across the Arab World (from Rabat to Muscat), in addition to Standard Arabic. Table 1 shows the break up we follow in choosing these cities. The table relates the typical five-way regional break up of Arabic dialects (Habash, 2010) to a more refined ten-way sub-region division, and even further into 25 cities.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1332–1344
Santa Fe, New Mexico, USA, August 20-26, 2018.

Region	Maghreb				Nile Basin	Levant		Gulf		Yemen
Sub-region	**Morocco**	**Algeria**	**Tunisia**	**Libya**	**Egypt/Sudan**	**South Levant**	**North Levant**	**Iraq**	**Gulf**	**Yemen**
Cities	**Rabat** (*RAB*) Fes (*FES*)	Algiers (*ALG*)	**Tunis** (*TUN*) Sfax (*SFX*)	Tripoli (*TRI*) Benghazi (*BEN*)	**Cairo** (*CAI*) Alexandria (*ALX*) Aswan (*ASW*) Khartoum (*KHA*)	Jerusalem (*JER*) Amman (*AMM*) Salt (*SAL*)	**Beirut** (*BEI*) Damascus (*DAM*) Aleppo (*ALE*)	Mosul (*MOS*) Baghdad (*BAG*) Basra (*BAS*)	**Doha** (*DOH*) Muscat (*MUS*) Riyadh (*RIY*) Jeddah (*JED*)	Sana'a (*SAN*)

Table 1: Different region, sub-region, and city dialects. The bolded cities are our regional representatives.

We build several classification systems and explore a large space of features. Our results show that we can identify the exact city of a speaker at an accuracy of 67.9% for sentences with an average length of 7 words (a 9% relative error reduction over the state-of-the-art technique for Arabic dialect identification (Zaidan and Callison-Burch, 2014)) and reach more than 90% when we consider 16 words.

We also report the results of training and evaluating our model using datasets obtained from different sources: (i) A large-scale parallel corpus of five regional representative dialects and MSA (CORPUS-6); (ii) A smaller-scale parallel corpus of 25 dialects and MSA (CORPUS-26); and (iii) A corpus of dialectal sentences extracted from Twitter. Furthermore, we report the additional insights we obtain from analyzing the data with respect to similarity and difference across Arabic dialects.

Our research contributions are the following:

- We extend the problem of Arabic DID to predict 25 fine-grained city-level dialects.
- We demonstrate a solution for leveraging relatively rich resources for a small number of city dialects to help with the fine-grained DID task for 25 city dialects.
- We present a detailed analysis of dialect similarity and confusability and add insights on top of the traditional map presented in the literature.
- We show that, on average, it takes 52 words to reach an optimal classification of the dialect and 16 words to reach 90% accuracy.
- We evaluate our system on dialectal sentences extracted from social media.

The remainder of this paper is organized as follows. In section 2, we review the main previous efforts for DID. In Section 3, we present the main challenges in processing Arabic and its dialects. In Section 4, we describe our experimental setup and discuss the datasets, models, features, evaluation metrics used as well as our results. In Section 5, we present a detailed analysis and discussion on dialect confusability, optimal classification and tweet dialect classification. Finally, we conclude and give our future directions in Section 6.

2 Related Work

Working on DID is more challenging than just recognizing a specific language (Etman and Beex, 2015). Since Arabic dialects use the same script and share part of the vocabulary, it is quite arduous to distinguish between them. Hence, developing an automatic identification system working at different levels of representation and exploring different datasets has attracted increasing attention in recent years. Shoufan and Alameri (2015) and Al-Ayyoub et al. (2017) present a survey on NLP and deep learning methods for processing Arabic dialectal data with an overview on Arabic DID of text and speech.

Biadsy and Hirschberg (2009) presented a system that identifies dialectal words in speech and their dialect of origin (on four regional Arabic dialects) from acoustic signals. In the same context, Bougrine et al. (2017) propose a hierarchical classification approach for spoken Arabic Algerian DID, using prosody.

Diab and Elfardy (2012) presented a set of guidelines for token-level identification of dialectness. They later proposed a supervised approach for identifying whether a given sentence is prevalently MSA or Egyptian (Elfardy and Diab, 2013) using the Arabic online commentary dataset (AOC) (Zaidan and Callison-Burch, 2011). Their system (Elfardy and Diab, 2012) combines a token-level DID approach with other features to train a Naive-Bayes classifier. Similarly, Tillmann et al. (2014) use a linear SVM

classifier to label the AOC dataset. Also, El-Haj et al. (2018) used grammatical, stylistic and Subtractive Bivalency Profiling features for dialect identification on the AOC dataset.

Sadat et al. (2014) presented a bi-gram character-level model to identify the dialect of sentences in the social media context among dialects of 18 Arab countries. More recently, discriminating between Arabic Dialects has been the goal of a dedicated shared task (Zampieri et al., 2017; Malmasi et al., 2016), encouraging researchers to submit systems to recognize the dialect of speech transcripts along with acoustic features for dialects of four main regions: Egyptian, Gulf, Levantine and North African, in addition to MSA. The dataset used in these tasks is different from the dataset we use in this work in its genre, size and the dialects covered.

Several systems implementing a range of traditional supervised learning and more advanced deep learning methods were submitted. High-order character n-grams extracted from speech or phonetic transcripts and i-vectors (a low-dimensional representation of audio recordings) were shown to be the most successful and efficient features (Butnaru and Ionescu, 2018), while deep learning approaches (Belinkov and Glass, 2016) did not perform well.

Recently, there are more efforts to collect and annotate datasets for dialect identification. Abdul-Mageed et al. (2018) present a large dataset from Twitter domain covering dialects from 29 major Arab cities in 10 Arab countries. Al-Badrashiny and Diab (2016) present a system that detects points of code-switching in sentences between MSA and dialectal Arabic.

Most, if not all of the approaches, proposed in the literature have been exploring DID at the regional or country level. To the best of our knowledge, this is the first fine-grained DID system covering the dialects of 25 cities from several countries, including cities in the same country in the Arab World. Moreover, this is the first study pinpointing Arabic DID, discussing the difference between regional and city-level identification and redrawing the geographical map for Arabic DID. Furthermore, this is the first work leveraging a parallel corpus covering 25 dialects in addition to MSA (Bouamor et al., 2018).

3 Arabic and its Dialects

Dialectal Arabic (DA) refers to the collection of language varieties used by Arabic speakers in their daily interactions. DA lives side by side with Modern Standard Arabic (MSA), the official language in most Arab countries. Although MSA is not acquired natively (through spoken input at home and in the community), it has an extensive presence in various settings: media, education, business, arts and literature, and official and legal written documents. The dialects are not standardized, they are not taught, and they do not have official status. However, they are the primary vehicles of communication (face-to-face and recently, online) and have a significant presence in the arts as well.

Arabic dialects are often classified in terms of geography. Typical regional groupings cluster the dialects into Levantine Arabic (Lebanon, Syria, Jordan, and Palestine), Gulf Arabic (Qatar, Kuwait, Saudi Arabia, United Arab Emirates and Bahrain, with Iraqi and Omani Arabic included sometimes), Egyptian Arabic (which may include Sudan), North African Arabic (vaguely covering Morocco, Algeria, Tunisia, Libya and Mauritania), and Yemeni Arabic (Habash, 2010). However, within each of these regional groups, there is significant variation down to the village, town, and city levels.

Arabic dialects differ from one another and from MSA on all levels of linguistic representation, from phonology and morphology to lexicon and syntax (Watson, 2007).[1] The number of lexical differences is significant i.e., Egyptian أوضة *ÂwDĎħ* 'room' corresponds to MSA غرفة *γrfħ*, Lybian دار *dAr* and Tunisian بيت *byt* (Habash et al., 2012a).[2] Morphological differences are also quite common. One example is the future marker particle which appears as +س *sa+* or سوف *sawfa* in MSA, +ح*Ha+* or رح *raH* in Levantine dialects and باش *bAš* in Tunisian. This together with the variation in the templatic morphology make the forms of some verbs rather different: e.g., *'I will write'* is سأكتب *sa Âaktubu* in MSA, حاكتب *HaÂaktub*

[1] Comparative studies of several Arabic dialects suggest that the syntactic differences between the dialects are minor (Benmamoun, 2012).

[2] Arabic transliteration is presented in the Habash-Soudi-Buckwalter scheme (Habash et al., 2007): (in alphabetical order) *AbtθjHxdðrzsšSDTĎς γfqklmnhwy* and the additional symbols: ' ء, Â أ, Ă إ, Ā آ, ŵ ؤ , ŷ ئ , ħ ة, ý ى.

in Palestinian, هكتب *haktib* in Egyptian and باش نكتب *bAš niktib* in Tunisian.

An example of phonological differences is in the pronunciation of dialectal words whose MSA cognate has the letter Qaf (ق *q*). It is often observed that in Tunisian Arabic, this consonant appears as /q/ (similar to MSA), while in Egyptian and Levantine Arabic it is /ʔ/ (glottal stop) and in Gulf Arabic it is /G/ (Haeri, 1991; Habash, 2010).

It should be also noted that while MSA has an established standard orthography, the dialects do not. Often people write words reflecting the phonology or the history (etymology) of these words. DA is sometimes written in Roman script (Bies et al., 2014). In the context of NLP, a set of conventional orthography guidelines (CODA) has been proposed, but only for specific dialects (Habash et al., 2018).

Despite these differences, distinguishing between dialects is a very challenging task because: (i) dialects use the same writing script (not in a conventionalized way) and share part of the vocabulary; and (ii) Arabic speakers usually resort to repeated code-switching between their dialect and MSA (Abu-Melhim, 1991; Bassiouney, 2009), creating sentences with different levels/percentages of dialectness. More discussion on the similarity between dialects of 25 cities in the Arab World and MSA is given in Section 4.1.

4 Experimental Setup

4.1 Data

In this work, we use a large-scale collection of parallel sentences built to cover the dialects of 25 cities from the Arab World (illustrated in Table 1), in addition to English, French and MSA (Bouamor et al., 2018). This resource was created as a commissioned translation of the Basic Traveling Expression Corpus (BTEC) (Takezawa et al., 2007) sentences from English and French to the different dialects. It contains two corpora. The first consists of 2,000 sentences translated into dialects of 25 cities. Each of these sentences has a corresponding 25 parallel translations. We refer to it as CORPUS-26 (25 cities plus MSA). The second corpus has 10,000 additional sentences (non-overlapping with the 2,000 sentences) from the BTEC corpus translated to the dialects of only five selected cities: Beirut, Cairo, Doha, Tunis and Rabat. We refer to it as CORPUS-6 (5 cities plus MSA). Effectively, the five selected cities will each have 12,000 sentences that are five-way parallel translations. An example of a 28-way parallel sentence (25 cities plus MSA, English and French) extracted from CORPUS-26 is given in Figure 1.

Data pre-processing and splitting In our experiments, we only tokenize the sentences in both CORPUS-6 and CORPUS-26 using punctuation marks. Morphological analysis has been shown to improve the performance of DID systems for a small number of dialects (Darwish et al., 2014). However, the number and sophistication of morphological analysis and segmentation tools for DA are very limited (Pasha et al., 2014), cover only a small number of dialects (Habash and Rambow, 2006; Habash et al., 2012b; Khalifa et al., 2017) and unavailable for most of the others. We split each corpus into Train, Development (Dev) and Test sets. The splits are balanced for each dialect and the distribution of each split is given in Table 2. We use TRAIN, DEVELOPMENT and TEST terms with CORPUS-6 and CORPUS-26 to refer to the training, development and test sets of the specified corpus. We use the term MODEL to refer to the trained system.

Pairwise similarity between dialects In order to get a sense of the complexity of our task, we explore the degrees of similarity and variation between the dialects in CORPUS-26. We accomplish this by building a similarity matrix representing the lexical similarity between the dialects of every two cities D_1 and D_2 (i.e., BEI and CAI, TUN and MUS, etc.). We measure the similarity by computing the percentage of common tokens between the corpus of D_1 and the corpus of D_2. This is solely a bag of word comparison.

	Train	Dev	Test	Classes
CORPUS-6	9000	1000	2000	6
CORPUS-26	1600	200	200	26

Table 2: Distribution of the Train, Dev and Test sets used in our experiments.

Then, we apply a hierarchical agglomerative clustering algorithm to the similarity matrix. We group the clusters using single linkage clustering, thus combining two clusters that contain the closest pair of

MSA	سوف آخذ هذه ، من فضلك . *swf Āxð hðh , mn fDlk .*
English	I'll take this one, please.
French	Je vais prendre celui-ci, s'il vous plaît.

Rabat	غادي ناخد هادي ، عافاك . *γAdy nAxd hAdy, ςAfAk.*	**Fes**	غادي ناخد هاد الواحد، عافاك. *γAdy nAxd hAd AlwAHd, ςAfAk.*
Algeria	ندي هاذا، من فضلك. *ndy hAðA, mn fDlk.*	**Tunis**	تو ناخو هاذي ، عيشك . *tw nAxw hAðy, ςyšk.*
Sfax	نحب ناخذ هذا، يعيشك. *nHb nAxð hðA, yςyšk.*	**Tripoli**	حناخد هدا ، من فضلك . *HnAxd hdA , mn fDlk .*
Benghazi	حناخذ هضي، لو سمحت. *HnAxð hDy, lw smHt.*	**Cairo**	حاخد ده ، إذا سمحت . *HAxd dh, ĂðA smHt.*
Alexandria	انا هاخد دة، لو سمحت. *AnA hAxd dħ, lw smHt.*	**Aswan**	أنا هاخد ده، لو سمحت. *ÂnA hAxd dh, lw smHt.*
Khartoum	ح اخد الواحد دا، من فضلك. *H Axd AlwAHd dA, mn fDlk.*	**Jerusalem**	رح أخد هدا ، لو سمحت . *rH Âxd hdA , lw smHt .*
Amman	راح آخد هاد، لو سمحت. *rAH Āxd hAd, lw smHt.*	**Salt**	راح اخد وحدة من هاي، لو سمحت . *rAH Axd wHdħ mn hAy, lw smHt .*
Beirut	رح اخد هيدا ، عمول معروف . *rH Axd hydA, ςmwl mςrwf.*	**Damascus**	رح آخد هاد ، إذا سمحت . *rH Āxd hAd , ĂðA smHt .*
Aleppo	بدي آخد هاد، أذا سمحت. *bdy Āxd hAd, ĂðA smHt.*	**Mosul**	غاح اخذ هذا الويحد، رجاءا. *γAH Axð hðA AlwyHd, rjA'A.*
Baghdad	راح اخذ هذا ، رجاء . *rAH Axð hðA , rjA'F .*	**Basra**	راح اخذ هذا، رجاءا. *rAH Axð hðA, rjA'A.*
Doha	باخذ هذي ، لو سمحت . *bAxð hðy, lw smHt.*	**Mascat**	بأخذ هاذي، من فضلك. *bÂxð hAðy, mn fDlk.*
Riyad	باخذ هي ، لو سمحت . *bAxð hy , lw smHt .*	**Jeddah**	حاخذ دا الشيء، لو سمحت. *HAxð dA Alšy', lw smHt.*
Sana'a	شا اشل هذا، لو سمحت. *šA Ašl hðA, lw smHt.*		

Figure 1: Sample of a 28-way parallel sentence extracted from CORPUS-26. The MSA and dialectal sentences are given along with their transliteration in the Habash-Soudi-Buckwalter scheme (Habash et al., 2007).

dialects (have the largest number of common tokens). The dendrogram in Figure 2 illustrates the result of this clustering algorithm, with the y-axis showing the token dissimilarity ratio between the clusters.

The dendrogram shows the not surprising closeness of dialects of cities within the same countries, and in the same geographic region. For example, Damascus and Aleppo dialects are different from each other only by 32% and from Beirut dialect by 38%. While the dissimilarity between the cluster enclosing Tunis and Sfax and the cluster containing the rest of the dialects is more than 50%. Thus, the high degree of similarity among some dialects shows that discriminating between dialects on the word-level is rather challenging. This can affect the accuracy of our models due to the increase of confusability among similar dialects.

4.2 Multi-level Dialect Identification Models

We formulate our DID problem as a multi-class classification task. We consider a Multinomial Naive Bayes (MNB) classifier for the learning task.[3] MNB is a variation of Naive Bayes that estimates the conditional probability of a token given its class as the relative frequency of the token t in all documents belonging to class c. MNB has proven to be suitable for classification tasks with discrete features (e.g.,

[3]Our experiments with MNB outperform other classification models such as Linear SVM, Convolutional Neural Networks models with multiple words and characters filter sizes (Belinkov and Glass, 2016), and Bi-directional LSTM models. The latter two had lower accuracy than the simple Language Model baseline, which could be explained by the small size of our training data.

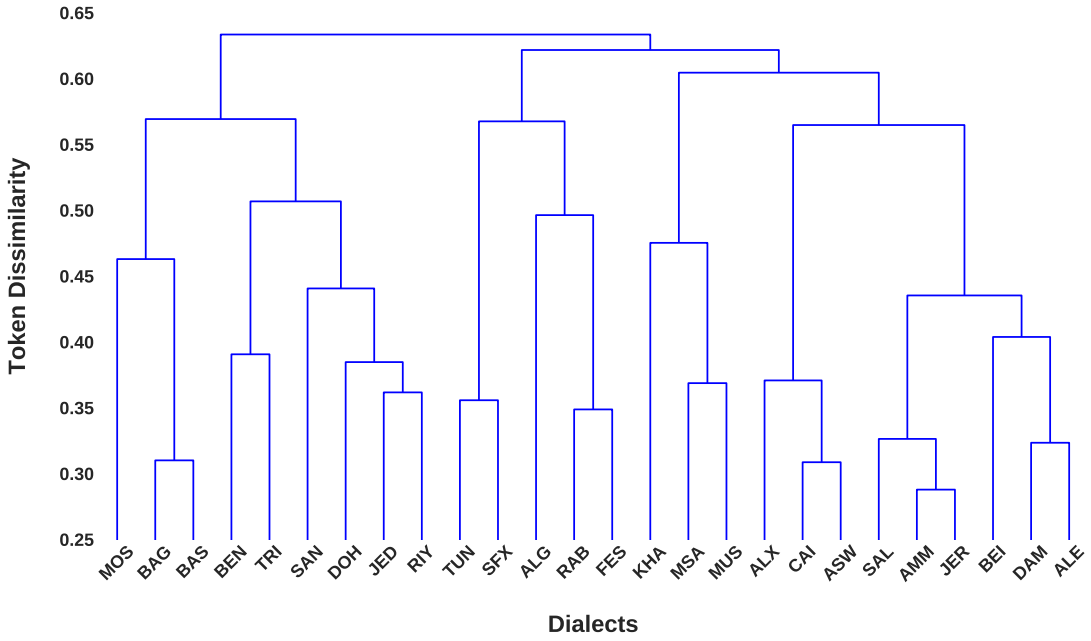


Figure 2: Pairwise similarity between dialects in Corpus-26

word or character counts or representation for text classification) (Manning et al., 2008).

Baseline We follow the approach described in Zaidan and Callison-Burch (2014) for dialect identification and adapt it to build our baseline model. For each dialect, we train a 5-gram character level language model (LM) using KenLM (Heafield, 2011) with default parameters and Kneser–Ney smoothing. Then, we use the LM to assign to each sentence S, the dialect D_i that maximizes its conditional probability score $\text{argmax}_i\, P(D_i|S)$. Character-based LMs leverage subword information and are generally good for capturing particular peculiarities that are specific to certain dialects such as the use of certain clitics, affixes or internal base word structure. For example, the word prefixes أدي *Âdy*, بت *bt* and هالـ *hAl* depicted by the character n-gram LM in أديش بتخدم هالفيزا ؟ *Âdyš btxdm hAlfyzA ? 'How long is this visa good for?'* are good indicators that the dialect of the sentence might be from the Levantine region. Furthermore, character-level models mitigate the ineffectiveness of word-based LMs caused by the presence of out-of-vocabulary words (OOVs) that are prominent in dialects, due to the lack of standard orthography (Habash et al., 2012a).

4.3 Learning Features

We use a suite of features that have been used in works related to DID and text classification. These features are extracted from CORPUS-6 and CORPUS-26 without any preprocessing beyond punctuation tokenization.

Word n-grams Word unigrams are extensively used in text classification tasks. For our task, we extract surface word n-grams ranging from unigrams to 5-grams and use them as features. Word unigrams are useful for our DID task as they depict words unique to some dialects. As shown in Figure 2, lexical variations are prominent and could be predictive for certain regions, countries, and cities.

Character n-grams Character n-grams have shown to be the most effective in language and dialect identification tasks (Zampieri et al., 2017). For DAs, Character n-grams are good at capturing several morphological features that are distinctive between Arabic dialects, especially the clitical and affixal use (as described in section 3). In our experiments, we extract character n-grams ranging from 1-grams to 5-grams. We use Term Frequency-Inverse Document Frequency (Tf-Idf) scores as it has been shown to empirically outperform count weights.

		N-gram Features		Other Features	Corpus-6		Corpus-26	
		Word	**Character**		**Dev**	**Test**	**Dev**	**Test**
a.	Baseline	–	–	Char 5-gram LM	92.2	92.7	66.2	64.7
b.	MNB	–	1		45.9	44.6	18.0	17.1
c.	MNB	–	1+2		70.9	70.4	40.6	37.4
d.	MNB	–	1+2+3		83.8	84.4	55.1	53.5
e.	MNB	–	1+2+3+4		87.3	88.2	60.0	58.2
f.	MNB	–	1+2+3+4+5		88.5	89.3	61.3	59.7
g.	MNB	1	–		90.8	91.1	63.9	63.0
h.	MNB	1+2	–		90.5	91.2	62.5	62.0
i.	MNB	1+2+3	–		90.1	90.9	62.2	61.2
j.	MNB	1+2+3+4	–		90.0	90.8	62.0	61.1
k.	MNB	1+2+3+4+5	–		89.8	90.7	62.0	60.9
l.	MNB	1	1+2+3		90.7	91.1	65.3	63.6
m.	MNB	1	1+2+3	Word 5-gram LM	91.5	91.9	62.6	62.8
n.	MNB	1	1+2+3	Char 5-gram LM	**92.7**	**93.2**	**67.6**	**66.4**
o.	MNB	1	1+2+3	Char/Word 5-gram LM	**93.1**	**93.6**	**68.5**	**67.5**
p.	MNB	1	1+2+3	Char/Word 5-gram LM + Corpus 6 Classifier Prob.	–	–	**68.9**	**67.9**

Table 3: Accuracy on the dev and test sets for both CORPUS-6 and CORPUS-26. The n-grams features show the n-gram orders for word and characters in the MNB models.

Language model probability scores We train n LMs each pertaining to the n dialects, on CORPUS-6 and CORPUS-26. We score the sentences in the TRAIN, DEVELOPMENT and TEST sets, using these LMs. Then, we use the probability scores of the sentence as features. Thus, each sentence will have n probability scores, one for each dialect. The probability scores measure how close each sentence is to the dialect. We experiment using probability scores from word 5-grams LM, character 5-gram LM, and adding both as features.

4.4 Evaluation Metrics

We report the results on CORPUS-6 and CORPUS-26 using the *accuracy* metric, which calculates the percentage of the sentences whose dialect is correctly predicted. We also report the precision, recall and F_1 scores metrics for our best systems on both corpora. The scores are calculated per class for our best system, which can provide a better understanding on the confusability of the classes and sensitivity of our model.

4.5 Results

In this section, we present the results of the different MNB models and compare them to the baseline. In Table 3, we report the results on the DEVELOPMENT and TEST sets for both CORPUS-6 and CORPUS-26 using the accuracy metric. First, we analyze the use of n-grams as features for our MNB model and the effect of increasing the n-gram order on the accuracy of the model. Training the MNB classifier on character n-grams (rows b. to f.) shows an increase in the accuracy on the DEVELOPMENT and TEST set of both corpora, when increasing the n-gram order. A steep increase is observed from unigrams to trigrams, while it diminishes from 4-grams to 5-grams. We hypothesize that the morphological features in the words' structure are well captured within character LMs.

However, increasing the word n-gram order has a negative effect on the system's accuracy (rows g. to k.). It results in a decrease of one and two accuracy points on CORPUS-6 and CORPUS-26 respectively, as we add higher order n-gram features on the top of unigrams. Still, the of use word unigrams features alone (row g.) is able to beat 1-to-5-grams character features (row f.)

We experiment with different combinations of word and character n-grams features. Our best combination is the one using word unigrams with character unigrams, bigrams and trigrams (row l.). Yet, this combination could not outperform the 5-gram LM baseline (row a.) for both CORPUS-6 and CORPUS-26, which emphasizes the power of LMs and align with previous results on language and dialect identification. This important result suggests adding LM probability scores as features to our model. Adding

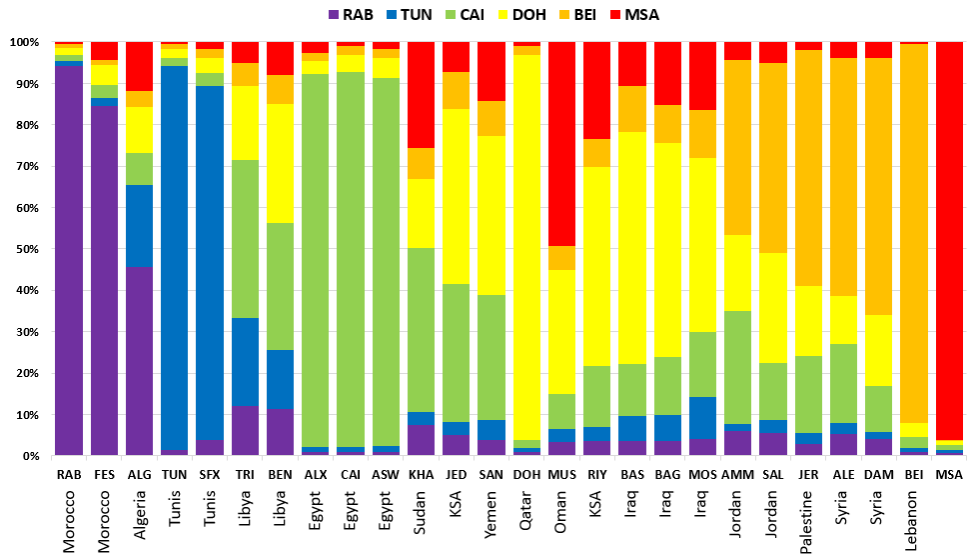


Figure 3: Confusability patterns of our 26 dialects in terms of the best MODEL-6 system.

LM scores (row m.) improves the accuracy on CORPUS-6 while it hurts the system on CORPUS-26. This can be attributed to the small size of CORPUS-26 TRAIN data used to train the 5-gram LM and the large number of classes. However, adding 5-gram character LM scores as features (row n.) beats the baseline scores (row a.). The accuracy is further improved when we include 5-gram word LM scores.

CORPUS-26 TRAIN is considered small with a large number of labels. We can make use of CORPUS-6 MODEL for providing evidence about the region that sentences of CORPUS-26 belong to. Our intuition is that sentences from CORPUS-26 can be weighed by how close to the five main cities and MSA in CORPUS-6. Thus, we train a model on CORPUS-6 TRAIN and run it on CORPUS-26 TRAIN, DEVELOPMENT, and TEST. We use the six probability scores generated by CORPUS-6 MODEL, each corresponding to a probability of the dialect given a sentence from CORPUS-6, as extra features to train CORPUS-26 model. The combination of these features with features from row (o.) resulted in the best performance of our model among all the experiments. Overall, our best CORPUS-6 and CORPUS-26 models achieve a 12% and 9% relative error reduction rate over the character Language Model baseline respectively.

5 Analysis and Discussion

5.1 Dialect Confusability and Identifiability

In this section, we present an analysis of the MODEL-6 *Classifier Probability* features that gave us our best MODEL-26 system results. In Figure 3, we show the average probability distribution of MODEL-6 for the sentences in CORPUS-26 TRAIN. The colors in the columns refer to the probability of assigning a specific MODEL-6 label from the six dialects we consider as *anchors*. The 25 cities are organized in a general West-to-East order, with some exceptions: we start from Rabat in the west and head to Alexandria; then we go up the Nile to southern Egypt and Sudan and jump over the Red Sea to the south of the Arabian Peninsula; then we head north through Iraq and visit the Levant ending in Beirut. MSA is presented at the end by itself. The first thing we observe is that there is general anchor-dialect diffusion pattern: e.g., the Rabat-ness is strong in Rabat, but it fades away in Algiers as more Tunis-ness sets in. Another example is how cities in the South Levant (Amman, Salt, Jerusalem) seem to have less of the Beirut-ness which strongly marks North Levantine cities, and more of Cairo-ness and Doha-ness. These confusability patterns correlate with geography independently of any pre-design of the data sets is a very interesting result. But furthermore, these patterns are valuable as unique *identifying* markers that help distinguish among the fine-grained 26-labels in CORPUS-26. It suggests that in the future as we go into more fine-grained distinctions, we can rely on a small number of anchors to help with identifiability through patterns of confusability.

5.2 Region Level Identification

Dialect	Precision	Recall	F_1
Corpus-26			
MOS	**88**	**83**	**86**
ALG	79	82	81
TRI	83	79	81
ALX	78	77	77
MSA	72	78	75
RAB	76	74	75
SAN	80	71	75
KHA	71	74	73
SFX	72	73	73
FES	74	70	72
TUN	75	69	72
BEI	76	62	69
BEN	73	65	69
DOH	64	76	69
ALE	73	64	68
BAS	69	66	67
BAG	66	65	65
ASW	61	66	63
JED	57	66	61
JER	61	60	61
RIY	56	61	59
AMM	61	56	58
CAI	64	52	57
DAM	47	66	55
SAL	52	59	55
MUS	55	51	53
Average	69%	68%	69%
Corpus-6			
MSA	97	97	97
RAB	96	95	95
TUN	95	94	95
BEI	94	93	94
DOH	91	94	93
CAI	92	92	92
Average	94%	94%	94%

Table 4: Results in terms of Precision, Recall and F_1 of our best model on the CORPUS-26 and CORPUS-6 test sets.

The upper part of Table 4 presents the precision, recall and F1 score for the 25 dialects, in addition to MSA from the best MODEL-26 system, ordered by F1 score. It is interesting to note that the top four cities classified with F1 scores ranging between 77% and 86% (MOS, ALG, TRI, ALX) are not members of CORPUS-6. Also, the dialects of CORPUS-6 keep the same relative order when classified using MODEL-26, that they have in MODEL-6 (See the bottom part of Table 4), but with Cairo lagging behind.

Upon examining the full confusion matrix (which we do not show in this paper), we observed two phenomena: (i) most confusion patterns tend to be bigger within limited geographical regions, e.g., Baghdad is more confused with other Iraqi cities, than with Maghreb (i.e., RAB, FES, etc.) or Egyptian cities (ALX, ASW, CAI); (ii) some cities are predicted a lot more than others, with Damascus being the most predicted (281 times) compared to Cairo (162 times). The most predicted cities tended to come from bigger regions (Levant and Gulf) which are more represented in our data.

In Table 5, we present a reduced confusion matrix in which we collapse our 25 dialects into eight *regions*. The regions are geographically organized and ordered followed by MSA. Naturally, the eight-region scores are higher than the 25-dialect and MSA scores (more coarse, less labels). However, interestingly, the scores are generally higher for smaller regions compared to larger regions. In the future, we will consider different methods for training varying regional granularities.

5.3 Sample Size and Optimal Classification

With more *test input* examples from the same source and the same dialect, the dialect could be determined with higher accuracy. This allows any system to get more evidence that could support the selection of one dialect over others. In this section, we present two experiments to measure the effect of increasing the length of the inputs of our test set by: (i) adding additional sentences, and (ii) adding additional words.

The main goal of these experiments is to answer the following question: how many sentences or words are needed on average to guarantee an optimal classification into a certain dialect? To answer this, we run CORPUS-26 MODEL on CORPUS-26 TEST.

For each sentence having an incorrectly predicted dialect label, we sample a sentence from the subset of sentences belonging to its correct dialect class and append it to it. We keep randomly sampling sentences until the system correctly predicts the right label. Interestingly, on average, it takes 1.52 sentences to predict the right class of a given sentence. With more examples of text by a writer, our system can confidently determine the correct dialect of the writer with a high accuracy reaching 100%.

Our CORPUS-26 has short sentences with an average length of seven words per sentence. The corpus includes sentences that are common among several dialects such as صباح الخير *SbAH Alxyr* 'good morn-

	MAG	TUN	LIB	EGY	GLF	IRQ	LEV	MSA	Match	Predict	Gold	Prec	Rec	F1
MAG 3 (RAB, FES, ALG)	**539**	7	8	10	19	3	10	4	539	591	600	91	90	91
TUN 2 (TUN, SFX)	15	**356**	4	0	10	6	9	0	356	386	400	92	89	91
LIB 2 (TRI, BEN)	8	8	**308**	20	34	2	20	0	308	370	400	83	77	80
EGY 4 (CAI, ALX, ASW, KHA)	8	4	8	**677**	45	8	41	9	677	784	800	86	85	85
GLF 5 (DOH, JED, RIY, SAN, MUS)	10	3	22	37	**798**	33	59	38	798	1046	1000	76	80	78
IRQ 3 (BAG, BAS, MOS)	1	5	3	6	49	**504**	26	6	504	578	600	87	84	86
LEV 6 (BEI, DAM, ALE, JER, AMM, SAL)	7	3	14	29	63	19	**1062**	3	1062	1230	1200	86	89	87
MSA 1 (MSA)	3	0	3	5	28	3	3	**155**	155	215	200	72	78	75

Table 5: Confusion matrix of MODEL-26 over eight geographical regions. Column 1 indicates the label of the region, the number of cities and the labels of the cities. The regions are (in order): Maghreb, Tunisia, Libya, Egypt and Sudan, Gulf, Iraq, Levant, and MSA.

ing'. We want to answer the question of how many words per sentence do we need in order to have an optimal classification. Given that short sentences and common phrases among dialects are less indicative of the dialect, we append the sentences in CORPUS-26 TEST with sentences randomly selected from the set with similar dialect. We continuously append each sentence until the total number of words reaches at least 65 words. We run CORPUS-26 MODEL on the modified test set over several iterations, by considering a fixed number of words at each iteration, starting from a sentence length of 1 until 60. Figure 4 illustrates the accuracy of CORPUS-26 MODEL with respect to the number of words in the sentences. It is important to note that the accuracy increases proportionally to the number of words considered in a sentences. Our system reaches an accuracy of 69.4% (compared to an accuracy of %67.9 on the original test set) using examples with fixed length of six words. We reach a score above 90% when we consider 16 words, while the optimal classification is reached using 51 words. This could be explained by the importance of longer context and its impact on improving the accuracy of the classifier.

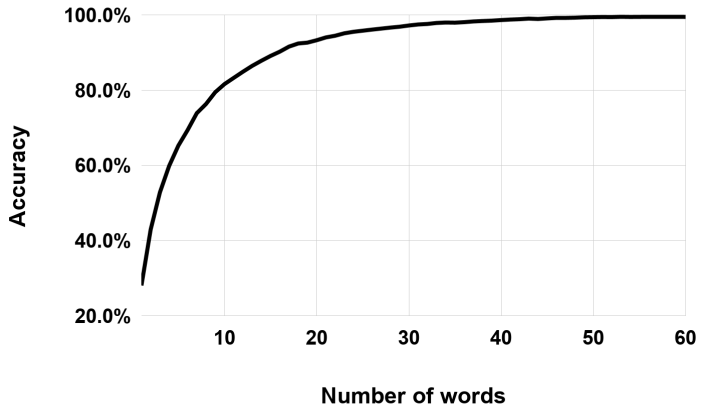


Figure 4: Accuracy on Corpus-26 with respect to the number of words in the input.

5.4 Initial Results on Classifying Tweets

The datasets we use in our experiments (CORPUS-6 and CORPUS-26) are built using a controlled approach, as dialectal translations of English and French sentences in the travel domain (Bouamor et al., 2018). We would like to evaluate the performance of our best CORPUS-6 MODEL on naturally occurring data from social media. For this, we revert to Twitter for collecting Arabic tweets using dialectal function words as seeds to guarantee that the content of the tweet is dialectal. We run CORPUS-6 MODEL on one million tweets. For each of the five cities, we retrieve the top 500 tweets predicted with a probability greater than 0.9. Each of the 500-tweets set is annotated and evaluated by an annotator who is familiar with the dialect. We evaluate the performance of CORPUS-6 MODEL on Beirut, Cairo, Doha, Tunis and Rabat dialects. We obtained an accuracy of 87.0%, 99.6%, 91.4%, 20.2%, 82.6%, respectively for each

city. The high accuracy on Cairo and Doha dialects could be explained by the large number of users who are actively using Twitter in this Arab cities.

As we noticed that the accuracy for Tunis dialect is lower than the other cities, we asked a native speaker to inspect a set of 100 tweets labeled by our system as 'Tunis'. The result obtained showed that 70% of the tweets happen to be tweets from Libya, which is the closest country to Tunisia geographically. Also, this could be explained by the fact that some of the words we consider in the "Tunis" seed list could also be used in Libya, especially that some of the Southern Tunisian dialects are structurally similar to those in close cities in the borders between Tunisia and Libya (Čéplö et al., 2016).

Overall, these initial results are encouraging and suggest a further exploration of Twitter, as it could be mined to extend CORPUS-26 in terms of size, dialects and text genres.

6 Conclusion and Future Work

In this paper, we explored the problem of Arabic DID from the typically studied variety of coarse-grained classification into a finer-grained classification problem covering 25 specific cities from the Arab World, in addition to Modern Standard Arabic (MSA). We presented a detailed analysis of dialect similarity and confusability and added interesting insights on top of the traditional map presented in the literature. We showed that using our best model, we can identify the exact city of a speaker at an accuracy of 67.9% for sentences with an average length of 7 words (a 9% relative error reduction over the state-of-the-art technique for Arabic dialect identification) and reach more than 90% when we consider 16 words. We also showed that a model trained on a commissioned dataset can be used to classify sentences in a corpus of naturally occurring dialectal sentences appearing in social media platforms such as Twitter.

In the future, we plan to explore DID for social media text and improve our model to deal with its complexities. We also plan to experiment on multi-level hierarchical classification, by classifying into regional, subregional and then city level.

Acknowledgements

This publication was made possible by grant NPRP 7-290-1-047 from the Qatar National Research Fund (a member of the Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

Muhammad Abdul-Mageed, Hassan Alhuzali, and Mohamed Elaraby. 2018. You Tweet What You Speak: A City-Level Dataset of Arabic Dialects. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan*.

Abdel-Rahman Abu-Melhim. 1991. Code-switching and Linguistic Accommodation in Arabic. In *Perspectives on Arabic Linguistics III: Papers from the Third Annual Symposium on Arabic Linguistics*, volume 80, pages 231–250. John Benjamins Publishing.

Mahmoud Al-Ayyoub, Aya Nuseir, Kholoud Alsmearat, Yaser Jararweh, and Brij Gupta. 2017. Deep Learning for Arabic NLP: A Survey. *Journal of Computational Science*.

Mohamed Al-Badrashiny and Mona Diab. 2016. LILI: A Simple Language Independent Approach for Language Identification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*.

Nora Al-Twairesh, Hend Al-Khalifa, and Abdulmalik AlSalman. 2016. AraSenTi: Large-Scale Twitter-Specific Arabic Sentiment Lexicons. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany.

Reem Bassiouney. 2009. *Arabic Sociolinguistics*. Edinburgh University Press.

Yonatan Belinkov and James Glass. 2016. A Character-level Convolutional Neural Network for Distinguishing Similar Languages and Dialects. In *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, Osaka, Japan.

Elabbas Benmamoun. 2012. Agreement and Cliticization in Arabic Varieties from Diachronic and Synchronic Perspectives. In Reem Bassiouney, editor, *Al'Arabiyya: Journal of American Association of Teachers of Arabic*, volume 44-45, pages 137–150. Georgetown University Press.

Fadi Biadsy and Julia Hirschberg. 2009. Using Prosody and Phonotactics in Arabic Dialect Identification. In *Proceedings of Interspeech*, Brighton, UK.

Ann Bies, Zhiyi Song, Mohamed Maamouri, Stephen Grimes, Haejoong Lee, Jonathan Wright, Stephanie Strassel, Nizar Habash, Ramy Eskander, and Owen Rambow. 2014. Transliteration of Arabizi into Arabic Orthography: Developing a Parallel Annotated Arabizi-Arabic Script SMS/Chat Corpus. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, Doha, Qatar.

Houda Bouamor, Nizar Habash, Mohammad Salameh, Wajdi Zaghouani, Owen Rambow, Dana Abdulrahim, Ossama Obeid, Salam Khalifa, Fadhl Eryani, Alexander Erdmann, and Kemal Oflazer. 2018. The MADAR Arabic Dialect Corpus and Lexicon. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan.

Soumia Bougrine, Hadda Cherroun, and Djelloul Ziadi. 2017. Hierarchical Classification for Spoken Arabic Dialect Identification using Prosody: Case of Algerian Dialects. *CoRR*, abs/1703.10065.

Andrei Butnaru and Radu Tudor Ionescu. 2018. UnibucKernel: A Kernel-based Learning Method for Complex Word Identification. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, Louisiana.

Slavomír Čéplö, Ján Bátora, Adam Benkato, Jiří Milička, Christophe Pereira, and Petr Zemánek. 2016. Mutual Intelligibility of Spoken Maltese, Libyan Arabic, and Tunisian Arabic Functionally Tested: A Pilot Study. *Folia Linguistica*, 50(2):583–628.

Kareem Darwish, Hassan Sajjad, and Hamdy Mubarak. 2014. Verifiably Effective Arabic Dialect Identification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar.

Mona Diab and Heba Elfardy. 2012. Simplified Guidelines for the Creation of Large Scale Dialectal Arabic Annotations. In *Proceedings of The eighth international conference on Language Resources and Evaluation (LREC 2012)*, Istanbul, Turkey.

Mahmoud El-Haj, Paul Rayson, and Mariam Aboelezz. 2018. Arabic Dialect Identification in the Context of Bivalency and Code-Switching. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan.

Heba Elfardy and Mona Diab. 2012. Token Level Identification of Linguistic Code Switching. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING), IIT Mumbai, India.*

Heba Elfardy and Mona Diab. 2013. Sentence Level Dialect Identification in Arabic. In *Proceedings of the Association for Computational Linguistics*, Sofia, Bulgaria.

Asma Etman and Louis Beex. 2015. Language and Dialect Identification: A Survey. In *Proceedings of The 2015 SAI Intelligent Systems Conference (IntelliSys)*, London, UK.

Nizar Habash and Owen Rambow. 2006. MAGEAD: A Morphological Analyzer and Generator for the Arabic Dialects. In *Proceedings of COLING/ACL*, Sydney, Australia.

Nizar Habash, Abdelhadi Soudi, and Timothy Buckwalter. 2007. On Arabic Transliteration. In *Arabic Computational Morphology*, volume 38 of *Text, Speech and Language Technology*, chapter 2, pages 15–22. Springer.

Nizar Habash, Mona Diab, and Owen Rambow. 2012a. Conventional Orthography for Dialectal Arabic. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 711–718, Istanbul, Turkey.

Nizar Habash, Ramy Eskander, and Abdelati Hawwari. 2012b. A Morphological Analyzer for Egyptian Arabic. In *NAACL-HLT 2012 Workshop on Computational Morphology and Phonology (SIGMORPHON2012)*, Montréal, Canada.

Nizar Habash, Fadhl Eryani, Salam Khalifa, Owen Rambow, Dana Abdulrahim, Alexander Erdmann, Reem Faraj, Wajdi Zaghouani, Houda Bouamor, Nasser Zalmout, Sara Hassan, Faisal Al shargi, Sakhar Alkhereyf, Basma Abdulkareem, Ramy Eskander, Mohammad Salameh, and Hind Saddiki. 2018. Unified guidelines and resources for arabic dialect orthography. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan.

Nizar Y Habash. 2010. *Introduction to Arabic natural language processing*. Morgan & Claypool Publishers.

Niloofar Haeri. 1991. Sociolinguistic Variation in Cairene Arabic: Palatalization and the qaf in the Speech of Men and Women.

Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*.

Salam Khalifa, Sara Hassan, and Nizar Habash. 2017. A Morphological Analyzer for Gulf Arabic Verbs. *The Third Arabic Natural Language Processing Workshop WANLP 2017*.

Shervin Malmasi, Marcos Zampieri, Nikola Ljubešić, Preslav Nakov, Ahmed Ali, and Jörg Tiedemann. 2016. Discriminating between Similar Languages and Arabic Dialect Identification: A Report on the Third DSL Shared Task. In *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, Osaka, Japan.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan M Roth. 2014. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC), Reykjavik, Iceland*.

Fatiha Sadat, Farzindar Kazemi, and Atefeh Farzindar. 2014. Automatic Identification of Arabic Language Varieties and Dialects in Social Media. In *Proceedings of the Second Workshop on Natural Language Processing for Social Media (SocialNLP)*, Dublin, Ireland.

Wael Salloum, Heba Elfardy, Linda Alamir-Salloum, Nizar Habash, and Mona Diab. 2014. Sentence Level Dialect Identification for Machine Translation System Selection. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, Baltimore, Maryland, USA.

Abdulhadi Shoufan and Sumaya Alameri. 2015. Natural Language Processing for Dialectical Arabic: A Survey. In *Proceedings of the Second Workshop on Arabic Natural Language Processing*, Beijing, China.

Toshiyuki Takezawa, Genichiro Kikui, Masahide Mizushima, and Eiichiro Sumita. 2007. Multilingual Spoken Language Corpus Development for Communication Research. *Computational Linguistics and Chinese Language Processing*, 12(3):303–324.

Christoph Tillmann, Saab Mansour, and Yaser Al-Onaizan. 2014. Improved Sentence-Level Arabic Dialect Classification. In *Proceedings of the First Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects*, Dublin, Ireland.

Janet Watson. 2007. *The Phonology and Morphology of Arabic*. Oxford University Press.

Omar F. Zaidan and Chris Callison-Burch. 2011. Crowdsourcing Translation: Professional Quality from Non-Professionals. In *Proceedings of ACL*, Portland, Oregon, USA.

Omar F. Zaidan and Chris Callison-Burch. 2014. Arabic Dialect Identification. *Computational Linguistics*, 40(1):171–202.

Marcos Zampieri, Shervin Malmasi, Nikola Ljubešić, Preslav Nakov, Ahmed Ali, Jörg Tiedemann, Yves Scherrer, and Noëmi Aepli. 2017. Findings of the VarDial Evaluation Campaign 2017. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Valencia, Spain.

Who Feels What and Why?
Annotation of a Literature Corpus with Semantic Roles of Emotions

Evgeny Kim and **Roman Klinger**
Institut für Maschinelle Sprachverarbeitung
University of Stuttgart, Pfaffenwaldring 5b, 70569 Stuttgart, Germany
evgeny.kim@ims.uni-stuttgart.de
roman.klinger@ims.uni-stuttgart.de

Abstract

Most approaches to emotion analysis in fictional texts focus on detecting the emotion expressed in text. We argue that this is a simplification which leads to an overgeneralized interpretation of the results, as it does not take into account who experiences an emotion and why. Emotions play a crucial role in the interaction between characters and the events they are involved in. Until today, no specific corpora that capture such an interaction were available for literature. We aim at filling this gap and present a publicly available corpus based on Project Gutenberg, REMAN (Relational EMotion ANnotation), manually annotated for spans which correspond to emotion trigger phrases and entities/events in the roles of experiencers, targets, and causes of the emotion. We provide baseline results for the automatic prediction of these relational structures and show that emotion lexicons are not able to encompass the high variability of emotion expressions and demonstrate that statistical models benefit from joint modeling of emotions with its roles in all subtasks. The corpus that we provide enables future research on the recognition of emotions and associated entities in text. It supports qualitative literary studies and digital humanities. The corpus is available at http://www.ims.uni-stuttgart.de/data/reman.

Title and Abstract in German

Wer fühlt was und warum?
Annotation eines Literaturkorpus mit Semantischen Rollen von Emotionen

Die meisten Ansätze in der Emotionsanalyse in Literatur beschränken sich auf die Erkennung der Emotion. Wir nehmen in dieser Arbeit an, dass dies eine starke Vereinfachung darstellt. Es wird ignoriert, welche Figur die Emotion empfindet und wodurch sie ausgelöst wurde. Dies ist ungünstig, da Emotionen eine entscheidende Rolle bei der Interaktion zwischen Figuren und mit Ereignissen spielen. Allerdings war bisher kein annotiertes Korpus verfügbar, welches all diese Komponenten erfasst. In diesem Aufsatz präsentieren wir das Korpus REMAN (Relational EMotion ANotation), welches diese Lücke füllt. Es basiert auf Ausschnitten von Texten aus dem Projekt Gutenberg, welche auf Phrasenebene mit Emotionen sowie dem Empfindenden, dem Ziel sowie der Ursache der Emotion annotiert sind. Wir präsentieren eine Analyse des Korpus und stellen erste Ergebnisse eines automatischen Vorhersagemodells vor, welches die Grenzen von Wörterbuch-Verfahren aufzeigt. Des Weiteren zeigen wir, dass statistische Modelle von einer gemeinsamen Modellierung der verschiedenen Teilaufgaben profitieren. Unser Korpus unterstützt die Literaturwissenschaften sowie digitalen Geisteswissenschaften und ermöglicht die Erstellung von Modellen zur feingranularen automatischen Vorhersage von Emotionen. Das Korpus ist verfügbar unter http://www.ims.uni-stuttgart.de/data/reman.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1345–1359
Santa Fe, New Mexico, USA, August 20-26, 2018.

1 Introduction

Emotions are one of the crucial aspects of compelling narratives (Oatley, 2002; Ingermanson and Economy, 2009; Hogan, 2015). Not only do emotions help readers in literature comprehension (Barton, 1996; Robinson, 2005) but they also improve readers' abilities of empathy and understanding of others' lives (Mar et al., 2009; Kidd and Castano, 2013). This makes literature an interesting resource for the study of emotions, hence there is a growing interest in emotion-oriented text analysis among digital humanities scholars.

Emotion analysis and classification is a challenging task which has mostly been tackled with comparably straight-forward approaches, at least in literary studies. For instance, Kim et al. (2017) and Reagan et al. (2016) show that emotions, recognized with dictionaries or bag-of-words models, serve as features for genre classification in fiction, however, only with limited performance. One reason is, presumably, that such approaches assume linearity of the story and ignore the semantic role structure of emotions: who feels the emotion and why, what caused the emotion, what is the target of it (Scarantino, 2016; Russell and Barrett, 1999). Consider the sentence "*Jack is afraid of John because John has a knife*". Following structural approaches to defining emotional episodes, the sentence can be rephrased as "emotion of fear is experienced by Jack (experiencer) because John (target) has a knife (cause)". Here, dictionary-based or bag-of-words approaches would probably capture that this sentence describes fear, however, would fail in attributing correct semantic roles to John and Jack and we would be forced to assume that their emotional experiences are equal, which is not the case.

To overcome these limitations of dictionary-based and bag-of-words approaches to emotion recognition from literary text, we contribute the corpus REMAN (Relational Emotion Annotation). To the best of our knowledge, this is the first dataset of literary excerpts which has annotations for emotions on a phrase level, for experiencers of each emotion, and for their targets and causes. Our work loosely follows the concept of directed emotions, as defined in FrameNet (Fillmore et al., 2003), and extends the work of Ghazi et al. (2015), who focus on detecting emotion stimulus in the FrameNet exemplary sentences annotated for emotions and causes. Our study is different in terms of the type of texts used for the annotation and the conceptualization of certain emotion components.

Our main contributions are therefore: (1) We present the first resource of fictional texts annotated for emotions, experiencers, causes, and targets. (2) We show that emotion annotation that takes into account not only strong emotion indicators ("afraid"), but also implicit emotions ("shaking fingers") is valuable for the study of the language of emotions. (3) We provide results of baseline models to predict emotion words and roles separately and (4) show that the prediction performance of all subtasks benefits from joint prediction of experiencer, emotion words, and targets.

2 Related Work

Emotions have strong linguistic markers that define the tone of the text (Johnson-Laird and Oatley, 1989). This allows for different granularities of emotion annotation. The corpus which originates from the ISEAR project (Scherer and Wallbott, 1994) is an example of document-level annotation that includes descriptions of situations in which respondents had experienced various emotions. Examples of sentence-level annotations include the work by Alm et al. (2005), who annotate a corpus of children stories, and Strapparava and Mihalcea (2007), who label news headlines, but without specifying the textual markers of emotion. An early work which includes textual markers of emotions is Aman and Szpakowicz (2007), who annotate blogposts. Wiebe et al. (2005) annotate a corpus of news articles with emotions at a word and phrase level.

Recent works have mainly diverged from plain emotion annotation, following the idea of emotion theorists (Russell, 2003, *i.a.*) that causes of emotions are inseparable from emotions: Russo et al. (2011) build a corpus of Italian newspaper articles annotated with emotion key words and emotion cause phrases. Both Mei et al. (2012) and Gui et al. (2016) construct emotion-cause-annotated corpora for Chinese. Chen et al. (2010) adopt a rule-based approach based on linguistic patterns to detect emotion causes in the annotated Chinese corpus. Gui et al. (2017) present a question-answering approach to emotion cause extraction, also for Chinese.

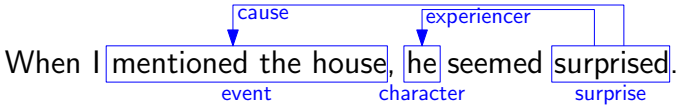


Figure 1: Example annotation from Hugo (1885), with one character, an emotion word, and event and cause and experiencer annotations.

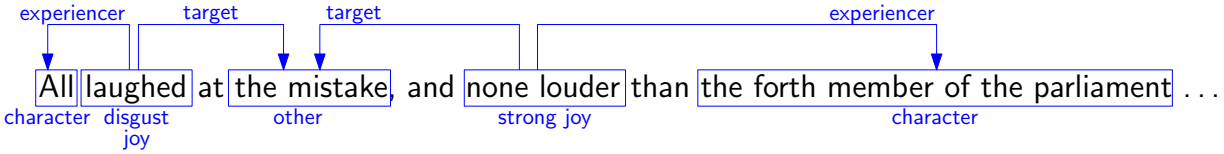


Figure 2: Example annotation from Stimson (1943), with two characters who are experiencers of different emotions. Disgust and joy are annotated as a mixture of emotions. Both emotions have the same target.

Fewer works exist for English. Neviarouskaya and Aono (2013) annotate 500 sentences from an online forum with experiencer, emotion, and emotion cause and present a method for extracting linguistic relations between an emotion and its cause. Ghazi et al. (2015) collect exemplary sentences from FrameNet that have cause annotation and implement a model that extracts the causes of emotions. Following a similar approach, Mohammad et al. (2014) annotate Tweets for semantic roles.

Conceptually, our work partially overlaps with the FactBank corpus (Saurí and Pustejovsky, 2009), where "who thinks what" is taken into account as well. However, in contrast to FactBank, we do not predefine event-selecting predicates for emotion causes and targets, as those are defined by the annotators. In this sense, our work is also different from aspect-based sentiment analysis, where aspects of reviewed products are often predefined.

3 Annotation Task

The goal of the REMAN annotation project is to create a dataset of excerpts from fictional texts that are annotated for the phrases that lead to the association of the text with an emotion, the experiencer of the emotion (a character in the text, if mentioned), the target and the cause of the emotion, if mentioned (*e. g.*, an entity, or event). An example of such an annotation is shown in Figures 1 and 2. As it can be seen from these depictions, each annotation includes textual span annotations such as emotions, characters, events, as well as relation annotations that establish relations between different text spans (cause, experiencer, target). In the following, we describe the conceptual background for each annotation layer in detail. The complete annotation guidelines are available online together with the corpus.

3.1 Phrase Annotation

3.1.1 Emotion

We conceptualize emotions as one's experience that falls in the categories in Plutchik's classification of emotions, namely *anger*, *fear*, *trust*, *disgust*, *joy*, *sadness*, *surprise*, and *anticipation*. In addition, we allow annotation with the class *other emotion* that covers cases when the emotion expressed in the text cannot be reliably categorized into one of the predefined eight classes. A list of the emotions along with example realizations can be found in Appendix A, Table 5.

Annotators are instructed to prefer span annotations of key words (*e. g.*, "afraid"), except cases when emotions are only expressed with a phrase (*e. g.*, "tense and frightened") or indirectly (*e. g.*, "the corners of her mouth went down"). Additionally, emotion spans are marked to be intensified (*i. e.*, amplified), diminished (*i. e.*, downtoned) and negated without marking the modifier or including the modifier. Each span is associated with one or more emotions (exemplified in Figure 2).

3.1.2 Entity

We conceptualize entities as mentions of something that has a clear identity of a person, object, concept, state, or event (see Table 6 in Appendix A). Entities are only annotated if they are experiencer, cause, or target of an emotion.

Character An entity that acts as a character in the text. Character annotation should not omit important information (*e. g.*, the annotation of "the man with two rings of the Royal Naval Reserve on his sleeve" is preferred over only annotating "the man").

Event An event is an occasion or happening that plays a role in the text. Events can be expressed in many ways (see Table 6 in Appendix A for examples from the annotated dataset) and annotators are instructed to label the entire phrases including complementizers or determiners.

Other This is an umbrella concept for everything else that is neither a character nor an event, but fills as relation, described in the following.

3.1.3 Relation Annotation

Relations are semantic links between an emotion and other text spans and can be of type *experiencer*, *cause*, and *target*. In addition, we partially annotate *coreferences* to link personal pronouns to proper nouns. All relations, except Coreference, can only originate from the emotion annotations.

Experiencer The experiencer relation links an emotion span and entity of type *character* who experiences the emotion. If the text contains multiple emotions with multiple experiencers, they all are subject to relation annotation.

Target The target relation links an emotion span and entity of any type towards which the emotion experienced by the experiencer is directed. If there are multiple targets of the emotion, then all of them should also be included in the relation annotation. See Figure 2 for the example of a target annotation.

Cause The cause relation links an emotion span and entity of any type, which serves as a stimulus, something that evokes the emotional response in the experiencer. If there are multiple causes for the emotion, then all of them are included in separate relation annotations.

Coreference The annotators are instructed to annotate as an experiencer the character that is the closest to the emotion phrase in terms of token distance. If the closest mention of the character is a pronoun and the text provides a referent that has a higher level of specificity than the pronoun (*i. e.*, a proper noun or a noun denoting a group or class of objects), then the annotators are asked to resolve the coreference.

4 Corpus Construction and Annotation

4.1 Selection

The corpus of 200 books is sampled from Project Gutenberg[1]. All books belong to the genre of fiction and were written by authors born after the year 1800. More detailed information on the distribution of authors and genres can be found in Appendix B.

We sample consecutive triples of sentences from this subsample of books. A triple is accepted for inclusion for annotation if the middle sentence includes a word from the NRC dictionary (Mohammad and Turney, 2013). We consider this middle sentence the target sentence and the annotators are instructed to label emotions in this second sentence only. Experiencers, causes and targets are annotated in the whole sentence triple if they refer to an emotion in the target sentence.

The sampling procedure is motivated by our observation that triples of sentences sampled with emotion dictionary show the best coverage in terms of the roles that are associated with the emotion. Ghazi et al. (2015) annotate only one sentence and speculate whether adding one sentence before and after will lead to better results. To check their hypothesis, we conduct a small pre-study experiment by extracting 100 random sentences from the Project Gutenberg with the NRC dictionary and analyze how often the roles of experiencer, cause, and target are found in the target sentence and in the window of up to five sentences before and after. The analysis shows that 98% of the texts include the experiencer in the target

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

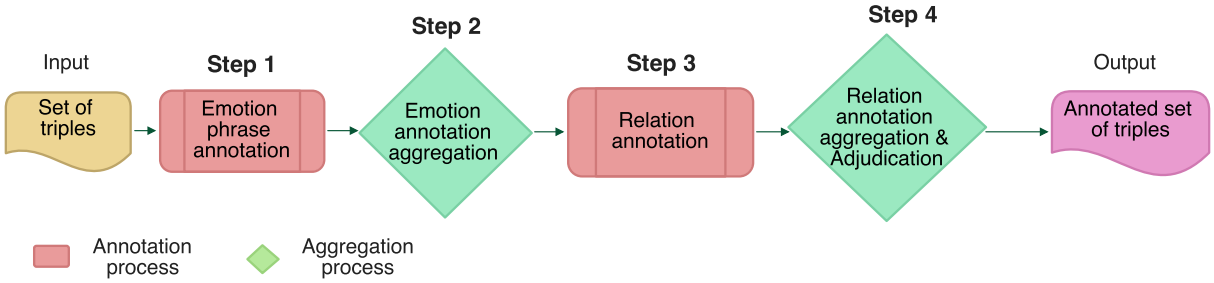


Figure 3: A visualization of the multi-step annotation process.

sentence, while cause and target is found in the target sentence in 67% of the texts. Another 29% of the texts include cause and target in the window of one sentence before and after the target sentence. The remaining texts include cause and target in the window of two (2%), three (1%), and four (1%) sentences around the target sentence. We therefore opt for three-sentence spans as they provide enough information regarding "who feels what and why" without creating unnecessary annotation overhead (cumulatively, 96 % of cause and target are found in such sentence triples).

4.2 Annotation Procedure

The annotations were generated in a multistep process, visualized in Figure 3. The people involved in the annotation were either *annotators* or *experts*, whose roles did not overlap. The annotations (of spans and relations) were performed by three graduate students of computational linguistics (two native English speakers, one non-native speaker) within a three-month period. Arising questions were discussed in weekly meetings with the experts (the authors of the paper) and the results documented in the annotation guidelines. We used WebAnno[2] (Yimam et al., 2013) as annotation framework. In the following, we discuss the four steps of generating the corpus.

Step 1: Emotion phrase annotation The *annotators* were asked to first decide whether the text expresses an emotion and which emotion that is. If any exists, they label the phrase, which led to their decision. The annotators were instructed to search for emotions that are expressed either as single words or phrases.

Step 2: Emotion phrase aggregation In the previous step, each annotator generates set of annotations. In this step, the *expert* heuristically aggregates all spans that overlap between annotators in a semi-automatic process: Concrete emotions are preferred over the "other-emotion", annotations with modifier are preferred over annotations without, and shorter spans are preferred over longer spans. Overlapping annotations with different emotion labels are all accepted.

Step 3: Relation annotation *Annotators* are given the same texts they annotated for emotions in Step 1 with the aggregation from Step 2 from all annotators. Therefore, all annotators see the same texts and annotations as input in this step. For each emotion, the task is to annotate entities that are experiencers, targets, or causes of the emotion and establish relations between them. The annotators were instructed to tag only those entities that have a role of an experiencer, cause, and target. The decision on the entity and relation annotation is made simultaneously: For each emotion the annotators find who experiences the emotion (which *character*) and why (because of event, object, or other character).

Step 4: Relation annotation aggregation and adjudication This final step is a manual *expert* step: Aggregate the relation annotations provided by the annotators. Heuristically, we prefer shorter spans for entities, but guide ourselves with common sense. For instance, consider the phrase "[. . .] *wishing rather to amuse and flatter himself by merely inspiring her with passion*". "*Wishing*" is labelled as emotion. One annotator tagged "*to amuse and flatter himself by merely inspiring her with passion*" as event, another tagged only "*by merely inspiring her with passion*", which is incomplete, as the target of the emotion is the act of amusing and flattering oneself.

Note that we do not discard the rejected annotations but publish all annotations of all annotators.

[2] https://webanno.github.io/webanno/

		a ↔ b			b ↔ c			a ↔ c		
	Type	κ	strict F_1	fuzzy F_1	κ	strict F_1	fuzzy F_1	κ	strict F_1	fuzzy F_1
Emotion	anger	.25	25	39	.15	15	38	.18	18	33
	anticipation	.09	9	23	.07	7	20	.18	18	39
	sadness	.32	32	41	.22	23	41	.19	20	29
	joy	.38	39	50	.40	40	55	.28	28	44
	surprise	.26	26	43	.22	23	33	.27	27	37
	trust	.17	17	26	.14	14	21	.12	13	32
	disgust	.23	23	41	.10	10	26	.19	19	31
	other	.07	7	7	.06	6	11	.08	8	22
	fear	.35	35	48	.28	28	35	.28	28	41
Entity	character	.63	63	68	.48	49	51	.48	48	54
	event	.29	31	60	.09	10	30	.32	34	44
	other	.11	12	28	.11	11	18	.20	21	23
Relation	experiencer		65	73		48	57		46	55
	cause		20	28		34	39		26	32
	target		27	36		18	29		14	28

Table 1: Pairwise inter-annotator agreement for the phrase annotation and relation annotation. F_1 is in %. Regarding the relation scores, in strict F_1, a TP holds if the relation annotation is the same and the entity it points to has the same label and span. In fuzzy F_1, a TP holds if the relation annotation is the same and the entity it points to is the same, but the span boundary of the entity is not necessarily the same.

5 Results

In the following, we first discuss annotation statistics and then provide results of baseline models trained on our resource.

5.1 Inter-annotator Agreement and Consistency of the Annotations

We use pairwise Cohen's Kappa coefficient (κ) on the token level and F_1 on a phrase level with exact and fuzzy match to calculate the agreement of the phrase annotation and F_1 to estimate the agreement of the relation annotation. For F_1 calculation, we use two approaches: strict that requires labels and spans to be identical, and fuzzy that accepts an annotation to be a true positive if the annotations of two annotators overlap by at least one token. Table 1 reports the agreement scores for emotion, entity, and relation annotations between each pair of annotators.

Joy has the highest number of instances (336) and the highest agreement scores (average $\kappa = 35$), followed by *fear* ($\kappa = 0.30$) and *sadness* ($\kappa = 0.24$). *Other emotion* has the lowest agreement with average $\kappa = 0.07$. For entity annotation, especially for *character* annotation, the agreement is higher, with the highest agreement between two annotators being $\kappa = 0.63$. The agreement on the *event* and *other* entities is low ($\kappa = 0.23$ and 0.14 and $F_1 = 25$ and 14, respectively). This is presumably the case because event annotations are often comparably long. This also holds, to a lower extend, for *character* annotations. If we allow partial overlaps to count as a match, the average F_1 increases to 57 for *character* (an increase of 4 percentage points (pp)), 44 for *event* (increase by 19 pp), and 23 for *other* category (increase by 9 pp).

For relation annotations, higher agreement scores are also observed with fuzzy evaluation (F_1 increase for *experiencer*, *cause* and *target* by 10 pp, 7 pp, and 12 pp respectively). These results are in line with previous studies on emotion cause annotation (Russo et al., 2011), and show that disagreements mainly come from the different spans of the entities, though they overlap.

5.2 Difficulties with Measuring IAA

As we showed in Section 5.1, the agreement across all annotation layers is comparably low. There are several reasons for that. Indeed, emotion annotation is highly subjective, but it is not the only subjective category. The cause and target of the emotion are not always clearly recognizable in the text and are also

	Type	Total	Adjudic.	Modifier strong	weak	neg.	Annotation Length 1 token		≥ 2 token		in NRC1		in NRC2	
Emotions	anger	192	156	5	12	7	106	68%	50	32%	36	33%	11	22%
	anticipation	248	201	5	3	11	161	80%	40	20%	28	17%	3	8%
	disgust	242	190	2	7	14	144	76%	46	24%	74	51%	16	34%
	fear	254	183	11	16	17	145	79%	38	21%	93	64%	20	52%
	joy	434	336	31	20	28	289	86%	47	14%	184	64%	29	61%
	sadness	307	224	10	2	13	168	75%	56	25%	100	59%	30	53%
	surprise	243	196	12	4	7	156	80%	40	20%	105	67%	19	47%
	trust	264	232	3	3	33	191	82%	41	18%	66	34%	26	63%
	other emotion	432	207	4	4	4	133	64%	41	36%	52	39%	0	0%
Entities	character	2072	1715				1288	75%	427	25%				
	event	858	615				38	6%	577	94%				
	other	771	485				114	24%	371	76%				

Table 2: Corpus statistics for emotions annotations. Columns indicate the number of times each emotion was annotated. "in NRC1" shows how many of 1 token annotations are in the NRC dictionary (percentage is given relative to 1 token annotations). "in NRC2" shows how many multi-word annotations include at least one word from NRC.

			Emotion that triggered the relation									Entities involved		
Relation	Total	Adjudicated	anger	anticip.	disgust	fear	joy	other	sadness	surprise	trust	char.	event	other
experiencer	2113	1717 48%	137	164	130	173	309	210	216	171	207	1704		
cause	1261	840 24%	48	45	70	95	174	74	134	125	75	87	398	343
target	1244	1017 28%	106	129	125	96	135	121	62	80	163	444	315	257
overall relations	4618	3574 77%	291	338	325	364	618	405	412	376	445	2238	717	601

Table 3: Corpus statistics for relation annotation. Columns indicate the number of times each role was assigned to an entity and how often the respective emotions are in relation to the entity.

subjective categories (two annotators may find two different causes for the same emotion), hence the low agreement scores across all categories. The only exception are *experiencer* annotations, which are the most reliable among all annotations and match the substantial agreement scores of character annotation (the only type of entities that can be involved in an experiencer relation).

We illustrate the difficulties the annotators face when annotating emotions with roles with the following example: "they had never seen ... what was really hateful in his face; ... they could only express it by saying that the arched brows and the long emphatic chin gave it always a look of being lit from below ..." All annotators agree on the character ("they") and the emotion ("hateful" expressing disgust). Similarly, both annotators agree that the disgust is related to properties of the face which is described, however, one annotator marks "his face" as target, the other marks the more specific but longer "the arched brows and the long emphatic chin gave it always a look of being lit from below" as cause.

If we abstract away from the text spans, both annotators agree that the emotion of disgust has something to do with "his face", however they disagree on the target annotation and the cause annotation. So, though conceptually, the annotations by two people are similar, this is not captured by our calculation of inter-annotator agreement.

5.3 Corpus Details

Tables 2 and 3 show the total number of annotations for each category. The REMAN corpus consists of 1720 sentence triples, 1115 of which include an emotion. For the emotion category, *joy* has the highest number of annotations, while *anger* has the lowest number of annotations. In most cases, emotion phrases are single tokens (*e. g.*, "monster","irksome"), out of which 47% on average are found in the NRC dictionary. *Other emotion* has the largest proportion of annotations that span more than one token (36% out of all annotations in this category), which is in line with our expectation that lower levels of specificity for emotion annotation make it more difficult to find a single token that indicates an emotion.

For entities, *character* has the highest number of annotations. As one can see, the *experiencer* relation dominates the dataset (48%), followed by *target* (28%) and *cause* (24%) relations. Note that each character can experience more than one emotion, hence the difference between the number of characters and the experiencers. Table 3 also shows how many times each emotion triggered certain relation. In this sense, *joy* has triggered the most *experiencer* and *cause* relations, which is still related to the prevalence of the annotations for this emotion in the dataset.

6 Baseline Model

We provide a baseline for automatically predicting the structures we annotated. For this first model, we map the relations to span prediction tasks. This is feasible because characters, entities, and other were only annotated if they fill one of the roles, *experiencer*, *target*, or *cause*. Therefore, the prediction task boils down to a sequence prediction task of emotion phrases (for the different emotions) and the potential mentions of experiencers, targets, and causes. Note that we lose the actual relation information in this simplification.

Consider the example depicted in Figure 1: The phrase "I mentioned the house" is labelled as an event and is assigned a role of a *cause* for the emotion of *surprise*, and the word "he" is labelled as a character and is assigned a role of an *experiencer* of the same emotion. We represent these relationships by tagging "I mentioned the house" as *cause* and "he" as *experiencer* using inside-outside-beginning (IOB) encoding capturing the text spans that are linked by relations with an emotion.

We use two sequence labelling models, conditional random fields (CRF) (Lafferty et al., 2001) and bidirectional long short-term memory networks with a CRF layer (biLSTM-CRF), which both provide a good performance in sequence prediction tasks (Benikova et al., 2014; Huang et al., 2015). In addition, we analyze the difficulty of predicting the emotion for a full sentence triple, independent of segments. In the following, we further specify the experimental setting in detail.

6.1 Experimental setting

Experiment 1: Coarse-grained emotion classification In this experiment, the task is to classify the emotions which occur in the sentence triple which forms the instance under consideration. This is therefore a coarse abstraction of the structured prediction tasks presented in this paper. However, this constitutes the most straight-forward task in emotion analysis. We use a dictionary-based approach and a bag-of-words-based classifier.

For the dictionary-based classification, we take the intersection between the words in the triple and NRC dictionary and assign the triple with the corresponding emotion labels. The F_1 score is calculated by comparing the set of labels predicted by dictionaries against the set of gold labels for each triple. The gold labels come from the annotation of words and phrases within each triple. For the BOW approach, we convert each triple into a sparse matrix using all words in the corpus as features. We then classify the triples with a multi-layer perceptron with three hidden layers, 128 neurons each, with an initial learning rate of 0.01 that is divided by 5 if the validation score does not increase after two consecutive epochs by at least 0.001.

Experiment 2: Fine-grained emotion and role detection In this experiment, we evaluate the performance of fine-grained emotion and role (experiencer, target, and cause) prediction in a sequence labelling fashion, as described above. We instantiate separate CRF and biLSTM-CRF models for each relation, as

Category	Annotations	Exp	Model	Features	Strict			Fuzzy		
					P	R	F_1	P	R	F_1
Emotion	1925	1	Rule-based	dictionary	19	83	31			
		1	MLP	BOW	55	21	31			
		2	CRF	all + dictionary	56	6	11	56	6	11
		3	CRF	all + dictionary + experiencer	55	9	16	69	12	20
		2	biLSTM-CRF	embeddings	57	35	43	62	39	48
Experiencer	1717	2	CRF	all + person	50	2	4	50	2	4
		3	CRF	all + person + emotion	74	15	24	78	15	26
		2	biLSTM-CRF	embeddings	49	21	30	49	21	30
Target	1017	3	CRF	all + emotion	50	3	6	50	3	6

Table 4: Results in % for the baseline experiments. F_1 for *cause* with CRF and biLSTM-CRF and for *target* with biLSTM-CRF is zero and therefore not shown here. The column Exp refers to the experimental settings described in Section 6.1.

some annotations overlap (*e. g.*, experiencers can also be targets/causes). The CRF uses part-of-speech tags (detected with spaCy[3] (Honnibal, 2013)), the head of the dependency, if it is capitalized, and offset conjunction with the features of previous and succeeding words as features. For the *emotion* category, we use the presence in the NRC dictionary in addition and, for *experiencer*, the presence in a list of English pronouns. We train for 500 iterations with L-BFGS (Liu and Nocedal, 1989) and L1 regularization.

The biLSTM-CRF model uses a concatenated output of two biLSTM models (one trained on word embeddings with dimension 300, and one trained on character embeddings from the corpus with dimension 100) as an input to a CRF layer. The word embeddings that we use as input are pre-trained on Wikipedia[4] using *fastText*. We use Adam as activation function, a dropout value of 0.5, and train the model for 100 epochs with early stopping if no improvement is observed after ten consecutive epochs.

Experiment 3: Potential for joint modelling of emotion and role prediction The goal of this experiment is to understand if joint modelling of relations has the chance to contribute over learning each relation separately. To that end, we analyze the potential interactions between predictions with gold labels of all other predictions. Specifically, when training our models, we provide the classifier with the information which sequence of tokens is an experiencer (in the case of emotion phrase prediction) and which sequence of tokens is an emotion (in case of experiencer, cause, and target detection).

6.2 Results and Discussion

The results of all the experiments are summarized in Table 4. We evaluate our models in the same way we use F_1 for inter-annotator agreement: Firstly, by accepting a TP if it is exactly found (exact) and secondly, if at least one token is overlapping with the annotation (fuzzy).

Experiment 1 Emotion classification with dictionaries and bag of words show mediocre performance. The recall with the dictionary classification is comparably high ($F_1 = 83$), which is due to the fact that texts were sampled using these dictionaries. However, as we said earlier, annotators are free to label any words and phrases as emotion-bearing, hence low precision and F_1 score. The MLP with BOW features does not perform better but shows increased precision at the cost of lower recall. A possible reason is that each triple may contain only one word that expresses the emotion with the rest of the words being neutral.

Experiment 2 As results of this experiment show, the recall is low for all categories. A presumable reason is, as discussed in Section 5, that substantial number of emotion annotations are words or phrases that are not found in the NRC dictionary. On average, only 46% of emotion annotations are single tokens

[3] `https://spacy.io/`

[4] As available at `https://github.com/facebookresearch/fastText` (Bojanowski et al., 2017)

that can be found in the NRC dictionary, but for some emotions this number is much lower (only 14% of *anticipation* annotation). The same applies to cause and target categories, as in most cases these are long spans of text (*e. g.*, 94% of events are multiword expressions). This explains zero F_1 score for cause prediction with CRF and biLSTM-CRF and a better performance for target prediction with CRF, taking into account that most target relations is triggered by characters, 75% of which are single tokens (see Table 3).

The highest precision and F_1 across all categories is observed for the *emotion* category with biLSTM-CRF (strict $F_1 = 43$ and fuzzy $F_1 = 48$). The strict F_1 is by 12 pp higher than predicted with dictionaries and with BOW in text classification experiment.

The *experiencer* category is second best, however, the recall for this category is still very low. This can be explained by the fact that experiencers are expressed in the text mostly as personal pronouns. As far as the number of personal pronouns in our texts is relatively low (13% of all tokens in a sentence on average), and only a small fraction of them act as experiencers ($< 1\%$ of all tokens in a sentence on average), the classifier cannot learn when an entity is an experiencer or not.

Experiment 3 The goal of this experiment was to estimate if joint modelling of emotion and roles is feasible. We observe that, for the *emotion* category, F_1 increases by 5 pp in strict and by 9 pp in fuzzy evaluation if we provide the classifier with the information, which sequence of tokens is an *experiencer*. For *experiencer* prediction, F_1 increases by 20 pp in strict and by 22 pp in fuzzy evaluation if we tell the classifier which word or sequence is labelled as emotion. These results indicate the complementarity of both categories. A qualitative study on a subsample of linguistic properties of emotions and experiencers shows that when the emotion expression and experiencer are parts of the same phrase (verb or adjectival phrase), the emotion word serves as a head to the word that represents an experiencer. Hence, the classifier is able to partially learn that any phrase that is a part of the emotion phrase, whose head is a personal pronoun or a proper name, is a potential *experiencer*.

The same applies to *experiencer*: if the head of the governing phrase is an emotion, then the head of the current phrase is a potential *experiencer*. However, due to variability of emotion expressions, this cannot always be the case.

7 Discussion, Conclusion, and Future Work

As evaluation of inter-annotator agreement and sequence labelling results of the baseline model show, the task of annotating emotions and corresponding roles, as well as their subsequent prediction is a difficult one. A high variability of emotion expressions (see Table 5) and a variability of cause and target expressions make it hard. At the same time, the resource we present provides interesting and valuable insights in the language of emotion expression and, therefore, is useful to the community of linguists who are interested in the study of linguistic properties of emotions.

However, we also note that developing such a resource has its limitations: Due to the subjective nature of emotions, it is challenging, if not impossible, to come up with an annotation methodology that would lead to less disparate annotations, especially if in addition to emotion, other categories should be annotated together with roles. That is in line with previous research. For instance Schuff et al. (2017) and Russo et al. (2011) found that aggregating labels by multiple annotators without a majority vote procedure but by merging is easier to model computationally.

We tackle this problem by employing a multi-step procedure that helps to improve the agreement of the relation annotation. This does not help in the emotion annotation itself, but helps in the role assignment. The introduction of our multi-step annotation procedure lead to an increased inter-annotator agreement for *experiencer* and *cause* annotations by 13 pp and 5 pp in strict evaluation. This indicates that the task seems easier to annotators if they perform role assignment with predefined emotion annotations.

Another difficulty arises from the nature of the texts we work with. Fictional texts are highly metaphoric and full of allusions and metonymies, which requires thoughtful reading (often reading between the lines) and a broader context. However, this is something that our annotators do not have: all the context they have at their disposal is a triple of sentences, each of which can rely on information that is available in other parts of the book, but not in the annotation unit. Therefore, it is not always possible to annotate the

cause, target, or even the experiencer. This is a trade-off: On the one side, we did not want to annotate full books to have a representative corpus. On the other side, we might not have provided sufficient context. Future work will therefore aim at better understanding how to preselect the relevant context that is needed for reliable annotation and secondly use such knowledge for a follow-up annotation project.

Nonetheless, we are confident that the dataset we present is useful to linguists and digital humanities scholars, as it contains valuable information about complex interactions of emotions, characters, and events in fictional texts, and gives interesting insights into the language of emotion expression in general.

Last but not least, the dataset constitutes a difficult task for structured prediction, as our baseline analysis has shown. Our experiments suggest that the prediction of emotions with their roles is a task that should be tackled with joint models. Therefore, this corpus adds an interesting relation extraction task to the set of existing challenges.

Acknowledgements

This research has been conducted within the CRETA project (`http://www.creta.uni-stuttgart.de/`) which is funded by the German Ministry for Education and Research (BMBF) and partially funded by the German Research Council (DFG), projects SEAT (Structured Multi-Domain Emotion Analysis from Text, KL 2869/1-1). We thank Laura-Ana-Maria Bostan, Sebastian Padó and the CRETA consortium for fruitful discussions.

References

Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. 2005. Emotions from text: Machine learning for text-based emotion prediction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 579–586, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Saima Aman and Stan Szpakowicz. 2007. Identifying expressions of emotion in text. In *International Conference on Text, Speech and Dialogue*, pages 196–205. Springer.

James Barton. 1996. Interpreting character emotions for literature comprehension. *Journal of Adolescent & Adult Literacy*, 40(1):22–28.

Darina Benikova, Chris Biemann, Max Kisselew, and Sebastian Pado. 2014. GermEval 2014 Named Entity Recognition Shared Task: Companion Paper. In *Workshop Proceedings of the 12th edition of the KONVENS conference*, pages 104–112.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ying Chen, Sophia Yat Mei Lee, Shoushan Li, and Chu-Ren Huang. 2010. Emotion cause detection with linguistic constructions. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 179–187, Beijing, China, August. Coling 2010 Organizing Committee.

Charles J. Fillmore, Miriam R.L. Petruck, Josef Ruppenhofer, and Abby Wright. 2003. Framenet in action: The case of attaching. *International Journal of Lexicography*, 16(3):297–332.

Diman Ghazi, Diana Inkpen, and Stan Szpakowicz. 2015. Detecting emotion stimuli in emotion-bearing sentences. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, pages 152–165, Cham. Springer International Publishing.

Lin Gui, Ruifeng Xu, Qin Lu, Dongyin Wu, and Yu Zhou. 2016. Emotion cause extraction, a challenging task with corpus construction. In Yuming Li, Guoxiong Xiang, Hongfei Lin, and Mingwen Wang, editors, *Social Media Processing*, pages 98–109, Singapore. Springer Singapore.

Lin Gui, Jiannan Hu, Yulan He, Ruifeng Xu, Lu Qin, and Jiachen Du. 2017. A question answering approach for emotion cause extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1593–1602. Association for Computational Linguistics.

Patrick Colm Hogan. 2015. What Literature Teaches Us About Emotion: Synthesizing Affective Science and Literary Study. In Lisa Zunshine, editor, *The Oxford Handbook of Cognitive Literary Studies*, chapter 13. Oxford University Press, March.

Matthew Honnibal. 2013. A Good Part-of-Speech Tagger in about 200 Lines of Python. Online: https://explosion.ai/blog/part-of-speech-pos-tagger-in-python.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Victor Hugo. 1885. *Les Misérables*. Project Gutenberg: http://www.gutenberg.org/ebooks/135.

Randy Ingermanson and Peter Economy. 2009. *Writing fiction for dummies*. John Wiley & Sons.

Philip Nicholas Johnson-Laird and Keith Oatley. 1989. The language of emotions: An analysis of a semantic field. *Cognition and emotion*, 3(2):81–123.

David Comer Kidd and Emanuele Castano. 2013. Reading literary fiction improves theory of mind. *Science*, 342(6156):377–380.

Evgeny Kim, Sebastian Padó, and Roman Klinger. 2017. Investigating the relationship between literary genres and emotional plot development. In *Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 17–26.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of the International Conference on Machine Learning*, pages 282–289.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

Raymond A Mar, Keith Oatley, and Jordan B Peterson. 2009. Exploring the link between reading fiction and empathy: Ruling out individual differences and examining outcomes. *Communications*, 34(4):407–428.

Lee Sophia Yat Mei, Chen Ying, Huang Chu-Ren, and Li Shoushan. 2012. Detecting emotion causes with a linguistic rule-based approach. *Computational Intelligence*, 29(3):390–416.

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Saif Mohammad, Xiaodan Zhu, and Joel Martin. 2014. Semantic role labeling of emotions in tweets. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 32–41, Baltimore, Maryland, June. Association for Computational Linguistics.

Alena Neviarouskaya and Masaki Aono. 2013. Extracting causes of emotions from text. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 932–936.

Keith Oatley. 2002. Emotions and the story worlds of fiction. *Narrative impact: Social and cognitive foundations*, 39:69.

Andrew J Reagan, Lewis Mitchell, Dilan Kiley, Christopher M Danforth, and Peter Sheridan Dodds. 2016. The emotional arcs of stories are dominated by six basic shapes. *EPJ Data Science*, 5(1):31.

Jenefer Robinson. 2005. *Deeper than reason: Emotion and its role in literature, music, and art*. Oxford University Press on Demand.

James A Russell and Lisa F Barrett. 1999. Core affect, prototypical emotional episodes, and other things called emotion: dissecting the elephant. *J Pers Soc Psychol*, 76(5):805–819, May.

James A Russell. 2003. Core affect and the psychological construction of emotion. *Psychological review*, 110(1):145.

Irene Russo, Tommaso Caselli, Francesco Rubino, Ester Boldrini, and Patricio Martínez-Barco. 2011. Emocause: an easy-adaptable approach to emotion cause contexts. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, pages 153–160. Association for Computational Linguistics.

Roser Saurí and James Pustejovsky. 2009. Factbank: a corpus annotated with event factuality. *Language resources and evaluation*, 43(3):227.

Andrea Scarantino. 2016. The philosophy of emotions and its impact on affective science. *The handbook of emotions*, pages 3–65.

Klaus R Scherer and Harald G Wallbott. 1994. Evidence for universality and cultural variation of differential emotion response patterning. *Journal of personality and social psychology*, 66(2):310.

Hendrik Schuff, Jeremy Barnes, Julian Mohme, Sebastian Padó, and Roman Klinger. 2017. Annotation, modelling and analysis of fine-grained emotions on a stance and sentiment detection corpus. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Copenhagen, Denmark. Workshop at Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics.

Frederic Jesu Stimson. 1943. *The King's Men: A Tale of Tomorrow*. Project Gutenberg: `http://www.gutenberg.org/ebooks/18960`.

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 70–74. Association for Computational Linguistics.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2):165–210, May.

Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. 2013. Webanno: A flexible, web-based and visually supported system for distributed annotations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 1–6, Sofia, Bulgaria, August. Association for Computational Linguistics.

A Concepts used for Phrase Annotation

Table 5 provides a list of concepts defined for the annotation of emotions, modifiers, and entities. The Examples column contains examples of annotations from the final corpus. Table 6 provides examples of linguistic realization of entities along with examples from the annotated dataset.

	Concept Value	Examples
Emotion	Anger	*angry, defend themselves by force, break your little finger, loss of my temper*
	Anticipation	*want, wish, wholly absorbed, looked listlessly round, wholly absorbed*
	Disgust	*repellent, cheap excitement, turn away from, beg never to hear again*
	Fear	*horrified, tense and frightened, shaking fingers*
	Joy	*cheerful, grateful, boisterous and hilarious, violins moved and touched him*
	Sadness	*failed, despair, the cloudy thoughts, staring at the floor*
	Surprise	*perplexing, suddenly, petrified with astonishment, loss for words, with his mouth open*
	Trust	*honor, true blue, immeasurable patience*
	Other	*careful, brave, had but a tongue, break in her voice, bit deeply into his thumb*
Modifier	strong	*I loved her the more*
	weak	*with a little pity*
	negated	*could not be content*
Entity	character	*the chairman of the board*
	event	*marry a man I did not love, because of his gold*
	other	*Lily's beauty*

Table 5: Concepts used for the phrase annotation layer.

Entity type	**Linguistic realiz.**	**Examples**
Character	noun phrase	*his son*
	adjectival phrase	*old man*
Event	verb phrase	*Mrs. Walton had got another baby.*
	adverbial phrase	*Jesus spoke unkindly to his mother when he said that to her.*
	prepositional phrase	[. . .] *giving her up.*
	clause	[. . .] *what she said to him* [. . .]
	noun phrase	*the journey*
Other	adjectival phrase	[. . .] *old age* [. . .]
	noun phrase	[. . .] *the heavens and the earth.*
	tense phrase	*She was the only treasure on the face of the Earth that my heart coveted.*

Table 6: Typical linguistic realization of entities.

B Genre and author composition

Subject headings	Most frequent author	# texts
Fiction, Christian fiction	MacDonald George	178
Historical fiction (translations), Epic literature	Hugo Victor	107
Social fiction	Dostoevsky Fyodor	63
Domestic fiction, Single women	Gissing George	45
Young men, Bildungsroman	Thackeray William	42
Love stories	James Henry	38
Didactic fiction	Eliot George	36
Political fiction	Atherton Gertrude Franklin Horn	35
Historical fiction (translations), France	Dumas Alexandre	35
German fiction (translations), Social classes	Freytag Gustav	22

Table 7: Most frequent subject headings and authors in the corpus. Subject headings are taken from Project Gutenberg metadata and are shortened for readability.

C Excerpt from the corpus file

```
<document author="Glasgow Ellen" author_death_year="1945" book_title="The Battle Ground" doc_id="6872"
      genre="Historical fiction" url="http://www.gutenberg.org/ebooks/6872">
  <text>In loving me, my darling?" "In loving you like that." "Nonsense.</text>
  <adjudicated>
    <spans>
      <span annotation_id="51002" annotatorId="B"
        cbegin="17" cend="24" type="character">darling</span>
      <span annotation_id="49637" annotatorId="A"
        cbegin="31" cend="37" type="joy">loving</span>
      <span annotation_id="49644" annotatorId="A"
        cbegin="31" cend="37" type="trust">loving</span>
      <span annotation_id="50015" annotatorId="B|A"
        cbegin="38" cend="41" type="character">you</span>
    </spans>
    <relations>
      <relation annotatorId="B" left="17" right="37" relation_id="51009" source_annotation_id="49637"
        target_annotation_id="51002" type="experiencer">darling[CHARACTER]...loving[JOY]</relation>
      <relation annotatorId="B|A" left="31" relation_id="50022" right="41" source_annotation_id="49637"
          target_annotation_id="50015" type="target">loving[JOY]...you[CHARACTER]</relation>
    </relations>
  </adjudicated>
  <other>
    <spans>
      <span altTo="49644" annotation_id="49581" annotatorId="C" cbegin="31" cend="37"
              type="other-emotion">loving</span>
    </spans>
    <relations />
  </other>
</document>
```

Figure 4: Excerpt from REMAN corpus.

Local String Transduction as Sequence Labeling

Joana Ribeiro[†] **Shashi Narayan**[†] **Shay B. Cohen**[†] **Xavier Carreras**[‡]
[†] School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh EH8 9AB, UK
[‡] dMetrics, Brooklyn, NY 11211, USA
s1536851@sms.ed.ac.uk shashi.narayan@ed.ac.uk
scohen@inf.ed.ac.uk xavier.carreras@dmetrics.com

Abstract

We show that the general problem of string transduction can be reduced to the problem of sequence labeling. While character deletions and insertions are allowed in string transduction, they do not exist in sequence labeling. We show how to overcome this difference. Our approach can be used with any sequence labeling algorithm and it works best for problems in which string transduction imposes a strong notion of locality (no long range dependencies). We experiment with spelling correction for social media, OCR correction, and morphological inflection, and we see that it behaves better than seq2seq models and yields state-of-the-art results in several cases.

1 Introduction

String transduction (mapping one string to another) is an essential ingredient in the natural language processing (NLP) toolkit that helps solve problems ranging from morphological inflection (Dreyer et al., 2008) and lemmatization to spelling correction (Cotterell et al., 2014), text normalization (Porta and Sancho, 2013) and machine translation (Kumar et al., 2006).

Finite-state technology is often used with string transduction (Allauzen et al., 2007), especially when there is a strong notion of locality in the transduction problem, i.e., when there are no long range dependencies between the predicted characters. More recently, neural network methods have become more common for such problems using the seq2seq models that were introduced for machine translation (Bahdanau et al., 2015).

Similarly, sequence labeling algorithms have been the mainstay for an array of problems in NLP, including part-of-speech tagging, named entity recognition and semantic role labeling. Various models have been proposed to do sequence labeling, including conditional random fields, hidden Markov models and neural networks.

String transduction and sequence labeling are often treated as two separate entities, and indeed, usually give treatment to different problems in NLP. We claim and show that there is a great similarity between them, and that one can be reduced to the other.

Sequence labeling is often associated with a set of problems in which the set of labels is small compared to the vocabulary. Still, this labeling essentially associates each symbol (usually a word) in a sequence (usually a sentence) with a label. As such, it can be seen as a specific case of string transduction that imposes a one-to-one correspondence between input and output symbols. This is the easier side of the reduction between sequence labeling and string transduction, and it has been exploited in the deep learning literature with seq2seq models.

In this paper, we come to address the reduction in the reverse direction – i.e. *we show how to frame string transduction as sequence labeling*. We do this by decomposing the transduction process into two parts: first, we induce an alignment between the characters of the input and output strings and then we use a sequence labeling model to predict the final output of the string.

The main difficulty here is not in the size of the vocabulary compared to the label set, but the fact that string transduction can yield to arbitrary output string length compared to the input string. In machine

Proceedings of the 27th International Conference on Computational Linguistics, pages 1360–1371
Santa Fe, New Mexico, USA, August 20-26, 2018.

translation, for example, a classic example of string transduction, the sentence in the target language can be shorter, longer, and contain a significant amount of re-ordering. Indeed, re-ordering is a challenge with string transduction. A sequence labeling model usually maintains higher order of monotonicity (such as with Markovian models), while for general string transduction problems, one needs models such as encoder-decoders or grammatical models. Such expressive models, on the other hand, are *not* a good fit for string transduction problems with a strong notion of locality, as they are too complex and lead to weaker generalization power for such problems (Rastogi et al., 2016). We show this weakness in our experiments with seq2seq models supporting similar conclusions in previous work such as by Schnober et al. (2016). Our experiments show that even when seq2seq models are incorporated with hard monotonic attention (Aharoni and Goldberg, 2017), our reduction to sequence labeling outperforms such models on certain problems with a strong notion of locality.

Our approach to reduce string transduction to sequence labeling relies on a simple observation: in most cases in transduction, it is easier to *delete* a symbol than to *insert* a symbol. This is true because insertion requires identifying both the point of insertion and the character that needs to be inserted, while deletion is a "binary" decision – either the decoding algorithm chooses to delete a specific symbol or not to. A similar observation was made by Schnober et al. (2016), Rosca and Breuel (2016) and Chandlee (2014) and is also present in much earlier work by Koskenniemi (1983) and later in Chandlee (2014). String transduction is treated by Koskenniemi (1983) as a two-layer problem: two strings (a surface representation and a lexical representation) were aligned according to a set of "two-layer" handwritten rules. In this paper, we propose a similar setup to Koskenniemi's but instead, we use various sequence labeling methods to automatically create these alignments (instead of hand-engineer them). This is possible by considering the three character operations used in string transduction – *deletions*, *insertions* and *substitutions* – as tagging operations. While Koskenniemi's work was done only in the context of morphology, we extend the idea further and apply it to a wider set of problems. However, our approach, like Koskenniemi's, is for now limited to character level tasks.

Our observation leads to an approach which is simple, yet powerful. We explain it in §3 and then perform an extensive experimentation with it in various scenarios, comparing it to models such as seq2seq. We test our approach thoroughly on three string transduction problems with a strong notion of locality: social media spelling correction, optical character recognition and morphological inflection. Each problem is tested with several sequence labeling algorithms. In all cases, we find that our approach is competitive with strong baselines and in some cases outperforms them. Our approach is independent of the sequence labeling algorithm that is used, and indeed we experiment with it in conjunction with four different sequence labeling learning algorithms.

2 Problem Formulation

We give a treatment to the problem of string transduction. Given an input alphabet, Σ, an output alphabet Λ, and training data, we aim to learn a function $f\colon \Sigma^* \to \Lambda^*$ that maps input strings to output strings. The training data comes in the form of pairs of strings $(x^{(i)}, y^{(i)})$ for $i \in \{1, \ldots, M\}$. Note that the input string and output string can be of different length.

We reduce the problem of learning f to sequence labeling. With sequence labeling, we aim to learn a function $g\colon \Sigma^* \to \Lambda^*$ such that the length of $y = g(x)$ is the same as of x. This means that the training data we receive to learn g has input and output strings of the same length. With sequence labeling, $g(x)$ is referred to as the "labels" of x, as each element in x is mapped to a single element in $g(x)$ (elements from the input string are not deleted and new ones are not inserted).

Clearly, the problem of string transduction subsumes the problem of sequence labeling, as one can always try to learn a mapping f from a given training data of strings of identical length. However, there is a significant distinction that is usually made between the two, as string transduction can re-write a string into a completely different string, while sequence labeling has a stronger notion of locality.

As such, sequence labeling is considered an easier problem than general string transduction. Yet, we show in this paper how to exploit sequence labeling algorithms, with their flexibility and efficiency, to do general string transduction.

3 Transduction as Insertion and Labeling

Most approaches to string transduction involve inducing an alignment between symbols in the input and output strings (Knight and Graehl, 1998; Clark, 2001; Eisner, 2002; Azawi et al., 2013; Bailly et al., 2013). In an alignment, unaligned input symbols are called deletions, while unaligned output symbols are called insertions.

It is challenging to jointly induce alignments and learn a transduction model.

A second challenge is that, at prediction time, it is difficult to predict the insertions, as there can be an arbitrary number of them between any two input symbols. The prediction problem would be much simpler if the insertion positions were in place, because the model would only need to decide which symbol goes in each position.

Our approach is based on this idea. We decompose the transduction process into first predicting insertion positions in the input string, and then labeling the modified input string to produce the final output string. The insertion function might predict insertion positions that should not be there —we call these *spurious* insertions. In this case we rely on the labeling function to eliminate these spurious insertions. It is an easier task to *delete* a symbol than to *insert* one for string transduction. Therefore, marking the potential insertions in the string and then deleting them if necessary during decoding reduces the problem of string transduction to sequence labeling and as such allows using the Viterbi algorithm for decoding.[1] In addition, to avoid the challenge of inducing alignments, our approach relies on existing ones — for our experiments, we use the edit distance algorithm. From these initial alignments, our approach learns an insertion function and a sequence labeler, which perform string transduction when composed. This simple approach is also very flexible, because potential insertions can later be deleted. We found this approach to yield quite good alignments, as the accuracy of our algorithm when the alignment is *known* is quite high and supersedes state-of-the-art results in several cases. We now explain this technique more in detail.

3.1 Insertion Algorithm

We learn an "insertion context function" $\sigma\colon \mathbb{N} \times \Sigma^* \to \mathbb{N}$. Given a string $x = x_1 \cdots x_k$ over Σ and an index in that string (between 0 and k),[2] this function tells how many *potential* insertions could happen in that position.

For example, say we would like to transduce the string `say` from its base form to its past form `said`. The insertion context function might learn that there needs to be an insertion following the `y` that corresponds to the `d`. Therefore, the insertion function values for `say` would be:

$$\sigma(i, \texttt{say}) = 0 \qquad \forall i \in \{0, 1, 2\}$$
$$\sigma(3, \texttt{say}) = 1.$$

and therefore the string we would try to transduce to `said` would be the string `say`ε, where ε is a special symbol marking potential insertion.

Learning σ To learn σ, we use the following procedure. We first align each pair of strings in the training set using an algorithm such as edit distance (see §4.1). We then check for all contexts in which an insertion, or a sequence of insertions happen. A context is defined to be the preceding character before the sequence of insertions and the character that follows the sequence of insertions – for instance, in the previous example `say`, the context for the letter `a` would be the letters `s` and `y`. All of these contexts are added to a dictionary of insertion contexts, always keeping the maximal number of insertions for each specific context. Then the function σ adds a sequence of insertions any time a specific context fires. For the example above, the context we learn is `y`ε\$ where \$ denotes the end of a string. Therefore, for a

[1] Finite state transducers that allow insertions or deletions, in practice, need to bound the number of insertions at any specific point, similarly to our insertion context function. This is done by operating on a lattice, where a path in that lattice corresponds to a string transduction with a specific alignment. Decoding using a lattice is equivalent to using the Viterbi algorithm. See, for example, Rastogi et al. (2016).

[2] The index really marks a *space* between two characters in the string, with 0 denoting the beginning of the string.

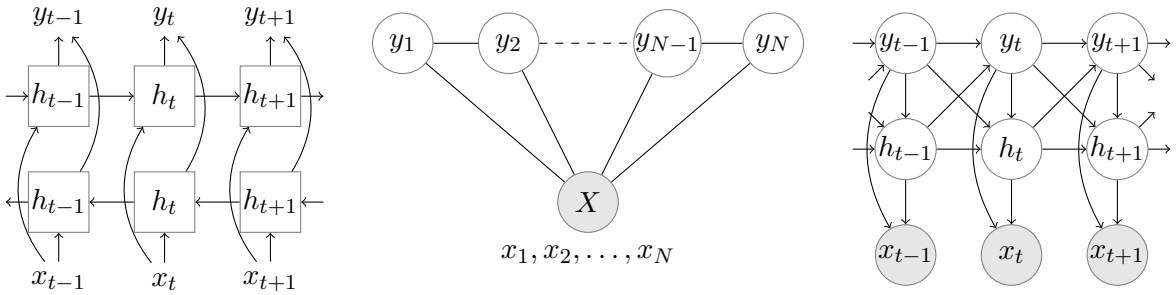


Figure 1: Graphical depictions of BiLSTMs (left), conditional random fields (middle) and refinement hidden Markov models (right). In practice, for the R-HMMs and CRFs we use a trigram model and not a bigram model as depicted.

decoded string that has the same context as the one found before, for instance, the string `slay` we would have $\sigma(4, \texttt{slay}) = 1$.

For all contexts, we calculate (based on the training data) a frequency table, that gives the probability of an insertion given the specific context. In some cases, we then tune a threshold (based on a development set) so that we add insertions only in cases where the frequency is larger than that threshold. We found out that while it perhaps prevents insertions in legitimate positions, if the frequency threshold is low enough, it does not have an adverse effect. On the contrary, it prevents adding spurious insertions that cannot be correctly recovered as deletions. For more details, see §4.

In the morphology tasks we experiment with, insertions tend to happen as suffixes or prefixes. To capture this, we also learn a pair of thresholds τ_0 and τ_1 (by grid search on the training set) such that $\sigma(i, x)$ is set to 0 if $\tau_0 \leq \frac{i}{|x|} \leq \tau_1$.

Learning the Labeling Model with σ Once we have the σ function applied to all aligned training data, we mark in the *output* sequences in the training data the spurious insertions that should not be in the input sequence with a special symbol `D` (for deletion). Therefore, the learning algorithm is now also required to learn where to delete these extra insertions that σ elicits. For example, if we consider the previously mentioned example of `say` and `said`, the input sequence in this case would be `say`ε and the output sequence would be `said`. However, it is not always the case that the past tense of a verb with a *ay* ending is the same word root with the suffix *-d* (and the transformation of the final *y* to *i* to accommodate the new suffix). For example, the past tense of `may` is also `may`, and in this case, if the string `may` was also in the training set, `may` should retain its form and be transduced to `may`. With the context function mentioned above we would have the pair `may`ε (input sequence), `mayD` (output sequence) to learn from. During decoding, we apply the σ function on the input string and then remove the `D` symbols from the string.

3.2 Sequence Labeling Models

We experiment with three models for sequence labeling: refinement hidden Markov models (R-HMMs; Stratos et al., 2013), conditional random fields (CRFs; Lafferty et al., 2001) and bidirectional Long Short Term Memory neural networks (biLSTMs; Graves and Schmidhuber, 2005). A graphical depiction of the models that we experiment with is given in Figure 1. For R-HMMs, we experiment with two learning algorithms, the expectation-maximization algorithm (EM; Dempster et al., 1977) and an L-PCFG spectral learning algorithm (Cohen et al., 2012; Cohen et al., 2013).[3]

4 Experiments

We describe in this section a set of experiments on datasets from three problems: social media spelling correction, optical character recognition (OCR) correction and morphological inflection. We believe that this array of datasets represents a broad set of problems in which string transduction as sequence

[3]We use the code from `https://github.com/shashiongithub/Rainbow-Parser`.

labeling can be tested. We apply the insertion technique to a set of sequence labelers: conditional random fields, refinement HMMs with expectation-maximization, spectral algorithms, and neural networks with biLSTM. All the results reported are at word level.

Our experiments require hyperparameter tuning of all algorithms, which we do over a fixed set of hyperparameters in the development set. For the latent-variable models, we vary the number of latent states between 1 and 48. For the biLSTM, we try a number of layers between 1 and 4 and a number of neurons ranging between 100 and 500.

The CRF is used with trigram modeling, and as such, the features it uses are very similar to the features used by the spectral algorithm. The main difference is that the spectral algorithm uses these features only during training, while the CRF uses them both during training and decoding.

In addition, we compare all our models to a seq2seq model implementation (Bahdanau et al., 2015). The model includes an attention mechanism, and we optimize the number of latent units over the development sets, ranging it from 10 to 100. We also include a comparison to seq2seq models with a "hard" attention mechanism (Aharoni and Goldberg, 2017), designed for string transduction with monotonic alignments.

4.1 Twitter Spelling Correction

In our first set of experiments, we use a corpus of spelling corrections for Twitter (Aramaki, 2010). The corpus includes 39,172 words in their original form and their correct form. We split this dataset into three parts (consecutively): 31,172 entries for a training set, 4,000 entries for a development set and another 4,000 entries for a test set.

Further Tuning In our experiments with EM and the spectral algorithm, we also tried to tune the σ context function on the development set. We discovered that the best context function is the one that adds ε slots for every context $a - b$ that appears in the data, if the relative frequency of $a - b$ to $a \star b$ is larger than $p = 0.0001$ (where $\star$ stands for a wildcard; the interpretation of this relative frequency division is that we check whether the conditional probability of having an ε given the context $a \star b$ is larger than p). This is not particularly surprising given the fact that the Twitter dataset is composed of single words, each having a single typo. Therefore, each pair of misspelled and correctly spelled words often needs at most one insertion to be aligned. Even though this threshold may seem small, it was enough to remove all the isolated, non-representative context cases.

We try both a bigram and a trigram model. The trigram model greatly outperforms the bigram model, so we focus on it. We also did some preliminary experiments with a 4-gram model, but this model did not seem to do better than the trigram model. For the biLSTM architecture, we find on the development set that the smallest error is with 2 layers and 350 neurons.

In our experiments, we compare two latent-variable learning methods: the EM algorithm and the spectral algorithm described by Stratos et al. (2013). EM was run for 8 iterations – in some preliminary experiments where we ran EM for much longer, we noticed it converges after very few iterations. We then choose the iteration that gave the best result on the development set. We range the number of latent states, m, between 2 and 48.

The Dictionary Layer We also experiment with an additional layer that we add to our transduction algorithm. In this layer, we use a dictionary (a fixed set of words) for which we find the closest entry (in edit distance) to the output of our algorithm. In that case, our transduction algorithm can be thought of as producing an *intermediate* string, i.e., a string that may or may not already be a correct word but that is closer to the correct word in terms of edit distance. The intermediate string representation is intended to correct mistakes while actually relying on the type of substitutions, deletions and insertions done by users of social media. However, it is still prone to character-based mistakes, since it is a learned component. This is the reason why we also include the second edit distance layer.

We use two types of dictionaries. The first dictionary is the set of all correctly spelled words in the training data. The second dictionary is the English dictionary extracted from the GNU aspell software.[4] In our results, we report four types of accuracy levels: the accuracy for the whole test set (where accuracy

[4] `ftp://ftp.gnu.org/gnu/aspell/dict/0index.html`.

method	acc	acc4+	acc6+	acc10+
aspell	30.1	41.1	56.6	82.6
train	40.1	59.6	**71.6**	71.9
GNU aspell	29.0	46.2	58.8	80.2
seq2seq	46.3	44.8	45.3	33.8
seq2seq+train	55.7	59.5	65.1	**86.2**
seq2seq+aspell	47.6	48.5	45.0	65.4
seq2seq (hard)	52.2	49.1	31.6	13.4
seq2seq+train (hard)	62.2	63.6	59.6	63.8
seq2seq+aspell (hard)	54.9	53.3	47.1	63.8
Spectral	54.3	52.6	44.0	28.1
Spectral+train	63.3	**68.4**	70.1	69.4
Spectral+aspell	56.6	59.4	58.6	70.2
biLSTM	54.1	54.1	55.4	49.3
biLSTM+train	**64.6**	64.6	68.7	72.1
biLSTM+aspell	56.9	57.0	59.1	63.0
CRF	34.2	34.2	32.2	19.0
CRF+train	55.9	55.9	59.5	67.8
CRF+aspell	45.2	45.2	46.4	50.8
EM	28.3	22.0	14.5	9.9
EM+train	52.8	64.0	70.8	72.7
EM+aspell	41.7	48.7	55.7	82.6

Table 1: Final results on the test set for the Twitter dataset. "aspell" stands for edit distance minimization using the aspell dictionary. "train" stands for the same with the dictionary extracted from the training set. "GNU aspell" stands for the results of running the GNU aspell software. "acc" denotes the accuracy over the whole dev/test set, "acc$n$+" for words of length $\geq n$. For each method the best model was chosen as follows: for CRF, over the dictionary; for EM, over iterations, number of latent states, and dictionary; and for spectral, over the hyperparameters, number of latent states and dictionary. The best model was chosen for each accuracy length criterion separately. The best result in each column is in bold. Top part are baselines. Bottom part uses our reduction to sequence labeling.

is measured as the number of words we fixed to the correct form), and also accuracy only for (correct) words which are longer than 4 characters, 6 characters and 10 characters.

We experimented with two types of edit distance algorithms. The first one allows only deletions, insertions and substitutions, and the other also allows transpositions. We discovered that edit distance with transposition behaves better, so we focus on it.

Results Table 1 gives the final result on the test set for the Twitter data (for different word lengths). The first thing to note is that there are some models that perform well on words longer than 10 characters. In general, it is easier to correct the spelling of longer words with the aspell dictionary because: (a) longer words are more likely to originate in words that appear in the dictionary; (b) there is more evidence in a long string for the misspelled string than in a short string.

Still, high accuracy on long words does not necessarily indicate the total accuracy is higher. We see that the spectral algorithm and the biLSTM model tend to give higher results on all words lengths overall than both the conditional random field and the EM algorithm. The GNU aspell software itself does not perform well on this dataset. This perhaps should not come as a surprise, as the GNU aspell software is designed for less informal text.

In addition, we see that sequence labeling performs consistently better than seq2seq models with attention. This should not come as a surprise, as seq2seq models encode the whole string before decoding it, even when the attention mechanism is used. This finding is also supported by (Schnober et al., 2016). The seq2seq method of Aharoni and Goldberg (2017) behaves even worse without using a dictionary,

context	dictionary	average	std
left	none	42.2	0.67
right	none	42.1	0.63
both	none	42.3	0.61
left	aspell	48.6	0.52
right	aspell	48.6	0.54
both	aspell	48.8	0.56
left	train	58.3	0.49
right	train	58.3	0.27
both	train	58.6	0.46

Table 2: Sensitivity analysis of the different context functions as a function of the probability threshold for insertion in a context. The contexts vary over left character ("left"), right character, or characters both on the left and the right ("both"). In addition, we use dictionaries (the aspell dictionary or the training data as dictionary). "average" gives the average result over the different thresholds, and "std" gives the standard deviation of that average.

especially for long words.

The Role of the Context Function We note that the choice of the context function is important, and extra ε symbols have to be added based on data-oriented choices. For example, when we experimented with a constant context function that adds an additional ε symbol after every character during training and decoding, the accuracy on the test set was much lower, whether we used a dictionary or not. To be more precise, the accuracy without using a dictionary was 33%, with the "train" dictionary mitigating the low performance to an accuracy of 61.1% (which is significantly lower than the biLSTM result and the spectral result) and 54.0% for the "aspell" dictionary.

To test the role of the context function, we further evaluated it more thoroughly. We created three types of context functions: one that looks at context only to the left of the current character, one that looks at context only on the right, and finally one that looks on context both on the left and on the right (as originally defined). In addition, we varied the threshold that activates an insertion in a specific context (see §3). The threshold was varied over the values in the set $\{0.0001, 0.0005, 0.001, 0.002, 0.005, 0.01, 0.05, 0.1, 0.15, 0.2\}$. The higher the threshold is, the less spurious insertions happen (or insertions in general).

We experimented with a single learning algorithm: the biLSTM neural network. Table 2 summarizes this sensitivity analysis. The "average" results give the performance on the development set, averaging over the different thresholds. The "std" result gives the corresponding standard deviation. As can be seen, on average, right context functions have smaller standard deviation than other contexts, while the accuracy levels are similar for all contexts. In addition, there is relatively not a very large sensitivity to the actual threshold use.

The same dataset was also used by Cotterell et al. (2014) and by Silfverberg et al. (2016) though in both cases the results are not comparable as the experimental setup and the errors reported are different than the one used in this paper. For example, Cotterell et al. do not report accuracy at all, but instead plot mean edit distance as a function of the number of training examples.

4.2 Optical Character Recognition Correction

In our second set of experiments we used the Finnish OCR dataset used by Silfverberg et al. (2016). This corpus consists of 36,470 word pairs and it was generated from processing Early Modern Finnish texts through an OCR engine. According to Silfverberg et al. (2016) this corpus is a subset of a larger corpus comprising historical newspapers and magazines digitized by the National Library of Finland. This dataset has been both manually edited by the Institute of Languages of Finland as well as through crowdsourcing. In order to run our experiments we used the same setup and the same 10 non-overlapping parts described in Silfverberg et al. (2016).

Method	acc	acc4+	acc6+	acc10+
Unstructured	20.0	-	-	-
+ Lexicon	21.6	-	-	-
Perceptron	32.1	-	-	-
+ Lexicon	35.1	-	-	-
seq2seq	79.9	80.0	78.8	78.6
seq2seq (hard)	58.4	55.4	55.3	53.7
Spectral	77.3	77.9	76.3	73.8
CRF	**81.8**	**81.9**	**80.7**	**78.8**
EM	79.9	80.5	79.3	77.4
biLSTM	81.7	81.7	80.3	78.3

Table 3: Finnish OCR results. The first set of baselines (upper part) are reported by Silfverberg et al. (2016), and as such, the numbers are not available for all columns. Bottom part is our method.

Further Tuning In our experiments, we use 10-fold cross validation. The results are averaged over these ten folds. The hyperparameters for the spectral algorithm were tuned on the development set of the first fold, and then we used the hyperparameters identified at that phase for the rest of the folds.

We tuned the neural network architecture parameters on the development set of the first fold. We found out that the biLSTM model works best with 1 layer with 200 neurons.

We tuned the context function on development data again (for fold 1). We discovered that the best context function includes a pair of ε in any context of the form $a - \ - b$ that it appeared in the training data, and a single ε in any context with conditional probability larger than 0.0001. See more in §4.1. The OCR dataset is quite different in its structure from the Twitter dataset. It has both words that do not need any correction, and words in which more than one character was not labeled with the correct character. Another important difference is the fact the "typos" were often grammatical in the Twitter dataset (as they were human misspellings), while that is not the case in the OCR dataset. It is also worth noting that Finnish relies heavily on double letters in its spelling, which could explain why this choice of σ function works best in this case but not in the Twitter one.

Silfverberg et al. (2016) present two approaches for this problem, both relying on finite-state technology. In one approach, "unstructured," they extract finite-state rules from pairs of strings and then apply them during decoding. In the "structured" case there is an additional Perceptron layer which is trained based on features extracted from the unstructured classifier. In addition, they have an additional layer which relies on a lexicon to find the best correction for a given input string.

Results Table 3 gives the results. We see that all uses of sequence labeling significantly outperform the results of Silfverberg et al. (2016). Even with an external lexicon, the results of the unstructured and structured classifiers of Silfverberg et al. significantly lag behind our results. Our results are also relatively stable with regards to word length. While the results are slightly lower for words of length 10 or more, the accuracy level is rather stable for other lengths. This is actually a different finding than the one we found with the Twitter spelling correction experiment. We see yet again that seq2seq modeling performs worse than our approach with biLSTM. seq2seq do not perform well with the hard monotonic attention model. This is on par with the results for the Twitter spelling dataset, on which we see bad behavior on long words. The OCR dataset indeed includes relatively long words.

4.3 Morphological Inflection

We also run experiments on the morphological inflection dataset used by Rastogi et al. (2016).[5] This dataset was first processed by Dreyer (2011) and we use the same setup in our experiments, i.e., in each experiment we use the same 2,500 example dataset sampled from the CELEX database (Baayen et al., 1993) divided into three sets, 500 examples for training, and 1,000 for both development and testing. This

[5] https://github.com/se4u/neural_wfst/.

Model	13SIA	2PIE	2PKE	rP
biLSTM-WFST	85.1	94.4	85.5	83.0
biLSTM (ensemble)	85.8	94.6	86.0	83.8
Moses15	85.3	94.0	82.8	70.8
Dreyer (Backoff)	82.8	88.7	74.7	69.9
Dreyer (Lat-Class)	84.8	93.6	75.7	81.8
Dreyer (Lat-Region)	**87.5**	93.4	87.4	84.9
Kann (MED)	82.3	94.4	86.8	83.9
Kann (MED+POET)	83.9	95.0	87.6	84.0
seq2seq	83.1	93.8	88	83.2
seq2seq (hard)	85.8	**95.1**	**89.5**	**87.2**
EM	68.5	93.5	77.2	75.8
biLSTM (ours)	80.2	93.8	84.2	83.0
CRF	79.2	88.1	72.4	84.7
Spectral	81.9	94.0	83.2	83.8
Spectral (ensemble)	84.0	93.8	83.7	83.5
Spectral (oracle)	*85.8*	*99.6*	*92.4*	*87.5*

Table 4: Results on the morphological inflection datasets. Baselines (upper part of the table) are reported by Rastogi et al. (2016) and Dreyer (2011). The best result in each column is in bold. Results for seq2seq are reported by Aharoni and Goldberg (2017). Kann results are reported by Kann and Schütze (2016).

corpus is divided in four subsets, each defining an inflection task between different verb tenses in German. These are labeled as follows:

- 13SIA→13SKE - 1st/3rd persons singular of the indicative past to 1st/3rd persons singular of the subjunctive present.
- 2PIE→13PKE - 2nd person plural of the indicative present to 1st/3rd person plural of the subjunctive present.
- 2PKE→z - 2nd person plural of the subjunctive present to the infinitive.
- rP→pA - imperative plural to the past participle.

Further Tuning For the 13SIA dataset, we discovered on development data (fold 1) that the best context function is one that adds a single ε any time it appeared in the training data with a context $a\varepsilon b$. For 2PIE, we discovered similarly that adding a single ε is best in all cases it appears in the end of the word in the training set. Finally, for 2PKE and rP, we discovered the best context function is such that it adds a single ε or a pair of ε in any context it appears in the training data (with a character to the left and a character to the right).

The morphological inflection dataset is the most different of the three datasets mostly because it allows for the insertion or deletion of full suffixes. We speculate it is the main factor as to why the best context functions for the morphological datasets are different than for the previous two datasets (Twitter and OCR).

With the biLSTM model we discovered on the development set of the first fold of each dataset that: 13SIA works best with 2 layers of size 300; 2PIE works best with 1 layer of size 150; 2PKE works best with 1 layer of size 250; and rP-pA works best with 2 layers of size 250.

Results The results are in Table 4. The biLSTM results reported in the upper part of the table do not use our sequence labeling technique, and are re-iterated as reported by Rastogi et al. (2016).

In general, the spectral algorithm does not perform as well as the LSTM baseline with finite state transducers, but does better than it on the rP-pA dataset. The spectral algorithm performs significantly better than some of the other baselines, such as Moses or the baselines from Dreyer (2011). This is especially true for the 2PKE and the rP-pA datasets.

An additional observation is that the way we handle insertions has a significant impact on the performance. In oracle mode (meaning, when the two strings are aligned using edit distance even at decoding time, and the spurious insertions are not used), the spectral algorithm performs significantly better. This should not come as a surprise, but does point to a direction on how to improve the use of our sequence labeling technique – better handling of insertions and the addition of less spurious insertion placeholders that are deleted during decoding.

In contrast to the other datasets, where perhaps the assumption of monotonicity is too strong, seq2seq models with an attention mechanism designed to handle monotonic alignments (Aharoni and Goldberg, 2017) perform quite well on this task, even better than the vanilla seq2seq models.

Finally, we also consider the case in which we use an ensemble method, combining several spectral models together (the top 50 performing models on the development set from the hyperparameter sweep). We combine the models using a MaxEnt reranker such as described by Charniak and Johnson (2005) and Narayan and Cohen (2015). We find that the ensemble approach does improve the results significantly for the 13SIA dataset, and also for the 2PKE-z dataset.

5 Conclusion

We presented a technique to frame general string transduction problems as sequence labeling. Our technique works by adding to the string to be transduced additional insertion markers, which are later potentially deleted during the sequence labeling process. Our approach is general and works with any sequence labeling algorithm. We developed our technique with conditional random fields, refinement hidden Markov models and neural networks. We tested our approach on problems from three challenging domains: spelling correction for social media, optical character recognition correction and morphological inflection. We demonstrated that our approach performs comparably to strong baselines and state of the art.

While the idea of using redundant epsilons in string transduction problems has been used in the past (Azawi et al., 2013; Schnober et al., 2016), it has traditionally been treated in an ad-hoc narrow manner, for example by adding a fixed number of epsilons after every character. To the best of our knowledge, we are the first to rigorously define this idea as a learning problem while performing a comprehensive empirical evaluation for it, both with respect to the type of datasets used and the sequence labeling algorithms covered.

Acknowledgements

We would like to thank Ariadna Quattoni, Adam Lopez and other members of the Edinburgh NLP group for their feedback and comments on this paper. We would also like to thank Tao Meng for his help with running the BiLSTM experiments. We gratefully acknowledge the support of the the European Union under the Horizon 2020 SUMMA project (grant agreement 688139), and the support of EPSRC (award number 1644752).

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 2004–2015, Vancouver, Canada.

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of the 12th International Conference on Implementation and Application of Automata*, Lecture Notes in Computer Science, pages 11–23, Prague, Czech Republic.

Eiji Aramaki. 2010. Typo corpus. Available at `http://luululu.com/tweet/#cr`.

Mayce Al Azawi, Muhammad Zeshan Afzal, and Thomas M. Breuel. 2013. Normalizing historical orthography for OCR historical documents using LSTM. In *Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing*, pages 80–85, Washington DC, USA.

Harald R. Baayen, Richard Piepenbrock, and Hedderik van Rijn. 1993. *The CELEX lexical data base on CD-ROMs*. Linguistic Data Consortium, University of Pennsylvania.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California, USA.

Raphael Bailly, Xavier Carreras, and Ariadna Quattoni. 2013. Unsupervised spectral learning of finite state transducers. In *Advances in Neural Information Processing Systems 26*, pages 800–808.

Jane Chandlee. 2014. *Strictly local phonological processes*. Ph.D. thesis, Department of Linguistics and Cognitive Science, University of Delaware.

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

Alexander Clark. 2001. Partially supervised learning of morphology with stochastic transducers. In *Proceedings of the 6th Natural Language Processing Pacific Rim Symposium*, pages 341–348, Tokyo, Japan.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2012. Spectral learning of latent-variable PCFGs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 223–231, Jeju Island, Korea.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2013. Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 148–157, Atlanta, Georgia.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2014. Stochastic contextual edit distance and probabilistic FSTs. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 625–630, Baltimore, Maryland.

Arthur Dempster, Natalie Laird, and Donald B. Rubin. 1977. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38.

Markus Dreyer, Jason Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1080–1089, Honolulu, Hawaii.

Markus Dreyer. 2011. *A non-parametric model for the discovery of inflectional paradigms from plain text using graphical models over strings*. Ph.D. thesis, Johns Hopkins University, Baltimore, MD.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 1–8, Philadelphia, Pennsylvania, USA.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 555–560, Berlin, Germany.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.

Kimmo Koskenniemi. 1983. *Two-level morphology*. Ph.D. thesis, University of Helsinki.

Shankar Kumar, Yonggang Deng, and William Byrne. 2006. A weighted finite state transducer translation template model for statistical machine translation. *Natural Language Engineering*, 12(1):35–75.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA.

Shashi Narayan and Shay B. Cohen. 2015. Diversity in spectral learning for natural language parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1868–1878, Lisbon, Portugal.

Jordi Porta and José-Luis Sancho. 2013. Word normalization in Twitter using finite-state transducers. In *Proceedings of the Tweet Normalization Workshop co-located with 29th Conference of the Spanish Society for Natural Language Processing*, pages 49–53, Madrid, Spain.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 623–633, San Diego, California.

Mihaela Rosca and Thomas Breuel. 2016. Sequence-to-sequence neural network models for transliteration. *arXiv preprint:1610.09565*.

Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych. 2016. Still not there? Comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks. In *Proceedings of the 26th International Conference on Computational Linguistics*, pages 1703–1714, Osaka, Japan.

Miikka Silfverberg, Pekka Kauppinenb, and Krister Lindénb. 2016. Data-driven spelling correction using weighted finite-state methods. In *Proceedings of the ACL Workshop on Statistical NLP and Weighted Automata*, pages 51–59, Berlin, Germany.

Karl Stratos, Alexander Rush, Shay B. Cohen, and Michael Collins. 2013. Spectral learning of refinement HMMs. In *Proceedings of the 17th Conference on Computational Natural Language Learning*, pages 56–64, Sofia, Bulgaria.

Deep Neural Networks at the Service of Multilingual Parallel Sentence Extraction

Ahmad Aghaebrahimian
University of Innsbruck
Department of Translation Studies
Herzog-Siegmund-Ufer 15, A-6020 Innsbruck, Austria
Ahmad.Aghaebrahimian@uibk.ac.at

Abstract

Wikipedia provides an invaluable source of parallel multilingual data, which are in high demand for various sorts of linguistic inquiry, including both theoretical and practical studies. We introduce a novel end-to-end neural model for large-scale parallel data harvesting from Wikipedia. Our model is language-independent, robust, and highly scalable. We use our system for collecting parallel German-English, French-English and Persian-English sentences. Human evaluations at the end show the strong performance of this model in collecting high-quality parallel data. We also propose a statistical framework which extends the results of our human evaluation to other language pairs. Our model also obtained a state-of-the-art result on the German-English dataset of BUCC 2017 shared task on parallel sentence extraction from comparable corpora.

Title and Abstract in German

Tiefe Neuronale Netze im Dienste der Extraktion mehrsprachiger paralleler Satzpaare

Wikipedia ist eine überaus wertvolle Quelle von mehrsprachigen Paralleldaten, die für eine Vielzahl von theoretischen und praktischen sprachbezogenen Fragestellungen benötigt werden. Wir stellen ein neuartiges neuronales End-to-End-System für das massenhafte Sammeln von Paralleldaten aus der Wikipedia vor. Das System ist sprachenpaarunabhängig, robust und weist eine hohe Skalierbarkeit auf. Wir nutzen es zur Extraktion von parallelen Satzpaaren in den Sprachenpaaren Deutsch-Englisch, Französisch-Englisch und Persisch-Englisch. Die hohe Genauigkeit unseres Systems wird durch manuelle Evaluation bestätigt. Darüber hinaus stellen wir einen statistischen Ansatz vor, mit dessen Hilfe menschliche Qualitätsurteile auf weitere Sprachenpaare übertragen werden können. Unser System erzielt State-of-the-Art-Ergebnisse gemessen am deutsch-englischen Datensatz der BUCC 2017 Shared Task zur Extraktion von parallelen Satzpaaren aus Vergleichskorpora.

1 Introduction

Parallel texts are an important resource in Natural Language Processing (NLP) applications and tasks. From Statistical and Neural Machine Translation (SMT, NMT) (Brown et al., 1990; Och and Ney, 2002; Kalchbrenner and Blunsom, 2013; Cho et al., 2014), to automatic lexical acquisition (Gale and Church, 1993; Melamed, 1997), cross-lingual Information Retrieval (Davis and Dunning, 1995; Oard, 1997) and annotation projection (Yarowsky et al., 2001; Diab and Resnik, 2002) all are dependent on parallel data.

Generating parallel corpora from scratch is a highly time consuming and expensive task. Therefore, many studies focus on extracting parallel texts from comparable corpora (Munteanu and Marcu, 2005) such as Wikipedia.

Wikipedia is a useful source of parallel sentences since humans already annotated its comparable documents. In Wikipedia, one can find both parallel, and comparable articles[1]. The reason is that some

[1]In a parallel bilingual corpus like Europarl (Koehn, 2005), all sentences in source and target languages are parallel (i.e., translations of each other) while in two comparable corpora the same topic may be described with entirely different sentences.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1372–1383
Santa Fe, New Mexico, USA, August 20-26, 2018.

authors prefer to explain an issue in their own words, while others use existing articles to translate them into their languages. Still, there are others who may combine the two approaches discussed above by borrowing some sentences from a source article and add some other contents by themselves. Generally, even in comparable articles with partial translations, there is a high chance of finding parallel sentences, since both articles are talking about the same topic.

A simple way of retrieving parallel sentences from comparable articles is to align the sentences in source and target pages together using a sentence alignment algorithm (Gale and Church, 1993; Fung and Church, 1994; Wu, 1994; Moore, 2002). However, these aligners are designed to align parallel corpora in which the source and target sentences are in the same order (i.e., no cross-alignment) or in proximity to each other and in which each sentence has only one matching sentence (i.e., no many-to-many alignment). These assumptions are largely violated in comparable corpora.

To address these issues, we designed a novel neural model which estimates a global probability distribution given each pair of sentences in comparable documents. We train an alignment model using German-English parallel data from the Europarl corpus (Koehn, 2005) and use the trained model to extract parallel sentences for the German-English, French-English, and Persian-English language pairs from Wikipedia.

Our model achieved statistically significant improvement over two baselines on Europarl test data (please see Section 5). Since we assumed we had no access to any gold parallel data in Wikipedia, we used human evaluation to determine the performance of our model using a two-tier evaluation design. Still, to make our results more comparable, we applied our model on the German-English dataset of BUCC 2017 second shared task on parallel sentence extraction from comparable corpora (Zweigenbaum et al., 2017) and we obtained a state-of-the-art result on it too.

In this work, we intend to model parallel sentences as accurate as possible. Hence we consider two sentences as parallel only if they have the same semantic content (i.e., convey the same message) and do not have any more or less content that is mentioned in one and missing in another (e.g., an extra or missing prepositional phrase). The sentences that do not satisfy this requirement are considered partial parallel sentences. The accuracy with which each of these partial parallel sentences represents the meaning of their source sentence is expressed in terms of a Normal distribution which is discussed in Section 6.

Moreover, we treat parallel sentences asymmetrically since professional translators often translate from their second language to their native language. Even when the translator's competency level in the source and target languages are the same, the target language could be influenced by the source language. Therefore, we trained all our models on language pairs whose target language is always English and recruited native English speakers for our evaluation tasks.

2 Related Work

Parallel data are considered an asset both in theoretical (e.g., contrastive corpus linguistics, translation studies, language use, and change) and applied (Machine Translation (MT), word sense disambiguation, bilingual lexicography) computational linguistics. There is a wealth of studies on the extraction of parallel data from the Internet in general and Wikipedia in particular. (Adafre and de Rijke, 2006) were among the first researchers who used Wikipedia for parallel data extraction. They generated a pack of source and target documents as the Cartesian product between 30 Wikipedia pages and utilized an MT system to translate target pages into English. Then, they used a similarity measure based on word overlap between the source and target sentences. In another approach, they used matching hyperlinks in Wikipedia pages to identify similar sentences.

To decrease the search space in (Adafre and de Rijke, 2006) work which is evidently too big for large-scale data extraction projects, (Mohammadi and GhasemAghaee, 2010) integrated the idea of length-based sentence alignment in (Gale and Church, 1993)[2] as a heuristic to decrease the complexity of the algorithm.

For larger-scale studies, (Barbosa et al., 2012) used bilingual dictionaries and online translation services (e.g., Google Translate or Microsoft Bing) and (Zhang et al., 2006) proposed the use of aligners

[2]Long source sentences are usually translated into long target ones and short source sentence to short target ones.

Language Pairs	Comparable articles
French-English	1,491,578
German-English	1,247,102
Persian-English	866,408

Table 1: The number of interlanguage links in Wikipedia for selected language pairs. English has interlanguage links with more than 300 other languages.

for content similarity estimation in candidate parallel web pages. (Štromajerová et al., 2016) enhanced Zhang et al.'s system by using Wikipedia's translation templates to locate comparable Czech-English parallel pages and subsequently by using the Hunalign tool (Varga et al., 2005) to extract parallel sentences.

In large-scale data extraction projects, checking all possible sentences for all pages in two languages is neither feasible nor necessary when document-level alignments are already available. (Smith et al., 2010) and (Ştefănescu and Ion, 2013) did their studies on document-aligned articles of Wikipedia. Smith et al. used a feature-based model on aligned documents. Similarly, Ştefănescu and Ion used cross-lingual Wikipedia links embedded within the documents and a trainable model to generate similarity scores for parallel sentence identification.

Classifiers are used for parallel sentence detection as well. (Chu et al., 2014) studied the use of classifiers for parallel sentence identification. They proposed a filtering scheme for Chinese-Japanese language pairs and used a binary classifier on the pruned sentences for parallel sentence classification.

In our work, we let a deep neural architecture learn the most relevant features on its own. We use Interwiki links available in Wikipedia to locate comparable pages. Using an end-to-end deep neural model we extract the most likely parallel sentences given two comparable pages by projecting the sentences into n-dimensional space. In this way, the model learns the most relevant features on its own without knowing much about the source and target languages.

3 Dataset

Wikipedia is an online encyclopedia of human knowledge. As of December 2017, it hosted over 14 million articles in more than 300 languages. While English as the biggest Wikipedia contains more than 3 million articles and 14K active users, there are 28 languages with more than 100K and 60 languages with more than 10K articles. Wikipedia is a crowd-sourced resource of information authored and translated collaboratively on a non-profit basis. Wikipedia provides a collection of similar pages in different languages by linking them together with interlanguage links. These links appear either in a sidebar on the left side or in the text of a page as in-line links. Table 1 represents the number of available English links for German, French, and Persian languages.

Our model is trained on parallel sentences from Europarl and is used to extract parallel sentences from German-English, French-English, and Persian-English comparable pages in Wikipedia. To train our model, we compiled a dataset of first 200K parallel German-English sentences from Europarl. The German sentences were translated using Google translate service and were used as pointers to original target sentences. Online translation in this phase is not a crucial step since a simple word replacement utilizing a dictionary can do the job.

We preprocessed all textual data including source and target sentences by eliminating all non-alphanumeric characters. Numeric characters were changed to 9 to retain the numeric semantic value of numeric tokens. We used 90%, 5%, and 5% of the sentences in our dataset to compile training, validation and test sets respectively in the following way.

Similar to (dos Santos et al., 2014), our model is trained by contrasting positive and negative parallel sentences. Therefore, to compile our training set, given each source sentence, we generated a positive parallel pair by taking its correct target sentence and a negative parallel pair by taking a randomly chosen sentence from target sentences. We make sure that the randomly selected sentence is not the same as the correct target sentence. To compile the validation and test sets, given each source sentence, we take a context of ten surrounding sentences including the correct target sentence. We used these sets to train,

validate and test the alignment system.

When the alignment system is trained and optimized, it is ready to extract parallel sentences from Wikipedia. For parallel sentence extraction, in the first step, we need to find comparable data in Wikipedia. Wikipedia connects comparable pages using interlanguage links. These links are available as an SQL database containing pointers to comparable pages. In Table 1 a few language pairs with their available comparable pages are reported. Using this database, we extract the page contents and use some simple preprocessing tasks to extract sentences from the pages by removing images, tables, graphs, formula, etc. In the end, we package source and target sentences of a comparable pair in one packet.

For each packet, our trained model estimates a probability distribution over all target sentences given each source sentence. In Wikipedia, we do not have access to standard gold data (i.e., we do not know which source-target sentence combination is parallel) hence, we use human evaluation (see Section 6) to estimate the system performance. We report the results of these experiments in Sections 5 and 6

4 Model Architecture

To compare source and target sentences in the mathematical sense, in the first step, we need to project them into n-dimensional space. To do this, we made use of Recurrent Neural Network (RNN) (Elman, 1990) architectures to encode textual strings into vector representations. Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014) are two widely studied variants of RNNs. In our study, we used LSTM cells since they showed better and more stable performance in our experiments.

To provide LSTM layers with their inputs, in the first layer, we used a lookup table to cast words into word embeddings (Equation 1). $\mathbf{W}$ in Equations 1 is the one-hot representation of the words in sentences whose production with pre-trained embedding matrix $\mathbf{E}$ generates word vector $\mathbf{W}_{i,t}$ for the words in the i$^{\text{th}}$ sample sentence each in time step t.

In the next layer, a forward RNN layer accepts word vectors and generates a sequence of vectors for each time step. A similar RNN does the same job but in opposite order to generate backward RNN vectors. We did max pooling ($\mathbf{MP}$) (Equations 3 and 5) over RNN vectors to get the most relevant features and then concatenated them in Equation 6. The final result of this process, $\mathbf{S}$ is a forward and backward vector representation of textual strings. We used this architecture to encode our source sentences.

$$\mathbf{W}_{i,t} = \mathbf{E}^{\intercal}\mathbf{W}_k \tag{1}$$

$$\overrightarrow{\mathbf{S}}_{i,t} = RNN(\overrightarrow{\mathbf{S}}_{i,t-1}, \mathbf{W}_{i,t}) \tag{2}$$

$$\overrightarrow{\mathbf{S}}_{i} = \mathbf{MP}(\overrightarrow{\mathbf{S}}_{i,t}) \tag{3}$$

$$\overleftarrow{\mathbf{S}}_{i,t} = RNN(\overleftarrow{\mathbf{S}}_{i,t+1}, \mathbf{W}_{i,t}) \tag{4}$$

$$\overleftarrow{\mathbf{S}}_{i} = \mathbf{MP}(\overleftarrow{\mathbf{S}}_{i,t}) \tag{5}$$

$$\mathbf{S}_i = [\overrightarrow{\mathbf{S}}_{i}; \overleftarrow{\mathbf{S}}_{i}] \tag{6}$$

Target sentences are encoded like source sentences with an additional attention layer, which helps the encoder to recognize the most relevant features by emphasizing on critical points of the target sentence given each source sentence. Likewise, the target language encoder receives the initial embeddings from a lookup table over a pre-trained embedding matrix. The forward RNN in the next layer transforms the embeddings into a sequence of vectors which finally are fed into an attention layer with attention on source sentence vectors (Equation 7). The resulted vector then is max pooled to be eliminated from non-relevant and useless features. The same process is done for the backward RNN, and the resulting vectors are concatenated.

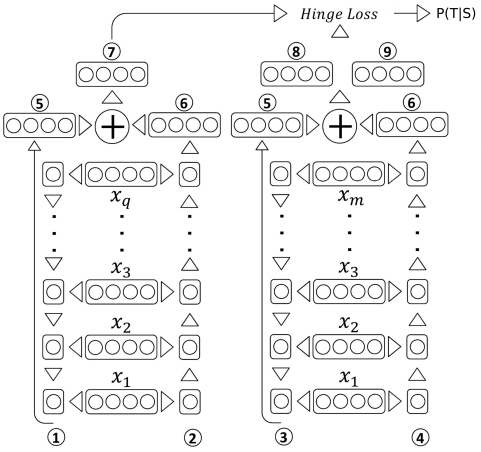


Figure 1: Abstract model architecture. Numbered components in the figure are 1-Forward LSTM, 2-Backward LSTM, 3-Forward Attentive LSTM, 4-Backward Attentive LSTM, 5-Backward max-pooling, 6-Forward max-pooling, 7-Question vector, 8-Positive sample vector, 9-Negative sample vector.

$$
\begin{aligned}
\mathbf{W}_{i,t} &= \mathbf{E}^{\intercal}\mathbf{W}_k \\
\overrightarrow{\mathbf{T}}_{i,t} &= RNN(\overrightarrow{\mathbf{T}}_{i,t-1}, \mathbf{W}_{i,t}) \\
\overrightarrow{\mathbf{T}}_{i,t} &= ATT(\overrightarrow{\mathbf{T}}_{i,t}, \mathbf{S}_i) \qquad (7) \\
\overrightarrow{\mathbf{T}}_{i} &= \mathbf{MP}(\overrightarrow{\mathbf{T}}_{i,t}) \\
\overleftarrow{\mathbf{T}}_{i,t} &= RNN(\overleftarrow{\mathbf{T}}_{i,t+1}, \mathbf{W}_{i,t}) \\
\overrightarrow{\mathbf{T}}_{i,t} &= ATT(\overleftarrow{\mathbf{T}}_{i,t}, \mathbf{S}_i) \qquad (8) \\
\overleftarrow{\mathbf{T}}_{i} &= \mathbf{MP}(\overleftarrow{\mathbf{T}}_{i,t}) \\
\mathbf{T}_i &= [\overrightarrow{\mathbf{T}}_i; \overleftarrow{\mathbf{T}}_i]
\end{aligned}
$$

All target sentences including positive and negative sentences are encoded using this procedure. In the end, source sentence vectors, positive target vectors, and negative target vectors are ready to be used as the inputs of a Hinge objective function. Using this function, we try to maximize the similarity in parallel sentences while minimizing the similarity in non-parallel ones. Therefore, the next step is to measure the similarity between sentences in a parallel set.

The similarity between two vectors can be estimated via different approaches such as Jaccard, Cosine, Polynomial or Manhattan, etc. However, we got better results using the Geometric mean of Euclidean and Sigmoid Dot product (GESD) proposed by (Feng et al., 2015). GESD (Equation 9) combines the angular and forward-line semantic distance between two vectors.

$$SIM(V1, V2) = \frac{1}{1+exp(-(V1 \cdot V2))} * \frac{1}{1+||V1-V2||} \qquad (9)$$

To distinguish parallel sentences from non-parallel ones, we need to train their vectors in a way that increases the similarity for parallel and decreases it for non-parallel sentences. Hinge objective function does the trick for us (Equation 10).

$$\ell = \sum_i \max(0, m + SIM(\mathbf{S}_i, \mathbf{T}_i^-) - SIM(\mathbf{S}_i, \mathbf{T}_i^+)) \qquad (10)$$

After training, the model estimates a probability for each pair of source and target sentences (i.e., $p(target|source)$). In the next section, we describe how we use these probabilities to distinguish parallel sentences from non-parallel ones.

5 Results

We trained our model on the dataset compiled from Europarl parallel data. We used 128-dimensional LSTMs for all RNNs in our model, and for the embedding layer, we used GloVe word vectors (Pennington et al., 2014). To prevent the model from over-fitting, we set the drop-out rate to 0.5 for the last layer

in each module. We used an attention model similar to the one proposed by (dos Santos et al., 2014). The model was trained on two GPUs and converged after around 3 hours of training. We used random assignment and the Bleualign tool (Sennrich and Volk, 2011) as two baselines. The results of the model on the test and validation sets are reported in Table 2. These results are the accuracy of the system for sentence alignment on the German-English dataset. Since our model is trained with source sentences translated into English, the same encoder can be used for any other source languages as long as we use the same mechanism for translation.

Up to this point, we trained our aligner. In the next step, we use this alignment system to extract parallel sentences. To estimate the performance of the model on parallel sentence extraction from Wikipedia, we designed a two-tier human evaluation, which is described in the next section.

Europarl German-English	**Baseline**	**Bleualign**	**This work**
Validation set	11.94 %	92.45 %	**96.83 %**
Test set	11.34 %	93.05 %	**97.24 %**

Table 2: Comparative evaluation of German-English alignment system. The baseline is random target sentence assignment.

6 Human Evaluation

As described in Section 3, our comparable data extraction yields more than one million comparable packets for the German and French and around 800K packets for the Persian languages (Table 1). Each comparable packet consists of two files, one of which contains source text lines and the other includes target text lines.

For each comparable packet, the model as described in Section 4 estimates a probability distribution over all target sentences given each source sentence. Each source sentence and the target sentence with the highest probability forms a probable parallel pair.

Not having access to standard gold data in Wikipedia, we do not know which of these pairs are, in fact, parallel, irrespective of the probability estimated by the model.

To estimate a threshold for the probabilities and to validate it, we designed a two-step human evaluation procedure. In the first step, we establish a threshold for the model-generated probabilities above which a sentence pair could be considered a parallel pair. Irrespective of the size of each comparable packet, this probability is a global metric of how much two sentences are semantically correlated. So we only need to establish a threshold and validate it using statistical inference. After that, we can accept source and target sentences with probabilities above the threshold as correct parallel sentences.

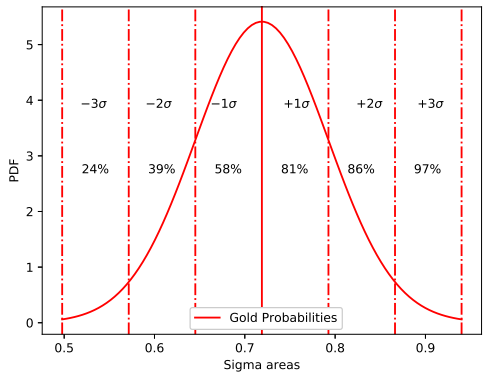
In the second step, we validate our thresholds by asking our evaluators to decide which of the extracted pairs are parallel. For this purpose, we randomly extract some sentence pairs under the curve of a Normal distribution parametrized by μ and σ obtained from the last step and ask some evaluators to decide which pairs are parallel and which ones are not.

Based on the information obtained from the evaluators, we try to reject the null hypothesis of our study (i.e., sentence pairs with a probability above the thresholds are not parallel) and to analyze the errors qualitatively.

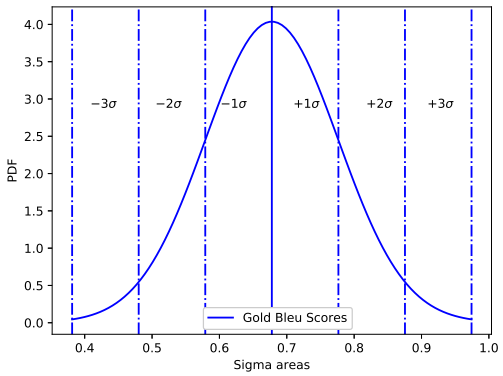
6.1 Establishing a Threshold

To establish a threshold for the model, we randomly extracted two hundred German sentences and asked two native English and fluent German translators to translate them into English. We returned these translations and their German source sentences into their original containing packets to provide them with their original contexts. Knowing that these are correct parallel sentences, we used our model to estimate their probabilities. Then, using these probabilities, we computed the μ and σ of the samples based on which we estimated a Normal distribution on the probability range of parallel sentences (Figure 2a).

Moreover, using Google translate service, we converted these sentences into English and computed their BLEU scores (Papineni et al., 1993) by using our gold translations produced by our translators as



(a) Normal Distribution on System Probabilities



(b) Normal Distribution on BLEU Scores

Figure 2: (a): Normal distribution parametrized by $\mu = 0.71$ and $\sigma = 0.07$ obtained from gold parallel sentences translated by human translators and scored by the model. The percentage in each σ region is the ratio of the sentence pairs evaluated as parallel in the second task.
(b): Normal distribution ($\mu = 0.67, \sigma = 0.09$) over the BLEU scores of English translations of 200 true parallel target sentences done by Google translate service. BLEU scores are transformed to percent.

the reference. We calculated the average of these two scores and used them for estimating a Normal distribution by computing their μ and σ (Figure 2b).

In any further parallel sentence extraction, to estimate the performance of the model without human evaluation, we only need to translate target sentences into English, to compute their BLEU score using source sentences as a reference and to compare them with the BLEU score curve obtained above. In the next section, we use this method to estimate the quality of extracted parallel sentences for French-English and Persian-English language pairs.

	-3σ	-2σ	-1σ	1σ	2σ	3σ
Observed Agreement	95 %	76 %	71 %	76 %	85 %	98 %
Cohen's kappa	85 %	69 %	45 %	50 %	81 %	89 %
Prediction Reliability	Strong	Average	Weak	Average	Strong	very Strong
Model probability Range	50%-57%	57%-64%	64%-71%	71%-78%	78%-85%	85%-92%
Bleu Score Range	40%-49%	49%-58%	58%-67%	67%-76% %	76%-85%	85%-94%
Collected parallel Sentences	24 %	39 %	58 %	81 %	86 %	97 %

Table 3: Human evaluation results. The Observed Agreement is the ratio of sentence pairs which were evaluated the same, either as true or false parallel pairs by both evaluators. Cohen's Kappa integrates chance agreement into the Observed Agreement. As mentioned in Prediction Reliability row, σ regions with Cohen's Kappa less than 0.8 are not reliably evaluated. Model Probability Ranges are the probabilities estimated by the model for 200 correct parallel sentences compile by our evaluators, and the BLEU Score Range is the BLEU scores of these sentences when translated using Google Translate. The ranges for last two rows are computed using a Normal distribution on their data. Collected Parallel Sentences are the ratio of sentence pairs which are evaluated as parallel in each σ region.

6.2 Validating the Thresholds

In Figure 2a the Normal distribution and the percentage of parallel data in each σ regions are illustrated. To validate these thresholds, we randomly and evenly sampled 1000 sentences from all σ regions and asked our evaluators to determine whether they are parallel or not. We statistically analyzed these data to assess the validity of the outcomes.

Given that our data in this task are nominal and have no order and that it is necessary to take into account the probability of chance in evaluation, we used Cohen's kappa to estimate the inter-rater reliability. We calculated the observed agreement probability for each σ region as well. The data analysis is done by considering $p > 0.95$ and 1000 data samples. The results are reported in Table 3. As is shown in this table, in $2\sigma^+$ and $3\sigma^+$ regions we have 81% and 89% inter-rater reliability accordingly, which indicates strong predictability in these regions.

The observed agreement and collected parallel sentences in these regions are quite high. In $3\sigma^+$ and $2\sigma^+$ regions, we managed to reject the null hypothesis (i.e., that all sentences in these regions are non-parallel), hence confirming that with 95% confidence, the established thresholds in these regions are statistically sufficient enough to decide whether a pair is parallel or not.

Since the thresholds are computed using the same scheme for both languages, the obtained results are valid for all language pairs. To prove this idea (i.e., the validity of the determined thresholds for other languages without human validation), we used this system for French-English and Persian-English language pairs as well. We randomly extracted 1000 pairs from the Normal distribution over Fr-En and Fa-En pairs and used Google translate service to generate the translations of these sentences in English. Then, we computed the BLEU scores of the English sentences in parallel sentences and compared them to the Normal curve of BLEU scores in Section 6.1. Since the translation service is the same, the computed BLEU scores are comparable. We observed that 98% of French-English and 96% of Persian-English sentence pairs, which fall in $3\sigma^+$ and $2\sigma^+$ regions of the model probability curve, are in the range of $2\sigma^+$ and $3\sigma^+$ area of the BLEU score Normal curve, too. This gives us an estimate of the quality of parallel data as well. As we discussed earlier, we are interested in perfect parallel sentences which are clustered in the highest σ regions. However partial parallel sentences in lower regions can be used for certain purposes too.

As we see in Table 3 there is a high correlation between the scores generated by the model and the BLEU scores of the parallel sentences. Therefore, we argue that irrespective of the language, the model is capable of extracting parallel sentences from any available language pair in Wikipedia with at least 95% accuracy in the last two σ regions. For other σ regions, although the confidence levels and respective accuracies are lower than the higher regions, partial parallel sentences still can be identified. In Figure 3 some parallel sentences with their probabilities estimated by the model are presented.

At the end to compare our results with other similar works, we applied our model on the German-English dataset of BUCC 2017 second shared task on parallel sentence extraction from comparable corpora (Zweigenbaum et al., 2017) and we obtained a state-of-the-art result on it. The results are presented in Table 4

Model	**Precision**	**Recall**	**F1**
VIC (Azpeitia et al., 2017)	88%	80%	84%
This work	**89%**	**83%**	**86%**

Table 4: System results on the German-English dataset of the second shared task of BUCC 2017

7 Error Analysis

As a qualitative study and a complement to the second task, we asked our evaluators to mention a reason why they think a pair of sentences might not be parallel. Based on a short data inquiry, we provided the evaluators with five major error types. We asked them to expand the list of errors if none of the provided error types explains why a given sentence pair is not parallel. They added other three errors to our list. These reasons are listed in Box 1. We can categorize these errors in noncritical, neutral and critical categories.

Sentences with noncritical errors such as error 4, 6 and 8 have slight problems and can be considered parallel in some cases. However, to enhance the quality of parallel sentences, these sentences are excluded from the system output. Neutral errors like errors 5 and 7 do not lead to a significant decrease in system performance, while critical mistakes like errors 1, 2, and 3, cause severe system malfunctioning.

Figure 3: Parallel Sentences and their model generated probabilities. The two boxes in the first row are gold parallel sentences translated by our translators. We returned the sentences to their original documents and used our model to estimate their probabilities. The two boxes in the second and third rows are test English-French and English-Persian sentences, respectively.

In the following, we analyze each of these categories in detail.

1. Errors 4 and 6 are opposite of each other: either source or target sentence contains more information than its counterpart. It is mainly caused by translators adding information to the translated version. In case 8, there are minor discrepancies between source and target sentences. Some of these cases can be considered parallel pairs though. The majority of errors in $3\sigma^+$ and $2\sigma^+$ belong to this category.

2. In case 5, a semi-parallel sentence is detected as a parallel sentence. However, the two sentences do not have enough semantic overlap and are therefore not parallel. In case 7, despite dealing with the same topic, sentence pairs have diverging contexts and settings and are consequently not parallel. We found a few errors of this type only in the $1\sigma^+$ region.

3. In case 1, the source and target sentences are different with no semantic overlap. In case 2, the source or target sentence is incomplete, which is caused by a malfunctioning sentence segmenter. In case 3, extracted source and target sentences are in the same language, which occasionally happens when

- for whatever reason - some sentences are not translated in the target text. The majority of errors in $3\sigma^-$, $2\sigma^-$, and $1\sigma^-$ belong to this category.

The errors in the first category require more elaborate linguistic analysis to be eradicated than the errors in the third category. The errors in the second category are barely detrimental to the system performance.

Default Reasons:
1- Totally different sentences
2- Incomplete sentences
3- Source and target sentences are in the same language.
4- More than half of the meaning is conveyed but not parallel.
5- Less than half of the meaning is conveyed.

Added Reasons:
6- Target sentence is correct but contains more information than the source sentence.
7- Information incorrect/different
8- Small detail(s) missing

Box 1: Human-judged reasons for lacking parallelity of extracted sentence pairs. Although items 4, 6, and 8 are strong candidates for parallel sentences, they are excluded from parallel sentence extraction to increase overall quality. Items 5 and 7 are trivial to the system performance. Items 1, 2, and 3 are serious system errors.

8 Conclusion

We introduced a language-independent parallel sentence extractor using an end-to-end deep neural network architecture. Our system extracts parallel sentences form comparable pages in Wikipedia. Using the gold parallel data compiled by our human evaluators for the German-English language pair, we showed that the system is highly accurate in extracting parallel sentences in other languages as well. Using the system thresholds estimated by human evaluation, we extracted high-quality parallel sentences for German-English, French-English, and Persian-English language pairs. Our model also obtained a state-of-the-art result on the German-English dataset of BUCC 2017 second shared task. In future work, we aim to improve the system by eradicating its errors and performing the translation step seamlessly without the need for any external translation services[3].

Acknowledgments

This work is part of the project ”TransBank: A Meta-Corpus for Translation Research”, funded by the Austrian Academy of Sciences (grant number GD 2016/56). The author thanks Michael Ustaszewski and Andy Stauder from TransBank group at the University of Innsbruck for their thoughtful comments on the final draft. He also appreciates Esther May Rathbone and Carolyn R. Atzl for their participation in human evaluation task. The computational results presented in this work have been achieved (in part) using the HPC infrastructure LEO of the University of Innsbruck.

[3]The data of this work are available at https://github.com/ExperimentalTransbank/WikiParal

References

Sisay Fissaha Adafre and Maarten de Rijke. 2006. Finding similar sentences across multiple languages in wikipedia. In *Proceedings of the EACL Workshop on New Text*.

Andoni Azpeitia, Thierry Etchegoyhen, and Eva Martinez Garcia. 2017. Weighted set-theoretic alignment of comparable sentences. In *Proceedings of the 10th Workshop on Building and Using Comparable Corpora. Association for Computational Linguistics (ACL)*.

Luciano Barbosa, Vivek Kumar Rangarajan Sridhar, Mahsa Yarmohammadi, and Srinivas Bangalore. 2012. Harvesting parallel text in multiple languages with limited supervision. In *Proceedings of The Conference on Computational Linguistics (COLING)*.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of the Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Chenhui Chu, Toshiaki Nakazawa, and Sadao Kurohashi. 2014. Constructing a chinese–japanese parallel corpus from wikipedia. In *Proceedings of the 9th Conference on International Language Resources and Evaluation*.

Mark W. Davis and Ted Dunning. 1995. A trec evaluation of query translation methods for multi-lingual text retrieval. In *Proceedings of Text Retrieval Conference (TREC)*.

Mona Diab and Philip Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL)*.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2014. Attentive pooling networks. *arXiv:1602.03609v1*.

Jeffry L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2).

Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: a study and an open task. In *Proceedings of IEEE ASRU Workshop*.

Pascale Fung and Kenneth Ward Church. 1994. Kvec: A new approach for aligning parallel texts. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*.

William A. Gale and Kenneth W. Church. 1993. A program for aligning sentences in bilingual corpora. *Computational Linguistics*.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit X*.

I. Dan Melamed. 1997. A portable algorithm for mapping bitext correspondence. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*.

Mehdi Mohammadi and Nasser GhasemAghaee. 2010. Building bilingual parallel corpora based on wikipedia. In *Proceedings of International Conference on Computer Engineering and Applications*.

Robert C. Moore. 2002. Fast and accurate sentence alignment of bilingual corpora. In *Lecture Notes in Computer Science*.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4).

Douglas W. Oard. 1997. Cross-language text retrieval research in the usa. In *Proceedings of the Third DELOS Workshop on Cross-Language Information Retrieval*.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kishore A. Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 1993. Bleu: a method for automatic evaluation of machine translation. *Technical Report RC22176 (W0109-022), IBM Research Division, Thomas J. Watson Research Center*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Rico Sennrich and Martin Volk. 2011. Iterative, mt-based sentence alignment of parallel texts. In *Proceedings of the 18th Nordic Conference of Computational Linguistics (COLING)*.

Jason R. Smith, Chris Quirk, and Kristina Toutanova. 2010. Extracting parallel sentences from comparable corpora using document level alignment. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Daniel Varga, Peter Halacsy, Andras Kornai, Viktor Nagy, Laszlo Nemeth, and Viktor Tron. 2005. Parallel corpora for medium density languages. In *Proceedings of the RANLP 2005*.

Dekai Wu. 1994. Aligning a parallel english-chinese corpus statistically with lexical criteria. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the First International Conference on Human Language Technology Research*.

Ying Zhang, Ke Wu, Jianfeng Gao, and Phil Vines. 2006. Automatic acquisition of chinese-english parallel corpus from the web. In *Proceedings of the 28th European Conference on Information Retrieval*.

Pierre Zweigenbaum, Serge Sharoff, and Reinhard Rapp. 2017. Overview of the second bucc shared task: Spotting parallel sentences in comparable corpora. In *Proceedings of the 10th Workshop Building Using Comparable Corpora*.

Dan Ştefănescu and Radu Ion. 2013. Parallel-wiki: A collection of parallel sentences extracted from wikipedia. In *Proceedings of the 14th Conference on Intelligent Text Processing and Computational Linguistics*.

Adéla Štromajerová, Vít Baisa, and Marek Blahuš. 2016. Between comparable and parallel: English-czech corpus from wikipedia. In *Proceedings of Advances in Slavonic Natural Language Processing*.

Diachronic word embeddings and semantic shifts: a survey

Andrey Kutuzov Lilja Øvrelid Terrence Szymanski[◇] Erik Velldal
University of Oslo, Norway
{andreku | liljao | erikve}@ifi.uio.no
[◇]ANZ, Melbourne, Australia
terry.szymanski@gmail.com

Abstract

Recent years have witnessed a surge of publications aimed at tracing temporal changes in lexical semantics using distributional methods, particularly prediction-based word embedding models. However, this vein of research lacks the cohesion, common terminology and shared practices of more established areas of natural language processing. In this paper, we survey the current state of academic research related to diachronic word embeddings and semantic shifts detection. We start with discussing the notion of semantic shifts, and then continue with an overview of the existing methods for tracing such time-related shifts with word embedding models. We propose several axes along which these methods can be compared, and outline the main challenges before this emerging subfield of NLP, as well as prospects and possible applications.

1 Introduction

The meanings of words continuously change over time, reflecting complicated processes in language and society. Examples include both changes to the core meaning of words (like the word *gay* shifting from meaning 'carefree' to 'homosexual' during the 20th century) and subtle shifts of cultural associations (like *Iraq* or *Syria* being associated with the concept of 'war' after armed conflicts had started in these countries). Studying these types of changes in meaning enables researchers to learn more about human language and to extract temporal-dependent data from texts.

The availability of large corpora and the development of computational semantics have given rise to a number of research initiatives trying to capture *diachronic semantic shifts* in a data-driven way. Recently, *word embeddings* (Mikolov et al., 2013b) have become a widely used input representation for this task. There are dozens of papers on the topic, mostly published after 2011 (we survey them in Section 3 and further below). However, this emerging field is highly heterogenous. There are at least three different research communities interested in it: natural language processing (and computational linguistics), information retrieval (and computer science in general), and political science. This is reflected in the terminology, which is far from being standardized. One can find mentions of 'temporal embeddings,' 'diachronic embeddings,' 'dynamic embeddings,' etc., depending on the background of a particular research group. The present survey paper attempts to describe this diversity, introduce some axes of comparison and outline main challenges which the practitioners face. Figure 1 shows the timeline of events that influenced the research in this area: in the following sections we cover them in detail.

This survey is restricted in scope to research which traces semantic shifts using distributional word embedding models (that is, representing lexical meaning with dense vectors produced from co-occurrence data). We only briefly mention other data-driven approaches also employed to analyze temporal-labeled corpora (for example, topic modeling). Also, we do not cover syntactic shifts and other changes in the functions rather than meaning of words.

The paper is structured as follows. In Section 2 we introduce the notion of 'semantic shift' and provide some linguistic background for it. Section 3 aims to compare different approaches to the task of

Proceedings of the 27th International Conference on Computational Linguistics, pages 1384–1397
Santa Fe, New Mexico, USA, August 20-26, 2018.

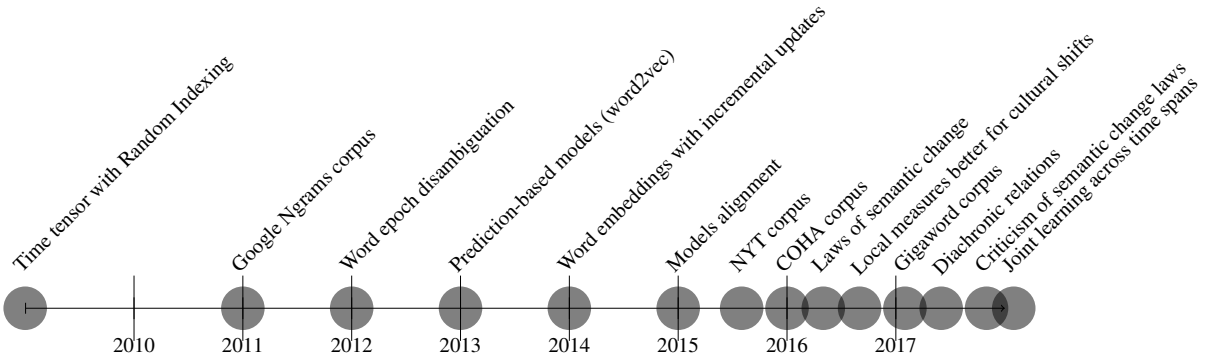


Figure 1: Distributional models in the task of tracing diachronic semantic shifts: research timeline

automatic detection of semantic shifts: in the choice of diachronic data, evaluation strategies, methodology of extracting semantic shifts from data, and the methods to compare word vectors across time spans. Sections 4 and 5 describe two particularly interesting results of diachronic embeddings research: namely, the statistical laws of semantic change and temporal semantic relations. In Section 6 we outline possible applications of systems that trace semantic shifts. Section 7 presents open challenges which we believe to be most important for the field, and in Section 8 we summarize and conclude.

2 The concept of semantic shifts

Human languages change over time, due to a variety of linguistic and non-linguistic factors and at all levels of linguistic analysis. In the field of theoretical (diachronic) linguistics, much attention has been devoted to expressing regularities of linguistic change. For instance, laws of phonological change have been formulated (e.g., Grimm's law or the great vowel shift) to account for changes in the linguistic sound system. When it comes to lexical semantics, linguists have studied the evolution of word meaning over time, describing so-called lexical *semantic shifts* or *semantic change*, which Bloomfield (1933) defines as "innovations which change the lexical meaning rather than the grammatical function of a form."

Historically, much of the theoretical work on semantic shifts has been devoted to documenting and categorizing various types of semantic shifts (Bréal, 1899; Stern, 1931; Bloomfield, 1933). The categorization found in Bloomfield (1933) is arguably the most used and has inspired a number of more recent studies (Blank and Koch, 1999; Geeraerts, 1997; Traugott and Dasher, 2001). Bloomfield (1933) originally proposed nine classes of semantic shifts, six of which are complimentary pairs along a dimension. For instance, the pair 'narrowing' – 'broadening' describes the observation that word meaning often changes to become either more specific or more general, e.g. Old English *mete* 'food' becomes English *meat* 'edible flesh,' or that the more general English word *dog* is derived from Middle English *dogge* which described a dog of a particular breed. Bloomfield (1933) also describes change along the spectrum from positive to negative, describing the speaker's attitude as one of either degeneration or elevation, e.g. from Old English *cniht* 'boy, servant' to the more elevated *knight*.

The driving forces of semantic shifts are varied, but include linguistic, psychological, sociocultural or cultural/encyclopedic causes (Blank and Koch, 1999; Grzega and Schoener, 2007). Linguistic processes that cause semantic shifts generally involve the interaction between words of the vocabulary and their meanings. This may be illustrated by the process of ellipsis, whereby the meaning of one word is transferred to a word with which it frequently co-occurs, or by the need for discrimination of synonyms caused by lexical borrowings from other languages. Semantic shifts may be also be caused by changes in the attitudes of speakers or in the general environment of the speakers. Thus, semantic shifts are naturally separated into two important classes: linguistic drifts (slow and regular changes in core meaning of words) and cultural shifts (culturally determined changes in associations of a given word). Researchers studying semantic shifts from a computational point of view have shown the existence of this division empirically (Hamilton et al., 2016c). In the traditional classification of Stern (1931), the semantic shift category of *substitution* describes a change that has a non-linguistic cause, namely that of technologi-

cal progress. This may be exemplified by the word *car* which shifted its meaning from non-motorized vehicles after the introduction of the automobile.

The availability of large corpora have enabled the development of new methodologies for the study of lexical semantic shifts within general linguistics (Traugott, 2017). A key assumption in much of this work is that changes in a word's collocational patterns reflect changes in word meaning (Hilpert, 2008), thus providing a usage-based account of semantics (Gries, 1999). For instance, Kerremans et al. (2010) study the very recent neologism *detweet*, showing the development of two separate usages/meanings for this word ('to delete from twitter,' vs 'to avoid tweeting') based on large amounts of web-crawled data. The usage-based view of lexical semantics aligns well with the assumptions underlying the distributional semantic approach (Firth, 1957) often employed in NLP . Here, the time spans studied are often considerably shorter (decades, rather than centuries) and we find that these distributional methods seem well suited for monitoring the gradual process of meaning change. Gulordava and Baroni (2011), for instance, showed that distributional models capture cultural shifts, like the word *sleep* acquiring more negative connotations related to sleep disorders, when comparing its 1960s contexts to its 1990s contexts.

To sum up, semantic shifts are often reflected in large corpora through change in the context of the word which is undergoing a shift, as measured by co-occurring words. It is thus natural to try to detect semantic shifts automatically, in a 'data-driven' way. This vein of research is what we cover in the present survey. In the following sections, we overview the methods currently used for the automatic detection of semantic shifts and the recent academic achievements related to this problem.

3 Tracing semantic shifts distributionally

Conceptually, the task of discovery of semantic shifts from data can be formulated as follows. Given corpora $[C_1, C_2, ...C_n]$ containing texts created in time periods $[1, 2, ...n]$, the task is to locate words with different meaning in different time periods, or to locate the words which changed most. Other related tasks are possible: discovering general trends in semantic shifts (see Section 4) or tracing the dynamics of the relationships between words (see Section 5). In the next subsections, we address several axes along which one can categorize the research on detecting semantic shifts with distributional models.

3.1 Sources of diachronic data for training and testing

When automatically detecting semantic shifts, the types of generalizations we will be able to infer are influenced by properties of the textual data being used, such as the source of the datasets and the temporal granularity of the data. In this subsection we discuss the data choices made by researchers (of course, not pretending to cover the whole range of the diachronic corpora used).

3.1.1 Training data

The time unit (the granularity of the temporal dimension) can be chosen before slicing the text collection into subcorpora. Earlier works dealt mainly with long-term semantic shifts (spanning decades or even centuries), as they are easier to trace. One of the early examples is Sagi et al. (2011) who studied differences between Early Middle, Late Middle and Early Modern English, using the Helsinki Corpus (Rissanen and others, 1993).

The release of the Google Books Ngrams corpus[1] played an important role in the development of the field and spurred work on the new discipline of 'culturomics,' studying human culture through digital media (Michel et al., 2011). Mihalcea and Nastase (2012) used this dataset to detect differences in word usage and meaning across 50-years time spans, while Gulordava and Baroni (2011) compared word meanings in the 1960s and in the 1990s, achieving good correlation with human judgments. Unfortunately, Google Ngrams is inherently limited in that it does not contain full texts. However, for many cases, this corpus was enough, and its usage as the source of diachronic data continued in Mitra et al. (2014) (employing syntactic ngrams), who detected word sense changes over several different time periods spanning from 3 to 200 years.

[1] https://books.google.com/ngrams

In more recent work, time spans tend to decrease in size and become more granular. In general, corpora with smaller time spans are useful for analyzing socio-cultural semantic shifts, while corpora with longer spans are necessary for the study of linguistically motivated semantic shifts. As researchers are attempting to trace increasingly subtle cultural semantic shifts (more relevant for practical tasks), the granularity of time spans is decreasing: for example, Kim et al. (2014) and Liao and Cheng (2016) analyzed the *yearly* changes of words. Note that, instead of using granular 'bins', time can also be represented as a continuous differentiable value (Rosenfeld and Erk, 2018).

In addition to the Google Ngrams dataset (with granularity of 5 years), Kulkarni et al. (2015) used Amazon Movie Reviews (with granularity of 1 year) and Twitter data (with granularity of 1 month). Their results indicated that computational methods for the detection of semantic shifts can be robustly applied to time spans less than a decade. Zhang et al. (2015) used another yearly text collection, the New-York Times Annotated Corpus (Sandhaus, 2008), again managing to trace subtle semantic shifts. The same corpus was employed by Szymanski (2017), with 21 separate models, one for each year from 1987 to 2007, and to some extent by Yao et al. (2018), who crawled the NYT web site to get 27 yearly subcorpora (from 1990 to 2016). The inventory of diachronic corpora used in tracing semantic shifts was expanded by Eger and Mehler (2016), who used the Corpus of Historical American (COHA[2]), with time slices equal to one decade. Hamilton et al. (2016a) continued the usage of COHA (along with the Google Ngrams corpus). Kutuzov et al. (2017b) started to employ the yearly slices of the English Gigaword corpus (Parker et al., 2011) in the analysis of cultural semantic drift related to armed conflicts.

3.1.2 Test sets

Diachronic corpora are needed not only as a source of *training* data for developing semantic shift detection systems, but also as a source of *test* sets to evaluate such systems. In this case, however, the situation is more complicated. Ideally, diachronic approaches should be evaluated on human-annotated lists of semantically shifted words (ranked by the degree of the shift). However, such gold standard data is difficult to obtain, even for English, let alone for other languages. General linguistics research on language change like that of Traugott and Dasher (2001) and others usually contain only a small number of hand-picked examples, which is not sufficient to properly evaluate an automatic unsupervised system.

Various ways of overcoming this problem have been proposed. For example, Mihalcea and Nastase (2012) evaluated the ability of a system to detect the time span that specific contexts of a word undergoing a shift belong to (*word epoch disambiguation*). A similar problem was offered as SemEval-2015 Task 7: 'Diachronic Text Evaluation' (Popescu and Strapparava, 2015). Another possible evaluation method is so-called cross-time alignment, where a system has to find equivalents for certain words in different time periods (for example, 'Obama' in 2015 corresponds to 'Trump' in 2017). There exist several datasets containing such temporal equivalents for English (Yao et al., 2018). Yet another evaluation strategy is to use the detected diachronic semantic shifts to trace or predict real-world events like armed conflicts (Kutuzov et al., 2017b). Unfortunately, all these evaluation methods still require the existence of large manually annotated semantic shift datasets. The work to properly create and curate such datasets is in its infancy.

One reported approach to avoid this requirement is borrowed from research on word sense disambiguation and consists of making a synthetic task by merging two real words together and then modifying the training and test data according to a predefined sense-shifting function. Rosenfeld and Erk (2018) successfully employed this approach to evaluate their system; however, it still operates on synthetic words, limiting the ability of this evaluation scheme to measure the models' performance with regards to real semantic shift data. Thus, the problem of evaluating semantic shift detection approaches is far from being solved, and practitioners often rely on self-created test sets, or even simply manually inspecting the results.

3.2 Methodology of extracting semantic shifts from data

After settling on a diachronic data set to be used in the system, one has to choose the methods to analyze it. Before the broad adoption of word embedding models, it was quite common to use change in raw

[2] http://corpus.byu.edu/coha/

word frequencies in order to trace semantic shifts or other kinds of linguistic change; see, among others, Juola (2003), Hilpert and Gries (2009), Michel et al. (2011), Lijffijt et al. (2012), or Choi and Varian (2012) for frequency analysis of words in web search queries. Researchers also studied the increase or decrease in the frequency of a word A collocating with another word B over time, and based on this inferred changes in the meaning of A (Heyer et al., 2009).

However, it is clear that semantic shifts are not always accompanied with changes in word frequency (or this connection may be very subtle and non-direct). Thus, if one were able to more directly model word meaning, such an approach should be superior to frequency-proxied methods. A number of recent publications have showed that *distributional word representations* (Turney et al., 2010; Baroni et al., 2014) provide an efficient way to solve these tasks. They represent meaning with sparse or dense (embedding) vectors, produced from word co-occurrence counts. Although conceptually the source of the data for these models is still word frequencies, they 'compress' this information into continuous lexical representations which are both efficient and convenient to work with. Indeed, Kulkarni et al. (2015) explicitly demonstrated that distributional models outperform the frequency-based methods in detecting semantic shifts. They managed to trace semantic shifts more precisely and with greater explanatory power. One of the examples from their work is the semantic evolution of the word *gay*: through time, its nearest semantic neighbors changed, manifesting the gradual move away from the sense of 'cheerful' to the sense of 'homosexual.'

In fact, distributional models were being used in diachronic research long before the paper of Kulkarni et al. (2015), although there was no rigorous comparison to the frequentist methods. Already in 2009, it was proposed that one can use distributional methods to detect semantic shifts in a quantitative way. The pioneering work by Jurgens and Stevens (2009) described an insightful conceptualization of a sequence of distributional model updates through time: it is effectively a Word:Semantic Vector:Time tensor, in the sense that each word in a distributional model possesses a set of semantic vectors for each time span we are interested in. It paved the way for quantitatively comparing not only words with regard to their meaning, but also different stages in the development of word meaning over time.

Jurgens and Stevens (2009) employed the *Random Indexing* (RI) algorithm (Kanerva et al., 2000) to create word vectors. Two years later, Gulordava and Baroni (2011) used explicit count-based models, consisting of sparse co-occurrence matrices weighted by Local Mutual Information, while Sagi et al. (2011) turned to Latent Semantic Analysis (Deerwester et al., 1990). In Basile et al. (2014), an extension to RI dubbed *Temporal Random Indexing* (TRI) was proposed. However, no quantitative evaluation of this approach was offered (only a few hand-picked examples based on the Italian texts from the *Gutenberg Project*), and thus it is unclear whether TRI is any better than other distributional models for the task of semantic shift detection.

Further on, the diversity of the employed methods started to increase. For example, Mitra et al. (2014) analyzed clusters of the word similarity graph in the subcorpora corresponding to different time periods. Their distributional model consisted of lexical nodes in the graphs connected with weighted edges. The weights corresponded to the number of shared most salient syntactic dependency contexts, where saliency was determined by co-occurrence counts scaled by Mutual Information (MI). Importantly, they were able to detect not only the mere fact of a semantic shift, but also its type: the birth of a new sense, splitting of an old sense into several new ones, or merging of several senses into one. Thus, this work goes into a much less represented class of 'fine-grained' approaches to semantic shift detection. It is also important that Mitra et al. (2014) handle natively the issue of polysemous words, putting the much-neglected problem of word senses in the spotlight.

The work of Kim et al. (2014) was seminal in the sense that it is arguably the first one employing *prediction-based word embedding models* to trace diachronic semantic shifts. Particularly, they used incremental updates (see below) and Continuous Skipgram with negative sampling (SGNS) (Mikolov et al., 2013a).[3] Hamilton et al. (2016a) showed the superiority of SGNS over explicit PPMI-based distributional models in semantic shifts analysis, although they noted that low-rank SVD approximations (Bullinaria and Levy, 2007) can perform on par with SGNS, especially on smaller datasets. Since then,

[3]Continuous Bag-of-Words (CBOW) from the same paper is another popular choice for learning semantic vectors.

the majority of publications in the field started using dense word representations: either in the form of SVD-factorized PPMI matrices, or in the form of prediction-based shallow neural models like SGNS.[4]

There are some works employing other distributional approaches to semantic shifts detection. For instance, there is a strong vein of research based on dynamic topic modeling (Blei and Lafferty, 2006; Wang and McCallum, 2006), which learns the evolution of topics over time. In Wijaya and Yeniterzi (2011), it helped solve a typical digital humanities task of finding traces of real-world events in the texts. Heyer et al. (2016) employed topic analysis to trace the so-called 'context volatility' of words. In the political science, topic models are also sometimes used as proxies to social trends developing over time: for example, Mueller and Rauh (2017) employed LDA to predict timing of civil wars and armed conflicts. Frermann and Lapata (2016) drew on these ideas to trace diachronic word senses development. But most scholars nowadays seem to prefer parametric distributional models, particularly prediction-based embedding algorithms like SGNS, CBOW or GloVe (Pennington et al., 2014). Following their widespread adoption in NLP in general, they have become the dominant representations for the analysis of diachronic semantic shifts as well.

3.3 Comparing vectors across time

It is rather straightforward to train separate word embedding models using time-specific corpora containing texts from several different time periods. As a consequence, these models are also time-specific. However, it is not that straightforward to compare word vectors across different models.

It usually does not make sense to, for example, directly calculate cosine similarities between embeddings of one and the same word in two different models. The reason is that most modern word embedding algorithms are inherently stochastic and the resulting embedding sets are invariant under rotation. Thus, even when trained on the same data, separate learning runs will produce entirely different numerical vectors (though with roughly the same pairwise similarities between vectors for particular words). This is expressed even stronger for models trained on different corpora. It means that even if word meaning is completely stable, the direct cosine similarity between its vectors from different time periods can still be quite low, simply because the random initializations of the two models were different. To alleviate this, Kulkarni et al. (2015) suggested that before calculating similarities, one should first *align* the models to fit them in one vector space, using linear transformations preserving general vector space structure. After that, cosine similarities across models become meaningful and can be used as indicators of semantic shifts. They also proposed constructing the time series of a word embedding over time, which allows for the detection of 'bursts' in its meaning with the *Mean Shift* model (Taylor, 2000). Notably, almost simultaneously the idea of aligning diachronic word embedding models using a distance-preserving projection technique was proposed by Zhang et al. (2015). Later, Zhang et al. (2016) expanded on this by adding the so called 'local anchors': that is, they used both linear projections for the whole models and small sets of nearest neighbors for mapping the query words to their correct temporal counterparts.

Instead of aligning their diachronic models using linear transformations, Eger and Mehler (2016) compared word meaning using so-called 'second-order embeddings,' that is, the vectors of words' similarities to all other words in the shared vocabulary of all models. This approach does not require any transformations: basically, one simply analyzes the word's position compared to other words. At the same time, Hamilton et al. (2016a) and Hamilton et al. (2016c) showed that these two approaches can be used simultaneously: they employed both 'second order embeddings' and orthogonal Procrustes transformations to align diachronic models.

Recently, it was shown in Bamler and Mandt (2017) ('*dynamic skip-gram*' model) and Yao et al. (2018) ('*dynamic Word2Vec*' model) that it is possible to learn the word embeddings across several time periods jointly, enforcing alignment across all of them simultaneously, and positioning all the models in the same vector space in one step. This develops the idea of model alignment even further and eliminates the need to first learn separate embeddings for each time period, and then align subsequent model pairs. Bamler and Mandt (2017) additionally describe two variations of their approach: a) for the cases when data slices arrive sequentially, as in streaming applications, where one can not use future observations, and b) for

[4]Levy and Goldberg (2014) showed that these two approaches are equivalent from the mathematical point of view.

the cases when data slices are available all at once, allowing for training on the whole sequence from the very beginning. A similar approach is taken by Rosenfeld and Erk (2018) who train a deep neural network on word and time representations. Word vectors in this setup turn into linear transformations applied to a continuous time variable, and thus producing an embedding of word w at time t.

Yet another way to make the models comparable is made possible by the fact that prediction-based word embedding approaches (as well as RI) allow for incremental updates of the models with new data without any modifications. This is not the case for the traditional explicit count-based algorithms, which usually require a computationally expensive dimensionality reduction step. Kim et al. (2014) proposed the idea of *incrementally updated diachronic embedding models*: that is, they train a model on the year y_i, and then the model for the year y_{i+1} is initialized with the word vectors from y_i. This can be considered as an alternative to model alignment: instead of aligning models trained from scratch on different time periods, one starts with training a model on the diachronically first period, and then updates this same model with the data from the successive time periods, saving its state each time. Thus, all the models are inherently related to each other, which, again, makes it possible to directly calculate cosine similarities between the same word in different time period models, or at least makes the models more comparable.

Several works have appeared recently which aim to address the technical issues accompanying this approach of incremental updating. Among others, Peng et al. (2017) described a novel method of incrementally learning the *hierarchical softmax* function for the CBOW and Continuous Skipgram algorithms. In this way, one can update word embedding models with new data and new vocabulary much more efficiently, achieving faster training than when doing it from scratch, while at the same time preserving comparable performance. Continuing this line of research, Kaji and Kobayashi (2017) proposed a conceptually similar incremental extension for *negative sampling*, which is a method of training examples selection, widely used with prediction-based models as a faster replacement for *hierarchical softmax*.

Even after the models for different time periods are made comparable in this or that way, one still has to choose the exact method of comparing word vectors across these models. Hamilton et al. (2016a) and Hamilton et al. (2016c) made an important observation that the distinction between linguistic and cultural semantic shifts is correlated with the distinction between *global* and *local* embedding comparison methods. The former take into account the whole model (for example, 'second-order embeddings,' when we compare the word's similarities to all other words in the lexicon), while the latter focus on the word's immediate neighborhood (for example, when comparing the lists of k nearest neighbors). They concluded that global measures are sensitive to regular processes of linguistic shifts, while local measures are better suited to detect slight cultural shifts in word meaning. Thus, the choice of particular embedding comparison approach should depend on what type of semantic shifts one seeks to detect.

4 Laws of semantic change

The use of diachronic word embeddings for studying the dynamics of word meaning has resulted in several hypothesized 'laws' of semantic change. We review some of these law-like generalizations below, before finally describing a study that questions their validity.

Dubossarsky et al. (2015) experimented with K-means clustering applied to SGNS embeddings trained for evenly sized yearly samples for the period 1850–2009. They found that the degree of semantic change for a given word – quantified as the change in self-similarity over time – negatively correlates with its distance to the centroid of its cluster. They proposed that the likelihood for semantic shift correlates with the degree of prototypicality (the *'law of prototypicality'* in Dubossarsky et al. (2017)).

Another relevant study is reported by Eger and Mehler (2016), based on two different graph models; one being a time-series model relating embeddings across time periods to model semantic shifts and the other modeling the self-similarity of words across time. Experiments were performed with time-indexed historical corpora of English, German and Latin, using time-periods corresponding to decades, years and centuries, respectively. To enable comparison of embeddings across time, second-order embeddings encoding similarities to other words were used, as described in 3.3, limited to the 'core vocabulary' (words occurring at least 100 times in all time periods). Based on linear relationships observed in the graphs, Eger and Mehler (2016) postulate two 'laws' of semantic change:

1. word vectors can be expressed as linear combinations of their neighbors in previous time periods;

2. the meaning of words tend to decay linearly in time, in terms of the similarity of a word to itself; this is in line with the '*law of differentiation*' proposed by Xu and Kemp (2015).

In another study, Hamilton et al. (2016a) considered historical corpora for English, German, French and Chinese, spanning 200 years and using time spans of decades. The goal was to investigate the role of frequency and polysemy with respect to semantic shifts. As in Eger and Mehler (2016), the rate of semantic change was quantified by self-similarity across time-points (with words represented by Procrustes-aligned SVD embeddings). Through a regression analysis, Hamilton et al. (2016a) investigated how the change rates correlate with frequency and polysemy, and proposed another two 'laws':

1. frequent words change more slowly ('*the law of conformity*');

2. polysemous words (controlled for frequency) change more quickly ('*the law of innovation*').

Azarbonyad et al. (2017) showed that these laws (at least the law of conformity) hold not only for diachronic corpora, but also for other 'viewpoints': for example, semantic shifts across models trained on texts produced by different political actors or written in different genres (Kutuzov et al., 2016). However, the temporal dimension allows for a view of the corpora under analysis as a sequence, making the notion of 'semantic shift' more meaningful.

Later, Dubossarsky et al. (2017) questioned the validity of some of these proposed 'laws' of semantic change. In a series of replication and control experiments, they demonstrated that some of the regularities observed in previous studies are largely artifacts of the models used and frequency effects. In particular, they considered 10-year bins comprising equally sized yearly samples from Google Books 5-grams of English fiction for the period 1990–1999. For control experiments, they constructed two additional data sets; one with chronologically shuffled data where each bin contains data from all decades evenly distributed, and one synchronous variant containing repeated random samples from the year 1999 alone. Any measured semantic shifts within these two alternative data sets would have to be due to random sampling noise.

Dubossarsky et al. (2017) performed experiments using raw co-occurrence counts, PPMI weighted counts, and SVD transformations (Procrustes aligned), and conclude that the 'laws' proposed in previous studies – that semantic change is correlated with frequency, polysemy (Hamilton et al., 2016a) and prototypicality (Dubossarsky et al., 2015) – are not valid as they are also observed in the control conditions. Dubossarsky et al. (2017) suggested that these spurious effects are instead due to the type of word representation used – count vectors – and that semantic shifts must be explained by a more diverse set of factors than distributional ones alone. Thus, the discussion on the existence of the 'laws of semantic change' manifested by distributional trends is still open.

5 Diachronic semantic relations

Word embedding models are known to successfully capture complex *relationships* between concepts, as manifested in the well-known word analogies task (Mikolov et al., 2013a), where a model must 'solve' equations of the form 'A is to B is as C is to what?' A famous example is the distributional model capturing the fact that the relation between '*man*' and '*woman*' is the same as between '*king*' and '*queen*' (by adding and subtracting the corresponding word vectors). Thus, it is a natural development to investigate whether changes in semantic relationships across time can also be traced by looking at the diachronic development of distributional models.

Zhang et al. (2015) considered the *temporal correspondences problem*, wherein the objective is to identify the word in a target time period which corresponds to a query term in the source time period (for example, given the query term *iPod* in the 2000s, the counterpart term in the 1980s time period is *Walkman*). This is proposed as a means to improve the results of information retrieval from document collections with significant time spans. Szymanski (2017) frames this as the *temporal word analogy* problem, extending the word analogies concept into the temporal dimension. This work shows that

diachronic word embeddings can successfully model relations like 'word w_1 at time period t_α is like word w_2 at time period t_β'. To this end, embedding models trained on different time periods are aligned using linear transformations. Then, the temporal analogies are solved by simply finding out which word vector in the time period t_β is the closest to the vector of w_1 in the time period t_α.

A variation of this task was studied in Rosin et al. (2017), where the authors learn the relatedness of words over time, answering queries like 'in which time period were the words *Obama* and *president* maximally related'. This technique can be used for a more efficient user query expansion in general-purpose search engines. Kutuzov et al. (2017a) modeled a different semantic relation: 'words w_1 and w_2 at time period t_α are in the same semantic relation as words w_3 and w_4 at time period t_β'. To trace the temporal dynamics of these relations, they re-applied linear projections learned on sets of w_1 and w_2 pairs from the model for the period t_n to the model trained on the subsequent time period t_{n+1}. This was used to solve the task of detecting lasting or emerging armed conflicts and the violent groups involved in these conflicts.

6 Applications

Applications of diachronic word embeddings approaches can generally be grouped into two broad categories: *linguistic studies* which investigate the how and why of semantic shifts, and *event detection* approaches which mine text data for actionable purposes.

The first category generally involves corpora with longer time spans, since linguistic changes happen at a relatively slow pace. Some examples falling into this category include tracking semantic drift of particular words (Kulkarni et al., 2015) or of word sentiment (Hamilton et al., 2016b), identifying the breakpoints between epochs (Sagi et al., 2011; Mihalcea and Nastase, 2012), studying the laws of semantic change at scale (Hamilton et al., 2016c) and finding different words with similar meanings at different points in time (Szymanski, 2017). This has been held up as a good use case of deep learning for research in computational linguistics (Manning, 2015), and there are opportunities for future work applying diachronic word embeddings not only in the field of historical linguistics, but also in related areas like sociolinguistics and digital humanities.

The second category involves mining texts for cultural semantic shifts (usually on shorter time spans) indicating real-world events. Examples of this category are temporal information retrieval (Rosin et al., 2017), predicting civil turmoils (Kutuzov et al., 2017b; Mueller and Rauh, 2017), or tracing the popularity of entities using norms of word vectors (Yao et al., 2018). They can potentially be employed to improve user experience in production systems or for policy-making in governmental structures.

We believe that the near future will see a more diverse landscape of applications for diachronic word embeddings, especially related to the real-time analysis of large-scale news streams. 'Between the lines,' these data sources contain a tremendous amount of information about processes in our world, manifested in semantic shifts of various sorts. The task of researchers is to reveal this information and make it reliable and practically useful.

7 Open challenges

The study of temporal aspects of semantic shifts using distributional models (including word embeddings) is far from being a solved problem. The field still has a considerable number of open challenges. Below we briefly describe the most demanding ones.

- The existing methods should be expanded to a *wider scope of languages*. Hamilton et al. (2016a), Kutuzov and Kuzmenko (2018) and others have started to analyze other languages, but the overwhelming majority of publications still apply only to English corpora. It might be the case that the best methodologies are the same for different languages, but this should be shown empirically.

- There is a clear need to devise algorithms that work on *small datasets*, as they are very common in historical linguistics, digital humanities, and similar disciplines.

- Carefully designed and robust *gold standard test sets* of semantic shifts (of different kinds) should be created. This is a difficult task in itself, but the experience from synchronic word embeddings evaluation (Hill et al., 2015) and other NLP areas proves that it is possible.

- There is a need for rigorous *formal mathematical models of diachronic embeddings*. Arguably, this will follow the vein of research in joint learning across several time spans, started by Bamler and Mandt (2017) and Yao et al. (2018), but other directions are also open.

- Most current studies stop after stating the simple fact that a semantic shift has occurred. However, more detailed analysis of the nature of the shift is needed. This includes:

    1. *Sub-classification of types of semantic shifts* (broadening, narrowing, etc). This problem was to some degree addressed by Mitra et al. (2014), but much more work is certainly required to empirically test classification schemes proposed in much of the theoretical work described in Section 2.
    2. *Identifying the source of a shift* (for example, linguistic or extra-linguistic causes). This causation detection is closely linked to the division between linguistic drifts and cultural shifts, as proposed in Hamilton et al. (2016c).
    3. *Quantifying the weight of senses* acquired over time. Many words are polysemous, and the relative importance of senses is flexible (Frermann and Lapata, 2016). The issue of handling senses is central for detecting semantic shifts, but most of the algorithms described in this survey are not sense-aware. To address this, methods from sense embeddings research (Bartunov et al., 2016) might be employed.
    4. *Identifying groups of words that shift together* in correlated ways. Some work in this direction was started in Dubossarsky et al. (2016), who showed that verbs change more than nouns, and nouns change more than adjectives. This is also naturally related to proving the (non-)existence of the 'laws of semantic change' (see Section 4).

- Last but not least, we believe that the community around diachronic word embeddings research severely lacks relevant forums, like *topical workshops* or *shared tasks*. Diachronic text evaluation tasks like the one at *SemEval-2015* (Popescu and Strapparava, 2015) are important but not enough, since they focus on identifying the time period when a text was authored, not the process of shifting meanings of a word. Organizing such events can promote the field and help address many of the challenges described above.

8 Summary

We have presented an outline of the current research related to computational detection of semantic shifts using diachronic (temporal) word embeddings. We covered the linguistic nature of semantic shifts, the typical sources of diachronic data and the distributional approaches used to model it, from frequentist methods to contemporary prediction-based models. To sum up, Figure 1 shows the timeline of events that have been influential in the development of research in this area: introducing concepts, usage of corpora and important findings.

This emerging field is still relatively new, and although recent years has seen a string of significant discoveries and academic interchange, much of the research still appears slightly fragmented, not least due to the lack of dedicated venues like workshops, special issues, or shared tasks. We hope that this survey will be useful to those who want to understand how this field has developed, and gain an overview of what defines the current state-of-the-art and what challenges lie ahead.

Acknowledgements

We thank William Hamilton, Haim Dubossarsky and Chris Biemann for their helpful feedback during the preparation of this survey. All possible mistakes remain the sole responsibility of the authors.

References

Hosein Azarbonyad, Mostafa Dehghani, Kaspar Beelen, Alexandra Arkut, Maarten Marx, and Jaap Kamps. 2017. Words are malleable: Computing semantic shifts in political and media discourse. In *Proceedings of the ACM on Conference on Information and Knowledge Management*, pages 1509–1518, Singapore.

Robert Bamler and Stephan Mandt. 2017. Dynamic word embeddings. In *Proceedings of the International Conference on Machine Learning*, pages 380–389, Sydney, Australia.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 238–247, Baltimore, USA.

Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, and Dmitry Vetrov. 2016. Breaking sticks and ambiguities with adaptive skip-gram. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 130–138, Cadiz, Spain.

Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. 2014. Analysing word meaning over time by exploiting temporal random indexing. In *Proceedings of the First Italian Conference on Computational Linguistics*, pages 38–42, Turin, Italy.

Andreas Blank and Peter Koch. 1999. *Historical semantics and cognition*. Walter de Gruyter.

David M Blei and John D Lafferty. 2006. Dynamic topic models. In *Proceedings of the 23rd International Conference on Machine learning*, pages 113–120, Pittsburgh, USA.

Leonard Bloomfield. 1933. *Language*. Allen & Unwin.

Michel Bréal. 1899. *Essai de sémantique*. Hachette, Paris.

John A Bullinaria and Joseph P Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods*, 39(3):510–526.

Hyunyoung Choi and Hal Varian. 2012. Predicting the present with Google trends. *Economic Record*, 88(s1):2–9.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Haim Dubossarsky, Yulia Tsvetkov, Chris Dyer, and Eitan Grossman. 2015. A bottom up approach to category mapping and meaning change. In *Proceedings of the NetWordS 2015 Word Knowledge and Word Usage*, pages 66–70, Pisa, Italy.

Haim Dubossarsky, Daphna Weinshall, and Eitan Grossman. 2016. Verbs change more than nouns: a bottom-up computational approach to semantic change. *Lingue e linguaggio*, 15(1):7–28.

Haim Dubossarsky, Daphna Weinshall, and Eitan Grossman. 2017. Outta control: Laws of semantic change and inherent biases in word representation models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1147–1156, Copenhagen, Denmark.

Steffen Eger and Alexander Mehler. 2016. On the linearity of semantic change: Investigating meaning variation via dynamic graph models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 52–58, Berlin, Germany.

John Firth. 1957. *A synopsis of linguistic theory, 1930-1955*. Blackwell.

Lea Frermann and Mirella Lapata. 2016. A bayesian model of diachronic meaning change. *Transactions of the Association of Computational Linguistics*, 4:31–45.

Dirk Geeraerts. 1997. *Diachronic prototype semantics: A contribution to historical lexicology*. Clarendon Press, Oxford.

Stefan Th. Gries. 1999. Particle movement: a cognitive and functional approach. *Cognitive Linguistics*, 10:105–145.

Joachim Grzega and Marion Schoener. 2007. English and general historical lexicology. *Eichstätt-Ingolstadt: Katholische Universität*.

Kristina Gulordava and Marco Baroni. 2011. A distributional similarity approach to the detection of semantic change in the Google Books Ngram corpus. In *Proceedings of the GEMS 2011 Workshop on Geometrical Models of Natural Language Semantics*, pages 67–71, Edinburgh, UK.

L. William Hamilton, Jure Leskovec, and Dan Jurafsky. 2016a. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1489–1501, Berlin, Germany.

William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016b. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 595–605, Austin, Texas.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016c. Cultural shift or linguistic drift? Comparing two computational measures of semantic change. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2116–2121, Austin, Texas.

Gerhard Heyer, Florian Holz, and Sven Teresniak. 2009. Change of topics over time – tracking topics by their change of meaning. In *Proceeding of the International Conference on Knowledge Discovery and Information Retrieval*, pages 223–228, Madeira, Portugal.

Gerhard Heyer, Cathleen Kantner, Andreas Niekler, Max Overbeck, and Gregor Wiedemann. 2016. Modeling the dynamics of domain specific terminology in diachronic corpora. In *Proceedings of the 12th International conference on Terminology and Knowledge Engineering (TKE 2016)*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Martin Hilpert and Stefan Th. Gries. 2009. Assessing frequency changes in multistage diachronic corpora: Applications for historical corpus linguistics and the study of language acquisition. *Literary and Linguistic Computing*, 24(4):385–401.

M. Hilpert. 2008. *Germanic future constructions: A usage-based approach to language change*. Benjamins, Amsterdam, Netherlands.

Patrick Juola. 2003. The time course of language change. *Computers and the Humanities*, 37(1):77–96.

David Jurgens and Keith Stevens. 2009. Event detection in blogs using Temporal Random Indexing.

Nobuhiro Kaji and Hayato Kobayashi. 2017. Incremental skip-gram model with negative sampling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 363–371, Copenhagen, Denmark.

Pentti Kanerva, Jan Kristofersson, and Anders Holst. 2000. Random indexing of text samples for latent semantic analysis. In *Proceedings of the 22nd annual conference of the cognitive science society*, volume 1036, pages 103–106, Mahwah, USA.

D. Kerremans, S. Stegmayr, and H.-J. Schmid. 2010. The neocrawler: Identifying and retrieving neologisms from the internet and monitoring ongoing change. In K. Allan and J. A. Robinson, editors, *Current methods in historical semantics*, pages 130–160. De Gruyter Mouton.

Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal analysis of language through neural language models. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 61–65, Baltimore, USA.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, pages 625–635, Florence, Italy.

Andrey Kutuzov and Elizaveta Kuzmenko. 2018. Two centuries in two thousand words: Neural embedding models in detecting diachronic lexical changes. *Quantitative Approaches to the Russian Language*, pages 95–112.

Andrey Kutuzov, Elizaveta Kuzmenko, and Anna Marakasova. 2016. Exploration of register-dependent lexical semantics using word embeddings. In *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*, pages 26–34, Osaka, Japan.

Andrey Kutuzov, Erik Velldal, and Lilja Øvrelid. 2017a. Temporal dynamics of semantic relations in word embeddings: an application to predicting armed conflict participants. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1824–1829, Copenhagen, Denmark.

Andrey Kutuzov, Erik Velldal, and Lilja Øvrelid. 2017b. Tracing armed conflicts with diachronic word embedding models. In *Proceedings of the Events and Stories in the News Workshop at ACL 2017*, pages 31–36, Vancouver, Canada.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, pages 2177–2185, Montreal, Canada.

Xuanyi Liao and Guang Cheng. 2016. Analysing the semantic change based on word embedding. In *Natural Language Understanding and Intelligent Applications*, pages 213–223. Springer International Publishing.

Jefrey Lijffijt, Tanja Säily, and Terttu Nevalainen. 2012. CEECing the baseline: Lexical stability and significant change in a historical corpus. In *Studies in Variation, Contacts and Change in English*, volume 10. Research Unit for Variation, Contacts and Change in English (VARIENG).

Christopher D. Manning. 2015. Computational linguistics and deep learning. *Computational Linguistics*, 41(4):701–707.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182.

Rada Mihalcea and Vivi Nastase. 2012. Word epoch disambiguation: Finding how words change over time. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 259–263, Jeju Island, Korea.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26:3111–3119.

Sunny Mitra, Ritwik Mitra, Martin Riedl, Chris Biemann, Animesh Mukherjee, and Pawan Goyal. 2014. That's sick dude!: Automatic identification of word sense change across different timescales. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1020–1029, Baltimore, Maryland.

Hannes Mueller and Christofer Rauh. 2017. Reading between the lines: Prediction of political violence using newspaper text. *American Political Science Review*, page 1–18.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword Fifth Edition LDC2011T07. Technical report, Technical Report. Linguistic Data Consortium, Philadelphia.

Hao Peng, Jianxin Li, Yangqiu Song, and Yaopeng Liu. 2017. Incrementally learning the hierarchical softmax function for neural language models. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 3267–327, San Francisco, California USA.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar.

Octavian Popescu and Carlo Strapparava. 2015. SemEval 2015, task 7: Diachronic text evaluation. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 870–878, Denver, Colorado.

Matti Rissanen et al. 1993. The helsinki corpus of english texts. *Kyttö et. al*, pages 73–81.

Alex Rosenfeld and Katrin Erk. 2018. Deep neural models of semantic shift. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 474–484, New Orleans, Louisiana, USA.

Guy D. Rosin, Eytan Adar, and Kira Radinsky. 2017. Learning word relatedness over time. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1179–1189, Copenhagen, Denmark.

Eyal Sagi, Stefan Kaufmann, and Brady Clark. 2011. Tracing semantic change with latent semantic analysis. *Current methods in historical semantics*, pages 161–183.

Evan Sandhaus. 2008. The New York Times annotated corpus overview. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Gustaf Stern. 1931. *Meaning and change of meaning; with special reference to the English language.* Wettergren & Kerbers.

Terrence Szymanski. 2017. Temporal word analogies: Identifying lexical replacement with diachronic word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 448–453, Vancouver, Canada.

Wayne A Taylor. 2000. Change-point analysis: a powerful new tool for detecting changes.

Elizabeth Closs Traugott and Richard B Dasher. 2001. *Regularity in semantic change*. Cambridge University Press.

Elizabeth Traugott. 2017. Semantic change. *Oxford Research Encyclopedias: Linguistics*.

Peter Turney, Patrick Pantel, et al. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: a non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–433, Philadelphia, PA, USA.

Derry Tanti Wijaya and Reyyan Yeniterzi. 2011. Understanding semantic change of words over centuries. In *Proceedings of the 2011 international workshop on Detecting and Exploiting Cultural diversity on the social web*, pages 35–40.

Yang Xu and Charles Kemp. 2015. A computational evaluation of two laws of semantic change. In *Proceedings of the 37th Annual Meeting of the Cognitive Science Society*, Austin, TX, USA.

Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong. 2018. Dynamic word embeddings for evolving semantic discovery. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 673–681, Marina Del Rey, CA, USA.

Yating Zhang, Adam Jatowt, Sourav Bhowmick, and Katsumi Tanaka. 2015. Omnia mutantur, nihil interit: Connecting past with present by finding corresponding terms across time. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 645–655, Beijing, China.

Yating Zhang, Adam Jatowt, Sourav S. Bhowmick, and Katsumi Tanaka. 2016. The past is not a foreign country: Detecting semantically similar terms across time. *IEEE Transactions on Knowledge and Data Engineering*, 28(10):2793–2807, October.

Interaction-Aware Topic Model for Microblog Conversations through Network Embedding and User Attention

Ruifang He[1,2,*], **Xuefei Zhang**[1,2,*], **Di Jin**[1,2,†], **Longbiao Wang**[1,2], **Jianwu Dang**[1,2,3], **Xiangang Li**[4]
[1]School of Computer Science and Technology, Tianjin University, Tianjin, China
[2]Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin, China
[3]Japan Advanced Institute of Science and Technology, Ishikawa, Japan
[4]AI Labs, Didi Chuxing, Beijing, China
{rfhe,jindi,longbiao_wang}@tju.edu.cn,zhangxuefei29@163.com
jdang@jaist.ac.jp,lixiangang@didichuxing.com

Abstract

Traditional topic models are insufficient for topic extraction in social media. The existing methods only consider text information or simultaneously model the posts and the static characteristics of social media. They ignore that one discusses diverse topics when dynamically interacting with different people. Moreover, people who talk about the same topic have different effects on the topic. In this paper, we propose an Interaction-Aware Topic Model (IATM) for microblog conversations by integrating network embedding and user attention. A conversation network linking users based on reposting and replying relationship is constructed to mine the dynamic user behaviours. We model dynamic interactions and user attention so as to learn interaction-aware edge embeddings with social context. Then they are incorporated into neural variational inference for generating the more consistent topics. The experiments on three real-world datasets show that our proposed model is effective.

1 Introduction

The prosperity of microblog platforms, such as Twitter[1] and Sina Weibo[2] brings the large scale, noisy and user-generated short posts. Automatically extracting topics in social media aims to reveal thematic information of the underlying collection, which can be used in summarization (Zhuang et al., 2016), hashtag recommendation (Li et al., 2016b), response generation (Xing et al., 2017) and so on.

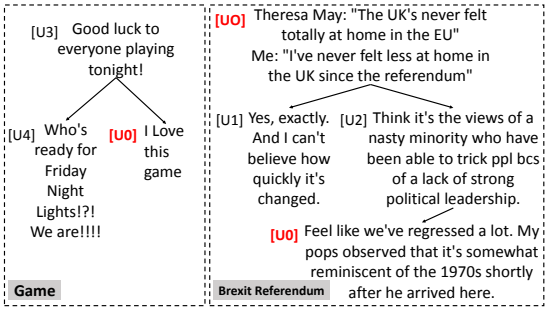
The conventional topic models, like Latent Dirichlet Allocation (LDA) (Blei et al., 2003), infer the hidden topics based on word co-occurrences in documents. They could not be directly transferred in social media due to the data sparsity and the noise of short texts.

The existing relevant researches can be roughly categorized into: (1) Methods exploit the content of short texts by aggregation strategy (Mehrotra et al., 2013) or modeling word-pair co-occurrence (Yan et al., 2013). (2) Models incorporating with word embedding try to deeply understand the posts by representation learning (Sridhar, 2015; Hu and Tsujii, 2016), yet they ignore the social context of microblog messages. Essentially, social media content and network structures influence each other (Bi et al., 2014), the only content analysis is insufficient. (3) (Li et al., 2016a; Chen and Ren, 2017) take into account the content and static characteristics of network structures to deduce topics. However, they ignore dynamic user behaviours.

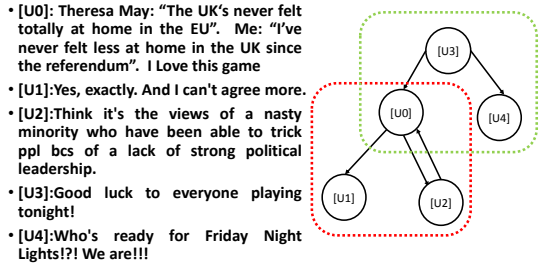
Actually, a user may talk about various topics when interacting with diverse neighbours in a social network. For instance, Fig. 1(a) shows two conversation trees, where [U0] communicates with [U1] and [U2] about the topic of United Kingdom European Union membership referendum, while [U0] talks a game with [U3]. Hence, we need to infer topics according to the interactions between users. Moreover, although [U0] and [U1] argue about the same topic, [U0] contains salient content in topic description, e.g., *UK, EU, referendum* while the reply of [U1] is nothing but simply response to [U0], e.g., *Yes, agree*.

* Indicates equal contribution.
†Corresponding author.
[1]https://twitter.com
[2]https://weibo.com

Proceedings of the 27th International Conference on Computational Linguistics, pages 1398–1409
Santa Fe, New Mexico, USA, August 20-26, 2018.



(a) Two conversation trees. [Ui]: the i-th user; Arrow lines: reposting/replying relations.



(b) An example of a conversation network. The green box shows the topic about a game; The red one exhibits the topic of Brexit referendum.

Figure 1: A conversation network is transformed from two conversation trees.

Thus, it is intuitive that during the topic extraction of United Kingdom European Union membership referendum, [U0] is more important than [U1]. The **dynamic interactions** and **diverse user attention** on the topic provide us heuristic insights for topic extraction.

Note that conversation trees can be transformed into conversation networks through user relations, as illustrated in the Fig. 1. Inspired by network embedding and social user behaviour researches, we propose an **I**nteraction **A**ware **T**opic **M**odel (IATM) integrating dynamic interactions and various effects of users on the topic. The contributions are as follows.

- We propose a novel Interaction-Aware Topic Model (IATM) for microblog conversations, which simultaneously considers social media content and network topology.
- Our method models dynamic interactions and various effects of users, and encodes them as interaction-aware edge embeddings through network embedding and user attention.
- The deeper edge representations are employed in neural variational inference for generating the more consistent topics.
- The experiments verify that the proposed IATM is effective. Dynamic user behaviours are useful for topic extraction of short texts in social media.

2 Related Work

Previous researches for topic extraction in social media can be mainly classified into two aspects.

2.1 Focusing on Content Information

They depend on pure content to generate document-topic distribution and topic-word distribution. (1) **Aggregation Strategy Based.** To solve the data sparsity short texts, some use different strategies to heuristically aggregate microblog messages based on authorship (Hong and Davison, 2010; Zhao et al., 2011) or hashtags (Mehrotra et al., 2013; Tang et al., 2013) before applying traditional topic model. Self-Aggregation based Topic Model (SATM) (Quan et al., 2015) combines short texts aggregation and topic induction into a unified model. Others use Biterm Topic Model (BTM) (Yan et al., 2013) and RNN-IDF based Biterm Short-text Topic Model (RIBSTM) (Lu et al., 2017) modeling biterm co-occurrence in the whole corpus to enhance topic discovery. (2) **Word Embedding Based.** Since word embeddings have been shown their ability to form clusters of conceptually similar words in the embedding space (Mikolov et al., 2013c), Gaussian Mixture Topic Model (GMTM) (Sridhar, 2015) and Latent Concept Topic Model (LCTM) (Hu and Tsujii, 2016) utilize word embeddings to improve topic generation. However, since social media content and network structures influence with each other, only focusing on content is insufficient.

2.2 Integrating Content and Network Structure information

This kind of researches considers social content and network structures together, including followship and repostship. (1) **Followship Based**. Lim et al.(2013) jointly model the text and user-follower network during topic inference. Yet the appearance of *Zombie Fans* purchased by those who would like to increase their influence, disturbs the followship network. (2) **Repostship Based**. LeadLDA (Li et al., 2016a) generates words according to topic dependencies derived from conversation trees by differentiating messages from leader and follower. Leader is the post that contains salient and new information in topic description while follower is the post that simply responds to its reposted or replied messages. Chen and Ren (2017) propose ForumLDA to infer topics by distinguishing response messages from relevant and irrelevant to the original post.

However, due to the zombie fans in followship network, or exploring the static characteristics in repostship network, these methods ignore the dynamic user behaviours latent in reciprocal interactions between users. Therefore we emphasize on exploring whether user interactions and user attention can help topic generation in social media.

2.3 Network Embedding and Neural Variational Inference

The further researches (Perozzi et al., 2014; Tu et al., 2017) about network embedding make it possible to suitably model the dynamic user behaviours. Meanwhile, comparing with topic models based on word embedding, Miao et al. (2017) and Srivastava and Sutton (2017) apply topic model with neural variational inference (Kingma and Welling, 2014) on traditional long documents, which consistently produces the better topics. A small change in the document will produce a small change in topics. Yet they have not taken into account sparse and noisy texts and network structures in social media.

Therefore, we model dynamic user behaviours and content information as interaction-aware edge embeddings, which combine network embedding and user attention. Further, they are incorporated into variational auto-encoder for topic extraction in social media, which can produce the more consistent and precise topics.

3 Interaction-Aware Topic Model

The large scale short and noisy texts in social media bring the more serious data sparseness and inconsistency in topics. Yet the dynamic user behaviours also provide us opportunities. Here, we propose two hypotheses and explore whether they are useful for topic extraction.

H1: Dynamic Interactions. One discusses various topics with different people.

H2: User Attention. Users who debate the same topic have diverse effects on the topic.

We need to construct a conversation network, shown in Fig. 1(b), transformed from conversation trees.

3.1 Conversation Network

Conversation networks are transformed from conversation trees through user relations, which are exhibited in Fig. 1(b). We first give the basic notations and definitions of a conversation network. Given $G = (V, E, T)$, where V is the set of vertices, $E \subseteq V \times V$ is the edges between vertices and T denotes text information of vertices. Each vertex $v \in V$ represents a user and every edge $e_{(u,v)} \in E$ stands for the reposting or replying relationship between two vertices (u, v). To solve the sparsity of short texts, we aggregate all the messages posted by the same user, including the original and the reposting messages[3]. Text information of the unique vertex $v \in V$ is the messages $M_v = (w_1, w_2, ..., w_n)$, where n is the count of words in M_v.

In order to illustrate the proposed IATM model, we firstly give two definitions:

D1: Edge Embedding. The edge embedding $\mathbf{e}$ for every edge $e_{(u,v)}$ is acquired by concatenating the vertex embeddings $\mathbf{u}$ and $\mathbf{v}$. It encodes user representations with respect to various neighbours.

D2: Interaction-aware Edge Embedding. It models the various effects of u and v on a topic via user attention mechanism, and makes edge embedding as an interaction-aware style.

[3]Both of replying and reposting messages are called reposting messages.

3.2 Model

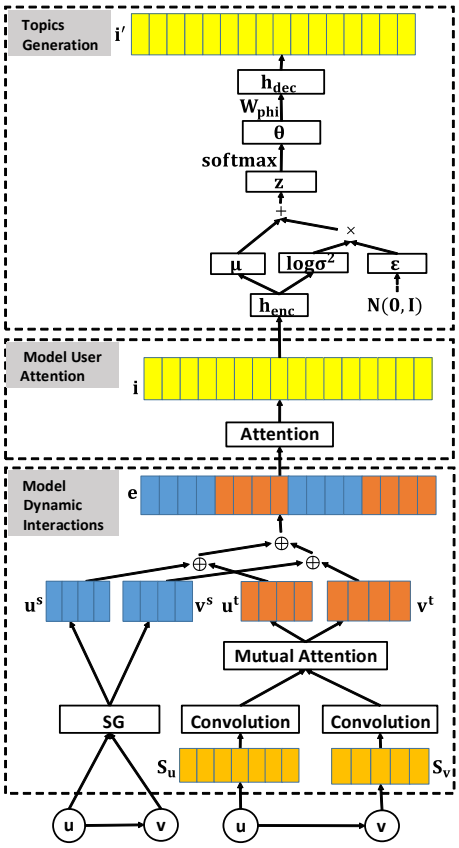


Figure 2: The IATM framework.

The proposed IATM framework is shown in Fig. 2. The generative process of IATM mainly includes three parts: (1) model dynamic interactions, (2) model user attention and (3) topics generation. A basis of three parts is that each word is represented as a low dimensional, continuous and real-valued vector, also known as word embedding (Mikolov et al., 2013c). Given text information of a vertex v, we take the embedding $\mathbf{w}_i \in \mathbb{R}^{d'}$ for each word to obtain embedding sequence as $\mathbf{S}_v = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_n)$. Here, d' indicates the dimension of word embeddings.

3.2.1 Model Dynamic Interactions

Corresponding to the assumption H1, a specific vertex should have its own diverse points with distinct neighbours, which leads to different edge embeddings.

To make full use of network structures and associated text information, we encode each vertex v as a concatenation of a structure-based embedding $\mathbf{v}^{(s)}$ and a text-based embedding $\mathbf{v}^{(t)}$, and get the vertex embedding $\mathbf{v} = \mathbf{v}^{(s)} \oplus \mathbf{v}^{(t)} (\mathbf{v} \in \mathbb{R}^d)$.

Structure-based Embedding. We adopt a neural language model (SkipGram, or SG for short) (Mikolov et al., 2013a) for $\mathbf{v}^{(s)}$. To maximize the probability of a node's neighbourhood co-occurrence, we define the objective function of structure-based embeddings as follows

$$L_s = -\log \sum_{-k \le j \le k} p(\mathbf{v}_{i+j}^{(s)} | \mathbf{v}_i^{(s)}) \tag{1}$$

where $\mathbf{v}_i^{(s)}$ is corresponding to the i-th vertex, the window size is k, and $p(\mathbf{v}_{i+j}^{(s)} | \mathbf{v}_i^{(s)})$ is defined using

the *softmax* fuction

$$p(\mathbf{v}_{i+j}^{(s)}|\mathbf{v}_i^{(s)}) = \frac{\exp((\mathbf{v}_{i+j}^{(s)})^T \mathbf{v}_i^{(s)})}{\sum_{t=1}^{|V|} \exp((\mathbf{v}_t^{(s)})^T \mathbf{v}_i^{(s)})}. \tag{2}$$

In this way, nodes with similar neighbours share the similar structure-based embeddings.

Text-based Embedding. To discover the thematic information of the vertex pair in an edge, we utilize mutual attention (Santos et al., 2016; Tu et al., 2017) to obtain text-based embeddings, which allows the pooling operation to be aware of the topic of an edge $e_{(u,v)}$. To some extent, content information from a vertex can directly affect the text-based embedding of the other vertex, and vice versa.

Convolutional neural network (CNN) has gained great performance on the textual information encoding (Chen et al., 2015; Wang et al., 2017). Given the embedding sequence $\mathbf{S}_v$, we conduct convolution operation over $\mathbf{S}_v$ within the i-th window as follows

$$\mathbf{x}_i = \mathbf{C} \cdot (\mathbf{S}_v)_{i:i+l-1} + \mathbf{b} \tag{3}$$

where $\mathbf{C} \in \mathbb{R}^{d \times (l \times d')}$ is a convolution matrix, $\mathbf{b}$ is the bias vector and window size is l. The same operation is also on $\mathbf{S}_u$. The outputs of convolution, $\mathbf{P} \in \mathbb{R}^{d \times m}$ and $\mathbf{Q} \in \mathbb{R}^{d \times n}$ where m and n mean the length of $\mathbf{S}_u$ and $\mathbf{S}_v$ respectively, are as the input of mutual attention layer to compute the correlation matrix $\mathbf{F}$.

$$\mathbf{F} = relu(\mathbf{P^T A Q})(\mathbf{F} \in \mathbb{R}^{m \times n}) \tag{4}$$

$\mathbf{A} \in \mathbb{R}^{d \times d}$ is a matrix to be learned by the neural network and we emlpoy $relu$ as activation function. Note that, the element $\mathbf{F}_{i,j} \in \mathbf{F}$ is the pair-wise correlation score of the hidden vectors $\mathbf{P}_i$ and $\mathbf{Q}_j$.

Then, we employ the mean-pooling along rows and columns of $\mathbf{F}$ to generate pooling vectors by

$$g_i^{(p)} = \mathbf{mean}(\mathbf{F}_{i,1}, ..., \mathbf{F}_{i,n}) \quad g_j^{(q)} = \mathbf{mean}(\mathbf{F}_{1,j}, ..., \mathbf{F}_{m,j}). \tag{5}$$

The pooling vectors of $\mathbf{P}$ and $\mathbf{Q}$ are obtained as

$$\mathbf{g}^{(p)} = (g_1^{(p)}, ..., g_m^{(p)})^T \quad \mathbf{g}^{(q)} = (g_1^{(q)}, ..., g_n^{(q)})^T. \tag{6}$$

Next, the $softmax$ function is operated on $\mathbf{g}^{(p)}$ and $\mathbf{g}^{(q)}$ to get the mutual attentive vectors $\mathbf{a}^{(p)}$ and $\mathbf{a}^{(q)}$. For instance, the i-th element of mutual attentive vector $\mathbf{a}^{(p)}$ is computed as

$$a_i^{(p)} = \frac{\exp(g_i^{(p)})}{\sum_{t=1}^{m} \exp(g_t^{(p)})}. \tag{7}$$

Finally, we get the text-based embeddings $\mathbf{u}^{(t)}$ and $\mathbf{v}^{(t)}$ by $\mathbf{u}^{(t)} = \mathbf{P}\mathbf{a}^{(p)}$ and $\mathbf{v}^{(t)} = \mathbf{Q}\mathbf{a}^{(q)}$. The objective function of text-based embeddings is as

$$L_t(e) = \alpha \log p(\mathbf{v}^{(t)}|\mathbf{u}^{(t)}) + \beta \log p(\mathbf{v}^{(t)}|\mathbf{u}^{(s)}) + \gamma \log p(\mathbf{v}^{(s)}|\mathbf{u}^{(t)}) \tag{8}$$

where α, β and γ control the weights of corresponding parts. Similarly, we employ $softmax$ function for calculating the conditional probabilities in Eq. (8) as in Eq. (2).

Edge Embedding. Here, we obtain the vertex embeddings $\mathbf{u}$ and $\mathbf{v}$ by $\mathbf{u} = \mathbf{u}^{(s)} \oplus \mathbf{u}^{(t)}$ and $\mathbf{v} = \mathbf{v}^{(s)} \oplus \mathbf{v}^{(t)}$. In this way, we operate $\mathbf{e} = \mathbf{u} \oplus \mathbf{v}$ to get the edge embedding $\mathbf{e} \in \mathbb{R}^{2d}$ for $e_{(u,v)}$, which is dynamic context-aware.

3.2.2 Model User Attention

As referred before, we assume that users who discuss the identical topic may take different effects on the topic. Attention mechanism is designed for mining the different significance of users on the topic. Since edge embedding is incorporated by the vertex-pair embeddings in an edge and each vertex denotes a user, the attention of users on the topic are transformed into the user attention vector $\mathbf{a}^{(e)} \in \mathbb{R}^{2d}$. $\mathbf{a}^{(e)}$

is computed by conducting $softmax$ function on the edge embedding $\mathbf{e}$. To formulate, the i-th element of $\mathbf{a}^{(e)}$ is computed as

$$a_i^{(e)} = \frac{\exp(\mathbf{e}_i)}{\sum_{t=1}^{2d} \exp(\mathbf{e}_t)}. \tag{9}$$

After that, through combining topic information of the vertex-pair in an edge with user attention on the topic, we obtain interaction-aware edge embedding $\mathbf{i} \in \mathbb{R}^{2d}$ by using element-wise product as $\mathbf{i} = (\mathbf{e}_1\mathbf{a}_1^{(e)}, ..., \mathbf{e}_{2d}\mathbf{a}_{2d}^{(e)})$.

3.2.3 Topics Generation

Neural variational inference (Miao et al., 2017) approximates the posterior of a generative model with a variational distribution parameterized by a neural network, which consistently generates the better topics. With respect to social media, we input the interaction-aware edge embedding, which is encoded with dynamic user behaviours and content information, into variational auto-encoder. To formulate, suppose $\hat{d}$ is a document, w is a word in $\hat{d}$ and the number of topics is K. Here, we adopt neural variational inference to infer the multinomial document-topic distribution $\theta_{\hat{d}} = (p(t_1|\hat{d}), ..., p(t_K|\hat{d}))$ and topic-word distribution $\phi_w = (p(w|t_1), ..., p(w|t_K))$, where t_i is the i-th topic.

Document-topic distribution. Precisely, given the interaction-aware edge embedding $\mathbf{i}$, we first encode it to a hidden space as

$$\mathbf{h}_{enc} = relu(\mathbf{W}^{(ih)}\mathbf{i} + \mathbf{b}^{(ih)}) \tag{10}$$

where $\mathbf{W}^{(ih)}$ and $\mathbf{b}^{(ih)}$ are the parameters of the neural network. Then the Gaussian parameters $\mu_{\hat{d}}$ and $\sigma_{\hat{d}}^2$ can be obtained as

$$\mu_{\hat{d}} = \mathbf{W}^{(h\mu)}\mathbf{h}_{enc} + \mathbf{b}^{(h\mu)} \quad \log(\sigma_{\hat{d}}^2) = \mathbf{W}^{(h\sigma)}\mathbf{h}_{enc} + \mathbf{b}^{(h\sigma)}. \tag{11}$$

The latent semantic vector $\mathbf{z} \in \mathbb{R}^K$ can be calculated using the reparameterization trick as $\mathbf{z} = \mu_{\hat{d}} + \epsilon \times \sigma_{\hat{d}}$ where $\epsilon \sim \mathcal{N}(\mu_0, \sigma_0^2)$ is the prior Gaussian distribution. The hyper-parameters μ_0 and σ_0^2 is set to a zero mean and unit variance Gaussian. Here we pass the Gaussian random vector $\mathbf{z}$ through the $softmax$ function to parameterize the multinomial document-topic distribution $\theta_{\hat{d}}$.

Topic-word distribution. The conventional topic models compute the conditional probability $p(w|\hat{d})$ as

$$p(w|\hat{d}) = \phi_w \times (\theta_{\hat{d}})^T. \tag{12}$$

Therefore, we compute the topic-word distribution ϕ_w as the parameter of neural network by

$$\mathbf{h}_{dec} = softmax(\phi_w \times (\theta_{\hat{d}})^T). \tag{13}$$

Thereafter, a new interaction-aware edge embedding $\mathbf{i}'$ is generated as

$$\mathbf{i}' = relu(\mathbf{W}^{(hi)}\mathbf{h}_{dec} + \mathbf{b}^{(hi)}) \tag{14}$$

where $\mathbf{W}^{(hi)}$ and $\mathbf{b}^{(hi)}$ are the parameters of neural network.

The objective for this part is as

$$L_{\theta,\phi}(e) = \mathbb{E}_{q(\theta,\mathbf{z}|\mathbf{i})}[\log p(\mathbf{i}|\mathbf{z},\theta,\phi)] - D_{KL}[q(\theta,\mathbf{z}|\mathbf{i})||p(\theta|\mu_{\hat{d}},\sigma_{\hat{d}}^2)] \tag{15}$$

where the variational distribution $q(\theta, \mathbf{z}|\mathbf{i})$ approximates the true posterior $p(\theta|\mu_{\hat{d}}, \sigma_{\hat{d}}^2)$ through Kullback-Leibler divergence.

3.3 Model Training

We need to minimize the overall objective function as

$$L = L_s + \sum_{e \in E}(L_t(e) + L_{\theta,\phi}(e)). \tag{16}$$

The conditional probability exploiting $softmax$ function is computationally expensive according to Eq. (1) and Eq. (8). Therefore, we employ negative sampling (Mikolov et al., 2013b) and Adam (Kingma and Ba, 2015) to optimize the overall objective function. In order to prevent overfitting, we also employ dropout (Srivastava et al., 2014) during the generation of document-topic and topic-word distribution.

4 Experiments

4.1 Datasets

We obtain the datasets shown in Tab. 1 based on the original microblog corpus used by Li et al., (2016a). They collected the posts of 50 frequent hashtags during May 1 -July 31, 2014 through Sina Weibo hashtag-search API[4]. Then they split the whole corpus into 3 datasets and each month represented a datasets. We further deal with the original datasets as follows: 1) Remove the posts whose length is less than 3 words or that have no poster username; 2) Filter users who have no reposting or replying relationship; 3) Aggregate all the original and the reposting posts from the same user to form text information of a user vertex.

Month	#Users	#Reposting
May	8907	10435
June	19293	25962
July	16990	20971

Table 1: Statistics of Datasets.

4.2 Evaluation Metrics

In previous work, a popular metric for topic model, perplexity, is evaluated based on the likelihood of held-out documents. Nonetheless, Chang et al. (2009) have proved that higher likelihood of held-out documents doesn't necessarily correspond to human perception of semantically coherent topics. Instead-ly, we follow (Mimno et al., 2011) to calculate the coherence score of a topic given the top *N* words as

$$\mathcal{C} = \frac{1}{K} \cdot \sum_{k=1}^{K} \sum_{i=2}^{N} \sum_{j=1}^{i-1} \log \frac{D(w_i^k, w_j^k) + 1}{D(w_j^k)} \tag{17}$$

where w_i^k refers to the i-th word in topic k ranked by topic distribution over words, $D(w_i^k, w_j^k)$ stands for the count of documents where w_i^k and w_j^k co-occur,and $D(w_j^k)$ indicates the number of documents that contain word w_j^k. In this paper, a document is the aggregated text of a user.

4.3 Baselines

To validate whether two assumptions mentioned in Section 3 are useful for topic extraction, we compare the proposed IATM with the following state-of-the-art baselines and its two variants:

Text Analysis Only:

(1) **SATM** (Quan et al., 2015) combines the aggregation of short texts and topic inference into an unified model.

(2) **BTM** (Yan et al., 2013) directly models the word co-occurrence patterns in the whole corpus.

(3) **LCTM** (Hu and Tsujii, 2016) uses word embeddings pretrained by a log-linear word2vec model[5] to deduce topics for tackling the data sparsity.

Text and Stucture Analysis:

(4) **LeadLDA** (Li et al., 2016a) generates words according to topic dependencies derived from conversation trees.

(5) **ForumLDA** (Chen and Ren, 2017) models the generation of topics by discriminating response messages relevant or irrelevant to the original post.

Two Variants:

(6) **IATM (-interaction)** is the IATM without dynamic interactions between users, which just takes text information into account and also uses neural variational inference to deduce topics.

[4]http://open.weibo.com/wiki/2/search/topics

[5]https://code.google.com/archive/p/word2vec/

(7) **IATM (-user attention)** considers the interactions between users, but not the various effects of users on the topic.

4.4 Experiment Settings

The hyperparameters of SATM, BTM, LCTM, LeadLDA and ForumLDA are set according to the best hyperparameters reported in their original papers. We run Gibbs samplings (in SATM, BTM, LCTM, LeadLDA and ForumLDA) with 1000 iterations to ensure convergence. For IATM (-interaction) and IATM (-user attention), the parameter settings are kept the same as IATM. Here, we set the vertex embedding dimension d as 200. We take the hyperparameters which achieve the best performance on datasets via an small grid search over combinations of the initial learning rate $[0.001, 0.0001]$, $\alpha \in [0.1, 1]$, $\beta \in [0.1, 1]$ and $\gamma \in [0.1, 1]$. Finally, learning rate is set as 0.001, α as 1.0, β as 0.3 and γ as 0.3. The other parameters are initialized by randomly sampling from normal distribution $\mathcal{N}(1.0, 0.3^2)$.

4.5 Performance Evaluation

Model	$K = 50$			$K = 100$		
	$N = 10$	$N = 15$	$N = 20$	$N = 10$	$N = 15$	$N = 20$
SATM	-76.10	-190.38	-341.22	-76.07	-190.48	-341.46
BTM	-79.27	-181.95	-326.77	-79.58	-183.00	-319.28
LCTM	-70.91	-165.37	-296.36	-58.65	-140.10	-261.40
LeadLDA	-53.91	-138.53	-258.38	-58.15	-141.34	-261.65
ForumLDA	-55.76	-129.57	-231.90	-55.84	-132.23	-236.89
IATM (-interaction)	-60.33	-145.98	-274.08	-64.26	-152.74	-280.40
IATM (-user attention)	-57.77	-126.06	-240.66	-59.57	-131.79	-233.54
IATM	**-43.34**	**-112.64**	**-228.27**	**-47.32**	**-121.46**	**-219.96**

Table 2: Coherence scores on May. Higher is better.

Model	$K = 50$			$K = 100$		
	$N = 10$	$N = 15$	$N = 20$	$N = 10$	$N = 15$	$N = 20$
SATM	**-31.04**	**-24.30**	**-58.39**	**-29.74**	**-22.02**	**-55.00**
BTM	-78.79	-179.99	-321.74	-75.77	-176.13	-315.43
LCTM	-91.72	-208.75	-367.76	-81.88	-181.57	-323.16
LeadLDA	-63.54	-150.18	-278.19	-72.07	-169.80	-309.40
ForumLDA	-78.22	-140.46	-229.62	-82.33	-160.46	-258.72
IATM (-interaction)	-74.61	-180.84	-340.83	-67.29	-161.30	-301.66
IATM (-user attention)	-69.75	-147.53	-234.73	-61.34	-148.01	-280.06
IATM	-46.69	-113.09	-213.61	-59.11	-133.96	-225.48

Table 3: Coherence scores on June. Higher is better.

Model	$K = 50$			$K = 100$		
	$N = 10$	$N = 15$	$N = 20$	$N = 10$	$N = 15$	$N = 20$
SATM	-128.28	-254.45	-431.88	-128.68	-254.74	-432.01
BTM	-73.26	-172.43	-313.12	-76.10	-176.67	-320.70
LCTM	-72.78	-160.08	-275.58	-63.56	-137.36	-238.31
LeadLDA	-70.40	-157.83	-268.23	-59.75	-130.83	-226.62
ForumLDA	-89.16	-215.47	-396.20	-89.96	-213.59	-386.65
IATM (-interaction)	-93.59	-224.17	-409.84	-91.96	-233.86	-396.22
IATM (-user attention)	-61.60	-144.75	-251.46	-57.00	-127.57	-227.81
IATM	**-50.75**	**-119.48**	**-212.26**	**-46.80**	**-110.27**	**-204.35**

Table 4: Coherence scores on July. Higher is better.

Following (Li et al., 2016a; Yan et al., 2013), we set the number of topics to 50 ($K = 50$) and 100 ($K = 100$). $K = 50$ is to match the count of hashtags and $K = 100$ is much larger than the real number of topics. As shown in Tab. 2, 3 and 4, we evaluate the top $N = 10$, 15, 20 words of $K = 50$ and $K = 100$. From these tables, we have the following overall observations:

(1) What call for special attention is that, SATM exhibits the unstable performance on various datasets. Specifically, SATM performs poorly on May, the worst on July and the best on June. It may be ascribed to its heavy reliance on the number of pseudo-documents. On the contrary, IATM has a stable performance and is only outperformed by SATM on June.

(2) Topic models that analyze text and structures perform better than the ones with only text except SATM on June. It indicates that considering texts and structures together is necessary due to their reciprocal influence in a social network.

(3) Our proposed IATM achieves crucial improvement in comparison with all the baselines on May and July, and it gives the greater coherence scores than the other baselines expect SATM on June. The reasons are two-fold: 1) It effectively identifies topics via simultaneously considering social media content and network topology. 2) It further produces interaction-aware edge embedding as a deeper edge representation combined dynamic user interactions and user attention.

Assumptions	$K = 50, N = 20$	$K = 100, N = 20$
Dynamic Interactions	27.32%	22.13%
User Attention	9.91%	11.87%

Table 5: The effects of two hypotheses by comparing IATM with its two variants.

To further evaluate the effectiveness of two assumptions, we compare IATM with its two variants, observations are as follows:

(4) IATM (-interaction) gives the worst coherence scores. The reason is that it has no consideration of social context. After introducing dynamic interactions, IATM (-user attention) obtains the considerable improvement. Moreover, IATM outperforms IATM (-user attention) on three datasets whatever different number of topics. It demonstrates the effectiveness of dynamic interactions (H1) and user attention (H2).

(5) For each component, we compute the average growth percentage under top $N = 20$ words of $K = 50$ and $K = 100$[6] in comparison of IATM and two variants. Seen from Tab. 5, two hypotheses are both useful for topic extraction in social media, and dynamic interactions modeling has greater influence than user attention.

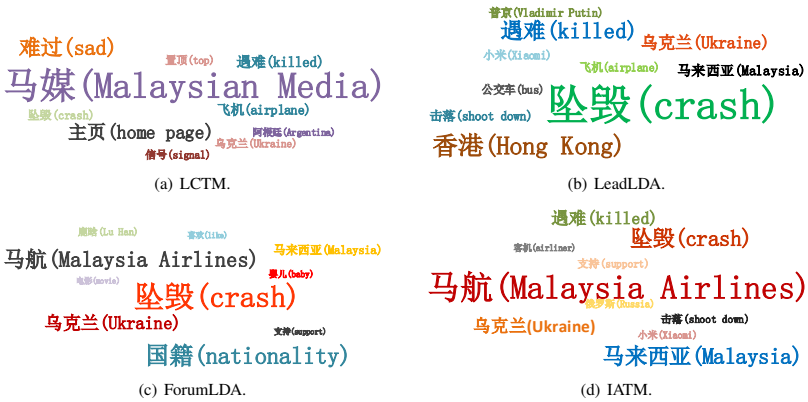
4.6 Case Study

To get an intuitive understanding of extracted topics, we design an experiment to visualize the top 10 words about "MH17 Crash" induced by the different models when $K = 50$, shown in Fig. 3. Due to "MH17 Crash" is included in the July dataset, LCTM gives the highest coherence score among the models of text analysis only. Then we choose LCTM, LeadLDA and ForumLDA as the competitors of our proposed method due to the limited space. Note that, individual words accompany with diverse font sizes. The larger the font size is, the more relevant the word is to the topic. We have the following observations from Fig. 3:

(1) As for LCTM based on word embeddings, "Malaysian Media" is the top one key word and "killed", "crash", "sad" and "Ukraine" is correlated to the topic. However, it mistakenly groups "Argentina lose the World Cup", which co-occurs with "sad" and "Argentina". Compared with LeadLDA, ForumLDA and IATM, LCTM performs the worst. It further testifies that the analysis of texts and structures in a social network is necessary due to their relevance in social media.

(2) With respect to LeadLDA, which distinguishes every message from leader and follower, "crash" is the top one key word and "killed", "Ukraine", "shoot down" and "Malaysia" is related to the topic. However, "bus" is falsely aggregated, which is relevant to the bus explosion in Guangzhou. Maybe it is

[6]The ones under other settings are not shown due to the limited space.



(a) LCTM. (b) LeadLDA. (c) ForumLDA. (d) IATM.

Figure 3: Word clouds describing "MH17 Crash" of different models. Each word cloud represents the similar topic generated by the corresponding model with top 10 words. English translations for the original Chinese words are inside the brackets.

because it wrongly recognizes the reposting message about the bus explosion in Guangzhou as a leader message.

(3) ForumLDA differentiates reposting messages from relevant and irrelevant response posts. It generates the top one key word "crash", and "Malaysia Airlines", "Ukraine", "Malaysia" and "nationality" are correlated to the topic. However, "Lu Han" and "movie" about a movie starring Lu Han, are mistakenly mixed. Perhaps it is because no prior knowledge is given to ensure the validity of relevance distinction.

(4) The top words generated by IATM, "Malaysia Airlines", "Malaysia", "crash" and "Ukrain", are related to the topic. Moreover, some detailed information "airliner" is also produced. Nevertheless, the generated "Xiaomi" is not relevant to the topic. We check the corpus and find that there are lots of posts tagged one hashtag, but expressing the other topic. For instance, someone makes product promotion of "Xiaomi" by utilizing the hot event "MH17 Crash". This belongs to a spam message.

5 Conclusion and Future Work

We propose an Interaction-Aware Topic Model (IATM) for microblog conversations through integrating dynamic interactions and various user attention. Our method not only makes full use of content and structure information in social media, but also keeps the better consistency of generated topics by variational auto-encoder. We model user behaviours through transforming conversation trees into network. Further interaction-aware embeddings produced by adding diverse effects of different users on a topic, encode the content and structure information of two neighbour users. This helps to incorporate the conversation context, and is easy to absorb user attention. Then it is plugged in neural variational inference to generate topics. Experiments on three real-world microblog datasets demonstrate that the proposed IATM is effective.

Yet lots of people would like to utilize hot events to have a product promotion and post some irrelevant messages. These spams damagingly affect the performance of IATM. In the future, we will unify spam posts separation during topics deduction.

Acknowledgments

The work was supported by National Natural Science Foundation of China (61472277, 61772361, 61771333). We would like to thank anonymous reviewers for the valuable and detailed comments and suggestions.

References

Bin Bi, Yuanyuan Tian, Yannis Sismanis, Andrey Balmin, and Junghoo Cho. 2014. Scalable topic-specific influence analysis on microblogs. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 513–522.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems*, pages 288–296.

Chaotao Chen and Jiangtao Ren. 2017. Forum latent dirichlet allocation for user interest discovery. *Knowledge-Based Systems*, 126:1–7.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, Jun Zhao, et al. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 167–176.

Liangjie Hong and Brian D Davison. 2010. Empirical study of topic modeling in twitter. In *Proceedings of the 1st Workshop on Social Media Analytics*, pages 80–88.

Weihua Hu and Junichi Tsujii. 2016. A latent concept topic model for robust topic inference using word embeddings. In *The 54th Annual Meeting of the Association for Computational Linguistics*, pages 380–386.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *Proceedings of the International Conference on Learning Representations*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations*.

Jing Li, Ming Liao, Wei Gao, Yulan He, and Kam-Fai Wong. 2016a. Topic extraction from microblog posts using conversation structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Yang Li, Ting Liu, Jing Jiang, and Liang Zhang. 2016b. Hashtag recommendation with topical attention-based lstm. In *Proceedings of the 26th International Conference on Computational Linguistics*.

Kar Wai Lim, Changyou Chen, and Wray L. Buntine. 2013. Twitter-Network Topic Model: A full Bayesian treatment for social network and text modeling. In *Advances in Neural Information Processing Systems: Topic Models Workshop*, pages 1–5.

Hengyang Lu, Lu-Yao Xie, Ning Kang, Chong-Jun Wang, and Jun-Yuan Xie. 2017. Don't forget the quantifiable relationship between words: Using recurrent neural network for short text topic discovery. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 1192–1198.

Rishabh Mehrotra, Scott Sanner, Wray Buntine, and Lexing Xie. 2013. Improving lda topic models for microblogs via tweet pooling and automatic labeling. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 889–892.

Yishu Miao, Edward Grefenstette, and Phil Blunsom. 2017. Discovering discrete latent topics with neural variational inference. In *Proceedings of the 34th International Conference on Machine Learning*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR, abs/1301.3781*, abs/1312.6114.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 13, pages 746–751.

David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 262–272.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710.

Xiaojun Quan, Chunyu Kit, Yong Ge, and Sinno Jialin Pan. 2015. Short and sparse text topic modeling via self-aggregation. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 2270–2276.

Cicero Dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR, abs/1602.03609*.

Vivek Kumar Rangarajan Sridhar. 2015. Unsupervised topic modeling for short texts using distributed representations of words. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 192–200.

Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. *Proceedings of the International Conference on Learning Representations*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Jian Tang, Ming Zhang, and Qiaozhu Mei. 2013. One theme in all views: modeling consensus topics in multiple contexts. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 5–13.

Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. Cane: Context-aware network embedding for relation modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1722–1731.

Xuepeng Wang, Kang Liu, and Jun Zhao. 2017. Handling cold-start problem in review spam detection by jointly embedding texts and behaviors. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 366–376.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 3351–3357.

Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. A biterm topic model for short texts. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1445–1456.

Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. 2011. Comparing twitter and traditional media using topic models. In *Proceedings of the 33rd European Conference on Information Retrieval*, pages 338–349.

Hao Zhuang, Rameez Rahman, Xia Hu, Tian Guo, Pan Hui, and Karl Aberer. 2016. Data summarization with social contexts. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 397–406.

Cross-media User Profiling with Joint Textual and Social User Embedding

Jingjing Wang[1,2], **Shoushan Li**[1,2], **Mingqi Jiang**[1,2], **Hanqian Wu**[3], **Guodong Zhou**[1,2,*]
[1]NLP Lab, School of Computer Science and Technology, Soochow University, China
[2]Collaborative Innovation Center of Novel Software Technology and Industrialization
[3]School of Computer Science and Engineering, Southeast University, China
djingwang@gmail.com, {lishoushan, gdzhou}@suda.edu.cn
20165227010@stu.suda.edu.cn, hanqian@seu.edu.cn

Abstract

In realistic scenarios, a user profiling model (e.g., gender classification or age regression) learned from one social media might perform rather poorly when tested on another social media due to the different data distributions in the two media. In this paper, we address cross-media user profiling by bridging the knowledge between the *source* and *target* media with a uniform user embedding learning approach. In our approach, we first construct a cross-media user-word network to capture the relationship among users through the textual information and a modified cross-media user-user network to capture the relationship among users through the social information. Then, we learn user embedding by jointly learning the heterogeneous network composed of above two networks. Finally, we train a classification (or regression) model with the obtained user embeddings as input to perform user profiling. Empirical studies demonstrate the effectiveness of the proposed approach to two cross-media user profiling tasks, i.e., cross-media gender classification and cross-media age regression.

1 Introduction

User profiling is a task which leverages user generated content (UGC) to automatically identify the data about a user, such as gender (Wang et al., 2017; Li et al., 2016; Li et al., 2015) and age (Marquardt et al., 2014) identification. Recently, along with the boom of social media, user profiling has been drawing more and more attention in various social applications, such as personality analysis (Li et al., 2016; Sarawgi et al., 2011; O'Connor et al., 2010), intelligent marking (Preotiuc-Pietro et al., 2015) and online advertising (Zhang et al., 2016; Volkova et al., 2013).

In the literature, conventional approaches normally recast user profiling as a supervised learning problem (Marquardt et al., 2014; Zhang et al., 2016; Ciot et al., 2013; Corney et al., 2002) by exploring various user generated textual features and user social connection features. However, the performance largely depends on a large amount of labeled data and the performance normally degrades dramatically when the model is tested on a different social media. Even worse, many real scenarios may involve several social media with some of them lacking sufficient labeled data to train a well-performed model in each social media. For example, Facebook.com is a social media site where many users publicize their age attribute in their homepages, making the collection of labeled data easy, while Linkedin.com is a social media site where age attribute information is not available in users homepages, making the collection of labeled data rather difficult. These scenarios raise a challenging user classification task, which leverages labeled data from a social media to train a model and evaluate it on the test data from a different social media. Briefly, we refer to this task as cross-media user profiling where the social media with labeled data is called the *source* media and the social media with only unlabeled data the *target* media.

In this paper, we address cross-media user profiling by bridging the knowledge between the *source* and *target* media with a uniform user embedding learning approach. As a user representation way, user

*Guodong Zhou is Corresponding Author.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1410–1420
Santa Fe, New Mexico, USA, August 20-26, 2018.

User **A** from *SINA*

Gender: *female* **Age:** *22*

Social Link Information (e.g., *followers*)
▷ **ID1:** *513950031*
▷ **ID2:** *205165814*

Textual Information (e.g., *messages*)
(1) *Goodbye, beautiful school and dear teacher. Its time to celebrate Christmas holiday, hahahaha.*
(2) *Big surprise! So amazing Louis Vuitton's necklace for Christmas gift. Love you forever, dear boyfriend.*

User **B** from *TIEBA*

Gender: *female* **Age:** *22*

Social Link Information (e.g., *followers*)
▷ **ID3:** *32d0aa2ce4*
▷ **ID4:** *7dfc636a6a*

Textual Information (e.g., *messages*)
(1) *UGH i don't wanna go to school tomorrow. Don't wanna see a teacher again. Oh wait, i have been 22! Wahooooo, forgive me, God..*
(2) *What a perfect necklace! Matching my earrings so perfect!! Don't cut my hands, please, haha.*

Figure 1: The user examples from two different social media, i.e., *SINA* and *TIEBA* social media

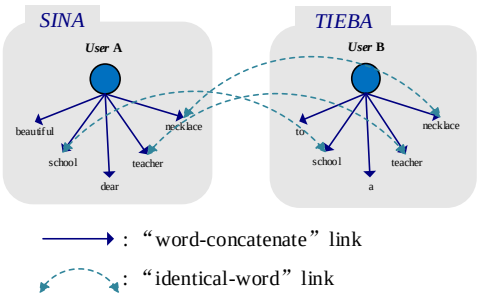


Figure 2: Cross-media user-word network

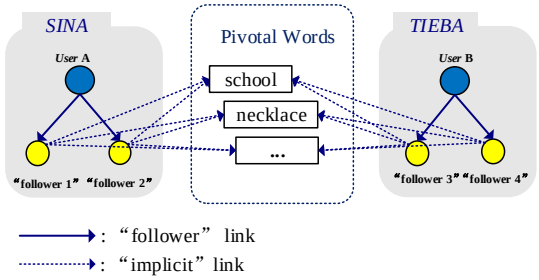


Figure 3: Cross-media user-user network with pivotal word vertices

embedding maps a user to a vector of real numbers with the motivation that embedding learning is good at capturing the relationship among the instances in the unlabeled data from both the *source* and *target* media.

One straightforward way to learn user embedding is to construct two user-word networks. Due to the phenomenon of sharing some identical words by two users, these two user-word networks are naturally connected and thus can be merged to be a mixture network, namely cross-media user-word network. For instance, Figure 1 shows two users, i.e., *User* **A** and *User* **B**, from two social media, i.e., *SINA* and *TIEBA*. It is easy to construct a user-word network wherein the users from different social media are naturally connected due to their sharing some identical words, as shown in Figure 2. Given this network, a network embedding approach, e.g., the LINE approach by Tang et al. (2015b), could be applied to learn vertex embedding and thus as user embedding.

Obviously, one main drawback of the above approach is that it much depends on the textual information and ignores the social link information which is important for user classification. Thus, a better approach to learn user embedding is to incorporate both the textual and social link information.

However, incorporating the social link information is challenging because user IDs in the different social media are totally different. As a result, the two user-user networks in the two social media are separated to each other, making it impossible to learn the relationship between two users from different social media. To tackle this challenge, we link the two user-user networks in the *source* and *target* media with some pivotal word vertices, as shown in Figure 3.

Besides, we learn better user embedding by learning a heterogeneous network composed of the two above networks, i.e., the cross-media user-word network and the cross-media user-user network. Once obtaining user embedding, we train a classifier (or regressor) on the labeled data from the *source* media

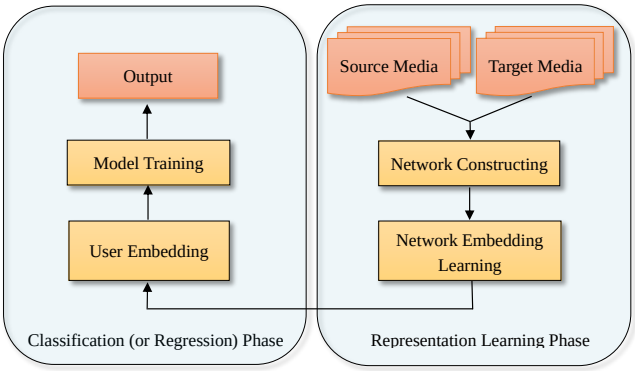


Figure 4: The overall architecture of our approach

and evaluate on the data from the *target* media. Empirical studies demonstrate that our approach outperforms both the straightforward baseline and conventional domain adaptation approaches on two different cross-media user profiling tasks, i.e., cross-media gender classification and age regression.

2 Related Work

In the last decade, many researchers have devoted their efforts on user profiling (e.g., gender identification and age identification) in several research communities, such as natural language processing and social network analysis.

For the gender identification task, Mohammad and Yang (2013) show that there are marked differences across genders in how they use emotion words in work-place email. Ciot et al. (2013) conduct the first assessment of latent attribute inference in various languages beyond English, focusing on gender inference of Twitter users. Li et al. (2015) aim to identify the genders of two interactive users on the basis of micro-blog text. Some other studies, such as Mukherjee and Liu (2010), Peersman et al. (2011) and Gianfortoni et al. (2011) focus on exploring more effective features to improve the performance. Wang et al. (2017) propose a joint learning approach in order to leverage the relationship features among relevant user attributes.

For the age identification task, most studies are devoted to explore efficient features in blog and social media. Schler et al. (2006) focus on textual features extracted from the blog text, such as word context features and POS stylistic features. Peersman et al. (2011) apply a text categorization approach to age classification with textual features extracted from the text in social media. More recently, Marquardt et al. (2014) propose a multi-label classification approach to predict both the gender and age of authors from texts adopting some sentiment and emotion features. Differently, in this paper, we cast age identification task as a regression problem instead of a classification problem.

Different from all above studies, we focus on the cross-media user profiling task. To the best of our knowledge, there are only two related papers by (Sap et al., 2014; Pardo et al., 2016) which mentions the cross-media user profiling issue. In their paper, only textual information is used in cross-media user profiling. Unlike their study, this paper firstly employs both textual and social information in cross-media user profiling.

3 Our Approach

3.1 Framework Overview

Our approach consists of two main phases, i.e., the representation learning phase and the classification (or regression) phase, which has been illustrated in Figure 4.

In the representation phase, we first construct two different types of cross-media networks, i.e., cross-media user-word network and cross-media user-user network. Second, we construct the heterogeneous user network, which is composed of the two cross-media networks. Third, we perform user embedding learning on a heterogeneous user network.

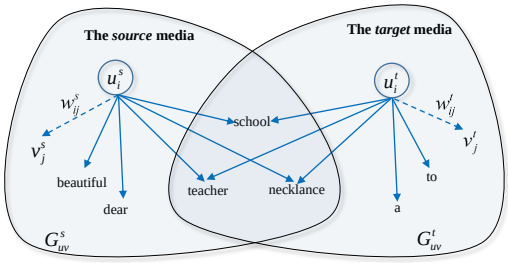


Figure 5: Cross-media user-word network G_{uv}^{cross}

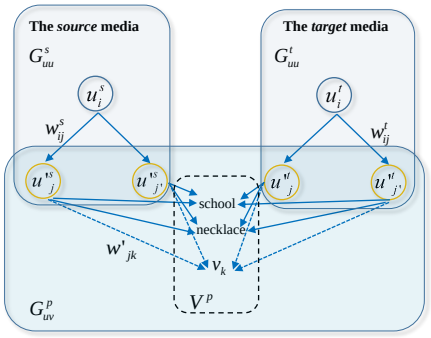


Figure 6: Cross-media user-user network G_{uu}^{cross}

In the classification (or regression) phase, we train a classifier (or regressor) with user embeddings in the *source* media for cross-media user profiling.

3.2 Basic Network Embedding Learning Approach

The goal of user embedding learning is to represent a user by a low dimensional real-value vector. In this study, we adopt the network embedding learning approach by Tang et al. (2015b) as our basic user embedding learning approach, of which we make use of the second-order proximity. Specifically, this approach is summarized as following.

Given a network $G = (V, E)$, where V is the set of vertices. E is the set of edges and the edge $e_{(i,j)}$ is directed from vertex v_i to the vertex v_j. To represent each vertex v_i with a low dimension vector $\vec{m}_i \in R^d$ where d is the dimension of this vector, we can minimize the following objective function (Tang et al., 2015b):

$$O = -\sum_{e_{(i,j)} \in E} w_{ij} \log p(v_j|v_i) \tag{1}$$

Where w_{ij} is the weight of the edge $e_{(i,j)}$. The conditional probability of vertex v_j generated by the vertex v_i, i.e., $p(v_j|v_i)$ is defined as follows:

$$p(v_j|u_i) = \frac{\exp(\vec{m}_j^\top \cdot \vec{m}_i)}{\sum_{k=1}^{|V|} \exp(\vec{m}_k^\top \cdot \vec{m}_i)} \tag{2}$$

where $\vec{m}_i$ is the embedding vector of the vertex v_i, and $\vec{m}_j$ is the embedding vector of the vertex v_j. $|V|$ is the number of all vertices in V.

3.3 Our Network Embedding Learning Approach

In our approach, we need to construct three different networks for learning network embedding, i.e., **(a)** Cross-Media User-Word Network; **(b)** Cross-Media User-User Network; **(c)** Heterogeneous User Network.

(a) Cross-Media User-Word Network Embedding

Figure 5 shows the cross-media user-word network G_{uv}^{cross}, which is composed of two user-word networks, i.e., G_{uv}^s and G_{uv}^t.

Formally, $G_{uv}^s = (U^s \bigcup V^s, E_{uv}^s)$ is a user-word network from the *source* media. It is a bipartite network where U^s and V^s are two disjoint sets of vertices, denoting the user vertices and word vertices respectively. E_{uv}^s is the set of edges between users and their published words in the *source* media.

$G_{uv}^t = (U^t \bigcup V^t, E_{uv}^t)$ is a user-word network from the *target* media. It is also a bipartite network where U^t and V^t are two disjoint sets of vertices, denoting the user vertices and word vertices respectively. E_{uv}^t is the set of edges between users and their published words in the *target* media.

For the network G_{uv}^{cross}, we can minimize the following objective function:

$$O_{uv} = -\sum_{e_{(i,j)} \in E_{uv}^{s}} w_{ij}^{s} \log p_{uv}(v_j^s | u_i^s) - \sum_{e_{(i,j)} \in E_{uv}^{t}} w_{ij}^{t} \log p_{uv}(v_j^t | u_i^t) \tag{3}$$

where the edge $e_{(i,j)} \in E_{uv}^{s}$ *or* E_{uv}^{t} is directed from user vertex u_i to word vertex v_j. The weight w_{ij} of the edge $e_{(i,j)}$ in both the *source* and *target* media is simply defined as the number of times v_j appears in the messages published by u_i. $p_{uv}(v_j^s|u_i^s)$ and $p_{uv}(v_j^t|u_i^t)$ are two conditional probabilities of a word vertex generated by a user vertex, which could be computed by the vertex embeddings, as shown in Equation (2).

(b) Cross-Media User-User Network Embedding

Figure 6 shows the cross-media user-user network G_{uu}^{cross}, which is a network composed of two user-user networks, i.e., G_{uu}^{s} and G_{uu}^{t}, and one user-pivotal_word network, i.e., G_{uv}^{p}.

Formally, $G_{uu}^{s} = (U^s, E_{uu}^{s})$ a user-user network from the *source* media. U^s is the entire set of users in the *source* media. E_{uu}^{s} is the set of edges between users and their "follower" users in the *source* media.

$G_{uu}^{t} = (U^t, E_{uu}^{t})$ is a user-user network from the *target* media. U^t is the entire set of users in *target* media. E_{uu}^{t} is the set of edges between users and their "follower" users in the *target* media.

As mentioned in Introduction, the user-user networks in both the *source* and *target* media share no identical users, i.e., $G_{uu}^{s} \bigcap G_{uu}^{t} = \Phi$, which makes it impossible to build connection among the users in both social media. Therefore, we attempt to construct another sub-network, a user-pivotal_word network G_{uv}^{p}, to bridge the users in both social media.

$G_{uv}^{p} = (U'^s \bigcup V^p \bigcup U'^t, E_{uu}^{p})$ is a user-pivotal_word network, which is constructed to link the above two networks G_{uu}^{s} and G_{uu}^{t}. It is a bipartite network where $U'^s \subseteq U^s$ is the set of "follower" users from the *source* media and $U'^t \subseteq U^t$ is the set of "follower" users from the *target* media. E_{uu}^{p} is the set of edges between each "follower" user and the words in V^p which are also published by his/her following user. The basic motivation of adding these edges is based on the assumption that one user's followers tend to have similar relationship to the words that are published by this user. The detailed process of constructing the network G_{uv}^{p} has been illustrated in **Algorithm 1**.

For the network G_{uu}^{cross}, we can minimize the following objective function:

$$O_{uu} = -\sum_{e_{(i,j)} \in E_{uu}^{s}} w_{ij}^{s} \log p_{uu}(u_j'^s | u_i^s) - \sum_{e_{(i,j)} \in E_{uu}^{t}} w_{ij}^{t} \log p_{uu}(u_j'^t | u_i^t) - \sum_{e_{(i,k)} \in E_{uu}^{p}} w_{ik}' \log p_{uu}(v_k | u_i') \tag{4}$$

where the edge $e_{(i,j)} \in E_{uu}^{s}$ *or* E_{uu}^{t} is directed from user vertex u_i to its "follower" vertex $u_j' \in U'^s or U'^t$. The edge $e_{(i,k)} \in E_{uu}^{p}$ is directed from a "follower" vertex u_i' to a pivotal word vertex $v_k \in V^p$. The weights of all above edges are defined as binary values (i.e., 0 or 1). $p_{uu}(u_j'^s|u_i^s)$ and $p_{uu}(u_j'^t|u_i^t)$ are two conditional probabilities of "follower" vertex $u_j'^s$ and $u_j'^t$ generated by user vertex u_i^s and u_i^t respectively. $p_{uu}(v_k|u_i')$ is the conditional probability of pivotal word vertex v_k generated by the "follower" vertex u_i'. These three kinds of probabilities could be computed by the vertex embeddings, as shown in Equation (2).

(c) Heterogeneous User Network Embedding

The heterogeneous user network G_{joint} is composed of two cross-media networks: the cross-media user-word network, i.e., G_{uv}^{cross} and the cross-media user-user network, i.e., G_{uu}^{cross}, where the user vertices are shared across the two networks.

To learn the embeddings of the heterogeneous user network, an intuitive approach is to collectively embed the two cross-media networks, which can be achieved by minimizing the following objective function:

$$O_{joint} = O_{uv} + O_{uu} \tag{5}$$

Algorithm 1: The Network G^p_{uv} Constructing

Input: U^s: users from the *source* media;
V^s: words from the *source* media;
U'^s: "follower" users from the *source* media;
U^t: users from the *target* media;
V^t: words from the *target* media;
U'^t: "follower" users from the *target* media
Output: V^p, E^p_{uu}
Initialization: $E^p_{uu} = \Phi$;
// V^p: words shared between *source* and *target* media;
$V^p = V^s \bigcap V^t$;
for $i = 1; i \leq |U^s|; i+ = 1$ **do**
  // $V_i \in V^s$: words published by u^s_i;
  for v_k *in* V_i **do**
    if $v_k \in V^p$ **then**
      // $U_i \in U'^s$: "follower" users of u^s_i;
      for u'_j *in* U_i **do**
        generate the edge $e_{(j,k)}$ from u'_j to v_k;
        $E^p_{uu} = E^p_{uu} \bigcup e_{(j,k)}$;
for $i = 1; i \leq |U^t|; i+ = 1$ **do**
  // $V_i \in V^t$: words published by u^t_i;
  for v_k *in* V_i **do**
    if $v_k \in V^p$ **then**
      // $U_i \in U'^t$: "follower" users of u^t_i;
      for u'_j *in* U_i **do**
        generate the edge $e_{(j,k)}$ from u'_j to v_k;
        $E^p_{uu} = E^p_{uu} \bigcup e_{(j,k)}$;
return V^p, E^p_{uu};

We adopt the asynchronous stochastic gradient descent (ASGD) (Recht et al., 2011) using the techniques of edge sampling (Tang et al., 2015b) and negative sampling (Mikolov et al., 2013) for optimizing Equation (3) and (4). In each step, we sample a binary edge $e_{(i,j)}$ with the sampling probabilities proportional to it's edge weight w_{ij}. Meanwhile, multiple negative edges $e_{(i,j)}$ are sampled from a noise distribution $p_n(i) \propto d_i^{3/4}$ as proposed in (Mikolov et al., 2013), where d_i is the out-degree of user vertex u_i. The Equation (5) can be optimized with the textual data (the user-word network) and the social link data (the user-user network) simultaneously. We call this approach joint learning. In joint learning, both the two types of cross-media networks are used together. When the network G_{joint} is heterogeneous, the weights of the edges in the two different networks, i.e., G^{cross}_{uv} and G^{cross}_{uu} are not comparable to each other. Therefore, in our approach, we alternatively sample from the two sets of edges instead of merging all the edges together to sample as the way by Tang et al. (2015a).

4 Experimentation

4.1 Experimental Settings

Data Setting: The users are collected from *SINA*[1] and *TIEBA*[2], two famous social media in China. From these two social media, we crawl each user's homepage containing user information (e.g. *name*, *gender*,*age*), messages and "follower" users. The data collection process starts from some randomly selected users, and then iteratively gets the data of their followers. In total, we collect 10000 users from *SINA* and *TIEBA* respectively. For cross-media gender classification task, in each social media, we randomly select a balanced data set containing 3000 male users and 3000 female users. For cross-media age regression task, we focus on the age from 19-28, totally 10 age categories and extract 6000 users from each social media. This data contains a balanced data set in each age, i.e., 600 samples in each age. For

[1] https://weibo.com/
[2] https://tieba.baidu.com/index.html

both above two cross-media user profiling tasks, We randomly select 80% users in each category from the *source* media as the labeled data, 80% users in each category from the *target* media as the unlabeled data and the remaining 20% users from the *target* media as test data. Then, we consider two cases: one is to consider *TIEBA* as the *source* media and *SINA* as the *target* media, i.e., TIEBA → SINA, and the other is to consider *SINA* as the *source* media and *TIEBA* as the *target* media, i.e., SINA → TIEBA.

Representations: In our experiment, we use three different representation models to represent each user. (1) **BOW**: Each user is represented by a bag of features consisting of four types of textual features, including word unigram and two kinds of complex features, i.e., F-measure, POS-pattern, are employed. These kinds of textual features yield the state-of-the-art performance for gender classification (Li et al., 2015); To get the word, POS and parse tree features, we use the public toolkit ICTCLAS[3] to perform word segmentation, POS tagging and stanford parser[4] to perform parsing on the Chinese text. (2) **Word Embeddings**: We learn word embedding matrices of each user from training a mixed training data set consisting of labeled data in the *source* media and unlabeled data in the *target* media with skip-gram algorithm[5]. The dimensionality of word vector is set to be 200. (3) **User Embeddings**: We learn the embedding vector of each user from training three different cross-media networks, i.e., G_{uv}^{cross}, G_{uu}^{cross} and G_{joint}, which are constructed from a mixed training data set consisting of the labeled data in *source* media and unlabeled data in *target* media. For the new user vertex which not present in the training data, we get its user embedding by averaging the embedding vectors of all words published by this user. The dimensionality of user embedding vector and word embedding vector are both set to be 200.

Model Training and Parameter Settings: In the classification (or regression) phase, we employ L2-regularized SVM (or support vector regression, SVR) model in the LibLinear package[6]. For Word Embeddings and User Embeddings, the mini-batch size of the stochastic gradient descent is set to be 1; the learning rate is set to be $\rho_t = \rho_0(1 - t/T)$, in which T is the total number of edge samples and $\rho_0 = 0.025$; the number of negative samples is set to be 5; the window size is set as 5 in Skip-gram.

Evaluation Metrics and Significance test: For gender classification task, the performance is evaluated using Macro-F1 (F). For age regression task, we employ the coefficient of determination R^2 to measure the regression performance (Cameron, 1996). Furthermore, t-test is used to evaluate the significance of the performance difference between two approaches (Yang and Liu, 1999).

4.2 Impact of Using User-pivotal_word Network G_{uv}^{p}

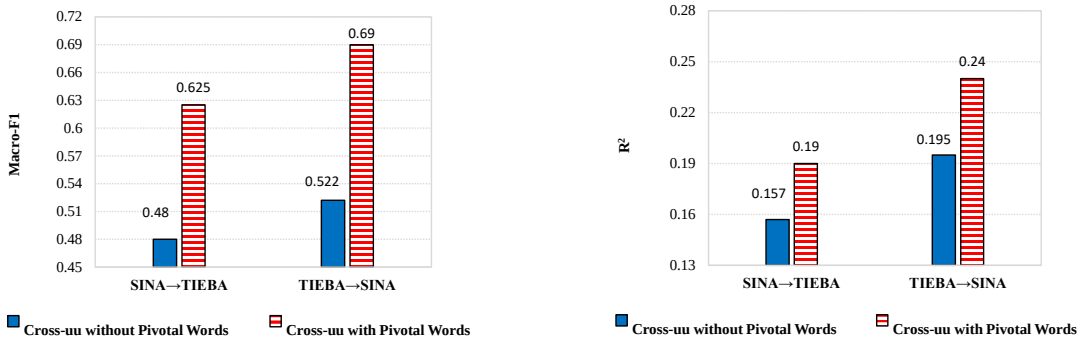
In order to test the effectiveness of using user-pivotal_word network G_{uv}^{p} to form the cross-media user-user network, the following two approaches are implemented for cross-media user profiling.

- **Cross-uu without Pivotal Words:** our model with **User Embeddings** as input. The user embeddings are learned from the cross-media user-user network which is composed of two user-user networks, i.e., $G_{uu}^{cross} = G_{uu}^{s} + G_{uu}^{t}$.

- **Cross-uu with Pivotal Words:** our model with **User Embeddings** as input. The user embeddings are learned from the cross-media user-user network which is composed of two user-user networks and one user-pivotal_word network, i.e., $G_{uu}^{cross} = G_{uu}^{s} + G_{uu}^{t} + G_{uv}^{p}$.

Figure 7(a) and 7(b) shows the results of the two approaches to cross-media gender classification and cross-media age regression respectively. From this figure, we can see that **Cross-uu without Pivotal Words** performs similar to a random classification, achieving about 50% in terms of Macro-F1 in gender classification task. This result is not strange because the involved two user-user networks share nothing and thus could not capture the relationship among the users in the two social media. In contrast, **Cross-uu with Pivotal Words** performs much better than a random performance. Especially, in gender classification task, in the case of TIEBA → SINA, our approach achieves 0.69 in terms of Macro-F1,

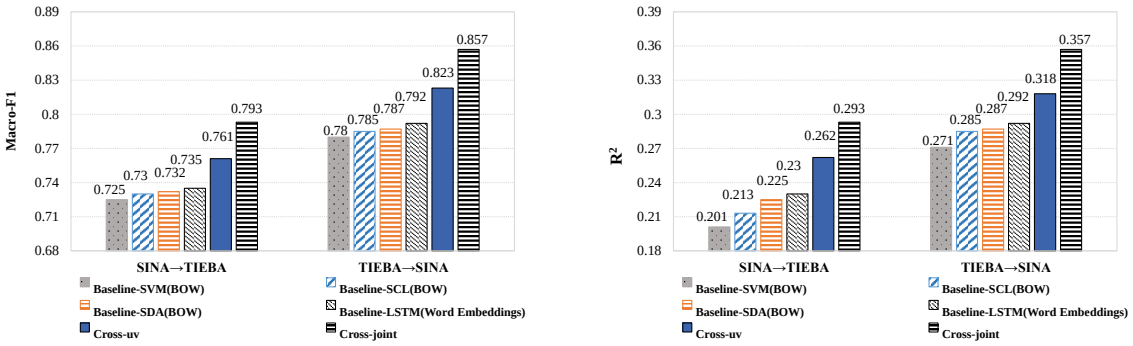
[3] http://www.ictclas.org/ictclas_download.aspx
[4] http://nlp.stanford.edu/software/lex-parser.shtml
[5] https://github.com/dav/word2vec
[6] http://www.csie.ntu.edu.tw/~cjlin/liblinear/



(a) Cross-media gender classification

(b) Cross-media age regression

Figure 7: Performances of two approaches (i.e., using or not using user-pivotal_word network).



(a) Cross-media gender classification

(b) Cross-media age regression

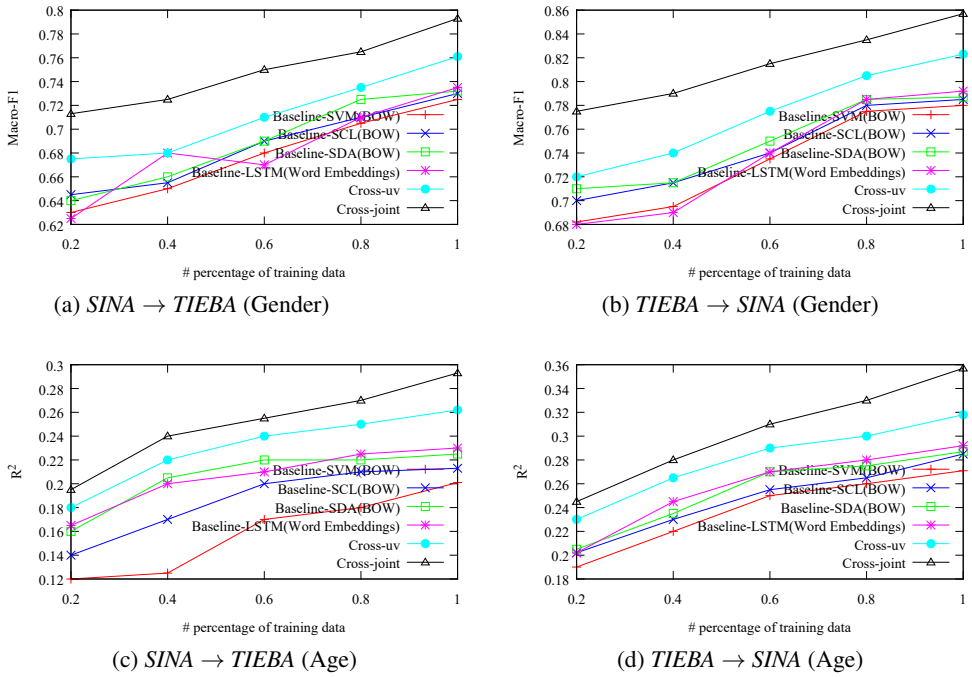
Figure 8: Performances of different approaches.

which is 16.8% better than **Cross-uu without Pivotal Words**. Moreover, in age aggression task, **Cross-uu with Pivotal Words** still outperforms **Cross-uu without Pivotal Words** about 4.5% in terms of R^2. These results confirm that constructing the user-pivotal_word network is beneficial for learning a better user embedding for cross-media user profiling.

4.3 Performance Comparison

For thorough comparison, several approaches are implemented for cross-media user profiling:

- **Baseline-SVM(BOW):** a SVM classifier (or SVR regressor) trained with only labeled data in the *source* media and the representation model is **BOW**. This approach is proposed by Li et al. (2015).
- **Baseline-SCL(BOW):** a famous textual domain adaptation approach named SCL which bridges the knowledge between the *source* and *target* domains using some pivotal features (Blitzer et al., 2007) and the representation model is **BOW**. We consider the *source* media as the *source* domain and the *target* media as the *target* domain. The number of the pivotal features is set to be 200.
- **Baseline-SDA(BOW):** another famous textual domain adaptation approach with stacked denoising auto-encoder to extract meaningful representation (Glorot et al., 2011) and the representation model is **BOW**. Similar to SCL, we consider the *source* media as the *source* domain and the *target* media as the *target* domain.
- **Baseline-LSTM(Word Embeddings):** a LSTM classification (or regression) model using **Word Embeddings** as input. This is a state-of-the-art approach to user profiling proposed by Wang et al. (2017).
- **Cross-uv:** our model with **User Embeddings** as input. The user embeddings are learned from the cross-media user-word network, i.e., G_{uv}^{cross}.



(a) *SINA* → *TIEBA* (Gender)

(b) *TIEBA* → *SINA* (Gender)

(c) *SINA* → *TIEBA* (Age)

(d) *TIEBA* → *SINA* (Age)

Figure 9: Performance of different approaches on various sizes of training data

- **Cross-joint:** our model with **User Embeddings** as input. The user embeddings are learned from the heterogeneous user network, i.e., G_{joint}.

Figure 8(a) and 8(b) shows the comparison results of different approaches to cross-media gender classification and cross-media age regression respectively. From these two figures, we can see that: The textual domain adaptation approaches including **Baseline-SCL(BOW)** and **Baseline-SDA(BOW)** are effective for cross-media user profiling and they both performs slightly better than **Baseline-SVM(BOW)**. Our approach **Cross-uv** performs consistently better than the textual domain adaptation approaches. The improvements are about 2% in terms of both Macro-F1 and R^2. The significance test shows that the improvements are significant (p-value<0.05). Besides, our approach performs better than **Baseline-LSTM(Word Embeddings)**, which indicates that our approach yields better user embeddings than the traditional word embedding approach.

Our approach **Cross-joint** performs best among all approaches. Compared to **Baseline-SVM(BOW)**, the average improvement is impressive, reaching 7.25% (Macro-F1) and 8.9% (R^2) in gender classification and age regression respectively. Furthermore, **Cross-joint** significantly performs better than **Cross-uv** (p-value<0.05), which encourages to incorporate the social link information for cross-media user profiling.

Figure 9 shows the performance of the approaches to cross-media user profiling by changing the sizes of the training data in the *source* media. From this figure, we can see that when the size of training data is small, the performance of **Baseline-LSTM(Word Embeddings)** is rather unstable, performing worse than **Baseline-SDA(BOW)** and **Baseline-SCL(BOW)** in some times and could even performs worse than **Baseline-SVM(BOW)**. However, our approach **Cross-joint** and **Cross-uv** consistently perform better than the other approaches, whatever sizes of training data are used.

5 Conclusion

In this paper, we propose a user embedding learning method to address cross-media user profiling. Specifically, we first propose a heterogeneous user network composed of two cross-media networks for learning user embedding and then employ the user embedding to train the model for classification (or regression). Empirical studies demonstrate that our approach significantly outperforms other strong baseline approaches in both cross-media gender classification and cross-media age regression tasks.

In our future work, we would like to incorporate some other types of information, e.g., label information, to better learn user embedding. Moreover, we would like to apply our approach to some other user profiling tasks, e.g., user profession identification.

Acknowledgments

We thank anonymous reviewers for their valuable suggestions and comments. The research work is supported by three National Natural Science Foundation of China (No.61751206, No.61331011 and No.61672366) and Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of ACL-2007*.

A. Colin Cameron. 1996. R-squared measures for count data regression models with applications to health-care utilization. *Journal of Business & Economic Statistics*, 14(2):209–220.

Morgane Ciot, Morgan Sonderegger, and Derek Ruths. 2013. Gender inference of twitter users in non-english contexts. In *Proceedings of EMNLP-2013*, pages 1136–1145.

Malcolm Corney, Olivier Y. de Vel, Alison Anderson, and George M. Mohay. 2002. Gender-preferential text mining of e-mail discourse. In *Proceedings of ACSAC-2002*, pages 282–289.

Philip Gianfortoni, David Adamson, and Carolyn P Rosé. 2011. Modeling of stylistic variation in social media with stretchy patterns. In *Proceedings of EMNLP-2011*, pages 49–59. Association for Computational Linguistics.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of ICML-2011*, pages 513–520.

Shoushan Li, Jingjing Wang, Guodong Zhou, and Hanxiao Shi. 2015. Interactive gender inference with integer linear programming. In *Proceedings of IJCAI-2015*, pages 2341–2347.

Shoushan Li, Bin Dai, Zhengxian Gong, and Guodong Zhou. 2016. Semi-supervised gender classification with joint textual and social modeling. In *Proceedings of COLING-2016*, pages 2092–2100.

James Marquardt, Golnoosh Farnadi, Gayathri Vasudevan, Marie-Francine Moens, Sergio Davalos, Ankur Teredesai, and Martine De Cock. 2014. Age and gender identification in social media. In *Proceedings of CLEF-2014*, pages 1129–1136.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Saif M. Mohammad and Tony Yang. 2013. Tracking sentiment in mail: How genders differ on emotional axes. *CoRR*, abs/1309.6347.

Arjun Mukherjee and Bing Liu. 2010. Improving gender classification of blog authors. In *Proceedings of EMNLP-2010*, pages 207–217.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of ICWSM-2010*.

Francisco Manuel Rangel Pardo, Paolo Rosso, Ben Verhoeven, Walter Daelemans, Martin Potthast, and Benno Stein. 2016. Overview of the 4th author profiling task at PAN 2016: Cross-genre evaluations. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016.*, pages 750–784.

Claudia Peersman, Walter Daelemans, and Leona Van Vaerenbergh. 2011. Predicting age and gender in online social networks. In *Proceedings of CIKM-2011*, pages 37–44.

Daniel Preotiuc-Pietro, Vasileios Lampos, and Nikolaos Aletras. 2015. An analysis of the user occupational class through twitter content. In *Proceedings of ACL-2015*, pages 1754–1764.

Benjamin Recht, Christopher Ré, Stephen J. Wright, and Feng Niu. 2011. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Proceedings of NIPS-2011*, pages 693–701.

Maarten Sap, Gregory J. Park, Johannes C. Eichstaedt, Margaret L. Kern, David Stillwell, Michal Kosinski, Lyle H. Ungar, and Hansen Andrew Schwartz. 2014. Developing age and gender predictive lexica over social media. In *Proceedings of EMNLP-2014*, pages 1146–1151.

Ruchita Sarawgi, Kailash Gajulapalli, and Yejin Choi. 2011. Gender attribution: Tracing stylometric evidence beyond topic and genre. In *Proceedings of CONLL-2011*, pages 78–86.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W. Pennebaker. 2006. Effects of age and gender on blogging. In *Proceedings of AAAI-2006*, pages 199–205.

Jian Tang, Meng Qu, and Qiaozhu Mei. 2015a. PTE: predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of SIGKDD-2015*, pages 1165–1174.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015b. LINE: large-scale information network embedding. In *Proceedings of WWW-2015*, pages 1067–1077.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring demographic language variations to improve multilingual sentiment analysis in social media. In *Proceedings of EMNLP-2013*, pages 1815–1827.

Jingjing Wang, Shoushan Li, and Guodong Zhou. 2017. Joint learning on relevant user attributes in micro-blog. In *Proceedings of IJCAI-2017*, pages 4130–4136.

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *Proceedings of SIGIR-1999*, pages 42–49.

Dong Zhang, Shoushan Li, Hongling Wang, and Guodong Zhou. 2016. User classification with multiple textual perspectives. In *Proceedings of COLING-2016*, pages 2112–2121.

Incorporating Syntactic Uncertainty in Neural Machine Translation with a Forest-to-Sequence Model

Poorya Zaremoodi **Gholamreza Haffari**
Faculty of Information Technology, Monash University
firstname.lastname@monash.edu

Abstract

Incorporating syntactic information in Neural Machine Translation (NMT) can lead to better reorderings, particularly useful when the language pairs are syntactically highly divergent or when the training bitext is not large. Previous work on using syntactic information, provided by top-1 parse trees generated by (inevitably error-prone) parsers, has been promising. In this paper, we propose a *forest*-to-sequence NMT model to make use of exponentially many parse trees of the source sentence to compensate for the parser errors. Our method represents the collection of parse trees as a packed forest, and learns a neural transducer to translate from the input forest to the target sentence. Experiments on English to German, Chinese and Farsi translation tasks show the superiority of our approach over the sequence-to-sequence and tree-to-sequence neural translation models.

1 Introduction

The neural approach is revolutionising machine translation (MT). The main neural approach to MT is based on the encoder-decoder architecture (Cho et al., 2014; Sutskever et al., 2014), where an encoder (e.g a recurrent neural network) reads the source sentences *sequentially* to produce a fixed-length vector representation. Then, a decoder generates the translation from the encoded vector, which can dynamically change using the *attention* mechanism.

One of the main premises about natural language is that words of a sentence are inter-related according to a (latent) hierarchical structure, i.e. a syntactic tree. Therefore, it is expected that modeling the syntactic structure should improve the performance of NMT, especially in low-resource or linguistically divergent scenarios, such as English-Farsi. In this direction, (Li et al., 2017) uses a sequence-to-sequence model, making use of *linearised* parse trees. (Chen et al., 2017b) has proposed a model which uses syntax to constrain the dynamic encoding of the source sentence via structurally constrained attention. (Bastings et al., 2017; Shuangzhi Wu, 2017) have incorporated syntactic information provided by the dependency tree of the source sentence. (Marcheggiani et al., 2018) has proposed a model to inject semantic bias into the encoder of NMT model.

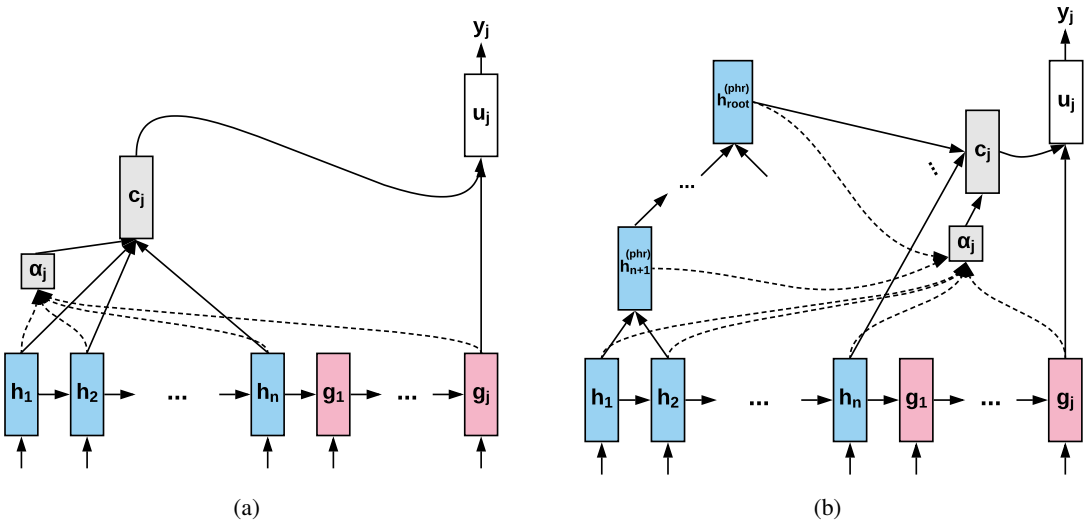
Recently, (Eriguchi et al., 2016; Chen et al., 2017a) have proposed methods to incorporate the hierarchical syntactic constituency information of the source sentence. In addition to the embedding of words, computed using the vanilla sequential encoder, they compute the embeddings of phrases recursively, directed by the top-1 parse tree of the source sentence generated by a parser. Though the results are promising, the top-1 trees are prone to parser error, and furthermore cannot capture semantic ambiguities of the source sentence.

In this paper, we address the aforementioned issues by using exponentially many trees encoded in a forest instead of a single top-1 parse tree. We capture the parser uncertainty by considering many parse trees and their probabilities. The encoding of each source sentence is guided by the forest, and includes the forest nodes whose representations are computed in a bottom-up fashion using our ForestLSTM architecture (§3). Thus, in the encoding stage of this approach, different ways of constructing a phrase

Proceedings of the 27th International Conference on Computational Linguistics, pages 1421–1429
Santa Fe, New Mexico, USA, August 20-26, 2018.

are taken into consideration along with the probability of rules in the corresponding trees. We evaluate our approach on English to Chinese, Farsi and German translation tasks, showing that forests lead to better performance compared to top-1 tree and sequential encoders (§4).

2 Neural Machine Translation



(a) (b)

Figure 1: (a) Attentional SEQ2SEQ model (b) Attentional TREE2SEQ model.

2.1 Attentional Sequence-to-Sequence model

This model is based on the attentional encoder-decoder architecture for SEQ2SEQ transduction. It contains a sequential encoder to read the input sentence and a sequential decoder to generate the output. An example is depicted in Figure 1(a).

Sequential Encoder The source sentence is encoded by a sequential LSTM:

$$\boldsymbol{h}_i = \text{SeqLSTM}(\boldsymbol{h}_{i-1}, \boldsymbol{E}_x[x_i])$$

where $\boldsymbol{E}_x[x_i]$ is the embedding of the word x_i in the embedding table $\boldsymbol{E}_x$ of the source language, and $\boldsymbol{h}_i$ is the context-dependent embedding of x_i.

Sequential Decoder For the generation of each target word j, a dynamic context $\boldsymbol{c}_j$ is produced to summarise the relevant parts of the source sentence. Then, the decoder generates the j-th word as follows:

$$\begin{aligned} \boldsymbol{g}_j &= \tanh(\boldsymbol{W}^{gh}\boldsymbol{g}_{j-1} + \boldsymbol{W}^{gi}\boldsymbol{E}_y[\boldsymbol{y}_{j-1}] + \boldsymbol{W}^{ga}\boldsymbol{c}_j + \boldsymbol{W}^{gu}\boldsymbol{u}_{j-1}) \\ \boldsymbol{u}_j &= \tanh(\boldsymbol{W}^{uc}\boldsymbol{c}_j + \boldsymbol{W}^{ui}\boldsymbol{E}_y[\boldsymbol{y}_{j-1}] + \boldsymbol{g_j}) \\ y_j|\boldsymbol{y}_{<j}, \boldsymbol{x} &\sim \text{softmax}(\boldsymbol{W}^{ou}\boldsymbol{u_j} + \boldsymbol{b}^o) \end{aligned}$$

where $\boldsymbol{E}_y[y_j]$ is the embedding of word y_j looked-up from target language embedding table $\boldsymbol{E}_y$, and $\boldsymbol{W}$ matrices and $\boldsymbol{b}^o$ are the parameters.

The attention mechanism provides relevant information from the source sentence with respect to the current state of the decoder. In each decoding step, the dynamic context is computed as follows:

$$\begin{aligned} a_{ij} &= \boldsymbol{v}^\top \tanh(\boldsymbol{W}^{ae}\boldsymbol{h}_i + \boldsymbol{W}^{ag}\boldsymbol{g}_{j-1}) \\ a_{i'j} &= \boldsymbol{v}^\top \tanh(\boldsymbol{W}^{ae}\boldsymbol{h}^{phr}_{i'} + \boldsymbol{W}^{ag}\boldsymbol{g}_{j-1}) \\ \boldsymbol{\alpha}_j &= \text{softmax}(\boldsymbol{a}_j) \end{aligned}$$

$$\boldsymbol{c}_j = \sum_{i=1}^{n} \alpha_{ji} \boldsymbol{h}_i$$

where a_{ij} and $a_{i'j}$ are the attention scores, and $\boldsymbol{h}_i$ refers to embedding of words. n is the length of the input sentence, and n_p is the number of forest nodes.

2.2 Attentional Tree-to-Sequence model

The majority of NMT models are based on sequential encoding of the source sentence. An exception is the TREE2SEQ model in (Eriguchi et al., 2016), where the encoding of the source sentence is directed by the top-1 parse tree. This model computes the embeddings of phrases in addition to the words, then attend on both words and phrases in the decoder. An example is depicted in Figure 1(b).

Tree Encoder It consists of sequential and recursive parts. The sequential part is the vanilla sequence encoder discussed in Section 2.1, which computes the embeddings of words. Then, the embeddings of phrases are computed using the embeddings of their constituent words in a recursive bottom-up fashion:

$$\boldsymbol{h}_k^{(phr)} = \text{TreeLSTM}(\boldsymbol{h}_k^l, \boldsymbol{h}_k^r).$$

where $\boldsymbol{h}_k^l$ and $\boldsymbol{h}_k^r$ are hidden states of left and right children respectively. This method uses TreeLSTM units (Tai et al., 2015) to calculate the embedding of a parent node using its two two children units as follow:

$$\begin{aligned}
\boldsymbol{i} &= \sigma(\boldsymbol{U}_l^{(i)} \boldsymbol{h}^l + \boldsymbol{U}_r^{(i)} \boldsymbol{h}^r + \boldsymbol{b}^{(i)}) \\
\boldsymbol{f}^l &= \sigma(\boldsymbol{U}_l^{(f_l)} \boldsymbol{h}^l + \boldsymbol{U}_r^{(f_l)} \boldsymbol{h}^r + \boldsymbol{b}^{(f_l)}) \\
\boldsymbol{f}^r &= \sigma(\boldsymbol{U}_l^{(f_r)} \boldsymbol{h}^r + \boldsymbol{U}_r^{(f_r)} \boldsymbol{h}^r + \boldsymbol{b}^{(f_r)}) \\
\boldsymbol{o} &= \sigma(\boldsymbol{U}_l^{(o)} \boldsymbol{h}^l + \boldsymbol{U}_r^{(o)} \boldsymbol{h}^r + \boldsymbol{b}^{(o)}) \\
\tilde{\boldsymbol{c}} &= \tanh(\boldsymbol{U}_l^{(\tilde{c})} \boldsymbol{h}^l + \boldsymbol{U}_r^{(\tilde{c})} \boldsymbol{h}^r + \boldsymbol{b}^{(\tilde{c})}) \\
\boldsymbol{c}^{(phr)} &= \boldsymbol{i} \odot \tilde{\boldsymbol{c}} + \boldsymbol{f}^l \odot \boldsymbol{c}^l + \boldsymbol{f}^r \odot \boldsymbol{c}^r \\
\boldsymbol{h}^{(phr)} &= \boldsymbol{o} \odot \tanh(\boldsymbol{c}^{(phr)})
\end{aligned}$$

where $\boldsymbol{i}$, $\boldsymbol{f}^l$, $\boldsymbol{f}^r$, $\boldsymbol{o}_j$,$\tilde{\boldsymbol{c}}_j$ are the input gate, left and right forget gates, output gate, and a state for updating memory cell; $\boldsymbol{c}^r$ and $\boldsymbol{c}^l$ are memory cells of the right and left units.

Sequential Decoder Eriguchi et al. set the initial state of the decoder by combining the final state of the sequential and tree encoders as follow:

$$\boldsymbol{g}_0 = \text{TreeLSTM}(\boldsymbol{h}_n, \boldsymbol{h}_{root}^{(phr)}),$$

The rest of the decoder is similar to the vanilla attentional decoder discussed in Section 2.1. The difference is that, in this model, the attention mechanism makes use of phrases as well as words. Thus, the dynamic context is computed as follows:

$$\boldsymbol{c}_j = \sum_{i=1}^{n} \alpha_{ji} \boldsymbol{h}_i + \sum_{i'=n+1}^{2n-1} \alpha_{ji'} \boldsymbol{h}_{i'}^{phr}$$

3 Neural Forest-to-Sequence Translation

The TREE2SEQ model uses the top-1 parse tree generated by a parser. Mistakes and uncertainty in parsing eventually affect the performance of the translation. To address these issues, we propose a method to consider exponentially many parse trees along with their corresponding probabilities. It consists of a *forest* encoder to encode a collection of packed parse trees, in order to reduce error propagation due to using only the top-1 parse tree. Our forest encoder computes representations for words and phrases of the source sentence with respect to its parse forest. A sequential decoder, then, generates output words one-by-one from left-to-right by attending to both words and phrases (i.e. forest nodes).

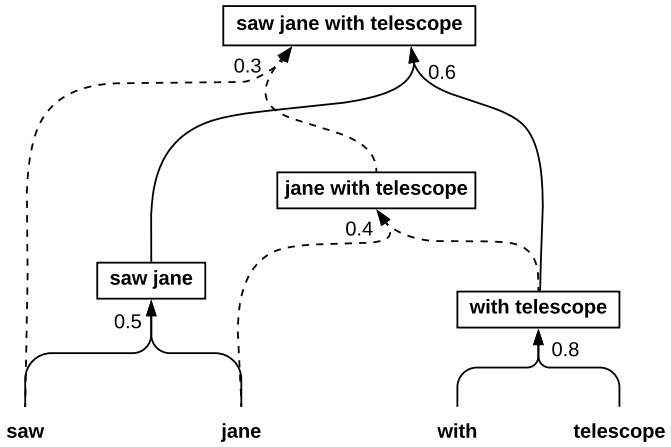


Figure 2: An example of generating a phrase from two different parse trees

3.1 Forest Encoder

The representation of words are computed using sequential encoder (Section 2.1). Then, the embeddings of phrases are computed with respect to the source sentence parse forest in a recursive bottom-up fashion.

Embedding of Phrases. We compute the embedding of the forest nodes (phrases) in a bottom-up fashion. For each hyperedge, we compute the embedding of the head with respect to its tails using a TreeLSTM unit (Tai et al., 2015). In a forest, however, a phrase can be constructed in multiple ways using the incoming hyperedges to a forest node, with different probabilities (see Figure 2). Our ForestLSTM combines the phrase embeddings resulted from these hyperedges, and takes into account their probabilities in order to obtain a unified embedding for the forest node and its corresponding phrase:

$$
\begin{aligned}
\boldsymbol{\gamma}^l &= \tanh\left(\boldsymbol{U}^{\gamma}\sum_{l'=1}^{N}\mathbf{1}_{l\neq l'}\boldsymbol{h}^{l'} + \boldsymbol{W}^{\gamma}\boldsymbol{h}^{l} + \boldsymbol{v}^{\gamma}p^{l} + \boldsymbol{b}^{\gamma}\right)\\
\boldsymbol{f}^l &= \sigma\left(\boldsymbol{U}^{f}\sum_{l'=1}^{N}\mathbf{1}_{l\neq l'}[\boldsymbol{h}^{l'};\boldsymbol{\gamma}^{l'}] + \boldsymbol{W}^{f}[\boldsymbol{h}^{l};\boldsymbol{\gamma}^{l}] + \boldsymbol{b}^{l}\right)\\
\boldsymbol{i} &= \sigma\left(\boldsymbol{U}^{i}\sum_{l=1}^{N}[\boldsymbol{h}^{l};\boldsymbol{\gamma}^{l}] + \boldsymbol{b}^{i}\right)\\
\boldsymbol{o} &= \sigma\left(\boldsymbol{U}^{o}\sum_{l=1}^{N}[\boldsymbol{h}^{l};\boldsymbol{\gamma}^{l}] + \boldsymbol{b}^{o}\right)
\end{aligned}
$$

where N is the number of incoming hyperedges, $\boldsymbol{h}^l$ is the embedding for the head of the l-th incoming hyperedge and p^l is its probability and $\boldsymbol{v}^{\gamma}$ is the learned weight for the probablity. $\boldsymbol{\gamma}^l$ is a probability-sensitive intermediate representation for the l-th incoming hyperedge, which is then used in the computations of the forget gate $\boldsymbol{f}^l$, the input gate $\boldsymbol{i}$, and the output gate $\boldsymbol{o}$. The representation of the phrase $\boldsymbol{h}^{phr}$ is then computed as

$$
\begin{aligned}
\tilde{\boldsymbol{c}} &= \tanh\left(\boldsymbol{U}^{\tilde{c}}\sum_{l=1}^{N}[\boldsymbol{h}^{l};\boldsymbol{\gamma}^{l}] + \boldsymbol{b}^{\tilde{c}}\right)\\
\boldsymbol{c}^{phr} &= \boldsymbol{i}\odot\tilde{\boldsymbol{c}} + \sum_{l=1}^{N}\boldsymbol{f}^{l}\odot\boldsymbol{c}^{l}\\
\boldsymbol{h}^{phr} &= \boldsymbol{o}\odot\tanh(\boldsymbol{c}^{phr})
\end{aligned}
$$

where $\boldsymbol{c}^l$ is the memory cell of the TreeLSTM unit used to compute the representation of the head for the l-th hyperedge from its tail nodes.

3.2 Sequential Decoder

We use a sequential attentional decoder similar to that of the TREE2SEQ model, where the attention mechanism attends to both words and phrases in the forest:

$$\boldsymbol{c}_j = \sum_{i=1}^{n} \alpha_{ji} \boldsymbol{h}_i + \sum_{i'=1+n}^{n_p+n} \alpha_{ji'} \boldsymbol{h}_{i'}^{phr}$$

where n is the length of the input sentence, and n_p is the number of forest nodes.

We initialize the decoder's first state by combining the embeddings of the last word in the source sentence and the root of the forest:

$$\boldsymbol{g}_0 = \text{TreeLSTM}(\boldsymbol{h}_n, \boldsymbol{h}_{root}^{phr}).$$

This provides a summary of phrases and words in the source sentence to the decoder.

3.3 Training

Training is done end-to-end by minimising the cross entropy objective:

$$J(\theta) = \sum_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{F}_{\boldsymbol{x}}) \in D} -\log p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{F}_{\boldsymbol{x}})$$

where D is the set of triples consists of the bilingual training sentences $(\boldsymbol{x}, \boldsymbol{y})$ paired with the parse forests of the source sentences $\boldsymbol{F}_{\boldsymbol{x}}$.

4 Computational Complexity Analysis

We now analyse the computational complexity of inference for SEQ2SEQ, TREE2SEQ and FOREST2SEQ models. We show that, interestingly, our method process exponentially many trees with only a small linear overhead.

Let $|x|$ denotes the length of the source sentence and $O(W_s)$ and $O(W_r)$ are computational complexity of forward-pass for the sequential and Tree/Forest LSTM units, respectively. Having N nodes in the tree/forest, the computational complexity of the encoding phase would be:

$$O(2W_s|x| + W_r N)$$

where the first term shows the computational complexity of a bidirectional sequential encoder to compute the embeddings of words, and the latter one is the time for computing the embeddings of phrases with respect to the corresponding tree/forest.

For generating each word in the target sentence, the attention mechanism performs soft attention on words and phrases of the source sentence. If $O(W_t)$ be the time for updating the decoder state and generating the next target word, for a target sentence with length $|y|$ the decoding phase computational complexity would be:

$$O(W_t|y| + |y|(N + |x|))$$

Hence, the total inference time for a sentence pair is:

$$O(2W_s|x| + W_r N + W_t|y| + |y|(N + x))$$

The difference among the three methods is N. For the SEQ2SEQ model N is 0. For the TREE2SEQ model the number of nodes in the tree is a constant function of the input size: $N = |x| - 1$. Since

Sentence Length	Avg. tree nodes	Avg. forest nodes	Avg. # of trees in forests
<10	7.94	9.77	6.13E+4
10-19	12.3	18.99	2.62E+16
20-29	21.18	41.79	2.76E+22
>30	31	78.72	2.21E+15
all	10.33	14.84	1.41E+20

Table 1: The average number of nodes in trees and forests along with average number of trees in forests for En→Fa bucketed dataset.

we used *pruned* forests obtained from the parser in (Huang, 2008), the number of nodes in the forest is variable. Table 1 shows the average value of N for trees/forests for different source lengths for one of the datasets we used in experiments. As seen, while forests contain exponentially many trees, on average, the number of nodes in parse forests is less than twice the number of nodes in the corresponding top-1 parse trees. It shows that our method considers exponentially many trees instead of top-1 tree using only a small linear overhead.

5 Experiments

5.1 The Setup

Datasets. We make use of three different language pairs: English (En) to Farsi (Fa), Chinese (Ch), and German (De). Our research focus is to tackle NMT issues for bilingually low-resource scenarios and En→Fa is intrinsically a low-resource language. Moreover, we used small datasets for En→Ch and En→De language pairs to simulate low-resource scenarios, where the source and target languages are linguistically divergent and close, respectively. For En→Fa, we use the TEP corpus (Tiedemann, 2009) which is extracted from movie subtitles. It has about 341K sentence pairs, where we split into 337K for training, 2K for development, and 2K for test. For En→Ch, we use BTEC where 'devset1_2' and 'devset_3' are used as the development and test sets, and training consists of 44,016 sentence pairs. For En→De, we use the first 100K sentences of Europarl[1] for training, 'newstest2013' for development, and 'newstest2014' for test.

We lowercase and tokenise the corpora using Moses scripts (Koehn et al., 2007). Sentences longer than 50 words are removed, and words with frequency less than 5 are replaced with <UNK>. Compact forests and trees for source English sentences are obtained from the parser in (Huang, 2008), where the forests are binarised, i.e. hyperedges with more than two tail nodes are converted to multiple hyperedges with two tail nodes. This is to ensure a fair comparison between our model and the TREE2SEQ model (Eriguchi et al., 2016) where they use binary HPSG parse trees. Furthermore, we prune the forests by removing low probability hyperedges, which significantly reduces the size of the forests. In all experiments, we use the development sets for setting the hyper-parameters, and the test sets for evaluation.

Implementation Details. We use Mantis implementation of attentional NMT (Cohn et al., 2016) to develop our code for FOREST2SEQ and TREE2SEQ with DyNet (Neubig et al., 2017). All neural models are trained end-to-end using Stochastic Gradient Descent, where the mini-batch size is set to 128. The maximum training epochs is set to 20, and we use early stopping on the development set as a stopping condition. We generate the translations using greedy decoding. The BLEU score is computed using 'multi-bleu.perl' script in Moses.

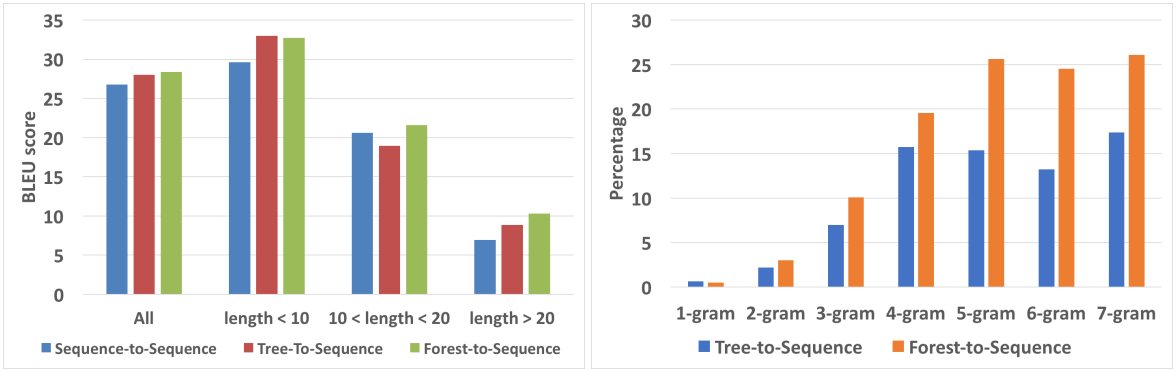
5.2 Results

The perplexity and BLEU scores of different models for all translation tasks are presented in Table 2. In all translation tasks, FOREST2SEQ outperforms TREE2SEQ as it reduces syntactic errors by using forests instead of top-1 parse trees. Our results confirm those in (Eriguchi et al., 2016), and show that using syntactic trees in TREE2SEQ improve the translation quality compared to the vanilla SEQ2SEQ.

[1] http://www.statmt.org/wmt14/translation-task.html

Method	H	En → De		En → Ch		En → Fa	
		Perplexity	BLEU	Perplexity	BLEU	Perplexity	BLEU
SEQ2SEQ	256	33.07	11.98	6.48	25.43	19.21	10.17
(Luong et al., 2015)	512	32.61	12.21	6.12	26.77	18.4	10.93
TREE2SEQ	256	30.13	13	6.17	26.85	17.94	11.32
(Eriguchi et al., 2016)	512	31.86	13.05	5.71	28	16.28	11.71
our	256	30.83	**13.54**	6.16	27.08	17.62	11.91
FOREST2SEQ	512	**29.25**	13.43	**5.49**	**28.39**	**16.66**	**12.38**

Table 2: Comparison of the methods together with different hidden dimension size (H) for all datasets.



(a) (b)

Figure 3: (a) BLEU scores for bucketed En→Ch dataset. (b) Percentage of more correct n-grams generated by the TREE2SEQ and FOREST2SEQ models compared to SEQ2SEQ model for En→Ch dataset.

Comparing BLEU scores of the forest-based and tree-based models, the largest increase is observed for En→Fa. This can be attributed to the syntactic divergence between English and Farsi (SVO vs SOV) as well as the reduction of significant errors in the top-1 parser trees for this translation task, resulted from the domain mismatch between the parser's training data (i.e. Penn Tree Bank) and the English source (i.e. informal movie subtitles).

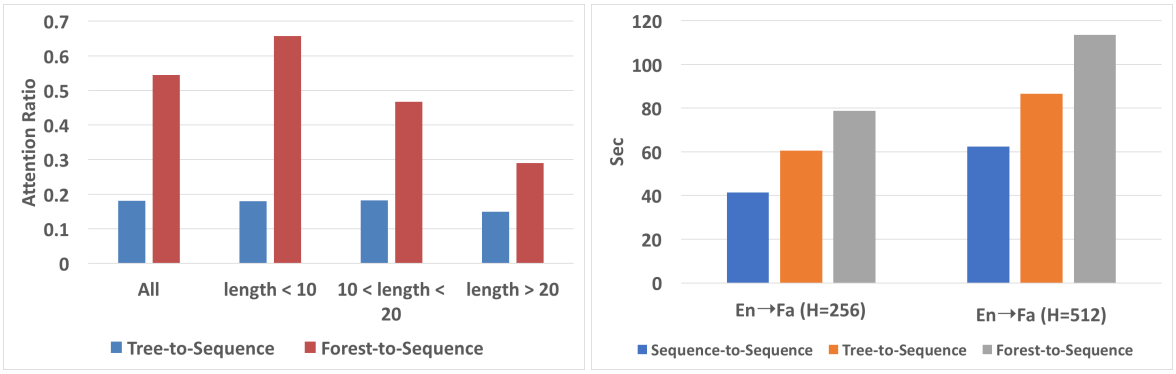
5.3 Analysis

The effect of sequential part in the forest encoder The forest encoder consists of sequential and recursive parts, where the former is the vanilla sequence encoder. We investigate the effect of the sequential part in the proposed forest encoder. Table 3 shows the results on the test set of En → Fa dataset. The results show that the sequential part in the forest encoder lead to improvement in results. Speculatively, the sequential part helps the forest encoder by providing the context-aware embeddings for words which then be used to construct phrase embeddings.

For which sentence lengths the forest-based model is more helpful? To investigate the effect of source sentence length, we divide En→Ch dataset into three buckets with respect to the length of source sentences. Figure 3(a) depicts the BLEU scores resulted from the models for different buckets. The FOREST2SEQ model performs better than vanilla SEQ2SEQ model on all buckets. Interestingly, while TREE2SEQ model has a lower BLEU compared to SEQ2SEQ model for the sentences whose lengths are between 10 and 20, the FOREST2SEQ model has a better BLEU possibly due to reducing parsing errors.

	Perplexity	BLEU
Forest encoder W/ sequential part	16.66	12.38
Forest encoder W/O sequential part	17.48	11.97

Table 3: The Effect of the sequential part in our proposed forest encoder on the En → Fa test set.



(a) (b)

Figure 4: (a) Attention ratios for En→Fa bucketed dataset.(b) Inference time (seconds) required for the test set of En→Fa dataset using trained models.

The forest-based model results in correct larger n-grams in translations We further analyse the effect of the incorporated syntactic knowledge on improving the number of generated gold n-grams in translations. For each sentence, we compute the number of n-grams in the generated translations which are common with those in the gold translation. Then, after aggregating the results over the entire test set, we compute the percentage of additional gold n-grams generated by syntax aware models, i.e. TREE2SEQ and FOREST2SEQ, compared to the SEQ2SEQ model. The results are depicted in Figure 3(b). Generating correct high order n-grams is hard, and results show that incorporating syntax is beneficial. As n increases, the FOREST2SEQ model performs significantly better than the TREE2SEQ model in generating gold n-grams, possibly due to better reorderings between the source and target.

How much attention the forest-based and tree-based models pay to the syntactic information? We next analyse the extent by which the syntactic information is used by the TREE2SEQ and FOREST2SEQ models. We compute the ratio of attention on phrases to words for both of the syntax-aware models in En→Fa translation task, where the source and target languages are highly syntactically divergent. For each triple in the test set, we calculate the sum of attention on words and phrases during decoding. Then, the ratio of attention on phrases to words are computed and is averaged for all triples. Figure 4(a) shows these attention ratios for bucketed En→Fa dataset. It shows that for all sentence lengths, the FOREST2SEQ model provides richer phrase embeddings compared to the TREE2SEQ model, leading to a more usage of the syntactic information.

Investigating the effect of using trees/forests on inference time We measured the inference time required for the test set of EN→FA dataset using the trained models. The results are depicted in Figure 4(b). As seen, while using one parse tree increases the inference time linearly, interestingly, our FOREST2SEQ model considers exponentially many trees also with a small linear overhead.

6 Conclusion

We have proposed a forest-to-sequence attentional NMT model, which uses a packed forest instead of the top-1 parse tree in the encoder.

Using a forest of parse trees, our method efficiently considers exponentially many constituency trees in order to take into account parser uncertainties and errors. Experimental results show our method is superior to the attentional tree-to-sequence model, which is more prone to the parsing errors.

Acknowledgments

This work was supported by the Multi-modal Australian ScienceS Imaging and Visualisation Environment (MASSIVE) (www.massive.org.au), and by the Australian Research Council through DP160102686. The first author was partly supported by CSIRO's Data61.

References

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. *arXiv preprint arXiv:1704.04675*.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017a. Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1936–1945.

Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2017b. Syntax-directed attention for neural machine translation. *arXiv preprint arXiv:1711.04231*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. pages 1724–1734.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 823–833. Association for Computational Linguistics.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *ACL*, pages 586–594.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. *arXiv preprint arXiv:1705.01020*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Diego Marcheggiani, Joost Bastings, and Ivan Titov. 2018. Exploiting semantics in neural machine translation with graph convolutional networks. *arXiv preprint arXiv:1804.08313*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Dongdong Zhang Shuangzhi Wu, Ming Zhou. 2017. Improved neural machine translation with source syntax. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4179–4185.

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

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Jörg Tiedemann. 2009. News from OPUS-A collection of multilingual parallel corpora with tools and interfaces. In *Recent advances in natural language processing*, volume 5, pages 237–248.

Ensure the Correctness of the Summary: Incorporate Entailment Knowledge into Abstractive Sentence Summarization

Haoran Li[1,2], **Junnan Zhu**[1,2], **Jiajun Zhang**[1,2] and **Chengqing Zong**[1,2,3]
[1] National Laboratory of Pattern Recognition, CASIA, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] CAS Center for Excellence in Brain Science and Intelligence Technology
{haoran.li, junnan.zhu, jjzhang, cqzong}@nlpr.ia.ac.cn

Abstract

In this paper, we investigate the sentence summarization task that produces a summary from a source sentence. Neural sequence-to-sequence models have gained considerable success for this task, while most existing approaches only focus on improving word overlap between the generated summary and the reference, which ignore the correctness, i.e., the summary should not contain error messages with respect to the source sentence. We argue that correctness is an essential requirement for summarization systems. Considering a correct summary is semantically entailed by the source sentence, we incorporate entailment knowledge into abstractive summarization models. We propose an entailment-aware encoder under multi-task framework (i.e., summarization generation and entailment recognition) and an entailment-aware decoder by entailment Reward Augmented Maximum Likelihood (RAML) training. Experimental results demonstrate that our models significantly outperform baselines from the aspects of informativeness and correctness.

1 Introduction

Sentence summarization is a well-studied task that creates a condensed version of a long source sentence. Sequence-to-sequence (seq2seq) model that encodes a source sequence into a latent representation and outputs another sequence is the dominating framework for sentence summarization (Rush et al., 2015; Chopra et al., 2016; Takase et al., 2016; Zhou et al., 2017; Li et al., 2017b; Li et al., 2018). Despite substantial improvements on this task, most of the existing researches typically aim to improve word overlap between the generated summary and the references, which is measured by n-gram matching metrics (e.g., ROUGE (Lin, 2004)). Hence, it cannot guarantee the semantic correctness of the summary as a whole. Therefore, in some cases, the summary giving high matching scores may contain critical error messages, which makes the summary fail to capture the correct information with respect to the source sentence. Previous study shows that about 30% of the summaries generated by state-of-the-art seq2seq system are subject to this problem (Cao et al., 2017). Here is an example (the digits are replaced by "#"):

Source sentence: franch won the gold medal at women 's epee team event of the fie #### world championships by beating china ##-## .
Reference: france beats china for women 's epee team gold
State-of-the-art seq2seq model: canada wins women 's epee team event

For the example shown above, the seq2seq system produces a fluent summary which contains an obvious mistake. The true winner of the "women 's epee team event" is "france", while the summarization model wrongly generates "canada", which is probably due to similar word representations for country names. Though the word overlap between the generated summary and the reference is considerable, leading to high ROUGE scores, the summary is invalid.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1430–1441
Santa Fe, New Mexico, USA, August 20-26, 2018.

We argue that correctness is an essential requirement for summarization systems, while most existing systems ignore it. Generally, a correct summary is semantically entailed by the source sentence, thus we believe entailment[1] (Bos and Markert, 2005) knowledge is beneficial to avoid producing contradictory or unrelated information in the summary.

To incorporate entailment knowledge into abstractive summarization models, we propose in this work an entailment-aware encoder and an entailment-aware decoder. We share the encoder of the summarization generation system with the entailment recognition system, so that the encoder can grasp both the gist of the source sentence and be aware of entailment relationships. Furthermore, we propose an entailment Reward Augmented Maximum Likelihood (RAML) (Norouzi et al., 2016) training that encourages the decoder of the summarization system to produce summary entailed by the source. Experimental results demonstrate that our models significantly outperform some solid baselines on objective evaluation for informativeness and manual evaluation for correctness. Further analysis suggests that our summarization model is aware of entailment knowledge.

Our main contributions are as follows:

- We incorporate entailment knowledge into summarization models to avoid producing unrelated information with respect to the source sentence.
- We propose an entailment-aware encoder by jointly modeling summarization generation and entailment recognition.
- We introduce an entailment-aware decoder via entailment RAML training.

2 Background: Seq2seq Learning

In this section, we describe the basic seq2seq learning framework. Given a dataset of input-output pairs, $\mathcal{D} \triangleq (\mathbf{x}_i, \mathbf{y}_i^*)_{i=1}^N$, the seq2seq model maximizes the conditional probability of a target sequence $\mathbf{y}^*$: $p(\mathbf{y}^*|\mathbf{x})$. Recurrent Neural Networks (RNN) encoder (Cho et al., 2014) reads and converts a variable-length input sequence $\mathbf{x}$ into a context representation c as follows:

$$h_t = f_{\text{enc}}(\mathrm{x_t}, h_{t-1}) \tag{1}$$

$$c_t = f_c(h_1, \cdots, h_t) \tag{2}$$

where $h_t \in \mathbb{R}^n$ is a hidden state at time t, and c_t is a context vector generated from the sequence of the hidden states. f_{enc} and f_{c} are nonlinear activation functions.

The decoder generates word $\mathrm{y_t}$ given the context vector c_t and the previously generated words:

$$p(\mathrm{y}_t|\{\mathrm{y}_1, \cdots, \mathrm{y}_{t-1}\}, c_t) = f_{\text{dec}}(\mathrm{y}_{t-1}, s_t, c_t) \tag{3}$$

where s_t is the hidden state of the decoder and f_{dec} is a nonlinear activation function. The maximum likelihood (ML) framework tries to minimize negative log-likelihood of the parameters as follows:

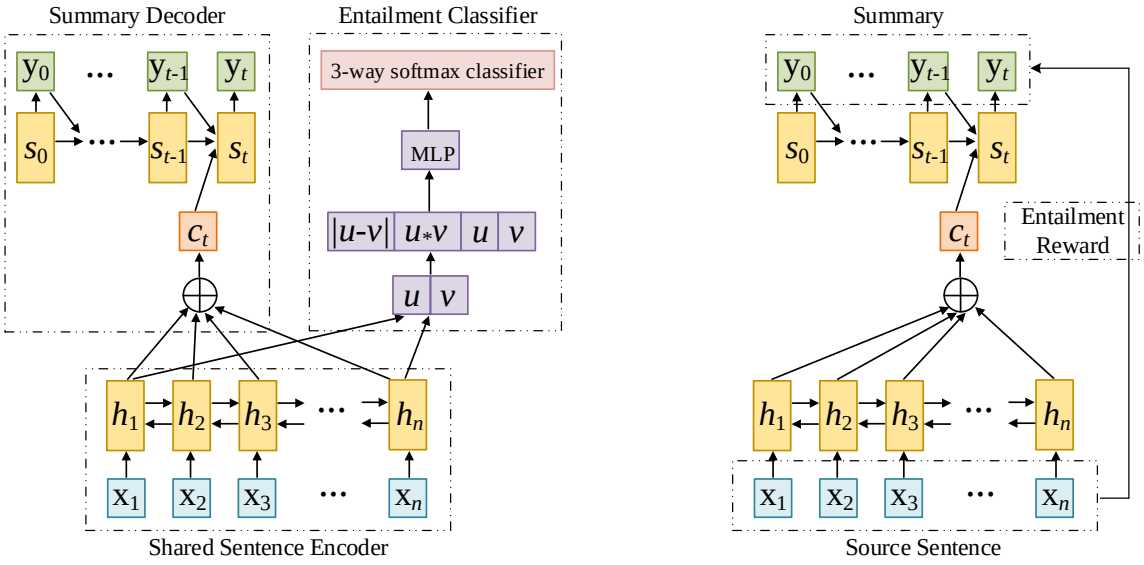
$$\mathcal{L}_{\text{ML}}(\mathcal{D}) = \sum_{(\mathbf{x},\mathbf{y}^*)\in\mathcal{D}} -\log p(\mathbf{y}^*|\mathbf{x}) \tag{4}$$

3 Our Proposed Model

3.1 Overview

In order to avoid generating unrelated summary with respect to the source sentence, we propose two strategies to incorporate entailment knowledge into seq2seq summarization model. We first introduce an entailment-aware encoder using multi-task learning for summarization generation and entailment recognition. Then, we introduce an entailment-aware decoder by entailment RAML training.

[1]Entailment is a kind of relationships between two sentences for natural language inference. Sentence A entailing sentence B means A can infer B. Other relationships include contradiction and neutral. A correct summary should be inferred by the source sentence. Thus, we argue that entailment is a useful criterion for the correctness of the summary.



(a) Entailment-aware Encoder

(b) Entailment-aware Decoder

Figure 1: The framework of our model. Entailment-aware encoder is learned by jointly training summarization generation (left part of (a), which is a seq2seq model) and entailment recognition (right part of (a), in which sentence pair in the entailment recognition corpus are encoded as u and v). Entailment-aware decoder is learned by entailment RAML training, in which the summary will be rewarded if it is entailed by the source sentence.

3.2 Entailment-aware Encoder

In this section, we propose a multi-task learning for abstractive summarization by sharing the encoder with the task of entailment recognition. By doing so, we can learn an entailment-aware encoder for sentence summarization task. In this way, we can improve the correctness aspect of the summarization model, while maintaining the salient information extraction aspects. Note that the training data for summarization and entailment task is from summarization and entailment corpus, respectively.

3.2.1 Shared Sentence Encoder

Given a source sentence $\mathbf{x} = (\mathrm{x}_1, \cdots, \mathrm{x}_n)$, we employ a bidirectional LSTM (BiLSTM) to build its hidden representation $(h_1, \cdots, h_n)$.

The BiLSTM encodes source sentence forwardly and backwardly to generate two sequences of the hidden states: $(\overrightarrow{h}_1, \cdots, \overrightarrow{h}_n)$ and $(\overleftarrow{h}_1, \cdots, \overleftarrow{h}_n)$, respectively, where:

$$\overrightarrow{h}_i = \mathrm{LSTM}(\mathrm{x}_i, \overrightarrow{h}_{i-1}) \tag{5}$$

$$\overleftarrow{h}_i = \mathrm{LSTM}(\mathrm{x}_i, \overleftarrow{h}_{i+1}) \tag{6}$$

The final sentence representation h_i is the concatenation of the forward and backward vectors: $h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i]$.

3.2.2 Attention-based Summarization Decoder

At each time step t, the state of the decoder s_t is calculated as follows:

$$s_t = \mathrm{LSTM}(s_{t-1}, \mathrm{y}_{t-1}, c_t) \tag{7}$$

$$s_0 = \tanh(\mathbf{W}_h[\overrightarrow{h}_n; \overleftarrow{h}_1]) \tag{8}$$

We compute the context vector c_t as a weighted sum of the source annotations as follows:

$$c_t = \sum_{i=1}^{N} \alpha_{t,i} h_i \tag{9}$$

where each vector is weighted by the attention weight $\alpha_{t,i}$, as calculated in Equations 10 and 11:

$$e_{t,i} = v_c^T \tanh(\mathbf{W}_s s_t + \mathbf{W}_e h_i) \tag{10}$$

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{j=1}^{N} \exp(e_{t,j})} \tag{11}$$

The probability for the next target word y_t is computed using hidden state s_t and the previously emitted word y_{t-1} as follows:

$$p(\mathrm{y}_t|\{\mathrm{y}_1, \cdots, \mathrm{y}_{t-1}\}) \propto \exp(\mathbf{L}_e \tanh(\mathbf{L}_s s_t + \mathbf{L}_y \mathrm{y}_{t-1})) \tag{12}$$

where $\mathbf{W}_h$, $\mathbf{W}_s$, $\mathbf{W}_e$, $\mathbf{L}_e$, $\mathbf{L}_s$ and $\mathbf{L}_y$ are model parameters. The summarization model is trained by minimizing negative log-likelihood loss as in Equation 4.

3.2.3 Matching-based Entailment Inference Model

To infer entailment relation, input sentence pairs from the entailment recognition corpus are fed into sentence encoder to obtain hidden representation $(h_1^u, \cdots, h_n^u)$ and $(h_1^v, \cdots, h_n^v)$, respectively. Then, the sentence pairs are encoded as vectors $u = [\overrightarrow{h}_n^u; \overleftarrow{h}_1^u]$ and $v = [\overrightarrow{h}_n^v; \overleftarrow{h}_1^v]$, respectively. Next, the absolute difference and the element-wise product for the tuple $[u, v]$ are concatenated with the original vectors u and v (Mou et al., 2016) as follows:

$$q = [|u - v|; u * v; u; v] \tag{13}$$

We then feed q into a 3-layer multilayer perceptron (MLP) classifier. The 3-class softmax output layer is on top of MLP. The entailment recognition model is trained by minimizing cross-entropy loss.

3.2.4 Multi-Task Learning (MTL)

In our multi-task setup, we share the encoder parameters of both the tasks, as shown in Figure 1(a). Traditional MTL considers equal contribution for all tasks. In our model, two tasks are significantly different. The summary generation task is much more complicated than entailment recognition, leading to different learning difficulties and convergence rates. Therefore, summarization generation is regarded as the main task and entailment recognition as the auxiliary task, and our goal is to optimize the main task with assistance of auxiliary task. To this end, we optimize the two loss functions alternatively during training. Let α be the number of mini-batches of training for entailment recognition after 100 mini-batches of training for summarization generation (Pasunuru et al., 2017). We adopt $\alpha = 10$ and performance with different α is discussed in Section 6.6.3.

3.3 Entailment-aware Decoder

In order to encourage the decoder of the summarization system to produce summary entailed by the source sentence, we apply an entailment-aware decoder by entailment RAML training (Norouzi et al., 2016).

3.3.1 Reward Augmented Maximum Likelihood (RAML) Training

RAML provides a computationally efficient approach to optimize task-specific reward (loss) directly. In our work, we apply RAML to incorporate entailment-based reward into our summarization model, as shown in Figure 1(b).

The RAML objective function is defined as follows:

$$\mathcal{L}_{\mathrm{RAML}} = \sum_{(\mathbf{x},\mathbf{y}^*)\in\mathcal{D}} \left\{ -\sum_{\mathbf{y}\in\mathcal{Y}} q\left(\mathbf{y}|\mathbf{x}, \mathbf{y}^*; \tau\right) \log p(\mathbf{y}|\mathbf{x}) \right\} \tag{14}$$

$$q\left(\mathbf{y}|\mathbf{x}, \mathbf{y}^*; \tau\right) = \frac{1}{Z(\mathbf{x}, \mathbf{y}^*, \tau)} \exp\{r(\mathbf{x}, \mathbf{y}, \mathbf{y}^*)/\tau\} \tag{15}$$

$$Z(\mathbf{x}, \mathbf{y}^*, \tau) = \sum_{\mathbf{y}\in\mathcal{Y}} \exp\{r(\mathbf{x}, \mathbf{y}, \mathbf{y}^*)/\tau\} \tag{16}$$

where $\mathcal{Y}$ is the set of possible model outputs. $r(\mathbf{x}, \mathbf{y}, \mathbf{y}^*)$ denotes the reward function and τ is the regularization parameter.

3.3.2 Optimizing by Entailment-based Sampling

We can express the gradient of $\mathcal{L}_{\mathrm{RAML}}$ in terms of an expectation over samples from $q(\mathbf{y}|\mathbf{x}, \mathbf{y}^*; \tau)$:

$$\mathcal{L}_{\mathrm{RAML}} = E_{q(\mathbf{y}|\mathbf{x},\mathbf{y}^*;\tau)}\left[-\nabla \log p(\mathbf{y}|\mathbf{x})\right] \tag{17}$$

RAML training adds a sampling step over typical ML objective. Instead of optimizing ML on training samples, given training input $(\mathbf{x}, \mathbf{y}^*)$, RAML training first samples an output $\mathbf{y}$ proportionally to the reward. Then, RAML optimizes log-likelihood on such sample given the corresponding input. Thus, we need to sample auxiliary outputs from the exponentiated payoff distribution, $q(\mathbf{y}|\mathbf{x}, \mathbf{y}^*; \tau)$. In this work, we first use reward values defined by negative Hamming distance and then re-weight the reward based on entailment reward $s(\mathbf{x}, \mathbf{y}, \mathbf{y}^*)$. Particularly, given a sentence $\mathbf{y}^*$ of length ℓ, we count the number of sentences within an edit distance d, where $d \in \{0, \ldots, 2\ell\}$. Then, we weight the counts by $\exp\{-d/\tau\}$ and perform normalization. Finally, we apply importance sampling by the weight $\exp\{(s(\mathbf{x}, \mathbf{y}, \mathbf{y}^*) + d)/\tau\}$ and perform normalization, where the proposal distribution is Hamming distance sampling[2].

We define entailment reward $s(\mathbf{x}, \mathbf{y}, \mathbf{y}^*)$ as follows:

$$s(\mathbf{x}, \mathbf{y}, \mathbf{y}^*) = \min\{e(\mathbf{x}, \mathbf{y}), e(\mathbf{x}, \mathbf{y}^*)\} \tag{18}$$

where $e(\mathbf{x}, \mathbf{y})$ denotes entailment score for sentence pairs $(\mathbf{x}, \mathbf{y})$. Our goal is to maximize the entailment reward of the summary towards the reference, given the source sentence. Here we adopt the model of Parikh et al. (2016) trained on the MultiNLI corpus[3] (Williams et al., 2017) to obtain $e(\mathbf{x}, \mathbf{y})$.

4 Related work

Text summarization methods can be categorized into extraction-based methods (Erkan and Radev, 2004; Wan et al., 2007; Cheng and Lapata, 2016; Zhang et al., 2016; Nallapati et al., 2017; Li et al., 2017a) and abstraction-based methods. Rush et al. (2015) first apply the seq2seq model to abstractive sentence summarization. They propose an attentive CNN encoder and a neural network language model (Bengio et al., 2003) decoder. Chopra et al. (2016) use RNN as the decoder and achieve better performance. Nallapati et al. (2016) further replace the encoder with an RNN, forming a full RNN seq2seq model. Gu et al. (2016) and Zeng et al. (2016) incorporate a copying mechanism into seq2seq learning and Gulcehre et al. (2016) propose a switch gate to control whether to copy from the source or generate a word by the decoder. Copying mechanism intends to replicate segments in the source to the target, which cannot guarantee the correctness of the summary as a whole. Ma et al. (2017) focus on improving the semantic relevance between source and summary by encouraging high similarity of their representation. Zhou et al. (2017) employ a selective encoding model to control the information flow from encoder to decoder. Li et al. (2017b) apply a deep recurrent generative decoder to seq2seq framework. Cao et al. (2017) solve the problem of fake facts in a summary. They use Open Information Extraction to extract fact descriptions in the source sentence and propose the dual-attention seq2seq framework to force the generation conditioned on both source sentence and the fact descriptions. To the best of our knowledge, our work is the first to directly explore the correctness of summary without any preprocessing.

Some previous work (Mehdad et al., 2013; Gupta et al., 2014) has used textual entailment recognition to reduce redundancy for extractive summarization task. Our work is partially inspired by the models of Pasunuru et al. (2017) with following differences: Pasunuru et al. (2017) model the entailment task as the seq2seq generation problem and enforce sharing of the same decoder between summarization and entailment. However, the entailment task is more reasonable to be considered as a multi-label classification problem rather than a generation problem. We thus design a multi-task learning framework in which the summarization generation task shares the same encoder with the entailment recognition task.

[2]We adopt the implement at https://github.com/pcyin/pytorch_nmt

[3]Multi-Genre Natural Language Inference (MultiNLI) is one of the largest corpora available for the task of natural language inference. It consists of sentences from ten different sources of text, which can be used for cross-genre domain adaptation.

5 Dataset

We conduct experiments on English Gigaword and DUC 2004 datasets.

Gigaword Corpus. We use the annotated Gigaword corpus provided by Rush et al. (2015). The dataset has about 3.8 million training pairs. Following Zhou et al. (2017), we use 8, 000 pairs as validation set and the test samples provided by Rush et al. (2015) and Zhou et al. (2017) as our test sets.

DUC 2004 Corpus. DUC-2004 corpus for tasks 1 & 2 (Over et al., 2007) consists of 500 documents. Each document in these datasets has four human annotated summaries. For experiments on this corpus, we directly use the model trained on the Gigaword to test on the DUC 2004 corpus.

6 Experiment

6.1 Experimental Settings

Word embedding size is set to 300 and LSTM hidden state size is set to 512. We use the full source and target vocabularies collected from the training data, which have 119, 505 and 68, 885 words, respectively. Adam (Kingma and Ba, 2014) optimizer is applied with the learning rate of 0.001, momentum parameters $\beta_1 = 0.9$ and $\beta_1 = 0.999$, and $\epsilon = 10^{-8}$. For RAML, $\tau = 0.85$. The mini-batch size is set to 64. We test the model performance (ROUGE-2 F1 score) on validation set for every 2,000 batches. We halve the learning rate if the ROUGE-2 F1 score drops for twelve consecutive tests on validation set. We also apply gradient clipping (Pascanu et al., 2012) with range $[-5, 5]$ during training. Our model with entailment-aware encode requires less than 300,000 training iterations to train with early stopping (Prechelt, 1998). To speed up the training for our RAML model, we continue the RAML training based on the pre-trained model with ML training with the current decayed learning rate. At test time, we use beam search with beam size 10 to generate the summary. We report ROUGE F1 score including ROUGE-1, ROUGE-2 and ROUGE-L for Gigaword corpus and ROUGE recall score for DUC 2004 corpus.

6.2 Comparative Methods

We compare a set of sentence summarization baselines.

ABS. Rush et al. (2015) first apply the seq2seq model to abstractive sentence summarization. They use an attentive CNN encoder and neural network language model decoder to summarize sentence.

ABS+. Rush et al. (2015) further tune ABS model on DUC 2003 dataset, then test on DUC 2004 test set.

CAs2s. Chopra et al. (2016) extend the ABS model with a convolutional encoder and RNN decoder, which performs better than the ABS model.

Feats2s. Nallapati et al. (2016) use a full RNN seq2seq model and add some lexical features to enhance the encoder, including POS, NER tags and TF-IDF values.

Luong-NMT. Luong et al. (2015) propose a neural machine translation model with two-layer LSTMs for the encoder-decoder.

Seq2seq. This is a standard seq2seq model with attention mechanism.

Seq2seq + MTL. This is our proposed model with entailment-aware encoder, which applies a multi-task learning (MTL) framework to seq2seq model.

Seq2seq + MTL (Share decoder). Pasunuru et al. (2017) propose a multi-task learning (MTL) framework in which the decoder is shared for summarization generation and entailment generation task.

Seq2seq + ERAML. This is our proposed model with entailment-aware decoder, which conducts an Entailment Reward Augmented Maximum Likelihood (ERAML) training framework.

Seq2seq + ROUGE-2 RAML. We apply ROUGE-2 RAML training for seq2seq model.

Seq2seq + RL. We implement Reinforcement Learning (RL) models (policy gradient) with reward metrics of **Entailment** and **ROUGE-2**.

Seq2seq + selective. Zhou et al. (2017) employ a selective encoding model to control the information flow from encoder to decoder. To verify the generalization of our entailment-based strategies, we adopt selective encoding mechanism to our seq2seq model and apply MTL and RAML to **Seq2seq + selective** model, which is denoted as the **Seq2seq + selective + MTL + RAML** model.

Model	ROUGE-1	ROUGE-2	ROUGE-L
ABS (Rush et al., 2015)	37.41	15.87	34.70
Seq2seq (Zhou et al., 2017)	43.76	22.28	41.14
Seq2seq + MTL	45.11	23.87	42.50
Seq2seq + MTL (Share decoder) (Pasunuru et al., 2017)	44.69	22.91	42.04
Seq2seq + ERAML	44.71	23.74	42.11
Seq2seq + ROUGE-2 RAML	43.75	23.63	41.31
Seq2seq + RL (Entailment)	44.39	23.31	41.86
Seq2seq + RL (ROUGE-2)	43.55	22.97	41.01
Seq2seq + MTL + ERAML	45.36	24.12	42.74
Seq2seq + selective	45.58	24.02	42.88
Seq2seq + selective + MTL + ERAML	**46.28**	**24.60**	**43.47**

Table 1: Experimental results (%) on the English Gigaword test set of Zhou et al. (2017). Our models perform significantly better than baselines by the 95% confidence interval measured by the official ROUGE script.

6.3 Experimental Results: Gigaword Corpus

In Table 1, we report the ROUGE F1 score of our model and the baseline methods on the English Gigaword test set provided by Zhou et al. (2017). Our entailment-aware models outperform all baseline models by a large margin. Our final model, **Seq2seq + selective + MTL + ERAML**, achieves the best results, which improves 2.52 (%) ROUGE-1, 2.32 ROUGE-2 and 2.33 ROUGE-L over seq2seq model. Our seq2seq model with entailment-aware encoder (**Seq2seq + MTL**) surpasses the state-of-the-art seq2seq model of 1.35 ROUGE-1, 1.59 ROUGE-2, 1.36 ROUGE-L, and entailment-aware decoder (**Seq2seq + ERAML**) gains improvement of 0.95 ROUGE-1, 1.46 ROUGE-2, 0.97 ROUGE-L. Compared to the another MTL model via sharing decoder for entailment generation task (**Seq2seq + MTL (Share decoder)**), our MTL model (**Seq2seq + MTL**) has obvious ROUGE score gains. The **Seq2seq + ROUGE-2 RAML** model also shows promising performance, especially for ROUGE-2 score. RAML has a clear advantage over RL. In principle, RL samples from the model distribution, which slows down training and several tricks are needed to get better estimates of the gradient (Ranzato et al., 2015). The comparison to the model of **Seq2seq + selective** shows that our entailment-aware strategies are also useful for seq2seq model with selective encoding framework, which demonstrates the good generalization of our method. The results on English Gigaword test set provided by Rush et al. (2015) are shown in Table 2. Our model performs better than the previous works.

6.4 Experimental Results: DUC 2004 Test Corpus

We evaluate our model with the ROUGE recall score. The reference summaries of the DUC 2004 test set are fixed to 75 bytes and we set the maximum length of the summary to 18 following Zhou et al. (2017). In Table 2, experimental results also show our **Seq2seq + selective + MTL + ERAML** model achieves significant improvements over baseline models, surpassing Feats2s (Nallapati et al., 2016) by 0.98% ROUGE-1, 0.78% ROUGE-2 and 0.65% ROUGE-L without fine-tuning on DUC data.

6.5 Manual Evaluation

Next, we conduct a manual evaluation to inspect the correctness of the generated summaries. We randomly select 500 samples in the test set and employ five postgraduates to classify the generated summaries as correct (i.e., not contain wrong information) or not. As shown in Table 3, 60.6% of the summaries generated by seq2seq model are correct, and it rises to 69.4% and 74.2% for our model with selective encoding and entailment-aware strategies, respectively, which indicates the effectiveness of our model to generate a correct summary.

	Test set of Rush et al. (2015)			DUC 2004 test set		
Model	RG-1	RG-2	RG-L	RG-1	RG-2	RG-L
ABS (Rush et al., 2015)	29.55	11.32	26.42	26.55	7.06	22.05
ABS+ (Rush et al., 2015)	29.76	11.88	26.96	28.18	8.49	23.81
Feats2s (Nallapati et al., 2016)	32.67	15.59	30.64	28.35	9.46	24.59
CAs2s (Chopra et al., 2016)	33.78	15.97	31.15	28.97	8.26	24.06
Luong-NMT (Luong et al., 2015)	33.10	14.45	30.71	28.55	8.79	24.43
Seq2seq (Zhou et al., 2017)	34.04	15.95	31.68	28.13	9.25	24.76
Seq2seq + MTL	34.69	16.68	32.32	28.61	9.67	24.86
Seq2seq + ERAML	34.34	16.59	32.26	28.37	9.61	24.81
Seq2seq + MTL + ERAML	34.88	16.86	32.51	28.89	9.87	24.94
Seq2seq + selective	35.01	16.71	32.88	29.01	9.89	25.01
Seq2seq + selective + MTL + ERAML	**35.33**	**17.27**	**33.19**	**29.33**	**10.24**	**25.24**

Table 2: Experimental results (%) on the English Gigaword test set of Rush et al. (2015) and DUC 2004 test set. Our models perform significantly better than baseline models by the 95% confidence interval measured by the official ROUGE (RG) script.

Model	Correctness(%)
Seq2seq	60.6
Seq2seq + selective	69.4
Seq2seq + selective + MTL + ERAML	74.2

Table 3: Manual evaluation for correctness.

6.6 Further Analysis

To further investigate the effectiveness of our model, we perform analysis on the entailment score improvement, the abstraction degree of our model and the impact for entailment recognition task.

6.6.1 Does our summarization model learn entailment knowledge?

The motivation of our work is to encourage summarization model to generate summaries that are entailed by the source sentences. To verify this goal, we investigate the entailment score for source-summary pairs for different models. For the test set of Zhou et al. (2017), the average entailment score for the reference is 0.72, while for the basic seq2seq model, the entailment score is only 0.46. When we adopt entailment-based strategies, the entailment score rises to 0.63 for seq2seq model. Note that the entailment score is 0.57 for seq2seq model with selective encoding, and we believe that the selective mechanism can filter out secondary information in the input, which will reduce the possibility to generate irrelevant information. Entailment-aware selective model achieves a high entailment reward of 0.71. In part at least, we can conclude that our model has successfully learned entailment knowledge.

6.6.2 Is it less abstractive for our model?

We have shown that our entailment-aware model can generate correct summaries more frequently (Section 6.5). Intuitively, it is more likely to be correct if summary segments are directly extracted from the source. Thus, readers may wonder whether our model is less abstractive. Figure 2 shows that the seq2seq model produces more novel words (i.e., words that do not appear in the article) than our model, indicating a lower degree of abstraction for our model. However, when we exclude all the words not in the reference (these words may lead to wrong information), our model generates more novel words, suggesting that our model provides a compromise solution for informativeness and correctness. Thus, our model can generate summary with fewer mistakes.

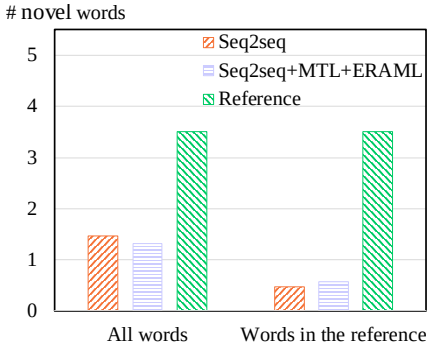


Figure 2: Average count of novel words (words that do not appear in the article). Seq2seq model generates more novel words, but less words are in the reference compared to our model.

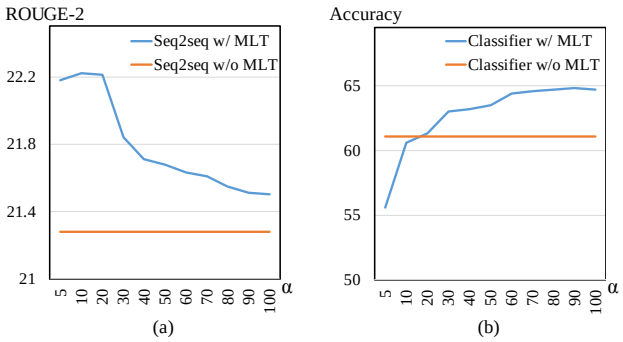


Figure 3: The performance of (a) summarization generation on Gigaword validation set and (b) entailment recognition on SNLI (Bowman et al., 2015) validation set with different task batch switches (α).

1	Source	brazilian stocks rose , led by consumer stocks , after the government said it would n't impose restraints on consumer credit
	Reference	brazil stocks rise after government rules out credit restraints
	Seq2seq	brazil stocks rise on consumer credit concerns
	Seq2seq + MTL + ERAML	brazil stocks rise after government says it wo n't impose credit restrictions
2	Source	authorities have denied neo-nazi groups permission to stage a demonstration next week in the austrian capital , where skinheads planned to gather on the ##th anniversary of nazi germany 's surrender ending world war ii in europe .
	Reference	vienna police deny neo-nazi groups permission to stage downtown
	Seq2seq	neo-nazi group to stage demonstration in vienna
	Seq2seq + MTL + ERAML	austrian authorities deny neo-nazi demonstration
3	Source	queens taxi driver osman chowdhury said he never thought of keeping the ## diamond rings he found inside a suitcase left in his trunk by a dallas woman who had given him a ##-cent tip , local media reported on thursday .
	Reference	u.s. taxi driver says he never thinks of keeping diamond rings left in trunk
	Seq2seq	queens taxi driver denies keeping ## diamond rings in trunk
	Seq2seq + MTL + ERAML	## diamond rings found in suitcase

Table 4: Cases Study.

6.6.3 Could the entailment recognition also be improved?

Multi-task learning (MTL) involves sharing parameters between related tasks, whereby each task can benefit from extra information of other tasks in the training process. In this section, we explore whether the entailment recognition can benefit from summarization generation task. Figure 3 shows that our summarization model with MTL outperforms basic seq2seq model. As α increases, the accuracy of entailment recognition improves and finally exceeds that of the model without MTL, which reveals the advantage of MTL framework.

7 Case Study

We illustrate the examples of outputs in Table 4. As shown in the table, seq2seq model generates summaries that are not relevant to the source sentence, while the output of our model obtains higher entailment scores than those of seq2seq model. For the first example, seq2seq model regards the reason for "brazil stocks rise" as "consumer credit concerns", while in fact, "consumer" is not worried because "government said it would n't impose restraints on consumer credit". By contrast, since our model incorporates entailment knowledge, the true reason is captured and the output of our model is related to the source sentence. A similar problem happens in example 2, and seq2seq model generates a summary that is contradictory to the source. The "demonstration" is "denied" by the "authorities", while seq2seq model confirms the "demonstration". In Example 3, neither seq2seq nor our model performs satisfactorily. Seq2seq model again misunderstands the meaning of the source and outputs summary containing wrong information. Though the summary generated by our model is entailed by the source, the summary fails to produce an integrated sentence and misses the key points of the source, such as the object of the event, "queens taxi driver". A mixed reward, i.e., combining entailment and ROUGE-2, may address this

issue. We leave it for our future work.

8 Conclusion

This paper investigates the correctness problem in abstractive summarization. We propose an entailment-aware encoder by jointly learning summarization generation and entailment recognition. We present an entailment-aware decoder by entailment reward augmented maximum likelihood training. By enriching the encoder and decoder with entailment information, our model makes the summary more likely be entailed by the source input. Experimental results on Gigaword and DUC 2004 datasets demonstrate that our model achieves significant improvements over strong baselines on both informativeness and correctness. Our code is available online[4].

Acknowledgements

The research work descried in this paper has been supported by the National Key Research and Development Program of China under Grant No. 2017YFC0820700 and the Natural Science Foundation of China under Grant No. 61333018.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, pages 1137–1155.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2017. Faithful to the original: Fact aware neural abstractive summarization. *arXiv:1711.04434*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *journal of artificial intelligence research*, 22:457–479.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1631–1640.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 140–149.

Anand Gupta, Manpreet Kaur, Shachar Mirkin, Adarsh Singh, and Aseem Goyal. 2014. Text summarization through entailment-based minimum vertex cover. In *Proceedings of the Third Joint Conference on Lexical and Computational Semantics (*SEM 2014)*, pages 75–80.

[4]https://github.com/bubei/entail_sum

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980*.

Haoran Li, Junnan Zhu, Cong Ma, Jiajun Zhang, and Chengqing Zong. 2017a. Multi-modal summarization for asynchronous collection of text, image, audio and video. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1092–1102. Association for Computational Linguistics.

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017b. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2091–2100.

Haoran Li, Junnan Zhu, Tianshang Liu, Jiajun Zhang, and Chengqing Zong. 2018. Multi-modal sentence summarization with modality attention and image filtering. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. AAAI Press.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Shuming Ma, Xu Sun, Jingjing Xu, Houfeng Wang, Wenjie Li, and Qi Su. 2017. Improving semantic relevance for sequence-to-sequence learning of chinese social media text summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 635–640.

Yashar Mehdad, Giuseppe Carenini, Frank Tompa, and N. G. Raymond T. 2013. Abstractive meeting summarization with entailment and fusion. In *European Workshop on Natural Language Generation*, pages 136–146.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 130–136.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, pages 3075–3081.

Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. 2016. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, pages 1723–1731.

Paul Over, Hoa Dang, and Donna Harman. 2007. Duc in context. In *Information Processing & Management*, pages 1506–1520.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

Ramakanth Pasunuru, Han Guo, and Mohit Bansal. 2017. Towards improving abstractive summarization via entailment generation. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 27–32.

Lutz Prechelt. 1998. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767.

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

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1054–1059.

Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. 2007. Manifold-ranking based topic-focused multi-document summarization. In *IJCAI*, volume 7, pages 2903–2908.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv:1704.05426*.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient summarization with read-again and copy mechanism. *arXiv:1611.03382*.

Jiajun Zhang, Yu Zhou, and Chengqing Zong. 2016. Abstractive cross-language summarization via translation model enhanced predicate argument structure fusing. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(10):1842–1853.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1095–1104.

Extracting Parallel Sentences with Bidirectional Recurrent Neural Networks to Improve Machine Translation

Francis Grégoire
Université de Montréal
francis.gregoire@rd.mila.quebec

Philippe Langlais
Université de Montréal
felipe@iro.umontreal.ca

Abstract

Parallel sentence extraction is a task addressing the data sparsity problem found in multilingual natural language processing applications. We propose a bidirectional recurrent neural network based approach to extract parallel sentences from collections of multilingual texts. Our experiments with noisy parallel corpora show that we can achieve promising results against a competitive baseline by removing the need of specific feature engineering or additional external resources. To justify the utility of our approach, we extract sentence pairs from Wikipedia articles to train machine translation systems and show significant improvements in translation performance.

1 Introduction

Parallel corpora are a prerequisite for many multilingual natural language processing (NLP) applications. Unfortunately, parallel data is available for a relatively small number of language pairs and for few specific domains. With the increasing amount of comparable corpora on the World Wide Web, a potential solution to alleviate the parallel data sparsity issue is to extract parallel sentences from this more abundant source of information. Therefore, the objective of parallel sentence extraction is to extract parallel sentences from such comparable corpora to increase the quantity of parallel data and the range of the covered domains. Comparable corpora can be defined as collections of topic-aligned but non-sentence-aligned multilingual texts.

Recent advances in deep learning architectures with recurrent neural networks (RNNs) have shown that they can successfully learn complex mappings from variable-length sequences to continuous vector representations. While numerous natural language processing tasks have successfully applied those models, ranging from handwriting generation (Graves, 2013) to machine comprehension (Hermann et al., 2015), most of the multilingual efforts have been devoted to machine translation (Sutskever et al., 2014; Cho et al., 2014), although more research interests have been recently devoted to multilingual semantic textual similarity.[1]

In this paper, we propose a parallel sentence extraction system to measure the translational equivalence between sentences in two languages. Our system is based on bidirectional recurrent neural networks that can learn sentence representations in a shared vector space by explicitly maximizing the similarity between parallel sentences. In contrast to previous approaches, by leveraging these continuous vector representation of sentences we remove the need to rely on multiple models and specific feature engineering. Experiments on noisy parallel corpora show that our approach outperforms a competitive baseline. To justify the utility of our approach, we add the sentence pairs extracted from Wikipedia articles to a parallel corpus to train machine translation systems and show improvements in translation performance. Our experimental results lead us to believe that our system is a promising tool to create new aligned multilingual resources.

[1] http://alt.qcri.org/semeval2017/task2/

Proceedings of the 27th International Conference on Computational Linguistics, pages 1442–1453
Santa Fe, New Mexico, USA, August 20-26, 2018.

2 Related work

Traditional systems developed to extract parallel sentences from comparable corpora typically rely on multiples models or metadata from articles structure. Munteanu and Marcu (2005) present a complete system based on statistical word alignment models and a maximum entropy classifier to automatically extract sentence pairs from newspaper collections. The authors evaluate the quality of the extracted sentences by showing that they improve the performance of statistical machine translation (SMT) systems. The overall structure of this approach is still considered state-of-the-art. Adafre and Rijke (2006) is the first work to observe that articles from Wikipedia are likely to generate parallel corpora useful for machine translation. These two approaches are extended by Smith et al. (2010) where the authors introduce several new features by exploiting the structure and metadata of Wikipedia article pairs. They use their augmented set of features in a conditional random field and obtain state-of-the-art results on a small set of 20 manually annotated Wikipedia article pairs. The work of Abdul-Rauf and Schwenk (2009) proposes a different approach, in which they use an SMT system built from a small parallel corpus. Instead of using a classifier, they translate the source language side of a comparable corpus to find candidate sentences on the target language side. They determine if a translated source sentence and a candidate target sentence are parallel by measuring the word error rate and the translation error rate. A simplification of these systems is proposed in (Azpeitia et al., 2017), where the similarity between two sentences is defined as the average of the Jaccard similarity coefficients obtained between sentence token sets and lexical translations determined by IBM models (Brown et al., 1993). Their approach also combine various features, such as longest common prefix matching, numbers and capitalized truecased tokens.

Chu et al. (2016) train a neural machine translation (NMT) system to generate sentence representations with the encoder, which are then used as additional features to the system of Munteanu and Marcu (2005). Similarly, Cristina et al. (2017) study sentence representations obtained from the encoder of an NMT system to detect new parallel sentence pairs. By comparing cosine similarities, they show that they can distinguish parallel and non-parallel sentences. A different approach exploiting continuous vector representations is proposed in Grover and Mitra (2017). After learning word representations using the bilingual word embeddings model of (Luong et al., 2015), they use a convolutional neural network on a similarity matrix to classify if a pair of sentences is aligned or not. These approaches are different from ours, where we use a single end-to-end model to estimate the conditional probability distribution that two sentences are parallel.

3 Approach

3.1 Negative sampling

As positive examples, we use a parallel corpus C consisting of n parallel sentence pairs $\left\{(\mathbf{s}_k^S, \mathbf{s}_k^T)\right\}_{k=1}^n$, where S and T denote the source sentences and target sentences. Since we want a model that learns differentiable vector representations to distinguish parallel from non-parallel sentences, we use negative sampling to generate negative examples. Therefore, we randomly sample m non-parallel target sentences for every positive source sentence, such that $(\mathbf{s}_k^S, \mathbf{s}_j^T)$ for $j \neq k$. This process is repeated at the beginning of each training epoch to allow the model to learn on a larger number of non-parallel sentence pairs. Hence, our training set contains $n(1+m)$ triples $(\mathbf{s}_i^S, \mathbf{s}_i^T, y_i)$, where $\mathbf{s}_i^S = (w_{i,1}^S, \ldots, w_{i,N}^S)$ is a source sentence of N tokens, $\mathbf{s}_i^T = (w_{i,1}^T, \ldots, w_{i,M}^T)$ is a target sentence of M tokens, and y_i is the label representing the translation relationship between $\mathbf{s}_i^S$ and $\mathbf{s}_i^T$, so that $y_i = 1$ if $(\mathbf{s}_i^S, \mathbf{s}_i^T) \in C$ and $y_i = 0$ otherwise.

The advantage of negative sampling is its simplicity. However, relying only on randomness to select negative sentence pairs makes most of the examples very non-parallel and easy to classify. An interesting way to generate negative examples would be to replace only a segment of a sentence. Similarly, we could replace a sentence from a parallel pair with another sentence that is close to it in the vector space. This would make the problem harder, but potentially could make the classifier stronger. We leave such investigations as future work.

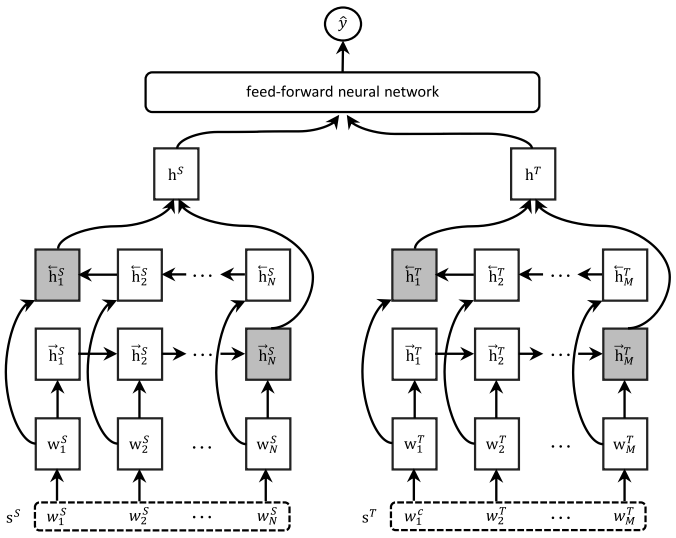


Figure 1: Graphical illustration of the siamese bidirectional recurrent neural networks. The final recurrent state of the forward and backward RNNs are concatenated and then fed into a feed-forward neural network culminating in a sigmoid output layer.

3.2 Model

Our idea is to use RNNs to learn cross-language semantics between sentence pairs to estimate the probability that they are translations of each other. The proposed model architecture consists of a bidirectional RNNs (BiRNN) (Schuster and Paliwal, 1997) sentence encoder with recurrent activation functions such as long short-term memory units (LSTM) (Hochreiter and Schmidhuber, 1997) or gated recurrent units (GRU) (Cho et al., 2014). Since we want vector representations in a shared vector space we use a siamese network (Bromley et al., 1993) with tied weights, which is equivalent to using a single BiRNN to encode a pair of sentences into two continuous vector representations. The source and target sentence representations are then fed into a feed-forward neural network with a sigmoid output layer which calculates the probability that they are parallel. The architecture of our approach is illustrated in Figure 1.

To avoid repetition and for clarity, we only define equations of the BiRNN encoding the source sentence. For the target sentence, simply substitute S for T. At each time step t, the token in the i-th sentence, $w^S_{i,t}$, defined by its integer index k in the vocabulary V^S, is represented as a one-hot vector, $\mathbf{x}^S_k \in \{0,1\}^{|V|}$. This one-hot vector is multiplied with an embedding matrix, $\mathbf{E}^S \in \mathbb{R}^{|V^S| \times d_e}$, to get a continuous vector representation of this token, $\mathbf{w}^S_{i,t} \in \mathbb{R}^{d_e}$, which serves as an input for the forward and backward recurrent states of the BiRNN encoder, $\overrightarrow{\mathbf{h}}^S_{i,t}$ and $\overleftarrow{\mathbf{h}}^S_{i,t}$. The forward RNN reads the variable-length sentence and updates its recurrent state from the first token until the last one to create a fixed-size continuous vector representation of the sentence, $\mathbf{h}^S_{i,N} \in \mathbb{R}^{d_h}$. The backward RNN processes the sentence in reverse. We use the concatenation of the last recurrent state in both directions as a final representation $\mathbf{h}^S_i = [\overrightarrow{\mathbf{h}}^S_{i,N} ; \overleftarrow{\mathbf{h}}^S_{i,1}]$.[2] The steps we described to encode a sentence are defined as:

$$\mathbf{w}^S_{i,t} = \mathbf{E}^{S^\top} \mathbf{w}^S_k, \tag{1}$$

$$\overrightarrow{\mathbf{h}}^S_{i,t} = \phi(\overrightarrow{\mathbf{h}}^S_{i,t-1}, \mathbf{w}^S_{i,t}), \tag{2}$$

$$\overleftarrow{\mathbf{h}}^S_{i,t} = \phi(\overleftarrow{\mathbf{h}}^S_{i,t+1}, \mathbf{w}^S_{i,t}), \tag{3}$$

where $\phi(\cdot)$ is an LSTM or GRU.

After both source and target sentences have been encoded, we capture their matching information by using their element-wise product and absolute element-wise difference. We estimate the conditional

[2] We considered combining the recurrent states with mean pooling and max pooling to obtain a fixed-size vector representation, but obtained inferior performance.

probability that the sentences are parallel by feeding the matching vectors into a feed-forward neural network with a sigmoid output layer:

$$\mathbf{h}_i^{(1)} = \mathbf{h}_i^S \odot \mathbf{h}_i^T, \tag{4}$$

$$\mathbf{h}_i^{(2)} = |\mathbf{h}_i^S - \mathbf{h}_i^T|, \tag{5}$$

$$\mathbf{h}_i = \tanh(\mathbf{W}^{(1)}\mathbf{h}_i^{(1)} + \mathbf{W}^{(2)}\mathbf{h}_i^{(2)} + \mathbf{b}), \tag{6}$$

$$p(y_i = 1|\mathbf{h}_i) = \sigma(\mathbf{v}\mathbf{h}_i + b), \tag{7}$$

where $\sigma(\cdot)$ is the sigmoid function, $\mathbf{W}^{(1)} \in \mathbb{R}^{d_f \times d_h}$, $\mathbf{W}^{(2)} \in \mathbb{R}^{d_f \times d_h}$, $\mathbf{v} \in \mathbb{R}^{d_f}$, $\mathbf{b} \in \mathbb{R}^{d_f}$ and b are model parameters. The value d_f is the size of the hidden layer in the feed-forward neural network.

The model is trained by minimizing the cross entropy of our labeled sentence pairs:

$$\mathcal{L} = -\sum_{i=1}^{n(1+m)} \Big(y_i \log \sigma(\mathbf{v}\mathbf{h}_i + b) + (1 - y_i) \log(1 - \sigma(\mathbf{v}\mathbf{h}_i + b)) \Big). \tag{8}$$

For prediction, a sentence pair is classified as parallel if its probability is greater than or equal to a decision threshold ρ that we need to fix:

$$\hat{y}_i = \begin{cases} 1 & \text{if } p(y_i = 1|\mathbf{h}_i) \geq \rho, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

The two embedding matrices $\mathbf{E}^S$ and $\mathbf{E}^T$ are parameters of the model that we must learn. As the size of the vocabulary and the dimension d_e increase, the number of parameters can become considerably expensive to estimate. For that reason, it is common to initialize the embedding matrices using word embeddings pre-trained on a large collection of texts.[3]

4 Experiments and Results

To assess the effectiveness of our approach, we compare it in multiple settings. In Section 4.1, we measure its capacity to extract parallel sentences found in parallel corpora in which we inserted non-parallel sentences. We extract sentence pairs from real comparable corpora in Section 4.2 and validate their utility by measuring their impact on SMT and NMT systems.

4.1 System comparison

4.1.1 Data

To create our training and test sets we use the WMT'15 English–French datasets.[4] The positive examples of our training set consist of 500,000 parallel sentence pairs randomly selected from the Europarl corpus (Koehn, 2005). To generate negative examples, at the beginning of each training epoch we sample m non-parallel sentences per positive example, as described in Section 3.1. The vocabulary size is 69,381 for English and 84,182 for French.

The most reliable way to create test sets to compare different approaches would be to have professional translators manually annotate parallel sentences from comparable corpora. However, this option is expensive and impractical. Therefore, it is common practice to compare parallel sentence extraction systems on artificially-created noisy parallel data by inserting non-parallel sentences into a parallel corpus. Thus, to create our test sets we use parallel sentences from the newstest2012 and Europarl corpora and insert artificial noise by substituting target sentences with non-parallel target sentences from their respective held-out dataset. We sample 1,000 parallel sentences from the newstest2012 corpus and create ten test sets with a noise ratio $r \in \{0\%, 10\%, \ldots, 80\%, 90\%\}$, where r is the ratio of artificial

[3] We pre-trained bilingual word embeddings as proposed by (Gouws et al., 2015) and (Smith et al., 2017). While we observed faster learning and a small improvement in the performance of our model, we prefer to learn the embedding matrices from scratch instead of relying on additional external resources.

[4] http://www.statmt.org/wmt15/translation-task.html

non-parallel sentences. The same process is repeated with the Europarl corpus to create three test sets with $r \in \{0\%, 50\%, 90\%\}$. Each final test set consists of 1,000,000 sentence pairs generated from the Cartesian product between the sentences in both languages. For example, 400 out of the 1,000 sentence pairs are parallel if $r = 60\%$, in which only 0.04% of the 1,000,000 sentence pairs generated from the Cartesian product are truly parallel.

We tokenize all datasets with the scripts from Moses (Koehn et al., 2007).[5] The maximum sentence length is set to 80 tokens. Each out-of-vocabulary word is mapped to a special UNK token.

4.1.2 Evaluation metrics

For evaluating the performance of our models, a sentence pair predicted as parallel is correct if it is present in the set of parallel sentences of the test set. Precision is the proportion of truly parallel sentence pairs among all extracted sentence pairs. Recall is the proportion of truly parallel extracted sentence pairs among all parallel sentence pairs in the test set. The F_1 score is the harmonic mean of precision and recall.

4.1.3 Baseline

For comparison, we use a parallel sentence extraction system developed in-house based on the works of (Munteanu and Marcu, 2005) and (Smith et al., 2010). The system consists of a candidate sentence pair filtering process and three models; two word alignment models and a maximum entropy classifier. The word alignment models are trained on both language directions using our training set of 500,000 parallel sentence pairs. To train the classifier, we select another 100,000 parallel sentence pairs from the held-out Europarl dataset. To generate negative examples, we use 100,000 non-parallel sentence pairs that have successfully passed the candidate sentence pair selection process.[6]

Candidate sentence pair selection A sentence pair filtering process is used to select a fixed number of negative sentence pairs to train the maximum entropy classifier. The objective is to select similar non-parallel sentence pairs to make the classifier more robust to noise. It is also used during prediction to filter out the unlikely sentence pairs of the Cartesian product. Since most of these sentence pairs are not parallel, the filtering process significantly reduces the number of sentence pairs to evaluate. The filtering process consists of a two-step procedure. First, it verifies that the ratio of the lengths between two sentences is not greater than 2.[7] It then uses a word-overlap filter to check for both sentences that at least 50% of their words have a translation in the other sentence. Every pair that does not fulfill these two conditions is discarded. The filtering process is only applied to the baseline model.

Word alignment models The translation and alignment tables are estimated using the HMM alignment model of (Vogel et al., 1996). These probability tables are required to calculate the value of many alignment features used in the classifier.

Maximum entropy classifier The classifier relies on word-level alignment features between two sentences, such as the number of connected words, top three largest fertilities, length of the longest connected substring, log probability of the alignments, and also general features, such as the lengths of the sentences, length difference and the percentage of words on each side that have a translation on the other side. For each sentence pair, a total of 31 features must be calculated and it is classified as parallel if the classifier outputs a probability score greater than or equal to a decision threshold ρ which needs to be fixed.

4.1.4 Training settings

The models of our approach are implemented in TensorFlow (Abadi et al., 2016). We use a BiRNN with a single layer in each direction with 512-dimensional word embeddings and 512-dimensional recurrent states. We use LSTM as recurrent activation functions. The hidden layer of the feed-forward neural network has 256 hidden units. To train our models, we use Adam (Kingma and Ba, 2014) with a learning rate of 0.0002 and a minibatch of 128 examples. Models are trained for a total of 15 epochs. To avoid

[5] https://github.com/moses-smt/mosesdecoder

[6] During our experiments, we did not observe any significant gain by using more than 100,000 parallel sentence pairs.

[7] Most parallel sentence extraction systems use a length ratio value equals to 2.

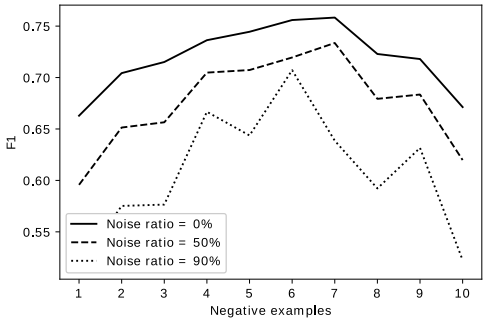


Figure 2: F_1 scores of BiRNN as we increase the number of negative examples m per parallel sentence in the training set. The models are evaluated on newstest2012 with noise ratios of 0%, 50% and 90%.

exploding gradients, we apply gradient clipping such that the norm of all gradients is no larger than 5 (Pascanu et al., 2013). We apply dropout to prevent overfitting with a probability of 0.2 and 0.3 for the non-recurrent input and output connections respectively (Zaremba et al., 2014).

4.1.5 Number of negative examples

We evaluate the performance of our approach as we increase the number of negative examples m per parallel sentence pair in our training set. We trained a total of 10 models with $m \in \{1, \ldots, 10\}$, such that with $m = 1$ and $m = 10$ a model is respectively trained on 1,000,000 and 5,500,000 sentence pairs per epoch, with a positive to negative sentence pairs ratio of 50% and 9%. The more the value of m increases, the more the training set becomes unbalanced. We know that extracting parallel sentences from comparable corpora in practice is an unbalanced classification task in which non-parallel sentences represent the majority class. Although an unbalanced training set is not desired since a classifier trained on such data will typically tend to predict the majority class and have a poor precision, the overall impact on the performance of BiRNN is not clear. Figure 2 shows the F_1 scores of BiRNN with respect to the value of m evaluated on newstest2012 with noise ratios of 0%, 50% and 90%. Each reported F_1 score is the one calculated at the decision threshold value maximizing the area under the precision-recall curve. For $m \geq 7$, we observe a deterioration in the performance of our systems evaluated on test sets with noise ratios of 0% and 50%, whereas $m = 6$ is the best performing model on the test set with a noise ratio of 90%. We see that having a balanced training set with $m = 1$ is not the optimal solution for our approach. On the contrary, having an unbalanced training set improves its performance. From these observations, in the following experiments we train our models with a value of m fixed at 6.[8]

4.1.6 Evaluation on noisy parallel corpora

In this experiment, we compare our approach to the baseline system described in Section 4.1.3. Table 1 shows the precision, recall and F_1 scores for the two systems evaluated on the newstest2012 and Europarl test sets with noise ratios of 0%, 50% and 90%. We see that BiRNN is able to constantly outperform the results obtained with the baseline by a significant margin. By using newstest2012 as out-of-domain test sets, our approach gets an absolute improvement in the F_1 score of 10.99%, 13.03% and 23.53%. The precision-recall curves of the two systems evaluated on test sets with noise ratios of 0% and 90% are shown in Figure 3. These curves illustrate the consistency of our approach, whereas the performance of the baseline is greatly impacted when the number of non-parallel sentences in the test set is high.

When BIRNN is evaluated on Europarl, it obtains F_1 scores over 95% on the three test sets. On the other hand, given that the underlying models of the baseline system are trained on Europarl data, it is surprising to observe again a significant deterioration in its performance when the number of non-parallel sentences increases. In Figure 4, we compare the precision, recall and F_1 scores as the noise ratio r applied to our newstest2012 test set increases. We observe that it becomes harder to identify

[8]It takes about 1.5 hours to train an epoch with the model settings described in 4.1.4 using a single GTX 1080 Ti GPU.

Noise	Model	newstest2012				Europarl			
		P (%)	**R (%)**	**F_1 (%)**	ρ	**P (%)**	**R (%)**	**F_1 (%)**	ρ
0%	BiRNN	**83.72**	**68.90**	**75.79**	0.99	**99.26**	**93.50**	**96.29**	0.99
	Baseline	73.88	57.70	64.80	0.93	87.72	84.30	85.98	0.98
50%	BiRNN	**79.95**	**65.40**	**71.95**	0.99	**98.32**	**93.60**	**95.90**	0.99
	Baseline	73.43	49.20	58.92	0.98	80.19	82.60	81.38	0.99
90%	BiRNN	**79.01**	**64.00**	**70.72**	0.99	**97.94**	**95.00**	**96.45**	0.99
	Baseline	53.85	42.00	47.19	0.99	63.89	69.00	66.35	0.99

Table 1: Precision (P), recall (R) and F_1 scores where the decision threshold ρ maximizes the area under the precision-recall curve of the test sets with noise ratios of 0%, 50% and 90%.

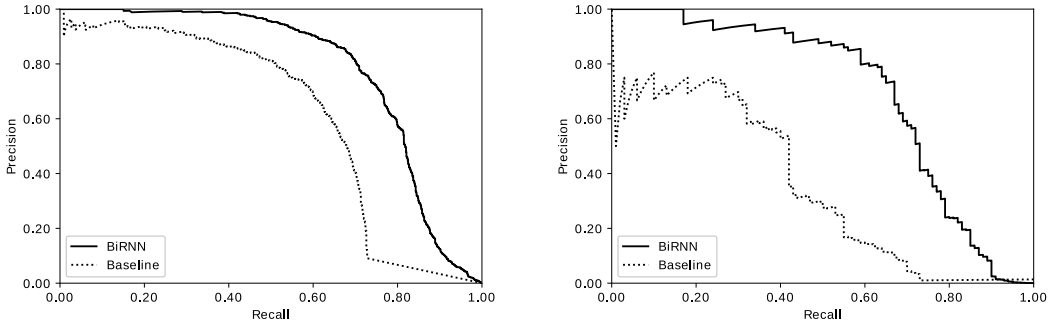


Figure 3: Precision-recall curves of the systems evaluated on newstest2012 with noise ratios of 0% (left) and 90% (right).

parallel sentences as the number of non-parallel sentences increases in the test set. However, we see that our neural network based approach obtains better performances by a significant margin. In contrast to the baseline, the performance of our method stays relatively stable and starts to degrade at very high noise ratios. At that level of noise, we believe our test set is more representative to texts found in real comparable corpora.

In this experiment, all our evaluation metrics (BiRNN and Baseline) are reported at the value ρ which maximizes the area under a precision-recall curve. These performances are therefore upper bound results. The decision threshold has a direct impact on the quality and the quantity of the extracted sentence pairs. Through our experiments, we observed that the optimal value ρ of BiRNN was constantly around a value of 0.99, while the one of the baseline system varied considerably from one test set to another. Thus, the stability of our approach is preferable in practice. We believe that the high optimal decision threshold value $\rho = 0.99$ is caused by the fact that we use a sigmoid output layer with highly unbalanced training and test sets.

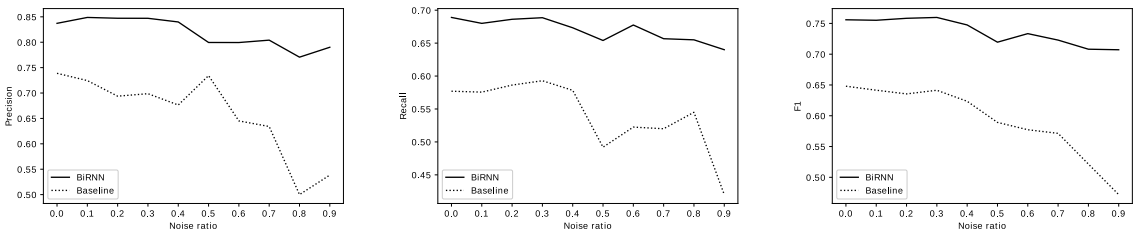


Figure 4: Precision (left), recall (middle) and F_1 (right) scores of the systems evaluated on newstest2012 as the noise ratio r increases.

4.2 Machine Translation Comparison

4.2.1 Data

As training sets, we use the English–French Europarl corpus and sentence pairs extracted by the two parallel sentence extraction systems. The comparable corpora used to extract parallel sentences are 919,000 pairs of English–French articles obtained from Wikipedia dumps.[9] The newstest2012 corpus is our validation set and we evaluate the machine translation systems on the newstest2013 corpus. All datasets are tokenized using the scripts from Moses. The maximum length of each sentence is set to 80 tokens. For the NMT systems, the vocabulary size for both languages is limited to the 150,000 most frequent words of their corresponding training set. All unknown words are replaced by the UNK token.

4.2.2 Evaluation metric

To evaluate the translation performance of our systems we use the BLEU score (Papineni et al., 2002) using the multi-bleu script from Moses.

4.2.3 Training settings

The SMT systems are phrase-based systems (Koehn et al., 2003) that are trained with Moses. We use the GIZA++ implementation (Och and Ney, 2003)[10] to train word alignment models in both directions. The phrases and lexical reordering are extracted using the default values of Moses. The language models are 5-gram models learned using the KenLM toolkit (Heafield, 2011)[11] on the monolingual parts of the same parallel corpus used for training the translation models. The parameters are optimized on the newstest2012 corpus.

To train the NMT systems, we use the PyTorch implementation of OpenNMT (Klein et al., 2017).[12] The NMT systems are one layer BiLSTMs with an attention mechanism (Bahdanau et al., 2014). The dimensions of the word embeddings and recurrent states for the encoder and decoder are all set to 256. The systems are trained for 10 epochs with minibatch of 64 sentence pairs using SGD with an initial learning rate of 1.0 and linear decay.[13] The norm of the gradient is clipped such that it is not greater than 5.

4.2.4 Evaluation on machine translation systems

For each of the SMT and NMT approaches, we trained 14 machine translation systems. The first two systems,[14] which serve as reference systems, are trained with 500,000 and 2,000,000 parallel sentences from the Europarl corpus, respectively. The system trained with 2,000,000 parallel sentences is used to compare the benefits of the sentence pairs extracted from Wikipedia against a large training corpus that we easily have at hand. For the 12 remaining systems, we sort the sentence pairs extracted by an extraction system in descending order according to the probability scores and append the top $\{250000, 500000, \ldots, 1250000, 1500000\}$ sentence pairs to the training set containing 500,000 parallel sentences from the Europarl corpus.

Table 2 shows the BLEU scores obtained by the machine translation systems we trained for the SMT and NMT approaches. The figures in parentheses show the absolute gains in BLEU score compared to the reference systems trained with 500,000 parallel sentences from the Europarl corpus.[15] We see that adding the subsets of sentence pairs extracted by both extraction systems leads to significant gains compared to the reference systems. Adding 1,500,000 sentence pairs extracted by BiRNN improves the BLEU score of the SMT and NMT systems by 3.71 and 9.47, respectively. We observe that translation systems trained on 1,000,000 sentence pairs have better BLEU scores than the reference systems trained on 2,000,000 parallel sentences from Europarl, showing that parallel sentence extraction allows translation systems to

[9] https://dumps.wikimedia.org/

[10] http://www.statmt.org/moses/giza/GIZA++.html

[11] https://github.com/kpu/kenlm

[12] https://github.com/OpenNMT/OpenNMT-py

[13] We could have trained NMT systems for more epochs to obtain better BLEU scores, but our objective is not to compare SMT with NMT systems.

[14] That is, the two first systems for each approach and not from the total of 28 machine translation systems.

[15] Which is the first line of Table 2.

Data	Model	BLEU		Pairs
		SMT	NMT	
Europarl		21.47	17.63	500k
		23.18	25.36	2M
+Top250k	BiRNN	**23.11 (+1.64)**	**24.44 (+6.81)**	750k
	Baseline	23.09 (+1.62)	24.22 (+6.59)	750k
+Top500k	BiRNN	**24.12 (+2.65)**	**25.93 (+8.30)**	1M
	Baseline	23.96 (+2.49)	25.82 (+8.19)	1M
+Top750k	BiRNN	**24.53 (+3.06)**	**26.39 (+8.76)**	1.25M
	Baseline	24.44 (+2.97)	26.19 (+8.56)	1.25M
+Top1M	BiRNN	**24.85 (+3.38)**	**26.64 (+9.01)**	1.5M
	Baseline	24.66 (+3.19)	26.59 (+8.96)	1.5M
+Top1.25M	BiRNN	**25.01 (+3.54)**	**26.80 (+9.17)**	1.75M
	Baseline	24.72 (+3.25)	26.64 (+9.01)	1.75M
+Top1.5M	BiRNN	**25.18 (+3.71)**	**27.10 (+9.47)**	2M
	Baseline	25.01 (+3.54)	26.85 (+9.22)	2M

Table 2: BLEU scores obtained by the SMT and NMT systems on newstest2013. Pairs is the number of sentence pairs in a training set. The numbers in parentheses are the translation gains with respect to the reference system trained with 500,000 sentence pairs from the Europarl corpus.

generalize better with less data. We are surprised that the BLEU scores obtained with the BiRNN data are so close to those obtained with the data from the baseline system. This may indicate that article pairs from Wikipedia share a relatively high degree of similarity. Still, the gain obtained with BiRNN compared to the baseline extraction system is consistent for all machine translation systems. These results confirm the quality of the 1,500,000 extracted sentence pairs. Hence, we could reduce the value of the decision threshold in order to extract corpora of larger sizes. Given the out-of-domain nature of the Wikipedia article pairs, we see that our approach could be applied to comparable corpora with a lower degree of comparability.

5 Discussion

Most parallel sentence extraction systems developed so far were based on a series of models that required the computation of several features that did not necessarily adapt well to different text structures. To alleviate this problem, we have shown that we can create new parallel corpora from a collection of texts in different languages with a single end-to-end model based on bidirectional recurrent neural networks. Our approach deals directly with raw sentence pairs to determine if two sentences are a translation of each other. In addition to being more flexible than traditional systems, we demonstrate that our approach obtains better results than a competitive baseline system. We found that new sentence pairs extracted from comparable corpora is a considerable resource to exploit for SMT and NMT systems.

Our parallel sentence extraction system is completely based on neural network models and we believe that our work enables exploration for researchers who want to apply future research ideas. In this study we did not consider out-of-vocabulary words. Therefore, it would be beneficial to use word segments (Sennrich et al., 2015) as lexical units instead of words to limit the size of the vocabulary and to be more robust when handling out-of-domain texts found in comparable corpora. The main challenge with our approach (and the majority of parallel sentence extraction systems) is that it needs to be trained on a parallel corpus. Thus, we need parallel sentences to extract parallel sentences, which can be problematic. Despite that it is possible to start from a small parallel corpus and to bootstrap the learning process with extracted sentence pairs, this method is limited for the many language pairs having few resources available. In order to lessen the need of a parallel corpus, an interesting and promising avenue could be

to apply methods from some of the recent works (Xia et al., 2016; Conneau et al., 2017; Artetxe et al., 2017) which allow machine translation systems to learn directly from non-parallel texts. Although our simple neural network architecture was able to extract parallel sentence pairs from comparable corpora, we believe that more sophisticated approaches, such as using two different sentence encoders trained by mutual information neural estimation (MINE) (Belghazi et al., 2018) could improve the flexibility and the performance of our approach. Even though we believe that our approach is easily scalable across multiple language pairs, in this paper we only evaluated it on English–French texts. Therefore, it would be interesting to study a larger set of language pairs. In addition, we emphasize that it would be pertinent to evaluate our approach with distant language pairs which are poor in parallel resources. Our code is available on GitHub and the extracted parallel corpora will be released to the public.[16]

Ackowledgements

This research was partially supported by the Natural Sciences and Engineering Research Council of Canada.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA. USENIX Association.

Sadaf Abdul-Rauf and Holger Schwenk. 2009. On the use of comparable corpora to improve smt performance. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 16–23, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sisay Fissaha Adafre and Maarten de Rijke. 2006. Finding similar sentences across multiple languages in wikipedia. In *Proceedings of the Workshop on NEW TEXT Wikis and blogs and other dynamic text sources*.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *CoRR*, abs/1710.11041.

Andoni Azpeitia, Thierry Etchegoyhen, and Eva Martínez Garcia. 2017. Weighted set-theoretic alignment of comparable sentences. In *Proceedings of the 10th Workshop on Building and Using Comparable Corpora*, pages 41–45. Association for Computational Linguistics.

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

Ishmael Belghazi, Sai Rajeswar, Aristide Baratin, R. Devon Hjelm, and Aaron C. Courville. 2018. MINE: mutual information neural estimation. *CoRR*, abs/1801.04062.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, pages 737–744, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311, June.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2016. Parallel sentence extraction from comparable corpora with neural network features. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

[16] https://github.com/FrancisGregoire/parSentExtract

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *CoRR*, abs/1710.04087.

Cristina España-Bonet, Ádám Csaba Varga, Alberto Barrón-Cedeño, and Josef van Genabith. 2017. An empirical analysis of nmt-derived interlingual embeddings and their use in parallel sentence identification. *CoRR*, abs/1704.05415.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 748–756. JMLR.org.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Jeenu Grover and Pabitra Mitra. 2017. Bilingual word embeddings with bucketed cnn for parallel sentence extraction. In *Proceedings of ACL 2017, Student Research Workshop*, pages 11–16. Association for Computational Linguistics.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197. Association for Computational Linguistics.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *CoRR*, abs/1506.03340.

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

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

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *CoRR*, abs/1701.02810.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *NAACL Workshop on Vector Space Modeling for NLP*, Denver, United States.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4):477–504, December.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Comput. Linguist.*, 29(1):19–51, March.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1310–III–1318. JMLR.org.

Mike. Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

Jason R. Smith, Chris Quirk, and Kristina Toutanova. 2010. Extracting parallel sentences from comparable corpora using document level alignment. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 403–411, Stroudsburg, PA, USA. Association for Computational Linguistics.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *CoRR*, abs/1702.03859.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*, COLING '96, pages 836–841, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yingce Xia, Di He, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. *CoRR*, abs/1611.00179.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

Fast and Accurate Reordering with ITG Transition RNN

Hao Zhang
Google
haozhang@google.com

Axel Ng
Google

Richard Sproat
Google

Abstract

Attention-based sequence-to-sequence neural network models learn to jointly align and translate. The quadratic-time attention mechanism is powerful as it is capable of handling arbitrary long-distance reordering, but computationally expensive. In this paper, with the goal of making neural translation both accurate and efficient, we follow the traditional pre-reordering approach to decouple reordering from translation. We add a reordering RNN that shares the input encoder with the decoder. The RNNs are trained jointly with a multi-task loss function and applied sequentially at inference time. The task of the reordering model is to predict the permutation of the input words following the target language word order. After reordering, the attention in the decoder becomes more peaked and monotonic. For reordering, we adopt Inversion Transduction Grammars (ITG) and propose a transition system to parse input to trees for reordering. We harness the ITG transition system with RNN. With the modeling power of RNNs, we achieve superior reordering accuracy without any feature engineering. In experiments, we apply the model to the task of text normalization. Compared to a strong baseline of attention-based RNN, our ITG RNN reordering model can reach the same reordering accuracy with only 1/10 of the training data and is 2.5x faster in decoding.

1 Introduction

The encoder-decoder neural network architecture for sequence-to-sequence problems has achieved enormous success especially after the introduction of the attention mechanism (Bahdanau et al., 2014). Its applications in NLP range from machine translation (Bahdanau et al., 2014) and sentence summarization (Rush et al., 2015) to text normalization (Sproat and Jaitly, 2017). The attention mechanism is effectively a random memory access mechanism, enabling access to any source sequence position at any decoding step. In principle, it can handle arbitrary reordering of any input length. But its power comes with a high computational cost. The time complexity of the attention mechanism is $O(N^2)$ if the number of decoding steps is proportional to the input length N. In addition to the computational concern, for dominantly-monotonic translation tasks, such as text normalization (Sproat and Jaitly, 2017), soft attention can be too relaxed and sub-optimal in terms of modeling efficiency. Hence, to reduce computational complexity as well as to improve model accuracy, recently there has been a surge of research interest in enforcing monotonic attention (Raffel et al., 2017) and hard attention (Aharoni and Goldberg, 2017) based on the observation that many sequence-to-sequence tasks are monotonic, including speech recognition and morphological inflection.

In reality, the monotonicity assumption has to be made carefully. Even in the highly monotonic translation task of text normalization, which is mapping written text to spoken text, there are systematic reordering patterns like from "*2018-10-03*" to "*October third, two thousand eighteen*", and from "*$100*" to "*one hundred dollars*". These reordering patterns can involve arbitrarily long chunks of input. Without handling the infrequent but systematic reordering, monotonic attention models are doomed. We claim there is hope for efficient and accurate systematic reordering by combining grammars with RNNs. The

Proceedings of the 27th International Conference on Computational Linguistics, pages 1454–1463
Santa Fe, New Mexico, USA, August 20-26, 2018.

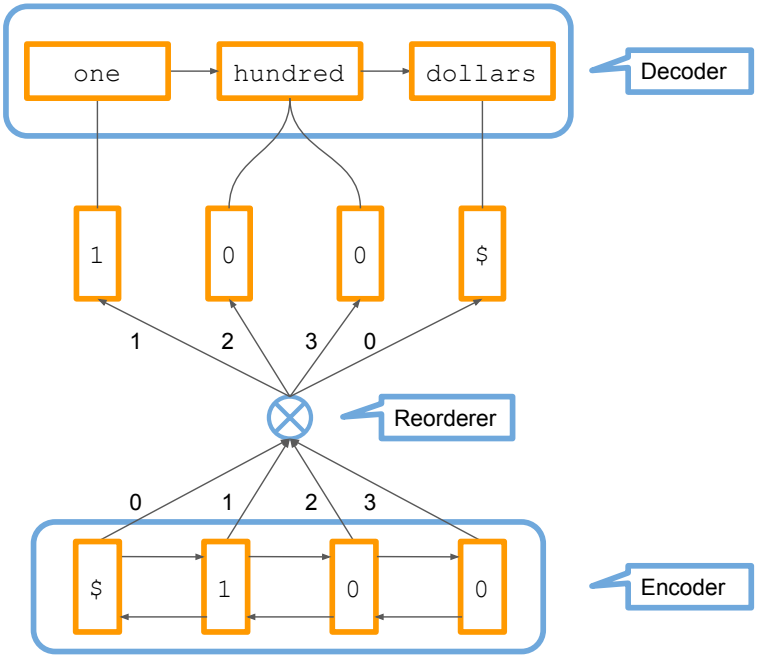


Figure 1: The encoder-reorderer-decoder architecture. The input encoder is shared by the reorderer and the decoder. The reorderer permutes the RNN states of the encoder.

key to our solution is the Inversion Transduction Grammars (Wu, 1997), a type of synchronous context free grammar limiting reordering to adjacent source spans. For machine translation across very different languages, ITGs have been reported to cover most of the alignments observed in parallel data (Zhang et al., 2006). For text normalization, we have not found a single example of reordering that cannot be covered by an ITG. This observation motivates us to factorize translation into two steps: an ITG reordering step followed by a monotonic translation step. The task of the reordering step is to handle systematic reordering through swapping of adjacent source spans. This step "normalizes" the input in terms of word order which opens up the capacity for improvement in translation accuracy. It also opens up the opportunity for more "lightweight" neural translation models such as monotonic attention models of Raffel et al. (2017) and Aharoni and Goldberg (2017). Of course, the burden of reordering has shifted from the attention mechanism to the dedicated reordering model. This idea has produced fruitful results in the era of phrase-based machine translation (Collins et al., 2005; Xu et al., 2009; Neubig et al., 2012; Lerner and Petrov, 2013; Nakagawa, 2015). In this paper, we revive the old idea with a neural treatment.

First of all, we modify the encoder-decoder architecture by adding a reorderer which shares the encoder with the decoder. Figure 1 shows the architecture. This change enables multi-task training of the encoder states and turns the inference into three steps: encoding, reordering, decoding, each of which is a linear chain RNN. Our main algorithmic contribution is an ITG-transition-based RNN reorderer. What we feed to the RNN for training are input strings paired with their permutations, for example *$100* with $(1, 2, 3, 0)$ to indicate it should be reordered to *100$* in order to be translated monotonically. Within the ITG framework, we will show the goal is equivalent to parsing the input into a parse tree annotated with reordering information, in this case $\langle$ *$* [[*1* *0*] *0*] $\rangle$, where $\langle$ $\rangle$ indicates a swapping of two subtrees and [ ] indicates no swapping. For efficiency and compatibility with linear chain RNN, we use a transition system for parsing. The transition system has three actions: SHIFT, REDUCE S, and REDUCE I. In this example, the transition sequence to learn is SHIFT, SHIFT, SHIFT, REDUCE S, SHIFT, REDUCE S, REDUCE I. To learn the conditional distributions of the three actions, we rely on the strength of RNN to condition on the entire transition history without any feature engineering. We report results for both the intrinsic task of reordering and the end-to-end task of translation.

The paper is organized in the following way. In Section 2, we formally introduce ITG and its transition-based parsing system. In Section 3, we introduce the ITG-based RNN model. In Section 4, we formally introduce the end-to-end neural architecture. In Section 5, we discuss related works. In Section 6.1, we carry out two sets of reordering experiments: one on synthetic sequence permutation data, the other on money expression reordering data coming from the task of English text normalization (Sproat and Jaitly, 2017). In Section 6.2, we conduct an end-to-end experiment for text normalization and compare the two-step approach with the baseline approach.

2 ITG Transition System

An Inversion Transduction Grammar is a grammar for generating a pair of languages recursively and synchronously. In our problem of source language reordering, we essentially pair the source language with a virtual target language that shares the vocabulary with the source but follows the word order of the real target language. To be concrete, we use the simplest ITG with just one nonterminal and the following context free production rules.

$$X \rightarrow [X\ X]$$
$$X \rightarrow \langle X\ X \rangle$$
$$X \rightarrow w_0/w_0$$
$$...$$
$$X \rightarrow w_{n-1}/w_{n-1}$$

The first rule is called the *straight rule* because it keeps the order of two constituents unchanged on the target side. The second rule is called the *inverted rule* because it inverts the order of two constituents on the target side. For example, a sentence pair $(w_0, w_1, w_2 \mid w_0, w_2, w_1)$ can be derived with three pre-terminal rules to link the words with the same subscripts on both sides, plus one inverted rule for grouping w_1 and w_2 and one straight rule on the top for grouping w_0 and the phrase of w_1, w_2.

We can also devise a transition system following (Nivre, 2003). The following is the deductive description of the transition system. Each configuration in the transition system consists of a stack and a pointer to the next word in the input buffer. At the beginning, we have an empty stack and a pointer to the first input word. At each time step, we can choose SHIFT if the buffer is not empty and choose to apply REDUCE S or REDUCE I if the stack has a height of at least two. We use four indices to uniquely represent each synchronous constituent that is constructed in the parsing process. The first two index into the source side and the last two index into the target side. The transition system stops when the buffer is empty and the stack has been reduced to size one.

$$\text{INITIAL} : [\phi,\ 0]$$
$$\text{SHIFT} : \frac{[S, i]}{[S|(i, i, i, i),\ i+1]}$$
$$\text{REDUCE S} : \frac{[S|(i, j, l_1, r_1)(j+1, k, l_0, r_0),\ k+1]}{[S|(i, k, l_1, r_0),\ k+1]}$$
$$\text{REDUCE I} : \frac{[S|(i, j, l_1, r_1)(j+1, k, l_0, r_0),\ k+1]}{[S|(i, k, l_0, r_1),\ k+1]}$$
$$\text{FINAL} : [(0, n-1, *, *),\ n]$$

When training a model, we are given a sequence of words $w_0, ..., w_{n-1}$, along with the permutation P indicating its corresponding target word order. It has the equivalent information as a pair of bijectively aligned sequences:

$$w_0, ..., w_{n-1} \quad w_{P(0)}, ...w_{P(n-1)}$$

Zhang et al. (2006) have shown for any P there exists a greedy shift-reduce algorithm to produce a *canonical parse* along with a *canonical transition sequence*. If we take the *$100* example, the canonical

transition sequence is SHIFT, SHIFT, SHIFT, REDUCE S, SHIFT, REDUCE S, REDUCE I, and the corresponding cannonical parse is $\langle$ *$* [[*1 0*] *0*] $\rangle$. Therefore, given $w_0, ..., w_{n-1}$ and P, the learning problem is mapping from $w_0, ...w_{n-1}$ to the canonical transition TRANSITION(P). There are two good properties of the output space. First, the output sequence is always of length $2n - 1$. Second, the output vocabulary is fixed to {SHIFT, REDUCE S, REDUCE I}.

Once a model is learned, at testing time, only the input sequence is given, the task is to predict a transition sequence which in turn corresponds to a permutation.

3 ITG RNN

A recurrent neural network computes the representation of a new state based on the representation of its previous state and input at the current step. Formally, this is

$$h_t = RNN(f(t), h_{t-1})$$

where h is the hidden state representation and f is the input function, and RNN can be a LSTM or GRU cell for instance.

A transition system also has a timeline recurrence. The recurrence of the ITG transition system is the following.

$$s_t = ITG(a(t), s_{t-1})$$

where s is the configuration and a is the action.

To make an ITG controlled by an RNN, we need to unroll an ITG transition system along with an RNN. Figure 2 shows an example of completely unrolled computation graph of ITG-RNN. For input f, we want this to be a function of s. The information in s is a stack along with an input buffer pointer. In Section 2, we specified each stack element as a tuple of (i, j, l, r) that indexes into the source sequence and the reordered target sequence. It is clear that the deduction rules involve at most top two stack elements. Therefore, we extract the indices stored at the top of the stack which are potentially directly involved in any next action: l_1, r_1, l_0, r_0, and the buffer pointer i. With these input indices, we can create a concatenated embedding vector of the input sequence. This will be what the ITG feeds into the RNN as input.

The task of the RNN is to predict the possible next actions. As the ITG transition system constrains the valid next actions at s_t, h_t should only emit action labels allowed by s_t. This will be what ITG places as constraints in the RNN output and will indirectly influence the training of the underlying RNN cells by moving probability mass away from invalid actions. The output constraints are implemented as masks over invalid output actions before applying softmax.

What RNN feeds back to ITG is the predicted actions after softmax. That is how the RNN controls the ITG at inference time. At training time, the action sequence is known.

4 Encoder-Reorderer-Decoder Architecture

As we discussed in Section 1 and demonstrated with Figure 1, a reorderer is designed to solve translation into two steps: reordering and monotonic decoding. In this section, we define the training and decoding objectives mathematically.

Given an input sequence $(x_1, \ldots, x_{T_x})$, a BiRNN, i.e., a pair of forward RNN $\overrightarrow{f}$ and backward RNN $\overleftarrow{f}$ is used to encode it into a pair of *forward hidden states* and *backward hidden states* $(\overrightarrow{h}_1, \ldots, \overrightarrow{h}_{T_x})$ and $(\overleftarrow{h}_1, \ldots, \overleftarrow{h}_{T_x})$, where $\overrightarrow{h}_t = \overrightarrow{f}(x_t, \overrightarrow{h}_{t-1})$ and $\overleftarrow{h}_t = \overleftarrow{f}(x_t, \overleftarrow{h}_{t+1})$. The forward and backward states are concatenated to summarize contexts in both directions: $h_t = [\overrightarrow{h}_t^\mathsf{T}; \overrightarrow{h}_t^\mathsf{T}]^\mathsf{T}$. This is how the encoder works.

Here we depart from the attention-based encoder-decoder architecture. The reorderer defines a probability of the ITG transition sequence $z_1, \ldots, z_{2T_x-1}$ as a product of conditionals.

$$p(z_1, \ldots, z_{2T_x-1} | x_1, \ldots, x_{T_x}) = \prod_{t=1}^{2T_x-1} p(z_t | z_1, \ldots, z_{t-1}, h_1, \ldots, h_{T_x}) \quad (1)$$

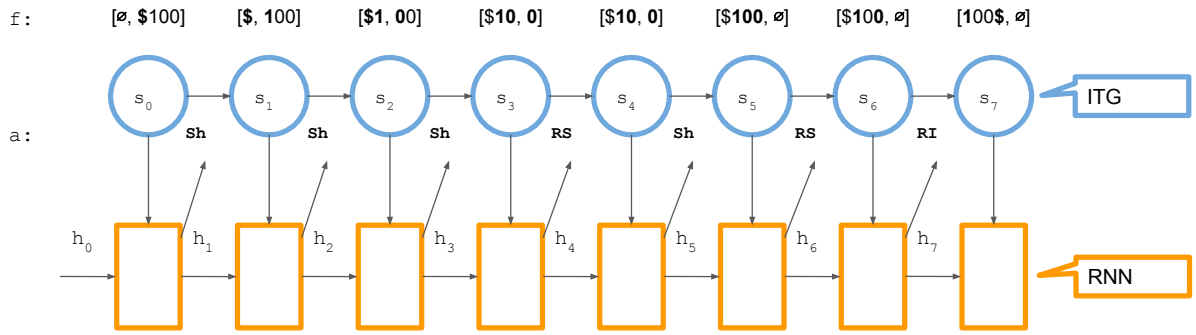


Figure 2: An unrolled computation graph of ITG-RNN. s is for the states of the ITG. h is for the states of the RNN. f is for the input function to the RNN. a is for actions of the ITG and softmax over output of the RNN. For conciseness, we use SH for SHIFT, RS for REDUCE S, and RI for REDUCE I. We use boldface to indicate function f as input focus over the stack and the buffer.

For the ITG RNN, the conditional probability is based on softmax over the output at each step t. As we mentioned in Section 2, for each transition sequence, there exists a functional mapping to a permutation P. Therefore, the probability of reordering is derived in the following way.

$$p(x_{P(1)}, \ldots, x_{P(T_x)} | x_1, \ldots, x_{T_x}) = p(z_1, \ldots, z_{2T_x-1} | x_1, \ldots, x_{T_x})$$

The key change following reordering is that the probability of the output sequence $(y_1, \ldots, y_{T_y})$ is based on the permuted sequence $(x_{P(1)}, \ldots, x_{P(T_x)})$ and we make an independence assumption with respect to the original input sequence.

$$p(y_1, \ldots, y_{T_y} | x_1, \ldots, x_{T_x}) = p(y_1, \ldots, y_{T_y} | x_{P(1)}, \ldots, x_{P(T_x)}) \cdot p(x_{P(1)}, \ldots, x_{P(T_x)} | x_1, \ldots, x_{T_x})$$

where

$$p(y_1, \ldots, y_{T_y} | x_{P(1)}, \ldots, x_{P(T_x)}) = \prod_{t=1}^{T_y} p(y_t | y_1, \ldots, y_{t-1}, h_{P(1)}, \ldots, h_{P(T_x)})$$

Usually, a soft attention is applied to attend enough context from h. But since the sequence of hidden states is reordered, the attention can be made monotonic.

Let's use $S = (x_1, \ldots, x_{T_x})$, $P(S) = (x_{P(1)}, \ldots, x_{P(T_x)})$, and $T = (y_1, \ldots, y_{T_y})$.

The training objective is

$$1/\mathcal{S} \sum_{(T,S) \in \mathcal{S}} \log p(P|S) + \log p(T|P(S)) \tag{2}$$

and the decoding objective is

$$\hat{T} = \arg \max_{T,P} p(P|S) \cdot p(T|P(S)) \tag{3}$$

The term $-\log p(P|S)$ is the reordering loss and the term $-\log p(T|P(S))$ is the translation loss. Both share the input S. In a computation graph, they share the BiRNN component. The hidden states $(h_1, \ldots, h_{T_x})$ are trained for both tasks. At decoding time, we approximate the decoding objective with a beam search over P, followed by another beam search over T.

5 Related Work

There are three papers that are most relevant to ours that relate to ITG pre-ordering. DeNero and Uszkoreit (2011) induce binary source trees first and learn pre-reordering rules for these binary trees from parallel data. Neubig et al. (2012) discriminatively train an ITG parser with CYK parsing for pre-reordering, essentially combining the two steps in DeNero and Uszkoreit (2011) into one. Nakagawa (2015) improve upon Neubig et al. (2012) with a linear time top-down ITG parsing algorithm. They all rely on feature engineering as they use linear models for training. None of them does transition-based parsing for ITG.

There are two papers most relevant to ours that relate to combining transition systems with RNNs. Dyer et al. (2015) propose a stack-LSTM for transition-based parsing. The difference with our approach to modeling is that we do not encode the entire stack explicitly. At each time step, we only feed the context indexed by the current stack and buffer configuration. We do not have an stack RNN, which can be expensive. Our transition RNN can be viewed as a special case of DRAGNN (Kong et al., 2017) which combines fixed features as input with recurrence links at each time step. Our recurrence link is only to the previous time step.

For the approach of incorporating syntactic constraints into neural translation, the following papers are most relevant. Eriguchi et al. (2016) and Chen et al. (2017) assume the existence of source parse trees and enhance the encoder and the attention mechanism to attend to both words and syntactic phrases. We do not rely on external parsers. Stahlberg et al. (2016) let a hierarchical phrase-based decoder guide neural machine translation decoding. Reordering decisions can only be indirectly influenced by the hierarchical decoder. In contrast, we have an explicit hierarchical reordering model applied pre-translation. Eriguchi et al. (2017) train a joint parsing and translation model to maximize the log likelihood of output sequence and input parsing action sequence. This is similar to our multi-task training setup. The key difference is our subtask is ITG parsing for reordering instead of linguistically-motivated parsing.

The idea of adding a reordering layer into neural MT models has also been studied by Huang et al. (2018). They use a simple feed-forward soft and local reordering layer similar to the soft attention mechanism. A fixed window size is used for local reordering. Our RNN reordering layer can handle long distance reordering. Another important difference is that we use discrete variables (permutations) for reordering while the soft reordering mechanism has no latent variables. We leave it as future work to train the end-to-end system by treating ITG transitions and permutations as latent variables.

6 Experiments

6.1 Reordering Experiments

In this part, we analyze the effectiveness of the ITG RNN reordering model. We first create a baseline system using a standard attention-based RNN. This basically implements the right-hand-side conditional in Equation 1 with an attention-based RNN instead of an ITG-RNN. The encoder is a bidirectional GRU cell RNN with one layer in each direction. The input embedding size is 32. The number of GRU cells is 32. The decoder is again a one-layer GRU RNN with 32 cells. The output embedding size is also 32. To make side-by-side comparison fair, we duplicate the network specification in ITG RNN. The only difference is that Equation 1 is implemented with ITG RNN.

6.1.1 Synthetic Reordering Data

The first set of experiments uses synthetic reordering data. We generated all permutations that have valid ITG transition action sequences up to permutation length 10. There are 258,563 such permutations. Then we scrambled letters "A" through "J" using these permutations and paired them up. So, the task is to sort letter sequences. But the models have no prior knowledge of the alphabet. We shuffled the examples and trained on 80% of the data, validated on 10% of the data, and tested on the rest 10%. Figure 3 shows the contrast between ITG RNN and attention-based RNN as we varied the training data size. ITG RNN is much more efficient in utilizing training examples. It only needs roughly 10% of the training data to get the same accuracy as the attention-based system. The attention-based model never reached zero error rate even after training 2 million steps with batch size 32 using the entire training set.

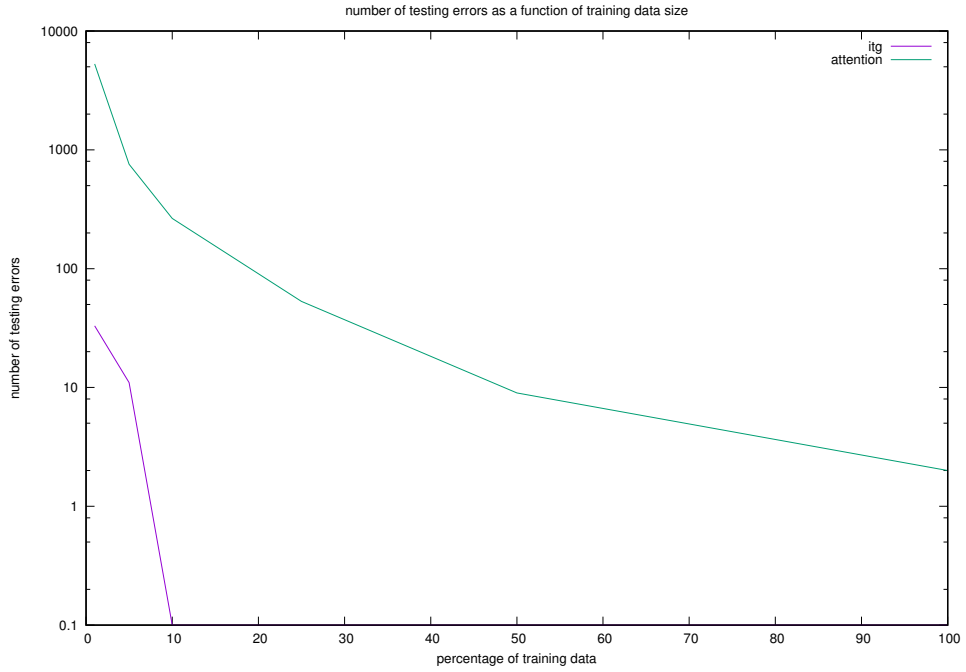


Figure 3: The ITG RNN model reaches zero error rate much faster than the attention-based RNN on the synthetic data set.

	attention-based RNN	*ITG RNN*
all	8×10^{-5}	6×10^{-6}
non-identity	8×10^{-3}	2×10^{-3}

Table 1: Text normalization reordering error rates.

6.1.2 Text Normalization Reordering

For text normalization, interesting reordering patterns include ISO dates such as *1990-09-01*, which should be verbalized as *September first, nineteen ninety*, which indicates a reordering of the last two date components. In measure expressions, interesting cases include *5* km^2, which should be read as *five square kilometers*. There are many more money expressions in real data such as *$100*, *USD100*, *RMB 2 million*, but also cases where no reordering is necessary, like *2 million dollars*. These cases make for a good mixture of reordering patterns.

Focusing on money expressions have a total of 615,642 such reordering examples. We did the same 80-10-10 split for train/validate/test. We use the same network configurations as in section 6.1.1. Figure 4 shows the accuracy comparison. We have a similar trend as with the synthetic data, namely that the ITG RNN is much more effective. We also measured the CPU decoding speed of the two models. Using a single core, the ITG RNN is 2.5x the speed of attention-based RNN. It is worth noting that the two models have encoder and decoder RNNs with the same network dimensions. The reduction of attention accounts for most of the saving.

6.2 End-to-end Text Normalization

For end-to-end translation, we annotated reordering information on 10,561,377 tokens of training data used by Sproat and Jaitly (2017). There are 349,537 DATE/MONEY expressions, and 14,795 non-identity reordering examples. This is a realistic data set because it reflects that only a tiny portion of all tokens require reordering. Table 1 shows the sequence error rate of the reordering model alone. It can reach 99.8% accuracy on the non-identity reordering subset, while an attention-based reordering component can only give 99.2%.

Table 2 summarizes the end-to-end text normalization results, comparing ITG RNN pre-reordering followed by hard monotonic attention (Raffel et al., 2017) against soft (unordered) attention (Bahdanau

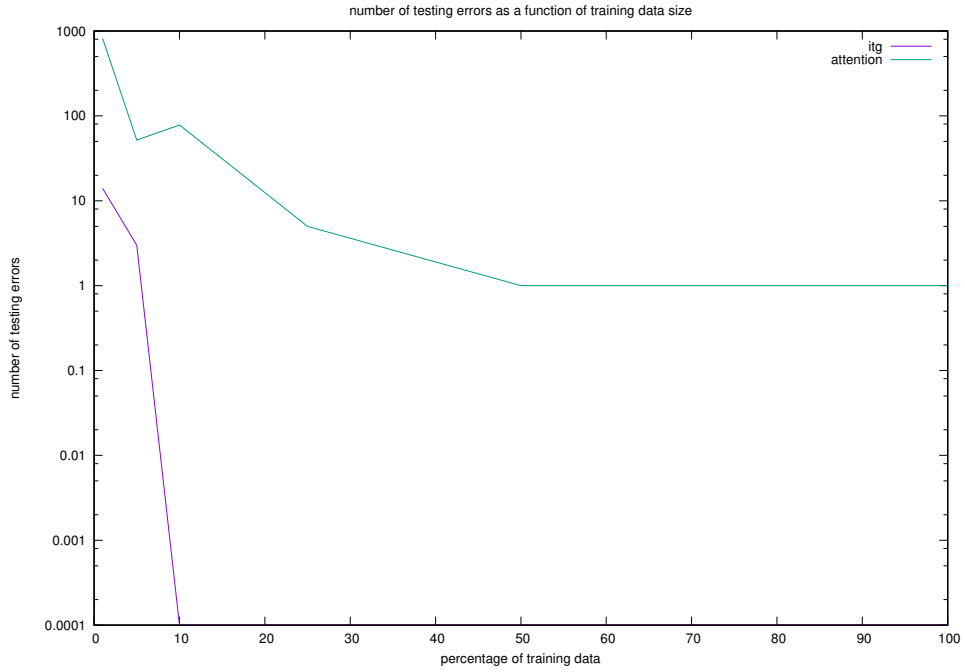


Figure 4: The ITG RNN model reaches zero error rate much faster than the attention-based RNN on the money expression text normalization data set.

	attention-based		*reorderer-based*	
Semiotic Class	Correct	Accuracy	Correct	Accuracy
ALL	143797	99.73%	143778	99.72%
DATE	10438	99.94%	10437	99.93%
MONEY	5838	97.07%	5815	96.69%

Table 2: Text normalization results.

et al., 2014). The details of the baseline normalization model can be found in Sproat and Jaitly (2017). The new system differs only in the attention mechanism. Essentially, the two systems have the same overall accuracy and DATE accuracy. But unfortunately the reorderer-based one is worse in MONEY accuracy.

The overall result indicates that for neural text normalization pre-reordering is feasible with comparable accuracy and has the potential for higher computational efficiency as shown by Raffel et al. (2017). But why do we not observe improvements in accuracy on this data set even though the intrinsic reordering accuracy has reached 99.8%? We suspect that interesting reordering is too rare on this data set and over time, the soft-attention model is fully capable of learning the reordering. We also suspect that the beam search for the reorderer-based system is more prune to search errors because it has two passes: one for the permutation and the other for the actual output based on the permutation produced by the first pass.

7 Conclusion

We propose a neural pre-reordering approach. It reorders the encoder output of the input sequence before feeding it to the decoder RNN. As a result, the burden of modeling reordering is shifted from the attention network to a dedicated reordering model. In particular, we find that combining an Inversion Transduction Grammar based transition system with a RNN results a reordering model that is both fast and accurate. It requires only 1/10 of samples to get to the same reordering sequence accuracy compared to an attention-based RNN reorderer. This demonstrates that reordering can be done more efficiently and accurately

with proper structural constraints.

Our future work has two directions. First, we will apply the model to machine translation data, using automatically aligned data like pre-reordering models for phrase-based machine translation models. Second, we will treat the reordering decisions as latent variables and train the model without alignment annotation and decode without an additional beam search.

Acknowledgements

We thank Shankar Kumar, Xiaochang Peng, and Michael Riley for discussions and comments on this work.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *ACL*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In *ACL*.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *ACL*.

John DeNero and Jakob Uszkoreit. 2011. Inducing sentence structure from parallel corpora for reordering. In *EMNLP*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *ACL*.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *ACL*.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *ACL*.

Po-Sen Huang, Chong Wang, Dengyong Zhou, and Li Deng. 2018. Neural phrase-based machine translation. In *ICLR*.

Lingpeng Kong, Chris Alberti, Daniel Andor, Ivan Bogatyy, and David Weiss. 2017. DRAGNN: A transition-based framework for dynamically connected neural networks. *arXiv preprint arXiv:1703.04474*.

Uri Lerner and Slav Petrov. 2013. Source-side classifier preordering for machine translation. In *EMNLP*.

Tetsuji Nakagawa. 2015. Efficient top-down btg parsing for machine translation preordering. In *ACL*.

Graham Neubig, Taro Watanabe, and Shinsuke Mori. 2012. Inducing a discriminative parser to optimize machine translation reordering. In *EMNLP-CoNLL*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *IWPT*.

Colin Raffel, Thang Luong, Peter J. Liu, Ron J. Weiss, and Douglas Eck. 2017. Online and linear-time attention by enforcing monotonic alignments. *arXiv preprint arXiv:1704.00784*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Richard Sproat and Navdeep Jaitly. 2017. An RNN model of text normalization. In *Interspeech*.

Felix Stahlberg, Eva Hasler, Aurelien Waite, and Bill Byrne. 2016. Syntactically guided neural machine translation. In *ACL*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational. Linguistics*, 23(3).

Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz Och. 2009. Using a dependency parser to improve smt for subject-object-verb languages. In *NAACL*.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *HLT-NAACL*.

Neural Machine Translation with Decoding-History Enhanced Attention

Mingxuan Wang[1] **Jun Xie**[1] **Zhixing Tan**[1] **Jinsong Su**[2] **Deyi Xiong**[3] **Chao bian**[1]
[1]Mobile Internet Group, Tencent Technology Co., Ltd
{xuanswang,stiffxie,zhixingtan,chaobian}@tencent.com
[2]Xiamen University, Xiamen, China
{jssu}@xmu.edu.cn
[3]Soochow University, Suzhou, China
{dyxiong}@suda.edu.cn

Abstract

Neural Machine Translation (NMT) with source side attention have achieved remarkable performance. however, there has been little work exploring to attend to the target side which can potentially enhance the memory capbility of NMT. We reformulate a *Decoding-History Enhanced Attention mechanism* (DHEA) to render NMT model better at selecting both source side and target side information. DHEA enables a dynamic control on the ratios at which source and target contexts contribute to the generation of target words, offering a way to weakly induce structure relations among both source and target tokens. It also allows training errors to be directly back-propagated through short-cut connections and effectively alleviates the gradient vanishing problem. The empirical study on Chinese-English translation shows that our model with proper configuration can improve by 0.9 BLEU upon Transformer and achieve the best reported results in the same dataset. On WMT14 English-German task and a larger WMT14 English-French task, our model achieves comparable results with the state-of-the-art NMT systems.

1 Introduction

Neural Machine Translation(Sutskever et al., 2014) generally adopts an encoder-decoder framework, where the encoder reads and encodes the input text into a distributed representation and the decoder generates translated text conditioned on the input representation. A potential issue with this encoder-decoder approach is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with translating long sentences. A recent successful extension of NMT models is the attention mechanism which conducts a soft search over source tokens and yields an *attentive vector* to represent the most relevant segments of the source sentence for the current decoding state (Bahdanau et al., 2014; Jean et al., 2015; Luong et al., 2014; Vaswani et al., 2017).

The typical attention mechanism frees the neural translation model from having to squash all the information of the source sentence into a fixed vector, however, it ignores some important information hidden in the target sequence (Cheng et al., 2016). At least three challenges still remains in the translation process. The first issue relates to the memory compression problems in the decoding process. During each translation step, a hidden state vector implicitly maintains at least two types of information, including both the most relevant source contexts and the language model over the partially translated sentence. The memory capacity of a single dense vector is not powerful enough to store these information. The second challenge comes with the training and optimization problem associated with vanishing or exploding gradients, which has been studied in the context of vanilla RNNs. Although some RNN variants such as LSTMs provide a temporal shortcut path to avoid the problem in the temporal domain, it is not guaranteed to be sufficient. At last, it should be acknowledged that the target side context is solely based on the sequence model which, in practice, is prone to a fixed vector and lacks the ability to capture effectively nonsequential dependencies among words.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1464–1473
Santa Fe, New Mexico, USA, August 20-26, 2018.

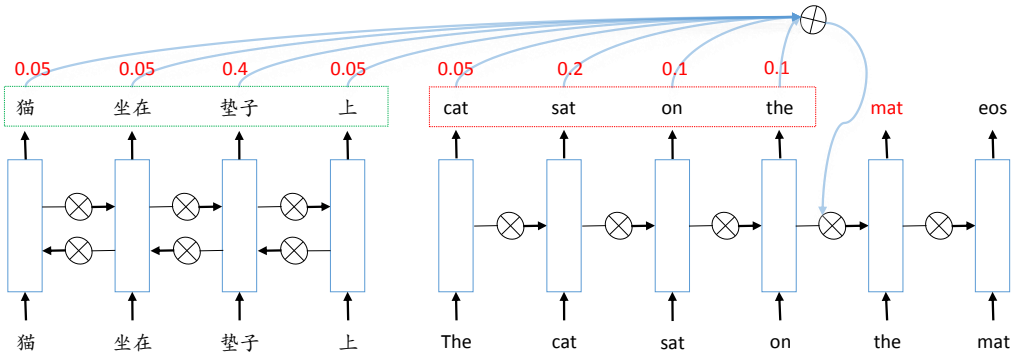


Figure 1: DHEA: *Decoding-History Enhanced Attention mechanism.* The interaction with hidden states that cross both the source and target side can be enhanced by these short-cut connections. The attention weights controls the individual contribution of each hidden states. In this example, the semantic dependency between the word "sat" and "mat" can be be explicitly captured.

In order to address these limitations, we introduce a *Decoding History Enhanced Attention* (DHEA) as a powerful extension to the typical attention based NMT architecture, where the attention over decoding history contributes directly to the prediction of the next word. As illustrated in Figure 1, DHEA explicitly looks back at multiple preceding steps and automatically decides how much previous information should be "seen" by weighting them. Basically, this sort of attention paid on the decoding history facilitates the flow of information from the distant past and is able to emphasize any of the previously translated words, hence it enables the learning of syntactic structures which are useful for the translation task. It also introduces some shortcut connections which connect far-away states to ensure training error signal to be back-propagated directly and alleviate gradient vanishing problem. Further more, we apply attention on the source and target side contexts synchronously, which can better characterize the dependencies within the two sides. Acting in this way, the system learns to determine the ratos of information from the two sides and properly select the amount of context information with regard to the generation of next word in the decoding process. For example, content words in the target sentence should depend more on the source context but less on its prefix (*e.g.* "*mat*" depends more one "*dianzi*" and takes "*cat*" as complementarity). In contrast, function words in the target sentence are often more related to the target context (*e.g.* "*on*" after "*sat*").

Our evaluation on three language pairs shows that the proposed model improves over several baselines, with only a small increase in computational overhead. On the NIST Chinese-English task, DHEA with proper settings yields the best reported result and a 0.9 BLEU improvement over a strong NMT system (Transformer). On WMT English-German and English-French tasks, it again leads to consistent improvements and achieves performance superior or comparable to the state-of-the-art. Finally, we analyze the learned attention function, providing additional insights to its actual contributions.

2 Neural Machine Translation

The NMT model aims to compute the conditional distribution of generating a sentence in a target language given a sentence in a source language, denoted by $p_\theta(y|x)$, where $x = \{x_1 \cdots, x_T\}$ and $y = \{y_1, \cdots, y_{T'}\}$ represent the source and target sentences as sequences of words respectively. θ is a set of parameters usually trained to maximize the conditional log-probability of the correct translation y given the source sentence x:

$$\theta^* = \arg\max_\theta \sum_{i=1}^{N} \log p(y^{(i)}|x^{(i)}) \quad (1)$$

where N is the size of the training corpus. In this section, we will first briefly introduce the NMT model used in our work. Our model modifies the attention based architecture proposed by Bahdanau et

al. (2014), and implements as a deep stack LSTM framework. The whole architecture has three main components: encoder, decoder and attention mechanism.

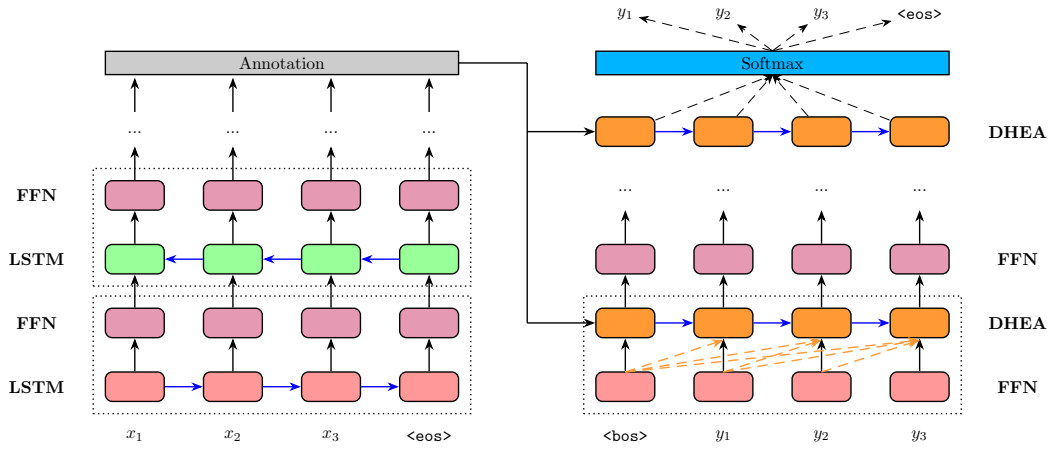


Figure 2: The structure of deep neural machine translation model with DHEA. The dashed arrow directly connects the decoding history and the current state. If we remove this connection, the model degrades to the source side attention based NMT.

Encoder The goal of the encoder is to build meaningful representations of source sentences. The typical encoder consists of a bidirectional RNN which processes the raw input in backward and forward direction with two separate layers, and then concatenates them together. In this work, we choose another bidirectional approach to process the sequence in order to learn more temporal dependencies. Specifically, an RNN layer processes the input sequence in a forward direction. The output of this layer is taken by an upper RNN layer as input, processed in a reverse direction (Zhou et al., 2016).

More formally,The encoder network reads the source input $x = \{x_1, ..., x_T\}$ and processes it into a source side memory $M^{\mathrm{s}} = \{h_1, h_2 \cdots, h_T\}$. where $x_i \in \mathbb{R}^{d_x}$. The output on layer ℓ is

$$h_t^\ell = \begin{cases} x_t, & \ell = 1 \\ \mathrm{LSTM}(h_{t+d}^\ell, h_t^{\ell-1}), & \ell > 1 \end{cases} \tag{2}$$

where

- $h_t^\ell \in \mathbb{R}^{d_h}$ gives the output of layer ℓ at location t.
- The directions are marked by a direction term $d = (-1)^\ell$. If we fixed d to -1, the input will be processed in forward direction, otherwise backward direction.
- We only apply the top-most hidden states as the source side memory which is then fed to the decoder.

Decoder The decoder uses another RNN to generate the translation $y = \{y_1, \cdots, y_{T'}\}$ based on the source side memory M^{s} produced by the encoder. In this work, we use a unidirectional deep stacked LSTM for the decoder RNN. More formally, the hidden state $s_t^\ell \in \mathbb{R}^{d_s}$ on decoder layer ℓ is

$$s_t^\ell = \begin{cases} y_t, & \ell = 1 \\ \mathrm{LSTM}(s_{t-1}^\ell, s_t^{\ell-1}, c_t^\ell), & \ell > 1 \end{cases} \tag{3}$$

Where y_t is the target word embedding at time step t and the context vector $c_t^\ell \in \mathbb{R}^{d_h}$ is dynamically obtained from M^{s} by the attention mechanism. At inference stage, we only utilize the top-most hidden states s to make the final prediction with a softmax layer:

$$p(y_i | y_{<i}, x) = \mathrm{softmax}(s_i W_o) \tag{4}$$

Attention The attention mechanism allows the decoder to select which parts of the source sentence are more useful to predict the next output word. It can be viewed as mapping a query and a set of key-value pairs to an output. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. More specifically, the attention model get context vector c_t after reading from the source representation M^{src}:

$$\begin{aligned} c_i^\ell &= \text{Read}(s_i, M^{\text{s}}) \\ &= \sum_{j=1}^{T} \alpha_{ij}^\ell (h_j W_V^\ell) \end{aligned} \tag{5}$$

Each weight coefficient, α_{ij}^ℓ , is computed using a softmax function:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n} \exp(e_{ik})} \tag{6}$$

And e_{ij}^ℓ is computed using a compatibility function that compares two input elements. The two most commonly used attention functions are additive attention, and dot-product attention. Additive attention computes the weights of the values using a feed-forward network with a single hidden layer. In this work, we consider dot-product with a scaling factor as the compatibility function for its simplicity and efficiency:

$$e_{ij}^\ell = \frac{1}{\sqrt{d_s}} (s_i^{\ell-1} W_Q^\ell)(h_j W_K^\ell)^T \tag{7}$$

Linear transformations of the inputs add sufficient expressive power. $W_K, W_V \in \mathbb{R}^{d_h \times d_s}$ and $W_Q \in \mathbb{R}^{d_s \times d_s}$ are parameter matrices. In general d_s and d_h are set to the same. These parameter matrices are unique per layer. It should be noticed that in Eqn.(7), we apply $s_i^{\ell-1}$ to compute the weights of layer ℓ on the values. In this way the query vector can be prepared in anvance which benifits the implementation of using highly optimized matrix multiplication code.

3 Decoding-History Enhanced Attention for NMT

From Eqn.(5), it is not difficult to observe that the attention weights are associated with the only source side memory. As we have argued before, it is easy for the conventional attention-based NMT models to suffer from generating incoherent texts due to the conflict between the source side attention mechanism and the decoding history. We therefore propose decoding-history enhanced attention to better control over target side contexts. In addition to the source memory M^{s}, DHEA is equipped with a buffer memory M^B as an extension to the conventional approach. M^B is identical to the source side memory M^{S}, except that M^B changes per time step. M_i^B is defined as $\{s_1, s_2, \cdots, s_i\}$. The new context vector is then defined as a combination of the two sides contexts:

$$\hat{c} = \text{Read}(s, M^{\text{S}}, M^{\text{B}}) \tag{8}$$

In this work investigate three strategies for integrating these contexts.

Sum Combination At time step i, DHEA reads from M^B:

$$\begin{aligned} z_i^\ell &= \text{Read}(s_i, M^{\text{B}}) \\ &= \sum_{j=1}^{i} \alpha_{ij}^\ell (s_j^{\ell-1} W_V^\ell) \end{aligned} \tag{9}$$

Similarly, α_{ij} is computed by Eqn.(6) where $j \leq i$ and e_{ij}^ℓ is computed as :

$$e_{ij}^\ell = \frac{1}{\sqrt{d_s}} (s_i^{\ell-1} W_Q^\ell)(s_j^{\ell-1} W_K^\ell)^T \tag{10}$$

One simple way to aggregate information from z_i and c_i is by summing them, then the new context vector is computed as:

$$\hat{c} = z + c \tag{11}$$

It is also worth mentioning that we use $s^{\ell-1}$ as the context information to update the hidden state on layer ℓ, since the lower layer states can be perpared in advance to facilitate parallel training.

Gate Combination As argued by Tu et al. (2016a), the source side context and the target side context plays a different role during the decoding process, we therefore design a context gate which assigns an element-wise weight to the two-side input:

$$\hat{c} = g(c, z) \cdot c + (1 - g(c, z)) \cdot z \tag{12}$$

where $g(c, z) \in (0, 1)$ is a sigmoid neural network which dynamically controls the amount of information flowing from the source and target contexts. $\hat{c}$ is the new context vector fed to the decoder in Eqn.(3) which refines $s_t^\ell = \text{LSTM}(s_{t-1}^\ell, s_t^{\ell-1}, \hat{c}_t^\ell)$.

With the gated control, the new context vector $\hat{c}$ can be rectified based on the decoding history and the source side information. The memory storing useful information of the partial translation can encourage the model to translate contents that are less repeated compared with the already translated contents.

Hybird Combination Considering that we apply the similar attention function to read from the sourc-side memory and the target side memory, it is natural to merge these two memorys into a hybird memory block $M^{\text{S+B}}$. After that, the decoder directly reads from this memory block.

$$\hat{c} = \text{Read}(s, M^{\text{S+B}}) \tag{13}$$

Where the memory block $M^{\text{S+B}}$ is the concatenation of M^{S} and M^{B}. The method can be viewd as a simple implementation of gate combination where the the gating weights are equal to the attention weights.

4 Experiments

4.1 Datasets

We mainly evaluated our approaches on the widely used NIST Chinese-English translation task. In order to show the usefulness of our approaches, we also provide results on other two translation tasks: English-French, English-German. The evaluation metric is BLEU. For Chinese-English task, we apply case-insensitive NIST BLEU. For other tasks, we tokenized the reference and evaluated the performance with multi-bleu.pl. The metrics are exactly the same as in previous work (Papineni et al., 2002).

For Chinese-English, our training data consists of 1.25M sentence pairs extracted from LDC corpora[1], with 27.9M Chinese words and 34.5M English words respectively. We choose NIST 2002 (MT02) dataset as our development set, and the NIST 2003 (MT03), 2004 (MT04), 2005 (MT05), and 2006 (MT06) datasets as our test sets.

For English-German, to compare with the results reported by previous work, we used the same subset of the WMT 2014 training corpus that contains 4.5M sentence pairs with 91M English words and 87M German words. The concatenation of news-test 2012 and news-test 2013 is used as the validation set and news-test 2014 as the test set.

To evaluate at scale, we also report the results of English-French. To compare with the results reported by previous work on end-to-end NMT, we used the same subset of the WMT 2014 training corpus that contains 36M sentence pairs. The concatenation of news-test 2012 and news-test 2013 serves as the validation set and news-test 2014 as the test set.

[1]The corpora include LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06.

4.2 Training details

Our training procedure and hyper parameter choices are similar to those used by (Vaswani et al., 2017). In more details, we limited the source and target vocabularies to the most frequent $30K$ words in Chinese-English translation. For English-German translation and English-French translation, we use $50K$ sub-word tokens as vocabulary based on Byte Pair Encoding(Sennrich et al., 2015).

We initialized parameters by sampling each element from the Gaussian distribution with mean 0 and variance 0.01^2. Parameter optimization was performed using stochastic gradient descent. Adam (Kingma and Ba, 2015) was used to automatically adapt the learning rate of each parameter ($\beta_1 = 0.9, \beta_2 = 0.98$ and $\epsilon = 10^{-8}$). To avoid gradient explosion, the gradients of the cost function which had ℓ_2 norm larger than a predefined threshold 25 were normalized to the threshold (Pascanu et al., 2013). We batched sentence pairs by approximate length, and limited input and output tokens per batch to 8192 per GPU. Each resulting training batch contained approximately 60,000 source and 60,000 target tokens. We trained our NMT model with the sentences of length up to 150 words in the training data. During training, we employed label smoothing of value $\epsilon = 0.1$(**?**).

Translations were generated by a beam search and log-likelihood scores were normalized by the sentence length. We used a beam width of 8 and length penalty $\alpha = 0.6$ in all the experiments. Dropout was applied on each layer to avoid over-fitting (Hinton et al., 2012). The dropout rate was set to 0.1. Except when otherwise mentioned, NMT systems had 6 layers encoders and 4 layers decoders. We trained for 300,000 steps on 8 M40 GPUs, and averaged the last 20 checkpoints, saved at 30 minute intervals. For our small model, the dimensions of all the hidden states were set to 512 and for the big model, the dimensions were set to 1024.

4.3 Results on Chinese-English Translation

SYSTEM	MT03	MT04	MT05	MT06	AVE.
Existing systems					
Moses	31.61	33.48	30.75	30.85	31.67
DEEPLAU(Wang et al., 2017)	39.35	41.15	38.07	37.29	38.97
FRNN+PRNN (Zheng et al., 2017)	37.90	40.37	36.75	34.55	37.39
COVERAGE+Context Gate(Tu et al., 2016a)	-	-	34.13	34.83	-
Transformer (Vaswani et al., 2017)	45.16	46.81	44.62	43.53	45.03
Our deep NMT systems					
Source Attn.	45.08	46.90	44.32	43.13	44.85
DHEA + Hybird Comb.	46.58	47.20	45.45	43.47	45.66
DHEA + Sum Comb.	46.38	47.15	45.30	43.60	45.50
DHEA + Gate Comb.	46.60	47.73	45.35	43.97	**45.90**

Table 1: Case-insensitive BLEU scores on Chinese-English translation.

Table 1 shows BLEU scores on Chinese-English datasets. Clearly DHEA leads to remarkable improvements over their competitors. Compared to the source side attention based model, DHEA is $+1.05$ BLEU score higher on average four test sets, showing the modeling power gained from the decoding history. Among the combination method of the contexts, the context gate function drives consistent improvements over sum and hybrid function by $+0.4$ and $+0.2$ in terms of BLEU. We conjecture it is because the adaptive gate function conditioned on the decoding history make it able to automatically decide how much information should be transferred during decoding to the next step.

To show the power of DHEA, we also make a comparison with previous work. Our best single model outperforms both a phrased-based MT system (Moses) as well as a strong source attention-based NMT system (DeepLau) by $+14.2$ and $+6.8$ BLEU points respectively on average. The result is also better than some other state-of-the-art variants of attention-based NMT model with big margins. Compared to Transformer, DHEA yields a gain of $+0.9$ BLEU and achieves the best performance ever reported this dataset.

4.4 Results on English-German and English-French Translation

SYSTEM	Architecture	EN-Fr BLEU	EN-DE BLEU
Existing systems			
Buck et al. (2014)	Winning WMT14 system	35.7	20.7
Wu et al. (2016)	GNMT + Ensemble	40.4	26.3
Gehring et al. (2017)	ConvS2S	40.51	25.16
Vaswani et al. (2017)	Transformer (small)	38.1	27.3
Vaswani et al. (2017)	Transformer (large)	41.0	28.4
Our deep NMT systems			
this work	Source Attn (small)	39.3	27.5
this work	DHEA + Gate Comb. (small)	40.4	27.9
this work	DHEA + Gate Comb. (large)	**41.6**	**28.7**

Table 2: Case-sensitive BLEU scores on English-German and English-French translation. Our proposed model outperforms the state-of-the-art models including the Transformer (Vaswani et al., 2017).

The results on English-German and English-French translation are presented in Table 2. We compare our NMT systems with various other systems including the winning system in WMT14 (Buck et al., 2014), a phrase-based system whose language models were trained on a huge monolingual text, the Common Crawl corpus. For end-to-end NMT systems, to the best of our knowledge, GNMT is the best RNN based NMT system. Transformer (Vaswani et al., 2017) is currently the SOTA system which is about 2 BLEU points better than GNMT on the English-German task and 0.6 BLEU points better than GNMT on the English-French task.

DHEA achieves a 0.6 BLEU score improvement over the state-of-the-art on the English-to-German task for the smaller network and 0.3 BLEU improvement for the larger network. In the case of the larger English-to-French task, we obtain a 2.3 BLEU improvement for the smaller model and a 0.6 improvement for the larger one. Also, note that the performance of the smaller model for DHEA is close to that of the larger baseline model, especially for the English-to-French task. This suggests that DHEA better utilizes available model capacity.

4.5 Analysis

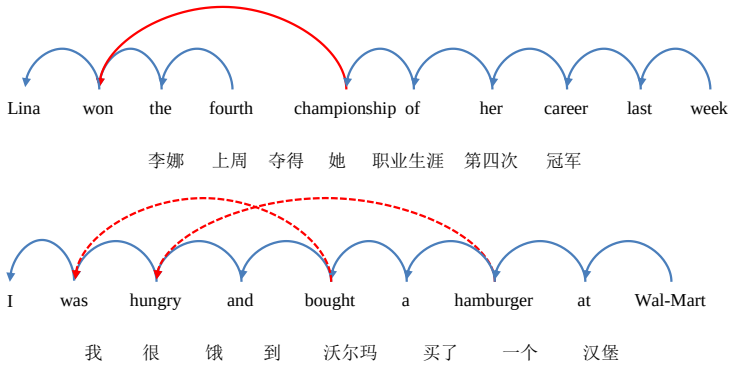


Figure 3: Examples of the target side attention during decoding process. Solid arrows denote which word is being focused when attention is computed, but not the direction of the relation. Red lines indicate long-term dependencies. Dashed lines denote which word is bing second focused.

In order to better understand whether the network benefits from the decoding history, we biefly visualized the skeleton of the target side attention. In Figure 3, although we explicitly allow DHEA to attend to any memory cell, much attention selects the immediately previous word. This agrees with the linguistic intuition that long-term dependencies are relatively rare and hard to learn. This model still finds some

long-term dependencies among words (e.g., the dependency between *won* and *championship*, *bought* and *was*, *hungry* and *hamburger*). These detected dependencies reflect some head-modifier relations in dependency graphs.

5 Related Work

Memory Networks There are variety of studies proposed to increase the LSTM memory capacity by using memory networks. The two most salient examples are Neural Turing Machine (NTM) (Graves et al., 2014) and Memory Network (Weston et al., 2014). Cheng et al. (2016) propose a machine reading simulator which processes text incrementally from left to right. In the NMT task, Wang et al. (2016) present a decoder enhanced decoder with an external shared memory which extends the capacity of the network and has the potential to read, write, and forget information. In fact DHEA can be viewed as a special case of memory networks, with only reading mechanism for the translation task. Quite remarkably DHEA incorporates two different types of memory (source memory and decoding history memory) and significantly improves upon state-of-the-arts.

Attention Mechanism Attention in neural networks (Bahdanau et al., 2014; Luong et al., 2015) is designed to assign weights to different inputs instead of threating all input sequences equally as original neural networks do. A number of efforts have been made to improve the attention mechanism(Tu et al., 2016b; Mi et al., 2016; Zhang et al., 2017). Some of them incorporated the previous attention history into the current attention for better alignment, but none of them are based on the decoding history.

The application of self-attention mechanisms in RNNs have been previously studied, and in general, it appears to capture syntactic dependencies among distant words (Liu and Lapata, 2017; Lee et al., 2017; Kim et al., 2017; Lin et al., 2017). Vaswani et al. (2017) resort to self-attention mechanism and showed outstanding performance. Our approach is diffrent from their work in two aspect. First, our method can be viewed as a variant of RNN decoder which allows a form of memory, thus has the the potential to better handle sentences of arbitrary length. Second, we forcus on controlling the information flow between the source side memory and the target side memory and design a gate to balance the contribution of the two-sides.

Recurrent Residual Networks Our work is also related to residual connections, which have been shown to improve the learning process of deep neural networks by addressing the vanishing gradient problem (He et al., 2015; Szegedy et al., 2016). Recently, several architectures using residual connections with LSTMs have been proposed (Kim et al., 2017; Wang, 2017; Wang and Tian, 2016) for sequence prediction. These connections create a direct path from previous layers, helping the transmission of information. Related to our work, Miculicich et al. (2018) propose a target sideattentive residual recurrent network for decoding, where attention over previous words contributes directly to the prediction of the next word. Comparatively, DHEA attends to the previous hidden state and make a combination with the source context.

Exploiting Contextual Information A thread of work in sequence to sequence learning attempts to exploit auxiliary context information(Wang and Cho, 2016; Li et al., 2017; Zhang and Zong, 2016). Recently Tu et al. (2016a) propose using context gates in NMT to dynamically control the contributions from the source contexts and the RNN hidden state. Our approach focuses on integrating the decoding history and the source side context to NMT architecture. In addition, we have a multi-layer approach to better utilize the contextual information. Experiments in Section 4.3 show the superiority of DHEA.

In the same period of our work, Lin et al. (2018) and Xia et al. (2017) first turn eyes to the target side attention of NMT architecture. Our approach share the similar idea with these work. The diffrence lies in that we concernd more about the integrating of the souce side context and the target side context and designed three types of combination functions. In addition, we approached in a multi-layer way which is more effective.

6 Conclusion

We presented a decoder history enhanced attention mechanism to enrich the target side contextual information for neural machine translation. Our empirical study in three translation tasks shows that it can significantly improve the performance of NMT.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Christian Buck, Kenneth Heafield, and Bas Van Ooyen. 2014. N-gram counts and language models from the common crawl. In *LREC*, volume 2, page 4. Citeseer.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *empirical methods in natural language processing*, pages 551–561.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China, July. Association for Computational Linguistics.

Jaeyoung Kim, Mostafa El-Khamy, and Jungwon Lee. 2017. Residual LSTM: design of a deep recurrent architecture for distant speech recognition. *CoRR*, abs/1701.03360.

Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. *international conference on learning representations*.

Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2017. Recurrent additive networks. *CoRR*, abs/1705.07393.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. pages 688–697.

Zhouhan Lin, Minwei Feng, Cicero Nogueira Dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding.

Junyang Lin, Shuming Ma, Qi Su, and Xu Sun. 2018. Decoding-history-based adaptive control of attention for neural machine translation.

Yang Liu and Mirella Lapata. 2017. Learning structured text representations. *CoRR*, abs/1705.09207.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation.

Lesly Miculicich Werlen, Nikolaos Pappas, Dhananjay Ram, and Andrei Popescu-Belis. 2018. Self-attentive residual decoder for neural machine translation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

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

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. 2016. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2016a. Context gates for neural machine translation. *CoRR*, abs/1608.06043.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016b. Modeling coverage for neural machine translation. *ArXiv eprints, January*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

Tian Wang and Kyunghyun Cho. 2016. Larger-context language modelling with recurrent neural network. In *Meeting of the Association for Computational Linguistics*, pages 1319–1329.

Yiren Wang and Fei Tian. 2016. Recurrent residual learning for sequence classification. In *Conference on Empirical Methods in Natural Language Processing*, pages 938–943.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2016. Memory-enhanced decoder for neural machine translation. *empirical methods in natural language processing*, pages 278–286.

Mingxuan Wang, Zhengdong Lu, Jie Zhou, and Qun Liu. 2017. Deep neural machine translation with linear associative unit. *CoRR*, abs/1705.00861.

Cheng Wang. 2017. RRA: recurrent residual attention for sequence learning. *CoRR*, abs/1709.03714.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR*, abs/1410.3916.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yingce Xia, Fei Tian, Tao Qin, Nenghai Yu, and Tie Yan Liu. 2017. *Sequence Generation with Target Attention*.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of EMNLP*.

Jinchao Zhang, Mingxuan Wang, Qun Liu, and Jie Zhou. 2017. Incorporating word reordering knowledge into attention-based neural machine translation. In *Meeting of the Association for Computational Linguistics*, pages 1524–1534.

Zaixiang Zheng, Hao Zhou, Shujian Huang, Lili Mou, Xinyu Dai, Jiajun Chen, and Zhaopeng Tu. 2017. Modeling past and future for neural machine translation. *CoRR*, abs/1711.09502.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. *arXiv preprint arXiv:1606.04199*.

Transfer Learning for a Letter-Ngrams to Word Decoder in the Context of Historical Handwriting Recognition with Scarce Resources

Adeline GRANET, Emmanuel MORIN, Harold MOUCHÈRE, Solen QUINIOU, Christian VIARD-GAUDIN

LS2N, UMR CNRS 6004, Université de Nantes, France
firstname.lastname@ls2n.fr

Abstract

Lack of data can be an issue when beginning a new study on historical handwritten documents. In order to deal with this, we present the character-based decoder part of a multilingual approach based on transductive transfer learning for a historical handwriting recognition task on Italian Comedy Registers. The decoder must build a sequence of characters that corresponds to a word from a vector of letter-ngrams. As learning data, we created a new dataset from untapped resources that covers the same domain and period of our Italian Comedy data, as well as resources from common domains, periods, or languages. We obtain a 97.42% Character Recognition Rate and a 86.57% Word Recognition Rate on our Italian Comedy data, despite a lexical coverage of 67% between the Italian Comedy data and the training data. These results show that an efficient system can be obtained by a carefully selecting the datasets used for the transfer learning.

1 Introduction

An increasing amount of handwritten historical documents is becoming digitally available, as a mean for preserving this heritage and making it accessible to all. However, digitization is not sufficient to make the documents usable: it is necessary to extract informations in order to index them. Researchers and historians in the humanities and in the social sciences need to be able to query them and find answers rapidly. Therefore, new projects involve the domains of language processing, document recognition, and information retrieval for historical studies.

In the field of Natural Language Processing (NLP) for historical documents, the main challenge is currently to analyse and normalize the spelling of texts (Garrette and Alpert-Abrams, 2016; Bollmann et al., 2017) whereas challenges of Computer Vision for historical documents are more diverse: they involve segmentation into words, lines, or paragraphs, as well as keyword spotting, or handwriting recognition (HWR). In recent years, the number of competitions regarding historical documents has increased (Cloppet et al., 2016; Pratikakis et al., 2016; Sanchez et al., 2017). Such systems have to deal with the complexity of the task as well as the document medium, its level of deterioration, or even its written language. All of these can have a strong impact on the system efficiency.

Lately, the trend in HWR is the use of deep neural networks and, more recently, the integration of attention models in the networks (Bluche et al., 2017). The handwriting systems are built with a Multidimensional Long Short-Term Memory (MDLSTM) network or even with a Convolutional Recurrent Neural Network (CRNN) stacked with a Bidirectional Long Short-Term Memory (BLSTM) (Granell et al., 2018). The neural network training includes a Connectionist Temporal Classification (CTC) cost function proposed by (Graves et al., 2006). Those approaches enable the use of all the available contexts. To decode sequences, there are several strategies: a dictionary, a language model, or a Weighted Finite State Transducer (WFST) including several dictionaries. Nevertheless, results are constrained by the size of the vocabulary used. When too many words are out-of-vocabulary, the results are degraded. To improve the results, methods can be employed to increase the size of the vocabulary by using Wikipedia

Proceedings of the 27th International Conference on Computational Linguistics, pages 1474–1484
Santa Fe, New Mexico, USA, August 20-26, 2018.

or other available resources. Whether improving the results or training the networks, the HWR systems require a lot of data.

We are now working on a new resource (the financial records of the Italian Comedy) with no ground-truth. The changing layout of these documents, the multilingualism (French and Italian), and their special writing make them more complex to study. Moreover, an annotation step would be time-consuming for experts to create a ground-truth. Therefore, transductive transfer learning is an interesting approach when no ground-truth is provided. Indeed, it uses different sources of data to train a system for a specific task by applying various target data (Pan and Yang, 2010). It enables us to use available existing resources to annotate unknown data. It is eagerly used to supply the lack of data for greedy systems such as word spotting in historical documents (Lladós et al., 2012) or translation models with multimodal systems such as (Nakayama and Nishida, 2017).

The standard of machine translation is an encoder-decoder system based on a recurrent neural network (Cho et al., 2014). The first component encodes the source language into a fixed vector and the last one decodes the sequence into the target language. Similarly, (Vinyals et al., 2015) proposed a neural image caption generator that consists of two sub-networks: a pre-trained Convolutional Neural Network (CNN) encoding an image into a fixed size vector, using the last hidden layer from GoogleNet, and a LSTM model generating the corresponding description. We were inspired by this idea to split up our handwriting recognition system into an encoder and a decoder. First, an image encoder takes a word image as input and, thanks to a fully convolutional network (FCN), converts it into a vector of letter-ngrams as in (Huang et al., 2013; Bengio and Heigold, 2014). Here, the transfer learning is performed during the training step that uses available labelled image data. Then, the decoder is built with a recurrent layer in order to generate a sequence of characters from the vector of letter-ngrams. Here, a second transfer learning is independently made on the vocabulary. Contrary to the machine translation and description generator works, the sub-networks are trained independently from each other.

In this paper, we investigate the robustness of the decoder part to infer rules on the order of characters from the letter-ngrams vector only. Within this context, we present three contributions:

1. we propose a character-based decoder and evaluate it on the historical vocabulary of the Italian Comedy;
2. we show that historical resources are more appropriate than Wikipedia to complement the vocabulary of historical documents;
3. we also show that the transfer learning across languages and periods of time works based on a bag of letter-ngrams.

This paper is organized as follows. We describe our transfer model within the encoder and decoder components in Section 2. Section 3 presents the datasets used and their preprocessing. The experimental setups are presented in Section 4 which is followed by the report of our experiments and results in Section 5.

2 Transfer Models

In this section, we present the architecture of our system. We aim to build a handwriting recognition system for multilingual historical documents using multiple resources with different languages and different creation periods.

2.1 Prior Work

Our first approach was based on a BLSTM-CTC network intergrating the transfer learning (Granet et al., 2018b). Firstly, an image is provided to a fully convolutional neural network to automatically extract features. Then, the extracted features were given to a BLSTM network. The cost was computed by the CTC activation function. The CTC enables the use of an unsegmented input image at a word or a character level and provides a sequence of characters as output. We implemented and evaluated this system with a training and test data from the same resource and, then, used the system on unknown data. The model did not use a dictionary nor a language model during the decoding. The resources were in French, Spanish, and English for the training set and in French, and Italian for the target datatest. The

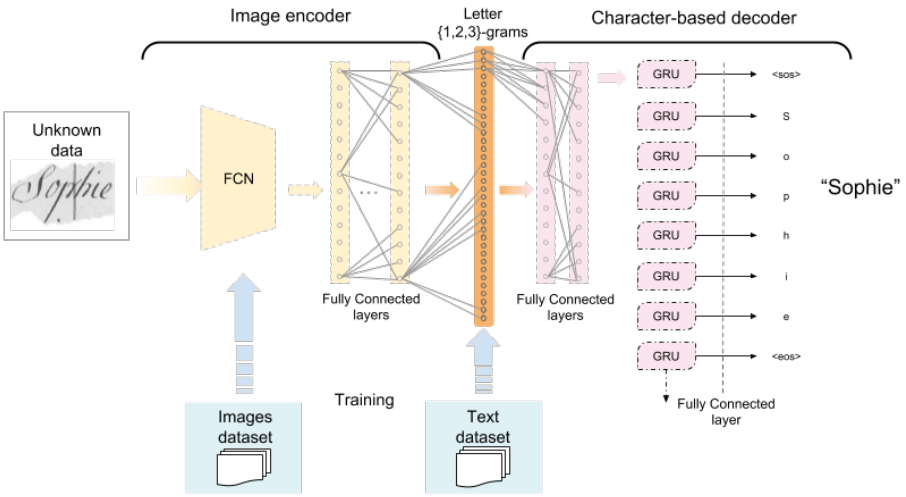


Figure 1: Overview of our encoder-decoder using transfer learning on new historical documents. On the left, the yellow component corresponds to the image encoder from which the features are extracted thanks to a FCN and two connected layers with 1,024 units each and a vector of letter-ngrams is built. On the right, the pink component corresponds to the sequence generation decoder that provides the word, character by character, corresponding to the input vector.

results of this system showed that a language model was learned implicitly to generate the sequence of characters and that the correctly recognized words were words close from one language to the other. Therefore, without a specialized training on a small part of the data to be transcribed, the transfer learning did not work well. The model allowed to reach only 10% of characters which are correctly recognized. From these results, we concluded that a BLSTM-CTC network was unsuitable for the transfer learning case.

Word classification could have been a solution. Nevertheless, in the field of transfer learning, this strategy quickly reaches its limits when working on target data with a closed vocabulary with many named entities, and a significant number of out-of-vocabulary words. Therefore, classification could not be applied to our data.

2.2 Encoder-Decoder for Transfer Learning

Based on the work on image description generation, we define a new system that is divided into two complementary components: an image encoder and a character decoder, as shown in Figure 1. The first one encodes a word image into a vector and the second one decodes the vector into a character sequence or word. The interface vector that links these two parts is a bag of characters or letter n-grams. We estimated the set of letter-ngrams on all resources used for both the training data and the transfer target data. The originality in using a letter-ngrams vector as a pivot is that we encode the input into a non-latent space which is transferable as long as the training data and transfer target data share the same alphabet. In this paper, we focus on the character-based decoder to generate corresponding words. Nonetheless, it is interesting to give informations on the whole system to understand the overall approach.

Image Encoder We define our encoder as shown on the left part of Figure 1:

- a fully convolutional network;
- 2 fully connected layers with a ReLU activation function (Nair and Hinton, 2010) and 1,024 hidden units each;
- one last fully connected layer with a sigmoid and L+1 units where L is the number of estimated letter-ngrams and the additional unitias a joker if the letter-ngram is unknown.

The last layer with the sigmoid activation function allows us to have a probability for each letter-ngram independently of the others.

Character-Based Decoder We turned to neural encoder-decoder architectures in NLP as it enables us to map one sequence to another one without having them to have the same length or word order such as in machine translation (Bahdanau et al., 2014) and description generation (Vinyals et al., 2015). We define the architecture for the character-based decoder as shown on the right part of Figure 1:

- one fully connected layer with a ReLU activation function;
- one Gated Recurrent Unit layer (GRU) (Cho et al., 2014);
- one fully connected layer with a softmax activation function.

These components were designed to be the most minimalist. Moreover, they do not use an embedding layer since it might interfer with the transfer learning that has to deal with different languages.

For the sequence generation task, we used a recurrent layer to add a temporality followed by a softmax layer that gives one character at the time until the word end symbol *J* is given. We chose not to include a bidirectional recurrent layer contrary to encoder-decoder models that are used in machine translation. Since our character-based decoder uses a vector without any order indication on characters from words, all the information and context are included in the vector. Therefore, a bidirectional recurrent layer is useless. Overall the system must infer sequences of characters from the vector content. If we analyse the simple short word "are", it consists of 3 unigrams, 4 bigrams, and 3 trigrams. This decomposition includes the n-grams with the beginning and end symbols of the word. So the decoder needs to deduce the order of characters from the letter-ngrams.

3 Dataset Collection and Statistics

In this section, we present the resources used in our experiments as well as their representation.

3.1 Target Data: Financial Records of the Italian Comedy

The studied documents consist of more than 28,000 pages of financial records of the Italian Comedy from 1716 to 1791. These pages give the list of daily, monthly and annual receipts, with details on the composition of the troupe of actors for each season. We noticed several evolutions over the years:

- the language switches from Italian (with several dialects) to French;
- the structure changes but keeps the same amount of information;
- the writing style is irregular at the beginning of the century but, after that, a cashier was nominated so the style became stable.

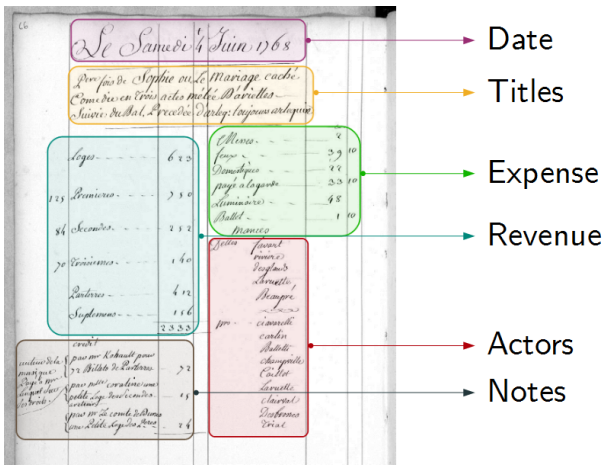


Figure 2: Example of a financial daily record of the Italian Comedy with fields identified.

Figure 2 shows all the information that can be found in a daily record such as the date, the titles of the plays, the revenues and expenses, the actor names, and also some notes. Our study focuses on the title field which contains the list of plays that were performed that day. This list can be extended by some indications as if it was the first performance of the play, if the performance took place at the king's court, or if a new actor started to play. An example of this additional information is shown in Figure 2: the

title field explains that it was a performance of *"Sophie ou le mariage caché"* (lit. *"Sophie or the secret marriage"*) which was a comedy in three acts preceded by *"Arlequin toujours Arlequin"* (lit. *"Arlequin always Arlequin"*). Moreover, a title can be written in many ways with the complete title or a shorter title. Titles were mostly constituted by named entities such as "Raton and Rosette" or "Zemire and Azor".

The language and writing style are essential for the decoder part. We explained above that the language can be either Italian or French. Compared to the modern writing of these languages, we note some differences such as special characters like the long form of 's', the evolution of the 'y' spelling to 'i' or using 'i' and 'j' indiscriminately, or some weak and strong abbreviations. We define as weak abbreviation, a symbol that replaces few characters of a word, while a strong abbreviation reduces a long title to only two or three words.

Thanks to recent works and a participative annotation website (Granet et al., 2018a), we collected and validated 971 annotated title lines. We used all of these annotations to test our decoder system: this dataset is thereafter called Italian Comedy Registers (ICR).

3.2 Training Datasets

In order to build an efficient decoder, we must carefully select the training datasets. Table 1 presents the vocabulary size for each dataset that are used to train the decoder. We choose to create a new dataset from available digitized historical books, as well as to use three existing image resources (RIMES, George Washington, and Los Esposalles) and the French version of Wikipedia since it is a widely used resource in NLP tasks.

	Google IC	RIMES	Los Esposalles	Georges Washington	Wikipedia Fr	IC Registers
Train	26,573	4,477	2,565	660	24,456	0
Validation	2,953	1,578	629	521	3,843	0
Test	0	1,627	629	431	1.928	1,431

Table 1: Vocabulary size of the training, validation, and test sets of each dataset.

New Dataset (Frinken et al., 2013) built a very large vocabulary gathered from a Google N-grams project and an edition of a manually transcribed book from the 16^{th} century. We had a similar approach to this work as we built a new dataset from digitized books of the same century of our data, with a similar closed vocabulary. To preserve the cultural heritage for future generations, copies of books have been digitized, automatically OCRized, and distributed via *Google Book*, for example. We selected 23 books dealing with the Italian Comedy from (and published in) the 18^{th} century. These books include bilingual scripts of plays (French and Italian), collections of plays, and anecdotal stories of the Italian theater. The extracted texts were cleaned to remove the noise such as the structure of the texts and special characters. This new resource, called Google Italian Comedy (GIC), has the advantage of having the same closed vocabulary with specific writings as in the records of the Italian Comedy.

Existing Datasets For our transfer learning approach, we carried out experiments with four available datasets of images, sharing at least one common feature with our target data, as well as Wikipedia to compare the effect of a modern and general resource on the Italian Comedy data:

- RIMES (RM) stands for Recognition and Indexing of Handwritten Documents and Faxes) and is a French database developed to evaluate automatic systems that recognizes and indexes handwritten letters (Grosicki and El-Abed, 2011);
- George Washington (GW) is an English database created from the George Washington Papers at the Library of Congress (Fischer et al., 2012);
- Los Esposalles (ESP) is a Spanish database compiled from a marriage licence book collection from the 15^th and 17^th centuries (Fischer et al., 2012);
- Wikipedia Fr data (Wiki), as used and distributed by (Bojanowski et al., 2017), provides all words whose frequency is greater than 5 in Wikipedia. From this dataset, we randomly selected 30 000

words.

To our knowledge, no other experiments dedicated to the construction of a decoder use the resources RM and ESP which are initially image resources. However, we found it interesting to observe the impact of trained models because ESP is mainly composed of named entities, and RM is a contemporary base with an administrative vocabulary. Therefore, they are far from our Italian Comedy data.

3.3 Representation

Preprocessing of the Training Datasets In all the datasets, we replaced the diacritical marks by their simple form, such as {é, è} are *e* characters, and the typical long form of *s* from the 18^{th} as a short form for the decoder input and also the target data. All words are case sensitive. For GIC only, text lines were split on whitespaces and punctuation marks. In contrast, ICR title lines were only split on whitespaces and we kept with the punctuations in the data.

Letter-Ngrams As suggested by (Bengio and Heigold, 2014; Vania and Lopez, 2017), we use letter-ngrams as a pivot in our multimodal encoder-decoder. Initially, the authors selected the 50 000 most popular letter-ngrams to represent words. Adding *[* and *]* to symbolize the beginning and end of a word, respectively, we computed all the possible letter-ngrams with a maximum length of 3 on all the datasets. Therefore, we selected around 12,500 letter-ngrams. The large difference in the number of ngrams comes from the choice of keeping only letter-ngrams appearing in at least 2 different datasets: a *joker* was created to replace the non-selected letter-ngrams. With our example in Figure 1, the name *Sophie* is thus decomposed as {S,o,p,[S,So,op,[So ,. . . ,ie],e]}.

Regarding the choice of having at most 3-grams, this allows to give the system an idea on the order in which the characters are arranged in the word. In the case where there are only unigrams, the learning process is longer and tends towards over-learning to be able to have sequences of characters forming words. Including 4-grams into the vector would dramatically increase its length and would make it more difficult to select the ngrams recognized in a word by the optical model.

Decoder Input and Output The input of the decoder input consists of the normalized frequencies of the n-grams in the input word. The frequencies are normalized by the length of the word to provide a vector with values between 0 and 1, corresponding to probabilities for each n-gram. Using frequencies enables us to hold information about the word length and to compensate the lost temporality. For the sequence generation, we have 79 output units for all the upper and lower characters, the digits, the punctuation symbols including the space, the start and the end of word. The last added symbol is a *blank* label that allows the system to provide an out-of-character answer after the end of a word.

4 Experimental Setup

This section presents the experimental setup of our decoder as well as the evaluation metrics we used.

4.1 Decoder Training

The size of the datasets varies from 660k to 25k words. For this reason, we trained the system with either one dataset or several datasets, except for GW. Its size is too small so we always combined it with other datasets.

To avoid overfitting, we used the classical technique of early stopping. The training process was stopped after five epochs if the validation loss did not decrease anymore. For the sequence generation, we used the loss defined by Equation (1) that computes the multi-classification with a normalization through all the output units.

$$L_i = -\sum_j t_{i,j} \log(p_{i,j}) \tag{1}$$

4.2 Decoder Parameters

We defined the structure of the decoder part with 1,024 units in the fully connected layer to extract features, 500 hidden units in the GRU layer, and 79 units with the Softmax activation function on the last

layer. To be sure to stay in the same configuration, we used the Adam function that controls the learning rate which is initialized at 0.0001 for the stochastic optimization. As we have less data than the others experiments of the state-of-the-art, our rate is lower. In the scope of sequence generation, we padded the length of the sequence with blank labels up to 50 characters. Therefore, the network is free to define any sequence without any constraints.

4.3 Evaluation Metrics

To evaluate the performance of our system, we used the recognition rate at the character level (*CRR*) and at the word level (*WRR*). The Character Recognition Rate is defined as:

$$\text{CRR} = \frac{N - (Ins + Subs + Dels)}{N}$$

with N the number of characters of the reference words, $Subs$ the number of character substitutions, $Dels$ the number of character deletions, and Ins the number of character insertions. The Word Recognition Rate corresponds to the Word Accuracy and is defined as the number of correctly recognized words divided by the total number of words to recognize. The WRR is computed with and without using the Levenshtein edit distance to correct the output sequence with a multilingual dictionary. The dictionary was built from the training and the validation parts of the vocabulary of all the datasets, except Wikipedia. The dictionary contains 39,051 words.

We also computed the lexical coverage of a training set with respect to the test set. It is defined as the number of words shared between the training set and the test set, normalized by the size of the test vocabulary. This metric defines a high boundary for the word recognition using a dictionary. Since the dictionary is built on the training and the validation sets, out-of-vocabulary words of the test set could not be correctly recognized.

4.4 Baseline Approach

As a baseline approach we used a vector of unigrams to represent each word in the decoder input. We added the symbols for the beginning and the end of words so each word is represented by 2 bigrams and several unigrams.

5 Results and Analysis

In this section, we present the results of our experiments for the sequence generation decoder, in Table 2 which is divided into 3 parts following the test resources. We performed three types of experiments corresponding to the sub-parts of the table, for each validation and test data:

1. the same resource for the training and the test parts (training domain = target domain);
2. adding other resources during the training part (training domain $>$ target domain);
3. different resources for the training and the test parts, *i.e.* transductive transfer learning (training domain $\neq$ target domain).

The experiments are done with the artificial true n-gram vectors so the input is not noisy. Overall, the results in Table 2 prove that using letter-ngrams is a better choice than only using unigrams and the use of a dictionary degrades the results. In all the experiments, the models provide a CRR greater than 75% on the validation sets and greater that 85% on the test sets, even if the lexical coverage is less than 25%. Furthermore, WRR also exceed the lexical coverage. To decode the ICR vocabulary, the model must be trained with at least GIC as a resource. All following analyses are presented on the validation sets to evaluate the best ressources to train the model. The last analysis relates to the test sets.

Baseline Whether experimenting with RM on RM (validation and test set) or with GIC on GIC (validation set) and ICR (test set), we found that the results obtained with letter-ngrams are better than with unigrams. On GIC, letter-ngrams dramatically outperformed unigrams with a 70% relative increase of WRR without the dictionary. It should also be noted that only 5% of the characters were incorrectly recognized with letter-ngrams. These results corroborate other studies using trigrams such as (Vania and Lopez, 2017).

Test	Train	Expe. Id	Letter ngrams	Lexical Coverage (%)	Validation			Test		
					CRR	WRR	WRR dict.	CRR	WRR	WRR dict.
ICR	GIC	1	1	65.57	69.11	13.05	14.65	69.27	14.54	10.83
	GIC	2	1,2,3		95.85	77.83	66.47	97.10	86.22	39.30
	GIC+RM	3	1,2,3	67.53	95.97	78.78	66.17	**97.27**	**86.57**	**39.23**
	GIC+ESP	4	1,2,3	65.83	**96.49**	**85.87**	**67.22**	96.96	81.46	39.09
	GIC+ESP+GW+RM	5	1,2,3	67.65	95.99	77.86	66.37	95.85	79.65	38.25
	RM	6	1,2,3	14.52	74.69	19.49	23.32	79.70	30.42	17.27
	RM+ESP+GW	7	1,2,3	23.39	78.11	25.36	28.44	83.68	40.21	23.99
	Wiki	8	1,2,3	0.0	**83.67**	**41.18**	**42.75**	**87.32**	**41.40**	**25.24**
RM	RM	9	1	75.09	82.66	37.83	27.58	83.97	43.07	28.49
	RM	10	1,2,3		93.95	79.93	37.45	94.72	79.50	37.78
	GIC+RM	11	1,2,3	83.83	**97.32**	**89.04**	**39.94**	**98.25**	**92.0**	**40.49**
	GIC+ESP+GW+RM	12	1,2,3	83.95	96.8	86.18	39.43	96.22	80.73	39.14
	GIC	13	1,2,3	58.55	**93.93**	**75.48**	**36.56**	**95.51**	**81.53**	**38.58**
	GIC+ESP	14	1,2,3	59.04	93.89	74.33	36.05	95.46	80.61	38.15
	Wiki	15	1,2,3	0.0	87.04	65.99	33.31	90.36	67.57	35.20
ESP	GIC+ESP	16	1,2,3	85.94	**98.25**	**91.61**	**62.92**	**98.57**	**91.11**	56.51
	GIC+ESP+GW+RM	17	1,2,3	86.10	98.14	90.48	61.94	98.40	90.79	**57.62**
	GIC	18	1,2,3	15.96	88.18	54.35	45.48	91.68	**65.87**	44.76
	RM	19	1,2,3	7.27	67.82	14.03	8.87	72.83	18.25	12.70
	GIC+RM	20	1,2,3	17.37	87.4	41.77	**51.61**	**92.05**	64.13	**46.51**
	GIC+RM+GW	21	1,2,3	17.69	**88.77**	**57.1**	44.35	91.68	64.60	44.60
	Wiki	22	1,2,3	0.0	78.89	27.26	24.52	84.52	34.28	32.38

Table 2: Sequence generation results. CRR and WRR with/without the dictionary on the Italian Comedy Records (ICR), Rimes (RM), and Los Esposalles (ESP) validation and test datasets. Several experiments are presented: evaluation of the baseline, combination of the resources, and transductive transfer learning with all the resources. The lexical coverage is computed between the training and the test datasets.

Use of a Dictionary Using the dictionary drops performances except for Expe. Id 6, 7, 8, and 20. We notice that when the lexicon coverage is greater than 20%, WRR always remains lower. We notice that a dictionary misleads the decoder when it generates a correct sequence but there is only another form (e.g., plural) of this word, in the dictionary. However, the dictionary sometimes helps to get closer to the original word even if this word does not exist in the dictionary and if the training and the test data are from different periods or different languages. Nevertheless, these cases are too rare to improve CRR with the dictionary.

Results on ICR (our Target Data) We focus on the central part of the ICR experiments, where GIC is combined with other resources. The CRR and WRR results are very similar to those of the training step with only GIC (Expe. Id 2 vs. Expe. Id 3, 4, and 5). Therefore, the increase in the number of resources does not always have a great impact on the performance. Nonetheless, training on GIC and ESP achieved better than GIC and RM despite the fact that languages are mixed.

For the transfer learning process with an out-of-domain vocabulary (Expe. Id 6, 7, and 8), the lexical coverage is extremely low (around 15%). Nevertheless, the system reaches a maximum of 19.49 % WRR using only RM and 41.18 % WRR using Wikipedia, which are the two modern French resources.

Results on Rimes (RM) These results are impressive because Rimes is the only modern image base we used. For transductive transfer learning, without RM during the training step, CRR and WRR reach the same results obtained with only RM as a training data. Moreover, they exceed the lexical coverage of 15%. When we work with historical training data applied to modern data but with the same language, the lexical coverage is greater than using a modern training data and testing on ICR. It seems that the historical spelling is easier to extend to the modern spelling for the decoder. The results obtained with Wikipedia show a CRR 6.89% lower and a WRR 9.49% lower than with GIC, despite the fact that the

vocabulary used in RM and Wikipedia is modern French.

Results on Los Esposalles (ESP) The vocabulary of Los Esposalles is built primarily with named entities as it is extracted from 18^{th} century Spanish wedding registers. This explains the low lexical coverage for the transductive transfer learning. Nevertheless, CRR is greater than 90% except with RM (Expe. Id 19) and WRR outperforms the lexical coverage. We did not experiment the decoding with unigrams on this resource because there are too few words. Expe. Id 21 using GIC, GW, and RM shows that this decoder enables to learn a representation from one language (mainly in French) to another.

Analysis on Test Sets Finally, creating and using a new dataset in the field of the Italian Comedy is an interesting approach to decode ICR. This enables a better word reconstruction since the lexical coverage is increased by more than 20%. Among the unknown but well-recognized words can be found light abbreviations such as *"arleq."* instead of *"Arlequin"*. Contrary to the validation results, the best results are obtained thanks to the combination with RM dataset (Expe. Id 3 versus 4). This can be explain by the difference of the vocabulary, the validation set is composed of GIC while the test set is composed of ICR. For the transductive transfer learning process, the results with Wikipedia are similar ($+1.20\%$) to using mixed resources with less data (Expe. Id 7 versus Expe. Id 8) contrary to the results on the validation set, where there is a significant difference around 15% on WRR with and without dictionary.

Error Analysis Table 3 shows some mistakes made by the decoder. For words with a repeated character such as *"cavalcade"* and *"clemence"*, it is common for the system to fail to generate the characters inserted between each repetition. Common mistakes are a permutation between two consecutive characters such as *"suitte"* and one doubled character to replace another one. In the last example, *"Soldat"*, the decoder predicts two start symbols: the second one replaces the first letter of the word. This happens when small resources are used to train the system.

Table 4 shows the distribution of the different types of errors presented in Table 3 for 3 selected experimentations (Expe. Id 2, 5, and 8). The most common mistake is a simple switch between two characters: this represents 65.14% to 88.32% of the errors, depending on the considered experiments. For the experiments including GIC in the training step, the redundant character errors and wrong starting character are similar. They represent only 10% and 20% of all the identified errors. Expe. Id 8 has a high rate of wrong starting characters that can be explained by the lack of lexical coverage between Wiki and ICR. Despite the equivalent quantity of words used for the training step with Wiki than with GIC, the system has difficulties finding out the start of words. Indeed, Wiki has been preprocessed and normalized by removing capitalization so the information seems to be important for the system.

To conclude, we can see that, in Expe. Id 2, there is on average 1.14 character error by word while, in Expe. Id 8, there is on average 1.73 character error by word: that can be explained by the 45.8% drop of WRR between these two experiments.

Type Error	Expe. Id	Correct Word	Retrieved Word
Multiple Char.	4	cavalcade	cacaadade
	3	clemence	ccceeene
Swich Char.	6	suitte	usitte
	5	belle	blll
Starting Char.	5	[diverstissemens]	ddevvestissemens]
	6	Soldat	[ollat

Table 3: Examples of mistakes made by our decoder.

Expe. Id	% Error Redundant Character	% Error Switch	% Error Starting Char.
2	9.13	81.22	9.65
5	4.81	88.32	6.87
8	8.44	65.14	26.42

Table 4: Quantitative analysis of the different error types on 3 ICR results.

6 Conclusion and Future Work

In this paper, we presented a transfer learning approach dealing with the lack of ground truth of our Italian Comedy data, for sequence generation. We selected some available resources with the same domain and from the same time period to achieve the transfer learning. We chose an approach that did not use

an indication of the order of characters in words rather than usual sequence-to-sequence approaches to facilitate the transfer learning.

Our results show that a character-based decoder can infer a word from a letter-ngrams vector. Moreover, the lexical coverage between the training and test data is reached by the word recognition rate and mostly exceeded without the dictionary. The dictionary misleads the decoder rather than helping it because only some forms of a word are included in the dictionary. We demonstrated that transfer learning is more efficient from historical spelling to modern vocabulary than the opposite. The difference in languages implies a low coverage of the words but it still enables a correct decoding of the words as long as the resources come from the same time period. Therefore, the letter-ngrams models are efficient across languages sharing the same alphabet. Nevertheless, the decoder encounters some difficulties with specific words. We might be wondering whether the letter-ngram size limit of 3 is large enough, or whether a higher amount of training data could solve this issue. Despite this, our simple decoder built with only 4 layers obtained good results. This strengthens our idea of looking for new untapped resources instead of relying on resources traditionally used such as Wikipedia.

Currently, we are carrying out experiments with this approach at the word-level only and without punctuation marks. The next step for the decoder would be to extend it to the sentence-level, more specifically to deal with the space character. Initially, we aim at implementing a new handwriting recognition system with an encoder-decoder model. We consider that the decoder presented in this article is operational. Our future work is going to look at the encoder components independently. The main difficulty could be to handle the errors by encoding them in the vector of letter-ngrams. Indeed, unlike the experiments carried out in this paper, the input data of the decoder will be noisy.

References

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate.

Samy Bengio and Georg Heigold. 2014. Word embeddings for speech recognition. In *Proceedings of the 15th Annual Conference of the International Speech Communication Association (Interspeech'14)*, Singapore.

Théodore Bluche, Jérôme Louradour, and Ronaldo Messina. 2017. Scan, Attend and Read: End-to-End Handwritten Paragraph Recognition with MDLSTM Attention. In *Proceedings of the 14th International Conference on Document Analysis and Recognition (ICDAR'17)*, Kyoto, Japan.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*, 5(1):135–146.

Marcel Bollmann, Joachim Bingel, and Anders Søgaard. 2017. Learning attention for historical text normalization by learning to pronounce. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL'17)*, pages 332–344, Vancouver, Canada, July.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of the 8th Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST'14)*, pages 103–111, Doha, Qatar.

Florence Cloppet, Véronique Eglin, Van Cuong Kieu, Dominique Stutzmann, and Nicole Vincent. 2016. ICFHR2016 Competition on Classification of Medieval Handwritings in Latin Script. In *Proceedings of the 15th International Conference on Frontiers in Handwriting Recognition (ICFHR'16)*, pages 590–595, Shenzhen, China.

Andreas Fischer, Andreas Keller, Volkmar Frinken, and Horst Bunke. 2012. Lexicon-free handwritten word spotting using character HMMs. *PRL*, 33(7):934–942.

Volkmar Frinken, Andreas Fischer, and Carlos-D Martínez-Hinarejos. 2013. Handwriting recognition in historical documents using very large vocabularies. In *Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing*, pages 67–72. ACM.

Dan Garrette and Hannah Alpert-Abrams. 2016. An unsupervised model of orthographic variation for historical document transcription. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HTL'16)*, pages 467–472, San Diego, CA, USA.

Emilio Granell, Edgard Chammas, Laurence Likforman-Sulem, Carlos-D Martínez-Hinarejos, Chafic Mokbel, and Bogdan-Ionuţ Cîrstea. 2018. Transcription of spanish historical handwritten documents with deep neural networks. *Journal of Imaging*, 4(1):15.

Adeline Granet, Benjamin Hervy, Geoffrey Roman-Jimenez, Marouane Hachicha, Emmanuel Morin, Harold Mouchère, Solen Quiniou, Guillaume Raschia, Francoise Rubellin, and Christian Viard-Gaudin. 2018a. Crowdsourcing-based Annotation of the Accounting Registers of the Italian Comedy. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan.

Adeline Granet, Emmanuel Morin, Harold Mouchre, Solen Quiniou, and Christian Viard-Gaudin. 2018b. Transfer learning for handwriting recognition on historical documents. In *Proceedings of the 7th International Conference on Pattern Recognition Applications and Methods ICPRAM*, pages 432–439.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *Proceedings of the 23rd International Conference on Machine Learning (ICML'06)*, pages 369–376, Pittsburgh, PA, USA.

Emmanuele Grosicki and Haikal El-Abed. 2011. ICDAR 2011 - French Handwriting Recognition Competition. In *Proceedings of the 11th International Conference on Document Analysis and Recognition (ICDAR'11)*, pages 1459–1463, Beijing, China.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM.

Josep Lladós, Marçal Rusiñol, Alicia Fornés, David Fernández, and Anjan Dutta. 2012. On the influence of word representations for handwritten word spotting in historical documents. *IJPRAI*, 26(05):1263002–1–25.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th international conference on machine learning (ICML'10)*, pages 807–814, Haifa, Israel.

Hideki Nakayama and Noriki Nishida. 2017. Zero-resource machine translation by multimodal encoder–decoder network with multimedia pivot. *Machine Translation*, 31(1-2):49–64.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Ioannis Pratikakis, Konstantinos Zagoris, George Barlas, and Basilis Gatos. 2016. ICFHR2016 Handwritten Document Image Binarization Contest (H-DIBCO 2016). In *Proceedings of the 15th International Conference on Frontiers in Handwriting Recognition (ICFHR'16)*, pages 619–623, Shenzhen, China.

Joan Andreu Sanchez, Veronica Romero, Alejandro H Toselli, Mauricio Villegas, and Enrique Vidal. 2017. ICDAR2017 Competition on Handwritten Text Recognition on the READ Dataset. In *Proceedings of the 14th International Conference on Document Analysis and Recognition (ICDAR'17)*, pages 1383–1388, Kyoto, Japan.

Clara Vania and Adam Lopez. 2017. From Characters to Words to in Between: Do We Capture Morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL'17)*, pages 2016–2027, Vancouver, Canada.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 3156–3164. IEEE.

SMHD: A Large-Scale Resource for Exploring Online Language Usage for Multiple Mental Health Conditions

Arman Cohan[1] * **Bart Desmet**[1,2] * **Andrew Yates**[1,3] *
Luca Soldaini[1] **Sean MacAvaney**[1] **Nazli Goharian**[1]

[1]IR Lab, Georgetown University, US
{firstname}@ir.cs.georgetown.edu

[2]LT3, Ghent University, BE
bart.desmet@ugent.be

[3]Max Planck Institute for Informatics, DE
ayates@mpi-inf.mpg.de

Abstract

Mental health is a significant and growing public health concern. As language usage can be leveraged to obtain crucial insights into mental health conditions, there is a need for large-scale, labeled, mental health-related datasets of users who have been diagnosed with one or more of such conditions. In this paper, we investigate the creation of high-precision patterns to identify self-reported diagnoses of nine different mental health conditions, and obtain high-quality labeled data without the need for manual labelling. We introduce the SMHD (Self-reported Mental Health Diagnoses) dataset and make it available. SMHD is a novel large dataset of social media posts from users with one or multiple mental health conditions along with matched control users. We examine distinctions in users' language, as measured by linguistic and psychological variables. We further explore text classification methods to identify individuals with mental conditions through their language.

1 Introduction

Mental health is a significant challenge in healthcare. Mental disorders have the potential to tremendously affect the quality of life and wellness of individuals in society (Strine et al., 2008; Mowery et al., 2017a). Social media have become an increasingly important source of data related to mental health conditions (Cohan et al., 2017; Mowery et al., 2017b; Coppersmith et al., 2017; Yates et al., 2017), as it is now a prominent platform for individuals to engage in daily discussions, share information, seek advice and simply communicate with peers that have shared interests. In addition to its ubiquity and ease of access, the possibility to disclose mental health matters anonymously or pseudo-anonymously further drives users to online self-disclosure.

At the same time, the close connection between language and mental health makes social media an invaluable resource of mental health-related data. Lack of data has been one of the key limitations to understanding and addressing the challenges the domain is facing (Coppersmith et al., 2014a). Data from social media can not only be used to potentially provide clinical help to users in need, but also to broaden our understanding of the various mental health conditions. Social media analysis has already been proven valuable for identifying depression (Coppersmith et al., 2014a; Yates et al., 2017), suicide ideation (Cohan et al., 2017; De Choudhury and Kıcıman, 2017; Kshirsagar et al., 2017; Desmet and Hoste, 2018), and other conditions such as schizophrenia (Mitchell et al., 2015). While social media data is abundantly available, the amount of labeled data for studying mental health conditions is limited. This is due to the high cost of annotation and the difficulty of access to experts.

Prior research has investigated self-disclosure as a means of obtaining labeled data from social media. De Choudhury et al. (2013a) used it to identify new mothers and track post-partum changes in emotions.

* Equal contribution.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1485–1497
Santa Fe, New Mexico, USA, August 20-26, 2018.

Condition	Twitter, (Coppersmith et al, 2015)	Reddit, SMHD (ours)
ADHD	102	10,098
Anxiety	216	8,783
Autism	n/a	2,911
Bipolar	188	6,434
Borderline	101	n/a
Depression	393	14,139
Eating	238	598
OCD	100	2,336
PTSD	403	2,894
Schizophrenia	172	1,331
Seasonal Affective	100	n/a

Table 1: Comparison between the number of self-reported diagnosed users per condition in the dataset of Coppersmith et al. (2015a) and ours (SMHD).

I was officially diagnosed with ADHD last year.
I have a diagnosed history of PTSD.
my dr just diagnosed me as schizo.

Figure 1: Examples of self-reported diagnoses statements.

Coppersmith et al. (2014a) specifically focused on self-reports of mental health diagnoses. In particular, Coppersmith et al. (2015a) constructed a dataset of various mental health conditions using Twitter statements. Finally, Yates et al. (2017) introduced a large dataset of depressed users obtained from Reddit.

We extend the previous efforts on addressing the lack of large-scale mental health-related language data. Particularly, we propose improved data collection methods through which we can obtain high-quality large-scale datasets of labeled diagnosed conditions paired with appropriate control users. Consequently, we introduce SMHD (Self-reported Mental Health Diagnoses), a large dataset of diverse mental health conditions that can provide further insight into the mental health-related language. We leverage self-reported diagnosis statements where a user declares to have been diagnosed with a mental health condition such as depression. Examples of self-reported diagnoses are shown in Figure 1. Our dataset can be used not only to develop methods for better identifying mental health conditions through natural language, but also allows us to investigate the characteristics of language usage within each condition. We hope the availability of this new resource will foster further research into these problems and enhance reproducibility of suggested approaches.

Our work has the following significant distinctions compared to existing social media datasets related to mental health. Previous work has studied self-reported diagnosis posts in Twitter (Coppersmith et al., 2015a), where the post length is limited to 140 characters.[1] This makes the Twitter language use rather different from real life discussions. Instead, we use data from Reddit, an interactive discussion-centric forum without any length constraints. Our dataset contains up to two orders of magnitude more diagnosed individuals for each condition than the Twitter dataset by Coppersmith et al. (2015a), making it suitable for exploring more recent data-driven learning methods (see Table 1). We choose our control users in a systematic way that makes classification experiments on the dataset realistic.

We normalize language usage between the users: by removing specific mental health signals and discussions, we focus on patterns of language in normal (general) discussions. While our dataset creation method is close to Yates et al. (2017), we extend theirs by investigating multiple high-precision matching patterns to identify self-reported diagnoses for a range of conditions. Part of our patterns are obtained through synonym discovery. Considering relevant synonyms from reliable sources increases the variety of the diagnosed users and linguistic nuances. We also explore nine common mental health

[1]This limitation was doubled to 280 characters in late 2017.

conditions while Yates et al. (2017) focus only on depression. We explore classification methods for identifying mental health conditions through social media language and provide detailed analysis that helps us understand the differences in language usage between conditions, and between diagnosed users and controls.

Our contributions are as follows: *(i)* We investigate the creation of high-precision matching patterns to identify self-reported diagnoses of nine different mental health conditions. *(ii)* We introduce a large-scale dataset of nine mental health conditions that has significant extensions to existing datasets and we make our data publicly available. Our dataset includes users who might suffer from more than one condition, thus allowing language study of interacting mental conditions. *(iii)* We investigate language characteristics of each mental health group. *(iv)* We explore classification methods for detecting users with various mental health conditions.

2 Related work

Social media offers a considerable amount of accessible common language, attracting the attention of those who study the language of individuals with mental health conditions. Twitter is a natural source, being a popular platform that enables users to share short messages publicly. Early work used crowd-sourcing to identify Twitter users who report a depression diagnosis in a survey, and proposed features that are able to identify depressed users prior to the onset of depression (De Choudhury et al., 2013b). Others found that these characteristics hold in both English and Japanese tweets (Tsugawa et al., 2015), indicating similar cross-cultural tendencies.

Due to the cost and bias introduced by relying on surveys, work shifted to identifying mental health conditions by examining the content shared by social media users. Coppersmith et al. (2014a) identified approximately 1,200 Twitter users with 4 mental health conditions (bipolar, depression, PTSD, SAD) using diagnosis statements found in tweets (e.g., "I was diagnosed with depression"). Following this work, detailed studies were conducted on users experiencing PTSD (Coppersmith et al., 2014b), and schizophrenia (Mitchell et al., 2015; Ernala et al., 2017). The shared task at the 2nd Computational Linguistics and Clinical Psychology Workshop (CLPsych 2015) focused on identifying depression and PTSD users on Twitter (Coppersmith et al., 2015b). This included a set of approximately 1,800 Twitter users with self-identified diagnoses. Leading submissions to CLPsych 2015 relied on the LIWC lexicon (Pennebaker et al., 2015), topic modeling, manual lexicons, and other domain-dependent features (Resnik et al., 2015; Preotiuc-Pietro et al., 2015). Coppersmith et al. (2015a) expands the research on Twitter to eleven self-identified mental health conditions. (Benton et al., 2017) uses this dataset (and others) with a neural multi-task learning approach to identify language characteristics. Mowery et al. (2016) investigates specific symptoms of depression in tweets, including depressed mood, disturbed sleep, and loss of energy.

While the abundant short texts of Twitter can provide some insight into language characteristics of those with mental health conditions, long-form content can provide additional linguistic insights. Some have investigated the language of users of an online crisis forum to identify posts of users who are at highest risk to allow for faster intervention (Milne et al., 2016; Cohan et al., 2017). Losada and Crestani (2016) applied the self-reported diagnosis strategy to identify approximately 150 Reddit users who suffer from depression, and paired them with 750 control users. Yates et al. (2017) also used self-reported diagnoses to identify clinically depressed users, but applied it to a larger set of Reddit, yielding the Reddit Self-reported Depression Diagnosis (RSDD) dataset of over 9,000 users with depression and over 100,000 control users (using an improved user control identification technique). The corpus was also used to study the temporal aspects of self-reported diagnoses (MacAvaney et al., 2018).

Others have used data sources beyond social media to examine the language of people with mental health conditions. Resnik et al. (2013) uses topic models to predict depression and neuroticism based on student-written essays, finding clear clusters of words when students are asked to write about their feelings in a stream-of-consciousness setting. Althoff et al. (2016) uses text message conversations from a mental health crisis center to improve counseling techniques.

This work addresses important limitations of previous efforts. Similar to RSDD (Yates et al., 2017), we

build our corpus from Reddit using self-reported diagnoses. This results in a large amount of long-form post content that is not constrained by a character limit. Furthermore, because there are no character limits (as exist for Twitter), the mode of language is more typical of general writing. Unlike Yates et al. (2017), we investigate extended self-diagnoses matching patterns derived from mental health-related synonyms. We also focus on nine mental health conditions (rather than just a single condition). This results in a collection that can be used to compare and contrast the language characteristics of each condition.

3 Data

In this section we describe the construction and characteristics of the Self-reported Mental Health Diagnoses (SMHD) dataset. The studied conditions correspond to branches in the DSM-5 (American Psychiatric Association, 2013), an authoritative taxonomy for psychiatric diagnoses. Six conditions are top-level DSM-5 disorders: schizophrenia spectrum disorders (*schizophrenia*), bipolar disorders (*bipolar*), depressive disorders (*depression*), anxiety disorders (*anxiety*), obsessive-compulsive disorders (*ocd*) and feeding and eating disorders (*eating*). The three other conditions are one rank lower: post-traumatic stress disorder (*ptsd*) is classified under trauma- and stress-related disorders, and autism spectrum disorders (*autism*) and attention-deficit/hyperactivity disorder (*adhd*) under neurodevelopmental disorders. We use these lower-rank conditions to provide more definition when they are clearly distinguishable from sibling disorders.

3.1 Dataset construction

The SMHD dataset was created by using high precision patterns to identify Reddit users who claimed to have been diagnosed with a mental health condition (*diagnosed users*) and using exclusion criteria to match these diagnosed users with control users who are unlikely to have one of the mental health conditions studied (*control users*). SMHD consists of user labels indicating the mental health condition(s) associated with each user and all Reddit posts made by each user between January 2006 and December 2017 (inclusive). Users and posts were extracted from a publicly available Reddit corpus[2]. This approach is based on the method used to create the Reddit Self-reported Depression Diagnosis (RSDD) dataset (Yates et al., 2017). SMHD expands on RSDD by incorporating synonyms in matching patterns and including diagnoses for eight new conditions in addition to depression.[3]

Diagnosed users were identified using high precision diagnosis patterns in a manner similar to that used in prior work that studied depression on Reddit (Yates et al., 2017); we describe these patterns in more detail in section 3.2. After identifying candidate diagnosed users who matched a diagnosis pattern (see Figure 1), we removed any candidates who *(1)* matched a negative diagnosis pattern[4] or *(2)* had fewer than 50 posts talking about topics other than mental health (*mental health posts*). This is done to ensure enough data remains for a diagnosed user after removing their mental health-related content. After removing these candidate diagnosed users, 36948 diagnosed users remain.

Mental health posts were defined as posts that were either made to a subreddit (i.e., a subforum devoted to a specific topic) related to mental health or that included language related to mental health, such as the name of a condition (e.g., *OCD*) and general terms like *diagnosis*, *mental illness*, or *suffering from*. We constructed a list of subreddits related to mental health by beginning with lists from prior work studying depression on Reddit (Pavalanathan and De Choudhury, 2015; Yates et al., 2017) and expanding them to include discussion and support subreddits for each of the other mental health conditions. All mental health posts are removed for diagnosed users and control users alike. Classification therefore happens on the set of posts that do not contain any of the mental health terms, and that have not been posted in any of the mental health-related subreddits. Our methodology does not guarantee, however, that all potentially relevant terms or subreddits have been excluded.

Control users were chosen from a pool of candidate control users based on their similarity with

[2] `https://files.pushshift.io/reddit/`

[3] Patterns are available from `http://ir.cs.georgetown.edu/data/smhd/`.

[4] e.g., *I was never clinically diagnosed.*

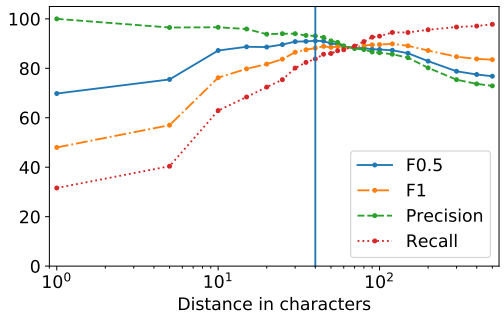


Figure 2: Precision, recall, F1 and F0.5 of the condition diagnosis patterns as a function of the maximum allowable distance (in characters) between a diagnosis and a condition keyword. Chosen threshold indicated at 40 characters.

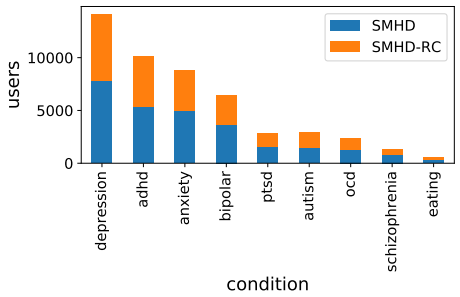


Figure 3: Number of users per condition for SMHD (in blue) and SMHD-RC (in orange).

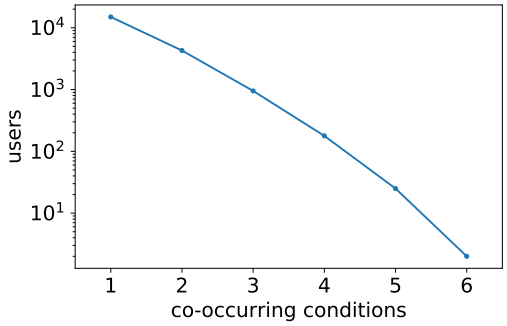


Figure 4: Number of users with a single or multiple co-occurring conditions.

Figure 5: Concomitance of condition diagnoses. Rows contain relative co-occurences of the row condition with other conditions.

the diagnosed users as measured by their number of posts and the subreddits they posted in. This is done to prevent biases between the control and diagnosed groups in the dataset and prevent making the task of identifying such users artificially easy. In more detail, every Reddit user who *(1)* had no mental health post[5] and *(2)* had at least 50 posts became a candidate control user. Given this pool of candidate control users, we matched each diagnosed user with 9 candidate control users (on average) after excluding controls who had never posted in the same subreddit as the diagnosed user or who had more than twice as many posts or less than half as many posts as the target diagnosed user.

The selection criteria for potential control users are stringent: users are removed if they do not have the required subreddit overlap or minimum post count, or if they use any of the mental health-related terms in any of their posts. The latter is necessary because we found that the prior probability of using an exclusion term in any given subreddit (e.g. *depressed*) is almost always higher for diagnosed users than for controls, even in popular subreddits on general topics (e.g. *r/politics*). As a result, the pool of candidate controls for matching to diagnosed users is significantly diminished. Because control users are picked from the pool without replacement, we were unable to meet the target of 9 appropriate controls for some of the diagnosed users. We release the users who have at least 9 control users as SMHD, and use this

[5]When defining language related to mental health, we consider *add* to be a mental health term. However, this term is often used as a verb that does not refer to Attention Deficit Disorder. We therefore do not exclude candidate control users who have used this term. We instead remove any post containing the term *add*, which matches our treatment of the diagnosed users where all mental health posts are removed.

dataset for all analyses in this paper. It contains 20,406 diagnosed users and 335,952 matched controls. A secondary dataset containing the remaining 16,542 diagnosed users with fewer than 9 controls will be made available as SMHD-RC (*Relaxed Controls*) for studies that require additional data for analysing differences between mental health conditions, rather than between diagnosed users and controls.

3.2 Diagnosis patterns

To identify Reddit users who report a diagnosis for one or more mental health conditions, we developed detection patterns with a focus on high precision. The patterns consist of two components: one that matches a self-reported diagnosis, and another that maps relevant diagnosis keywords to the 9 mental health conditions. A user is included for a condition if one of the condition keywords occurs within a certain distance of the diagnosis pattern (as discussed below).

For each condition, a seed list of diagnosis keywords was collected from the corresponding DSM-5 headings. To increase the likelihood of matching diagnostic posts, we expanded each set of medical expressions to include synonyms, common misspellings, vernacular terms and abbreviations. Our steps mirror the ones of Soldaini and Yom-Tov (2017), who were also interested in identifying self-diagnosed users, albeit on query logs. In particular, we leveraged two synonym mappings to generate alternative formulations of the disorders of interest:

- *MedSyn* (Yates and Goharian, 2013) is a laypeople-oriented synonym mapping ontology. It was generated from a subset of *UMLS*[6] filtered to remove irrelevant semantic types.
- *Behavioral* (Yom-Tov and Gabrilovich, 2013) maps expressions commonly used by laypeople to describe their medical condition to concepts in UMLS. The synonyms were generated by first identifying the most frequent search terms Yahoo! search used to retrieve Wikipedia medical pages. Then, frequent lexical affinities (Carmel et al., 2002) to the aforementioned were added to synonyms lists.

The resulting expansions were vetted manually to remove terms that did not describe the condition precisely (e.g., hypernyms) or that were ambiguous (e.g., "add"), and missing items were added.

Using the diagnosis patterns without condition keywords, all self-reported diagnosis posts were collected from the Reddit dataset, and 500 posts were randomly sampled for manual annotation. False positive matches were found in 18 of those, i.e. the precision for self-diagnosis detection was 96.4%. False positives included negations, hypotheticals (e.g., "I'm not sure if I was diagnosed"), and diagnoses that are uncertain or wrong ("I was diagnosed with autism, then undiagnosed."). Out of 500 diagnosis posts, 241 reported a mental health condition. The ones that were annotated as belonging to one of the 9 conditions were used to tune the condition term lists and the optimal distance between a diagnosis pattern and a condition term. Figure 2 plots the effect of this distance on precision, recall, F1 and F0.5 (which emphasizes precision). A maximum distance of 40 characters was chosen, where F0.5 score is highest. We thus achieve high precision (93%) with good recall (84%). Since the optimal distance threshold was tuned on this development set, it may overfit the data and the reported scores should be considered a ceiling performance.

To validate the final diagnosis matching approach on a held-out set, 900 posts (corresponding to 9 samples of 100 diagnosis posts, one for each condition) was manually checked for false positives. We obtain high precision, with a minimum precision of 90% for *anxiety* and macro-averaged precision of 95.8%. Most false positives are caused by terms for a condition occurring close to a diagnosis for another condition (e.g. "My doctor diagnosed me with depression, and I also have an anxiety problem."). While a user might also suffer from these conditions, they are not explicitly reporting a diagnosis.

3.3 Dataset statistics

Figure 3 shows the distribution of diagnosed users per condition in both datasets. Users who self-reported a diagnosis of depression, ADHD, anxiety or bipolar are most common. Interestingly, 26.7% of diagnosed users in the dataset reported more than one diagnosis (Figure 4). Such concomitant conditions are not uncommon, and were also reported in the work of Coppersmith et al. (2015a). As can be seen in

[6]Unified Medical Language System

condition	posts		tokens		characters
	per user	total	per post	total	per post
control	310.0 (157.8)	115,669k	26.2 (48.3)	3,031.6M	133.9 (252.9)
depression	162.2 (84.2)	1,272k	45.1 (80.0)	57.4M	227.5 (406.9)
adhd	164.7 (83.6)	872k	46.5 (82.7)	40.5M	237.5 (433.5)
anxiety	159.7 (83.0)	795k	46.4 (83.0)	36.9M	233.9 (422.8)
bipolar	157.6 (82.4)	575k	45.5 (86.5)	26.2M	230.6 (447.0)
ptsd	160.7 (84.7)	258k	53.1 (114.0)	13.7M	267.8 (581.7)
autism	168.3 (84.5)	248k	46.5 (82.3)	11.6M	237.9 (434.0)
ocd	158.8 (81.4)	203k	46.4 (90.1)	9.4M	234.2 (459.5)
schizophrenia	157.3 (80.5)	123k	49.2 (105.6)	6.1M	253.8 (566.6)
eating	161.4 (81.0)	53k	46.3 (73.7)	2.5M	232.6 (372.8)

Table 2: Average (Stdev.) and count of posts, tokens and characters for diagnosed and control users.

Figure 5, depression co-occurs with high frequency with most of the other conditions, almost 30% of users with depression, OCD or PTSD also suffer from anxiety, and schizophrenia is most likely to be diagnosed alongside bipolar disorder.

An important characteristic of the SMHD dataset is its scale. Reddit allows its users to write long-form posts, so unlike datasets collected on Twitter, a large amount of text is available for each diagnosed user. Table 2 gives an overview of the average and total number of posts, tokens and characters per condition, and for controls. Control users post on average twice as many posts as diagnosed users, but these posts tend to be considerably shorter. Although this may be a valid signal for certain mental health conditions, it can be removed for classification experiments by truncating the length and number of posts. This is common practice for technical reasons, and truncating post length has been shown in previous work to improve classification performance (Yates et al., 2017).

3.4 Ethics and privacy

Even though we rely on publicly available Reddit posts in our work, mental health is a sensitive matter and measures to prevent risk to individuals in social media research should always be considered (Hovy and Spruit, 2016; Šuster et al., 2017). The risks associated with the data collection methods and our resulting SMHD dataset is minimal. We refrain from publicly posting any excerpts of the data, we made no attempt to contact users, and we made no attempt to identify or link users to other social media accounts. We further replace usernames with random identifiers to prevent users' identities from being known without the use of external information. The SMHD dataset is available through a Data Usage Agreement (DUA)[7] protecting the users' privacy. In particular, the DUA specifies that no attempt should be made to publish portions of the dataset (which could result in users being identified), contact users, identify them, or link them with other user information.

4 Analysis

To investigate the differences between the language of mental health condition groups and the control group, we categorize language of users based on measures of psycholinguistic attributes through the LIWC lexicon (Pennebaker et al., 2015). These categories include variables that characterize linguistic style as well as psychological aspects of language (e.g., cognitive attributes and affective attributes). For each user, we obtain LIWC categories based on their posts and then compare these categories across users in each mental health condition group versus the control group using Welch's t-test (Welch, 1947). We adjust p-values with Bonferroni correction. To better see the differences between the categories, we also report the Cohen's d statistic (Cohen, 1988). Table 3 shows the results.

In general, we observe a variety of differences in language use between the diagnosed and the control groups. The effect sizes range from a very small effect to medium ($0.05<d<0.5$), which is typical of large datasets (Sakai, 2016). These differences span across both the linguistic style and psychological

[7] http://ir.cs.georgetown.edu/resources/

LIWC category	depression	adhd	anxiety	bipolar	ptsd	autism	ocd	schizophrenia	eating
Summary Language Variables									
Clout	$-0.06^{\ddagger}$	–	$-0.1^{\ddagger}$	–	–	–	–	–	–
Authentic	$0.2^{\ddagger}$	$0.15^{\ddagger}$	$0.22^{\ddagger}$	$0.18^{\ddagger}$	$0.21^{\ddagger}$	$0.14^{\ddagger}$	$0.23^{\ddagger}$	–	$0.24^{\star}$
WPS	$0.08^{\ddagger}$	$0.1^{\ddagger}$	$0.08^{\ddagger}$	$0.06^{\ddagger}$	$0.1^{\ddagger}$	$0.12^{\ddagger}$	$0.08^{\dagger}$	–	–
Dictionary words	$0.27^{\ddagger}$	$0.22^{\ddagger}$	$0.28^{\ddagger}$	$0.24^{\ddagger}$	$0.31^{\ddagger}$	$0.2^{\ddagger}$	$0.28^{\ddagger}$	$0.22^{\ddagger}$	$0.3^{\ddagger}$
Total function words	$0.27^{\ddagger}$	$0.21^{\ddagger}$	$0.28^{\ddagger}$	$0.24^{\ddagger}$	$0.3^{\ddagger}$	$0.26^{\ddagger}$	$0.28^{\ddagger}$	$0.23^{\ddagger}$	$0.27^{\ddagger}$
Total pronouns	$0.22^{\ddagger}$	$0.14^{\ddagger}$	$0.24^{\ddagger}$	$0.2^{\ddagger}$	$0.25^{\ddagger}$	$0.17^{\ddagger}$	$0.26^{\ddagger}$	$0.18^{\ddagger}$	$0.26^{\dagger}$
Personal pronouns	$0.23^{\ddagger}$	$0.14^{\ddagger}$	$0.26^{\ddagger}$	$0.21^{\ddagger}$	$0.22^{\ddagger}$	$0.14^{\ddagger}$	$0.23^{\ddagger}$	$0.2^{\ddagger}$	$0.27^{\dagger}$
1st pers singular	$0.23^{\ddagger}$	$0.16^{\ddagger}$	$0.28^{\ddagger}$	$0.22^{\ddagger}$	$0.23^{\ddagger}$	$0.17^{\ddagger}$	$0.26^{\ddagger}$	$0.17^{\dagger}$	$0.28^{\dagger}$
3rd pers singular	$0.09^{\ddagger}$	–	$0.1^{\ddagger}$	$0.08^{\star}$	$0.17^{\star}$	–	–	–	–
Impersonal pronouns	$0.06^{\ddagger}$	$0.05^{\star}$	$0.07^{\dagger}$	–	$0.11^{\dagger}$	$0.09^{\star}$	$0.13^{\dagger}$	–	–
Prepositions	$0.12^{\ddagger}$	$0.12^{\ddagger}$	$0.12^{\ddagger}$	$0.12^{\ddagger}$	$0.16^{\ddagger}$	$0.12^{\ddagger}$	$0.11^{\dagger}$	–	–
Auxiliary verbs	$0.12^{\ddagger}$	$0.1^{\ddagger}$	$0.14^{\ddagger}$	$0.11^{\ddagger}$	$0.14^{\ddagger}$	$0.15^{\ddagger}$	$0.13^{\ddagger}$	–	–
Common Adverbs	$0.09^{\ddagger}$	$0.07^{\ddagger}$	$0.08^{\ddagger}$	$0.06^{\star}$	–	–	–	–	–
Conjunctions	$0.17^{\ddagger}$	$0.14^{\ddagger}$	$0.18^{\ddagger}$	$0.16^{\ddagger}$	$0.17^{\ddagger}$	$0.13^{\ddagger}$	$0.11^{\dagger}$	–	$0.3^{\ddagger}$
Other Grammar									
Common verbs	$0.15^{\ddagger}$	$0.11^{\ddagger}$	$0.17^{\ddagger}$	$0.13^{\ddagger}$	$0.15^{\ddagger}$	$0.13^{\ddagger}$	$0.19^{\ddagger}$	–	–
Numbers	$-0.1^{\ddagger}$	$-0.09^{\ddagger}$	$-0.11^{\ddagger}$	$-0.09^{\ddagger}$	$-0.11^{\ddagger}$	–	$-0.1^{\ddagger}$	–	$-0.13^{\dagger}$
Psychological Variables									
Positive emotion	–	–	–	–	–	$-0.08^{\star}$	–	–	–
Anxiety	$0.07^{\dagger}$	–	$0.07^{\star}$	–	–	–	–	–	–
Social processes	$0.11^{\ddagger}$	$0.07^{\ddagger}$	$0.11^{\ddagger}$	$0.1^{\ddagger}$	$0.15^{\ddagger}$	–	–	–	–
Family	$0.06^{\ddagger}$	–	$0.06^{\star}$	–	$0.12^{\dagger}$	–	–	–	–
Female references	$0.13^{\ddagger}$	$0.07^{\ddagger}$	$0.1^{\ddagger}$	$0.13^{\ddagger}$	$0.22^{\dagger}$	–	–	–	–
Cognitive processes	$0.12^{\ddagger}$	$0.13^{\ddagger}$	$0.14^{\ddagger}$	$0.09^{\ddagger}$	$0.12^{\dagger}$	$0.16^{\ddagger}$	$0.13^{\dagger}$	–	–
Insight	$0.09^{\ddagger}$	$0.07^{\ddagger}$	$0.1^{\ddagger}$	$0.08^{\ddagger}$	–	$0.1^{\star}$	$0.17^{\dagger}$	–	–
Discrepancy	–	$0.06^{\star}$	–	–	–	–	–	–	–
Tentative	$0.07^{\ddagger}$	$0.08^{\ddagger}$	$0.08^{\ddagger}$	$0.07^{\star}$	–	–	–	–	–
Differentiation	$0.08^{\ddagger}$	$0.08^{\ddagger}$	$0.1^{\ddagger}$	–	–	$0.09^{\star}$	–	–	–
Biological processes	$0.06^{\ddagger}$	–	–	–	–	–	–	–	–
Health	$0.08^{\ddagger}$	$0.07^{\dagger}$	$0.08^{\ddagger}$	$0.11^{\ddagger}$	–	–	–	–	–
Time orientation									
Past focus	$0.08^{\ddagger}$	$0.05^{\star}$	$0.09^{\ddagger}$	$0.08^{\dagger}$	$0.09^{\star}$	–	$0.11^{\star}$	–	–
Present focus	$0.09^{\ddagger}$	$0.06^{\ddagger}$	$0.1^{\ddagger}$	$0.07^{\star}$	–	–	–	–	–
Personal concerns									
Relativity	$0.05^{\dagger}$	$0.06^{\dagger}$	–	–	–	–	–	–	–
Time	$0.06^{\ddagger}$	–	$0.06^{\star}$	–	–	–	–	–	–
Work	–	$0.06^{\star}$	–	–	–	–	–	–	–
Leisure	$-0.07^{\ddagger}$	$-0.07^{\ddagger}$	$-0.07^{\ddagger}$	$-0.09^{\ddagger}$	$-0.12^{\ddagger}$	$-0.09^{\dagger}$	–	–	–
Money	$-0.06^{\ddagger}$	–	$-0.05^{\ddagger}$	$-0.06^{\ddagger}$	–	$-0.1^{\ddagger}$	–	–	$-0.12^{\ddagger}$
Informal language	$-0.07^{\ddagger}$	$-0.07^{\ddagger}$	$-0.06^{\ddagger}$	–	$-0.12^{\ddagger}$	–	$-0.09^{\star}$	–	–
Netspeak	$-0.07^{\ddagger}$	$-0.06^{\ddagger}$	$-0.06^{\ddagger}$	$-0.06^{\star}$	$-0.1^{\ddagger}$	$-0.08^{\star}$	–	–	$-0.15^{\ddagger}$
Assent	–	–	–	–	$-0.06^{\star}$	–	–	–	–

Table 3: Differences between the linguistic and psycholinguistic variables obtained from LIWC between users treatment groups and control users. Statistical significance is based on t-test with Bonferroni adjustment. Numbers in the cells are effect sizes (Cohen's d). $\star$, $\dagger$, and $\ddagger$ respectively show adjusted p-values of <0.01, <0.001 and <0.0001. This is also indicated by shading colors, red shading shows that the corresponding category is observed with more intensity in the treatment group, where as blue shading shows more significance in the control group. For brevity, only significant attributes are shown.

attributes. Particularly, we observe that clout, an attribute regarding language that indicates high social status, is more prominent among control users compared to the depression and anxiety groups. Related work in psychology shows that depression and anxiety are more prevalent in lower socioeconomic status demographics (Murphy et al., 1991), which is in line with the observation in our dataset. Next, authentic language is stronger among most of the mental health groups compared with control groups. This correlates with the use of personal and first person pronouns. This is backed by prior research showing that fewer self-references are associated with unauthentic language (Newman et al., 2003). We also observe that among most of the conditions the usage of first-person singular pronouns is higher than in control groups. Related studies arrive at the same conclusion that people with mental health conditions tend to use more first-person pronouns and references, possibly because mental health conditions result in greater self-focused attention and rumination (Bucci and Freedman, 1981; Pennebaker et al., 2015; Watkins and Brown, 2002; Van der Zanden et al., 2014).

Social processes, the language related to human interaction and sharing, appears to be higher in depressed, anxiety and PTSD users. A potential explanation for this is that sharing and support-seeking behaviors have been shown to improve an individual's state of mind (Turner et al., 1983; De Choudhury and Kıcıman, 2017). People with conditions such as bipolar, depression and anxiety show significantly

	Depression	ADHD	Anxiety	Bipolar	PTSD	Autism	OCD	Schizophrenia	Eating	Multi-label
Logistic Regression BoW features	**P=85.00**	**P=88.60**	**P=85.80**	**P=87.06**	P=92.44	**P=93.18**	**P=95.00**	**P=100.00**	P=0.00	P=17.22
	R=28.65	R=19.22	R=27.04	R=21.57	R=19.71	R=7.93	R=4.87	R=2.25	R=0.00	R=24.38
	F=42.85	F=31.59	F=41.13	F=34.58	F=32.50	F=14.62	F=9.27	F=4.40	F=0.00	F=19.72
XGBoost BoW features	P=81.40	P=85.71	P=83.52	P=84.94	P=89.83	P=84.09	P=86.96	P=86.84	P=94.12	P=16.87
	R=29.49	R=21.25	R=31.46	R=30.31	R=37.99	R=21.47	R=25.64	R=24.72	R=14.29	R=29.38
	F=43.29	F=34.05	F=45.71	F=44.68	F=53.40	F=34.21	F=39.60	F=38.48	F=24.81	F=20.51
Linear SVM BoW features	P=78.70	P=83.74	P=81.76	P=84.36	**P=93.12**	P=91.19	P=86.29	P=93.06	**P=100.00**	P=19.20
	R=39.75	R=30.69	R=39.88	R=37.21	R=41.22	R=28.05	R=27.44	R=25.09	R=13.39	R=34.86
	F=52.82	F=44.92	F=53.61	**F=51.64**	F=57.14	F=42.90	F=41.63	F=39.53	F=23.62	F=23.71
Supervised FastText	P=66.80	P=62.11	P=67.63	P=62.63	P=70.76	P=68.47	P=65.02	P=69.23	P=64.15	**P=23.02**
	R=44.70	**R=37.77**	R=44.54	**R=42.74**	R=48.57	R=39.07	R=33.85	R=33.71	R=30.36	R=44.44
	F=53.56	**F=46.98**	**F=53.71**	F=50.81	**F=57.60**	**F=49.75**	**F=44.52**	**F=45.34**	F=41.21	**F=27.83**
Convolutional Neural Network (CNN)	P=45.85	P=37.86	P=51.33	P=37.21	P=47.00	P=30.82	P=22.97	P=30.12	P=38.56	P=20.65
	R=56.07	R=32.77	**R=45.01**	R=39.53	**R=65.95**	**R=50.68**	**R=57.95**	**R=47.94**	**R=52.68**	**R=48.12**
	F=50.45	F=35.13	F=47.96	F=38.34	F=54.88	F=38.33	F=32.90	F=36.99	**F=44.53**	F=26.55

Table 4: Classification performance on target users in SMHD. All methods rely on the official SMHD split introduced in Section 3 for training, tuning, and testing. We report the performance of each classifier as binary predictor (i.e., given a condition, predict if a user belongs to the diagnosed or control group) as well as in a multi-label, multi-class setting (i.e., given a user, predict if and which conditions they were diagnosed with).

more female references. This might point to a gender bias in the corpus towards males, which would shift an increased preoccupation with romantic relationships towards references of female partners. Cognitive processes (markers of cognitive activity associated with rational thought and argumentation) are also observed more in people with mental health conditions. This could be due to the observation that these groups tend to express their feelings more. Other attributes such as health, focus on past time, and biological processes are observed more in conditions such as depression. Unsurprisingly, attributes like leisure and money are more prevalent among controls.

5 Experiments

In order to provide a more thorough understanding of SMHD, we explored several classification methods for each of the conditions included in the dataset. We train our classifiers both in a binary and multi-label multi-class setting. For the binary tasks, all classifiers are trained only on the subset of diagnosed users in SMHD who are associated with any condition and their respective control users. Recall that, as discussed in 3.3, users can be diagnosed with one or more conditions; as such, we also evaluate the baseline classifiers in a multi-class, multi-label setting. In other words, given a user, classifiers are trained to predict if and which conditions the user was diagnosed with. In detail, we consider the following methods:

Logistic regression: we trained a simple discriminative model using bag-of-words features extracted over all posts by all users in the training set. Posts were concatenated and split on non-alphanumeric characters; we normalized tokens by folding them to lowercase. Words occurring less than 20 times in the training set were removed. Features were weighted using *tf-idf* and ℓ_2-normalized.

XGBoost (Chen and Guestrin, 2016): we evaluated the performance of an ensemble of decision trees. For this model, the same features described above were used. We set the number of estimators to 100.

Support vector machine (SVM): we included a SVM classifier with linear kernel among our baselines. Like the methods above, this model was trained on *tf-idf* bag-of-words features.

Supervised FastText (Joulin et al., 2016): we trained a shallow neural net model using FastText. After tuning on the development set, we set the dimension of the hidden layer to 100. The model leverages sub-word information by using character ngrams of size 3 to 6; it was trained for 100 epochs.

Convolutional neural network (CNN): Finally, we tested a CNN approach to learn ngram sequences indicative of each condition. We used a model architecture based on the work of Kim (2014), with

a filter size of 3 and a pooling size of 4. A model was trained for each condition, and each model was trained for 100 epochs. Each user's posts are concatenated and truncated to 15,000 tokens. This representation combines tractability with density. We do not pad posts individually hence no space is wasted by including short posts. Each token is represented by the FastText embeddings from above.

Results of the classification methods are reported in Table 4. All classifiers are trained, tuned, and tested on binary labels, except for the results reported in the rightmost column ("multi-label"). Overall, FastText obtained the best performance in terms of F1 score across all conditions except bipolar, in which it is outperformed by the SVM model, and eating disorder, where it is outperformed by a CNN. FastText also outperforms all other classifiers in the multi-label, multi-class test setting. We observe that models trained on bag-of-words features favor precision over recall, while the two neural models we tested offer more balanced performance. This can be explained by the fact that bag-of-words models are more likely to give more weight to strongly indicative text features, potentially causing some overfitting on the training data.

Comparing across conditions, we note that the performance on models is strongly affected by the number of diagnosed users in the training set. All classifiers struggle on the three conditions with fewest users (OCD, schizophrenia, and eating). This observation is consistent with the analysis on LIWC categories reported in Section 4.

Finally, we note that the simpler neural model (FastText) outperformed the richer CNN method. We explain this result by observing that FastText considers all posts by a user, while CNN samples a portion of them. This strategy, which is motivated by efficiency reasons, was based on the work of Yates et al. (2017), in which sampling was found to be an effective strategy to make the classification problem tractable. However, this result suggests that novel methods ought to be studied to achieve a robust sampling strategy across all conditions included in SMHD.

6 Conclusion

We presented SMHD, a large dataset of Reddit users with diverse mental health conditions and matched control users. Our dataset was constructed using high-precision diagnosis patterns and carefully selected control users. To our knowledge, SMHD is the largest dataset that supports a variety of mental health conditions and is up to two orders of magnitude larger than the largest published similar resource. We further examined differences in language use between mental health condition (diagnosed users) and control groups as measured by various linguistic and psychological signals. Several text classification methods were explored to identify the diagnosed users, with FastText being the most effective approach overall. We make our dataset available to the community, and hope that it will foster further research into these problems and enhance reproducibility of suggested approaches.

References

Tim Althoff, Kevin Clark, and Jure Leskovec. 2016. Large-scale analysis of counseling conversations: An application of natural language processing to mental health. In *TACL*.

American Psychiatric Association. 2013. *Diagnostic and Statistical Manual of Mental Disorders (DSM-5)*. American Psychiatric Association Publishing.

Adrian Benton, Margaret Mitchell, and Dirk Hovy. 2017. Multitask learning for mental health conditions with limited social media data. In *EACL*.

Wilma Bucci and Norbert Freedman. 1981. The language of depression. *Bulletin of the Menninger Clinic*, 45(4):334.

David Carmel, Eitan Farchi, Yael Petruschka, and Aya Soffer. 2002. Automatic query refinement using lexical affinities with maximal information gain. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 283–290. ACM.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM.

Arman Cohan, Sydney Young, Andrew Yates, and Nazli Goharian. 2017. Triaging content severity in online mental health forums. *Journal of the Association for Information Science and Technology (JASIST)*.

Jacob Cohen. 1988. Statistical power analysis for the behavioral sciences 2nd edition.

Glen Coppersmith, Mark Dredze, and Craig Harman. 2014a. Quantifying mental health signals in Twitter. In *Proceedings of the Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, pages 51–60.

Glen Coppersmith, Craig Harman, and Mark Dredze. 2014b. Measuring post traumatic stress disorder in Twitter. In *ICWSM*.

Glen Coppersmith, Mark Dredze, Craig Harman, and Kristy Hollingshead. 2015a. From ADHD to SAD: Analyzing the language of mental health on Twitter through self-reported diagnoses. In *Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, pages 1–10.

Glen Coppersmith, Mark Dredze, Craig Harman, Kristy Hollingshead, and Margaret Mitchell. 2015b. CLPsych 2015 shared task: Depression and PTSD on Twitter. In *CLPsych HLT-NAACL*.

Glen Coppersmith, Casey Hilland, Ophir Frieder, and Ryan Leary. 2017. Scalable mental health analysis in the clinical whitespace via natural language processing. In *Biomedical & Health Informatics (BHI), 2017 IEEE EMBS International Conference on*, pages 393–396. IEEE.

Munmun De Choudhury and Emre Kıcıman. 2017. The language of social support in social media and its effect on suicidal ideation risk. In *Proceedings of the... International AAAI Conference on Weblogs and Social Media. International AAAI Conference on Weblogs and Social Media*, volume 2017, page 32. NIH Public Access.

Munmun De Choudhury, Scott Counts, and Eric Horvitz. 2013a. Major life changes and behavioral markers in social media: case of childbirth. In *Proceedings of the 2013 conference on Computer supported cooperative work*, pages 1431–1442. ACM.

Munmun De Choudhury, Michael Gamon, Scott Counts, and Eric Horvitz. 2013b. Predicting depression via social media. In *ICWSM*.

Bart Desmet and Veronique Hoste. 2018. Online suicide prevention through optimised text classification. *Information Sciences*, 439-440:61–78.

Sindhu Kiranmai Ernala, Asra F. Rizvi, Michael L. Birnbaum, John M. Kane, and Munmun De Choudhury. 2017. Linguistic markers indicating therapeutic outcomes of social media disclosures of schizophrenia. *PACMHCI*, 1:43:1–43:27.

Dirk Hovy and Shannon L Spruit. 2016. The social impact of natural language processing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 591–598.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Rohan Kshirsagar, Robert Morris, and Sam Bowman. 2017. Detecting and explaining crisis. *arXiv preprint arXiv:1705.09585*.

David E. Losada and Fabio Crestani. 2016. A test collection for research on depression and language use. In *CLEF*.

Sean MacAvaney, Bart Desmet, Arman Cohan, Luca Soldaini, Andrew Yates, Ayah Zirikly, and Nazli Goharian. 2018. RSDD-Time: Temporal annotation of self-reported mental health diagnoses. In *Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic*, pages 168–173.

David N. Milne, Glen Pink, Ben Hachey, and Rafael Alejandro Calvo. 2016. CLPsych 2016 shared task: Triaging content in online peer-support forums. In *CLPsych@HLT-NAACL*.

Margaret Mitchell, Kristy Hollingshead, and Glen Coppersmith. 2015. Quantifying the language of schizophrenia in social media. In *Proceedings of the 2nd workshop on Computational linguistics and clinical psychology: From linguistic signal to clinical reality*, pages 11–20.

Danielle L Mowery, Albert Park, Craig Bryan, and Mike Conway. 2016. Towards automatically classifying depressive symptoms from Twitter data for population health. In *Proceedings of the Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media (PEOPLES)*, pages 182–191.

Danielle Mowery, Craig Bryan, and Mike Conway. 2017a. Feature studies to inform the classification of depressive symptoms from Twitter data for population health. *arXiv preprint arXiv:1701.08229*.

Danielle Mowery, Hilary Smith, Tyler Cheney, Greg Stoddard, Glen Coppersmith, Craig Bryan, and Mike Conway. 2017b. Understanding depressive symptoms and psychosocial stressors on Twitter: a corpus-based study. *Journal of medical Internet research*, 19(2).

Jane M Murphy, Donald C Olivier, Richard R Monson, Arthur M Sobol, Elizabeth B Federman, and Alexander H Leighton. 1991. Depression and anxiety in relation to social status: A prospective epidemiologic study. *Archives of General Psychiatry*, 48(3):223–229.

Matthew L Newman, James W Pennebaker, Diane S Berry, and Jane M Richards. 2003. Lying words: Predicting deception from linguistic styles. *Personality and social psychology bulletin*, 29(5):665–675.

Umashanthi Pavalanathan and Munmun De Choudhury. 2015. Identity management and mental health discourse in social media. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 315–321, New York, NY, USA. ACM.

James W Pennebaker, Ryan L Boyd, Kayla Jordan, and Kate Blackburn. 2015. The development and psychometric properties of LIWC2015. Technical report.

Daniel Preotiuc-Pietro, Johannes C. Eichstaedt, Gregory J. Park, Maarten Sap, Laura Smith, Victoria Tobolsky, H. Andrew Schwartz, and Lyle H. Ungar. 2015. The role of personality, age, and gender in tweeting about mental illness. In *CLPsych HLT-NAACL*.

Philip Resnik, Anderson Garron, and Rebecca Resnik. 2013. Using topic modeling to improve prediction of neuroticism and depression in college students. In *EMNLP*.

Philip Resnik, William Armstrong, Leonardo Claudino, and Thang Nguyen. 2015. The University of Maryland CLPsych 2015 shared task system. In *CLPsych HLT-NAACL*.

Tetsuya Sakai. 2016. Statistical significance, power, and sample sizes: A systematic review of SIGIR and TOIS, 2006-2015. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 5–14. ACM.

Luca Soldaini and Elad Yom-Tov. 2017. Inferring individual attributes from search engine queries and auxiliary information. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*, pages 293–301.

Tara W Strine, Ali H Mokdad, Lina S Balluz, Olinda Gonzalez, Raquel Crider, Joyce T Berry, and Kurt Kroenke. 2008. Depression and anxiety in the United States: findings from the 2006 behavioral risk factor surveillance system. *Psychiatric services*, 59(12):1383–1390.

Simon Šuster, Stéphan Tulkens, and Walter Daelemans. 2017. A short review of ethical challenges in clinical natural language processing. *arXiv preprint arXiv:1703.10090*.

Sho Tsugawa, Yusuke Kikuchi, Fumio Kishino, Kosuke Nakajima, Yuichi Itoh, and Hiroyuki Ohsaki. 2015. Recognizing depression from Twitter activity. In *CHI*.

R Jay Turner, B Gail Frankel, and Deborah M Levin. 1983. Social support: Conceptualization, measurement, and implications for mental health. *Research in community & mental health*.

Rianne Van der Zanden, Keshia Curie, Monique Van Londen, Jeannet Kramer, Gerard Steen, and Pim Cuijpers. 2014. Web-based depression treatment: Associations of clients word use with adherence and outcome. *Journal of Affective Disorders*, 160:10–13.

E Watkins and RG Brown. 2002. Rumination and executive function in depression: An experimental study. *Journal of Neurology, Neurosurgery & Psychiatry*, 72(3):400–402.

Bernard L Welch. 1947. The generalization ofstudent's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35.

Andrew Yates and Nazli Goharian. 2013. ADRTrace: detecting expected and unexpected adverse drug reactions from user reviews on social media sites. In *European Conference on Information Retrieval*, pages 816–819. Springer.

Andrew Yates, Arman Cohan, and Nazli Goharian. 2017. Depression and self-harm risk assessment in online forums. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2968–2978, Copenhagen, Denmark, September. Association for Computational Linguistics.

Elad Yom-Tov and Evgeniy Gabrilovich. 2013. Postmarket drug surveillance without trial costs: discovery of adverse drug reactions through large-scale analysis of web search queries. *Journal of medical Internet research*, 15(6).

Crowdsourcing a Large Corpus of Clickbait on Twitter

Martin Potthast[1] **Tim Gollub**[2] **Kristof Komlossy**[2] **Sebastian Schuster**[2]
Matti Wiegmann[2,3] **Erika Patricia Garces Fernandez**[2] **Matthias Hagen**[4] **Benno Stein**[2]

[1]Leipzig University [2]Bauhaus-Universität Weimar
[3]German Aerospace Center (DLR) [4]Martin-Luther-Universität Halle-Wittenberg

Abstract

Clickbait has become a nuisance on social media. To address the urging task of clickbait detection, we constructed a new corpus of 38,517 annotated Twitter tweets, the Webis Clickbait Corpus 2017. To avoid biases in terms of publisher and topic, tweets were sampled from the top 27 most retweeted news publishers, covering a period of 150 days. Each tweet has been annotated on 4-point scale by five annotators recruited at Amazon's Mechanical Turk. The corpus has been employed to evaluate 12 clickbait detectors submitted to the Clickbait Challenge 2017.

Download: https://webis.de/data/webis-clickbait-17.html Challenge: https://clickbait-challenge.org

1 Introduction

For publishers of online content, the most popular revenue model is advertising. On freely accessible web pages ads are displayed along the content, and publishers earn fees for each ad seen by the visitors of these pages. Online content is often expected "to pay for itself:" it has to attract enough visitors such that the earned fees exceed the investment made in creating it. Assuming that each piece of content has its target audience, generating profit boils down to letting the respective target audience know where to find it. Ironically, this requires the content itself to be advertised.

Notwithstanding other means of advertisement, today, most publishers advertise their content on social media. By spreading "teaser messages" (be it with or without paying promotion fees), publishers engage in viral marketing. Teaser messages consist of two or three parts: (1) a short text, (2) an optional media attachment such as an image or a video, and (3) a link to the publisher's page where the advertised content is found. From a publisher's perspective, the ideal teaser message discloses its advertised content to the least possible extent, so that its target audience (better: everyone) is tempted to visit the publisher's pages. Conversely, from a reader's perspective, the ideal teaser message is a self-contained summary, so that visiting the publisher's page is subject only to cases where not everything important could be fitted into the summary, or where the reader wants to learn more about the backgrounds. The two ideals are naturally at odds, and the social network operators may need to reconcile between them.

Figure 1 shows examples of teaser messages in the above sense, advertising news content on Twitter. The left-to-right ordering of the messages in the figure is indicative of the degree to which an average reader might feel sufficiently informed without clicking the link (left), less informed but rather curious and urged to click (right), or something in-between (middle). The messages more to the right side of the figure are examples of what is called "clickbait" today. Clickbait is an umbrella term that encompasses all kinds of teaser messages in social media capable of instigating an (on average) increased click-through.

This paper introduces the Webis Clickbait Corpus 2017, a large-scale corpus of teaser messages, which has been built to address automatic clickbait detection. All messages in the corpus have been carefully sampled and annotated via crowdsourcing regarding their degree of clickbaitiness. After reviewing related work in Section 2, Section 3 defines clickbait and its operationalization, and Sections 4 and 5 detail the corpus construction process, including a first corpus analysis.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1498–1507
Santa Fe, New Mexico, USA, August 20-26, 2018.

Figure 1: Examples of teaser messages from news publishers on Twitter. The tweets have been redacted, removing information identifying the publishers in order not to pillory them.

2 Related Work

To the best of our knowledge, Table 1 lists all clickbait-related resources available to date. The Webis Clickbait Corpus 2017 (Webis-Clickbait-17, for short) was preceded by the Webis Clickbait Corpus 2016 built by Potthast et al. (2016) to study clickbait detection for the first time. Given that clickbait itself is a fairly recent subject of inquiry, all of these resources have been published within the past two years. The resources can be distinguished in terms of genre, teaser type, approaches to acquisition and annotation, whether the linked articles are archived, and their size. A discussion of each aspect follows.

Genre. All resources to date, including ours, cover news. News are of particular importance, since the use of clickbait in this domain raises concerns: rather than maximizing the informative value of the teaser messages employed for news dissemination, news publishers also misuse them to maximize their ad revenue. Because of their important societal role, though, news publishers must be held to high standards, and the self-imposed codes of conduct of major news publishers clearly reject manipulative means to increase readership. Clickbait has been dismissed as "digital yellow journalism," i.e., as nothing new or noteworthy,[1] but this view is short-sighted: our previous work shows that nearly 25% of a sample of 2,992 tweets shared by the top-20 most retweeted news publishers are clickbait (Potthast et al., 2016)—a finding which is now confirmed by our 10 times larger corpus. Besides news, clickbait does occur in other genres, too. For example, on entertainment platforms such as YouTube (Qu et al., 2018). In fact, all kinds of social media where users are referred to another web page are susceptible to clickbait.

Teaser type. Two teaser types have been covered for corpus construction so far, namely the headline of a news article, and tweets about news articles. Headlines must be considered an approximation of an actual teaser message on social media; they are often used as textual component of a teaser, but not necessarily verbatim, and typically in combination with media attachments. The corpora using headlines lack associated media attachments, however. Only our previous corpus (Potthast et al., 2016) and the one presented in this paper incorporate realistic teaser messages. The teasers found on platforms like Facebook, LinkedIn, YouTube, and the like, are all similar to tweets, but differ in terms of allowed text length, media arrangement, and available meta data. Other teaser types are found on portal pages of websites where users are referred to more specific content found elsewhere (e.g., consider the teaser messages found on a news publisher's front page). Yet others may be found in discussion forums, comment boards, and emails, where more free-form text is allowed. However, when used as clickbait, these kinds of messages more often resemble spam, which is distinctly different from clickbait, but shares the goal of luring readers to their sender's web pages. One may expect that the clickbait message style will eventually become part of the repertoire of spammers.

Acquisition approach. Key to the representativeness of a language resource is the corpus acquisition strategy applied. Here, the news domain presents difficulties for sampling, since an average researcher hardly has access to all news published in a given period of time, so that directly sampling from the population of news is infeasible. To circumvent this issue, three strategies have been devised for clickbait corpus acquisition, namely reputation-based, gatekeeper-based, and importance-based acquisition.

With reputation-based acquisition, teaser messages (or news articles) are selected from publishers with a good reputation to obtain a relatively clean sample of non-clickbait, and from publishers with correspondingly bad reputation to obtain as much clickbait as possible. For instance, according to the

[1] http://theconcourse.deadspin.com/shut-up-about-clickbait-1551902024

Publication	Genre	Teaser	Acquisition	Annotation (Scale, People, κ)			Articles	Size
Agrawal (2016)	News	Headline	Gatekeeper-based	Binary	3	0.84	No	2,388
Potthast et al. (2016)	News	Tweet	Importance-based	Binary	3	0.35	Yes	2,992
Biyani et al. (2016)	News	Headline	Reputation-based?	Binary	?	?	?	4,073
Chakraborty et al. (2016)	News	Headline	Reputation-based	Binary	3	0.79	No	15,000
Rony et al. (2017)	News	Headline	Reputation-based	Binary	3	0.79?	No	32,000
Webis-Clickbait-17	News	Tweet	Importance-based	Graded	5	0.36	Yes	38,517

Table 1: Overview of clickbait corpora. All are in English; question marks (?) indicate lack of clarity.

literature, the latter publishers include BuzzFeed, Huffington Post, Upworthy, ViralNova, Scoopwhoop, and ViralStories. This approach has been applied by Chakraborty et al. (2016) whose corpus formed the basis for the one of Rony et al. (2017). The description of Biyani et al. (2016) is very superficial, but they do mention certain publishers, and the fact that almost half of their teasers are clickbait suggests that they have actively searched for such messages. Gatekeeper-based acquisition makes use of, e.g., web forums where news are curated. Agrawal (2016) obtain their selection of news from Reddit forums, where one forum pre-selects news of interest for the benefit of users, and where another collects clickbait to "spoil" it, e.g., by giving away the information withheld from a teaser. Both, reputation-based and gatekeeper-based acquisition, directly induce a ground truth. However, especially the reputation-based approach must be taken with a grain of salt, since many publishers with a comparably low reputation, such as BuzzFeed and Huffington Post, still publish a significant amount of news without resorting to clickbait teaser messages. Rony et al. (2017), Chakraborty et al. (2016), and Agrawal (2016) report to have double-checked the induced labels; each instance has been reviewed by at least three student volunteers. But the substantial inter-annotator agreements must be taken with a grain of salt, too, since as Agrawal (2016) reports, "the reason for such a high value of inter-assessor agreement is due to the fact that these headlines are already in their well defined categories owing to the nature of data collection." This conscientious observation hints at a bias that has been introduced by reputation-based acquisition and *not* disguising the sources or at least randomizing the messages before review.

By comparison, under importance-based acquisition as applied in (Potthast et al., 2016) as well as for our new corpus, publishers are ranked with respect to a specific criterion, such as sales or shares, and teasers are sampled from the top-ranked publishers. This way, it is ensured that a corpus covers the news seen by the largest portion of the target audience. Moreover, no assumptions are made about the nature of a publisher, so that the proportion of clickbait observed is not artificially amplified as is the case with the aforementioned acquisition approaches. Importance-based acquisition does not induce a ground truth, so that all teasers collected must be reviewed and annotated.

Annotation approach. Previous clickbait corpora consider only a binary scale for annotation, where a teaser is either clickbait or not. In our own annotation experiments, we found the binary scale to be lacking, since, as illustrated by the examples depicted in Figure 1, the degree to which a certain teaser message tries to manipulate its reader varies depending on the person reading it. For example, it may be that one person is hardly affected by any of these examples, whereas another more strongly experiences an urge to click. This is also why a single opinion does not suffice to arrive at an objective judgment about a teaser message, but a number of opinions should be collected and averaged.

Article archival. Teaser messages do not occur in isolation but link to some content on another web page. While the task of clickbait detection has typically been cast as an assessment of teaser messages in isolation, the linked content may still play an important role for subsequent analyses. Only our two corpora Webis-Clickbait-16 and 17 incorporate the linked web pages.

Size. Finally, in terms of size, our corpus is on par with that of Rony et al. (2017), albeit, as discussed above, our sampling and annotation approaches are significantly more sophisticated.

3 Defining, Understanding, and Operationalizing Clickbait

Clickbait turns out to be an elusive concept. Characterizing clickbait as an enumeration of properties exhibited by a teaser message proves difficult. Having reviewed thousands of examples, we would say that clickbait withholds crucial information, and that it is emotional, sensational, condescending, inflammatory, know-it-all, or any combination thereof. However, this list of properties is likely not exhaustive, leaves lots of room for interpretation, and none appear to be a strict necessity. Furthermore, these properties are difficult to be operationalized. But despite the apparent difficulty of capturing in a clear-cut definition the essence of what makes clickbait clickbait, the term has been in use since 1999, according to Merriam Webster's. Apparently, the underlying phenomenon, namely that messages are spread designed to entice readers into clicking a link, has been around ever since the ad revenue business model took off on the web. And the phenomenon has been nagging enough so that web users took note of it and coined the term. This is evidence that people are able to tell apart clickbait from other forms of messages found online and gives us hope that machine learning technology, too, can be taught to detect clickbait.

A better way of understanding and explaining clickbait is by describing it in terms of what happens at publisher site and at reader site. The following four definitions contrast the emergence of clickbait and publisher intentions with the effects of clickbait and reader perception:

- *Emergence.* Clickbait is the result of optimizing teaser messages to maximize click-through.
- *Intention.* Clickbait are teaser messages whose authors intend to lure as many people as possible to a web page, disregarding the content's target audience.
- *Effects.* Clickbait are teaser messages that manipulate its readers into clicking a link.
- *Perception.* Clickbait are teaser messages perceived by (some) readers as bait to click a link.

These definitions capture important aspects of clickbait that a mere listing of properties cannot convey, and they reflect how clickbait is described, e.g., at Merriam Webster and on Wikipedia. The question remains, though, which of these definitions is best suited to *operationalize* clickbait.

For publishers, the availability of facilities to analyze user interactions with postings in real time on social networks naturally has given rise to their optimization to maximize click-through. Today's shape and form of clickbait can be attributed to Upworthy (Koechley, 2012), who "pioneered" and popularized clickbait by flooding Facebook at a scale that prompted Facebook to change its algorithms to reduce it (El-Arini and Tang, 2014). Other publishers quickly learned from Upworthy's example. To learn more, we asked professional journalists who use social network analytics on a daily basis: On the upside, seeing a recently finished article spread is gratifying, and they are thankful for the opportunity to adjust teasers, e.g., to fix mistakes or misconceptions. On the downside, this feedback about personal success or failure is also available to their department heads. When encountering clickbait, some form of malicious intent, be it voluntary or forced, can hence be presumed. From a reader's perspective, clickbait is mainly a form of manipulation. The manipulative power of clickbait has been attributed to the curiosity gap (Loewenstein, 1994), a gap of knowledge created in a reader's mind by withholding crucial information from a teaser message. Once experienced, the gap causes an urge to be filled, which may be explained as a kind of curiosity. This is at least what is proclaimed by Koechley (2012) as to why clickbait worked for Upworthy, and what has henceforth been cited. The connection between Loewenstein's theory and clickbait has not been shown scientifically, though, so that it remains an open question how the relatively short teaser messages can have such a strong effect. More generally, we say that clickbait appeals to human urges and exploits cognitive dissonances to fulfill its goal.

Our best bet at operationalizing clickbait is the fact that clickbait is perceptible to (some) readers, whereas proving malicious intent of individual journalists or publishers as a whole (i.e., reputation-based annotation) is virtually impossible. An obvious limitation of the former operationalization is that we cannot capture clickbait that eludes conscious reflection while still successfully manipulating (some) readers. This is why each teaser is judged by more than one annotator, and why our assessors have the freedom to judge the baitiness of a teaser message on a Likert scale rather enforcing a binary decision.

4 Corpus Acquisition

In order to make a valuable contribution to the research community, we set out to render the Webis Clickbait Corpus 2017 authentic, representative, rich in terms of potential features, unbiased, and large-scale. An overview of the main characteristics of our acquisition approach is provided in Table 2. We follow and extend the approach taken to construct our previous, small Webis-Clickbait-16 corpus (Potthast et al., 2016), rendering both corpora comparable. In what follows, we outline the design choices we were required to make regarding the selection of publishers, the crawling of their news, and the sampling of news items for annotation.

4.1 Platform and Publisher Selection

As teaser type for our corpus, we chose Twitter tweets. Twitter was chosen because (1) the platform has a large user base, and (2) virtually all major US news publishers disseminate their articles through Twitter. Facebook would have been equally suited, but we opted for Twitter for its more rigorous limitations of shape and form of teaser messages. This way, we hoped to ensure less variability in terms of how clickbait can be constructed, lest the more free-form teaser messages on Facebook may have caused more ambiguity. At any rate, all publishers under consideration disseminate their news on both platforms simultaneously. A cursory comparison of teasers shared on both platforms for the same news item shows that their contents are typically roughly equivalent, so that our platform choice may not significantly limit the generalizability of results obtained based on our corpus.

Our sample of news publishers is governed by publisher importance in terms of retweets. Restricting ourselves to English-language publishers, we obtain a ranking of the top-most retweeted news publishers from the NewsWhip social media analytics service.[2] Taking the top 27 publishers,[3] we ensured that the news covered reflect what a large population of English-speaking internet users is confronted with on a daily basis. Although many of the top-most publishers (say, the top 10) remain the same over long time spans, others make an appearance on lower ranks and are displaced again as time goes by and publishers compete for user attention. Our sample of publishers reflects the ranking of October 2016.

4.2 Crawling, Archiving, and Preprocessing News Teasers and Articles

Given the Twitter handles of the 27 news publishers selected, we used Twitter's API to record every tweet they published in the period from December 1, 2016, through April 30, 2017. To enable the research community to develop and experiment with a rich set of features, we included the tweet text, media attachements, and the meta data provided by Twitter. Due to technical limitations we had to omit tweets that contained video content. Further discarding tweets that did not contain exactly one hyperlink, we crawled the news article advertised. Besides technical difficulties arising from the HTTP forwarding associated with links on Twitter as well as publisher-specific forwarding, we found it insufficient to only download the HTML content of the linked news articles: many news publishers paginate articles, and some are at the forefront of dynamic web page design, so that we found cases where the article content was not part of the initial download, but resulted from JavaScript-based content loading. That, and the fact that the articles of some publishers tend to disappear or be reorganized rather quickly after publication, led us to adopt a more reproducible archiving approach. Using the tools underlying the Wayback Machine of the Internet Archive, we recorded the whole communication that takes place between a client (browser) requesting the web page of a news article and the publisher's web server hosting it, storing it within web archive (WARC) files (including, e.g., HTML, CSS, Javascript, and images). This way, every article page that forms part of our corpus can be reviewed as it was on the day we crawled it, allowing for corpus reviews even after years, hence maximizing its reproducibility.

Nevertheless, users of our corpus will not have to handle the raw WARC files. For convenience, we applied publisher-specific wrappers extracting a set of content-related fields (cf. fields prefixed with "target" in Table 2). In this regard, we discarded Boilerpipe and other generic content extraction tools, since their performance was surprisingly poor.

[2] https://www.newswhip.com

[3] We aimed for the top 30, but accidentally used the wrong Twitter handles of three, so that we excluded them altogether.

Corpus Acquisition	
Platform:	Twitter
Crawling period:	Dec 1 2016 – Apr 30 2017
Crawled tweets:	459,541
Publishers:	27. Names: abc, bbcworld, billboard, bleacherreport, breitbartnews, business, businessinsider, buzzfeed, cbsnews, cnn, complex, espn, forbes, foxnews, guardian, huffpost, independent, indiatimes, mailonline, mashable, nbcnews, nytimes, telegraph, usatoday, washingtonpost, wsj, yahoo
Filters:	- No videos in tweets. - Exactly one hyperlink in tweet. - Article archiving succeeded. - Main content extraction succeeded.
Recorded fields:	12. Names: postId, postTimestamp, postText, postMedia, postPublisher, targetUrl, targetTitle, targetDescription, targetKeywords, targetParagraphs, targetCaptions, targetWarcArchive
Sampling strategy:	Maximally 10 tweets per day and publisher
Sampled tweets:	38,517

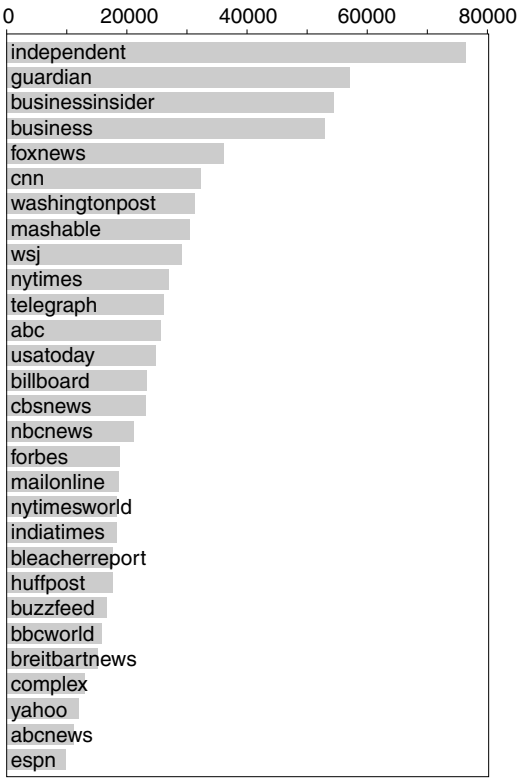


Table 2: Corpus acquisition overview (left), and number of tweets crawled from every publisher (right).

4.3 Sampling News

When sampling from a stream of news, special attention must be paid to avoid topic bias. This pertains particularly to the task of constructing a clickbait corpus, since clickbait is more a matter of teaser style rather than teaser topic. A clickbait detection model trained on too narrow a range of topics may not generalize well since it will likely pick up stylistic differences that result from differences between topics rather than differences between the style of clickbait and that of non-clickbait. Our previously published corpus Webis-Clickbait-16 suffers from this shortcoming, since it covers only one week worth of tweets. Although news coverage switches topics rather quickly, this is hardly enough to ensure topic diversity. To obtain a sample of tweets that has a high topic diversity, we crawled news as described above for five months in a row, yielding almost half a million tweets that fit our criteria and that were successfully archived.

From this population, we drew a random sample for annotation where, for budgetary reasons, the goal was to draw at least 30,000 tweets and at most 40,000. Since the distribution of tweets per publisher is highly skewed, we apply stratified sampling to avoid a corresponding publisher bias. Similarly, we ensure that tweets are sampled from each day of the five months worth of tweets to ensure coverage of the whole time period. Selecting a maximum of ten tweets per day and publisher yielded a set of 38,517 tweets and archived articles, which were then subjected to manual annotation.

5 Corpus Annotation

The annotation of the acquired tweets regarding their clickbaitiness was implemented with the crowdsourcing platform Amazon Mechanical Turk (AMT). To obtain high-quality annotations we invested substantial effort into preliminary evaluations of different task designs, alternative annotation scales, and, in particular, into the review of the annotations made. The following subsections detail our considerations, experiences, and insights.

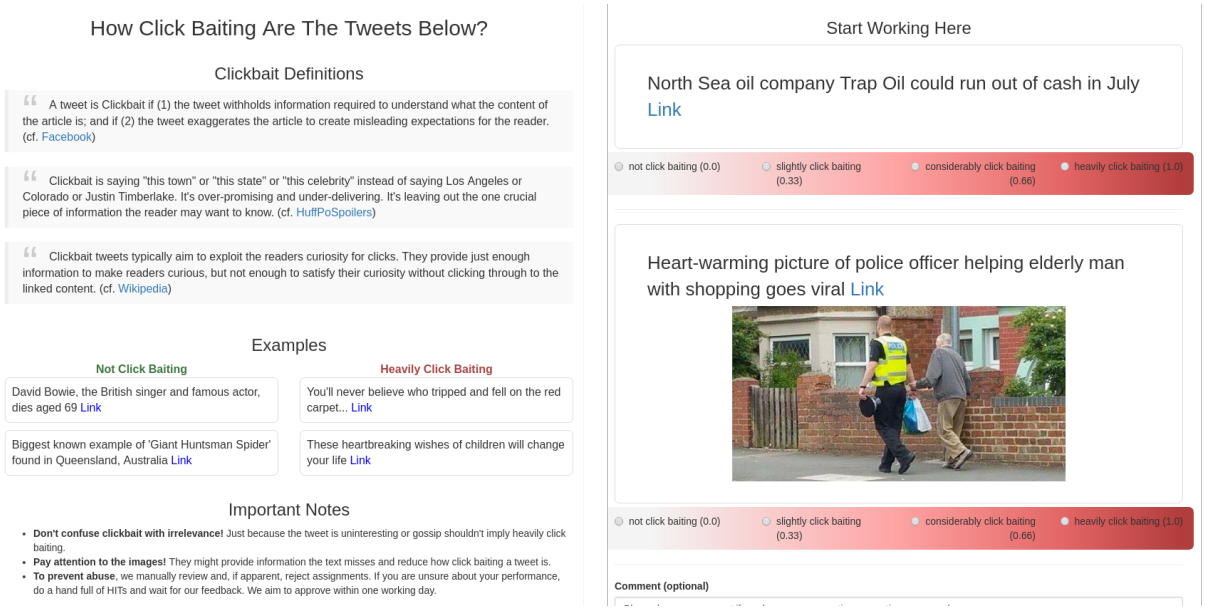


Figure 2: Screenshot of the final AMT task design.

5.1 Task Design

Designing a successful crowdsourcing task entails a number of design decisions, which can hardly be made without pilot studies. We conducted a series of AMT pilot studies to optimize the task design. In this regard, we employed the 2992 tweets of our previous clickbait corpus Webis-Clickbait-16. Figure 2 shows the final task design we used.

Clickbait is a concept with which crowd workers may not be familiar. To develop their "mental clickbait model," we quoted colloquial clickbait definitions from Facebook, HuffPoSpoilers, and Wikipedia as part of the task instructions (i.e., the aforementioned definitions are not suited for laymen). In addition, two typical clickbait and two non-clickbait examples were shown, drawn from our previous corpus. Since the annotations of the first pilot studies' tweets still showed an overly high variance, we extended the instructions again: first, since gossip was often misclassified as clickbait, we added a note that workers should not mix up irrelevance with clickbait. Second, since information in images attached to tweets was often overlooked, we added a note that special attention has to be paid to images. To avoid that workers get a high number of rejections because of misunderstanding the task, we suggested to complete at most two task instances and to await their approval before continuing.

The task instructions were followed by a list of tweets, each of which to be annotated regarding its clickbaitiness. We compared different annotation scales, the lengths of the tweet lists (five versus ten), and the number of *check instances* (one or two). As check instances we employed those tweets where all workers agreed; in order to bootstrap a pool of check instances, we started with tweets from our previous corpus. Check instances are of great value for the assessment of crowd workers. In our case it turned out that having two check instances per task instance—one that is clickbait and one that is not—allowed us to pinpoint underperforming workers of two kinds: (1) those with a tendency towards one of the extreme values, and (2) those who do not use the entire grading scale. Because of the (high) number of required check instances, we decided to have ten instead of five tweets per task in the final task design.

Within all studies, we payed one cent per tweet, an amount commensurate with the average time it took workers to complete a task so that a reasonable, albeit not generous, hourly wage would result. An exception was a particular pilot study where, in addition to the assessment of clickbait, workers had to mark words in the tweets which they considered indicative for their judgments. Having such a subtask was recently recommended for a search result relevance annotation task (McDonnell et al., 2016). In our case, this additional task increased the overall task completion time considerably while not improving worker agreement. We hence omitted this additional step in the final task design.

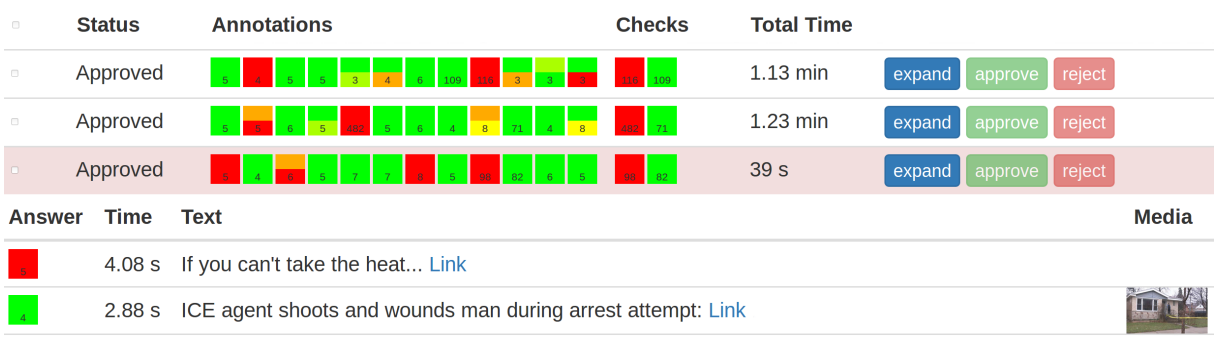


Figure 3: Screenshot of the system used for reviewing the crowdsourced clickbait assessments. Each row in the table refers to a HIT that has been worked on by a specific crowd worker.

5.2 Annotation Scale

Existing clickbait corpora use a binary scale for assessing clickbait. Although there are teaser messages that are obviously clickbait ("You won't believe what happened next!"), we noticed that making a binary decision about a teaser message is often not useful: even a teaser that can be rephrased in a more informative way may not be considered as clickbait if it conveys the idea of what the linked article is about. Hence, to account for different degrees of clickbait, we compared the use of a graded scale to the use of a binary scale. The graded scale has four values (Likert scale with forced choice) and ranges from "not" over "slightly" and "considerably" to "heavily" clickbaiting. For the ten annotations that we collected for each of the 2992 tweets in our previous corpus, the binary scale achieved an agreement of 85% (on average 8.5 annotators voted for the majority class). For the four point graded scale, the agreement dropped to 63%. However, when binarizing these annotations by combining the first two and the second two classes, an agreement of 84% is obtained. This shows that the graded scale is the more flexible choice: it can be abstracted into a binary scale without loss of agreement, and it allows for casting clickbait detection as a regression problem.

5.3 Reviewing Process

To guarantee a high quality dataset, all crowdsourced assessments were reviewed and, if necessary, discarded, resubmitting the respective assignment to AMT. Figure 3 shows a screenshot of the system that we have developed for efficient review of massive amounts of annotations: each row in the table shows a set of annotations (a single assignment) that has been submitted by a crowd worker. The Annotations column in turn shows a colored square for each tweet of the assignment. The upper half of a square color-codes the annotations made by the specific crowd worker, the lower half color-codes how (and how many) other workers annotated the tweet. By default, the color encodes the statistical mode of the annotations; other statistical location parameters such as mean and median are available from a menu. As mentioned above, each assignment included two check instances, which were displayed in the Checks column and which serve as an effective heads-up indicator: only if a worker frequently misclassified check instances, a more in-depth reviewing of the annotations was conducted. To this end, each of the rows in the table is expandable to display the tweets underlying the squares. As a further indicator, task completion time was measured and displayed as well (see the Total Time column). Obviously, a low agreement paired with a short completion time gives a strong bias that a worker did not work conscientiously. Altogether, more than 600 000 individual annotations were reviewed with the system for the clickbait corpus project; our rejection rate was 35.9%.

5.4 Corpus Analysis

The histogram in Figure 4 (left) shows the distribution of tweets across the four classes of our graded scale in the form of stacked bars. To classify a tweet into one of the four classes, the mode of its annotations is used, where, in case of multiple modes, the fifth annotation is used to determine the class. The different colors in the bars encode different levels of agreement. With a value of 0.21 in terms of Fleiss' κ, the annotator agreement is between slight and fair. However, when binarizing the classes

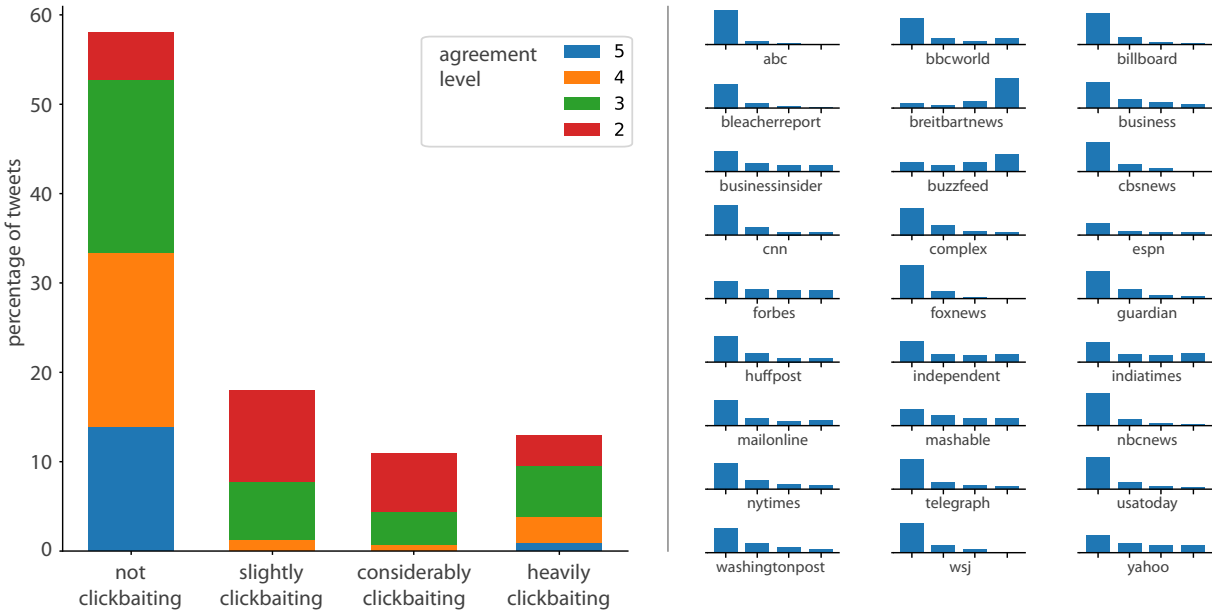


Figure 4: Left: Distribution of tweets over the four clickbait classes; the level of agreement about the class is color-coded. Right: Class distribution per publisher; no color coding is applied.

as described before, κ becomes 0.36, which corresponds to the respective value of 0.35 reported for our previous clickbait corpus (Potthast et al., 2016). Also note that the distribution of tweets across the binarized classes matches that of our previous corpus. Recalling that our previous corpus has been assessed by few trained experts, we conclude that our crowdsourcing strategy lives up to the state of the art and that it can be considered as successful: the two independently designed and operationalized annotation studies still achieve the same result, and hence our annotation experiment can be understood as a reproduction of our previous efforts, only at a larger scale. We summarize this results as follows: to scale the annotation of clickbait corpora, resorting to trained experts is not a necessary requirement.

Figure 4 (right) shows the distribution of tweets across the classes per publisher. Two of the 27 publishers, breitbartnews and buzzfeed, do obviously not follow the overall distribution; both send significantly more clickbait than the others. However, their contribution to the total amount of clickbaiting tweets (and hence to the corpus' publisher bias) is moderate only. Interestingly, TV networks in particular are among the publisher with least clickbait, avoiding even weakly clickbaiting messages.

6 Conclusion

Clickbait has quickly become one of the pests of social media, not unlike spam for email. While its working mechanisms are still barely understood, nearly every major publisher on social media employs it already to a greater or lesser extent to increase their readership. Although clickbait is apparently an effective marketing instrument, the ends do not justify the means: clickbait is also an effective instrument of manipulation. For example, the propagators of fake news make use of clickbait, too, to spread their disinformation. For news dissemination in general, clickbait must be rejected as well, since readers expect to be comprehensively informed, and journalistic codes of ethics typically prohibit unethical means of marketing.

To lay the groundwork for the emerging and ongoing investigation of clickbait by computer linguists and natural language processing alike, we have constructed the Webis Clickbait Corpus 2017, the first large-scale corpus of clickbait which has been carefully designed to be as representative as possible of teaser messages sent by the major publishers on Twitter. This way, our corpus will serve qualitative as well as quantitative research alike, allowing for deeper insights into the nature of clickbait, and the construction of technology to handle it, respectively. To foster the emerging research community on this subject, we share the resource itself and all the technologies that helped to compile it free of charge under permissible open source licenses.

To kickstart the use of our corpus, and to foster the development of new detection technology for clickbait messages, we have organized the Clickbait Challenge 2017, a shared task which used our corpus for evaluation. The corpus was divided into a training and test set, each comprising roughly half of the total number of tweets. Twelve teams of researchers participated in the challenge, submitting as many approaches for evaluation. A detailed review of the submitted approaches is out of the scope of this paper, however, it can be found in (Potthast et al., 2018).

References

A. Agrawal. 2016. Clickbait detection using deep learning. In *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, pages 268–272, Oct.

Prakhar Biyani, Kostas Tsioutsiouliklis, and John Blackmer. 2016. "8 amazing secrets for getting more clicks": Detecting clickbaits in news streams using article informality. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 94–100.

Abhijnan Chakraborty, Bhargavi Paranjape, Sourya Kakarla, and Niloy Ganguly. 2016. Stop clickbait: Detecting and preventing clickbaits in online news media. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2016, San Francisco, CA, USA, August 18-21, 2016*, pages 9–16.

Khalid El-Arini and Joyce Tang. 2014. News Feed FYI: Click-baiting. http://web.archive.org/web/20150529104738/http://newsroom.fb.com/news/2014/08/news-feed-fyi-click-baiting/.

Peter Koechley. 2012. Why The Title Matters More Than The Talk. http://web.archive.org/web/20150611110506/http://blog.upworthy.com/post/26345634089/why-the-title-matters-more-than-the-talk.

George Loewenstein. 1994. The Psychology of Curiosity: A Review and Reinterpretation. *Psychological Bulletin*, 116(1):75.

Tyler McDonnell, Matthew Lease, Mucahid Kutlu, and Tamer Elsayed. 2016. Why is that relevant? collecting annotator rationales for relevance judgments. In *Proceedings of the Fourth AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*.

Martin Potthast, Sebastian Köpsel, Benno Stein, and Matthias Hagen. 2016. Clickbait Detection. In Nicola Ferro, Fabio Crestani, Marie-Francine Moens, Josiane Mothe, Fabrizio Silvestri, Giorgio Maria Di Nunzio, Claudia Hauff, and Gianmaria Silvello, editors, *Advances in Information Retrieval. 38th European Conference on IR Research (ECIR 16)*, volume 9626 of *Lecture Notes in Computer Science*, pages 810–817, Berlin Heidelberg New York, March. Springer.

Martin Potthast, Tim Gollub, Matthias Hagen, and Benno Stein. 2018. The Clickbait Challenge 2017: Towards a Regression Model for Clickbait Strength. In *Proceddings of the Clickbait Chhallenge (to appear)*.

Jiani Qu, Anny Marleen Hißbach, Tim Gollub, and Martin Potthast. 2018. Towards Crowdsourcing Clickbait Labels for YouTube Videos. In *Proceedings of the 6th AAAI Conference on Human Computation and Crowdsourcing (HCOMP 18)*, July.

Md Main Uddin Rony, Naeemul Hassan, and Mohammad Yousuf. 2017. Diving deep into clickbaits: Who use them to what extents in which topics with what effects? *CoRR*, abs/1703.09400.

Cross-lingual Knowledge Projection Using Machine Translation and Target-side Knowledge Base Completion

Naoki Otani*
Carnegie Mellon University
notani@cs.cmu.edu

Hirokazu Kiyomaru
Kyoto University
kiyomaru@nlp.ist.i.kyoto-u.ac.jp

Daisuke Kawahara
Kyoto University
dk@i.kyoto-u.ac.jp

Sadao Kurohashi
Kyoto University
kuro@i.kyoto-u.ac.jp

Abstract

Considerable effort has been devoted to building commonsense knowledge bases. However, they are not available in many languages because the construction of KBs is expensive. To bridge the gap between languages, this paper addresses the problem of projecting the knowledge in English, a resource-rich language, into other languages, where the main challenge lies in projection ambiguity. This ambiguity is partially solved by machine translation and target-side knowledge base completion, but neither of them is adequately reliable by itself. We show their combination can project English commonsense knowledge into Japanese and Chinese with high precision. Our method also achieves a top-10 accuracy of 90% on the crowdsourced English–Japanese benchmark. Furthermore, we use our method to obtain 18,747 facts of accurate Japanese commonsense within a very short period.

1 Introduction

Commonsense has been considered to play a vital role in language understanding (LoBue and Yates, 2011), and considerable effort has been devoted to building knowledge bases (KBs) that organize commonsense (Zang et al., 2013). The largest multi-lingual commonsense KB is ConceptNet (Liu and Singh, 2004b). ConceptNet maintains knowledge as a triple of two concepts and relationship between them, which we call *fact*. The characteristic of ConceptNet is that concepts are represented in undisambiguated forms of words or phrases, which facilitates commonsense acquisition and inference in practice (Liu and Singh, 2004a) and recently has proven useful for building word representations (Speer et al., 2017; Camacho-Collados et al., 2017). We target ConceptNet in this paper.

A major problem lies in a large gap of quantity and quality between languages. The latest release (v5.5.0) of ConceptNet has 2,828,394 unique English facts[1], but the number of Japanese facts in ConceptNet is only 69,902 ($\approx 2.5\%$) even though Japanese knowledge takes up the eighth-largest portion of the database. This problem is not specific to ConceptNet. English KBs are typically larger and of higher quality than other languages. Although an adequate amount of knowledge of named entities is often available in many languages thanks to semi-structured text on the web such as Wikipedia infobox (Lehmann et al., 2014), commonsense is hard to obtain due to the lack of tractable and objective information (Gordon and Van Durme, 2013).

It is not realistic to develop large knowledge resources in every language from scratch because of cost constraints. Instead, this paper focuses on cross-lingual knowledge projection. We translate English commonsense facts into a target language, aiming to gain large commonsense resources in the target language efficiently.

The main challenge is projection ambiguity. For example, consider translating (bat, *CapableOf*, fly) shown in Figure 1. Bat has four Japanese translations such as koumori [bat (animal)] and batto [bat (stick)]. Fly has 27 translations such as tobu [fly (verb)] and hae [fly (insect)]. Thus, (bat, *CapableOf*, fly) results in $4 \times 27 = 108$ Japanese translation candidates in total, and even after filtering them

*This work was conducted while the first author was at Kyoto University.

[1] An English fact is a fact with two English concepts.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1508–1520
Santa Fe, New Mexico, USA, August 20-26, 2018.

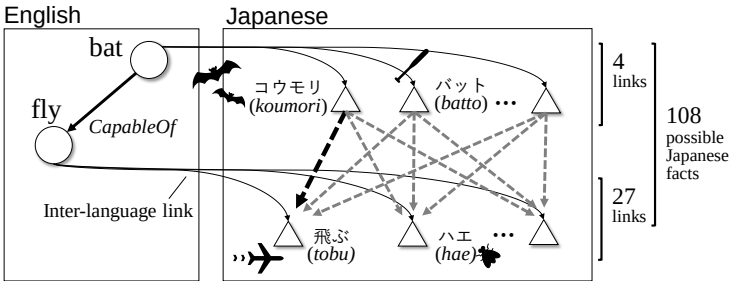


Figure 1: **Ambiguity of knowledge projection.** (bat, *CapableOf*, fly) in English and 108 possible translations in Japanese. The task is to identify the correct link between Japanese words (dash line) that corresponds to the English link (solid line.)

by considering part-of-speech constraints we still have 64 candidates. This problem happens very frequently because 42% of English concepts appearing in inter-language links have more than one Japanese translation.

This is in contrast to previous studies explored cross-lingual knowledge projection focused on knowledge of named entities (Feng et al., 2016; Klein et al., 2017; Chen et al., 2017). Their methods assume one-to-one mapping of concepts across languages. This assumption is reasonable if concepts are named entities because the majority of named entities has one or a few translations, *e.g.*, France (English) and Francia (Spanish).

In contrast, commonsense concepts are represented by common nouns, verbs, and phrases, and those words/phrases have many translations by nature as shown in Figure 1. Such translation ambiguity, that is, the knowledge projection ambiguity can be partially solved by machine translation (MT) and knowledge base completion (KBC) techniques. Cross-lingual knowledge projection can be seen as a structured version of an MT task. KBC models complete missing relations between concepts based on existing relations, which are also closely related to our task. Neither of them, however, can disambiguate knowledge projection with adequate precision. We do not have sufficient training data for building a translation model of facts because MT systems are generally developed not for structured knowledge but for unstructured text. KBC models need to be trained on a sufficiently large KB in a target language.

To alleviate these problems, we combine MT and target-side KBC. The MT and KBC models are trained on separate datasets, and our model weights the estimates from the two models to generate final results. To compute translation probabilities of facts with MT, we propose to convert a fact into plain text with hand-crafted templates.

Our contributions are three-fold.

1. We propose a cross-lingual projection method for undisambiguated forms of commonsense. Our method combines MT and target-side KBC to disambiguate knowledge projection across languages. To utilize an MT model trained on unstructured text, we develop rule-based conversion of structured knowledge.
2. We demonstrate that our method outperforms a projection method that assumes one-to-one mapping of concepts, and single KBC and MT models. Furthermore, an experiment on a crowdsourced dataset shows our method can find correct translations with a top-10 accuracy of 90%.
3. We obtained 18,747 accurate facts of Japanese commonsense using our method and crowdsourcing, which are an equivalent or larger amount of the existing facts in ConceptNet for 12 relation types. We release the resulting datasets as well as code to reproduce our experiments to the research community.[2]

2 Related Work

Developing human language technologies for low-resource languages has been an important challenge for years, and several studies attempted to bridge the resource gap across languages by cross-lingual

[2] https://github.com/notani/CLKP-MTKBC

Relation	e_1	e_2	English	Japanese	Chinese
AtLocation	NP	NP	You are likely to find e_1 in e_2.	e_2 de e_1 wo miru koto ga aru.	Ni keyi zai e_2 zhaodao e_1.
CapableOf	NP	VP	e_1 can e_2	e_1 wa e_2 koto ga dekiru .	e_1 hui e_1.
MadeOf	NP	NP	e_1 is made of e_2.	e_1 wa e_2 kara tsukurareru.	e_1 keyi yong e_2 zhi cheng.
UsedFor	NP	VP	You can use e_1 to e_2.	e_1 wa e_2 tame ni tsukawareru.	e_2 de shihou keneng hui yong dao e_1.

Table 1: **Examples of templates for converting facts into sentences.** Constraints of part-of-speech on e_1 and e_2 (Speer and Havasi, 2012) are also presented. Some templates were developed by the ConceptNet organizers. The rest of the templates can be found in the released code.

knowledge projection.

Klein et al. (2017) and Chen et al. (2017) represented concepts in multiple languages in a unified vector space, and built knowledge base completion models based on vector representations. Their methods ensure a concept in the source language has a similar vector representation to its target-side counterpart, assuming each concept in the source language corresponds to exactly one concept in the target language.

There is a rich body of work on sense embedding, which allows one surface form of a word to have sense-specific vectors (Neelakantan et al., 2014; Iacobacci et al., 2015). However, to the best of our knowledge, previous studies in this field do not target sense vectors of concepts for cross-lingual knowledge projection.

Several studies proposed methods for one-to-one projection of facts (Kuo and Hsu, 2010; Feng et al., 2016). The work by Feng et al. (2016) is the most related to our study. Their model learns mappings between English and Chinese facts by manually annotated alignments. Their experimental result showed the model successfully resolved the projection ambiguity. Their experiment was, however, limited to a narrow domain due to the cost of manual annotations, indicating the difficulty of obtaining sufficient resources for learning a model.

Various types of commonsense is vital to understanding languages in a wide range of tasks such as recognizing textual entailment (LoBue and Yates, 2011). Researchers have compiled resources to maintain such knowledge. Cyc (Lenat, 1995) is a seminal big project that aims to organize commonsense in logical forms. Logical forms are suitable for disambiguating the meaning of language, but we need high expertise to acquire or utilize them. In contrast, ConceptNet (Liu and Singh, 2004b; Speer et al., 2017) adopted natural language expressions such as words and phrases that may have ambiguities to represent knowledge, which made it possible to collect millions of commonsense facts in multiple languages via crowdsourcing.

3 Problem Setting

Suppose we project a fact f^s in a source language into a target language. We obtain n candidate translations by following inter-language links. In ConceptNet, the links are built from data sources such as Wiktionary and WordNet. We denote these candidates as $f_1^t, \cdots, f_n^t$.

Our goal is to estimate a projection score $h(f_i^t|f^s)$, and find the most appropriate target-side fact that maximizes the score.

$$\hat{f}^t = \underset{f_i^t}{\operatorname{argmax}}\, h(f_i^t|f^s) \tag{1}$$

4 Method

We propose two methods to combine MT and target-side KBC models for estimating projection scores.

4.1 Machine Translation (MT)

MT models consider contexts to find bilingual mapping of sentences. Given "I saw a bat in the zoo." as a source sentence, the model will assign a higher translation probability to "doubutsuen de koumori wo

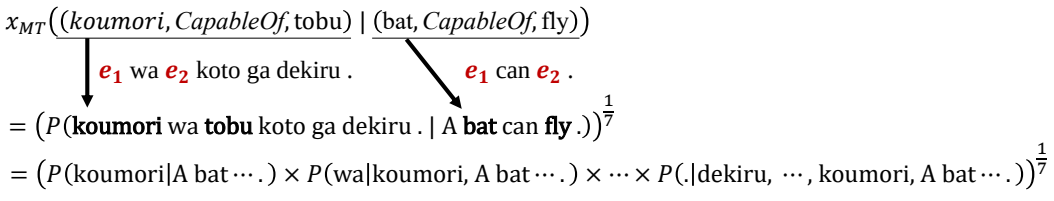


Figure 2: **Calculating a translation probability of a fact using an MT model.** For simplicity, this example does not use subword units. In addition, we omit special symbols that represent the beginning and end of a sentence in this figure.

mita." [I saw a bat (animal) in the zoo.] than to "doubutsuen de batto wo mita." [I saw a bat (stick) in the zoo.]

In contrast to our problem, typical MT focuses on plain texts, and only unstructured parallel texts are normally available for training MT models. Thus, we convert facts into natural language expressions beforehand. We use a rule-based approach to generate an expression for each fact, for example, "e_1 can e_2" corresponding to (e_1, *CapableOf*, e_2). Fortunately, some facts in ConceptNet already have such language expressions (Speer and Havasi, 2012). For the rest of the facts, we develop simple templates based on the existing expressions. Table 1 shows examples of templates in English, Japanese and Chinese.[3] We refer to part-of-speech tags of concepts to generate natural-sounding sentences.

Using sentences of facts, we define a score from the MT model as a translation probability of the language expression normalized by the target-side length m.

$$x_{\mathrm{MT}}(f^t|f^s) = \left(\mathrm{P}(W^t|W^s)\right)^{1/m}, \tag{2}$$

where W^t and W^s are sentences of f^t and f^s, respectively. To define $\mathrm{P}(W^t|W^s)$, we employ an off-the-shelf sequence-to-sequence model with an attention mechanism (Bahdanau et al., 2014), which is one of the recent successful MT models.

Figure 2 illustrates the translation probability of a Japanese fact (koumori, *CapableOf*, tobu) given an English fact (bat, *CapableOf*, fly). We first obtain language expressions of the facts using hand-crafted templates and compute a translation probability with the translation model.

4.2 Knowledge Base Completion (KBC)

KBC models evaluate the plausibility of a given fact based on existing information on the KB. For example, if we already know many animals with wings can fly and bats have wings, we can imagine that bats also can fly. We train a KBC model on the target-side KB.

We use a bilinear model used in several previous studies (*e.g.*, (Li et al., 2016)) as a component of our model, where concepts and relations are represented as vectors and matrices, respectively. This component can also be replaced with other KBC models.[4]

Given a fact $f^t = (e_1, r, e_2)$, the bilinear model outputs the value of plausibility as follows.

$$x_{\mathrm{KBC}}(f^t) = \sigma(\boldsymbol{u}_1^T \boldsymbol{M}_r \boldsymbol{u}_2), \tag{3}$$

where σ is a sigmoid function, $\boldsymbol{u}_i \in \mathbb{R}^d$ $(i = 1, 2)$ corresponds to vectors of concepts e_1 and e_2, $\boldsymbol{M}_r \in \mathbb{R}^{d \times d}$ corresponds to a matrix of relation r, and d is a hyper parameter. We construct concept vectors by averaging pre-trained d'-dimensional word embeddings as several previous studies did to boost the predictive performance (Socher et al., 2013; Li et al., 2016). The following nonlinear transformation reduces the dimensionality for computational efficiency.

$$\boldsymbol{u}_i = \tanh(\boldsymbol{W}\boldsymbol{v}_i + \boldsymbol{b}), \tag{4}$$

[3] We provide the templates in the released code.

[4] Potential alternatives can be found in the survey paper by Wang et al. (2017).

where $\boldsymbol{v}_i \in \mathbb{R}^{d'} (i = 1, 2)$ is a pre-trained concept vector, and $\boldsymbol{W} \in \mathbb{R}^{d \times d'}$ is a weight, $\boldsymbol{b} \in \mathbb{R}^d (d < d')$ is a bias term. The model parameters are learned to minimize a cross-entropy function on training facts.

4.3 Combination of Scores

We combine the two scores explained above to generate a score for each pair of f_i^t and f^s. Our model h takes $\boldsymbol{x}(f_i^t|f^s) = \left(x_{\text{KBC}}(f_i^t), x_{\text{MT}}(f_i^t|f^s)\right)$ as an input (for simplicity, we omit f_i^t and f^s, hereafter), and calculates a projection score, where x_{KBC} and x_{MT} are normalized before calculation.

We first describe two options for $h(\boldsymbol{x})$, (1) a linear transformation and (2) a multi-layer perceptron, and next explain the inference procedure of parameters below.

Linear Transformation: First, a *linear transformation* (LIN) model combines x_{KBC} and x_{MT} linearly. The model has a different weight vector and a bias term for each relation because the accuracy of KBC and MT varies for different relation types.

$$h(\boldsymbol{x}) = \boldsymbol{w}_r^T \boldsymbol{x} + b_r \quad \boldsymbol{w}_r \in \mathbb{R}^2, b_r \in \mathbb{R}. \tag{5}$$

Multi-layer Perceptron: LIN is a very simple model and may cause underfitting. Thus, we introduce a *multi-layer perceptron* (MLP) model with one hidden layer to increase the model capacity. The model has an input layer, one hidden layer, and an output layer. MLP calculates $h(\boldsymbol{x})$ by the equation below.

$$h(\boldsymbol{x}) = {\boldsymbol{w}_r^{(2)}}^T \boldsymbol{z}(\boldsymbol{x}) + b_r^{(2)} \tag{6}$$

$$\boldsymbol{z}(\boldsymbol{x}) = \tanh\left({\boldsymbol{W}^{(1)}}^T \boldsymbol{x} + \boldsymbol{b}^{(1)}\right) \tag{7}$$

$$\boldsymbol{W}^{(1)} \in \mathbb{R}^{2 \times c}, \boldsymbol{b}^{(1)} \in \mathbb{R}^c, \boldsymbol{w}_r^{(2)} \in \mathbb{R}^c, b_r^{(2)} \in \mathbb{R}.$$

Note that the common weight matrix and bias are used across relations in Equation (7) in order to capture the global intermediate representations of projections.

Inference: Given training instances of fact-to-fact projection, we estimate the model parameters that compute high scores to correct translations and low scores to incorrect translations. The training data consists of a correct translation set T_+ and an incorrect translation set T_-. $T_-(\boldsymbol{x}_+)$ denotes a set of incorrect translations that have the same English fact as $\boldsymbol{x}_+ \in T_+$.

For each $x_+ \in T_+$, we define the following margin-based loss function.

$$\text{loss}(\boldsymbol{x}_+) = \max\left(0, 1 + h(\boldsymbol{x}_+) - h(\boldsymbol{x}_-)\right), \tag{8}$$

where x_- is randomly extracted from $T_-(\boldsymbol{x}_+)$. We sum this loss function over T_+, and obtain the model parameters Θ by minimizing the summed loss function.

$$\hat{\Theta} = \underset{\Theta}{\text{argmin}} \sum_{\boldsymbol{x}_+ \in T_+} \text{loss}(\boldsymbol{x}_+) \tag{9}$$

5 Experiments

We conducted two experiments to compare our method with baseline methods.

5.1 Data

We use automatically and manually constructed datasets for evaluating knowledge projection methods. Throughout the experiments, facts are obtained from ConceptNet version 5.5.0.[5]

5.1.1 Automatically Built Datasets

The first experiment used semi-automatically built datasets of English–Japanese and English-Chinese projection. We call these AUTO datasets. The English–Japanese AUTO dataset was constructed in three steps:

[5] https://github.com/commonsense/conceptnet5/wiki/Downloads

Language	Fact (unique) f^t	Fact (unique) f^s	Translation T_+	Translation T_-
En→Ja	22,508	2,767,450	33,724	5,612,716
En→Zh	2,294	11,648	2,861	12,348

(a) AUTO

Fact (unique) f^t	Fact (unique) f^s	Translation T_+	Translation T_-
13194	200	832	12,365

(b) MANUAL (En → Ja)

Table 2: **Statistics of dataset.**

1. We translated each English fact f^s into Japanese facts $f^t_1, f^t_2, \cdots$ with inter-language links in ConceptNet. Those that violated part of speech constraints (Speer and Havasi, 2012) were discarded.
2. If a translated fact f^t_i already existed in Japanese ConceptNet, we considered the pair of the English and Japanese facts to be a positive projection, *i.e.*, $(f^s, f^t_i) \in T_+$. Otherwise, we include the pair in a set of negative projection T_-.
3. The previous step resulted in millions of obvious negative projection, which is often directed to rare Japanese words. To reduce such projection, we counted co-occurrences of all pairs of concepts in 200 million Japanese web sentences and discarded Japanese facts whose concepts do not occur together.

We applied the same procedure to Chinese facts, where we used the Chinese Gigaword Fifth Edition[6] in step 3. The size of the AUTO dataset is reported in Table 2(a). The English–Japanese dataset is larger than the English–Chinese dataset because the number of English–Japanese inter-language links in ConceptNet is four times larger than English–Chinese links. We can gain data by harvesting links from lexical resources such as dictionaries and multi-lingual WordNet, which is left as future work.

We conducted five-fold cross validation by splitting the datasets into training (60%), validation (20%) and test (20%) sets.

5.1.2 Manually Built Dataset

The AUTO datasets are large but may not be accurate enough to test methods because the target KBs were small by nature, and many true Japanese facts were not identified as correct projection in step 2. Thus, we next built an accurate but small testing dataset annotated by humans. We call this dataset MANUAL. Due to the cost constraint, this dataset was only built for English–Japanese projection.

We used crowdsourcing to annotate the data. Human workers were gathered in a Japanese crowdsourcing platform Yahoo! Crowdsourcing[7].

1. We extracted the 200 most confident English facts based on the scores in ConceptNet. We only used English concepts with fewer than 20 inter-language links.[8] In addition, we removed facts containing dirty words.[9]
2. We projected the English facts into Japanese with inter-language links as we did in step 1 of the AUTO datasets.
3. Crowd workers annotated the Japanese facts with five-level labels: (1) "false, or does not make sense", (2) "true only in a few contexts", (3) "true in several contexts", (4) "true in many contexts", and (5) "true". Each Japanese fact was judged by five workers.
4. We aggregated the collected judgments by taking median.

The size of the resulting dataset is reported in Table 2(b). An English fact had 66 translations on average. We used this dataset only for evaluation. To conduct this evaluation, we trained our models on the whole AUTO datasets.

[6]The LDC catalog number is LDC2011T13.

[7]`http://crowdsourcing.yahoo.co.jp`

[8]We found a few concepts had extremely many links (*e.g.*, `/c/en/person`), and most of the links are inappropriate.

[9]We use a dirty word list on google_twunter_lol (`https://gist.github.com/jamiew/1112488`).

	MRR	Acc@1	Acc@10
PPMI	.183	.081	.406
MT	.306	.190	.546
KBC	.352	.232	.610
MTransE	.185	.097	.372
LIN (EQ)	.363†	.233	.626†
LIN	.363†	.230	.633†
MLP	**.370**†	**.234**	**.644**†

(a) En→Ja

	MRR	Acc@1	Acc@10
PPMI	.667	.466	.980
MT	.742	.585	.982
KBC	.673	.480	.975
MTransE	.762	.626	.976
LIN (EQ)	.768	.624	.986
LIN	.766	.620	.985
MLP	**.772**	**.629**	**.988**†

(b) En→Zh

Table 3: **Results on the AUTO dataset.** † denotes LIN (EQ), LIN and MLP outperformed all the baselines significantly (paired t-test with $\alpha = 0.05$.)

5.2 Baselines and Proposed Methods

We compare the performance of our proposed methods with the following baselines.

- **PPMI:** Positive pointwise mutual information of two concepts consisting of a target-side fact f^t. We count the co-occurrence of the concepts in the 200 million web sentences for Japanese, and in the Chinese Gigaword Fifth Edition for Chinese concepts.
- **MT:** The neural MT model with an attention mechanism, which computes x_{MT} in the proposed methods. We used an implementation by Neubig (2015) and train a model on 3.25M (en-ja) and 2.97M (en-zh) sentence pairs from dictionaries and newswire corpora. BPE (Sennrich et al., 2016) was used to reduce the vocabulary size.
- **KBC:** The target-side bilinear KBC model which was used as the component to produce x_{KBC}. The Japanese and Chinese models were trained on 59,274 and 318,361 facts, respectively.
- **MTransE:** The multi-lingual translation-based KBC model (Chen et al., 2017) which learns TransE (Bordes et al., 2013) and concept-to-concept alignment jointly. Chen et al. (2017) proposed five different alignment models and reported the fourth variant performed best in their experiments. Thus we use the variant in our experiments.

The proposed methods **LIN** and **MLP** use estimates of MT and KBC models above. We also include **LIN (EQ)**, a variant of LIN, which equally combines scores from MT and KBC after normalizing each score to a $[0, 1]$-range. We use the Adam optimizer (Kingma and Ba, 2014) for training models. We provide implementation details and hyperparameter settings in the appendix.

5.3 Results

We report mean reciprocal rank (MRR), top-1 and -10 accuracy (Acc@1 and Acc@10) on the test set.[10] To calculate these metrics on the MANUAL dataset with five-way labels, we binarized the labels by considering (5) true label as positive and the others as negative because we aim to find the most appropriate projection for each English fact. We removed English facts which only had positive/negative Japanese translations when calculating MRR and accuracy. Besides, we also report nDCG (normalized discounted cumulated gain) using the five-way labels.

5.3.1 AUTO

Table 3 shows MRR, Acc@10 and Acc@1 on the test datasets for English–Japanese and English–Chinese projection. LIN (EQ), LIN and MLP outperformed MT and KBC in most cases, indicating combining them helped find correct translations. LIN (EQ) performed on par with LIN even though LIN (EQ) does not learn combination weights from the training data. This result could be attributed to the limited capacity of a linear model and motivates us to use MLP.

MTransE achieved high precision on the English–Chinese dataset, but failed to yield correct predictions on the English–Japanese dataset. The essential difference between the two language pairs is in

[10] Some previous studies reported meanrank, but MRR is more robust to outliers than meanrank.

En→Ja				
	MRR	Acc@1	Acc@10	nDCG@10
PPMI	.450	.282	.859	.632
MT	.544	.380	.866	.689
KBC	.448	.303	.775	.610
MTransE	.260	.148	.521	.473
LIN (EQ)	.600†	**.472**†	.894	.711†
LIN	.602†	**.472**†	.894	.711†
MLP	**.606**†	**.472**†	.901	**.714**†
#facts$_{\text{en}}$		142		200

Table 4: **Result on the MANUAL dataset.** † denotes LIN (EQ), LIN and MLP outperformed all the baselines significantly (paired t-test with $\alpha = 0.05$.)

Rank			Concepts		
MLP	MT	KBC	e_1	e_2	Label
1	6	8	ao [blue]	shikisai [color]	**true**
2	9	4	heki [blue]	shikisai [color]	**true**
3	2	35	brû [blue]	shikisai [color]	**true**

(a) (blue, RelatedTo, color)

MLP	MT	KBC	e_1	e_2	Label
1	5	1	rokku [rock (music)]	myûjikku [music]	**true**
2	9	2	rokku [rock (music)]	ongaku [music]	**true**
3	25	5	rokku [rock (music)]	fumen [music score]	true in several contexts
			...	...	
15	1	49	iwa [rock (stone)]	myûjikku [music]	false

(b) (rock, IsA, music)

Table 5: **Improved examples.** Top ranked projections by MLP are reported with labels and ranks given by MT and KBC.

the degree of the ambiguity, that is, the English–Chinese projection is not as ambiguous as the English–Japanese projection on our dataset because of the lack of English–Chinese links in ConceptNet v5.5.0. This characteristic boosted the performance of MTransE, which assumes one-to-one projection of concepts.

We observed the performance of the baselines varied across relations. MT was inaccurate at lexical relations such as *Antonym* and *Synonym* but outperformed KBC on *HasFirstSubevent*, *HasLastSubevent*, and *UsedFor*. MLP outpeformed single MT and KBC for the most of the relations by combining them.

5.3.2 MANUAL

Table 4 shows the result on the MANUAL dataset. The differences between the proposed methods and the baselines are statistically significant except for Acc@10 (paired t-test with $\alpha = 0.05$.) All the methods resulted in better scores on this dataset than on the AUTO dataset. Although PPMI appears to be accurate, in fact it failed to provide valid scores to 8,894 out of 13,197 examples as the co-occurrences of their concepts were not observed in the corpus. Likewise, the MANUAL dataset had many concepts that were not in the training data for MTransE, which seriously degraded its performance.

The examples in Table 5 show that MLP well combined the strength of MT and KBC models as we hypothesized. In Table 5(a), MLP put a weight on MT since it learned on the training set that MT tends to be more reliable at *RelatedTo* relation than KBC. The ratio *RelatedTo* facts was higher on the MANUAL dataset than the AUTO dataset, and we think this was the reason why MT outperformed KBC. Table 5(b) is an example in which KBC mitigated erroneous predictions by MT. MT preferred iwa [stone] to rokku [rock music], whereas KBC provided the high score to the latter. We found a pre-trained word vector of rokku was similar to those of music genres such as hevimetaru [heavy metal] and jazu [jazz], which could help KBC to identify the word sense of rock occurring with music.

Rank			Concepts		
MLP	KBC	MT	e_1	e_2	Label
1	11	5	*iso [rocky coast]	*sunago [sand/grit]	false
2	1	61	*iso [rocky coast]	suna [sand/grit]	true only in a few contexts
3	4	43	*umibe [seashore]	sand [sand/grit]	true in several contexts

(beach, *RelatedTo*, sand)

Table 6: **Failed example.** Top ranked projections by MLP are reported with labels and ranks given by MT and KBC. * denotes a rare Japanese word.

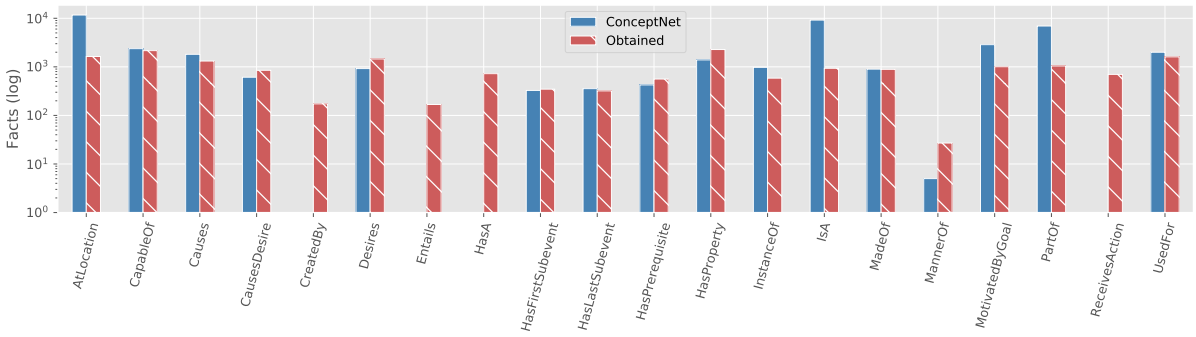


Figure 3: **Number of obtained and existing Japanese facts.**

Table 6 shows failed examples. KBC, MT, and the proposed methods produced low scores for the correct facts in these examples. This was because some negative examples contained rare words, and both MT and KBC gave them high scores. We found KBC was particularly inaccurate for facts containing OOV words. In practice, discarding rare words would be a reasonable choice in order to achieve high precision.

6 Japanese Commonsense Knowledge Construction

The previous experiments have shown our method can score projection candidates with high precision. We now use our method to collect Japanese commonsense resources of high quality.

We first sampled 10,000 English facts that cover 20 relation types.[11] In the same way as step 1 in Section 5.1.1, we obtained Japanese counterparts of them. We then used the MLP model, which achieved the best score on the MANUAL dataset, and computed scores of the projection candidates.

Although top-10 predictions are likely to contain correct projection as shown in Table 4, we further used crowdsourcing to refine the projected knowledge. We showed crowd workers top 10 confident Japanese facts that were generated from the same English fact, and the workers chose all correct Japanese facts if any. Each set of candidates was judged by five workers. Here, we converted facts into natural language by the hand-crafted templates described in Section 4.1 so that workers can easily understand the meaning. All the facts were checked by 838 workers only for 25 hours. Their annotations were aggregated by majority voting.

As a result, we obtained 18,747 facts. Note that one English fact can have multiple Japanese counterparts. Figure 3 shows the distribution of the obtained facts against the existing Japanese facts in ConceptNet. The current Japanese facts concentrated on a few relation types such as *IsA* and *RelatedTo*, and most of the relation types do not have many facts. Indeed, we have already collected an equivalent or larger amount of commonsense knowledge for 12 relation types.[12]

[11]The sampled English facts include *AtLocation*, *CapableOf*, *Causes*, *CausesDesire*, *CreatedBy*, *Desires*, *Entails*, *HasA*, *HasFirstSubevent*, *HasLastSubevent*, *HasPrerequisite*, *HasProperty*, *InstanceOf*, *IsA*, *MadeOf*, *MannerOf*, *MotivatedByGoal*, *PartOf*, *ReceivesAction*, and *UsedFor*.

[12]12 types include *CapableOf*, *CausesDesire*, *CreatedBy*, *Desires*, *Entails*, *HasA*, *HasFirstSubevent*, *HasPrerequisite*, *HasProperty*, *MadeOf*, *MannerOf*, and *ReceivesAction*.

7 Conclusion and Future Work

We proposed a method to project knowledge stored in English into other languages. We focused on commonsense knowledge that is required to understand human communications. The main challenge of cross-lingual knowledge projection is the ambiguity of projection. To resolve this ambiguity, our method combines MT and target-side KBC models. Experiments showed the proposed method outperformed baseline methods by large margins consistently. We projected 10,000 English into Japanese and obtained 18,747 accurate facts using our method and crowdsourcing. There are still more than 450,000 English facts with inter-language links to Japanese, and we are planning to project them into Japanese by our proposed method and crowdsourcing refinement. We will release the resulting resources to research communities in order to facilitate research in many languages.

Acknowledgments

We thank anonymous reviewers for their valuable suggestions. This work was partially supported by Yahoo Japan Corporation.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473:1–15.

Francis Bond and Ryan Foster. 2013. Linking and extending an Open Multilingual Wordnet. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1352–1362, Sofia, Bulgaria. Association for Computational Linguistics.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26 (NIPS)*, pages 2787–2795.

Jose Camacho-Collados, Mohammad Taher Pilehvar, Nigel Collier, and Roberto Navigli. 2017. SemEval-2017 Task 2: Multilingual and Cross-lingual Semantic Word Similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval)*, pages 15–26, Vancouver, Canada, August. Association for Computational Linguistics.

Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. 2017. Multi-lingual knowledge graph embeddings for cross-lingual knowledge alignment. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, Melbourne, Australia, November.

Xiaocheng Feng, Duyu Tang, Bing Qin, and Ting Liu. 2016. English-chinese knowledge base translation with neural network. In *Proceedings of the 26th International Conference on Computational Linguistics: (COLING)*, pages 2935–2944. The COLING 2016 Organizing Committee, December.

Jonathan Gordon and Benjamin Van Durme. 2013. Reporting bias and knowledge acquisition. In *Proceedings of The 3rd Workshop on Automated Knowledge Base Construction (AKBC)*. ACM Press, October.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. SensEmbed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 95–105, Beijing, China, July. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6, December.

Patrick Klein, Simone Paolo Ponzetto, and Goran Glavaš. 2017. Improving neural knowledge base completion with cross-lingual projections. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 516–522. Association for Computational Linguistics, April.

Yen-Ling Kuo and Jane Yung-Jen Hsu. 2010. Bridging common sense knowledge bases with analogy by graph similarity. In *Proceedings of the 2nd AAAI Workshop on Collaboratively-Built Knowledge Sources and Artificial Intelligence (WikiAI)*, pages 22–27, Atlanta, Georgia, USA, July. AAAI Press.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Chris Bizer. 2014. DBpedia – A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*.

Douglas B. Lenat. 1995. Cyc: a large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1445–1455, Berlin, Germany, August. Association for Computational Linguistics.

Hugo Liu and Push Singh. 2004a. Commonsense Reasoning in and Over Natural Language. In *Proceedings of the 8th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES)*, pages 293–306, Wellington, New Zealand. Springer, Berlin, Heidelberg.

Hugo Liu and Push Singh. 2004b. ConceptNet A practical commonsense reasoning tool-kit. *BT Technology Journal*, 22(4):211–226.

Peter LoBue and Alexander Yates. 2011. Types of Common-Sense Knowledge Needed for Recognizing Textual Entailment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 329–334, Portland, Oregon, USA, June. Association for Computational Linguistics.

Hajime Morita, Daisuke Kawahara, and Sadao Kurohashi. 2015. Morphological analysis for unsegmented languages using recurrent neural network language model. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2292–2297. Association for Computational Linguistics, sep.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1059–1069, Doha, Qatar, October. Association for Computational Linguistics.

Graham Neubig. 2015. lamtram: A toolkit for language and translation modeling using neural networks. http://www.github.com/neubig/lamtram.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 86–96, Berlin, Germany, August. Association for Computational Linguistics.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 26 (NIPS)*, pages 926–934. Stateline, Nevada, USA.

Robert Speer and Catherine Havasi. 2012. Representing general relational knowledge in ConceptNet 5. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC)*, pages 3679–3686, Istanbul, Turkey, May. European Language Resources Association.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*, pages 4444–4451, San Francisco, California, USA, February. AAAI Press.

Masao Utiyama and Hitoshi Isahara. 2003. Reliable measures for aligning japanese-english news articles and sentences. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 72–79, Sapporo, Japan, July. Association for Computational Linguistics.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, December.

Liangjun Zang, Cong Cao, Yanan Cao, Yuming Wu, and Cungen Cao. 2013. A Survey of Commonsense Knowledge Acquisition. *Journal of Computer Science and Technology*, 28(4):689–719, July.

A Experimental Setup

In experiments (Section 5), we built the baselines and our combination methods (Section 4) in the following procedures. Facts are from ConceptNet version 5.5.0. Japanese words in concepts are normalized using the morphological analyzer JUMAN++ (Morita et al., 2015). We used the Adam (Kingma and Ba, 2014) for training models. Initial learning rates α are described below.

A.1 MT Model

We built the neural MT model with lamtram (Neubig, 2015). The English–Japanese parallel corpora included 3,253,923 sentence pairs from dictionaries and newswire:

- JEC Basic Sentence Data[13]
- EDICT [14]
- Eijiro[15]
- Tatoeba Project[16]
- Open Multilingual WordNet (Bond and Foster, 2013)
- JENAAD (Utiyama and Isahara, 2003)

We extracted 2,965,845 English–Chinese sentence pairs from LDC[17]. We segmented words by byte-pair encoding (BPE) with vocabulary size 8,003 for each language. This vocabulary includes three special symbols indicating the beginning and the end of the sentence, and the out-of-vocabulary word. We trained the BPE model (Sennrich et al., 2016) on the training set with sentencepiece[18]. Hyper parameters were tuned based on perplexity on randomly selected 1,000 sentence pairs. The best setting was the encoder/decoder of LSTM with 512 hidden nodes, dropout of rate 0.25, and learning rate $\alpha = 0.001$ for the both language pairs.

A.2 KBC Model

A bilinear model (Section 4.2) was trained on 59,274 Japanese and 317,161 Chinese facts in ConceptNet. They are already sorted by a confidence score. We excluded facts in the training and evaluation datasets for knowledge projection (Section 5.1.)

Following Li et al. (2016), we define concept vectors by taking an average of predefined word embeddings. Japanese word embeddings of 256 dimensions were trained on 200 million sentences from the web. Chinese word embeddings of 300 dimensions were obtained from the CoNLL 2016 data.[19] We normalized concept vectors using the mean and variance calculated on the training set.

In hyper parameter tuning, the most confident 600 facts were used for evaluation, the second most confident 600 facts were used for early stopping in training, and the remaining facts were used for training. These facts were treated as positive examples, and we generated negative examples by randomly swapping one component of each fact. Once the best parameters were determined, the most confident 600 facts were used for early stopping, and the other facts were used to train a model.

We removed positive examples with out-of-vocabulary (OOV) words from the datasets. Instead, we added facts with a OOV vector as negative examples to the training data because rare words are often incorrect.

We selected dimensionality $d \in \{100, 150\}$, regularization coefficients for $\boldsymbol{M}_r$ and the other parameters $\lambda_1, \lambda_2 \in \{0.001, 0.0001, 0.00001\}$, learning rate $\alpha \in \{0.1, 0.01\}$, and batch size $\beta \in 200, 400, 800$.

[13] http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JEC\%20Basic\%20Sentence\%20Data

[14] http://www.edrdg.org/jmdict/edict.html

[15] ISBN: 978-4757428126

[16] https://tatoeba.org

[17] The catalog numbers are LDC2002L27, LDC2002T01, LDC2003T17, LDC2004T07, LDC2004T08, LDC2005T10, LDC2005T34, LDC2006T04, LDC2007T02, LDC2012T16, LDC2012T20, and LDC2012T24.

[18] https://github.com/google/sentencepiece

[19] CoNLL 2016 Shared Task Multilingual Shallow Discourse Parsing, http://www.cs.brandeis.edu/~clp/conll16st/dataset.html

The resulting parameters for the Japanese facts were $d = 150$, $\lambda_1 = 0.00001$, $\lambda_2 = 0.001$, $\alpha = 0.1$, and $\beta = 200$. Those for the Chinese facts were $d = 100$, $\lambda_1 = 0.001$, $\lambda_2 = 0.0001$, $\alpha = 0.1$, and $\beta = 400$.

A.3 MTransE

MTransE (Chen et al., 2017) was trained with the implementation provided by Chen et al. (2017). We set batch size 100 to speed up training although the original code adopted online learning (*i.e.*, batch size is 1). We empirically found mini batch learning did not impair the performance. We enumerated all combinations reported by Chen et al. (2017) for tuning dimensionality d, weight of the alignment model γ, norm l, learning rate α. The resulting parameters for the English–Japanese dataset are $d = 100$, $\gamma = 2.5$, $l = L2$, and $\alpha = 0.5$. Those for the English–Chinese dataset are $d = 100$, $\gamma = 2.5$, $l = L2$, and $\alpha = 0.1$.

A.4 LIN and MLP

The proposed methods were trained using estimates of KBC and MT described above. The validation set was used for early stopping in training, and selecting the best hyper parameters. We conducted a grid search for learning rate $\alpha \in \{0.5, 0.25, 0.125, 0.0625, 0.03125\}$ and dimensionality $d \in \{8, 16, 32, 64, 128\}$ (only for MLP). The resulting configurations are reported below.

- **LIN:** $\alpha = 0.125$ for the English–Japanese dataset, and $\alpha = 0.5$ for the English–Chinese dataset
- **MLP:** $\alpha = 0.5$ and $d = 16$ for the English–Japanese dataset, and $\alpha = 0.5$ and $d = 32$ for the English–Chinese dataset

Assessing Quality Estimation Models for Sentence-Level Prediction

Hoang Cuong and Jia Xu
Cuny University of New York

Abstract

This paper provides an evaluation of a wide range of advanced sentence-level Quality Estimation models, including *Support Vector Regression*, *Ride Regression*, *Neural Networks*, *Gaussian Processes*, *Bayesian Neural Networks*, *Deep Kernel Learning* and *Deep Gaussian Processes*. Beside the accurateness, our main concerns are also the robustness of Quality Estimation models. Our work raises the difficulty in building strong models. Specifically, we show that Quality Estimation models often behave differently in Quality Estimation feature space, depending on whether the scale of feature space is small, medium or large. We also show that Quality Estimation models often behave differently in evaluation settings, depending on whether test data come from the same domain as the training data or not. Our work suggests several strong candidates to use in different circumstances.

1 Introduction

Quality Estimation (QE) (Blatz et al., 2004; Specia et al., 2009) is an important topic in Natural Language Processing (NLP). QE aims to predict the quality of Machine Translation (MT) outputs without human references. QE has a lot of potential. Such a case is automatically optimizing MT systems without having reference of translation outputs. Another example is MT for gisting by users of online translation systems. QE can also reduce post-editing human effort in disruptive ways. As most QE research has conducted at sentence level, we focus on the sentence level in this study.

Prior work developed strong QE predictors by creating powerful QE features. Hand-craft and syntactic feature templates have been proposed for a while (e.g. see Specia et al. (2009) and Martins et al. (2017)). The representation of words, phrases and sentences can also be learned automatically using Neural Networks, serving as powerful QE features (Kreutzer et al., 2015; Shah et al., 2015b; Kim and Lee, 2016; Kim et al., 2017; Chen et al., 2017; Biçici, 2017; Martins et al., 2017).

Meanwhile, using a relevant QE model is also very important in QE. Various QE models have been applied in different QE settings, such as *Support Vector Regression* (Cortes and Vapnik, 1995; Specia et al., 2009), *Ride Regression* (Hoerl and Kennard, 2000; Wisniewski et al., 2013), *Neural Networks* (NNs) (Avramidis, 2017; Paetzold and Specia, 2016; Paetzold and Specia, 2017) and *Gaussian Processes* (GPs) (Rasmussen and Williams, 2005; Cohn and Specia, 2013).

Given that there are many powerful regression models, it is tempting to have an understanding of which sentence-level QE models we should use. Prior work has been attempted to provide an answer (e.g. see Cohn and Specia (2013), Soricut et al. (2012), Wisniewski et al. (2013), and Beck et al. (2016)). However, the assessment of QE models is often limited in the number of QE features. For instance, Cohn and Specia (2013) conduct experiments with only 17 QE features. Also, the assessment is in only the *Standard* setting (i.e. test data come from the same domain as training data), which is often violated in practice.

Interesting and important questions are still open to date. For instance, given different scale of feature space, it is not clear which QE models we should use and how to build them properly (e.g. how *deep*

Proceedings of the 27th International Conference on Computational Linguistics, pages 1521–1533
Santa Fe, New Mexico, USA, August 20-26, 2018.

a NN we should use). Given different training/test settings, it is also not clear which models produce more robust QE performance than others.[1] Specifically, we are interested in both *Standard* and *Domain Adaptation* (DA) settings (i.e. test data come from a different domain to the training data). We also focus on *Knowledge Transfer* (KT) setting (i.e. QE model is trained on a dataset from a specific language pair but the model is tested on a test set with a different language pair).

This work provides an evaluation of a wide range of QE models and settings as the main contribution. Not only assessing popular QE models, we also propose to use several novel models for QE including *Bayesian Neural Networks* (Neal, 1996), *Deep Kernel Learning* (Wilson et al., 2016b), *Deep Gaussian Processes* (Damianou and Lawrence, 2013). Our results also raise concern about overfitting in applying QE models in different settings. We also show how complex the interactions are between QE features. Meanwhile, we also show that our proposed models work very well in certain circumstances.

In detail, we summarize the most interesting findings as follows. First, Support Vector Regression, despite being considered a strong QE model, is far from achieving top performance. It also provides a rather weak performance in DA and KT setting. Meanwhile, our experiments show that the QE feature space is highly complex. Specifically, we show that while the feature space could be large, most of the features in the space are usually useful so that applying sparse QE models is often less effective than non-sparse QE models.

Second, a shallow NN seems to gain a strong performance in the Standard setting. This is interesting, as previous attempt to use shallow NNs for QE (e.g. see Avramidis (2017)) failed to deliver a competitive result in the standard WMT setting (see Bojar et al. (2017) for a reference). Perhaps, their network parameter settings are not good enough to make the model work. We also attempt to use deep NNs for QE, and show that they do not contribute a better performance in the Standard setting. This is because training deep NNs for QE requires lots of labeled data to control the risk of overfitting.

Increasing the number of units in the hidden layer larger could degrade the performance. Specifically, we show that training a large NN could be overfitting, and available QE datasets are not large enough to reduce the risk. But can we still make a very large NN work for QE? This paper shows that we can do so by using Gaussian Processes (GPs).

We investigate GPs from the viewpoint of a non-parametric version of shallow NNs, and analyze their advantages over NNs: GP has an infinite number of hidden units in the hidden layer, GP has a built-in feature weighting mechanism and model is more robust to over-fitting. We also analyze their disadvantages, showing the cases where GPs completely fail to deliver good results. Finally, we also propose a solution to address their drawbacks.

Third, we observe a very weak performance of NNs in the DA and KT setting. While we are aware of concerns of applying NNs to different domains in other NLP tasks (e.g. Machine Translation (Koehn and Knowles, 2017) and Sentiment Analysis (Radford et al., 2017)), this is the first study to raise this issue in QE. We propose three methods that make NNs work well with the challenging setting.

First, we propose to use deep NNs for QE to DA and KT settings. We attribute this observation to the fact that deep NNs learn high-level (instead of low-level) features, which may work well across domains/datasets. Second, we propose to use GPs under these conditions because of their robustness.

These models, however, work well only when we train them with a small set of features. We then propose to use a combination of the two, a.k.a Deep Kernel Learning, to address the problem. For this deep model, the GP is put on top of a NN, aiming to combine the best of both worlds. We show that it provides an even stronger performance.

Third, we propose to use Bayesian NNs to the DA and KT setting. It is a type of neural networks with a prior distribution on its weights. The common perception is that it is hard to apply the model to NLP because of its difficulty in training. This paper shows that it is not the case, thanks to recent advances in Variation Inference (with stochastic variational inference (Hoffman et al., 2013), black-box variational inference (Ranganath et al., 2014) and the reparameterization trick (Kingma and Welling, 2014; Rezende et al., 2014)). We show that Bayesian NNs provide another powerful solution to the problem.

[1] We should note that we focus on the accurateness and robustness of QE models. However, calculation cost is not considered in comparing QE methods.

2 Related Work

Developing different regression QE model at sentence-level has been one of the main focuses in QE. *Support Vector Regression* is perhaps the most popular QE sentence-level model (Specia et al., 2009; Soricut et al., 2012; Kozlova et al., 2016). *Gaussian Processes* (GPs) (Cohn and Specia, 2013; Shah et al., 2013; Beck et al., 2015; Beck et al., 2016) and *Ride Regression* (Wisniewski et al., 2013) are also popular models in QE. NNs have been widely deployed as a non-linear classification model in QE (e.g. see Blatz et al. (2004), Esplà-Gomis et al. (2015) and Esplà-Gomis et al. (2016)). Recently, NNs have also been attempted to use as a regression QE model at sentence-level (Avramidis, 2017; Paetzold and Specia, 2016; Paetzold and Specia, 2017). Other less popular models are proposed, e.g. Regression Trees (Soricut et al., 2012; Wisniewski et al., 2013), Extremely Randomized Trees (Negri et al., 2014).

Given different QE models, it is tempting to have an understanding of which QE models we should use. There have been several attempts to answer this important question (e.g. (Cohn and Specia, 2013; Soricut et al., 2012; Wisniewski et al., 2013; Beck et al., 2016)). For instance, Cohn and Specia (2013) and Beck et al. (2015) show that choosing a model between GPs and SVR could make a difference. Meanwhile, Avramidis (2017) compares NNs with SVM. The comparison, however, limits in only a specific pair of models (e.g. GPs-SVR and SVR-NNs). Experiments are also with only medium-size feature space (17 features) and in the Standard setting. We extend the work extensively in regards to having a larger number of QE models, and having a systematic manner with different settings (testing with different feature space, different training/test settings). Moreover, our results not only provide a better understanding in regards to QE Models for sentence-level prediction, but also raise concern about overfitting in applying QE models in different settings.

Our work also puts attention to the setting where test data comes from different distributions or domains to the training/dev data (i.e. DA and KT). We share the same views with the work of Ríos and Sharoff (2016), Shah and Specia (2016), de Souza et al. (2014a), de Souza et al. (2014b) and de Souza et al. (2015) in regards to this setting. Specifically, we believe utilizing a dataset from one specific distribution/domain to use for other distribution/domain is very useful. The related studies focus on using multiple task learning (Shah and Specia, 2016; de Souza et al., 2014a), or utilizing unlabeled data (Ríos and Sharoff, 2016) to address the problem. Meanwhile, our work focuses on assessing the robustness of QE models to improve model performance in this setting.

Prior work revealed that the interactions between QE features are non-linear (e.g. see Shah et al. (2015a)). We extend the result in the sense that we show the interactions are not only non-linear but also highly complex. However, we show that we do not need a *non-stationary* function to model the relationship (i.e. whether there is a jump discontinuity or an isolated tall peak).

A distant research line is *feature selection*. Specifically, Soricut et al. (2012) use a computationally intensive method to find all 2^{24} possible combinations, from an initial set of 24 features to find the best combinations. Another distantly related work focuses on developing non-linear methods for feature selection in QE (Shah et al., 2015a). While the research line is distant to our study, we share the same views with the studies in that a complex combination of features, rather than a simple linear combination, may bring significant benefits to QE.

Our work also introduces Deep Kernel Learning, a hybrid NN with GPs, as a regression model to not only QE but also NLP for the first time. This combination model has been attempted recently in the Machine Learning community, pioneered by the work of Wilson et al. (2016a) and Al-Shedivat et al. (2017). Specifically, Wilson et al. (2016a) apply the framework to *airline* and *image* classification. Al-Shedivat et al. (2017) apply the framework to different regression problems in Machine Learning. Recently, Bradshaw et al. (2017) also apply the framework to image classification and *transfer* learning.

3 Models

This section contributes a survey of different models we used in the study.

3.1 Support Vector Regression

Support Vector Regression (SVR) is a standard model in QE. Our experiments are with a non-linear Radial Basis Function (RBF) kernel. Model parameters are optimised via grid search with 5-fold cross validation on the training set. SVR is a standard QE model. It is widely considered a very strong and robust model. However, this paper shows that some models can do far better.

3.2 Kernel Ridge Regression

The form of the model learned by Kernel Ridge Regression (KRR) is identical to SVR. However, KRR uses squared error loss while SVR uses epsilon-insensitive loss. The learned SVR is thus sparse while KRR is non-sparse. We are interested in KRR because a comparison between the model and SVR gives some hints about the feature space. If a sparse model (SVR) produces a better performance, the feature space may contain lots of irrelevant features. If this is not the case, then the feature space is highly complex as most of the features in the space are useful.

3.3 Neural Networks

Assessing the fitness of NNs to QE is the core of our study. We first study shallow NNs with three layers: 1 input layer, 1 hidden layer and 1 output layer. Prior work (e.g. see Avramidis (2017)) has tried shallow NNs, but failed to deliver a competitive result in the standard WMT setting (see Bojar et al. (2017) for a reference). This work, however, shows that shallow NNs bring competitive performance in the Standard setting. Extensive experiments are also conducted with deeper NNs with more hidden layers (e.g. 2, 3, 4, 5). Our goal is to investigate, for the first time, how deep models can help QE.

3.3.1 Gaussian Processes

Giving an infinite number of units plus a prior over weights and a Bayesian interpretation altogether turns Neural Networks to *Gaussian Processes* (GPs). Specifically, HTER scores Y are represented as a sample from a multivariate Gaussian distribution from input X as $Y \sim \mathcal{N}(0, K(X, X)$. Here, its mean is a zero-vector with length N as the number of data points, and K is an $N \times N$ matrix that we get by applying the kernel function to our data points.

The kernel function estimates the similarity of observed inputs. The intuition here is that the closer data points are, the closer their prediction is to the others. We use the standard Squared Exponential (SE) kernel function in our work:

$$k(x_i, x_{i'}) = \delta^2 \exp\Big[-\frac{1}{2}\sum\nolimits_{d=1}^{D}\Big(\frac{x_{i,d} - x_{i',d}}{\lambda_d}\Big)^2\Big]. \quad (1)$$

Here, $x_{i,d}$ denotes the d^{th} feature value of data point x_i, and D denotes the total number of features. The output variance δ and lengthscale λ_d are hyperparameters.

We now discuss how to do regression with GPs. First, note that for each new input x^*, its corresponding output y^* forms a distribution:

$$\begin{bmatrix} Y \\ y^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) & K(X,x^*) \\ K(x^*,X) & K(x^*,x^*) \end{bmatrix}\right). \quad (2)$$

We then can derive the conditional distribution $P(y^*|Y)$ as $P(y^*|Y) = \mathcal{N}(\bar{y^*}, \sigma^*)$, where: $\bar{y^*} = K(x^*, X)K(X, X)^{-1}Y$ and $\sigma^* = K(x^*, x^*) - K(x^*, X)K(X, X)^{-1}K(X, x^*)$. The model is robust because they marginalise over an infinitely large class of models.

For learning and prediction with GPs the inverse of the covariance matrix needs to be computed. This inversion of a matrix scales with the number of training data points, N, as O(N^3). Inference with a GP model is thus expensive, but this can be addressed by learning a small number M of pseudo data points and using them to train the model instead (Snelson and Ghahramani, 2006).[2] In other words, we can

[2]We can also use a very large but placed in a computationally efficient structure instead of a small number of pseudo data points (Wilson et al., 2015; Wilson and Nickisch, 2015).

approximate the inverse of the covariance matrix with the inverse of the $M \times M$ "pseudo matrix" instead. For this reason, we do not observe any problem with scaling GPs to QE datasets in our experiments. For our implementation, GPs are implemented in the standard GPflow, a Gaussian process library using TensorFlow (de G. Matthews et al., 2016).

3.3.2 Deep Gaussian Processes

This work will show that the relationship between the space of input features and corresponding output is non-linear and highly complex. A further question we asked is whether we need a *non-stationary* function to model the relationship (i.e. whether there is a jump discontinuity or an isolated tall peak). One additional contribution of this work is to provide an understanding of this issue.

Specifically, we introduce another kind of deeper NNs for QE, namely *Deep Gaussian Processes* (Damianou and Lawrence, 2013). Deep GPs combine GPs with deep architectures. Let us denote a GP model for Y as $Y \sim GP(0, K(X, X))$. Instead of having only a simple layer of GP, we can stack many more. This is mathematically equivalent to having the form:

$$Y \sim f^L(f^{L-1}(\dots f^1(X,\ X)\dots)), \tag{3}$$

where $f^l(\cdot) \sim GP(0, K^l(\cdot, \cdot))$. Stacking many more layers helps the model able to learn to approximate non-stationary functions (e.g. see Damianou and Lawrence (2013) and Vafa (2016)).

Training the model is more complicate, but doable with recent advanced approximate methods (e.g. Approximate Expectation Propagation (Bui et al., 2016), Doubly Stochastic Variational Inference (Salimbeni and Deisenroth, 2017)). In this work we choose the work of Salimbeni and Deisenroth (2017) to train the model because of its simplicity and efficiency.

3.3.3 Deep Kernel Learning

In theory, GP with the SE kernel function is an universal approximator (Micchelli et al., 2006). Unfortunately, the representational power offered by the kernel can be very limited as the kernel is perhaps too simple. This may make it less effective when applying the model to a high dimensional QE feature space. It is not easy to address the problem.

In this work, we propose to use Deep Kernel Learning (DKL) to address the challenge. This type of network is a deep network: it has input layer, hidden layers, and a GP model on the top. The formulation for the model is thus $Y \sim GP(0, NN(X))$ instead of $Y \sim GP(0, K(X, X))$, where $NN(X)$ represents a NN instead of a simple kernel function $K(\cdot, \cdot)$. In our experiments, DKL is a shallow NN plus a GP model stacked on the top.

Technically, all GPs parts are differentiable (Rasmussen and Williams, 2005), including the Cholesky decomposition that computes the inverse of the $M \times M$ "pseudo matrix". For this reason, the model can be trained in an end-to-end fashion with back-propagation. Specifically, the model can be trained end-to-end using stochastic variational approach (Wilson et al., 2016a) or semi-stochastic block-gradient approach (Al-Shedivat et al., 2017). We choose the work of Al-Shedivat et al. (2017) to train the model because of its efficiency.

We propose to use DKL for QE because of two potential benefits. First, such a combination may work well with high dimensional input space since the flow goes from a NN to the GP layer. Second, it may be more robust because the model enjoys the robustness of GPs in general. Indeed, while deep NNs are found to be overfitting, DKLs are found to be very robust in our experiments.

3.3.4 Bayesian Neural Networks

Bayesian NNs are a type of NNs with a prior distribution on the weights. We introduce the model to QE because of their robustness.

Specifically, we put network weights $\mathbf{w}$ with normal priors $p(\mathbf{w}) = \mathcal{N}(0, I)$ in our experiments. Pre-

dictive distribution of output y^* given a new input x^* can be computed as follows:

$$p(y^*|\ x^*, X, Y) = \int p(y^*|\ x^*, \mathbf{w}) p(\mathbf{w}|X, Y) d\mathbf{w}.$$

$$p(\mathbf{w}|X, Y) = \frac{p(\mathbf{w}) \prod_{i=1}^{N} p(y_i|x_i, \mathbf{w})}{p(Y|\ X)}.$$

Given a new input x^*, Monte Carlo estimation can be used to get an unbiased estimate of it by sampling from the variational posterior

$$p(y^*|x^*, X, Y) \simeq \frac{1}{M} \sum_{i=1}^{M} p(y^*|\ x^*, \mathbf{w}_i). \tag{4}$$

where each $\mathbf{w}_i$ is sampled from $p(\mathbf{w}|X,\ Y)$. We use Bayesian NNs with 50 hidden units in our experiments. We found that using a larger number of units hurts the performance.

The model comes with a cost: it is non-trivial to compute the posterior distribution of network parameters $p(\mathbf{w}|X, Y)$ because of the intractable marginal distribution $p(Y|\ X)$. Variational Inference can be used to address the problem. It uses a variational distribution $q_\theta(\mathbf{w}) = \prod_{i=1}^{L} q_{\theta_i}(\mathbf{w}_i)$ to directly approximate $p(\mathbf{w}|X, Y)$. Each $q_{\theta_i}(\mathbf{w}_i)$ is basically a normal distribution paramaterized by mean and standard deviation:

$$q_{\theta_i}(\mathbf{w}_i) = \mathcal{N}(\mathbf{w}_i|\mu_i, \delta_i^2) \tag{5}$$

Training the model is efficient, thanks to stochastic variational inference (Hoffman et al., 2013) and the reparameterization trick (Kingma and Welling, 2014; Rezende et al., 2014)).

As a side note, our Bayesian NNs implementation is based on ZhuSuan, a well-developed framework for Bayesian Deep Learning (Shi et al., 2017).

4 Experiments Settings

Our training data is a collection of a pair of source/target sentences together with a HTER score (Snover et al., 2006) - the minimum edit distance between the machine translation and its manually post-edited version. All the datasets we conducted experiments in our study are standard WMT shared task datasets. The datasets are all publicly available. We also release our implementation for the code,[3] making it available for reproducing results.

This work studies three different QE scenarios:

- Standard Setting: The test data comes from the same distribution as the training/validating data.
- Domain Adaptation Setting: The test data comes from a different distribution as the training/validating data.
- Knowledge Transfer Setting: The test data comes from a different language pair as the training/validating data.

Previous QE studies mainly focus on Standard setting. Our focus is also on DA setting, because we inevitably will come across data that is sampled from a different distribution to our training data when using QE models in the wild. Closely related to this is fine-tuning models on a new domain. However, we are interested in the aspect of NOT fine-tuning our models. This is because information of test data is not provided in advance when applying a QE model in the wild.

We are also interested in KT setting. Utilizing a dataset from one language pair to use for other language pairs is very useful, given that QE datasets are limited and language-specific.

Our experiments are with different feature sets. For medium-scale QE feature space, we use the standard 17 QE features (see Bojar et al. (2017)). For large-scale QE feature space, we enrich the feature

[3]The code is available at: *https://github.com/hoangcuong2011/QESentlevel*.

Model	Standard Setting		Domain Adaptation	Knowledge Transfer	
	EN-DE	DE-EN	Train/Valid: WMT EN-DE 2016 Test: WMT EN-DE 2017	Train/Valid: WMT DE-EN 2017 Test: WMT EN-DE 2017	Train/Valid: WMT DE-ES 2015 Test: WMT EN-DE 2017
Linear Model	23.48	29.47	–	–	–
Support Vector Regression	17.47	17.79	20.28	20.69	22.68
Kernel Ridge Regression	17.41	17.47	20.55	28.67	21.19
Shallow Neural Networks	17.46	17.15	20.81	24.96	23.46
Deep Neural Networks	17.51	17.32	19.45	22.85	20.51
Gaussian Processes	17.39	17.05	20.23	21.03	21.12
Deep Gaussian Processes	17.70	17.36	20.20	22.46	21.15
Deep Kernel Learning	17.35	17.28	19.68	21.93	19.93
Bayesian Neural Networks	17.57	17.76	20.65	21.64	23.77
Deep Bayesian Neural Networks	18.44	18.34	18.83	21.19	20.23

Table 1: Assessing QE Models for Sentence-Level Prediction (Medium-scale QE feature space).

set using unsupervised learning. Specifically, we trained a multiplicative LSTM (Krause et al., 2017) with $4,096$ units on a very large public *English* corpus (see McAuley et al. (2015)) to predict the next character for a string. These $4,096$ units are found to be meaningful in NLP tasks in general such as Sentiment Analysis (Radford et al., 2017). This work shows that they are very useful to QE as well. While we can add all units as extra features, this turns out to be expensive. We rather use only a *random* subset of them (53 and 123 units). In the end, our large-scale QE feature space experiments are with two feature sets with 70 and 140 features respectively.[4]

5 Standard Setting

We first present results with WMT QE 2017 shared task (EN-DE and DE-EN). There are various evaluation metrics that are often used in Quality Estimation, including Pearson's correlation, Mean Average Error and standard Root Mean Squared Error. Because of space constraints, we report only root-mean-square error (RMSE):

$$RMSE = \sqrt{\frac{\sum_1^N (HTER_{ref} - HTER_{pred})^2}{N}} * 100, \quad (6)$$

where N is the size of test data. Meanwhile, we emphasize that the important findings in our work are consistent with the other metrics.

5.1 Medium-scale setting

Table 1 presents the result with medium-scale QE feature space. We detail most important findings.

First, in this setting, we do not observe important advantages of using different models. There is not a clear winner, and the difference between the best model and other top models is quite *marginal*. As a site note, SVR is far from achieving top performance (e.g. 17.79 vs 17.01 for WMT QE DE-EN 2017 task).

Meanwhile, Table 1 reveals that the relationship between the space of input features and corresponding output is fully non-linear. Specifically, having a linear QE model provides a far suboptimal performance (e.g. 23.48 with EN-DE), compared to other complex models (e.g. Deep Kernel Learning: 17.35. We also do not observe any significant improvement by using SVR instead of KRR. The fact that a sparse model (SVR) does not help over a non-sparse model (KRR) indicates most features are useful. Combining with the fact that the feature space is non-linear, the QE feature space seems highly complex.

Shallow NNs with 512 units produce a competitive result. This is interesting, as previous attempt to use shallow NNs for QE (e.g. see Avramidis (2017)) failed to deliver a competitive result in the standard WMT setting (see Bojar et al. (2017) for a reference). Perhaps, their network parameter settings are not good enough to make the model work. For the sake of completeness, our network parameter setting is

[4]Surprisingly, we found that training a good QE model only with the set of 53 or 123 extra features improves the performance over the WMT 2017 shared task baseline (Bojar et al., 2017). These units are indeed very meaningful.

as follows. The models were regularized with dropout (Srivastava et al., 2014) with rate 0.5 and were trained with Adam (Kingma and Ba, 2014) with rate 0.0001.

Meanwhile, Bayesian NNs do not shine with this setting. Given that Bayesian NNs are suitable to the scenario of having little training data, we conclude that the quantity of being "little" should be much smaller than our QE training data. (Our training data are with around $10K$-$20K$ sentences.) Deep NNs and deep Bayesian NNs do not help much with Standard setting. Table 2 presents results with other settings of hidden layers for deep NNs.

DNNs	EN-DE	DE-EN	NNs	EN-DE	DE-EN	DGPs	DE-EN	GPs	DE-EN	EN-DE
1 Layer	17.46	17.15	512	17.46	17.15	1 Layer	17.01	w. FW	17.36	17.01
2 Layers	17.47	17.26	1024	17.44	17.14	2 Layers	17.28	w.o. FW	17.72	17.56
3 Layers	17.51	17.29	16384	17.51	17.34	3 Layers	17.36			
4 Layers	17.48	17.37	32768	17.40	17.32	4 Layers	17.22			
5 Layers	17.54	17.35	Infinite (GP)	17.36	17.05	5 Layers	17.34			

Table 2: Detailed analyses with different models in the Medium-scale setting. Specifically, using deep NNs (DNNs) does not help much. Using large NNs improves the performance, but only if the model is robust in training (i.e. as in GPs). It is also unlikely to need a QE model that is capable of approximating non-stationary functions, as we found deep GPs (DGPs) does not gain any significant improvements. Finally, feature weighting (F.W.) mechanism in GPs is also important. Turning it off rather gives worse results.

GPs are competitive with Standard setting and medium-scale QE feature space. We recall that GP is a non-parametric version of a shallow NN with an infinite number of units in the hidden layer (Neal, 1996). Detail analyses confirm that it is a good fit to medium-scale QE feature space because of the following reasons. First, having a substantially larger number of hidden units may help. However, we notice that having a larger NN may not improve performance, as in Table 2. That is, it helps only the case when the model is robust in training, as in GPs.

Second, most features in the feature space are useful. However, each of them should have a certain degree of relevance to the task. GPs can capture this. As in Eq 1, if λ_d is larger, then the d^{th} feature is less important. Meanwhile, a small λ_d indicates that the feature is crucial to the task. To this end, we noticed that their values are different in our experiments, and neither of them are too large or too small. We also noticed that once we use only a single hyper-parameter λ for all the features, we observe a significant drop in performance, as shown in Table 2. This confirms that the feature weighting mechanism is also important in GPs.

Meanwhile, deep GPs do not provide any improvement over GPs. We provide detailed statistics with other configurations of deep layers in Table 2. It is therefore unlikely to need a QE model that is capable of approximating non-stationary functions. In other words, the relationship between the space of input features and corresponding output is unlikely to have jump discontinuities or isolated tall peaks.

5.2 Large-scale setting

We now turn our attention to large-scale setting.[5] Table 3 presents the result in detail. Here, we do not report KRR and Deep GPs because they seem not to be a right model for QE. [6]

The three most important observations are as follows. First, it is surprising to see that there is also not a clear winner, and the difference between the best model and other top models is also quite *marginal*. Second, GPs do not provide competitive results with this setting. This is because simple SE kernel function is not representative enough to learn the similarity in high-dimensional QE feature space. We also tried other advanced kernel functions, such as Matern kernel (Rasmussen and Williams, 2005), Polynomial kernel (Rasmussen and Williams, 2005), mixture of SE kernels (Wilson and Adams, 2013) and Additive Kernels (Duvenaud et al., 2011) without success.

[5]We report the Standard setting for large-scale experiments only for DE-EN. Specifically, the pretrained LSTM model is made available at *https://github.com/openai/generating-reviews-discovering-sentiment* by Radford et al. (2017). Our large-scale feature experiments are thus reproducible.

[6]Our result with these models in the setting is also not promising.

Model	Standard Setting Train/Valid/Test: WMT DE-EN 2017	Knowledge Transfer Train/Valid: WMT DE-EN 2017 Test: WMT EN-ES 2013	Domain Adaptation Train/Valid: WMT DE-EN 2017 Test: WMT DE-EN 2013
70 features			
Support Vector Regression	16.71	38.84	43.68
Shallow Neural Networks	16.39	36.02	142.27
Deep Neural Networks	16.80	43.15	95.71
Gaussian Processes	24.37	62.05	60.14
Deep Kernel Learning	16.37	37.11	44.13
Bayesian Neural Networks	16.94	32.71	42.37
Deep Bayesian Neural Networks	16.99	33.51	46.46
140 features			
Support Vector Regression	17.23	38.99	45.03
Shallow Neural Networks	16.52	36.22	257.30
Deep Neural Networks	16.82	42.44	131.81
Gaussian Processes	24.37	62.05	60.14
Deep Kernel Learning	16.82	35.44	40.99
Bayesian Neural Networks	17.22	36.39	45.48
Deep Bayesian Neural Networks	17.00	36.60	59.12

Table 3: Assessing QE Models for Sentence-Level Prediction (large-scale QE feature space)

Perhaps, making GPs work well with this setting needs a fully different type of kernel function. As shown in Table 3, we found that DKL seems to be a right approach to address the difficulty. That is, replacing the simple SE kernel function with a NN improves the performance of GPs significantly. Overall DKL and shallow NNs provide the best result with large-scale QE feature space.

In summary, we recommend using shallow NNs and DKL for QE with Standard setting. We recommend using GPs only with medium-scale QE feature space. We recommend NOT using a deep NN in QE because of overfitting. Our results also show that using a suitable model is very important. For instance, good models (shallow NN and DKL) with a set of 70 features produce competitive results (16.39 and 16.37) to the best *single* model in WMT QE 2017 shared task (SHEF/QUEST-EMB-SCALE - 16.1) (see Bojar et al. (2017) for a reference). Other models including SVR are far from achieving top performance.[7]

6 Domain Adaptation and Knowledge Transfer

We study different DA and KT settings. Specifically, we use EN-DE training/validating data come from WMT 2016, and evaluate on WMT 2017 EN-DE test data. We also use WMT 2017 DE-EN training/validating data to train the model, but we evaluate on WMT 2013 DE-EN test data. Note that with the EN-DE training/validating data come from WMT 2016, the datasets contain HTER scores greater than 100. We therefore scale the HTER score in $[0, 1]$ by dividing the maximum HTER score.

In KT setting, we experiment with WMT 2015 EN-ES and WMT 2017 DE-EN training/validating data sets, and evaluate on WMT 2017 EN-DE test data. We also train models using WMT 2017 DE-EN training/validating data, but test on WMT 2013 ES-EN test data.

Tables 1 and 3 present results. There are several important findings as follows. First, there is a substantial difference in model performance in these settings. Second, we raise our concern of applying shallow NNs to these settings. Specifically, shallow NNs often predict an HTER score that is outside the range of [0:1] given an input that is not close to the input space of training data. This explains a substantially worse RMSE with 142.27 and 257.30 in Table 3.[8] While analysis of these cases requires future work, we should note that we did not observe similar problems for other models.

Using deep NNs for QE could reduce the effect of DA and KT setting. We attribute this observation

[7]We recall that we randomly selected 53 units from our $4,096$ units. We notice that by selecting top most relevant 53 units (by feature selection), we improve the performance of shallow NN and DKL to 16.1. However, our main concerns are assessing QE models rather than beating SOTA.

[8]Technically, we can "post-process" the output so that a prediction that is outside the range of [0:1] will be set to 0 or 1. We do not apply such post-processing procedures here.

to the fact that deep NNs learn high-level features, which may work well across domains/datasets. Unfortunately, this seems to be only the case with medium-scale QE feature space. With large-scale setting deep NNs produce a suboptimal performance. It is indeed non-trivial to train deep NNs with the settings given small training data we have. Deep NNs also often produce HTER scores that are outside the range of $[0{:}1]$ with these settings.

Third, using GPs for QE gains robust performance across various medium-scale QE feature space adaptation/knowledge transfer tasks. Unfortunately, this is not the case with large-scale setting.

Using Bayesian NNs and DKL provides a powerful solution to the DA and KT setting. Robustness is the key to their success under these settings. As a side note, deep Bayesian NNs provide good solution to DA and KT setting only with medium-scale setting.

Finally, our work shows if we choose a right QE model to apply in the wild, the QE prediction is robust. For instance, the performance of using deep NNs, DKL and deep Bayesian NNs is still very promising (19.45, 19.68 and 18.83 respectively) when they are trained with WMT EN-DE 2016 dataset but tested with WMT EN-DE 2017 test data.

Moreover, our results show that it is promising to utilize a QE dataset from one language pair to use for other language pairs. Let us present our good results with the WMT QE 2017 EN-DE test set as an example. Our result with DKL trained with WMT DE-ES 2015 is 19.93 while the best QE model trained on the in-domain training/validating data is 17.35.

Our work also raises the challenge of building a better QE model with a large feature set. As in Table 3, model performance is often far from being robust/accurate across different domains/datasets.

7 Conclusion

This paper contributes a significant amount of efforts of assessing advanced sentence-level QE models. We show that having powerful features is not enough, as we also need a right model to use. We recommend to use shallow NNs with Standard setting, but we do NOT recommend deep NNs with this setting. When it comes to medium-scale QE feature space, we recommend using GPs.

Meanwhile, we observe a very weak performance of NNs when applying the model in the wild (i.e. Adaptation and Knowledge Transfer setting). GPs model is a right option in these cases, but only with medium-scale setting. Bayesian NNs work well for both DA and KT settings, but not Standard setting.

Each model works well in certain circumstances. There is an exception, though, with Deep Kernel Learning. We found that the QE model is a strong candidate to use in real-world scenario.

Acknowledgements

We would like to thank reviewers for their insightful comments on the paper. This research was partially supported by National Science Foundation (NSF) Award No. 1747728 and National Science Foundation of China (NSFC) Award No. 61672524. We express our gratitude to the Provost office, the Dean's office, the Department of Computer Science, and the Research Foundation of Hunter College at CUNY for partially funding our research.

References

Maruan Al-Shedivat, Andrew Gordon Wilson, Yunus Saatchi, Zhiting Hu, and Eric P. Xing. 2017. Learning scalable deep kernels with recurrent structure. https://arxiv.org/abs/1610.08936.

Eleftherios Avramidis. 2017. Sentence-level quality estimation by predicting hter as a multi-component metric. pages 534–539. Association for Computational Linguistics.

Daniel Beck, Trevor Cohn, Christian Hardmeier, and Lucia Specia. 2015. Learning structural kernels for natural language processing. *Transactions of the Association for Computational Linguistics*, 3:461–473.

Daniel Beck, Lucia Specia, and Trevor Cohn. 2016. Exploring prediction uncertainty in machine translation quality estimation. In *CoNLL*, pages 208–218. ACL.

Ergun Biçici. 2017. Predicting translation performance with referential translation machines. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 540–544, Copenhagen, Denmark, September.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence estimation for machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 169–214, Copenhagen, Denmark, September.

John Bradshaw, Alexander G. de G. Matthews, and Zoubin Ghahramani. 2017. Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks. https://arxiv.org/abs/1707.02476.

Thang D. Bui, José Miguel Hernández-Lobato, Daniel Hernández-Lobato, Yingzhen Li, and Richard E. Turner. 2016. Deep gaussian processes for regression using approximate expectation propagation. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1472–1481. JMLR.org.

Zhiming Chen, Yiming Tan, Chenlin Zhang, Qingyu Xiang, Lilin Zhang, Maoxi Li, and Mingwen WANG. 2017. Improving machine translation quality estimation with neural network features. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 551–555, Copenhagen, Denmark, September.

Trevor Cohn and Lucia Specia. 2013. Modelling annotator bias with multi-task gaussian processes: An application to machine translation quality estimation. In *ACL (1)*, pages 32–42. The Association for Computer Linguistics.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September.

Andreas C. Damianou and Neil D. Lawrence. 2013. Deep gaussian processes. In *AISTATS*, volume 31 of *JMLR Workshop and Conference Proceedings*, pages 207–215. JMLR.org.

Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. 2016. Gpflow: A gaussian process library using tensorflow. https://arxiv.org/abs/1705.08933.

José G. C. de Souza, Marco Turchi, and Matteo Negri. 2014a. Machine translation quality estimation across domains. In *COLING*, pages 409–420. ACL.

José G. C. de Souza, Marco Turchi, and Matteo Negri. 2014b. Towards a combination of online and multitask learning for mt quality estimation: a preliminary study.

José Guilherme Camargo de Souza, Matteo Negri, Elisa Ricci, and Marco Turchi. 2015. Online multitask learning for machine translation quality estimation. In *ACL (1)*, pages 219–228. The Association for Computer Linguistics.

David K Duvenaud, Hannes Nickisch, and Carl E. Rasmussen. 2011. Additive gaussian processes. In *Advances in Neural Information Processing Systems 24*, pages 226–234. Curran Associates, Inc.

Miquel Esplà-Gomis, Felipe Sánchez-Martínez, and Mikel L. Forcada. 2015. Using on-line available sources of bilingual information for word-level machine translation quality estimation. In *Proceedings of the 18th Annual Conference of the European Association for Machine Translation*, pages 19–26, Antalya, Turkey, May.

Miquel Esplà-Gomis, Felipe Sánchez-Martínez, and Mikel Forcada. 2016. Ualacant word-level and phrase-level machine translation quality estimation systems at wmt 2016. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 782–786. Association for Computational Linguistics.

Arthur E. Hoerl and Robert W. Kennard. 2000. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 42(1):80–86.

Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. 2013. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347.

Hyun Kim and Jong-Hyeok Lee. 2016. Recurrent neural network based translation quality estimation. In *Proceedings of the First Conference on Machine Translation*, pages 787–792, Berlin, Germany, August.

Hyun Kim, Hun-Young Jung, Hong-Seok Kwon, Jong-Hyeok Lee, and Seung-Hoon Na. 2017. Predictor-estimator: Neural quality estimation based on target word prediction for machine translation. *ACM Trans. Asian & Low-Resource Lang. Inf. Process.*, 17(1):3:1–3:22.

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

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. *ICLR*, abs/1312.6114.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver, August.

Anna Kozlova, Mariya Shmatova, and Anton Frolov. 2016. Ysda participation in the wmt'16 quality estimation shared task. In *Proceedings of the First Conference on Machine Translation*, pages 793–799, Berlin, Germany, August. Association for Computational Linguistics.

Ben Krause, Liang Lu, Iain Murray, and Steve Renals. 2017. Multiplicative lstm for sequence modelling. https://arxiv.org/abs/1609.07959.

Julia Kreutzer, Shigehiko Schamoni, and Stefan Riezler. 2015. Quality estimation from scratch (quetch): Deep learning for word-level translation quality estimation.

André Martins, Marcin Junczys-Dowmunt, Fabio Kepler, Ramón Astudillo, Chris Hokamp, and Roman Grundkiewicz. 2017. Pushing the limits of translation quality estimation. *Transactions of the Association for Computational Linguistics*, 5:205–218.

Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 785–794, New York, NY, USA. ACM.

Charles A. Micchelli, Yuesheng Xu, and Haizhang Zhang. 2006. Universal kernels. *Journal of Machine Learning Research*, 6:2651–2667.

Radford M. Neal. 1996. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Matteo Negri, Marco Turchi, José G. C. de Souza, and Daniele Falavigna. 2014. Quality estimation for automatic speech recognition. In *COLING*, pages 1813–1823. ACL.

Gustavo Paetzold and Lucia Specia. 2016. Simplenets: Quality estimation with resource-light neural networks. In *WMT*, pages 812–818. The Association for Computer Linguistics.

Gustavo Paetzold and Lucia Specia. 2017. Feature-enriched character-level convolutions for text regression. In *WMT*, pages 575–581. Association for Computational Linguistics.

Alec Radford, Rafal Józefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *CoRR*, abs/1704.01444.

Rajesh Ranganath, Sean Gerrish, and David M. Blei. 2014. Black box variational inference. *AISTAT*, abs/1401.0118.

Carl Edward Rasmussen and Christopher K. I. Williams. 2005. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1278–1286. JMLR.org.

Miguel Ríos and Serge Sharoff. 2016. Language adaptation for extending post-editing estimates for closely related languages.

Hugh Salimbeni and Marc Deisenroth. 2017. Doubly stochastic variational inference for deep gaussian processes. https://arxiv.org/abs/1705.08933.

Kashif Shah and Lucia Specia. 2016. Large-scale Multitask Learning for Machine Translation Quality Estimation. In *NAACL HLT 2016 : The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies : Proceedings of the Conference, June 12-17, 2016 San Diego, California, USA*, pages 558–567, Stroudsburg, PA, Jun. 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, Calif. (USA), 12 Jun 2016 - 17 Jun 2016, Association for Computational Linguistics (ACL). Zweitveröffentlicht auf dem Publikationsserver der RWTH Aachen University.

Kashif Shah, Trevor Cohn, and Lucia Specia. 2013. An investigation on the effectiveness of features for translation quality estimation.

Kashif Shah, Trevor Cohn, and Lucia Specia. 2015a. A bayesian non-linear method for feature selection in machine translation quality estimation. *Machine Translation*, 29(2):101–125, June.

Kashif Shah, Varvara Logacheva, Gustavo Paetzold, Frédéric Blain, Daniel Beck, Fethi Bougares, and Lucia Specia. 2015b. Shef-nn: Translation quality estimation with neural networks. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 342–347, Lisbon, Portugal, September. Association for Computational Linguistics.

Jiaxin Shi, Jianfei Chen, Jun Zhu, Shengyang Sun, Yucen Luo, Yihong Gu, and Yuhao Zhou. 2017. Zhusuan: A library for bayesian deep learning. cite arxiv:1709.05870Comment: The GitHub page is at https://github.com/thu-ml/zhusuan.

Edward Snelson and Zoubin Ghahramani. 2006. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pages 1257–1264. MIT Press.

Matthew Snover, Bonnie Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *AMTA*.

Radu Soricut, Nguyen Bach, and Ziyuan Wang. 2012. The sdl language weaver systems in the wmt12 quality estimation shared task. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, WMT '12, pages 145–151, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lucia Specia, Marco Turchi, Nicola Cancedda, Marc Dymetman, and Nello Cristianini. 2009. Estimating the Sentence-Level Quality of Machine Translation Systems. In *EAMT*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January.

Keyon Vafa. 2016. Training deep gaussian processes with sampling. *NIPS 2016 Workshop on Advances in Approximate Bayesian Inference*.

Andrew Wilson and Ryan Adams. 2013. Gaussian process kernels for pattern discovery and extrapolation. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1067–1075, Atlanta, Georgia, USA, 17–19 Jun. PMLR.

Andrew Gordon Wilson and Hannes Nickisch. 2015. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1775–1784. JMLR.org.

Andrew Gordon Wilson, Christoph Dann, and Hannes Nickisch. 2015. Thoughts on massively scalable gaussian processes. *CoRR*, abs/1511.01870.

Andrew G Wilson, Zhiting Hu, Ruslan R Salakhutdinov, and Eric P Xing. 2016a. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems 29*, pages 2586–2594.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. 2016b. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51, pages 370–378, Cadiz, Spain, 09–11 May. PMLR.

Guillaume Wisniewski, Anil Kumar Singh, and François Yvon. 2013. Quality estimation for machine translation: Some lessons learned. *Machine Translation*, 27(3-4):213–238, December.

User-Level Race and Ethnicity Predictors from Twitter Text

Daniel Preoţiuc-Pietro
Computer and Information Science
University of Pennsylvania
danielpr@sas.upenn.edu

Lyle Ungar
Computer and Information Science
University of Pennsylvania
ungar@cis.upenn.edu

Abstract

User demographic inference from social media text has the potential to improve a range of downstream applications, including real-time passive polling or quantifying demographic bias. This study focuses on developing models for user-level race and ethnicity prediction. We introduce a data set of users who self-report their race/ethnicity through a survey, in contrast to previous approaches that use distantly supervised data or perceived labels. We develop predictive models from text which accurately predict the membership of a user to the four largest racial and ethnic groups with up to .884 AUC and make these available to the research community.

1 Introduction

The popularity and ubiquity of social media allows access to a wide variety of spontaneous language enabling researchers to study language variation across space and time at large scale (Eisenstein et al., 2010; Eisenstein, 2016). Through language analysis, social media data showed the potential to compliment or in part replace traditional polling (O'Connor et al., 2010), with the caveat that its demographics is not a representative sample of the real population (Culotta, 2014). One of the most important demographic differences – especially across the US – is that of race and ethnicity. For example, in Twitter-based political polling applications it is important to adjust for the fact that ethnic minorities are overall less likely to support either party, but prefer the Democratic Party over the Republican Party (Hajnal and Lee, 2011).

In this paper, we present the first extensive study on identifying user-level race and ethnicity. So far, linguistic differences have mostly been studied in the context of dialects, usually African-American Vernacular English – AAVE (Jørgensen et al., 2015), using message-level data which offered insight into syntactic (Stewart, 2014) or lexical markers (Blodgett et al., 2016). However, not all users from a racial or ethnic group use these markers or, more generally, an associated dialect and usage is different across socio-demographic traits – use of the AAVE is correlated with lower income and education (Rickford, 1999). In addition, there are differences in language use across racial/ethnic groups not caused by dialects e.g., in the US, African Americans prefer basketball, Whites ice-hockey and Hispanic/Latinos football (OpenDorse, 2013) – and are consequently more likely to post about these sports.

In addition, previous studies used either perceived race labels (Mohammady and Culotta, 2015; Volkova and Bachrach, 2016; Culotta et al., 2016) which are subject to human stereotypes (Flekova et al., 2016a) or mapped geo-located tweets to census statistics (Mohammady and Culotta, 2014; Blodgett et al., 2016) which lead to other biases (Jørgensen et al., 2015) because: 1) census statistics are outdated; 2) the Twitter population is not a representative sample of the general population (Eisenstein et al., 2011) with African Americans over-represented on Twitter (Duggan, 2015); 3) users who geo-locate tweets are not a representative sample of the Twitter population (Eisenstein, 2016); 4) geo-located tweets might be posted from a different location than the user's home.

In this study, we introduce a new data set where user-level race and ethnicity labels are collected through an online survey of Twitter users. We build models for accurately classifying Twitter users into four of the largest racial and ethnic groups in the U.S. as defined by the Census Bureau: Non-Hispanic Whites,

Proceedings of the 27th International Conference on Computational Linguistics, pages 1534–1545
Santa Fe, New Mexico, USA, August 20-26, 2018.

Hispanic/Latinos, African-Americans and Asians.[1] We also measure the predictive power of various types of linguistic features and quantify usage of race/ethnicity revealing language across demographic groups. Finally, we perform a linguistic feature analysis to reveal the most important linguistic markers of race/ethnicity.

2 Data Set

We build a data set of Twitter users from participants in larger surveys taken through Qualtrics,[2] the largest platform for online experiments in social science, for which each user was compensated with 3$ per study.

As a first step, all participants were asked to take a standard demographic questionnaire which included selecting their race/ethnicity with the following options consistent with the US Census: African-American (AA, 374 users), Hispanic/Latino (Latino, 241 users), Asian (140 users), Non-Hispanic White (White, 3,182 users), Multiracial (153 users) or Other (free-text field prompt; 40 users, most frequently mentioned to be 'Native American'). In addition, we collected gender ('Male', 'Female' or open-ended field), age (13–90 interval), education (6 ordinal values ranging from 'No high-school' to 'Advanced Degree') and income level (8 ordinal values representing annual income from '<20k$/y' to '>200k$/y'). We restricted participation in our surveys to users based in the U.S. to limit the impact of potential cultural factors by using Qualtrics' filtering mechanisms.

The participants were asked to provide their public Twitter handle from which they had posted more than 100 tweets. We performed several checks on this Twitter handle to ensure that it was the user's own: *after* compensation we asked the users if they were truthful in reporting their handle and if not, we removed their data from analysis (22 users). We manually checked all handles which were verified by Twitter or had over 5000 followers and eliminated them if they were celebrities, as these were unlikely the users who participated in the survey (10 users in total). For each user, we downloaded their most recent 3,200 tweets, leading to a total data set size of 5,415,985 tweets.

We asked users to disclose their information, as this is the most efficient way to collect multiple demographic traits of users, despite the small possibility that some users are deceitful in reporting these traits. For ethical reasons, we only recruited participants who were willing to share their Twitter handles. This means our sample may not be fully representative of the general population, although we will account for any possible demographic skew in all our experiments.

After reporting demographics (including race/ethnicity) and their Twitter handles, users were directed to one of four collections of psychological questionnaires. The list of questions for each of these collections is very long (>30 questions) and bears no impact on the initial demographic information provided or to our experiments, hence we omit them from this paper. We have more 'White' users than the general U.S. population because participants were required to be part of this group in one of the sub-studies.

For the rest of the paper, we remove from our analysis the 'Multiracial' (153 users) and 'Other' (40 users) groups as these are small in sample size, heterogeneous in make-up and the sample may have been skewed towards a certain race/ethnicity mixture, but we could not know which.

3 About Race/Ethnicity

In our study, we used the definition of race/ethnicity as presented in the U.S. Census.[3,4]

We have chosen the U.S. Census definition of race/ethnicity for the following reasons:

- it is arguably most familiar to the participants, as users are all from the U.S., thus most have participated in the Census previously;
- multiple previous studies and applications such as polls rely on this classification to study the relationship between race/ethnicity and other variables;
- previous work on race-specific language used the same racial categorization (Blodgett et al., 2016);
- U.S. Census race/ethnicity surname statistics are available to benchmark our methods.

[1] Please refer to the 'About Race/Ethnicity' for a discussion about the definition of race/ethnicity used in this study.

[2] https://www.qualtrics.com/

[3] https://www.census.gov/topics/population/race/about.html

[4] https://www.census.gov/topics/population/hispanic-origin/about.html

We acknowledge that other racial/ethnic classifications exist or some groups are heterogeneous (e.g., Asian contains Chinese, Indian and Middle Eastern ethnicities) and that some argue the validity of the race construct entirely (Mukhopadhyay et al., 2013; Sen and Wasow, 2016).

4 Ethical Considerations

This experiment has received ethics approval from the Institutional Review Board (IRB) of our institution.

For reproducibility purposes, the data used in this paper is released in an anonymized and aggregated format.[5]. Any academic who wishes to gain access to the original user id's and labels will need to register with the authors, as per our IRB terms. Access will not be granted for any commercial applications and purposes.

Any individual-level predictions such as gender, age, religion, race or politics should be used with caution as they represent sensitive information. Race is, however, perhaps even more fraught with potential misunderstandings and abuses. We expect models for race prediction to be useful at an aggregate level, as the performance of the models do not warrant being be used at an individual level. For example, one could study how, in aggregate, tweets from youth of different races tweet about drug and alcohol use, where age and race make-up is estimated using predictive models.

We also highlight that Twitter's Terms of Service disallow targeting of individual users based on sensitive traits, including race: 'our Twitter Ads Policy also prohibits advertisers from targeting ads based on categories we consider sensitive, such as race, religion, politics, sex life, or health'.[6,7]

5 Features

In our analysis and predictive experiments, we use a broad range of linguistic features which are briefly described below.

Unigrams We use the bag-of-words representation to reduce each user's posting history to a normalised frequency distribution over the vocabulary consisting of all words used by at least 1% of the users (32,142 words).

LIWC Traditional psychological studies use a dictionary-based approach to representing text. The most popular method is based on Linguistic Inquiry and Word Count (LIWC) (Pennebaker et al., 2001) consisting of 73 manually constructed lists of words (Pennebaker et al., 2015) including some specific parts-of-speech, topical or stylistic categories. Each user is thereby represented as a frequency distribution over these categories.

Word2Vec Topics An alternative to LIWC is to use automatically generated word clusters. These clusters of words can be thought of as *topics*, i.e., groups of words that are semantically and/or syntactically similar. The clusters help reduce the feature space and provide good interpretability. To create these groups of words, we use an automatic method that leverages word co-occurrence patterns in large corpora by making use of the distributional hypothesis: similar words tend to co-occur in similar contexts (Harris, 1954).

We use the method from (Preoţiuc-Pietro et al., 2015a) to compute topics using word2vec similarity (Mikolov et al., 2013) and spectral clustering (Shi and Malik, 2000; von Luxburg, 2007) of different sizes (from 30 to 2000). We have tried other alternatives to building clusters: using other word similarities to generate clusters – such as NPMI (Lampos et al., 2014) or GloVe (Pennington et al., 2014) as proposed in (Preoţiuc-Pietro et al., 2015a) – or using standard topic modelling approached to create soft clusters of words e.g., Latent Dirichlet Allocation (Blei et al., 2003). For brevity and clarity, we present experiments with the best performing feature set containing 1000 Word2Vec topics after experimenting with other numbers of topics (from 30 through 2000). We aggregated all the words posted in a users tweets and represent each user as a distribution of the fraction of words belonging to each cluster.

Sentiment & Emotions We also investigate the extent to which racial groups differ in the type of emotions they express through their posts. The most popular model of discrete emotions is the Ekman model (Ekman, 1992; Strapparava and Mihalcea, 2008; Strapparava et al., 2004) which posits the existence of six basic

[5] http://www.preotiuc.ro

[6] https://twitter.com/privacy/

[7] https://support.twitter.com/articles/20170368\#/

emotions: anger, disgust, fear, joy, sadness and surprise. We automatically quantify these emotions from our Twitter data set using a publicly available model, which in addition to emotions also includes positive and negative sentiment (Volkova and Bachrach, 2016). We also used the word lexicon derived using crowd-sourcing from (Mohammad and Turney, 2010; Mohammad and Turney, 2013) and found it had slightly worse predictive results. Using these models, we assign sentiment and emotion probabilities to each message and then average across all users' posts to obtain user level emotion expression scores.

Part-of-Speech Tags We analyze part-of-speech tag usage across groups by POS tagging all tweets using the Twitter model of the Stanford Tagger (Derczynski et al., 2013), which showed best tagging results for AAVE (Jørgensen et al., 2015) and also uses the finer grained Penn Treebank tagset. Each user is thus represented as a distribution over POS tags.

Linear Ensemble Finally, we build a logistic regression model having as features the real-valued predictions of the models trained on all previous feature sets.

6 Baselines

We introduce the following competitive baselines:

Demographics User demographics collected from the users using our survey (age, gender, education and income level) are used as features. A classifier using only these demographics as features will establish the demographic tendencies of race/ethnicity in the data set.

Blodgett – User Model Blodgett et al. (2016) release a set of tweets with race/ethnicity probabilities obtained through distant supervision. In order to create user-level models for comparison, we take two separate approaches. From that data set, we identify the users that have more than 50 messages in the data set and label them with one of the four groups (AA, Latino, Other – mostly Asian – and White) if more than half of their messages are assigned a probability higher than 0.8 for that group. We build four one-vs-all unigram user-level classifiers and use these four models to predict the race/ethnicity of the users in our data set. These models were internally validated on the data set from (Blodgett et al., 2016) obtaining an average AUC $= .970$ prediction accuracy on 10-fold cross-validation within that data set. We also note that this method is trained on data from 26,009 users, an order of magnitude larger than our data set.

Blodgett – Message Model The second approach based on the (Blodgett et al., 2016) data set is to first train message-level logistic regression models for predicting how likely each message belongs to one of the four race/ethnicity groups. On internal validation, the average Pearson correlation between these models' prediction and the message-level scores released is R=.534 as measured using 10 fold cross-validation. We use these models to assign four prediction scores to all messages in our data set and compute an user-level average across these four predictions. We note that in this method we use 50 million tweets for training the message-level race/ethnicity classifiers, again an order of magnitude larger than our data set.

Perceived Race Labels We experiment with two data sets consisting of users for which race was determined by external annotators who analyzed the Twitter profile and tweets of users. The first data set was introduced in (Mohammady and Culotta, 2014) and consists of 362 users (all remaining public accounts from the original 770 users) labelled with three race categories (AA – 70, Latino – 104, White – 188). The second data set was introduced in (Volkova and Bachrach, 2016) and consists of 3380 users (all remaining public accounts from the original 5000 users) labelled with four race categories (AA – 1216, Latino – 150, Asian – 316, White – 1698) . All models trained on this data use bag-of-word unigram features and have been pre-processed in the same way as all other data sets.

Census Surnames Previous research used aggregate statistics about name distributions to estimate the demographic traits of Twitter users based on their name field, especially focusing on gender (Knowles et al., 2016). The U.S. 2010 Census aggregates statistics on race distribution for all surnames common to at least 100 US citizens, covering 95.9% of the population.[8] In total, 65.9% (2,592) of the users in our data set have a valid surname in their Twitter name field and could be matched to the Census statistics. For these, we selected the race with the highest frequency in the Census data as our prediction and for the rest, we use the most frequent race, namely White, as the prediction (**Census Surname – Most Likely**). We

[8] http://www.census.gov/topics/population/genealogy/data/2010_surnames.html

	AA	Latino	Asian	White
Baseline	.500	.500	.500	.500
Demographics	.630	.638	.659	.618
Models using Surnames				
Census Surname – Most Likely	.536	.610	.630	.571
Census Surname – Distribution	.634	.710	.736	.678
NamePrism - Most Likely	.525	.648	.657	.572
NamePrism - Distribution	.681	*.740*	.765	.719
Models based on (Blodgett et al., 2016)				
Blodgett – User Model	.729	.579	.560	.645
Blodgett – Message Model	.802	.700	.614	.718
Models Trained on Perceived Race Labels				
(Mohammady and Culotta, 2014)	.798	.603	–	.700
(Volkova and Bachrach, 2016)	.859	.693	.736	.765
User-level models proposed in this paper				
Emotions	.621	.617	.578	.609
LIWC	.775	.651	.656	.689
POS	.721	.666	.619	.686
Topics	.840	.670	.731	.758
Unigrams	.866	.708	.768	.795
Linear Ensemble	*.870*	*.710*	*.781*	*.797*
Linear Ensemble & NamePrism	**.884**	**.781**	**.832**	**.825**

Table 1: User-level race one-vs-all classification results measured in ROC AUC.

	Original		Balanced	
	Acc	F1	Acc	F1
Random Guess	.808	.223	.250	.100
Demographics	.809	.283	.391	.384
Census Surname – Most Likely	.804	.372	.358	.303
Census Surname – Distribution	.806	.403	.411	.411
NamePrism - Most Likely	.798	.386	.388	.342
NamePrism - Distribution	.801	.418	.480	.463
Trained on (Mohammady and Culotta, 2014)	.780	.338	.367	.287
Trained on (Volkova and Bachrach, 2016)	.820	.401	.383	.318
Trained on this data set	*.837*	*.485*	*.555*	*.557*
Trained on this data set & NamePrism – Distribution	**.842**	**.537**	**.617**	**.614**

Table 2: User-level four way race classification results on both the original data set and on balanced classes obtained by over-sampling the less frequent classes. Bag-of-words unigram features are used in all text-based methods.

also use the four Census race frequencies for the user's matched surname as features (**Census Surname – Distribution**). For users that are not matched, we use the average distribution. This has the effect to adjust for, at least partially, the race differences between the general U.S. population and the Twitter users in our data set.

NamePrism In addition to only looking at surnames as extracted from the Census, we also use a publicly available API [9] that enables race and nationality prediction from word formation patterns in surnames. This calls models that have obtained state-of-the-art accuracy for this task (Ye et al., 2017; Ambekar et al., 2009). Similarly to the Census name, we use both the actual prediction (**NamePrism – Most Likely**) and a model trained on the race probabilities predicted by the API **(NamePrism – Distribution**).

7 Predictive Experiments

In our predictive experiments we use logistic regression classification with Elastic Net regularization in a 10-fold cross-validation setup, where 8 folds are used for training, 1 for tuning the regularization parameters using grid search and 1 for testing. We have experimented with other methods for non-linear classification (e.g. SVMs), but results did not improve significantly.

7.1 One vs. All Classification

Results of one vs. all classifications are presented in Table 1. These are evaluated using ROC AUC (area under the receiver operating characteristic curve) because: a) the class distribution is imbalanced (up to

[9] http://www.name-prism.com/

1:19 for Asian vs. rest); b) AUC does not favor methods that use in training a similar class distribution as in the test set. Results using F1 score show the same patterns of improvement.

Overall performance of our best models are $>.7$ AUC (compared to AUC $= .5$ if random), with performance on the 'African American' group being highest (AUC $= .884$). Unigrams consistently obtain the best predictive results out of all feature sets, as they can capture idiosyncratic words and spellings which are very specific to certain racial/ethnic groups, while topic choice can only capture more generic topical relationships. Textual features – with a few exceptions for the 'Asian' group – make better predictions when compared to demographics, showing that language use carries information beyond purely demographics. Combining the predictions of all feature sets using a linear ensemble obtains the best results.

The two methods based on the data from (Blodgett et al., 2016) obtain overall good prediction results, with the message-level model surpassing the user-level model in all four cases. However, the performance is consistently lower than the best classifier trained on our data set, despite this being an order of magnitude smaller. Similar classifiers trained on perceived age labels result in models that are slightly, but consistently worse (.022 on average when comparing the 'Unigrams' only models) than when training on our data set. The results indicate that using perceived traits results in less accurate models when predicting real traits (Flekova et al., 2016a), as data set size is similar when using the data set from (Volkova and Bachrach, 2016). In addition, we have tried standard domain adaptation methods for combining both data sets (Daumé III, 2007) but found no additional gains.

The methods based on surnames obtain performance above chance, but below our best predictive models, except for the 'Latino' group where it exceeds our best model's performance due to the peculiarities posed by Hispanic/Latino surnames. Combining surname-based methods with the best text-based model results leads to significantly better results than using text alone, showing that these contain complementary information. However, further adding the demographics (age, gender, income, education) as features does not add to the predictive performance, showing that the results are not impacted by any imbalance in the demographic makeup of our data set.

7.2 Four-way Classification

Next, we experiment with four-way race/ethnicity classifiers using only unigram features for text-based methods, as these performed best in the previous experiment. In addition to experiments on the original data set, we also performed experiments with a balanced data set obtained by oversampling the smaller classes. In the 'Trained on this data set' setup, we oversampled in each fold and data split, such that no users from training are present in testing.

The results are presented in Table 2 with results showing both accuracy and F1 score (macro averaged). We notice a higher margin of improvement when using the data from our data set when compared perceived race labelled users. Surname distributions perform second best and are the only features that bring additional performance on top of the best performing method.

7.3 In-Sample Perceived Race Label Prediction

Finally, we compute the performance of models trained in-sample on perceived labels from previous work (Mohammady and Culotta, 2014; Volkova and Bachrach, 2016). All models are logistic regression classifiers with Elastic Net regularization. If these models achieve significantly higher accuracies compared to when tested on our data set, we can conclude that the race/ethnicity prediction task was over-simplified by the method of selecting users in the data set.

Results on in-sample 10-fold cross-validation on the two data sets with perceived race labels are presented for one-vs-all classification in Table 3. We notice that the results are very high compared to when applying the models on data with real race labels, with models suffering on average $>.10$ drop in AUC on data from (Volkova and Bachrach, 2016) and up to a .30 drop when trained on data from (Mohammady and Culotta, 2014).

Similarly, Table 4 shows 4-way classification results for in-sample perceived race prediction. We again note the larger margins of improvement over the baseline for both classifiers when compared to the same classifiers applied on real race labelled data in our data set in Table 2.

	AA	Latino	Asian	White
Baseline	.500	.500	.500	.500
Tested using Survey Labels				
(Mohammady and Culotta, 2014)	.798	.603	–	.700
(Volkova and Bachrach, 2016)	.859	.693	.736	.765
Tested using Perceived Labels				
(Mohammady and Culotta, 2014)	.954	.912	–	.928
(Volkova and Bachrach, 2016)	.946	.780	.767	.892

Table 3: Predictive results of unigram models using perceived race labels. Results are in ROC AUC.

	Original		Balanced	
	Acc	F1	Acc	F1
Random Guess	.519	.227	.333	.250
Trained on (Mohammady and Culotta, 2014)	.771	.770	.777	.768
Random Guess	.502	.167	.250	.100
Trained on (Volkova and Bachrach, 2016)	.764	.557	.639	.637

Table 4: User-level four way race unigram-based classification results on predicting perceived race labels. Results are on both the original data set split and on balanced classes obtained by over-sampling the less frequent classes.

8 Demographic Covariates

Next, we explore the hypothesis that the accuracy of race/ethnicity classification varies with the demographic traits of the users. For example, previous hypotheses state that AAVE usage is more prevalent in males, people of lower income and lower education levels (Rickford, 1999), thus making race prediction for this groups more accurate.

We focus our analysis on the African American user group. We build data splits that are matched in size, one demographic trait at a time (gender, age, education, income), by sub-sampling the larger demographic group when necessary. The, we sub-sample users such that the label proportions (AA vs. other) are the same in each task. For example, in studying the impact of gender in race classification we built a AA classifier for male users (122 AA vs. 122 Non-AA) and another one for female users (122 AA vs. 122 Non-AA). The split points between the two compared groups for age, education level and income level are 26 years old, completed High School degree and 40,000$/y respectively. All classifiers were trained using logistic regression with unigram features.

This experiment allows to uncover for which demographics automated methods are able to better classify users in belonging to the African American group, while keeping the training data size and label distribution constant. Classification results are shown in Table 5.

Classifiers reach different accuracies for age, gender and income, with younger, female and lower income users all being significantly easier to predict if they belong to the African American group. Further, in a separate experiment, the final classifier is more accurate for the same user categories (e.g., AUC $= .894$ for females compared to AUC $= .810$ for males). These results have implications in biases and classifier fairness.

Gender	Female: 0.858	Male: 0.772
Age	$<$26: 0.878	$>$26: 0.759
Education	$\leq$High School: 0.819	$>$High School: 0.802
Income	$<$40,000$/y: 0.853	$>$40,000$/y: 0.770

Table 5: Classification performance in ROC AUC when holding data set size constant across different demographic traits.

9 Feature Analysis

Finally, we perform a linguistic feature analysis with the goal of identifying the linguistic markers specific of each race/ethnicity group on Twitter.

First, Table 6 shows the features with the highest weights for each race/ethnicity category. The features represent the weights learnt a regularized bag-of-words unigram logistic regression models trained for predicting if a user is part of a race/ethnicity group, as described in Section 7.1. The results are intuitive, highlighting known dialectal variations (e.g., 'bout', 'naw') and references to in-group figures ('trayvon',

Latino	Asian
latinas, barely, maze, latina, remorse, dd, absorbed, ay	asian, neighbourhood, pho, ud, jg, consciousness
White	**African-American**
of, seriously, unhappy, sure, pumped, great, someday	trayvon, meek, bout, yaaasss, beyonce, smh, naw

Table 6: Unigrams most predictive of each race/ethnicity group when examining the feature weights learned by the classifiers.

r	Cluster	Words	r	Cluster	Words
Latino			**Asian**		
.144 → .063		22 Significant Clusters with mostly Spanish words	.145→ .060		13 Significant Clusters with words in Asian languages
.148	Spanish City names	los, san, diego, angeles, francisco, antonio, jose, fran	.107	Asian Place-names	china, japan, korea, tokyo, asia, philippines, seoul, hk
.125	S.American Places	mexico, venezuela, brazil, puerto, rico, buenos, costa, argentina	.105	Elongated 'yes'	yah*, yaa*, yee*, wahh, hee
.088	Politeness	no, reason, matter, problem, sense, offense, worries, excuses	.098	Phonetic Spelling	ikr, rly, h8, bcuz, rlly, coz, pple, realy, ure, completely
.078	Frustration	dayum, man*, damn*, shi*t, ma*n	.087	Intejections	eh, ye, heh, tis, pun, poke, wink, teh, mam
.070	Love	333*, love*, lo*ve, ilove	.084	Cricket	england, india, kenya, #worldcup, cricket, batting, aus, pakistan
White			**African-American**		
.121	Time References	years, hours, minutes,ago, weeks, months, minute, seconds, min	.281	Slang	bout, wit, kno, sayin, talkin, doe, lowkey, wut, naw, thinkin
.110	Superlatives	ever, worst, biggest, cutest, funniest, coolest, longest, sweetest	.252	Group References	bitches, hoes, cops, dudes, fools, fucks, females, chicks, shits
.101	Adverbs of degree	absolutely, quite, extremely, perfectly, incredibly, certainly	.241	Person References	lil, boo, buddy, mama, sis, bby, tina, dee, missy, mister, sissy
.100	Rest	bed, couch, laying, cuddle, blanket, comfy, cuddling, blankets	.224	Phonetic Spelling	somethin, urself, some1, every1, any1, wha, sum1, no1
.091	Modals	say, could, wish, i'd, wouldn't, couldn't, meant, you'd, couldnt	.171	G-Dropping	tl, freaky, callin, actin, seein, tweetin, followin, ppls, sendin

Table 7: Word2Vec clusters most correlated with each race/ethnicity group (maximum 5 clusters per group, excluding topics made up of foreign words). The cluster name is manually assigned. All correlations are significant at $p < .05$, two-tailed t-test, **Simes corrected** for multiple comparisons and are controlled for gender, age, education and income. Words in a category are sorted by frequency in our data set. * highlights word variants where a character is repeated.

'beyonce') for AA users, names specific to a racial or ethnic group (e.g. 'latinas', 'latina', 'asian'), frequent foreign words (e.g. 'ay', 'ud') or more salient linguistic cues ('seriously', 'of', 'great', 'barely').

However, examining only the most predictive features assigned by a model with both L1 and L2 regularization is likely to overlook features are co-linear. For this reason, we perform analysis of Word2Vec clusters, POS Tags and Emotions using univariate correlation as introduced in (Schwartz et al., 2013). We compute univariate Pearson correlations independently for each feature between its distribution across users (features are first normalized to sum up to unit for each user) and the user-level outcome of interest (e.g., whether it belongs to a race/ethnicity group or not). In Section 8, we showed that user demographics (age, gender, education and income levels) impact the classification accuracies. In order to mitigate this effect, we introduce age, gender, education and income levels as controls and compute partial linear correlation for each feature. Since a large number of features are explored simultaneously, we consider coefficients significant if they meet a Simes-corrected two-tailed p-value of less than 0.05.

Table 7 shows the Word2Vec clusters with the highest correlations with users belonging to a group. For the 'Latino' and 'Asian' groups, we do not show clusters that contain more than 70% non-English words in their most frequent 20 words, as these only reflect the use of a particular language (e.g. Hindi or Tagalog for 'Asian' and Spanish for 'Latino') and are not interesting for analysis.

For Latinos we first note the use of Spanish-origin place names in the US and place names from Latin America. In addition, Latinos use words associated with polite constructs (e.g. 'offense' as in 'no offense', 'worries' as in 'no worries', 'excuses') and express frustration (variants of 'man', 'shit' and 'damn', as well as the Spanish version 'dayum') or love (heart shapes or variants of 'love'). The Asian group is, similarly to the 'Latino' group, first characterized by the use of words belonging to Asian languages and Asian name places. Asian users prefer a peculiar type of expressing 'yes' (e.g.,' 'yah', 'yaa', 'yee'), use a specific category of interjections (e.g., 'heh', 'pole') and prefer certain phonetic spellings (e.g., 'bcuz', 'h8') or contractions to typical English words (e.g. ikr – 'I know, right?', 'rly', 'rlly' – 'really'). Finally,

r	Tag	Words	r	Tag	Words
Latino			**Asian**		
.100	FW	Foreign Word	.079	UH	Interjection
.097	UH	Interjection	.057	NNP	Proper Noun, singular
.091	NNP	Proper Noun, singular			
White			**African-American**		
.101	IN	Subordinating conj. – of, on	.098	RP	Adverb, particle – about
.093	JJS	Adjective, superlative – best	.097	VB	Verb, base form – think
.087	DT	Determiner – the	.094	PRP	Personal Pronoun – I
.085	EX	Existential 'there'	.084	WP	Wh-pronoun – Who
.083	VBZ	Verb, 3rd pers. singular present – moves	.053	VBP	Verb, non-3rd pers. singular present – move
.082	TO	to	.051	JJR	Adjective, comparative – big
.069	MD	Modal – could			
.058	RB	Adverb – extremely			
.054	WDT	Wh-determiner – which			
.054	CC	Coordinating conj. – and			
.044	NNS	Noun, plural – years			

Table 8: Part-of-Speech tags most correlated with each race/ethnicity group. All correlations are significant at $p < .05$, two-tailed t-test, **Simes corrected** for multiple comparisons and are controlled for gender, age, education and income.

Emotion	Latino	Asian	White	AA
Positive	-.060	–	–	.036
Negative	-.049	–	–	–
Anger	–	–	-.067	.118
Disgust	–	–	–	.083
Fear	–	–	–	.050
Joy	-.044	–	–	–
Sadness	-.040	–	–	.048
Surprise	-.041	–	–	–

Table 9: Pearson correlations between emotions and race/ethnicity groups. All reported correlations are significant at $p < .05$, two-tailed t-test, **Simes corrected** for multiple comparisons and are controlled for gender, age, education and income.

the cluster containing words about cricket highlights a more topical difference specific of Asian users, likely driven by users with ethnicity in the Indian subcontinent where this sport is popular. White users are best characterized by the use of temporal references (e.g., 'years', 'ago', 'weeks'). Many clusters uncover syntactic preferences of the White group including the use of superlatives (e.g., 'cutest'), adverbs of degree (e.g., 'quite') and modal verbs (e.g. 'could'). Finally, a topical cluster around the theme of rest (e.g., 'laying', 'cuddle') is also specific of the White group. Finally, the African American groups has the topics with the highest correlation coefficients. Clusters capture specific terms that reference other persons (e.g., 'lil', 'boo') or groups (e.g., 'dudes') and other typical AAVE words (e.g., 'naw', 'bout'). Similarly to the Asian group, we also observe a cluster dominated by phonetic spelling variants of words (e.g., 'urself', 'some1'). Finally, we observe a topic dominated by verbs missing the final 'g' (e.g., 'callin', 'sendin'). This represents one of the most well-known distinctive features of AAVE (Rickford, 1999).

Table 8 shows Part-of-Speech tags correlated with each race/ethnicity group. The Latino and Asian groups are characterized by the use of foreign words, interjections and singular proper nouns. This matches the findings of the cluster analysis, leading to the conclusion that these two groups use references of proper names and foreign words both related to their ethnic origin and, increased use of interjections. The POS usage results for the White and African-American groups highlight intriguing contrasts. While the White users prefer superlative adjectives, African American users prefer comparative adjectives. White users are predisposed to use more adverbs, however African Americans use more adverb particles (e.g., 'take off', 'put away'). African-American prefer non 3-rd person singular verbs and use more personal pronouns, while White users prefer 3-rd person verbs. This shows the tendency of African-Americans to post more on Twitter about their activities and whereabouts rather than other events or impersonal reporting. White users also use more conjunctions (subordinating and coordinating), perhaps an indicator of more complex syntactic constructions.

Lastly, emotion analysis results are presented in Table 9. The results show that emotions are mainly correlated with the Latino and AA groups, with only a single other significant correlation for the Asian and White groups. African American users on Twitter express overall more emotions than other groups, including both positive sentiment as well as a range of negative emotions, especially anger. On the other hand, Latino users express overall less sentiment, both positive and negative, and fewer emotions.

10 Conclusion

We presented a detailed study of user-level race/ethnicity prediction along the lines of previous work on predicting user traits from text (Burger et al., 2011; Rao et al., 2010; Pennacchiotti and Popescu, 2011; Schwartz et al., 2013; Sap et al., 2014; Volkova et al., 2014; Preoţiuc-Pietro et al., 2015b; Flekova et al., 2016b; Preoţiuc-Pietro et al., 2016; Preoţiuc-Pietro et al., 2017). In contrast with previous research on race/ethnicity, we used labels obtained by directly surveying Twitter users, rather than distantly supervised geo-located data or perceived race labels, which lead to multiple biases and lower accuracies on real data.

We built models that obtain state-of-the-art out-of-sample accuracy on predicting the four prominent racial/ethnic groups in the US. We presented an extensive linguistic feature analysis for each group and brought new evidence towards linguistic hypotheses on dialect use in Twitter across demographic groups. We believe this paper offers a solid basis for the study of race prediction online. Our models are readily usable for large-scale passive polling scenarios, where automatically quantifying existing racial sample differences can lead to improved predictive performance (Culotta, 2014).

Acknowledgments

The authors acknowledge the support of the Templeton Religion Trust, grant TRT-0048.

References

Anurag Ambekar, Charles Ward, Jahangir Mohammed, Swapna Male, and Steven Skiena. 2009. Name-ethnicity Classification from Open Sources. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, KDD, pages 49–58.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Su Lin Blodgett, Lisa Green, and Brendan O'Connor. 2016. Demographic Dialectal Variation in Social Media: A Case Study of African-American English. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 1119–1130.

D. John Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Discriminating Gender on Twitter. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 1301–1309.

Aron Culotta, Nirmal Ravi Kumar, and Jennifer Cutler. 2016. Predicting Twitter User Demographics using Distant Supervision from Website Traffic Data. *Journal of Artificial Intelligence Research*, 55:389–408.

Aron Culotta. 2014. Reducing Sampling Bias in Social Media Data for County Health Inference. In *Joint Statistical Meetings Proceedings*, pages 1–12.

Hal Daumé III. 2007. Frustratingly Easy Domain Adaptation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, ACL, pages 256–263.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter Part-of-Speech Tagging for All: Overcoming Sparse and Noisy Data. In *Proceedings of Recent Advances in Natural Language Processing*, RANLP, pages 198–206.

Maeve Duggan. 2015. *Mobile Messaging and Social Media – 2015*. Pew Research Center.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2010. A Latent Variable Model for Geographic Lexical Variation. In *Proceedings of the 2010 Conference on Empirical Methods for Natural Language Processing*, EMNLP, pages 1277–1287.

Jacob Eisenstein, Noah A Smith, and Eric P Xing. 2011. Discovering Sociolinguistic Associations with Structured Sparsity. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, ACL, pages 1365–1374.

Jacob Eisenstein. 2016. Written Dialect Variation in Online Social Media. In Charles Boberg, John Nerbonne, and Dom Watt, editors, *Handbook of Dialectology*. Wiley.

Paul Ekman. 1992. An Argument for Basic Emotions. *Cognition & Emotion*, 6(3-4):169–200.

Lucie Flekova, Jordan Carpenter, Salvatore Giorgi, Lyle Ungar, and Daniel Preoţiuc-Pietro. 2016a. Analyzing Biases in Human Perception of User Age and Gender from Text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL, pages 843–854.

Lucie Flekova, Lyle Ungar, and Daniel Preoţiuc-Pietro. 2016b. Exploring Stylistic Variation with Age and Income on Twitter. In *Proceedings of the 54th annual meeting of the Association for Computational Linguistics*, ACL.

Zoltan L Hajnal and Taeku Lee. 2011. *Why Americans don't Join the Party: Race, Immigration, and the Failure (of Political Parties) to Engage the Electorate*. Princeton University Press.

Z. Harris. 1954. Distributional Structure. *Word*, 10(23):146 – 162.

Anna Katrine Jørgensen, Dirk Hovy, and Anders Søgaard. 2015. Challenges of Studying and Processing Dialects in Social Media. In *Proceedings of the First Workshop on Noisy User-Generated Text (W-NUT)*, ACL, pages 9–18.

Rebecca Knowles, Josh Carroll, and Mark Dredze. 2016. Demographer: Extremely Simple Name Demographics. In *Proceedings of the Workshop on NLP and Computational Social Science*, EMNLP, pages 108–113.

Vasileios Lampos, Nikolaos Aletras, Daniel Preoţiuc-Pietro, and Trevor Cohn. 2014. Predicting and Characterising User Impact on Twitter. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, EACL, pages 405–413.

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2010 annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL, pages 746–751.

Saif M. Mohammad and Peter D. Turney. 2010. Emotions Evoked by Common Words and Phrases: Using Mechanical Turk to Create an Emotion Lexicon. In *Proceedings of the Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, NAACL, pages 26–34.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a Word-Emotion Association Lexicon. *Computational Intelligence*, 29(3):436–465.

Ardehaly Ehsan Mohammady and Aron Culotta. 2014. Using County Demographics to Infer Attributes of Twitter Users. In *Proceedings of the Joint Workshop on Social Dynamics and Personal Attributes in Social Media*, ACL, pages 7–16.

Ardehaly Ehsan Mohammady and Aron Culotta. 2015. Inferring Latent Attributes of Twitter Users with Label Regularization. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL, pages 185–195.

Carol C Mukhopadhyay, Rosemary Henze, and Yolanda T Moses. 2013. *How Real is Race?: A Sourcebook on Race, Culture, and Biology*. Rowman & Littlefield.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, and Noah A Smith. 2010. From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, ICWSM, pages 122–129.

OpenDorse. 2013. 2013 Sports Fan Demographics. `http://opendorse.com/blog/2013-sports-fan-demographics/`. Accessed: 2017-02-05.

Marco Pennacchiotti and Ana-Maria Popescu. 2011. A Machine Learning Approach to Twitter User Classification. In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, ICWSM, pages 281–288.

James W. Pennebaker, Martha E. Francis, and Roger J. Booth. 2001. *Linguistic Inquiry and Word Count*. Mahway: Lawrence Erlbaum Associates.

James W. Pennebaker, Roger J. Booth, Ryan L. Boyd, and Martha E. Francis. 2015. *Linguistic Inquiry and Word Count: LIWC2015*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 1532–1543.

Daniel Preoţiuc-Pietro, Vasileios Lampos, and Nikolaos Aletras. 2015a. An Analysis of the User Occupational Class through Twitter Content. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL, pages 1754–1764.

Daniel Preoţiuc-Pietro, Svitlana Volkova, Vasileios Lampos, Yoram Bachrach, and Nikolaos Aletras. 2015b. Studying User Income through Language, Behaviour and Affect in Social Media. *PLoS ONE*, 10(9), 09.

Daniel Preoţiuc-Pietro, Jordan Carpenter, Salvatore Giorgi, and Lyle Ungar. 2016. Studying the Dark Triad of Personality using Twitter Behavior. In *Proceedings of the 25th ACM Conference on Information and Knowledge Management*, CIKM, pages 761–770.

Daniel Preoţiuc-Pietro, Ye Liu, Daniel J. Hopkins, and Lyle Ungar. 2017. Beyond Binary Labels: Political Ideology Prediction of Twitter Users. In *Proceedings of the 55th Conference of the Association for Computational Linguistics*, ACL.

Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. 2010. Classifying Latent User Attributes in Twitter. In *Proceedings of the 2nd International Workshop on Search and Mining User-generated Contents*, SMUC, pages 37–44.

John Russell Rickford. 1999. *African American Vernacular English: Features, Evolution, Educational Implications*. Wiley-Blackwell.

Maarten Sap, Gregory Park, Johannes Eichstaedt, Margaret Kern, Lyle Ungar, and H Andrew Schwartz. 2014. Developing Age and Gender Predictive Lexica over Social Media. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 1146–1151.

H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, and Martin EP Seligman. 2013. Personality, Gender, and Age in the Language of Social Media: The Open-vocabulary Approach. *PloS ONE*, 8(9).

Maya Sen and Omar Wasow. 2016. Race as a Bundle of Sticks: Designs that Estimate Effects of Seemingly Immutable Characteristics. *Annual Review of Political Science*, 19:499–522.

Jianbo Shi and Jitendra Malik. 2000. Normalized Cuts and Image Segmentation. *Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.

Ian Stewart. 2014. Now we Stronger than Ever: African-American Syntax in Twitter. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, EACL, pages 31–37.

Carlo Strapparava and Rada Mihalcea. 2008. Learning to Identify Emotions in Text. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, pages 1556–1560.

Carlo Strapparava, Alessandro Valitutti, et al. 2004. WordNet Affect: an Affective Extension of WordNet. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, volume 4 of *LREC*, pages 1083–1086.

Svitlana Volkova and Yoram Bachrach. 2016. Inferring Perceived Demographics from User Emotional Tone and User-Environment Emotional Contrast. In *Proceedings of the 54rd Annual Meeting of the Association for Computational Linguistics*, ACL, pages 1567–1578.

Svitlana Volkova, Glen Coppersmith, and Benjamin Van Durme. 2014. Inferring User Political Preferences from Streaming Communications. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, ACL, pages 186–196.

Ulrike von Luxburg. 2007. A Tutorial on Spectral Clustering. *Statistics and Computing*, 17(4):395–416.

Junting Ye, Shuchu Han, Yifan Hu, Baris Coskun, Meizhu Liu, Hong Qin, and Steven Skiena. 2017. Nationality classification using name embeddings. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM, pages 1897–1906.

Multi-Source Multi-Class Fake News Detection

Hamid Karimi, Proteek Chandan Roy, Sari Saba-Sadiya, and Jiliang Tang
Department of Computer Science and Engineering, Michigan State University
{karimiha,royprote,sadiyasa,tangjili}@msu.edu

Abstract

Fake news spreading through media outlets poses a real threat to the trustworthiness of information and detecting fake news has attracted increasing attention in recent years. Fake news is typically written intentionally to mislead readers, which determines that fake news detection merely based on news content is tremendously challenging. Meanwhile, fake news could contain true evidence to mock true news and presents different degrees of fakeness, which further exacerbates the detection difficulty. On the other hand, the spread of fake news produces various types of data from different perspectives. These multiple sources provide rich contextual information about fake news and offer unprecedented opportunities for advanced fake news detection. In this paper, we study fake news detection with different degrees of fakeness by integrating multiple sources. In particular, we introduce approaches to combine information from multiple sources and to discriminate between different degrees of fakeness, and propose a **M**ulti-source **M**ulti-class **F**ake news **D**etection framework MMFD, which combines automated feature extraction, multi-source fusion and automated degrees of fakeness detection into a coherent and interpretable model. Experimental results on the real-world data demonstrate the effectiveness of the proposed framework and extensive experiments are further conducted to understand the working of the proposed framework.

1 Introduction

Given its negative impacts, fake news has been identified as a global threat (Webb et al., 2016). Detecting fake news has become increasingly important and can benefit individuals and even our society in many aspects. First, people will be well-informed about events and news and their political and social activities will not be misguided. Second, despite a few recent initiatives by some social media providers like Facebook, there is no systematic fake news detection by social media platforms. Third, identifying fake news is one step toward targeting financial incentives encouraging the spreaders running their "business". For instance, Google attempts to stop providing its ad services to fake news websites (Wingfiled et al., 2016). Fourth, people would not lose their trust in web and social media.

However, fake news detection is naturally challenging especially in the era of social media. First, fake news is usually written intentionally to mislead its readers and the content of fake news is rather diverse in terms of length, topics, and styles (Shu et al., 2017a). For example, fake news in social media is short, informal and is often related to newly emerging and time-critical events. Thus, since we have not gained enough insights into the nature of fake news, hand-crafted features based on the news content are generally not sufficient (Ruchansky et al., 2017). Second, to mock true news, fake news could mix false statements with true ones. For example, fake news can cite true evidence to support a non-factual claim (Shu et al., 2017a). Hence, fake news has different degrees of *fakeness* such as half-true, false, etc. However, the majority of existing algorithms consider fake news detection as a binary classification in the form of the *true/false* dichotomy. Considering degrees of fakeness adds further difficulty on fake news detection.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1546–1557
Santa Fe, New Mexico, USA, August 20-26, 2018.

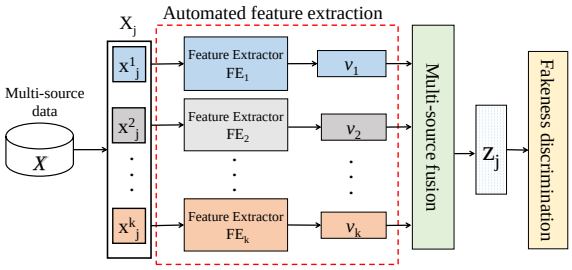


Figure 1: An overview of the proposed model for fake news detection (MMFD)

On the other hand, recent advances in mobile techniques and the popularity of emerging media (e.g., social media) enable the widespread of fake news and this process produces a large amount of data, which allows us to collect information about fake news from various perspectives such as the origin of fake news, the fake news writers and their historical data, etc. Such data provides rich contextual information beyond the news content. For example, a writer who created plenty of fake news is likely to create more fake news (Del Vicario et al., 2016); while news from authoritative organizations such as governments is less likely to be false. Thus, the availability of multiple sources related to fake news has the great potential to help fake news detection.

In this paper, we study the problem of multi-class fake news detection with multiple sources. In particular, we aim to answer two major research questions – (1) how to effectively combine information from multiple sources for fake news detection and (2) how to discriminate between degrees of fakeness mathematically. Our solutions to these two questions result in the proposed Multi-source Multi-class Fake news Detection (MMFD) framework. Our main contributions are summarized as a) we introduce an automated and interpretable way to integrate information from multiple sources; b) we provide a principled approach to discriminate between degrees of fakeness mathematically; c) the proposed framework MMFD coherently incorporates automated feature extraction, multi-source fusion and degrees of fakeness discrimination in an end-to-end way; and d) we conduct experiments on real-world data to demonstrate the effectiveness of the proposed framework.

The rest of the paper is organized in the following manner. In Section 2, we formally define the multi-source multi-class fake news detection problem. Section 3 describes the proposed approach for fake news detection. Section 4 evaluates the proposed method and presents the experimental results and discussions. In Section 5, we present the related work. Section 6 concludes the paper and sheds light onto the future directions.

2 Problem Definition

In this section, we introduce the mathematical notations and formally define the multi-source multi-class fake news detection problem. We follow the previous work (Allcott and Gentzkow, 2017; Shu et al., 2017b) and define fake news as follows.

Definition. A news item is called *fake* if its content is verified to be false and *true* otherwise.

Let $\mathcal{X} = \{X_1, X_2, \cdots, X_n\}$ denote a multi-source dataset containing n news items. Each new item $j \in [1, n]$ contains k sources of data and is denoted as $X_j = \{x_j^1, x_j^2, \cdots x_j^k\}$. Additionally, let $Y = \{y_1, y_2, \cdots, y_n\}$ denote a set of class labels associated with news items of dataset $\mathcal{X}$. Each class label $y_i \in Y$ takes a label from the label set $L = \{l_1, l_2, \cdots, l_m\}$ where m denotes the number of recognized degrees of fakeness in our framework and $l_j \in L$ is a particular degree of fakeness e.g., *half-true*. With the aforementioned notations and definitions, the problem of multi-source multi-class fake news detection is formally defined as follows:

Given the multi-source dataset $\mathcal{X}$ and its corresponding multi-class labels Y, we aim to learn the model $\mathcal{M}$ mapping $\mathcal{X}$ to Y, which can automatically predicts the degrees of fakeness for unlabeled news.

3 The Proposed Framework

Multi-source multi-class fake news detection faces three challenges. First, hand-crafted features based on news content are not very efficient for fake news, which calls for an automated feature extraction approach from multiple sources. Second, multiple sources of fake news data offer complementary information and combining the multiple sources while delivering an interpretable solution is another challenge. Third, degrees of fakeness offer a better understanding of fake news; however, fake news with different degrees may not be easily separable. In this work, we propose a framework which can tackle these challenges simultaneously. Figure 1 demonstrates an overview of the proposed **M**ulti-source **M**ulti-class **F**ake news **D**etection framework (MMFD). MMFD incorporates three coherent components into an end-to-end way – automated feature extraction (Section 3.1), interpretable multi-source fusion (Section 3.2), and fakeness discrimination (Section 3.3). In the following, we detail each component, followed by presenting the training procedure of the proposed model in Section 3.4.

3.1 Automated feature extraction

Since hand-crafted features (Gupta et al., 2014; Yang et al., 2012; Khurana and Intelligentie, 2017) are not very effective, automated feature extraction is more desired. Thus, we employ a deep neural network model to extract powerful features from each source inspired by the promising performance of deep learning in representation learning (Bengio et al., 2013). Different types of sources may need different deep network architectures. Since most of the sources in our dataset are textual sources (see Section 4), we propose a method to automatically extract features from a textual source.

The textual data can contain indicative information revealing the nature of fake news. Inspired by recent advancements in deep neural networks for modeling the text, we propose a deep model to extract features from textual sources based the CNN (convolutional Neural Network) (LeCun et al., 2015) and the LSTM (Long Short-Term Memory) network (Hochreiter and Schmidhuber, 1997). A CNN extracts local patters from a text similar to n-grams features (Kalchbrenner et al., 2014). Then, we apply an LSTM on top of the features extracted from the CNN aiming at capturing the temporal dependencies in the entire text.

Suppose a textual source contains x words. To apply the CNN model, we represent the text by an input matrix of word embeddings denoted as $W \in \mathbb{R}^{x \times e}$ where e is the dimension of the word embedding. More specifically, $w_j \in W$ is a e dimensional vector representing j-th word of the text and populates j-th row of matrix W. This matrix is produced from a word representation method such as word2vec (Mikolov et al., 2013). Then, convolution operations on W are performed M iterations (e.g., $\mathsf{M} = 2$ in Figure 2). At each iteration $m \in [1, \mathsf{M}]$, a filter (a weight matrix) $f^m \in \mathbb{R}^{l^m \times e}$ is convolved with W where l^m is the length of the filter. Convolution of f^m with the input matrix W produces feature maps $\mathbf{p}^m \in \mathbb{R}^{x-l^m+1}$. Each entry $p_j \in \mathbf{p}^m$ $(1 \leq j \leq x - l^m + 1)$ is generated as follows:

$$p_j = \mathcal{G}(r_j \odot f^m + b) \tag{1}$$

where $r_j = \{w_j, w_{j+1}, \cdots, w_{j+l^m-1}\}$ is a window of l^m consecutive words in W (a region of m words), $\mathcal{G}$ is a non-linear activation function such as sigmoid, $\odot$ denotes element-wise product operation, and $b \in \mathbb{R}$ is a bias term. Following the common way in CNNs, we repeat the process described above n_m times producing the sequence $U^m = [\mathbf{p}_1^m || \mathbf{p}_2^m || \cdots || \mathbf{p}_{n_m}^m]$ where $||$ denotes the vector concatenation operator (one can see U^1 and U^2 in Figure 2 shown as colorful column vectors). Rows of U^m represent the **local patterns** extracted from a text via the CNNs. In order to find the **global patterns** throughout the entire text, we feed rows of U^m to an LSTM network as depicted in Figure 2. The output of the LSTM network is considered as the hidden output of its last unit and is denoted as s^m having size q i.e., $|s^m| = q$. Then, the final feature vector of the proposed CNN-LSTM model is generated by passing the concatenation of all last hidden outputs through a Fully Connected Network (denoted as FCN in Figure 2) as follows:

$$v = \mathcal{G}([s^1 || s^2 || \cdots || s^{\mathsf{M}}]^T \times A + b) \tag{2}$$

where v denotes the extracted feature vector of a textual source, $A \in \mathbb{R}^{(\mathsf{M} \times q) \times d}$ is a weight matrix, and d is the desired final feature vector size. Note that $[s^1 || s^2 || \cdots || s^{\mathsf{M}}]$ is flattened into a vector of size $\mathsf{M} \times q$.

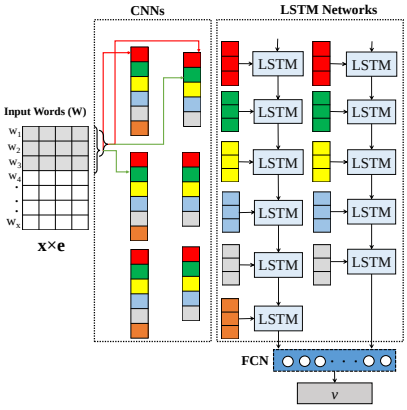


Figure 2: An illustration of the proposed deep model for fake news textual source feature extraction.

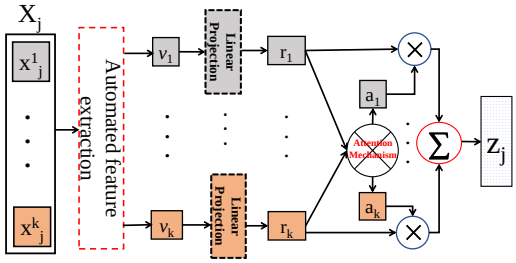


Figure 3: An illustration of the proposed interpretable multi-source fusion.

3.2 Interpretable multi-source fusion

The interpretable multi-source fusion component aims to combine features from different sources. A naive way of doing this is concatenating the feature vectors extracted by automated feature extraction component i.e., v_1 to v_k in Figure 1, into a single vector. This scheme considers all sources equally. However, different sources may have substantially different powers to detect fake news. Hence, when combining multiple sources, we propose to consider their contributions via an attention mechanism as shown in Figure 3. The attention mechanisms have achieved great success in many applications such as language translation (Bahdanau et al., 2014), image captioning (Xu et al., 2015), etc. Next, we present the details of the interpretable multi-source fusion component

The set of source-specific features of a news item is extracted by the automated feature extraction component as described in Section 3.1. Then suppose each source's extracted feature vector is denoted by $v_i \in \mathbb{R}^{d_i}$ $(1 \leq i \leq k)$ and has the dimension d_i. Now, to ensure feature vectors from different sources all have the same dimension h, we map each v_i to the vector r_i via a linear projection as follows:

$$r_i = v_i^T \times W_i + b_i, \quad \forall 1 \leq i \leq k \tag{3}$$

where $W_i \in \mathbb{R}^{d_i \times h}$ is a weight matrix, and $b_i \in \mathbb{R}^h$ is a bias vector. Then, the proposed attention method is carried out as follows:

$$z_j = \mathcal{G}(\sum_{i=1}^{k} a_i r_i), \quad \forall 1 \leq j \leq n \tag{4}$$

where z_j denotes the final feature vector of j-th news item and the scalar a_i is the attention score associated with i-th source i.e., the contribution of i-th source at z_j. Typically the attention scores are represented by a probability distribution. Therefore, each attention score a_i is normalized by the softmax function as follows.

$$a_i = \frac{e^{u_i}}{\sum_{l=1}^{k} e^{u_l}}, \quad \forall 1 \leq i \leq k \tag{5}$$

where u_i is a real value number denoting attention score of source i and is calculated according to Eq. 6:

$$u_i = w^T tanh(r_i), \quad \forall 1 \leq i \leq k \tag{6}$$

where $w \in \mathbb{R}^h$ is a weight matrix.

We should emphasize that attention scores a_i $(1 \leq i \leq k)$ are learned along with other parts of the model in an end-to-end manner where they are optimized to capture the informativeness of the different sources at the fake news detection task. As a result, capturing contributions of different sources reflected

at scalar values a_i ($1 \leq i \leq k$) makes the proposed framework somehow interpretable. This is due to the fact that now we can identify the roles of different sources in the determining the veracity of a news item as will be verified in Section 4.4.

3.3 Fakeness discrimination

Many of the existing fake news detection approaches look into the problem from a binary perspective. However, in practice, a piece of news might be a mixture of factual and false statements. Therefore, it is crucial to classify fake news into multiple classes reflecting degrees of fakeness. Nevertheless, multi-class fake news detection (i.e., fakeness discrimination) is challenging. The classifier needs to offer a better discriminative power because the boundary between two classes becomes more and more intertwined as the number of classes increases. To handle this challenge, we propose **M**ulti-class **D**iscriminative **F**unction (MDF) and is described in the following.

Classification accuracy on unknown test cases mostly depends on how we map our input data into a new feature space. In general, we want to group similar samples together and dissimilar ones far apart. To achieve this, we can learn the 'centers' of the classes in the feature space. If the centers are too close to one another, their samples might overlap resulting in less discriminative feature learning. Additionally, samples should be close to their class centers. This guarantees that samples belonging to the same class will be mapped into the same cluster (Wen et al., 2016). In general, the two aforementioned goals can be regarded as optimizing the precision and recall of the clusters in the learned feature space. Therefore, we include two terms in MDF: one which *pushes* the centers by a margin referred as *inter-class* term and one which *pulls* the samples toward their centers referred as *intra-class* term. The proposed inter-class term helps to learn discriminative features. Next, we introduce the MDF mathematically.

We present the MDF in a generalized non-convex form. The interpretable multi-source fusion component yields output feature vector of each news item (z_j in Eq. 4). For convenience, let $\mathcal{D}$ denote a dataset that contains features of all news items in $\mathcal{X}$ (i.e., $\mathcal{D} = \{z_1, z_2, \cdots, z_n\}$) and $\mathcal{D}_i$ denote all samples belonging to class i where $1 \leq i \leq m$. We calculate the centers of the different classes in the output feature space by the following equation.

$$c_i = \frac{1}{|\mathcal{D}_i|} \sum_{z_j \in \mathcal{D}_i} z_j, \forall i \in \mathcal{C} \tag{7}$$

During the optimization process, centers are not fixed anymore, as both weights and centers are getting updated. Also, the centers and feature outputs are normalized as $||c_j||_2 = 1$ and $||z_j||_2 = 1$, and therefore they reside on a unit hyper-sphere. Eq. 8 shows the formulation of MDF.

$$\begin{aligned} \varepsilon^{intra} &= \frac{1}{|\mathcal{D}|} \sum_{\forall \mathcal{D}_i \in D} \sum_{z_j \in \mathcal{D}_i} \|z_j - c_i\|_2^2 \\ \varepsilon^{inter} &= \frac{1}{\binom{m}{2}} \sum_{o=1}^{m} \sum_{r=1, r \neq o}^{m} max(0, \alpha - \|c_o - c_r\|_2^2) \\ MDF &= \beta_1 \varepsilon^{intra} + \beta_2 \varepsilon^{inter} \end{aligned} \tag{8}$$

where ε^{intra} term is the intra-class term that tries to minimize a sample's distance to its class center, and ε^{inter} is the inter-class term that is responsible for enforcing margin α between centers of every two classes. Two hyperparameters β_1 and β_2 control the intra-class and inter-class terms, respectively.

3.4 Training procedure

In this subsection, we describe the training procedure of MMFD as shown in Algorithm 1. At each iteration of the algorithm, a mini-batch of samples is selected for training (line 2). The features of each source are extracted by the automated feature extraction component described in Section 3.1 (line 5). The extracted features are re-weighted according to the proposed attention mechanism described in Section 3.2 (line 8). We combine MDF (Eq. 8) and cross entropy as the model loss function and their contributions

Algorithm 1: Training procedure of MMFD.

Data: Sample $\mathcal{X} = \{X_1, \cdots, X_n\}$ and labels $Y = \{y_1, \cdots, y_n\}$
Input: Learning rate: η; weight of MDF : λ; and hyperparameters of MDF: β_1, β_2 and α
Output: Model $\mathcal{M}$ and parameters $\theta = \{\theta_1, \cdots, \theta_p\}$

1 **while** *Not convergent* **do**
2 Select mini-batch $\mathcal{B} \subseteq \mathcal{X}$
3 **foreach** $X_j \in \mathcal{B}$ **do**
4 **foreach** $x_j^i \in X_j$ **do**
5 $v_i = FE_i(x_j^i)$ /* see Figure 1 */
6 $r_i = v_i^T \times W_i + b_i$ /* see Figure 3 */
7 **end**
8 $z_j = \mathcal{G}(\sum_{i=1}^{k} a_i r_i)$ /* see Figure 3 */
9 $p(z_j) = Softmax(z_j)$
10 **end**
11 $CrossEntropy = -\sum_{z_j} p(y_j) \times log(p(z_j))$
12 $J = \lambda \times CrossEntropy + (1 - \lambda)(\beta_1 \varepsilon^{intra} + \beta_2 \varepsilon^{inter})$ /* refer to Eq. 8 for ε terms */
13 Backpropagate the error to get $\nabla_\theta J(\theta)$
14 $\forall \theta_i \in \theta, \theta_i = \theta_i - \eta \nabla_{\theta_i} J(\theta)$
15 **end**
16 **if** *condition* = *TRUE* **then**
17 $CalculateCenters(\mathcal{X}; \mathcal{M})$
18 **end**
19 **return** $\mathcal{M}$, $\{\theta_1, \cdots, \theta_p\}$

are controlled by the hyperparameter λ (line 12). Algorithm 1 backpropagates the error to compute gradients of the loss function with respect to each parameter of the model. Following the common way, we use Stochastic Gradient Descent (SGD) to update the parameters (line 14). Finally, once the training converges, the optimized parameters are returned, which can be used for prediction.

4 Experiment

In this section, we empirically evaluate the proposed framework on a real-world fake news dataset. We demonstrate the effectiveness of the approach by conducting a set of experiments. Particularly, we seek to answer the following research questions.

- *Q1*. How does the proposed framework perform on fake news detection?
- *Q2*. Can the proposed framework handle multiple sources effectively?
- *Q3*. How does each component of the proposed framework affect the performance?

We first present the experimental settings. Then, the experiments and their results are conducted to answer the aforementioned questions. Finally, we present a case study to qualitatively demonstrate the effectiveness of the proposed approach.

4.1 Experimental settings

In this work, we use the dataset published in (Wang, 2017) known as LIAR. LIAR is one of the largest publicly available fake news datasets. It has been constructed from a collection of statements investigated by experts in *politifact.com*, which is a well-known fact-checking service. The author did split the dataset into three sets: train, test, and development. His sample selection for sets is random where the train, test, and development sets contain, respectively, 80%, 10%, and 10% of the entire dataset. We use the same split as Wang (Wang, 2017). LIAR contains three sources of data. We supplement it and add another source. In the following, we describe the sources used to evaluate the proposed framework.

- **Statements.** The statements are short sentence(s), mostly from American politicians covering different topics. The statements have been manually classified into six classes including *True*, *Mostly-True*, *Half-True*, *Barely-True*, *False*, and *Pants-on-Fire*. We use the architecture presented in Section 3.1 (Figure 2) to model the statements.

- **Metadata.** The metadata is a textual context about the statements or their speakers if known. This includes the name of a statement's speaker, his/her job title, his/her political party affiliation, the U.S state associated with a speaker, and the venue/location that a statement has been made. When a speaker is unknown, the related fields are blank. We combine all metadata fields and use the architecture presented in Section 3.1 (Figure 2) to model it.

- **History.** The history is a non-textual source. It is a 5-dimensional vector representing a speaker's count of statements on five classes, namely *Mostly-True*, *Half-True*, *Barely-True*, *False*, and *Pants-on-Fire*. The history is modeled by a single-layer fully connected network with 10 hidden units.

- **Report.** In addition to the sources described above, we supplement the dataset by adding the verdict reports generated by experts in *politifact.com*. The reports are longer than the statements and metadata. We remove the class labels from the reports. Similar to other textual sources, the architecture presented in Section 3.1 (Figure 2) is utilized to model the reports.

We use the train set to train the model. For the textual sources, we populate word embeddings (see Figure 2) from the Google word2vec embeddings trained on roughly 100 billion words from Google News (Mikolov et al., 2013). In all experiments, we utilize the development set to tune the hyperparameters, which have been done carefully with grid search over ranges of different values. We utilize the mini-batch 32, apply the dropout rate 60%, and use Adam optimizer (Kingma and Ba, 2014) to apply the gradient descent with the learning rate of 0.001. For center computation in MDF (Eq. 7), we gradually increase the center computation period starting from 20 optimization steps to 160 steps as the training proceeds. The hyperparameters β_1, β_2, and α are set to 0.6, 0.4, and 0.3, respectively. The number of hidden units in LSTM network is set to 200. The test set is used to evaluate the effectiveness of the model and the performance reported in this paper is based on the evaluation of the test set. For the performance metric, we use accuracy since the dataset is fairly balanced and also consistent with Wang (Wang, 2017), we found that f-measure performs similarly as accuracy.

4.2 Performance comparison

To answer questions *Q1* and *Q2*, we compare our approach, MMFD, with a set of representative baselines:

- Basic-SVM. For this baseline, we extract a set of features from sources. For the textual sources, we extract Bag-of-Words, bigrams, and 3-grams, and simply combine them with the history. Since the number of features is large, we apply PCA to reduce the dimension to 300. Finally, we train a SVM model (Support Vector Machine) on the extracted features. We use a grid search on the performance of the development set to tune the SVM classifier hyperparameters.

- Basic-RandomForests. This baseline is similar to Basic-SVM except that Random Forests is employed as the classifier.

- Basic-NN. In addition to the employed traditional classifiers, i.e., SVM and Random Forests, we also use a NN (Neural Network) as the baseline. The input to NN is the same with other two classifiers. We use a fully connected network with one layer. Again, the hyperparameters are set using the development set. Basic-NN further refines the initial features.

- Wang (Wang, 2017). Wang (Wang, 2017) developed a model based on CNN and BLSTM (Bi-directional LSTM) for fake news detection on LIAR dataset.

- Random. Random includes randomly selecting the class of a test sample.

- Majority. In this method, each test news item is labeled with a class having the largest number of samples i.e., the *Half-True* class.

Table 1: Performance comparison. S1: Statement, S2: Metadata, S3: History, S4: Report

Sources	Method	Accuracy (%)
–	Random	17.4
–	Majority	20.8
S1	Basic-SVM	20.12
	Basic-RandomForests	23.19
	Basic-NN	21.73
	Wang (Wang, 2017)	27.00
	MMFD	**29.06**
S1+S2+S3	Basic-SVM	25.07
	Basic-RandomForests	27.63
	Basic-NN	28.16
	Wang (Wang, 2017)	27.04
	MMFD	**34.77**
S1+S2+S3+S4	Basic-SVM	29.98
	Basic-RandomForests	27.01
	Basic-NN	29.12
	Wang (Wang, 2017)	N/A
	MMFD	**38.81**

The comparison results on different combinations of sources have been shown in Table 1. As mentioned before, the original dataset (Wang, 2017) contains only the sources "S1: Statement", "S2: Metadata" and "S3: History" and we enrich the dataset by adding the fourth source "S4: Report". Thus, we only report combinations with news content (i.e., S1), all sources in the original dataset (i.e., S1+S2+S3) and all sources in the enriched dataset (i.e., S1+S2+S3+S4). Note that the performance of Wang (Wang, 2017) is not available on the enriched dataset. We make the following observations from these results:

- MMDF outperforms Wang (Wang, 2017) when only using the news content (e.g., S1). Both methods are based on CNN + LSTM. However, the proposed framework also has the model component to discriminate between multiple classes. This observation supports the importance of the proposed discriminative function i.e., MDF.

- By incorporating more sources, the performance of all methods tends to increase. Compared to baselines, the proposed framework MMDF enjoys more performance improvement with more sources. These observations suggest that the proposed framework can effectively combine multiple sources.

- The proposed framework always obtains the best performance with all settings. In the following subsection, we further investigate how each model component contributes to the performance improvement.

To sum up, the proposed framework outperforms representative baselines and it is also effective in integrating information from multiple sources.

4.3 MMFD component analysis

In this subsection, we conduct several experiments to study the impact of model components on the performance of the proposed framework to answer *Q3*. The experiment are as follow:

- $MMFD_{PCA}$. In this experiment, we remove feature extractors in MMDF (FEs in Figure 1). Instead, we apply PCA (with dimension 100) on each input of the textual sources. We keep the history and the interpretable multi-source fusion and the fakeness discrimination components unchanged.

Table 2: Impact of model components on the proposed framework.

Investigated component	Setting	Accuracy (%)
Automated feature extraction	$MMDF_{PCA}$	26.06
	$MMDF_{NN}$	30.95
Multi-source fusion	$MMDF_{Concat}$	28.69
	$MMDF_{EQ}$	32.08
Fakeness discrimination	$MMDF_{CE}$	31.11
	$MMDF_{CL}$	35.17
–	MMFD	**38.81**

Table 3: A case study demonstrating the interpretability of the proposed framework.

Source	Content	Attention score
Statement	"Virtually every person across this country has seen premiums going up and up and up due to Obamacare"	0.17
Metadata	Ted Cruz, Senator Texas, Republican a conversation with reporters	0.09
History	[36 33 15 19 8]	0.43
Report	The full report is available from the URL[1]	0.31

- $MMFD_{NN}$. For this configuration, we perform the same feature extraction as $MMFD_{PCA}$ except that instead of PCA, we extract features from a NN.
- $MMFD_{Concat}$. In this setting, we keep the automated feature extraction and the fakeness discrimination components. However, we replace the interpretable multi-source fusion component by concatenating extracted features.
- $MMFD_{EQ}$. Similar to $MMFD_{Concat}$, in this setting, we keep the automated feature extraction and the fakeness discrimination components. However, we replace the interpretable multi-source fusion component by explicitly considering all the sources equally important. In other words, in Eq. 4 all a_is are set to the same value.
- $MMDF_{CE}$. In this setting, we evaluate the effectiveness of the fakeness discrimination component. We keep two other components and remove MDF from the loss function of the model and just keep Cross Entropy (CE).
- $MMDF_{CE}$. Center Loss (CL) was proposed in (Wen et al., 2016). This function penalizes cross entropy by an intra-class sparsity. In other words, it just tries to move samples toward their class centers. In this setting, we use Center Loss and cross entropy.

The results of component analysis with all sources in the enriched dataset are presented in Table 2. It can be observed that:

- Automated feature extraction effectively extracts features. Neither $MMFD_{FCN}$ nor $MMFD_{PCA}$ can extract informative features compared to the automated feature extraction component.
- Hand-crafted features are ineffective. Features in $MMFD_{FCN}$ and $MMFD_{PCA}$ are basic linguistic features which are not effective as much as the automated ones from deep networks.
- The interpretable multi-source fusion component integrates features effectively. Simple concatenation methods such as $MMFD_{Concat}$ and $MMFD_{EQ}$ fail to distinguish different sources. The reason for better performance of $MMFD_{EQ}$ is because the linear projection presented in Eq. 3.
- MMFD can discriminate different classes effectively. Neither $MMDF_{CE}$ nor $MMDF_{CL}$ can effectively discriminate different classes. The reason for better accuracy of $MMDF_{CL}$ is due to the extra penalty added to the cross entropy.

In conclusion, the three major model components including automated feature extraction, multi-source fusion, and fakeness discrimination can help boosting the performance of the proposed framework in detecting fake news.

[1] http://www.politifact.com/truth-o-meter/statements/2013/oct/17/ted-cruz/sen-ted-cruz-says-premiums-have-gone-virtually-eve/

4.4 A case study

As mentioned before, the interpretable multi-source fusion components equips the proposed framework with a sort of interpretability. To investigate the interpretability power of the framework , we present a case study demonstrated in Table 3. In this case study, we randomly select a sample news item from the test set. The ground truth label of this news item is *False* and the model correctly predicts it as *False*. Then, we show the attention score of each source. As shown in Table 3, the highest score is given to the history source. This seems reasonable as the history of the speaker, Ted Cruz, regarding the number of previous false statements is quite high. Moreover, the report attains the second highest attention score. This seems reasonable as well because the report includes contextual detail about a news item. In addition, it is quite hard to predict the fakeness of news from the short news statement and the writer's profile; hence low scores are given to these two sources.

5 Related Work

There have been substantial number of works studying fake news and misinformation detection in recent years. One well-known approach is taking advantage of linguistic features. In their seminal work (DePaulo et al., 2003), DePaulo et al. shed light on cues of fake stories from physiological point of view. They pointed out that fake stories contain an unusual language. Li et al. (Li et al.,) discovered that linguistic features such as sentiments and singular pronouns are informative in online spam reviews. In (Qazvinian et al., 2011), the authors used unigram, bigram and Part-of-Speech (POS) features of tweets for rumor detection. Martinez-Romo and Araujo (Martinez-Romo and Araujo, 2013) used discrepancy and lack of semantic relation between the language of spam tweets and that of the websites redirected by those tweets. Kumar et al. (Kumar et al., 2016) showed that Wikipedia hoaxes tend to contain more words than genuine articles. Also, Linguistic Inquiry and Word Count (LIWC) (Pennebaker et al., 2015) has been employed to investigate the role of individual words in a document for deception detection (Ott et al., 2011). For a review of other linguistic features, you can refer to survey papers (Conroy et al., 2015; Heydari et al., 2015).

Despite the success of aforementioned works, we still lack a comprehensive and optimized set of features for fake news. This is even more stressed for short statements as they offer little to fake news detection. Hence, deep neural networks as automatic feature extractors have gained attention for fake news detection (Ruchansky et al., 2017). Closely related to our work is (Wang, 2017). Wang (Wang, 2017) introduced the dataset used for our evaluation. Further, he developed a combination of CNN and LSTM for modeling fake news detection. Our model is substantially different from that of Wang (Wang, 2017) because, 1) we provide an interpretable model component to combine multiple sources, (2) we propose a new discriminative function, MDF, to discriminate degrees of fakeness, (3) we supplemented the dataset and modeled the verdict reports.

6 Conclusion

In this study, we investigated the challenging task of multi-source multi-class fake news detection. The task faces several challenges – how to incorporate multiple sources and how to discriminate degrees of fakeness. To address these challenges, we proposed a coherent and interpretable framework MMFD, which incorporates automated feature extraction, multi-source fusion and fakeness discrimination. Through extensive experiments, we demonstrated that our model can effectively distinguish different degrees of the fakeness of news. In fakeness discrimination, we treated all classes the same. Thus, we would like to incorporate class differences by enforcing larger margin between certain classes. By doing so, several classes can be merged easily if needed, allowing for a less fine grained but possibly more precise detection. Another possible research direction is to incorporate more sources such as temporal information, social networks and user interactions.

Acknowledgments

This work is supported by the National Science Foundation (NSF) under grant number IIS-1714741 and IIS-1715940. We would like to thank Nazanin Donyapour for her help and support.

References

[Allcott and Gentzkow2017] Hunt Allcott and Matthew Gentzkow. 2017. Social media and fake news in the 2016 election. *Journal of Economic Perspectives*, 31(2):211–36.

[Bahdanau et al.2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Bengio et al.2013] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

[Conroy et al.2015] Niall J Conroy, Victoria L Rubin, and Yimin Chen. 2015. Automatic deception detection: methods for finding fake news. In *Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community*, page 82. American Society for Information Science.

[Del Vicario et al.2016] Michela Del Vicario, Alessandro Bessi, Fabiana Zollo, Fabio Petroni, Antonio Scala, Guido Caldarelli, H Eugene Stanley, and Walter Quattrociocchi. 2016. The spreading of misinformation online. *Proceedings of the National Academy of Sciences*, 113(3):554–559.

[DePaulo et al.2003] Bella M DePaulo, James J Lindsay, Brian E Malone, Laura Muhlenbruck, Kelly Charlton, and Harris Cooper. 2003. Cues to deception. *Psychological Bulletin*, 129(1):74.

[Gupta et al.2014] Aditi Gupta, Ponnurangam Kumaraguru, Carlos Castillo, and Patrick Meier. 2014. Tweetcred: Real-time credibility assessment of content on twitter. In *International Conference on Social Informatics*, pages 228–243. Springer.

[Heydari et al.2015] Atefeh Heydari, Mohammad ali Tavakoli, Naomie Salim, and Zahra Heydari. 2015. Detection of review spam: A survey. *Expert Systems with Applications*, 42(7):3634–3642.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Kalchbrenner et al.2014] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

[Khurana and Intelligentie2017] Urja Khurana and Bachelor Opleiding Kunstmatige Intelligentie. 2017. The linguistic features of fake news headlines and statements.

[Kingma and Ba2014] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kumar et al.2016] Srijan Kumar, Robert West, and Jure Leskovec. 2016. Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes. In *Proceedings of the 25th international conference on World Wide Web*, pages 591–602. International World Wide Web Conferences Steering Committee.

[LeCun et al.2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436–444.

[Li et al.] Jiwei Li, Myle Ott, Claire Cardie, and Eduard H Hovy. Towards a general rule for identifying deceptive opinion spam.

[Martinez-Romo and Araujo2013] Juan Martinez-Romo and Lourdes Araujo. 2013. Detecting malicious tweets in trending topics using a statistical analysis of language. *Expert Systems with Applications*, 40(8):2992–3000.

[Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Ott et al.2011] Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 309–319. Association for Computational Linguistics.

[Pennebaker et al.2015] James W Pennebaker, Ryan L Boyd, Kayla Jordan, and Kate Blackburn. 2015. The development and psychometric properties of liwc2015. Technical report.

[Qazvinian et al.2011] Vahed Qazvinian, Emily Rosengren, Dragomir R Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1589–1599. Association for Computational Linguistics.

[Ruchansky et al.2017] Natali Ruchansky, Sungyong Seo, and Yan Liu. 2017. Csi: A hybrid deep model for fake news. *arXiv preprint arXiv:1703.06959*.

[Shu et al.2017a] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017a. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1):22–36.

[Shu et al.2017b] Kai Shu, Suhang Wang, and Huan Liu. 2017b. Exploiting tri-relationship for fake news detection. *arXiv preprint arXiv:1712.07709*.

[Wang2017] William Yang Wang. 2017. "liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*.

[Webb et al.2016] Helena Webb, Marina Jirotka, Bernd Carsten Stahl, William Housley, Adam Edwards, Matthew Williams, Rob Procter, Omer Rana, and Pete Burnap. 2016. Digital wildfires: hyper-connectivity, havoc and a global ethos to govern social media. *ACM SIGCAS Computers and Society*, 45(3):193–201.

[Wen et al.2016] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. 2016. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515. Springer.

[Wingfiled et al.2016] Nick Wingfiled, Mike Isaac, and Benner Kate. 2016. Google and facebook take aim at fake news sites, November. [Online; posted Nov. 14, 2016].

[Xu et al.2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

[Yang et al.2012] Lei Yang, Tao Sun, Ming Zhang, and Qiaozhu Mei. 2012. We know what@ you# tag: does the dual role affect hashtag adoption? In *Proceedings of the 21st international conference on World Wide Web*, pages 261–270. ACM.

Killing Four Birds with Two Stones: Multi-Task Learning for Non-Literal Language Detection

Erik-Lân Do Dinh, Steffen Eger, and Iryna Gurevych
Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
http://www.ukp.tu-darmstadt.de

Abstract

Non-literal language phenomena such as idioms or metaphors are commonly studied in isolation from each other in NLP. However, often similar definitions and features are being used for different phenomena, challenging the distinction. Instead, we propose to view the detection problem as a generalized non-literal language classification problem. In this paper, we investigate multi-task learning for related non-literal language phenomena. We show that in contrast to simply joining the data of multiple tasks, multi-task learning consistently improves upon four metaphor and idiom detection tasks in two languages, English and German. Comparing two state-of-the-art multi-task learning architectures, we also investigate when soft parameter sharing and learned information flow can be beneficial for our related tasks. We make our adapted code publicly available[1].

1 Introduction

As Lakoff and Johnson (1980) argued, metaphorical concept mappings, often from concrete to more abstract concepts, are ubiquitous in everyday life, thus they are ubiquitous in written texts. But the same is true for other kinds of so-called *non-literal language*, e.g., for idioms. Boundaries between these instances of non-literality are not always clear: for instance, "[...] a swathe of sunlight lay across the red-tiled floor."[2] can be classified as a metaphor because of its non-lexicalized figurative meaning. Few would dispute the idiomaticity of "kicking the bucket", as the non-literal meaning in this multi-word expression is largely conventionalized. However, in the sentence "One approach would be to draw the line by reference [...]" the expression "draw the line" could be classified as either metaphorical (because it still evokes the literal senses of its constituents) or idiomatic (as it is a fixed expression with lexicalized figurative sense).

Much effort has been spent on the detection of metaphors, idioms, and general non-literal language use (Shutova, 2015). However, because of the named vague and subjective nature of these phenomena, a multitude of datasets using differing definitions has emerged in the process. Even when datasets address the same aspect of non-literality, they may use diverging or underspecified definitions; compare, e.g., the guidelines for annotating metaphors of Tsvetkov et al. (2014) ("[...] all words that, in your opinion, are used non-literally [...]") and Mohammad et al. (2016) ("more complex; more distant from our senses; more abstract; more vague; ...").

The fuzziness of non-literality has two natural consequences: (1) training data is sparse, because different researchers may use diverging definitions, and hence may annotate different things rather than extend "the same story"; (2) high-quality training data is costly to obtain because there may be considerable disagreement among crowd-workers and even trained experts regarding the labels for different instances of (non-)literality.

In this work, we address two research questions:

[1] https://github.com/UKPLab/coling2018-nonliteral-mtl

[2] VUAMC (Steen et al., 2010)

Proceedings of the 27th International Conference on Computational Linguistics, pages 1558–1569
Santa Fe, New Mexico, USA, August 20-26, 2018.

- (how) can we leverage existing datasets for non-literality, possibly using diverging definitions and addressing different aspects of non-literality, to bootstrap classifiers on new sparse data? This question is particularly relevant for non-literal language detection due to problem (2) above;

- do existing datasets for non-literality share common ground in the first place or are many of them using arbitrary and mutually exclusive definitions? If the answer to this question is "yes", then this would call for a re-thinking of a whole computational research area, and a return to safe grounds.

To address our research questions, we apply *multi-task learning* (MTL) to four non-literal language datasets. Introduced by Caruana (1993), MTL has recently been used to improve a variety of NLP tasks with the help of other, auxiliary tasks. This includes chunking (Søgaard and Goldberg, 2016), using POS tagging, and spelling normalization (Bollmann and Søgaard, 2016), using orthographic data from other domains, but also semantic tasks like entailment (Hashimoto et al., 2017) or argument mining (Schulz et al., 2018). The underlying idea of MTL is that it is beneficial to learn several tasks jointly because of "spill-over" effects. Some researchers have claimed that a requirement for success in MTL is task relatedness. If this is true, then MTL is a formidable testbed for *both* of our research questions.

We aim at **four birds**, namely: We consider detection of non-literal phenomena in four different datasets, regarding each as a separate task. These are (A) *metaphor detection in content tokens*, (B) *classification of metaphorical adjective-noun constructions*, (C) *detection of idiomatic use of infinitive-verb compounds*, and (D) *non-literal usage of particle verbs*. Two of the datasets comprise English data, the other two consist of German data.

We throw **two stones** at them, namely: (i) an MTL sequence tagging framework using hard-parameter sharing (Kahse, 2017) and (ii) Sluice Networks (Ruder et al., 2017), for which sharing of information is not hard-wired, but can adjust softly. Both frameworks yield different insights because the first has to make use of all available data, while the second can freely decide if other tasks contain relevant information and if so, how much sharing to enable.

This work is structured as follows. We first ground our motivation to tackle four non-literal language tasks in Section 2. There, we also discuss applications of MTL, from pure syntactic to higher-level, semantic tasks. In Section 3, we describe the two systems that we compare in our experiments. We specify the used datasets and describe their differences in Section 4. In Section 5, we discuss our experiments and results. Finally, we conclude with an outlook on possible future work in Section 6.

2 Related Work

Work in non-literal or figurative language classification usually revolves around one specific phenomenon, be it metaphor, idiom or also metonymy detection. While many approaches are monolingual, some explore cross-lingual non-literal language classification. Tsvetkov et al. (2014) train models separately on English adjective-noun and subject-verb-object metaphors using random forests and a variety of semantic features, including supersenses and concreteness values. The models are then used to classify metaphors in similarly annotated Spanish, Russian, and Farsi test sets.

To the best of our knowledge, a combined detection of multiple non-literal phenomena has not been conducted before. This is surprising because common semantic features have already been used to classify different kinds of non-literal language.

One such typical feature in non-literal language classification is the violation of selectional preferences (Wilks, 1978). It is used, e.g., in *met** (Fass, 1991) to classify metaphors and metonyms. While the system distinguishes between both phenomena, it does so only after using the selectional preference information. In a related task, Horbach et al. (2016) employ this information for classifying idiomatic uses of infinitive verb compounds. Another feature used across different non-literal language detection tasks is topic information. While the work by Horbach et al. (2016) includes this feature for idiom detection, Beigman Klebanov et al. (2014) utilize it to classify metaphorically used tokens. Additionally, they make use of concreteness ratings, grounded in the Conceptual Metaphor Theory (Lakoff and Johnson, 1980). However, as argued in our introduction, concreteness is also useful for the detection of other kinds of non-literal language. For example, Zhang and Gelernter (2015) utilize such ratings to detect

metonymy. Further, supersenses are employed to detect metaphors (Tsvetkov et al., 2014) or non-literal language in general (Köper and Schulte im Walde, 2017). One more feature that is often integrated is textual cohesion, e.g., in metaphor (Schulder and Hovy, 2014) and idiom detection (Sporleder and Li, 2009). The use of such common features suggests that different aspects of non-literality require similar information and that representation sharing may thus turn out beneficial.

Introduced in the early nineties (Caruana, 1993), multi-task learning (MTL) has been more widely and successfully used in NLP recently (Ruder, 2017). MTL denotes a machine learning technique in which multiple tasks are trained in parallel in the same system, using a shared representation. The goal is to take advantage of commonalities between the different tasks. Bollmann and Søgaard (2016) use multi-task learning for historical spelling normalization. They employ texts from different domains as main respectively auxiliary tasks and improve upon a CRF baseline. Søgaard and Goldberg (2016) show that task supervision at lower layers for lower-level syntactic tasks like POS-tagging is beneficial to higher-level syntactic tasks such as chunking and CCG supertagging. Since their experiments with semantic tasks (NER, supersense tagging) on the higher levels show no improvement, they conclude that tasks need to be "sufficiently similar, e.g., all of syntactic nature" for multi-task learning to increase performance. Their conclusion is challenged by Hashimoto et al. (2017), who create a similar multi-task learning network in which lower layers predict syntactic tasks, while higher layers predict sentential relatedness and entailment. Their semantic tasks also improve when introducing *shortcut connections*, i.e., feeding the word representations into all layers of their network. In contrast, Alonso and Plank (2017) find generally mixed performance of MTL for semantic tasks. They also use syntactic tasks as low-level auxiliary tasks, but cannot improve performance over a single-task learning baseline for three out of five investigated semantic tasks (NER, supersense classification, frame identification). Schulz et al. (2018) apply MTL to another semantic task, argumentation mining. Instead of syntactic tasks as auxiliaries, they use diverse argumentation mining datasets from different domains. Similar to our tasks, annotation and labels vary across their different tasks due to inherent subjectivity and vagueness of argumentation mining (as for non-literal language detection). Experimenting with artificially downsized datasets, they show that—especially when data for the main task is sparse—MTL improves sequence tagging results for argumentation mining, even when the tasks cover different domains. In contrast, we also investigate the effect of auxiliary tasks with small datasets on a large-data main task.

3 System Descriptions

We conduct our multi-task learning experiments using two different architectures: a sequence tagging framework (MTL-SEQ) (Kahse, 2017), and Sluice Networks (SLUICE) (Ruder et al., 2017).

We adapt both systems to our datasets, which only have labels for few tokens in a sentence. Thus, overfitting on the missing/"empty" labels seems probable, and is confirmed by preliminary experiments. To alleviate this issue, we exclude the loss on the tokens with empty labels from the total loss calculation, so that they do not influence weight updates.

Multi-Task Learning Sequence Tagging Framework (MTL-SEQ) We employ the system by Kahse (2017), a framework for multi-task learning sequence tagging generalizing the model of Søgaard and Goldberg (2016). An example of a two-task setup is shown in Figure 1. It uses English metaphor classification on a token level as the main task, and English metaphor detection in adjective-noun constructions as an auxiliary task (both described in Section 4).

After an input layer that reads in word embeddings, there are multiple shared, bi-directional LSTM layers, thus implementing hard parameter sharing. The shared layers are followed by a number of fully connected task-specific layers (Peng et al., 2017), storing private information for each task. At their end, a softmax classifier predicts labels for each input token.

In addition to using pre-trained word embeddings, the architecture incorporates character-level information to improve handling of out-of-vocabulary words. The framework can further be configured to terminate different tasks at different layers. We set this option to randomly use one of two scenarios: either all tasks use all BiLSTM layers, or all auxiliary tasks terminate one layer before the main task.

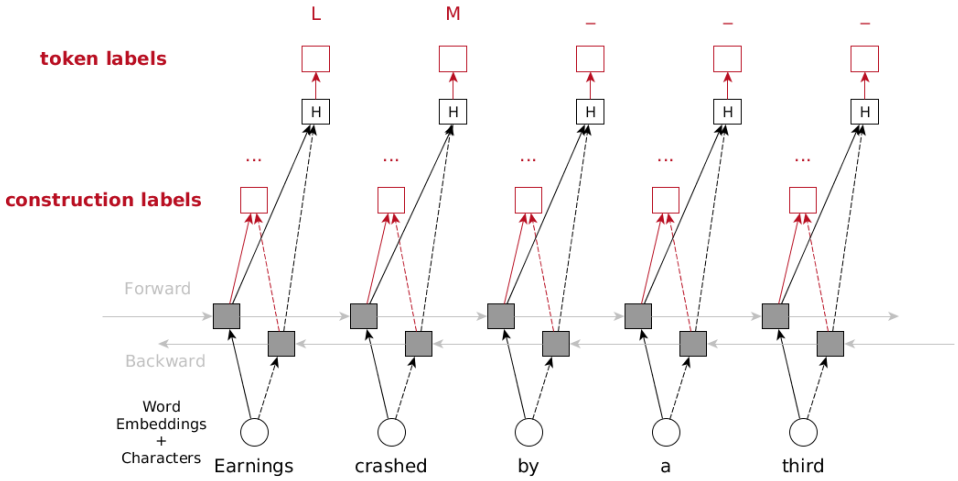


Figure 1: Example setup for the sequence tagging MTL framework (Kahse, 2017). It shows auxiliary task `adj-noun-met` and main task `tok-met`. Gray blocks represent shared BiLSTM units. In this example, both tasks are terminated after one shared BiLSTM and multiple private fully connected layers (white *H* blocks). The input is task-specific and only evaluated for the respective task; pictured is input for `tok-met`. "Earnings" is labeled as literal, "crashed" is labeled as a metaphor; the remaining tokens have no label and their loss is excluded from the total loss calculation.

We optimize over several hyper-parameters, which include the number of task-specific fully connected layers, the layer at which tasks should terminate, and word dropout rate. An overview of the hyper-parameters and the range from which we randomly sample their values is shown in Table 1.

Sluice Networks (SLUICE) Ruder et al. (2017) introduced *sluice networks* as a generalization and combination of various MTL architectures that use different information transfer techniques. Thus, it can emulate, e.g., hard parameter sharing (Caruana, 1993), where the same layer weights (i.e., the same layers) are used for different tasks, and cross-stitch networks (Alonso and Plank, 2017), in which trainable parameters decide the amount of shared information between layers of separate networks. A generic example is given in Figure 2.

The system first reads input in an embedding layer, and combines the word embeddings with character-level embeddings (produced by a BiLSTM). This representation is passed to task-specific BiLSTM networks, which learn to share parts of their layer outputs, i.e., activations, in the following way. Each layer contains two different subspaces, to store both task-specific, and shared representations. Trainable parameters α determine the amount of sharing between the subspaces of the task BiLSTMs. Parameters β decide to which degree the output of the BiLSTM layers should be used for the task-specific classifier (a multilayer perceptron). This way, they allow for hierarchical relations.

Most hyper-parameters regarding information and weight sharing are encoded into trainable parameters for sluice networks. Thus, the most important hyper-parameters are the number of BiLSTM layers in the network, the way layer output is combined, and constraints on the subspaces (see Table 2).

Hyper-parameter	value range
number of of shared BiLSTM layers	{1, 2, 3}
number of task specific fully connected layers	{0, 2}
Word dropout	[0.0, 0.5]
Dropout	[0.0, 0.5]
Activation function for fully connected layers	{tanh, relu}
Output layer	{0, 1, 2}

Table 1: Hyper-parameters for the multi-task learning sequence tagging framework (Kahse, 2017).

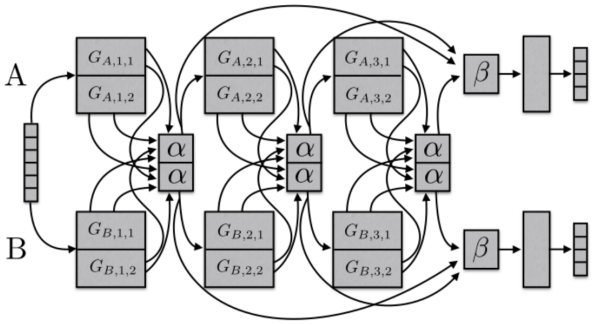


Figure 2: Example for sluice networks (Ruder et al., 2017). A single input representation is passed into task networks A and B. Both of the two networks i have three RNN layers j, consisting of two subspaces $G_{i,j,k}$ each. For each layer output, α determines the amount of sharing between the layers of different tasks while β controls the layer at which a task network terminates.

Conceptually, the largest difference between both systems is the following: MTL-SEQ implements hard parameter sharing, i.e., the layer weights are the same for all tasks, and the structure of the network is predefined; in particular, at which layer a task should terminate. Essentially, this amounts to using one network for multiple tasks, trained jointly. In contrast, SLUICE has separate layers for each task, and learns parameters which control the information flow between those task-specific layers. Further, the level at which task predictions take place is also learned for each task. This means that two networks are trained in parallel, that share a learnable amount of information. Comparing both approaches, we can analyze which tasks profit from which degree of information sharing.

Hyper-parameter	value range	notes
number of BiLSTM layers	{2, 3}	
layer connections	{stitch, skip}	*stitch* denotes learnable β parameters, while *skip* only uses the last BiLSTM layer as classification input
subspace constraint weights	{0.1, 0.2}	

Table 2: Hyper-parameters for the sluice networks.

4 Datasets and Tasks

In our experiments, we consider four different non-literal language tasks (Table 3). Specifically, we include two metaphor detection tasks, with data being annotated on a token and on a construction level (i.e., grammatical constructions that share one label). We further include data from two tasks classifying idiomatic language use. The latter two tasks use German data, while the metaphor task datasets are in English. While those tasks all cover distinct, or at least distinctly annotated, non-literal language phenomena, on a textual level they all are examples of irregular polysemy. Thus, in many feature-based detection systems similar or even the same features and resources are used to classify these phenomena.

Token-level Metaphor Classification We consider metaphor detection on a token level, i.e., each (content) token should be classified as belonging to a metaphor or not. We employ the *VU Amsterdam Metaphor Corpus* (Steen et al., 2010), short *VUAMC*, for our experiments. It is the largest available resource in which each token is specifically annotated as being used metaphorically or literally, and comprises four genres: *academic*, *conversation*, *fiction*, and *news* texts.

The VUAMC is annotated using very rigid annotation guidelines, *MIPVU* (Steen et al., 2010). In short, each token that is not used in its most basic (i.e., most concrete, body-related, historically oldest) sense is labeled as metaphoric. Unclear cases are resolved using specified online dictionaries. These guidelines lead to a high inter-annotator agreement of 0.84 Fleiss' κ. At the same time, they introduce the problem of very commonly used word senses being annotated as metaphoric. For example, prepositions such

Task	dataset	size	M	lang	example
Token-level metaphor detection	VUAMC	103,865	15%	en	"Along with Sir James he **found** the US much more **attractive**, [...]"
Construction level metaphor detection	Tsvetkov et al., (2014)	1,738	47%	en	Wind and wave power providing the **green energy** of the future.
Classification of idiomatically used verb compounds	Horbach et al., (2016)	5,249	64%	de	"Auch eine Uhr, die **stehen geblieben** ist, geht zweimal am Tag richtig", sagt er. ("A clock that has **stopped running**, is correct two times a day, too," he says.)
Classification of non-literally used particle verbs	Köper and Schulte im Walde (2016)	6,436	35%	de	Auf Decken sitzt man ums Feuer und lässt den ereignisreichen Tag **nachklingen**. (You sit around the fire on blankets and let the day **linger on** [lit.: ring on].)

Table 3: Investigated tasks and datasets. *Size* describes labeled tokens in case of token-level metaphor detection (content tokens), and labeled constructions for the other tasks respectively, *M* denotes percentage of non-literal labels. Non-literal use of tokens/constructions in the examples is marked bold.

as *on* are labeled as metaphoric when not being used in their positional sense. We partially avoid this problem by filtering out non-content words and auxiliary verbs. Dataset statistics are given in Table 3.

Construction-level Metaphor Classification Many datasets in metaphor detection tasks label *grammatical constructions* instead of tokens, prominently, e.g., adjective-noun compounds or subject-verb/verb-object compounds. We use the adjective-noun dataset by Tsvetkov et al. (2014). Their training set was created by two annotators and later curated, while the test set—comprised of sentences from the TenTen web corpus (Jakubíček et al., 2013)—was annotated by five annotators. Fleiss' κ for the latter is reported as 0.74.

Compared to the VUAMC, the guidelines for this dataset are substantially less specific, appealing to the intuition of the annotators (in fact, annotators were asked to mark all *non-literally* used words). Due to this difference in guidelines and the focus on adjective-noun metaphors, the resulting annotations also differ from the VUAMC, which is why we consider this a different task.

For the test set, the original 200 sentences containing the construction are available. We automatically crawl large corpora[3] to obtain sentences for the constructions in the training set. Note that both training and test sets are substantially smaller than those of the VUAMC (Table 3), by a factor of 50.

Classification of idiomatically used infinitive-verb compounds In contrast to the metaphor datasets which can also include more novel meanings of the annotated words, e.g., "spot of information", Horbach et al. (2016) focus on idiomatic usage of verb compounds. This means they consider only conventionalized figurative meanings, i.e., compounds whose figurative sense is already lexicalized. They create a corpus of literal and idiomatic uses of six German infinitive-verb compounds (e.g., *sitzen lassen—to leave sitting/abandon*). Thus, this dataset contains considerably more samples for a specific compound (up to 1,000 for each) compared to the previous two datasets. Genres covered are newspaper and magazine articles published from 1993–2013.

Two lexicographers were tasked to annotate occurrences of the compounds within a three-sentence

[3]ukWac (Baroni et al., 2009), British National Corpus (BNC Consortium, 2007)

context as *literal* or *idiomatic*, but were given no further guidelines. Still, they achieve an inter-annotator agreement of 0.72 Cohen's κ. The dataset contains the unanimously labeled instances and a portion of adjudicated ambiguous cases.

Classification of non-literally used particle verbs Similar to the infinitive-verb compound dataset, Köper and Schulte im Walde (2016) investigate pre-set constructions, specifically 165 different German particle verbs. For each particle verb, they selected up to 50 sentences containing *literal* and *non-literal* uses from a German web corpus, *DECOW14AX* (Schäfer and Bildhauer, 2012; Schäfer, 2015); however, they do not discuss the types of non-literality covered. Annotation was conducted by three German speakers with "linguistic background" on a 6-point scale; guidelines are not disclosed. Köper and Schulte im Walde (2016) report an inter-annotator agreement of 0.70 Fleiss' κ after mapping the 6-point scale to a binary annotation.

5 Experiments

We conduct three types of experiments for MTL-SEQ and SLUICE:

- a regular single-task learning baseline *(STL)*
- a combined-data single-task learning baseline *(MERGED)*, where we merge the training data of the different tasks
- a multi-task learning setup *(MTL-all)*, where one main task is supported by three auxiliary tasks

The MERGED baseline allows us to differentiate between causes for possible improvements, since we want to investigate in which way additional tasks and data can influence performance. We further partition our MTL experiments along languages, i.e., we carry out additional experiments separately for German *(MTL-de)* and English *(MTL-en)* tasks. Those mono-lingual setups thus only include one auxiliary task.

Setup We use the same train and test split as in Tsvetkov et al. (2014) for `adj-noun-met`. Thus, we use 11% of the data as test data and the remainder as training data. We use 20% of the training data as development data. For `tok-met`, we use a split of 70%/20%/10%. For the two German idiom datasets, we use a training/development/test set split of roughly 80%/10%/10%.

As input to both neural networks we use 100-dimensional, bilingual word embeddings trained using bivec (Luong et al., 2015) on a parallel (en-de) version of the Europarl corpus (Koehn, 2005).

Results We report test set results for the best performing system configurations on the development sets and use macro F_1-score to compare our results. We first analyze the results of the different architectures separately, before comparing both approaches.

Task type	`tok-met`	`adj-noun-met`	`inf-verb`	`part-verb`
STL	0.4399	0.5172	0.8507	0.7037
MERGED	0.4600	0.5976	0.8405	0.6875
MTL-all	**0.4897**	**0.6705**	**0.8602**	**0.7083**
MTL-en	0.4277	0.5357	–	–
MTL-de	–	–	0.8519	0.7065

Table 4: **MTL-SEQ** F_1-scores on the test sets for the different tasks and experiment types. *MTL-all* specifies that the respectively remaining three tasks are used as auxiliary tasks. *MTL-en* and *MTL-de* stand for mono-lingual experiments, i.e., containing only one auxiliary task.

Task type	tok-met	adj-noun-met	inf-verb	part-verb
STL	0.4833	0.5487	0.8609	0.7894
MERGED	0.4570	0.5143	0.8401	0.7965
MTL (all)	**0.5604**	**0.6333**	0.8763	0.8007
MTL (en)	0.5467	0.6207	–	–
MTL (de)	–	–	**0.8833**	**0.8146**

Table 5: **SLUICE** F_1-scores on the test sets for all four tasks and experiment types.

MTL-SEQ Our results for MTL-SEQ are given in Table 4.

For tok-met, MERGED already improves over the STL baseline. MTL-all further improves upon the two. In contrast, including only adj-noun-met as auxiliary task (MTL-en) performs even worse than STL.

For adj-noun-met, the pattern is identical except that MTL-en slightly outperforms STL. Comparing MERGED with MTL-all we see that the performance difference is mainly due to MTL-all introducing fewer new classification errors over the STL baseline.

In other words: both metaphor tasks, where training (and evaluation) data for a particular token or construction is sparse, profit heavily from inclusion of the idiom datasets. In contrast, using only the respective other English metaphor dataset does not help. Here, the difference between STL and MTL-en is not statistically significant (McNemar's Test, $p = 0.22$). It is notable that the German idiom datasets improve the English metaphor tasks by jointly fitting the same network.

The results for inf-verb and part-verb display a different pattern. MERGED performs worse than STL and even though MTL-all improves over both baselines, the difference is not significant. Unlike for the English datasets, MTL-de performs better than STL. The network can better fit to the datasets which are designed to contain many instances of a particular token or construction (inf-verb and part-verb). This is evidenced by the large differences in F_1-scores between the tasks. But these idiom datasets also substantially improve classification for the English metaphors, regardless of differences in language and annotated phenomenon.

SLUICE Results for SLUICE are found in Table 5.

For tok-met, the MERGED baseline performs weaker than STL. In contrast, both MTL approaches improve substantially over the baselines, by more than 6 percentage points over STL. Further, MTL-all outperforms MTL-en, although not significantly so.

Similar to tok-met, for adj-noun-met MTL-all improves significantly over STL. The performance drop from STL to MERGED is similarly large. Also, MTL-all outperforms MTL-en here, even though not significantly. Even though the difference in F_1-score between MTL-all and MTL-en is small, they make considerably different predictions. 16 instances were labeled correctly using MTL-all, but misclassified by MTL-en; and the same number vice versa. Thus, 16% of all 200 test instances were classified differently by the MTL setups. This means that adding the additional idiom tasks does indeed add further information that is helpful for some instances, but detrimental to others—more so than for tok-met.

For inf-verb, STL again outperforms MERGED. MTL-all and MTL-de marginally improve upon the baselines, with the latter performing best, though slightly. Again, both MTL setups show differences in the correctly labeled instances, albeit less than it was the case for adj-noun-met (12%). These differences are distributed similarly over the six compounds. An exception is *sitzen bleiben (to stay seated/to be stuck with something)*, which is considerably better classified using MTL-de than MTL-all.

Results for part-verb differ from the previous tasks, as the MERGED baseline actually perform better than STL. Still, MTL-all and MTL-de improve upon both baselines, with the mono-lingual approach again performing slightly better.

STL		actual	
		M	L
predicted	M	1,396	975
	L	2,580	20,660

MTL-all		actual	
		M	L
predicted	M	1,622	1,027
	L	2,354	20,608

Table 6: `tok-met`: Confusion matrices for MTL-SEQ.

STL		actual	
		M	L
predicted	M	1,567	942
	L	2,409	20,745

MTL-all		actual	
		M	L
predicted	M	2,119	1,467
	L	1,857	20,220

Table 7: `tok-met`: Confusion matrices for SLUICE.

Analysis For both MTL-SEQ and SLUICE, the MTL-all approach outperforms both STL and MERGED baselines. Contrary to MTL-SEQ, SLUICE performs worse on the MERGED baseline than STL for the English tasks.

Consistent between both architectures is that they show significant—and indeed, very substantial—performance gains using MTL for the metaphor datasets. For the idiom and non-literal language datasets, while also improving using MTL, increases are smaller. We attribute this to the nature of the datasets `inf-verb` and `part-verb`, as both comprise multiple annotated instances of few particular compounds/verbs. Thus, the STL approach already yields good performance, and adding the more unstructured information of the metaphor datasets does not improve the MTL settings.

An important difference can be observed in the specific MTL setups. While MTL-SEQ fails to profit from just one (in-language) auxiliary task, SLUICE improves significantly over STL even in this lower-resource setup. Adding more auxiliary tasks boosts MTL-SEQ, but does not change SLUICE performance significantly. For MTL-SEQ, this indicates that including information from just one auxiliary task skews the main task too much in one direction, due to hard parameter sharing, while including multiple auxiliary tasks appears to better "balance" the architecture. In contrast, SLUICE can choose to only share the information necessary to improve the tasks, and thus already profits heavily from one auxiliary task. While we expected an increase in performance for the small-data task `adj-noun-met` using more data-heavy auxiliary tasks, it is interesting that `tok-met` also yields better results when including `adj-noun-met` as auxiliary. We attribute this to the network successfully learning a common representation.

We investigate the improvements of MTL-all over STL on the `tok-met` task, using the confusion matrices of MTL-SEQ (Table 6) and SLUICE (Table 7). For both networks, the increase in F_1-score mainly arises due to an increase in recall. This advantage, especially the increase in correctly labeled metaphoric instances, is comparable between SLUICE and MTL-SEQ. For example, "covering" in "So who's covering tomorrow?" is wrongly classified as literal by both STL approaches, but correctly labeled as metaphoric by both MTL-all configurations, i.e., with the help of auxiliary tasks. However, most of the newly identified metaphors (i.e., found by MTL-all but not by STL) differ between the approaches. So, in contrast to the first example, "sweet" in "That's a sweet little village" is misclassified by both networks in STL configuration, but correctly labeled as metaphoric only using SLUICE in the MTL-all setting.

Finally, we compare to the state-of-the-art (SOA) on our chosen tasks. We note that such a comparison with SOA is difficult for most tasks due to non-described test splits or usage of cross validation. The latter was too costly in terms of computation time for our tested architectures. Nonetheless, we include reference numbers for an approximate comparison. `adj-noun-met` is the only dataset for which we have the original test split. Here, the original feature based implementation of Tsvetkov et al. (2014) ($F_1 = 0.85$) outperforms our approach (MTL: 0.63) by a large margin. We attribute this to heavy feature-engineering on their part, using supersenses and concreteness information. In contrast, we perform on par with the state-of-the-art on `tok-met` (Do Dinh and Gurevych (2016): $F_1 = 0.56$, our system:

$F_1 = 0.56$). They implement a simple MLP, incorporating also POS tags and concreteness features. Their test set is similarly large as ours. Horbach et al. (2016) report only accuracy (A = 0.86) for their cross validation experiments on `inf-verb`. Our system results are comparable to their approach, albeit in a different setup using a dedicated test set (A = 0.85 for MTL-de). However, as described in Section 2, Horbach et al. (2016) employ a multitude of semantic features, including selectional preferences and topic information. For `part-verb`, we do not reach the results of Köper and Schulte im Walde (2017). They attain $F_1 = 0.88$ using multi-sense embeddings with Multinomial Naive Bayes in a cross validation setting (vs. our $F_1 = 0.81$ on a test set). Since the multi-sense embeddings vastly outperform their own single-sense baseline, it would be interesting for future work to also include this approach into our model.

6 Conclusion

To the best of our knowledge, we are the first to abstract from related non-literal language detection tasks metaphor and idiom classification to a more general model, using multi-task learning. To this end, we presented an evaluation of two different multi-task learning models on four related semantic, non-literal language tasks in English and German: detection of metaphoric tokens, classification of metaphoric adjective-noun constructions, classification of idiomatic use of infinitive-verb compounds, and non-literal particle verbs. We compared both performances to respective single-task learning baselines, and baselines which use merged training data. While results for the latter systems are mixed, the multi-task learning setups improved performance over the baselines in all cases. Especially the metaphor datasets profit substantially from the multi-task setups, and the inclusion of auxiliary tasks—in the case of hard parameter sharing even for out-of-language idiom tasks.

For all but the smallest dataset, soft parameter sharing and learned architecture outperformed hard parameter sharing. However, the latter approach could benefit from more information (using all related tasks as auxiliary tasks), while the former performed best in a mono-lingual setting. In both cases, data from related non-literal language tasks increase classifier performance, which means that a common representation within the network could be established.

Future work could explore more specifically which non-literal language tasks benefit from the inclusion of which auxiliary tasks. Further, it could be investigated how classic features such as the concreteness or supersenses can increase performance when viewed as lower-level semantic auxiliary tasks.

Acknowledgements

This work has been supported by the German Federal Ministry of Education and Research (BMBF) under the promotional reference 01UG1816B (CEDIFOR).

References

Héctor Martínez Alonso and Barbara Plank. 2017. When is multitask learning effective? semantic sequence prediction under varying data conditions. In *Proceedings of EACL 2017*, pages 44–53, Valencia, Spain. Association for Computational Linguistics.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Beata Beigman Klebanov, Chee Wee Leong, Michael Heilman, and Michael Flor. 2014. Different texts, same metaphors: Unigrams and beyond. In *Proceedings of the Second Workshop on Metaphor in NLP*, pages 11–17, Baltimore, MD, USA. Association for Computational Linguistics.

The BNC Consortium. 2007. The British National Corpus, version 3 (BNC XML Edition).

Marcel Bollmann and Anders Søgaard. 2016. Improving historical spelling normalization with bi-directional lstms and multi-task learning. In *Proceedings of COLING 2016*, pages 131–139. The COLING 2016 Organizing Committee.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 41–48, Amherst, MA, USA. Morgan Kaufmann.

Erik-Lân Do Dinh and Iryna Gurevych. 2016. Token-level metaphor detection using neural networks. In *Proceedings of the Fourth Workshop on Metaphor in NLP*, pages 28–33, San Diego, CA, USA. Association for Computational Linguistics.

Dan Fass. 1991. met *: A method for discriminating metonymy and metaphor by computer. *Computational Linguistics*, 1(1):49–90.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of EMNLP 2017*, pages 1923–1933, Copenhagen, Denmark. Association for Computational Linguistics.

Andrea Horbach, Andrea Hensler, Sabine Krome, Jakob Prange, Werner Scholze-Stubenrecht, Diana Steffen, Stefan Thater, Christian Wellner, and Manfred Pinkal. 2016. A corpus of literal and idiomatic uses of german infinitive-verb compounds. In *Proceedings of LREC 2016*, pages 836–841, Portorož, Slovenia. European Language Resources Association.

Miloš Jakubíček, Adam Kilgarriff, Vojtěch Kovář, Pavel Rychlý, and Vít Suchomel. 2013. The tenten corpus family. In *7th International Corpus Linguistics Conference CL*, pages 125–127, Lancaster, UK.

Tobias Kahse. 2017. Multi-task learning for argumentation mining. Master's thesis, TU Darmstadt.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the Tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.

Maximilian Köper and Sabine Schulte im Walde. 2016. Distinguishing literal and non-literal usage of german particle verbs. In *Proceedings of NAACL-HLT 2016*, pages 353–362, San Diego, CA, USA. Association for Computational Linguistics.

Maximilian Köper and Sabine Schulte im Walde. 2017. Applying multi-sense embeddings for german verbs to determine semantic relatedness and to detect non-literal language. In *Proceedings of EACL 2017*, pages 535–542, Valencia, Spain. Association for Computational Linguistics.

George Lakoff and Mark Johnson. 1980. *Metaphors we live by*. University of Chicago Press, Chicago, IL, USA.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *NAACL Workshop on Vector Space Modeling for NLP*, pages 151–159, Denver, CO, US. Association for Computational Linguistics.

Saif Mohammad, Ekaterina Shutova, and Peter Turney. 2016. Metaphor as a medium for emotion: An empirical study. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 23–33, Berlin, Germany. Association for Computational Linguistics.

Hao Peng, Sam Thomson, and Noah A. Smith. 2017. Deep multitask learning for semantic dependency parsing. *arXiv preprint, arXiv:1704.06855*.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017. Sluice networks: Learning what to share between loosely related tasks. *arXiv preprint, arXiv:1705.08142*.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint, arXiv:1706.05098*.

Marc Schulder and Eduard Hovy. 2014. Metaphor detection through term relevance. In *Proceedings of the Second Workshop on Metaphor in NLP*, pages 18–26, Baltimore, MD, USA. Association for Computational Linguistics.

Claudia Schulz, Steffen Eger, and Johannes Daxenberger. 2018. Multi-task learning for argumentation mining in low-resource settings. In *Proceedings of NAACL-HLT 2018*, pages 35–41, New Orleans, LA, USA. Association for Computational Linguistics.

Roland Schäfer and Felix Bildhauer. 2012. Building large corpora from the web using a new efficient tool chain. In *Proceedings of LREC 2012*, pages 486–493, Istanbul, Turkey.

Roland Schäfer. 2015. Processing and querying large web corpora with the cow14 architecture. In Piotr Bański, Hanno Biber, Evelyn Breiteneder, Marc Kupietz, Harald Lüngen, and Andreas Witt, editors, *Proceedings of the 3rd Workshop on Challenges in the Management of Large Corpora*, pages 28–34, Lancaster. IDS.

Ekaterina Shutova. 2015. Design and evaluation of metaphor processing systems. *Computational Linguistics*, 41(4):579–623.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of ACL 2016*, pages 231–235, Berlin, Germany. Association for Computational Linguistics.

Caroline Sporleder and Linlin Li. 2009. Unsupervised recognition of literal and non-literal use of idiomatic expressions. In *Proceedings of EACL 2009*, pages 754–762, Athens, Greece. Association for Computational Linguistics.

Gerard J. Steen, Aletta G. Dorst, J. Berenike Herrmann, Anna Kaal, Tina Krennmayr, and Trijntje Pasma. 2010. *A Method for Linguistic Metaphor Identification. From MIP to MIPVU*. John Benjamins, Amsterdam, Netherlands.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. In *Proceedings of ACL 2014*, pages 248–258, Baltimore, MD, USA. Association for Computational Linguistics.

Yorick Wilks. 1978. Making preferences more active. *Artificial Intelligence*, 11(3):197–223.

Wei Zhang and Judith Gelernter. 2015. Exploring metaphorical senses and word representations for identifying metonyms. *arXiv preprint, arXiv:1508.04515*.

Twitter corpus of Resource-Scarce Languages for Sentiment Analysis and Multilingual Emoji Prediction

Nurendra Choudhary, Rajat Singh, Vijjini Anvesh Rao, Manish Shrivastava
Language Technologies Research Center, Kohli Center on Intelligent Systems,
International Institute of Information Technology, Hyderabad, India
{nurendra.choudhary, rajat.singh, vijjinianvesh.rao}@research.iiit.ac.in
m.shrivastava@iiit.ac.in

Abstract

In this paper, we leverage social media platforms such as twitter for developing corpus across multiple languages. The corpus creation methodology is applicable for resource-scarce languages provided the speakers of that particular language are active users on social media platforms. We present an approach to extract social media microblogs such as tweets (Twitter). In this paper, we create corpus for multilingual sentiment analysis and emoji prediction in Hindi, Bengali and Telugu. Further, we perform and analyze multiple NLP tasks utilizing the corpus to get interesting observations.

1 Introduction

Twitter has become a valuable source of data for various NLP studies such as sentiment analysis, polarity detection and emoji prediction. Although, copious amount of research studies have been conducted on twitter data, a majority of them deal with English tweets. This is due to several factors. First, English dominates in the mix of languages on Twitter. According to (Hong et al., 2011), more than 50% of tweets are in English. Outside of the largest five Twitter languages (given in Figure 1), other languages represent just under 1% of Twitter traffic each. We primarily focus on collecting data for sentiment analysis. Additionally, taking the premise that social media devices like emojis convey sentiment of their respective tweet, we provide a methodology for collecting tweets with emojis for any language irrespective of its resource availability. Going beyond sentiment analysis, we also collect data without any preconditions on presence or absence of emojis in the tweets which could be used to draw interesting social media analytics both within and across languages or linguistic communities.

We discuss, specifically, about resource-poor languages because such discourse is available on resource-rich languages like English and Spanish, whereas resource-poor languages are largely ignored. For such resource-poor, but yet widely spoken, languages (especially in multilingual communities that have their predominant literature in a language other than their native tongue), we observe social media as a good data source. In this paper, we look at Twitter to collect data in Telugu, Hindi and Bengali, predominantly spoken in the Indian subcontinent. In these regions, English is the language of administration and is increasingly becoming the lingua franca. This, also, explains the lack of sizable corpora despite the relatively large number of speakers.

2 Related Work

Sharing of corpora or resources is important for researchers to compare results with each other, pushing the boundaries of the state-of-the-art model. Openly available corpora reduce the efforts of researchers in constructing the same corpus others developed merely for comparison. However, Twitter's terms of service, under which sharing of aggregated resources (tweets) is barred, prove an unnecessary obstacle in this regard. For example, the Edinburgh Twitter corpus (Petrovic et

Proceedings of the 27th International Conference on Computational Linguistics, pages 1570–1577
Santa Fe, New Mexico, USA, August 20-26, 2018.

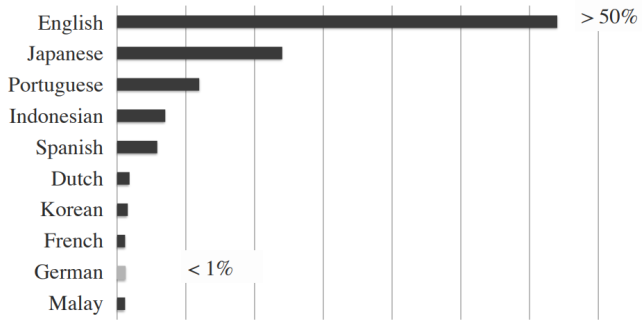


Figure 1: Top ten languages on Twitter. Data from (Hong et al., 2011)

al., 2010) was dissolved as a result of their policies. However, researchers found certain routes to progress even with such issues, following are some examples:

1. Distribution of only lists of tweet IDs, e.g.; in this TREC 2011 microblog task [1]

2. Distribution of the derivatives of data, e.g.; sharing n-gram counts, instead of the original tweets (Herdağdelen, 2013).

However, the second method loses important information about the data such as word order and limits the analyses of experiments. For example, in (Herdağdelen, 2013), the analysis in section 4 would require more than just n-gram counts since the model deals with tweets per day of the week. Hence, primary metadata regarding the tweets is required.

Twitter provides a streaming API under a public "gardenhouse" setting to build corpora. For example, the Edinburgh corpus (Petrovic et al., 2010), the Tweets2011 corpus from the TREC microblog shared task, and the Rovereto n-gram corpus (Herdağdelen, 2013). In this method, a considerably small fraction of tweets, over a time period are collected. However, the way in which Twitter makes these set of tweets is unclear, inducing a possible bias. Even after these restrictions, the corpora that exist are predominantly in English, hence extracting twitter for resource-scarce languages is necessary.

In their attempt to construct language specific corpora, some approaches choose certain sites to crawl based on results from using medium frequency terms of the language as search terms (Baroni and Bernardini, 2004; Schäfer and Bildhauer, 2012). Following a similar approach, we employ the top most frequent words of the required language as keywords. (Rehbein et al., 2013) proposes an efficient approach to collect German tweets using geolocation features with language filter. However, the data encounters certain biases:

1. Only a fraction of users have GPS access while tweeting. These tweets are included whereas other kinds of users' tweets are completely ignored. Hence, the collected data is not a representative of the larger sample.

2. Tweets that do not originate from devices with geolocation features like smartphones are also completely excluded. Curated content or in-depth political discussions are not necessarily tweeted from a smartphone.

Similarly, (Scheffler, 2014) have created a German twitter corpus using Twitter APIs.

(Cui et al., 2011) analyze emotion tokens, including emotion symbols (e.g. emoticons) for sentiment analysis and emotion analysis of Twitter snapshots. (Barbieri et al., 2017) established a sentiment analysis architecture for Twitter and released a data set for the same in English. They released the data set of roughly 500K tweets for English with emojis as labels. On a

[1] `http://trec.nist.gov/data/tweets`

similar track, our work focuses on resource-poor languages. Here, the architecture addresses sentiment analysis as a supervised multi class classification problem where each English tweet is "annotated" with its emoji. Hence their data consists of tweets with exactly a single emoji which later is treated as the sentiment label of the tweet, while the tweet itself is striped of the emoji.

We continue their task and collect data on the same lines i.e. we collect Tweets with exactly one emoji in the given aforementioned Languages and more.

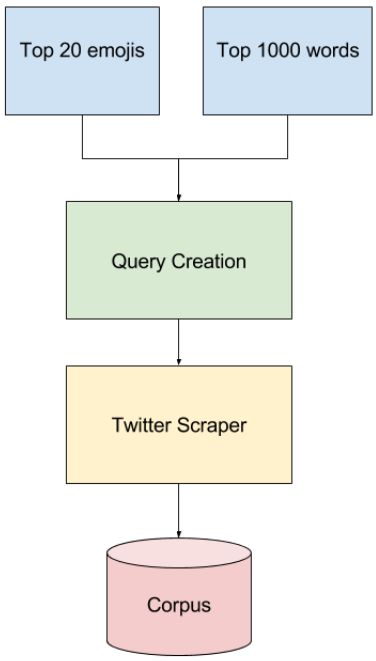


Figure 2: Corpus creation pipeline.

3 Building the corpus

While the previous work for gathering the tweets has been restricted to using twitter search APIs [2], the free usage is limited to only a history of 7 days of tweets. Therefore, we use twitter scraper for no limits with respect to temporal history, as it scraps the web search results of the given query and returns tweets from them. We attempt to scrap tweets using prominent keywords of particular language. Our methodology which is both efficient in time and memory, can easily be extended to any other language irrespective of Resource availability. Following is the description of our methodology to collect tweets with emojis for emoji prediction tasks (Barbieri et al., 2017) which can be extended to collecting just the tweets irrespective of emojis for various other NLP tasks:

3.1 Keywords Extraction

First, we get 1000 most frequent words of the target language for which we want to create the corpus. There are various websites which showcase most frequent used words in a particular language. We use 1000mostcommonwords.com [3] for this project. The most common words usually contain stop words and commonly occuring verbs of the language.

[2] `https://developer.twitter.com/en/docs/tweets/search/overview`
[3] `http://1000mostcommonwords.com/`

Language	Without Emojis	With Emojis as classes	Total
Hindi	194,063	202,415	396,478
Telugu	161,851	16,990	178,841
Bengali	78,308	59,528	137,836
Total	434,222	278,933	**713,155**

Table 1: Total Number of Collected Tweets

3.2 Query creation

To create a search query for the scraper, in preparing corpus for emoji prediction task. we use an emoji and one of the most frequent words of the language for eg.< ❤️,की >. For creating a general twitter corpus irrespective of emojis, we just drop the emoji from the search query.

3.3 Scraper

We then scrap tweets using the Python package *twitterscraper* (Taspinar, 2016 2017) [4]. Our corpus collection pipeline is shown in Figure 2. The scraper returns tweets for the given query based on the condition that the tweet contains the query keyword. We also keep two hashmaps, for tweet id and tweet text's initial 30 characters. These hashmaps are used to remove repetition in incoming tweet results. Every time a tweet is processed, its availability in previously appeared tweets that are already present in corpus is checked, this ensures data is enough diverse and we avoid duplicate entries. Spam or automated tweets from bots, or tweets feed on a celebrity's birthday will be overflowing with very similar tweets like: "जन्मदिन पर बहुत बहुत शुभकामनाएँ बच्चन जी", (Huge Wishes on your Birthday Bacchan sir) and "जन्मदिन पर बहुत शुभकामनाएं अमिताभ जी (Huge Wishes on your Birthday Amitabh sir)" etc. will be avoided in this method.
Collecting such tweets as separate entries is only going to deplete the quality of our data as the later tweet doesn't give much information which the first won't. Any kind of learning on such a data, is only going to reinforce a specific data point rather than giving diversity in label or dataset.
We, also, store a preprocessed version of the collected data by removing mentions and urls because they do not carry significant information for some major tasks such as sentiment analysis and emoji prediction.

4 Corpus Analysis

In total we collected 713,155 tweets in three languages, 396K in Hindi, 178K in Telugu and 137K in Bengali (Table 1) out of which 279K tweets are with emojis as labels and remaining we collected irrespective of emojis. In both the cases we maintain two distinct versions of the data: Cleaned and Uncleaned. Where Cleaning is done with the emoji prediction task in consideration, the changes made include:

1. Trailing symbols removed, for example "????" becomes "?" or "...." becomes ".".

2. Words and symbols which are foreign to the language of the tweet are removed, for example "मुर्गी के अंडे और पा जी के फंडे रोके नही रुकते Just #ViruPanti #INDvAUS" will be cleaned to "मुर्गी के अंडे और पा जी के फंडे रोके नही रुकते"

However, As we maintain an uncleaned raw version too, depending on the application, data can be used accordingly.

The most frequent emojis in Telugu, Hindi and Bengali can be found in Table 2 , Table 3, Table 4 respectively.

[4] https://pypi.python.org/pypi/twitterscraper/

❤️	😊	😍	😏	😐	😠	😢	🤔	😈
9.54	9.51	9.42	8.58	7.72	7.50	7.22	6.78	6.73
😭	💕	😉	😁	💙	😘	😶	😜	💜
4.92	4.32	3.82	2.90	2.81	2.56	2.33	1.87	1.46

Table 2: Percentage of emojis in Hindi tweets

😊	❤️	😍	😐	😠	😏	😘	😢	🤔
15.65	13.99	10.46	10.22	8.66	8.59	7.44	5.80	5.70
😶	😭	😈	😁	😜	😉	💕	💙	💜
4.64	3.66	1.32	1.08	1.04	0.67	0.52	0.49	0.07

Table 3: Percentage of emojis in Bengali tweets

😊	😠	😍	❤️	😐	🤔	😏	😢	😭
22.67	16.55	13.38	10.77	6.68	5.67	5.05	4.19	3.91
😈	😜	😁	😉	💕	😶	😘	💜	💙
3.01	2.29	1.57	1.36	0.94	0.84	0.54	0.31	0.25

Table 4: Percentage of emojis in Telugu tweets

4.1 Relationship between emojis and Sentiments

A random sampling of 500 tweets of each language, from the extracted tweets with emojis, are manually annotated into three sentiment classes: Positive, Negative, and Neutral to study a relationship between emojis and annotated sentiments. Results shown in Table 5.

The tabulated statistics suggest that sentiments and emojis in most of the cases share a strong correlation, which further reinforces the idea of using emoji as a sentiment label. Furthermore, we note from Tables 5,6 and 7 that even across languages the annotation of different emoji tweets into the three sentiment classes is similar, indicating that the sentiment or emoji's semantic value is preserved across languages till a large extent.

5 Applications

The entire corpus has been hosted on the link given in the footnote [5]. Potential use of the corpus can be really varied:

- `Sentiment Analysis:` As discussed in related work, we built a Sentiment Analysis tool. The emoji data be used as label for Sentiment Analysis, where we could go one step further and predict emoji for the emoji-less data we collected.

- `Enrich Resource-scarce Language:` Siamese network based approaches (Choudhary et al., 2018b; Choudhary et al., 2018c; Choudhary et al., 2018a) are capable of utilizing these emojis of resource-poor language concurrently with emojis of resource-rich languages to enhance the overall performance of both resource-poor and resource-rich languages.

- `Social Media Analytics:` Beyond Sentiment Analysis too, We could learn a lot, especially in Topic modeling across languages for example, what domain do most Telugu tweets belong to. Are Telugu tweets more about Movies than say Bengali tweets which might be into Politics. Social Media is always heavy with Demographic trends which can be analyzed.

[5] `https://figshare.com/articles/Twitter_corpus_of_Resource-Scarce_Languages_for_Sentiment_Analysis_and_Multilingual_Emoji_Prediction/6477782`

Emoji	Positive	Neutral	Negative	Emoji	Positive	Neutral	Negative
😊	98	2	0	😶	10	73	7
❤️	100	0	0	😭	3	14	83
😍	99	1	0	😡	0	3	97
😐	3	84	13	😁	93	5	2
😠	0	2	98	😜	76	21	3
😒	9	79	12	😉	67	25	8
😘	96	4	0	💕	99	1	0
😢	9	3	87	💙	87	13	0
🤔	3	90	7	💜	82	8	10

Table 5: Distribution of annotated sentiment classes of Hindi tweets with emojis in percentage, symbolized by their emojis here

Emoji	Positive	Neutral	Negative	Emoji	Positive	Neutral	Negative
😊	90	9	1	😶	7	78	5
❤️	99	0	1	😭	9	80	11
😍	95	3	2	😡	0	7	93
😐	2	81	17	😁	90	7	3
😠	0	3	97	😜	71	18	11
😒	5	75	20	😉	71	27	2
😘	93	7	0	💕	100	0	0
😢	11	5	85	💙	91	9	0
🤔	5	89	6	💜	79	11	10

Table 6: Distribution of annotated sentiment classes of Bengali tweets with emojis in percentage, symbolized by their emojis here

Emoji	Positive	Neutral	Negative	Emoji	Positive	Neutral	Negative
😊	90	9	1	😶	5	74	11
❤️	100	0	0	😭	3	9	88
😍	100	0	0	😡	1	5	94
😐	7	83	10	😁	89	4	7
😠	0	0	100	😜	69	18	13
😒	11	82	7	😉	71	22	7
😘	91	5	4	💕	98	1	1
😢	7	3	90	💙	77	21	2
🤔	2	98	0	💜	91	1	8

Table 7: Distribution of annotated sentiment classes of Telugu tweets with emojis in percentage, symbolized by their emojis here

6 Limitations and Future Work

Certain issues with our data which require further investigation would include

- Code Mixed Data: Because we only look for just one word to decide the language of the tweet, It's possible we may take a lot of code mixed data as compared to a corpus which is exclusively in one language, for example
  - "तू कल भी दिल में थी... और आज भी है...बस कल तक favorite list मे थी...आज block list मे है... ", (*You were in my heart till yesterday and so today too, but yesterday you were in my favorite list and today in block list*)
  - "#Sun #Pagli.... अगर तू #doll है तो में #Dollar... अगर तू #Brand है तो में #Branded हु ", (*Listen, O mad woman, If you are a doll, then I am Dollar, if you are a brand, then I am branded*)

  Such tweets may also get included in our data, even though the amount of Hindi information they have is minimal

- In any data Noise is unfavorable, Noise in such social media platforms could be
  - "सिर्फ $attitude होने से कुछ नही होता #smile ऐसी दो की हर एक #लोग बोल पड़े ", (*only having $attitude won't do anything, give a #smile which will make #people talk*)
  - "####french_fries....!!! : तले भुने आलू सदा सुखी रहो ", (*####french_fries....!!!* .. semantically non compliant)

  Such a noise will corrupt any learning to be performed on this data.

- Twitter like any other social media is a live, thriving platform, taking a snapshot at one time and presuming it to work forever is not a sound judgment. Which is why such data needs to be updated regularly.

7 Conclusion

In this paper, we present a twitter corpora for Telugu, Bengali and Hindi along with a methodology which is scalable across languages. A data which could be used for tasks like Sentiment Analysis,often done on Resource heavy languages only. Along with it we also release a regular data which can be used for tasks beyond. In addition to this, we further present that such a data provides more information than already present in the field, by providing its use for some applications. We hope this corpus can serve as a basis for more work to be done upon such Resource Scarce languages.

References

Francesco Barbieri, Miguel Ballesteros, and Horacio Saggion. 2017. Are emojis predictable? *arXiv preprint arXiv:1702.07285*.

Marco Baroni and Silvia Bernardini. 2004. Bootcat: Bootstrapping corpora and terms from the web. In *LREC*, page 1313.

Nurendra Choudhary, Rajat Singh, Ishita Bindlish, and Manish Shrivastava. 2018a. Contrastive learning of emoji-based representations for resource-poor languages. *arXiv preprint arXiv:1804.01855*.

Nurendra Choudhary, Rajat Singh, Ishita Bindlish, and Manish Shrivastava. 2018b. Emotions are universal: Learning sentiment based representations of resource-poor languages using siamese networks. *arXiv preprint arXiv:1804.00805*.

Nurendra Choudhary, Rajat Singh, Ishita Bindlish, and Manish Shrivastava. 2018c. Sentiment analysis of code-mixed languages leveraging resource rich languages. *arXiv preprint arXiv:1804.00806*.

Anqi Cui, Min Zhang, Yiqun Liu, and Shaoping Ma. 2011. Emotion tokens: Bridging the gap among multilingual twitter sentiment analysis. *Information retrieval technology*, pages 238–249.

Amaç Herdağdelen. 2013. Twitter n-gram corpus with demographic metadata. *Language resources and evaluation*, 47(4):1127–1147.

Lichan Hong, Gregorio Convertino, and Ed H Chi. 2011. Language matters in twitter: A large scale study. In *ICWSM*.

Sasa Petrovic, Miles Osborne, and Victor Lavrenko. 2010. The edinburgh twitter corpus. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media*, pages 25–26.

Ines Rehbein, Sören Schalowski, Nadja Reinhold, and Emiel Visser. 2013. Uhm... uh.. filled pauses in computer-mediated communication. In *Talk presented at the Workshop on" Modelling Non-Standardized Writing" at the 35th Annual Conference of the German Linguistic Society (DGfS)*.

Roland Schäfer and Felix Bildhauer. 2012. Building large corpora from the web using a new efficient tool chain. In *LREC*, pages 486–493.

Tatjana Scheffler. 2014. A german twitter snapshot. In *LREC*, pages 2284–2289.

Ahmet Taspinar. 2016–2017. Python twitterscraper.

Towards identifying the optimal datasize for lexically-based Bayesian inference of linguistic phylogenies

Taraka Rama[♠] **Søren Wichmann**[♢,♡]
[♠]Department of Informatics, University of Oslo, Norway
[♢]Leiden University Centre for Linguistics, Leiden University, Netherlands
[♡]Laboratory of Quantitative Linguistics, Kazan Federal University, Russia
tarakark@ifi.uio.no, wichmannsoeren@gmail.com

Abstract

Bayesian linguistic phylogenies are standardly based on cognate matrices for words referring to a fix set of meanings—typically around 100-200. To this day there has not been any empirical investigation into which datasize is optimal. Here we determine, across a set of language families, the optimal number of meanings required for the best performance in Bayesian phylogenetic inference. We rank meanings by stability, infer phylogenetic trees using first the most stable meaning, then the two most stable meanings, and so on, computing the quartet distance of the resulting tree to the tree proposed by language family experts at each step of datasize increase. When a gold standard tree is not available we propose to instead compute the quartet distance between the tree based on the n-most stable meaning and the one based on the $n+1$-most stable meanings, increasing n from 1 to $N-1$, where N is the total number of meanings. The assumption here is that the value of n for which the quartet distance begins to stabilize is also the value at which the quality of the tree ceases to improve. We show that this assumption is borne out. The results of the two methods vary across families, and the optimal number of meanings appears to correlate with the number of languages under consideration.

1 Introduction

Phylogenetic methods—both distance-based and character-based methods (specifically Bayesian phylogenetic inference)—from computational biology are being widely used in historical linguistics and typology, for phylogenetic inference, for triangulating homelands of proto-languages (Wichmann et al., 2010b), for investigating rates of lexical change (Greenhill et al., 2017), dating and spread of language families (Gray et al., 2009; Holman et al., 2011; Bouckaert et al., 2012), reconstruction of proto-languages (Bouchard-Côté et al., 2013), and for investigating typological universals (Dunn et al., 2011).

Most of the above studies require cognate matrices based on wordlists annotated by experts for cognacy. Each column in such a matrix represents a *cognate class*, with 1's and 0's indicating whether or not a form in the shared ancestral language has a reflex in the descendant languages with a given predefined meaning pertaining to the fixed list of meanings. Such cognate matrices are fed to Bayesian phylogenetic software such as MrBayes (Ronquist et al., 2012b), BEAST (Drummond et al., 2012) or BayesPhylogenies (Pagel and Meade, 2004). An alternative approach is to apply distance-based methods (Wichmann et al., 2010a; Jäger, 2013; Rama and Borin, 2015). Such methods have the advantage that they do not necessarily require labor-intensive cognate identification or the assumption which goes with cognate identification, namely that the languages analyzed are actually related. Instead, an aggregate phonetic distance can be computed. As in biology, the most popular algorithm for inferring phylogenies from linguistic distance data has been Neighbor-Joining (Saitou and Nei, 1987). All the above methods have typically been applied using variants of Swadesh meaning lists (Swadesh, 1952), sometimes adapted for the language family in focus.

Some information is already available concerning the optimal word list size for distance-based linguistic phylogenetic inference. The pertinent literature is reviewed in the next section. As for

Proceedings of the 27th International Conference on Computational Linguistics, pages 1578–1590
Santa Fe, New Mexico, USA, August 20-26, 2018.

character-based methods, such as Bayesian ones, there is no study investigating the optimal meaning list length when inferring linguistic phylogenies. One reason for this could be the computational costs involved in inferring millions of trees for each of the cumulative meaning lists. To the best of our knowledge the present paper represents the first known computationally intensive study in this direction. We infer trees for eleven language families (a total of 313 languages) using Bayesian phylogenetic inference procedure and apply two methods to determine whether the quality of trees stops changing as the size of the word list increases. The first method requires expert phylogenetic trees for validation. It is similar in nature to one applied by Holman et al. (2008), described in the next section. The second method is entirely novel and does not require expert trees. Both methods are applied to the five language families in our sample for which reasonably reliable expert trees were available, and we show that the results are similar. Thus support is found for also including results for the second method, which does not require expert trees. Thereby results could be added the remaining six language families and more reliable statistics was obtained.

In this paper, we ask the following research questions:

- Is there any optimal list length after which there is no improvement in the trees inferred from Bayesian phylogenetic inference procedure?
- Is there any procedure to determine the optimal list length when there is sparse historical linguistic research regarding the genetic tree for a language family?

The structure of the paper is as follows. We discuss some work related to determining ranks of item stability in section 2. The problem and our approach are described in section 3. Section 4 presents the datasets and outlines how Bayesian phylogenetic inference is applied. Section 5 presents and discusses the implications of our results and is followed by conclusion in section 6.

2 Related work

Holman et al. (2008) used 245 100-item word lists from languages of 69 families previously compiled for another paper (Brown et al., 2008) and then devised a stability measure quantifying the resistance of a meaning to undergo lexical replacement. The authors ranked the 100 meanings by the order of stability and then created lists consisting of the 5, 6 ... 100 most stable meanings. They then proceeded to infer linguistic trees from the meaning lists, computing the fit of the trees to expert trees. The authors found that the quality of the trees ceased to improve when using more than the 40 most stable meanings (and subsequently 40-item lists were collected for as many languages of the world as possible, ultimate leading to the so-called ASJP database of Wichmann et al. (2018), covering more than two thirds of the world's languages). Evaluation studies (Wichmann et al., 2010a; Pompei et al., 2011; Jäger, 2013; Rama and Borin, 2015) have suggested that the quality of the phylogenetic trees inferred from 40-item word lists is quite high when compared to gold standard phylogenetic trees.

Another investigation of the effect of the number of meanings on the quality of a phylogeny is represented by Petroni and Serva (2010). The authors rank 200-item meaning lists of 50 Austronesian and 50 Indo-European languages through a stability measure derived from the Levenshtein distance between synonyms across word lists. They then infer a distance-based tree for each cumulative meaning list and compare the tree with the Levenshtein distance tree inferred from the full dataset using the Robinson-Foulds distance (Robinson and Foulds, 1979). The authors observe that the Robinson-Foulds distance drops as the length of meaning list changes from 1 to 200. The authors do not compare the inferred trees from subsets to the gold standard family tree. Neither do the authors use the cognate information to infer trees. Instead, the distance between two languages is based on aggregate Levenshtein distance computed between synonyms in the word lists.

In another work, Rama and Borin (2013) computed phoneme n-gram entropy for synonyms belonging the 100-item meaning lists from 245 languages used by Holman et al. (2008). Then they rank the meanings in order of increasing entropy. The meaning with the lowest entropy is the most stable item.

The authors find that the entropy-ordered meaning lists' ranks correlate highly with the order inferred by Holman et al. (2008).

In a related work, which uses Bayesian inference, Pagel and Meade (2006) employ a Bayesian phylogenetic inference model where each meaning is treated as a separate partition within the complete dataset (cf. table 2). Each partition or meaning has a separate weight that contributes to the likelihood calculation. The weight for each meaning is inferred through an MCMC procedure (cf. section 4.2 for details on Bayesian phylogenetic inference procedure). Then the authors correlated the mean weight for each meaning estimated from the posterior samples with the number of cognate classes in the meaning for 87 Indo-European languages and 95 Bantu languages. The authors find a positive correlation ($r > 0.6$) between the mean weights and the number of cognate classes for both language families.

In a follow-up paper, the same authors Pagel et al. (2007) interpreted the inferred weight from posterior samples as an index for lexical replacement and found a negative correlation with corpus-based estimates of the frequency of words representing the different meanings. In essence, meanings with low frequency undergo lexical replacement more easily than meanings with high frequency. In the present paper, we also apply Bayesian inference procedure to cumulative datasets ranked by the diachronic lexical stability.

3 Problem and approach

To investigate the influence of the datasize on the quality of a Bayesian phylogeny, word lists of different sizes must be produced. In doing so, it is necessary to take the differential stabilities of the words corresponding to different meanings into account, because more stable items are expected to lead to better phylogenies and the influence of stability can thus mask the influence of the number of items. It is well known that the words for some meanings tend to be more quickly replaced than words for some other meanings.

Different methods and datasets discussed in section 2 lead to somewhat different assessments of relative stabilities, but not to such a degree that any major controversies have arisen. The measure of stability used in the present paper is based on the simple and quite old idea idea (cf. Thomas (1960), Kroeber (1963), and Oswalt (1970)) that words for more stable items can be identified by their greater tendency to yield cognates within groups of closely related languages than words for less stable items. Here we use the number of cognate classes as a measure of stability: the smaller the number of cognate classes found for a given meaning is, the fewer word substitutions there would have been, and thus the greater the stability. Having ranked the meanings in this way for a given language family we can go on to define word lists of different sizes, n, each consisting of words for the n-most stable meanings. As a note, in case of ties—two or more equally stable meanings—these meanings will be arbitrarily ranked.

We propose two tests for comparing the quality of trees based on two meaning lists differing by the number of items they contain. In the first test, for each of the two meaning lists, we report the average of the generalized quartet distances (GQD) between the posterior tree distribution and a gold standard tree. We hypothesize that after reaching a certain size, n, of meaning list, the difference between the average GQD distance between the list having n meanings and the one having $n + 1$ meanings will stabilize. This first test is designed for cases where the gold standard family tree is reasonably well established and uncontroversial. This is not always the case, since the amount of historical linguistic work dedicated to language families throughout the world varies greatly (Campbell and Poser, 2008). For cases where expert classifications are lacking or not very trustworthy we propose a second test that does not depend on the gold standard tree. This test consists in computing the GQD's between trees inferred from meaning lists of size n and $n + 1$, where n ranges from 1 to the next to largest number of meanings covered in the dataset used, and then plot the average quartet distance for each such pair. Our hypothesis is that after a point, the difference of average quartet distances will begin to stabilize.

4 Materials and Methods

In this section, we describe the datasets, tree inference procedure, and the evaluation procedure.

Family	Languages	Meanings	Sources
Austronesian	96	210	Gray et al. (2009)
Mayan	30	100	Wichmann and Holman (2013)
Mixe-Zoque	10	100	Cysouw et al. (2006)
Indo-European	52	208	Dunn (2012)
Uto-Aztecan	31	100	Miller (1984)
Afro-Asiatic	25	100	Militarev (2000)
Bai	9	110	Wang (2006)
Chinese	18	180	Dàxué (1964)
Mon-Khmer	16	100	Peiros (1998)
Ob-Ugrian	21	110	Zhivlov (2011)
Tujia	5	109	Starostin (2013)

Table 1: Number of languages, meanings, and sources for each dataset.

4.1 Datasets

The experiments involving expert trees are performed on datasets belonging to five different language families: Austronesian, Mayan, Mixe-Zoque, Indo-European, and Uto-Aztecan. In addition to this dataset we draw on another six language families or subfamilies: Afro-Asiatic, Bai, Chinese, Mon-Khmer, Ob-Ugrian, and Tujia. The number of languages, number of meanings, and sources for each dataset is given in Table 1. In case of Austronesian and Indo-European, we use a subset of languages provided in Jäger et al. (2017).

Each of the above datasets is a multilingual word list with expert annotated cognate judgments concerning the words representing each meaning. An excerpt of a multilingual word list from Indo-European is shown in table 2a. The meanings are ordered by the number of cognate classes. We also show the corresponding binary matrix in table 2b. The binary matrix has 2 columns and 3 columns for the meanings ALL and AND since there are, respectively, two and three cognate classes for the meanings ALL and AND. In contrast, the meaning NAIL, for instance, has only one cognate class since all the daughter languages show words that can be traced back to the Proto-Indo-European stage. The meanings are accumulated at each rank and converted to a binary matrix of dimensions $L \times C$ where L is the number of languages and C is the number of cognate sets in the list of meanings accumulated. We explain the Bayesian phylogenetic inference and the tree evaluation algorithms below.

Language	ALL	AND	...
English	ɔːl^1	aɛnd^1	...
German	alə^1	ʊnt^1	...
French	tu^2	e^2	...
Spanish	toðo^2	i^2	...
Swedish	ˈalːa^1	ɔkɾ^3	...

(a) Forms and cognate classes

Language	ALL		AND		
English	1	0	1	0	0
German	1	0	1	0	0
French	0	1	0	1	0
Spanish	0	1	0	1	0
Swedish	1	0	0	0	1

(b) Binary Matrix

Table 2: Excerpt from meaning list showing cognate classes (table 2a) and the binary cognate matrix (table 2b) for ALL and AND in Germanic and Romance subfamilies of Indo-European. The superscript indicates words that are cognate.

The meanings ranked by the number of cognate classes in each of the five families for which we make use of expert trees are provided below. A number in a square bracket introduces meanings for which there are as many cognate classes as the number indicates. The lists have some interest in their own right since they contribute to the available empirical data on word stabilities across different language families. For reasons of page limitations we do not include the remaining six families.

Austronesian : [1] fifty, [2] twenty, six, [3] stickwood, seven, louse, five, [4] to yawn, breast, ten, eye, [5] two, that, to die/be dead, three, fruit, eight, nine, [6] one thousand, father, [7] liver, [8] we, to vomit, they, new, four, i, to pound/beat, [9] thou, who, one hundred, [10] name, to dig, to choose, stone, he/she, branch, when, to come, above, [11] thin, what, wife, mother, to hide, to grow, one, to buy, [12] dust, worm/earthworm, root, star, salt, toeat, [13] person/human being, in/inside, this, to drink, leaf, woman/female, feather, needle, to dream, [14] fog, head, where, lake, bird, rain, ear, thick, to hear, if, to fear, bad/evil, road/path, [15] fire, to sleep, right, blood, hair, tongue, to climb, sky, moon, to chew, [16] to kill, you, to shoot, to burn, man/male, to throw, child, and, to steal, thatch/roof, house, ash, [17] shoulder, how, night, tail, to cry, bone, to blow, nose, spider, other, shy/ashamed, tooth, to open/uncover, year, [18] flower, to swell, to think, to say, water, below, sea, to bite, to hunt, to count, to swim, to walk, to plant, [19] to suck, fish, old, back, egg, rope, dry, to stand, narrow, to flow, heavy, day, all, [20] to see, far, mosquito, to cook, painful/sick, wide, black, dull/blunt, snake, to spit, fat/grease, skin, green, [21] rat, sand, hand, intestines, to sew, at, to live/be alive, to turn, cold, white, to fall, to split, [22] correct/true, warm, wet, sharp, to squeeze, to hold, smoke, leg/foot, grass, [23] to sniff/smell, to lie down, to scratch, left, near, woods/forest, to stab/pierce, earth/soil, husband, [24] short, no/not, to hit, to cut/hack, wing, mouth, dog, [25] to work, lightning, meat/flesh, wind, small, to laugh, to fly, cloud, [26] to tie up/fasten, red, yellow, [27] big, neck, belly, to breathe, to sit, [28] rotten, long, [29] to know/be knowledgeable, good, [30] dirty, thunder.

Mayan: [1] green, white, tail, two, water, [2] tongue, path, die, yellow, bone, rain, tree, red, [3] name, tooth, mouth, mountain, hand, one, louse, sleep, foot, dry, [4] person, fire, ash, [5] claw, flesh, heart, blood, dog, sun, [6] neck, black, ear, fish, moon, I, [7] smoke, walk, see, earth, seed, breasts, man, skin, drink, [8] we, grease, kill, night, nose, cloud, leaf, hair, [9] head, star, stone, say, horn, woman, come, give, [10] sand, hot, cold, hear, full, [11] eat, egg, you, feather, new, bird, fly, stand, [12] bark, big, bite, eye, [13] not, [14] liver, this, burn, belly, good, swim, long, [15] know, knee, [16] round, who, [17] many, [18] all, root, [19] that, [20] lie, small, [21] what, [24] sit.

Mixe-Zoque: [1] ear, tooth, green, see, star, yellow, earth, stone, what, blood, bone, rain, white, leaf, thou, hair, drink, road/path, dry, water, red, [2] name, black, eat, head, person, bark, ashes, tongue, feather, walk, mountain, die, night, liver, new, moon, I, hand, tree, louse, hear, one, come, sun, tail, skin, good, no, clawnail, stand, [3] we, neck, sand, fire, smoke, mouth, kill, horn, knee, seed, round, give, eye, who, two, meat, root, [4] egg, fish, big, know, heart, say, bird, this, nose, fly, woman, sleep, man, [5] warm, sit, breast, belly, cloud, long, [6] cold, bite, that, dog, full, small, swim, many, [7] fat, foot, [8] lie, all, burn.

Indo-European: [1] name, four, three, fingernail, five, two, [2] we, what, I, who, tongue, when, ant, [3] how, new, egg, sun, tooth, one, [4] where, give, heart, thou, mother, full, drink, [5] sit, die, night, ear, not, star, horn, salt, day, [6] you, spit, root, snow, father, knee, [7] know, fish, this, seed, flower, eye, sew, water, long, foot, [8] warm, there, nose, other, dry, hear, sleep, [9] fly, sand, hand, bone, wind, leaf, stand, he, feather, right side, [10] eat, ice, far, old, that, liver, louse, sing, swim, with, smoke, wing, sea, moon, live, suck, [11] head, they, fire, all, rain, small, flow, blow, some, wash, dog, skin, grass, here, [12] lake, freeze, blood, bird, see, hold, red, cold, bark, laugh, green, and, breathe, lie, snake, year, meat, [13] rub, dig, sharp, tie, man, person, heavy, fall, sky, yellow, worm, cloud, straight, [14] burn, thin, short, stone, round, river, black, mouth, animal, think, fruit, in, come, [15] few, bite, neck, leg, breast, tree, play, left, guts, earth, woman, white, [16] fog, count, big, rotten, float, dust, hair, wide, thick, narrow, near, say, mountain, ashes, [17] wet, smooth, fear, pull, smell, cut, vomit, [18] right, turn, at, split, many, [19] road, belly, hunt, good, [20] fat, back, kill, hit, wipe, if, [21] stab, scratch, walk, dull, wife, [24] tail, swell, throw, woods, stick, [25] rope, fight, [26] child, [28] squeeze, husband, [29] dirty, [32] bad, push, [37] because.

Uto-Aztecan: [1] tooth, bite, [2] path, liver, water, [3] ear, moon, breasts, horn, sun, tail, eye, heavy, two, dry, [4] eat, tongue, smoke, stone, salt, nose, drink, [5] name, neck, die, knee, blood, hand, bone, [6] star, night, earth, one, sleep, clawnail, stand, [7] fire, mouth, kill, sky, dog, louse, hear, wind, root, [8] fish, grease, heart, no, cold, mountain, seed, [9] person, snow, see, bellybutton, give, ash, [10] head, know, sit, snake, egg, belly, bark, hair, leaf, meat, long, [11] black, foot, sand, hot, big, tree, white, [12]

lie, feather, vomit, sing, man, many, red, [13] skin, walk, rain, [14] burn, green, yellow, new, fly, [15] year, cloud, small, [16] bird, [17] woman, good, [18] old, come, [19] rope.

4.2 Phylogenetic inference

The Bayesian phylogenetic inference (Ronquist and Huelsenbeck, 2003) is based on the following Bayes rule:

$$f(\tau, \mathbf{T}, \theta|X) = \frac{f(X|\tau, \mathbf{T}, \theta) f(\tau, \mathbf{T}, \theta)}{f(X)}, \tag{1}$$

where X is the binary data matrix of dimensions $L \times C$, τ is the tree topology, θ is the substitution model parameters, $\mathbf{T}$ is the branch length vector of the tree. The posterior distribution $f(\tau, \mathbf{T}, \theta|X)$ is difficult to calculate analytically since one has to sum over all the possible rooted topologies ($\frac{(2L-3)!}{2^{L-2}(L-2)!}$). Therefore, Markov Chain Monte Carlo (MCMC) methods are used to calculate the posterior probability of the parameters $\tau, \mathbf{T}, \theta$. The Metropolis-Hastings algorithm (a MCMC algorithm) is used to sample the parameters from the posterior distribution. This algorithm constructs a Markov chain by proposing change to one or a block of parameters and then accepts the proposal with the following probability:

$$r = \frac{f(X|\tau, t^*, \theta)}{f(X|\tau, t, \theta)} \frac{f(t^*)}{f(t)} \frac{q(t|t^*)}{q(t^*|t)} \tag{2}$$

We assume that the parameters $\tau, \mathbf{T}, \theta$ are independent of each other. Therefore, the joint prior probability $f(\tau, \mathbf{T}, \theta)$ can be decomposed into $f(\tau) f(\mathbf{T}) f(\theta)$. In equation 2, the inference program proposes a change to a branch length t to generate a new branch length t^*. Subsequently, the likelihood of the data to the new parameters is computed using the pruning algorithm (Felsenstein, 2004), which is a special case of the Sum-Product algorithm (Jordan, 2004).

All our Bayesian analyses use binary datasets with states 0 and 1 (cf. table 2b). We employ the Generalized Time Reversible Model (Yang, 2014) for computing the transition probabilities between individual states. The rate variation across sites is modeled using a four category discrete Γ distribution (Yang, 1994). We follow Lewis (2001) and Felsenstein (1992) in correcting the likelihood calculation for ascertainment bias resulting from unobserved 0 patterns. We used a uniform tree prior (Ronquist et al., 2012a) in all our analyses which constructs a rooted tree and draws internal node heights from a uniform distribution. All our experiments are performed using MrBayes 3.2.6 (Zhang et al., 2015).

A Markov chain is initiated at a random starting point which consists of random tree, random branch lengths, and random substitution model parameters. In our experiments, we run two independent chains and sample every thousandth state in the chain and write it to the file. The runs are stopped until the average standard deviation of split (unique bipartition in a tree) frequencies does not differ beyond a threshold of 0.01. The size of the dataset influences the number of states needed for convergence. The tree space required to explore grows factorially with the number of languages whereas the cost of likelihood calculation increases linearly with the number of languages. In the case of Austronesian, we run the analysis for fifty million states; in the case of Indo-European and Uto-Aztecan, we run the analysis for ten million and five million states respectively. For the rest of the datasets, we run the Markov chains for two million states.

The Bayesian phylogenetic inference is performed on each meaning list's binary matrix. Each meaning list is processed cumulatively and the phylogenetic trees are stored to disk. In summary, each language family has binary matrices equal to the number of meanings in the family. Once the inference is finished, we proceed to evaluate the phylogenetic trees as described in the next subsection. For the evaluation we discarded the initial 25% of the trees as part of burnin and used the rest of the trees for evaluation.

4.3 Evaluation of inferred trees

The quality of inferred phylogenetic trees is evaluated using the quartet distance (Christiansen et al., 2006). The quartet distance measures the distance between two trees in terms of the number of different quartets. A quartet is a subtree consisting of four languages. A quartet for languages a, b, c, d can either be a *star* quartet when there is no internal branch separating the leaves or a *butterfly* quartet when there is an internal branch separating any two leaves from the two other leaves. The quartet distance measures the total number of different butterflies and star quartets divided by the total number of possible quartets in the tree. However, this definition of quartet distance does not work well when the gold standard tree is unresolved and has non-binary internal nodes. Expert linguistic phylogenies often have unresolved internal nodes due to lack of detailed knowledge causing the splits. In such a case the quartet distance penalizes the inferred tree because a butterfly quartet in the inferred tree is a star quartet in the gold standard tree and the former will be counted as erroneous. To remedy this, Pompei et al. (2011) proposed an extension to quartet distance known as Generalized Quartet Distance (GQD), which measures the distance between a binary tree and the gold standard tree as the number of different butterflies divided by the number of butterflies in the gold standard tree. We extract the gold standard trees from Glottolog (Hammarström et al., 2017)—a publicly available repository of language references and phylogenetic trees.

5 Results and Discussion

5.1 Comparison with gold standard trees

The results of the first test are given in Figure 1. Here, we plot the GQD as a function of number of meanings in each meaning list. Each point in the graph presents the GQD score for each meaning list. We fit a LOESS curve for each language family.[1] For all families but Austronesian the most drastic effect of list length size is seen for the first 50 or so items. For Mayan and Uto-Aztecan the GQDs appear to fluctuate not far after this point, whereas for Indo-European the drop continues to somewhat beyond 100 items. For Austronesian the regime with the most drastic drop is for the first 100 or so items, and then a slower drop sets in, continuing for the remainder of the curve.

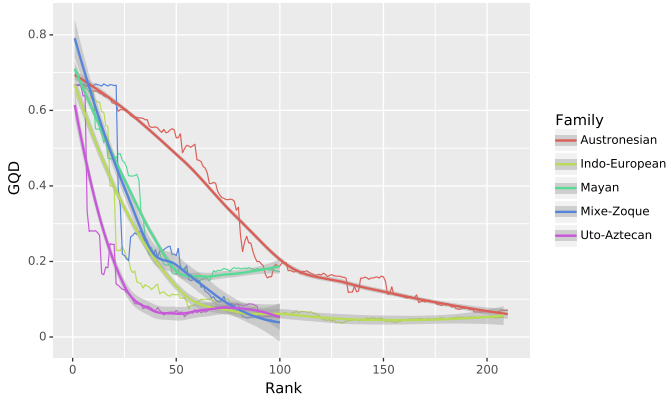


Figure 1: Lineplot of GQD of meaning lists sorted by the number of cognate classes when compared with gold standard trees. The trendlines are drawn using LOESS smoothing.

5.2 Comparison between successive trees

The second method is designed for related languages where the gold standard tree is not determined to the fullest satisfaction of the language family experts. Following this method, we compute the GQD between the inferred trees from successive meaning lists and then plot the GQD. The results, shown in Figure 2 are remarkably similar to the ones making reference to a gold standard tree, displayed in Figure

[1] All the plots are generated using `plotnine` python library available at `http://plotnine.readthedocs.io/en/stable/index.html`.

1. The visual impression of similarities in the shape of the curves is supported by Pearson correlations, which all lie in the range $0.890 > R^2 > 0.967$.

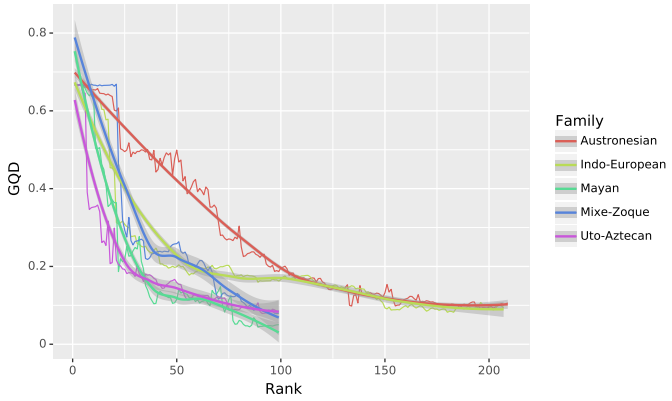


Figure 2: Lineplot of GQD between two successive meaning lists. We show the LOESS fit for each family.

Finally, we provide Figure 3, which shows plots for the remaining six language groups in our sample for which expert trees are generally absent or less reliable.

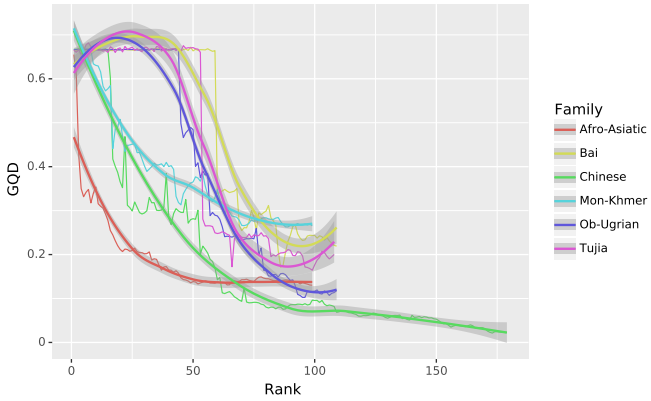


Figure 3: Lineplot of GQD between two successive meaning lists for the remaining language groups in our sample. We show the LOESS fit for each family.

5.3 Analyzing the results

In order to replace eye-balling with a more consistent identification of the cut-off between the approximately monotonic regime of each curve and the regime (if any) where fluctuations set in, we tested two different approaches. The first approach, inspired by Grieve et al. (2017), was as follows. We compute the Spearman rank correlation between the mean GQDs in each segment of the left tail with the corresponding numbers of meaning items, increasing the size of the segment in steps of one datapoint from 2 to the maximal number of meanings. For instance, for a segment of the curve corresponding to rank 1-10 we correlate the 10 GQD values with the series of numbers 1-10. As long as the result continues to be a strong negative correlation one could consider the curve to be approximately monotonically falling. Here we experimented with different cut-offs, including $\rho = -0.96$. If the increase of a segment of the curve by one meaning results in a weaker correlation than the cut-off, this point could then be considered to pertain to the fluctuating regime. As it turned out, there was no sensible way of defining a cut-off that would work across all eleven cases. Moreover, this method was too sensible to local fluctuations. Instead we applied a second and much simpler approach, which is to assume that the relevant point identifying the point where the GQD ceases to drop as simply the minimum. Table 3 summarizes the results.

Family	Lgs.	Meanings	Min (gold)	CC (gold)	Min (consec.)	CC (consec.)
Austronesian	96	210	203	3251	197	3091
Mayan	30	100	61	310	90	655
Mixe-Zoque	10	100	100+	300+	99	292
Indo-European	52	208	130	1078	181	1883
Uto-Aztecan	31	100	100+	846+	92	711
Afro-Asiatic	25	100			100+	1273+
Bai	9	110			87	137
Chinese	18	180			174	1115
Mon-Khmer	16	100			90	606
Ob-Ugrian	21	110			96	172
Tujia	5	109			100	152

Table 3: The information on languages and total meanings from Table 1 is repeated, and in addition the last four columns indicate the point where the curves reach their minimum ("Min"), and the corresponding number of cognate classes ("CC") for respectively the method using a gold standard ("gold") and the one using comparisons between consecutive size-increased meaning lists ("consec."). A plus sign following a number indicates that the minimum is reached when all items are used, such that we can suspect, although we cannot know for certain, that adding some more items (also implying more cognate classes) would continue to bring improvements.

The results in Table 3 suggest the existence of a linear correlation between the optimal number of meanings and the number of languages to be classified for the gold standard method. The correlation is strong ($R^2 = 0.786$) but only barely below the 0.05 significance level ($p = 0.0452$). As just mentioned, however, the real impact should come not from the mere number of items, but from the number of cognate classes they produce. Thus, not unexpectedly, the correlation is stronger yet between the optimal number of cognate classes and the number of languages ($R^2 = 0.923$) and also significant ($p = 0.0094$). For the method using consecutive datasets for eleven language families and groups the correlation between the number of meanings and the number of languages is weak albeit significant ($R^2 = 0.537$, $p = 0.010$); but again the situation radically improves when the number of languages is correlated with the number of cognate classes ($R^2 = 0.876$, $p < 0.0001$).

The linear model for the optimal number of cognate classes as a function of the number of languages intercepts at -11.24. Since to classify zero languages zero cognate classes are needed the intercept can be set at zero, something which only requires a small adjustment, and the slope of the resulting function will then be 32.397, indicating that ∼33 cognate classes per language classified is optimal. The gold standard results from the smaller set of five families gives us a slope of 29.297 when the intercept is at zero, indicating that ∼30 cognate classes is optimal. We choose to place more weight on the more conservative of these estimates.

Our result should be highly useful in the design of future studies in Bayesian phylogenetics. The initial decision on how many items to include in a lexical dataset can be difficult since the amount of resulting cognate classes may be unknown, but across our case studies the number of items required was always at least 87. Thus, for a small or medium-sized family (∼30 languages or less) the researcher could initially aim at 100 items, but for larger families this will clearly not suffice. For around 31-100 languages 200 items may be needed, and for yet larger families more than that. Counting cognate classes will yield a much more precise estimate of the adequacy of the sample. As an example, a recent study of Dravidian languages (Kolipakam et al., 2018) reports that 778 cognate classes were used for inferring a phylogeny for 20 languages. Since $20 \times 33 = 660$, we now have reasons to believe that this is an adequate sample. The sample of 20 languages, however, only contains around one fourth of the total set of Dravidian languages (Hammarström et al., 2017), so a future investigation may be directed at extending the phylogeny. In that case it can be anticipated that it will not suffice to add data for the list

of meanings used in Kolipakam et al. (2018)—rather, new items may have to be added.

6 Conclusion

This paper has, for the first time, addressed the question of the size of meaning lists required for the optimal Bayesian-based inferencing of linguistic phylogenies. In our search for this optimum, that is, the point where Bayesian inference stops improving, we find the following:

- Bayesian inference does not necessarily improve when given more cognate classes. In fact, more data can reduce the fit with expert trees, as is seen in the case of the Mayan family. Such a situation can be explained by a high level of borrowing where less stable words are exchanged more among lineages than more stable words, such that including these less stable words is not necessarily advantageous. (Indeed, Mayan languages are known to borrow heavily from one another, cf. Wichmann and Brown (2003)).

- The curves representing fits with gold standard trees and the ones representing fits between trees produced by successively adding items in descending order of stability are highly similar, so the latter can be used as a proxy for the former in order to increase the number of families sampled for the purpose of determining the optimal list size.

- The optimal list size for Bayesian phylogenetic inference depends on the number of cognate classes represented by the words corresponding to the items on the meaning list, and the number of cognate classes, in turn, is strongly correlated with the number of languages under investigation. Our results indicate that the optimal list size is one that produces around 33 cognate classes times the number of languages to be classified.

Acknowledgments

The first author is supported by BIGMED project (a Norwegian Research Council LightHouse grant, see bigmed.no). The second author is supported by a subsidy of the Russian Government to support the Programme of Competitive Development of Kazan Federal University. The experiments were performed when both authors took part in the ERC Advanced Grant 324246 EVOLAEMP project led by Gerhard Jäger. All these sources of support are gratefully acknowledged.

References

Alexandre Bouchard-Côté, David Hall, Thomas L. Griffiths, and Dan Klein. 2013. Automated reconstruction of ancient languages using probabilistic models of sound change. *Proceedings of the National Academy of Sciences*, 110(11):4224–4229.

Remco Bouckaert, Philippe Lemey, Michael Dunn, Simon J. Greenhill, Alexander V. Alekseyenko, Alexei J. Drummond, Russell D. Gray, Marc A. Suchard, and Quentin D. Atkinson. 2012. Mapping the origins and expansion of the Indo-European language family. *Science*, 337(6097):957–960.

Cecil H. Brown, Eric W. Holman, Søren Wichmann, and Viveka Velupillai. 2008. Automated classification of the world's languages: A description of the method and preliminary results. *STUF—Sprachtypologie und Universalienforschung*, 61(4):285–308.

Lyle Campbell and William J. Poser. 2008. *Language Classification: History and Method*. Cambridge University Press, Cambridge.

Chris Christiansen, Thomas Mailund, Christian N. S. Pedersen, Martin Randers, and Martin Stig Stissing. 2006. Fast calculation of the quartet distance between trees of arbitrary degrees. *Algorithms for Molecular Biology*, 1(1).

Michael Cysouw, Søren Wichmann, and David Kamholz. 2006. A critique of the separation base method for genealogical subgrouping, with data from Mixe-Zoquean. *Journal of Quantitative Linguistics*, 13(2-3):225–264.

Alexei J. Drummond, Marc A. Suchard, Dong Xie, and Andrew Rambaut. 2012. Bayesian phylogenetics with BEAUti and the BEAST 1.7. *Molecular Biology and Evolution*, 29(8):1969–1973.

Michael Dunn, Simon J. Greenhill, Stephen C. Levinson, and Russell D. Gray. 2011. Evolved structure of language shows lineage-specific trends in word-order universals. *Nature*, 473(7345):79–82.

Michael Dunn. 2012. Indo-European lexical cognacy database (IELex). http://ielex.mpi.nl/.

Běijīng Dàxué, editor. 1964. *Hànyǔ fāngyán cíhuì [Chinese dialect vocabularies]*. Wénzì Gǎigé.

Joseph Felsenstein. 1992. Phylogenies from restriction sites: A maximum-likelihood approach. *Evolution*, 46(1):159–173.

Joseph Felsenstein. 2004. *Inferring Phylogenies*. Sinauer Associates, Sunderland, Massachusetts.

Russell D. Gray, Alexei J. Drummond, and Simon J. Greenhill. 2009. Language phylogenies reveal expansion pulses and pauses in Pacific settlement. *Science*, 323(5913):479–483.

Simon J. Greenhill, Chieh-Hsi Wu, Xia Hua, Michael Dunn, Stephen C. Levinson, and Russell D. Gray. 2017. Evolutionary dynamics of language systems. *Proceedings of the National Academy of Sciences*, 114(42):E8822–E8829.

Jack Grieve, Andrea Nini, and Diansheng Guo. 2017. Analyzing lexical emergence in Modern American English online. *English Language & Linguistics*, 21(1):99–127.

Harald Hammarström, Robert Forkel, and Martin Haspelmath. 2017. *Glottolog*. Max Planck Institute for Evolutionary Anthropology, Leipzig. http://glottolog.org/.

Eric W. Holman, Søren Wichmann, Cecil H. Brown, Viveka Velupillai, André Müller, and Dik Bakker. 2008. Explorations in automated language classification. *Folia Linguistica*, 42(3-4):331–354.

Eric W. Holman, Cecil H. Brown, Søren Wichmann, André Müller, Viveka Velupillai, Harald Hammarström, Sebastian Sauppe, Hagen Jung, Dik Bakker, Pamela Brown, et al. 2011. Automated dating of the world's language families based on lexical similarity. *Current Anthropology*, 52(6):841–875.

Gerhard Jäger, Johann-Mattis List, and Pavel Sofroniev. 2017. Using support vector machines and state-of-the-art algorithms for phonetic alignment to identify cognates in multi-lingual wordlists. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics. Long Papers*, pages 1204–1215, Valencia. Association for Computational Linguistics.

Gerhard Jäger. 2013. Phylogenetic inference from word lists using weighted alignment with empirically determined weights. *Language Dynamics and Change*, 3(2):245–291.

Michael I. Jordan. 2004. Graphical models. *Statistical Science*, 19(1):140–155.

Vishnupriya Kolipakam, Fiona M. Jordan, Michael Dunn, Simon J. Greenhill, Remco Bouckaert, Russell D. Gray, and Annemarie Verkerk. 2018. A Bayesian phylogenetic study of the Dravidian language family. *Royal Society Open Science*, 5:171504.

Alfred Louis Kroeber. 1963. *Yokuts Dialect Survey*, volume 11(3) of *University of California Publications: Anthropological Records*. University of California Press, Berkeley.

Paul O. Lewis. 2001. A likelihood approach to estimating phylogeny from discrete morphological character data. *Systematic Biology*, 50(6):913–925.

Alexander Militarev. 2000. Towards the chronology of Afrasian (Afroasiatic) and its daughter families. In Colin Renfrew, April McMahon, and Larry Trask, editors, *Time Depth in Historical Linguistics*, pages 267–307. McDonald Institute for Archaelogical Research, Cambridge.

Wick R. Miller. 1984. The classification of the Uto-Aztecan languages based on lexical evidence. *International Journal of American Linguistics*, 50(1):1–24.

R. L. Oswalt. 1970. The detection of remote linguistic relationships. *Computer Studies in the Humanities and Verbal Behavior*, 3(3):117–129.

Mark Pagel and Andrew Meade. 2004. A phylogenetic mixture model for detecting pattern-heterogeneity in gene sequence or character-state data. *Systematic Biology*, 53:571–581.

Mark Pagel and Andrew Meade. 2006. Estimating rates of lexical replacement on phylogenetic trees of languages. In Peter Forster and Colin Renfrew, editors, *Phylogenetic Methods and the Prehistory of Languages*, pages 173–182. McDonald Institute Monographs, Cambridge.

Mark Pagel, Quentin D. Atkinson, and Andrew Meade. 2007. Frequency of word-use predicts rates of lexical evolution throughout Indo-European history. *Nature*, 449(7163):717–720.

Ilia Peiros. 1998. *Comparative Linguistics in Southeast Asia*. Number 142. Pacific Linguistics, Research School of Pacific and Asian Studies, National University of Australia.

Filippo Petroni and Maurizio Serva. 2010. Lexical evolution rates derived from automated stability measures. *Journal of Statistical Mechanics: Theory and Experiment*, 2010:P03015.

Simone Pompei, Vittorio Loreto, and Francesca Tria. 2011. On the accuracy of language trees. *PloS one*, 6(6):e20109.

Taraka Rama and Lars Borin. 2013. N-gram approaches to the historical dynamics of basic vocabulary. *Journal of Quantitative Linguistics*, 21(1):50–64.

Taraka Rama and Lars Borin. 2015. Comparative evaluation of string similarity measures for automatic language classification. In Ján Mačutek and George K. Mikros, editors, *Sequences in Language and Text*, pages 203–231. Walter de Gruyter.

D. F. Robinson and L. R. Foulds. 1979. Comparison of weighted labelled trees. *Combinatorial Mathematics*, 6:119–126.

Fredrik Ronquist and John P. Huelsenbeck. 2003. MrBayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics*, 19(12):1572–1574.

Fredrik Ronquist, Seraina Klopfstein, Lars Vilhelmsen, Susanne Schulmeister, Debra L. Murray, and Alexandr P. Rasnitsyn. 2012a. A total-evidence approach to dating with fossils, applied to the early radiation of the hymenoptera. *Systematic Biology*, 61(6):973–999.

Fredrik Ronquist, Maxim Teslenko, Paul van der Mark, Daniel L. Ayres, Aaron Darling, Sebastian Höhna, Bret Larget, Liang Liu, Marc A. Suchard, and John P. Huelsenbeck. 2012b. MrBayes 3.2: efficient Bayesian phylogenetic inference and model choice across a large model space. *Systematic Biology*, 61(3):539–542.

Naruya Saitou and Masatoshi Nei. 1987. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425.

George S Starostin. 2013. Annotated swadesh wordlists for the tujia group. *The Global Lexicostatistical Database, RGGU, Moscow. URL: http://starling. rinet. ru/new100/tuj. xls*.

Morris Swadesh. 1952. Lexico-statistic dating of prehistoric ethnic contacts: with special reference to North American Indians and Eskimos. *Proceedings of the American Philosophical Society*, 96(4):452–463.

David Thomas. 1960. Basic vocabulary in some Mon-Khmer languages. *Anthropological Linguistics*, 2(3):7–11.

Feng Wang. 2006. *Comparison of languages in contact: the distillation method and the case of Bai*, volume 3. Institute of Linguistics, Academia Sinica.

Søren Wichmann and Cecil H. Brown. 2003. Contact among some mayan languages: inferences from loanwords. *Anthropological Linguistics*, 45(1):57–93.

Søren Wichmann and Eric W Holman. 2013. Languages with longer words have more lexical change. In Lars Borin and Anju Saxena, editors, *Approaches to Measuring Linguistic Differences*, pages 249–281. Mouton de Gruyter, Berlin.

Søren Wichmann, Eric W. Holman, Dik Bakker, and Cecil H. Brown. 2010a. Evaluating linguistic distance measures. *Physica A: Statistical Mechanics and its Applications*, 389:3632–3639.

Søren Wichmann, André Müller, and Viveka Velupillai. 2010b. Homelands of the world's language families: A quantitative approach. *Diachronica*, 27(2):247–276.

Søren Wichmann, Eric W. Holman, and Cecil H. Brown, editors. 2018. *The ASJP Database (version 18)*. http://asjp.clld.org/.

Ziheng Yang. 1994. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: Approximate methods. *Journal of Molecular evolution*, 39(3):306–314.

Ziheng Yang. 2014. *Molecular Evolution: A Statistical Approach*. Oxford University Press, Oxford.

Chi Zhang, Tanja Stadler, Seraina Klopfstein, Tracy A. Heath, and Fredrik Ronquist. 2015. Total-evidence dating under the fossilized birth–death process. *Systematic Biology*, 65(2):228–249.

M Zhivlov. 2011. Annotated swadesh wordlists for the ob-ugrian group (uralic family). *The Global Lexicostatistical Database. RGU*.

A Supplemental Material

The code and data used in this paper are uploaded as a zip file along with this paper. In addition, they are available for download via Zenodo at `https://doi.org/10.5281/zenodo.1287317`

The Road to Success: Assessing the Fate of Linguistic Innovations in Online Communities

Marco Del Tredici and **Raquel Fernández**
Institute for Logic, Language and Computation
University of Amsterdam
{m.deltredici|raquel.fernandez}@uva.nl

Abstract

We investigate the birth and diffusion of lexical innovations in a large dataset of online social communities. We build on sociolinguistic theories and focus on the relation between the spread of a novel term and the social role of the individuals who use it, uncovering characteristics of innovators and adopters. Finally, we perform a prediction task that allows us to anticipate whether an innovation will successfully spread within a community.

1 Introduction

Language is an incessantly evolving system, and linguistic innovations of different kind are continuously created. New variants originate in concrete communicative situations with particular speakers. After being introduced, some of them are adopted by other speakers belonging to the same community and possibly spread until they become community norms. In contrast, other innovations do not manage to make their way into the community conventions and just disappear after a certain period of time. This process raises many intriguing questions, which have been the focus of attention in Sociolinguistics since the late 1960's (Weinreich et al., 1968; Chambers and Schilling, 2013). For example, at what linguistic levels (phonology, lexicon, syntax) do innovations arise and succeed? Who are the leaders of change and what are their social characteristics? In this paper, we investigate lexical innovation in online communities, focusing on the latter type of question, by means of a large-scale data-driven approach.

We study the interplay between the birth and spread of new terms and users' social standing in large online social communities, taking as starting point hypotheses put forward in the Sociolinguistics literature (see Section 2). We present a longitudinal study on a large social media dataset, including 20 online forums and around 10 million users overall. We consider each forum as an independent social network, and investigate how lexical innovations arise and spread within a forum. We analyse the fate of approximately 8 thousand innovations (focusing on acronyms such as *lol*, phonetic spellings such as *plis*, and other linguistic phenomena usually collected under the term of *Internet slang*), and relate their spread within a social network to the role of the individuals who use them. We characterise users' roles within a community by means of a novel, theoretically-motivated *tie-strength* measure, combined with well-known centrality measures.

We show that (1) innovators (users who introduce a new term) are central members of a community, connected to many other users but with relatively low tie-strength, and (2) strong-tie users (who belong to cliques or sub-groups within the community) effectively contribute to the dissemination of a new term. This pattern is surprisingly consistent across the 20 online communities under investigation. In addition, we show that, by solely using information on speakers' tie strength as predictor variable, we can anticipate whether an innovation will successfully spread within a community.

Our work yields new theoretical insights into the applicability of sociolinguistic theories to online settings. It also has practical significance for NLP systems encountering novel terms in social media: While new terms that do not succeed in becoming community norms may safely be treated as out-of-vocabulary words, a better understanding of the dissemination process of novel terms beyond their

Proceedings of the 27th International Conference on Computational Linguistics, pages 1591–1603
Santa Fe, New Mexico, USA, August 20-26, 2018.

frequency can help identify those innovations that a system should be able to deal with. Together with the in-depth analysis presented in the paper, we make available our dataset, which includes interactions scraped from 20 topic-based forums over a period of 4 to 8 years, as well as the code for preprocessing the data, extracting the dissemination trajectories of ∼8k internet slang terms, and computing the strength of the ties among users.[1]

2 Background

2.1 Sociolinguistic Theories

Our investigation of innovations in online social communities builds on Milroy's (1987) theory, according to which linguistic innovations propagate and become community norms when individuals with *strong ties* adopt them early on; however, it is usually individuals with only *weak ties* who are the actual *innovators*, i.e., those who introduce novel linguistic variants. Milroy's proposal is thus an application to Sociolinguistics of *The Strength of Weak Ties* theory put forward by Granovetter (1973) to explain the diffusion of innovations more broadly.

In Milroy's (1987) original study, the strength of a tie between two individuals is given by a combination of factors, such as the amount of time spent together, the level of intimacy, etc. As a result, strong ties are usually those connecting family members or close friends, while weak ties connect acquaintances. Individuals linked by strong ties form close-knit sub-communities or clusters, whereas weak ties act as bridges between these clusters. According to Milroy's theory, individuals without strong ties are more likely to innovate. Thanks to their bridging role, they can introduce an innovation into one or more close-knit clusters. Once introduced, the innovation can be adopted by closely connected individuals within a cluster, who quickly propagate it to others.

Milroy's theory has found confirmation both in small-scale sociolinguistic studies carried out in the protestant enclaves of Belfast (Milroy and Milroy, 1985; Milroy and Milroy, 1987) and in agent-based models with simulated data (Fagyal et al., 2010). However, it has not been tested at a large scale on actual linguistic data. Moreover, the theory contrasts with other well-established models regarding the source of novel variants, which casts some doubt on the generality of its application: Labov's studies on New York adolescent gangs and the neighbourhoods of Philadelphia (Labov, 1972; Labov, 2001) identified *leaders*, i.e., those who have strong networks and are the most popular in their communities, as the drivers of change. Similarly, Eckert's (2000) studies of adolescent groups in Detroit showed that language novelty and change is led by charismatic leaders with strong ties to the local community.

We assess the applicability of Milroy's theory to the emergence of lexical innovations in online social communities, where relationships between individuals and their exposure to new terms can only be modelled in terms of their directly observable linguistic interaction in online discussion threads.

2.2 Related Computational Work

Research on language change has recently experienced a boost within the NLP community (Hamilton et al., 2016; Frermann and Lapata, 2016). Here we focus on reviewing approaches that are directly related to the work we present in this paper, i.e., approaches that investigate linguistic innovation within relatively short time spans of a few years (rather than looking at historical time) and that do so by leveraging data from social media to create a graph of connected users. More general literature on graph modelling and innovation diffusion is out of the scope of our review.

Graphs representing social networks have been used to analyse the extralinguistic factors that drive diffusion, such as geographical (Eisenstein, 2015) and demographic variables (Eisenstein et al., 2014), as well as the position of individuals in the social network (Paolillo, 1999) and the kind of user interactions that foster the diffusion of the innovations. For example, Goel et al. (2016) and Paradowski and Jonak (2012) investigate the amount of interaction required to adopt an innovation, showing that while some innovations are adopted by a new user after multiple interactions, for others a single contact is sufficient. Rotabi et al. (2017a) relate the success of competing lexical variants to the seniority of the users who use them. This work is extended by Rotabi et al. (2017b), who introduce an inheritance graph

[1] https://github.com/marcodel13/The-Road-to-Success

to represent how speakers pass innovations (in this case called *practices*) on to others in time through real-life collaborations.

Perhaps the work that is most directly connected to ours is that of Rotabi and Kleinberg (2016), who investigate the diffusion patterns of words that experience a frequency burst at a given time, which the authors call *trends*, and analyse the level of activity of the users involved in the different phases of the diffusion process. While related, our work differs on key points: we focus on linguistic innovations rather than trends and characterise users in terms of tie strength rather than level of activity.

A parallel line of work has investigated linguistic diffusion in relation to social ties using agent-based computer simulations (Ke et al., 2008; Fagyal et al., 2010; Swarup et al., 2011). In particular, Fagyal et al. (2010) address questions similar to ours via simulated data and find that peripheral users are crucial in introducing new linguistic variants, which are then adopted and spread by central agents in the community. The current availability of large social media datasets containing real linguistic interactions allows us to study these questions with more ecological validity.

3 Methodology

We describe our dataset, the methodology we use to define the social network of a community and the social role of its members, and the procedure to identify linguistic innovations and characterise their diffusion.

3.1 Data

We use data from Reddit,[2] a popular website which includes around 1 million communities called *subreddits*. A subreddit is a topic-based forum where users can submit posts, comment on existing posts or score them. While in this paper we treat each forum independently (leaving the analysis of innovation diffusion across forums for future work), in order to conduct a large-scale analysis and draw conclusions that generalise across communities, we analyse 20 different forums that show substantial variability in terms of subject matter and size (see Table 1 for an overview). Despite their rich heterogeneity, all subreddits we consider share the following features, which are indicative of highly active and interconnected communities, where the spreading cascade process at the base of innovation diffusion finds a favourable environment (Hamilton et al., 2017; Guille et al., 2013):

- a topic that reflects a strong external interest, such as sport teams, videogames or TV series;
- small-to-medium size, which in contrast to very large subreddits such as r/news (15M users) or r/funny (18M), are less dispersive and favour tighter connections;
- high density, i.e., high ratio of existing connections over the number of potential connections;[3]

We downloaded the entire content of each subreddit from its first post to the end of 2016. We segment the data from each subreddit into consecutive time bins corresponding to one month.[4] We discard time bins with less than 200 active users, which are common during the first few months of a subreddit lifespan, and ignore any posts whose author is unknown.[5]

Next, we explain how we leverage this longitudinal data to extract information about the social role of forum members, as well as to detect linguistic innovations and characterise their diffusion.

3.2 Social Network

We create a graph representing a community's social network for each month t during the community lifespan. In these graphs, nodes are users and edges encode whether users have interacted. We consider two users to be connected by an edge if they comment within the same thread in close proximity.[6] Given

[2] `https://www.reddit.com`

[3] Hamilton et al. (2017) report density values in the range $[0.001 - 0.016]$ in their set of subreddits.

[4] We also experimented with smaller bins of one week, obtaining similar results.

[5] When users delete their account, the posts, comments, and messages submitted prior to the deletion are still visible to others, but information about the user is not available (see Reddit Privacy Policy at `https://www.redditinc.com/policies/privacy-policy`).

[6] In particular, if they are separated by at most two posts, as done e.g., by Hamilton et al. (2017).

subreddit	years	tokens	users	density	innovations
r/Android	7	158	1.03M	0.006	730
r/apple	8	89	580k	0.006	584
r/baseball	6	101	576k	0.014	520
r/beer	7	29	291k	0.008	360
r/boardgames	6	88	313k	0.004	380
r/cars	6	101	544k	0.014	605
r/FinalFantasy	4	22	137k	0.009	218
r/Guitar	7	71	387k	0.009	496
r/harrypotter	5	39	287k	0.005	227
r/hockey	7	191	847k	0.012	602
r/Liverpool	5	40	173k	0.018	314
r/Patriots	5	26	151k	0.009	231
r/pcgaming	5	52	350k	0.003	360
r/photography	8	81	353k	0.006	485
r/pokemon	6	107	1.02M	0.006	695
r/poker	6	28	104k	0.012	258
r/reddevils	4	49	186k	0.008	329
r/running	6	56	279k	0.008	367
r/StarWars	6	56	542k	0.008	381
r/subaru	5	21	187k	0.005	340

Table 1: Statistics of the subreddits in our dataset, including: years of activity considered until end of 2016; total # of tokens (in millions); total # of active users (including users who may have left the community); average ratio of network ties over all possible ties computed over all the time bins (density); total # of linguistic innovations analysed.

that arguably lexical diffusion follows a simple "contagion" model whereby a single contact is sufficient for the spreading process (Goel et al., 2016), our graphs are undirected and unweighted.[7]

Milroy's (1987) theory relates the diffusion of a linguistic innovation to the local topology of individuals within a network, distinguishing between close-knit sub-groups and individuals outside of these groups. In line with other studies such as Weng et al. (2015) and Zhao et al. (2010), we build on a measure introduced by Onnela et al. (2007), which determines the strength of the edge between two individuals i and j in terms of the overlap O_{ij} of their adjacent neighbourhoods, as follows:

$$O_{ij} = \frac{n_{ij}}{(k_i - 1) + (k_j - 1) - n_{ij}} \quad [1]$$

where n_{ij} is the number of adjacent nodes between i and j, and k_i, k_j their respective degree, i.e., the number of edges incident to each of them. Possible values for O_{ij} are in the range $[0, 1]$, where 0 indicates no common neighbours (weakest possible connection between i and j) and 1 exactly the same adjacent neighbours (strongest possible connection between i and j).

We now leverage Equation [1] to characterize users given the strength of their connections. According to Milroy (1987), weak-tie individuals have only weak connections (are not part of close-knit clusters), while strong-tie individuals have strong connections with other users, but may also have weak connections if they are linked to weak-tie users. To capture this, we define the tie strength of each individual i as the highest value of her incident edges. That is, for all individuals j directly connected to i:

$$\textit{tie-strength}(i) := \max(O_{ij}) \quad [2]$$

Taking the maximum captures what we are after: A community member who only has weak connections will have low tie-strength and be considered a *weak-tie user*, while a member with either strong connections only or with both strong and weak connections will have high tie-strength and be considered a *strong-tie user*, who will be part of a local clique (Luce and Perry, 1949). Unlike mean or median values, which tend to be rather balanced and thus do not help to distinguish between innovators and non-innovators, taking the maximum is in line with Milroy's theory and captures the key difference between these two groups of users, as will be shown in the next section.

[7]The graphs are implemented with Python's package `networkx`.

Besides computing tie-strength as defined in Equation [2] for all users in our social graphs, we also compute their centrality values for three common measures of network centrality — degree, betweenness, and eigenvector — which are global indices of the importance of a node with respect to all other nodes in a graph (Newman, 2010).

3.3 Linguistic Innovations

We focus on *Internet slang*, a general term commonly used to refer to a range of linguistic phenomena such as abbreviations (*cu* for *see you*), acronyms (*IIRC* for *if I remember/recall correctly*) and phonetic spellings (*dat* for *that*). The motivation behind this choice is twofold: firstly, forms of this kind are very abundant and continuously introduced in online communication; and secondly, they are easier to identify and track than innovations at other levels, such as those related to meaning shift. In the present study, we do not focus on tracking the co-evolution of two variants (e.g., *dat* vs. *that*) that may be competing, but rather on analysing the emergence of new forms and their trajectories independently, in light of the tie-strength of the members who use them.

Our starting point are the terms in the dictionary available at `NoSlang.com`,[8] a comprehensive record of Internet slang that is constantly updated. After removing terms including non-alphabetic characters, we obtain a list of approximately 6k terms. For each subreddit, we only consider terms that:

- are used at least 10 times in the subreddit;
- are not present during the first 3 months of the community's existence; and
- are introduced within the initial quarter of the community lifespan.

That is, we restrict our analysis to *newly* introduced Internet slang terms that are not present from the very beginning of the community's activity, but are not introduced too late so as to be able to observe their trajectory for a substantial period of time.[9]

The number of innovations considered across all subreddits is 7962, while the number of unique innovations amounts to 1456. Most of the terms (around 76%) occur in more than one community, although no innovation is present in all the subreddits in our dataset. Thus, around 24% of innovations tracked occur in just one community — some of these are clearly topic-related, e.g., *pkemon* in /r/pokemon, while others are more general purpose abbreviations that, in principle, could appear in any community, such as *txs* (*thanks*, in /r/Android) or *omgz* (*oh my god/gosh*, in /r/subaru). Regarding frequency, while we set a minimum threshold of 10 occurrences, in practice 72% of terms occur at least 50 times on average.

Dissemination trajectory. Once introduced, innovations may have different fates: they can spread widely within the community, be used by just a small sub-group, or fail to make an impact and disappear altogether. We define the fate of a term as its *dissemination*, which we compute as the proportion of community members who use it at a given moment in time (Del Tredici and Fernández, 2017; Altmann et al., 2011). It should be stressed that dissemination differs from frequency. Although often they are highly correlated, in principle a term can have high relative frequency but low dissemination and vice versa. We quantify the diffusion of innovations in terms of their dissemination since this gives us a measure of their spread within a community. To this end, for each innovation we calculate its *dissemination trajectory* as the vector of its monthly dissemination values since the innovation was introduced.

Tie-strength trajectory. Similarly, we compute the *tie-strength trajectory* of each innovation as a vector whose features correspond to the maximum tie-strength value among the users that used it in the corresponding month. We choose the maximum value in order to test Milroy's (1987) hypothesis, according to which the crucial factor in the diffusion of an innovation is its adoption by strong-tie members (see Section 2). Considering only the maximum value provides a simple way to test whether any individual with high tie-strength has used the term in a given month.[10]

[8] `https://www.noslang.com/dictionary/`

[9] To assess whether ambiguity may be an issue, we checked whether the terms also appear in a standard English dictionary, PyDictionary. For example, the slang term *bra* for *brother* also has the standard meaning of *brasserie*. Given that under 2% of terms in our dataset could potentially be ambiguous, we decided to not treat them in any special way.

[10] The dissemination and the tie-strength vectors always have the same magnitude.

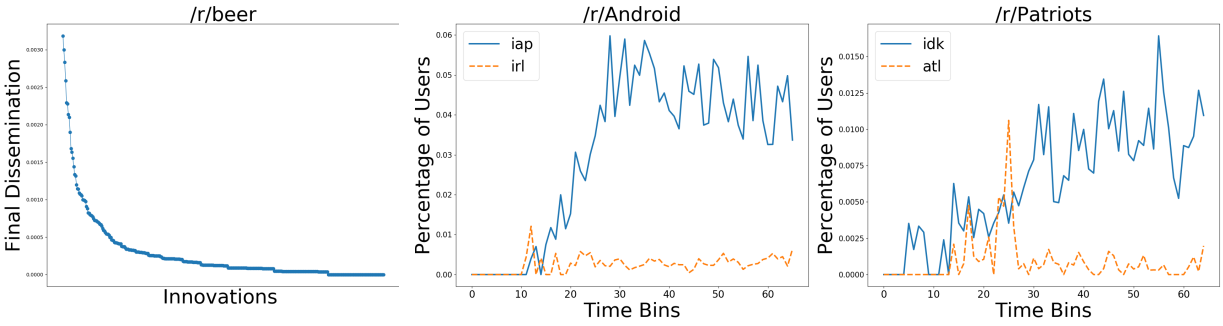


Figure 1: **Left**: Distribution of the final dissemination values in /r/beer. **Center/Right**: Examples of dissemination trajectories of *successful* (blue solid line) and *unsuccessful* (orange dashed line) innovations.

4 Empirical Observations

Our analysis reveals some patterns common to all communities, which we present in this section.

4.1 Linguistic Innovations

In order to explore the relative success of innovations, we consider the level of dissemination reached by an innovation to be the average dissemination value in the last six months.[11] Results show that the distribution of these final dissemination values is highly skewed for all the subreddits — see, for example, Figure 1 (left) for /r/beer. While a few innovations disseminate successfully (i.e., are adopted by a relatively high number of community members), most of them do not spread, and either disappear or barely appear in the last period.

In Figure 1 (center/right) we show examples of successful and unsuccessful innovations. Successful innovations such as *iap* (*In App Purchases*) and *idk* (*I don't know*) show a stable increase in dissemination after their introduction, which can reach a plateau at some point (*iap*) — thus showing the S-shaped curve typical of general processes of innovation adoption (Rogers, 2010) — or can still be ongoing at the end of the period covered by our analysis (*idk*). Unsuccessful innovations, in contrast, can either have a flat dissemination trajectory, as in the case of *irl* (*In Real Life*), indicating that the term has never experienced a spread in the community, or present a peak at some point, followed by a sudden decrease with no stable recovery, as for *atl* (*Atlanta*).

Given these observations, we formally define the classes of *successful* and *unsuccessful* innovations based on the dissemination *slope* of a term, computed as the difference between its average dissemination value in the first six months and in the last six months in the dissemination trajectory vector. We include in the *unsuccessful* class innovations with slope index ≤ 0, i.e., those with trajectories similar to the *irl* and *atl* examples in Figure 1 (center/right). In order to discard innovations with very low positive slope (i.e., those that do not disappear, but are only sporadically used) we only include in the *successful* class terms whose slope index is above the average value of the community.[12] We will make use of these two classes in the prediction experiment we present in Section 6.

4.2 Social Networks

Next, we analyse the distribution of users' tie-strength values in the social graphs derived for the communities in our dataset.[13] We find a clear pattern for all subreddits: the large majority of users have low tie-strength, with around 39% having values ≤ 0.05 and almost 50% having values ≤ 0.1; while, around 15 to 20% of users have strong tie-strength, with values ≥ 0.5. Figure 2 shows the average tie-strength value distribution computed over all the monthly graphs of all subreddits in the dataset, with probabilities calculated for bins of size 0.1 for illustration purposes.

[11] This makes our measurements more robust than taking only the very last bin.

[12] Average slope index is positive for all subreddits.

[13] A sample graph is available at `https://github.com/marcodel13/The-Road-to-Success`.

This distribution mirrors the typical power-law distribution observed for centrality measures in online communities (Mihalcea and Radev, 2011). The topological properties captured by our tie-strength measure (Equation [2]), however, are different from those captured by centrality, as already hinted at in Section 3.2. The three centrality measures considered (degree, betweenness, and eigenvector) correlate strongly with each other (Spearman's r in range 0.85–0.89).[14] But there is only a moderate correlation with tie-strength: r=0.63 degree, r=0.61 betweenness, and r=0.47 eigenvector. In addition, we observe that the three centrality measures correlate strongly with number of posts (r=[0.78–0.91]) in all subreddits, while we find low correlation between number of posts and tie-strength (r=0.31, std=0.08, $p < .05$).

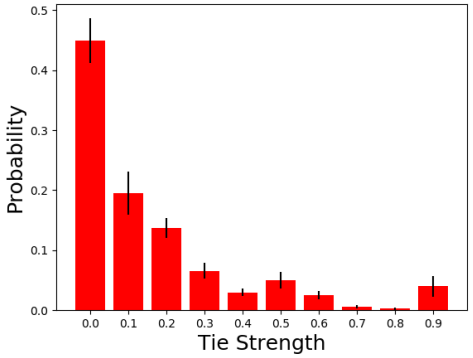


Figure 2: Average tie-strength distribution for all subreddits, with standard deviation.

These results confirm the difference between our tie-strength measure and centrality. While centrality values are *global* indices of the role of a node with respect to the entire graph (Newman, 2010), tie-strength captures the *local* topological information around a node. In the online social communities we investigate, individuals at the core of the social network, who interact with many other individuals and have high posting activity, receive high centrality values. In contrast, high tie-strength values are the signature of users who belong to small cliques, but who do not act as hubs for the entire network. We take this as confirmation that our tie-strength measure does capture key features of the social structures underpinning Milroy's (1987) theory. It remains to be seen, however, whether these social structures, as realised in large online social communities, lend support to the theory's main claims.

5 Assessing Sociolinguistic Claims

In this section, we uncover the features that characterise innovators and analyse the role of strong-tie users in the dissemination process.

5.1 Innovators

We consider *innovators* those members who introduce a new term, i.e., those who use it for the very first time in a community. In order to verify whether innovators are weak-tie users (as hypothesised by Milroy (1987)), we compare the distribution of the tie-strength values of innovators to the general tie-strength distribution in the community. Since innovations are introduced at different points in time, the general tie-strength distribution in the community is defined as the average of the months at which innovations were introduced. We compute Kullback-Leibler divergence (KL), which measures how similar the behaviour of two distributions is. We find KL values in the range 0.15–0.4, which indicates a moderate difference between the distributions. The source of this difference can be appreciated in Figure 3: the tie-strength values of innovators cluster around 0.1–0.3.[15] This contrasts with the overall tie-strength distribution: innovators tend to *not* be strong-tie users (lower blue than red bars for tie-strength ≥ 0.4 in Figure 3) and are far less likely to

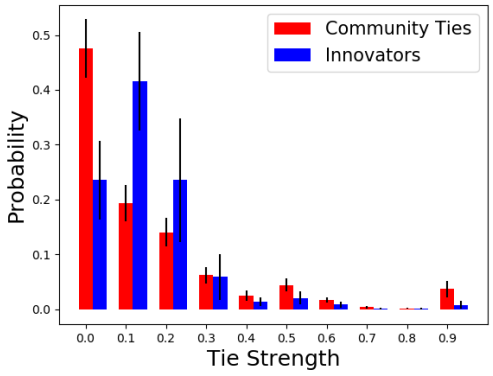


Figure 3: Comparison of the probability mass distribution of innovators' tie-strength values and avg. values of all users in all subreddits.

[14] All correlation coefficients reported are averages across time bins and subreddits and are all significant with $p < 0.05$.

[15] The same trend holds for all subreddits. Individual plots per subreddit can be found at `https://github.com/marcodel13/The-Road-to-Success`.

have very weak tie-strength than most users (lower blue than red bar for tie-strength < 0.1).

In addition to analysing tie-strength, we compare the centrality values and the posting activity of innovators and non-innovators. We observe that innovators are significantly more central than other users (for all measures considered: degree, betweenness and eigenvector) and are significantly more active in terms of number of posts than other individuals in the community (unpaired Welch's t-tests, all with $p < 0.05$).

Thus, a very robust pattern emerges across all subreddits, showing that innovators do have a particular profile in terms of their social standing: they do not belong to tightly connected cliques and occupy a central position in the network, as hubs with many connections of relatively low strength. On the one hand, this seems in line with Milroy's hypothesis, since it confirms that innovations do not arise within sub-communities that are close-knit. On the other hand, our results may also be interpreted as lending support to Labov's (1972; 2001) characterisation of innovators as *leaders*, since they occupy a core position in the network.

5.2 Strong-Tie Users and Innovation Spread

We consider strong-tie users those with a tie-strength value ≥ 0.5. By definition, these are users who are part of cliques or sub-communities within a subreddit. As mentioned in Section 4.2, strong-tie users constitute between 15 and 20% of the total number of community members. Thus, in contrast to the small scale social communities examined by Milroy and other sociolinguists, where most community members are part of some tight-knit sub-group (such as a family or a church congregation), in large online social networks strong-tie users are a minority. Despite this, strong-tie users are not isolated in remote cliques: Our analysis shows that, in all subreddits, strong-tie users are significantly more central (with respect to all, degree, betweenness, and eigenvector centrality) and have significantly more posting activity than users with weak tie-strength values ≤ 0.05, who make up the vast majority of community members — around 39% on average (unpaired Welch's t-test, $p < 0.05$). Thus, in terms of centrality, strong-tie members occupy an intermediate position in the social network, forming a loose ring around the core innovators and the majority of members, who are on the periphery and are characterised by very weak tie-strength.

To analyse the role of strong-tie users in the innovation diffusion process, we proceed as follows: We identify any time t_i in the tie-strength trajectory vector when the innovation is used by some strong-tie member for k consecutive months. We then check its average dissemination in the period up to time t_i and compare it to the average dissemination in the six months following t_{i+k-1}. We find that when $k = 1$ (i.e., when an innovation has been used by a strong-tie member only in one month) the probability that the dissemination increases in the next six months is around 50% for all subreddits — a value similar to the likelihood of dissemination increase after the usage by a weak-tie user. However, as k increases, and thus the adoption by strong-tie users becomes more stable, a future increase in dissemination becomes progressively more likely, for all the subreddits. Importantly, the same effect is not observed for weak-tie users, for whom, independently from the number of months, the probability of a future increase in dissemination is always approximately the same. Figure 4 shows examples of how the probability of dissemination changes after the adoption by either strong- or weak-tie users (see Appendix A for full results).

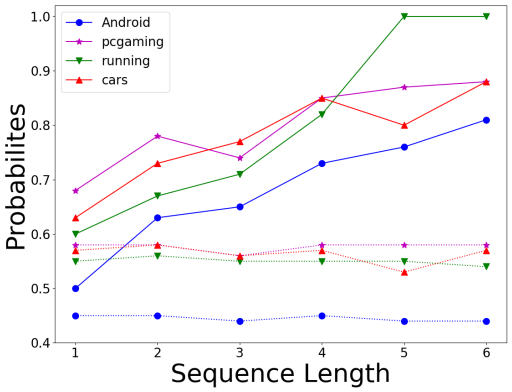


Figure 4: Probability of dissemination increase after a term is adopted by a strong-tie user (solid line) or by a weak-tie user (dotted line) for k consecutive months, computed for k in range $[1 - 6]$.

These results, thus, are consistent with Milroy's (1987) claims, and furthermore show that innovation diffusion is connected to *sustained* adoption by strong-tie community members.

6 Predicting Innovation Success

Most innovations do not succeed in becoming community norms, but some do. Here we assess whether information about the tie-strength of members who use an innovation in the first months after its introduction can predict whether it will be successful in the future. This provides further theoretical insight into the importance of tie-strength for innovation diffusion, and has practical significance by contributing to identifying new terms that NLP systems should be able to process. Our aim here is not to maximise prediction accuracy—which is likely to require taking into account several factors beyond users' tie-strength—but rather to explore whether the statistical effects we have uncovered are strong enough to have some predictive power.

We approach this as a binary classification task, making use of the distinction between *successful* and *unsuccessful* innovations defined in Section 4. We extract a subvector of length k from the tie-strength trajectory vector of innovations and use it as features for the prediction. For instance, with $k = 3$, we use the tie-strength information from the first three months of usage of a term to predict if it will be successful or not, leveraging subvectors of increasing magnitudes. We use Python's `scikit-learn` Random Forest classifier with default parameters[16] and perform 100-run cross-validation, using 90% of the data for training and 10% for testing. We compare our results against a *weighted baseline*, whereby the two labels (successful/unsuccessful) are randomly assigned taking into account their frequency in the training set. The classes are fairly balanced across subreddits, with an average proportion of 55% successful and 45% unsuccessful.

When leveraging tie-strength information from only the first 3 months of usage, we obtain F1 results that are significantly higher than the baseline for 12 out the 20 subreddits. But overall, performance remains rather low, with an average F1-score for the *successful* class of 0.62 vs. 0.58 for the baseline. Given that new terms are introduced by users with relatively low tie-strength (as shown in Section 5.1), arguably in the initial few months before a novel term is picked up by a strong-tie user, there is little difference between successful and unsuccessful innovations. With tie-strength information from the first 6 months of usage, we are able to make predictions with results significantly above baseline for 18 out 20 subreddits, with an average F1-score of 0.68. Not surprisingly, performance increases substantially when information for a longer period (first/second year of usage) is exploited, reaching an average F1-score of 0.76, significantly above the baseline for all communities. Detailed results, including precision, recall, and F1-score for each subreddit, can be found in Appendix B. In Figure 5 we graphically illustrate the results for a few subreddits, which are representative of the general trend observed.

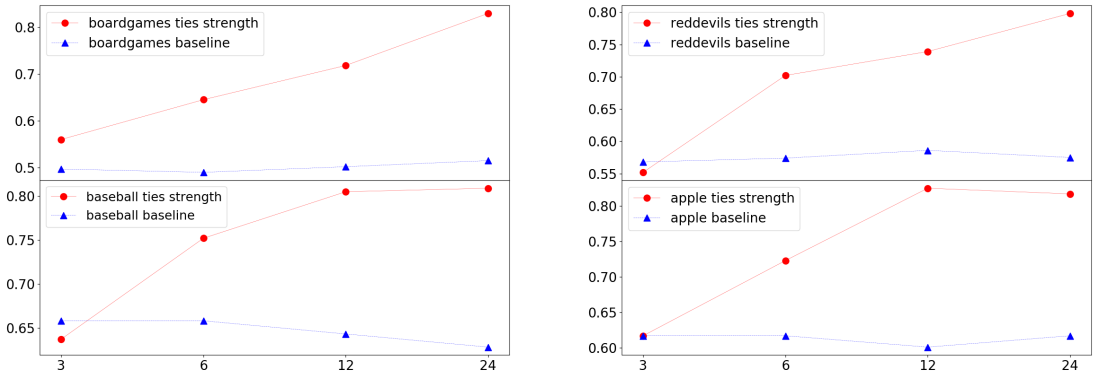


Figure 5: F-score (y-axis) for the successful class obtained with the ties-strength values of the first k months (x-axis) after the introduction of a term.

[16] `http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`.

7 Conclusions

This work has provided a large-scale analysis of the interplay between the birth and spread of new terms and users' social standing in large online social communities. Building on sociolinguistic theories — in particular, Milroy's (1987) version of *The Strength of Weak Ties* theory — we have proposed a simple measure to quantify tie-strength in Milroy's sense and have used it in combination with common centrality measures to uncover the characteristics of innovators and to assess the role of strong-tie users in the dissemination process.

Regarding innovators, our results show that they are central community members, connected to many other users but with relatively low tie-strength. As for strong-tie users, we find that in online social networks they are a small proportion of community members, organised in small cliques, and that they do play an important role in the spread of an innovation, presumably by spreading new terms introduced by innovators into their sub-groups. The patterns we have revealed are surprisingly consistent across the 20 online communities we have investigated.

Our work opens a range of interesting questions. In particular, we are looking forward to investigating how the patterns we have observed are affected by users' membership in multiple forums and how these multi-community users transfer new terms between communities.

Acknowledgements

This research has received funding from the Netherlands Organisation for Scientific Research (NWO) under VIDI grant nr. 276-89-008, *Asymmetry in Conversation*. We thank the anonymous reviewers for their comments as well as the area chairs and PC chairs of COLING 2018.

References

Eduardo G. Altmann, Janet B. Pierrehumbert, and Adilson E. Motter. 2011. Niche as a determinant of word fate in online groups. *PLoS ONE*, 6(5):e19009.

Jack K Chambers and Natalie Schilling. 2013. *The handbook of language variation and change*, volume 129. John Wiley & Sons.

Marco Del Tredici and Raquel Fernández. 2017. Semantic variation in online communities of practice. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS)*.

Penelope Eckert. 2000. *Language variation as social practice: The linguistic construction of identity in Belten High*. Wiley-Blackwell.

Jacob Eisenstein, Brendan O'Connor, Noah A Smith, and Eric P Xing. 2014. Diffusion of lexical change in social media. *PLoS ONE*, 9(11):e113114.

Jacob Eisenstein. 2015. Identifying regional dialects in online social media. In C. Boberg, J. Nerbonne, and D. Watt, editors, *Handbook of Dialectology*. Wiley.

Zsuzsanna Fagyal, Samarth Swarup, Anna María Escobar, Les Gasser, and Kiran Lakkaraju. 2010. Centers and peripheries: Network roles in language change. *Lingua*, 120(8):2061–2079.

Lea Frermann and Mirella Lapata. 2016. A bayesian model of diachronic meaning change. *Transactions of the Association for Computational Linguistics*, 4:31–45.

Rahul Goel, Sandeep Soni, Naman Goyal, John Paparrizos, Hanna Wallach, Fernando Diaz, and Jacob Eisenstein. 2016. The social dynamics of language change in online networks. In *International Conference on Social Informatics*, pages 41–57. Springer.

Mark S Granovetter. 1973. The strength of weak ties. *American journal of sociology*, 78(6):1360–1380.

Adrien Guille, Hakim Hacid, Cecile Favre, and Djamel A Zighed. 2013. Information diffusion in online social networks: A survey. *ACM Sigmod Record*, 42(2):17–28.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of ACL 2016*.

William L Hamilton, Justine Zhang, Cristian Danescu-Niculescu-Mizil, Dan Jurafsky, and Jure Leskovec. 2017. Loyalty in online communities. In *Proceedings of the eleventh International Conference on Web and Social Media*.

Jinyun Ke, Tao Gong, and William SY Wang. 2008. Language change and social networks. *Communications in Computational Physics*, 3(4):935–949.

William Labov. 1972. *Language in the inner city: Studies in the Black English vernacular*. University of Pennsylvania Press.

William Labov. 2001. *Principles of linguistic change: Social factors*. Blackwell.

R Duncan Luce and Albert D Perry. 1949. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116.

Rada Mihalcea and Dragomir Radev. 2011. *Graph-based natural language processing and information retrieval*. Cambridge University Press.

James Milroy and Lesley Milroy. 1985. Linguistic change, social network and speaker innovation. *Journal of linguistics*, 21(2):339–384.

James Milroy and Lesley Milroy. 1987. Belfast: change and variation in an urban vernacular. In *Sociolinguistic patterns in British English*, pages 19–36. E. Arnold.

Lesley Milroy. 1987. *Language and social networks*. Blackwell.

Mark Newman. 2010. *Networks: an introduction*. Oxford University Press.

Jukka-Pekka Onnela, Jari Saramäki, Jorkki Hyvönen, György Szabó, David Lazer, Kimmo Kaski, János Kertész, and A-L Barabási. 2007. Structure and tie strengths in mobile communication networks. *Proceedings of the National Academy of Sciences*, 104(18):7332–7336.

John Paolillo. 1999. The virtual speech community: Social network and language variation on irc. *Journal of Computer-Mediated Communication*, 4(4):JCMC446.

Michał B Paradowski and Łukasz Jonak. 2012. Diffusion of linguistic innovation as social coordination. *Psychology of Language and Communication*, 16(2):131–142.

Everett M Rogers. 2010. *Diffusion of innovations*. Simon and Schuster.

Rahmtin Rotabi and Jon M Kleinberg. 2016. The status gradient of trends in social media. In *Proceedings of the tenth International Conference on Web and Social Media*, pages 319–328.

Rahmtin Rotabi, Cristian Danescu-Niculescu-Mizil, and Jon Kleinberg. 2017a. Competition and selection among conventions. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1361–1370. International World Wide Web Conferences Steering Committee.

Rahmtin Rotabi, Cristian Danescu-Niculescu-Mizil, and Jon Kleinberg. 2017b. Tracing the use of practices through networks of collaboration. In *Proceedings of the 11th International Conference on Web and Social Media*.

Samarth Swarup, Andrea Apolloni, and Zsuzsanna Fagyal. 2011. A model of norm emergence and innovation in language change. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 693–700. International Foundation for Autonomous Agents and Multiagent Systems.

Uriel Weinreich, William Labov, and Marvin I. Herzog. 1968. Empirical foundations for a theory of language change. In *Directions for Historical Linguistics: a Symposium*, pages 95–188. University of Texas Press.

Lilian Weng, Márton Karsai, Nicola Perra, Filippo Menczer, and Alessandro Flammini. 2015. Attention on weak ties in social and communication networks. *arXiv preprint arXiv:1505.02399*.

Jichang Zhao, Junjie Wu, and Ke Xu. 2010. Weak ties: Subtle role of information diffusion in online social networks. *Physical Review E*, 82(1):016105.

Appendix A: Probability of Dissemination Increase

The table in this appendix complements the results presented in Section 5.2.

	k=1		k=2		k= 3		k=4		k=5		k=6	
subreddit	**strong**	**weak**	**strong**	**weak**	**strong**	**weak**	**strong**	**weak**	**strong**	**weak**	**strong**	**weak**
r/Android	0.5	0.45	0.63	0.45	0.65	0.44	0.73	0.45	0.76	0.44	0.81	0.44
r/apple	0.58	0.47	0.7	0.48	0.73	0.48	0.8	0.47	0.8	0.47	0.81	0.47
r/baseball	0.62	0.55	0.73	0.55	0.71	0.54	0.76	0.56	0.75	0.55	0.75	0.54
r/beer	0.51	0.49	0.62	0.49	0.68	0.49	0.69	0.48	0.72	0.48	0.73	0.48
r/boardgames	0.71	0.6	0.83	0.6	0.8	0.58	0.88	0.6	0.7	0.56	0.9	0.6
r/cars	0.63	0.57	0.73	0.58	0.77	0.56	0.85	0.57	0.8	0.53	0.88	0.57
r/FinalFantasy	0.61	0.58	0.59	0.57	1.0	0.56	–	–	–	–	–	–
r/Guitar	0.57	0.53	0.7	0.54	0.7	0.52	0.81	0.53	0.89	0.53	0.77	0.53
r/harrypotter	0.53	0.51	0.5	0.5	1.0	0.51	–	–	–	–	–	–
r/hockey	0.74	0.61	0.8	0.61	0.89	0.62	0.76	0.58	0.83	0.61	0.94	0.62
r/Liverpool	0.61	0.52	0.61	0.51	0.67	0.53	0.67	0.49	–	–	–	–
r/Patriots	0.65	0.64	0.62	0.63	0.6	0.65	0.67	0.65	–	–	–	–
r/pcgaming	0.68	0.58	0.78	0.58	0.74	0.56	0.85	0.58	0.87	0.58	0.88	0.58
r/photography	0.65	0.57	0.73	0.57	0.82	0.57	0.84	0.58	0.75	0.57	0.73	0.57
r/pokemon	0.54	0.48	0.66	0.48	0.69	0.48	0.69	0.48	0.71	0.47	0.71	0.47
r/poker	0.57	0.57	0.7	0.57	1.0	0.57	–	–	–	–	–	-
r/reddevils	0.56	0.54	0.54	0.53	0.6	0.52	0.6	0.5	0.4	0.5	1.0	0.49
r/running	0.6	0.55	0.67	0.56	0.71	0.55	0.82	0.55	1.0	0.55	1.0	0.54
r/StarWars	0.61	0.5	0.77	0.51	0.82	0.51	0.88	0.5	0.9	0.5	0.91	0.5
r/subaru	0.49	0.53	0.53	0.51	0.92	0.52	0.8	0.48	–	–	–	–

Table 2: Probability of increase in dissemination of a linguistic innovation after being used by a **strong-tie** or a **weak-tie** user for k consecutive months. Missing values indicate no such condition was found in a community.

Appendix B: Detailed Results on Success Prediction

The following table gives detailed results for the prediction task described in Section 6. Statistical significance between tie-strength features and the weighted baseline is computed with the dependent t-test for paired samples implemented by Python's `scipy` package on results from 100 runs.

		k=3			k=6			k=12			k=24		
subreddit		**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Android	**t**	0.49	0.5	0.49	0.55	0.57	0.56	0.59	0.59	0.59	0.68	0.69	0.68
	b	0.37	0.38	0.37	0.38	0.38	0.38	0.4	0.41	0.4	0.39	0.39	0.39
apple	**t**	0.64$^{\#}$	0.6$^{\#}$	0.62$^{\#}$	0.72	0.73	0.72	0.8	0.85	0.82	0.78	0.86	0.82
	b	0.64$^{\#}$	0.62$^{\#}$	0.63$^{\#}$	0.66	0.6	0.63	0.62	0.61	0.61	0.65	0.61	0.63
baseball	**t**	0.63	0.65$^{\#}$	0.64$^{\#}$	0.71	0.79	0.75	0.73	0.89	0.8	0.74	0.89	0.81
	b	0.67	0.67$^{\#}$	0.67$^{\#}$	0.66	0.68	0.67	0.64	0.67	0.65	0.63	0.65	0.64
beer	**t**	0.52$^{\#}$	0.52$^{\#}$	0.52 $^{\#}$	0.58	0.62	0.6	0.59	0.65	0.62	0.64	0.7	0.67
	b	0.52$^{\#}$	0.51$^{\#}$	0.51$^{\#}$	0.51	0.49	0.5	0.49	0.5	0.49	0.52	0.5	0.51
boardgames	**t**	0.56$^{\#}$	0.56$^{\#}$	0.56	0.64	0.65	0.64	0.69	0.74	0.71	0.81	0.85	0.83
	b	0.52$^{\#}$	0.51$^{\#}$	0.51	0.5	0.51	0.5	0.55	0.5	0.52	0.54	0.52	0.53
cars	**t**	0.66	0.66	0.66	0.7	0.76	0.73	0.71	0.82	0.76	0.72	0.85	0.78
	b	0.6	0.61	0.6	0.6	0.61	0.6	0.6	0.6	0.6	0.58	0.59	0.58
Finalfantasy	**t**	0.67$^{\#}$	0.6$^{\#}$	0.63$^{\#}$	0.65$^{\#}$	0.7	0.67	0.76	0.84	0.8	0.79	0.86	0.82
	b	0.65$^{\#}$	0.58$^{\#}$	0.61$^{\#}$	0.63$^{\#}$	0.58	0.6	0.63	0.6	0.61	0.64	0.6	0.62
Guitar	**t**	0.68	0.65	0.66	0.68	0.76	0.72	0.71	0.81	0.76	0.75	0.85	0.8
	b	0.58	0.6	0.59	0.58	0.61	0.59	0.56	0.6	0.58	0.58	0.6	0.59
harrypotter	**t**	0.53$^{\#}$	0.55$^{\#}$	0.54	0.57	0.58	0.57	0.56	0.6	0.58	0.51	0.55	0.53
	b	0.49$^{\#}$	0.51$^{\#}$	0.5	0.47	0.48	0.47	0.48	0.52	0.5	0.45	0.48	0.46
hockey	**t**	0.72	0.75	0.73	0.72	0.79	0.75	0.68$^{\#}$	0.81	0.74	0.74	0.88	0.8
	b	0.64	0.64	0.64	0.68	0.62	0.65	0.66$^{\#}$	0.63	0.64	0.64	0.62	0.63
Liverpool	**t**	0.59	0.59	0.59	0.62	0.64	0.63	0.58$^{\#}$	0.61	0.59	0.69	0.78	0.73
	b	0.53	0.5	0.51	0.53	0.52	0.52	0.54$^{\#}$	0.51	0.52	0.53	0.53	0.53
Patriots	**t**	0.72$^{\#}$	0.74	0.73	0.8	0.81	0.8	0.81	0.88	0.84	0.81	0.92	0.86
	b	0.71$^{\#}$	0.69	0.7	0.64	0.69	0.66	0.67	0.71	0.69	0.65	0.69	0.67
pcgaming	**t**	0.78	0.7$^{\#}$	0.74	0.77	0.83	0.8	0.82	0.86	0.84	0.79	0.89	0.84
	b	0.66	0.66$^{\#}$	0.66	0.68	0.68	0.68	0.67	0.68	0.67	0.68	0.7	0.69
photography	**t**	0.72	0.64	0.68	0.74	0.72	0.73	0.74	0.8	0.77	0.77	0.85	0.81
	b	0.63	0.62	0.62	0.65	0.61	0.63	0.64	0.61	0.62	0.63	0.61	0.62
pokemon	**t**	0.65	0.67	0.66	0.58	0.66	0.62	0.62	0.77	0.69	0.68	0.79	0.73
	b	0.53	0.52	0.52	0.51	0.5	0.5	0.53	0.52	0.52	0.52	0.49	0.5
poker	**t**	0.58$^{\#}$	0.62$^{\#}$	0.6$^{\#}$	0.6$^{\#}$	0.64$^{\#}$	0.62$^{\#}$	0.58	0.63$^{\#}$	0.6$^{\#}$	0.65	0.78	0.71
	b	0.62$^{\#}$	0.6$^{\#}$	0.61$^{\#}$	0.61$^{\#}$	0.59$^{\#}$	0.6 $^{\#}$	0.63	0.59$^{\#}$	0.61$^{\#}$	0.58	0.6	0.59
reddevils	**t**	0.57$^{\#}$	0.54$^{\#}$	0.55$^{\#}$	0.68	0.72	0.7	0.7	0.78	0.74	0.74	0.86	0.8
	b	0.59$^{\#}$	0.58$^{\#}$	0.58$^{\#}$	0.58	0.59	0.58	0.58	0.61	0.59	0.59	0.59	0.59
running	**t**	0.58$^{\#}$	0.55$^{\#}$	0.56$^{\#}$	0.64	0.68	0.66	0.65	0.77	0.7	0.66	0.81	0.73
	b	0.57$^{\#}$	0.58$^{\#}$	0.57$^{\#}$	0.58	0.61	0.59	0.56	0.61	0.58	0.58	0.58	0.58
Starwars	**t**	0.63	0.63$^{\#}$	0.63$^{\#}$	0.57$^{\#}$	0.63$^{\#}$	0.6$^{\#}$	0.6$^{\#}$	0.67	0.63	0.65	0.76	0.7
	b	0.55	0.58$^{\#}$	0.56$^{\#}$	0.6$^{\#}$	0.61$^{\#}$	0.6$^{\#}$	0.56$^{\#}$	0.59	0.57	0.6	0.59	0.59
subaru	**t**	0.69	0.64$^{\#}$	0.66	0.69	0.79	0.74	0.69	0.83	0.75	0.69	0.84	0.76
	b	0.63	0.61$^{\#}$	0.62	0.61	0.61	0.61	0.61	0.61	0.61	0.62	0.59	0.6
Average	**t**	0.63	0.62	0.62	0.7	0.66	0.68	0.68	0.76	0.72	0.71	0.81	0.76
	b	0.58	0.57	0.58	0.58	0.57	0.58	0.58	0.58	0.58	0.58	0.58	0.58

Table 3: Average precision (P), recall (R), and F1-score for the *successful* class over 100 runs with 10-fold cross-validation. k= length of the tie-strength vector used for the prediction (corresponding to number of months); **t / b**= results obtained using tie-strength information and the weighted baseline, respectively. Difference between **t** and **b** is significant ($p < 0.05$) except when marked with $^{\#}$.

Ab Initio: Automatic Latin Proto-word Reconstruction

Alina Maria Ciobanu, Liviu P. Dinu
Faculty of Mathematics and Computer Science, University of Bucharest
Human Language Technologies Research Center, University of Bucharest
alina.ciobanu@my.fmi.unibuc.ro, ldinu@fmi.unibuc.ro

Abstract

Proto-word reconstruction is central to the study of language evolution. It consists of recreating the words in an ancient language from its modern daughter languages. In this paper we investigate automatic word form reconstruction for Latin proto-words. Having modern word forms in multiple Romance languages (French, Italian, Spanish, Portuguese and Romanian), we infer the form of their common Latin ancestors. Our approach relies on the regularities that occurred when the Latin words entered the modern languages. We leverage information from all modern languages, building an ensemble system for proto-word reconstruction. We use conditional random fields for sequence labeling, but we conduct preliminary experiments with recurrent neural networks as well. We apply our method on multiple datasets, showing that our method improves on previous results, having also the advantage of requiring less input data, which is essential in historical linguistics, where resources are generally scarce.

1 Introduction

Proto-language reconstruction is one of the main concerns of historical linguistics and is central to understanding the evolution of the world's languages. Reconstructing an ancestral language from the modern languages descending from it not only allows an analysis of language change across space and time, but also reveals information about the historical relationships between languages. The traditional process of reconstructing ancient languages is called the comparative method (Anttila, 1989) and relies on cognates (words in different languages descending from a common proto-word). The main idea of the comparative method is to perform a property-based comparison on multiple sister languages in order to infer properties of their common ancestor. For a long period, the comparative reconstruction has been a time-consuming manual process that required a large amount of intensive work. The first step of the process consists in identifying cognate pairs.

Identifying cognates and borrowings by means of computational approaches has attracted considerable attention in recent years (Hall and Klein, 2010; Tsvetkov et al., 2015; Ciobanu and Dinu, 2015). However, few studies went beyond this step, and beyond the comparative method, to automate the process of proto-language reconstruction (Oakes, 2000; Bouchard-Côté et al., 2013; Atkinson, 2013). Reconstructing proto-words is a challenging task. While the main hypothesis in this research problem is that there are regularities and patterns in how words evolved from the ancestor language to its modern daughter languages, there are also words which diverged significantly from their ancestor. Take, for example, the Latin word *umbilicu(lu)s*. It evolved into *buric* (Romanian), *nombril* (French), and *umbigo* (Portuguese).

One of the best approaches to proto-word reconstruction (Bouchard-Côté et al., 2013) relies on an analogy with reconstructing the genealogy of the species from genetic sequences in biology. This approach requires an existing phylogenetic tree and the phonetic transcripts of the words, to infer the ancient word forms based on probability estimates for all the possible sound changes on each branch of the tree.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1604–1614
Santa Fe, New Mexico, USA, August 20-26, 2018.

Contributions of the present work. In this paper, we investigate proto-word reconstruction starting from modern word forms. The main goal of our study is to achieve state-of-the-art performance in proto-word reconstruction using less resources than in previous studies. We aim at providing a tool for linguists to use in their research on endangered and extinct languages. Our hypothesis is that orthographic changes represent sound correspondences to a large extent, and thus we attempt to reconstruct proto-words from the orthographic form of the modern words.

We address this problem in two steps. Firstly, given cognate pairs in multiple modern languages and their common ancestors, we apply a word production method based on sequence labeling for reconstructing proto-words. We apply the method on each modern language individually. Secondly, we propose several ensemble methods for combining information from multiple word production systems, with the purpose of joining the best productions from all modern languages. Through experiments, we show that our best ensemble system outperforms previous results (Bouchard-Côté et al., 2007). The novelty of our approach is enhancing the sequence alignment system with an additional step of reranking. We also compute the results of an oracle, which shows the potential of this approach. The proposed methods have the advantage of not requiring phonetic transcripts and other data besides the training word pairs (such as a corpus in the target language, as some of the existing methods require); external information regarding language evolution is difficult to obtain for some languages, and this method can be applied on low-resourced and endangered languages. Moreover, as opposed to previous methods, our system is able to reconstruct proto-words even from incomplete cognate sets (cognate pairs in multiple modern languages descending from a common proto-language).

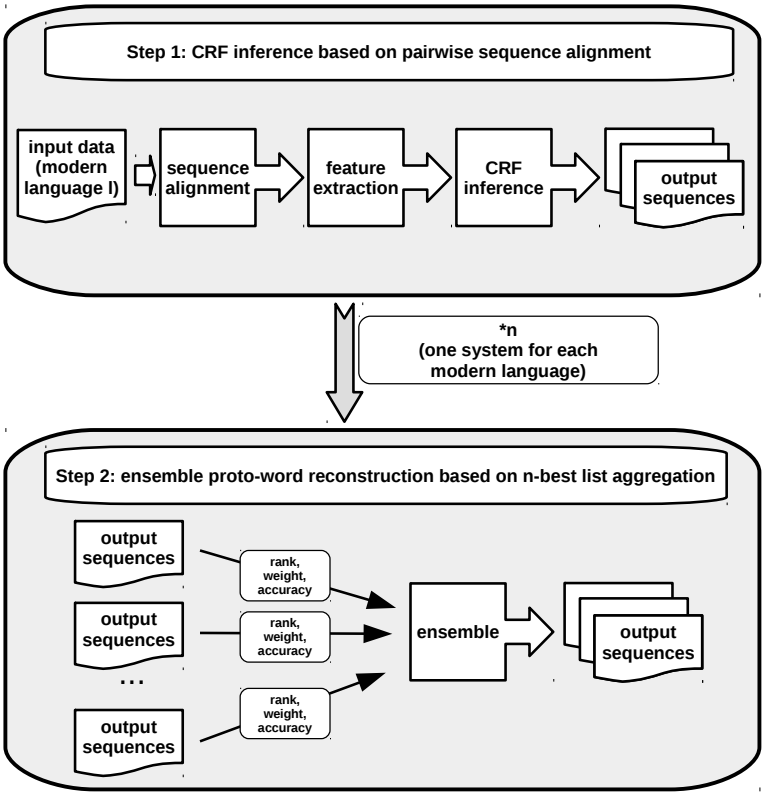


Figure 1: Methodology for proto-word reconstruction.

2 Methodology and Algorithm

In this section we introduce a new technique for proto-word reconstruction. Given cognate sets in several Romance languages, we infer the form of their common Latin ancestors. Our goal is to produce a tool that would provide support in historical linguistics, where the produced proto-words would be further analyzed by domain experts. We propose an approach based on conditional random fields. For each

modern language independently, we first apply a sequence labeling method that produces the form of the Latin ancestors. For this first step, we only require *(word, proto-word)* pairs (not cognate sets), as the input data is formed of modern word forms and their ancestors. Then we define several ensemble methods to take advantage of the information provided by all languages, in order to improve performance on Latin proto-word reconstruction. Our methodology is illustrated in Figure 1.

2.1 Conditional Random Fields

From the alignment of related words in the training set, the system learns orthographic patterns for the changes in spelling between each modern language and the proto-language. We apply a sequence labeling method to infer the form of the Latin proto-words for the modern words in the test set. The method that we employ is based on sequence labeling, an approach that has been proven useful in generating transliterations (Ganesh et al., 2008; Ammar et al., 2012) and in cognate production (Ciobanu, 2016).

In our case, the words in the modern languages are the sequences, and their characters are the tokens. Our purpose is to obtain, for each input word, a sequence of characters that compose its proto-word. To this end, we use conditional random fields (CRFs) (Lafferty et al., 2001). As features for the CRF system, we use character n-grams from the input words, extracted from a fixed window w around the current token.

2.2 Pairwise Sequence Alignment

To align pairs of words, we experimented with two alignment methods that have been proven useful in natural language processing and computational biology: the alignment method based on profile hidden Markov models proposed by Bhargava and Kondrak (2009) and the Needleman-Wunsch global alignment algorithm (Needleman and Wunsch, 1970).

Since the alignment is only a pre-processing step for our task, we evaluate the alignment methods by the downstream results, i.e., the accuracy obtained by the CRF system when using one or the other alignment method. We observed that the results obtained with the two alignment methods were very similar, slightly better for the latter. Thus, we report the results obtained with the Needleman Wunsch alignment algorithm.

The alignment algorithm uses, in our case, words as input sequences and a basic substitution matrix, which gives equal scores to all substitutions, disregarding diacritics (e.g., we ensure that *e* and *è* are matched). For example, for the Romanian word *frumos* (meaning *beautiful*) and its Latin ancestor *formosus*, the alignment is as follows:

```
f - r u m o s - -
f o r - m o s u s
```

For each character in the modern word (after the alignment), the associated label is the character which occurs on the same position in its proto-word. In the case of insertions, we add the new character to the previous label, because there is no input character in the source language to which we could associate the inserted character as label. We account for affixes separately: for each input word, we add two more characters B and E, marking the beginning and the end of the word. The characters that are inserted in the target word at the beginning or at the end of the word are associated to these special characters. In order to reduce the number of labels, we replace the label with $\star$ for input tokens that are identical to their labels. Thus, for the previous example, the labels are as follows:

```
B  f  r  u  m  o  s  E
↓  ↓  ↓  ↓  ↓  ↓  ↓  ↓
*  fo *  -  *  *  *  usE
```

2.3 Ensembles

Ensembles of classifiers combine the results of multiple classifiers, in order to improve performance. In our case, the classifiers use different input data: we train a classifier for each modern language and

combine their results, in order to obtain Latin proto-words with higher accuracy. Our goal is to improve the performance of the system by taking advantage of the information provided by all languages.

Each classifier produces, for each input word, an n-best list of possible proto-words. To combine the outputs of the classifiers, we propose four fusion methods based on the ranks in the n-best lists and the probability estimates provided by the individual classifiers for each possible production.

Given a cognate set, we combine the n-best lists previously obtained to compute a joint n-best list that leverages information from all modern languages.

Fusion by rank. We compute an average rank for each production from the n-best lists, as the average rank from all modern languages. We favor words that occur on the first position in multiple n-best lists. In other words, for a given n-best list, we associate weights in a Borda sense (Borda, 1781) to all the words from the list: we give weight *n* to the word produced with the highest confidence, *n - 1* to the second one, and so on, until weight *1* to the *n*-th produced word. For an n-best list L_i and a word u, we denote by $w(u_i)$ the weight of word *u* in L_i; if *u* is not a word from list L_i, then $w(u_i) = 0$. Given *k* n-best lists (one for each modern language) and a produced word u, we define the rank weight of u over the *k* n-best lists as:

$$w_r(u) = (1/k) * \sum_{i=1}^{k} w(u_i). \tag{1}$$

Fusion by rank and accuracy. Starting from the previous fusion method, we also take into account the training accuracy for each language. We give more importance to the *vote* of the languages that obtained a better performance, multiplying the weight by the training accuracy. In other words, for each n-best list L_i, let $\pi(i)$ be the training accuracy for language i (COV_{10}). Given *k* n-best lists (one for each modern language) and a produced word u, we define the rank-accuracy weight of u over the *k* n-best lists as:

$$w_{ra}(u) = (1/k) * \sum_{i=1}^{k} w(u_i)\pi(i). \tag{2}$$

Fusion by weight. We compute an average confidence score for each production from the n-best lists, using the confidence score reported by the sequence labeling system. We rerank the productions based on the average confidence score. Given *k* n-best lists (one for each modern language), a produced word u, and $w(u_i)$ the confidence score of the sequence labeling system in list L_i, we define the confidence weight of u over the *k* n-best lists as:

$$w_c(u) = (1/k) * \sum_{i=1}^{k} w(u_i). \tag{3}$$

This is similar to the first fusion method, but uses the sequence labeling system's weights instead of the weights obtained from the ranking in the n-best lists.

Fusion by weight and accuracy. Starting from the previous fusion method, we also take into account the training accuracy for each language. We give more importance to the *vote* of the languages that obtained a better performance, multiplying the score by the training accuracy. Given *k* n-best lists (one for each modern language), a produced word u, and $w(u_i)$ the confidence score of the sequence labeling system in list L_i, we define the confidence-accuracy weight of u over the *k* n-best lists as:

$$w_{ca}(u) = (1/k) * \sum_{i=1}^{k} w(u_i)\pi(i). \tag{4}$$

This is similar to the second fusion method, but uses the sequence labeling system's weights instead of the weights obtained from the ranking in the n-best lists.

The output of each ensemble is a new n-best list in which the words are sorted in descending order of their computed weights, as described in Equations 1-4.

Oracle. An oracle classifier is an ensemble method that produces the correct result if any of the comprising classifiers produces the correct result. The purpose of an oracle is to determine the upper limit of an ensemble. The probability of obtaining correct results from an oracle is the following (Kuncheva et al., 2003):

$$P(oracle) = 1 - P(all\_wrong) \tag{5}$$

3 Experimental Setup

In this section we describe the experimental setup that we employed to assess the performance and to analyze the proposed method for proto-word reconstruction.

3.1 Datasets

We use datasets of Romance languages with Latin ancestors:

Dataset 1. The dataset proposed by Bouchard-Côté et al. (2007). It contains 585 complete cognate sets in three Romance languages (Spanish, Portuguese, Italian) and their common Latin ancestors. It is provided in two versions: orthographic and phonetic (IPA transcriptions). This dataset allows us to compare our results with a previous state-of-the-art method for proto-word reconstruction.

Dataset 2. The dataset proposed by Reinheimer Ripeanu (2001). It consists of 1,102 cognate sets in five Romance languages (Spanish, Italian, Portuguese, French, Romanian) and their common Latin ancestors. Note that not all of these cognate sets are complete (that is, for some of them there are not cognates provided in all five modern languages).

Dataset 3. The dataset proposed by Ciobanu and Dinu (2014b) and previously used for cognate detection (Ciobanu and Dinu, 2014a). It contains 3,218 complete cognate sets in five Romance languages (Spanish, Italian, Portuguese, French, Romanian) and their common Latin ancestors. Below we provide an example of a cognate set from this dataset:

vehicul (**Ro**) | véhicule (**Fr**) | veicolo (**It**) | vehículo (**Es**) | veículo (**Pt**) | vehiculum (**Lat**)

3.2 Task Setup

We split each dataset in subsets for train, dev and test (3:1:1 ratio). For inferring the form of the proto-words, we use the CRF implementation provided by the Mallet toolkit (McCallum, 2002). For parameter tuning, we perform a grid search for the number of iterations in $\{1, 5, 10, 25, 50, 100\}$ and for the size of the window w in $\{1, 2, 3, 4, 5\}$.

3.3 Evaluation Measures

Following previous work in this area (Bouchard-Côté et al., 2007; Beinborn et al., 2013), we use the evaluation measures listed below to assess the performance of our method:

Average edit distance. To assess how close the productions are to the correct form of the proto-words, we report the edit distance between the produced words and the gold standard. We report both the unnormalized and the normalized edit distance. For normalization in the [0,1] interval, we divide the edit distance by the length of the longest string.

Coverage. Also known as *top n accuracy*, the coverage is a relaxed metric which computes the percentage of input words for which the n-best output list contains the correct proto-word (the gold standard). We use n $\in \{1, 5, 10\}$. The practical importance of analyzing the top n results is that we offer a filter to narrow down the possible forms of the output words to a low-dimensional list, that linguists can analyze, aiming to identify the correct form of the proto-word. Note that the coverage for n = 1 is the well-known measure accuracy.

Mean reciprocal rank. The mean reciprocal rank is an evaluation measure which applies to systems that produce an ordered output list for each input instance. Given an input word, the higher the position of its correct proto-word in the output list, the higher the mean reciprocal rank value:

$$MRR(w_i) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{rank_i}, \tag{6}$$

where m is the number of input instances, and $rank_i$ is the position of w_i's proto-word in the output list. If w_i's correct proto-word is not in the output list, we consider the reciprocal rank 0.

4 Results

In this section we analyze and compare the performance of our systems in different scenarios. We also compare our results with previous work.

4.1 Results Analysis

In Table 1 we report the results of our individual systems (one for each modern language) and the ensemble results. For individual experiments, Italian obtains the lowest average edit distance on all datasets.
For ensembles, we experimented with the four fusion methods described in the previous section, but report only the best performing one. Out of the four proposed fusion methods, the first two lead to similar results, that are superior to the other two. The best-performing ensemble uses the second fusion method, which assigns scores based on the rank in the n-best lists and the training accuracy for each individual system. We also tried applying the ensembles on language subsets (that is, not to take all modern languages into account at once). We investigated all combinations, and in the majority of cases using all modern languages lead to the highest performance among all ensembles.
In Table 2 we show an example of our systems' output n-best lists. This example illustrates how the ensemble can improve over the individual classifiers, by ranking the correct production higher than all the other systems. For all datasets we obtained performance improvements for proto-word reconstruction when we combined individual results using ensembles. As expected, the highest performance was obtained by the oracle classifiers. The results show the high potential of the ensemble methods for proto-word reconstruction.
The average edit distance of our best-performing ensemble, on Dataset 3, is 1.07, meaning that, on average, the proto-word reconstructions obtained by the system are a little more than one character different from the correct proto-words. Furthermore, the correct proto-word is listed among the 5-best list productions of our system in 70% of the cases (increasing to 74% for 10-best lists). These results are encouraging, having in mind the purpose of our system: to be a tool to be used by linguists (not to substitute the work of the experts).
Overall, we notice that the results are significantly better for Datasets 2 and 3 than for Dataset 1. One possible explanation is the nature of the dataset: while Datasets 2 and 3 are built based on the etymologies of the words (that is, the genetic relationships are taken into account), for Dataset 1 the cognacy decisions have been made based on an edit distance threshold between the words (Bouchard-Côté et al., 2007). To test this assumption, we ran an additional experiment, training the system on a subset of Dataset 3, having the same size as Dataset 1. The performance was close to that reported for Dataset 3 in Table 1, confirming that it is the nature of the dataset rather than its size that influences the results.

4.2 Additional Experiments: Neural Word Production

We performed additional experiments using a recurrent neural network (RNN) system with an encoder-decoder architecture instead of a CRF. RNNs have been proven useful in many applications, and are suitable for sequence labeling problems. However, they require large amounts of training data. In historical linguistics in general, and in the problem of proto-word reconstruction in particular, the resources are often scarce. Thus, we were interested to find out if RNNs can outperform the CRF system. We experimented with an encoder-decoder system with two long short-term memory (LSTM) layers, with stochastic gradient descent (SGD) optimization and a global attention mechanism (Luong et al.,

Language	EDIT	COV_1	COV_5	COV_{10}	MRR
Italian	2.45 (0.29)	0.14	0.32	0.35	0.22
Spanish	2.51 (0.29)	0.15	0.21	0.23	0.18
Portuguese	2.61 (0.30)	0.16	0.24	0.31	0.21
Ensemble	**2.31 (0.27)**	**0.22**	**0.32**	**0.42**	**0.27**
Oracle	1.76 (0.20)	0.28	0.41	0.47	0.34

(a) Dataset 1 orthographic (Bouchard-Côté et al. (2007))

Language	EDIT	COV_1	COV_5	COV_{10}	MRR
Italian	2.52 (0.29)	0.16	0.28	0.32	0.21
Spanish	2.61 (0.30)	0.12	0.22	0.25	0.17
Portuguese	2.95 (0.34)	0.07	0.17	0.22	0.13
Ensemble	**2.28 (0.26)**	**0.14**	**0.32**	**0.36**	**0.21**
Oracle	1.93 (0.22)	0.23	0.36	0.39	0.30

(b) Dataset 1 phonetic (Bouchard-Côté et al. (2007))

Language	EDIT	COV_1	COV_5	COV_{10}	MRR
Italian	1.57 (0.24)	0.25	**0.52**	**0.55**	0.37
Spanish	1.78 (0.27)	0.22	0.35	0.39	0.28
Portuguese	1.76 (0.28)	0.19	0.34	0.39	0.26
Romanian	2.12 (0.32)	0.18	0.31	0.36	0.25
French	2.31 (0.35)	0.13	0.24	0.30	0.18
Ensemble	**1.55 (0.23)**	**0.29**	0.49	**0.55**	**0.38**
Oracle	0.95 (0.14)	0.43	0.60	0.66	0.51

(c) Dataset 2 (Reinheimer Ripeanu (2001))

Language	EDIT	COV_1	COV_5	COV_{10}	MRR
Italian	1.12 (0.14)	0.46	0.62	0.66	0.54
Spanish	1.31 (0.16)	0.42	0.59	0.61	0.49
Portuguese	1.30 (0.16)	0.41	0.58	0.61	0.49
Romanian	1.36 (0.16)	0.43	0.61	0.64	0.51
French	1.52 (0.18)	0.43	0.57	0.61	0.50
Ensemble	**1.07 (0.13)**	**0.50**	**0.70**	**0.74**	**0.59**
Oracle	0.65 (0.08)	0.66	0.77	0.79	0.71

(d) Dataset 3 (Ciobanu and Dinu (2014a))

Table 1: Proto-word reconstruction for Latin proto-words. The first column indicates the modern language (or ensemble) that we used for training. For ensembles we report the results for the best performing ensemble. We report the average edit distance between the produced form and the correct form of the proto-word (**EDIT**) un-normalized (and in parantheses the normalized version), the coverage (**COV** for n $\in \{1, 5, 10\}$) and the mean reciprocal rank (**MRR**).

2015). We used the RNN implementation provided by TensorFlow (Luong et al., 2017). We experimented with Word2Vec character embeddings as features, and also with embeddings extracted from the aligned words (similar to the features used for the CRF system).

The results of the RNN system are lower than those of the CRF system (for example, COV_{10} is about

Language	Word	5-best productions
French	voisin	vosinum, vosnum, vosine, vosinus, voiinum
Italian	vicino	vicinum, **vicinus**, vicenum, vicenus, vicnum
Portuguese	vizinho	vizinus, vizinum, **vicinus**, vizinium, vizinhum
Spanish	vecino	vecinum, vecinus, vicinum, vecenum, **vicinus**
Romanian	vecin	vicenus, vicenum, **vicinus**, vicinum, vecenus
Ensemble	all words	**vicinus**, vicinum, vicenus, vicenum, vecinus

Table 2: Proto-word reconstruction example for the Latin proto-word **vicinus** (meaning **neighbor**). The correct productions are highlighted in bold.

0.10 for Dataset 1, while the CRF system obtains about 0.15 on the same dataset). The results are not included here, since they do not provide better performance. We leave for future work experimenting with additional features and setups, in order to compensate for the lack of training data.

4.3 Error Analysis

In this subsection we perform a brief analysis of the errors of our ensemble systems. The purpose of this step is to understand where the systems are not able to learn the correct production rules, in order to improve them in the future.

Looking at the incorrect productions that have one character different from the correct proto-word, we notice that sometimes the final vowel is mistaken. Most commonly, *um* instead of *us*: *serenum* instead of *serenus*, *cantum* instead of *cantus*, *novum* instead of *novus*. Another one-character mistake is, sometimes, failing to double a consonant. For example, *ll* or *ss*: *colapsus* instead of *collapsus*, *intervalum* instead of *intervallum*, *disociatio* instead of *dissociatio*, *esentia* instead of *essentia*.

For productions that have two characters different from the correct proto-word , we notice the following patterns in the incorrect productions: sometimes the character *f* is mistakenly obtained instead of *ph*: *asfaltus* instead of *asphaltus*, *eufonia* instead of *euphonia*, *diafragma* instead of *diaphragma*. Another interesting pattern is obtaining the suffix *a* instead of *us*: *citrina* instead of *citrinus*, *alba* instead of *albus*. When this occurs for adjectives, the productions are not incorrect words in Latin; we obtain the feminine form instead of the masculine.

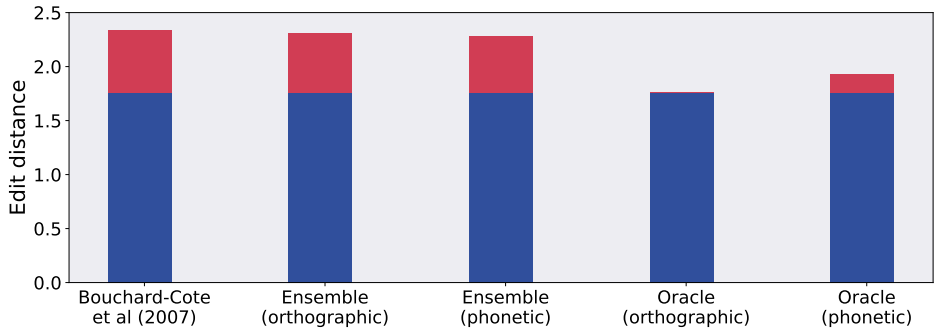


Figure 2: Comparison between previous work (Bouchard-Côté et al., 2007) and our ensembles and oracles on Dataset 1.

4.4 Comparison with Previous Work

Dataset 1 allows us a fair comparison with the state-of-the-art method proposed by Bouchard-Côté et al. (2007), as illustrated in Figure 2. On the same dataset, the authors report 2.34 average edit distance

between the produced words and the gold standard, when recreating Latin proto-words using the phonetic transcriptions. On the same dataset, we obtain better results on both the orthographic and the phonetic version of the dataset. The best of our systems (ensemble with fusion method based on the rank in the n-best lists and the training accuracy) obtains 2.28 average edit distance on the phonetic version of the dataset and 2.31 average edit distance on the orthographic version. The oracle obtains 1.76 average edit distance on the orthographic version of the dataset and 1.93 on the phonetic version.
The improvement of our systems is significant because we are able to obtain these results with less data than in previous experiments – which in historical linguistics is essential, as resources are most of the time scarce: the system is able to perform well even when not having the phonetic transcripts of the words.

5 Related Work

Researchers have been continuously interested in language derivation (Pagel et al., 2013). One of the main concerns in historical linguistics is language change across space and time (Rama and Borin, 2014). The first attempts to address this problem focused on regular sound correspondences to construct modern forms of the words, given a proto-language, or vice-versa. Traditionally, proto-language reconstruction has been investigated with comparative linguistics instruments (Campbell, 1998) and required a manual process and intensive work from linguistics and domain experts. Modern approaches impose the use and development of quantitative and computational methods in this field (McMahon et al., 2005; Heggarty, 2012; Atkinson, 2013), or even cross-disciplinary methods (such as those borrowed from biology).
Some of the early studies on partially automating proto-language reconstruction belong to Covington (1998) – investigating multiple alignment for historical comparison, and Kondrak (2002) – proposing, among others, methods for cognate alignment and identification.
More recent approaches addressed the complete automation of the reconstruction process. Oakes (2000) proposed two systems, Jakarta and Prague, that combined cover the steps of the comparative method for proto-language reconstruction. Another probabilistic approach belongs to Hall and Klein (2010), who obtained an average edit distance of 3.8 on the dataset of Romance languages proposed by Bouchard-Côté et al. (2009). Bouchard-Côté et al. (2013) used probabilistic models to trace language change in the Austronesian languages, based on a given phylogenetic tree.

6 Conclusions

In this paper, we proposed an automatic method for proto-word reconstruction. We applied our method on multiple datasets of Romance languages, in order to reconstruct Latin proto-words.
We used an approach based on sequence labeling and sequence alignment, combining the results of individual systems using ensembles. We obtained fairly good results (our best performance is 70% top 5 accuracy), and improved over previous work in this area. Our method has the advantage of requiring less input data than previous methods, and also accepting incomplete data, which is essential in historical linguistics, where resources are scarce.
We conclude that leveraging information from multiple modern languages, in ensemble systems, improves the performance on this task, producing n-best list of proto-words to be further analyzed by linguists and to aid in the process of comparative reconstruction for endangered or extinct languages.
As future work, we intend to refine the fusion methods for the ensemble classifiers – as the oracle results showed the high potential of the approach – and to evaluate our method on other datasets that cover more languages (for example, the Swadesh lists or the Austronesian basic vocabulary database (Greenhill et al., 2008)). We also intend to study rhotacism (Campbell, 1998), to investigate further ways of improving the performance of our CRF systems, and to enhance the RNN system even with little data available.

Acknowledgments

We thank the anonymous reviewers for their helpful and constructive comments. The contribution of the authors to this paper is equal. Research supported by UEFISCDI, project number 53BG/2016.

References

Waleed Ammar, Chris Dyer, and Noah A Smith. 2012. Transliteration by Sequence Labeling with Lattice Encodings and Reranking. In *Proceedings of the 4th Named Entity Workshop*, pages 66–70.

Raimo Anttila. 1989. *Historical and Comparative Linguistics*. Benjamins.

Quentin D. Atkinson. 2013. The Descent of Words. *Proceedings of the National Academy of Sciences*, 110(11):4159–4160.

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2013. Cognate Production using Character-based Machine Translation. In *Proceedings of IJCNLP 2013*, pages 883–891.

Aditya Bhargava and Grzegorz Kondrak. 2009. Multiple Word Alignment with Profile Hidden Markov Models. In *Proceedings of NAACL-HLT 2009, Companion Volume: Student Research Workshop and Doctoral Consortium*, pages 43–48.

Jean Charles Borda. 1781. Memoire sur les elections au scrutin. *Memoires de lAcademie Royale des Sciences de Paris pour l'Anne 1781*, pages 657–665.

Alexandre Bouchard-Côté, Percy Liang, Thomas Griffiths, and Dan Klein. 2007. A Probabilistic Approach to Diachronic Phonology. In *Proceedings of EMNLP 2007*, volume 12, pages 887–896.

Alexandre Bouchard-Côté, Thomas L. Griffiths, and Dan Klein. 2009. Improved Reconstruction of Protolanguage Word Forms. In *Proceedings of NAACL 2007*, volume 7, pages 65–73.

Alexandre Bouchard-Côté, David Hall, Thomas L. Griffiths, and Dan Klein. 2013. Automated Reconstruction of Ancient Languages Using Probabilistic Models of Sound Change. *Proceedings of the National Academy of Sciences*, 110(11):4224–4229.

Lyle Campbell. 1998. *Historical Linguistics. An Introduction*. MIT Press.

Alina Maria Ciobanu and Liviu P. Dinu. 2014a. Automatic Detection of Cognates Using Orthographic Alignment. In *Proceedings of ACL 2014, Volume 2: Short Papers*, pages 99–105.

Alina Maria Ciobanu and Liviu P. Dinu. 2014b. Building a Dataset of Multilingual Cognates for the Romanian Lexicon. In *Proceedings of LREC 2014*, pages 1038–1043.

Alina Maria Ciobanu and Liviu P. Dinu. 2015. Automatic Discrimination between Cognates and Borrowings. In *Proceedings of ACL 2015, Volume 2: Short Papers*, pages 431–437.

Alina Maria Ciobanu. 2016. Sequence Labeling for Cognate Production. In *Proceedings of KES 2016*, pages 1391–1399.

Michael A. Covington. 1998. Alignment of Multiple Languages for Historical Comparison. In *Proceedings of ACL 1998, Volume 1*, pages 275–279.

Surya Ganesh, Sree Harsha, Prasad Pingali, and Vasudeva Verma. 2008. Statistical Transliteration for Cross Language Information Retrieval Using HMM Alignment Model and CRF. In *Proceedings of CLIA 2008*, pages 42–47.

Simon J Greenhill, Robert Blust, and Russel D. Gray. 2008. The Austronesian Basic Vocabulary Database: From Bioinformatics to Lexomics. *Evolutionary Bioinformatics*, 4:271–283.

David Hall and Dan Klein. 2010. Finding Cognate Groups Using Phylogenies. In *Proceedings of ACL 2010*, pages 1030–1039.

Paul Heggarty. 2012. Beyond Lexicostatistics: How to Get More out of "Word List" Comparisons. In *Quantitative Approaches to Linguistic Diversity: Commemorating the Centenary of the Birth of Morris Swadesh*, pages 113–137. Benjamins.

Grzegorz Kondrak. 2002. *Algorithms for Language Reconstruction*. Ph.D. thesis, University of Toronto.

Ludmila I. Kuncheva, Christopher J. Whitaker, Catherine A. Shipp, and Robert P. W. Duin. 2003. Limits on the Majority Vote Accuracy in Classifier Fusion. *Pattern Analysis and Applications*, 6(1):22–31.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of ICML 2001*, pages 282–289.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of EMNLP 2015*, pages 1412–1421.

Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. 2017. Neural Machine Translation (seq2seq) Tutorial. *https://github.com/tensorflow/nmt*.

Andrew Kachites McCallum. 2002. MALLET: A Machine Learning for Language Toolkit. http://mallet.cs.umass.edu.

April McMahon, Paul Heggarty, Robert McMahon, and Natalia Slaska. 2005. Swadesh Sublists and the Benefits of Borrowing: an Andean Case Study. *Transactions of the Philological Society*, 103(2):147–170.

Saul B. Needleman and Christian D. Wunsch. 1970. A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. *Journal of Molecular Biology*, 48(3):443–453.

Michael P. Oakes. 2000. Computer Estimation of Vocabulary in a Protolanguage from Word Lists in Four Daughter Languages. *Journal of Quantitative Linguistics*, 7:233–243.

Mark Pagel, Quentin D Atkinson, Andreea S Calude, and Andrew Meade. 2013. Ultraconserved Words Point to Deep Language Ancestry across Eurasia. *Proceedings of the National Academy of Sciences*, 110(21):8471–8476.

Taraka Rama and Lars Borin. 2014. Comparative Evaluation of String Similarity Measures for Automatic Language Classification. In George K. Mikros and Jn Macutek, editors, *Sequences in Language and Text*. De Gruyter Mouton.

Sanda Reinheimer Ripeanu. 2001. *Lingvistica Romanica: Lexic, Morfologie, Fonetica*. Ed. All. Bucuresti.

Yulia Tsvetkov, Waleed Ammar, and Chris Dyer. 2015. Constraint-Based Models of Lexical Borrowing. In *Proceedings of NAACL-HLT 2015*, pages 598–608.

A Computational Model for the Linguistic Notion of Morphological Paradigm

Miikka Silfverberg and **Ling Liu** and **Mans Hulden**
Department of Linguistics
University of Colorado
first.last@colorado.edu

Abstract

In supervised learning of morphological patterns, the strategy of generalizing inflectional tables into more abstract paradigms through alignment of the longest common subsequence found in an inflection table has been proposed as an efficient method to deduce the inflectional behavior of unseen word forms. In this paper, we extend this notion of morphological 'paradigm' from earlier work and provide a formalization that more accurately matches linguist intuitions about what an inflectional paradigm is. Additionally, we propose and evaluate a mechanism for learning full human-readable paradigm specifications from incomplete data—a scenario when we only have access to a few inflected forms for each lexeme, and want to reconstruct the missing inflections as well as generalize and group the witnessed patterns into a model of more abstract paradigmatic behavior of lexemes.

1 Introduction

Most work in phonology and morphology assumes that the ability to inflect previously unseen or unheard word forms is through a model of analogy (Blevins and Blevins, 2009) to known forms. In languages with poor inflectional morphology, this often manifests itself in the capacity to inflect previously unwitnessed lexemes, e.g. **wug** $\mapsto$ **wugs** by analogy to some other lexeme, perhaps a noun like **bug** $\mapsto$ **bugs**. A closely related problem, relevant for languages that have complex inflectional morphology and where a single part-of-speech can be inflected in perhaps thousands of ways, is the capacity to inflect a lexeme with a previously unheard combination of inflectional features. For example, Finnish nouns are inflected in 2,253 different ways,[1] and no speaker will have witnessed all variants for any given noun. This capacity to fill in missing forms has been dubbed the **paradigm cell filling problem** (PCFP) (Ackerman et al., 2009).

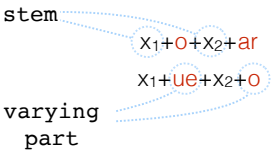
The frequency of references to the nebulous label 'analogy' and its responsibility for human performance in these tasks stands in contrast to the dearth of proposed exact formal or computational models of analogical word-formation in the literature. Even in second-language teaching material, speakers are assumed to possess the capacity to directly generalize a pattern from a collection of inflected forms. Table 1 shows a partial inflection table for the Spanish verb **tostar** ('to toast') given in the resource *501 Spanish Verbs* (Kendris and Kendris, 2007). By contrast, the verb entry for **colar** ('to strain') contains no table at all, but simply provides a pointer to the inflection table for **tostar**. In other words, it is presupposed that, for example, given the analogy **tostar:tuesto::colar:x**, the astute reader can solve for **x** without undue problems, though no explicit procedure for doing so is given.[2]

One formalization of this implicit procedure is offered in Ahlberg et al. (2014) and Hulden (2014). In those works, a collection of inflected forms is first subjected to identifying the *longest common subsequence* (LCS) among the given forms. For example, the set of words {**tostar, tuesto, ...**} would contain

[1] http://www.ling.helsinki.fi/~fkarlsso/genkau2.html

[2] Note that to perform this 'analogy', it is not sufficient to merely identify a monolithic stem and possible suffix in the pair **tostar/tuesto** and add the discovered suffix to the presupposed stem of **colar**. This strategy works for simple cases like **bug:bugs::wug:wugs**, but obviously cannot be the basis for general analogical modeling in morphology.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1615–1626
Santa Fe, New Mexico, USA, August 20-26, 2018.



(a) Stem and varying part extracted for word forms **tostar** and **tuesto**.

			P1	P2
NOM	koira	mäyrä	x_1	x_1
ADE	koiralla	mäyrällä	x_1+**ll**a	x_1+**ll**ä
ELA	koirasta	mäyrästä	x_1+**st**a	x_1+**st**ä

(b) Paradigms extracted by LCS do not correspond to linguistic generalizations because they do not factor out regular phonological alternations.

Figure 1: Illustration of the LCS method for paradigm extraction in 1a. In 1b, we show the inability of the LCS method to factor out vowel harmony.

the LCS **tst**, as that subsequence occurs in every form. After locating this, the generalization is set up so that the LCS part is defined as a discontinuous *stem* that recurs throughout the inflectional paradigm, while the remainder can vary. In the above example, this would yield the generalization in Figure 1a where the $\mathbf{x_i}$s represent (in this case) the discontinuous 'stem'. Fitting another word such as **colar** into the first form would implicitly yield that the stem of that word is $\mathbf{x_1} = \mathbf{c}$, $\mathbf{x_2} = \mathbf{l}$. Having identified the stem, we can now produce the form $\mathbf{x_1}$ + **ue** + $\mathbf{x_2}$ + **o** which becomes **cuelo**, concluding the analogy as **tostar:tuesto::colar:cuelo**.

This strategy works reasonably well for supervised learning of morphological paradigms and provides a human-readable generalization as a result (Ahlberg et al., 2014; Ahlberg et al., 2015; Forsberg and Hulden, 2016). However, we identify two weaknesses in the model. First, the approach undergeneralizes in that it produces separate generalizations of paradigms for what is arguably very similar inflectional behavior. Secondly, it does not offer a method to learn paradigms from partial data—it operates on full inflectional tables—something that would address the paradigm cell filling problem.

To illustrate the first point, consider the two Finnish inflection tables and resulting paradigms in Figure 1b. Here, the only difference between the two learned patterns is that whenever one (P1) has an **a**-segment in its slots, the other (P2) has **ä**. This difference prevents the two patterns from being deemed alike. Naturally, a more thorough linguistic analysis would uncover that this is an instance of back/front vowel harmony dictated by the vowels found in the stem, the sequences **lla/llä** and **sta/stä** in effect being allomorphs of the same morpheme.

In this paper we propose a method that will discover such allomorphy through a purely data-driven approach—i.e. it will not be necessary to postulate any phonological evidence or analysis to be able to discover that several paradigms learned through the LCS method are actually alike and to deduce all the allomorphs. This, as we will show, will substantially cut down on the number of different inflectional paradigms discovered by the algorithm to something very close to that postulated by linguists after thorough phonological and morphological analysis.

Further, we propose a method on top of the LCS-paradigm generalization to discover partial paradigms from partially given inflection tables and to collapse these into full-size paradigms that allow us to address the paradigm cell filling problem.

tostar (474)		**colar**
tuesto	tostamos	*[inflect like 474]*
tuestas	tostáis	
tuesta	tuestan	
...	...	

Table 1: Information given in Kendris & Kendris (2007) about Spanish verb inflection. Note the reference from **colar** to **tostar**.

1.1 Terminology

To avoid terminological confusion, we will adhere to some conventions in this paper. First, we make a distinction between an **inflection table** and **paradigm**: an **inflection table** is a concrete manifestation of forms such as the two in the left hand box in Figure 1b. By contrast, a **paradigm** is the result of generalizing into stems (the $\mathbf{x_i}$s) and affixes (the remainder), such as the two paradigms in the right-hand box. This is in line with the idea that a mere collection of inflected forms is not a generalization per se, and we therefore reserve the term **paradigm** to the generalized version of an inflection table.[3] A single **slot** in an inflection table which contains a **form** (such as 'koiralla' in Figure 1b) will be generalized into

[3]Some authors call the generalization an **inflectional class** (Haspelmath and Sims, 2013).

a **form pattern**, such as $\mathbf{x_i}$**+lla**. The features corresponding to a form, such as NOM;SG will be referred to as the **morphosyntactic description** (MSD).

2 Related Work

As noted above, we build on the work of Ahlberg et al. (2014), Ahlberg et al. (2015), and Forsberg and Hulden (2016) which introduced the LCS as a core strategy in formal definitions of morphological paradigm and analogy. Further linguistic arguments favoring the LCS-model are given in Lee (2015).

Learning to generalize from inflection tables to unseen forms has attracted much recent interest in natural language processing (Dreyer and Eisner, 2011; Durrett and DeNero, 2013). In addition, two recent shared tasks hosted by SIGMORPHON and CoNLL have contributed to the research by producing comparable large multilingual data sets (Cotterell et al., 2016; Cotterell et al., 2017) consistently annotated with the Unimorph scheme (Sylak-Glassman et al., 2015; Kirov et al., 2018). Overwhelmingly, most recent work in inflection generation has employed neural sequence-to-sequence models with attention (Sutskever et al., 2014; Bahdanau et al., 2015) in supervised scenarios (Kann and Schütze, 2016; Faruqui et al., 2016; Östling, 2016; Makarov et al., 2017; Aharoni and Goldberg, 2017), often with data augmentation mechanisms in low-resource settings (Bergmanis et al., 2017; Silfverberg et al., 2017; Kann and Schütze, 2017). The PCFP has been explicitly addressed by recurrent neural generators as well (Malouf, 2016; Malouf, 2017). While neural models perform quite well on such tasks—even in low-resource settings—their parameter opacity makes it difficult to extract concise linguistic generalizations that can be interpreted and compared with linguist analyses.

Since morphological annotation is usually done at the word-level (**flowers** $\leftrightarrow$ ❀**+PL**),[4] recent work has also attempted to learn the latent segmentation and recover the different allomorphs for stems and affixes (**flower** = ❀, **s=PL**). This is arguably similar to what an L1-learner does, and the kind of evidence L1 learners have—words with semantic (❀) and grammatical (**+PL**) content deducible from context, but crucially missing information about how these correspond to some subsequence of phonemes in an inflected word form. Silfverberg and Hulden (2017a) and Silfverberg and Hulden (2017b) propose a data-driven method that searches the space of possible allomorph and grammatical tag associations, and favors a small model. Similar approaches are found in Hayes (2018) who similarly argues that allomorphs can be detected "even if we don't yet understand the phonology". Our work, apart from providing an explicit model for paradigms, also implicitly extracts the different allomorphs used, something that falls out as a byproduct of the paradigm generalizations in section 4 and inference from partial inflection tables in section 5.

3 Data

We use two different data sets for experiments. The first one is the training and development parts from the data set developed by Durrett and DeNero (2013) for inflection generation, including Finnish, German and Spanish.[5] The data set was scraped from the English Wiktionary. It is used both for experiments on paradigm generalization and learning of full paradigms from partial paradigms. We describe the data set in Table 2.

The second data set comes from the CoNLL-SIGMORPHON 2017 joint shared task on morphological reinflection (Cotterell et al., 2017). We use the paradigm completion subtask (subtask 2) data. We use this data exclusively for the paradigm generalization task.[6] The CoNLL-SIGMORPHON shared task data set contains full morphological inflection tables. We use the task's train-dev-test splits. Like the data set by Durrett and DeNero (2013), the CoNLL-SIGMORPHON data set is also scraped from Wiktionary. It contains data from 52 languages representing a wide variety of language families. However, we perform experiments on a selection of 33 languages out of these 52 languages because of the variance in inflection table size for some languages as explained below.[7]

[4] An illustration of a semantic representation.

[5] `http://www.cs.utexas.edu/~gdurrett/`

[6] We do not use CoNLL-SIGMORPHON data for paradigm learning because of the limited size of the data set.

[7] These 33 languages are Albanian, Bengali, Catalan, Danish, English, Estonian, Faroese, Finnish, French, Georgian, Ger-

The paradigm extraction method developed by Ahlberg et al. (2014), as well as the paradigm generalization method to be proposed in section 4, operates on full paradigms, requiring that all of the inflection tables for a given part-of-speech have the same number of MSDs. However, this is not the case in the CoNLL-SIGMORPHON shared task data: inflection tables of German nouns, for instance, encompass between four and eight forms. There are several reasons for this discrepancy. For example, some nouns may lack plural or singular forms altogether (as an example, consider the English *plurale tantum* noun **pants**). We, therefore, first determine the maximal number of MSDs for inflection tables of each part-of-speech in each language and perform experiments only on those inflection tables where all MSDs are present. This is not possible for all languages because many do not contain a single table which would give forms corresponding to the full set of MSDs. Consequently, we limit our experiments to those languages where we can successfully find lexemes with a maximal number of MSDs.

4 Generalized Paradigms

We now turn to the method for making further generalizations over the basic paradigm model given in Ahlberg et al. (2014). This method also extracts sets of allomorphs.

As mentioned in section 1, the paradigm extraction procedure presented in Ahlberg et al. (2014) cannot group inflection tables under the same paradigm unless the non-stem parts of corresponding forms in the tables are identical. For example, the two Finnish stems **lauk** and **hyp** in Figure 2 cannot belong to the same paradigm because back vowels (**a**) and front vowels (**ä**) vary across the two paradigms, though the form patterns are similar in other respects. This does not adhere to standard linguistic assumption whereby regular phonological alternations—such as vowel harmony in this case—should not affect paradigm membership.

Our method for paradigm generalization groups together paradigms which exhibit such regular alternations, under the same *generalized paradigm*. The method is based purely on a formal comparison of the paradigms, not linguistic considerations; it is in effect 'phonology-free'. We discover regular alternations using a variable substitution described below, and group multiple paradigms under the same *generalized paradigm* whenever they exhibit regular alternation without regard to the identity of the variants.

4.1 Method

The basis for paradigm generalization is identification of regular alternations between paradigms. Consider the example in Figure 2. Both of the nominal lexemes **laukku** ('bag') and **hyppy** ('jump') undergo consonant gradation. In what is called the 'strong grade', the last stem consonant is doubled both in the nominative (NOM) and partitive (PART) cases. The alternation between the paradigms is regular in the sense that whenever the form for **laukku** has an additional **k**, **hyppy** will have an additional **p**. We abstract out the differences between variants by replacing the varying parts in form patterns with consecutively numbered new variables $\mathbf{y_i}$, where multiple occurrences of the same varying part all get the same variable $\mathbf{y_i}$. For the current case, the **k** in the paradigm for **laukku** and the **p** in the paradigm extracted from **hyppy** in the strong grade are replaced by $\mathbf{y_1}$ both in the nominative and partitive forms, **lla** and **llä** are both replaced with $\mathbf{y_2}$, **sta** and **stä** are both replaced with $\mathbf{y_3}$, and **a** and **ä** are both replaced with $\mathbf{y_4}$. Applying this second-order generalization procedure on top of the LCS-generalization results in identical generalized paradigm structure for **laukku** and **hyppy**, seen on the right in Figure 2.

4.2 Experiments and Results

As mentioned in Section 3 we use two data sets for experiments: the data set from Durrett and DeNero (2013), and part of the data (33 languages) for the second subtask of the CoNLL-SIGMORPHON 2017 shared task on universal morphological reinflection. The number of paradigms and generalized paradigms in the Durrett and DeNero (2013) data set is shown in Table 2 in the last two columns. The number of generalized paradigms agrees much better with linguist analyses regarding the number of

man, Hebrew, Hindi, Hungarian, Icelandic, Irish, Italian, Latin, Latvian, Lithuanian, Lower Sorbian, Northern Sami, Persian, Portuguese, Romanian, Slovak, Slovene, Spanish, Swedish, Turkish, Urdu, and Welsh, which are from the 11 different language groups (based on the original data source grouping given by Cotterell et al. (2017)) including Romance, Germanic, Indo-Aryan, Uralic, Kartvelian, Semitic, Celtic, Baltic, Iranian, Slavic, and Turkic

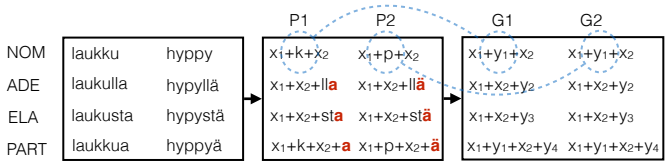


Figure 2: Forming a generalized paradigm (right) based on a paradigm discovered by the LCS method (Ahlberg et al., 2014) (middle) from raw inflection tables (left).

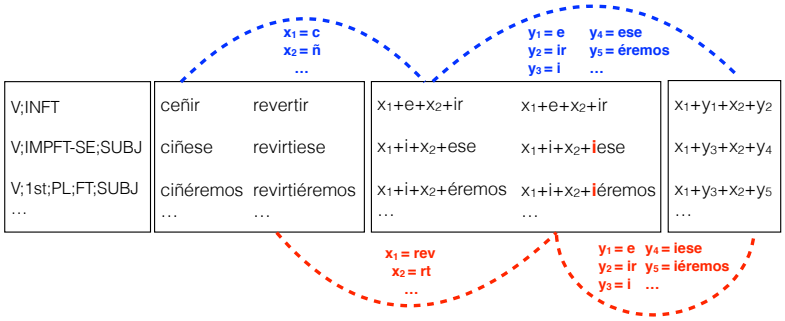


Figure 3: Illustration of a latent phonological generalization in Spanish.

types of inflection tables within languages. For example, Nykysuomen sanalista,[8] a lexical resource developed by the Institute for the Languages of Finland, recognizes 51 nominal inflection types and 27 verb inflection types for Finnish. The number of generalized paradigms for Finnish nouns and adjectives[9] produced by the current method is 40 for nouns and 30 for verbs. The average ratio between the number of generalized paradigms and the number of original LCS-paradigms across languages are 39.07% for verbs, 26.29% for nouns and adjectives, and 35.02% for all parts-of-speech together, showing a significant reduction in number of disctinct patterns identified.

In addition, generalized paradigms capture linguistically regular alternations. Figure 3 illustrates how the generalized paradigm captures a phenomenon surrounding the grapheme **ñ** in Spanish. Verbs that don't contain stem-final **ñ** [ɲ] insert an **i** in some forms, while verbs with **ñ** forgo the **i** as the **ñ** itself can be interpreted as a coalesced form that already contains an **i**, i.e. [ni] ↦ [ɲ]. Spanish verbs like **ceñir** ('to cling to') and verbs like **revertir** ('to revert to') would fall into two separate paradigms using the paradigm extraction method proposed in Ahlberg et al. (2014). These two LCS-paradigms are provided in the second box from the right in the figure. The dotted blue curve on top shows the first-order generalization provided by the LCS-strategy that assigns $\mathbf{x_i}$-variables, and our subsequent second-order abstraction that assigns $\mathbf{y_i}$-variables. After these two steps, the behavior of the the two verbs is seen to be identical in the new representation shown in the rightmost box. Other words which belong to the same LCS-paradigm as **ceñir** that all have **ñ** as their second variable include words like **reñir** ('to scold'), **constreñir** ('to restrict'), **teñir** ('to stain'). The second variable for words which get the same paradigm as **revertir** can be **rt**, **r**, **nt** etc. Words in this paradigm include **convertir** ('to convert'), **ingerir** ('to ingest'), **sentir** ('to feel') etc. The generalized paradigm groups the two paradigms together because the two paradigms become identical after $\mathbf{y_i}$-replacement following the method described in section 4.1. The $\mathbf{y_i}$s from the first paradigm are given along the blue dotted curve connecting the paradigm with the generalized paradigm, and the $\mathbf{y_i}$s from the second paradigm are added to the red dotted curve at the bottom combining the second paradigm with the generalized paradigm. By grouping the two paradigms together as one generalized paradigm, the alternation between **ñ** and any other consonants followed by **i** emerges.

With the second data set of 33 languages, we combine all the training, development and test sets together as the data set is small (less than 300 inflection tables altogether) for each language. For all the

[8] `http://kaino.kotus.fi/sanat/nykysuomi`

[9] Finnish nouns and adjectives are inflected in the same way in the data set.

languages we experiment with, the number of paradigms decreases consistently after generalization: the average ratio between the number of generalized paradigms and the number of LCS-paradigms across languages on this data set is 0.69 for verbs, 0.55 for nouns and adjectives, and 0.59 for all parts-of-speech together. The ratio is higher on this data set than on the Durrett and DeNero (2013) data set because the data size here is much smaller, and yields fewer paradigms in general.

LANGUAGE	POS	TABLE COUNT	TABLE SIZE	PARADIGM COUNT	GENERALIZED PARADIGM COUNT
FINNISH	NOUN & ADJ	6,400	28	259	40
FINNISH	VERB	7,249	53	276	30
GERMAN	NOUN	2,764	8	70	26
GERMAN	VERB	2,027	27	135	107
SPANISH	VERB	4,055	57	96	26

Table 2: Data sets by Durrett and DeNero (2013). We show counts of inflection tables and inflection table sizes for nouns, adjectives and verbs. Additionally, we show the number of paradigms found the LCS method (Ahlberg et al., 2014) and the resulting number of 'generalized paradigms' found by subsequently applying our method.

An example of how generalization over paradigms collapses them and catches regularities from the CoNLL-SIGMORPHON data set can be found by examining English verbal inflections. Different paradigms are extracted from verbs like **contest**, **establish**, **benefit**, **occur** and **bag**, though the paradigm for **establish** differs from that for **contest** only in the present tense for the third-person singular, with the last variable as **-es** for **establish**, but **-s** for **contest**; the paradigms for **benefit**, **occur** and **bag** differ from that of **contest** in the present participle, past participle and past tense, for which the last variables for **benefit**, **occur** and **bag** are **-ting**, **-ring** and **-ging** (for the present participle) and **-ted**, **-red** and **-ged** (for the past participle and past tense) respectively, but **-ing** and **-ed** for **contest**. The generalized paradigm approach groups all these patterns together, which completely agrees with the linguistic regularity.

5 Learning Full Paradigms from Partial Paradigms

In this section we present an algorithm for learning full morphological paradigms given partial inflection tables. Our algorithm takes as input a set of partial inflection tables, that is, inflection tables with a number of missing forms. It first derives a morphological LCS-paradigm for each partial inflection table and then imputes missing form patterns into the paradigm. At test time, form patterns are substituted by concrete word forms (see Figure 4a). A key component of the proposed algorithm is a stem sampling step which counteracts bad hypotheses about stems caused by missing information when performing paradigm extraction on only partial inflection tables.

V;INF	speak		V;INF	speak
V;PRES	?		V;PRES	speaks
V;PAST	?	$\Longrightarrow$	V;PAST	spoke
V;PAST;PCPLE	spoken		V;PAST;PCPLE	spoken
V;PRES;PCPLE	?		V;PRES;PCPLE	speaking

(a) Our inputs are partial inflection tables and the final outputs are completed inflection tables.

V;INF	-		V;INF	-
V;PRES	-		V;PRES	-
V;PAST	-	$\Longrightarrow$	V;PAST	-
V;PAST;PCPLE	eaten		V;PAST;PCPLE	$\mathbf{x_1{+}e{+}x_2}$
V;PRES;PCPLE	eating		V;PRES;PCPLE	$\mathbf{x_1{+}i{+}x_2{+}g}$

(b) Paradigm extraction is applied to a partial inflection table. The stem is incorrectly identified as **eatn**. This makes it impossible to correctly infer the infinitive **eat** and past tense **ate**.

Figure 4: We illustrate the paradigm cell filling task in 4a and show an example of over-generalization effects due to data sparsity in 4b.

We evaluate the presented algorithm with regard to its ability to reconstruct missing forms in partial paradigms and compare it against two baselines: majority voting and matrix completion. Our experiments demonstrate that the proposed algorithm delivers clear improvements over the baselines.

5.1 Methods

As mentioned above, our input consists of partial inflection tables. The first step is to apply the LCS procedure for paradigm extraction (Ahlberg et al., 2014) presented in Section 1. This results in a partial

paradigm as shown in Figure 4. We now present four different methods for imputing the missing forms into the resulting partial paradigm.

First Baseline using Majority Voting For each partial paradigm P, we apply a simple variant of majority voting in order to infer missing form patterns. We form the set of all partial paradigms which have a form pattern corresponding to a missing MSD, m, and perform a majority vote among the stem patterns. A slight variant of standard majority voting is, however, required because we do not want to infer a candidate with an incorrect number of stem variables. For example, if the PAST and PRES;PCPLE forms in P are $\mathbf{x_1}$**+ed** and $\mathbf{x_1}$**+ing** respectively, we do not want to infer a form $\mathbf{x_1}$**+**$\mathbf{x_2}$**+n** for PAST;PCPLE because the stem corresponding to P has only one variable $\mathbf{x_1}$. Therefore, we restrict the vote to paradigms which have the same number of variables ($\mathbf{x_i}$s) as P.

We want to further restrict the set of candidates, because whenever two paradigms P and P' disagree on a known form, it is likely that they will disagree on missing forms as well. For example, if the PAST form in P is $\mathbf{x_1}$**+ed** and the PAST form for P' is $\mathbf{x_1}$**+n**, that is, the paradigms disagree on the PAST MSD, then the PAST;PCPLE forms are unlikely to agree either. Therefore, we further restrict voting to those candidate paradigms P', where all the shared MSDs with P correspond to identical form patterns. For each MSD m, we now take a majority vote in the restricted set of candidate paradigms.

Priority Voting As a small additional restriction to the majority voting process, we consider restricting the set of candidates even further. Whereas we include candidates with no overlap, i.e. common forms, in majority voting, we now exclude such paradigms from voting. That is, only include candidates which have some common MSDs with the target paradigm P. These are, after all, more likely to agree other forms as well. This set may, however, be empty. We, therefore, resort to applying majority voting as fall-back.

Stem Sampling We apply the paradigm extraction procedure outlined in Section 1 to partial inflection tables. Unfortunately, this frequently leads to incorrectly identified stems as shown in Figure 4b. This happens when all of the given forms (for example, **eating** and **eaten**) share a substring (**n**) which, nevertheless, is a part of the inflectional affixes and not the actual word stem. The common substring then becomes incorrectly incorporated in the word stem (**eat**, **n**), because the stem is identified greedily as the longest common subsequence of all of the forms in the partial inflection table. This gives rise to implausible form patterns ($\mathbf{x_1}$**+i+**$\mathbf{x_2}$**+g** and $\mathbf{x_1}$**+e+**$\mathbf{x_2}$, from **eat**<u>**i**</u>**ng**, **eat**<u>**e**</u>**n**). To counteract this problem, we perform a sampling step before imputing missing forms.

By examining all partial paradigms extracted from partial inflection tables at once, we can make a better guess concerning the identity of the stems. It is a reasonable assumption that an MSD such as PAST;PCPLE have few distinct realizations. In English, most words take a past participle ending of either **-n** or **-ed**. Although there are examples of other PAST;PCPLE markers, the majority of words belong to one of these classes. Therefore, we would expect most form patterns in the past perfect slot to look similar to $\mathbf{x_1}$**+ed** or $\mathbf{x_1}$**+**$\mathbf{x_2}$**+n**. However, incorrectly identified stems will frequently result in implausible variants like $\mathbf{x_1}$**+**$\mathbf{x_2}$, $\mathbf{x_1}$**+d** or $\mathbf{x_1}$**+e+**$\mathbf{x_2}$. We can expect to find better stem candidates if we attempt to find a set of stems which minimize the number of realizations for each MSD over the whole data set, i.e. all partial paradigms.

The general plan of attack is to sample stems in a way that favors fewer realizations for each MSD. Let the partial inflection table be as in Figure 4b. The longest common subsequence between all of the given forms is **eatn**, which leads us to postulate it as the stem. We also get forms $\mathbf{x_1}$**+e+**$\mathbf{x_2}$ and $\mathbf{x_1}$**+i+**$\mathbf{x_2}$**+g**. We can consider additional stem candidates, for example, **ea** and **eat**, which are substrings of the original stem. Each stem candidate leads to different realizations for the inflections. For example, the candidate **eat** will give realizations $\mathbf{x_1}$**+en** and $\mathbf{x_1}$**+ing** for the PAST;PCPLE and PRES;PCPLE forms, respectively.

At the start of the sampling process, each paradigm P has a stem S_0, which is a collections of stem parts $S_0(1), ..., S_0(n)$ (for example $S_0(1) =$ **eat** and $S_0(2) =$ **n** in Figure 4b). Each stem part is a sequence of letters $S_0(i)^1, ..., S_0(i)^m$. Intuitively, we sample a new stem S_1 by replacing a random stem part $S_0(i)$ by a random continuous substring of $S_0(i)^k, ..., S_0(i)^{k+l}$. This gives us a new stem proposal

S_1, where each stem part $S_1(j) = S_0(j)$, except $S_1(i)$, which is a substring of $S_0(i)$. For example we could sample a new stem part $S_1(2) = \epsilon$ based on the existing stem part $S_0(2) = \mathbf{n}$. The new stem will now define new realizations for each of the MSDs in paradigm P (for example, $\mathbf{x_1}$**+en** and $\mathbf{x_1}$**+ing**) and we can use statistics about MSD realizations over the entire set of partial paradigms to determine whether to accept or discard the new stem.

Note that whenever we sample a new stem S_k, we sample a substring of the original stem part $S_0(i)$. This allows stem parts to both shrink and grow during the sampling process but they can never grow beyond the stem parts in the original stem S_0.

We set up stem sampling as a product of Chinese Restaurant Processes (CRP) and use Gibbs sampling to optimize the stem assignments for all inflection tables in the data set. This has the effect of preferring MSD realizations which are common. Therefore, sampling will indirectly minimize the number of distinct MSD realizations, which is our goal.

Let P be a randomly selected paradigm, S_i be a stem for P and S_{i+1} a sampled stem. Let $F_i = \{F_i(m_1), ..., F_i(m_k)\}$ and $F_{i+1} = \{F_{i+1}(m_1), ..., F_{i+1}(m_k)\}$ be the set of form patterns, determined by S_i and S_{i+1}, respectively, for each MSD m_1,...,m_k in paradigm P. Further, let $\#(f, m)$ be the count of form pattern f for MSD m over all paradigms $\neq P$. We now accept the new stem setting S_{i+1} with probability $p_a = \prod_j \frac{G_{i+1}(m_j)}{G_i(m_j)}$, where $G_k(m_j) = \alpha$, if $\#(F_k(m_j), m_j) = 0$, and $G_k(m_j) = \#(F_k(m_j), m_j)$, otherwise. Whenever, $p_a > 1$, we automatically accept the new setting S_{i+1}. If we reject the new setting, we set $S_{i+1} := S_i$. Based on preliminary experiments, we set α to 0.1.

We run the sampler for 100 steps over the entire training set. In order to ensure that each paradigm is sampled, we do not randomly select paradigms but instead loop through and sample all paradigms shuffling the data set after each epoch.

Second Baseline using Matrix Completion As an additional baseline, we frame the task of learning complete paradigms as a matrix completion task. We first infer the paradigm structure for each partial inflection table. We then encode each form pattern present in paradigm P into a one-hot representation $\mathbf{v_m} \in \mathbb{R}^N$, where N is the number of distinct realizations of MSD m as shown in Figure 5.[10]

Collecting the rows for each paradigm P, we then form a matrix M, whose rows $\mathbf{r_P} = [\mathbf{v_{m_1}}; ...; \mathbf{v_{m_n}}]$, where m_1, ..., m_n are all the distinct MSDs in the data set. Since the paradigms P are incomplete, M will have a number of missing entries. We complete these entries using the SOFT-IMPUTE algorithm introduced by Mazumder et al. (2010).[11]

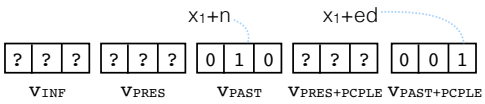


Figure 5: Form patterns are encoded into one-hot vectors and concatenated.

Completed entries $\mathbf{v_m}$ may not be one-hot vectors. Therefore, we transform them into one-hot vectors by applying the transformation $\mathbf{v_m}'[\arg\max_i \mathbf{v_m}[i]] = 1$ and $\mathbf{v_m}'[i] = 0$ otherwise. Finally, we decode the one-hot vectors defined by M into form patterns.

5.2 Experiments and Results

We start with complete inflection tables and uniformly sample a fixed number of forms n from each to form partial inflection tables (n=2,3,4,5 and 6). This gives us collections of partial inflection tables where each one has exactly n forms. We then extract paradigms from the partial inflection tables and use the different methods presented in Section 5.1 to impute the missing forms into the paradigms. More specifically, we apply the following methods (1) sampling of stems followed by priority voting, (2) Priority voting, (3) Majority voting (baseline I), and (4) matrix completion (baseline II).

To evaluate the method, we, after inferring the complete paradigms, substitute stem parts in to form patterns, for example $\mathbf{x_1}$**+ed** $\rightarrow$ **eased**. Thus, we end up with a complete inflection table. We evaluate

[10] In preliminary experiments, we encoded each form pattern as a distinct integer. This, however, delivered poorer performance.

[11] We use the implementation in the `fancyimpute` Python package `https://pypi.python.org/pypi/fancyimpute` by Alex Rubinsteyn and Sergey Feldman.

the result against the gold standard inflection tables. using two metrics: how many of the missing forms were correctly reconstructed and how many of the entire tables were correctly reconstructed.

We perform experiments separately on nouns (nouns and adjectives for Finnish) and verbs. We do this because (1) the performance varies quite a lot between nouns and verbs for the same language, and (2) we only have data for Spanish verbs but we have data both for nominals and verbs for Finnish and German.

Table 3 in Appendix A presents results of the experiment. Overall, we can see that the number of successfully reconstructed forms increases as the number of given forms in the paradigm increases. The best results are attained for German nouns where the number of MSDs in the paradigm is the smallest (8 MSDs per paradigm). Conversely, results are the poorest for Finnish verbs, which have a large number of MSDs per paradigm (54). However, all methods except matrix completion fare quite well on full paradigm learning for Spanish verbs which have even more MSDs per paradigm than Finnish verbs (57).

Matrix completion performs very poorly in comparison to the other methods on all data sets except German nouns. Even for German nouns it, however, performs worse than Baseline I, namely, majority voting. Priority voting consistently outperforms majority voting (by over 30%-points for Spanish verbs when three forms are given). We can also see that sampling gives a consistent improvement over other methods except for German nouns which have the smallest inflection tables (only nine forms per table).

6 Discussion and Conclusions

Our method for paradigm generalization results in a marked decrease in paradigm counts in both Durrett and DeNero (2013) and CoNLL-SIGMORPHON data sets. For Finnish, our paradigm counts for verbs and nouns very closely match the counts arrived at by linguists (40 vs. 51 and 30 vs. 27 for verbs and nouns+adjectives respectively). Qualitative analysis of generalized paradigms shows that the method is clearly capable of capturing regularities like the examples illustrated about Finnish, Spanish, and English.

On the full morphological paradigm learning task, the baseline voting methods (majority and priority voting) are surprisingly effective. Priority voting is able to reconstruct 84% of missing Spanish verb forms and 96% of missing German noun forms when six forms are given in the input inflection table. However, stem sampling on top of priority voting still results in sizable gains over the baseline methods especially in the presence of few given forms in the input table. It is to be expected that the effect diminishes when the number of input forms increases as this reduces the risk of mis-identifying stems. Nevertheless, even with six input forms, stem sampling still results in relatively large gains of around 4%-points for Finnish, German and Spanish verbs which have the largest table sizes in the data set.

Both methods proposed above can be useful for developers of linguistic resources. Paradigm generalization from partial paradigms can be valuable for development of morphological analyzers. Generalized paradigms can guide a linguist working on the phonological and morphological description of a language, and also extract all the allomorphs that are related to a grammatical category.

A clear future research direction for full paradigm learning is exploration of neural methods for the task. While it is to be expected that neural methods will reach high performance, it will be substantially more difficult to draw clear linguistic generalizations from the models they produce. As an added advantage of the current method, we automatically split word forms into stems and inflectional material. Such a split can be difficult to derive from a neural model which can reduce its usefulness as an aid in writing linguistic descriptions.

We have presented a model for automatically inducing generalized morphological paradigms which agrees well with the linguist intuitions concerning morphological pattern generalization since it factors out regular phonological alternations, without actually being 'phonology-aware'. Moreover, we have presented a method for learning full morphological paradigms given training data of partial inflection tables only, and have shown that the method is capable of inferring 90% of missing forms for paradigms as large as 57 forms from only six given forms.

Acknowledgements We want to thank the anonymous reviewers for their valuable comments. The first author is supported by The Society of Swedish Literature in Finland (SLS).

References

Farrell Ackerman, James P. Blevins, and Robert Malouf. 2009. Parts and wholes: Implicative patterns in inflectional paradigms. In James P. Blevins and Juliette Blevins, editors, *Analogy in Grammar*, pages 54–82. Oxford University Press.

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2004–2015, Vancouver, Canada, July. Association for Computational Linguistics.

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2014. Semi-supervised learning of morphological paradigms and lexicons. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 569–578, Gothenburg, Sweden, April. Association for Computational Linguistics.

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2015. Paradigm classification in supervised learning of morphology. In *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*, pages 1024–1029, Denver, CO. Association for Computational Linguistics.

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

Toms Bergmanis, Katharina Kann, Hinrich Schütze, and Sharon Goldwater. 2017. Training data augmentation for low-resource morphological inflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 31–39, Vancouver, August. Association for Computational Linguistics.

James P. Blevins and Juliette Blevins, editors. 2009. *Analogy in Grammar*. Oxford University Press.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22, Berlin, Germany, August. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 1–30, Vancouver, August. Association for Computational Linguistics.

Markus Dreyer and Jason Eisner. 2011. Discovering morphological paradigms from plain text using a Dirichlet process mixture model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 616–627. Association for Computational Linguistics.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1185–1195.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 634–643, San Diego, California, June. Association for Computational Linguistics.

Markus Forsberg and Mans Hulden. 2016. Learning transducer models for morphological analysis from example inflections. In *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata (StatFSM)*, pages 42–50, Berlin, Germany, August. Association for Computational Linguistics.

Martin Haspelmath and Andrea Sims. 2013. *Understanding Morphology*. Routledge.

Bruce Hayes. 2018. Allomorph discovery as a basis for learning alternations. In *Proceedings of the Society for Computation in Linguistics: Vol. 1 , Article 33.*

Mans Hulden. 2014. Generalizing inflection tables into paradigms with finite state operations. In *Proceedings of the 2014 Joint Meeting of SIGMORPHON and SIGFSM*, pages 29–36. Association for Computational Linguistics.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 555–560, Berlin, Germany, August. Association for Computational Linguistics.

Katharina Kann and Hinrich Schütze. 2017. The LMU system for the CoNLL-SIGMORPHON 2017 shared task on universal morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 40–48, Vancouver, August. Association for Computational Linguistics.

Christopher Kendris and Theodore Kendris. 2007. *501 Spanish verbs*. Barron's.

Christo Kirov, Ryan Cotterell, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sebastian J. Mielke, Arya McCarthy, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. Unimorph 2.0: Universal morphology. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA).

Jackson L. Lee. 2015. Morphological Paradigms: Computational Structure and Unsupervised Learning. In *Proceedings of NAACL-HLT 2015 Student Research Workshop (SRW)*, pages 161–167, Denver, Colorado, June. Association for Computational Linguistics.

Peter Makarov, Tatiana Ruzsics, and Simon Clematide. 2017. Align and copy: UZH at SIGMORPHON 2017 shared task for morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 49–57, Vancouver, August. Association for Computational Linguistics.

Robert Malouf. 2016. Generating morphological paradigms with a recurrent neural network. *San Diego Linguistic Papers*, 6:122–129.

Robert Malouf. 2017. Abstractive morphological learning with a recurrent neural network. *Morphology*, 27:431–458.

Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. 2010. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11:2287–2322.

Robert Östling. 2016. Morphological reinflection with convolutional neural networks. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 23–26, Berlin, Germany, August. Association for Computational Linguistics.

Miikka Silfverberg and Mans Hulden. 2017a. Automatic morpheme segmentation and labeling in universal dependencies resources. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 140–145, Gothenburg, Sweden, May. Association for Computational Linguistics.

Miikka Silfverberg and Mans Hulden. 2017b. Weakly supervised learning of allomorphy. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 46–56, Copenhagen, Denmark, September. Association for Computational Linguistics.

Miikka Silfverberg, Adam Wiemerslage, Ling Liu, and Lingshuang Jack Mao. 2017. Data augmentation for morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 90–99, Vancouver, August. Association for Computational Linguistics.

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

John Sylak-Glassman, Christo Kirov, David Yarowsky, and Roger Que. 2015. A language-independent feature schema for inflectional morphology. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 674–680, Beijing, China, July. Association for Computational Linguistics.

Appendix A. Detailed Results

Method	forms = 2	forms = 3	forms = 4	forms = 5	forms = 6
FINNISH NOUNS AND ADJECTIVES					
1. Sampling	**38.08** ±0.36 (0.09 ±0.11)	**53.71** ±3.27 (4.88 ±1.31)	**60.39** ±0.37 (8.25 ±1.23)	**66.57** ±0.26 (11.59 ±0.80)	**71.32** ±0.95 (21.81 ±1.25)
2. Priority Voting	24.64 ±0.79 (0.00 ±0.00)	45.95 ±3.11 (1.75 ±0.59)	56.09 ±0.45 (7.47 ±0.94)	62.80 ±0.33 (12.59 ±0.95)	68.85 ±1.11 (20.50 ±1.35)
3. Majority Voting (Baseline I)	15.40 ±0.75 (0.00 ±0.00)	27.02 ±1.10 (0.00 ±0.00)	31.40 ±0.28 (0.03 ±0.07)	31.12 ±0.22 (0.38 ±0.30)	29.61 ±0.47 (1.20 ±0.35)
4. Soft Impute (Baseline II)	1.23 ±0.16 (0.00 ±0.00)	2.14 ±0.51 (0.00 ±0.00)	2.53 ±0.12 (0.00 ±0.00)	3.11 ±0.14 (0.00 ±0.00)	3.26 ±0.18 (0.00 ±0.00)
FINNISH VERBS					
1. Sampling	**27.98** ±0.25 (0.00 ±0.00)	**44.29** ±0.16 (0.00 ±0.00)	**54.56** ±0.36 (0.00 ±0.00)	**57.27** ±0.17 (0.00 ±0.00)	**63.31** ±0.20 (0.00 ±0.00)
2. Priority Voting	13.65 ±0.20 (0.00 ±0.00)	30.95 ±0.19 (0.00 ±0.00)	44.68 ±0.28 (0.00 ±0.00)	52.02 ±0.20 (0.00 ±0.00)	59.91 ±0.20 (0.00 ±0.00)
3. Majority Voting (Baseline I)	10.79 ±0.19 (0.00 ±0.00)	22.12 ±0.15 (0.00 ±0.00)	22.10 ±0.22 (0.00 ±0.00)	20.92 ±0.12 (0.00 ±0.00)	20.07 ±0.14 (0.00 ±0.00)
4. Soft Impute (Baseline II)	0.38 ±0.03 (0.00 ±0.00)	0.80 ±0.04 (0.00 ±0.00)	1.02 ±0.04 (0.00 ±0.00)	1.34 ±0.06 (0.00 ±0.00)	1.54 ±0.05 (0.00 ±0.00)
GERMAN NOUNS					
1. Sampling	55.55 ±4.53 (10.64 ±2.10)	65.22 ±7.24 (26.70 ±3.13)	83.45 ±1.99 (66.93 ±2.00)	92.75 ±0.92 (85.02 ±2.90)	95.26 ±1.26 (92.15 ±1.46)
2. Priority Voting	55.28 ±4.40 (10.82 ±2.09)	64.33 ±6.80 (25.43 ±2.63)	82.39 ±1.79 (65.41 ±2.64)	92.12 ±0.85 (84.33 ±2.89)	95.53 ±1.22 (92.40 ±1.41)
3. Majority Voting (Baseline I)	43.64 ±6.32 (2.71 ±1.07)	44.10 ±11.40 (1.45 ±0.99)	50.10 ±7.14 (14.04 ±1.88)	51.95 ±1.26 (21.78 ±2.33)	52.42 ±1.98 (33.32 ±3.20)
4. Soft Impute (Baseline II)	18.59 ±0.99 (0.18 ±0.20)	32.34 ±8.27 (1.45 ±0.81)	32.45 ±5.95 (6.15 ±1.87)	35.06 ±0.73 (10.53 ±2.16)	36.14 ±2.32 (19.57 ±2.33)
GERMAN VERBS					
1. Sampling	**61.24** ±2.41 (**11.74** ±2.27)	**76.23** ±0.90 (**34.68** ±2.76)	**81.54** ±0.52 (**40.16** ±2.40)	**82.79** ±0.99 (**42.03** ±3.19)	**84.49** ±0.34 (**44.89** ±2.42)
2. Priority Voting	40.55 ±3.79 (0.00 ±0.00)	64.39 ±0.86 (11.15 ±2.84)	70.16 ±0.59 (16.18 ±2.01)	75.49 ±0.91 (24.12 ±3.33)	78.65 ±0.44 (28.52 ±2.87)
3. Majority Voting (Baseline I)	14.57 ±2.43 (0.00 ±0.00)	32.12 ±0.38 (0.00 ±0.00)	47.31 ±0.41 (15.98 ±2.99)	72.63 ±0.92 (40.11 ±3.20)	74.43 ±0.42 (41.74 ±2.52)
4. Soft Impute (Baseline II)	8.55 ±1.57 (0.00 ±0.00)	8.35 ±0.42 (0.00 ±0.00)	11.65 ±0.45 (0.00 ±0.00)	12.34 ±0.37 (0.00 ±0.00)	15.67 ±0.35 (0.00 ±0.00)
SPANISH VERBS					
1. Sampling	**49.69** ±2.83 (0.00 ±0.00)	**72.38** ±1.23 (25.38 ±1.72)	**86.23** ±0.15 (**53.88** ±3.37)	**87.26** ±0.14 (**58.35** ±2.07)	**89.29** ±0.11 (**67.08** ±2.38)
2. Priority Voting	34.22 ±1.55 (0.00 ±0.00)	60.01 ±1.02 (25.33 ±1.75)	72.02 ±0.18 (35.71 ±2.74)	80.01 ±0.42 (45.77 ±2.69)	83.64 ±0.21 (53.91 ±2.45)
3. Majority Voting (Baseline I)	21.41 ±1.81 (0.00 ±0.00)	31.00 ±0.59 (0.00 ±0.00)	46.31 ±0.26 (0.00 ±0.00)	53.81 ±0.09 (47.79 ±2.73)	57.94 ±0.05 (51.94 ±2.65)
4. Soft Impute (Baseline II)	0.57 ±0.07 (0.00 ±0.00)	1.28 ±0.06 (0.00 ±0.00)	2.48 ±0.08 (0.00 ±0.00)	3.63 ±0.15 (0.00 ±0.00)	4.46 ±0.09 (0.00 ±0.00)

Table 3: Results for learning full paradigms from partial paradigms. The columns give results in presence of n forms in the partial paradigms. We give the amount of missing forms which were successfully reconstructed. In parentheses, we give the amount of complete paradigms which were successfully reconstructed. We give confidence intervals at the 99% level from a one-sided t-test and show the cases where sampling gives significantly better results than the other methods in bold typeface.

Relation Induction in Word Embeddings Revisited

Zied Bouraoui
CRIL CNRS UMR 8188
Artois University
F-62300 Lens, France
bouraoui@cril.fr

Shoaib Jameel
School of Computing
Medway Campus
University of Kent, UK
M.S.Jameel@kent.ac.uk

Steven Schockaert
School of Computer Science
and Informatics
Cardiff University, UK
schockaerts1@cardiff.ac.uk

Abstract

Given a set of instances of some relation, the relation induction task is to predict which other word pairs are likely to be related in the same way. While it is natural to use word embeddings for this task, standard approaches based on vector translations turn out to perform poorly. To address this issue, we propose two probabilistic relation induction models. The first model is based on translations, but uses Gaussians to explicitly model the variability of these translations and to encode soft constraints on the source and target words that may be chosen. In the second model, we use Bayesian linear regression to encode the assumption that there is a linear relationship between the vector representations of related words, which is considerably weaker than the assumption underlying translation based models.

1 Introduction

It has been observed that many lexical relationships can be modelled as vector translations in a word embedding space (Mikolov et al., 2013b; Pennington et al., 2014). Even though this remarkable property is now well-established, and is commonly used as a basis for evaluating word embedding models, its potential for knowledge base completion has only been explored in a preliminary way. In this paper, we are particularly interested in the following relation induction task, which was proposed in (Vylomova et al., 2016): given a set $\{(s_1, t_1), ..., (s_n, t_n)\}$ of word pairs that are related in a given way, identify new word pairs (s, t) that are likely to be related in the same way. We will refer to s and t as the source and target word respectively; we will write $\mathbf{w}$ for the vector representation of the word w.

One natural approach is to model the considered relation using the average translation vector $\mathbf{r} = \frac{1}{n}\sum_i(\mathbf{t_i} - \mathbf{s_i})$, and to accept (s, t) as a likely instance if $\cos(\mathbf{s} + \mathbf{r}, \mathbf{t})$ is sufficiently high. While this approach was found in (Drozd et al., 2016) to work well for analogy completion, for relation induction it typically leads to too many false positives. This is illustrated in Table 1 for the case where $\mathbf{r}$ is constructed from the instances of the *capital of* relation of the BATS dataset[1]. As can be seen in the table, most of the top-ranked pairs are actually incorrect. In practice, this problem is further exacerbated by the dramatic class imbalance: for typical vocabulary sizes there are tens (or hundreds) of billions of incorrect pairs, compared to only a few hundred correct instances. While correct instances may, on average, get a higher score than incorrect instances, as a result of this imbalance there will still be many incorrect instances that receive a very high score.

Another problem relates to the use of the cosine similarity, which treats each dimension in the same way when comparing the vectors $\mathbf{r}$ and $\mathbf{t} - \mathbf{s}$. In practice, however, some dimensions of the word embedding may correspond to features of meaning that are irrelevant for the considered relationship. When we are only given one example (s, t) of a correct instance, as in the most common version of the analogy completion task, the cosine similarity is a suitable choice as we cannot determine which dimensions are most relevant. For relation induction, however, we can use the empirical variance of the translation vectors $\mathbf{t_i} - \mathbf{s_i}$ to make a more informed choice.

[1] See section 4 for more details about the experimental set-up, including how negative test examples were chosen.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1627–1637
Santa Fe, New Mexico, USA, August 20-26, 2018.

word pair	cos	word pair	cos
(horse, horses)	0.84	*(baghdad,iraq)*	0.64
(boy, girl)	0.79	(aware, unaware)	0.64
(madrid, spain)	0.73	*(moscow, russia)*	0.63
(london, england)	0.69	*(berlin, germany)*	0.63
(spain, madrid)	0.68	(look, looking)	0.61
(walk, walks)	0.65	(moscow,germany)	0.59

Table 1: Cosine scores for the average translation model applied to the *capital of* relation; correct instances are shown in italics.

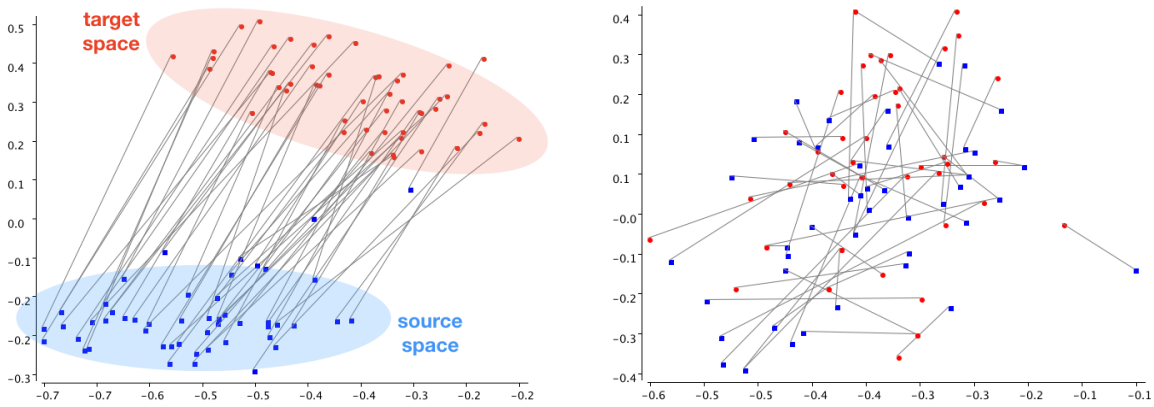
In this paper, we propose a probabilistic relation induction model that addresses both problems. First, to reduce the number of spurious instances that are detected, our model learns a soft constraint on which words are likely to occur as source and target words. Second, we use a (Bayesian estimation of a) Gaussian distribution over translation vectors to encode which features of word meaning are most important for the considered relation. We also consider a variant that is not based on translations and merely assumes that there is a linear mapping between source and target words, which we formalize using Bayesian linear regression. Being more general than the translation based approach, this model can potentially be more faithful, but it needs a larger number of training instances to be effective.

2 Related Work

Predicting Relations. At least three different types of approaches have been studied for predicting relations that are missing from a given knowledge base. First, there is a large body of work on relation extraction from text, for instance by using the known instances as a form of distant supervision (Mintz et al., 2009; Riedel et al., 2010; Surdeanu et al., 2012). The second type of approach relies on modeling statistical dependencies among the known instances of the considered relations, for instance by learning latent representations (Kok and Domingos, 2007; Speer et al., 2008; Nickel et al., 2012; Riedel et al., 2013; Bordes et al., 2013; Wang et al., 2014; Yang et al., 2015), or probabilistic rules (Schoenmackers et al., 2010; Lao et al., 2011; Wang et al., 2015). The third type of approach, which is the focus of this paper and is reviewed in more detail below, relies on vector space representations. A standard approach is to model relations as translations in the vector space (Mikolov et al., 2013b), although various other approaches have also been investigated (Weeds et al., 2014).

These three types of methods are highly complementary. While relation extraction methods can predict very fine-grained relations, they require that at least one sentence in the corpus states the relation explicitly. Statistical methods can predict relations even without access to a text corpus, but they are limited to predicting what can plausibly derived from what is already known. From a knowledge base completion point of view, the main appeal of word embeddings is that they may be able to reveal commonsense relationships which are rarely stated explicitly in text.

Modeling Relations in a Vector Space. It has been shown that word embeddings can be used to complete analogy questions of the form *a*:*b*::*c*:?, asking for a word that relates to c in the same way that b relates to a (e.g. *france*:*wine*::*germany*:?), by predicting the word w that maximizes $\cos(\mathbf{b} - \mathbf{a} + \mathbf{c}, \mathbf{w})$ (Mikolov et al., 2013b; Pennington et al., 2014). Similarly, several types of interpretable features can be modeled as directions in word embeddings. For example, in (Rothe and Schütze, 2016), it was shown that word embeddings can be decomposed in orthogonal subspaces that capture particular semantic properties, including a one-dimensional subspace (i.e. a direction) that encodes polarity. Along similar lines, in (Kim and de Marneffe, 2013) it was found that the direction defined by a word and its antonym (e.g. "good" and "bad") can be used to derive adjectival scales (e.g. bad $<$ okay $<$ good $<$ excellent). In (Gupta et al., 2015), it was shown that many types of numerical attributes can be predicted from word embeddings (e.g. GDP, fertility rate and CO_2 emissions of a country) using linear regression, again supporting the view that directions can model meaningful relations. Finally, in (Derrac and Schockaert, 2015) an unsupervised method was proposed to decompose domain-specific vector spaces into interpretable directions. For instance, in a space of movies, directions modeling terms such as "scary", "romantic" or "hilarious" were found.



(a) Superlative degree relation.

(b) Meronymy relation.

Figure 1: Superlative degree and Meronymy relations.

Several authors have focused on extracting hypernym relations from word embeddings. To decide whether a word h is a hypernym of w, in (Baroni et al., 2012) it is proposed to use an SVM with a polynomial kernel, using the concatenation of $\mathbf{h}$ and $\mathbf{w}$ as feature vector. In (Roller et al., 2014) it was shown that vector differences can lead to good results with a linear SVM, provided that the vectors are normalized, and that the squared differences of each coordinate are added as additional features. Intuitively, this allows the SVM classifier to express that h and w need to be different in particular aspects (using the vector differences) but similar in other aspects (using the squared differences). Some authors have also proposed to identify hypernyms by using word embedding models that represent words as regions or densities (Erk, 2009; Vilnis and McCallum, 2015; Jameel and Schockaert, 2017).

Beyond hypernyms, most work has focused on completing analogies. The problem of relation induction was studied in (Vylomova et al., 2016), where a linear SVM trained on vector differences was used. While strong results were obtained for several relations in a controlled setting (e.g. predicting which among a given set of relations a word pair belongs to), many false positives were obtained when random word pairs were added as negative examples. A variant of the relation induction problem was also studied in (Drozd et al., 2016), where the focus was on predicting the target word t given a valid source word s (as in analogy completion), given a set of training instances (as in relation induction). Two strong baselines were introduced in that paper, which will be explained in Section 4.1.

3 Modeling Relations

In this section, we propose two models for relation induction. We assume that we are given a set of pairs $\{(s_1, t_1), ..., (s_n, t_n)\}$ as training data, and we need to determine whether a given pair of words (s, t) are related in the same way.

3.1 Translation Model

Intuition. The source words $s_1, ..., s_n$ typically belong to some semantic or syntactic category, and their representations can often be expected to approximately belong to some relatively low-dimensional subspace of the word embedding. This is illustrated in Figure 1a for the 'superlative degree' relation from the BATS dataset[2] (see Section 4). Irrespective of the translation $\mathbf{t} - \mathbf{s}$, if (s, t) is a valid relation instance, we would expect $\mathbf{s}$ to be approximately in the same subspace as $\mathbf{s_1}, ..., \mathbf{s_n}$, and similar for the target words. Imposing this condition will intuitively allow us to ensure that only pairs where s and t are of the correct type are considered. As we will see, this will substantially reduce the number of false positives that are predicted by the model.

In Figure 1a we can see that there is no single translation vector that perfectly models the relation, although the translation vectors $\mathbf{t_i} - \mathbf{s_i}$ are all rather similar. This can be naturally modeled by considering

[2]In particular, the figure shows the two first principle components of the set $\{\mathbf{s_1}, ..., \mathbf{s_n}, \mathbf{t_1}, ..., \mathbf{t_n}\}$. As word vectors, we used the Skip-gram embedding that was learned from the Google news corpus.

a probability distribution over vector translations. In the example, this distribution would have a small variance along directions which are orthogonal to the average translation vector (as most vectors are almost parallel), but a larger variance along the direction of the average translation vector itself (as the translation vectors have varying lengths).

Putting everything together, we want to accept (s, t) as a valid instance if (i) $\mathbf{s}$ and $\mathbf{t}$ are sufficiently similar to the vector representations of the given source and target words, and (ii) the translation $\mathbf{t} - \mathbf{s}$ has a sufficiently high probability.

Model description. Let us write δ_s and δ_t be the event that the source word s and target word t are of the correct type, and let θ_{st} be the event that s and t are in the considered relation. Note that θ_{st} entails δ_s and δ_t, hence we have:

$$P(\theta_{st}|\mathbf{s}, \mathbf{t}) = P(\delta_s|\mathbf{s}) \cdot P(\delta_t|\mathbf{t}) \cdot P(\theta_{st}|\mathbf{s}, \mathbf{t}, \delta_s, \delta_t)$$

By making the core assumption that the relation can be modelled in terms of vector translations, together we Bayes' rule, we obtain:

$$P(\theta_{st}|\mathbf{s}, \mathbf{t}) \propto \frac{f(\mathbf{s}|\delta_s)}{f(\mathbf{s})} \cdot \frac{f(\mathbf{t}|\delta_t)}{f(\mathbf{t})} \cdot \frac{f(\mathbf{t} - \mathbf{s}|\theta_{st})}{f(\mathbf{t} - \mathbf{s}|\delta_s, \delta_t)}$$

We now discuss how the densities occurring in this latter expression can be estimated. First, $f(\mathbf{s}|\delta_s)$ models the density of words that can appear as source words in instances of the considered relation (i.e. the words which are intuitively of the correct type). We make the simplifying assumption that the vector representations of these words follow a Gaussian distribution. Note, however, that the number of available training instances n is typically small, and in particular smaller than the number of dimensions. In such cases, the covariance matrix cannot be reliably estimated, and we have to impose strong regularity assumptions. A common approach, which we will adopt, is to only allow diagonal covariance matrices. In this case, we have

$$f(\mathbf{s}|\delta_s) = \prod_{i=1}^{m} f(x_i^s|\delta_s)$$

where m is the number of dimensions in the word embedding, and we write x_i^s for the i^{th} coordinate of $\mathbf{s}$. The density $f(x_i^s|\delta_s)$ is then a univariate Gaussian distribution with an unknown mean and variance. Given the typically small number of training examples, a maximum likelihood estimation of the parameters would lead to a form of over-fitting. To address this, we use a Bayesian approach to estimate $f(x_i^s|\delta_s)$ as follows:

$$\int G(x_i^s; \mu, \sigma^2) NI\chi^2(\mu, \sigma^2|\mu_0, \kappa_0, \nu_0, \sigma_0^2) d\mu d\sigma$$

where G represents the Gaussian distribution and $NI\chi^2$ is the normal inverse χ^2 distribution. Note that in the standard relation induction setting, we have no prior information about μ and σ^2. The parameters $\mu_0, \kappa_0, \nu_0, \sigma_0^2$ can then be chosen such that they correspond to a flat prior; we refer to (Murphy, 2007) for details. For this choice, it can be shown that the integral evaluates to:

$$f(x_i^s|\delta_s) = t_{n-1}\Big(\overline{x_i}, \frac{(n+1)\sum_{j=1}^{n}(x_i^{s_j} - \overline{x_i})^2}{n(n-1)}\Big)$$

where $\overline{x_i} = \frac{1}{n}\sum_{j=1}^{n} x_i^{s_j}$ and t_{n-1} is the Student t-distribution with $n-1$ degrees of freedom. The density $f(\mathbf{t}|\delta_t)$ is evaluated in the same way. To evaluate $f(\mathbf{s})$ and $f(\mathbf{t})$, we make the simplifying assumption that the overall distribution of word vectors also follows a Gaussian distribution. Given the typical vocabulary sizes, we can reliably use the sample mean and covariance matrix as estimations of the parameters of this Gaussian.

The density $f(\mathbf{t}-\mathbf{s}|\theta_{st})$ is estimated similarly to $f(x_i^s|\delta_s)$, which corresponds to an assumption that the translations $\mathbf{t}-\mathbf{s}$ also follow a Gaussian distribution. In particular, we estimate $f(\mathbf{t}-\mathbf{s}|\theta_{st})$ as $\prod_{i=1}^{m} f(x_i^s - x_i^t|\theta_{st})$, where x_i^s is again the i^{th} coordinate of $\mathbf{s}$ and similar for x_i^t. Each univariate Gaussian $f(x_i^s -$

$x_i^t|\theta_{st})$ is then again estimated using the t-distribution, from the set of data points $\{x_i^{s_1} - x_i^{t_1}, ..., x_i^{s_n} - x_i^{t_n}\}$. We similarly estimate $f(\mathbf{t} - \mathbf{s}|\delta_s, \delta_t)$ as $\prod_{i=1}^m f(x_i^s - x_i^t|\delta_s, \delta_t)$. The mean of $f(x_i^s - x_i^t|\delta_s, \delta_t)$ is the same as the mean of $f(x_i^s - x_i^t|\theta_{st})$, but the variance is estimated from the differences $x_i^{s_l} - x_i^{t_k}$ corresponding to n randomly sampled pairs (s_l, t_k), where s_l and t_k are respectively sampled from the source and target words occurring in the training examples. Note that if the assumption that the considered relation corresponds to a translation is wrong, we can expect the variance of $f(x_i^s - x_i^t|\theta_{st})$ and $f(x_i^s - x_i^t|\delta_s, \delta_t)$ to be similar, in which case the last factor in the evaluation of $P(\theta_{st}|\mathbf{s}, \mathbf{t})$ will be approximately 1. In other words, the model implicitly takes into account how much the translation assumption appears to be satisfied.

3.2 Regression Model

Intuition. While there are several relations that can be approximately modelled as vector translations, there are many other relations for which this is not the case. To illustrate this, Figure 1b has been constructed in the same way as Figure 1a, but using the instances of the meronymy relation from the DiffVec dataset (Gladkova et al., 2016). This figure clearly suggests that we cannot expect an approach that models meronymy in terms of translations to perform well (for this word embedding). As an alternative, in this section we propose a model which weakens the translation assumption, and merely assumes that there is a linear mapping from source to target words. We can expect that the resulting model should perform well for a broader set of relations; for example, as we will see in Section 4, the meronymy relation can be modelled rather accurately in this way. The most important drawback of this approach is that we need more training examples to reliably estimate a linear mapping than to estimate a translation. In fact, while a translation can be estimated from a single example, we can only learn a linear mapping if the number of training examples is higher than the number of dimensions. We address this issue by reducing the number of dimensions of the source space, based on the available number of training examples.

Model description. We now estimate the probability that (s, t) is a valid instance of the considered relation as follows:

$$\begin{aligned} P(\theta_{st}|\mathbf{s}, \mathbf{t}) &\propto \frac{f(\mathbf{s}|\delta_s)}{f(\mathbf{s})} \cdot \frac{f(\mathbf{t}|\delta_t)}{f(\mathbf{t})} \cdot \frac{f(\mathbf{t}|\mathbf{s}, \theta_{st})}{f(\mathbf{t}|\mathbf{s}, \delta_s, \delta_t)} \\ &= \frac{f(\mathbf{s}|\delta_s)}{f(\mathbf{s})} \cdot \frac{f(\mathbf{t}|\delta_t)}{f(\mathbf{t})} \cdot \frac{f(\mathbf{t}|\mathbf{s}, \theta_{st})}{f(\mathbf{t}|\delta_t)} \\ &= \frac{f(\mathbf{s}|\delta_s)}{f(\mathbf{s})} \cdot \frac{f(\mathbf{t}|\mathbf{s}, \theta_{st})}{f(\mathbf{t})} \end{aligned}$$

The densities $f(\mathbf{s}|\delta_s)$, $f(\mathbf{s})$ and $f(\mathbf{t})$ are estimated as before. We estimate $f(\mathbf{t}|\mathbf{s}, \theta_{st})$ as $\prod_{i=1}^m f(x_i^t|\mathbf{s}, \theta_{st})$, where x_i^t is again the i^{th} coordinate of $\mathbf{t}$. Each univariate density $f(x_i^t|\mathbf{s}, \theta_{st})$ is estimated using a Bayesian linear regression model that predicts the possible representations of the target word from $\mathbf{s}$. However, this is only feasible if $\mathbf{s}$ has at most $n - 2$ coordinates. Therefore, we use a low-rank approximation of the source word representations, as follows.

Let A be a matrix whose rows are the vectors $\mathbf{s_1}, ..., \mathbf{s_n}$ and let $A = U\Sigma V^T$ be the SVD decomposition of A. Let $\mathbf{v_1}, ..., \mathbf{v_k}$ be the first k row vectors of V, for some $k < n - 1$. For a given vector $\mathbf{p}$, we can think of $\mathbf{p}^S = (\mathbf{p} \cdot \mathbf{v_1}, ..., \mathbf{p} \cdot \mathbf{v_k})$ as the representation of $\mathbf{p}$ in the source subspace. Given that we typically need far fewer dimensions to represent the source space than the total number of dimensions in the word embedding, we should be able to predict the target word from $\mathbf{s}^S$, even for relatively small values of k. In any case, the choice of k represents a trade-off: the lower the value of k, the better we can characterize the uncertainty underlying our predictions, but the less information we have for making predictions. In the experiments, we have used $k = \frac{n-1}{2}$. We estimate $f(x_i^t|\mathbf{s}, \theta_{st})$ as follows:

$$\begin{aligned} &\int G(x_i^t; \mathbf{s}^*\beta, \sigma^2) \cdot \\ &G(\beta; (X^TX)^{-1}X^T\mathbf{b^i}, (X^TX)^{-1}\sigma^2) \cdot \\ &NI\chi^2(\sigma^2|\nu_0, \sigma_0^2) d\beta d\sigma \end{aligned}$$

where $\mathbf{b^i} = (x_i^{t_1}, ..., x_i^{t_n})$, X is composed of the first k columns of $U\Sigma$ (with U and Σ the matrices from the SVD decomposition of A) with an additional 1 appended at the end of each row for the bias term, and $\mathbf{s}^*$ is the vector $\mathbf{s}^S$ with an additional 1 appended. Assuming a flat prior on the residual variance σ^2, the parameters ν_0 and σ_0^2 can be estimated from the training data as:

$$\nu_0 = n - k - 1$$
$$\sigma_0^2 = \frac{1}{n-k-1}(\mathbf{b^i} - X\hat{\beta})^T(\mathbf{b^i} - X\hat{\beta})$$

with $\hat{\beta}$ the least squares solution.

4 Evaluation

In this section, we experimentally compare the two proposed models with a number of baseline methods from the literature. The relations we consider are taken from three standard benchmark datasets, each containing a mixture of syntactic and semantic relationships: (i) the Google Analogy Test Set (Google), which contains 14 types of relations with a varying number of instances per relation (Mikolov et al., 2013a), (ii) the Bigger Analogy Test Set (BATS), which contains 40 relations with 50 instances per relation (Gladkova et al., 2016), and (iii) the DiffVec Test Set (DV), which contains 36 relations with a varying number of instances per relation (Vylomova et al., 2016). We report results for two embeddings that have been learned using Skip-gram, one from the Wikipedia dump of 2 November 2015 (SG-Wiki) and one from a 100B words Google News data set[3] (SG-GN). We also use two embeddings that have been learned with GloVe, one from the same english Wikipedia dump (GloVe-Wiki) and one from the 840B words Common Crawl data set[4] (GloVe-CC).

For relations with at least 10 instances, we use 10-fold cross validation, whereas for relations with less than 10 instances, we use a leave-one-out evaluation. Note that the considered datasets only contain positive examples. To generate negative test examples, we use four strategies. First, for each pair (s, t) in the test fold, we add (t, s) as a negative example. Second, for each source word s in the test fold, we randomly sample two target words from the test fold (provided that the test fold contains enough pairs), which do not occur together with s, and for each such target word t, we add (s, t) as a negative example. Third, for each positive example, we randomly select a pair from the other relations. Finally, for each positive example, we generate a random word pair from the words available in the dataset. This ensures that the evaluation involves negative examples that consist of related words, as well as negative examples that consist of unrelated words.

If we consider the task as a classification task, i.e. deciding for an unseen pair (s, t) whether it has the considered relation, we need to select a threshold, as the considered methods only produce a confidence score. To choose this threshold, we randomly select 10% of the 9 training folds as validation data. In the results below, we separately report precision, recall and F1. We can also evaluate this task as a ranking problem, where we merely evaluate to what extent each method assigns the highest score to the correct pairs. In that case, we use mean average precision (MAP).

4.1 Baselines

The first baseline we consider is the 3CosAvg (or 3CA) method proposed in (Drozd et al., 2016), which essentially treats the relation induction problem like an analogy completion problem, where we use the average translation vector across all pairs (s_i, t_i) from the training data. In particular, this method assigns the following score to the test pair (s, t):

$$\mathit{score}_{3CA}(t, s) = \cos\left(\mathbf{s} + \frac{\sum_i \mathbf{t_i} - \mathbf{s_i}}{n}, \mathbf{t}\right)$$

Another method proposed in (Drozd et al., 2016), called LRCos (or LRC), is based on the assumption that (s, t) is likely correct if $\cos(\mathbf{s}, \mathbf{t})$ is high and t is of the correct type, where a logistic regression

[3] https://code.google.com/archive/p/word2vec/
[4] https://nlp.stanford.edu/projects/glove/

Table 2: Results of the relation induction experiments (macro-averages).

		SG-Wiki			GloVe-Wiki			SG-GN			GloVe-CC		
		Google	BATS	DV	Google	BATS	DV	Google	BATS	DV	Google	BATS	DV
3CA	Pr	0.651	0.433	0.210	0.510	0.330	0.205	0.466	0.377	0.214	0.642	0.428	0.225
3CA	Rec	0.602	0.620	0.491	0.541	0.635	0.478	0.592	0.619	0.507	0.665	0.609	0.506
3CA	F1	0.626	0.510	0.294	0.525	0.434	0.287	0.522	0.469	0.301	0.653	0.503	0.311
3CA	MAP	0.736	0.463	0.274	0.663	0.376	0.261	0.615	0.424	0.273	0.741	0.480	0.288
LRC	Pr	0.516	0.475	0.374	0.366	0.256	0.166	0.488	0.486	0.389	0.427	0.383	0.257
LRC	Rec	0.646	0.672	0.527	0.527	0.577	0.439	0.670	0.646	0.570	0.659	0.596	0.474
LRC	F1	0.573	0.557	0.437	0.432	0.355	0.241	0.565	0.555	0.462	0.518	0.466	0.333
LRC	MAP	0.710	0.580	0.519	0.508	0.322	0.265	0.713	0.614	0.545	0.628	0.481	0.389
SVM	Pr	0.407	0.336	0.198	0.383	0.365	0.215	0.464	0.398	0.276	0.407	0.381	0.225
SVM	Rec	0.680	0.417	0.412	0.628	0.461	0.376	0.646	0.531	0.384	0.671	0.501	0.408
SVM	F1	0.509	0.372	0.267	0.476	0.408	0.274	0.540	0.455	0.321	0.507	0.433	0.290
SVM	MAP	0.494	0.366	0.283	0.502	0.404	0.298	0.611	0.467	0.366	0.502	0.425	0.296
Trans	Pr	0.794	0.627	0.449	0.635	0.445	0.284	0.741	0.660	0.498	0.744	0.571	0.378
Trans	Rec	0.649	0.708	0.563	0.618	0.620	0.446	0.771	0.705	0.604	0.713	0.689	0.552
Trans	F1	0.714	0.665	0.500	0.626	0.518	0.347	0.756	0.682	0.546	0.728	0.624	0.449
Trans	MAP	0.906	0.729	0.596	0.791	0.541	0.387	0.890	0.773	0.635	0.898	0.678	0.520
Regr	Pr	0.668	0.474	0.410	0.536	0.281	0.259	0.627	0.476	0.469	0.613	0.401	0.357
Regr	Rec	0.603	0.470	0.471	0.580	0.403	0.422	0.665	0.449	0.537	0.646	0.439	0.467
Regr	F1	0.634	0.472	0.439	0.557	0.331	0.321	0.646	0.462	0.501	0.629	0.419	0.404
Regr	MAP	0.834	0.618	0.570	0.741	0.434	0.381	0.801	0.639	0.621	0.793	0.549	0.506

classifier was trained on the target words $\{t_1, ..., t_n\}$ to predict the probability that t is a valid 'target word'. To adapt this method to our setting, we also need to consider the probability that s is a valid 'source word' (which is not needed in the analogy completion setting considered in (Drozd et al., 2016), since s is always given as a valid source word). To allow for a more direct comparison with our methods, instead of using a logistic regression classifier, we will use our Bayesian estimation for the probability that s and t are of the correct type. In particular, we use the score $score_{LRC}(t, s)$ defined as follows:

$$\frac{P(\mathbf{s}|\delta_s)}{P(\mathbf{s})} \cdot \frac{P(\mathbf{t}|\delta_t)}{P(\mathbf{t})} \cdot \cos(\mathbf{s}, \mathbf{t})$$

As our final baseline, we train a linear SVM classifier using the training pairs $(s_1, t_1), ..., (s_n, t_n)$ as positive examples. Following (Vylomova et al., 2016), we use negative examples of the form (t_i, s_i), obtained by swapping the position of source and target word, as well as negative examples of the form (s_i, t_j), obtained by swapping t_i by the target word of another instance (while ensuring that (s_i, t_j) does not appear in the training data as well). Finally, we also add n random word pairs as negative examples. The C parameter (i.e. the cost parameter for mislabeled examples) of the SVM is tuned for each relation separately (choosing values from $\{0.01, 0.1, 1, 10, 100\}$), using the same validation data that is used for selecting the thresholds in the other models. To address class imbalance, negative examples were weighted by the ratio of positive to negative examples.

4.2 Results

The results are summarized in Table 2. As can be observed, our translation model consistently outperforms all other methods in both MAP and F1 score. Moreover, the regression model consistently outperforms the baselines in terms of MAP score, and outperforms the baselines for the Google and DV test sets in terms of F1 score (with the exception of the Glove-CC embedding, where 3CA has a better F1 score for the Google test set). The performance of the baselines varies, with 3CA generally performing best for the Google test set and LRC generally performing best for DiffVec.

To compare the performance of the methods across different types of relations, Table 3 contains the MAP scores for a number of selected relations from the DiffVec and BATS test sets, for the SG-GN word embedding. For the BATS dataset, the translation model consistently outperforms the baseline across all relations (including the relations that are not shown in the table). In the case of DiffVec there are a few exceptions, as can be seen in Table 3, but in such cases the differences with the translation model are small. The regression model also outperforms the baselines in most cases, but there are a few

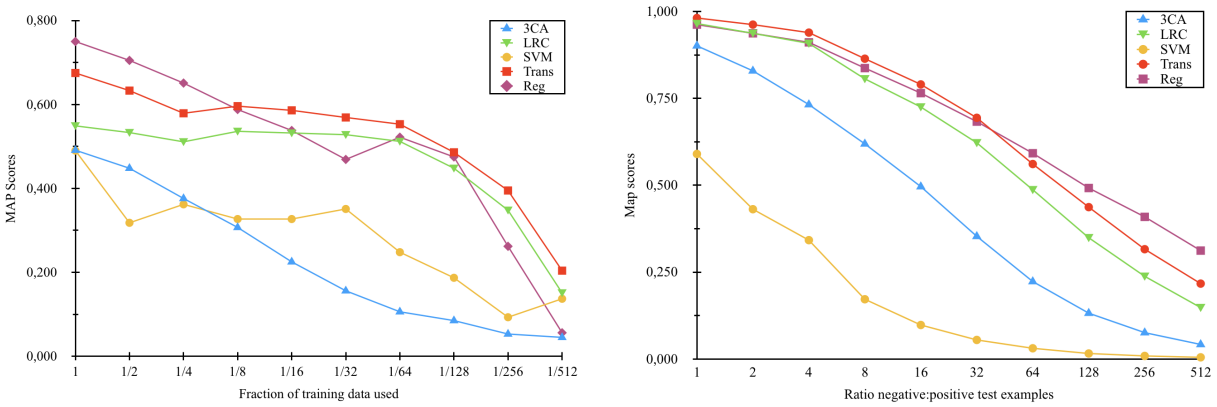
Table 3: MAP scores for selected relations (based on the SG-GN word embedding).

DiffVec	3CA	LRC	SVM	Trans	Regr
Object:State	0.11	**0.59**	0.27	0.57	0.50
CompensatoryAction	0.08	0.58	0.41	0.62	**0.66**
IntendedAction	0.14	0.53	0.35	**0.62**	0.58
Prevention	0.17	0.66	0.55	**0.71**	0.62
Collective noun	0.37	0.58	0.39	0.56	**0.69**
Event	0.61	0.72	0.40	0.74	**0.94**
Hyper	0.48	0.55	0.39	0.75	**0.91**
Lvc	0.11	0.71	**0.77**	0.74	0.22
Mero	0.49	0.55	0.40	0.67	**0.83**
Prefix re	0.40	0.49	0.30	**0.72**	0.68
Expression	0.14	0.82	0.51	0.81	**0.82**
Knowledge	0.14	0.69	0.51	**0.72**	0.70
Plan	0.08	0.55	0.29	0.57	**0.62**
Representation	0.09	**0.50**	0.40	0.49	0.39
Contiguity	0.11	0.53	0.30	**0.63**	0.61
Sign:Significant	0.08	0.38	0.30	**0.39**	0.38
Loc:Action/Activity	0.22	0.73	0.51	**0.75**	0.72
Loc:Process/Product	0.13	0.41	0.57	0.48	**0.66**
Verb 3rd	0.96	0.61	0.40	**0.98**	0.96
Verb Past	0.93	0.64	0.32	**0.99**	0.90
BATS	**3CA**	**LRC**	**SVM**	**Trans**	**Regr**
Regular plurals	0.83	0.59	0.40	**0.88**	0.79
Comparative degree	0.93	0.65	0.47	**0.96**	0.88
Superlative degree	0.87	0.71	0.61	**0.93**	0.87
Infinitive: past	0.78	0.58	0.46	**0.96**	0.72
3Ps.Sg: past	0.72	0.70	0.56	**0.98**	0.95
Noun+less	0.38	0.58	0.43	0.62	**0.63**
Un+adj	0.30	0.55	0.35	**0.77**	0.69
Over+adh./Ved	0.20	0.63	0.38	0.74	**0.77**
Re+verb	0.39	0.66	0.37	**0.82**	0.74
Verb+able	0.21	0.63	0.57	0.76	**0.76**
Verb+ment	0.34	0.54	0.47	**0.78**	0.69
Hypernyms animals	0.43	0.71	0.64	**0.85**	0.28
Hypernyms misc	0.35	0.64	0.54	**0.77**	0.23
Meronyms substance	0.23	0.51	0.36	**0.61**	0.27
Synonyms intensity	0.34	0.50	0.29	**0.67**	0.23
Synonyms exact	0.28	0.45	0.26	**0.51**	0.19
Antonyms binary	0.22	0.44	0.31	**0.50**	0.24
Capitals	0.25	0.68	0.47	**0.75**	0.74
Country:language	0.19	0.64	0.52	0.66	**0.71**
Nationalities	0.22	0.77	0.60	**0.85**	0.63
Animals sounds	0.25	0.65	0.45	0.66	**0.77**
thing:color	0.44	0.77	0.66	0.76	**0.79**
Male:female	0.61	0.61	0.47	**0.84**	0.72

exceptions where it performs much worse (e.g. *Lvc* for DiffVec, and *Hypernyms-animals*, *Meronyms-substance*, *Synonyms-intensity* and *Antonyms-binary* in the case of BATS). While the regression model is outperformed by the translation model on average, there are several cases where it performs better. For relations such as *Event*, *Hyper* and *Mero* from DiffVec, where the number of examples is rather large (resp. 3583, 1173, 2825), we can see that the regression model actually substantially outperforms the translation model. The main weakness of the regression model is that it needs more training data: while a vector translation can be estimated from a single training example, learning an arbitrary linear mapping requires the number of training examples to be larger than the number of dimensions. While this can be addressed by using a low-dimensional approximation of the source word, information is lost in this way.

The impact of the amount of training data on the relative performance of the regression model is further analyzed in Figure 2a, taking the Mero relation from DiffVec as an example (for SG-GN). While the regression model performs best if all the training data is used, the translation model is less sensitive to the amount of training data, and starts outperforming the regression model if less than 1/8th of the training data is used.

We also noticed that the performance of the methods crucially depends on the number of negative test



(a) MAP scores for the Mero relation, in function of training data (as a fraction of the total amount).

(b) MAP scores for the Mero relation, in function of the ratio of negative to positive test examples.

Figure 2: MAP scores for the Mero relation, in function of training data/ratio of negative to positive test examples.

examples that are considered, even if these negative examples are randomly chosen word pairs. This is illustrated in Figure 2b, which shows the MAP scores for the Mero relation from DiffVec (for SG-GN), for different numbers of negative examples. For these experiments, as negative examples, we only considered random word pairs. The total number of such negative examples was varied from 1 times the number of positive examples, which is equal to the number of positive examples, to 512 times the number of positive examples. While the performance of each of the methods is affected by the number of such negative examples, the performance of the baseline models drops more quickly. Moreover, the regression model is more robust than the translation model, and starts outperforming it if the ratio of negative examples to positive examples is higher than 32:1.

5 Conclusions

We have proposed two probabilistic models for identifying word pairs that are in a given relation. The first model is based on the common assumption that lexical relations correspond to vector translations in a word embedding. The other model is based on linear regression, relying on the weaker assumption that there is a linear relationship between the source and target words of the considered relation. Both models implicitly factor in whether their underlying assumption is satisfied, and could thus easily be used in combination with each other, or with additional models. In our experimental evaluation, we have found both models to outperform existing approaches, with the translation model outperforming the regression model on average. However, in cases where sufficient training data is available, the regression model tends to perform better. We have also found some evidence that the regression model is better able to handle cases of extreme imbalance between positive and negative examples.

There are several interesting avenues for future work. First, a number of variants of the proposed models can be developed. For example, a model based on vector concatenations could intuitively model similar kinds of relationships as the regression model. However, in the case of vector concatenations, we can no longer use a diagonal covariance matrix, as that would mean that no interactions between source and target words are being captured. One solution could be to use a low-rank approximation of the vector concatenations and estimate full covariance matrices in a lower-dimensional space. Another interesting option to explore would be to estimate prior probabilities from coarser grained relations for which more training data is available. For example, we could learn a generic model for causal relations, and use that as a prior for the specific types of causal relationships that are considered in the DiffVec test set. It may even be useful to learn priors capturing e.g. syntactic relations, which would intuitively amount to finding a subspace of the embedding that relates to syntactic features.

Acknowledgments

This work was supported by ERC Starting Grant 637277.

References

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 23–32.

A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, pages 2787–2795.

J. Derrac and S. Schockaert. 2015. Inducing semantic relations from conceptual spaces: a data-driven approach to plausible reasoning. *Artificial Intelligence*, pages 74–105.

Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. 2016. Word embeddings, analogies, and machine learning: Beyond king - man + woman = queen. In *Proceedings of the 26th International Conference on Computational Linguistics*, pages 3519–3530.

Katrin Erk. 2009. Representing words as regions in vector space. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 57–65.

Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. 2016. Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *Proceedings of the Student Research Workshop at NAACL 2016*, pages 8–15.

Abhijeet Gupta, Gemma Boleda, Marco Baroni, and Sebastian Padó. 2015. Distributional vectors encode referential attributes. In *Proc. EMNLP*, pages 12–21.

Shoaib Jameel and Steven Schockaert. 2017. Modeling context words as regions: An ordinal regression approach to word embedding. In *Proceedings of the 21st Conference on Computational Natural Language Learning*, pages 123–133.

Joo-Kyung Kim and Marie-Catherine de Marneffe. 2013. Deriving adjectival scales from continuous space word representations. In *Proc. EMNLP*, pages 1625–1630.

Stanley Kok and Pedro Domingos. 2007. Statistical predicate invention. In *Proc. ICML*, pages 433–440.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proc. EMNLP*, pages 529–539.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proc. NAACL-HLT*, pages 746–751.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the 47th Annual Meeting of the ACL*, pages 1003–1011.

Kevin Murphy. 2007. Conjugate Bayesian analysis of the Gaussian distribution. Technical report, University of British Columbia.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing YAGO: Scalable machine learning for linked data. In *Proceedings of the 21st International Conference on World Wide Web*, pages 271–280.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proc. ECML/PKDD*, pages 148–163.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proc. HLT-NAACL*, pages 74–84.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proc. COLING*, pages 1025–1036.

Sascha Rothe and Hinrich Schütze. 2016. Word embedding calculus in meaningful ultradense subspaces. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 512–517.

Stefan Schoenmackers, Jesse Davis, Oren Etzioni, and Daniel S. Weld. 2010. Learning first-order horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1088–1098.

Robert Speer, Catherine Havasi, and Henry Lieberman. 2008. Analogyspace: reducing the dimensionality of common sense knowledge. In *Proceedings of the 23rd AAAI Conference on Artificial intelligence*, pages 548–553.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465.

Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding. In *Proceedings of the International Conference on Learning Representations*.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119.

William Yang Wang, Kathryn Mazaitis, Ni Lao, and William W. Cohen. 2015. Efficient inference and learning in a large knowledge base - reasoning with extracted information using a locally groundable first-order probabilistic logic. *Machine Learning*, 100(1):101–126.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 2249–2259.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations*.

Contextual String Embeddings for Sequence Labeling

Alan Akbik
Zalando Research
Mühlenstraße 25
10243 Berlin

Duncan Blythe
Zalando Research
Mühlenstraße 25
10243 Berlin

Roland Vollgraf
Zalando Research
Mühlenstraße 25
10243 Berlin

{firstname.lastname}@zalando.de

Abstract

Recent advances in language modeling using recurrent neural networks have made it viable to model language as distributions over characters. By learning to predict the next character on the basis of previous characters, such models have been shown to automatically internalize linguistic concepts such as words, sentences, subclauses and even sentiment. In this paper, we propose to leverage the internal states of a trained character language model to produce a novel type of word embedding which we refer to as *contextual string embeddings*. Our proposed embeddings have the distinct properties that they (a) are trained without any explicit notion of words and thus fundamentally model words as sequences of characters, and (b) are *contextualized* by their surrounding text, meaning that the same word will have different embeddings depending on its contextual use. We conduct a comparative evaluation against previous embeddings and find that our embeddings are highly useful for downstream tasks: across four classic sequence labeling tasks we consistently outperform the previous state-of-the-art. In particular, we significantly outperform previous work on English and German named entity recognition (NER), allowing us to report new state-of-the-art F1-scores on the CoNLL03 shared task.

We release all code and pre-trained language models in a simple-to-use framework to the research community, to enable reproduction of these experiments and application of our proposed embeddings to other tasks: https://github.com/zalandoresearch/flair

1 Introduction

A large family of NLP tasks such as named entity recognition (NER) and part-of-speech (PoS) tagging may be formulated as *sequence labeling* problems; text is treated as a sequence of words to be labeled with linguistic tags. Current state-of-the-art approaches for sequence labeling typically use the LSTM variant of bidirectional recurrent neural networks (BiLSTMs), and a subsequent conditional random field (CRF) decoding layer (Huang et al., 2015; Ma and Hovy, 2016).

A crucial component in such approaches are *word embeddings*, typically trained over very large collections of unlabeled data to assist learning and generalization. Current state-of-the-art methods concatenate up to three distinct embedding types:

1. Classical word embeddings (Pennington et al., 2014; Mikolov et al., 2013), pre-trained over very large corpora and shown to capture latent syntactic and semantic similarities.

2. Character-level features (Ma and Hovy, 2016; Lample et al., 2016), which are not pre-trained, but trained on task data to capture task-specific subword features.

3. Contextualized word embeddings (Peters et al., 2017; Peters et al., 2018) that capture word semantics in context to address the polysemous and context-dependent nature of words.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1638–1649
Santa Fe, New Mexico, USA, August 20-26, 2018.

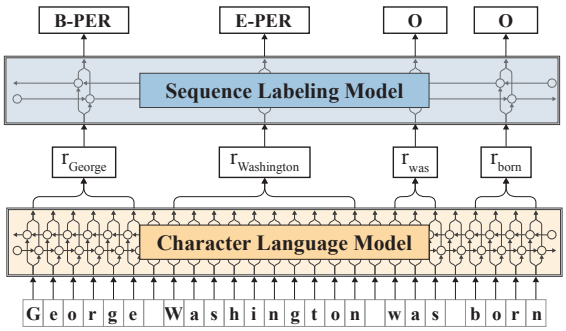


Figure 1: High level overview of proposed approach. A sentence is input as a character sequence into a pre-trained bidirectional character language model. From this LM, we retrieve for each word a contextual embedding that we pass into a vanilla BiLSTM-CRF sequence labeler, achieving robust state-of-the-art results on downstream tasks (NER in Figure).

Contextual string embeddings. In this paper, we propose a novel type of contextualized character-level word embedding which we hypothesize to combine the best attributes of the above-mentioned embeddings; namely, the ability to (1) pre-train on large unlabeled corpora, (2) capture word meaning in context and therefore produce different embeddings for polysemous words depending on their usage, and (3) model words and context fundamentally as sequences of characters, to both better handle rare and misspelled words as well as model subword structures such as prefixes and endings.

We present a method to generate such a contextualized embedding for any string of characters in a sentential context, and thus refer to the proposed representations as *contextual string embeddings*.

Neural character-level language modeling. We base our proposed embeddings on recent advances in neural language modeling (LM) that have allowed language to be modeled as distributions over sequences of characters instead of words (Sutskever et al., 2011; Graves, 2013; Kim et al., 2015). Recent work has shown that by learning to predict the next character on the basis of previous characters, such models learn internal representations that capture syntactic and semantic properties: even though trained without an explicit notion of word and sentence boundaries, they have been shown to generate grammatically correct text, including words, subclauses, quotes and sentences (Sutskever et al., 2014; Graves, 2013; Karpathy et al., 2015). More recently, Radford et al. (2017) showed that individual neurons in a large LSTM-LM can be attributed to specific semantic functions, such as predicting sentiment, without explicitly trained on a sentiment label set.

We show that an appropriate selection of hidden states from such a language model can be utilized to generate word-level embeddings that are highly effective in downstream sequence labeling tasks.

State-of-the-art sequence labeling. Based on this, we propose the sequence tagging architecture illustrated in Figure 1: each sentence is passed as a sequence of characters to a bidirectional character-level neural language model, from which we retrieve for each word the internal character states to create a contextual string embedding. This embedding is then utilized in the BiLSTM-CRF sequence tagging module to address a downstream NLP task (NER in the Figure).

We experimentally verify our approach in the classic sequence labeling tasks of named entity recognition for English and German, phrase chunking and part-of-speech tagging, and find that our approach reliably achieves state-of-the-art results. In particular, for both German and English NER, our approach significantly improves the state-of-the-art. But even for highly saturated tasks such as PoS tagging and chunking we find slight improvements over the already strong state-of-the-art (see Table 1).

We also find that our proposed embeddings on some tasks subsume previous embedding types, enabling simplified sequence labeling architectures. In addition to this, the character-level LM is compact and relatively efficient to train in comparison to word-level models. This allows us to easily train models for new languages or domains.

Contributions. To summarize, this paper proposes contextual string embeddings, a novel type of word embeddings based on character-level language modeling, and their use in a state-of-the-art sequence labeling architecture. Specifically, we

- illustrate how we extract such representations from a character-level neural language model, and

Task	PROPOSED	Previous best
NER English	**93.09**±0.12	*92.22*±0.1 (Peters et al., 2018)
NER German	**88.32**±0.2	*78.76* (Lample et al., 2016)
Chunking	**96.72**±0.05	*96.37*±0.05 (Peters et al., 2017)
PoS tagging	**97.85**±0.01	*97.64* (Choi, 2016)

Table 1: Summary of evaluation results for best configuration of proposed architecture, and current best published results. The proposed approach significantly outperforms previous work on the CONLL03 NER task for German and English and slightly outperforms previous works on CONLL2000 chunking and Penn treebank PoS tagging.

integrate them into a simplified sequence labeling architecture;

- present experiments in which we quantiatively evaluate the usefulness and inherent semantics of the proposed embeddings against previous embeddings and their stacked combinations in downstream tasks;

- report a new state-of-the-art on the CONLL03 NER task for English (**93.09** F1, ↑0.87 pp vs. previous best) and German (**88.33** F1, ↑9.56 pp vs. previous best), and state-of-the-art scores for chunking and PoS;

- release all code and pre-trained language models in a simple-to-use framework to the research community, to enable reproduction of these experiments and application of our proposed embeddings to other tasks.

This paper is structured as follows: we present our approach for extracting contextual string embeddings from character-level language models in Section 2. We evaluate our approach against prior work in Section 3. We then discuss the results and present an outlook into future work in Section 4.

2 Contextual String Embeddings

Our proposed approach passes sentences as sequences of characters into a character-level language model to form word-level embeddings. Refer to Figure 2 for an example illustration.

2.1 Recurrent Network States

Like recent work, we use the LSTM variant (Hochreiter and Schmidhuber, 1997; Graves, 2013; Zaremba et al., 2014) of recurrent neural networks (Sutskever et al., 2011) as language modeling architecture. These have been shown to far outperform earlier n-gram based models (Jozefowicz et al., 2016) due to the ability of LSTMs to flexibly encode long-term dependencies with their hidden state. We use characters as atomic units of language modeling (Graves, 2013), allowing text to be treated as a sequence of characters passed to an LSTM which at each point in the sequence is trained to predict the next character[1]. This means that the model possesses a hidden state for each character in the sequence.

Formally, the goal of a character-level language model is to estimate a good distribution $P(\mathbf{x}_{0:T})$ over sequences of charaters $(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T) =: \mathbf{x}_{0:T}$ reflecting natural language production (Rosenfeld, 2000). By training a language model, we learn $P(\mathbf{x}_t|\mathbf{x}_0, \ldots, \mathbf{x}_{t-1})$, an estimate of the predictive distribution over the next character given past characters. The joint distribution over entire sentences can then be decomposed as a product of the predictive distribution over characters conditioned on the preceeding characters:

$$P(\mathbf{x}_{0:T}) = \prod_{t=0}^{T} P(\mathbf{x}_t|\mathbf{x}_{0:t-1}) \tag{1}$$

[1]Note that character-level LM is different from *character-aware* LM (Kim et al., 2015) which still operates on the word-level, but also takes into account character-level features through an additional CNN encoding step.

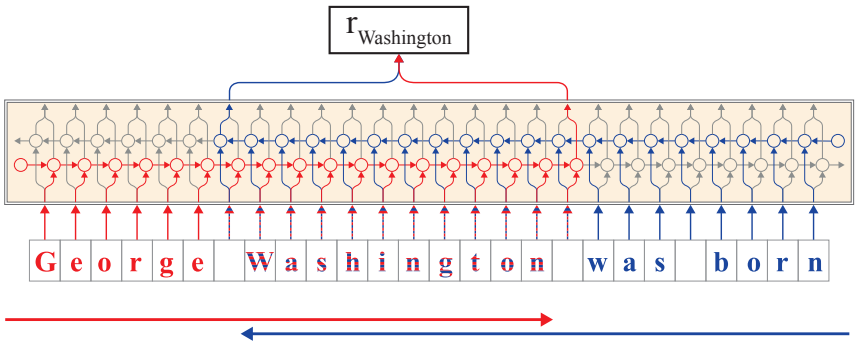


Figure 2: Extraction of a contextual string embedding for a word ("Washington") in a sentential context. From the forward language model (shown in red), we extract the output hidden state after the last character in the word. This hidden state thus contains information propagated from the beginning of the sentence up to this point. From the backward language model (shown in blue), we extract the output hidden state before the first character in the word. It thus contains information propagated from the end of the sentence to this point. Both output hidden states are concatenated to form the final embedding.

In the LSTM architecture, the conditional probability $P\left(\mathbf{x}_t|\mathbf{x}_{0:t-1}\right)$ is approximately a function of the network output $\mathbf{h}_t$.

$$P\left(\mathbf{x}_t|\mathbf{x}_{0:t-1}\right) \approx \prod_{t=0}^{T} P\left(\mathbf{x}_t|\mathbf{h}_t;\theta\right) \tag{2}$$

$\mathbf{h}_t$ represents the entire past of the character sequence. In an LSTM in particular, it is computed recursively, with the help of an additional recurrent quantity $\mathbf{c}_t$, the memory cell,

$$\begin{aligned}\mathbf{h}_t\left(\mathbf{x}_{0:t-1}\right) &= f_{\mathbf{h}}\left(\mathbf{x}_{t-1},\mathbf{h}_{t-1},\mathbf{c}_{t-1};\theta\right)\\ \mathbf{c}_t\left(\mathbf{x}_{0:t-1}\right) &= f_{\mathbf{c}}\left(\mathbf{x}_{t-1},\mathbf{h}_{t-1},\mathbf{c}_{t-1};\theta\right)\ ,\end{aligned}$$

where θ denotes all the parameters of the model. $\mathbf{h}_{-1}$ and $\mathbf{c}_{-1}$ can be initialized with zero or can be treated as part of the model parameters θ. In our model, a fully conected softmax layer (without bias) is placed ontop of $\mathbf{h}_t$, so the likelihood of every character is given by

$$P\left(\mathbf{x}_t|\mathbf{h}_t;\mathbf{V}\right) = \text{softmax}\left(\mathbf{V}\mathbf{h}_t+\mathbf{b}\right) \tag{3}$$

$$= \frac{\exp\left(\mathbf{V}\mathbf{h}_t+\mathbf{b}\right)}{\left\|\exp\left(\mathbf{V}\mathbf{h}_t+\mathbf{b}\right)\right\|_1} \tag{4}$$

where $\mathbf{V}$ and $\mathbf{b}$, weights and biases, are part of the model parameters θ (Graves, 2013; Jozefowicz et al., 2016).

2.2 Extracting Word Representations

We utilize the hidden states of a forward-backward recurrent neural network to create contextualized word embeddings. This means, alongside with the forward model (2), we also have a backward model, which works in the same way but in the reversed direction:

$$P^b\left(\mathbf{x}_t|\mathbf{x}_{t+1:T}\right) \approx \prod_{t=0}^{T} P^b\left(\mathbf{x}_t|\mathbf{h}_t^b,\theta\right) \tag{5}$$

$$\mathbf{h}_t^b = f_{\mathbf{h}}^b\left(\mathbf{x}_{t+1},\mathbf{h}_{t+1}^b,\mathbf{c}_{t+1}^b;\theta\right) \tag{6}$$

$$\mathbf{c}_t^b = f_{\mathbf{c}}^b\left(\mathbf{x}_{t+1},\mathbf{h}_{t+1}^b,\mathbf{c}_{t+1}^b;\theta\right) \tag{7}$$

Note that, in the following, we will use the superscript $\cdot^f$ to define $\mathbf{h}_t^f := \mathbf{h}_t$, $\mathbf{c}_t^f := \mathbf{c}_t$ for the forward model described in the previous section.

From this forward-backward LM, we concatenate the following hidden character states for each word: from the fLM, we extract the output hidden state after the last character in the word. Since the fLM

is trained to predict likely continuations of the sentence after this character, the hidden state encodes semantic-syntactic information of the sentence up to this point, including the word itself. Similarly, we extract the output hidden state before the word's first character from the bLM to capture semantic-syntactic information from the end of the sentence to this character. Both output hidden states are concatenated to form the final embedding and capture the semantic-syntactic information of the word itself as well as its surrounding context.

Formally, let the individual word-strings begin at character inputs with indices $t_0, t_1, \ldots, t_n$, then we define contextual string embeddings of these words as:

$$\mathbf{w}_i^{CharLM} := \begin{bmatrix} \mathbf{h}^f_{t_{i+1}-1} \\ \mathbf{h}^b_{t_i-1} \end{bmatrix} \tag{8}$$

We illustrate our approach in Figure 2 for a word in an example sentence, with the fLM in red and the bLM shown in blue.

Our approach thus produces embeddings from hidden states that are computed not only on the characters of a word, but also the characters of the surrounding context, since it influences the LM's ability to predict likely continuations of a sentence. As we later illustrate in Section 3.4, our proposed approach thus produces different embeddings for the same lexical word string in different contexts, and is able to accurately capture the semantics of contextual use together with word semantics itself.

2.3 Sequence Labeling Architecture

In the default configuration of our approach, the final word embeddings are passed into a BiLSTM-CRF sequence labeling module as proposed by Huang et al. (2015) to address downstream sequence labeling tasks. Then let us call the inputs to the BiLSTM g_l: $\mathbf{w}_0, \ldots, \mathbf{w}_n$. Then we have that:

$$\mathbf{r}_i := \begin{bmatrix} \mathbf{r}^f_i \\ \mathbf{r}^b_i \end{bmatrix} \tag{9}$$

Where $\mathbf{r}^f_i$ and $\mathbf{r}^f_i$ are the forward and backward output states of the BiLSTM g_l. The final sequence probability is then given by a CRF over the possible sequence labels y:

$$\widehat{P}(\mathbf{y}_{0:n}|\mathbf{r}_{0:n}) \propto \prod_{i=1}^{n} \psi_i(y_{i-1}, y_i, \mathbf{r}_i) \tag{10}$$

Where:

$$\psi_i(y', y, \mathbf{r}) = \exp(\mathbf{W}_{y',y}\mathbf{r} + \mathbf{b}_{y',y}) \tag{11}$$

Alternatively, we also experiment with directly applying a simple feedforward linear architecture (essentially multinomial logistic regression (Menard, 2018)). This configuration simply linearly projects the hidden states of the neural character LM to make predictions:

$$\mathbf{r}_i = \mathbf{W}_r\mathbf{w}_i + \mathbf{b}_r \tag{12}$$

Then the prediction of the label is given by:

$$P(\mathbf{y}_i = j|\mathbf{r}_i) = \text{softmax}(\mathbf{r}_i)[j] \tag{13}$$

Stacking Embeddings. Current sequence labeling models often combine different types of embeddings by concatenating each embedding vector to form the final word vectors. We similarly experiment with different stackings of embeddings; for instance in many configurations it may be benficial to add classic word embeddings to add potentially greater latent word-level semantics to our proposed embeddings. In this case, the final words representation is given by

$$\mathbf{w}_i = \begin{bmatrix} \mathbf{w}_i^{CharLM} \\ \mathbf{w}_i^{GloVe} \end{bmatrix} \tag{14}$$

Here $\mathbf{w}_i^{GloVe}$ is a precomputed GLOVE embedding (Pennington et al., 2014). We present different configurations of stacked embeddings in the next section for the purpose of evaluation.

3 Experiments

We conduct an experimental evaluation to assess in how far our proposed contextual string embeddings are useful for sequence labeling, and how they compare to existing embedding types. In addition, we aim to gain insights into the inherent semantics of the proposed embeddings.

Tasks. To this end, we evaluate our embeddings in four classic sequence labeling tasks in their default evaluation setups, namely named entity recognition in the CoNLL03 setup (Tjong Kim Sang and De Meulder, 2003), chunking in the CoNLL2000 setup (Tjong Kim Sang and Buchholz, 2000) and PoS tagging in the default Penn treebank setup described by Collins (2002). We evaluate NER to determine in how far our embeddings are helpful for shallow semantic tasks, while we use chunking and PoS to determine in how far they are helpful for shallow syntactic tasks. We also include the German NER task from the CoNLL03 setup to evaluate our embeddings in a language with richer morphology.

3.1 Experimental Setup

We utilize the BiLSTM-CRF sequence labeling architecture proposed by Huang et. al (2015) in all configurations of our comparative evaluation. In this architecture, we evaluate our proposed contextual string embeddings in stacked combinations with the three types of previous embeddings discussed in Section 1; namely (1) pre-trained static ("classic") word embeddings (Pennington et al., 2014), (2) task-trained (i.e. not pre-trained) static character features (Ma and Hovy, 2016; Lample et al., 2016), and (3) pre-trained contextual word embeddings (Peters et al., 2018).

Baselines. We also evaluate setups that involve only previous word embeddings. These are effectively reimplementations of state-of-the-art approaches within our sequence labeling architecture. We conduct these experiments to isolate the impact of our proposed embeddings vis-a-vis earlier approaches. The setups are as follows:

HUANG : A standard BiLSTM-CRF setup with pre-trained word embeddings, and thus a reimplementation of Huang et al. (2015).

LAMPLE : A hierarchical BiLSTM-CRF setup with pre-trained word embeddings, in which task-trained character features are additionally computed by a character-level BiLSTM for each word. It is thus a reimplementation of Lample et al. (2016).

PETERS : A BiLSTM-CRF setup that utilizes the release of Peters et al. (2018) in the ALLENNLP library to produce contextualized word embeddings, and thus is effectively a reimplementation of Peters et al. (2018) in our framework. Note that only English embeddings are provided, so we use this baseline only on the English tasks.

Proposed approach. We evaluate our proposed contextual string embeddings in the following configurations:

PROPOSED : The simplest setup of our proposed approach that relies solely on our proposed contextual string embeddings, passed to a standard BiLSTM-CRF architecture. This configuration thus matches the illustration in Figure 1.

PROPOSED$_{+\text{WORD}}$: An extension of PROPOSED in which we concatenate pre-trained static word embeddings with our contextual string embeddings as per Equation (14), to determine whether they complement or subsume our proposed embeddings.

PROPOSED$_{+\text{CHAR}}$:A similar extension in which we concatenate task-trained character features in a hierarchical BiLSTM-CRF architecture to our contextual string embeddings, to determine whether they complement or subsume our proposed embeddings.

PROPOSED$_{+\text{WORD+CHAR}}$: We also evaluate a setting in which we add both pre-trained word and task-trained character embeddings.

PROPOSED$_{+\text{ALL}}$: Finally, we evaluate a setup that uses all four embeddings. Since Peters et al. (2018) embeddings are only distributed for English, we use this setup only on the English tasks.

Approach	NER-English F1-score	NER-German F1-score	Chunking F1-score	POS Accuracy
proposed				
PROPOSED	91.97±0.04	85.78 ± 0.18	96.68±0.03	97.73±0.02
PROPOSED$_{+\text{WORD}}$	93.07±0.10	88.20 ± 0.21	96.70±0.04	97.82±0.02
PROPOSED$_{+\text{CHAR}}$	91.92±0.03	85.88 ± 0.20	**96.72**±0.05	97.8±0.01
PROPOSED$_{+\text{WORD+CHAR}}$	**93.09**±0.12	**88.32** ± 0.20	96.71±0.07	97.76±0.01
PROPOSED$_{+\text{ALL}}$	92.72±0.09	n/a	96.65±0.05	**97.85**±0.01
baselines				
HUANG	88.54±0.08	82.32 ± 0.35	95.4±0.08	96.94±0.02
LAMPLE	89.3±0.23	83.78 ± 0.39	95.34±0.06	97.02±0.03
PETERS	92.34±0.09	n/a	96.69±0.05	97.81± 0.02
best published				
	92.22±0.10	78.76	96.37±0.05	97.64
	(Peters et al., 2018)	(Lample et al., 2016)	(Peters et al., 2017)	(Choi, 2016)
	91.93±0.19	77.20	95.96±0.08	97.55
	(Peters et al., 2017)	(Seyler et al., 2017)	(Liu et al., 2017)	(Ma and Hovy, 2016)
	91.71±0.10	76.22	95.77	97.53±0.03
	(Liu et al., 2017)	(Gillick et al., 2015)	(Hashimoto et al., 2016)	(Liu et al., 2017)
	91.21	75.72	95.56	97.30
	(Ma and Hovy, 2016)	(Qi et al., 2009)	Søgaard et al. (2016)	(Lample et al., 2016)

Table 2: Summary of evaluation results or all proposed setups and baselines. We also list the best published scores for each task for reference. We significantly outperform all previous works on NER, and slightly outperform the previous state-of-the-art in PoS tagging and chunking.

3.2 Model Training and Parameters

Character-level language models. We train our LMs using SGD to perform truncated backpropagation through time (BPTT) with a window length of 250, a non-annealed learning rate of 20.0, a batch size of 100, clipping gradients at 0.25 and dropout probabilities of 0.25. We prepare the data by unking-the rarest 0.0001 percent of characters. We set the number of hidden states of the (one-layered) LSTM to 2048. We halt training by tracking the performance on the validation set, stopping when negligible gains were observed, or after 1 week, whichever came first.

We train our English models on the 1-billion word corpus (Chelba et al., 2013) and our German models on a corpus of half a billion words aggregated from various open source corpora in the OPUS project[2]. After training, we find the models trained for the full week achieve character level perplexity on the supplied test-set of 2.42 for English and 2.38 for German. Resources did not allow us to train for longer or on larger language model variants. We hypothesize that more resources and time would permit learning an even better model.

Sequence tagging model. We train the sequence tagging model using vanilla SGD with no momentum, clipping gradients at 5, for 150 epochs. We employ a simple learning rate annealing method in which we halve the learning rate if training loss does not fall for 5 consecutive epochs. Following recommendations of Reimers et al. (2017)'s in-depth analysis of hyperparameters in sequence labeling, we utilize variational dropout, set the number of hidden states per-layer of the LSTM to 256, set the number of LSTM layers to 1, and perform model selection over the learning rate $\in \{0.01, 0.05, 0.1\}$ and mini-batch size $\in \{8, 16, 32\}$, choosing the model with the best F-measure (for NER and chunking) or accuracy (for PoS) in the best epoch as judged by performance on the validation set. Following Peters et al. (2017), we then repeat the experiment for the model chosen 5 times with different random seeds, and train using both train and development set, reporting both average performance and standard deviation over these runs on the test set as final performance.

Classic word embeddings. Following Reimers et al. (2017), we use GLOVE embeddings for English NER and KOMNIOS embeddings (Komninos and Manandhar, 2016) for PoS tagging and chunking. For German, we use the German FASTTEXT embeddings (Grave et al., 2018). In configurations that train character features, we apply a BiLSTM with 25 hidden states to each word separately and extract the final hidden output states (see Lample et al. (2016)).

[2]We aggregate over the PARACRAWL, EUROPARL, OPENSUBTITLES2018, and WIKIPEDIA releases in OPUS for versatile text and to allow reproduction of our results.

Embedding + Architecture	NER-English F1-score	NER-German F1-score	Chunking F1-score	POS Accuracy
PROPOSED$_{+\text{WORD}}$				
+BiLSTM-CRF	93.07 ± 0.10	88.20 ± 0.21	96.70 ± 0.04	97.82 ± 0.02
+Map-CRF	90.17 ± 0.06	85.17 ± 0.04	96.05 ± 0.04	97.62 ± 0.01
+Map	79.86 ± 0.12	76.97 ± 0.16	90.55 ± 0.05	97.35 ± 0.01
PROPOSED				
+BiLSTM-CRF	91.97 ± 0.04	85.78 ± 0.18	96.68 ± 0.03	97.73 ± 0.02
+Map-CRF	88.62 ± 0.15	82.27 ± 0.22	95.96 ± 0.05	97.53 ± 0.02
+Map	81.42 ± 0.16	73.90 ± 0.09	90.50 ± 0.06	97.26 ± 0.01
CLASSIC WORD EMBEDDINGS				
+BiLSTM-CRF	88.54 ± 0.08	82.32 ± 0.35	95.40 ± 0.08	96.94 ± 0.02
+Map-CRF	66.53 ± 0.03	72.69 ± 0.12	91.26 ± 0.04	94.06 ± 0.02
+Map	48.79 ± 0.27	57.43 ± 0.12	65.01 ± 0.50	89.58 ± 0.02

Table 3: Additional ablation experiment in which we evaluate our approach without BiLSTM and CRF. Here, we instead use a simple linear map over the embeddings to determine their direct information content.

3.3 Results

Our experimental results are summarized in Table 2 for each of the four tasks. We find that our approach either slightly or strongly surpasses all previously published results. This shows that our proposed contextual string embeddings are indeed highly useful for sequence labeling. In more detail, we make the following observations:

New state-of-the-art for NER. We find that our approach performs particularly well on the task of named entity recognition. For English, we surpass the previous state-of-the-art approach by a significant margin (**93.09** F1, ↑0.87 pp vs. Peters et al. (2018)). For German in particular, we significantly raise the state-of-the-art (**88.33** F1, ↑9.56 pp vs. Lample et al. (2016)). We hypothesize that pre-trained contextual character-level features are particularly helpful for this task, since entities are an open vocabulary of names that are often indicated by character features (such as capitalization or endings), as well as their contextual use. In addition, the CONLL03 shared task data set contains many sentences (such as article titles) that are in all-caps; since most traditional embeddings perform lower casing to manage vocabulary size, we assume that our character model can better capture the features of such sentences.

Good performance on syntactic tasks. We also find that our approach slightly outperforms the latest state-of-the-art approaches for chunking and PoS tagging. We do not however see the same improvements as for NER, which we mainly attribute to the fact that syntactic tasks are already solved by current approaches to very high accuracy, making it difficult to record further strong improvements (see Manning et al. (2014) for a discussion of this issue).

Traditional word embeddings helpful. Additionally, we observe that the added use of classic word embeddings in setup PROPOSED$_{+\text{WORD}}$ often produces the best results. Especially for NER, the use of classic word embeddings increases average F1 score by 1.1 pp to 93.07, compared with 91.97 for PROPOSED. We hypothesize that classic word embeddings capture word-level semantics that complement the strong character-level features of our proposed embeddings. We therefore recommend the setup PROPOSED$_{+\text{WORD}}$ as default setup for sequence labeling tasks.

Task-specific character features unnecessary. Another observation is that setups with task-specific character features (PROPOSED$_{+\text{CHAR}}$ and PROPOSED$_{+\text{WORD+CHAR}}$) do not perform better than those without. This indicates that they are largely subsumed by our proposed embeddings, and are thus no longer required. A positive side-effect of this is that this simplifies training of the sequence labeler (see Section 3.5).

3.4 Inherent Semantics

Quantiative investigation (Table 3). To gain additional insight into the content of the contextual string embeddings relative to the task, we replace the standard BiLSTM-CRF in the sequence labeling architecture with a direct feedforward map as per Equation (12), with and without a CRF decoder. This simplified approach makes predictions directly on the basis of the proposed embeddings without additional learning of the BiLSTM recurrence. The results for this simplified architecture are displayed in Table 3 for the

word	context	selected nearest neighbors
Washington	(a) ***Washington*** *to curb support for [..]*	(1) ***Washington*** *would also take [..] action [..]* (2) ***Russia*** *to clamp down on barter deals [..]* (3) ***Brazil*** *to use hovercrafts for [..]*
Washington	(b) *[..] Anthony* ***Washington*** *(U.S.) [..]*	(1) *[..] Carla* ***Sacramento*** *(Portugal) [..]* (2) *[..] Charles* ***Austin*** *(U.S.) [..]* (3) *[..] Steve* ***Backley*** *(Britain) [..]*
Washington	(c) *[..] flown to* ***Washington*** *for [..]*	(1) *[..] while visiting* ***Washington*** *to [..]* (2) *[..] journey to New York City and* ***Washington*** *[..]* (14) *[..] lives in* ***Chicago*** *[..]*
Washington	(d) *[..] when* ***Washington*** *came charging back [..]*	(1) *[..] point for victory when* ***Washington*** *found [..]* (4) *[..] before* ***England*** *struck back with [..]* (6) *[..] before* ***Ethiopia*** *won the spot kick decider [..]*
Washington	(e) *[..] said* ***Washington*** *[..]*	(1) *[..] subdue the never-say-die* ***Washington*** *[..]* (4) *[..] a private school in* ***Washington*** *[..]* (9) *[..] said* ***Florida*** *manager John Boles [..]*

Table 4: Examples of the word "Washington" in different contexts in the CoNLL03 data set, and nearest neighbors using cosine distance over our proposed embeddings. Since our approach produces different embeddings based on context, we retrieve different nearest neighbors for each mention of the same word.

setups PROPOSED and PROPOSED$_{+\text{WORD}}$, as well as for a setup that involves only traditional word embeddings (GLOVE for English NER, KOMNIOS for English PoS and chunking, FASTTEXT for German NER).

We find that the effect of removing the BiLSTM layer on downstream task accuracy is far lower for the proposed embeddings than for classic embeddings. For the setups PROPOSED and PROPOSED$_{+\text{WORD}}$, we record only an average drop of 3% in F-score/accuracy between the BiLSTM-CRF and Map-CRF architectures. This stands in contrast to classic embeddings in which we find an average drop of 20% from BiLSTM-CRF to Map-CRF. This indicates that the inherent semantics of the proposed embeddings are meaningful enough as to require much less powerful learning architectures on top to perform downstream sequence labeling tasks. In particular, for PoS tagging, the simple feedforward map is competitive to BiLSTM and much more effective to train.

Qualitative inspection (Table 4). To illustrate the contextualized nature of our proposed embeddings, we present example embeddings of the polysemous word "Washington" in different contexts. We compute contextual string embeddings for all words in the English CoNLL03 corpus and compute nearest neighbors in the embedding space using the cosine distance. We then look up nearest neighbors for different mentions of the word "Washington".

As Table 4 shows, the embeddings successfully pry apart person, place, legislative entity and team (a-d). For instance, "Washington" used as last name in context (b) is closest to other last names, many of which are also place names ("Carla Sacramento"); "Washington" used as a sport team name in context (d) is closest to other place names used in sports team contexts. We include a negative example (e) in Table 4 in which the context is not sufficient to determine the type of mention. We hypothesize that modeling semantics in context is a key feature that allows our proposed embeddings to better address downstream sequence labeling task.

3.5 Discussion

Our proposed approach is one of the first to leverage hidden states from a language model to improve sequence labeling performance. Two prior works have suggested related approaches: The first is Liu et al. (2017) that jointly train a character-level language model together with the sequence labeling BiLSTM. In effect, this means that the language model is trained only on labeled task data and therefore has orders of magnitude fewer data available than our proposed approach (which we can pre-train on basically unlimited amounts of unlabled data). We hypothesize that this is the main reason for why our approach outperforms Liu et al. (2017) across all tasks.

A second approach is the method by Peters et. al (2017) which proposed to extract hidden states from pre-trained word-level language models as features for downstream NLP tasks. They report new state-

of-the-art results for NER (Peters et al., 2018) and chunking (Peters et al., 2017), but require the training of massive word-level language models: their best configuration uses a language model that was trained for 5 weeks on 32 GPUs (Jozefowicz et al., 2016), and still requires lower casing of all words to deal with the large vocabulary size. Their approach is the strongest baseline against which we compared.

We similarly propose to utilize trained LMs to generate embeddings, but consider LM at the character level. This has a number of advantages: First, character-level LM is independent of tokenization and a fixed vocabulary. Second, they produce stronger character-level features, which is particularly useful for downstream tasks such as NER. Finally, such models have a much smaller vocabulary size (distinct characters vs. distinct words) and are thus significantly easier to train and deploy in applications: the LM parameter-count scales according to $n_{layers}\, n^2_{hidden}$ (not true for word-level LM since these contain many inputs) and may be trained in 1 week on 1 GPU (e.g. significantly less than the 5 weeks on 32 GPUs used by Peters et al. (2017)). This allowed us to train models for other languages such as German with moderate resources, and thus allows our method to more effectively scale to new languages or domains.

4 Conclusion

We have proposed novel *contextual string embeddings* and a method for producing them using neural character LM. We find that they capture syntactic-semantic word features and disambiguate words in context, resulting in state-of-the-art performance in classic NLP sequence labeling tasks. To facilitate reproduction of our experiments, and enable application of our proposed embeddings to other tasks, we release all code and pre-trained language models in a simple-to-use framework to the research community[3]. Future work will focus on applying these embeddings to additional sentence level tasks such as image-retrieval and neural translation.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732328 ("FashionBrain").

References

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Jinho D Choi. 2016. Dynamic feature induction: The last gist to the state-of-the-art. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 271–281.

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

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2015. Multilingual language processing from bytes. *arXiv preprint arXiv:1512.00103*.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

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

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*.

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

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

[3] https://github.com/zalandoresearch/flair

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

Alexandros Komninos and Suresh Manandhar. 2016. Dependency based embeddings for sentence classification tasks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1490–1500.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2017. Empower sequence labeling with task-aware neural language model. *arXiv preprint arXiv:1709.04109*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTMs-CNNs-CRF. *arXiv preprint arXiv:1603.01354*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Scott Menard. 2018. *Applied logistic regression analysis*, volume 106. SAGE publications.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1765, Vancouver, Canada, July. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, and Christopher Clark Kenton Lee Luke Zettlemoyer Mohit Iyyer, Matt Gardner. 2018. Deep contextualized word representations. *6th International Conference on Learning Representations*.

Yanjun Qi, Ronan Collobert, Pavel Kuksa, Koray Kavukcuoglu, and Jason Weston. 2009. Combining labeled and unlabeled data with word-class distribution learning. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1737–1740. ACM.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark, 09.

Ronald Rosenfeld. 2000. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278.

Dominic Seyler, Tatiana Dembelova, Luciano Del Corro, Johannes Hoffart, and Gerhard Weikum. 2017. Knowner: Incremental multilingual knowledge in named entity recognition. *arXiv preprint arXiv:1709.03544*.

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

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

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

Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 127–132. Association for Computational Linguistics.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Learning Word Meta-Embeddings by Autoencoding

Cong Bao
Department of Computer Science
University of Liverpool
sgcbao2@liverpool.ac.uk

Danushka Bollegala
Department of Computer Science
University of Liverpool
danushka@liverpool.ac.uk

Abstract

Distributed word embeddings have shown superior performances in numerous Natural Language Processing (NLP) tasks. However, their performances vary significantly across different tasks, implying that the word embeddings learnt by those methods capture complementary aspects of lexical semantics. Therefore, we believe that it is important to combine the existing word embeddings to produce more accurate and complete *meta-embeddings* of words. We model the meta-embedding learning problem as an autoencoding problem, where we would like to learn a meta-embedding space that can accurately reconstruct *all* source embeddings simultaneously. Thereby, the meta-embedding space is enforced to capture complementary information in different source embeddings via a coherent common embedding space. We propose three flavours of autoencoded meta-embeddings motivated by different requirements that must be satisfied by a meta-embedding. Our experimental results on a series of benchmark evaluations show that the proposed autoencoded meta-embeddings outperform the existing state-of-the-art meta-embeddings in multiple tasks.

1 Introduction

Representing the meanings of words is a fundamental task in Natural Language Processing (NLP). A popular approach to represent the meaning of a word is to *embed* it in some fixed-dimensional vector space (Turney and Pantel, 2010). In contrast to sparse and high-dimensional counting-based distributional word representation methods that use co-occurring contexts of a word as its representation, dense and low-dimensional prediction-based distributed word representations (Pennington et al., 2014; Mikolov et al., 2013a; Huang et al., 2012; Collobert and Weston, 2008; Mnih and Hinton, 2009) have obtained impressive performances in numerous NLP tasks such as sentiment classification (Socher et al., 2013), and machine translation (Zou et al., 2013).

Previous works studying the differences in word embedding learning methods (Chen et al., 2013; Yin and Schütze, 2016) have shown that word embeddings learnt using different methods and from different resources have significant variation in quality and characteristics of the semantics captured. For example, Hill et al. (2014; 2015) showed that the word embeddings trained from monolingual vs. bilingual corpora capture different local neighbourhoods. Bansal et al. (2014) showed that an ensemble of different word representations improves the accuracy of dependency parsing, implying the complementarity of the different word embeddings. This suggests the importance of *meta-embedding* – creating a new embedding by combining different existing embeddings. We refer to the input word embeddings to the meta-embedding process as the *source embeddings*. Yin and Schütze (2016) showed that by meta-embedding five different pre-trained word embeddings, we can overcome the out-of-vocabulary problem, and improve the accuracy of cross-domain part-of-speech (POS) tagging. Encouraged by the above-mentioned prior results, we expect an ensemble containing multiple word embeddings to produce better performances than the constituent individual embeddings in NLP tasks.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1650–1661
Santa Fe, New Mexico, USA, August 20-26, 2018.

Despite the above-mentioned benefits, learning a single meta-embedding from multiple source embeddings remains a challenging task due to several reasons. First, the algorithms used for learning the source word embeddings often differ significantly and it is non-obvious how to reconcile their differences. Second, the resources used for training source embeddings such as text corpora or knowledge bases are different, resulting in source embeddings that cover different types of information about the words they embed. Third, the resources used to train a particular source embedding might not be publicly available for us to train all source embeddings in a consistent manner. For example, the Google news corpus containing 100 billion tokens on which the skip-gram embeddings are trained and released originally by Mikolov et al. (2013b) is not publicly available for us to train other source embedding learning methods on the same resource. Therefore, a meta-embedding learning method must be able to learn meta-embeddings from the given set of source embeddings without assuming the availability of the training resources.

Autoencoders (Kingma and Welling, 2014; Vincent et al., 2008) have gained popularity as a method for learning feature representations from unlabelled data that can then be used for supervised learning tasks. Autoencoders have been successfully applied in various NLP tasks such as domain adaptation (Chen et al., 2012; Ziser and Reichart, 2016), similarity measurement (Li et al., 2015; Amiri et al., 2016), machine translation (P et al., 2014; Li et al., 2013) and sentiment analysis (Socher et al., 2011). Autoencoder attempts to reconstruct an input from a possibly noisy version of the input via a non-linear transformation. The intermediate representation used by the autoencoder captures the essential information about the input such that it can be accurately reconstructed. This setting is closely related to the meta-embedding learning where we must reconstruct the information contained in individual source embeddings using a single meta-embedding. However, unlike typical autoencoder learning where we have a single input, in meta-embedding learning we must reconstruct multiple source embeddings.

We propose three types of autoencoders for the purpose of learning meta-embeddings. The three types consider different levels of integrations among the source embeddings. To the best of our knowledge, autoencoders have not been used for meta-embedding learning in prior work. We compare the proposed autoencoded meta-embeddings (AEME) against previously proposed meta-embedding learning methods and competitive baselines using five benchmark tasks. Our experimental results show that AEME outperforms the current state-of-the-art meta-embeddings in multiple tasks.

2 Related Work

Yin and Schütze (2016) proposed a meta-embedding learning method (1TON) that projects a meta-embedding of a word into the source embeddings using separate projection matrices. The projection matrices are learnt by minimising the sum of squared Euclidean distance between the projected source embeddings and the corresponding original source embeddings for all the words in the vocabulary. They propose an extension (1TON+) to their meta-embedding learning method that first predicts the source word embeddings for out-of-vocabulary words in a particular source embedding, using the known word embeddings. Next, 1TON method is applied to learn the meta-embeddings for the union of the vocabularies covered by all of the source embeddings. Experimental results in semantic similarity prediction, word analogy detection, and cross-domain POS tagging tasks show the effectiveness of both 1TON and 1TON+.

Although not learning any meta-embedding, several prior works have shown that incorporating multiple word embeddings learnt using different methods improve performance in various NLP tasks. For example, Tsuboi (2014) showed that by using both word2vec and GloVe embeddings together in a POS tagging task, it is possible to improve the tagging accuracy, if we had used only one of those embeddings. Similarly, Turian et al. (2010) collectively used Brown clusters, CW and HLBL embeddings, to improve the performance of named entity recognition and chucking tasks.

Luo et al. (2014) proposed a multi-view word embedding learning method that uses a two-sided neural network. They adapt pre-trained CBOW (Mikolov et al., 2013b) embeddings from Wikipedia and click-through data from a search engine. Their problem setting is different from ours because their source embeddings are trained using the same word embedding learning method but on different resources

whereas, we consider source embeddings trained using different word embedding learning methods and resources. Although their method could be potentially extended to meta-embed different source embeddings, the unavailability of their implementation prevented us from exploring this possibility.

Goikoetxea et al. (2016) showed that concatenation of word embeddings learnt separately from a corpus and the WordNet to produce superior word embeddings. Moreover, performing Principal Component Analysis (PCA) on the concatenated embeddings slightly improved the performance on word similarity tasks.

Chandar et al. (2015) proposed a correlation neural network that reconstruct an input using two views such that their correlation in a hidden layer os maximised. The setting of this work, using multiple views for reconstructing, is similar to our multiple source embeddings. However, we do not optimise simply for correlation, but require reconstructing the sources from meta-embeddings. Moreover, Chandar et al. (2015) did not consider meta-embedding learning, which is the focus of this paper.

3 Autoencoded Meta-Embeddings

To simplify the disposition of the proposed method, we focus on the problem of learning a single meta-embedding from two given source embeddings. The autoencoding methods we propose in this paper can be easily generalised to more than two source embeddings. Let us denote the two source embeddings by $\mathcal{S}_1$ and $\mathcal{S}_2$. Moreover, the dimensionalities of two source embeddings $\mathcal{S}_1$ and $\mathcal{S}_2$ are given respectively d_1 and d_2. Note that we do not assume d_1 and d_2 to be equal. The two word embeddings of a word $w \in \mathcal{V}$ is given respectively by $\boldsymbol{s}_1(w) \in \mathbb{R}^{d_1}$ and $\boldsymbol{s}_2(w) \in \mathbb{R}^{d_2}$. Here, the vocabulary $\mathcal{V}$ is the intersection set of two vocabularies $\mathcal{V}_1$ and $\mathcal{V}_2$ of source embeddings $\mathcal{S}_1$ and $\mathcal{S}_2$, i.e. $\mathcal{V} = \mathcal{V}_1 \cap \mathcal{V}_2$. For out of vocabulary words, we can first train a regression model using source embeddings for the common vocabulary as done by Yin and Schütze (2016) and then use it to predict the source embeddings for the words that do not occur in the intersection of the vocabularies.

Next, let us consider two encoders E_1 and E_2, which encode the two source embeddings to a common meta-embedding space $\mathcal{M}$ with dimensionality d_m for each word $w \in \mathcal{V}$. Dimensionalities of the encoded source embeddings are denoted respectively by d'_1 and d'_2. We denote the meta-embedding of a word $w \in \mathcal{V}$ as $\boldsymbol{m}(w) \in \mathbb{R}^{d_m}$. Correspondingly, we have two decoders D_1 and D_2, which will decode each word $w \in \mathcal{V}$ in the meta-embedding space $\mathcal{M}$ back to the two original source embeddings $\mathcal{S}_1$ and $\mathcal{S}_2$. Both encoders and decoders can be single or multi-layer neural networks.

We consider the problem of learning E_1, E_2, D_1 and D_2 such that we can learn a meta-embedding $\boldsymbol{m}(w)$ for a word w considering the complementary information from its source embeddings $\boldsymbol{s}_1(w)$ and $\boldsymbol{s}_2(w)$. We propose three different autoencoding methods for this purpose. The architectures of the three autoencodes are visualised in Figure 1.

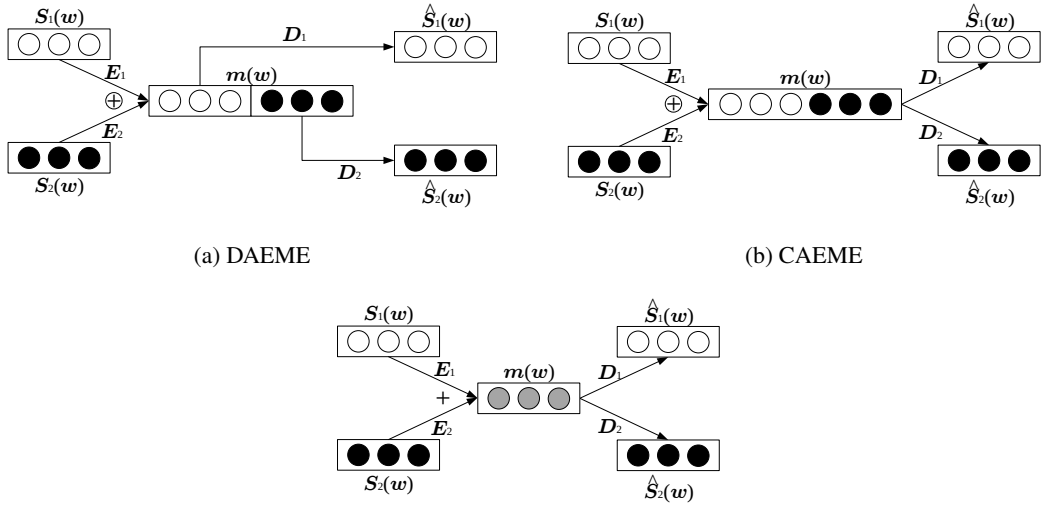
3.1 Decoupled Autoencoded Meta-Embedding (DAEME)

In DAEME, the meta-embedding $\boldsymbol{m}(w)$ is represented as the concatenation of two encoded source embeddings $E_1(\boldsymbol{s}_1(w))$ and $E_2(\boldsymbol{s}_2(w))$ for each word $w \in \mathcal{V}$, as given by (1).

$$\boldsymbol{m}(w) = E_1(\boldsymbol{s}_1(w)) \oplus E_2(\boldsymbol{s}_2(w)) \tag{1}$$

Concatenation has been found to be a simple yet effective baseline for creating meta-embeddings from multiple source embeddings. DAEME can be seen as an extension of concatenation that has non-linear neural networks applied on the raw model. Here, each encoder can be seen as independently performing a transformation to the respective source embedding so that it can learn to retain essential information rather than simply concatenate features.

The dimensionality of meta-embedding space $\mathcal{M}$ is therefore computed as the sum of dimensionalities of source embeddings $\mathcal{S}_1$ and $\mathcal{S}_2$ after encoding, i.e. $d_m = d'_1 + d'_2$. We then decode the meta-embedding to reconstruct the original source embeddings as $\hat{\boldsymbol{s}}_1(w)$ and $\hat{\boldsymbol{s}}_2(w)$ using two decoders. Specifically, we reconstruct the two components of the meta-embedding $\boldsymbol{m}(w)$ in (1) independently to the corresponding source embeddings. Because of this behaviour we call this approach *decoupled* autoencoded meta-embedding.



(a) DAEME

(b) CAEME

(c) AAEME

Figure 1: The architectures of proposed AEMEs. Rectangles represent word vectors, circles represent a single dimension of a word vector, and filled circles represent source of word embeddings. In (c), greyed circles in $\boldsymbol{m}(w)$ indicates mixing of vectors from the two source embeddings.

The reconstructed versions of the two source embeddings are given by (2) and (3).

$$\hat{\boldsymbol{s}}_1(w) = D_1(E_1(\boldsymbol{s}_1(w))) \tag{2}$$

$$\hat{\boldsymbol{s}}_2(w) = D_2(E_2(\boldsymbol{s}_2(w))) \tag{3}$$

To encourage encoded embeddings share common information from the two sources, while retaining complementary information, we propose the loss given by (4).

$$\mathcal{L}(E_1, E_2, D_1, D_2) = \sum_{w \in \mathcal{V}} \Big(\lambda_1 \left|\left| E_1(s_1(w)) - E_2(s_2(w)) \right|\right|^2 + \lambda_2 \left|\left| \hat{\boldsymbol{s}}_1(w) - \boldsymbol{s}_1(w) \right|\right|^2 + \lambda_3 \left|\left| \hat{\boldsymbol{s}}_2(w) - \boldsymbol{s}_2(w) \right|\right|^2 \Big) \tag{4}$$

The first term in the RHS of (4) emphasises the common information to the two source, whereas the second and third terms force the meta-embedding to retain sufficient information to reconstruct the two source embeddings. The coefficients λ_1, λ_2, and λ_3 in (4) can be used to control the effect of the three errors on the overall objective. Later in our experiments, we set these coefficients using validation data. We jointly learn E_1, E_2, D_1 and D_2 such that the total reconstruction error given by (4) is minimised. To prevent the weight matrices of the autoencoders degrading to the identity matrix $\mathbf{I}$ during training, we use a non-linear activation function (ReLU in our experiments) in encoders.

3.2 Concatenated Autoencoded Meta-Embedding (CAEME)

Similar to DAEME, the meta-embedding in CAEME is also constructed as the concatenation of two encoded source embeddings as described in (1). However, instead of treating the meta-embedding as two individual components, CAEME reconstructs the source embeddings from the *same* meta-embedding, thereby implicitly using both common and complementary information in the source embeddings. The reconstructed source embeddings for CAEME are given by (5) and (6).

$$\hat{\boldsymbol{s}}_1(w) = D_1(\boldsymbol{m}(w)) \tag{5}$$

$$\hat{\boldsymbol{s}}_2(w) = D_2(\boldsymbol{m}(w)) \tag{6}$$

In CAEME, the dimensionality of the meta-embedding space $\mathcal{M}$ is identical to that of DAEME, which is $d_m = d'_1 + d'_2$. The overall objective of CAEME is given by (7). Similar to DAEME, the coefficients λ_1 and λ_2 can be used to give different emphasis to the reconstruction of the two sources. For example, if we would like to emphasis reconstructing $\mathcal{S}_2$ more than $\mathcal{S}_1$ in the meta-embedding process, we can set $\lambda_2 > \lambda_1$. The coefficients are also tuned experimentally using validation data.

$$\mathcal{L}(E_1, E_2, D_1, D_2) = \sum_{w \in \mathcal{V}} \left(\lambda_1 \left|\left| \hat{\boldsymbol{s}}_1(w) - \boldsymbol{s}_1(w) \right|\right|^2 + \lambda_2 \left|\left| \hat{\boldsymbol{s}}_2(w) - \boldsymbol{s}_2(w) \right|\right|^2 \right) \tag{7}$$

Compared to DAEME, CAEME imposes a tighter integration between the two sources in their meta-embedding. We jointly learn E_1, E_2, D_1 and D_2 that minimises the total reconstruction error given by (7).

3.3 Averaged Autoencoded Meta-Embedding (AAEME)

AAEME can be seen as a special case of CAEME, where we compute the meta-embedding by averaging the two encoded sources in (1) instead by their concatenation. A recent work (Coates and Bollegala, 2018) shows that for approximately orthogonal source embedding spaces, averaging performs comparably to concatenation, without increasing the dimensionality. However, our AAEME can be seen as a more general version of averaging in the sense that we first transform each source embedding independently using two encoders before we compute their average. This operation has the benefit that we can transform the sources such that they could be averaged in the same vector space, and also guarantees orthogonality between the encoded vectors. AAEME computes the meta-embedding of a word w from its two source embeddings $\boldsymbol{s}_1(w)$ and $\boldsymbol{s}_2(w)$ as the ℓ_2-normalised [1] sum of two encoded versions of the source embeddings $E_1(\boldsymbol{s}_1(w))$ and $E_2(\boldsymbol{s}_2(w))$, given by (8).

$$\boldsymbol{m}(w) = \frac{E_1(\boldsymbol{s}_1(w)) + E_2(\boldsymbol{s}_2(w))}{||E_1(\boldsymbol{s}_1(w)) + E_2(\boldsymbol{s}_2(w))||_2} \tag{8}$$

Note that unlike CAEME, where the outputs of the two encoders might be of different dimensionalities, in AAEME they should be equal in order to facilitate summation. One could set the output dimensionalities of the two encoders to be different and pad zeros to the shorter output vector. However, we did not find this trick to perform well empirically in our preliminary experiments and do not consider output encodings of different dimensionalities in AAEME. Similar to CAEME, the two source embeddings are reconstructed from the same meta-embedding using two separate decoders D_1 and D_2 as given by (5) and (6). The overall reconstruction error for AAEME is the same as in (7).

4 Experiments

4.1 Source Word Embeddings

We use the following two source embeddings in our experiments:

CBOW: The Continuous Bag-Of-Words (CBOW) embeddings proposed by Mikolov et al. (2013a). The released version we used contains 929,019 word embeddings (phrase embeddings are discarded) with dimensionality of 300, and is trained on Google News corpus (about 100 billion words).

GloVe: The Global Vectors for word representation proposed by Pennington et al. (2014). The released version we used contains 1,917,494 word embeddings with dimensionality of 300 that is trained on Common Crawl dataset.

The intersection of two vocabularies contains 154,076 words for which we create meta-embeddings. We choose CBOW and GloVe as the source embeddings because according to Yin and Schütze (2016), CBOW and GloVe outperform other previously proposed word embeddings such as HLBL (Mnih and Hinton, 2009) and Huang (Huang et al., 2012) in several benchmark tasks such as semantic similarity measurement and word analogy prediction. However, we emphasise that all three proposed methods can be trained with more than two source embeddings.

[1] In our preliminary experiments we found ℓ_2-normalisation to improve performance.

4.2 Training Details

In our experiments, each autoencoder is implemented as a neural network with a single hidden layer. The weights of the encoders and decoders are randomly initialised by sampling from a Gaussian with zero mean and 0.01 standard deviation. We use Adam (Kingma and Ba, 2014) with mini-batches of size 128 for minimising the reconstruction error in each variant of AEME. Masking noises (MN) (Vincent et al., 2010) is applied during training, which will randomly set a fraction of the elements in an input vector to zero. Table 1 summarises hyperparameters, overall trainable parameters, and the time consumed by each method. The optimal hyperparameters were found using the Miller-Charles dataset as a validation dataset. Code implementing AEMEs is publicly available [2].

Method	bs	lr	epoch	activation	noise	λ_1	λ_2	λ_3	hidden neurons	parameters	time used (on CPU)
DAEME	128	0.001	500	ReLU	5% MN	1.0	1.0	5.0	600	361,200	3h 57min
CAEME	128	0.001	500	Sigmoid / ReLU	5% MN	1.0	1.0	/	600	541,200	6h 53min
AAEME	128	0.001	500	Sigmoid / ReLU	5% MN	1.0	1.0	/	600	361,200	2h 10min

Table 1: Training details. bs: batch size. lr: learning rate. Both Sigmoid and ReLU activations performed comparably for CAEME and AAEME in our experiments.

4.3 Evaluation Tasks

Following the common practice for evaluating word embeddings, we use the meta-embeddings created by the proposed method in a set of NLP tasks and measure the increase/decrease of the performance of those tasks. If a meta-embedding can improve the performance of an NLP task, then it can be considered as an accurate semantic representation for the words. All meta-embeddings compared in our experiments cover the same subset of words that appear in the benchmark dataset. Therefore, there is no unfair advantage to a particular meta-embedding method due to its coverage of the vocabulary. We use the following five evaluation tasks:

Semantic Similarity: The semantic similarity between two words is measured as the cosine similarity between the corresponding word embeddings. The computed semantic similarity scores are compared against human-rated similarity scores using the Spearman correlation coefficient. A high degree of correlation with the human ratings is considered as an indication of the accuracy of the word embeddings. We use the following benchmark datasets for this evaluation: Word Similarity 353 dataset (**WS**, 2023 word pairs) (Finkelstein et al., 2002), Rubenstein-Goodenough dataset (**RG**, 65 word pairs) (Rubenstein and Goodenough, 1965), Miller-Charles dataset (**MC**, 30 word pairs) (Miller and Charles, 1998), and the **MEN** dataset (3000 word pairs) (Bruni et al., 2012).

Word Analogy: Word analogy task consists of questions like "a to b is c to what?". Different methods have been proposed in the literature for finding fourth word d that completes an analogy. In our experiments we use the CosAdd method, which finds d such that the cosine similarity between the vector $(\boldsymbol{b} - \boldsymbol{a} + \boldsymbol{c})$ and $\boldsymbol{d}$ is maximised. We use the following benchmark datasets for this task: Google dataset (**GL**) (Mikolov et al., 2013b), **MSR** dataset (Levy and Goldberg, 2014), SemEval 2012 Task 2 dataset (**SE**) (Jurgens et al., 2012), and the **SAT** (Slack, 1980) dataset. The evaluation measure is the ratio of the questions that were answered correctly in each dataset using a particular word embedding.

Relation Classification: The DiffVec dataset (**DV**) (Vylomova et al., 2016) contains 12,458 triples of the form (r, w_1, w_2), where the relation r exists between the two words w_1 and w_2. DiffVec dataset contains tuples covering 15 different relation types. The task is to predict the relation that exists between two words w_1 and w_2 from the 15 relation types in the dataset. The relation classification accuracy is computed as the ratio of the correctly predicted instances to the total instances in the

[2] https://github.com/CongBao/AutoencodedMetaEmbedding

DiffVec dataset. We represent the relation between two words as the vector offset, $\boldsymbol{w}_1 - \boldsymbol{w}_2$, between the corresponding word embeddings. Next, we measure cosine similarity between the target test word-pair and word-pairs in the training dataset and use a 1-nearest neighbour classifier to predict the relation for the test word-pair.

Short-text Classification: In the short-text classification task, a text is represented by the centroid of the embeddings of the words contained in that text. All datasets used in the short-text classification tasks contain binary target labels. We train a binary logistic regression classifier using the train part of each dataset and the classification accuracy is measured on the test part of the corresponding dataset. We use the following short-text classification datasets in our experiments: Stanford Sentiment Treebank (**TR**) (Socher et al., 2013), Movie Review dataset (**MR**) (Pang and Lee, 2005), Customer Review dataset (**CR**) (Hu and Liu, 2004), and Subjectivity dataset (**SUBJ**) (Pang and Lee, 2004).

Psycholinguistic Score Prediction: Word embeddings can be used as features for predicting psycholinguistic ratings of a word (Paetzold and Specia, 2016). We use the learnt meta-embeddings in a neural network (containing a single hidden layer of 100 neurones and ReLU as the activation function) to learn a regression model for predicting different psycholinguistic ratings. We use a randomly selected 80% of words from the MRC database [3] and the ANEW dataset (Beth Warriner et al., 2013) to train regression models for arousal (**AS**), valence (**VAL**), and dominance (**DOM**). Pearson correlation between the predicted ratings and human ratings is used as the evaluation measure.

4.4 Baselines

We compare the meta-embeddings produced by the proposed methods against the following baselines:

Concatenation (CONC): Concatenation of the source embeddings for a particular word has been found to be an effective method for creating meta-embeddings citeYin:ACL:2016. Despite being simple, meta-embeddings created via concatenation have shown good performance in benchmark tasks such as semantic similarity measurement and word analogy detection. We create meta-embeddings by first ℓ_2 normalising the CBOW and GloVe embeddings for a word, and then concatenating the normalised embeddings. The normalisation operation gives an equal importance to the source embeddings when we measure cosine similarity using the concatenated meta-embeddings.

Singular Value Decomposition (SVD): A disadvantage of concatenation as a method for creating meta-embeddings is that it increases the dimensionality of the meta-embedding space. For example, if we concatenated two source embeddings of dimensionalities d_1 and d_2, the resultant meta-embedding will have a dimensionality of $(d_1 + d_2)$. SVD has been used in various tasks in NLP such as latent semantic analysis (Deerwester et al., 1990) and latent relational analysis (Turney, 2005) as a technique to reduce the dimensionality of a feature space. Yin and Schütze (2016) proposed the use of SVD to reduce the dimensionality of the meta-embeddings created from concatenation. Specifically, let N represents the number of words common to the vocabularies of CBOW and GloVe embeddings. We first create an $N \times (d_1 + d_2)$ matrix $\mathbf{C}$ representing the concatenation meta-embedding of all words in $\mathcal{V}$. Next, we perform SVD decomposition on $\mathbf{C} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices and $\mathbf{\Sigma}$ is a diagonal matrix containing the singular values of $\mathbf{C}$. We then select largest k singular values from $\mathbf{\Sigma}$ to construct a diagonal matrix $\mathbf{\Sigma}_k$, and the corresponding left singular vectors from $\mathbf{U}$ to construct a matrix $\mathbf{U}_k$. Finally, we obtain our k-dimensional meta-embeddings given by the rows of $\mathbf{C}_k = \mathbf{U}_k\mathbf{\Sigma}_k$. SVD is guaranteed to produce the best (in the sense of least square error) rank k approximation of a matrix. In our experiments, we use CBOW and GloVe source embeddings of 300 dimensions ($d_1 = d_2 = 300$) and created $k = 300$ dimensional SVD meta-embeddings.

[3] http://websites.psychology.uwa.edu.au/school/MRCDatabase/uwa_mrc.htm

Averaging (AVG): Coates and Bollegala (2018) proposed averaging the source embeddings for a word as a method for creating meta-embeddings. Source embeddings are often trained independently and their vector spaces as well as dimensionalities are different, which is problematic when computing averages. Although it is possible to pad a source embeddings that have fewer dimensions with zero such that all source embeddings have the equal dimensionality, it is still not mathematically valid to add vectors in different spaces. Surprisingly however, Coates and Bollegala (2018) show that averaging performs well in a series of benchmark tasks. Moreover, they prove that if word embeddings can be shown to be approximately orthogonal, then averaging will approximate the same information as concatenation, without increasing the dimensionality. Averaging can be seen as a special case of AAEME. Similar to concatenation, we first perform ℓ_2-normalisation on both CBOW and GloVe embeddings prior to averaging them to create meta-embeddings. Since averaging of embedding does not increase the dimensionality, the final dimension of averaged meta-embedding in our experiments is $d_m = 300$, the same as the dimensionalities of the CBOW and GloVe source embeddings.

4.5 Evaluation Results

The evaluation results of different embeddings on different tasks or datasets are summarised in Table 2. In Table 2, rows 1 and 2 show the performance of the source embeddings. Performance of baseline methods are shown in rows 3-5, and rows 8-10 are the results for the proposed method. We also include 1TON and 1TON+ proposed by Yin and Schütze (2016) in rows 6 and 7 as a comparison, which is the current state-of-the-art. To make a fair comparison, we use the publicly available meta-embeddings released by Yin and Schütze (2016) in our evaluation and do not retrain their method.

		Model	WS	RG	MC	MEN	GL	MSR	SE	SAT	DV	TR	MR	CR	SUBJ	AS	VAL	DOM
sources	1	CBOW	69.1	76.0	82.2	78.2	68.2	76.3	43.4	30.2	88.5	80.3	76.2	79.2	90.6	1.97	0.26	2.70
	2	GloVe	75.4	82.9	87.0	81.7	70.7	72.3	41.6	27.0	87.8	80.0	75.9	81.5	90.0	3.66	1.19	0.85
ensemble	3	CONC	76.4	83.0	88.8	82.5	75.5	79.4	43.5	29.7	88.5	80.6	76.5	80.9	90.7	2.29	1.98	0.00
	4	SVD	76.8	83.3	86.7	82.6	**76.9**	79.8	43.2	30.7	**89.5**	80.3	**77.0**	80.9	90.1	1.29	2.07	0.64
	5	AVG	76.4	83.0	88.8	82.5	75.5	79.4	43.5	29.7	88.5	81.1	76.8	81.2	90.7	3.18	0.11	0.00
	6	1TON	76.5	83.2	88.1	82.8	74.6	78.0	42.3	22.5	87.6	80.7	75.6	81.5	88.7	1.24	5.73	2.74
	7	1TON+	76.8	79.4	85.7	81.8	68.1	72.2	40.1	21.7	83.9	79.4	74.2	68.5	87.1	2.28	11.7	**8.27**
	8	DAEME	**77.3**	83.0	89.1	83.0	75.9*	79.4*	43.7	**39.3***	88.6*	**82.3**	76.6	**84.2**	**90.8***	6.97*	13.3*	6.76
	9	CAEME	76.4	**84.0**	**89.3**	82.3	74.8	78.6	**43.9**	38.0*	88.2*	80.5	76.7	78.9	90.3*	5.27*	12.1	6.41
	10	AAEME	76.1	82.7	88.8	**83.2**	**76.9***	**80.0***	43.8	38.0*	**89.5***	81.4	76.8	82.6	90.3*	**7.03***	**13.6***	7.35

Table 2: Results on the benchmark tasks. Best results are in bold, whereas statistical significance over 1TON is indicated by an asterisk.

From Table 2, we see that the ensemble methods (3-10) outperform the individual source embeddings (1-2) in most tasks. Among the proposed methods, DAEME performs best in short-text classification tasks, while CAEME is competitive in semantic similarity measurement tasks. On the other hand, AAEME performs overall well and obtains the best performance in word analogy, relation classification, and psycholinguistic score prediction tasks. We evaluate statistical significance against both 1TON and 1TON+. For the semantic similarity and psycholinguistic score prediction benchmarks we use Fisher transformation to compute $p < 0.05$ confidence intervals for Spearman correlation coefficients. In all other datasets, we use Clopper-Pearson binomial exact confidence intervals at $p < 0.05$.

Comparing DAEME, CAEME, and CONC, we see that using auto-encoder with nonlinear neural network can achieve better performance in most tasks as they combine complementary information from sources and also reduce probability that features from two source embedding counteract with each other. If we compare AAEME with AVG, we see that AAEME improves the performance of most tasks, especially in psycholinguistic score prediction. This result shows that nonlinear transformation helps to

ensure orthogonality between feature vectors, which is the reason to outperform AVG.

Concerning dimensionality, we see that performing SVD on CONC does not reduce the performance of most tasks, and even outperforms other methods in tasks such as GL, DV, and MR. Among the three proposed methods, both DAEME and CAEME have dimensionality of 600, and AAEME has dimensionality of 300. Although AAEME has smaller dimensionality, it performs similarly to DAEME and CAEME and outperforms them in 6 tasks. Considering the smaller dimensionality and the robust performance, among the three variants of AEME proposed in the paper, AAEME is the recommended meta-embedding for practical applications.

Prior work on word embedding learning shows that the dimensionality of the embedding can affect the performance of a downstream NLP application that uses the word embeddings as features (Bollegala et al., 2015; Bollegala et al., 2016). In order to directly compare the proposed meta-embedding learning methods against the current state-of-the-art meta-embedding learning methods under the same dimensionality, we reduce the dimensionality of the meta-embeddings created by the proposed methods to 200 dimensions using SVD, which is the dimensionality of the publicly released 1TON and 1TON+ meta-embeddings. Table 3 shows the results for this evaluation.

From Table 3, we see that the proposed AEMEs still perform well in tasks of word analogy, relation classification, and short-text classification. Comparing with the results in Table 2, we see that the reduction of dimensionality has not significantly affected the performance on those benchmark datasets. On the other hand, there is a slight drop in performance for the semantic similarity and psycholinguistic score prediction tasks when the dimensionality is reduced. Nevertheless, the proposed methods still outperforms 1TON and 1TON+ in majority of the tasks, which shows that the nonlinear transformations learnt by autoencoders to be superior to the linear transformations learnt by 1TON and 1TON+.

		Model	WS	RG	MC	MEN	GL	MSR	SE	SAT	DV	TR	MR	CR	SUBJ	AS	VAL	DOM
ensemble	1	1TON	76.5	**83.2**	**88.1**	**82.8**	74.6	78.0	42.3	22.5	87.6	80.7	75.6	81.5	88.7	1.24	5.73	2.74
	2	1TON+	**76.8**	79.4	85.7	81.8	68.1	72.2	40.1	21.7	83.9	79.4	74.2	68.5	87.1	2.28	**11.7**	**8.27**
	3	DAEME	74.0	81.7	83.7	82.4	77.5*	79.2*	43.1	**38.0***	**89.7***	**81.5**	76.1	82.9	**90.7***	**5.85***	8.81	7.13
	4	CAEME	73.7	82.9	85.8	82.5	77.5*	80.2*	43.6	36.9*	**89.7***	80.6	75.9	79.5	89.8	3.85*	0.00	3.20
	5	AAEME	74.5	82.0	86.1	82.5	**77.6***	**80.8***	**43.6**	35.0*	**89.7***	81.4	**76.4**	**83.2**	90.5*	4.13	7.63*	6.41

Table 3: Results on the benchmark tasks for 200 dimensional meta-embeddings. Best results are in bold, whereas statistical significance over 1TON is indicated by an asterisk.

5 Conclusion

We proposed three autoencoder-based approaches DAEME, CAEME, and AAEME for learning meta-embeddings from multiple pre-trained source embeddings. The experimental results on a series of benchmark datasets show that AEME outperforms previously proposed meta-embeddings on multiple tasks. We plan to extend the proposed method to create meta-embeddings from multi-lingual word embeddings.

References

Hadi Amiri, Philip Resnik, Jordan Boyd-Graber, and Hal Daumé III. 2016. Learning text pair similarity with context-sensitive autoencoders. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1882–1892, Berlin, Germany, August. Association for Computational Linguistics.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continouos word representations for dependency parsing. In *Proc. of ACL*.

Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 english lemmas. 45, 02.

Danushka Bollegala, Takanori Maehara, and Ken ichi Kawarabayashi. 2015. Embedding semantic relationas into word representations. In *Proc. of IJCAI*, pages 1222 – 1228.

Danushka Bollegala, Alsuhaibani Mohammed, Takanori Maehara, and Ken ichi Kawarabayashi. 2016. Joint word representation learning using a corpus and a semantic lexicon. In *Proc. of AAAI*, pages 2690–2696.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam Khanh Tran. 2012. Distributional semantics in technicolor. In *Proc. of ACL*, pages 136–145.

Sarath Chandar, Mitesh M. Khapra, Hugo Larochelle, and Balaraman Ravindran. 2015. Correlational neural networks. *CoRR*, abs/1504.07225.

Minmim Chen, Zhixiang (Eddie) Xu, and Kilian Q. Weinberger. 2012. Marginalized denoising autoencoders for domain adaptation. In *Proc. of ICML*.

Yanqing Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2013. The expressive power of word embeddings. In *Proc. of ICML Workshop*.

Joshua Coates and Danushka Bollegala. 2018. Frustratingly easy meta-embedding – computing meta-embeddings by averaging source word embeddings. In *Proc. of NAACL-HLT*.

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

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407.

L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, z. Solan, G. Wolfman, and E. Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20:116–131.

Josu Goikoetxea, Eneko Agirre, and Aitor Soroa. 2016. Single or multiple? combining word representations independently learned from text and wordnet. In *Proc. of AAAI*, pages 2608–2614.

Felix Hill, Kyunghyun Cho, Sébastian Jean, Coline Devin, and Yoshua Bengio. 2014. Not all neural embeddings are born equal. In *NIPS workshop*.

Felix Hill, Kyunghyun Cho, Sébastian Jean, Coline Devin, and Yoshua Bengio. 2015. Embedding word similarity with neural machine translation. In *ICLR Workshop*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proc. of ACL*, pages 873–882.

David A. Jurgens, Saif Mohammad, Peter D. Turney, and Keith J. Holyoak. 2012. Measuring degrees of relational similarity. In *Proc. of SemEval*.

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

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proc. ICLR*.

Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *CoNLL*.

Peng Li, Yang Liu, and Maosong Sun. 2013. Recursive autoencoders for ITG-based translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 567–577, Seattle, Washington, USA, October. Association for Computational Linguistics.

Jiwei Li, Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1106–1115, Beijing, China, July. Association for Computational Linguistics.

Yong Luo, Jian Tang, Jun Yan, Chao Xu, and Zheng Chen. 2014. Pre-trained multi-view word embedding using two-side neural network. In *Proc. of AAAI*, pages 1982–1988.

Tomas Mikolov, Kai Chen, and Jeffrey Dean. 2013a. Efficient estimation of word representation in vector space. In *Proc. of ICLR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*, pages 3111–3119.

G. Miller and W. Charles. 1998. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

Andriy Mnih and Geoffrey E. Hinton. 2009. A scalable hierarchical distributed language model. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Proc. of NIPS*, pages 1081–1088.

Sarath Chandar A P, Sanislas Lauly, Hugo Larochelle, Mitesh M. Khapra, Balaraman Ravindran, Vikas C. Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *NIPS'14*.

Gustavi Henrique Paetzold and Lucia Specia. 2016. Inferring psycholinguistic properties of words. In *Proc. of NAACL-HLT*, pages 435–440.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity. In *Proceedings of ACL*, pages 271–278.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proc. of ACL*, pages 115–124.

Jeffery Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: global vectors for word representation. In *Proc. of EMNLP*, pages 1532–1543.

H. Rubenstein and J.B. Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8:627–633.

Warner Slack. 1980. The scholastic aptitude test: A critical appraisal. *Harvard Educational Review*, 50(2):154–175.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*, pages 1631–1642.

Yuta Tsuboi. 2014. Neural networks leverage corpus-wide information for part-of-speech tagging. In *Proc. of EMNLP*, pages 938–950.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proc. of ACL*, pages 384 – 394.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Aritificial Intelligence Research*, 37:141 – 188.

P.D. Turney. 2005. Measuring semantic similarity by latent relational analysis. In *Proc. of IJCAI'05*, pages 1136–1141.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antonie Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *ICML'08*, pages 1096 – 1103.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relational learning. In *ACL*, pages 1671–1682.

Wenpeng Yin and Hinrich Schütze. 2016. Learning meta-embeddings by using ensembles of embedding sets. In *Proc. of ACL*, pages 1351–1360.

Yftah Ziser and Roi Reichart. 2016. Neural structural correspondence learning for domain adaptation. *arXiv*.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proc. of EMNLP*, pages 1393–1398.

GenSense: A Generalized Sense Retrofitting Model

Yang-Yin Lee[1*], Ting-Yu Yen[1*], Hen-Hsen Huang[1], Yow-Ting Shiue[1], and Hsin-Hsi Chen[1,2]
[1]Department of Computer Science and Information Engineering
National Taiwan University
No. 1, Sec. 4, Roosevelt Road, Taipei, 10617 Taiwan
[2]MOST Joint Research Center for AI Technology and All Vista Healthcare, Taiwan
{yylee, tyyen, hhhuang}@nlg.csie.ntu.edu.tw,
orina1123@gmail.com, hhchen@ntu.edu.tw

Abstract

With the aid of recently proposed word embedding algorithms, the study of semantic similarity has progressed and advanced rapidly. However, many natural language processing tasks need sense level representation. To address this issue, some researches propose sense embedding learning algorithms. In this paper, we present a generalized model from the existing sense retrofitting model. The generalization takes three major components: semantic relations between the senses, the relation strength and the semantic strength. In the experiments, we show that the generalized model outperforms the previous approaches in three aspects: semantic relatedness, contextual word similarity and semantic difference.

1 Introduction

The distributed representation of word model (word embedding) has drawn great interest in recent years due to its ability to acquire syntactic and semantic information from a large unannotated corpus (Mikolov et al., 2013; Pennington et al., 2014). With the pre-trained word embedding, some researches propose post-processing models that incorporate with the existing semantic knowledge into the word embedding model (Faruqui et al., 2015; Yu and Dredze, 2014). However, word embedding models use only one vector to represent a word, and are problematic in some natural language processing applications that require sense level representation (e.g., word sense disambiguation, semantic relation identification, etc.). As a result, some researches try to resolve the polysemy and homonymy issue and introduce sense level embedding, either act as pre-process (Iacobacci et al., 2015) or post-process (Jauhar et al., 2015) fashion.

In this research, we focus on the post-processing sense retrofitting approach and propose *GenSense*, a generalized sense embedding learning framework that retrofits a pre-trained word embedding via incorporating with the semantic relations between the senses, the relation strength and the semantic strength. Although some parts of the idea are not new, it is the first time of putting all the parts into a generalized framework. Our proposed *GenSense* for generating low-dimensional sense embedding is inspired from *sense retro* (Jauhar et al., 2015), but has three major differences. First, we generalize the semantic relations from positive relations (e.g., synonyms, hyponyms, paraphrase, etc.) to positive and negative relations (e.g., antonyms). Second, each relation incorporates with both the semantic strength and the relation strength. Within a semantic relation, there should be a weighting for each semantic strength. For example, although *jewel* has the synonyms *gem* and *rock*, it is clear that the similarity between (*jewel*, *gem*) is higher than (*jewel*, *rock*), and thus (*jewel*, *gem*) should have higher weight. Last, *GenSense* gives different relations with different relation strengths. For example, if the objective is to train a sense embedding that can distinguish between the positive and negative sense, then the weight for the negative relation (e.g., antonyms) should be higher, and vice versa. The experimental results suggest the relation strengths play a role in balancing the relations and are application dependent. With an objective that considers these three parts, the sense vectors can be learned and updated via running a belief propagation process on the relation constrained network.

*These authors contributed equally to this work.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1662–1671
Santa Fe, New Mexico, USA, August 20-26, 2018.

In the experiments, we show that our proposed *GenSense* model outperforms the previous approaches in three types of datasets: semantic relatedness, contextual word similarity and semantic difference. While the generalized model of considering all the relations performs well in the semantic and relatedness tasks, we also find that the antonym relation is in favor of the semantic difference experiment. The remainder of this paper is organized as follows. Section 2 gives a survey on the related works. Section 3 defines the generalized sense retrofitting model. The experimental setup is in Section 4. Section 5 shows and discusses the experimental results. Section 7 concludes the remarks.

2 Related Works

The study of the representation of words has a long history. Early approaches include utilizing the term-document occurrence matrix from a large corpus and then perform dimension reduction techniques such as singular value decomposition (latent semantic analysis) (Bullinaria and Levy, 2007; Deerwester et al., 1990). Beyond that, recent word embedding approaches are more focus on neural-style (Dragoni and Petrucci, 2017; Mikolov et al., 2013; Pennington et al., 2014) and performs well on syntactic and semantic tasks. Apart from the unsupervised word embedding learning models, there are plenty of ontologies that contain lexical knowledge, such as WordNet (Fellbaum, 1998), Roget's 21st Century Thesaurus (Kipfer and Institute, 1993) or the paraphrase database (Pavlick et al., 2015). As a result, many researches combine the word embedding with ontological resources, either in a joint training (Bian et al., 2014; Liu et al., 2016; Yu and Dredze, 2014) or a post-processing (Faruqui et al., 2015) fashion. When the need for sense embedding is getting higher, some researches are inspired from the word level embedding learning model and propose sense level embedding (Iacobacci et al., 2015; Jauhar et al., 2015; Lee and Chen, 2017). Although some evidence shows that the sense embedding cannot improve every natural language processing task (Li and Jurafsky, 2015), the benefit of having a sense embedding for improving tasks that need sense level representation is still in great need (Azzini et al., 2012; Ettinger et al., 2016; Qiu et al., 2016).

3 Generalized Sense Retrofitting Model

Let $V = \{w_1, \dots, w_n\}$ be a vocabulary of a trained word embedding and $|V|$ be its size. The matrix $\hat{Q}$ will be the pre-trained collection of vector representations $\hat{q}_i \in \mathbb{R}^d$, where d is the dimensionality of a word vector. Each $w_i \in V$ is learned using a standard word embedding technique (e.g., GloVe (Pennington et al., 2014) or Word2Vec (Mikolov et al., 2013)). Let $\Omega = (T, E)$ be an ontology that contains the semantic relationship, where $T = \{t_1, \dots, t_m\}$ is a set of senses and $|T|$ is total number of senses. The edge $(i, j) \in E$ indicates a semantic relationship of interest (e.g., synonym) between t_i and t_j. In our scenario, the edge set E consists of several disjoint subsets of interest (i.e., $E = E_{s_1} \cup E_{s_2} \cup \dots \cup E_{s_k}$). For example, $(i, j) \in E_{s_1}$ if and only if t_j is the synonym of t_i. We use $\hat{q}_{t_j}$ to denote the word form vector of t_j (one should notice that $\hat{q}_{t_j}$ and $\hat{q}_{t_k}$ may map to the same vector representation even if $j \neq k$). Then the goal is to learn a new matrix $S = (s_1, \dots, s_m)$ such that each new sense vector is close to its word form vertex and its synonym neighbors. The basic form that considers only synonym relation for the objective of the sense retrofitting model is:

$$\sum_{i=1}^{m} \left[\alpha_1 \beta_{ii} \left\| s_i - \hat{q}_{t_i} \right\|^2 + \alpha_2 \sum_{(i,k) \in E_{s_1}} \beta_{ij} \| s_i - s_k \|^2 \right] \quad (1)$$

where α balances the importance of the word form vertex and the synonym, and βs control the strength of the semantic relations. From equation 1, the learned new sense vectors will close to its synonyms, meanwhile constraining its distance with its original word form vector. In addition, this equation can be further generalized to consider all the relations:

$$\sum_{i=1}^{m} \left[\alpha_1 \beta_{ii} \left\| s_i - \hat{q}_{t_i} \right\|^2 + \alpha_2 \sum_{(i,k) \in E_{s_1}} \beta_{ij} \| s_i - s_k \|^2 + \cdots \right] \quad (2)$$

Apart from the positive sense relation, we now introduce three types of special relations. The first one is the positive contextual neighbor relation s_2. $(i, j) \in E_{s_2}$ if and only if t_j has only one sense. In our model, we use the word form vector to represent the neighbors of the t_is in E_{s_2}. Those neighbors

are viewed as positive contextual neighbors as they learned from the context of a corpus (e.g., word2vec trained with Google News corpus) with positive meaning. The second is the negative sense relation s_3 (e.g., antonym). The negative senses are used in a subtraction fashion for pushing the sense away from the positive meaning. The last is the negative contextual neighbors s_4. Just like the positive contextual neighbors, the negative contextual neighbors were learned from the context of a corpus, but with negative meaning.

Figure 1 illustrates an example of the relation network. In Figure 1, *gay* may have two meanings: (1) bright and pleasant; promoting a feeling of cheer and (2) someone who is sexually attracted to persons of the same sex. If we focus on the first sense, then our model can attract s_{gay_1} to its word form vector $\hat{q}_{gay_1}$, its synonym s_{glad_1} its positive contextual neighbor $\hat{q}_{jolly}$. But in the same time, it will push s_{gay_1} from its antonym s_{sad_1} and its negative contextual neighbor $\hat{q}_{dull}$.

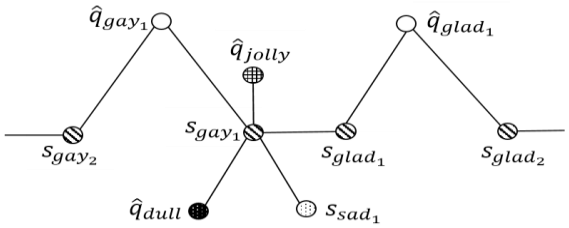


Figure 1. An illustration of the relation network. Different textures of the nodes represent different roles (e.g., synonym, antonym, etc.) in the GenSense model.

To formulize the above scenario and consider all the parts, the equation 2 would become:

$$\sum_{i=1}^{m}\left[\alpha_1\beta_{ii}\left\|s_i-\hat{q}_{t_i}\right\|^2+\alpha_2\sum_{(i,j)\in E_{s_1}}\beta_{ij}\left\|s_i-s_j\right\|^2+\alpha_3\sum_{(i,j)\in E_{s_2}}\beta_{ij}\left\|s_i-\hat{q}_j\right\|^2\right.$$
$$\left.-\alpha_4\sum_{(i,j)\in E_{s_3}}\beta_{ij}\left\|s_i-s_j\right\|^2-\alpha_5\sum_{(i,j)\in E_{s_4}}\beta_{ij}\left\|s_i-\hat{q}_j\right\|^2\right] \quad (3)$$

We therefore apply an iterative updating method to the solution of the above convex objective function (Bengio et al., 2006). Initially, the sense vectors are set to their corresponding word form vectors (i.e., $s_i \leftarrow \hat{q}_{t_i}\ \forall i$). Then in the following iterations, the updating formula for s_i would be:

$$s_i=\frac{-\alpha_5\sum_{j:(i,j)\in E_{s_4}}\beta_{ij}\hat{q}_j-\alpha_4\sum_{j:(i,j)\in E_{s_3}}\beta_{ij}s_j+\alpha_3\sum_{j:(i,j)\in E_{s_2}}\beta_{ij}\hat{q}_j+\alpha_2\sum_{j:(i,j)\in E_{s_1}}\beta_{ij}s_j+\alpha_1\beta_{ii}\hat{q}_{t_i}}{-\alpha_5\sum_{j:(i,j)\in E_{s_4}}\beta_{ij}-\alpha_4\sum_{j:(i,j)\in E_{s_3}}\beta_{ij}+\alpha_3\sum_{j:(i,j)\in E_{s_2}}\beta_{ij}+\alpha_2\sum_{j:(i,j)\in E_{s_1}}\beta_{ij}+\alpha_1\beta_{ii}} \quad (4)$$

A formal description of our proposed GenSense method is shown in Algorithm 1. In Algorithm 1, the β parameters are retrieved from the ontology. The ϵ is a threshold for deciding whether to update the sense vector or not, which is used as a stopping criteria when the difference between the new sense vector and the original sense vector is too small. Experimentally, 10 iterations are sufficient to minimize the objective function from a set of starting vectors to produce effective sense retrofitted vectors.

4 Experiments

We evaluate GenSense with three types of experiments: semantic relatedness, contextual word similarities, and semantic difference. In testing phase, if a test dataset has missing words, we use the average of all sense vectors to represent the missing word. Note that our reported results of vanilla sense embedding may be slightly different from the other researches due to the treatment of missing words and the similarity computation method. Some researches use zero vector to represent the missing words, whereas some remove those missing words from the dataset. However, within this research the reported performance can be compared due to the same missing word processing method and the same similarity computation method.

Algorithm 1 GenSense

Input: A pre-trained word embedding $\hat{Q}$, a relation ontology $\Omega = (T, E)$, hyper-parameters α and parameters β, number of iterations max_it, the convergence criteria for sense vectors ϵ.

Output: A trained sense embedding S

for $i = 1$ **to** m **do**
  $s_i^{(0)} \leftarrow \hat{q}_{t_i}$
end for
for $it = 1$ **to** $max\_it - 1$ **do**
  for $i = 1$ **to** m **do**
    Compute s_i^{tmp} using equation (4).
    if $\left\| s_i^{tmp} - s_i^{(it-1)} \right\| \geq \epsilon$ **then**
      $s_i^{(it)} \leftarrow s_i^{tmp}$
    else
      $s_i^{(it)} \leftarrow s_i^{(it-1)}$
    end if
  end for
end for
return S

4.1 Experimental Setup

We adopt the GloVe model in our experiment (Pennington et al., 2014). The pre-trained GloVe word embedding is trained on Wikipedia and Gigaword-5 (6B tokens, 400k vocab, uncased, 50d vectors). Roget's 21st Century Thesaurus (Kipfer and Institute, 1993) (Roget) is selected for building ontology in our experiments as it contains the strength information of the senses. As Roget does not provide the ontology directly, we manually built a synonym ontology and an antonym ontology from the resource. The vocabulary from GloVe pre-trained word embedding is used for fetching and building the ontology from Roget. In Roget, there are three levels of synonym relations, we set βs to 1.0, 0.6 and 0.3 for the nearest to the farthest synonyms. The antonym relation is built in the same way. For each sense, β_{ii} is set to the sum of all the relation specific weights. Unless specifically address, αs are set to 1 in the experiments. We set the convergence criteria for sense vectors to $\epsilon = 0.1$ with the number of iterations of 10. With the capability of generalization, we run three types of the model: GenSense-syn (only considers the synonyms and positive contextual neighbors), GenSense-ant (only considers the antonyms and negative contextual neighbors) and GenSense-all (considers everything).

4.2 Semantic Relatedness

We downloaded four semantic relatedness benchmark datasets from the web: MEN (Bruni et al., 2014), MTurk (Radinsky et al., 2011), Rare Words (RW) (Luong et al., 2013) and WordSim353 (WS353) (Finkelstein et al., 2001). In MEN dataset, there are two versions of the word pairs: lemma and natural form. We show the natural form in the experimental result, but the performances on two datasets are similar. In each dataset, there is a list of word pairs together with their corresponding human rated scores. A higher score value indicates higher semantic similarity. For example, the score of (*journey*, *voyage*) is 9.29 and the score of (*king*, *cabbage*) is 0.23 in WS353. For measuring the semantic similarity between a word pair (w, w') in the datasets, we adopt the sense evaluation metrics AvgSim and MaxSim (Reisinger and Mooney, 2010):

$$\text{AvgSim}(w, w') \stackrel{\text{def}}{=} \frac{1}{K_w K_{w'}} \sum_{j=1}^{K_w} \sum_{k=1}^{K_{w'}} \cos\left(v_{w_j}, v_{w'_k}\right) \tag{5}$$

$$\text{MaxSim}(w, w') \stackrel{\text{def}}{=} \max_{1 \le j \le K_w, 1 \le k \le K_{w'}} \cos\left(v_{w_j}, v_{w'_k}\right) \quad (6)$$

where K_w and $K_{w'}$ denote the number of senses of w and w', respectively. The AvgSim can be seen as a *soft* metric as it averages all the similarity scores. Whereas the MaxSim can be seen as a *hard* metric as it only selects the senses with maximum similarity score. For measuring the performance of the sense embedding, we compute the spearman correlation between the human rated scores and the AvgSim/MaxSim scores. Table 1 shows a summary of the benchmark datasets and their relationship with the ontologies. In Table 1, row 3 shows the number of words that are both listed in the datasets and the ontology. The word count in Roget is 63,942.

	MEN	**MTurk**	**RW**	**WS353**
Pair count	3,000	287	2,034	353
Word count	751	499	2,951	437
Roget	707	416	2,371	412

Table 1. A summary of the semantic relatedness benchmark datasets.

4.3 Contextual Word Similarity

Although the semantic relatedness datasets are used in many researches, one major disadvantage is that the words in those word pairs do not have contexts. Therefore, we also conduct experiments with the Stanford's Contextual Word Similarities (SCWS) dataset (Huang et al., 2012). SCWS consists of 2,003 word pairs together with human rated scores. A higher score value indicates higher semantic similarity. Different from the semantic relatedness datasets, the words in the SCWS have their contexts and part-of-speech tags. That is, the human subjects can know the usage of the word when they rate the similarity. For each word pair, we compute its AvgSimC/MaxSimC scores from the learned sense embedding (Reisinger and Mooney, 2010):

$$\text{AvgSimC}(w, w') \stackrel{\text{def}}{=} \frac{1}{K^2} \sum_{j=1}^{K} \sum_{k=1}^{\text{K}} d_{c,w,k} d_{c',w',j} d\left(\pi_k(w), \pi_j(w')\right) \quad (7)$$

$$\text{MaxSimC}(w, w') \stackrel{\text{def}}{=} d\left(\hat{\pi}(w), \hat{\pi}(w')\right) \quad (8)$$

where $d_{c,w,k} \stackrel{\text{def}}{=} d\left(v(c), \pi_k(w)\right)$ is the likelihood of context c belonging to cluster π_k, and $\hat{\pi}(w) \stackrel{\text{def}}{=} \pi_{\arg\max_{1 \le k \le K} d_{c,w,k}}(w)$, the maximum likelihood cluster for w in context c. We use a window size of 5 for the words in the word pairs (i.e., 5 words prior to w/w' and 5 words after w/w'). Stop words are removed from the context. For measuring the performance, we compute the spearman correlation between the human rated scores and the AvgSimC/MaxSimC scores.

4.4 Semantic Difference

This task is defined to answer if a word has a closer semantic feature to a concept than another word (Krebs and Paperno, 2016). In this dataset, there are 528 concepts, 24,963 word pairs, and 128,515 items. Each word pair comes with a feature. For example, in the test $(airplane, helicopter)$: $wings$, choosing the first word if and only if $\cos(airplane, wings) > cos(helicopter, wings)$, otherwise, choose the second word. As this dataset does not provide context for disambiguation, we use the similar strategies from the semantic relatedness task:

$$\text{AvgSimD}(w, w') \stackrel{\text{def}}{=} \frac{1}{K_w K_{w'}} \sum_{j=1}^{K_w} \sum_{k=1}^{K_{w'}} \cos\left(v_{w_j}, v_{w'_k}\right) \quad (9)$$

$$\text{MaxSimD}(w, w') \stackrel{\text{def}}{=} \max_{1 \le j \le K_w, 1 \le k \le K_{w'}} \cos\left(v_{w_j}, v_{w'_k}\right) \quad (10)$$

In AvgSimD, we choose the first word iff $\text{AvgSimD}(w_1, w') > \text{AvgSimD}(w_2, w')$. In MaxSimD, we choose the first word iff $\text{MaxSimD}(w_1, w') > \text{MaxSimD}(w_2, w')$. The performance is determined by computing the accuracy.

5 Results and Discussion

Table 2 shows the spearman correlation ($\rho \times 100$) of AvgSim and MaxSim between human scores and sense embedding's scores on each benchmark dataset. Row 2 shows the performance of vanilla GloVe word embedding. Note that the MaxSim and AvgSim scores will be the same when there is only one sense for each word (word embedding). Row 3 shows the performance of the retro model (Jauhar et al., 2015).

	MEN	**MTurk**	**RW**	**WS353**	Macro	Micro
GloVe	65.7	61.9	30.3	50.3	52.1	51.9
retro	62.4/67.7	57.4/60.1	15.1/26.9	43.9/51.1	44.7/51.5	44.0/51.6
GenSense-syn	67.6/67.9	64.1/64.0	**33.8/33.6**	50.5/52.8	54.0/54.6	54.3/54.5
GenSense-ant	65.1/65.0	62.1/63.1	31.0/30.9	48.4/47.1	51.6/51.5	51.7/51.6
GenSense-all	**68.8/68.6**	**65.1/64.8**	33.3/33.2	**53.2/54.0**	**55.1/55.2**	**54.9/54.8**

Table 2. $\rho \times 100$ of (MaxSim / AvgSim) on semantic relatedness benchmark datasets.

From Table 2, we find that our proposed model outperforms the comparison models *retro* and GloVe in all the datasets. When comparing our model with retro, the spearman correlation scores of MaxSim of each dataset grows at least 6.4. In RW, the spearman correlation score of GenSense exceed *retro* by 18.2. We also discover a significant growth of spearman correlation between GenSense-syn and GenSense-all. Surprisingly, the model that only adopts synonyms and positive contextual information can outperform *retro* and GloVe. After utilizing antonym knowledge from Roget, its performance can further be improved in all but RW dataset. This result supports an assumption that the antonyms in Roget are quite informative and useful. Moreover, GenSense can adapt information from synonyms and antonyms to boost its performance. Although our model can pull sense vectors away from its reverse sense with the help of antonym and negative contextual information. This shift cannot guarantee the new sense vectors will move to a better place with only negative relations. As a result, the GenSense-ant does not perform as well as GenSense-syn. Table 2 also shows the macro-averaged and micro-averaged results in the rightmost two columns. In both of the additional evaluation metrics, we find that the GenSense model outperforms *retro* with a large margin. These two metrics suggest the robustness of our proposed model when comparing to the *retro* model.

We also conduct an experiment to test how much benefit we can get from the relation strength. We run GenSense-syn over the Roget ontology with a grid of $(\alpha_1, \alpha_2, \alpha_3)$ parameters. Each parameter is tuned from 0.0 to 1.0 with a 0.1 step size. Table 3 shows the results with MaxSim metric and Table 4 shows the results with AvgSim metric. Note that the $\alpha_1/\alpha_2/\alpha_3$ parameter combinations of the worst or the best case may be more than one. In that case, we only report one $\alpha_1/\alpha_2/\alpha_3$ setting in Table 3 and Table 4 due to the space limitation. From Table 3, we find that the default setting can achieve relatively good results when comparing to the best case. Another point worth mentioning is that the worst performance happens under .1/1./.1 setting except the WS353 dataset. Similar results can be found in Table 4's AvgSim metric. The results demonstrate the importance of the original word vector and the synonyms sense vectors in the model.

	MEN	$\alpha_1/\alpha_2/\alpha_3$	**MTurk**	$\alpha_1/\alpha_2/\alpha_3$	**RW**	$\alpha_1/\alpha_2/\alpha_3$	**WS353**	$\alpha_1/\alpha_2/\alpha_3$
GenSense default	67.6	1./1./1.	64.1	1./1./1.	33.8	1./1./1.	50.5	1./1./1.
GenSense worst	52.4	.1/1./.1	50.3	.1/1./.1	25.1	.1/1./.1	34.8	.1/1./.8
GenSense best	68.1	.8/.5/.8	64.4	.4/.3 /.4	35.8	.1/.1/1.	52.0	1./.6/.3

Table 3. $\rho \times 100$ of MaxSim on semantic relatedness benchmark.

	MEN	$\alpha_1/\alpha_2/\alpha_3$	MTurk	$\alpha_1/\alpha_2/\alpha_3$	RW	$\alpha_1/\alpha_2/\alpha_3$	WS353	$\alpha_1/\alpha_2/\alpha_3$
GenSense default	67.9	1./1./1.	64.0	1./1./1.	33.6	1./1./1.	52.8	1./1./1.
GenSense worst	60.1	.1/1./.1	58.7	.1/1./.1	30.5	.1/1./.1	43.3	.1/1./1.
GenSense best	68.1	.5/.5/.8	64.2	.3/.5/.4	35.8	.1/.1/1.	53.1	.5/.5/.8

Table 4. ρ × 100 of AvgSim on semantic relatedness benchmark datasets.

Figure 2 shows the ρ × 100 of MaxSim on the semantic relatedness benchmark datasets as function of vector dimension. All GloVe pre-trained models are trained on the 6 billion tokens corpus of 50d, 100d, 200d and 300d. We use the GenSense-all model on the GloVe pre-trained models. Figure 2 shows the proposed GenSense-all outperforms GloVe in all the datasets of all the tested dimensions. In GloVe's original paper, they showed GloVe's performance (in terms of accuracy) is proportional to the dimension in the range within 50d and 300d. In this experiment, we show that both GloVe and GenSense-all's performance is proportional to the dimension in the range within 50d and 300d in terms of ρ × 100 of MaxSim. Similar results can be found in the AvgSim metric.

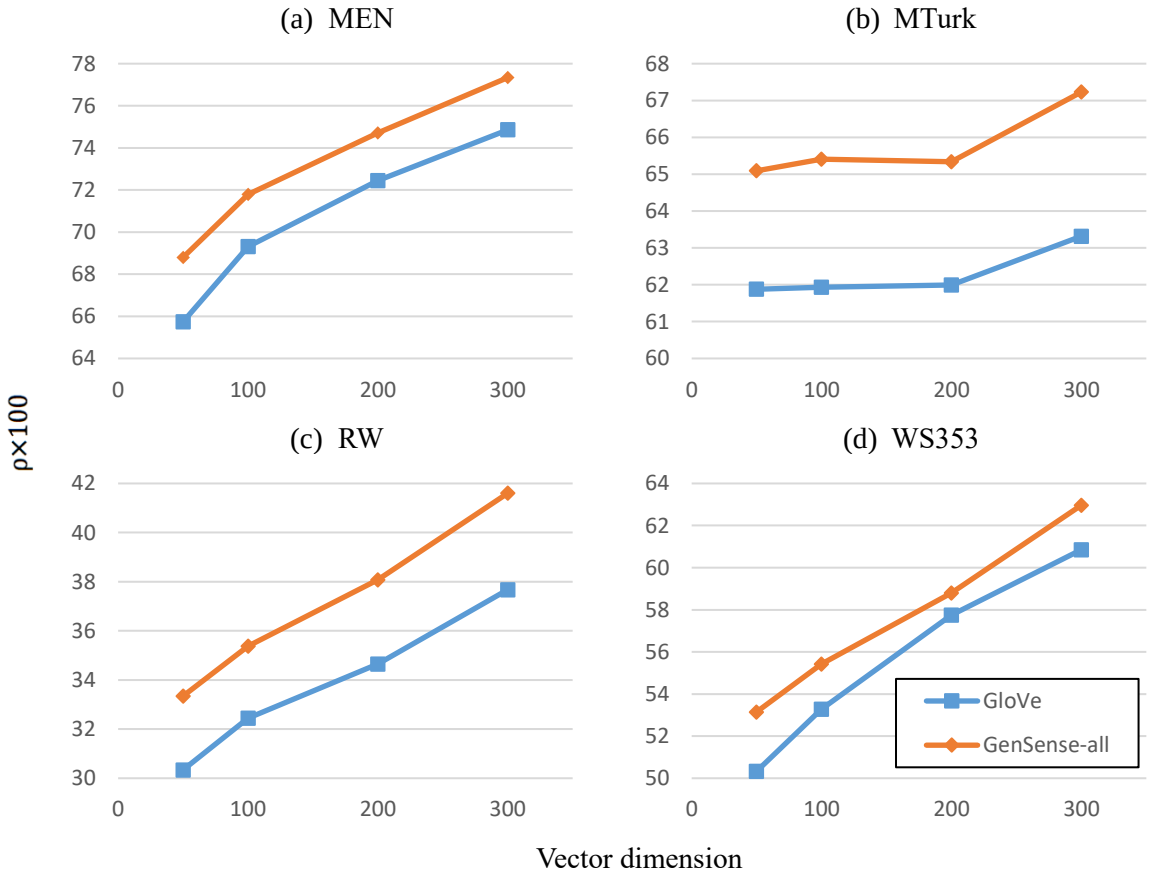


Figure 2. ρ × 100 of MaxSim on semantic relatedness benchmark datasets as function of vector dimension. GloVe model is compared.

Table 5 shows the selected MEN's word pairs and its corresponding GenSense-all, GloVe and retro scores for case study. For GenSense-all, GloVe and retro, we use the MaxSim scores and then sort and re-scale to MEN's score distribution. From Table 5, we find that Gensense-all can improve pre-trained word-embedding model (in terms of closeness to MEN's score, smaller is better) in the following situations: (1) both words have a few senses (*lizard*, *reptiles*), (2) both words have many senses (*stripes*, *train*) and (3) one word has many senses and one word has a few senses (*rail*, *railway*). Sometimes, the retro model increases the closeness to MEN's score. In other words, GenSense-all can handle all the possible situations well and outperform *retro*.

Word pair	#senses	GenSense-all	GloVe	retro
(*rail*, *railway*)	(15, 2)	1	3	36
(*stripes*, *train*)	(20, 17)	2	6	23
(*pregnant*, *women*)	(3, 4)	0	8	17
(*curve*, *dance*)	(10, 7)	2	6	25
(*blue*, *happy*)	(16, 4)	0	5	21
(*dripping*, *round*)	(5, 25)	0	4	24
(*nails*, *wolf*)	(10, 6)	2	6	26
(*action*, *truck*)	(12, 3)	0	9	19
(*lizard*, *reptiles*)	(2, 1)	7	13	28
(*amphibians*, *lizard*)	(3, 2)	9	16	34

Table 5. Selected MEN's word pairs and their scores difference with GenSense-all, GloVe and retro models. (the smaller the better).

Table 6 shows the spearman correlation ($\rho \times 100$) of Stanford's Contextual Word Similarity dataset. With the sense level information, both GenSense and retro can outperform the word embedding model GloVe. The GenSense model performs slightly better than *retro*. Again, we find that the retrofitting model cannot benefit with only negative relation information.

	SCWS
GloVe	52.9
retro	54.2/55.9
GenSense-syn	**54.8/56.0**
GenSense-ant	52.9/52.7
GenSense-all	54.2/55.3

Table 6. $\rho \times 100$ of (MaxSimC / AvgSimC) on SCWS dataset.

Table 7 shows the results of the semantic difference experiment. From Table 7, we find that GenSense outperforms *retro* and GloVe. The accuracy of *retro* declines in this experiment. This finding demonstrates the effectiveness and robustness of our proposed framework. Surprisingly, the antonym relation plays an important role when computing the semantic difference.

	Accuracy	Precision	Recall
GloVe	58.5	53.3	59.4
retro	57.5/57.3	52.2/51.9	58.0/52.0
GenSense-syn	57.8/57.6	52.5/52.3	61.2/59.8
GenSense-ant	58.0/**58.7**	52.7/**53.3**	59.7/**61.7**
GenSense-all	**58.7**/57.6	**53.3**/52.3	**62.4**/61.0

Table 7. $(\text{Accuracy}, \text{Precision}, \text{Recall}) \times 100$ of (MaxSimD / AvgSimD) on the semantic difference dataset.

6 Conclusion

In this paper we present *GenSense*, a generalized framework for learning sense embedding. The generalization takes in three parts: (1) we extend the synonym relation to positive contextual neighbor relation, antonym relation and negative contextual neighbor relation; (2) within each relation, we consider the semantic strength; and (3) we use relation strength between relations to balance different components. We then conduct experiments in three types of experiments: semantic relatedness, contextual word similarity, and semantic difference and show that the GenSense model outperforms the previous approaches. In the future, one of the possible applications is to apply the generalized sense

representation learnt by the proposed method in downstream natural language processing applications to conduct extrinsic evaluations. We release the source code and the pre-trained model as resource for the research community. [12] Other versions of the sense retrofitted embeddings can be found in the website.

Acknowledgements

This research was partially supported by Ministry of Science and Technology, Taiwan, under grants MOST-107-2634-F-002-019, MOST-107-2634-F-002-011 and MOST-106-2923-E-002-012-MY3.

Reference

Antonia Azzini, Célia da Costa Pereira, Mauro Dragoni, and Andrea GB Tettamanzi. 2012. A neuro-evolutionary corpus-based method for word sense disambiguation. *IEEE Intelligent Systems*, 27(6):26–35.

Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. 2006. Label Propagation and Quadratic Criterion. In Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors, *Semi-Supervised Learning*, pages 193–216. MIT Press.

Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Knowledge-powered deep learning for word embedding. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 132–148. Springer.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *The Journal of Artificial Intelligence Research*, 49:1–47.

John A Bullinaria and Joseph P Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods*, 39(3):510–526.

Scott C. Deerwester, Susan T Dumais, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

Mauro Dragoni and Giulio Petrucci. 2017. A neural word embeddings approach for multi-domain sentiment analysis. *IEEE Transactions on Affective Computing*, 8(4):457–470.

Allyson Ettinger, Philip Resnik, and Marine Carpuat. 2016. Retrofitting sense-specific word vectors using parallel text. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1378–1383.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615, Denver, Colorado.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics,* pages 873–882, Jeju, Republic of Korea.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. SensEmbed: learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association*

[1] http://nlg.csie.ntu.edu.tw/nlpresource/GenSense
[2] https://github.com/y95847frank/GenSense

for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 95–105, Beijing, China.

Sujay Kumar Jauhar, Chris Dyer, and Eduard Hovy. 2015. Ontologically grounded multi-sense representation learning for semantic vector space models. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 683–693.

B.A. Kipfer and Princeton Language Institute. 1993. *Roget's 21st Century Thesaurus in Dictionary Form: The Essential Reference for Home, School, or Office*. Dell Pub.

Alicia Krebs and Denis Paperno. 2016. Capturing discriminative attributes in a distributional space: Task proposal. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 51–54.

Guang-He Lee and Yun-Nung Chen. 2017. MUSE: Modularizing unsupervised sense embeddings. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 327–337, Copenhagen, Denmark.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1722–1732, Lisbon, Portugal.

Xiaojie Liu, Jian-Yun Nie, and Alessandro Sordoni. 2016. Constraining word embeddings by prior knowledge–application to medical information retrieval. In *Asia Information Retrieval Symposium*, pages 155–167.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevich, and Chris Callison-Burch Ben Van Durme. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers), pages 425–430, Beijing, China.*

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar.

Lin Qiu, Kewei Tu, and Yong Yu. 2016. Context-dependent sense embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 183–191.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346.

Joseph Reisinger and Raymond J Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 545–550.

Variational Attention for Sequence-to-Sequence Models

Hareesh Bahuleyan[*†] **Lili Mou**[*‡] **Olga Vechtomova**[†] **Pascal Poupart**[†]
[†]University of Waterloo, Canada
{hpallika, ovechtomova, ppoupart}@uwaterloo.ca
[‡]AdeptMind Research, Toronto, Canada
doublepower.mou@gmail.com

Abstract

The variational encoder-decoder (VED) encodes source information as a set of random variables using a neural network, which in turn is decoded into target data using another neural network. In natural language processing, sequence-to-sequence (Seq2Seq) models typically serve as encoder-decoder networks. When combined with a traditional (deterministic) attention mechanism, the variational latent space may be bypassed by the attention model, and thus becomes ineffective. In this paper, we propose a variational attention mechanism for VED, where the attention vector is also modeled as Gaussian distributed random variables. Results on two experiments show that, without loss of quality, our proposed method alleviates the bypassing phenomenon as it increases the diversity of generated sentences.[1]

1 Introduction

The variational autoencoder (VAE), proposed by Kingma and Welling (2014), *encodes* data to latent (random) variables, and then *decodes* the latent variables to reconstruct the input data. Theoretically, it optimizes a variational lower bound of the log-likelihood of the data. Compared with traditional variational methods such as mean-field approximation (Wainwright et al., 2008), VAE leverages modern neural networks and hence is a more powerful density estimator. Compared with traditional autoencoders (Hinton and Salakhutdinov, 2006), which are *deterministic*, VAE populates hidden representations to a region (instead of a single point), making it possible to generate diversified data from the vector space (Bowman et al., 2016) or even control the generated samples (Hu et al., 2017).

In natural language processing (NLP), recurrent neural networks (RNNs) are typically used as both the encoder and decoder, known as a sequence-to-sequence (Seq2Seq) model. Although variational Seq2Seq models are much trickier to train in comparison to the image domain, Bowman et al. (2016) succeed in training a sequence-to-sequence VAE and generating sentences from a continuous latent space. Such an architecture can further be extended to a variational encoder-decoder (VED) to transform one sequence into another with the "variational" property (Serban et al., 2017; Zhou and Neubig, 2017).

When applying attention mechanisms (Bahdanau et al., 2015) to variational Seq2Seq models, however, we find the generated sentences are of less variety, implying that the variational latent space is ineffective. The attention mechanism summarizes source information as an *attention vector* by weighted sum, where the weights are a learned probabilistic distribution; then the attention vector is fed to the decoder. Evidence shows that attention significantly improves Seq2Seq performance in translation (Bahdanau et al., 2015), summarization (Rush et al., 2015), etc. In variational Seq2Seq, however, the attention mechanism unfortunately serves as a "bypassing" mechanism. In other words, the variational latent space does not need to learn much, as long as the attention mechanism itself is powerful enough to capture source information.

In this paper, we propose a variational attention mechanism to address this problem. We model the attention vector as random variables by imposing a probabilistic distribution. We follow traditional VAE

[*]The first two authors contributed equally.

[1]Code is available at https://github.com/HareeshBahuleyan/tf-var-attention

Proceedings of the 27th International Conference on Computational Linguistics, pages 1672–1682
Santa Fe, New Mexico, USA, August 20-26, 2018.

and model the prior of the attention vector by a Gaussian distribution, for which we further propose two plausible priors, whose mean is either a zero vector or an average of source hidden states.

We evaluate our approach on two experiments: question generation and dialog systems. Experiments show that the proposed variational attention yields a higher diversity than variational Seq2Seq with deterministic attention, while retaining high quality of generated sentences. In this way, we make VED work properly with the powerful attention mechanism.

In summary, the main contributions of this paper are two-fold: (1) We discover a "bypassing" phenomenon in VED, which could make the learning of variational space ineffective. (2) We propose a variational attention mechanism that models the attention vector as random variables to alleviate the above problem. To the best of our knowledge, we are the first to address the attention mechanism in variational encoder-decoder neural networks. Our model is a general framework, which can be applied for various text generation tasks.

2 Background and Motivation

In this section, we introduce the variational autoencoder and the attention mechanism. We also present a pilot experiment motivating our variational attention model.

2.1 Variational Autoencoder (VAE)

A VAE encodes data $\boldsymbol{Y}$ (e.g., a sentence) as hidden random variables $\boldsymbol{Z}$, based on which the decoder reconstructs $\boldsymbol{Y}$. Consider a generative model, parameterized by $\boldsymbol{\theta}$, as

$$p_{\boldsymbol{\theta}}(\boldsymbol{Z}, \boldsymbol{Y}) = p_{\boldsymbol{\theta}}(\boldsymbol{Z}) p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{Z}) \tag{1}$$

Given a dataset $\mathcal{D} = \{\boldsymbol{y}^{(n)}\}_{n=1}^{N}$, the likelihood of a data point is

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{y}^{(n)}) \geq \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{y}^{(n)})}\left[\log\left\{\frac{p_{\boldsymbol{\theta}}(\boldsymbol{y}^{(n)}, \boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{y}^{(n)})}\right\}\right]$$

$$= \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{y}^{(n)})}\Big[\log p_{\boldsymbol{\theta}}(\boldsymbol{y}^{(n)}|\boldsymbol{z})\Big] - \mathrm{KL}\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{y}^{(n)})\|p(\boldsymbol{z})\Big) \triangleq \mathcal{L}^{(n)}(\boldsymbol{\theta}, \boldsymbol{\phi}) \tag{2}$$

VAE models both $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{y})$ and $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{z})$ with neural networks, parametrized by $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, respectively. Figure 1a shows the graphical model of this process. The training objective is to maximize the lower bound of the likelihood $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi})$, which can be rewritten as minimizing

$$J^{(n)} = J_{\text{rec}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{y}^{(n)}) + \mathrm{KL}\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{y}^{(n)})\|p(\boldsymbol{z})\Big) \tag{3}$$

The first term, called *reconstruction loss*, is the (expected) negative log-likelihood of data, similar to traditional deterministic autoencoders. The expectation is obtained by Monte Carlo sampling. The second term is the KL-divergence between $\boldsymbol{z}$'s posterior and prior distributions. Typically the prior is set to standard normal $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

2.2 Variational Encoder-Decoder (VED)

In some applications, we would like to transform source information to target information, e.g., machine translation, dialogue systems, and text summarization. In these tasks, "auto"-encoding is not sufficient, and an encoding-decoding framework is required. Different efforts have been made to extend VAE to variational encoder-decoder (VED) frameworks, which transform an input $\boldsymbol{X}$ to output $\boldsymbol{Y}$. One possible extension is to condition all probabilistic distributions further on $\boldsymbol{X}$ (Zhang et al., 2016; Cao and Clark, 2017; Serban et al., 2017). In this case, the posterior of $\boldsymbol{z}$ is given by $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{X}, \boldsymbol{Y})$. This, however, introduces a discrepancy between training and prediction, since $\boldsymbol{Y}$ is not available during the prediction stage.

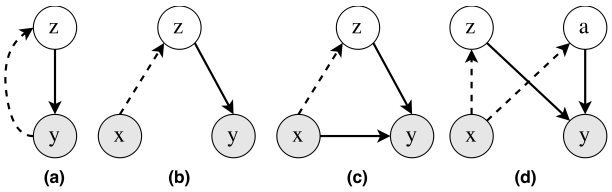


Figure 1: Graphical model representations. **(a)** Variational autoencoder (VAE). **(b)** Variational encoder-decoder (VED). **(c)** VED with deterministic attention (VED+DAttn). **(d)** VED with variational attention (VED+VAttn). **Dashed lines**: Encoding phase. **Solid lines**: Decoding phase.

Input: *the men are playing musical instruments*	
(a) VAE w/o hidden state init. (Avg entropy: 2.52)	**(b) VAE w/ hidden state init. (Avg entropy: 2.01)**
the men are playing musical instruments	*the men are playing musical instruments*
the men are playing video games	*the men are playing musical instruments*
the musicians are playing musical instruments	*the men are playing musical instruments*
the women are playing musical instruments	*the man is playing musical instruments*

Table 1: Sentences obtained by sampling from the VAE's latent space. (a) VAE without hidden state initialization. (b) VAE with hidden state initialization.

Another approach is to build a recognition model using only $\boldsymbol{X}$ (Zhou and Neubig, 2017). Making the assumption that $\boldsymbol{Y}$ is a function of $\boldsymbol{X}$, i.e., $\boldsymbol{Y} = \boldsymbol{Y}(\boldsymbol{X})$, we have $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{y}) = q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{Y}(\boldsymbol{x})) \triangleq q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. In this work, we follow Zhou and Neubig (2017) and adopt this extension. Figure 1b shows the graphical model of the VED used in our work.

2.3 Seq2Seq and Attention Mechanism

In NLP, sequence-to-sequence recurrent neural networks are typically used as the encoder and decoder, as they are suitable for modeling a sequence of words (i.e., sentence). Figure 2a shows a basic Seq2Seq model in the VAE/VED scenario (Bowman et al., 2016). The encoder has an input $\boldsymbol{x}$, and outputs $\boldsymbol{\mu}_z$ and $\boldsymbol{\sigma}_z$ as the parameters of $\boldsymbol{z}$'s posterior normal distribution. Then a decoder generates $\boldsymbol{y}$ based on a sample $\boldsymbol{z}$, drawn from its posterior distribution.

Attention mechanisms are proposed to dynamically align $\boldsymbol{y} = (y_1, \cdots, y_{|\boldsymbol{y}|})$ and $\boldsymbol{x} = (x_1, \cdots, x_{|\boldsymbol{x}|})$ during generation. At each time step j in the decoder, the attention mechanism computes a probabilistic distribution by

$$\alpha_{ji} = \frac{\exp\{\widetilde{\alpha}_{ji}\}}{\sum_{i'=1}^{|\boldsymbol{x}|} \exp\{\widetilde{\alpha}_{ji'}\}} \tag{4}$$

where $\widetilde{\alpha}_{ji}$ is a pre-normalized score, computed by $\widetilde{\alpha}_{ji} = \boldsymbol{h}_j^{(\text{tar})} W^T \boldsymbol{h}_i^{(\text{src})}$ in our model. Here, $\boldsymbol{h}_j^{(\text{tar})}$ and $\boldsymbol{h}_i^{(\text{src})}$ are the hidden representations of the jth step in target and ith in the source, and W is a learnable weight matrix.

Then the source information $\{\boldsymbol{h}_i^{(\text{src})}\}_{i=1}^{|\boldsymbol{x}|}$ is summed by weights α_{ji} to obtain the attention vector

$$\boldsymbol{a}_j = \sum_{i=1}^{|\boldsymbol{x}|} \alpha_{ji} \boldsymbol{h}_i^{(\text{src})} \tag{5}$$

which is fed to the decoder RNN at the jth step. Figure 2b shows the variational Seq2Seq model with such traditional attention.

2.4 "Bypassing" Phenomenon

In this part, we explain the "bypassing" phenomenon in VAE/VED, if the network is not designed properly; this motivates our variational attention described in Section 3.

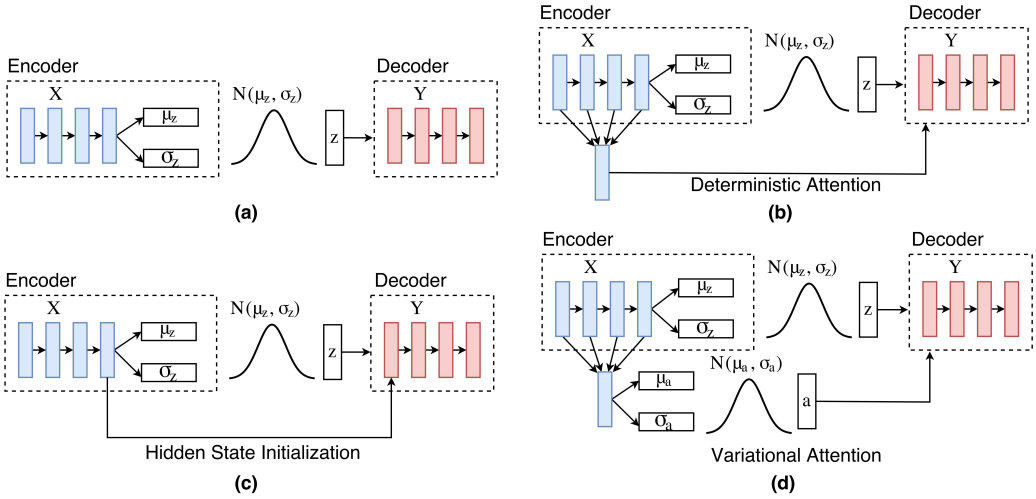


Figure 2: (a) Variational Seq2Seq model. (b) Variational Seq2Seq with deterministic attention. (c) Variational Seq2Seq with hidden state initialization. (d) Variational Seq2Seq with variational attention.

We observe that, if the decoder has a direct, deterministic access to the source, the latent variables $\boldsymbol{Z}$ might not capture much information so that the VAE or VED does not play a role in the process. We call this a *bypassing phenomenon*.

Theoretically, if $p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\cdot)$ is aware of $\boldsymbol{X}$ by itself, i.e., $p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\cdot)$ becomes $p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{Z})$, it could be learned as $p_{\boldsymbol{\theta}}(\boldsymbol{Y}|\boldsymbol{X})$ without hurting the reconstruction loss J_{rec}, but the KL term in Eq. (3) can be minimized by fitting the posterior to its prior. This degrades a variational Seq2Seq model to a deterministic one.

The phenomenon can be best shown with a bypassing connection between the encoder and decoder for hidden state initialization. Some previous studies using VEDs set the decoder's initial state to be the encoder's final state (Cao and Clark, 2017), shown in Figure 2c. We conducted a pilot study with a Seq2Seq VAE with a subset (∼80k samples) of the massive dataset provided by Bowman et al. (2015), and show generated sentences and entropy in Table 1. We see that the variational Seq2Seq can only generate very similar sentences with such bypassing connections (Table 1b), as opposed to generating diversified samples from the latent space (Table 1a). We also computed the entropy for 10 randomly sampled outputs for a given input sentence. Quantitatively, the entropy decreases by 0.5 on average for 1k unseen input sentences. This shows a significant difference because entropy is a logarithmic metric. Our analysis sheds light on the design philosophy of neural architectures in VAE or VED.

Since attention largely improves model performance for deterministic Seq2Seq models, it is tempting to include attention in the variational Seq2Seq as well. However, our pilot experiment raises the doubt if a traditional attention mechanism, which is deterministic, may bypass the latent space in VED, as illustrated by a graphical model in Figure 1c. Also, evidence in Zheng et al. (2018) shows the attention mechanism is so powerful that removing other connections between the encoder and decoder has little effect on BLEU scores in machine translation. Therefore, a VED with deterministic attention might learn reconstruction mostly from attention, whereas the posterior of the latent space can fit to its prior in order to minimize the KL term.

To alleviate this problem, we propose a variational attention mechanism for variational Seq2Seq models, as is described in detail in the next section.

3 The Proposed Variational Attention

Let us consider the decoding process of an RNN. At each timestep j, it adjusts its hidden state $\boldsymbol{h}_j^{(\text{tar})}$ with an input of a word embedding $\boldsymbol{y}_{j-1}$ (typically the groundtruth during training and the prediction from the previous step during testing). This is given by $\boldsymbol{h}_j^{(\text{tar})} = \text{RNN}_{\boldsymbol{\theta}}(\boldsymbol{h}_{j-1}^{(\text{tar})}, \boldsymbol{y}_{j-1})$. In our experiments, we use long short-term memory units (Hochreiter and Schmidhuber, 1997) as RNN's transition. Enhanced

with attention, the RNN is computed by $\boldsymbol{h}_j^{(\text{tar})} = \text{RNN}_{\boldsymbol{\theta}}(\boldsymbol{h}_{j-1}^{(\text{tar})}, [\boldsymbol{y}_{j-1}; \boldsymbol{a}_j])$. The predicted word is given by a softmax layer $p(y_j) = \text{softmax}(W_{\text{out}}\boldsymbol{h}_j^{(\text{tar})})$, where W_{out} is a weight matrix. As discussed earlier, traditional attention computes $\boldsymbol{a}_j$ in a deterministic fashion by Eq. (5).

To build a variational attention, we treat both the traditional latent space $\boldsymbol{z}$ and the attention vector $\boldsymbol{a}_j$ as random variables. The recognition and reconstruction graphical models are shown in Figure 1d.

3.1 Lower Bound

Since the likelihood of the nth data point decomposes for different time steps, we consider the lower bound $\mathcal{L}_j^{(n)}(\boldsymbol{\theta}, \boldsymbol{\phi})$ at the jth step. The variational lower bound in Eq. (2) becomes

$$\mathcal{L}_j^{(n)}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{z},\boldsymbol{a}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z},\boldsymbol{a}|\boldsymbol{x}^{(n)})}\Big[\log p_{\boldsymbol{\theta}}(\boldsymbol{y}^{(n)}|\boldsymbol{z},\boldsymbol{a})\Big] - \text{KL}\Big(q_{\boldsymbol{\phi}}(\boldsymbol{z},\boldsymbol{a}|\boldsymbol{x}^{(n)})\|p(\boldsymbol{z},\boldsymbol{a})\Big) \tag{6}$$

$$= \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}^{(z)}(\boldsymbol{z}|\boldsymbol{x}^{(n)}),\boldsymbol{a}\sim q_{\boldsymbol{\phi}}^{(a)}(\boldsymbol{a}|\boldsymbol{x}^{(n)})}\Big[\log p_{\boldsymbol{\theta}}(\boldsymbol{y}^{(n)}|\boldsymbol{z},\boldsymbol{a})\Big] \tag{7}$$
$$- \text{KL}\Big(q_{\boldsymbol{\phi}}^{(z)}(\boldsymbol{z}|\boldsymbol{x}^{(n)})\|p(\boldsymbol{z})\Big) - \text{KL}\Big(q_{\boldsymbol{\phi}}^{(a)}(\boldsymbol{a}|\boldsymbol{x}^{(n)})\|p(\boldsymbol{a})\Big)$$

Eq. (7) is due to the independence in both recognition and reconstruction phrases. The posterior factorizes as $q_\phi(\boldsymbol{z},\boldsymbol{a}|\cdot) = q_\phi^{(z)}(\boldsymbol{z}|\cdot)\, q_\phi^{(a)}(\boldsymbol{a}|\cdot)$ because $\boldsymbol{z}$ and $\boldsymbol{a}$ are conditionally independent given $\boldsymbol{x}$ (dashed lines in Figure 1d), whereas the prior factorizes because $\boldsymbol{z}$ and $\boldsymbol{a}$ are marginally independent (solid lines in Figure 1d). In this way, the sampling procedure can be done separately and the KL loss can also be computed independently.

3.2 Prior

We propose two plausible prior distributions for $\boldsymbol{a}_j$.

- The simplest prior, perhaps, is the standard normal, i.e., $p(\boldsymbol{a}_j) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. This follows the prior of the latent space $\boldsymbol{z}$ as in a conventional autoencoder (Kingma and Welling, 2014; Bowman et al., 2016).
- We observe that the attention vector has to be inside the convex hull of hidden representations of the source sequence, i.e., $\boldsymbol{a}_j \in \text{conv}\{\boldsymbol{h}_i^{(\text{src})}\}$. We impose a normal prior whose mean is the average of $\boldsymbol{h}_i^{(\text{src})}$, i.e., $p(\boldsymbol{a}_j) = \mathcal{N}(\bar{\boldsymbol{h}}^{(\text{src})}, \mathbf{I})$, where $\bar{\boldsymbol{h}}^{(\text{src})} = \frac{1}{|\boldsymbol{x}|}\sum_{i=1}^{|\boldsymbol{x}|}\boldsymbol{h}_i^{(\text{src})}$, making the prior non-informative.

3.3 Posterior

We model the posterior of $q_{\boldsymbol{\phi}}^{(a)}(\boldsymbol{a}_j|\boldsymbol{x})$ as a normal distribution $\mathcal{N}(\boldsymbol{\mu}_{a_j}, \boldsymbol{\sigma}_{a_j})$, where the parameters $\boldsymbol{\mu}_{a_j}$ and $\boldsymbol{\sigma}_{a_j}$ are obtained by a recognition neural network. Similar to VAEs, we compute parameters as if in the deterministic attention in Eq. (5) (denoted by $\boldsymbol{a}_j^{\text{det}}$ in this part) and then transform them by another layer, shown in Figure 2d.

For the mean $\boldsymbol{\mu}_{a_j}$, we apply an identity transformation, i.e., $\boldsymbol{\mu}_{a_j} \equiv \boldsymbol{a}_j^{\text{det}}$. The identify transformation makes much sense as it preserves the spirit of "attention." To compute $\boldsymbol{\sigma}_{a_j}$, we first transform $\boldsymbol{a}_j^{\text{det}}$ by a neural layer with $\tanh$ activation. The resulting vector then undergoes a linear transformation followed by an $\exp$ activation function to ensure that the values are positive.

3.4 Training Objective

The overall training objective of Seq2Seq with both variational latent space $\boldsymbol{z}$ and variational attention $\boldsymbol{a}$ is to minimize

$$J^{(n)}(\boldsymbol{\theta}, \boldsymbol{\phi}) = J_{\text{rec}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{y}^{(n)}) + \lambda_{\text{KL}}\Big[\text{KL}\Big(q_\phi^{(z)}(\boldsymbol{z}|\boldsymbol{x}^{(n)})\|p(\boldsymbol{z})\Big) + \gamma_a \sum_{j=1}^{|\boldsymbol{y}|}\text{KL}\Big(q_\phi^{(a)}(\boldsymbol{a}_j|\boldsymbol{x}^{(n)})\|p(\boldsymbol{a}_j)\Big)\Big] \tag{8}$$

Here, we have a hyperparameter λ_{KL} to balance the reconstruction loss and KL losses. γ_a further balances the attention's KL loss and $\boldsymbol{z}$'s KL loss. Since VAE and VED are tricky with Seq2Seq models

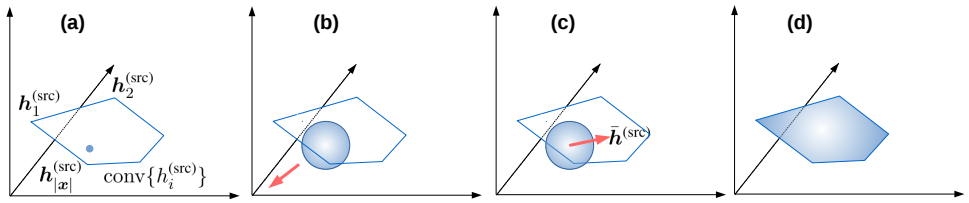


Figure 3: Geometric interpretation of attention mechanisms.

(e.g., requiring KL annealing), we tie the change of both KL terms and only anneal λ_{KL}. (Training details will be presented in Section 4.1.)

Notice that if $\boldsymbol{a}_j$ has a prior of $\mathcal{N}(\bar{\boldsymbol{h}}^{(\text{src})}, \mathbf{I})$, the derivative of the KL term also goes to $\bar{\boldsymbol{h}}^{(\text{src})}$. This can be computed straightforwardly or by auto-differentiation tools, e.g., TensorFlow.

3.5 Geometric Interpretation

We present a geometric interpretation of both deterministic and variational attention mechanisms in Figure 3.

Suppose the hidden representations $\boldsymbol{h}_i^{(\text{src})}$ is of k-dimensional space (represented as a 3-d space in Figure 3). In the deterministic mechanism, the attention model is a convex combination of $\{\boldsymbol{h}_i^{(\text{src})}\}_{i=1}^{|\boldsymbol{x}|}$, as the weights in Eq. (5) are a probabilistic distribution. The attention vector $\boldsymbol{a}_j$ is a point in the convex hull $\text{conv}\{\boldsymbol{h}_i^{(\text{src})}\}$, shown in Figure 3a.

For variational attention in Figures 3b and 3c, the mean of posterior is still in the convex hull, but the sample drawn from the posterior is populated over the entire space (although mostly around the mean, shown as a ball). The difference between the two variants is that the standard normal prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$ pulls the posterior to the origin, whereas the prior $\mathcal{N}(\bar{\boldsymbol{h}}^{(\text{src})}, \mathbf{I})$ pulls the posterior to the mean of $\boldsymbol{h}_1^{(\text{src})}, \boldsymbol{h}_2^{(\text{src})}, \cdots, \boldsymbol{h}_{|\boldsymbol{x}|}^{(\text{src})}$ (indicated by red arrows).

Finally we would like to present a (potential) alternative of modeling variational attention. Instead of treating $\boldsymbol{a}_j$ as random variables, we might also treat $\boldsymbol{\alpha}_j$ as random variables. Since $\boldsymbol{\alpha}_j$ is the parameter of a categorical distribution, its conjugate prior is a Dirichlet distribution. In this case, the resulting attention vector populates the entire convex hull (Figure 3d). However, it relies on a reparametrization trick to propagate reconstruction error's gradient back to the recognition neural network (Kingma and Welling, 2014). In other words, the sampling of latent variables should be drawn from a fixed distribution (without parameters) and then transformed to a desired sample using the distribution's parameters. This is nontrivial for Dirichlet distributions and further research is needed to address this problem.

4 Experiments

We evaluated our model on two tasks: question generation (Section 4.1) and dialog systems (Section 4.2).

4.1 Experiment I: Question Generation

Task, Dataset, and Metrics. We first evaluated our approach on a question generation task. It uses the Stanford Question Answering Dataset (Rajpurkar et al., 2016, SQuAD), and aims to generate questions based on a sentence in a paragraph. We used the same train-validation-test split as in Du et al. (2017). According to Du et al. (2017), the attention mechanism is especially critical in this task in order to generate relevant questions. Also, generated questions do need some variety (e.g., in the creation of reading comprehension datasets), as opposed to machine translation, which is typically deterministic.

We followed Du et al. (2017) and used BLEU-1 to BLEU-4 scores (Papineni et al., 2002) to evaluate the quality (in the sense of accuracy) of generated sentences. Besides, we adopted entropy and *distinct* metrics to measure the diversity. Entropy is computed as $-\sum_w p(w) \log p(w)$, where $p(\cdot)$ is the unigram probability in generated sentences. *Distinct* metrics—used in previous works to measure diversity (Li et al., 2016)—computes the percentage of distinct unigrams or bigrams (denoted as *Dist-1* and *Dist-2*, respectively).

Model	Inference	BLEU-1	BLEU-2	BLEU-3	BLEU-4	Entropy	Dist-1	Dist-2
DED (w/o Attn) (Du et al., 2017)	MAP	31.34	13.79	7.36	4.26	-	-	-
DED (w/o Attn)	MAP	29.31	12.42	6.55	3.61	-	-	-
DED+DAttn	MAP	30.24	14.33	8.26	4.96	-	-	-
VED+DAttn	MAP	**31.02**	14.57	8.49	5.02	-	-	-
	Sampling	30.87	**14.71**	**8.61**	**5.08**	2.214	0.132	0.176
VED+DAttn (2-stage training)	MAP	28.88	13.02	7.33	4.16	-	-	-
	Sampling	29.25	13.21	7.45	4.25	2.241	0.140	0.188
VED+VAttn-0	MAP	29.70	14.17	8.21	4.92	-	-	-
	Sampling	30.22	14.22	8.28	4.87	**2.320**	**0.165**	**0.231**
VED+VAttn-$\bar{h}$	MAP	30.23	14.30	8.28	4.93	-	-	-
	Sampling	30.47	14.35	8.39	4.96	2.316	0.162	0.228

Table 2: BLEU, entropy, and distinct scores. We compare the deterministic encoder-decoder (DED) and variational encoder-decoders (VEDs). For VED, we have several variates: deterministic attention (DAttn) and the proposed variational attention (VAttn). Variational models are evaluated by both max *a posteriori* (MAP) inference and sampling.

Training Details. We used LSTM-RNNs with 100 hidden units for both the encoder and decoder; the dimension of the latent vector $\boldsymbol{z}$ was also 100d. We adopted 300d word embeddings (Mikolov et al., 2013), pretrained on the SQuAD dataset. For both the source and target sides, the vocabulary was limited to the most frequent 40k tokens. We used the Adam optimizer (Kingma and Ba, 2015) to train all models, with an initial learning rate of 0.005, a multiplicative decay of 0.95, and other default hyperparameters. The batch size was set to be 100.

As shown in Bowman et al. (2016), Seq2Seq VAE is hard to train because of the issues associated with the KL term vanishing to zero. Following Bowman et al. (2016), we adopted KL cost annealing and word dropout during training. The coefficient of the KL term λ_{KL} was gradually increased using a logistic annealing schedule, allowing the model to learn to reconstruct the input accurately during the early stages of training. A fixed word dropout rate of 25% was used.

All the hyperparameter tuning was based on validation performance on the motivating Seq2Seq VAE discussed in Section 2.4, and the same hyperparameters were used for all of the models described in Section 3.

Overall Performance. Table 2 represents the performance of various models. We first implemented a traditional vanilla Seq2Seq model, which we call a deterministic encoder-decoder (DED), and generally replicated the results on the question generation task as reported in Du et al. (2017), showing that our implementation is fair. Incorporating attention mechanism in this model (DED+DAttn) improves BLEU scores, as expected. In the variational encoder-decoder (VED) framework, we report results obtained by both max *a posterior* (MAP) inference as well as sampling. In the sampling setting, we draw 10 samples ($\boldsymbol{z}$ and/or $\boldsymbol{a}$) from the posterior given $\boldsymbol{x}$ for each data point, and report average BLEU scores.

The proposed variational attention model (VED+VAttn) largely outperforms deterministic attention (VED+DAttn) in terms of all diversity metrics. It should be noted that entropy is a logarithmic measure, and hence a difference of 0.1 in Table 2 is significant; VED+VAttn also generates more distinct unigrams and bigrams than VED+DAttn.

Regarding the prior of variational attention, we propose two variants: $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\mathcal{N}(\bar{\boldsymbol{h}}^{(\mathrm{src})}, \mathbf{I})$, denoted as VED+VAttn-0 and VED+VAttn-$\bar{h}$, respectively. VED+VAttn-0 has slightly lower BLEU but higher diversity. The results are generally comparable, showing both priors are reasonable.

We also tried a heuristic of 2-stage training (VED+DAttn 2-stage), in which the VED is first trained without attention for 6 epochs, and then the attention mechanism is added to the model. This heuristic is proposed in hopes of better training the variational latent space at the beginning stages. However, experiments show that such simple heuristic does not help much, and is worse than the principled variational attention mechanism in terms of all BLEU and diversity metrics.

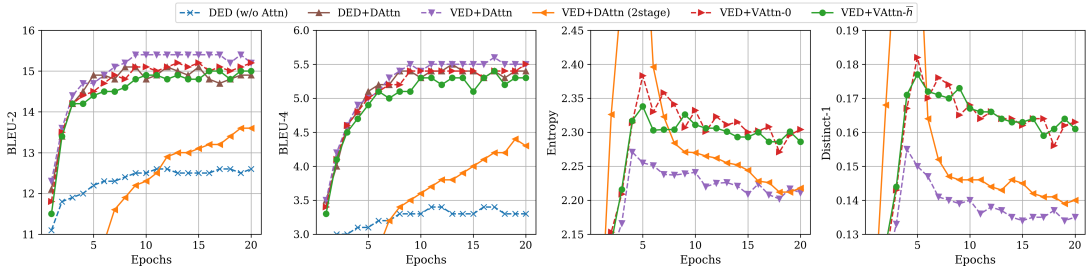


Figure 4: BLEU-2, BLEU-4, Entropy, and *Dist-1* calculated on the validation set as training progresses.

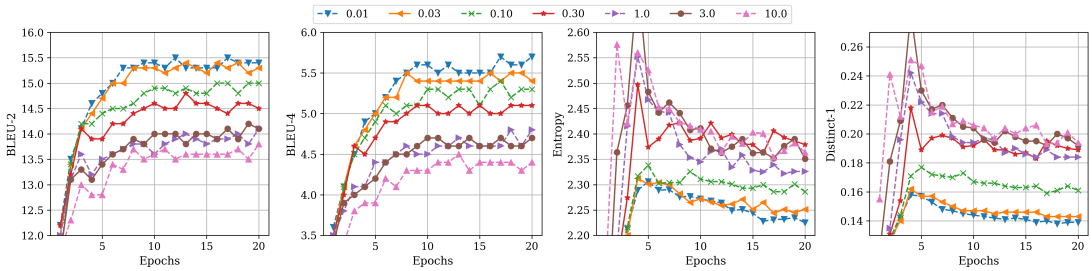


Figure 5: BLEU-2, BLEU-4, Entropy, and *Dist-1* with different γ_a values.

Source	*when the british forces evacuated at the close of the war in 1783 ,* they transported 3,000 freedmen for resettlement in nova scotia .	**Source**	*downstream , more than 200,000 people were evacuated from mianyang by june 1 in anticipation of the dam bursting .*
Reference	*in what year did the american revolutionary war end ?*	**Reference**	*how many people were evacuated downstream ?*
VED+DAttn	*how many people evacuated in newfoundland ?* *how many people evacuated in newfoundland ?* *what did the british forces seize in the war ?*	**VED+DAttn**	*how many people evacuated from the mianyang basin ?* *how many people evacuated from the mianyang basin ?* *how many people evacuated from the mianyang basin ?*
VED+Vattn-$\bar{h}$	*how many people lived in nova scotia ?* *where did the british forces retreat ?* *when did the british forces leave the war ?*	**VED+VAttn-$\bar{h}$**	*how many people evacuated from the tunnel ?* *how many people evacuated from the dam ?* *how many people were evacuated from fort in the dam ?*

Table 3: Case study of question generation.

Human Evaluation. In order to assess the quality of the generated text in terms of language fluency, a human evaluation study was carried out. For each of the two models under comparison (VED+DAttn and VED+VAttn-$\bar{h}$), a randomly shuffled subset of 100 generated questions were selected. Six human evaluators were asked to rate the fluency of these 200 questions on a 5-point scale: 5-Flawless, 4-Good, 3-Adequate, 2-Poor, 1-Incomprehensible, following Stent et al. (2005). The average rating obtained for VED+DAttn was 3.99 and for VED+VAttn-$\bar{h}$ was 4.01, the difference between which is not statistically significant. The human annotations achieved 0.61 average Spearman correlation coefficient (measuring order correlation) between any two annotators. According to Swinscow (1976), this indicates moderate to strong correlation among different annotators. Hence, we conclude variational attention does not negatively affect the fluency of sentences.

Learning curves. Figure 4 shows the trends of sentence quality (BLEU-2 and BLEU-4) and diversity (entropy and *Dist-1*) of all models on the validation set, as training progresses.[2] We see that BLEU and diversity are conflicting objectives: a high BLEU score indicates resemblance to the groundtruth, resulting in low diversity. However, the variational attention mechanisms (red and green lines in Figure 4) remain high in both aspects, showing the effectiveness of our model.

Strength of Attention's KL Loss. We tuned the KL term's strength in variational attention, i.e., γ_a in Eq. (8), and plot the BLEU and diversity metrics in Figure 5. In this experiment, we used the VED+DAttn-$\bar{h}$ variant. As shown, a decrease in γ_a increases the quality of generated sentences at the cost of diversity. This is expected because a lower γ_a gives the model less incentive to optimize the attention's KL term, which then causes the model to behave more "deterministic." Based on this experi-

[2]Other metrics are omitted because the trend is the same.

Model	Inference	BLEU-2	Entropy	Dist-1	Dist-2
DED+DAttn	MAP	1.84	–	–	–
VED+DAttn	MAP	1.68	–	–	–
	Sampling	1.68	2.113	0.311	0.450
VED+VAttn-$\bar{h}$	MAP	1.78	–	–	–
	Sampling	1.79	**2.167**	**0.324**	**0.467**

Table 4: Performance on conversation systems.

ment, we chose a value of 0.1 for γ_a, as it yields a learning curve in the middle among those of different hyperparameters, being a good balance between quality and diversity.

It should be further mentioned that, with a milder γ_a (e.g., 0.01), VED+VAttn outperforms VED+DAttn in terms of both quality and diversity (on the validation set). This is consistent with the evidence that variational latent space may serve as a way of regularization and improves quality (Zhang et al., 2016). However, a small γ_a only slightly improves diversity, and hence we did not choose this hyperparameter in Table 2.

Case study. We show in Table 3 two examples of generated sentences by VED+DAttn and VED+VAttn-$\bar{h}$, each containing three random sentences drawn from the variational latent space(s) for a given input. In both examples, the variational attention generates more diversified sentences than deterministic attention. The quality of generated sentences is close in both models.

4.2 Experiment II: Dialog Systems

We present another experiment on generative conversation systems. The goal is to generate a reply based on a user-issued utterance. We used the Cornell Movie-Dialogs Corpus[3] (Danescu-Niculescu-Mizil and Lee, 2011) as our dataset, which contains more than 200k conversational exchanges. All the settings in this experiment were the same as in Subsection 4.1 except that we had 30k words as the vocabulary for both the encoder and decoder. We evaluated the quality of generated replies with BLEU-2, as it has been observed to be more or less correlated with human annotators among the BLEU metrics (Liu et al., 2016).

Table 4 shows the performance of our model (VED-VAttn-$\bar{h}$) compared with two main baselines. We see that VEDs are slightly worse than the deterministic encoder-decoder (DED) in this experiment. However, variational attention outperforms deterministic attention in terms of both quality and diversity, showing that our model is effective in different applications. However, we find the improvement is not so large as in the previous experiment. We conjecture that in conversational systems, there is a weaker alignment between the source and target information. Hence, the attention mechanism itself is less effective.

5 Related Work

The variational autoencoder (VAE) was proposed by Kingma and Welling (2014) for image generation. In NLP, it has been used to generate sentences (Bowman et al., 2016). Serban et al. (2017) propose a variational encoder-decoder (VED) model to generate better (more diverse and thus meaningful) replies in a dialog system. VED frameworks have also been applied to knowledge base reasoning (Zhang et al., 2018). Another thread of VAE/VED applications is to control some characteristics of generated data, such as the angle of a face image (Kumar et al., 2017), and the sentiment of a sentence (Hu et al., 2017).

In this paper, the focus is on the scenario where VED is combined with attention mechanism. We show that the variational attention space is effective, in terms of the diversity of sampled sentences (since VEDs are probabilistic models). Although previous studies have addressed diversity using diversified beam search (Vijayakumar et al., 2016) and determinantal point processes (Song et al., 2018), we would like to point out that our paper is "orthogonal" to those studies. The diversity in our approach arises

[3] https://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html

through probabilistic modeling, as opposed to a manually specified heuristic function of the diversity metric. It is to be noted that our approach can be naturally combined with the above methods.

6 Conclusion and Future Work

In this paper, we proposed a variational attention mechanism for variational encoder-decoder (VED) frameworks. We observe that, in VEDs, if the decoder has direct access to the encoder, the connection may bypass the variational space. Traditional attention mechanisms might serve as bypassing connection, making the output less diverse. Our variational attention mechanism imposes a probabilistic distribution on the attention vector. We also proposed different priors for the attention vector. The proposed model was evaluated on two tasks: question generation and dialog systems, showing that variational attention yields more diversified samples while retaining high quality.

In future work, it would be interesting to investigate VEDs that model the attention probability with Dirichlet distributions (see Figure 3d). Our framework also provides a principled methodology for designing variational encoding-decoding models without the bypassing phenomenon.

Acknowledgments

We thank Hao Zhou for helpful discussions. The Titan Xp GPU used for this research was donated by the NVIDIA Corporation to Olga Vechtomova.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21.

Kris Cao and Stephen Clark. 2017. Latent variable dialogue models and their diversity. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 182–187.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 76–87.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

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

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1587–1596.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *Proceedings of the International Conference on Learning Representations*.

Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. 2017. Variational inference of disentangled latent concepts from unlabeled observations. *arXiv preprint arXiv:1711.00848*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 3295–3301.

Yiping Song, Rui Yan, Yansong Feng, Yaoyuan Zhang, Zhao DongYan, and Ming Zhang. 2018. Towards a neural conversation model with diversity net using determinantal point processes. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 5932–5939.

Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *Proceedings of International Conference on Intelligent Text Processing and Computational Linguistics*, pages 341–351.

TD Swinscow. 1976. Statistics at square one: Xviii-correlation. *British Medical Journal*, 2(6037):680.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2016. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*.

Martin J Wainwright, Michael I Jordan, et al. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, pages 1–305.

Biao Zhang, Deyi Xiong, jinsong su, Hong Duan, and Min Zhang. 2016. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 521–530.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

Zaixiang Zheng, Hao Zhou, Shujian Huang, Lili Mou, Xinyu Dai, Jiajun Chen, and Zhaopeng Tu. 2018. Modeling past and future for neural machine translation. *Transactions of the Association for Computational Linguistics*, pages 145–157.

Chunting Zhou and Graham Neubig. 2017. Morphological inflection generation with multi-space variational encoder-decoders. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 58–65.

A New Concept of Deep Reinforcement Learning based Augmented General Sequence Tagging System

Yu Wang
Samsung Research America

Abhishek Patel
Samsung Research America

Hongxia Jin
Samsung Research America

Abstract

In this paper, a new deep reinforcement learning based augmented general sequence tagging system is proposed. The new system contains two parts: a deep neural network (DNN) based sequence tagging model and a deep reinforcement learning (DRL) based augmented tagger. The augmented tagger helps improve system performance by modeling the data with minority tags. The new system is evaluated on SLU and NLU sequence tagging tasks using ATIS and CoNLL-2003 benchmark datasets, to demonstrate the new system's outstanding performance on general tagging tasks. Evaluated by F1 scores, it shows that the new system outperforms the current state-of-the-art model on ATIS dataset by 1.9 % and that on CoNLL-2003 dataset by 1.4 %.

1 Introduction

Sequence tagging/labeling is one of the general techniques required for implementing different natural/spoken language understanding (NLU/SLU) tasks, such as: Named-entity recognition (NER), Part-of-speech (POS) Tagging in NLU, or Slot filling in SLU. Though the purpose of the tasks and their sequence labels are different, most of the state-of-the-art results for these tasks are generated using neural network based architectures. For example, Ma et al. demonstrates that their BLSTM-CNN-CRF model (Ma and Hovy, 2016) can achieve state-of-the-art performance for the NER task (without using extra features) on CoNLL-2003 dataset; Liu et al. gives an attention-based bi-directional LSTM model (Liu and Lane, 2016) for the slot-filling task, which also gives the best performance on the popular ATIS dataset (Hemphill et al., 1990) for SLU tasks (This result is obtained by intial paper submission, a better result is published later by Yu et al. (Wang et al., 2018)). The details of these related works will be given in next background section. The solution for general sequence tagging tasks (like NER/Slot filling) should mainly contain two properties: the first one is that the model can handle the language sequence in a word/token level manner, *i.e.*, it can read in the sequence word by word and outputs their corresponding labels; the second property is that the model should capture the contextual information/constraints which helps understand the word/token's tags, either as named-entity or simply slot tags. In(Lample et al., 2016; Ma and Hovy, 2016), the authors use a conditional random filed (CRF) to assist their basic bidirectional LSTM model to capture the contextual constraints in NER task. Similarly, Liu et al. uses an attention based mechanism to take advantages of the weighted collective information from the hidden states of all words in an utterance.

Due to the imbalance of data distribution under different labels, normally the evaluations for different models will be based on their precision and recall, or simply the F1 scores. It can be observed that, though the neural network based models are becoming more and more complicated, it becomes harder to make further improvement on these tagging tasks by using more advanced network structures. The main reasons are from two aspects:

1. Though the tagging models are designed to be more and more complicated, they also becomes more likely to be over-fitting by using more layers, hidden nodes or advance recurrent neural network structures

*: The authors' emails are {yu.wang1@samsung.com, abhishek.p@partner.samsung.com, hongxia.jin@samsung.com}.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1683–1693
Santa Fe, New Mexico, USA, August 20-26, 2018.

(like bidirectional LSTM). A model's performance may easily reach a bottom-neck by simply tuning hyper-parameters or changing network structures.
2. Considering the fact that the numbers of words/tokens under some wrongly labeled tags may be quite small, it may not give us more improvement if we still use the same group of training data by simply changing the model. Normally a model will focus on finding the pattern of the data under majority tags instead of minority. Sometimes, even a new model can improve the performance on minority tags' classes, at the cost of degrading the model's performance on the majority tags with more training data, which is a common issue resulting from the imbalanced data distribution under different tags (He and Garcia, 2009; Chawla et al., 2004).

Weighted sampling is one general solution to resolve the imbalanced data issue by giving more weights to minority tags and fewer to the majority ones (He and Garcia, 2009; Sun et al., 2009). The technique, however, gives the issue of distorting the original tagging distribution, which may adversely affect model's performance on majority tags. Similar approach, like adaptive synthetic sampling (He et al., 2008), also suffers from the same problem.

This gives us the dilemma on dealing with imbalanced tagged data (either for slot filling or NER task): On the one hand, it is a non-trivial task to re-organizing/sampling the data under different tags without any distortion of information; on the other hand, the improvement by changing models has an upper bound which is mainly decided by the data pattern itself instead of the model structure. Due to these two reasons, in this paper, a new concept of augmented tagging system based on deep reinforcement learning is proposed to furture improve the performances of sequence labeling tasks without sacrificing their original data distributions. This novel system will use a deep reinforcement learning based compensatory model to capture the wrong labeled tags and learn their correct labels, hence it can improve the whole model system performance without affecting the original correctly labeled tags. Detailed experiments will be further conducted on two sequence labeling tasks in the domain of SLU/NLU, as NER and slot filling, using public datasets.

The paper is organized as following: In section 2, a brief overview of two common NLP/SLU tagging tasks, *i.e.* NER and Slot filling, are given. Their current state-of-the-art models will also be explained briefly. Section 3 gives a background of deep reinforcement learning (DRL) first, then illustrates our DRL based augmented general tagging system (DAT) in detail. In section 4, two different tagging tasks on different datasets are given, One is a slot filling task performed on the ATIS benchmark dataset. The other is a named-entity recognition (NER) task, tested on the CoNLL-2003 benchmark dataset. Both experiments will compare our new algorithm with their current best-performed models with state-of-the-art results.

2 Background

In this section, a brief background overview on the models for two sequence labeling tasks will be given. Their state-of-the-art neural network structures will also be discussed for further comparison purpose.

2.1 Slot filling in SLU

Slot filling task is one of the fundamental tasks in spoken language understanding. It sequentially labels the words in an utterance using the pre-defined types of attributes or slot tags. The most straight-forward approach for this task is by using a single recurrent neural network (RNN) to generate sequential tags by reading an input sentence word by word. The current state-of-the-art result on ATIS is an attention based bidirectional LSTM model (Liu and Lane, 2016). A brief overview about this approach is given for a better illustration of our new algorithm. The model introduced in (Liu and Lane, 2016) covers both of the intent detection and slot filling tasks. Considering the focus of this paper, we will only discuss the slot filling part of the model. The structure of the attention based BLSTM model for slot filling task is as shown in Figure 1.

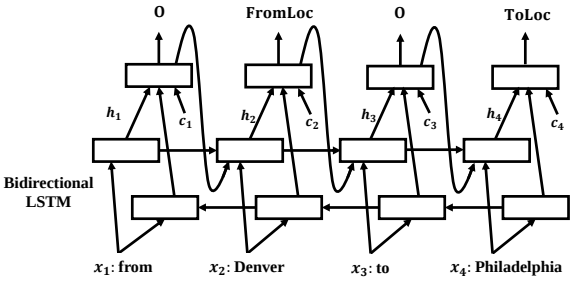


Figure 1: Attention based Bidirectional LSTM for Slot Filling

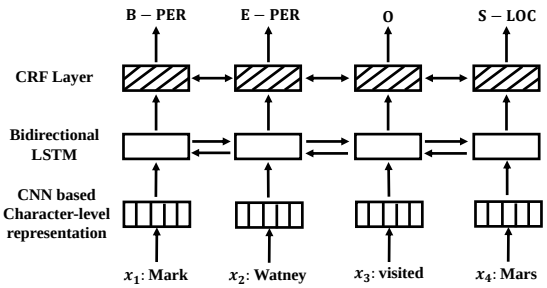


Figure 2: BLSTM-CNN-CRF Structure for Named-entity Recognition

As shown in the figure, a contextual vector $c_i(\cdot)$ is defined using the attention of hidden states h_j:

$$c_i = \sum_{j=1}^{L} \alpha_{i,j} h_j \tag{1}$$

where $\alpha_{i,j}$ is the attention coefficient defined as:

$$\begin{aligned} \alpha_{i,j} &= \frac{e^{e_{i,j}}}{\sum_{k=1}^{L} e^{e_{i,k}}} \\ e_{i,k} &= \phi(s_{i-1}, h_k) \end{aligned} \tag{2}$$

where $\phi(\cdot)$ is a feed-forward neural network and s_{i-1} is the decoder hidden state.
This model consists of tw0 main properties:
1. It uses the Bi-direction LSTM (BLSTM) structure to capture the long-term dependencies in both directions.
2. The attention vector c_i gives additional contextual information, for which cannot be captured by the hidden states in BLSTM.

The model gives the state-of-the-art slot filling performance on ATIS dataset. In this paper, we will also use this model as a baseline for comparison purpose. Also, this attention based model is chosen as the DNN part of our new system, *i.e.*f_{dnn}, to generate the filtered input data to the deep reinforcement learning based augmented tagger (DAT) sub-model of the system.

2.2 Named-entity recognition in NLU

Another very common NLU sequence labeling task is named-entity recognition (NER), which seeks to locate and classify entities into some pre-defined named labels. In this section, a brief overview about the neural network based NER model as in (Ma and Hovy, 2016) will be given. The model gives the state-of-the-art performance on CoNLL-2003 dataset (without lexicon features).

The structure contains a convolutional neural network for generating character-level representations, which is used as an input to a bidirectional LSTM network followed by a conditional random field layer. The conditional random field layer is designed to capture the strong dependency across the labels, which can be formulated as:

$$\begin{aligned} s(X, y) &= \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=0}^{n} P_{i, y_i} \\ p(y|X) &= \frac{e^{s(X,y)}}{\sum_{\tilde{y} \in Y_x} e^{s(X,\tilde{y})}} \end{aligned} \tag{3}$$

where $s(X, y)$ is the score defined a predicted labeling sequence y given X. A_{y_i, y_j} represents transition score from the tag y_i to tag y_j, and P_{i,y_i} is the probability of the y_i output tag of the i^{th} input word

Table 1: Comparison between Minority Tags and Majority Tags on ATIS and CoNLL-2003

	ATIS			CoNLL-2003		
	Training Data		**Test Data**	**Training Data**		**Test Data**
	# of tag types	# of tokens	% of wrongly labeled data	# of tag types	# of tokens	% of wrongly labeled data
Minority Tags (<1% of total # of data)	119	6,323	92%	2	2,312	78%
Majority Tags (≥ 1% of total # of data)	8	38,707	8%	7	202,255	22%
Total	127	45,030	100%	9	204,567	100%

generated by the bidirectional LSTM. Y_X represents all possible tag sequences for an input sentence X, and $p(y|X)$ is the predicted output y's tagging probability giving an input words sequence $X = \{x_1, x_2, \cdots, x_n\}$.

The model shown above gives the state-of-the-art result on the English NER task with an F1 score of 91.2 on CoNLL-2003 dataset (without extra features).

Though decent experimental results are obtained by using deep learning models on the slot filling and NER tagging tasks, the potentials of these models are still limited by the imbalanced data distribution, since most of the tokens don't have any entity names (labeled as 'O'). The model can easily cover the majority tags but not the minority ones. Table 1 gives a summary of minority tags (each tag has less than 1% of entire data) and majority tags on ATIS and CoNLL-2003 datasets, and their corresponding occurrence ratio among the wrongly labeled data in test dataset.

In Table 1, it shows that about 92% of the wrongly labeled data are from minority tags in ATIS dataset and 78% in CoNLL03 dataset. This gives us an indication that, in order to further improve the model performance, the new model needs to figure out how to better label the minority tags and wrongly labeled tags without sacrificing the model performance on majority tags.

3 Deep Reinforcement Learning

In this section, a novel deep reinforcement learning (DRL) based augmented tagging system is proposed to address the issue as described earlier. The system contains two parts: one is the original deep neural network(NN) based tagger (f_{dnn}) as in section 2.1 for slot filling and section 2.2 for NER, the second part is a deep reinforcement learning (DRL) based augmented tagger (f_{dat}), which can be trained to learn the correct tags that are labeled wrongly by the deep learning based tagger, such that it can compensate the weakness of the original single tagger.

3.1 Reinforcement Learning and Deep Q Network (DQN)

Reinforcement learning (RL) is a bit different from the supervised learning and unsupervised learning algorithm. Formulated as a markov decision process (MDP) , an RL based model mainly contains several key elements: state (s_t), action (a_t), rewards (r_t) and policy (π). In a given a state of a stochastic environment, an agent is seeking its best action to perform in order to maximize its expected rewards to be obtained, by following some policy. The main target of a RL based model is to seek the best policy, hence the corresponding action, for an agent to perform. There are mainly three types of reinforcement learning algorithms: value-based, policy gradient and actor-critic. In our scenario, due to the necessity to define rewards at each state and small discrete action space, a value-based model design using Q-learning is proposed for our problem. Its optimal action-value function Q^π, which is also the maximum expected reward obtained by selecting the best policy π, is defined as:

$$Q^\pi(s_t, a_t) = \max_\pi \mathbb{E}[R_t \mid a_t, s_t, \pi] \tag{4}$$

One well-known issue of applying reinforcement learning in real problems is that, in order to compute this optimal action-value function Q^π, it is necessary to store all of the Q values for each state action pair (s_t, a_t) in a table, which is not very practical if the state or action space is large. This problem is described as "the curse of dimensionality" by Bellman (Powell, 2007). One approach to overcome it is

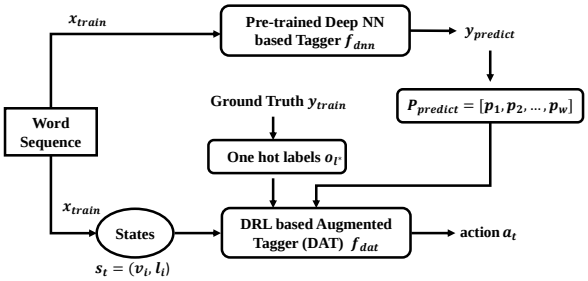


Figure 3: Training Model of DRL based Augmented Tagger

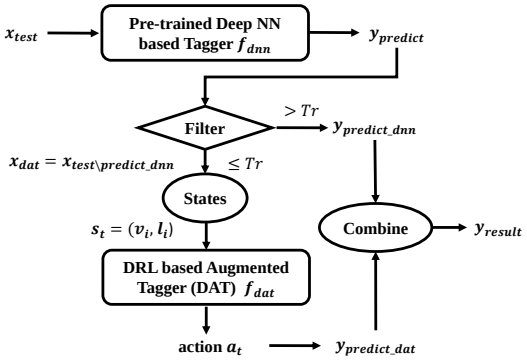


Figure 4: Inference Model of DRL based Augmented Tagger

by training a deep neural network based estimator $Q(s_t, a_t|\theta_t)$ to estimate $Q^\pi(s_t, a_t)$ at time t. However, these types of estimators tend to be unstable since the convergence cannot be fully guaranteed. Recently, the deep Q network (DQN) (Mnih et al., 2013; Mnih et al., 2015) demonstrates much better convergence performance on large state space, by taking advantage of a novel training technique called "experience replay". In this paper, we will use DQN as our selected deep reinforcement learning structure to satisfy the model requirement due to large state space.

In DQN structure, the optimal action-value function $Q^\pi(s_t, a_t)$ is estimated by a neural network based estimator $Q(s_t, a_t|\theta_t)$, *i.e.*

$$Q(s_t, a_t|\theta_t) \sim Q^\pi(s_t, a_t) \tag{5}$$

where θ_t is the neural network parameter, and the state s_t serves as the input of the network. In next section we will describe how to use the techniques in DQN to build our augmented tagger from a system perspective. Also, the states, actions and rewards of the DRL based tagger are also going to be carefully designed next.

3.2 DRL based Augmented Tagger (DAT) System

The design of DRL based augmented tagging (DAT) system is as given in Figure 3 and Figure 4. Due to the differences of using augmented tagger in training and inference, we discuss these two parts separately.

3.2.1 Training DAT System

In order to train the entire tagging system, one may need to first pre-train a deep NN based tagger f_{dnn}, this tagger will follow the design as discussed in section 2.1 for slot filling task or section 2.2 for the NER task. Once f_{dnn} is fully trained, the output labels of f_{dnn} using training input x_{train} are stored as $y_{predict}$ with a probability distribution $P_{y_{predict}} = [p_1, p_2, \cdots, p_w]$, where w is the total number of tags in the entire training dataset. Following is the DRL based design of our new DAT structure:

States (s_t): The DAT model's state s_t is shown in Figure 5. The state is defined by each word/token w_i in sentences, it mainly contains two parts: the first part contains the word level information represented by an n-gram (n is odd) averaged vector v_i, and the other part is a given label l_i of w_i. During the generation of training states, l_i uses all possible tags for the word/token w_i. v_i is defined as the average of the vector of word sequences from $w_{i-(n-1)/2}$ to $w_{i+(n-1)/2}$, where w_j is the center of this sequence:

$$v_i = \frac{1}{n} \sum_{j=i-(n-1)/2}^{j=i+(n-1)/2} w_j \tag{6}$$

Remarks: The reason to use an n-gram vector v_i to substitute w_i as a word level information in a state is due to that we want to better capture the contextual information compared to that using a single word vector. When $n = 1$, v_i is the same as word vector w_i. Also, the average of fewer words/tokens can be used if the index of word sequences is out of the boundary of an input sentence.

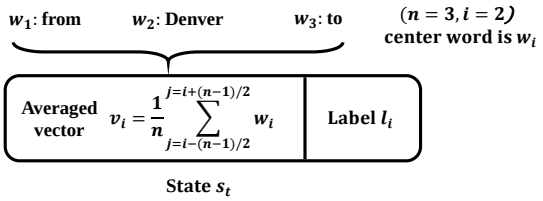


Figure 5: State of DAT

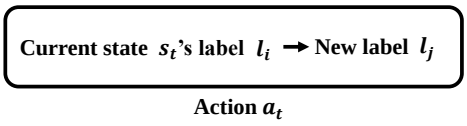


Figure 6: Action of DAT

Actions (a_t): The DAT model's action a_t at time step t is defined as in Figure 6. The action gives a transition signal such that the state will change from its current label l_i to its next label l_j by keep the same n-gram vector v_i. Simply speaking, the action set A contains all possible labels/tags for the word vector w_i or its correspond n-gram substitute v_i, *i.e.* $A = \{l_1, \cdots, l_k\}$. At each time step t, the action a_t with highest predicted action-reward value is chosen as:

$$a_t = \arg\max_a Q(s_t, a|a \in A, \theta_t) \tag{7}$$

Then the state will transit from its current label to the label directed by action a_t.

Rewards (r_t): The reward defined at a state s_t containing an n-gram vector v_i (with a center word/token w_i) will use the one-hot representation $o_{l_i^*}$ of w_i's true label l_i^*, the one hot vector o_{l_i} of the label l_i in current state s_t, and the predicted probability p_i for the word w_i using f_{dnn} as:

$$r_t = tanh(log(\frac{||o_{l_i^*} - p_i||_2}{||o_{l_i} - o_{l_i^*}||_2 + \epsilon})) \tag{8}$$

where $tanh(\cdot)$ is used to normalize the reward to be within -1 and 1, $||\cdot||_2$ is the $\mathcal{L}_2$ norm, and ϵ is a very small value added to avoid zero denominator.

The insight behind our reward design is using the ratio of the distance from f_{dnn} predicted label to $w_i's$ true label and that from current state's label to its true label. Based on our definition, a higher reward will be assigned to a state in which its label is more closer to the true label compared with the f_{dnn}'s predicted one. The reward function is one of the key factors for further improvement using our new augmented tagger.

Remarks: It is worth noticing that p_i should be generated by f_{dnn} using the same word sequence as preparing v_i. Similarly, l_i^* is the true label of w_i in the same sentence of preparing p_i and v_i.

Training: The training algorithm for DAT is as shown in Algorithm 1. The loss function is defined based on the difference of two estimated expected rewards at state s_t using an unsupervised approach as:

$$\begin{aligned} \mathcal{L}_{s_t} &= (\hat{Q}(s_t, a_t|\theta_t) - Q(s_t, a_t|\theta_t))^2 \\ &= (r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}|\theta_t) - Q(s_t, a_t|\theta_t))^2 \end{aligned} \tag{9}$$

where $Q(s_t, a_t|\theta_t)$ is the predicted expected reward using (s_t, a_t) generated by the neural network-based estimator in DAT directly, and $\hat{Q}(s_t, a_t|\theta_t)$ is an estimation of $Q(s_t, a_t|\theta_t)$ using the current state reward r_t and its neighbors' estimations $Q(s_{t+1}, a_{t+1}|\theta_t)$ as in (9).

One training technique we borrowed from (Mnih et al., 2013) is the experience replay used in DQN. It improves the convergence issue in neural network-estimator based Q-learning by storing the state s_t visited before, action performed a_t, state's reward r_t and the next state s_{t+1} after performing the action a_t in an experience tuple (s_t,r_t, a_t, s_{t+1}). This tuple is then pushed into the experience replay memory queue M. The size of M is pre-defined based on experiment.

Whenever an action is performed then a new state is arrived, the past experience tuple will be pushed into the replay memory queue M if M is not full, otherwise M will pop out the first tuple and push in the latest one as First-in First-out (FIFO). At each training iteration, a random tuple is selected from M and the loss function value is calculated based on the s_t, r_t, a_t and s_{t+1} given by the tuple.

Algorithm 1 DAT Training using Experience Replay

Epochs: N
Batch Size: K
Initialize a replay memory M with size μ
for n=1 $\rightarrow$ N **do**
  Time step in state space: $t \leftarrow 0$
  Randomly select an initial state $s_t = (v_i, l_i)$, $l_i^* \leftarrow$ true label of v_i
  while l_i!=l_i^* **do**
    $Q(s_t, a_t|\theta_t) \leftarrow$ DAT(s_t)
    a_t=arg max$_a$ $Q(s_t, a|\theta_t)$
    Perform action a_t, generating next state s_{t+1}
    Push tuple (s_t, r_t, a_t, s_{t+1}) into replay memory M
    for b=1 $\rightarrow$ K **do**
      Randomly select a tuple m from M
      $s_t \leftarrow m(0)$, $r_t \leftarrow m(1)$, $s_{t+1} \leftarrow m(3)$
      $Q(s_t, a_t|\theta_t) \leftarrow$ DAT(s_t)
      $Q(s_{t+1}, a_{t+1}|\theta_t) \leftarrow$ DAT(s_{t+1})
      Obtain p_i from f_{dnn} and o_{l_i} in state s_t, $o_{l_i^*}$ from the ground truth in training data
      $r_t \leftarrow tanh(log(\frac{||o_{l_i^*} - p_i||_2}{||o_{l_i} - o_{l_i^*}||_2 + \epsilon}))$
      Update $\hat{Q}(s_t, a_t|\theta_t)$ in (9)
      Update the network by minimizing loss $\mathcal{L}_{s_t}$ in (9)
    end for
    $t \leftarrow t + 1$
  end while
end for

Remarks: 1. It is worth noticing that it is necessary to use reinforcement learning (RL) in our problem, since RL based model can learn the long-term dependency much better than the "conventional" deep learning approach. In our case, since we split a sentence into multiple word level vectors and select them randomly during training, not only we need to consider what the states current label is, but also how this state/word level vector is affected by its connected state in the network, such that it wont have any difficulty to find the correct label from its current wrong label.
2. In our model, we use a cascaded multiple model structure to boost the system performance. The concept of using information from multiple models (with the same or different structures) to achieve better performance has been widely used in deep learning ((Dean et al., 2012; Ngiam et al., 2011; Wang et al., 2018)), system identification ((Narendra et al., 2014; Narendra et al., 2015b; Murray-Smith and Johansen, 1997; Narendra et al., 2015c; Wang, 2017a; Wang, 2017b)) and also reinforcement learning field ((Narendra et al., 2015a; Wang and Jin, 2018; Narendra et al., 2016)) recently . Instead of using collective information, in this paper, we takes advantage of two models (DNN-based and DRL-based) to compensate each other, such that minor tags can be taken care of and learned.

3.2.2 Model Inference

As shown in Figure 4, the inference part of the DAT model is a bit different from training the DAT. Since during the training procedure, the entire training dataset (x_{train}, y_{train}) is used to train both DNN and DAT models. Comparatively, during inference, only partial of test data with "unsatisfied performance" will be collected and further evaluated by DAT. This is mainly because that we use DAT as an augmented tagger to compensate the minority cases when the original DNN based tagger f_{dnn} does not perform well. For the majority test data $x_{predict\_dnn}$, on which f_{dnn} can perform very well, we still use f_{dnn} to generate their tags $y_{predict\_dnn}$. In order to filter those data with "unsatisfied performance", a threshold value T_r is defined. All the tokens with their predicted tags' probabilities below T_r are filtered as the

minority cases and further used as the inference input of DAT, *i.e.* $x_{dat} = x_{test \backslash predict\_dnn}$. The outputs of DAT are the actions that will transfer the states from their current labels l_i to the target label l_i^*, which gives the output of minority cases, *i.e.* $y_{predict\_dat}$.

Remarks: The choice of T_r will slightly affect the performance of DAT as different percentage of data will be filtered. A general recommended T_r is to use a similar percentage of data under minority tags in training data as shown in section 2.2. For example, we use a T_r to filter around 14%(=6,323/45,030) percent of test data in ATIS for DAT since the training tokens under minority tags in ATIS are roughly around this ratio, as shown in Table 1.

4 Experiment

4.1 Data Sets

In our experiment, we will evaluate our deep reinforcement learning based augmented tagging system on two sequence labeling tasks: Slot filling task and NER task.

Slot Filling: For Slot filling task, we use the public ATIS dataset which follows the same format as in (Liu and Lane, 2015; Mesnil et al., 2015; Xu and Sarikaya, 2013; Liu and Lane, 2016). The training set contains 4,978 utterances/sequences, and the test dataset contains 893 utterances. There are totally 127 slot tags.

NER: For NER task, we use the public CoNLL-2003 dataset as in (Lample et al., 2016; Chiu and Nichols, 2016; Ma and Hovy, 2016; Xu et al., 2017; Huang et al., 2015). The training set contains 14,987 sentences, and the test dataset contains 3,684 sentences. The dataset contains four different types of named entities: *PERSON, LOCATION, ORGANIZATION, and MISC*. By using the BIO2 tagging scheme, the total number of different labels is 9. There are a total of 204,567 tokens in training set, and 46,666 tokens in test set.

4.2 Training Setup

Slot Filling: For slot filling task, the pre-trained DNN model f_{dnn} has the same set-up as in (Liu and Lane, 2016), by using an attention based bi-directional LSTM. The number of states in LSTM cell is 128. The randomly initialized word embedding size is also 128.The batch size is 16 and the dropout rate for non-recurrent connection is 0.5.

NER: For NER task, the pre-trained DNN model f_{dnn} follows the BLSTM-CNNS-CRF structure as in (Ma and Hovy, 2016), which gives the current state-of-the-art result on CoNLL-2003 dataset. The word embedding is chosen as the GloVe 100-dimensional embedding (Pennington et al., 2014). The CNN's window size is chosen as 3 and the number of filters is 30. The number of states in LSTM cell is 200. The batch size is 10 and the dropout rate is 0.5.

The DAT model f_{dat} used for Slot fill and NER tasks are almost the same except the batch size. The network structure chosen to estimate the action-value function Q is an LSTM structure with 100 states. The averaged word vector in a reinforcement learning state is chosen as a trigram, *i.e.* n=3. The discount factor γ in (9) is selected as 0.5, 0.7 and 0.9 for difference experiments. The minibatch size is $K = 16$ for slot filling task and $K = 10$ for NER in order to keep the same training batch size as f_{dnn} in different tasks, and the replay memory size is pre-defined as $\mu = 5,000$. The thresholds for both experiment are set as $T_r = 0.95$.

4.3 Performance on ATIS dataset

Our first experiment is performed on ATIS dataset and compared with other existing approaches, by evaluating their slot filling $F1$ scores. A detailed comparison is given in Table 2.

By using the same DNN based model for f_{dnn} as in (Liu and Lane, 2016), our new model surpassed the previous state-of-the-art result by 0.9%,1.2% and 1.9% for γ= 0.5, 0.7 and 0.9 separately. It is also worth noticing that, a larger discount factor γ gives a better performance as shown in Table. One empirical explanation is that when a larger discount factor is used, the model puts more weights on future states during training, hence it can search the correct label for the current word within a state in a faster manner.

Table 2: Performance of Different Models on ATIS Dataset

Model	F1 Score
Recursive NN (Guo et al., 2014)	93.96%
RNN with Label Sampling (Liu and Lane, 2015)	94.89%
Hybrid RNN (Mesnil et al., 2015)	95.06%
RNN-EM (Peng and Yao, 2015)	95.25%
CNN CRF (Xu and Sarikaya, 2013)	95.35%
Encoder-labeler Deep LSTM (Kurata et al., 2016)	95.66%
Attention Encoder-Decoder NN (Liu and Lane, 2016)	95.87%
Attention BiRNN (Liu and Lane, 2016)	95.98%
DRL based Augmented Tagging System ($\gamma = 0.5$)	**96.85%**
DRL based Augmented Tagging System ($\gamma = 0.7$)	**97.23%**
DRL based Augmented Tagging System ($\gamma = 0.9$)	**97.86%**

Table 3: Performance of Different Models on CoNLL-2003 Dataset

Model	F1 Score
NN+SLL+LM2(Collobert et al., 2011)	88.67%
NN+SLL+LM2+Gazetter* (Collobert et al., 2011)	89.59%
BI-LSTM-CRF* (Huang et al., 2015)	90.10%
ID-CNN-CRF (Strubell et al., 2017)	90.54%
FOFE (Xu et al., 2017)	90.71%
BLSTM-CNN+emb (Chiu and Nichols, 2016)	90.91%
BLSTM-CRF(Lample et al., 2016)	90.94%
BLSTM-CNN-CRFs (Ma and Hovy, 2016)	91.21%
BLSTM-CNN+emb+lex* (Chiu and Nichols, 2016)	91.62%
DRL based Augmented Tagging System ($\gamma = 0.5$)	**91.92%**
DRL based Augmented Tagging System ($\gamma = 0.7$)	**92.23%**
DRL based Augmented Tagging System ($\gamma = 0.9$)	**92.67%**

Table 4: Change on Tags' Distribution of Wrongly Labeled Data on ATIS and CoNLL-2003 Test Datasets

	ATIS Test Dataset		CoNLL-2003 Test Dataset	
	% of wrongly labeled data using f_{dnn}	% of wrongly labeled data using f_{dnn}+f_{dat}	% of wrongly labeled data using f_{dnn}	% of wrongly labeled data using f_{dnn}+f_{dat}
Minority Tags (<1% of total # of data)	92%	56%	78%	48%
Majority Tags ($\geq$ 1% of total # of data)	8%	44%	22%	52%
Total	100%	100%	100%	100%

4.4 Performance on CoNLL-2003 dataset

Our second experiment is conducted on the CoNLL-2003 dataset and compared with current existing neural network-based approaches for the NER tasks. The metric is also using their F1 scores, and the result is as shown in Table 3 (The results are collected before initial submission, some better results are released during final submission (Peters et al., 2018), despite our model's performance is still better).

The methods marked using $*$ are using extra features like lexicons and etc. It can be observed that our DAT system's result outperforms the state-of-the-art results in (Ma and Hovy, 2016) (without lexicon features) and (Chiu and Nichols, 2016) (lexicon features) by 1% and 1.4% using $\gamma = 0.9$ separately. Similar to the SLU task, a larger discount factor γ also boosts our system's performance.

4.5 Change on Result Distribution

Another observation from our experiment's result is that the tags' distribution of wrongly labeled data changed. Table 4 gives a summary about these changes on ATIS and CoNLL-2003 two test dataset.

It can be observed that the percentage of wrongly labeled data with minority tags decreases for both ATIS and CoNLL-2003 datasets by adding the DAT model f_{dat} to the DNN based model f_{dnn}. It indicates that the DAT model f_{dat} can help improve the performance of general sequence tagging model by correctly labeling data with minority tags, which is also the weakness of a single DNN based model f_{dnn}.

5 Conclusion

In this paper, a new DRL based augmented general tagging system is designed. The system use two sequence labeling models: one is a "conventional" DNN based tagger, and the other is a novel DRL based augmented tagger, *i.e.* DAT. By filtering DNN's output and picking out the "unsatisfied data", DAT can further improve sequence labeling tasks' performance by correctly relabeling the data below the threshold T_r, especially for those under minority tags. The experiment results on two sequence labeling tasks, *i.e.* Slot fillings and NER in SLU and NLU, both outperform the current state-of-the-art models. Besides the decent experimental performance obtained, more importantly, the new augmented approach can be generalized to more general sequence labeling models without changing their original model setups.

References

Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. 2004. Special issue on learning from imbalanced data sets. *ACM Sigkdd Explorations Newsletter*, 6(1):1–6.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. 2012. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231.

Daniel Guo, Gokhan Tur, Wen-tau Yih, and Geoffrey Zweig. 2014. Joint semantic utterance classification and slot filling with recursive neural networks. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 554–559. IEEE.

Haibo He and Edwardo A Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284.

Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. 2008. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1322–1328. IEEE.

Charles T Hemphill, John J Godfrey, George R Doddington, et al. 1990. The atis spoken language systems pilot corpus. In *Proceedings of the DARPA speech and natural language workshop*, pages 96–101.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. Leveraging sentencelevel information with encoder lstm for natural language understanding. *arXiv preprint*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Bing Liu and Ian Lane. 2015. Recurrent neural network structured output prediction for spoken language understanding. In *Proc. NIPS Workshop on Machine Learning for Spoken Language Understanding and Interactions*.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(3):530–539.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

Roderick Murray-Smith and T Johansen. 1997. *Multiple model approaches to nonlinear modelling and control*. CRC press.

Kumpati S Narendra, Yu Wang, and Wei Chen. 2014. Stability, robustness, and performance issues in second level adaptation. In *American Control Conference (ACC), 2014*, pages 2377–2382. IEEE.

Kumpati S Narendra, Snehasis Mukhopadyhay, and Yu Wang. 2015a. Improving the speed of response of learning algorithms using multiple models. *arXiv preprint arXiv:1510.05034*.

Kumpati S Narendra, Yu Wang, and Wei Chen. 2015b. Extension of second level adaptation using multiple models to siso systems. In *American Control Conference (ACC), 2015*, pages 171–176. IEEE.

Kumpati S Narendra, Yu Wang, and Wei Chen. 2015c. The rationale for second level adaptation. *arXiv preprint arXiv:1510.04989*.

Kumpati S Narendra, Yu Wang, and Snehasis Mukhopadhay. 2016. Fast reinforcement learning using multiple models. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 7183–7188. IEEE.

Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. 2011. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696.

Baolin Peng and Kaisheng Yao. 2015. Recurrent neural networks with external memory for language understanding. *arXiv preprint arXiv:1506.00195*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Warren B Powell. 2007. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate sequence labeling with iterated dilated convolutions. *arXiv preprint arXiv:1702.02098*.

Yanmin Sun, Andrew KC Wong, and Mohamed S Kamel. 2009. Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(04):687–719.

Yu Wang and Hongxia Jin. 2018. A boosting-based deep neural networks algorithm for reinforcement learning. In *Proceedings of the 2018 American Control Conference*.

Yu Wang, Yilin Shen, and Hongxia Jin. 2018. A bi-model based rnn semantic frame parsing model for intent detection and slot filling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2*, volume 2, pages 309–314.

Yu Wang. 2017a. *Adaptive control and learning using multiple models*. Ph.D. thesis, Yale University.

Yu Wang. 2017b. A new concept using lstm neural networks for dynamic system identification. In *American Control Conference (ACC), 2017*, pages 5324–5329. IEEE.

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 78–83. IEEE.

Mingbin Xu, Hui Jiang, and Sedtawut Watcharawittayakul. 2017. A local detection approach for named entity recognition and mention detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1237–1247.

Learning from Measurements in Crowdsourcing Models: Inferring Ground Truth from Diverse Annotation Types

Paul Felt
IBM Watson*
plfelt@us.ibm.com

Eric K. Ringger
Zillow*
ericri@zillow.com

Jordan Boyd-Graber
Computer Science, iSchool, UMIACS,
and Language Science, University of Maryland
jbg@umiacs.umd.edu

Kevin Seppi
Dept. of Computer Science
Brigham Young University
kseppi@byu.edu

Abstract

Annotated corpora enable supervised machine learning and data analysis. To reduce the cost of manual annotation, tasks are often assigned to internet workers whose judgments are reconciled by crowdsourcing models. We approach the problem of crowdsourcing using a framework for learning from rich prior knowledge, and we identify a family of crowdsourcing models with the novel ability to combine annotations with differing structures: e.g., document labels and word labels. Annotator judgments are given in the form of the predicted expected value of measurement functions computed over annotations and the data, unifying annotation models. Our model, a specific instance of this framework, compares favorably with previous work. Furthermore, it enables active sample selection, jointly selecting annotator, data item, and annotation structure to reduce annotation effort.

Annotierte Korpora ermöglichen überwachtes maschinelles Lernen und Datenanalyse. Um die Kosten für manuelle Annotationen zu vermeiden, werden Aufgaben häufig Internetarbeitern zugewiesen, deren Urteile durch Crowdsourcing-Modelle abgeglichen werden. Wir nähern uns dem Problem des Crowdsourcings, indem wir einen Rahmen für das Lernen aus reichem Vorwissen vorschlagen, und wir bestimmen eine Familie von Crowdsourcing-Modellen mit der Fähigkeit, Annotationen mit unterschiedlichen Strukturen zu kombinieren: z.B., Dokumentbezeichnungen und Wortbezeichnungen. Bewertungen werden in Form des vorhergesagten erwarteten Werts von Messfunktionen (measurement functions) gegeben, die über Annotationen und die Daten berechnet werden. Darin werden die vorherige Annotationsmodelle vereinheitlicht. Unser Modell, eine spezifische Instanz dieses Rahmens, schneidet im Vergleich zu früheren Arbeiten positiv ab. Darüber hinaus ermöglicht es die aktive Stichprobenauswahl, indem Kommentator, Datenelement, und Annotationsstruktur gemeinsam ausgewählt werden, um den Annotationskosten zu reduzieren.

1 Introduction

Supervised machine learning is data hungry: new approaches require massive training sets. These training sets can come from inexpensive crowdsourcing platforms, but consistency is often sacrificed for speed and thrift. Sophisticated models (Surowiecki, 2005; Snow et al., 2008; Jurgens, 2013) can overcome the intrinsic annotation noise by reconciling redundant annotation and predicting true labels by modeling the error patterns associated with individual labels, documents, or annotators.

These models have typically assumed that we collect document-level *labels* and nothing else from annotators. But crowd workers could provide other valuable information. For example, if we wanted to predict the sentiment of documents about the weather (Figure 1), intuition says that words like "sunny",

* This work was completed while the first and second authors were at Brigham Young University.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1694–1704
Santa Fe, New Mexico, USA, August 20-26, 2018.

- *. . . Sunshine: 60s and partly sunny? OK!*
- *. . . shopping, sunshine, margaritas, happiness!*
- *Going to the zoo . . . the weather is perfect.*
- *Another rainy day! Blah.*
- *Cold and rainy here in Boston, Wish I was in ATL*
- *Damn its hot out. Even when not working . . .*
- *Horrible wet morning . . . bring back the sunshine!*

Figure 1: Example tweets from the "Weather Sentiment" dataset available at `http://www.crowdflower.com/data-for-everyone`. Annotators can annotate both documents and *measurements* (e.g., whether a word appears in a document) to label sentiment.

"sunshine", and "perfect" will often appear in positive tweets, and words like "blah", "cold", and "hot" will tend to be more negative. This information is often complementary to labels attached to documents.

However, existing crowdsourcing models cannot simultaneously model multiple kinds of information. This paper introduces a framework and model to combine such diverse information from crowd workers, to measure and model individual annotator errors, and to form consensus final predictions.

We build on the measurements annotation framework (Section 2), which has previously only been applied to the scenario of a single, trusted annotator. Section 3 discusses the application of the framework to crowdsourcing scenarios with multiple untrusted annotators and show that our annotator-aware measurement framework subsumes existing crowdsourcing models. In Section 4 we develop inference for a crowdsourcing measurements model and show that it captures per-annotator variance on both simulated and crowdsourced data in Section 5. We then further extend the model for active query strategies, asking crowd workers for mixed annotations (Section 6).

2 Single-Annotator Measurements

Liang et al. (2009) introduce a supervised learning framework that uses more than the annotation of a document. This section reviews the intuition and notation of this framework that we extend to multiple annotators in the following sections.

The key intuition of the measurement framework is that annotators often have insights that generalize beyond a single document x and label y. For example, in a sentiment labeling task, an annotator could supply the clue that the word "lackluster" often appears in documents with negative sentiment.

These insights are encoded through functions called *measurement features*: $\sigma_k : \mathcal{X}, \mathcal{Y} \mapsto \mathcal{R}$. The measurement feature σ_k tests whether property k holds for the pair x, y, where x is a data item such as a sentence or document and y is its (possibly structured) label. The index k encodes everything needed by a measurement feature to fire only in an extremely specific situation. Measurement features are more specific and correspondingly more sparse than traditional NLP features.

Measurement features can encode traditional supervised labeling. For example, in sentiment classification some measurement feature $\sigma_{k'}$ might encode the fact that Document 343 has a positive label by being zero except when x exactly matches Document 343 and y is positive: $\sigma_{k'} = \mathbb{1}(x = x_{343}, y = Positive)$.

But introducing measurement functions also allows more flexibility. Returning to our "lackluster" example, we can include a function that is one if that word is in the example and the label is negative: $\sigma_{k''} = \mathbb{1}(\text{"lackluster"} \in x, y = Negative)$.

Thus measurement features map the data into an extremely sparse and high-dimensional (partially) observed space. Moreover, measurement features can span multiple documents, so each measurement feature is summed over the dataset $\sigma_k(x, y) = \sum_i \sigma_k(x_i, y_i)$. The learning from measurements framework defines K measurement features, one for every possible labeling. Table 1 provides examples of properties that measurements can encode.

The measurements framework treats observed measurement values τ as the result of measurement noise ϵ applied to the result of measurement features σ.

Measurement Type	Observed $\sigma_k(x, y)$	Partially Observed $\sum_i E_{q(y_i)}[\sigma_k(x, y)]$	Maximum $\max(\sigma_k)$
Document Label	$\mathbb{1}(x = x_m, y = c)$	$q(y_m = c)$	1
Word Label	$\mathbb{1}(f(x) = 1, y = c)$	$\sum_{i \in f(X)} q(y_i = c)$	$\sum_i \mathbb{1}(f(x_i) = 1)$
Label Proportion	$\mathbb{1}(y = c)$	$\sum_i q(y_i = c)$	N

Table 1: The measurement paradigm reformulates the direct labels of traditional supervised learning as indirect measurement features σ and their expected values. If we could directly observe class labels c then we would compute σ (*Observed* column). Since we only have indirect annotation evidence we must learn via the expected values in the *Partially Observed* column. $\mathbb{1}(\cdot)$ is an indicator function. Expected values are defined with respect to the approximate model q (defined in Section 4). All table values remain the same when dealing with annotator-indexed measurements σ_{jk}, although σ_{jk} additionally encodes annotator identity j as well as an implicit annotation value k.

Figure 2 illustrates the measurement framework's generative[1] story:

1. Draw parameter vector θ.
2. Draw measurement noise prior γ.
3. For $i \in N$ instances:
   (a) Draw label y_i from conditional exponential model family $p(y_i \mid x_i, \theta)$.
4. For $k \in K$ measurements:
   (a) Draw measurement noise ϵ_k from $p(\epsilon_k \mid \gamma)$.
   (b) Draw measurement τ_k from $p(\tau_k \mid \sum_i \sigma_k(x_i, y_i), \epsilon_k)$.

Although document labels y are part of the hypothetical generative story according to this model, at inference time y is always unobserved (Figure 2). Like many crowdsourcing models, while the true labels y cannot be observed directly, some byproduct τ of label y can provide evidence for inferring y.

While measurement noise ϵ_k is a part of the generative model, previous implementations of measurement models ignore this component, effectively assuming that all measurements have the same noise. Prior work ignored these considerations because traditional supervised learning training sets lack information about annotators to effectively model per-annotator or per-measurement noise.

In the next section, we correct this omission by extending the measurement model to specifically model not just the measurement noise model but to also estimate the *source* of these errors.

3 Connecting Measurements and Crowdsourcing

This section applies the measurements framework (Section 2) to crowdsourcing scenarios with multiple untrusted annotators and shows the connection to existing crowdsourcing frameworks. The adapted framework (Figure 2) requires two changes to the original measurements framework. First, each measurement k is replicated for each annotator j. This re-indexing of k accommodates annotator-specific parameters that can encode the expertise or focus of particular annotators in a crowdsourcing framework. The generative process is unchanged except for the final step:

4. For $j \in J$ annotators:
   (a) For $k \in K$ measurements:
      i. Draw measurement noise ϵ_{jk} from $p(\epsilon_{jk} \mid \gamma)$.
      ii. Draw measurement τ_{jk} from $p(\tau_{jk} \mid \sum_i \sigma_{jk}(x_i, y_i), \epsilon_{jk})$.

In addition to drawing per-annotator measurements, we also add a hierarchical prior γ over noise parameters (omitted here, as we consider multiple forms of the prior later): this induces parameter tying among measurement noise distributions. For example, this tied parameter can encode trust in annotator j by tying all ϵ_{jk} for annotator j and be left with a single noise parameter ϵ_j per annotator by imposing a prior where $\forall j \exists k, k'(\epsilon_{jk} \neq \epsilon_{jk'}) \implies p(\epsilon|\gamma) = 0$.

[1]The measurements framework omits the extra plate indexed by j. It will be addressed in the next section.

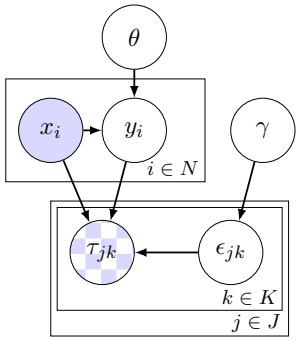


Figure 2: Plate diagram of the measurements framework (Liang et al., 2009) adapted for multiple untrusted annotators. Shaded nodes have observed values. Partially shaded nodes have some observed values. This model subsumes other crowdsourcing models with naïve measurements and extends them to varied annotation environments with richer measurement functions.

The measurements framework leaves the structure of y, the conditional exponential family used to model $p(y \mid x, \theta)$, and definitions of the distributions $p(\theta)$ and $p(\epsilon_{jk})$ unspecified. This paper situates the measurements framework into existing crowdsourcing models. To see the connection, let y_i take on discrete class values, let $p(y_i \mid x_i, \theta) = p(y_i \mid \theta)$ be a data-agnostic multinomial distribution, and let each observed annotation define a document label measurement σ_{jk} (Table 1) where k encodes a specific instance indexed by i, annotation value c', and label value c. Finally, let noise parameters ϵ_j for annotator j be tied to a confusion matrix with Dirichlet-distributed rows such that ϵ_{jk} selects the value at cell (c, c'), encoding how likely j is to produce annotation c' when shown a document whose true label is c. These settings recover the traditional item-response crowdsourcing model (Dawid and Skene, 1979). Using the same settings but defining $p(y \mid x, \theta) \propto exp[\theta^T f(x, y)]$, recovers a popular data-conditional crowdsourcing model (Raykar et al., 2010; Yan et al., 2014; Felt et al., 2015b). However, existing crowdsourcing models lack the representational richness of the measurements framework; we address this lacuna in the next section.

4 Per-annotator Normal Measurement Model for Classification

Having shown how the annotator measurements framework can capture existing crowdsourcing models, this section presents a novel crowdsourcing model that instantiates the richness of the measurements framework that we use in Sections 5–6. For brevity, we refer to this model as PAN (per-annotator normal) measurement model.[2] The generative story is:

1. Draw a stochastic vector θ over C classes from a symmetric Dirichlet Dir(δ)
2. For $i \in N$ documents, draw label y_i from categorical Mult(θ).
3. For $j \in J$ annotators:
   (a) draw measurement noise ϵ_j from inverse gamma IG(α, β).
   (b) For $k \in K$, draw measurement τ_{jk} from a normal Norm($\sum_i \sigma_{jk}(x_i, y_i), \epsilon_j$))

4.1 Learning by Variational Inference

The PAN model's conditional log joint distribution is

$$\log p(\theta, y, \tau \mid x) = \tag{1}$$
$$-\log \text{Beta}(\delta) + J\alpha \log \beta - J \log \Gamma(\alpha) - \sum_j \frac{K_j}{2} \log(2\pi) + \sum_c (\delta + n_c - 1) \log \theta_c$$
$$+ \sum_j (-(\alpha + \frac{K_j}{2}) - 1) \log \epsilon_j - \left(\frac{\beta + \frac{1}{2} \sum_k (\tau_{jk} - \sum_i \sigma_{jk}(x_i, y_i))^2}{\epsilon_j} \right)$$

[2]Data and code available at `https://github.com/BYU-NLP-Lab/Experiments`.

where $\beta\cdot$ is the multivariate beta function and K_j is the number of measurements values observed by annotator j. We use mean-field variational inference as an efficient approximation to full likelihood maximization by assuming a fully factorized approximate model:

$$q(\theta, y \mid \nu) = q(\theta \mid \nu^{(\delta)}) \prod_i q(y_i \mid \nu_i^{(y)}) \prod_j q(\epsilon_j \mid \nu_j^{(\alpha)}, \nu_j^{(\beta)})$$

and then minimizing Kullback-Leibler divergence $\mathrm{KL}(q \,\|\, p)$ via coordinate ascent by iteratively updating the variational parameters ν. That is, $q(\theta \mid \nu^{(\delta)}) \sim \mathrm{Dir}(\nu^{(\delta)})$ where

$$\nu_c^{(\delta)} = \delta + \sum_i \nu_{y_i,c}^{(\delta)} \ . \tag{2}$$

Similarly, $q(\epsilon_j | \nu_j^{(\alpha)}, \nu_j^{(\beta)}) \sim \mathrm{IG}(\nu^{(\alpha)}, \nu^{(\beta)})$ where

$$\nu_j^{(\alpha)} = \alpha + \frac{K_j}{2}, \quad \nu_j^{(\beta)} = \beta + \frac{1}{2} \sum_{k \in S(j)} \mathbb{E}_{q(y_i)} \left[\left(\sum_i \sigma_{jk}(x_i, y_i) \right)^2 \right] \tag{3}$$

where $S(j)$ is the set of measurements provided by annotator j. Although the term $\mathbb{E}_{q(y_i)} \left[(\sum_i \sigma_{jk}(x_i, y_i))^2 \right]$ appears intractable, we can simplify it by introducing terms to complete the square.

$$\mathbb{E}_{q(y_i)} \left[\left(\sum_i \sigma_{jk}(x_i, y_i) \right)^2 \right] = \tag{4}$$
$$\sum_i \left(\mathbb{E}_{q(y_i)} \left[\sigma_{jk}(x_i, y_i) \right] \right)^2 - \sum_i \mathbb{E}_{q(y_i)} \left[\sigma_{jk}(x_i, y_i) \right]^2 + \mathbb{E}_{q(y_i)} \left[\sigma_{jk}(x_i, y_i)^2 \right] .$$

Finally, $q(y_i \mid \nu_i^{(y)}) \sim \mathrm{Mult}(\nu_i^{(y)})$. We update $\nu_i^{(y)}$ by evaluating $\log \nu_i^{(y)} + const$ for each setting of y_i and then exponentiating and normalizing the resulting vector.

$$\log \nu_i^{(y)} = \psi(\nu_{y_i}^{(\delta)}) - \psi(\sum_{y_i} \nu_{y_i}^{(\delta)}) + const + \sum_j \frac{\nu_j^{(\alpha)}}{2\nu_j^{(\beta)}} \Bigg(\sum_{k \in S(i,j)} 2\tau_{jk}\sigma_{jk}(x_i, y_i) \tag{5}$$
$$- \sigma_{jk}(x_i, y_i)^2 - 2\sigma_{jk}(x_i, y_i) \sum_{i' \neq i} \mathbb{E}_{q(y_{i'})} \left[\sigma_{jk}(x_{i'}, y_{i'}) \right] \Bigg)$$

where $S(i, j)$ is the set of measurements provided by annotator j that relate to instance i. More formally, $S(i, j)$ is the set of measurements k where there is some setting of y_i that makes the measurement feature $\sigma_{jk}(x_i, y_i)$ evaluate to a non-zero value.

To get a taste of inference in PAN, consider a simple concrete scenario where four people are labeling the sentiment of a tweet: "Wishing good weather were here again". First Alice labels the tweet positive. Lacking any other information, PAN accepts that for now, calling $\nu_0^{(y)} = [0.75, 0.25]$ and assigning Alice moderate trust in the form of $\mathrm{IG}(\nu_{alice}^{(\alpha)} = 1.6, \nu_{alice}^{(\beta)} = 1.22)$ with a mean error rate of 2.03. Next Bob labels the tweet negative. Since it has no reason to trust Bob more than Alice, the model splits the class vote evenly $\nu_0^{(y)} = [0.5, 0.5]$ and downgrades its trust in both Alice and Bob because of the conflict, assigning them both error rate 2.25. Next Carol labels the word "good" as being positive. Since the word "good" is in our document this tips the balance in favor of positive: $\nu_0^{(y)} = [0.8, 0.2]$. Notice the positive balance is more than $2/3$ since Alice and Carol are now aligned with the model's belief about the true labels and their error rate is upgraded to 2.0, while dissenting Bob's error rate is downgraded to 2.5. Finally, Dave, who happens to be highly trusted *a priori* (perhaps because of admin status or previous good work) comes along and labels the word "wishing" as negative. Since Dave has clout the document swings negative $\nu_0^{(y)} = [0.2, 0.8]$, and Dave and Bob are now aligned with the truth, while Carol and Alice are not. Dave and Bob's error rates go to 0.56 and 2.0, respectively, while Alice and Carol's degrade to 2.5.

4.2 Implementation Considerations

The updates in the previous section contain terms like $\sum_j \sum_k \sum_i \mathbb{E}_{q(y_i)} [\sigma_{jk}(x_i, y_i)]$. Despite the apparent expense of these terms, many of these sums are very sparse and may be computed efficiently. In addition, these values may be cached and incrementally updated as necessary to reduce computational expense.

The scale of measurement features can confuse both users and the algorithm. In Equation 1, observed measurement values τ are compared with the value of measurement features summed over the dataset $\sum_i \sigma(x_i, y_i)$. This latter quantity is bounded by a different range for each measurement feature (see Table 1). However, humans may prefer to give measurement values between 0 and 1, where 1 means "this happens as often as possible." Such values must be scaled by $\max(\sigma_{jk})$.

A related point is that each measurement type is defined on a different scale, but the PAN model fits a common variance to all types. For this to make sense, it is necessary to re-scale each measurement to a common range before running inference. For all experiments reported in this paper, we choose the range $[0 \dots 1]$. Concretely, substitute $\frac{\sigma_{jk}(x_i,y_i)}{\max(\sigma_{jk})}$ for $\sigma_{jk}(x_i, y_i)$, $\frac{E_{q(y_i)}[\sigma_{jk}(x_i,y_i)]}{\max(\sigma_{jk})}$ for $E_{q(y_i)}[\sigma_{jk}(x_i, y_i)]$, and $\frac{\tau_{jk}}{\max(\sigma_{jk})}$ for τ_{jk}.

5 Experiments

This section explores the performance of PAN in rich annotation scenarios. Our goal here is not to establish the PAN model as the state of the art but rather to assess and validate the utility of incorporating diverse annotation types.

5.1 Baselines

We choose two common crowdsourcing baselines which are widely compared against. Despite their simplicity, previous work in establishing benchmark crowdsourcing tasks indicates that these baselines are surprisingly competitive with more sophisticated methods (Sheshadri and Lease, 2013).

Majority Vote (MV) chooses the label with the largest number of votes for each item, ignoring annotator identify. Ties are broken randomly. Although simple, majority vote is widely used and surprisingly successful across a variety of tasks.

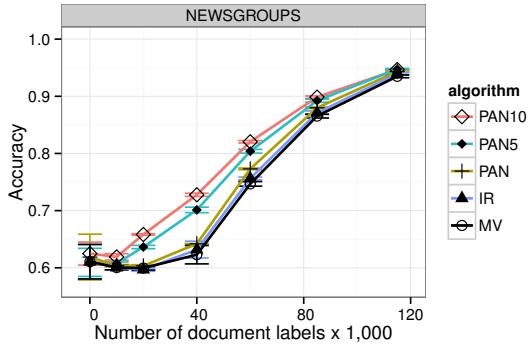
Dawid & Skene (IR). The *item response model* proposed by Dawid and Skene (1979) models annotator error over discrete responses via per-annotator confusion matrices. Much subsequent work uses the same basic structure, making it a good point of comparison. We use a Bayesian version of this model with variational inference adapted from Felt et al. (2015b). The general learning from measurements crowdsourcing framework generalizes this model and many of its extensions (Section 3).

5.2 Simulated Data

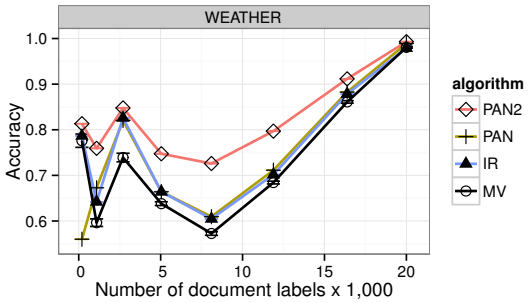
We first generate confidence in our implementation by running on the well-known *20 News Groups* dataset with simulated annotator measurements consistent with PAN's Gaussian noise assumptions. We simulate three different measurements: document labels, word labels, and estimated label proportions. Document label judgments are simulated by corrupting the known true document labels via confusion matrices for five annotators with 50%, 55%, 60%, 65%, and 70% accuracy rates, using the following process:

1. Choose a document randomly without replacement. If none are left, begin again.
2. Choose a simulated annotator randomly with replacement and annotate the document according to that annotator's accuracy.
3. Stop when each of the 20,000 documents has approximately six label annotations.

Word label judgments are simulated by choosing a word w and document label y uniformly at random, and then calculating the empirical rate of seeing documents with label y given that they contain word w. We add Gaussian noise proportional to the total number of documents containing word w and inversely proportional to the accuracy of the annotator. Finally, we manually specify twenty label proportion judgments to encode our *a priori* belief that classes appear roughly the same number of times in the data.



(a) Simulated annotations over the 20 newsgroups corpus. The corpus is annotated until each document has approximately six document label annotations. Performance is competitive with methods with only document labels. However, as we add additional word labels—5,000 (*PAN5*), and 10,000 (*PAN10*)—via the measurements framework, performance increases further.



(b) Real annotations over the weather dataset. PAN is with (PAN2) and without (PAN) access to 2,266 labeled words from CrowdFlower. Labeled words substantially improve PAN's predictions.

Figure 3: Inferred labeled accuracy of Majority vote (MV), the item-response (IR) model, and the per-annotator normal (PAN) measurement model.

In practice, such prior knowledge about approximate label proportions may be available depending on how the data was gathered.

5.3 Inferred label accuracy curves

When the annotation process begins, few documents have annotations. Because crowdsourcing models require annotations to make a prediction, we can plot the accuracy of inferred labels only over those documents having at least one annotation. This means that until the dataset is annotated once, the denominator of the accuracy calculation is growing. Thus inferred label accuracy reflects the corpus accuracy over the subset of documents that have at least one label. This can make inferred label accuracy curves look unlike traditional learning curves, especially when annotation order is not controlled. In real annotation projects, some documents might be annotated multiple times before other documents receive any annotations, resulting in potential accuracy dips as the denominator changes.

PAN is competitive with baselines using only simulated document labels, and it benefits from additional simulated word labels (Figure 3a). Unsurprisingly, there are diminishing returns from additional labeled words: the difference between 0 and 5,000 labeled words is more dramatic than the difference between $5,000$ and $10,000$.

5.4 Sentiment Classification

The same trends hold with real annotator judgments using the "Weather Sentiment" dataset. In this dataset twenty annotators label 1,000 tweets as either `Positive`, `Negative`, `Neutral`, or `Not weather`. Majority vote labels are then evaluated in the related "Weather Sentiment Evaluated" task where a ten secondary annotators judge whether each consensus label is correct or not. For 724 of the tweets, at least nine secondary annotators agree that the consensus majority vote label is correct. We use these high confidence tweets as our gold standard.

Document label judgments are already available for this dataset, but no labeled words or label proportion judgments. We paid CrowdFlower workers to generate labeled words by showing them groups of then randomly selected weather tweets and asking them to list words that characterize each class of tweet. Although a more highly trained workforce could have generated and labeled more sophisticated measurements such as regular expressions, labeled words are a first test. Furthermore, we encode an *a priori* belief that each class occurs roughly the same number of times by manually adding four trusted label proportion measurements stating that each class occurs $N/C = 250$ times. Trusted measurements

are expressed in this framework by authoring measurements under the id of an artificial annotator who is assigned a strong prior distribution of low measurement noise.

From the workers we gather 864 word lists containing 2,482 individual words and covering a vocabulary of 995 unique words. Of the 2,482 word labels, 216 did not match any words in the corpus (we allowed users the freedom to draw on their own background knowledge as well as words of the corpus that they were shown). Interestingly, a brief manual examination of the word lists did not uncover any abusive behavior, although several annotators clearly wanted to match phrases rather than just words. Although the model would permit the use of labeled phrases, our Crowdflower task was not designed to distinguish the two cases, so for now we treat all words as individual judgments. We match word labels to document words after normalizing both by removing punctuation, converting to lower case, applying a Porter stemmer, and removing words from the MALLET stopword list (McCallum, 2002).

Labeled words improve PAN's accuracy. In Figure 3b, the PAN2 line uses $2k$ labeled words and beats vanilla PAN substantially. And while labeled words are not free, it took only 864 judgments to generate our labeled word set, which works out to about $9 at $0.01 per judgment; whereas the 20,000 judgments comprising the document labels would have cost $200 at $0.01 per judgment.

6 Active Measurement Selection

The learning curves in Section 5 assume that annotations were obtained in some arbitrary order, either randomly (Section 5.2) or else ordered by timestamp (Section 5.4). Active learning minimizes annotation costs by obtaining annotations in an order that maximizes their utility. Previous work in active learning either assumes a single, infallible annotator (Settles, 2010; Liang et al., 2009), or else allows multiple annotators but assumes a single kind of annotation (Donmez and Carbonell, 2008; Haertel, 2013; Yan et al., 2011; Nguyen et al., 2015). This paper presents the first active learning results in a setting with multiple annotators and diverse annotation types, jointly selecting annotator, document, and annotation type.

We adapt the active measurement selection algorithm of Liang et al. (2009) to fit the crowdsourcing scenario in Algorithm 1. At each step in the MEASUREMENTSELECTION subroutine, we have a set of observed measurement labels τ_0 and wish to observe a new measurement feature $\sigma_{j,k}$ encoding both the annotator j and the annotation type k (including the document to be annotated for document-centric measurement features).

The NEXTMEASUREMENT subroutine in Algorithm 1 contains our selection criteria, and can be understood as approximating expected net utility:

$$U(\sigma_{jk}) = \mathbb{E}_{p(\tau_{jk} \mid \tau_0, X)} \left[R(\sigma_{jk}, \tau_{jk}) - C(\sigma_{jk}) \right]$$

where $R(\sigma_{jk}, \tau_{jk})$ is the expected reward for obtaining judgment value τ_{jk} for measurement feature σ_{jk}, and $C(\sigma_{jk})$ is the expected cost of obtaining that judgment. To make this computation tractable, we introduce a number of approximations. Lines 11–15 of Algorithm 1 compute the expectation over τ_{jk} using stochastic integration. For PAN, we approximate sampling from the posterior $p(\tau_{jk} \mid \tau_0, X)$ by parameterizing the Normal distribution of the original PAN model with the mean value of our variational parameters: $p(\tau_{jk} \mid \tau_0, q_0) = \text{Norm}(\sum_i \sigma_{jk}(x_i, y_i), \frac{\nu^{(\beta)}}{\nu^{(\alpha)} - 1})$.

The expected reward function R should reflect our expected satisfaction at having observed τ_{jk}. Ideally, we would be able to compute model improvement directly by comparing true document labels y with our predicted labels $\hat{y}$ after adding the new observation: $\mathbb{E}_{p*(x)} \left[\max_{\hat{y}} r(y, \hat{y}) \right]$ where $r(y, \hat{y})$ is an internal reward function like label accuracy and $p^*(x)$ is the empirical distribution. In reality the true values y are unobservable, but we can expect over them using our posterior approximation $\tilde{q}$; thus $R_{\tilde{q}} = \mathbb{E}_{p*(x)} \left[\max_{\hat{y}} \mathbb{E}_{\tilde{q}(y)} \left[r(y, \hat{y}) \right] \right]$. With a label accuracy reward function the expected reward simplifies to $R_{\tilde{q}} = \sum_i \max_{\hat{y}} q(y_i = \hat{y}) = \sum_i \max_{\hat{y}} \nu_{i\hat{y}}^{(y)}$. For simplicity, we set the cost function $C_{\tilde{q}}$ to a constant, but leave it in the equations since future work should estimate and use annotation cost.

By default, Algorithm 1 jointly selects an annotator j and measurement k, but it could be used in other ways. Haertel et al. (2010) argue that in realistic scenarios the active learning algorithm typically cannot control when annotators are available but rather must respond to annotator requests for work. Algorithm 1

1: **function** MEASUREMENTSELECTION(τ_0)
2: $q_0 \leftarrow$ Inference(τ_0)
3: **while** more measurements are desired **do**
4: $j, k \leftarrow$ NextMeasurement(τ_0, q_0)
5: $\tau_0 \leftarrow \tau_0 \cup$ ObserveMeasurement(j, k)
6: $q_0 \leftarrow$ Inference(τ_0)
7: **return** q
8: **function** NEXTMEASUREMENT(τ_0, q_0)
9: **for** annotator j **do**
10: **for** measurement feature k **do**
11: draw t samples from $p(\tau_{jk}|\tau_0, q_0)$
12: **for** sampled τ_{jkt} **do**
13: $\tau_1 \leftarrow \tau_{jkt} \cup \tau_0$
14: $\tilde{q} \leftarrow$ Inference(τ_1)
15: $\mu_{\tau_{jkt}} \leftarrow R_{\tilde{q}}(\tau_1) - C_{\tilde{q}}(\tau_1)$
16: $\mu_{\tau_{jk}} \leftarrow \frac{1}{t} \sum_t \mu_{\tau_{jkt}}$
17: **return** $\operatorname{argmax}_{j,k} \mu_{\tau_{jk}}$

Algorithm 1: Active measurement selection algorithm for jointly selecting annotator j and annotation type k. The NEXTMEASUREMENT subroutine approximates expected utility.

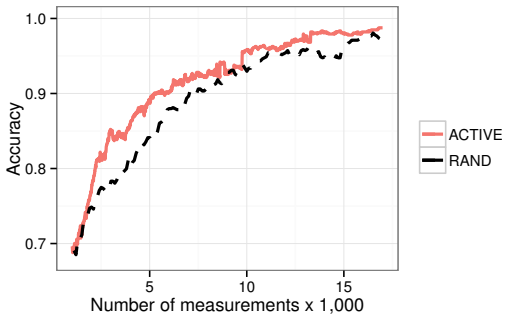


Figure 4: Inferred labeled accuracy of the PAN model selecting measurements randomly (*RAND*) compared with a strategy that selects measurements and annotators actively (*ACTIVE*).

can return the best measurement k given annotator $\hat{j}$ by computing $\operatorname{argmax}_k \mu_{\tau_{\hat{j}k}}$. Similarly, one can calculate the best annotator j for a desired measurement $\hat{k}$ as $\operatorname{argmax}_j \mu_{\tau_{j\hat{k}}}$.

Unfortunately, Algorithm 1 is computationally expensive. Prior to selection, each candidate measurement must be considered and a model retrained using t sampled candidate measurements. However, by applying a number of additional approximations we can run on the weather sentiment dataset from Section 5.4 with over 22,000 candidate measurements. We set the number of samples to three, and models trained in the inner loop (line 14 of Algorithm 1) are initialized using q_0 and then trained only one additional iteration. More importantly, we score only twenty-five randomly selected candidates at each round and select batches of the ten most promising measurements at each round.

We compare random and active selection of measurements (Algorithm 1) from the sentiment classification experiment in Section 5.4 in Figure 4. The active measurement selection improves over a random baseline.

7 Additional Related Work

Other supervised learning frameworks that incorporate rich prior knowledge include constraint-driven learning based on integer linear programming (Chang et al., 2008), generalized expectation criteria (Druck et al., 2008), and the posterior regularization (Ganchev et al., 2010). Ganchev et al. (2010) explain each of these three frameworks can be derived as a special case of the learning from measurements framework of Liang et al. (2009) by making particular approximations for the sake of tractability.

Traditional corpus construction assesses inferred label quality using annotator agreement heuristics such as Krippendorff's alpha (Krippendorff, 2012). Passonneau and Carpenter (2014) argue that inference in probabilistic models yields higher quality labels at lower cost, and should be preferred over agreement heuristics. Among crowdsourcing models, Hovy et al. (2013) include Bernoulli switching variables to identify and eliminate malicious contributors (spammers). Raykar and Yu (2012) iteratively

run inference and exclude problematic annotators in order to eliminate spammers. Raykar et al. (2010), Yan et al. (2014), and Felt et al. (2015a) model data jointly with labels, allowing patterns in the data to inform inferred labels. Simpson and Roberts (2015) model annotator dynamics, tracking the ways that annotator decision making changes over time in response to factors such as training and fatigue. Welinder et al. (2010) and Whitehill et al. (2009) both model item difficulty, reducing the effect of inherently ambiguous or difficult items on annotator reliability estimates.

Each of these crowdsourcing models focuses on incorporating one or more insights about the annotation process. We leave it to future work to incorporate these insights into crowdsourcing models that learn from measurements, either via measurement noise or via new measurement formulations. For example, item difficulty could be modeled by creating a measurement feature $\sigma_{jk}(x_i = x_{i'})$ for each $x_{i'}$ and assigning a separate measurement noise to each (perhaps with a shared hierarchical prior noise).

8 Conclusion and Future Work

The success of machine learning depends on data, and those data often come from human annotators. Asking the right questions of the right people is an often overlooked challenge of building an effective machine learning system. Extending the measurements framework allows modeling users' quirks and using their insights more effectively.

This paper makes a focused contribution by connecting the measurements framework with crowdsourcing models and using initial experiments to showcase the flexibility and promise of this connection. However, the measurements framework is far more general than this first round of experiments is able to show. One primary direction of follow-on work will be to extend crowdsourcing measurement models to more sophisticated structured prediction applications. Another will be to develop inference for a more robust noise model. For example, Gaussian Processes could be used as measurement noise priors to capture more complex interactions between measurement types and annotators.

As we move toward richer annotations, we also need to consider the implications for human interactions. Modeling the costs of annotations can prevent crowdsourcing from asking difficult, ambiguous, or annoying questions. Potentially interesting measurements may include allowing annotators to report their own reliability, to assess the reliability of other annotators, or to label locations in a semantically meaningful space rather than discrete words or documents.

Acknowledgments This work was supported by the collaborative NSF Grant IIS-1409739 (BYU) and IIS-1409287 (UMD). Boyd-Graber is also supported by NSF grant IIS-1652666. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

References

Ming-Wei Chang, Lev-Arie Ratinov, Nicholas Rizzolo, and Dan Roth. 2008. Learning and inference with constraints. In *Proc. Conference on Artificial Intelligence (AAAI)*.

Alexander P. Dawid and Allan M. Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*, pages 20–28.

Pinar Donmez and Jaime G. Carbonell. 2008. Proactive learning: Cost-sensitive active learning with multiple imperfect oracles. In *Proc. ACM Conference on Information and Knowledge Management*.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *Proc. International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Paul Felt, Eric K. Ringger, Jordan Boyd-Graber, and Kevin Seppi. 2015a. Making the most of crowdsourced document annotations: Confused supervised LDA. In *Proc. Conference on Computational Natural Language Learning (CoNLL)*.

Paul Felt, Eric K. Ringger, Kevin Seppi, and Robbie A. Haertel. 2015b. Early gains matter: A case for preferring generative over discriminative crowdsourcing models. In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 11:2001–2049.

Robbie A. Haertel, Paul Felt, Eric K. Ringger, and Kevin Seppi. 2010. Parallel active learning: Eliminating wait time with minimal staleness. In *Proc. HLT-NAACL 2010 Workshop on Active Learning for Natural Language Processing*.

Robbie A. Haertel. 2013. *Practical Cost-Conscious Active Learning for Data Annotation in Annotator-Initiated Environments*. Ph.D. thesis, Brigham Young University.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard H. Hovy. 2013. Learning whom to trust with MACE. In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

David Jurgens. 2013. Embracing ambiguity: A comparison of annotation methodologies for crowdsourcing word sense labels. In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Klaus Krippendorff. 2012. *Content Analysis: An Introduction to its Methodology*. Sage.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning from measurements in exponential families. In *Proc. International Conference on Machine Learning (ICML)*.

Andrew McCallum. 2002. MALLET: A machine learning for language toolkit. http://mallet.cs.umass.edu.

An T. Nguyen, Byron C. Wallace, and Matthew Lease. 2015. Combining crowd and expert labels using decision theoretic active learning. In *Proc. 3rd AAAI Conference on Human Computation (HCOMP)*.

Rebecca J. Passonneau and Bob Carpenter. 2014. The benefits of a model of annotation. *Transactions of the Association for Computational Linguistics*, 2:311–326.

Vikas C. Raykar and Shipeng Yu. 2012. Eliminating spammers and ranking annotators for crowdsourced labeling tasks. *The Journal of Machine Learning Research*, 13:491–518.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *The Journal of Machine Learning Research*, 11:1297–1322.

Burr Settles. 2010. Active learning literature survey. *University of Wisconsin, Madison*.

Aashish Sheshadri and Matthew Lease. 2013. Square: A benchmark for research on computing crowd consensus. In *Proc. Conference on Human Computation and Crowdsourcing (HCOMP)*.

Edwin Simpson and Stephen Roberts. 2015. Bayesian methods for intelligent task assignment in crowdsourcing systems. In *Decision Making: Uncertainty, Imperfection, Deliberation and Scalability*, pages 1–32. Springer.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

James Surowiecki. 2005. *The Wisdom of Crowds*. Random House LLC.

Peter Welinder, Steve Branson, Pietro Perona, and Serge J. Belongie. 2010. The multidimensional wisdom of crowds. In *Proc. Advances in Neural Information Processing Systems (NIPS)*.

Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L. Ruvolo. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Proc. Advances in Neural Information Processing Systems (NIPS)*.

Yan Yan, Glenn M. Fung, Rómer Rosales, and Jennifer G. Dy. 2011. Active learning from crowds. In *Proc. International Conference on Machine Learning (ICML)*.

Yan Yan, Rómer Rosales, Glenn Fung, Ramanathan Subramanian, and Jennifer Dy. 2014. Learning from multiple annotators with varying expertise. *Machine Learning*, 95(3):291–327.

Reproducing and Regularizing the SCRN Model

Olzhas Kabdolov, Zhenisbek Assylbekov, Rustem Takhanov
School of Science and Technology, Nazarbayev University
{olzhas.kabdolov, zhassylbekov, rustem.takhanov}@nu.edu.kz

Abstract

We reproduce the Structurally Constrained Recurrent Network (SCRN) model, and then regularize it using the existing widespread techniques, such as naïve dropout, variational dropout, and weight tying. We show that when regularized and optimized appropriately the SCRN model can achieve performance comparable with the ubiquitous LSTM model in language modeling task on English data, while outperforming it on non-English data.

Title and Abstract in Russian

Воспроизведение и регуляризация SCRN модели

Мы воспроизводим структурно ограниченную рекуррентную сеть (SCRN), а затем добавляем регуляризацию, используя существующие широко распространенные методы, такие как исключение (дропаут), вариационное исключение и связка параметров. Мы показываем, что при правильной регуляризации и оптимизации показатели SCRN сопоставимы с показателями вездесущей LSTM в задаче языкового моделирования на английских текстах, а также превосходят их на неанглийских данных.

1 Introduction

Recurrent neural networks (RNN) have demonstrated tremendous success in sequence modeling in general and in language modeling in particular. The most basic RNN (Elman, 1990) suffers from the problem of vanishing and exploding gradients (Bengio et al., 1994) and is hard to train efficiently. One of the most widespread and efficient alternatives to the basic RNN is the Long-Short Term Memory (LSTM) model (Hochreiter and Schmidhuber, 1997), which effectively addresses the problem of vanishing gradients. However, LSTM is a fairly complex model with excessive number of parameters and its inner functionality is not obvious. This complexity has motivated some of the researchers to find more apparent and less complex alternatives. One of such alternative models is a Structurally Constrained Recurrent Network (SCRN) proposed by Mikolov et al. (2015). They encouraged some of the hidden units to change their state slowly by making part of the recurrent weight matrix close to identity, thus forming a kind of longer term memory and showed that their SCRN model can outperform the simple RNN and achieve the performance comparable with the LSTM under no regularization and small parameter budget. It is natural to try to regularize the SCRN model under larger budgets: *Will it approach the performance of LSTM?* Our experiments show that (1) under naïve dropout the SCRN demonstrates performance close to that of the LSTM, but (2) under variational dropout and weight tying the LSTM demonstrates better performance.

2 Related Work

There has been several attempts on simplifying the ubiquitous LSTM model while not losing in performance. E.g., Ororbia II et al. (2017) introduced Delta-RNN architecture for which SCRN serves as a

Proceedings of the 27th International Conference on Computational Linguistics, pages 1705–1716
Santa Fe, New Mexico, USA, August 20-26, 2018.

predecessor. They showed that, when regularized using naïve dropout (Zaremba et al., 2014), Delta-RNN performs comparably to LSTM and GRU (Chung et al., 2014). However, they did not compare to regularized SCRN, and they did not consider more recent regularization techniques, such as variational dropout (Gal and Ghahramani, 2016) and weight tying (Inan et al., 2017; Press and Wolf, 2017).

Lei and Zhang (2017) proposed the Simple Recurrent Unit (SRU) architecture, a recurrent unit that simplifies the computation and exposes more parallelism. In SRU, the majority of computation for each step is independent of the recurrence and can be easily parallelized. SRU is as fast as a convolutional layer and 5–10x faster than an optimized LSTM implementation. The authors study SRUs on a wide range of applications, including classification, question answering, language modeling, translation and speech recognition. However, one important thing which needs to be mentioned is that this model also exploits the idea of highway connections (Zilly et al., 2017) letting the input directly flow into the hidden state. The use of highway connections could be the main reason why the model achieves high performance, especially in a multi-stacked setting.

Lee et al. (2017) introduced Recurrent Additive Network (RAN), a new gated RNN which is distinguished by the use of purely additive latent state updates. At every time step, the new state is computed as a gated component-wise sum of the input and the previous state, without any of the non-linearities commonly used in RNN transition dynamics. The authors show that the model performs on par with the LSTM, and claim that it has significantly less parameters. However, in their language modeling experiments the authors specify only the number of parameters in the recurrent units, and do not take into account parameters of the embedding and softmax layers, which actually make up most of the language model parameters.

3 Baseline SCRN Model

Let $\mathcal{W}$ be a finite vocabulary of words. We assume that words have already been converted into indices. Based on one-hot word embeddings $\mathbf{x_{1:k}} = \mathbf{x_1}, \ldots, \mathbf{x_k}$ for a sequence of words $w_{1:k}$, the baseline SCRN model (Mikolov et al., 2015) produces two sequences of states, $\mathbf{s_{1:k}}$ and $\mathbf{h_{1:k}}$, according to[1]

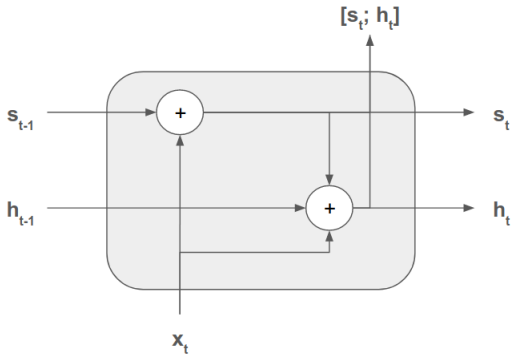


Figure 1: SCRN cell.

$$\mathbf{s_t} = (1-\alpha)\mathbf{x_t}\mathbf{B} + \alpha\mathbf{s_{t-1}}, \tag{1}$$

$$\mathbf{h_t} = \sigma(\mathbf{x_t}\mathbf{A} + \mathbf{s_t}\mathbf{P} + \mathbf{h_{t-1}}\mathbf{R}), \tag{2}$$

where $\mathbf{B} \in \mathbb{R}^{|\mathcal{W}| \times d_s}$, $\mathbf{A} \in \mathbb{R}^{|\mathcal{W}| \times d_h}$, $\mathbf{P} \in \mathbb{R}^{d_s \times d_h}$, $\mathbf{R} \in \mathbb{R}^{d_h \times d_h}$, d_s and d_h are dimensions of $\mathbf{s_t}$ and $\mathbf{h_t}$, $\sigma(\cdot)$ is the logistic sigmoid function. Mikolov et al. (2015) refer to $\mathbf{s_t}$ as a slowly changing *context state*, and to $\mathbf{h_t}$ as a quickly changing *hidden state*. The last couple of states $(\mathbf{s_k}, \mathbf{h_k})$ is assumed to contain information on the whole sequence $w_{1:k}$ and is further used for predicting the next word w_{k+1} of a sequence according to the probability distribution

$$\Pr(w_{k+1}|w_{1:k}) = \text{softmax}(\mathbf{s_k}\mathbf{U} + \mathbf{h_k}\mathbf{V}), \tag{3}$$

where $\mathbf{U} \in \mathbb{R}^{d_s \times |\mathcal{W}|}$ and $\mathbf{V} \in \mathbb{R}^{d_h \times |\mathcal{W}|}$ are output embedding matrices. For the sake of simplicity we omit bias terms in (2) and (3).

Training the model involves minimizing the negative log-likelihood over the corpus $w_{1:K}$:

$$-\sum_{k=1}^{K} \log \Pr(w_k|w_{1:k-1}) \longrightarrow \min_{\Theta}, \tag{4}$$

which is usually done by truncated backpropagation through time (Werbos, 1990). Here Θ denotes the set of all model parameters.

[1] Vectors are assumed to be row vectors, which are right multiplied by matrices ($\mathbf{xW} + \mathbf{b}$). This choice is somewhat non-standard but it maps better to the way networks are implemented in code using matrix libraries such as `TensorFlow`.

Notice that SCRN is a slight modification of the vanilla RNN model (Elman, 1990), and its simplicity (see Fig. 1) is in stark contrast with the complexity of the widespread LSTM model.

Dense embeddings: The original model spends

$$2 \cdot |\mathcal{W}| \cdot (d_s + d_h) \tag{5}$$

parameters to embed words into dense vectors at input and at output. We believe that it is more beneficial to first embed words w_t into dense vectors $\mathbf{w_t} = \mathbf{x_t}\mathbf{E} \in \mathbb{R}^{d_h}$ using only one embedding matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{W}| \times d_h}$ and then use $\mathbf{w_t}$ instead of $\mathbf{x_t}$ in (1) and (2) with the appropriate change of shapes for matrices: $\mathbf{B} \in \mathbb{R}^{d_h \times d_s}$ and $\mathbf{A} \in \mathbb{R}^{d_h \times d_h}$. In this case, the model spends $|\mathcal{W}| \cdot (2d_h + d_s)$ parameters on input/output embeddings, which is $d_s \cdot |\mathcal{W}|$ parameters less than (5). E.g., on the Penn Tree Bank (PTB) dataset (Marcus et al., 1993), where $|\mathcal{W}| = 10,000$, the reduction is 1M parameters for the SCRN model with $d_s = 100$.

4 Stacking and Regularizing the SCRN

4.1 Stacking recurrent cells

It is well known that stacking at least two layers in recurrent neural networks is beneficial, but between two and three layers the results are mixed (Karpathy et al., 2016; Laurent and von Brecht, 2017). To make our results comparable to the previous works on LSTM language modeling (Zaremba et al., 2014; Gal and Ghahramani, 2016; Inan et al., 2017), we experiment with two-layered architectures: the output of the first layer (1, 2) is concatenated $[\mathbf{s_t}; \mathbf{h_t}]$ and is fed as input into the second layer.

In what follows the superscript index in round brackets denotes the layer index, $\boldsymbol{\xi}(p) = [\xi_1, \ldots, \xi_d]$ is a random vector (dropout mask) with $\xi_i \sim \text{Bernoulli}(1-p)$, d is the dimensionality of the corresponding layer, p is a dropout rate, and $\odot$ is the element-wise (Hadamard) product. One time-step of a stacked SCRN model is fully specified by the following equations:

- Input embedding layer:

$$\mathbf{w_t} := \mathbf{y_t}^{(0)} = \mathbf{x_t}\mathbf{E}. \tag{6}$$

- Two SCRN layers: for $l = 1, 2$

$$\begin{aligned}
\mathbf{s_t}^{(l)} &= (1-\alpha)\mathbf{y_t}^{(l-1)}\mathbf{B}^{(l)} + \alpha\mathbf{s_{t-1}}^{(l)}, \\
\mathbf{h_t}^{(l)} &= \sigma\left(\mathbf{y_t}^{(l-1)}\mathbf{A}^{(l)} + \mathbf{s_t}^{(l)}\mathbf{P}^{(l)} + \mathbf{h_{t-1}}^{(l)}\mathbf{R}^{(l)}\right), \\
\mathbf{y_t}^{(l)} &= [\mathbf{s_t}^{(l)}; \mathbf{h_t}^{(l)}].
\end{aligned} \tag{7}$$

- Softmax prediction:

$$\Pr(w_{t+1}|w_{1:t}) = \text{softmax}\left(\mathbf{y_t}^{(2)}\begin{bmatrix}\mathbf{U} \\ \mathbf{V}\end{bmatrix}\right). \tag{8}$$

In the equations above, the matrices $\mathbf{U}$ and $\mathbf{V}$ are concatenated along the first dimension to produce a $(d_s + d_h) \times |\mathcal{W}|$ matrix.

4.2 Naïve dropout

Due to high complexity of deep neural networks the regularization techniques are crucial for good generalization performance. Zaremba et al. (2014) proposed one of the ways how dropout (Srivastava et al., 2014) can be used to regularize recurrent neural networks. They apply dropout only to non-recurrent connections, while keeping the recurrent connections without change. This method, usually referred to as *naïve dropout*, improves the baseline results of the LSTM model without any other modifications. Application of the same dropout technique to the SCRN model (6, 7, 8) is specified by

$$\begin{aligned}
\hat{\mathbf{y}}_\mathbf{t}^{(0)} &= \mathbf{y_t}^{(0)} \odot \boldsymbol{\xi_t}^{(0)}(p_i), \\
\hat{\mathbf{y}}_\mathbf{t}^{(l)} &= \mathbf{y_t}^{(l)} \odot \boldsymbol{\xi_t}^{(l)}(p_o), \quad l = 1, 2
\end{aligned}$$

correspondingly, where p_i and p_o are input and output dropout rates.

4.3 Variational dropout

The approach of Zaremba et al. (2014) have led many to believe that dropout cannot be extended to recurrent connections, leaving them with no regularization. However, Gal and Ghahramani (2016) showed that it is possible to derive a variant of dropout which successfully regularizes recurrent connections. In their dropout variant, which is usually referred to as *variational dropout*, they repeat the same dropout mask at each time step for inputs, recurrent layers, and outputs:

$$\begin{aligned}
\tilde{\mathbf{y}}_\mathbf{t}^{(0)} &= \mathbf{y}_\mathbf{t}^{(0)} \odot \boldsymbol{\xi}^{(0)}(p_i), \\
\hbar_\mathbf{t}^{(l)} &= \mathbf{h}_\mathbf{t}^{(l)} \odot \boldsymbol{\xi}^{(l)}(p_h), \quad l = 1, 2 \\
\tilde{\mathbf{y}}_\mathbf{t}^{(l)} &= \mathbf{y}_\mathbf{t}^{(l)} \odot \boldsymbol{\xi}^{(l)}(p_o), \quad l = 1, 2
\end{aligned} \tag{9}$$

where p_h is a recurrent dropout rate. This is in contrast to the naïve dropout where different masks are sampled at each time step for the inputs and outputs alone and no dropout is used with the recurrent connections. Notice that we do not regularize the context state $\mathbf{s_t}$ in horizontal (recurrent) direction.

4.4 Tying word embeddings

Tying input and output word embeddings in word-level RNNLM is a regularization technique, which was introduced earlier (Bengio et al., 2003; Mnih and Hinton, 2007) but has been widely used relatively recently, and there is empirical evidence (Press and Wolf, 2017) as well as theoretical justification (Inan et al., 2017) that such a simple trick improves language modeling quality while decreasing the total number of trainable parameters almost two-fold, since most of the parameters are due to embedding matrices. In case of the SCRN model, reusing input embeddings at output can be done by setting

$$\mathbf{V} = \mathbf{E}^\top$$

in the softmax layer (8).

5 Experimental setup

We use perplexity (PPL) to evaluate the performance of the language models. Perplexity of a model over a sequence $[w_1, \ldots, w_K]$ is given by

$$\text{PPL} = \exp\left(-\tfrac{1}{K}\sum_{k=1}^{K} \log \Pr(w_k | w_{1:k-1})\right).$$

Data sets: The baseline model is trained and evaluated on the PTB (Marcus et al., 1993), while all regularized and stacked configurations are trained and evaluated on the PTB and the WikiText-2 (Merity et al., 2017) data sets. For the PTB we utilize the standard training (0-20), validation (21-22), and test (23-24) splits along with pre-processing per Mikolov et al. (2010). WikiText-2 is an alternative to PTB, which is approximately two times as large in size and three times as large in vocabulary.

Baseline Model: To reproduce the results of the baseline (single-layer and non-regularized) SCRN model we use the original `Torch` implementation[2] released by Mikolov et al. (2015). We have spent fair amount of time and effort to make their script run, as several of its dependencies have not been updated for few years, and are not compatible with the up-to-date versions of the others. To simplify the path for other researchers we release a script[3], which installs necessary versions of the dependencies. We use exactly the same set of hyperparameters reported in the original paper (Table 1). We also implement the baseline SCRN model ourselves[4] using `TensorFlow` (Abadi et al., 2016) which, unlike `Torch`, uses static computational graphs, and thus has different style of truncated backpropagation through time[5] (BPTT). Because of this difference, we chose different set of hyperparameters for our implementation (Table 1),

[2] `https://github.com/facebookarchive/SCRNNs`

[3] `https://github.com/zh3nis/scrn/blob/master/scrnn_deps.sh`

[4] Our implementation is available at `https://github.com/zh3nis/scrn`

[5] `https://r2rt.com/styles-of-truncated-backpropagation.html`

Hyperparameter	Mikolov et al. (2015)	Our implementation
α in (1)	0.95	0.95
batch size	32	20
initial LR	0.05	0.8
LR decay	1/1.5	0.5
LR decayed if	valid PPL doesn't improve	valid PPL doesn't improve
BPTT steps	50	35
BPTT frequency	5	35
gradients	renormalized	norms clipped at 5
weights initialized over	$[-0.05, 0.05]$	$[-0.3, 0.3]$ ($[-0.2, 0.2]$)

Table 1: Hyperparameters of the baseline SCRN model. Abbreviations: LR — learning rate, BPTT — backpropagation through time. Values in brackets correspond to the $(d_h, d_s) = (300, 40)$ configuration when they differ from others.

and this choice is motivated by the previous work on word-level language modeling (Zaremba et al., 2014), which has an open-source implementation in `TensorFlow`[6].

Stacked and Regularized Models: In the previous works on regularizing the LSTM, small-sized models usually had $d_h = 200$ and medium-sized models had $d_h = 650$. The inner simplicity of the SCRN cell allows us slighlty larger hidden sizes: we use $d_h = 240$ for small models and $d_h = 750$ for medium models. Context state sizes d_s are chosen to be 40 (small) and 120 (medium), so that total number of parameters does not exceed the budget, which is 5M parameters for small models, and 20M parameters for medium models. We find empirically, that a good ratio between context size and hidden size in the SCRN model is around $1/6$. We optimize hyperparameters separately under naïve dropout and under variational dropout. Some of the hyperparameters are tuned using random search according to the marginal distributions:

- $p_i \sim U[0.01, 0.6]$,
- $p_h \sim U[0.01, 0.6]$,
- $p_o \sim U[0.01, 0.6]$,
- initial learning rate $\sim U[0.5, 0.99]$,
- learning rate decay $\sim U[0.5, 0.89]$,
- initialization scale $\sim U[0.05, 0.3]$.

where $U[a, b]$ means continuous uniform distribution over the interval $[a, b]$, and initialization scale is a number r such that all model weights are initialized uniformly over $[-r, r]$. Other hyperparameters are tuned manually through trial-and-error. When performing random search we first choose ranges mentioned above. After 100 runs the initial ranges are shrinked to the neighborhoods of the values that give best performances, and the random search is performed again. We repeat this procedure until hyperparameters converge to their (sub)optimal values. To prevent exploding gradients we clip the norm of the gradients (normalized by minibatch size) at 5. For training (4) we use stochastic gradient descent.

6 Results

Baseline: To assure that our implementation of the baseline SCRN is adequate, we evaluate it against the original SCRN code by Mikolov et al. (2015) (Table 2). As one can see, the original SCRN code does *not* fully reproduce the results reported in the original paper. Their hyperparameters (Table 1) work well for the case when $(d_s, d_h) \in \{(40, 10), (90, 10)\}$, but are not optimal for the other two configurations. Our implementation together with our set of hyperparameters (Table 1) brings the validation and test perplexities of *all* the configurations closer to those reported in the paper.

Stacked and Regularized Models: Tuning the stack of two SCRNs results in hyperparameters in Table 3. The results of evaluating these models against regularized and stacked LSTMs on PTB and WikiText-2 are provided in Table 4. Regularization *does* benefit the simple and intuitive SCRN model, which

[6] `https://github.com/tensorflow/models/tree/master/tutorials/rnn/ptb`

Hidden size	Context size	Mikolov et al. (2015) Valid PPL	Mikolov et al. (2015) Test PPL	Our implementation Valid PPL	Our implementation Test PPL
40	10	133.6 (133)	127.5 (127)	134.5	128.0
90	10	125.4 (124)	120.3 (119)	124.9	118.6
100	40	127.6 (120)	122.9 (115)	124.9	118.7
300	40	130.1 (120)	124.4 (115)	127.2	120.6

Table 2: Reproducing the baseline model on PTB data. For the original implementation (columns 3 and 4), values outside brackets were obtained when running the script from `https://github.com/facebookarchive/SCRNNs`, and values in brackets were reported in the paper of Mikolov et al. (2015).

Hyperparameter	SCRN + ND		SCRN + VD	
	Small	Medium	Small	Medium
p_i	0.2 (0.15)	0.55 (0.45)	0.15 (0.1)	0.4 (0.3)
p_h	—	—	0.15 (0.1)	0.4 (0.3)
p_o	0.2 (0.15)	0.55 (0.45)	0.15 (0.1)	0.4 (0.3)
α	0.9	0.9	0.9	0.9
initial LR	0.8	0.8	0.8	0.6
LR decay	0.5	0.65	0.87	0.9
LR decayed	when valid PPL does not improve		after 10 epochs	after 25 epochs
initialization scale	0.3	0.3	0.3	0.3

Table 3: Tuned hyperparameters: ND – naïve dropout, VD – variational dropout. Values in brackets correspond to the WikiText-2 in cases when they differ from those used for the PTB.

demonstrates performance comparable to the sophisticated LSTM model under the naïve dropout, but lags behind the LSTM under the variational dropout regularization. This shows that at some point less complex models become less competitive, even when regularized and optimized appropriately. It is important to mention that no architectural modifications were applied to the original SCRN model except stacking.

Our feeling is that a little over-parameterization of the recurrent connections model *is* needed for the recent regularization techniques to work well. For example, consider the variational dropout of the recurrent connections (9) in the SCRN: dropping the coordinates $i_1, \ldots, i_l$ in the row $\mathbf{h_t}$ is equivalent to zeroing out the rows $i_1, \ldots, i_l$ in the matrices $\mathbf{A}$, $\mathbf{P}$, $\mathbf{R}$. Now consider one layer of the LSTM model:

$$\begin{aligned}
\mathbf{f_t} &= \sigma(\mathbf{x_t W_f} + \mathbf{h_{t-1} U_f} + \mathbf{b_f}) \\
\mathbf{i_t} &= \sigma(\mathbf{x_t W_i} + \mathbf{h_{t-1} U_i} + \mathbf{b_i}) \\
\mathbf{o_t} &= \sigma(\mathbf{x_t W_o} + \mathbf{h_{t-1} U_o} + \mathbf{b_o}) \\
\mathbf{c_t} &= \mathbf{f_t} \odot \mathbf{c_{t-1}} + \mathbf{i_t} \odot \tanh(\mathbf{x_t W_c} + \mathbf{h_{t-1} U_c} + \mathbf{b_c}) \\
\mathbf{h_t} &= \mathbf{o_t} \odot \tanh(\mathbf{c_t})
\end{aligned}$$

When one drops the coordinates $i_1, \ldots, i_l$ of $\mathbf{h_t}$ in LSTM, this can be understood as zeroing out the rows $i_1, \ldots, i_l$ in the matrices $\mathbf{W_f}$, $\mathbf{W_i}$, $\mathbf{W_o}$, $\mathbf{U_f}$, $\mathbf{U_i}$, $\mathbf{U_o}$, i.e. around 3 times more parameters are zeroed out given the same hidden state size d_h. In other words, the number of recurrent weights is 3 times larger in LSTM than in SCRN. We believe this is one of the reasons why the variational dropout works worse in SCRN. Roughly speaking, there is nothing much to regularize in the horizontal (recurrent) direction in SCRN, as most parameters are in the embedding and softmax layers.

6.1 Ablation analysis

We consider removal of some parts or forms of regularization from one of our well-performing configurations, SCRN + ND + WT, to see whether such removal degrades the performance. It also tells us which

Word-level model	PTB		WikiText-2	
	Small	Medium	Small	Medium
LSTM + ND (Jozefowicz et al., 2015)	—	**79.8**	—	—
LSTM + ND (Kim et al., 2016)	97.6	85.4	116.8†	—
LSTM + ND (Zaremba et al., 2014)	—	82.7	—	**96.2**‡
SCRN + ND	95.8	85.6	115.0	100.8
SCRN + ND + WT	**94.1**	86.1	**112.0**	98.5
LSTM + VD (Gal and Ghahramani, 2016)	—	78.6	—	—
LSTM + VD (Inan et al., 2017)	87.3	77.7	105.9	95.3
LSTM + VD + WT (Inan et al., 2017)	**85.1**	**73.9**	**100.5**	**87.7**
SCRN + VD	97.2	90.7	120.1	107.6
SCRN + VD + WT	96.8	90.7	112.2	106.0

Table 4: Evaluation of the SCRN against LSTM under different regularization techniques: ND – naïve dropout, VD – variational dropout, WT – weight tying. †We reproduced the LSTM-Word-Small model from Kim et al. (2016) on PTB and then evaluated it on WikiText-2. ‡We ran the open-source implementation from `https://github.com/tensorflow/models/tree/master/tutorials/rnn/ptb` at medium config on WikiText-2 data.

parts of the model are more important. The model and dropout variations are listed below.

Removing regularization: To understand how much improvement is gained by the use of regularization we completely remove dropout and weight tying:

$$p_i = p_o = 0, \qquad \mathbf{V} \neq \mathbf{E}^\top$$

Removing dropout of the context state: According to (1), context state $\mathbf{s_t}$ changes linearly and thus should not suffer from over-fitting. Thus it seems reasonable to try to not regularize it and apply dropout only to the hidden states, i.e. replacing the equation (7) by

$$\mathbf{y_t}^{(l)} = [\mathbf{s_t}^{(l)}; \mathbf{h_t}^{(l)} \odot \boldsymbol{\xi}_\mathbf{t}^{(l)}(p_o)], \qquad l = 1, 2.$$

Removing the context state from the softmax layer: As in the case of the SCRN, the inner state of the LSTM model also consists of two vectors $\mathbf{c_t}$ and $\mathbf{h_t}$, and usually the state $\mathbf{c_t}$ is not used at softmax. We do the same for the context state $\mathbf{s_t}$ in our model, i.e. the equation (8) is replaced by

$$\Pr(w_{t+1}|w_{1:t}) = \text{softmax}(\mathbf{h_t}^{(2)}\mathbf{V}).$$

The meaningfulness of removing the context state from the softmax is that, in our opinion, it plays the role of a long-term memory and thus should not be crucial for predicting the next word of a sequence. Moreover, such removal reduces the model size by at least $d_s \cdot |\mathcal{W}|$ parameters, which can be significant (see Section 3).

The results of the ablation analysis are provided in Table 5. As we can see, without regularization our SCRN + ND + WT model fails to generalize well on validation and test sets. Regularizing only the hidden state (and keeping the context state untouched) is less harmful but still degrades the performance of the model. Finally, not using the context state in the output only slightly worsens the performance but at the same time leads to a significant reduction in model size.

6.2 Hidden state changes

We performed an analysis of the SCRN's hidden state evolution as in the work of Ororbia II et al. (2017) on Delta-RNN (see their Figure 2). We found out that the hidden state changes in SCRN are similar to those in Delta-RNN: the L_1-norm of the state change is higher for informative words, such as "government", and the difference is in general more pronounced than in LSTM. See the Figure 2.

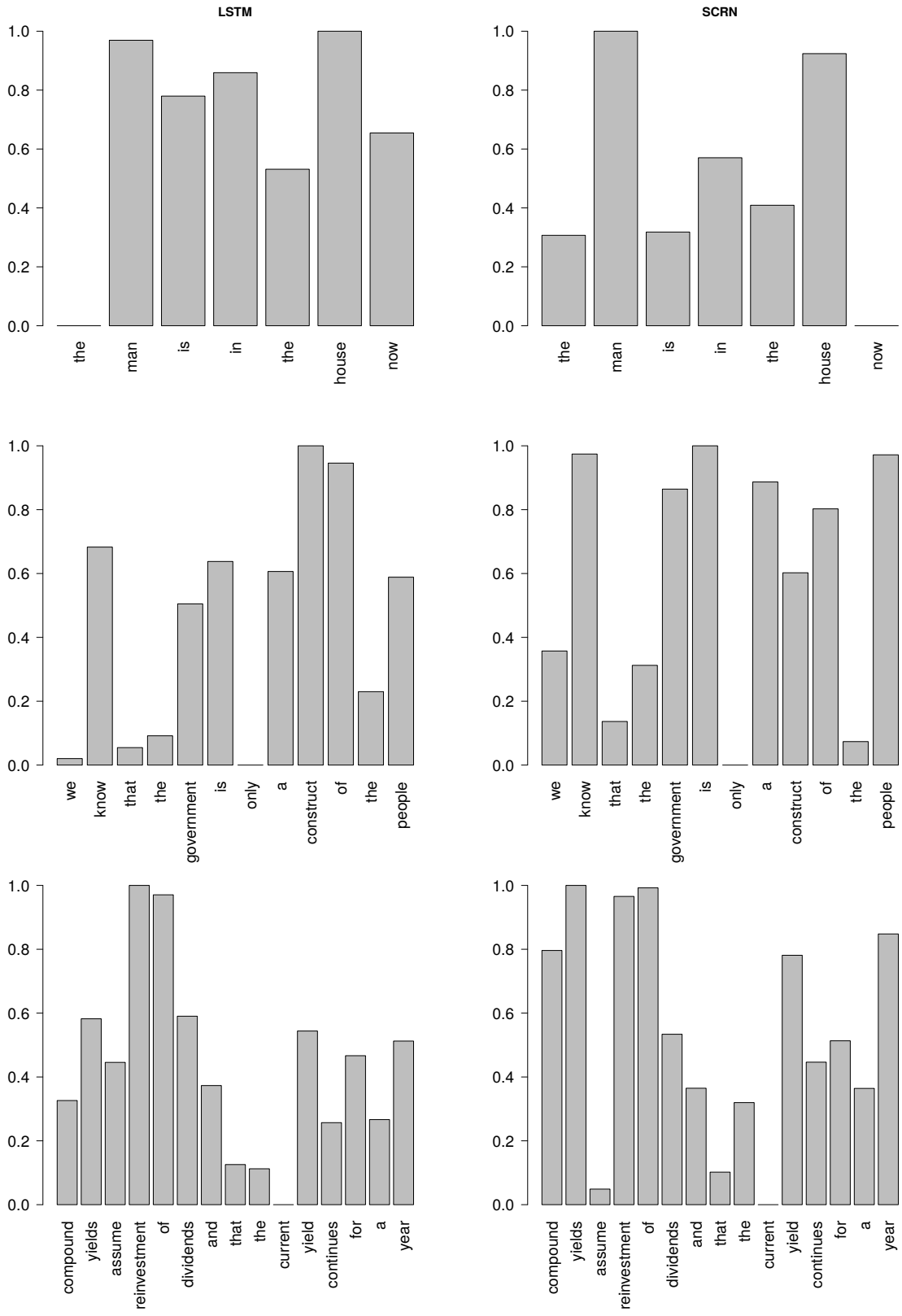


Figure 2: L_1 norm of deltas between consecutive states of LSTM (left) and SCRN (right) trained on Penn Treebank plotted over words of example sentences. The main observation is that the norm is in general lower for low-information content words, such as the article *the*, and higher for informative words, such as *government*, and this difference is more pronounced in SCRN.

Model	Small		Medium	
	Valid	Test	Valid	Test
SCRN + ND + WT	98.2	94.1	90.1	86.1
− regularization	123.3	117.6	146.1	140.2
− dropout of context	108.4	103.6	115.0	109.4
− context in softmax	100.8	96.4	91.7	87.6

Table 5: Model ablations for the small and medium SCRN + ND + WT models on PTB set.

7 Sobolev Regularization

Together with dropout and weight tying techniques we conducted some experiments with a Sobolev-type regularization. By the Sobolev-type regularization we understand penalization of large gradients of certain parts of neural architecture treated as functions. The idea of penalizing norm of gradients can be traced back to the works on double backpropagation (Drucker and Le Cun, 1992). Zilly et al. (2017) demonstrated that an important property of the hidden vector's dynamics is its stability, which they decribe in terms of an upper bound on norm of Jacobian matrix $\frac{\partial \mathbf{h_t}}{\partial \mathbf{h_{t-1}}}$. Exploiting the latter intuition we suggest the following regularization term:

$$\mathcal{L}_{\text{Sobolev}} = \beta \sum_{t=1}^{K} \left\| \frac{\partial \mathbf{h_t}}{\partial \mathbf{h_{t-1}}} \right\|_F^2, \tag{10}$$

where $\|\mathbf{A}\|_F = \sqrt{\text{Tr}(\mathbf{A}^\top \mathbf{A})}$ is the Frobenius norm of a matrix $\mathbf{A}$. Note that we do not specify the layer, since our regularization can be applied to all layers, as well as to certain layers only. The strength of such penalty is controlled by the hyperparameter β.

Overall, our experimental results are consistent with conclusions to which we came for variational dropout, with a perplexity gain even smaller than in the latter case. A typical gain from the Sobolev-type regularization was up to 1 perplexity point when it was applied on top of naïve or variational dropout, and up to 7 perplexity points when it was the only regularization used. Based on those results, the conclusion from Section 6 can be reinforced: probably, the main weakness of SCRN model, in comparison with popular RNNs for which the regularization of recurrent connections is effective, is under-parameterization of those connections. The latter "does not give a space" for such regularization techniques. For completeness of narrative, let us give some details of implementing the Sobolev term in `TensorFlow`.

Since $\mathbf{h_t}$ is a row, then $\frac{\partial \mathbf{h_t}}{\partial \mathbf{h_{t-1}}}$ is understood as $\left[\frac{\partial h_t^j}{\partial h_{t-1}^i}\right]$ (i.e. transposed Jacobian $\frac{\partial \mathbf{h_t}^\top}{\partial \mathbf{h_{t-1}}^\top}$). According to (2), we have $\mathbf{h_t} = \sigma(\mathbf{x_t}\mathbf{A} + \mathbf{s_t}\mathbf{P} + \mathbf{h_{t-1}}\mathbf{R})$. After an application of the matrix chain rule, we obtain:

$$\frac{\partial \mathbf{h_t}}{\partial \mathbf{h_{t-1}}} = \mathbf{R} \cdot \text{diag}(\sigma'(\mathbf{a_t} + \mathbf{h_{t-1}}\mathbf{R})),$$

where $\mathbf{a_t} = \mathbf{x_t}\mathbf{A} + \mathbf{s_t}\mathbf{P}$ and $\text{diag}(\mathbf{v})$ is a diagonal matrix with its diagonal consisting of the components of a vector $\mathbf{v}$. Further, we have:

$$\left\| \frac{\partial \mathbf{h_t}}{\partial \mathbf{h_{t-1}}} \right\|_F^2 = \text{Tr}\left(\frac{\partial \mathbf{h_t}}{\partial \mathbf{h_{t-1}}}^\top \frac{\partial \mathbf{h_t}}{\partial \mathbf{h_{t-1}}} \right) = \text{Tr}\left(\text{diag}(\sigma'(\mathbf{a_t} + \mathbf{h_{t-1}}\mathbf{R})) \cdot \mathbf{R}^\top \mathbf{R} \cdot \text{diag}(\sigma'(\mathbf{a_t} + \mathbf{h_{t-1}}\mathbf{R})) \right).$$

Using the fact that trace is invariant w.r.t. to the transform $\mathbf{A} \to \mathbf{S}^{-1}\mathbf{A}\mathbf{S}$, we can set $\mathbf{S} = \text{diag}(\sigma'(\mathbf{a_t} + \mathbf{h_{t-1}}\mathbf{R}))$ and obtain the final formula:

$$\left\| \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} \right\|_F^2 = \text{Tr}(\mathbf{R}^\top \mathbf{R} \cdot \text{diag}(\sigma'(\mathbf{a_t} + \mathbf{h_{t-1}}\mathbf{R}))^2) = (\sigma')^2(\mathbf{a_t} + \mathbf{h_{t-1}}\mathbf{R})\mathbf{v}(\mathbf{R}),$$

where for $\mathbf{R} = [\mathbf{r_1}, \ldots, \mathbf{r_{d_h}}]$, $\mathbf{v}(\mathbf{R})$ is defined as the column $\left[\|\mathbf{r_1}\|^2, \ldots, \|\mathbf{r_{d_h}}\|^2\right]^\top$. Finally, the Sobolev term (10) can be given as

$$\mathcal{L}_{\text{Sobolev}} = \beta \left[\sum_{t=1}^{K} (\sigma')^2 (\mathbf{a_t} + \mathbf{h_{t-1}}\mathbf{R}) \right] \mathbf{v}(\mathbf{R}),$$

which is a form of a loss function that is suitable for an efficient implementation in `TensorFlow` (we also used the standard trick that $\sigma'(x) = \sigma(x)(1 - \sigma(x))$).

8 Performance on non-English data

It is interesting to see the performance of SCRN on texts written in non-English languages. For this purpose we conduct evaluation of the medium-sized versions of LSTM + VD + WT against SCRN + ND + WT on non-English data which comes from the 2013 ACL Workshop on Machine translation[7] with pre-processing per Botha and Blunsom (2014). Hyperparameters tuned on Wikitext-2 (Table 3) were used for all languages and we did not perform any language-specific tuning. Corpora statistics and the results of evaluation are provided in Table 6. Surprisingly, the SCRN model performs very well (compared to

	French	Spanish	German	Czech	Russian
Number of tokens	1M	1M	1M	1M	1M
Vocabulary size	25K	27K	37K	46K	62K
LSTM + ND (Kim et al., 2016)	222	200	286	493	357
LSTM + VD + WT	205	193	277	488	351
SCRN + ND + WT	**199**	**179**	**258**	**420**	**306**

Table 6: Evaluation of medium-sized models on non-English data.

LSTM) when it comes to modeling morphologically rich languages. Also, it is noteworthy that the higher the type-token ratio, the bigger the advantage of SCRN over LSTM. Our hypothesis for this phenomenon is as follows: in a morphologically rich language one needs less previous words (context) on average to predict the next word than in English, e.g.

German: Pr(lag | Der, einschlagspunkt)
English: Pr(was | The, location, of, the, impact)

The parameter α in the SCRN model (1) controls how much of the previous context is stored in the slowly changing state $\mathbf{s_t}$. Under the mentioned hypothesis, for the morphologically rich language an optimal value of α should be lower than for English. To verify this we train the medium-sized SCRN + ND + WT on all datasets for different values of α, and the results are provided in Table 7. Indeed, as we can see,

α	PTB	WikiText-2	FR	ES	DE	CS	RU
0.85	88.0	101.1	205	184	264	439	313
0.88	87.5	99.5	198	183	262	422	309
0.90	86.1	98.5	199	**179**	**258**	**420**	**306**
0.95	87.2	97.8	**197**	**179**	260	425	307
0.97	**86.0**	**96.5**	199	180	261	437	313
0.99	92.4	100.0	213	190	287	479	326

Table 7: Performance of SCRN + ND + WT for different values of α.

[7] `http://www.statmt.org/wmt13/translation-task.html`

$\alpha = 0.9$ is more beneficial for German, Czech and Russian (TTR > 0.03), but $\alpha = 0.97$ is better for English PTB and WikiText-2 (TTR < 0.03), while $\alpha = 0.95$ is optimal for French and Spanish (TTR ≈ 0.03). Also, notice that $\alpha = 0.99$ brings the SCRN's results closer to the LSTM's results on non-English data. Therefore, we think LSTM stores too much of the long-term memory via its trainable forget gate, while in SCRN we can directly control this through the α.

9 Conclusion

Being originally implemented in `Torch`, the SCRN model is fully reproducible in `Tensorflow` despite the difference in styles of truncated BPTT in these two libraries. Being conceptually much simpler, the SCRN architecture demonstrates performance comparable to the widely used LSTM model in language modeling task under naïve dropout, but it underperforms the LSTM under variational dropout regularization on English data. However, on texts written in morphologically rich languages, the SCRN with appropriately chosen hyperparameter α outperforms the LSTM.

Acknowledgement

We gratefully acknowledge the NVIDIA Corporation for their donation of the Titan X Pascal GPU used for this research. The work of Zhenisbek Assylbekov has been funded by the Committee of Science of the Ministry of Education and Science of the Republic of Kazakhstan, contract # 346/018-2018/33-28, IRN AP05133700. The authors would like to thank anonymous reviewers for their valuable feedback.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: a system for large-scale machine learning. In *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation*, pages 265–283. USENIX Association.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Jan Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *International Conference on Machine Learning*, pages 1899–1907.

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

Harris Drucker and Yann Le Cun. 1992. Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, 3(6):991–997.

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

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

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

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *International Conference on Learning Representations*.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning*, pages 2342–2350.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2016. Visualizing and understanding recurrent networks. In *International Conference on Learning Representations (Workshop Track)*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2741–2749. AAAI Press.

Thomas Laurent and James von Brecht. 2017. A recurrent neural network without chaos. In *International Conference on Learning Representations*.

Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2017. Recurrent additive networks. *arXiv preprint arXiv:1705.07393*.

Tao Lei and Yu Zhang. 2017. Training rnns as fast as cnns. *arXiv preprint arXiv:1709.02755*.

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

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. 2015. Learning longer memory in recurrent neural networks. In *International Conference on Learning Representations (Workshop Track)*.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM.

Alexander G Ororbia II, Tomas Mikolov, and David Reitter. 2017. Learning simpler language models with the differential state framework. *Neural computation*, 29(12):3327–3352.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 157–163.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2017. Recurrent highway networks. In *International Conference on Machine Learning*, pages 4189–4198.

Structure-Infused Copy Mechanisms for Abstractive Summarization

Kaiqiang Song
Computer Science Dept.
University of Central Florida
Orlando, FL 32816, USA
kqsong@knights.ucf.edu

Lin Zhao
Research and Tech. Center
Robert Bosch LLC
Sunnyvale, CA 94085, USA
lin.zhao@us.bosch.com

Fei Liu
Computer Science Dept.
University of Central Florida
Orlando, FL 32816, USA
feiliu@cs.ucf.edu

Abstract

Seq2seq learning has produced promising results on summarization. However, in many cases, system summaries still struggle to keep the meaning of the original intact. They may miss out important words or relations that play critical roles in the syntactic structure of source sentences. In this paper, we present structure-infused copy mechanisms to facilitate copying important words and relations from the source sentence to summary sentence. The approach naturally combines source dependency structure with the copy mechanism of an abstractive sentence summarizer. Experimental results demonstrate the effectiveness of incorporating source-side syntactic information in the system, and our proposed approach compares favorably to state-of-the-art methods.

1 Introduction

Recent years have witnessed increasing interest in abstractive summarization. The systems seek to condense source texts to summaries that are concise, grammatical, and preserve the important meaning of the original texts (Nenkova and McKeown, 2011). The task encompasses a number of high-level text operations, e.g., paraphrasing, generalization, text reduction and reordering (Jing and McKeown, 1999), posing a considerable challenge to natural language understanding.

Src	A Mozambican man suspect of murdering Jorge Microsse, director of Maputo central prison, has ***escaped*** from the city's police headquarters, local media reported on Tuesday.
Ref	Mozambican suspected of killing Maputo prison director ***escapes***
Sys	mozambican man ***arrested*** for murder
Src	An Alaska father who was too drunk to drive ***had*** his 11-year-old son take the wheel, authorities said.
Ref	Drunk Alaska dad ***has*** 11 year old drive home
Sys	alaska father who was too drunk to drive

Table 1: Example source sentences, reference and system summaries produced by a neural attentive seq-to-seq model. System summaries fail to preserve summary-worthy content of the source (e.g., main verbs) despite their syntactic importance.

The sequence-to-sequence learning paradigm has achieved remarkable success on abstractive summarization (Rush et al., 2015; Nallapati et al., 2016; See et al., 2017; Paulus et al., 2017). While the results are impressive, individual system summaries can appear unreliable and fail to preserve the meaning of the source texts. Table 1 presents two examples. In these cases, the syntactic structure of source sentences is relatively *rare* but perfectly normal. The first sentence contains two appositional phrases ("suspect of murdering Jorge Microsse," "director of Maputo central prison") and the second sentence has a relative clause ("who was too drunk to drive"), both located between the subject and the main verb. The system, however, fails to identify the main verb in both cases; it instead chooses to focus on the first few words of the source sentences. We observe that *rare syntactic constructions* of the source can pose problems for neural summarization systems, possibly for two reasons. First, similar to *rare words*, certain syntactic constructions do not occur frequently enough in the training data to allow the system to learn the patterns. Second, neural summarization systems are not explicitly informed of the syntactic structure of the source sentences and they tend to bias towards sequential recency.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1717–1729
Santa Fe, New Mexico, USA, August 20-26, 2018.

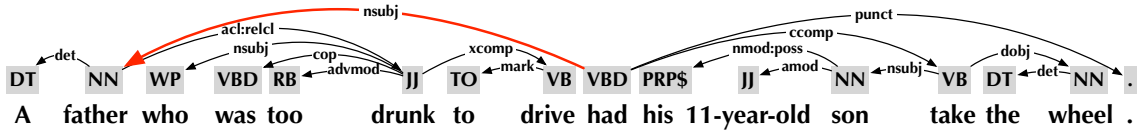


Figure 1: An example dependency parse tree created for the source sentence in Table 1. If important dependency edges such as "father ← had" can be preserved in the summary, the system summary is likely to preserve the meaning of the original.

In this paper we seek to address this problem by incorporating source syntactic structure in neural sentence summarization to help the system identify summary-worthy content and compose summaries that preserve the important meaning of the source texts. We present structure-infused copy mechanisms to facilitate copying source words and relations to the summary based on their semantic and structural importance in the source sentences. For example, if important parts of the source syntactic structure, such as a dependency edge from the main verb to the subject ("father" ← "had," shown in Figure 1), can be preserved in the summary, the "missing verb" issue in Table 1 can be effectively alleviated. Our model therefore learns to recognize important source words and source dependency relations and strives to preserve them in the summaries. Our research contributions include the following:

- we introduce novel neural architectures that encourage salient source words/relations to be preserved in summaries. The framework naturally combines the dependency parse tree structure with the copy mechanism of an abstractive summarization system. To the best of our knowledge, this is the first attempt at comparing various neural architectures for this purpose;
- we study the effectiveness of several important components, including the vocabulary size, a coverage-based regularizer (See et al., 2017), and a beam search with reference mechanism (Tan et al., 2017);
- through extensive experiments we demonstrate that incorporating syntactic information in neural sentence summarization is effective. Our approach surpasses state-of-the-art published systems on the benchmark dataset.[1]

2 Related Work

Prior to the deep learning era, sentence syntactic structure has been utilized to generate summaries with an "*extract-and-compress*" framework. Compressed summaries are generated using a joint model to extract sentences and drop non-important syntactic constituents (Daume III and Marcu, 2002; Berg-Kirkpatrick et al., 2011; Thadani and McKeown, 2013; Durrett et al., 2016), or a pipeline approach that combines generic sentence compression (McDonald, 2006; Clarke and Lapata, 2008; Filippova et al., 2015) with a sentence pre-selection or post-selection process (Zajic et al., 2007; Galanis and Androutsopoulos, 2010; Wang et al., 2013; Li et al., 2013; Li et al., 2014). Although syntactic information is helpful for summarization, there has been little prior work investigating how best to combine sentence syntactic structure with the neural abstractive summarization systems.

Existing neural summarization systems handle syntactic structure only implicitly (Kikuchi et al., 2016; Chen et al., 2016; Zhou et al., 2017; Tan et al., 2017; Paulus et al., 2017). Most systems adopt a "*cut-and-stitch*" scheme that picks words either from the vocabulary or the source text and stitch them together using a recurrent language model. However, there lacks a mechanism to ensure structurally salient words and relations in source sentences are preserved in the summaries. The resulting summary sentences can contain misleading information (e.g., "mozambican man arrested for murder" flips the meaning of the original) or grammatical errors (e.g., verbless, as in "alaska father who was too drunk to drive").

Natural language generation (NLG)-based abstractive summarization (Carenini and Cheung, 2008; Gerani et al., 2014; Fabbrizio et al., 2014; Liu et al., 2015; Takase et al., 2016) also makes extensive use of structural information, including syntactic/semantic parse trees, discourse structures, and domain-specific templates built using a text planner or an OpenIE system (Pighin et al., 2014). In particular, Cao et al. (2018) leverage OpenIE and dependency parsing to extract fact tuples from the source text and use those to improve the faithfulness of summaries.

[1]We made our system publicly available at: `https://github.com/KaiQiangSong/struct_infused_summ`

Different from the above approaches, this paper seeks to directly incorporate source-side syntactic structure in the copy mechanism of an abstractive sentence summarization system. It learns to recognize important source words and relations during training, while striving to preserve them in the summaries at test time to aid reproduction of factual details. Our intent of incorporating source syntax in summarization is different from that of neural machine translation (NMT) (Li et al., 2017a; Chen et al., 2017), in part because NMT does not handle the information loss from source to target. In contrast, a summarization system must selectively preserve source content to render concise and grammatical summaries. We specifically focus on sentence summarization, where the goal is to reduce the first sentence of an article to a title-like summary. We believe even for this reasonably simple task there remains issues unsolved.

3 Our Approach

We seek to transform a source sentence $\mathbf{x}$ to a summary sentence $\mathbf{y}$ that is concise, grammatical, and preserves the meaning of the source sentence. A source word is replaced by its Glove embedding (Pennington et al., 2014) before it is fed to the system; the vector is denoted by $\mathbf{x}_i$ ($i \in [S]$; 'S' for source). Similarly, a summary word is denoted by $\mathbf{y}_t$ ($t \in [T]$; 'T' for target). If a word does not appear in the input vocabulary, it is replaced by a special '$\langle \text{unk} \rangle$' token. We begin this section by describing the basic summarization framework, followed by our new copy mechanisms used to encourage source words and dependency relations to be preserved in the summary.

3.1 The Basic Framework

We build an encoder-decoder architecture for this work. An encoder condenses the entire source text to a continuous vector; it also learns a vector representation for each unit of the source text (e.g., words as units). In this work we use a two-layer stacked bi-directional Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) networks as the encoder, where the input to the second layer is the concatenation of hidden states from the forward and backward passes of the first layer. We obtain the hidden states of the second layer; they are denoted by $\mathbf{h}_i^e$. The source text vector is constructed by averaging over all $\mathbf{h}_i^e$ and passing the vector through a feedforward layer with $\tanh$ activation to convert from the encoder hidden states to an initial decoder hidden state ($\mathbf{h}_0^d$). This process is illustrated in Eq. (2).

$$\mathbf{h}_i^e = f_e(\mathbf{h}_{i-1}^e, \mathbf{x}_i) \quad \mathbf{h}_t^d = f_d(\mathbf{h}_{t-1}^d, \mathbf{y}_{t-1}) \tag{1}$$

$$\mathbf{h}_0^d = \tanh(\mathbf{W}^{h_0} \frac{1}{S} \sum_{i=1}^{S} \mathbf{h}_i^e + \mathbf{b}^{h_0}) \tag{2}$$

A decoder unrolls the summary by predicting one word at a time. During training, the decoder takes as input the embeddings of ground truth summary words, denoted by $\mathbf{y}_t$, while at test time $\mathbf{y}_t$ are embeddings of system predicted summary words (i.e., teacher forcing). We implement an LSTM decoder with the attention mechanism. A context vector $\mathbf{c}_t$ is used to encode the source words that the system attends to for generating the next summary word. It is defined in Eqs (3-5), where $[\cdot||\cdot]$ denotes the concatenation of two vectors. The $\boldsymbol{\alpha}$ matrix measures the strength of interaction between the decoder hidden states $\{\mathbf{h}_t^d\}$ and encoder hidden states $\{\mathbf{h}_i^e\}$. To predict the next word, the context vector $\mathbf{c}_t$ and $\mathbf{h}_t^d$ are concatenated and used as input to build a new vector $\widetilde{\mathbf{h}}_t^d$ (Eq. (6)). $\widetilde{\mathbf{h}}_t^d$ is a surrogate for semantic meanings carried at time step t of the decoder. It is subsequently used to compute a probability distribution over the output vocabulary (Eq. (7)).

$$e_{t,i} = \mathbf{v}^\top \tanh(\mathbf{W}^e[\mathbf{h}_t^d||\mathbf{h}_i^e] + b^e) \tag{3}$$

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{i'=1}^{S} \exp(e_{t,i'})} \tag{4}$$

$$\mathbf{c}_t = \textstyle\sum_{i=1}^{S} \alpha_{t,i} \mathbf{h}_i^e \tag{5}$$

$$\widetilde{\mathbf{h}}_t^d = \tanh(\mathbf{W}^h[\mathbf{h}_t^d||\mathbf{c}_t] + \mathbf{b}^h) \tag{6}$$

$$P_{vocab}(w) = \text{softmax}(\mathbf{W}^y \widetilde{\mathbf{h}}_t^d + \mathbf{b}^y) \tag{7}$$

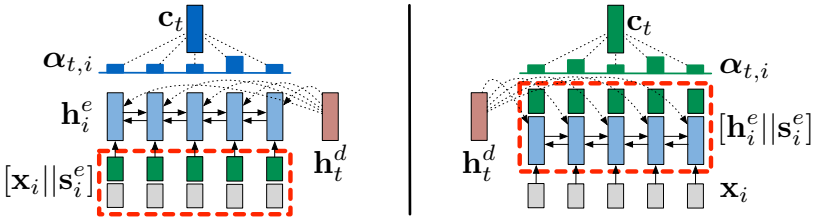


Figure 2: System architectures for 'Struct+Input' (left) and 'Struct+Hidden' (right). A critical question we seek to answer is whether the structural embeddings ($\mathbf{s}_i^e$) should be supplied as input to the encoder (left) or be exempted from encoding and directly concatenated with the encoder hidden states (right).

The copy mechanism (Gulcehre et al., 2016; See et al., 2017) allows words in the source sequence to be selectively copied to the target sequence. It expands the search space for summary words to include both the output vocabulary and the source text. The copy mechanism can effectively reduce out-of-vocabulary tokens in the generated text, potentially aiding a number of applications such as MT (Luong et al., 2015b) and text summarization (Gu et al., 2016; Cheng and Lapata, 2016; Zeng et al., 2017).

Our copy mechanism employs a 'switch' to estimate the likelihood of generating a word from the vocabulary (p_{gen}) vs. copying it from the source text ($1 - p_{gen}$). The basic model is similar to that of the pointer-generator networks (See et al., 2017). The switch is a feedforward layer with sigmoid activation (Eq. (8)). At time step t, its input is a concatenation of the decoder hidden state $\mathbf{h}_t^d$, context vector $\mathbf{c}_t$, and the embedding of the previously generated word $\mathbf{y}_{t-1}$. For predicting the next word, we combine the generation and copy probabilities, shown in Eq. (9). If a word w appears once or more in the input text, its copy probability ($\sum_{i:w_i=w} \alpha_{t,i}$) is the sum of the attention weights over all its occurrences. If w appears in both the vocabulary and source text, $P(w)$ is a weighted sum of the two probabilities.

$$p_{gen} = \sigma(\mathbf{W}^z[\mathbf{h}_t^d||\mathbf{c}_t||\mathbf{y}_{t-1}]) + b^z) \tag{8}$$

$$P(w) = p_{gen}P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} \alpha_{t,i} \tag{9}$$

3.2 Structure-Infused Copy Mechanisms

The aforementioned copy mechanism attends to source words based on their "semantic" importance encoded in $\{\alpha_{t,i}\}$, which measures the semantic relatedness of the encoder hidden state $\mathbf{h}_i^e$ and the decoder hidden state $\mathbf{h}_t^d$ (Eq. (4)). However, the source syntactic structure is ignored. This is problematic, because it hurts the system's ability to effectively identify summary-worthy source words that are syntactically important. We next propose three strategies to inject source syntactic structure to the copy mechanism.

3.2.1 Shallow Combination

Inspired by compressive summarization via structured prediction (Berg-Kirkpatrick et al., 2011; Almeida and Martins, 2013), we hypothesize that structural labels, such as the incoming dependency arc and the depth in a dependency parse tree, can be helpful to predict word importance. We consider six categories of structural labels in this work; they are presented in Table 2. Each structural label is mapped to a fixed-length, trainable structural embedding. However, a critical question remains as to where the structural embeddings should be injected in the existing neural architecture. This problem has not yet been systematically investigated. In this work, we compare two settings:

Structural info	Example
(1) depth in the dependency parse tree	0
(2) label of the incoming edge	'root'
(3) number of outgoing edges	3
(4) part-of-speech tag	'VBD'
(5) absolution position in the source text	9
(6) relative position in the source text	(0.5, 0.6]

Table 2: Six categories of structural labels. Example labels are generated for word 'had' in Figure 1. Relative word positions are discretized into ten buckets.

- **Struct+Input** concatenates structural embeddings of position i (flattened into one vector $\mathbf{s}_i^e$) with the source word embedding $\mathbf{x}_i$ and uses them as a new form of input to the encoder: $\mathbf{x}_i \Rightarrow [\mathbf{x}_i||\mathbf{s}_i^e]$;

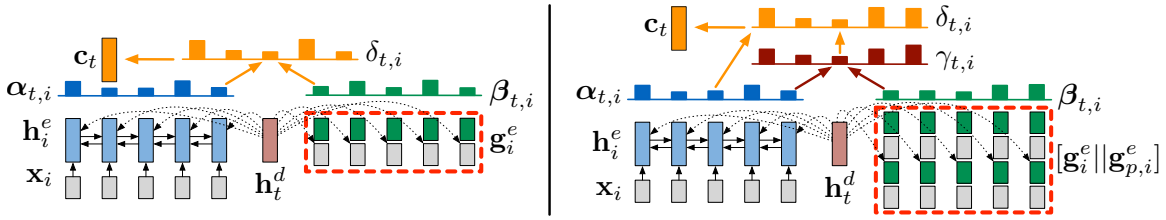


Figure 3: System architectures for 'Struct+2Way+Word' (left) and 'Struct+2Way+Relation' (right). $\beta_{t,i}$ (left) measures the structural importance of the i-th source word; $\beta_{t,i}$ (right) measures the saliency of the dependency edge pointing to the i-th source word. $\mathbf{g}_{p,i}^e$ is the structural embedding of the parent. In both cases $\delta_{t,i}$ replaces $\alpha_{t,i}$ to become the new attention value used to estimate the context vector $\mathbf{c}_t$.

- **Struct+Hidden** concatenates structural embeddings of position i (flattened) with the encoder hidden state $\mathbf{h}_i^e$ and uses them as a new form of hidden states: $\mathbf{h}_i^e \Rightarrow [\mathbf{h}_i^e||\mathbf{s}_i^e]$.

The architectural difference is illustrated in Figure 2. Structural embeddings are important complements to existing neural architectures. However, it is unclear whether they should be supplied as input to the encoder or be left out of the encoding process and directly concatenated with the encoder hidden states. This is a critical question we seek to answer by comparing the two settings. Note that an alternative setting is to separately encode words and structural labels using two RNN encoders, we consider this as a subproblem of the "Struct+Input" case.

The above models complement state-of-the-art by combining semantic and structural signals to determine summary-worthy content. Intuitively, a source word is copied to the summary for two reasons: it contains salient semantic content, or it serves a critical syntactic role in the source sentence. Without explicitly modeling the two factors, 'semantics' can outweigh 'structure,' resulting in summaries that fail to keep the original meaning intact. In the following we propose a two-way mechanism to separately model the "semantic" and "structural" importance of source words.

3.2.2 2-Way Combination (+Word)

Our new architecture involves two attention matrices that are parallel to each other, denoted by $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. $\alpha_{t,i}$ is defined as previously in Eq. (3-4). It represents the "semantic" aspect, calculated as the strength of interaction between the encoder hidden state $\mathbf{h}_i^e$ and the decoder hidden state $\mathbf{h}_t^d$. In contrast, $\beta_{t,i}$ measures the "structural" importance of the i-th input word to generating the t-th output word, calculated by comparing the structure-enhanced embedding $\mathbf{g}_i^e$ with the decoder hidden state $\mathbf{h}_t^d$ (Eq. (10-11)). We use $\mathbf{g}_i^e = [\mathbf{s}_i^e||\mathbf{x}_i]$ as a primitive (unencoded) representation of the i-th source word.

We define $\delta_{t,i} \propto \alpha_{t,i} + \epsilon\beta_{t,i}$ as a weighted sum of $\alpha_{t,i}$ and $\beta_{t,i}$, where a trainable coefficient ϵ is introduced to balance the contribution from both sides (Eq. (12)). Merging semantic and structural salience at this stage allows us to acquire an accurate estimate of how important the i-th source word is to predicting the t-th output word. $\delta_{t,i}$ replaces $\alpha_{t,i}$ to become the new attention value. It is used to calculate the context vector $\mathbf{c}_t$ (Eq. (13)). A reliable estimate of $\mathbf{c}_t$ is crucial as it is used to estimate the generation probability over the vocabulary ($P_{vocab}(w)$, Eq. (6-7)), the switch value (p_{gen}, Eq. (8)), and ultimately used to predict the next word ($P(w)$, Eq. (9)).

$$f_{t,i} = \mathbf{u}^\top \tanh(\mathbf{W}^f[\mathbf{g}_i^e||\mathbf{h}_t^d] + \mathbf{b}^f) \tag{10}$$

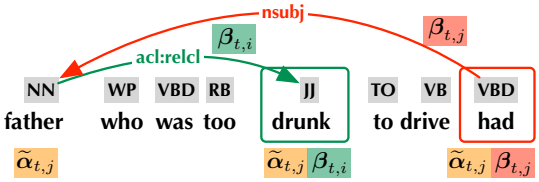
$$\beta_{t,i} = \frac{\exp(f_{t,i})}{\sum_{i'=1}^{S}\exp(f_{t,i'})} \tag{11}$$

$$\delta_{t,i} = \frac{\alpha_{t,i} + \epsilon\beta_{t,i}}{\sum_{i'=1}^{S}(\alpha_{t,i'} + \epsilon\beta_{t,i'})} \tag{12}$$

$$\mathbf{c}_t = \sum_{i=1}^{S}\delta_{t,i}\mathbf{h}_i^e \tag{13}$$

3.2.3 2-Way Combination (+Relation)

We observe that salient source relations also play a critical role in predicting the next word. For example, if a dependency edge ("father" $\xleftarrow{\text{nsubj}}$ "had") is salient and "father" is selected to be included in the summary, it is likely that "had" will be selected next such that a salient source relation ("nsubj") is preserved in the summary. Because summary words tend to follow the word order of the original, we assume selecting a source word and including it in the summary has an impact on its subsequent source words, but not the reverse.



In this formulation we use $\beta_{t,i}$ to capture the saliency of the dependency edge pointing to the i-th source word. Thus, an edge $w_j \leftarrow w_i$ has its salience score saved in $\beta_{t,j}$; and conversely, an edge $w_j \rightarrow w_i$ has its salience score in $\beta_{t,i}$. $\boldsymbol{\beta}$ is calculated in the same way as described in Eq. (10-11). However, we replace $\mathbf{g}_i^e$ with $[\mathbf{g}_i^e||\mathbf{g}_{p,i}^e]$ so that a dependency edge is characterized by the embeddings of its two endpoints ($\mathbf{g}_{p,i}^e$ is the parent embedding). The architectural difference between "Struct+2Way+Word" and "Struct+2Way+Relation" is illustrated in Figure 3.

To obtain the likelihood of w_j being selected to the summary prior to time step t, we define $\widetilde{\alpha}_{t,j} = \sum_{t'=0}^{t-1} \alpha_{t',j}$ that sums up the individual probabilities up to time step t-1. Assume there is a dependency edge $w_j \rightarrow w_i$ (j<i) whose salience score is denoted by $\beta_{t,i}$. At time step t, we calculate $\widetilde{\alpha}_{t,j}\beta_{t,i}$ (or $\widetilde{\alpha}_{t,j}\beta_{t,j}$ for edge $w_j \leftarrow w_i$) as the probability of w_i being selected to the summary, given that *one* of its prior words w_j (j<i) is included in the summary and there is a dependency edge connecting the two. By summing the impact over *all* its previous words, we obtain the likelihood of the i-th source word being included to the summary at time step t in order to preserve salient source relations; this is denoted by $\gamma_{t,i}$ (Eq. (15)). Next, we define $\delta_{t,i} \propto \alpha_{t,i} + \epsilon\gamma_{t,i}$ as a weighted combination of semantic and structural salience (Eq. (16)). $\delta_{t,i}$ replace $\alpha_{t,i}$ to become the new attention values used to estimate the context vector $\mathbf{c}_t$ (Eq. (13)). Finally, the calculation of generation probabilities $P_{vocab}(w)$, switch value p_{gen}, and probabilities for predicting the next word $P(w)$ remains the same as previously (Eq. (6-9)).

$$\widetilde{\alpha}_{t,j} = \sum_{t'=0}^{t-1} \alpha_{t',j} \tag{14}$$

$$\gamma_{t,i} = \sum_{j:j<i} \begin{cases} \widetilde{\alpha}_{t,j}\beta_{t,i} & \text{if } w_j \rightarrow w_i \\ \widetilde{\alpha}_{t,j}\beta_{t,j} & \text{if } w_j \leftarrow w_i \end{cases} \tag{15}$$

$$\delta_{t,i} = \frac{\alpha_{t,i} + \epsilon\gamma_{t,i}}{\sum_{i'=1}^{S}(\alpha_{t,i'} + \epsilon\gamma_{t,i'})} \tag{16}$$

3.3 Learning Objective and Beam Search

We next describe our learning objective, including a coverage-based regularizer (See et al., 2017), and a beam search with reference mechanism (Tan et al., 2017). We want to investigate the effectiveness of these techniques on sentence summarization, which has not been explored in previous work.

Learning objective. Our training proceeds by minimizing a per-target-word cross-entropy loss function. A regularization term is applied to the $\boldsymbol{\alpha}$ matrix. Recall that $\alpha_{t,i} \in [0, 1]$ measures the interaction strength between the t-th output word and the i-th input word. Naturally, we expect a 1-to-1 mapping between the two words. The coverage-based regularizer, proposed by See et al., (2017), encourages this behavior by tracking the historical attention values attributed to the i-th input word (up to time step t-1), denoted by $\widetilde{\alpha}_{t,i} = \sum_{t'=0}^{t-1} \alpha_{t',i}$. The approach then takes the minimum between $\widetilde{\alpha}_{t,i}$ and $\alpha_{t,i}$, which has the practical effect of forcing $\alpha_{t,i}$ ($\forall t$) to be close to either 0 or 1, otherwise a penalty will be applied. The regularizer Ω is defined in Eq. (17), where M is the size of the mini-batch, S and T are the lengths of the source

and target sequences. For two-way copy mechanisms, $\boldsymbol{\delta}$ replaces $\boldsymbol{\alpha}$ to become the new attention values, we therefore apply regularization to $\boldsymbol{\delta}$ instead of $\boldsymbol{\alpha}$. When the regularizer applies, the objective becomes minimizing $(\mathcal{L} + \Omega)$.

$$\Omega = \lambda \sum_{m=1}^{M} \frac{1}{T^{(m)} S^{(m)}} \sum_{t=1}^{T^{(m)}} \sum_{i=1}^{S^{(m)}} \Big(\min(\widetilde{\alpha}_{t,i}, \alpha_{t,i}) \Big) \tag{17}$$

Beam search with reference. During testing, we employ greedy search to generate system summary sequences. For the task of summarization, the ground truth summary sequences are usually close to the source texts. This property can be leveraged in beam search. Tan et al., (2017) describe a beam search with reference mechanism that rewards system summaries that have a high degree of bigram overlap with the source texts. We describe it in Eq. (18), where where $\mathcal{S}(w)$ denotes the score of word w. $\mathcal{B}(\mathbf{y}_{<t}, \mathbf{x})$ measures the number of bigrams shared by the system summary (up to time step t-1) and the source text; $\{\mathbf{y}_{<t}, w\}$ adds a word w to the end of the system summary. The shorter the source text (measured by length S), the more weight a shared bigram will add to the score of the current word w. A hyperparameter η controls the degree of closeness between the system summary and the source text.

$$\mathcal{S}(w) = \log P(w) + \eta \frac{\mathcal{B}(\{\mathbf{y}_{<t}, w\}, \mathbf{x}) - \mathcal{B}(\mathbf{y}_{<t}, \mathbf{x})}{S} \tag{18}$$

4 Experiments

We evaluate the proposed structure-infused copy mechanisms for summarization in this section. We describe the dataset, experimental settings, baselines, and finally, evaluation results and analysis.

4.1 Data Sets

We evaluate our proposed models on the Gigaword summarization dataset (Parker, 2011; Rush et al., 2015). The task is to reduce the first sentence of an article to a title-like summary. We obtain dependency parse trees for source sentences using the Stanford neural network parser (Chen and Manning, 2014). We also use the standard train/valid/test data splits. Following (Rush et al., 2015), the train and valid splits are pruned[2] to improve the data quality. Spurious pairs that are repetitive, overly long/short, and pairs whose source and summary sequences have little word overlap are removed. No pruning is performed for instances in the test set. The processed corpus contains 4,018K training instances. We construct two (non-overlapped) validation sets: "valid-4096" contains 4,096 randomly sampled instances from the valid split; it is used for hyperparameter tuning and early stopping. "valid-2000" is used for evaluation; it allows the models to be trained and evaluated on pruned instances. Finally, we report results on the standard Gigaword test set (Rush et al., 2015) containing 1,951 instances ("test-1951").

4.2 Experimental Setup

We use the Xavier scheme (Glorot and Bengio, 2010) for parameter initialization, where weights are initialized using a Gaussian distribution $\mathbf{W}_{i,j} \sim \mathcal{N}(0, \sigma)$, $\sigma = \sqrt{\frac{2}{n_{in}+n_{out}}}$; n_{in} and n_{out} are numbers of the input and output units of the network; biases are set to be 0. We further implement two techniques to accelerate mini-batch training. First, all training instances are sorted by the source sequence length and partitioned into mini-batches. The shorter sequences are padded to have the same length as the longest sequence in the batch. All batches are shuffled at the beginning of each epoch. Second, we introduce a variable-length batch vocabulary containing only source words of the current mini-batch and words of the output vocabulary. $P(w)$ in Eq. (9) only needs to be calculated for words in the batch vocabulary. It is magnitudes smaller than a direct combination of the input and output vocabularies. Finally, our input vocabulary contains the most frequent 70K words in the source texts and summaries. The output vocabulary contains 5K words by default. More network parameters are presented in Table 3.

[2] https://github.com/facebookarchive/NAMAS/blob/master/dataset/filter.py

Input vocabulary size	70K
Output vocabulary size	5K (default)
Dim. of word embeddings	100
Dim. of structural embeddings	16
Num. of encoder/decoder hidden units	256
Adam optimizer (Kingma and Ba, 2015)	lr = 1e-4
Coeff. for coverage-based regularizer	$\lambda = 1$
Coeff. for beam search with reference	$\eta \approx 13.5$
Beam size	$K = 5$
Minibatch size	$M = 64$
Early stopping criterion (max 20 epochs)	valid. loss
Gradient clipping (Pascanu et al., 2013)	g $\in$ [-5, 5]

Table 3: Parameter settings of our summarization system.

System	Gigaword Valid-2000 R-1	R-2	R-L
Baseline	42.48	21.34	40.18
Struct+Input	42.44	21.75	40.46
Struct+Hidden	42.88	21.81	40.63
Struct+2Way+Word	**43.21**	21.84	**40.86**
Struct+2Way+Relation	42.83	**21.85**	40.60

Table 4: Results on the Gigaword valid-2000 set (full-length F1). Models implementing the structure-infused copy mechanisms ("Struct+*") outperform the baseline.

S:	the government filed another round of criminal charges in a widening stock options scandal
T:	options scandal widens
B:	government files more charges in stock options scandal
I:	another round of criminal charges in stock options scandal
H:	charges filed in stock options scandal
W:	another round of criminal charges in stock options scandal
R:	government files another round of criminal charges in options scandal

Table 5: Example system summaries. 'S:' source; 'T:' target; 'B:' baseline; 'I:' Struct+Input; 'H:' Struct+Hidden; 'W:' 2Way+Word; "R:" 2Way+Relation. "2Way+Relation" is able to preserve important source relations in the summary, e.g., "government ~~‹nsubj~~ files," "files ~~dobj›~~ round," and "round ~~nmod›~~ charges."

S:	red cross negotiators from rivals north korea and south korea held talks wednesday on emergency food shipments to starving north koreans and agreed to meet again thursday
T:	koreas meet in beijing to discuss food aid from south eds
B:	north korea , south korea agree to meet again
I:	north korea , south korea meet again
H:	north korea , south korea meet on emergency food shipments
W:	north korea , south korea hold talks on food shipments
R:	north korea , south korea hold talks on emergency food shipments

Table 6: Example system summaries. "Struct+Hidden" and "2Way+Relation" successfully preserve salient source words ("emergency food shipments"), which are missed out by other systems. We observe that copying "hold talks" from the source also makes the resulting summaries more informative than using the word "meet."

4.3 Results

ROUGE results on valid set. We first report results on the Gigaword valid-2000 dataset in Table 4. We present R-1, R-2, and R-L scores (Lin, 2004) that respectively measures the overlapped unigrams, bigrams, and longest common subsequences between the system and reference summaries[3]. Our baseline system ("Baseline") implements the seq2seq architecture with the basic copy mechanism (Eq. (1-9)). It is a strong baseline that resembles the pointer-generator networks described in (See et al., 2017). The structural models ("Struct+*") differ from the baseline only on the structure-infused copy mechanisms. All models are evaluated without the coverage regularizer or beam search (§3.3) to ensure fair comparison. Overall, we observe that models equipped with the structure-infused copy mechanisms are superior to the baseline, suggesting that combining source syntactic structure with the copy mechanism is effective. We found that the "Struct+Hidden" architecture, which directly concatenates structural embeddings with the encoder hidden states, outperforms "Struct+Input" despite that the latter requires more parameters. "Struct+2Way+Word" also demonstrates strong performance, achieving 43.21%, 21.84%, and 40.86% F_1 scores, for R-1, R-2, and R-L respectively.

ROUGE results on test set. We compare our proposed approach with a range of state-of-the-art neural summarization systems. Results on the standard Gigaword test set ("test-1951") are presented in Table 7. Details about these systems are provided in Table 8. Overall, our proposed approach with structure-infused pointer networks perform strongly, yielding ROUGE scores that are on-par with or surpassing state-of-the-art published systems. Notice that the scores on the valid-2000 dataset are generally higher than those of test-1951. This is because the (source, summary) pairs in the Gigaword test set are not pruned (see §4.1). In some cases, none (or very few) of the summary words appear in the source. This may cause difficulties to the systems equipped with the copy mechanism. The "Struct+2Way+Word" architecture that respectively models the semantic and syntactic importance of source words achieves the highest scores. It outperforms its counterpart of "Struct+2Way+Relation," which seeks to preserve source dependency relations in summaries. We conjecture that the imperfect dependency parse trees generated

[3] w/ ROUGE options: -n 2 -m -w 1.2 -c 95 -r 1000

System	Gigaword Test-1951 R-1	R-2	R-L
ABS (Rush et al., 2015)	29.55	11.32	26.42
ABS+ (Rush et al., 2015)	29.76	11.88	26.96
Luong-NMT (Chopra et al., 2016)	33.10	14.45	30.71
RAS-LSTM (Chopra et al., 2016)	32.55	14.70	30.03
RAS-Elman (Chopra et al., 2016)	33.78	15.97	31.15
ASC+FSC1 (Miao and Blunsom, 2016)	34.17	15.94	31.92
lvt2k-1sent (Nallapati et al., 2016)	32.67	15.59	30.64
lvt5k-1sent (Nallapati et al., 2016)	35.30	16.64	32.62
Multi-Task (Pasunuru et al., 2017)	32.75	15.35	30.82
DRGD (Li et al., 2017b)	36.27	17.57	33.62
Baseline (this paper)	35.43	17.49	33.39
Struct+Input (this paper)	35.32	17.50	33.25
Struct+2Way+Relation (this paper)	35.46	17.51	33.28
Struct+Hidden (this paper)	**35.49**	17.61	33.33
Struct+2Way+Word (this paper)	35.47	**17.66**	**33.52**

Table 7: Results on the Gigaword test-1951 set (full-length F1). Models with structure-infused copy mechanisms ("Struct+*") perform well. Their R-2 F-scores are on-par with or outperform state-of-the-art published systems.

ABS and **ABS+** (Rush et al., 2015) are the first work introducing an encoder-decoder architecture for summarization.

Luong-NMT (Chopra et al., 2016) is a re-implementation of the attentive stacked LSTM encoder-decoder of Luong et al. (2015a).

RAS-LSTM and **RAS-Elman** (Chopra et al., 2016) describe a convolutional attentive encoder that ensures the decoder focuses on appropriate words at each step of generation.

ASC+FSC1 (Miao and Blunsom, 2016) presents a generative auto-encoding sentence compression model jointly trained on labelled/unlabelled data.

lvt2k-1sent and **lvt5k-1sent** (Nallapati et al., 2016) address issues in the attentive encoder-decoder framework, including modeling keywords, capturing sentence-to-word structure, and handling rare words.

Multi-Task w/ Entailment (Pasunuru et al., 2017) combines entailment with summarization in a multi-task setting.

DRGD (Li et al., 2017b) describes a deep recurrent generative decoder learning latent structure of summary sequences via variational inference.

Table 8: Existing summarization methods.

by the parser may affect the "Struct+2Way+Relation" results. However, because the Gigaword dataset does not provide gold-standard annotations for parse trees, we could not easily verify this and will leave it for future work. In Table 5 and 6, we present system summaries produced by various models.

Linguistic quality. To further gauge the summary quality, we hire human workers from the Amazon Mechanical Turk platform to rate summaries on a Likert scale of 1 to 5 according to three criteria (Zhang and Lapata, 2017): *fluency* (is the summary grammatical and well-formed?), *informativeness* (to what extent is the meaning of the original sentence preserved in the summary?), and *faithfulness* (is the summary accurate and faithful to the original?). We sample 100 instances from the test set and employ 5 turkers to rate each summary; their averaged scores are presented in Table 9. We found that "Struct+2Way+Relation" outperforms "Struct+Input" on all three criteria. It also compares favorably to ground-truth summaries on "fluency" and "faithfulness." On the other hand, the ground-truth summaries, corresponding to article titles, are judged as less satisfying according to human raters.

System	Info.	Fluency	Faithful.
Struct+Input	2.9	3.3	3.0
Struct+2Way+Relation	3.0	3.4	3.1
Ground-truth Summ.	3.2	3.5	3.1

Table 9: Informativeness, fluency, and faithfulness scores of summaries. They are rated by Amazon turkers on a Likert scale of 1 (worst) to 5 (best). We choose to evaluate Struct+2Way+Relation (as oppose to 2Way+Word) because it focuses on preserving source relations in the summaries.

Dependency relations. We investigate the source dependency relations preserved in the summaries in Table 10. A source relation is considered preserved if both its words appear in the summary. We observe that the models implementing structure-infused copy mechanisms (e.g., "Struct+2Way+Word") are more likely to preserve important dependency relations in the summaries, including nsubj, dobj, amod, nmod, and nmod:poss. Dependency relations that are less important (mark, case, conj, cc, det) are less likely to be preserved. These results show that our structure-infused copy mechanisms can learn to recognize the importance of dependency relations and selectively preserve them in the summaries.

Coverage and reference beam. In Figure 11, we investigate the effect of applying the coverage regularizer ("coverage") and reference-based beam search ("ref_beam") (§3.3) to our models. The coverage regularizer is applied in a second training stage, where the system is trained for an extra 5 epochs with coverage and the model yielding the lowest validation loss is selected. Both coverage and ref_beam can improve the system performance. Our observation suggests that ref_beam is an effective addition to shorten the gap between different systems.

System	nsubj	dobj	amod	nmod	nmod:poss	mark	case	conj	cc	det
Baseline	7.23	12.07	20.45	8.73	12.46	15.83	14.84	9.72	5.03	2.22
Struct+Input	7.03	11.72	19.72	**9.17**↑	12.46	15.35	14.69	9.55	4.67	1.97
Struct+Hidden	**7.78**↑	**12.34**↑	**21.11**↑	**9.18**↑	**14.86**↑	14.93	**15.84**↑	9.47	3.93	**2.65**↑
Struct+2Way+Word	**7.46**↑	**12.69**↑	**20.59**↑	**9.03**↑	**13.00**↑	15.83	14.43	8.86	3.48	1.91
Struct+2Way+Relation	**7.35**↑	**12.07**↑	**20.59**↑	8.68	**13.47**↑	15.41	14.39	9.12	4.30	1.89

Table 10: Percentages of source dependency relations (of various types) preserved in the system summaries.

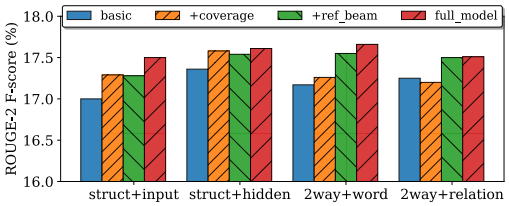


Table 11: Effects of applying the coverage regularizer and the reference beam search to structural models, evaluated on test-1951. Combining both yields the highest scores.

$\|V\|$	R-2	Train Speed	InVcb	InVcb+Src
1K	13.99	2.5h/epoch	60.57	76.04
2K	15.35	2.7h/epoch	69.71	80.72
5K	17.25	3.2h/epoch	79.98	86.51
10K	17.62	3.8h/epoch	88.26	92.18

Table 12: Results of the "Struct+2Way+Relation" system trained using output vocabularies of various sizes ($|V|$), evaluated on test-1951 w/o coverage or ref_beam. The training speed is calculated as the elapsed time (hours) per epoch, tested on a GTX 1080Ti GPU card.

Output vocabulary size. Finally, we investigate the impact of the output vocabulary size on the summarization performance in Table 12. All our models by default use an output vocabulary of 5K words in order to make the results comparable to state-of-the-art-systems. However, we observe that there is a potential to further boost the system performance (17.25→17.62 R-2 F1-score, w/o coverage or ref_beam) if we had chosen to use a larger vocabulary (10K) and can endure a slightly longer training time (1.2x). In Table 12, we further report the percentages of reference summary words covered by the output vocabulary ("InVcb") and covered by either the output vocabulary or the source text ("InVcb+Src"). The gap between the two conditions shortens as the size of the output vocabulary is increased.

5 Conclusion

In this paper, we investigated structure-infused copy mechanisms that combine source syntactic structure with the copy mechanism of an abstractive summarization system. We compared various system architectures and showed that our models can effectively preserve salient source relations in summaries. Results on benchmark datasets showed that the structural models are on-par with or surpass state-of-the-art published systems.

References

Miguel B. Almeida and Andre F. T. Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proceedings of ACL*.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Giuseppe Carenini and Jackie Chi Kit Cheung. 2008. Extractive vs. NLG-based abstractive summarization of evaluative text: The effect of corpus controversiality. In *Proceedings of the Fifth International Natural Language Generation Conference (INLG)*.

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

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for document summarization. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of ACL*.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of NAACL*.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*.

Hal Daume III and Daniel Marcu. 2002. A noisy-channel model for document compression. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Giuseppe Di Fabbrizio, Amanda J. Stent, and Robert Gaizauskas. 2014. A hybrid approach to multi-document summarization of opinions in reviews. *Proceedings of the 8th International Natural Language Generation Conference (INLG)*.

Katja Filippova, Enrique Alfonseca, Carlos Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Dimitrios Galanis and Ion Androutsopoulos. 2010. An extractive supervised two-stage method for sentence compression. In *Proceedings of NAACL-HLT*.

Shima Gerani, Yashar Mehdad, Giuseppe Carenini, Raymond T. Ng, and Bita Nejat. 2014. Abstractive summarization of product reviews using discourse structure. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of ACL*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

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

Hongyan Jing and Kathleen McKeown. 1999. The decomposition of human-written summary sentences. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Chen Li, Fei Liu, Fuliang Weng, and Yang Liu. 2013. Document summarization via guided sentence compression. In *Proceedings of EMNLP*.

Chen Li, Yang Liu, Fei Liu, Lin Zhao, and Fuliang Weng. 2014. Improving multi-documents summarization by sentence compression based on expanded constituent parse tree. In *Proceedings of EMNLP*.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017a. Modeling source syntax for neural machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017b. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Chin-Yew Lin. 2004. ROUGE: a package for automatic evaluation of summaries. In *Proceedings of ACL Workshop on Text Summarization Branches Out*.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *Proceedings of EACL*.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*.

Robert Parker. 2011. English Gigaword fifth edition LDC2011T07. *Philadelphia: Linguistic Data Consortium*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Ramakanth Pasunuru, Han Guo, and Mohit Bansal. 2017. Towards improving abstractive summarization via entailment generation. In *Proceedings of the Workshop on New Frontiers in Summarization*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference Empirical Methods in Natural Language Processing (EMNLP)*.

Daniele Pighin, Marco Cornolti, Enrique Alfonseca, and Katja Filippova. 2014. Modelling events through memory-based, open-ie patterns for abstractive summarization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for sentence summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kapil Thadani and Kathleen McKeown. 2013. Sentence compression with joint structural inference. In *Proceedings of CoNLL*.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multi-document summarization. In *Proceedings of ACL*.

David Zajic, Bonnie J. Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing and Management*.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2017. Efficient summarization with read-again and copy mechanism. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Measuring the Diversity of Automatic Image Descriptions

Emiel van Miltenburg
Vrije Universiteit Amsterdam
emiel.van.miltenburg@vu.nl

Desmond Elliott
University of Edinburgh
d.elliott@ed.ac.uk

Piek Vossen
Vrije Universiteit Amsterdam
piek.vossen@vu.nl

Abstract

Automatic image description systems typically produce generic sentences that only make use of a small subset of the vocabulary available to them. In this paper, we consider the production of generic descriptions as a lack of diversity in the output, which we quantify using established metrics and two new metrics that frame image description as a word recall task. This framing allows us to evaluate system performance on the head of the vocabulary, as well as on the long tail, where system performance degrades. We use these metrics to examine the diversity of the sentences generated by nine state-of-the-art systems on the MS COCO data set. We find that the systems trained with maximum likelihood objectives produce less diverse output than those trained with additional adversarial objectives. However, the adversarially-trained models only produce more types from the head of the vocabulary and not the tail. Besides vocabulary-based methods, we also look at the compositional capacity of the systems, specifically their ability to create compound nouns and prepositional phrases of different lengths. We conclude that there is still much room for improvement, and offer a toolkit to measure progress towards the goal of generating more diverse image descriptions.

1 Introduction

Automatic image description is a challenging task because natural language and the visual world both have an unbounded range of variation (Bernardi et al., 2016). Computational image description models are trained to generalize over datasets of images with multiple human descriptions, however, much of the variation present in these descriptions is lost in a trained model. Dai et al. (2017) note that the descriptions generated by recurrent neural networks using a maximum-likelihood objective are "overly rigid and lacking in variability." This rigidity and lack of variability in the output of state-of-the-art models is unfortunate because human descriptions are the exact opposite of this: Devlin et al. (2015) found that humans typically produce unique descriptions, i.e. only 4.8% of the human-described evaluation data in the MS COCO data set (Lin et al., 2014) also occur in the training data.

The lack of variability in machine-generated text is not limited to automatic image description; it is a general problem in natural language generation. Simply put: automatically generated text quickly becomes boring or repetitive. Recent efforts to address this problem include using maximum mutual information as an objective function, rather than the likelihood of the output, to improve the variablity of a neural conversation model (Li et al., 2016). Castro Ferreira et al. (2016) focused on the deterministic nature of NLG systems, in the sense that they repeatedly use the same referential forms to refer to the same entity in longer stretches of text. They addressed this problem by explicitly training their model to mimic human variability for referring expression generation.

In the image description literature, there have been two recent approaches to generating diverse outputs: (i) learn different description distributions simultaneously to generate multiple different descriptions for the same image (Wang et al., 2016); and (ii) augmenting a model with an additional (conditional) Generative Adversarial Network objective (Goodfellow et al., ; Mirza and Osindero, 2014, GAN)

Proceedings of the 27th International Conference on Computational Linguistics, pages 1730–1741
Santa Fe, New Mexico, USA, August 20-26, 2018.

to generate more natural and diverse descriptions. In this setting, the caption generator tries to fool a discriminator that is trying to distinguish human image descriptions from machine-generated ones (Dai et al., 2017; Shetty et al., 2017). From these papers, two definitions of diversity emerge:

Local diversity: The ability to generate many different descriptions for the same image.
Global diversity: The ability to use (many different combinations of) many different words.

The former is *local* because it can be evaluated for individual images. The latter is *global*, because it is a property at the corpus level. This paper focuses on global diversity, which means that we will study whether systems are able to produce as many different words and phrases as humans do in their descriptions of images. We know that word frequencies follow a *Zipfian* (or *power law*) distribution (Zipf, 1949; Van Heuven et al., 2014; Corral et al., 2015), which means that a small subset of the vocabulary accounts for the largest part of the data. Natural language processing systems trained on corpus data are sensitive to this, and tend to overfit on the head of the distribution (e.g. Postma et al., 2016). We will show that this also holds for the output of image description systems: all systems considered in this paper mainly use the top 20% most frequent words.

In this paper, we consider the following question: **How can we measure the diversity of the output generated by an image description model?** There is currently a lack of consensus about how to measure the diversity of model output but the metrics used thus-far fall into four broad areas:

(i) Modified[1] type-token ratio: the number of distinct unigrams or bigrams, divided by the total number of generated words (Li et al., 2016; Shetty et al., 2017).
(ii) mBLEU: compute the average BLEU score (Papineni et al., 2002) between each description and the other descriptions generated for the same image. This metric can only be used to evaluate models that produce multiple descriptions per image (Wang et al., 2016; Shetty et al., 2017).
(iii) Model-internal: a Generative Adversarial evaluator network that judges whether descriptions are more natural-sounding and semantically relevant than human descriptions (Dai et al., 2017); and
(iv) vocabulary size and the proportion of uniquely generated sentences (Shetty et al., 2017).

In addition to this lack of consensus about which metrics should be used to measure diversity, it is not known how state-of-the-art systems differ in terms of output diversity because it has not been standard practice to report this type of performance statistics. In this paper, we present an overview of metrics to assess the diversity of automatically generated English image descriptions, and compare them using nine state-of-the-art image description systems (Section 2). Besides covering existing metrics, like TTR and average sentence length, we also propose two word recall metrics that provide more information about the output vocabulary (Section 3). We also look at the compositional capacity of the different systems, by examining how many different compound nouns and prepositional phrases they can produce. We use these metrics to analyse how image description systems differ from human descriptions (Section 4). It is not our goal to evaluate the quality of the descriptions, though future research may find that more diverse descriptions are also more attractive for human readers (Section 5.2).

The main finding of our analysis is that recent GAN-based systems (Dai et al., 2017; Shetty et al., 2017), designed to produce more human-like image descriptions, do indeed produce more diverse output than the other MLE-based systems, but this increased diversity still mostly comes from the head of the vocabulary. In order to support future analyses, we release a toolkit to assess the output of any system and to compare the results with existing approaches.[2]

2 Existing metrics

This section discusses six general metrics to measure output diversity at the word level, along with a method to visually inspect the differences between systems. All of these methods require tokenized image descriptions – we use SpaCy 2.0.4 for this purpose and lowercase all of the tokens. The validation data is different from the system output, in that it consists of 5 reference descriptions per image, while

[1]This is similar to the type-token ratio (TTR; number of types divided by number of tokens), except that it is customary to compute TTR over a fixed number of tokens, as TTR decreases with corpus size (Youmans, 1990).

[2]Toolkit: `https://github.com/evanmiltenburg/MeasureDiversity`

Type	System	BLEU	Meteor	ASL	SDSL	Types	TTR_1	TTR_2	%Novel	Cov	Loc_5
MLE	Liu et al. 2017	32.3	25.8	10.3	1.32	598	0.17	0.38	50.1	0.05	0.70
	Mun et al. 2017	32.6	25.7	9.4	1.12	1009	0.16	0.38	50.0	0.08	0.78
	Shetty et al. 2016	31.9	25.2	9.0	1.03	1112	0.15	0.34	43.0	0.08	0.74
	Tavakoli et al. 2017	28.7	23.5	9.2	1.03	917	0.15	0.33	38.8	0.07	0.66
	Vinyals et al. 2017	32.1	25.7	10.1	1.28	953	0.21	0.43	90.5	0.07	0.69
	Wu et al. 2016	31.0	25.0	9.1	1.03	849	0.14	0.32	44.5	0.06	0.72
	Zhou et al. 2017	30.0	24.8	9.3	1.20	1334	0.22	0.51	60.1	0.10	0.80
GAN	Dai et al. 2017	20.7	22.4	9.8	1.63	1922	0.23	0.55	87.7	0.15	0.76
	Shetty et al. 2017	–	23.6	9.4	1.31	2611	0.24	0.54	80.5	0.20	0.71
	Validation data	–	–	11.3	2.61	9200	0.32	0.72	95.3	–	–

Table 1: System results: BLEU and Meteor scores; average sentence length; standard deviation of sentence length; mean-segmented type-token ratio (TTR); bigram TTR; percentage novel descriptions; coverage; and local recall with importance class 5. BLEU/Meteor scores are originally reported values, except for (Dai et al., 2017) and (Vinyals et al., 2017), which we computed on the validation set.

the systems only produce one description per image. Hence, for the validation data, we compute each score 5 times – once per reference description – and report the average.

1. The **average sentence length (ASL)** corresponds to the mean number of tokens per sentence.
2. The **standard deviation of the sentence length (SDSL)** is a measure of how much systems vary in their description lengths.
3. The **number of types** measures the number of unique word types in the output vocabulary.
4. The **mean segmented type-token ratio (TTR_1)** is the average number of types per 1000 tokens (Johnson, 1944). It is not affected by sentence length because it is computed for a fixed number of tokens. It is more difficult to artificially increase than the number of types because it is an average.
5. The **bigram TTR (TTR_2)** is the average number of bigram types per 1000 bigram tokens. This is based on Li et al.'s (2016) diversity metric (looking at bigram diversity), and the MSTTR metric (using a fixed size, averaging over multiple samples) so that it is not biased by description length.
6. The **percentage novel descriptions (%Novel)** refers to the generated descriptions that do not occur in the training data. Note that there may be duplicates among the novel descriptions.

2.1 Systems

For any analysis of output diversity, it is essential to have the generated descriptions. Unfortunately, this data is generally not available for most published systems. We contacted the authors of papers that appeared in relevant conferences and journals between 2016–2017[3], and received nine responses with descriptions generated for the MS COCO validation set. All these systems are listed in Table 1. With the exception of the two GAN-based systems (Dai et al., 2017; Shetty et al., 2017), the other systems are based on a conditioned recurrent neural network, trained using a Maximum Likelihood (MLE) objective.

2.2 Results

Table 1 presents the results for the metrics discussed above. We discuss each of them in turn.

Average sentence length. We observe that all models produce shorter sentences than humans, on average, perhaps also conveying less information. It also means that the BLEU *brevity penalty* (Papineni et al., 2002) and Meteor *length penalty* (Denkowski and Lavie, 2014) are affecting the metric scores. However, *producing shorter sentences* does not necessarily mean *producing worse descriptions*.

Standard deviation of sentence length. We observe that the GAN-based systems vary more than most other systems, but the systems by Liu et al. (2017) and Vinyals et al. (2017) have more variation than other MLE-based systems. Humans vary much more than any model in the length of their descriptions.

Number of types. The model by Liu et al. (2017) produces the fewest distinct word types (598), which severely limits the output diversity of the system. The two GAN-based models produce the most

[3]We surveyed AAAI, ACL, BMVC, COLING, CVPR, EACL, EMNLP, ICCV, ICLR, ICPR, IJCAI, NAACL, and NIPS.

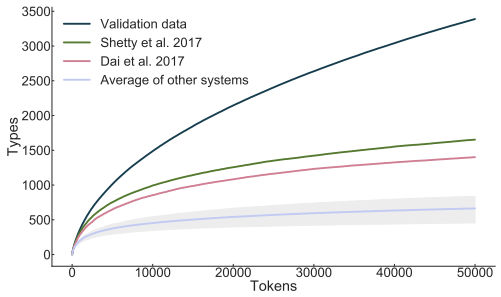


Figure 1: Type-token curves for nine systems. The validation data grows much faster than any of the systems and the GAN-based systems clearly outperform the other systems (shaded, with a line plotting the average performance).

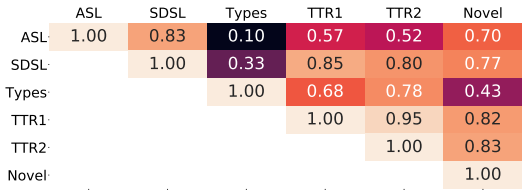


Figure 2: Absolute Spearman correlation between the different diversity metrics, computed over the results for the 9 different systems

distinct word types: 1,922 and 2,611. This is still much lower than the human type count, which averages at 9,200. The total number of types in the validation set is much higher, at 17,557.

TTR$_{\{1,2\}}$ We find that the GAN-based models again outperform the rest. But in terms of variation, there is still much room for improvement before they reach human parity.

Percentage novel descriptions. We find that the model by Vinyals et al. (2017) outperforms the rest (90.5% novel), with the GAN-based systems following close behind at 87.7% and 80.5% novel. The remainder of the systems reproduce a sentence from the training data approximately 50% of the time.

We visualize the differences between the systems using a **type-token curve (TTC)**, which shows how the number of types develops as one reads more output tokens (Youmans, 1990). This curve was originally proposed to compare different texts, which means that sentence order is fixed. With automatic image description, we do not have this constraint. Rather than taking a single sample, and reading the image descriptions in a single order, we randomized the order of the descriptions ten times, and computed the average TTC for the validation data for each system. Figure 1 shows the type-token curves for the validation data and all systems. We observe that the TTC for the validation data develops much more rapidly than that of the systems. Moreover, we can clearly see how the two GAN-based systems stand out from the others in producing more diverse output.

We now inspect how strongly the different existing metrics correlate with each other. Figure 2 shows the correlation matrix between the different general metrics for measuring diversity. We observe that TTR_1 and TTR_2 are almost perfectly correlated. We conclude from this that a single type-token ratio measure is enough to capture differences between systems in their use of different types. The number of novel descriptions is strongly correlated with the type-token ratio. An intuitive explanation for this is that whenever a model produces more varied output, it is also more likely to produce novel output. In this light, it is interesting to observe the lower correlation between the number of types and the percentage of novel sentences. An explanation for this may be that producing more different types in total does not necessarily mean more diverse output. A system has to *consistently* produce more different types to have an impact.

3 Image description as word recall

Image description can be simplified to a word recall problem, where the goal is simply to produce a bag of words that should overlap with the reference data. By ignoring sentence structure, we can focus on the richness of the vocabulary, and study system performance for different classes of words. We distinguish between global recall, looking at the corpus as a whole, and local recall, looking at the corpus image-by-image. We also introduce ranking measures based on these concepts.

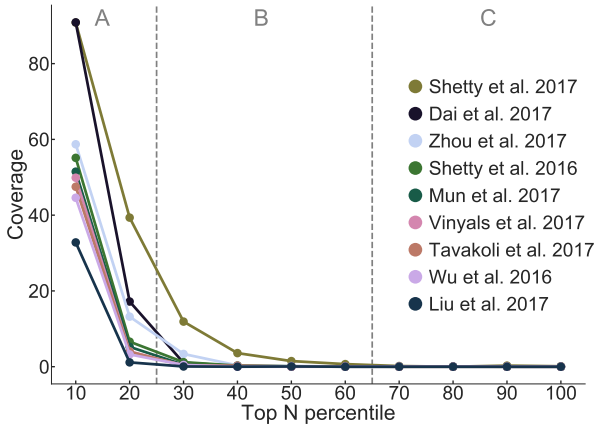


Figure 4: Coverage (equation 3) for different subsets of the learnable words. Recall for all systems is best for the top 10% most frequent words, but immediately drops for the next 10% of most frequent words.

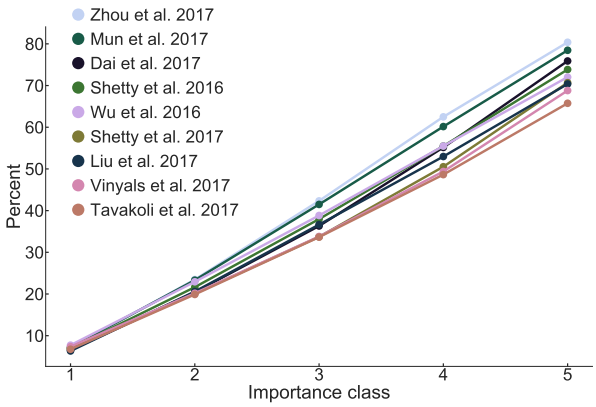


Figure 5: Local recall scores for all systems for each word importance class. Systems have low recall for words that occur only once in the reference descriptions, but their recall grows to 65-80% when all five references mention the same word.

3.1 Global recall

We formally define the *global recall* metrics in Figure 3. The sets TRAIN, EVAL, GEN correspond to the words that are in the training set, evaluation set, or those generated by the model. Any word type that is both in TRAIN and EVAL is *learnable* from the training data (Eq. 1).[4] Recalled words are those that are both learnable and generated by the model (Eq. 2). We quantify the success of a system as the percentage of learnable words it can recall, i.e. coverage (Eq. 3). Since the set of learnable word types is a subset of the word types in EVAL (this follows from (Eq. 1)), systems that are trained on the training data alone cannot recall *all* word types in EVAL. We define this limit in (Eq. 4). Intuitively, a model that has a higher coverage (Eq. 3) can recall more types from the learnable set (Eq. 1), therefore the model is producing a more globally diverse output.

Using the coverage metric to evaluate the nine systems, we find that the GAN-based systems of Shetty et al. (2017) and Dai et al. (2017) once again achieve the highest scores, achieving 15-20% coverage. This still leaves much room for improvement. We further explore coverage for 10 different subsets of the learnable word types, ranging from the 10% most to the 10% least frequent types in the validation data (based on the counts in the validation set).

$$\text{Learnable} = \text{TRAIN} \cap \text{EVAL} \quad (1)$$

$$\text{Recalled} = \text{GEN} \cap \text{Learnable} \quad (2)$$

$$\text{Coverage} = \frac{|\text{Recalled}|}{|\text{Learnable}|} \quad (3)$$

$$\text{Limit} = \frac{|\text{Learnable}|}{|\text{Eval}|} \quad (4)$$

Figure 3: Definitions for global recall.

Figure 4 shows the results. We see that the two GAN-based systems achieve almost 90% coverage of the most frequent types, but this score quickly degrades. Other systems only achieve about 60% coverage of the head, and degrade even more quickly than the GAN-based systems. Furthermore, we observe that the GAN-based systems only achieve better coverage than the other systems on the head of the distribution. Dai et al.'s system is only better for the $0-20\%$ most frequent terms (part A), and Shetty et al.'s (2017) system still shows higher coverage than the others up to the 60% mark (part B), but there is no difference for the rest of the lexicon (part C). We emphasize that, for global recall, a system only has to use a type *once* for it to be counted. The Limit for the MS COCO validation set is 0.75. This means that the other 25% (4356 words) in the validation set cannot be learned on the basis of the training set.

[4] We ignore zero-shot learning approaches that could learn to describe images using words outside the training data.

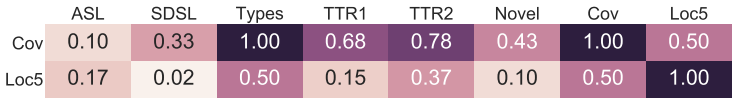


Figure 6: Spearman correlations between our coverage and local recall metric and the existing metrics.

3.2 Local recall

Local recall considers each image in the evaluation data as a separate word recall problem. We define the local target set as the union of the descriptions (sets of words, D) for an image I_i (Eq. 5). The goal is to recall the content words that are important to describe the image. We used SpaCy 2.0.4 to tag the descriptions and we only use adjectives, verbs, nouns, and adverbs as content words for the analysis.

Recalled words are those that are generated for a specific image I_i and occur in the local target set (Eq. 6). We define the importance of a word w for an image I in terms of the number of descriptions D that the word w occurs in (Eq. 7), resulting in a value between 1 and N (here N=5, as there are 5 descriptions per image). We use the importance metric to measure how well a system recalls the essential (with a score of 5) or the majority (3 or higher) words.

$$\text{Local}_i = \bigcup_{D_j \in I_i} \{w : w \in D_j\} \tag{5}$$

$$\text{Recalled}_i = \text{Gen}_i \cap \text{Local}_i \tag{6}$$

$$\text{Importance}(w, I) = |\{D : w \in D \wedge D \in I\}| \tag{7}$$

$$\text{Local recall score}_k = \frac{1}{|\text{Val}|} \sum_{I \in \text{Val}} \frac{|\{w : w \in \text{Recalled}_i \wedge \text{Importance}(w, I_i) = k\}|}{|\{w : w \in \text{Local}_i \wedge \text{Importance}(w, I_i) = k\}|} \tag{8}$$

The *local recall score* for words of *k* importance is computed by dividing the total number of recalled words with an importance of *k* by the total number of words with an importance of *k* (Eq. 8). Figure 5 shows the scores for all 9 systems. All models achieve local recall scores between 65% and 80% for types that are mentioned in all five references. This time, the GAN-based models do not outperform the rest, although they still have recalls around 75%. Although local recall is not strictly about diversity in output vocabulary, it does test each system's ability to use the right words at the right time (even if those words are rare).

Figure 6 shows the correlations between coverage (Eq. 3) and the local recall metric with the existing measures of diversity that were discussed earlier. We find that coverage and the number of types are perfectly correlated. Future work may find that these two measures do not always correlate perfectly, since coverage is based on the word types in the evaluation set. If future systems start producing more word types that are not in the evaluation set, we would see a divergence between coverage and number of types. Local recall (Loc_5 in the table), does not correlate as strongly with the other metrics.

3.3 Global ranking of omitted words

Instead of using local and global recall to produce scores summarizing model performance, we can use these metrics to construct a ranking of words typically produced by a model, or that a model typically fails to produce. We refer to ranking on the basis of global recall as *global ranking*. The most straightforward way to use global ranking is to construct a frequency table for all words in the evaluation set that are not recalled by a model. This gives us a list of the *most common omissions* for that model. Table 2 presents the 15 most frequent words that *all* systems failed to produce. The first ranking is based on the frequency in the training set; the second ranking on the basis of the validation set frequency. The advantage of the former is that we see which words are omitted even though there is sufficient evidence. The advantage of the latter is that we see which words are omitted, even though there are sufficient contexts in which those words could have been used.

Two types that immediately stand out are *'s* and *n't*. One possible reason that both these types were never produced by any system is that they are (cognitively) complex. The possessive *'s* indicates abstract

Global ranking	
Train	Validation
's	's
elderly	elderly
toast	toast
we	we
whole	thrown
laughing	whole
displays	ham
meadow	located
located	driver
ham	mat
nicely	n't
n't	heading
almost	displays
more	amongst
picking	simple

Local ranking		
Absolute	Relative	Relative$_{10}$
man	pillow	door
dog	kitten	paper
woman	flag	van
people	turkey	pink
cat	milk	head
umbrella	ice	doll
dogs	chips	hair
sign	rainbow	pool
pizza	potatoes	fork
ball	map	tray
cake	eggs	carrot
bear	cream	girls
bed	butter	apple
table	strawberries	women
elephant	pregnant	rice

Table 2: Global and local rankings of omitted words. These rankings show the most frequent words that *are not* produced at all (Global ranking), or that are most commonly omitted by the 9 image description systems (Local ranking).

relations between animate entities and objects that vary from scene to scene, making it difficult to learn how to use this type on the basis of visual information alone. The use of negations like *n't* typically requires the speaker to reason about whether or not an image conforms with their expectations (van Miltenburg et al., 2016). Another difficult case is *thrown*, which refers to a throwing action taking place before the picture was taken. Completing the top-3 in both rankings are *elderly* (253 occurrences in the training data, 140 in the validation data) and *toast* (237 and 124). These are less complex than the examples mentioned above, and could be determined on the basis of visual information alone. Further research is needed to determine why these words could not be produced by any system.

3.4 Local ranking of omitted words

We refer to rankings produced on the basis of local recall as *local ranking*. With local ranking, we can look at the words that models failed to produce most often. For reasons of space, we will only look at the words with importance class $k = 5$. Table 2 presents three local rankings:

1. An *absolute* ranking, where we look at the aggregate number of times each word was missed by the models under investigation.
2. A *relative* ranking, where we look at the rate at which each word was missed (Eq. 9). In the case of a tie, the most frequent word 'wins', so that words with the largest impact on model performance are ranked higher.

$$\text{MissRatio}(w) = \frac{\text{missed}(w)}{\text{missed}(w) + \text{recalled}(w)} \tag{9}$$

3. A relative ranking with an *occurrence threshold*, where each word with importance class $k = 5$ has to occur at least $n = 10$ times for each system. This eliminates words from the ranking that occur only a few times, but that are missed by all systems (so MissRatio(w) = 1).

All three rankings provide a starting point to explore system performance. For example, in the first ranking, we observe that some of the most common terms in the MS COCO dataset overall (*man* and *woman*) are often missed by image description systems, when all annotators *do* use those terms. Since these words are ranked high, they have a big impact on the quality of the descriptions. For reasons of space, we cannot discuss this example in depth, but a natural next step would be to look at example descriptions where systems fail to produce *man* or *woman* and identify potential causes of this behavior (e.g. an inability to determine people's gender using only visual information).

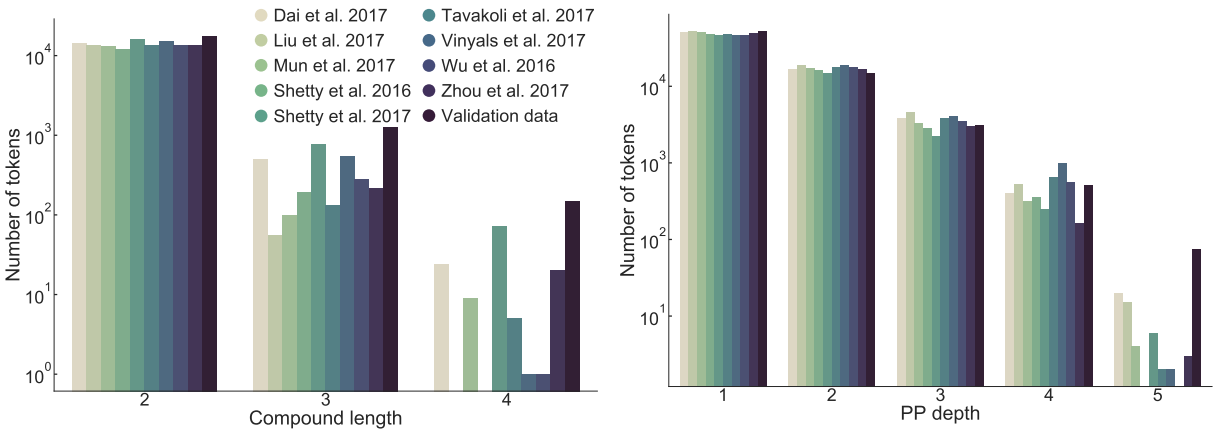


Figure 7: Histograms showing the number of tokens with compound length 2, 3, and 4, and the number of tokens with PP-depth 1–5, for all 9 systems and the MS COCO validation data. The validation data and the two GAN-based systems (Dai et al., 2017; Shetty et al., 2017) clearly have more compound tokens than the other systems. We do not observe this difference with PP depth.

4 Compound nouns and prepositional phrases

Beyond the word level, we can look at how words are combined to form new phrases (i.e. *compositionality* (Szabó, 2017)). We detect compound nouns using a part-of-speech tagger (SpaCy 2.0.4), assuming that any sequence of nouns is a nominal compound. We also compute the *compound ratio*: the average number of compounds per description. Figure 7 and Table 3 show the results. We observe that the validation data has a larger number of compound nouns, resulting in a higher compound ratio. When we separate the compounds by length, we see that humans produce most compounds in any category, and the GAN-based systems (Dai et al., 2017; Shetty et al., 2017) produce more compounds of length 3 and 4 than the other systems. The system by Vinyals et al. (2017) also stands out in this regard. Finally, we see that the GAN-based systems produce more compound types of length 2 than any other system, but there is still a big gap between the GAN-based systems and human performance.

We detect prepositional phrases (PPs), such as *in the kitchen*, using SpaCy's part-of-speech tagger and dependency parser. First, we identify each preposition in the description (e.g. *in, with, on*). Then we inspect the subtree headed by those prepositions. For each of those subtrees, we count their depth in terms of PP embeddings, e.g. *on top of a pan on a table* (1) has a depth of 3.

(1) [**on** top [**of** a pan [**on** a table]]]

We also compute the *preposition ratio*, which is the average number of prepositions per description. Table 3 shows the results. We do not see a big difference between the validation data and the systems. The only difference is that humans produce more types of PPs with depth 1: twice as many as the System by Dai et al. (2017). We conclude that image description systems still have much to gain in terms of compositionality. For further discussion of this topic, also see recent work by Lake and Baroni (2017).

	Compound stats		PP stats	
	Ratio	Types-2	Ratio	Types-1
Liu et al. 2017	0.33	122	1.86	1145
Mun et al. 2017	0.33	300	1.74	2423
Shetty et al. 2016	0.30	319	1.65	2426
Tavakoli et al. 2017	0.33	259	1.72	1888
Vinyals et al. 2017	0.39	275	1.74	1678
Wu et al. 2016	0.34	237	1.69	1732
Zhou et al. 2017	0.34	472	1.71	3451
Dai et al. 2017	0.37	2576	1.78	11709
Shetty et al. 2017	0.42	1446	1.58	8439
Validation data	0.47	6089	1.74	22237

Table 3: Statistics for nominal compounds and prepositional phrases. Compound ratio corresponds to the number of compounds per description. Types-2 refers to the number of compound types of length 2. Preposition ratio corresponds to the number of prepositional phrases per description. Types-1 refers to the number of PP types of depth 1.

5 Discussion and Future Research

5.1 Other metrics

In addition to the metrics proposed in this paper, there are other options that could be explored in future work. For reasons of space, we were also not able to cover metrics based on the **frequency distribution** of words in the training and validation data. We already mentioned Shetty et al.'s (2017) use of frequency ratios in the introduction. Their approach could be extended (perhaps also using log-likelihood (Rayson and Garside, 2000)) to produce a ranking of words that are over- or underused by a particular system. Overused words could be further analyzed by computing a '**local precision**' metric, measuring how often a generated word is also used in at least one reference description. Ferraro et al. (2015) present other metrics in their survey of datasets for vision and language research, including:

Yngve and Frazier measurements of syntactic complexity (Yngve, 1960; Frazier, 1985). Ferraro et al. (2015) found that the MS COCO and Flickr30K datasets have the most complex sentences, compared to other vision & language datasets. It is still an open question whether machine-generated descriptions are of equal complexity and, if not, what are the differences.

Abstract-to-concrete ratio The authors also compare the proportion of abstract words that each corpus contains. They count abstract words by using a list of abstract words compiled in earlier work. In the literature, there are two definitions of abstractness and concreteness. Concrete words are either said to be (1) more closely tied to perception, or (2) more specific (Spreen and Schulz, 1966; Theijssen et al., 2011). It is unclear which is meant by Ferraro et al., but it would be interesting to see whether machine-generated descriptions are more closely tied to perception than human descriptions, who also speculate about the context of the images (van Miltenburg, 2016).

Part-of-speech distribution Ferraro et al. (2015) compared the distribution of *nouns, verbs, adjectives*, and *other* parts of speech. Our work on detecting prepositional phrases and compound nouns (Section 4) suggests that differences in the distribution of parts of speech between human- and machine-generated descriptions could be an interesting avenue to explore.

Besides the measures discussed above, it may also be interesting to study some types of linguistic phenomena in more detail. For example, van Miltenburg et al. (2016) provide a thorough overview of the uses of *negations* in human-generated image descriptions. Even though this is a low-frequent (or long-tail) phenomenon, studying a subset of the image descriptions informs us about the human image description process, and the *cognitive requirements* to produce a description containing a negation. It remains to be seen whether image description systems could produce similar descriptions.

5.2 Limitations and human validation

Earlier work has shown that automated evaluation metrics do not correlate well with human judgments (Elliott and Keller, 2014; Kilickaya et al., 2017). For this reason, we should not blindly trust evaluation metrics in their assessment of system performance. Still, this paper only includes automatic, intrinsic metrics. This is by design: we want to gain insight into the descriptions, not to evaluate their quality.

While you cannot evaluate a system using only automated metrics, they do tell us something about how a system behaves. Future researchers could try to improve the diversity metrics while maintaining or improving the quality of the descriptions (ideally measured by human judgments). At that point, we should determine if more diverse descriptions (as measured by the metrics covered in this paper) are perceived by humans as more interesting to read. One issue is that it is unclear how human judgments could be used to rate the diversity of the generated descriptions, because diversity is a *global* property of the data. In other words: you cannot judge the diversity of a single description, because that is not what diversity is about. You can only judge the diversity of a larger collection of descriptions. One way to do this might be to generate descriptions for sets of very similar images, and have participants rate the diversity of different batches of descriptions.

6 Conclusion

We explored several metrics to analyze the richness of computer-generated image descriptions, most of which focus on diversity at the word level. In our analysis of the output of nine state-of-the-art systems, we found that there are clear differences between human and system output: humans produce more word types; more different types when averaged over multiple 1000-token samples; more compound nouns per description; more long compound nouns; and more compound noun types than image description systems. Not all of these observations hold for prepositional phrases: humans *don't* produce more prepositional phrases per description, and neither do they produce more embedded prepositional phrases, however, they *do* produce a larger number of different prepositional phrases than the systems. At the sentence level, we found that humans produce longer descriptions, vary more in their description length, and produce more novel descriptions. We also found that GAN-based systems produce more diverse descriptions than MLE-based systems. However, we caution that the GAN-based systems are the only ones in our evaluation that are designed with diversity in mind. Further research is needed to find out what kind of approach is best for producing diverse descriptions.

We also proposed to frame image description as a word recall task to further explore the differences highlighted above. *Global recall* looks at the types from all the validation data that are learnable from the training data. *Local recall* measures whether systems are able to produce content words that are mentioned in n reference descriptions for a single image. These metrics show that there is plenty of room for improvement, both in terms of vocabulary size, as well as using the right words at the right time. One way to approach this challenge is by *ranking* terms that are often missed by a system, and looking for ways to learn when to use these words.

We provide all the code and data to to apply the metrics discussed in this paper and compare systems. We encourage readers to use this overview to start exploring the output of their own image description systems, but note that the metrics covered here are just the tip of the iceberg. As more researchers focus on producing more diverse descriptions, we will hopefully also develop a better understanding of what makes a description human-like. Formalizing these notions enables us to measure our progress towards richer and more diverse descriptions.

Acknowledgements

This research is funded through the 2013 NWO Spinoza prize, awarded to Piek Vossen. Desmond Elliott is supported by an Amazon Research Award.

References

Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. 2016. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *Journal of Artificial Intelligence Research*, 55:409–442.

Thiago Castro Ferreira, Emiel Krahmer, and Sander Wubben. 2016. Towards more variation in text generation: Developing and evaluating variation models for choice of referential form. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 568–577, Berlin, Germany.

Álvaro Corral, Gemma Boleda, and Ramon Ferrer-i Cancho. 2015. Zipf's law for word frequencies: word forms versus lemmas in long texts. *PloS one*, 10(7):e0129031.

Bo Dai, Sanja Fidler, Raquel Urtasun, and Dahua Lin. 2017. Towards diverse and natural image descriptions via a conditional gan. In *Proceedings of the 2017 International Conference on Computer Vision*, pages 2970–2979, Venice, Italy.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong He, Geoffrey Zweig, and Margaret Mitchell. 2015. Language models for image captioning: The quirks and what works. In *Proceedings of the 53rd*

Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 100–105, Beijing, China.

Desmond Elliott and Frank Keller. 2014. Comparing automatic evaluation measures for image description. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 452–457, Baltimore, Maryland, June. Association for Computational Linguistics.

Francis Ferraro, Nasrin Mostafazadeh, Ting-Hao Huang, Lucy Vanderwende, Jacob Devlin, Michel Galley, and Margaret Mitchell. 2015. A survey of current datasets for vision and language research. In *EMNLP*, pages 207–213, Lisbon, Portugal.

Lyn Frazier. 1985. Syntactic complexity. In D. R. Dowty, L. Karttunen, and A. M. Zwicky, editors, *Natural language parsing: Psychological, computational, and theoretical perspectives*, pages 129–189. Cambridge University Press, Cambridge.

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

Wendell Johnson. 1944. I. a program of research. *Psychological Monographs*, 56(2):1.

Mert Kilickaya, Aykut Erdem, Nazli Ikizler-Cinbis, and Erkut Erdem. 2017. Re-evaluating automatic metrics for image captioning. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 199–209, Valencia, Spain, April. Association for Computational Linguistics.

Brenden M Lake and Marco Baroni. 2017. Still not systematic after all these years: On the compositional skills of sequence-to-sequence recurrent networks. *arXiv preprint arXiv:1711.00350*. Under review at ICLR 2018.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL:HLT*, pages 110–119, San Diego, California, June. ACL.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer.

Chang Liu, Fuchun Sun, Changhu Wang, Feng Wang, and Alan Yuille. 2017. Mat: A multimodal attentive translator for image captioning. In *IJCAI*, pages 4033–4039.

Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv:1411.1784*.

Jonghwan Mun, Minsu Cho, and Bohyung Han. 2017. Text-guided attention model for image captioning. In *AAAI Conference on Artificial Intelligence*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Marten Postma, Ruben Izquierdo, Eneko Agirre, German Rigau, and Piek Vossen. 2016. Addressing the mfs bias in wsd systems. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Paul Rayson and Roger Garside. 2000. Comparing corpora using frequency profiling. In *Proceedings of the Workshop on Comparing Corpora - Volume 9*, pages 1–6.

Rakshith Shetty, Hamed R.-Tavakoli, and Jorma Laaksonen. 2016. Exploiting scene context for image captioning. In *Proceedings of the 2016 ACM Workshop on Vision and Language Integration Meets Multimedia Fusion*, iV&L-MM '16, pages 1–8, New York, NY, USA. ACM.

Rakshith Shetty, Marcus Rohrbach, Lisa Anne Hendricks, Mario Fritz, and Bernt Schiele. 2017. Speaking the same language: Matching machine to human captions by adversarial training. In *ICCV*, Oct.

Otfried Spreen and Rudolph W Schulz. 1966. Parameters of abstraction, meaningfulness, and pronunciability for 329 nouns. *Journal of Verbal Learning and Verbal Behavior*, 5(5):459–468.

Zoltán Gendler Szabó. 2017. Compositionality. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2017 edition.

Hamed R Tavakoli, Rakshith Shetty, Ali Borji, and Jorma Laaksonen. 2017. Paying attention to descriptions generated by image captioning models. In *CVPR*, pages 2487–2496.

DL Theijssen, H van Halteren, LWJ Boves, and NHJ Oostdijk. 2011. On the difficulty of making concreteness concrete. *Computational Linguistics in the Netherlands Journal*, 1:61–77, December.

Walter JB Van Heuven, Pawel Mandera, Emmanuel Keuleers, and Marc Brysbaert. 2014. Subtlex-uk: A new and improved word frequency database for british english. *The Quarterly Journal of Experimental Psychology*, 67(6):1176–1190.

Emiel van Miltenburg, Roser Morante, and Desmond Elliott. 2016. Pragmatic factors in image description: The case of negations. In *Proceedings of the 5th Workshop on Vision and Language*, pages 54–59, Berlin, Germany, August. ACL.

Emiel van Miltenburg. 2016. Stereotyping and bias in the flickr30k dataset. In Jens Edlund, Dirk Heylen, and Patrizia Paggio, editors, *Proceedings of Multimodal Corpora: Computer vision and language processing (MMC 2016)*, pages 1–4.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2017. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):652–663, April.

Zhuhao Wang, Fei Wu, Weiming Lu, Jun Xiao, Xi Li, Zitong Zhang, and Yueting Zhuang. 2016. Diverse image captioning via grouptalk. In *IJCAI*, pages 2957–2964.

Qi Wu, Chunhua Shen, Peng Wang, Anthony Dick, and Anton van den Hengel. 2017. Image captioning and visual question answering based on attributes and external knowledge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Victor H Yngve. 1960. A model and an hypothesis for language structure. *Proceedings of the American philosophical society*, 104(5):444–466.

Gilbert Youmans. 1990. Measuring lexical style and competence: The type-token vocabulary curve. *Style*, pages 584–599.

George Kingsley Zipf. 1949. *Human behaviour and the principle of least effort: an introduction to human ecology.* Cambridge, MA: Addison-Wesley.

Extractive Headline Generation Based on Learning to Rank for Community Question Answering

Tatsuru Higurashi[†] **Hayato Kobayashi**[†‡] **Takeshi Masuyama**[†] **Kazuma Murao**[†]
[†]Yahoo Japan Corporation [‡]RIKEN AIP
{thiguras,hakobaya,tamasuya,kmurao}@yahoo-corp.jp

Abstract

User-generated content such as the questions on community question answering (CQA) forums does not always come with appropriate headlines, in contrast to the news articles used in various headline generation tasks. In such cases, we cannot use paired supervised data, e.g., pairs of articles and headlines, to learn a headline generation model. To overcome this problem, we propose an extractive headline generation method based on learning to rank for CQA that extracts the most informative substring from each question as its headline. Experimental results show that our method outperforms several baselines, including a prefix-based method, which is widely used in real services.

1 Introduction

Community question answering (CQA) is a service where users can post their questions and answer the questions from other users. The quality of a CQA service is inevitably linked to how many questions are answered in a short amount of time. To this end, the headline of a posted question plays a key role, as it is the first thing users see in a question list or push notification on a smartphone. However, questions in a CQA service do not always have appropriate headlines because the questions are written by various users who basically do not have any specialized knowledge in terms of writing such content, in contrast to news articles written by professional editors. In fact, the biggest CQA service in Japan, Yahoo! Chiebukuro[1], does not even provide an input field for headlines in the submission form of questions, as general users do not have enough patience and tend not to post questions if even just one required field is added to the form. This service alternatively uses the prefix of a question as its headline as in Figure 1(a), where the headline "はじめまして。よろしくお願いします。仕事…(*Nice to meet you. Thank you in advance. At work ...*)" is created from the prefix of the content. Obviously, this headline is uninformative because

(a) Example of posted question. (b) Example of push notification.

Figure 1: Examples of (a) posted question and (b) push notification.

[1]https://chiebukuro.yahoo.co.jp/

Proceedings of the 27th International Conference on Computational Linguistics, pages 1742–1753
Santa Fe, New Mexico, USA, August 20-26, 2018.

of the lack of actual content, which is related to an initial unrequited love of a woman in the workplace. Figure 1(b) shows how ineffective such an uninformative headline is for a smartphone push notification, where users would have practically zero motivation to click on it and answer since they cannot imagine what kind of question is being asked. This negative effect has been confirmed on a commercial CQA service (Section 2).

In this work, we take an extractive approach for improving uninformative headlines. Although there have recently been many studies on abstractive headline generation (as described in Section 6), we do not follow any of these approaches because the content we deal with has no correct headlines and also the abstractive methods can yield erroneous output. This latter issue is important in practical terms because correct output is critical for a commercial service. If by some chance an erroneous headline, or one including politically incorrect phrases, is sent to all users, the service's credibility can be lost instantly. Therefore, we formalize our task as an extraction problem of a fixed-length substring from a question as its headline. In this setting, we can assume that outputs never include such errors caused by the service, as the outputs are substrings of user-posted questions. Note that they have no coherence errors in selecting multiple sentences as in normal extractive summarization tasks. While it is true that these outputs might contain inappropriate expressions authored by users, this type of error is beyond our scope since it is a different inevitable problem. Furthermore, the situation of "the service generated an inappropriate expression by itself" is significantly worse than the situation of "a user posted an inappropriate question to the service, and the service displayed it". Therefore, it is difficult to directly use abstractive methods for commercial services from a business standpoint.

Our approach involves preparing headline candidates and ranking them. The formal description is as follows. Let q be a target question to be translated into a headline. We first prepare a set $S(q)$ of headline candidates from the question q. Note that the set $S(q)$ is restricted to a set of fixed-length substrings given a length n, i.e., $S(q) \subseteq \{x \mid x \preceq q, |x| = n\}$, where $x \preceq q$ means that x is a substring of q. Then we extract the best headline that maximizes a score function $f_q(x)$, which represents the "headline-ness" score of a candidate $x \in S(q)$ with respect to the target question q, as follows:

$$\mathrm{argmax}_{x \in S(q)} f_q(x). \tag{1}$$

To ensure simplicity of implementation and understandability from users, we use a set of the fixed-length prefixes of all sentences in a question q as the candidate set $S(q)$ (Section 3). Because the problem is to select the best candidate from among several targets, the score function $f_q(x)$ is naturally trained by learning to rank (Section 4).

The main contributions of this paper are as follows.

- We report empirical evidence of the negative effect of uninformative headlines on a commercial CQA service (Section 2). We additionally show that our task can reduce uninformative headlines by using simple dictionary matching, which dramatically improves the average answer rate of questions by as much as 2.4 times.
- We propose an extractive headline generation method based on learning to rank for CQA (Section 4) that extracts the most informative substring (prefix of mid-sentence) from each question as its headline. To the best of our knowledge, our work is the first attempt to address such a task from a practical standpoint, although there have been many related studies (Section 6). Experimental results show that our method outperforms several baselines, including the dictionary-based method (Section 5).
- We create a dataset for our headline generation task (Section 3), where headline candidates extracted from questions are ranked by crowdsourcing with respect to "headline-ness", that is, whether or not each headline candidate is appropriate for the headline of the corresponding question.

2 Negative Effect of Uninformative Headlines

We conducted A/B testing on the push notifications of smartphones in collaboration with Yahoo! Chiebukuro, as shown in Figure 1(b). We first prepared a dictionary of typical first sentences that cause uninformative headlines. This dictionary was manually selected from frequent first sentences and consists of 913 sentences including greetings such as "おはようございます (Good morning)", "こんにち

Changed uninformative	Unchanged uninformative	Informative
0.75%	0.31%	0.45%

Table 1: Average answer rates of three question groups in A/B testing.

は (Good afternoon)", and "はじめまして (Nice to meet you)", and fixed phrases such as "質問させてください (Can I ask you something)", "教えてください (Please tell me)", and "よろしくお願いします (Thank you in advance)". We assumed that a question with a prefix match in the dictionary has an uninformative prefix headline and classified such questions into an *uninformative group*. For convenience, we also classified the other questions into an *informative group*, although they might include not so informative headlines. We further randomly divided the uninformative group into two equal groups: *changed* and *unchanged*. In the changed uninformative group, each headline is extracted as the prefix of the first (informative) sentence that does not match with the dictionary, which is the same as `DictDel` explained in Section 5.2. The unchanged group remains in uninformative. For comparison of these groups, we used the average answer rate over notified questions in each group as an evaluation measure, defined as

$$\text{Average answer rate} = \frac{\text{No. of questions answered from the notification}}{\text{No. of notified questions}}. \quad (2)$$

Note that we use a percentage expression (%) for easy reading.

Table 1 shows the evaluation results of the A/B testing during a 1-month period (Feb. 2 – Mar. 4, 2018), where about three million questions were sent to users. Comparing the unchanged uninformative group with the informative group, we can see that the average answer rate of the uninformative questions, 0.31%, is actually lower than that of the informative questions, 0.45%. Comparing the changed and unchanged uninformative groups, the average answer rate of the changed questions, 0.75%, is much higher than that of the unchanged questions, 0.31%. This means that even a simple dictionary-based method can dramatically improve the quality of the uninformative headlines, i.e., by as much as 2.4 times. We confirmed that the difference is statistically significant on a one-tailed Wilcoxon signed-rank test ($p < 0.05$). Note that the average answer rate represents a conversion rate (or rate of target actions), which is more important than a click-through rate (or rate of initial actions). The average answer rate is one of the most important indicators for a CQA service, while the click-through rate can be meaninglessly high if the service sends headlines that are fake or too catchy. We should point out that low answer rates are sufficient for the service, since it has 44M users: i.e., each question has an average of 2.4 answers, as the service currently has 189M questions and 462M answers.

3 Dataset Creation

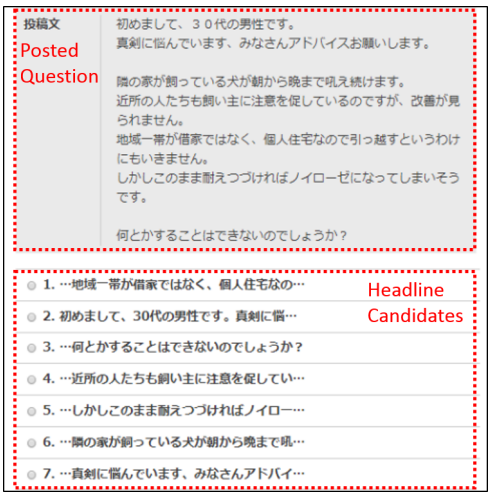
We created a dataset for our headline generation task based on the Yahoo! Chiebukuro dataset[2], which is a dataset including questions and answers provided from a Japanese CQA service, Yahoo! Chiebukuro. We first prepared headline candidates from this dataset as in Section 3.1 and then conducted a crowdsourcing task specified in Section 3.2. In Section 3.3, we report the results of the crowdsourcing task.

3.1 Preparation of Headline Candidates

We extracted only questions from the Chiebukuro dataset and split each question into sentences by using punctuation marks (i.e., the exclamation ("！"), question ("？"), and full stop ("。") marks). We regarded 20 Japanese characters that are basically extracted from each sentence as a headline candidate $x \in S(q)$ in Eq. (1), since this setting is used for push notifications in the actual service in Figure 1(b). More specifically, the headline candidate is created as follows:

1. If the sentence is the first one in the question, we extract the first 19 characters and put an ellipsis mark ("…") at the end.
2. Otherwise, we extract the first 18 characters and put ellipses at both the beginning and the end. This expression explicitly represents being mid-sentence so as to avoid weird headlines.

[2] `http://www.nii.ac.jp/dsc/idr/en/yahoo/yahoo.html`



(a) Example of our crowdsourcing task.

Posted Question:
Nice to meet you, I am a man in my 30s.
Please give me your advice on a pressing concern I have.

A dog kept in the next house barks from morning to night.
Neighbors have given the owner cautions against it, but there is no improvement.
This area has only private houses, not rented houses, so I cannot move out.
However, I will go crazy if I have to keep enduring this.

How can I effectively manage this problem?

Headline Candidates:
1. ... This area has only private houses, not rented ...
2. Nice to meet you, I am a man in my 30s. Please ...
3. ... How can I effectively manage this problem?
4. ... Neighbors have given the owner cautions against ...
5. ... However, I will go crazy if I have to keep ...
6. ... A dog kept in the next house barks from morning ...
7. ... Please give me your advice on a pressing concern ...

(b) English translation of left example.

Figure 2: Examples of (a) our crowdsourcing task and (b) its English translation.

In the case where the length of a candidate is less than 20 characters, we include some of the next sentence in order to maximize use of display space. We included questions with more than five sentences for the purpose of efficiently collecting ranking information. All told, we prepared 10,000 questions containing more than five headline candidates each.

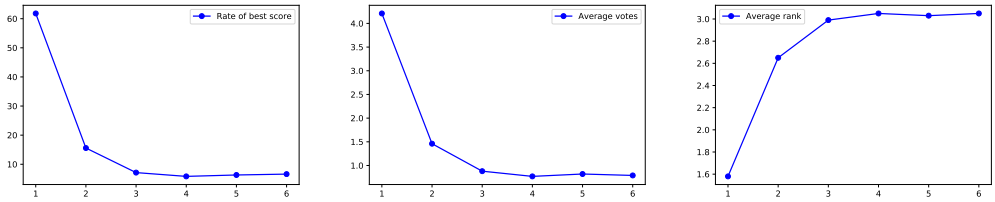
3.2 Crowdsourcing Task

Figure 2 shows an example of our crowdsourcing task (a) and its English translation (b). This task involves a posted question and headline candidates corresponding to the question. We asked workers to select the best candidate from options after reading the posted question. A relative evaluation, where workers select the best candidate, was used instead of an absolute evaluation, where workers select a score from 0 to 10 for each candidate, because we wanted to obtain as accurate a headline as possible, and it might be difficult for workers to select an appropriate absolute score. The workers were instructed as follows (English translation):

> Various candidate headlines are listed as options for a posted question in a Q&A service. After reading the question, please select the best option from the list so that users can guess the content of the question and distinguish it from other ones. Please remove uninformative ones such as greetings, self-introductions, and unspecific expressions.

We explain how to judge the appropriateness of each candidate by means of the example in Figure 2. After examining the posted question, we can assume that the most important content is "he is annoyed by the barking of a dog kept in the next house". On the basis of this assumption, option 6 is the best one, since the headline "A dog kept in the next house barks from morning" is enough to help answerers deduce that "the questioner is annoyed by the sound". Option 1 is inappropriate because although the answerers might be able to guess that "the questioner cannot move out", this matter is not the central one. Option 2 is uninformative because it consists merely of greetings and self-introduction, and option 3, while a question sentence, is unspecific. Option 4 enables answerers to guess that this question is related to an issue involving pets, but they cannot grasp the specific content. Option 5 specifies a likely damage due to the trouble, but the reason (trouble) is more important for the answering than the result (damage). Option 7 directly shows that "the questioner is annoyed and wants some advice", but answerers cannot understand why he/she is annoyed.

The detailed implementation of our task is as follows. First we randomly sorted the candidates of each question (shown in Figure 2(a)) to avoid position bias by the workers. We included ten actual questions and a dummy question so that workers would always have to answer one dummy question per every ten actual questions. A dummy question is a question with a clear answer inserted to eliminate fraud



(a) Ratio of true best candidates. (b) Average votes for position. (c) Average rank for position

Figure 3: Ratio of true best candidates and average votes/rank (for position) versus sentence position.

workers (i.e., workers who randomly select answers without actually reading them). Each question was answered by ten workers so each headline candidate had a vote score from 0 to 10 representing whether or not the candidate was appropriate for a headline. This task took nine days and was answered by 1,558 workers. As a result, our dataset consists of 10,000 questions, each of which has more than five headline candidates with accompanying "headline-ness" scores.

3.3 Analysis of Crowdsourcing Results

We analyzed our dataset to determine how much room for improvement our task has compared to the prefix headline. It is well known that the first sentence can be a strong baseline in many summarization tasks, so the prefix headline is also expected to perform well. Figure 3 shows the statistical information based on sentence position that includes the (a) ratio of the true best (most voted) candidates, (b) average votes, and (c) average rank (in order of votes) over the candidates at each sentence position. Looking at the true best candidates (a), the ratio 61.8% for the 1st sentence clarifies the effectiveness of the prefix headline, as expected. Conversely, we still have room for improvement for the prefix headline up to 38.2%. Our goal is to improve the uninformative headlines of 38.2% while keeping the remaining 61.8% unchanged. The other two figures (b) and (c) also support the above discussion.

Furthermore, we qualitatively checked the crowdsourcing quality. Workers successfully eliminated uninformative candidates including greetings and self-introductions, while one or two workers sometimes chose ones that included a fixed phrase such as "Please tell me ...". This is probably because workers had different criteria regarding the "unspecific expressions" described in the instructions. Since we cannot enumerate all of the concrete bad examples, we ignore this phenomenon with the expectation that a learning algorithm will reduce its negative effect.

4 Proposed Method

In this section, we explain how to construct a headline generation model from the dataset presented in Section 3. We took a ranking approach, i.e., learning to rank, for our task, rather than a simple regression one, since estimating absolute scores is not required for our purpose. Even if two headline candidates (of different questions) have the same expression, their votes can be significantly different since the votes in our dataset are based on relative evaluation. For example, the best candidate for Figure 2 was No. 6, but it might not be selected in other questions such as "A dog kept in the next house barks from morning. Does anybody know why dogs generally want to bark?".

Learning to rank is an application of machine learning that is typically used for ranking models in information retrieval systems. The ranking models are basically learned from a supervised dataset consisting of triples (q, x, y), where q is a user's query, x is a document corresponding to q, and y is a relevance score of x with respect to q. In this work, we formalize our task by regarding q, x, and y, as a posted question, a headline candidate, and a voted score in our dataset, respectively.

We used a pairwise ranking method that is also implemented as an instance of the well-known SVM tools LIBLINEAR (Lee and Lin, 2014) and LIBSVM (Kuo et al., 2014). We used a linear model based on LIBLINEAR, an L2-regularized L2-loss linear rankSVM, for the experiments. Let D be a dataset that consists of triples including a posted question q, a headline candidate x, and a voted score y, i.e., $(q, x, y) \in D$. In the pairwise ranking method, we train a ranking model as a binary classifier that determines whether the condition $y_i > y_j$ is true or false for two candidates x_i and x_j in the same

question ($q_i = q_j$). Specifically, we first define the index pairs of positive examples by $P = \{(i,j) \mid q_i = q_j, y_i > y_j, (x_i, y_i, q_i) \in D, (x_j, y_j, q_j) \in D\}$. Note that we do not need to consider the negative examples $N = \{(j,i) \mid (i,j) \in P\}$ since they yield the same formula as P in the optimization process. The training of the pairwise ranking method is achieved by solving the following optimization problem using the set P of the index pairs:

$$\min_w \ \frac{1}{2} w^\top w + C \sum_{(i,j)\in P} \ell(w^\top \tilde{x}_i - w^\top \tilde{x}_j), \tag{3}$$

where w is a weight vector to be learned, $\tilde{x}_i$ is a feature vector extracted from a headline candidate x, and C is the regularization parameter. The function ℓ is a squared hinge loss, which is defined as $\ell(d) = \max(0, 1-d)^2$. Finally, we define the score function in Eq. (1) as $f_q(x) = w^\top \tilde{x}$, where $\tilde{x}$ can be created by using q as well as x. This score means the relative "headline-ness" of x.

5 Experiments

5.1 Basic Settings

The basic settings of the experiments are as follows. We split our dataset into training and test sets consisting of 9,000 and 1,000 examples, respectively. We used an implementation[3] based on LIBLINEAR for training our ranking model, i.e., a linear L2-regularized L2-loss rankSVM model (Lee and Lin, 2014), as described in Section 4. The regularization parameter was optimized by cross validation and set as $C = 0.125$.

The feature vector for a headline candidate consists of three kinds of features: bag-of-words, embedding, and position information. The bag-of-words feature is a sparse vector of 30,820 dimensions based on the tf-idf scores of nouns, verbs, interjections, conjunctions, adverbs, and adjectives in a candidate, where we used a Japanese morphological analyzer, MeCab[4] (Kudo et al., 2004), with a neologism dictionary, NEologd[5] (Toshinori Sato and Okumura, 2017). The embedding feature is a dense vector of 100 dimensions based on a doc2vec model (Le and Mikolov, 2014) trained with all 3M sentences in the Chiebukuro dataset using the Gensim tool[6]. The position feature is a binary vector of ten dimensions, where each dimension represents the coarse position (or coverage) of a headline candidate for a question. Specifically, we equally split a question (character sequence) into ten parts and set one to each dimension if and only if the corresponding part overlaps a candidate. For example, candidate No. 2 in Figure 2 had a position feature $(1, 1, 0, \cdots, 0)$, since the candidate covers the first 2/10 of the whole question. Similarly, No. 6 and No. 3 had $(0, 0, 1, 1, 0, \cdots, 0)$ and $(0, \cdots, 0, 1)$, respectively. For constructing the feature vector of each headline candidate, we used the previous and next candidates in sentence order, in addition to the target candidate. This is based on the idea that near candidates might have useful information for the target candidate. Finally, we prepared each feature vector by concatenating nine feature vectors, i.e., the above three kinds of features for three candidates, and normalizing them.

5.2 Compared Methods

We compared our method, `MLRank`, with the baselines listed below. `Prefix`, `DictDel`, and `Random` are simple baselines, while `Prefix` and `DictDel` are practically strong. `ImpTfidf`, `SimTfidf`, `SimEmb`, and `LexRank` are unsupervised baselines, and `SVM` and `SVR` are supervised ones.

- `Prefix`: Selects the first candidate in sentence order.
- `DictDel`: Selects the first (informative) candidate that does not match in the dictionary of uninformative headlines (Section 2).
- `Random`: Randomly selects a candidate.
- `ImpTfidf`: Selects the most important candidate with the highest tf-idf value, where a tf-idf value is calculated by the sum of the elements in a bag-of-words feature (described in Section 5.1).

[3] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/#large_scale_ranksvm`
[4] `https://taku910.github.io/mecab/`
[5] `https://github.com/neologd/mecab-ipadic-neologd`
[6] `https://radimrehurek.com/gensim/models/doc2vec.html`

- `SimTfidf`: Selects the most similar candidate to the original question, which is calculated by the cosine similarity between the bag-of-words features (in Section 5.1) of each candidate and the question.
- `SimEmb`: An embedding-based variation of `SimTfidf` with embedding features (in Section 5.1).
- `LexRank`: Selects the candidate with the highest score based on LexRank[7] (Erkan and Radev, 2004), which is a widely used unsupervised extractive summarization method based on the PageRank algorithm. The graph expression of each question was constructed on the basis of cosine similarity of the tf-idf vectors corresponding to candidates.
- `SVM`: Selects the candidate with the highest confidence based on a model learned as a classification task, where candidates with nonzero votes were labeled as positive. This setting was the best in our preliminary experiments. We used the L2-regularized L2-loss support vector classification model ($C = 0.0156$) in LIBLINEAR. The other settings were the same as those described in Section 5.1.
- `SVR`: Selects the candidate with the highest predicted votes based on a model learned as a regression task, where the target variable is the number of votes. We used the L2-regularized L2-loss support vector regression model ($C = 0.0625$) in LIBLINEAR. The other settings were the same as above.
- `MLRank`: Proposed method described in Section 4.

5.3 Evaluation Measures

We defined three evaluation measures for evaluating each method on our headline generation task.

Change Rate from Prefix Headline

We measured how much each method changed the default prefix headline in order to determine the effect of application to an actual CQA service. We defined this measure as *change rate from the prefix headline*, as follows:

$$\text{Change rate} = \frac{\text{No. of questions where the best candidate is not the prefix headline}}{\text{No. of all questions}}. \quad (4)$$

Clearly, the change rate of the default method that selects the prefix headline is 0%. If the value is small, the effect on the service will be small, but if the value is higher than the ideal change rate of 38.2% (Section 3), there can be side effects even if the average result is good. A higher change rate up to the ideal rate is desirable from a practical standpoint.

Winning Rate against Prefix Headline

We measured how much each method won against the prefix headline to directly assess the quality of changed headlines. We defined this measure as *winning rate against the prefix headline*, as follows:

$$\text{Winning rate} = \frac{\text{No. of questions where the best candidate got more votes than the prefix headline}}{\text{No. of questions where the best candidate is not the prefix headline}}. \quad (5)$$

We did not consider the first candidate (in sentence order) selected by each method, which is the same as the prefix headline, since they obviously have the same number of votes.

Average Votes

We measured how appropriate the candidates selected by each method are in order to determine the overall performance. We defined this measure as *average votes*, as follows:

$$\text{Average votes} = \frac{\text{Sum of votes for the best candidates for all questions}}{\text{No. of questions}}. \quad (6)$$

Note that the average votes score is different from the average votes score for position (Figure 3(b)) in that the former is the average over the selected candidates while the latter is the average over the candidates at a sentence position.

[7] `https://pypi.org/project/lexrank/0.0.1b0`

E.g.	Prefix method (Prefix)	Proposed method (MLRank)
1	27 歳女です。環境的になかなか新しい出… I am a 27-year-old woman. Owing to my environment, there is little chance of new ...	…環境的になかなか新しい出会いがなくて… ... Owing to my environment, there is little chance of new encounters with men ...
2	情緒不安定なものです。回答して下さった… I am emotionally unstable. Those who answered ...	…新婚旅行は、行きは別々で行き、現地で… ... For the honeymoon, we went separately and in the field ...
3	カテ違いならゴメンナサイ。今、財布が破… I am sorry if the category is wrong. Now, my wallet is torn	…今、財布が破れてツライ状況です。新し… ... Now, my wallet is torn, and I'm having a hard time. A new one ...
4	前にとても大変で嫌な思いをしたので携帯… Because things felt very hard and painful before, my mobile ...	…携帯メールお断りというのはこの落札者… ... Rejecting e-mails from a mobile phone means this winning bidder ...
5	現在、私の父は 60 歳で定年退職を迎えま… Currently my father is 60 years old and will retire from employment ...	…厚生年金の支払いは何歳までなんですか… ... Until what age should he pay the welfare pension ...

Table 2: Examples of prefix method Prefix and proposed method MLRank.

This measure is related to (normalized) discounted cumulative gain (DCG), which is widely used as an evaluation measure of ranking models. We often use DCG@k for evaluating top-k rankings, and the above definition acutally corresponds to DCG@1. According to a well-known paper (Järvelin and Kekäläinen, 2002) in the information retrieval field, DCG is appropriate for graded-relevance judgments like our task, while precision (described below) is appropriate for binary-relevance ones. Average votes is expected to be more appropriate than precision for our task because we want "averagely better headlines than default ones" rather than "best ones" from a practical standpoint.

Precision

Precision is a widely used evaluation measure for classification tasks, and we added it to support an evaluation based on average votes. We defined it with respect to the best candidate, i.e., precision@1, as follows:

$$\text{Precision} = \frac{\text{No. of questions where the best candidate had the maximum votes}}{\text{No. of questions}}. \quad (7)$$

5.4 Results

Qualitative Analysis

Table 2 shows examples of headlines generated by the prefix method, Prefix, and the proposed method, MLRank. Looking at the first example, we can see that our method successfully eliminated the self-introduction phrase ("I am a 27-year-old woman"). The headline (right) of our method allows answerers to know that the questioner is discouraged about how to encounter men from the phrase "little chance of new encounters with men", while the headline (left) of the prefix method lacks this important clue. The second and third examples show similar effects to the first example. In the second one, although there are few clues about what the question is with the prefix method, our method correctly included the important clue ("honeymoon"). In the third one, our method appropriately eliminated the uninformative long phrase ("I am sorry if the category is wrong"), which is not a frequent fixed phrase. The fourth example shows a slightly challenging case, where both headlines make it difficult to understand the question. However, the headline of our method included the term "winning bidder", so at least the answerer can assume that the question is about some sort of auction trouble. The fifth example is a clearly successful result, where our method extracted the main question point about "welfare pension" as a headline. These results qualitatively demonstrate the effectiveness of our method.

Quantitative Analysis

We compared our method MLRank with the baselines in Section 5.2 on the headline generation task for our dataset in Section 3. Table 3 shows the evaluation results based on the change rates, winning rates, average votes, and precision. Looking at the average votes and precision, which represent the overall performances, our method MLRank clearly performed the best among all methods. We confirmed that

	Change rate (%)	Winning rate (%)	Average votes	Precision (%)
Ref (reference)	38.2	100.0	5.56	100.0
Prefix (default)	0	–	4.19	61.8
DictDel	2.2	72.0	4.23	61.3
Random	85.9	11.7	1.39	16.1
ImpTfidf	81.1	12.7	1.68	20.0
SimTfidf	79.3	18.7	2.27	20.0
SimEmb	**88.2**	13.0	1.40	15.4
LexRank	55.7	19.3	2.27	29.9
SVM	16.7	50.1	4.09	60.3
SVR	52.5	25.7	3.00	42.1
MLRank (proposed)	9.9	**94.9**	**4.28**	**62.6**

Table 3: Evaluation results of our headline generation task for proposed method MLRank and baselines.

the relative improvement of the average votes of our method MLRank against every baseline including the prefix method Prefix is statistically significant on the basis of a one-tailed Wilcoxon signed-rank test ($p < 0.01$). The change and winning rates of our method are 9.9% and 94.9%, respectively. This means that our method detected 9.9% of the uninformative headlines and improved them with the high accuracy of 94.9%. In other words, our method could successfully improve the overall performance while simultaneously avoiding any negative side effects. The ideal results (Ref) based on correct labels suggest that our method still has room for improvement, especially for the change rate.

The results of the other baselines are as follows. Not surprisingly, the prefix method Prefix performed well. This is consistent with the fact that the first sentence can be a good summary in many summarization tasks. The random method Random performed the worst, also as expected. The dictionary-based deletion method DictDel was relatively useful, although the change rate was small. The reason the winning rate of DictDel is relatively low compared with MLRank is that there are some cases where a combination of uninformative expressions can yield likely clues. For example, the self-introduction "I am a newbie of this forum" itself is basically uninformative for a question, but a combination with additional information such as "I am a newbie of this forum. Where can I change the password ..." can be more informative than only the additional information "... Where can I change the password since I forgot it after ..." because the combination specifies the target site by the expression "this forum".

The unsupervised methods, SimTfidf, SimTfidf, SimEmb, and LexRank, which are widely used for summarization tasks, performed worse than the prefix method Prefix. Although the change rates are higher than our method, the winning rates are lower. In other words, they yielded many bad headlines. These results suggest that supervised learning specialized to the target task would be required. Comparing the important sentence extraction method ImpTfidf and the similarity-based summarization method SimTfidf, we found that SimTfidf performed better. This implies that the content information of each question is useful for our headline generation task, as is the case with other summarization tasks. The similarity-based method SimEmb with embeddings performed worse than our expectation. The reason seems to be that it was difficult to obtain meaningful document embeddings from long questions. The graph-based method LexRank had a similar performance to SimTfidf, because LexRank tends to select a candidate similar to the question when only one candidate was selected. The supervised methods, SVM and SVR, performed relatively well compared to the unsupervised methods, but they did not outperform the strong simple baselines, Prefix and DictDel, nor our method MLRank. These results support the appropriateness of our approach.

6 Related Work

In this section, we briefly explain several related studies from two aspects: headline generation task and CQA data. As discussed below, our work is the first attempt to address an extractive headline generation task for a CQA service based on learning to rank the substrings of a question.

After Rush et al. (2015) proposed a neural headline generation model, there have been many studies on the same headline generation task (Takase et al., 2016; Chopra et al., 2016; Kiyono et al., 2017; Zhou et al., 2017; Suzuki and Nagata, 2017; Ayana et al., 2017; Raffel et al., 2017). However, all of them are abstractive methods that can yield erroneous output, and the training for them requires a lot of

paired data, i.e., news articles and headlines. There have also been several classical studies based on non-neural approaches to headline generation (Woodsend et al., 2010; Alfonseca et al., 2013; Colmenares et al., 2015), but they basically addressed sentence compression after extracting important linguistic units such as phrases. In other words, their methods can still yield erroneous output, although they would be more controllable than neural models. One exception is the work of Alotaiby (2011), where fixed-sized substrings were considered for headline generation. Although that approach is similar to ours, Alotaiby only considered an unsupervised method based on similarity to the original text (almost the same as `SimTfidf` in Section 5.2), in contrast to our proposal based on learning to rank. This implies that Alotaiby's method will also not perform well for our task, as shown in Section 5.4. There have been several studies on extractive summarization (Kobayashi et al., 2015; Yogatama et al., 2015) based on sentence embeddings, but they were basically developed for extracting multiple sentences, which means that these methods are almost the same as `SimEmb` in Section 5.2 for our purpose, i.e., extraction of the best candidate. This also implies that they will not be suitable for our task. Furthermore, recent sophisticated neural models for extractive summarization (Cheng and Lapata, 2016; Nallapati et al., 2017) basically require large-scale paired data (e.g., article-headline) to automatically label candidates, as manual annotation is very costly. However, such paired data do not always exist for real applications, as in our task described in Section 1.

There have been many studies using CQA data, but most of them are different from our task, i.e., dealing with answering questions (Surdeanu et al., 2008; Celikyilmaz et al., 2009; Bhaskar, 2013; Nakov et al., 2017), retrieving similar questions (Lei et al., 2016; Romeo et al., 2016; Nakov et al., 2017), and generating questions (Heilman and Smith, 2010). Tamura et al. (2005) focused on extracting a core sentence and identifying the question type as classification tasks for answering multiple-sentence questions. Although their method is useful to retrieve important information, we cannot directly use it since our task requires shorter expressions for headlines than sentences. In addition, they used a support vector machine as a classifier, which is almost the same as `SVM` in Section 5.2, and it is not expected to be suitable for our task, as shown in Section 5.4. The work of Ishigaki et al. (2017) is the most related one in that they summarized lengthy questions by using both abstractive and extractive approaches. Their work is promising because our task is regarded as the construction of short summaries, but the training of their models requires a lot of paired data consisting of questions and their headlines, which means that their method cannot be used to our task.

7 Conclusion

We proposed an extractive headline generation method based on learning to rank for CQA that extracts the most informative substring in each question as its headline. We created a dataset for our task, where headline candidates in each question are ranked using crowdsourcing. Our method outperformed several baselines, including a prefix-based method, which is widely used for cases where the display area is limited, such as the push notifications on smartphones. The dataset created for our headline generation task will be made publicly available[8]. Although our task is basically designed for extractive summarization, this dataset can also be used for abstractive summarization as a side information for training abstractive models.

In future work, we will investigate how effectively our method can perform in practical situations, e.g., push notifications. In addition, we will consider how to improve the change rate of our method while keeping its winning rate and how to create a useful dataset even if removing the length limitation.

Acknowledgements

We would like to thank the engineers at Yahoo! Chiebukuro for supporting the A/B testing in Section 2 and the anonymous reviewers for giving us many useful comments.

[8] `https://research-lab.yahoo.co.jp/en/software/`

References

Enrique Alfonseca, Daniele Pighin, and Guillermo Garrido. 2013. HEADY: News headline abstraction through event pattern clustering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 1243–1253. Association for Computational Linguistics.

Fahad Alotaiby. 2011. Automatic Headline Generation using Character Cross-Correlation. In *Proceedings of the ACL 2011 Student Session*, pages 117–121. Association for Computational Linguistics.

Ayana, Shi-Qi Shen, Yan-Kai Lin, Cun-Chao Tu, Yu Zhao, Zhi-Yuan Liu, and Mao-Song Sun. 2017. Recent Advances on Neural Headline Generation. *Journal of Computer Science and Technology*, 32(4):768–784.

Pinaki Bhaskar. 2013. Answering Questions from Multiple Documents – the Role of Multi-Document Summarization. In *Proceedings of the Student Research Workshop associated with RANLP 2013*, pages 14–21. INCOMA Ltd. Shoumen, BULGARIA.

Asli Celikyilmaz, Marcus Thint, and Zhiheng Huang. 2009. A graph-based semi-supervised learning for question-answering. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 719–727, Suntec, Singapore, August. Association for Computational Linguistics.

Jianpeng Cheng and Mirella Lapata. 2016. Neural Summarization by Extracting Sentences and Words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 484–494. Association for Computational Linguistics.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pages 93–98. Association for Computational Linguistics.

Carlos A. Colmenares, Marina Litvak, Amin Mantrach, and Fabrizio Silvestri. 2015. HEADS: Headline Generation as Sequence Prediction Using an Abstract Feature-Rich Space. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2015)*, pages 133–142. Association for Computational Linguistics.

Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based Lexical Centrality As Salience in Text Summarization. *Journal of Artificial Intelligence Research (JAIR)*, 22(1):457–479.

Michael Heilman and Noah A. Smith. 2010. Good Question! Statistical Ranking for Question Generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2010)*, pages 609–617. Association for Computational Linguistics.

Tatsuya Ishigaki, Hiroya Takamura, and Manabu Okumura. 2017. Summarizing Lengthy Questions. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (IJCNLP 2017)*, pages 792–800. Asian Federation of Natural Language Processing.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

Shun Kiyono, Sho Takase, Jun Suzuki, Naoaki Okazaki, Kentaro Inui, and Masaaki Nagata. 2017. Source-side Prediction for Neural Headline Generation . *CoRR*, abs/1712.08302.

Hayato Kobayashi, Masaki Noguchi, and Taichi Yatsuka. 2015. Summarization based on embedding distributions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1984–1989, Lisbon, Portugal, September. Association for Computational Linguistics.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to japanese morphological analysis. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 230–237, Barcelona, Spain, July. Association for Computational Linguistics.

Tzu-Ming Kuo, Ching-Pei Lee, and Chih-Jen Lin. 2014. Large-scale Kernel RankSVM. In *Proceedings of the 2014 SIAM International Conference on Data Mining (SDM 2014)*, pages 812–820.

Quoc Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning (ICML 2014)*, pages 1188–1196. JMLR.org.

Ching-Pei Lee and Chih-Jen Lin. 2014. Large-scale Linear Ranksvm. *Neural Computation*, 26(4):781–817.

Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi Jaakkola, Kateryna Tymoshenko, Alessandro Moschitti, and Lluís Màrquez. 2016. Semi-supervised Question Retrieval with Gated Convolutions. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pages 1279–1289. Association for Computational Linguistics.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. SemEval-2017 Task 3: Community Question Answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 27–48. Association for Computational Linguistics.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A Recurrent Neural Network Based Sequence Model for Extractive Summarization of Documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017)*, pages 3075–3081. AAAI Press.

Colin Raffel, Minh-Thang Luong, Peter J. Liu, Ron J. Weiss, and Douglas Eck. 2017. Online and Linear-Time Attention by Enforcing Monotonic Alignments. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, pages 2837–2846.

Salvatore Romeo, Giovanni Da San Martino, Alberto Barrón-Cedeño, Alessandro Moschitti, Yonatan Belinkov, Wei-Ning Hsu, Yu Zhang, Mitra Mohtarami, and James Glass. 2016. Neural Attention for Learning to Rank Questions in Community Question Answering. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016)*, pages 1734–1745. The COLING 2016 Organizing Committee.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 379–389. Association for Computational Linguistics.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2008. Learning to Rank Answers on Large Online QA Collections. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2008)*, pages 719–727. Association for Computational Linguistics.

Jun Suzuki and Masaaki Nagata. 2017. Cutting-off redundant repeating generations for neural abstractive summarization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 291–297. Association for Computational Linguistics.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural Headline Generation on Abstract Meaning Representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 1054–1059. Association for Computational Linguistics.

Akihiro Tamura, Hiroya Takamura, and Manabu Okumura. 2005. Classification of Multiple-Sentence Questions. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP 2005)*, pages 426–437. Springer-Verlag Berlin Heidelberg.

Taiichi Hashimoto Toshinori Sato and Manabu Okumura. 2017. Implementation of a word segmentation dictionary called mecab-ipadic-neologd and study on how to use it effectively for information retrieval (in japanese). In *Proceedings of the Twenty-three Annual Meeting of the Association for Natural Language Processing*, pages NLP2017–B6–1. The Association for Natural Language Processing.

Kristian Woodsend, Yansong Feng, and Mirella Lapata. 2010. Title Generation with Quasi-Synchronous Grammar. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, pages 513–523. Association for Computational Linguistics.

Dani Yogatama, Fei Liu, and Noah A. Smith. 2015. Extractive Summarization by Maximizing Semantic Volume. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1961–1966. Association for Computational Linguistics.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1095–1104. Association for Computational Linguistics.

A Multi-Attention based Neural Network with External Knowledge for Story Ending Predicting Task

Qian Li[1], Ziwei Li[1], Jin-Mao Wei[1], Yanhui Gu[2], Adam Jatowt[3], Zhenglu Yang[1]
CCCE, Nankai University,China[1]
School of CS and Technology, Nanjing Normal University, China[2]
Graduate School of Informatics, Kyoto University, Japan[3]
{liqian515,lzw_nku}@mail.nankai.edu.cn, weijm@nankai.edu.cn,
gu@jnu.edu.cn, adam@dl.kuis.kyoto-u.ac.jp,
yangzl@nankai.edu.cn

Abstract

Enabling a mechanism to understand a temporal story and predict its ending is an interesting issue that has attracted considerable attention, as in case of the ROC Story Cloze Task (SCT). In this paper, we develop a multi-attention based neural network (MANN) with well-designed optimizations, like Highway Network, and concatenated features with embedding representations into the hierarchical neural network model. Considering the particulars of the specific task, we thoughtfully extend MANN with external knowledge resources, exceeding state-of-the-art results obviously. Furthermore, we develop a thorough understanding of our model through a careful hand analysis on a subset of the stories. We identify what traits of MANN contribute to its outperformance and how external knowledge is obtained in such an ending prediction task.

1 Introduction

The prediction on story endings is an important and interesting application because it is involved with several essential issues, such as textual semantic understanding, logical reasoning and natural text generation. Most previous studies on the subject of common sense story understanding mainly focus on generating guesses for a missing event, such as matching explicit information in a given context (Chambers and Jurafsky, 2008), paying attention to specific types of common sense knowledge, like event schema (Chambers and Jurafsky, 2009), or concentrating on unsupervised learning (Chambers and Jurafsky, 2008). Although numerous studies have addressed the issue, training machines to be able to understand underlying narrative structures is still a challenging task. Previous research is limited at the shallow technique requirement of evaluation and noisy knowledge resources.

To facilitate the evaluation and benchmark the problem in the literature, the Story Cloze Task (SCT) has been introduced to predict what should be the "right" ending to a story (Mostafazadeh et al., 2016), which consists of daily events. The common strategies utilized to perform the task can be classified into two kinds of approaches: (1) traditional machine learning techniques with optimal feature engineering; and (2) deep learning-based models with effective strategies (Cai et al., 2017). The task is published with an unlabeled training set that consists of one-correct-ending stories, which is a notable impediment for further research. Some studies investigated fake ending generation, which obtained far more satisfying results. Most supervised learning approaches are trained from the finite evaluation set, ignoring the sheer volume of training corpus.

Understanding daily stories requires not only common sense experience sharing, but also a thorough understanding of text learned from common sense knowledge resources. We propose an effective multi-attention based neural network (MANN) and broaden our model with external knowledge. The model is superior in three aspects: (i) it adds features of sentences as embedding representations; (ii) it features a self-matched attention mechanism that functions through one sentence of the story, while interaction attention functions across the story plot and ending option to obtain word-level interaction; and (iii) it involves external knowledge to augment text coherence understanding.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1754–1762
Santa Fe, New Mexico, USA, August 20-26, 2018.

The proposed model is comprehensively evaluated by comparing it with state-of-the-art approaches. Results show that the performance of the MANN strategy is superior to that of its competitors by up to 7% in terms of accuracy.

Furthermore, we conduct ablation experiments to illustrate the effect of each component in the MANN framework. Through elaborate evaluation, we demonstrate the superiorities and different characteristics of the components in the proposed model and the beneficial effect of the external knowledge utilized.

The contributions of this work are as follows:

- We build a MANN with deliberately devised structures, consisting of components that were not previously applied in this task, such as synthesis embeddings, multiple attentions and Highway, to characterize the semantic coherence of temporal stories from SCT.

- Unlike previous work on generating fake options or that trained only on the labeled set, we extend our model by regarding the unannotated corpus as proportions of external knowledge to enrich insufficient information to remedy the issue of limited resources.

- We conduct comprehensive analysis by manually labeling a subset of 300 stories to further study how our method performs in the story ending prediction task. We demonstrate that the proposed model outperforms the state-of-the-art methods by up to 7%. We will provide the full list of these annotated samples for further research.

2 Related Work

The issue of story ending prediction is related to several other research topics, such as reading comprehension and common sense learning, which will be briefly surveyed as follows.

Reading comprehension is the ability to read and understand text, and it has attracted much attention in natural language processing (NLP) to evaluate the level a machine can reach in understanding text. Two popular forms of evaluation tasks exist in this field: cloze-style query and text-span matching. Cloze-style query, such as SQuAD published by Stanford University, focuses on predicting existing text from the original corpus when given a relevant context. Text-span matching is different from selecting a possible word from the provided text to replenish the blank areas, such as CNN/DailyMail by Hermann and Hinton. Existing tasks are constructed with fragments, whereas examples from SCT are complete and independent stories that has short and meaningful sentence. SCT is also different in that it requires the prediction of development of a story, which is not provided in the given hypothesis. This novel task calls for stronger relation extraction and external inferential capability to identify the correct ending. Our model paid attention on through structure and proved to be effective during experiments.

Common sense learning is a challenging aspect in NLP. The limitation of other rich knowledge structures is that they mostly either focus on shallower representations, such as semantic roles like PropBank (Palmer et al., 2005), or pay attention to specific types of knowledge, i.e., unsupervised co-reference in the text (Chambers and Jurafsky, 2009) and event temporal relation (Modi and Titov, 2014). Learning from structural event knowledge is proposed to enrich this field, including narrative schema (Chambers and Jurafsky, 2009) and event frames (Sha et al., 2016). Unlike the above tasks, SCT (Mostafazadeh et al., 2016) provides large-scale supervised training stories of temporal and causal relations, ensuring a high-quality evaluation for common sense knowledge understanding of mechanisms.

However, the published ROCStories could not be used directly in supervised learning. Considering the use of the training set without negative endings, researchers proposed strategies to generate incorrect options. A conditional generative adversarial network has been proposed, achieving a moderate result with an accuracy of 60.9% (Wang et al., 2017). Roemmele (Roemmele et al., 2017) designed four generative models for fake options, namely, random, backward, nearest-ending and language model. The best result is produced from samples of all four types of endings (67.2%).

Other researchers have attempted to learn from the limit-scale validation set and augment the capability of the relation extractor. Schwartz (Schwartz et al., 2017) is the champion of the LSDSem 2017 Shared

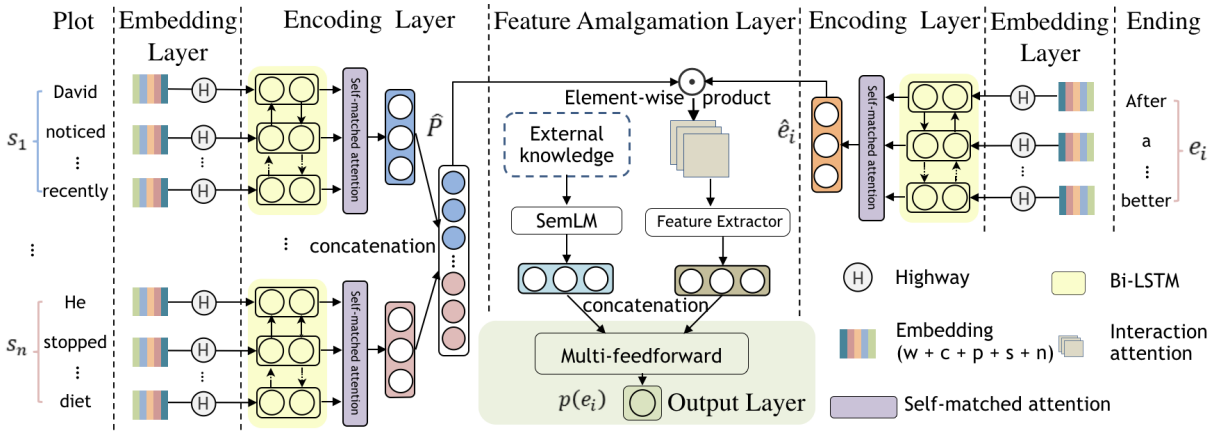


Figure 1: Architecture of our model.[1]

Task, which achieved a score of 75.2% by associating writing style features in endings and training a linear regression. HCM (Chaturvedi et al., 2017) trained a joint model with feature engineering to obtain representations of event sequence, sentiment, and topic from validation set and a hidden variable approach as a voter, thereby obtaining 77.6%. The previous NN-based models did not perform well. Cai (Cai et al., 2017) constructed a model with hierarchical long short-term memory network (LSTM) to encode plot and an ending2sentence attention, then concatenated the two representations through feed-forward network and outputting the final prediction, obtaining 74.7% accuracy. We pursue the same strategy to construct our principal model MANN and see opportunity to utilize external knowledge in the technique of combining semantic sequence information.

3 Proposed Model

This study aims to deduce the right ending given its previous context in the story. Formally, the story ending prediction task is defined as follows: given the story $\langle P, E\rangle$, where P=$\langle s_1, \ldots, s_n\rangle$ is a story plot, and the ending options E=$\langle e_1, \ldots, e_k\rangle$, the task is to select an appropriate ending e_i $(1 \leq i \leq k)$ from E. We address the task as a regression problem, mapping the right ending as 1 and the wrong ending as 0. We identify the highest-scored option as the answer. We first introduce the MANN model, followed with the extension of the model; this extension is constructed to learn from external knowledge. The whole model is shown in Fig.1.

3.1 Multi-attention Neural Network Model

The proposed MANN model consists of four layers. The embedding layer maps each word to a high-dimension vector representation. Then, the encoding layer encodes the representation of the context, and the feature amalgamation layer extracts features from the interaction between plot and the ending. Finally, the output layer provides the probability of the right ending.

Embedding Layer: We concatenate five representations: (i) word embedding; (ii) character feature; (iii) part-of-speech (POS) tagging; (iv) sentiment polarity of a word; and (v) negation.

Word embedding is conducted by converting a token to a high-dimensional vector space, and we obtain a fixed word embedding of each word by using pre-trained vocabulary. The character feature is obtained by mapping each word to a high-dimensional vector space to better handle out-of-vocab or rare words. We abstract character-level representations for tokens through a one-dimension convolutional neural network(CNN). The vectors embedded from characters have the same size as the input channel size of the CNN. By max-pooling the outputs of the CNN over the entire width, we obtain fixed-size vectors for words. By utilizing pre-processing tools, we tackle the last three aspects of a word as one-hot representations, while we collect the POS tagging feature with a natural language toolkit, sentiment polarity of a word with a look-up from pre-trained sentiment lexica, and negation with a corpus of

[1] w for word, c for character, p for POS, s for sentiment polarity, n for negation.

negation words (i.e., "not", "neither", "nor" and "n't"). The concatenation of the five representations of a word is then passed through a two-layer Highway Network (Srivastava et al., 2015) to fuse the information of features, which is processed as follows:

$$trans = ReLU(w_t x + b_t) \tag{1}$$

$$gate = \sigma(w_g x + b_g) \tag{2}$$

$$H(x) = gate * trans + x(1 - gate) \tag{3}$$

where $w_g, w_t \in \mathbb{R}^{D\times D}$ and $b_t, b_g \in \mathbb{R}^D$, D is the dimension of input.

Encoding Layer: The encoding layer encodes the sequence and semantic abstraction of a single sentence and then compromises them, which are from an identical plot to obtain a fusion premise.

Sentence encoding constructs an LSTM in both directions on top of the embeddings provided by the previous layer and concatenate the outputs of forward and backward LSTMs, to learn high-level abstractions from time-sequence features of the context. We obtain $s_{lj,(j=1,\ldots,n)} = [\overrightarrow{LSTM}; \overleftarrow{LSTM}]$ where ; represents the concatenation between two directional LSTM thus $s_{lj} \in \mathbb{R}^{T\times 2d}$ denotes each sentence in plot and $e_{li} \in \mathbb{R}^{G\times 2d}$ denotes the ending.

The new representations are passed into self-matched attention to model the temporal interactions between words. Taking vector V as example, self-matched attention $\widetilde{V} = att(V)$ is defined as follows:

$$M_{ij} = W^{\mathrm{T}}[V_i; V_j; V_i \circ V_j] \tag{4}$$

$$a_i = softmax(M_i) \tag{5}$$

$$\widetilde{V_i} = \sum_j a_{ij} V_j \tag{6}$$

where $W^T \in \mathbb{R}^{6d}$ is a weight matrix and $\circ$ is element-wise multiplication. The higher-level semantics can directly tackled from encoded sequences through the attention mechanism.

Plot encoding compounds the interacted representation of sentences from the same story context. We studied several commonly used implementations for sentence combination, such as LSTM, summation, weighted summation, and concatenation operations. We find simple concatenation useful. The encodings of the story plot is concatenated as $\hat{P} = [att(s_{l1}); \cdots ; att(s_{ln})]$, $\hat{P} \in \mathbb{R}^{nT\times 2d}$, while the ending is simply represented as $\hat{e} = att(e_{li})$, $\hat{e} \in \mathbb{R}^{G\times 2d}$.

Feature Amalgamation Layer: We focus on characterizing diverse interaction information among representations of the plot and the ending, thus extracting feature from them. This layer is inspired by the IIN model (Gong et al., 2018). We combine the plot vectors and ending representations to create a word-by-word interaction attention tensor, in which each channel represents the interaction of the word in one dimension. We tried processing, such as $F_{ij} = \hat{P}_i \circ \hat{e}_j$, $F_{ij} = \hat{P}_i + \hat{e}_j$ and $F_{ij} = |\hat{P}_i - \hat{e}_j|$ and found that the most effective one is $F_{ij} = \hat{P}_i \circ \hat{e}_j$, where we define $i \in [1, \ldots, nT]$, $j \in [1, \ldots, G]$, $\circ$ is element-wise multiplication and $F \in \mathbb{R}^{nT\times G\times 2d}$.

Then, we adapt a feature extractor on F to extract semantic features from word-by-word interaction.Unlike extractors (i.e., VGG and ResNets (He et al., 2016)), DenseNet (Huang et al., 2017) strengthens feature propagation and reduces information disappearance through time because of the structure of pre-activation.

Output Layer: A multi-feedforward neural layer is used. We apply three $tanh$ layers to calculate a score for prediction support.

3.2 Extension of MANN

Labeling large-scale examples requires considerable expertise and manpower. With the pattern of supervised learning, MANN can merely use limited annotated examples to train finite information, thereby preventing us from obtaining more powerful statistical model. To eliminate the restriction of labeled data scarcity and ensure the robustness of our model, we introduce semantic sequence information extracted from external knowledge onto MANN, thus building an extended model, i.e., sequence based MANN (SeqMANN).

External knowledge is mostly used to enrich word implication, thereby ensuring that semantic information can be extracted. To address the combination of external knowledge and the neural network model, Chen (Chen et al., 2017) added extra relation informations of word pair into the encoder. Other researchers studied embedding representations (Bordes et al., 2013) to learn complex reasoning capacities. Considering specific ending-prediction task, we aim to further derive coherence among sentences from stories. Directly working on the large-scale ROCStories is not easy due to its deficiency of negative samples. We use SemLM (Peng and Roth, 2016) to model the distribution over a meaningful sequence chain, with ROCStories regarded as portion of our extra resources while the other are from news data.

SemLM is a language model that first uses FrameNet to split sentences by semantic sequence, and then represents these pieces of sequence with semantic frames and discourse markers from an extended vocabulary. The abstraction of a sentence is $[f_1, dis_1, f_2, \cdots, o]$ where f_i denotes semantic frames, dis_i denotes discourse markers and o denotes period symbol. The SemLM is trained with a log-bilinear model (Mnih and Hinton, 2007) on ROC corpus and news data, and obtains the probability of two words appearing simultaneously in a sentence. The log-bilinear model computes the sequence probability of the next word w_i given the previous words (context), which is defined as follows:

$$p(w_i|c(w_i)) = \frac{exp(v(w_i)^{\mathrm{T}} u(c(w_i)) + b(w_i))}{\sum_{w \in V} exp(v(w)^{\mathrm{T}} u(c(w_i)) + b(w))} \tag{7}$$

We define $v(w)$ as the target vector, $v'(w)$ as the context vector, and $b(w)$ as a bias of a token. Here, V is the vocabulary, $u(c(w_i)) = \sum_{c_t \in c(w_i)} q_t \circ v'(c_t)$, $\circ$ is element-wise multiplication and q_i is a model parameter that depends on the position of a token in the context. The final sequence probability is $\prod_{i=1}^{k} p(w_i|c(w_i))$.

The trained language model is then used to calculate the conditional probability of semantic frames from each option when given the same hypothesis, inspired by HCM (Chaturvedi et al., 2017). For each ending e_i with frames represented as f_{e_i} and $\langle f_1, f_2, \cdots, f_T \rangle, T \geq n$ indicating semantic frames evoked in the story plot, the following features are captured considering the sequence of frames in corresponding story plot: $P(f_{e_i}|f_T), P(f_{e_i}|f_T, f_{T-1}), \ldots, P(f_{e_i}|f_T, f_{T-1}, \cdots, f_1)$.

Finally we learn the semantic sequence feature of plot-ending pairs which represents the interaction information between the plot and the ending of stories, extending the study on external knowledge. We fuse it with features extracted from DenseNet in the feature amalgamation layer through concatenation.

4 Experiments

4.1 Data

As described in (Mostafazadeh et al., 2016), SCT was constructed based on ROCStories. The ROC corpus consists of 100,000 five-sentence cases, each of which was written as a logically meaningful story. After eliminating original endings, writers develop both a "right" ending and a "wrong" ending for the context of examples which are randomly chosen from the corpus. The published SCT is constructed with ROCStories as a large training set, an evaluation set, and a test set, which have the same structure and a size of 1,871.

We evaluate our model by using the benchmark SCT (Mostafazadeh et al., 2016). Notably, the training set contains four-sentence articles with one correct ending, while the evaluation set consists of four-sentence stories with two ending options.

4.2 Training details

To train our neural algorithm, we apply word embeddings of a look-up from 100-d GloVe pre-trained on Wikipedia and Gigaword (Pennington et al., 2014). We set $hiddensize = 100$ for LSTM. An Adam optimizer with a mini-batch size of 120 and an initial learning rate of 0.01 is applied. In the feature amalgamation layer, DenseNet consists of three pairs of dense blocks with a following transition block. The number of layers in a dense block is set as 10, and a ReLU activation function is applied for the whole convolutions. In the output layer, we use three full-connected layers with ReLU activation function. We use mean squared error as the loss function. We decide the model based on the average accuracy of the held-out folds through 5-fold cross validation.

Table 1: Performance comparison.

	Acc.
Machine Learning Algorithm	
Acoli (Schenk and Chiarcos, 2017)	70.0%
Schwartz (Schwartz et al., 2017)	75.2%
HCM (Chaturvedi et al., 2017)	77.6%
Neural Network Algorithm	
DSSM (Mostafazadeh et al., 2016)	58.5%
CGAN (Wang et al., 2017)	60.9%
Lin (Lin et al., 2017)	67.0%
LSTM (Mihaylov and Frank, 2017)	72.8%
Cai (Cai et al., 2017)	74.7%
MANN	78.3%
SeqMANN	84.7%
Human Performance	100%

Table 2: Ablation study.

	MANN	SeqMANN
Embedding Ablation		
-character feature	76.5%	84.3%
-sentiment polarity	75.7%	84.3%
-POS	77.5%	84.5%
-negation	76.3%	84.4%
-word embedding	73.4%	82.9%
Component Ablation		
-Highway	76.3%	83.0%
-biLSTM	74.3%	82.3%
-self_matched attention	75.1%	82.4%
-interaction attention	74.6%	80.0%
-MANN	—	71.3%
Full	78.3%	84.7%

4.3 Results on Our Model

We compare our model with some distinctive methods and approaches that rank high in LSDSem 2017 Shared Task (Mostafazadeh et al., 2017) in Tabel 1. Cai (Cai et al., 2017) is state-of-the-art model in NN-methods while HCM (Chaturvedi et al., 2017) is state-of-the-art model in all published methods. Under the condition in which external resources are not used, MANN outperforms Cai by 3% and even performs better than the highest-level method. Benefiting from external knowledge, SeqMANN achieves a 6.4% improvement over MANN. Nevertheless, the superiority of SeqMANN does not rely on only the external resources utilized, but the correlated effect between MANN and the external resources, as we can see that after removing MANN from SeqMANN (as shown in Table 2 and will described in Section 4.4), the remaining external resources based model only obtained 71.3%[1].

4.4 Ablation Study

We conduct an ablation study on the proposed model to evaluate the effectiveness of each feature and component involved. Results are shown in Table 2.

Embedding Ablation: All embedding features contribute to MANN. However, the influence is not as crucial in SeqMANN. We conjecture that contextual information in the embedding process partly overlaps with the features in external knowledge.

Component Ablation: We respectively remove each component to study their contribution. For interaction attention ablation, we replace it with flattening between plot and ending representations, followed by removing DenseNet and replacing a three-layer $tanh$ operation as feature extractor. For MANN ablation, we only retain the extraction of external knowledge and the output layer (we model SemanticSequence by the same way as MANN ablation processing in Section 5).

5 Data Analysis

To further analyze the model, we seek to determine the abilities required to predict the right ending, and which aspects of questions among the specific dataset are solved by our model. We sample 300 examples from the validation set randomly and annotate them manually from two cognitive chunks, namely, by labeling samples with the difficulty degree in human-understandable rationale for prediction and by tagging samples according to the linguistic phenomena they contain.

[1]Code and the annotated samples are available at https://github.com/StoryDevelopment/SCT.

Table 3: Some examples from each human-understandable difficulty labeling category.

Category	Story plot	Ending options
Relevant Word	Kyle invited everyone he works with bowling one night. Most people could not go but Matt and John showed up. Matt had never been bowling before so they had to show him how to. After a few games, Matt picked up how to play better.	e_1: Now Matt and Kyle can go bowling more then. e_2: Kyle took the children shopping for a gift for their mother.
Compatible Sentence	It was the last day of our vacation.We were eating lunch on the patio of the hotel. We laughed and smiled because it was a great vacation.Then we packed our bags and drove to the airport.	e_1: We want to revisit someday. e_2: We all vowed to never go back again.
Plot-level Paraphrasing	Rory was allergic to gluten and strawberries.One day she sat down to eat lunch at school.She opened her lunch box, and stared at a sandwich with strawberries.Her new step mom had packed her lunch for the first time.	e_1: Rory had to buy a school lunch that day. e_2: Rory ate the sandwich.
Ambiguous Inference	Tina always wore a red bikini when she went to the beach. She was known for it and everyone expected to see her in one. One day she met her friends in a blue bikini and surprised them. They could not understand why she would wear something different.	e_1: They liked the new bikini though. e_2: Tina's friends knew how unpredictable she was.

5.1 Human-understandable Difficulty Labeling

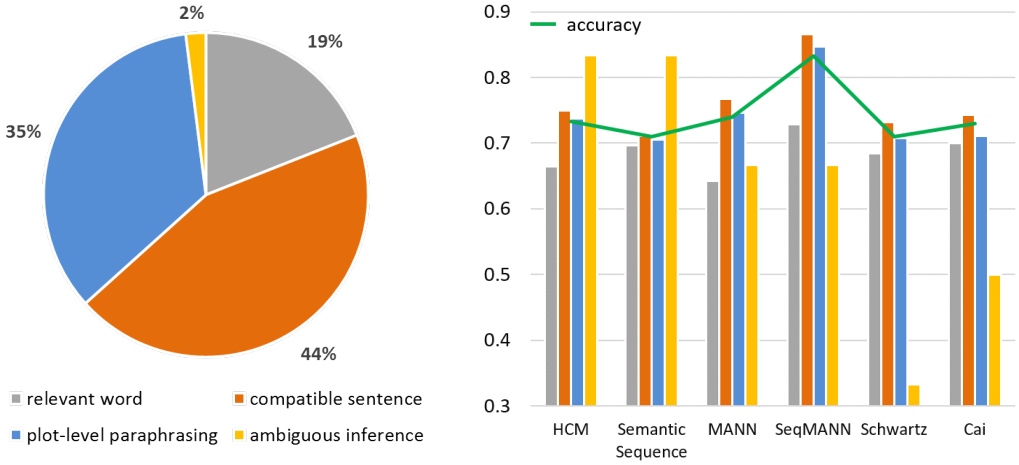
We classify the 300 samples into the following categories (as shown in Table 3, where the right ending is e_1 and the wrong ending is e_2):

Relevant Word. The verbs and noun phases in e_1 are more relevant to the plot than those in e_2. This category includes examples that are thought to be correctly predicted at the word level.

Compatible Sentence. We could derive the correct answer by understanding a single sentence in the hypothesis, which means that the answer is similar to some sentence in an earlier context.

Plot-level Paraphrasing. It requires a full understanding of multiple sentences to infer the answer.

Ambiguous Inference. It includes examples in which we think both endings are logically reasonable for the story, while the correct one is more answerable. This category consists of poorly designed cases, which we consider miscarriage examples in this task.



(a) The proportion of each category.

(b) Results in each category.

Figure 2: Results of human understandable difficulty labeling.

Fig.2a shows the proportion of each category. The second and third cases have the highest proportion, followed by the first and the least "Ambiguous Inference" cases. We observe that deeper information processing, which means sentence understanding (for "Compatible Sentence" case) and plot-level understanding (for "Plot-level Paraphrases" case), is crucial to good performance in this task. The low proportion of "Ambiguous Inference" shows the satisfactory quality of this task.

To further analyze the depth of prediction from our models compared with other methods, we reproduce the other methods and test the 300 annotated examples based on the above categorization. Results are presented in Fig.2b. We conclude the following: (i) The comparison between MANN and semantic sequence shows that semantic sequence is superior in shallow natural language comprehension, such as word-level understanding, while MANN performs better in deep contextual comprehension, such as sentence and plot level understanding; (ii) Successfully, SeqMANN combines the two advantages to out-

Table 4: Linguistic phenomena tagging results.

Phenomena Tag	Label Frequency	Cai	Schwartz	HCM	SemanticSequence	MANN	SeqMANN
CONDITIONAL	2.0	83.3	83.3	83.3	66.7	66.7	**100.0**
NEGATION	42.7	70.3	67.9	69.5	64.1	70.3	76.6
SENTENCE LENGTH	42.0	76.1	76.5	75.3	70.3	72.2	81.7
QUANTITY/TIME REASONING	21.0	73.0	60.3	66.7	65.1	74.6	82.5
COREF	39.3	83.1	75.4	78.0	75.4	77.1	**88.1**
QUANTIFIER	31.3	73.4	69.1	67.0	61.7	74.6	82.5
MODAL	27.0	75.3	76.5	75.3	70.3	69.1	80.2
BELIEF VERBS	5.0	53.3	53.3	60.0	60.0	73.3	73.3
CONVERSATIONAL PIVOTS	31.3	76.6	69.1	74.5	75.5	70.0	**84.0**
ANTO	24.7	73.5	66.3	74.7	67.5	68.7	83.1
EMOTIONAL COMMONALITY	68.7	73.7	72.3	74.3	71.3	78.2	**86.9**
ENDING ONLY	4.0	58.3	76.5	60.0	60.0	91.6	**83.3**
ACCURACY	—	72.3	71.3	73.3	71.0	74.0	83.3

perform other methods on this task. (iii) The model is a stable one with external knowledge resources, thus still maintaining high accuracy under decreasing training data.

5.2 Linguistic Phenomena Tagging

To obtain an idea of the linguistic phenomena in SCT and to conduct a detailed analysis of the semantical performance of these models, imitating MultiNLI (Williams et al., 2018), we design a set of annotation tags to label the subset as follows:

CONDITIONAL: Whether the example contains the word "if".
NEGATION: Whether the example contains negation words, e.g., not, none, neither.
SENTENCE LENGTH: Whether the right ending is longer than the other endings.
QUANTITY/TIME REASONING: Whether understanding the ending options contains quantity or time reasoning that needs to be explained from the plot.
COREF: Whether ending options contain referring expressions.
QUANTIFIER: Whether the example contains quantifier words, e.g., more, most, enough.
MODAL: Whether the example contains modal verbs.
BELIEF VERBS: Whether the example contains belief verbs such as think, believe and doubt.
CONVERSATIONAL PIVOTS: Whether the example contains discourse cohesion, e.g., but, yet, however, though, while.
ANTO: Whether the example contains an antonym pair.
EMOTIONAL COMMONALITY: Whether the sentiment throughout the plot is consistent with that through the right ending.
ENDING ONLY: Whether the ending options contradict themselves.

Results are shown in Table 4, and we observe the following: (i) The poor performances on SENTENCE LENGTH indicates that such feature, which is regarded as valuable bias (Cai et al., 2017; Schwartz et al., 2017), does not have an influential contribution to our model. (ii) Examples with ENDING ONLY tag shows that our attention components recognize not only paraphrases in relation among sentences but also the intra-semantic implications of each sentence. (iii) The model outperforms in terms of EMOTIONAL COMMONALITY, which shows effects of adding sentiment polarity in embedding representation.

6 Conclusion

In this paper, we proposed a MANN model with external knowledge. The results of this model outperformed state-of-the-art results by 7%. We carefully examined a subset of the corpora from SCT to analyze the performance of our models. The competitive model benefits not only from our thoughtfully designed structure but also from the combination of semantic relations learned from external resources.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant No.U1636116, 11431006, 61772288, the Research Fund for International Young Scientists under Grant No. 61650110510 and 61750110530, and the Ministry of education of Humanities and Social Science project under grant 16YJC790123.

References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795.

Zheng Cai, Lifu Tu, and Kevin Gimpel. 2017. Pay attention to the ending: Strong neural baselines for the roc story cloze task. In *ACL*, pages 616–622.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *ACL*, pages 789–797.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *ACL and AFNLP*, pages 602–610.

Snigdha Chaturvedi, Haoruo Peng, and Dan Roth. 2017. Story comprehension for predicting what happens next. In *EMNLP*, pages 1603–1614.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, and Diana Inkpen. 2017. Natural language inference with external knowledge. *arXiv preprint arXiv:1711.04289*.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural language inference over interaction space. In *ICLR*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*, pages 770–778.

Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. 2017. Densely connected convolutional networks. In *CVPR*, pages 4700–4708.

Hongyu Lin, Le Sun, and Xianpei Han. 2017. Reasoning with heterogeneous knowledge for commonsense machine comprehension. In *EMNLP*, pages 2032–2043.

Todor Mihaylov and Anette Frank. 2017. Story cloze ending selection baselines and data examination. In *LSDSem*, pages 87–92.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *ICML*, pages 641–648.

Ashutosh Modi and Ivan Titov. 2014. Inducing neural models of script knowledge. In *CoNLL*, pages 49–57.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. In *NAACL*, pages 839–849.

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James Allen. 2017. Lsdsem 2017 shared task: The story cloze test. In *LSDSem*, pages 46–51.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Haoruo Peng and Dan Roth. 2016. Two discourse driven language models for semantics. In *ACL*, pages 290–300.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Melissa Roemmele, Sosuke Kobayashi, Naoya Inoue, and Andrew M Gordon. 2017. An rnn-based binary classifier for the story cloze test. In *LSDSem*, pages 74–80.

Niko Schenk and Christian Chiarcos. 2017. Resource-lean modeling of coherence in commonsense stories. In *LSDSem*, pages 68–73.

Roy Schwartz, Maarten Sap, Ioannis Konstas, Leila Zilles, Yejin Choi, and Noah A Smith. 2017. The effect of different writing tasks on linguistic style: A case study of the roc story cloze task. In *CoNLL*, pages 15–25.

Lei Sha, Sujian Li, Baobao Chang, and Zhifang Sui. 2016. Joint learning templates and slots for event schema induction. In *NAACL-HLT*, pages 428–434.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *NIPS*, pages 2377–2385.

Bingning Wang, Kang Liu, and Jun Zhao. 2017. Conditional generative adversarial networks for commonsense machine comprehension. In *IJCAI*, pages 4123–4129.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*.

A Reinforcement Learning Framework for Natural Question Generation using Bi-discriminators

Zhihao Fan[1], **Zhongyu Wei**[1]*, **Siyuan Wang**[1], **Yang Liu**[2], **Xuanjing Huang**[3]
[1]School of Data Science, Fudan University, China
[2]Liulishuo Company
[3]School of Computer Science, Fudan University, China
{14300180043,zywei,14302010062}@fudan.edu.cn, yang.liu@liulishuo.com, xjhuang@fudan.edu.cn

Abstract

Visual Question Generation (VQG) aims to ask natural questions about an image automatically. Existing research focuses on training model to fit the annotated data set that makes it indifferent from other language generation tasks. We argue that natural questions need to have two specific textual attributes from the perspectives of content and linguistic respectively, namely, *natural* and *human-written*. Inspired by the setting of discriminator in adversarial learning, we propose two discriminators, one for each attribute, to enhance the training. We then use the reinforcement learning framework to incorporate scores from the two discriminators as the reward to guide the training of the question generator. Experimental results on a benchmark VQG dataset show the effectiveness and robustness of our model compared to some state-of-the-art models in terms of both automatic and human evaluation metrics.

1 Introduction

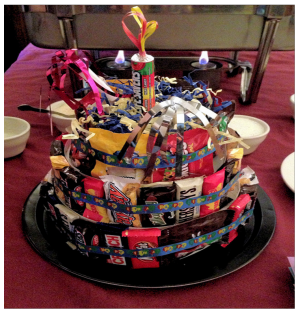
Recent years see the popularity of multi-modal research on vision and language. Visual caption generation (VCG) (Xu et al., 2015; Vinyals et al., 2015) and visual question answering (VQA) (Antol et al., 2015) attract increasing attention from research communities. VCG aims to generate descriptions for a given image with the goal of scene understanding, while VQA asks visual questions and requires an answer to it. Research for these two tasks are fueled by several manually generated corpora (Lin et al., 2014; Zhu et al., 2016).

Different from generating a statement (descriptions or answers) about an image, visual question generation (VQG) (Mostafazadeh et al., 2016) is tasked with generating a natural question which can potentially engage a human in starting a conversation when shown an image. Under this guidance, Mostafazadeh et al. (2016) collect natural questions for images via a crowd-sourcing platform and construct the first dataset for VQG. They also explore some neural network-based models for natural question generation. Those models are trained to better fit the VQG dataset that makes them indifferent from other language generation models and hard to identify the progress in naturalness for generated ones. Some human generated questions for VQG and questions for VQA are shown in Figure 1, we name questions for VQA descriptive questions and those for VQG natural ones. As we can see, VQA questions are much simpler and can be easily answered using information from the source image directly. In contrast, VQG questions are more complex and answers are not trivial. We therefore argue that the specialty in terms of content needs to be considered for natural question generation.

In this paper, We formulate the task of visual natural question generation as language generation task with specific attributes in terms of content and linguistics, i.e. *natural* and *human-written*. Recently, adversarial learning approaches (Goodfellow et al., 2014) have been applied to various tasks and show advantage of learning boundary for target data distribution. Inspired by the setting of discriminator, we propose to use two discriminators to better learn these two textual attributes. For the attribute of *human written*, we use a generative adversarial network (GAN) to learn a dynamic discriminator to

*Corresponding author

Proceedings of the 27th International Conference on Computational Linguistics, pages 1763–1774
Santa Fe, New Mexico, USA, August 20-26, 2018.



VQA:
1. How many plates are seen ?
2. Is this some sort of craft idea ?
3. How many different candies are visible ?

VQG:
1. Is this serving as a birthday cake ?
2. Where can I buy that birthday cake ?
3. How many different candies did you use to make this cake ?
4. Whose idea was it for a candy cake ?
5. Who made that candy cake ?

Figure 1: Annotated questions for tasks of VQG and VQA to an image from the dataset of MSCOCO (Antol et al., 2015).

distinguish human generated questions and machine generated questions. For the attribute of *natural*, we use questions from VQA as negative samples and questions from VQG as positive samples to train a static discriminator.

It is difficult to come up some differentiable objective function for our target. Recently, reinforcement learning has been introduced to optimize model in terms of non-differentiable metrics (Ranzato et al., 2015; Rennie et al., 2016; Zhang et al., 2017; Feng et al., 2018). Therefore, we propose a reinforcement learning framework (Williams, 1992) to incorporate scores from the two discriminators as the reward to guide the optimization of the question generator. Experiment results on a benchmark dataset show the effectiveness of our proposed framework in terms of both automatic and human evaluation.

We will introduce our framework in detail in section 2. In section 3, we present our experiment results. In section 4, we list some related works. In section 5, we conclude our work and point out some future directions.

2 Framework

Given an image I, we aim to train a generative model G_θ with parameter θ that is able to produce natural questions. The generator is designed following the fashion of Seq2Seq (Cho et al., 2014) that takes the representation of the image as input and generates a question word by word. We take two attributes of natural questions into consideration while training the generator. In particular, two discriminators are proposed to distinguish samples from two pairs of counter-question-distributions, namely *human written* vs *machine generated* and *natural* vs *descriptive*. A reinforcement learning framework is then used to combine results from the two discriminators as the reward to train the generator. The overall framework can be see in Figure 2.

2.1 Bi-discriminator configuration

We first introduce our setup of bi-discriminators in this sub-section starting with the design of a hierarchical structure for the distribution of questions.

Hierarchical structure for question distribution Suppose we have an overall domain $\mathcal{D}$ for all the questions, it can be split into two antithetic domains $\mathcal{D}_g$ (*machine generated*) and $\mathcal{D}_h$ (*human written*) according to linguistic attribute. The human written domain can be further split into $\mathcal{D}_{n+}$ (*natural*) and $\mathcal{D}_{n-}$ (*descriptive*) according to the content attribute *natural*. The hierarchical structure is described in Equation 1.

$$\begin{aligned} \mathcal{D} = \mathcal{D}_g \cup \mathcal{D}_h, \;\; \mathcal{D}_h = \mathcal{D}_{n+} \cup \mathcal{D}_{n-} \\ \mathcal{D}_{n+} \subset \mathcal{D}_h \subset \mathcal{D} \end{aligned} \tag{1}$$

To distinguish questions from the two pairs for counter-question-domains ($\mathcal{D}_g$ vs $\mathcal{D}_h$ and $\mathcal{D}_{n+}$ vs $\mathcal{D}_{n-}$), we propose two discriminators, D_1 and D_2. Discriminator D_1 is trained to discriminate whether a question comes from $\mathcal{D}_h$ or $\mathcal{D}_g$, and D_2 is trained to discriminate whether a question belongs to $\mathcal{D}_{n+}$

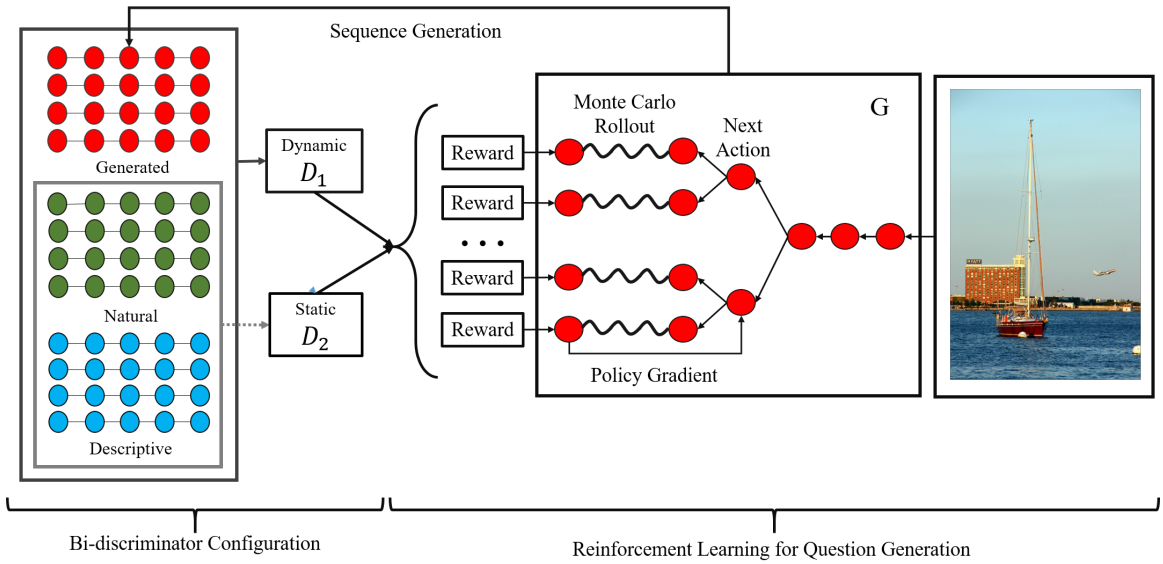


Figure 2: The overall framework of our proposed model.

or $\mathcal{D}_{n-}$. Final layer's activation of two discriminators is sigmoid, which represents the possibility that question belong to the positive domain $\mathcal{D}_*$ as formula 2 shown.

$$P_1\big(G_\theta(I) \in \mathcal{D}_h | I\big), \quad P_2\big(G_\theta(I) \in \mathcal{D}_{n+} | I\big) \tag{2}$$

where $G_\theta(I)$ stands for questions generated by the generator given image I, P_1 stands for the likelihood that a generated question is from domain $\mathcal{D}_h$ and P_2 stands for the likelihood that a generated question is from $\mathcal{D}_{n+}$.

Scores of the two discriminators are served as the reward in our reinforcement learning framework to guide G_θ to generate questions closer to questions in target domain $\mathcal{D}_*$. Under such setting, it is easy to observe that question similar to $\mathcal{D}_{n+}$ would be encouraged to generate by both D_1 and D_2, which means that our bi-discriminator environment configuration is able to encourage natural question generation in theory. In conclusion, maximizing score of $G_\theta(I)$ assigned by two discriminators is equal to encourage generate question with attributes of $\mathcal{D}_h$ (human-written) and $\mathcal{D}_{n+}$ (natural).

Discriminator D_1 for question domains $\mathcal{D}_g$ and $\mathcal{D}_h$ Discriminator D_1 is proposed to distinguish human written questions and machine generated questions. It is used to guide the generator to produce questions closer to samples from the domain of $\mathcal{D}_h$. We propose to use questions from a human generated dataset as positive samples while questions from our generator as negative samples. Considering the generator is updating during the training process, discriminator D_1 needs to be re-newed accordingly. We introduce generative adversarial network (GAN) for the training of D_1 to learn the border between $\mathcal{D}_h$ and $\mathcal{D}_g$ in pace with the updating of G_θ. The target of GAN is shown in Equation 3.

$$\min_G \max_D V(D,G) = \mathbb{E}_{p_{data}(\mathbf{x})}\big[log\, D(\mathbf{x})\big] + \mathbb{E}_{p_z(\mathbf{z})}\big[log\,\big(1 - D(G(\mathbf{z}))\big)\big] \tag{3}$$

The discriminator would converge to $D^*(\mathbf{x}) = p_{data}(x)/\big(p_{data}(\mathbf{x}) + p_g(\mathbf{x})\big)$ during the training. The optimal generator G in GAN aims to mock D to be unable to recognize generated samples. In our case, the optimal condition is that questions generated can be completely mixed with human-written questions, and the dynamic discriminator D_1 is incapable to distinguish these two kinds of questions and thus assign equal probabilities to both categories.

During training, once we have a batch of generated questions, we re-train the discriminator to minimize the loss in Equation 4,

$$L_{D_1} = -\mathbb{E}_{Q\sim D_g}\Big[log\,\big(1 - D_1(Q)\big)\Big] - \mathbb{E}_{Q\sim D_h}\Big[log\, D_1(Q)\Big] \tag{4}$$

where positive samples are from human-written domain $\mathcal{D}_h$, and negative samples are generated by current generator G_θ.

Discriminator D_2 for question domains $\mathcal{D}_{n+}$ and $\mathcal{D}_{n-}$ Discriminator D_2 is proposed to distinguish natural questions and descriptive questions. It is used to guide the generator to produce questions with information beyond the image to meet the attribute of *natural*. With human generated samples from both domains $\mathcal{D}_{n+}$ and $\mathcal{D}_{n-}$, discriminator can be trained before-hand and stay static during the training of the generator. Cross-entropy loss is used for the training of D_2. Considering that labeled samples for *natural* questions are much less than *descriptive* ones in reality, we need to consider the problem of class imbalance. As proved in the application of object detection (Lin et al., 2017b), focal loss can reduce the problem of imbalance. It slows the updating speed of a certain class when that class has been trained well. Similarly, we train the discriminator D_2 using focal loss following Equation 5.

$$
\begin{aligned}
p_t(Q,I) &= \begin{cases} P_2(Q \in \mathcal{D}_{n+}|I) & Q \in \mathcal{D}_{n+}|I \\ P_2(Q \in \mathcal{D}_{n-}|I) & Q \in \mathcal{D}_{n-}|I \end{cases} \\
L_{D_2} &= -\big(1 - p_t(Q,I)\big)^\gamma \, log \;\; p_t(Q,I)
\end{aligned} \tag{5}
$$

2.2 Reinforcement Learning for Question Generation

The reinforcement learning algorithm mainly consists of the generative model G_θ and the reward function R.

Generative model Our generator G_θ follows the design of Seq2Seq model. The only difference is that it takes image features as input instead of a sequence of words. We use *fc7* feature extracted from VGGNet to represent a given image. The decoder is a recurrent neural network that generates a question word by word. In the framework of reinforcement learning, the generation process can be described as a sequence of states and actions (known as trajectory as well), $\big(s_0, a_0, s_1, a_1, \cdots, s_T, a_T, s_{T+1}\big)$, where a_0 is the input image feature. $\mathcal{A}_t (t \geq 1)$ denotes the action space at time t. For text generation, action is the generation of a word, the action space is thus the whole vocabulary. In this task, when action is determined, the following state is also determined. Therefore, we can denote a trajectory as $\big(a_0, a_1, \cdots, a_T\big)$ for simplicity.

Reinforce learning and policy gradient training Based on the reward function R and the generative model G_θ, our goal is to maximize the expectation of reward $R(Q,I)$, where Q is a set of questions produced by G_θ and subject to $p_\theta(Q|I)$. By means of REINFORCE algorithm (Williams, 1992), the objective function is shown in Equation 6.

$$
\mathcal{J}(\theta) = \mathbb{E}_{Q \sim p_\theta(Q|I)}\big[R(Q,I)\big] \tag{6}
$$

Assuming that $p_\theta(Q|I)$ is continuously differentiable with respect to θ, the gradient of the equation 6 with respect to θ can be solved by policy gradient method shown in Equation 7 (Aleksandrov et al., 1968; Glynn, 1990; Williams, 1992).

$$
\frac{\partial \mathcal{J}(\theta)}{\partial \theta} = \mathbb{E}_{Q \sim p_\theta(Q|I)}\left[\left(\sum_{t=1}^{T} \frac{\partial}{\partial \theta} log\, p_\theta(a_t|a_{0:t-1})\right) R(Q,I)\right] \tag{7}
$$

During the training, after generating questions, we compute rewards based on reward function R and update G_θ using gradient ascent algorithm.

Monte Carlo Rollout The disadvantage of REINFORCE algorithm in language generation is that the reward can only be assigned to a complete sentence. Therefore, every single action taken for generating a complete sentence share the same reward. This hurts the effectiveness of training the generator.

To deal with such problem, we employ the strategy of Monte Carlo Rollout to assign a specific reward to each action. Suppose that we have a partial trajectory $a_{0:t-1}$ at t time, then we sample word a_t from

action space $\mathcal{A}_t$ according to policy G_θ. Next, we use Monte Carlo Rollout to generate the remaining $T-t$ tokens. The m_{th} Monte Carlo Rollout consequence is represented in equation 8:

$$\mathrm{MC}^m_{G_\theta}(a_{1:t-1}, a_t) = (a_{1:t-1}, a_t, a^m_{t+1:T}) \tag{8}$$

The reward for action a_t is computed as the mean reward of the sampling sentences, which is the following equation 9:

$$R\big((a_{0:t-1}, a_t), I\big) = \begin{cases} \frac{1}{M}\sum_{m=1}^{M} R\big(\mathrm{MC}^m_{G_\theta}(a_{1:t-1}, a_t), I\big) & t < T \\ R\big((a_{0:t-1}, a_t), I\big) & t = T \end{cases} \tag{9}$$

With Monte Carlo Rollout, the policy gradient is reconstructed as Equation 10.

$$\frac{\partial \mathcal{J}(\theta)}{\partial \theta} = \mathbb{E}_{p_\theta(Q|I)}\left[\left(\sum_{t=1}^{T} \frac{\partial}{\partial \theta} log\, p_\theta(a_t|a_{0:t-1})\right) R\big((a_{0:t-1}, a_t), I\big)\right] \tag{10}$$

Teacher Force As shown in Equation 10, ground-truth samples are not directly used to optimize the generator in the training process. In practice, this is usually in-efficient and is difficult to train a good question generator. Once the performance of generator G_θ is poor and the discriminator is able to do a good job identifying the origin of questions, it is non-trivial for the generator to find a way for improvement without guidance of ground-truth samples. To alleviate the issue, it is necessary for the generator to directly access the ground-truth questions, through which, generator G_θ learns the knowledge of target questions so that it is able to improve the performance. We follow Sutskever et al. (2014) to use the strategy of teacher force that trains the generator via MLE loss together with rewards from discriminators.

Training The overall training process of our proposed model is shown in Table 1.

Algorithm 1 Sketch of proposed model

Q stands for sample questions instances from human-generated dataset
$\hat{Q}$ stands for questions generated by the generator G(I)
Pre-train generator on VQA and VQG dataset
Pre-train D_1 and D_2
For number of training iterations **do**
  for i=1,D1-steps do
    Sample (Q, I) from real data
    Sample $\hat{Q} \sim G(I)$
    Update D_1 using (Q, I) as positive examples and $(\hat{Q}, I)$ as negative examples.

  for i=1,G1-steps do
    Sample (Q, I) from real data
    Sample $\hat{Q} \sim G(I)$
    Compute Reward r_1 for $(\hat{Q}, I)$ using D_1
    Update G on $(\hat{Q}, I)$ using reward r_1

  for i=1,G2-steps do
    Sample (Q, I) from real data
    Sample $\hat{Q} \sim G(I)$
    Compute Reward r_2 for $(\hat{Q}, I)$ using D_2
    Update G on $(\hat{Q}, I)$ using reward r_2

  for i=1,G3-steps do
    Sample (Q, I) from real data
    Teacher Force: Update G using (Q, I)
  End

3 Experiments

3.1 Dataset

We evaluate our models using MSCOCO part of Visual Question Generation (VQG) dataset[1] (Mostafazadeh et al., 2016). It contains 2500, 1250 and 1250 images for training, validation and testing respectively. Each image is accompanied with 5 natural questions produced by human annotators.

Another dataset involved is VQA. For each image in VQA, three questions are collected. The intention of building such dataset is to teach model to response to the questions with respect to concepts in some certain images. Therefore, questions in VQA are much simpler than those ones in VQG. VQG dataset contains about 80000, 40000, 80000 images for training, validation and testing respectively. VQA is used to pre-train our question generator and questions are also served as negative samples to train the discriminator D_2.

3.2 Models for Comparison

We compare our models with some baselines and some state-of-the-art methods.

- **KNN** (Mostafazadeh et al., 2016): using the question from the most similar image as the question for a target image. Cosine similarity based on *fc7* features is utilized to search for similar images.
- **Img2Seq**: it generates a question from image features following Seq2Seq fashion (Cho et al., 2014). The model is trained using the word-level loss maximum likelihood estimation (MLE).
- **Img2Seq**$_{pre-train}$: different from *Img2Seq*, this model is pre-trained on VQA.
- **MIXER-BLEU-4** (Ranzato et al., 2015): it follows the framework of reinforcement learning that uses BLEU-4 between the generated question and the human-written question as the reward to guide the parameter update of the generator with policy gradient. Since the model is trained by optimizing BLEU-4 directly, it is able to generate higher BLEU-4 score in general.
- **Reinforce**$_{\mathbf{D_1}}$: it uses the score of D_1 as the reward to guide the training of the generator. The setting is quite similar to *SeqGAN* (Yu et al., 2017). This model is a upgrade version of **Img2Seq**. It introduces adversarial learning network to better train the generator under the reinforcement learning framework.
- **Reinforce**$_{\mathbf{D_2}}$: it uses the score of D_2 as the reward to guide the training of the generator. This model is comparable to *MIXER-BLEU-4* because both models utilize a static way to produce reward (BLEU score with ground truth questions in *MIXER-BLEU-4* and classification confidence in Reinforce$_{\mathbf{D_2}}$)
- **Reinforce**$_{\mathbf{D_1+D_2}}$: this is our proposed model.

3.3 Training Details

For the generator, we use GRU cell and the number of cells is 512; the dimension of word embedding is 300 and is pre-trained using GloVe (Pennington et al., 2014). The image feature, *fc7* is the output of the 7th fully-connected layer in *VGGNet*. The original dimension of *fc7* is 4096, and we compress it to a 300-dimension vector using 2-layer fully-connected layer. The upper settings are the same for all neural network models. We set batch size, rollout size, D1-step, G1-step, G2-step and G3-step as 64, 16, 5, 1, 2 and 1, respectively. γ for the training of D_2 is set to 2.0 empirically.

3.4 Automatic Evaluation

There is no direct evaluation metric to determine whether a question is natural or not. We thus use several relevance scores for the automatic evaluation following the setting of existing researches, including Corpus BLEU-4, BLEU-4 (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), ROUGE (Lin and Hovy, 2003) and CIDEr (Vedantam et al., 2015). The overall experiment results in terms of five relevance scores are shown in Table 1. We have some findings as follows:

[1]The original dataset contains three parts, namely, MS-COCO, Flickr and Bing, *VQG-MSCOCO*. However, images from Flickr and Bing are quite different from those in Visual Question Answering (VQA) dataset (Antol et al., 2015), and this makes it difficult to use questions from VQA as negative samples to train D_2.

Model	BLEU-4	corpus BLEU-4	METEOR	ROUGE	CIDEr
KNN	37.062	19.799	22.413	52.324	50.199
Img2Seq	36.744	21.028	23.125	54.089	51.171
Img2Seq$_{pre-train}$	37.522	22.106	23.877	55.310	54.076
MIXER-BLEU-4	**41.674**	24.808	24.382	**57.777**	60.527
Reinforce$_{\mathbf{D_1}}$	38.945	24.420	24.665	56.196	59.513
Reinforce$_{\mathbf{D_2}}$	40.063	25.237	25.492	57.503	61.745
Reinforce$_{\mathbf{D_1+D_2}}$	41.098	**26.265**	**25.634**	57.679	**63.388**

Table 1: Overall experiment results of different models in terms of relevance scores. **bold**: the best performance in that column.

- Although *KNN* model retrieves questions from the original dataset, the result is not good enough. Further analysis reveals that questions generated are not relevant to the target image. Therefore, the strategy of reusing questions from similar images is not sufficient for generating question with high quality.
- Performance of *Img2Seq*$_{pre-train}$ is better than that of *Img2Seq* in terms of all the five metrics. This indicates that samples from the domain of $\mathcal{D}_{n-}$ are also helpful for generating high quality natural questions. This is reasonable because samples from VQA also locate in the domain of $\mathcal{D}_h$ therefore pre-training is able to guide the generator to better learn the attribute of *human written*.
- By incorporating the discriminator D_2, both models of *Reinforce*$_{\mathbf{D_2}}$ and *Reinforce*$_{\mathbf{D_1+D_2}}$ is able to improve the performance in terms of all the five relevance scores compared with their counter-part models *Img2Seq*$_{pre-train}$ and *Reinforce*$_{\mathbf{D_1}}$ respectively. This confirms the effectiveness of discriminator D_2. The existence of $\mathcal{D}_{n-}$ helps the generator G_θ learns more about the unique features for questions in $\mathcal{D}_{n+}$ and better demarcates the boundary of itself.
- By incorporating the discriminator D_1, both *Reinforce*$_{\mathbf{D_1}}$ and *Reinforce*$_{\mathbf{D_1+D_2}}$ produce better performance than their counter-part approaches *Img2Seq* and *Reinforce*$_{\mathbf{D_2}}$ respectively. The disadvantage of training using MLE is that only human-generated training samples are exposed to generator (known as exposure bias) and loss is computed in word-level without considering sentence-level performance. By adding D_1, the generator is able to observe generated question during training. Therefore the issue of exposure bias can be partially addressed. Besides, the sentence level performance is also computed and used as reward to guide the training process.
- *MIXER-BLEU-4* produces the highest score of BLEU-4 and ROUGE but it performs much worse in terms of other three metrics. This indicates that simply using one criteria value as the reward is not universal.
- Our proposed model *Reinforce*$_{\mathbf{D_1+D_2}}$ produces three best values out of the five evaluation metrics. This confirms the effectiveness of our proposed framework for natural question generation. Although our model is not optimizing BLUE-4 directly, it performs comparable to *MIXER-BLEU-4* in terms of BLUE-4, indicating the robustness of our model.

3.5 Human Evaluation

We also perform human evaluation on generated questions from different models to evaluate their naturalness (Mostafazadeh et al., 2016). For a given image, we first list questions generated by different models and randomly sample one question from ground-truth to form the question pool. Then we present the image and the corresponding question pool to the annotator. S/he is asked to assign a score to each single question in term of naturalness. A guidance from (Mostafazadeh et al., 2016) is presented during the annotation. We set the score range in $(1\ 2\ 3)$ following (Mostafazadeh et al., 2016). The better the question is, the higher score it would get.

KNN What type of bird are these ?
MIXER-BLEU-4 Are those bananas ?
Img2Seq What kind of bananas are those ?
$\boldsymbol{Reinforce_{D_1}}$**:** Are these bananas fresh ?
$\boldsymbol{Reinforce_{D_1+D_2}}$**:** How much do you think those bananas cost ?

KNN How old is the man riding the horse ?
MIXER-BLEU-4 Is that a horse ?
Img2Seq Is this horse racing today ?
$\boldsymbol{Reinforce_{D_1}}$ What is the name of the horse ?
$\boldsymbol{Reinforce_{D_1+D_2}}$ Why is this horse fitted with a harness ?

Figure 3: Generated question by different models.

Model	1	2	3	Avg
KNN	214	120	66	1.63
Img2Seq	182	147	71	1.72
MIXER-BLEU-4	153	172	75	1.81
Reinforce$_{\mathbf{D_1}}$	167	153	80	1.78
Reinforce$_{\mathbf{D_1+D_2}}$	149	160	91	1.86
Ground-truth	50	79	271	2.55

Table 2: Results of human evaluation for different models.

Two annotators are invited to label the questions independently. We collect the number of various ratings for different models and add up the value of the two annotators. We randomly choose 200 images for human evaluation. Result is shown in Table 2. Figure 3 shows two sample images and corresponding questions generated by different models. Based on the result of human evaluation and the sample questions generated by the system, we have some findings as follows:

- Relatively poor performance of retrieval model indicates that questions retrieved by KNN are not relevant to the image as we can see in Figure 3.

- The performance of *Reinforce*$_{\mathbf{D_1+D_2}}$ is the best among all automatic question generators in terms of the number of rating 3 it obtains. This reconfirms the effectiveness of our framework. Besides, the sample questions in Figure 2. also reveals that our system is able to ask more complex questions.

- The performance of *MIXER-BLEU-4* is middle-level. This indicates that optimizing a single evaluation metric is not sufficient enough for generating high quality natural questions.

- The gap between ground-truth questions and machine generated questions is still large. This indicates that there is still a large room for question generation system to improve.

4 Related Works

This paper locates in the research filed of question generation and reinforcement learning for sequence generation. We will focus on related works from these two domains.

Question Generation Question generation has been researched for years from textual input (Rus et al., 2010; Heilman, 2011). Researchers start from rule-based method that extracts key aspects from the input text and then insert these aspects into human generated templates for interrogative sentence generation (Heilman, 2011). Recently, sequence-to-sequence model is utilized for question generation in description-question pairs (Du et al., 2017; Tang et al., 2017; Serban et al., 2016). Although these models generate better performance, the characteristics of question is still ignored. On the other hand, research about visual question generation is much less (Ren et al., 2015; Vijayakumar et al., 2018;

Mostafazadeh et al., 2016; Shijie et al., 2017). Diversity as another important characteristic of question also draws much attention. Li et al. (2016) proposed to use Maximum Mutual Information (MMI) as the objective function for result diversification. Vijayakumar et al. (2018) proposed a diverse beam search for generated multiple questions. Fan et al. (2018) utilized question type driven framework to diversify question generation.

Natural question generation cares more about a specific attribute of the generated text in terms of content. Although some attempts have been explored for this task, researchers ignore the attribute of *natural* in general. In our work, we treat this task as language generation with an additional attribute and propose a reinforcement learning framework for it. Our framework can also be used to other language generation tasks like dialogue generation with emotion information.

Reinforcement Learning For Sequence Generation *MIXER* (Ranzato et al., 2015) model uses REINFORCE method with a baseline reward estimator, directly optimizes BLEU-4 score of generated sequence. In order to compensate same reward for all action in generation and lack of ground-truth sequence knowledge, curriculum learning is utilized. Although performance in corresponding metric gets better, but model still lacks robustness and distinguishing reward for every generate step. Actor-Critic algorithm (Konda and Tsitsiklis, 2000) in sequence generation (Bahdanau et al., 2017) utilizes value network to estimate reward for every generate step. Rennie et al. (2016) utilizes self-critic $\arg\max$ to estimate generator's reward more accurately and does not need additional value network. Different reward function also gets attention for better evaluation and robustness. Liu et al. (2017) uses a combination of different metrics as the reward. In the task of visual caption generation, visual-semantic embedding (Frome et al., 2013; Kiros et al., 2015) is used to be reward (Ren et al., 2017) for better matching image and caption. SeqGAN (Yu et al., 2017), RankGAN (Lin et al., 2017a), LeakGAN (Guo et al., 2018) are also based on Reinforcement Learning, they aim to generate sequence more similar to human-written ones, thus the similarity is assigned as reward to generator.

In our setting, we borrow the idea of GAN to train one of our discriminators. Adversarial training is effective in improving generator's capability in general tasks, and it helps our generator get better guidance of natural attribute. Most of existing research under reinforcement learning focuses on using a single reward to guide the training of the generator. In this paper, we propose a setting of bi-discriminator to consider two attributes of target text and it is easy to be generalized to multiple discriminators incorporating other information.

5 Conclusion and Future Work

In this paper, we propose a reinforcement learning framework for natural question generation. It incorporates two discriminators to take two specific attributes of natural question into consideration. Experimental results on a benchmark VQG dataset show the effectiveness and robustness of our proposed model. The proposed framework can be applied to other language generation tasks with additional attribute, such as dialogue generation with emotion information. Besides, the setting of bi-discriminators can be extended to multi-discriminators to incorporate more information.

The future research can be carried out in several directions. First, during our experiment, we find that the time and space consumption of Monte Carlo Rollout is expensive. More effective and powerful methods of assigning reward for every generate step deserve research in the future. Second, We will explore more structural scoring system and better collaborative method of multiple discriminators. Third, another future direction is to incorporate some automatic evaluation metrics into our reinforcement learning framework to improve the performance further.

Acknowledgements

The work is partially supported by National Natural Science Foundation of China (Grant No.61702106), Shanghai Science and Technology Commission (Grant No.17JC1420200, Grant No.17YF1427600 and Grant No.16JC1420401).

References

VM Aleksandrov, VI Sysoyev, and SHEMENEV. VV. 1968. Stochastic optimization. *Engineering Cybernetics*, (5):11–+.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *Proceedings of ICLR*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Kyunghyun Cho, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Association for Computational Linguistics (ACL)*.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. 2013. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pages 2121–2129.

Peter W Glynn. 1990. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84.

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

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In *AAAI*, pages 2852–2858.

Michael Heilman. 2011. *Automatic factual question generation from text*. Ph.D. thesis, Carnegie Mellon University.

Chris Brockett Jianfeng Gao Jiwei Li, Michel Galley. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of NAACL-HLT*.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2015. Unifying visual-semantic embeddings with multimodal neural language models. In *Transactions of the Association for Computational Linguistics (TACL)*.

Vijay R Konda and John N Tsitsiklis. 2000. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 71–78. Association for Computational Linguistics.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Kevin Lin, Dianqi Li, Xiaodong He, Ming-ting Sun, and Zhengyou Zhang. 2017a. Adversarial ranking for language generation. In *Advances in Neural Information Processing Systems*, pages 3158–3168.

Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. 2017b. Feature pyramid networks for object detection. In *CVPR*, pages 936–944. IEEE Computer Society.

Siqi Liu, Zhenhai Zhu, Ning Ye, Sergio Guadarrama, and Kevin Murphy. 2017. Improved image captioning via policy gradient optimization of spider. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1802–1813, Berlin, Germany, August. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732.

Mengye Ren, Ryan Kiros, and Richard Zemel. 2015. Exploring models and data for image question answering. In *Advances in neural information processing systems*, pages 2953–2961.

Zhou Ren, Xiaoyu Wang, Ning Zhang, Xutao Lv, and Li-Jia Li. 2017. Deep reinforcement learning-based image captioning with embedding reward. In *Proceeding of IEEE conference on Computer Vision and Pattern Recognition (CVPR)*.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2016. Self-critical sequence training for image captioning. *CoRR*, abs/1612.00563.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2010. The first question generation shared task evaluation challenge. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 251–257. Association for Computational Linguistics.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 588–598, Berlin, Germany, August. Association for Computational Linguistics.

Zhang Shijie, Qu Lizhen, You Shaodi, Yang Zhenglu, and Zhang Jiawan. 2017. Automatic generation of grounded visual questions. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4235–4243.

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

Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *CoRR*, abs/1706.02027.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search: Decoding diverse solutions from neural sequence models. In *Proceedings of AAAI*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858.

Li Zhang, Flood Sung, Feng Liu, Tao Xiang, Shaogang Gong, Yongxin Yang, and Timothy M. Hospedales. 2017. Actor-critic sequence training for image captioning. *CoRR*, abs/1706.09601.

Fan Zhihao, Wei Zhongyu, Li Piji, Lan Yanyan, and Huang Xuanjing. 2018. A question type driven framework to diversify visual question generation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-18*.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4995–5004.

Embedding Words as Distributions with a Bayesian Skip-gram Model

Arthur Bražinskas[1] **Serhii Havrylov**[2] **Ivan Titov**[1,2]
[1]ILLC, University of Amsterdam
[2]ILCC, School of Informatics, University of Edinburgh
arthur.brazinskas@gmail.com s.havrylov@ed.ac.uk ititov@inf.ed.ac.uk

Abstract

We introduce a method for embedding words as probability densities in a low-dimensional space. Rather than assuming that a word embedding is fixed across the entire text collection, as in standard word embedding methods, in our Bayesian model we generate it from a word-specific prior density for each occurrence of a given word. Intuitively, for each word, the prior density encodes the distribution of its potential 'meanings'. These prior densities are conceptually similar to Gaussian embeddings of Vilnis and McCallum (2015). Interestingly, unlike the Gaussian embeddings, we can also obtain context-specific densities: they encode uncertainty about the sense of a word given its context and correspond to the approximate posterior distributions within our model. The context-dependent densities have many potential applications: for example, we show that they can be directly used in the lexical substitution task. We describe an effective estimation method based on the variational autoencoding framework and demonstrate the effectiveness of our embedding technique on a range of standard benchmarks.

1 Introduction

Distributed representations of words induced from large unlabeled text collections have had a large impact on many natural language processing (NLP) applications, providing an effective and simple way of dealing with data sparsity. Word embedding methods typically represent words as vectors in a low-dimensional space (Deerwester et al., 1990; Collobert et al., 2011; Mikolov et al., 2013; Pennington et al., 2014). In contrast, we encode them as probability densities (see Figure 1). Intuitively, the densities represent the distributions over possible meanings[1] of a word. Representing a word as a distribution provides many potential benefits. For example, such embeddings let us encode generality of terms (e.g., kakapo is a hypernym of bird), characterize uncertainty about semantic properties of the corresponding referent (e.g., a proper noun, such as John, encodes little about the person it refers to) or represent polysemy (e.g., kiwi may refer to a fruit, a bird or a New Zealander).

The main inspiration for this work is Gaussian embeddings (*word2gauss*, W2G) introduced by Vilnis and McCallum (2015). They represent words as Gaussian distributions and directly optimize an objective expressed in terms of divergences (e.g., the Kullback-Liebler divergence) between the distributions. In contrast, we approach the problem from the generative Bayesian perspective. Though, as in W2G, context-agnostic densities are present in our model (they correspond to data-dependent priors), unlike W2G, we can also perform posterior inference and obtain context-specific densities. These posterior densities encode semantic properties of a word in a given context. For example, in Figure 1, when 'kiwi' appears in a context suggesting the 'bird' sense, the posterior (represented by the shaded ellipsoid) becomes more 'peaky' and moves towards the representation of word 'bird'. We use a lexical substitution task (McCarthy and Navigli, 2007) to demonstrate that the posterior densities are effective in predicting potential replacements of a word given a context. Importantly, though the Gaussian assumption for context-agnostic embeddings is questionable (e.g., polysemous words would need multimodal distributions to accurately represent their meanings), the same assumption for the context-specific posterior

[1]We use the term 'meaning' somewhat liberally in this work: we are not arguing that embeddings capture actual meanings.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1775–1789
Santa Fe, New Mexico, USA, August 20-26, 2018.

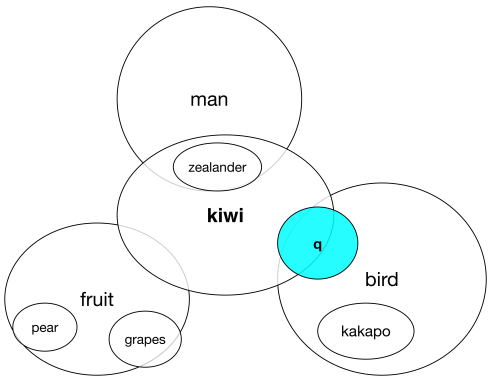


Figure 1: An idealized illustration of density embeddings. Unshaded ellipsoids encode prior densities. The shaded ellipsoid corresponds to the posterior for 'kiwi' when it appears in a context indicating that 'kiwi' refers to a bird.

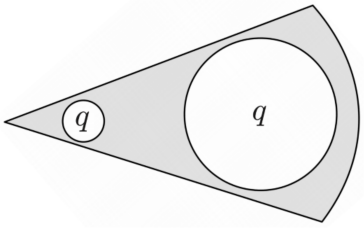


Figure 2: Shaded cone is a fixed angle, and ellipses are approximate posterior Gaussian distributions. The corner of the cone is at the origin.

densities is more reasonable: the word is likely to be disambiguated by the provided context and hence does not require complex families of distributions.

In principle, using densities to represent words provides a natural way of encoding entailment: the decision regarding entailment relation can be made by testing the level sets of the distributions for soft inclusion. For example, in Figure 1, the ellipse for 'kakapo' lies within the ellipse for 'bird'. Vilnis and McCallum (2015) proposed to use the KL divergence to detect entailment. In our analysis, we observe that, though the covariances indeed encode information relevant to entailment, their direct use is somewhat problematic both with W2G and with our model.

We are not the first to propose a Bayesian version of word embedding methods (Zhang et al., 2014; Sakaya, 2015; Barkan, 2017). However, our approach is crucially different from the previous work. The previous methods can be regarded as applications of Bayesian matrix factorization (BMF) (Salakhutdinov and Mnih, 2008) to word co-occurrence matrices. Hence they assume that every word is associated with a fixed (but unknown) real-valued vector which is shared across the entire text collection. Instead, in our approach, we acknowledge that word meaning inherently depends on a context and do not assume that any fixed vector exists at type level: we draw it at token level (i.e. for each word occurrence) from a parameterized word-specific prior distribution. Consequently, unlike us and also unlike densities in W2G, the posteriors in previous Bayesian embeddings models encode uncertainty about the embedding parameters; they will converge to a delta distribution as the amount of data increases. In contrast, our densities encode the distribution of senses and, as confirmed in our experiments, do not follow this trend.

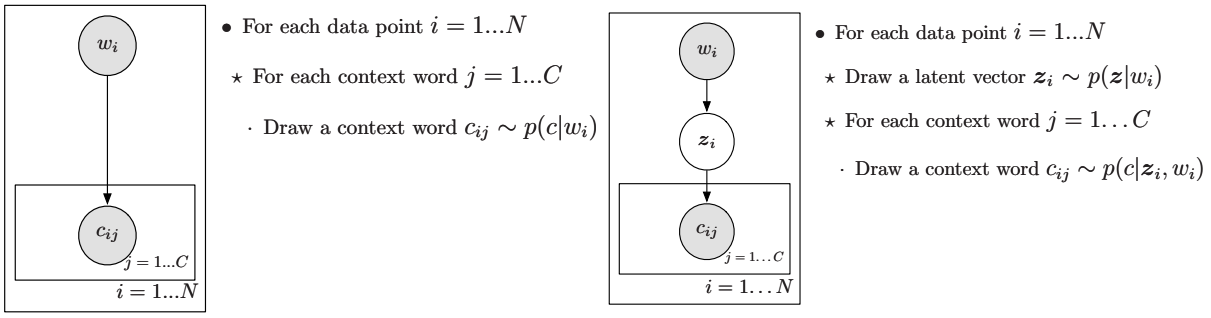
More formally, our model can be regarded as a form of coupled matrix factorization where each sliding window is factorized individually, but priors for words are shared across all the windows. We describe an effective estimation method based on the variational autoencoding (VAE) framework (Kingma and Welling, 2014). The context-specific densities are provided by the encoder component of VAE (i.e. the inference network). Our main contributions can be summarized as follows:[2]

- we proposed a Bayesian model for embedding words as probability densities;
- we derived a computationally-efficient inference algorithm, which, as a by-product, yields context-sensitives densities;
- we demonstrate their effectiveness, including on the lexical substitution task.

2 Bayesian Skip-gram

In this section, we provide detailed description of the novel model. To motivate our Bayesian extension of the Skip-gram model, consider the polysemous word 'kiwi' (Figure 1). Its meaning changes depending on the context: for example, when it appears in 'I like apples, kiwi, and bananas', we can

[2]The implementations of our model is available at `https://github.com/ixlan/BSG`.



(a) SG

(b) BSG

Figure 3: Bayesian networks corresponding to Skip-gram and Bayesian skip-gram.

deduce that 'kiwi' refers to a fruit. Polysemy cannot be effectively captured with a single representation induced by standard word embedding methods (e.g., skipgram (Mikolov et al., 2013)) or even with a single distribution, as in W2G. We also do not want to assume that there is a finite set of discrete senses, as made in multi-sense mixture models (Li and Jurafsky, 2015; Neelakantan et al., 2014). Sense distinctions are often gradual and discrete senses are only clusters that approximate underlying meaning distributions (Kilgarriff, 1997; Erk and McCarthy, 2009). Our Bayesian Skip-gram (BSG) model addresses this problem. It predicts a distribution of 'meanings' given a context. Intuitively, if the context is very discriminative, this density becomes very peaky, assigning the entire probability mass to the predicted 'meaning' (in the limit, encoding the word as a point vector). If the context is less informative, the distribution will remain flat, representing uncertainty about the word sense.

2.1 Generative model

The skip-gram (SG) model aims at maximizing the probability $p_{\boldsymbol{\theta}}(\mathbf{c}|w)$ of words in a context window given its central word. The log probability of each context word is assumed to be proportional to the dot product of its representation and that of the central word (see the graphical model in Figure 3a). In contrast, BSG assumes that the choice of context words is dependent on the context-specific (latent) meaning of the central word (see Figure 3b).

The generative story of BSG is the following: take a word from the dataset (e.g. 'kiwi'), sample its latent meaning $\mathbf{z} \sim p_{\boldsymbol{\theta}}(\mathbf{z}|w)$ (e.g., a vector that encodes the meaning 'bird'), and finally draw context words $c \sim p_{\boldsymbol{\theta}}(c|\mathbf{z})$ (e.g., 'flightless', 'forest', and 'feather').

2.2 Model definition

Ideally, we would like to maximize the log-likelihood, which is the sum of the following terms, one per each window:

$$\log p_{\boldsymbol{\theta}}(\mathbf{c}|w) = \log \int \prod_{j=1}^{C} p_{\boldsymbol{\theta}}(c_j|\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{z}|w) \mathrm{d}\boldsymbol{z} \tag{1}$$

where C is the size of the context window $\mathbf{c}$, c_j is a context word for the central word w, and θ are model parameters. Unfortunately, as $p_{\boldsymbol{\theta}}(c_j|\mathbf{z})$ is a neural network, integration over the latent space is intractable. Hence, the marginal log-likelihood and its derivatives cannot be efficiently computed. In our case posterior distribution $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{c}) = p_{\boldsymbol{\theta}}(\mathbf{c}|\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{z}) / p_{\boldsymbol{\theta}}(\mathbf{c})$ is intractable as well. This means that the expectation-maximization (EM) algorithm cannot be used. Instead, we rely on variational inference, specifically the variational auto-encoding (VAE) framework (Kingma and Welling, 2014).

2.3 Bayesian Skip-gram ELBO

For BSG model we optimize the variational lower bound of the marginal likelihood:

$$\log p_{\boldsymbol{\theta}}(\mathbf{c}|w) \geq \sum_{j=1}^{C} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{c},w)} \left[\log p_{\boldsymbol{\theta}}(c_j|\mathbf{z})\right] - \mathbb{D}_{KL}\left[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{c},w) \| p_{\boldsymbol{\theta}}(\mathbf{z}|w)\right] \tag{2}$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{c}, w)$ is the approximate posterior distribution, which can be used to infer the representation ('meaning') of the central word w in the context $\mathbf{c}$. For now, we will assume that the approximate posterior is a Gaussian distribution with a diagonal covariance matrix, namely $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{c}, w) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$. The covariance matrix of the approximate posterior encodes uncertainty or generality of meaning of the central word w in the context $\boldsymbol{c}$. Intuitively, if the meaning is concrete, we would like to get small variance values and large, otherwise.

Another important component of our model is $p_{\boldsymbol{\theta}}(\mathbf{z}|w)$, to which we will refer as a *prior*, and we will assume that it is also a Gaussian distribution with a diagonal covariance[3] and individual parameters for each central word $p_{\boldsymbol{\theta}}(\mathbf{z}|w) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$. Fortunately, the KL divergence term between two normal distributions can be expressed in closed form.

2.4 Reconstruction error

The most troublesome component of the lower bound is the expected reconstruction term $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{c},w)}\left[\log p_{\boldsymbol{\theta}}(c_j|\mathbf{z})\right]$ which can be decomposed as shown in Eq. (3):

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{c},w)}\left[\log p_{\boldsymbol{\theta}}(c_j|\mathbf{z})\right] = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{c},w)}\left[\log \frac{\exp(f_{\boldsymbol{\theta}}(\mathbf{z}, c_j))}{\sum_{k=1}^{|V|}\exp(f_{\boldsymbol{\theta}}(\mathbf{z}, c_k))}\right] \tag{3}$$

where $p_{\boldsymbol{\theta}}(c_j|\mathbf{z})$ is represented by a neural network with the softmax activation function and $f_{\boldsymbol{\theta}}(\mathbf{z}, c_j)$ is a function that models relationship between latent vector $\mathbf{z}$ and context word c_j. In the original skip-gram model it was the dot product. The dot product is based on the angle between two vectors scaled by their norms, which makes a lot of sense with point estimates. However, it is problematic when we use normal distributions. Consider the fixed angle cone and two densities in Figure 2. Both densities encode the same uncertainty about the angle, whereas their variances are very different. Hence, the uncertainty in the dot product can be increased either by moving the density towards the origin or by increasing the variance. In other words, the model has too many degrees of freedom. Consequently, words that are more general will not necessarily have a large variance. We observed this behaviour in our toy experiment.

To address this problem, we propose to use a different form of the function f: $f_{\boldsymbol{\theta}}(\mathbf{z}, c_j) = \log\left(\mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{c_j}, \boldsymbol{\Sigma}_{c_j}) \times p(c_j)\right)$:[4]

$$\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log \frac{\exp(f_{\boldsymbol{\theta}}(\mathbf{z}, c_j))}{\sum_{k=1}^{|V|}\exp(f_{\boldsymbol{\theta}}(\mathbf{z}, c_k))}\right] = \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log\left(\mathcal{N}(\mathbf{z};\boldsymbol{\mu}_{c_j}, \boldsymbol{\Sigma}_{c_j})p(c_j)\right) - \log\sum_{k=1}^{|V|}\mathcal{N}(\mathbf{z};\boldsymbol{\mu}_{c_k}, \boldsymbol{\Sigma}_{c_k})p(c_k)\right] \tag{4}$$

Intuitively, scaling with $p(c_j)$ (e.g., the empirical unigram probability) encodes that frequent words are more likely to appear in any context. The Gaussian density evaluates how well a word fits context given the meaning $\mathbf{z}$ of the central word and is more sensitive to the variance, hence addressing the above-mentioned problem.

2.5 Encoder

An important component of our model is the inference network $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{c}, w)$ (also known as an encoder), which approximates the posterior distribution $p_\theta(\mathbf{z}|\mathbf{c}, w)$. As we discussed, we make the Gaussian assumption for q. Similarly to Kingma and Welling (2014), we use a one-layer feed-forward neural network to compute variance and mean parameters of q. Its architecture is shown in Figure 4. The network takes embeddings of the central word and those of context words, passes each of them through a non-linearity (ReLU) and sums the vectors together: $\mathbf{h} = \sum_{j=1}^{C} relu\left(\mathbf{M}\begin{bmatrix}\mathbf{R}_{c_j}\\ \mathbf{R}_w\end{bmatrix}\right)$. The resulting vector is then used to generate $\boldsymbol{\mu}_q$ and $\log\sigma^2$ with a linear model: $\log\sigma_q^2 = \mathbf{W}\mathbf{h} + b_2; \boldsymbol{\mu}_q = \mathbf{U}\mathbf{h} + \mathbf{b}_1$. Using logarithm is necessary to ensure that the matrix is positive definite. The encoder can be understood as a parametrized

[3]Though this assumption is questionable for polysemous words, what is crucial is that in BSG (unlike W2G) we have access to approximate context-specific posterior densities encoding word senses. Nevertheless, using more expressive priors may be beneficial. Expressive data-dependent priors may also lead to over-fitting as using simple priors can be regarded as a form of regularization. We leave investigation of richer priors for future work.

[4]For brevity, we use q_ϕ to denote $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{c}, w)$.

function that produces distributions corresponding to meanings of central words in different contexts. Using this function, we can model infinitely many meanings of words without explicitly storing their representations.

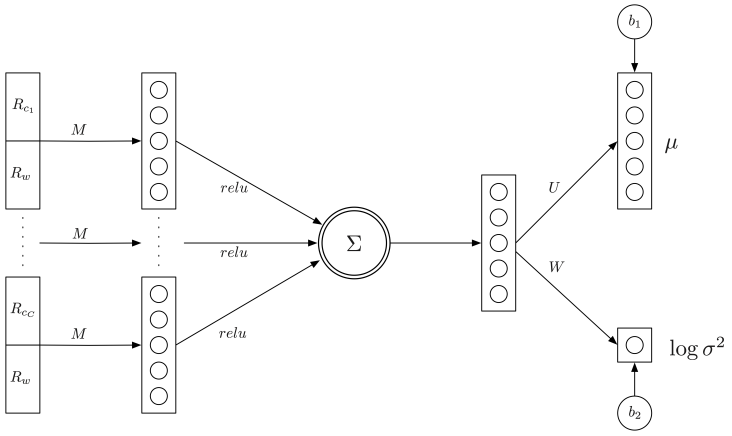


Figure 4: Encoder architecture.

2.6 Approximation of the log-partition function

While we can obtain a low-variance estimate of the first part of the expectation from the Eq. (4) using the re-parameterization trick (Kingma and Welling, 2014), the expected log-partition function (the second part) is very computationally demanding, as it involves summation over all words in the vocabulary. In order to scale it to large vocabularies, we compute the lower bound of the log-partition function and use a Monte Carlo (MC) approximation to obtain an unbiased estimate of the expectation over all words:

$$\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log \mathbb{E}_{p(\tilde{c})}\left[\mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\tilde{c}}, \boldsymbol{\Sigma}_{\tilde{c}})\right]\right] \geq \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\mathbb{E}_{p(\tilde{c})}\left[\log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\tilde{c}}, \boldsymbol{\Sigma}_{\tilde{c}})\right]\right] \approx \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\tilde{c}_j}, \boldsymbol{\Sigma}_{\tilde{c}_j})\right] \quad (5)$$

The estimate involves only one word $\tilde{c}_j$ ('negative word') which can be easily sampled from $p(\tilde{c})$. By replacing the log-partition function from Eq. (4) with the unbiased estimate of its lower bound derived in Eq. (5), we can get the approximate lower-bound of marginal log-likelihood[5] shown in Eq. (6). The sum $\sum_{(j,k)}$ in the equation is now over pairs of positive c_j and negative $\tilde{c_k}$ context words. Furthermore, we can transform the difference of negative cross-entropies into the difference of Kullback-Leibler divergences by adding and subtracting entropy of $q_\phi(\mathbf{z}|\mathbf{c}, w)$:

$$\begin{aligned}\hat{\mathcal{L}} = &\sum_{(j,k)} \left(\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{c_j}, \boldsymbol{\Sigma}_{c_j})\right] - \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\tilde{c_k}}, \boldsymbol{\Sigma}_{\tilde{c_k}})\right]\right) - \mathbb{D}_{KL}\left[q_{\boldsymbol{\phi}} \| p_{\boldsymbol{\theta}}(\mathbf{z}|w)\right] = \\ &\sum_{(j,k)} \left(\mathbb{D}_{KL}\left[q_{\boldsymbol{\phi}} \| \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\tilde{c}_k}, \boldsymbol{\Sigma}_{\tilde{c}_k})\right] - \mathbb{D}_{KL}\left[q_{\boldsymbol{\phi}} \| \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{c_j}, \boldsymbol{\Sigma}_{c_j})\right]\right) - \mathbb{D}_{KL}\left[q_{\boldsymbol{\phi}} \| p_{\boldsymbol{\theta}}(\mathbf{z}|w)\right]\end{aligned} \quad (6)$$

The first term in the above equation suggests maximization of the margin between divergences that involve negative and positive words. We transform it into the hard-margin form (i.e. use the hinge loss, as done in Weston et al. (2013)). This is the final objective function that we maximize:

$$\sum_{(j,k)} \max\left(0,\ \mathbb{D}_{KL}\left[q_{\boldsymbol{\phi}} \| \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{c_j}, \boldsymbol{\Sigma}_{c_j})\right] - \mathbb{D}_{KL}\left[q_{\phi} \| \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{\tilde{c}_k}, \boldsymbol{\Sigma}_{\tilde{c}_k})\right] + m\right) + \mathbb{D}_{KL}\left[q_{\boldsymbol{\phi}} \| p_{\boldsymbol{\theta}}(\mathbf{z}|w)\right] \quad (7)$$

This transformation of the objective is necessary because the KL terms are unconstrained, and, during optimization, the model tends to assign extreme values for the frequent negative words and this has a detrimental effect on the overall performance. The hinge loss solves this problem by setting the gain to zero when the KL term involving a negative context word is larger by a margin than the KL term involving a positive one.

[5]Notice that $p(c_j)$ is omitted because it factors out as a constant and does not change the optimum.

Datasets	BSG	WG(S)	WG(D)	SG
MC-30	0.71	0.69	0.70	**0.72**
MEN-TR-3k	**0.73**	0.72	0.71	0.72
MTurk-287	0.70	0.70	0.69	**0.70**
MTurk-771	**0.67**	0.65	0.64	0.65
RG-65	0.70	0.69	0.71	**0.72**
RW-STNFRD	0.43	0.43	0.42	**0.44**
SIMLEX-999	**0.35**	0.34	0.34	0.34
VERB-143	0.32	**0.38**	0.29	0.36
WS-353-ALL	**0.72**	0.68	0.67	0.69
WS-353-REL	**0.68**	0.66	0.65	0.65
WS-353-SIM	**0.75**	0.70	0.68	0.71
YP-130	**0.50**	0.46	0.46	0.45
Sum	**7.26**	7.10	6.95	7.15

Table 1: Word similarity benchmarks.

Model	GAP
BSG (Encoder)	**0.461**
BSG (Add)	0.437
BSG (Mult)	0.439
W2G(S) (Add)	0.431
W2G(S) (Mult)	0.432
W2G(D) (Add)	0.427
W2G(D) (Mult)	0.427
SG (Add)	0.426
SG (Mult)	0.428

Table 2: Lexical substitution task. Average precision for SemEval-07 dataset.

The intuition behind the objective is the following: once we generated the posterior distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{c}, w)$, we optimize parameters to make $\mathbb{D}_{KL}\left[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{c}, w)||p_{\boldsymbol{\theta}}(\mathbf{z}|w)\right]$ small, which can be intuitively understood as a regularization preventing $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{c}, w)$ ellipsoid from diverging from $p_{\boldsymbol{\theta}}(\mathbf{z}|w)$ ellipsoid. In addition, we discriminate positive context words from negative ones by comparing their divergences from $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{c}, w)$.

The expected value of the log-partition function in Eq. (3) is with the negative sign, and because we approximate its lower-bound, the resulting objective function in Eq. (6) is not a lower bound of the likelihood anymore. We still found that proposed approximation works quite well in practice. As an alternative, we considered a Monte Carlo approximation of the partition function proposed by Botev et al. (2017). However, this led to inferior performance on our benchmarks.

3 Experiments

In this section we empirically evaluate our approach. First, in subsections 3.2 - 3.4, we use standard benchmarks to evaluate and better understand properties of our context-agnostic embeddings (i.e. the prior distributions $\mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_{c_j}, \boldsymbol{\Sigma}_{c_j})$). In these experiments, we can directly compare BSG to W2G and the vanilla version of skip-gram. In subsection 3.5, we also demonstrate that the context-sensitive posterior distributions $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{c}, w)$, estimated by the VAE inference network, can be used to select potential substitutes of a word in a given context.

3.1 Experimental settings

We train all our models on a concatenation of ukWaC and WaCkypedia (Baroni et al., 2009) corpora, resulting in approximately 3 billion tokens. The embedding dimensionality was set to 100 for all the models. For BSG, we used spherical covariances both for the posterior and prior densities, as they resulted in better performance in preliminary experiments: this is consistent with results reported for W2G in Vilnis and McCallum (2015). In order to enable fair comparison, we used our own implementation of W2G and SG, which shared the hyperparameters with BSG.[6] Throughout the text we use WG(S) and WG(D) to denote W2G with the spherical covariances and W2G with the diagonal covariance matrix, respectively. For additional details see supplementary materials.

[6]The original implementation of W2G is not publicly available. Our re-implementation yields stronger results on similarity benchmarks but weaker on the entailment dataset of Baroni et al. (2012). Our implementation is also stronger across the board (including entailment) than the third-party implementation used in Vulić et al. (2017) (`https://github.com/seomoz/word2gauss`). Athiwaratkun and Wilson (2017) also report W2G numbers very similar to ours.

	BBDS		BLESS	
Model	KL	Cos	KL	Cos
BSG	76.2	**75.9**	20.0	20.8
W2G(S)	**77.0**	75.7	18.9	20.4
W2G(D)	76.5	74.9	18.7	20.3
SG	-	75.7	-	20.3

Table 3: F1 scores on entailment recognition.

Model	BBDS	BLESS
BSG	78.23	**67.34**
W2G(S)	78.41	57.50
W2S(D)	78.05	54.58
Freq. Baseline	**78.84**	55.26

Table 4: Entailment directionality detection.

3.2 Word similarity

Table 1 presents similarity results computed using the online tool of Faruqui and Dyer (2014). In these experiments, we used only the mean vectors (from the prior) and ignored the covariance information, both for BSG and W2Gs. First, we observe that BSG has a slight edge over both the original SG and also over W2G versions. Second, contrary to observations in Vilnis and McCallum (2015), W2G does not reach a performance of SG in our experiments. As their original implementation is not available, it is hard to pinpoint the reason for the discrepancy. Note that their SG baseline is considerably weaker than ours, so it may be easier to beat. Our SG baselines outperforms their reported results on all similarity datasets (they considered only 8 out of 12 in Table 1) and also in average (0.6248 vs. 0.5724).[7] The results suggest that prior means induced by BSG are indeed effective in capturing semantic properties of words.

3.3 Entailment recognition

In this section, we consider the lexical entailment task (i.e. essentially hyponymy detection). Note that, as with word similarity, word context is not provided, and, thus, we cannot showcase the main advantage of our approach: its disambiguation capabilities.

Given an ordered pair of words, the task is to predict if the first word (w_1) entails the second one (w_2). We use two entailment measures: the negated KL divergence $-D_{KL}\left[\mathcal{N}(\mathbf{z};\boldsymbol{\mu}_{w_1},\boldsymbol{\Sigma}_{w_1})\|\mathcal{N}(\mathbf{z};\boldsymbol{\mu}_{w_2},\boldsymbol{\Sigma}_{w_2})\right]$ and the cosine similarity between the means. We predict that w_1 entails w_2 if the corresponding score is above a certain threshold. As in Vilnis and McCallum (2015), the scores are optimistic, as the threshold is chosen on the test set. The thresholds are set individually for each method (including the baselines) and, hence, the comparison is fair.

Intuitively, KL should be a good choice: it would favor word pairs such that, not only their means are similar, but also such that the region, where the density function for w_1 is non-negligible, lies within the area where the density of w_2 is also high enough. Roughly speaking, level sets for w_1 should lie within level sets for w_2 (as with ('pear', 'fruit') in Figure 1).

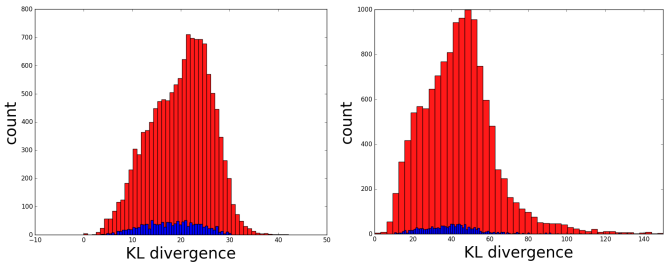
First, we consider the entailment benchmark of Baroni et al. (2012), we will refer to it as BBDS (Table 3, left part). The results are generally consistent with the ones reported by Vilnis and McCallum. The best W2G(S) model outperforms SG. They also have the edge over BSG. We also observe that, unlike W2G, covariance information appears not to be particularly beneficial for BSG (i.e. cosine performs essentially as well as KL).

Second, we turn to the BLESS dataset (Baroni and Lenci, 2011).[8] All the considered methods perform badly on BLESS. This is even more evident from Figure 5, where we plot the histogram of the KL divergence for the entailing and not entailing pairs. Clearly, the two classes cannot be separated based on KL alone, neither for BSG, nor W2G.

We hypothesize that the reason for these, seemingly inconsistent results, is that all considered methods struggle with distinguishing hypernymy ('dog' and 'pet') from co-hyponymy ('dog' and 'cat'), as well as from other types of semantic relatedness. Making such distinction is a challenging for an unsupervised method (Weeds et al., 2014). Indeed, the BLESS dataset is harder than BBDS: it is unbalanced and

[7] The popular Gensim SG implementation is even slightly stronger than ours (0.6365), though not really comparable because of major differences in optimization.

[8] The version we are using is from Levy et al. (2015)



(a) BSG (b) W2G(S)

Figure 5: Histograms of KL scores on BLESS. Blue and red bars correspond to 'entailment' and 'no entailment' pairs, respectively.

word 1	word 2	KL	cosine sim.
dog	cat	15.47	0.71
dog	pet	18.52	0.70
dog	hound	21.20	0.64
dog	animal	27.69	0.52

Table 5: KL and cosine similarity of selected words. We used $D_{KL}\left[p(w_1)||p(w_2)\right]$, where $p(w)$ is the prior density for word w.

contains about ten times more negatives examples than positive ones, and the negative examples are often co-hyponyms.

So, why does KL not work as well as we expected? Intuitively, KL can be regarded as balancing two trade-offs: penalizing for divergences between means and also penalizing for the 'wrong' type of 'inclusion' between the distributions. It is very easy to see this for the one-dimensional case, where KL can be written as: $\log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2+(\mu_1-\mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$, but holds for the multivariate case as well. If the variances are sufficiently similar, then the distance between means is all that matters. We hypothesize that for the Gaussian priors within BSG and for both versions of W2G, covariances end up playing a relatively minor role. Indeed, when we checked for correlations between KL and cosine similarity, we observed that it is very strong, confirming our hypothesis. For example, dog entails animal, but dog does not entail cat, while scores in Table 5 indicate the opposite (for more examples see supplementary materials). Generally, the Pearson correlation coefficient between cosine similarity and KL equals -0.652 and -0.703, for BLESS and BBDS, respectively, indicated a very strong (linear) relation between the two. Here, we considered W2G(S), as it seemed the most promising on the basis of BBDS results, but the trend is the same for BSG and W2G(D).

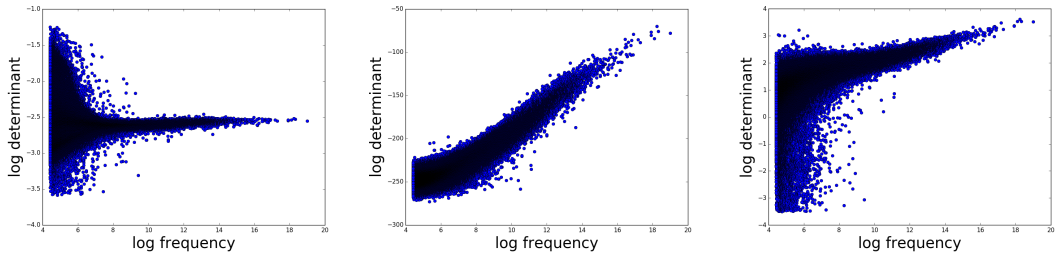
Though we see that KL on its own does not seem to be sufficient for detecting entailment, we still hypothesize that it does capture information directly relevant to this task, namely, represents a degree of 'generality' of a term. In order to see that this is the case, in next subsection, we consider a modification of the entailment task. It will also hint at why W2G performs better than BSG on BBDS, while, as we will see, BSG appears to capture generality more accurately.

3.4 Entailment directionality prediction

In these experiments, given a pair of words, the system needs to predict if the first word entails the second one or the other way around. In other words, it is known that entailment holds for the pair but its directionality needs to be predicted. As we would like to get extra insights into results obtained in the preceding section, we extract these pairs from the same datasets, BBDS and BLESS. As the symmetric cosine similarity would be useless here, we use only KL. As a baseline, we consider the following heuristic: we compare frequencies of the two words in our corpus and predict that the less frequent word (e.g., 'kiwi') entails the more frequent one (e.g., 'fruit').

Results are presented in Table 4. First, we observe that the situations are very different for the two datasets. On BBDS, the scores are much higher for all the approaches. However, depressingly, the frequency baseline appears the strongest: neither model manages to beat it. In contrast, on BLESS, the frequency baseline is weak (only 5% higher than chance), and both W2G versions achieve results very similar to the baseline. However, BSG outperforms them by a substantial margin (10%).

These results hint at the possibility that the main reason for slightly stronger results of W2Gs in the original set-up on BBDS (Table 3) is that covariances in W2Gs capture information about token frequencies. We verify this by plotting log-determinants of covariance matrices (representing the spread of the 'ellipsoids'), as a function of token frequencies (Figure 6). The plots confirm our hypothesis: covariances for W2Gs are growing with the frequency, and hence KL with W2Gs will prefer to label frequent words



(a) BSG $\rho = -0.0706$. (b) W2G(D) $\rho = 0.9098$. (c) W2G(S) $\rho = 0.5915$.

Figure 6: Log determinant vs. log frequency; ρ is the Pearson correlation coefficient.

as hypernyms. This is not the case at all for BSG. Note that BSG achieves similar results in directionality detection on BBDS without directly capturing frequency information (and hence 'mimicking' the frequency baseline). This, together with strong results on directionality detection for BLESS, suggests that covariances of BSG are better at capturing genuine information about the generality of a term.

3.5 Lexical substitution

We argued that the posterior densities in our model encode semantic properties of a word given its context. In other words, the posteriors can be used to disambiguate the word. In order to show that this is indeed the case, we consider the *lexical substitution* task, where the goal is to choose a suitable replacement for a word given its context. For example, the word 'bright' in an expression 'bright child' can be substituted with 'smart' or 'gifted' rather than 'shining'.

We used the SemEval-2007 task 10 dataset (McCarthy and Navigli, 2007). In this task, a system has to select a target word from a list of replacement candidates. We followed the set-up of Melamud et al. (2015) but kept the models the same as in the preceding sections. Namely, we did not use syntactic dependency information (shown highly beneficial in (Melamud et al., 2015)) but rather relied on a bag-of-word representation of a window of 5 words on each side. We also used 100 dimensional embeddings instead of 600 in Melamud et al. (2015).

Our model relies bag-of-word context, and the task is closely related to language modeling where representing ngrams is known to be crucial. Thus, we do not expect BSG to reach performance of state-of-the-art techniques relying on richer representations of context, such as LSTMs (e.g. (Melamud et al., 2016; Yuan et al., 2016)). We use the lexical substitution task as a way to test that the approximate posterior indeed encodes meaningful information, useful for predicting substitutions. To make the comparison fair, we use baselines which also rely on bag-of-word representations of context windows, and leave the investigation of BSG models with richer representations of context for future work.

For BSG, we ranked candidates s from the set of candidates S by the KL divergence: $\mathbb{D}_{KL}\left[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{c}, w)||p_{\boldsymbol{\theta}}(\mathbf{z}|s)\right]$, where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{c}, w)$ is the approximate posterior computed by the encoder, and $p_{\boldsymbol{\theta}}(\mathbf{z}|s)$ is the prior distribution for the candidate word. As baselines, we used the two best performing approaches from Melamud et al. (2015), namely *Add* and *Mult*. Those heuristics score potential substitutes with respect to contexts and center words using cosine similarity of word embeddings, and mathematical operations as their names suggest. We used these heuristics on top of embeddings produced by SG and mean vectors for W2Gs and BSG. Note that they do not use covariance information.

The results (*generalized average precision*, GAP) are shown in Table 2. The encoder indeed appears substantially more effective in predicting word substitutes than the alternative methods. This shows that the posterior densities are effective at disambiguating words. In Table 6, we show top 3 substitutes proposed by the BSG encoder in several example sentences.

4 Related Work

Our model bears some relation to topic models: for example, one can interpret the latent meaning vector z of the central word as encoding topics of the window. Topic models relying on word embeddings has been considered in the past (Das et al., 2015; Li et al., 2016). However, their modeling approaches,

Excerpts	Top 3 Substitutes
at that size it would have a **mass** of about the same as an average galaxy	conglomeration, magnitude, bulk
few people parallels the growing poverty of the **masses**	multitude, proletariat, throng
he was **really** he always wanted to be a politician	genuinely, very, definitely
it is his passion and to be denied that is **really** hard	absolutely, very, truly
between shutdown and power up when the latchup **cross** section is evaluated	sequence, segment, transverse
and hurting like the torments of hell for being so close to the **cross**	crucifix, sequence, angry
something more sedate there are quieter sophisticated **bars** in the hotels	pub, saloon, lounge
put granola **bars** in bowl	snack, pub, biscuit
i never expected to be **touched** by a weird global media event personally	affect, feel, stir
these stripes are continuous and do not **touch** each other	contact, finger, stroke

Table 6: Substitutes proposed by the BSG encoder on SemEval-2007.

inference methods and applications have been quite different. Methods for inducing context-sensitive representations relying on syntactic dependency tree (using forward-backward-style algorithms) have also been studied in the past (Grave et al., 2014). Their approach is unlikely to be scalable. Very recent work of Peters et al. (2018) showed how LSTMs can be trained to provide dynamic word embeddings. These embeddings are context-aware but do not explicitly represent uncertainty about word meanings.

Reisinger and Mooney (2010) introduced a method that represents words as collections of prototype vectors. Their multi-prototype approach uses word sense discovery to partition contexts of a word and construct "sense-specific" prototypes for each cluster. Huang et al. (2012) extended this approach by incorporating global document context to learn multiple dense, low-dimensional embeddings using neural networks. This method performs word sense discrimination as a preprocessing step by clustering contexts for each word type. To model polysemous words Neelakantan et al. (2014) proposed an extension to the skip-gram model that efficiently learns multiple embeddings per word type. Their method performs word sense discrimination and embedding learning by using a nonparametric estimate of the number of senses per word type. In contrast, our model does not assume that there is a finite set of discrete senses per word. Liu et al. (2015) introduced an extension to the skip-gram model that can model the interaction between words and topics under different contexts using a tensor layer.

Our method and W2G are specifically designed to induce information about the generality of a word and capture entailment. Previous work has shown that entailment decision can also be made by 'post-processing' representations produced by standard embedding methods (e.g., (Weeds et al., 2014; Henderson and Popa, 2016)). Henderson and Popa (2016) is perhaps the most related one, as they use a probabilistic motivation for their preprocessing technique. All these post-processing methods deal with fixed rather than context-specific embeddings.

5 Conclusion

We introduced a method for embeddings words as probability densities. Our method produces two types of embeddings: (1) 'prior' / static embeddings representing a word type and collapsing all word senses; (2) 'posterior' / dynamic embeddings encoding a representation of a word given its context. The Gaussian embeddings of Vilnis and McCallum (2015) have been shown effective in a range of applications (e.g., modeling knowledge graphs (He et al., 2015) and cross-modal transfer from language to vision (Mukherjee and Hospedales, 2016)), our framework, providing more flexible context-sensitive alternatives, is likely to be beneficial in these settings.

Acknowledgments

This project is supported by SAP Innovation Center Network, NWO Vidi Grant (639.022.518) and ERC Starting Grant BroadSem (678254). We would like to thank anonymous reviewers for their helpful suggestions and comments as well as Luke Vilnis for answering questions about their model.

References

[Athiwaratkun and Wilson2017] Ben Athiwaratkun and Andrew Gordon Wilson. 2017. Multimodal word distributions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*, pages 1645–1656.

[Barkan2017] Oren Barkan. 2017. Bayesian neural word embedding. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 2017*, pages 3135–3143.

[Baroni and Lenci2011] Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10. Association for Computational Linguistics.

[Baroni et al.2009] Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.

[Baroni et al.2012] Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 23–32. Association for Computational Linguistics.

[Botev et al.2017] Aleksandar Botev, Bowen Zheng, and David Barber. 2017. Complementary sum sampling for likelihood approximation in large scale classification. In *Artificial Intelligence and Statistics*, pages 1030–1038.

[Collobert et al.2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

[Das et al.2015] Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian LDA for topic models with word embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

[Deerwester et al.1990] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

[Erk and McCarthy2009] Katrin Erk and Diana McCarthy. 2009. Graded word sense assignment. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 440–449. Association for Computational Linguistics.

[Faruqui and Dyer2014] Manaal Faruqui and Chris Dyer. 2014. Community evaluation and exchange of word vectors at wordvectors.org. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics.

[Grave et al.2014] Edouard Grave, Guillaume Obozinski, and Francis Bach. 2014. A markovian approach to distributional semantics with application to semantic compositionality. In *International Conference on Computational Linguistics (Coling)*, pages 1447–1456.

[He et al.2015] Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. 2015. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 623–632. ACM.

[Henderson and Popa2016] James Henderson and Diana Popa. 2016. A vector space for distributional semantics for entailment. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2052–2062. Association for Computational Linguistics.

[Huang et al.2012] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 873–882.

[Kilgarriff1997] Adam Kilgarriff. 1997. I dont believe in word senses. *Computers and the Humanities*, 31(2):91–113.

[Kingma and Welling2014] Diederik P Kingma and Max Welling. 2014. Auto-encoding Variational Bayes. In *Proceedings of the 3rd International Conference for Learning Representations*.

[Levy et al.2015] Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2015*, pages 970–976.

[Li and Jurafsky2015] Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Portugal, 2015*, pages 1722–1732.

[Li et al.2016] Chenliang Li, Haoran Wang, Zhiqian Zhang, Aixin Sun, and Zongyang Ma. 2016. Topic modeling for short texts with auxiliary word embeddings. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 165–174.

[Liu et al.2015] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2015. Learning context-sensitive word embeddings with neural tensor skip-gram model. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015*, pages 1284–1290.

[McCarthy and Navigli2007] Diana McCarthy and Roberto Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 48–53. Association for Computational Linguistics.

[Melamud et al.2015] Oren Melamud, Omer Levy, Ido Dagan, and Israel Ramat-Gan. 2015. A simple word embedding model for lexical substitution. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 1–7.

[Melamud et al.2016] Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, Berlin, Germany, 2016*, pages 51–61.

[Mikolov et al.2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Mukherjee and Hospedales2016] Tanmoy Mukherjee and Timothy M. Hospedales. 2016. Gaussian visual-linguistic embedding for zero-shot recognition. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016*, pages 912–918.

[Neelakantan et al.2014] Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1059–1069.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

[Peters et al.2018] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

[Reisinger and Mooney2010] Joseph Reisinger and Raymond J. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, pages 109–117.

[Sakaya2015] Joseph Hosanna Sakaya. 2015. Scalable bayesian induction of word embeddings.

[Salakhutdinov and Mnih2008] Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887.

[Vilnis and McCallum2015] Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding. In *Proceedings of the 4th International Conference for Learning Representations*.

[Vulić et al.2017] Ivan Vulić, Daniela Gerz, Douwe Kiela, Felix Hill, and Anna Korhonen. 2017. Hyperlex: A large-scale evaluation of graded lexical entailment. *Computational Linguistics*, 43(4):781–835.

[Weeds et al.2014] Julie Weeds, Daoud Clarke, Jeremy Reffin, David J. Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers*, pages 2249–2259.

[Weston et al.2013] Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1366–1371.

[Yuan et al.2016] Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers*, pages 1374–1385.

[Zhang et al.2014] Jingwei Zhang, Jeremy Salwen, Michael R. Glass, and Alfio Massimiliano Gliozzo. 2014. Word semantic representations using bayesian probabilistic tensor factorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1522–1531.

Supplementary Material for COLING 2018 Paper
Embedding Words as Distributions with a Bayesian Skip-gram Model

Arthur Bražinskas[1] **Serhii Havrylov**[2] **Ivan Titov**[1,2]
[1]ILLC, University of Amsterdam
[2]ILCC, School of Informatics, University of Edinburgh
arthur.brazinskas@gmail.com s.havrylov@ed.ac.uk ititov@inf.ed.ac.uk

1 KL divergence and cosine similarity of semantically related words

word 1	word 2	KL	cosine sim.
dog	cat	15.47	0.71
dog	pet	18.52	0.70
dog	hound	21.20	0.64
dog	animal	27.69	0.52
cappuccino	espresso	12.59	0.76
cappuccino	latte	13.39	0.7
cappuccino	coffee	22.54	0.69
cappuccino	drink	30.81	0.54
microsoft	windows	24.41	0.65
microsoft	google	24.44	0.60
microsoft	corporation	39.40	0.29
microsoft	company	46.05	0.19

Table 1: KL and cosine similarity of semantically related words. We used $D_{KL}\left[p(w_1)||p(w_2)\right]$, where $p(w)$ is the prior density for word w.

2 Experimental settings

We train all our models on a concatenation of ukWaC and WaCkypedia (Baroni et al., 2009) corpora, resulting in approximately 3 billion tokens. We restricted the vocabulary size to 280,000 most frequent word types and used the sub-sampling procedure introduced in Mikolov et al. (2013) with $t = 10^{-4}$. The window size was set to 5 words on each side, the dimensionality of the embeddings and of the hidden layer of the encoder were both set to 100. The number of negative samples was equal to the number of positive ones (i.e. 10). For BSG, we used spherical covariances both for the posterior and prior densities, as they resulted in better performance in preliminary experiments: this is consistent with results reported for W2G in Vilnis and McCallum (2015). As an optimizer we used Adam (Kingma and Ba, 2014). We used large batches of 22,000 prediction tasks each. We used grid-search by running each model for 5 times on the concatenation of ukWaC and WaCkypedia dataset with different learning rates and selected the best ones based on their benchmark performance.

In order to enable fair comparison, we used our own implementation of W2G and SG, which shared the above hyperparameters with BSG.[1] The implementations of our model and the baselines will be publicly released should the paper get accepted. The initial learning rates were tuned for all models individually and set to 0.00055, 0.0065, 0.0015 and 0.0015 for BSG, W2G with the spherical covariances

[1]The original implementation of W2G is not publicly available. Our re-implementation yields stronger results on similarity benchmarks but weaker on the entailment dataset of Baroni et al. (2012). Our implementation is also stronger across the board (including entailment) than the third party implementation used in Vulić et al. (2017) (https://github.com/seomoz/word2gauss). Athiwaratkun and Wilson (2017) also report W2G numbers very similar to ours.

(WG(S)), W2G with the diagonal covariance matrix (WG(D)) and SG, respectively. All results presented in this work were obtained from one model of each type (i.e. no tuning was performed for individual benchmarks).

References

[Athiwaratkun and Wilson2017] Ben Athiwaratkun and Andrew Gordon Wilson. 2017. Multimodal word distributions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*, pages 1645–1656.

[Baroni et al.2009] Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.

[Baroni et al.2012] Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 23–32. Association for Computational Linguistics.

[Kingma and Ba2014] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations*.

[Mikolov et al.2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Vilnis and McCallum2015] Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding. In *Proceedings of the 4th International Conference for Learning Representations*.

[Vulić et al.2017] Ivan Vulić, Daniela Gerz, Douwe Kiela, Felix Hill, and Anna Korhonen. 2017. Hyperlex: A large-scale evaluation of graded lexical entailment. *Computational Linguistics*, 43(4):781–835.

Assessing Composition in Sentence Vector Representations

Allyson Ettinger[1], **Ahmed Elgohary**[2], **Colin Phillips**[1], **Philip Resnik**[1,3]
[1]Linguistics, [2]Computer Science, [3]Institute for Advanced Computer Studies
University of Maryland, College Park, MD
{aetting,colin,resnik}@umd.edu, elgohary@cs.umd.edu

Abstract

An important component of achieving language understanding is mastering the composition of sentence meaning, but an immediate challenge to solving this problem is the opacity of sentence vector representations produced by current neural sentence composition models. We present a method to address this challenge, developing tasks that directly target compositional meaning information in sentence vector representations with a high degree of precision and control. To enable the creation of these controlled tasks, we introduce a specialized sentence generation system that produces large, annotated sentence sets meeting specified syntactic, semantic and lexical constraints. We describe the details of the method and generation system, and then present results of experiments applying our method to probe for compositional information in embeddings from a number of existing sentence composition models. We find that the method is able to extract useful information about the differing capacities of these models, and we discuss the implications of our results with respect to these systems' capturing of sentence information. We make available for public use the datasets used for these experiments, as well as the generation system.[1]

1 Introduction

As natural language processing strives toward language understanding, it is important that we develop models able to extract and represent the *meaning* of sentences. Such representations promise to be applicable across a variety of tasks, and to be more robust than non-meaning-based representations for any given task requiring meaning understanding. To accomplish meaning extraction, a particular need is that of mastering composition: systematic derivation of the meaning of a sentence based on its parts.

In this paper we tackle compositional meaning extraction by first addressing the challenge of evaluation and interpretability: after all, in order to improve meaning extraction, we need to be able to evaluate it. But with sentence representations increasingly taking the form of dense vectors (embeddings) from neural network models, it is difficult to assess what information these representations are capturing—and this problem is particularly acute for assessing abstract content like compositional meaning information.

Here we introduce an analysis method for targeting and evaluating compositional meaning information in sentence embeddings. The approach builds on a proposal outlined in Ettinger et al. (2016), and involves designing classification tasks that directly target the information of interest (e.g., "Given a noun n, verb v, and an embedding $\mathbf{s}$ of sentence s: is n the *agent* of v in s?"). By contrast to related work analyzing surface variables like word content and word order in sentence embeddings (Adi et al., 2016), we specifically target compositional meaning information relevant to achieving language understanding—and in order to isolate this more abstract information, we exert careful control over our classification datasets to ensure that we are targeting information arrived at by composition of the source

[1]Code for the generation system, as well as a pointer to the classification datasets, can be found at https://github.com/aetting/compeval-generation-system

Proceedings of the 27th International Conference on Computational Linguistics, pages 1790–1801
Santa Fe, New Mexico, USA, August 20-26, 2018.

sentence, rather than general statistical regularities. Our approach is informed by methods in cognitive neuroscience and psycholinguistics, where such controls are standard practice for studying the brain.

In particular, to ensure validity of our tests we introduce three mechanisms of control. First, to create controlled datasets at the necessary scale, we develop a generation system that allows us to produce large sentence sets meeting specified semantic, syntactic and lexical constraints, with gold-standard meaning annotation for each sentence. Second, we control the train-test split so as to require more robust generalization in order to perform the tasks successfully. Third, we employ a sanity check leveraging known limitations of bag-of-words (BOW) composition models: for any tasks requiring order information *from the source sentence*, which BOW models cannot logically retain, we check to ensure that BOW composition models are at chance performance.

These controls serve to combat a problem that has gained increasing attention in recent work: many existing evaluation datasets contain biases that allow for high performance based on superficial cues, thus inflating the perceived success of systems on these downstream tasks (Gururangan et al., 2018; Bentivogli et al., 2016). In the present work, our first priority is careful control of our tasks such that biases are eliminated to the greatest extent possible, allowing more confident conclusions about systems' compositional capacities than are possible with existing metrics.

The contributions of this paper are threefold. 1) We introduce a method for analyzing compositional meaning information in sentence embeddings, along with a generation system that enables controlled creation of datasets for this analysis. 2) We provide experiments with a range of sentence composition models, to demonstrate the capacity of our method to shed light on compositional information captured by these models. 3) We make available the classification datasets used for these experiments, as well as the generation system used to produce the sentence sets, to allow for broader testing of composition models and to facilitate creation of new tasks and classification datasets.

Although we focus on neural composition models and sentence embeddings in the present paper—due to the current dominance of these methods and the need to evaluate their compositional capacities—it is important to note that this analysis method can also be applied more broadly. Since the method simply operates by classification of sentence representations, it can be applied to any format of sentence representation that can be input as features to a classifier.

2 Meaning and composition

In this section we will briefly explain the concepts of *meaning* and *composition*, which are the central targets of our analyses in this work.

Our approach assumes there to be identifiable components of *meaning* that we can expect in well-formed sentence representations. For instance, the sentence "the dog chased the girl" contains the information that there was a chasing event, and a dog was the chaser (*agent* of chasing) and a girl the chasee (*patient* of chasing). The sentence "the dog did not bark" conveys that a barking event did not happen.

Humans are able to extract meaning with remarkable robustness, and a key factor in human language understanding is *composition*: the productive combinatory capacity that allows sentence meaning to be derived systematically based on the meanings of its parts (Heim and Kratzer, 1998). To illustrate the power of this systematicity, consider a nonsensical sentence like the following:

The turquoise giraffe recited the sonnet but did not forgive the flight attendant.

Though this sentence describes an entirely implausible scenario, and though nothing like it should ever have occurred in any corpus or conversation, any English speaker is able to extract the meaning of this sentence without difficulty. This is because language is highly systematic, and the meanings of the parts of the sentence can be combined predictably to arrive at the full meaning.

Regardless of how closely NLP systems should draw on human strategies for language processing, the need for composition is clear: if systems do not construct meanings of sentences based on their parts, then the alternative is memorization of all possible sentences, which is neither practical nor possible.

In this work, critically, we are focused on the results of systematic compositional processes, to be distinguished from biases based on general statistical regularities. The importance of this distinction is

highlighted by the result reported in Adi et al. (2016), which shows a BOW composition model attaining 70% accuracy on a binary word order classification task. This result is surprising given that BOW models (which simply average together word-level representations) necessarily sacrifice any order information from the source sentence. This suggests that the above-chance performance relies on statistical regularities of word ordering in the data as a whole, independent of the source sentence—that is, the model's above-chance performance must be dependent on some correspondence between word orders being tested and word orders seen when training the word embeddings.

Although sensitivity to such regularities is often useful, in this work we are concerned with *systematic composition of the source sentence itself*, abstracting away from general statistical regularities. This is critical for our purposes: to master composition, models must be able to construct the meaning of a sentence not only when it matches commonly-seen patterns (e.g., "the cat chased the mouse") but also when it deviates from such patterns (e.g., "the mouse chased the cat"). This is the reasoning behind our BOW sanity check, discussed in Section 3.3, which serves to ensure that our tests cannot be solved by simple averaging. Additionally, the biases in naturally-occurring data, further highlighted by the Adi et al. result, motivate our use of generated data for the sake of maintaining the necessary level of control.

3 The present method

3.1 Approach

The approach that we take to probe for compositional meaning information in sentence embeddings is inspired by the neuroscience technique of multivariate pattern analysis (Haxby et al., 2014), which tests for encoding of information in patterns of neural data by means of classification tasks designed to be contingent on the information of interest. Our use of careful control in implementing this approach is also informed more generally by the methodologies of cognitive neuroscience and psycholinguistics, which standardly use these kinds of controls to draw conclusions about information in human brain activity. The approach that we develop here builds on the proposal of Ettinger et al. (2016)—which described the basic form of the method and provided a simple validation with a small set of active and passive sentences. In the present work we flesh out and strengthen the method with a number of more rigorous controls aimed at better isolating the information of interest, and we substantially expand the scope of the tests through the use of a more sophisticated sentence generation system.

3.2 Classification tasks

As proposed by Ettinger et al. (2016), we target two meaning components as our starting point: semantic role and negation. These components are priorities because they are fundamental to the meaning of a sentence, having bearing on the key questions of "what happened (and what didn't)" and "who did what to whom". Additionally, they represent information types that can be heavily distorted with respect to surface variables like word content and order: to know semantic role and negation information, it is not enough to know which words are in the sentence or which words come earlier in the sentence.

We formulate the semantic role classification task ("**SemRole**") as follows: "Given representation $\mathbf{n}$ of probe noun n, representation $\mathbf{v}$ of probe verb v, and embedding $\mathbf{s}$ of sentence s (with s containing both n and v), does n stand in the AGENT relation to v in s?" For example, an input of $\{n$: "professor", v: "help", s: "the professor helped the student"$\}$ would receive a positive label because *professor* is AGENT of *help* in the given sentence.

We formulate the negation classification task ("**Negation**") as follows: "Given a representation $\mathbf{v}$ of a probe verb v, and an embedding $\mathbf{s}$ of sentence s (with s containing v, one negation, and one other verb), is v positive or negated in s?" For example, an input of $\{v$: "sleep", s: "the professor is not actually helping the student who is totally sleeping"$\}$ receives a positive label because *sleep* is not negated in that sentence. To decouple this from a simpler task of identifying adjacency between negation and a verb, we insert variable-length adverb sequences (e.g., *not **really, actually** helping*) before the verbs in the dataset (negated and non-negated), to ensure that the negation is not always adjacent to the verb that it affects.

These formulations differ from those in the original Ettinger et al. (2016) proposal, instead making use of variable word probes as employed by Adi et al. (2016). This adjustment was made to maximize

the generalization required for strong performance on the tasks, and to further reduce vulnerability to biasing cues in the datasets. More detail on our implementation of this formulation is given in Section 5.

3.3 Means of control

The most critical consideration in this work is ensuring that we can draw valid conclusions about composition from performance on our classification tasks. To this end, we take a number of measures to control our data, to avoid biasing cues that would make the tasks solvable independent of the information of interest—a problem observed in many existing datasets, as mentioned above (Gururangan et al., 2018).

Generation system A critical component of isolating abstract meaning information is employing syntactic variation, such that the meaning information of interest is the single underlying variable distinguishing label categories. For instance, we might use sentences like "the professor helped the student", "the student was helped by the professor", and "the student that the professor helped was sleeping"—which vary in structure, but which share an underlying event of a professor helping a student.

In order to produce sentence sets that exhibit this level of variation—and that reach the necessary scale for training and testing classifiers—without allowing the statistical biases of naturally-occurring data, we developed a generation system that takes as input lexical, semantic and syntactic constraints, and that produces large sentence sets meeting those constraints. In addition to allowing us to produce controlled datasets, this system also ensures that the generated datasets are annotated with detailed semantic and syntactic information. This generation system is described in greater detail in Section 4.

Train/test splits To be confident that the classifier is picking up on underlying meaning information and not simply a union of different superficial cues across syntactic structures, we make careful provisions in our train/test split to ensure generalization (beyond the obvious split such that sentences in test do not appear in training). For our semantic role task, certain (n,v) probe combinations are held out for test, such that no combinations seen at test time have been seen during training. This is done to ensure that the classifier cannot rely on memorized sequences of words. For our negation task, which uses only one probe, we hold out certain adverbs from training (as described above, adverbs are used as material to separate the negation and the verb), such that at test time, the material separating the negation and the verb (or preceding the non-negated verb) has never been seen in training.

BOW as control As described above, it is logically impossible for BOW models to encode information that requires access to word order from the source sentence itself. We leverage this knowledge to create a sanity check baseline for use in monitoring for lexical biases: if, for any task requiring access to word order information, the BOW baseline performs above chance, we know that the datasets contain lexical biases affecting the classification results, and we can modify them accordingly.

4 Generation system

In this section we describe the generation system that we use to create large, controlled datasets for our classification tasks. As described above, this system takes input constraints targeting semantic, syntactic, and lexical components, and produces diverse, meaning-annotated sentences meeting those constraints.

4.1 Event/sentence representations

As a framework for specifying semantic and syntactic constraints, we use a class of event representations that contain both lexicalized semantic information and necessary syntactic information, such that there is a deterministic mapping from a fully-populated event representation to a corresponding surface sentence form. These representations fall roughly within the category of "lexicalized case frame" outlined by Reiter and Dale (2000) for natural language generation. Figure 1 shows an example representation, in fully-specified textual form, and in simplified graphical form.

Our representations are currently restricted to events denoted by transitive and intransitive verbs, with the arguments of those verbs and optional transitive or intransitive relative clauses on those arguments.

These representations are comparable in many ways to abstract meaning representation (AMR) (Banarescu et al., 2012), but rather than abstracting entirely away from syntactic structure as in AMR, our

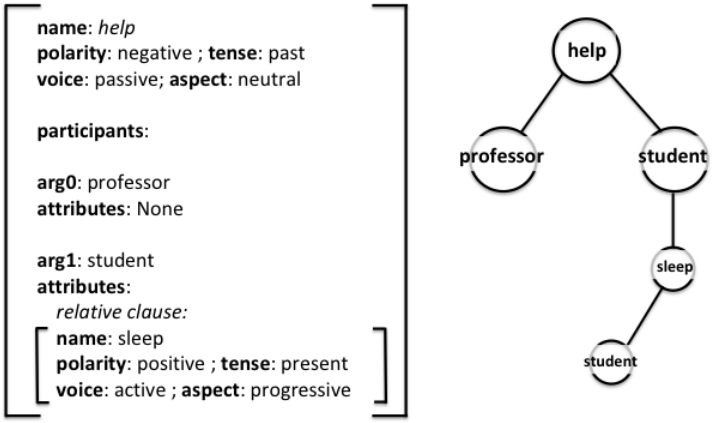


Figure 1: Event representation for *"The student who is sleeping was not helped by the professor"*

event representations encode syntactic information directly, along with the more abstract meaning information, in order to maintain a deterministic mapping to surface forms. Relatedly, while AMR uses PropBank frames (Palmer et al., 2005) to encode meaning information, we encode information via English lemmas, to maintain control over lexical selection during generation.

These representations can be partially specified to reflect a desired constraint, and can then be passed in this partial form as input to the generation system—either as a required component, or as a prohibited component. This allows us to constrain the semantic and syntactic characteristics of the output sentences. In addition to partial events, the system can also take lists of required or prohibited lexical items.

4.2 Event population

The system uses a number of structural templates into which partial events can be inserted. Structural templates vary based on the transitivity of verbs and the presence or absence of relative clauses on arguments—for instance, if the nodes in the right side of Figure 1 were unpopulated, it would depict an empty structural template consisting of a transitive main verb with an intransitive relative clause on arg1. Once we have inserted a partial event into a subsection of an empty structural template (events can be inserted into either the main clause or a relative clause), the system populates the remainder of the event components by iterating through available verbs and nouns of the vocabulary, and through available values for unfilled syntactic characteristics (such as polarity, tense, voice, etc.).

For simplicity, we control plausibility of argument/predicate combinations by setting the system vocabulary such that it contains only animate human nouns, and only verbs that can take any of those nouns in the relevant argument slots. This is a reasonable task due to the capacity of the system to generate thousands of sentences from only a handful of nouns and verbs. We leave incorporation of more sophisticated selectional preference methods (Van de Cruys, 2014; Resnik, 1996) for future work.

Our goal is to find the optimal balance between the critical need of this method for structurally variable, carefully controlled sentences, and the practical need to avoid substantial deviation from sentence types to which systems will have been exposed during training. To this end, we draw our vocabulary from comparatively frequent words, and we impose structural constraints to limit the complexity of sentences—specifically, in the current experiments we restrict to sentences with no more than one relative clause, by omitting templates that include relative clauses on both arguments of a main verb.

4.3 Syntactic realization

Once an event representation is fully populated, it is submitted to a surface realization module that maps from the event to a surface sentence via a simple rule-based mapping. Since the representations specify syntactic information and use lexicalized meaning information, there is no significant process of lexical selection required during surface realization—only morphological inflection derivable from syntactic characteristics. As a result, the event representations map deterministically to their corresponding surface

forms. We use a grammar specified using the NLTK feature grammar framework (Bird et al., 2009). Morphological inflections are drawn from the XTAG morphological database (Doran et al., 1994).

4.4 Sentence quality

To ensure the quality of generation system output, we manually inspected large samples of generated sentences throughout development and after generation of the final sets, to confirm that sentences were grammatical and of the expected form. Table 1 shows a representative sample of generated sentences.[2]

the men were sleeping
the woman followed the lawyer that the student is meeting
the women were being helped by the lawyers
the student called the man
the scientist that the professors met is dancing
the doctors that helped the lawyers are being recommended by the student

Table 1: Example generated sentences

5 Implementation of lexical variability

As discussed above, we adopt the variable probe formulation used by Adi et al. (2016). This adds a dimension to the learning task that is not present in the original Ettinger et al. (2016) task formulation: the classifier needs not only to identify meaning information in the input sentence—it needs to identify meaning information contingent on the identities of the particular probe words.

To identify the probe word(s) in the input features, Adi et al. (2016) use the source word embeddings, but this is problematic for our purposes, given that we want to test a wide variety of models, which use word embeddings of different types and sizes. To avoid this variability, it would be preferable to use one-hot vectors to identify word probes. To this end, we performed a series of experiments testing whether classification accuracy was affected by use of one-hot probe representations by comparison to embedding probes, in a replication of the word content task of Adi et al. (2016). Finding almost equivalent accuracies between the two input types, we use one-hot probe representations in all subsequent experiments.

Note that as a result, by contrast to Adi et al. (2016) we are not assuming the classifier to identify words in the sentence representation based on resemblance to their original word embeddings—this may not in fact be a safe assumption, given that the word's representation may distort during composition of the sentence. Instead, the classifier must learn a mapping from each one-hot representation to its manifestation in sentences. This means that all words must appear as probes in training. To facilitate the learning of this mapping, we restrict to a small (14-lemma) vocabulary of noun and verb probes in these experiments.[3] Because the generation system is able to produce thousands of sentences from even such a restricted vocabulary as this, this limitation does not prevent generation of adequately large datasets.

A note about the size and selection of the vocabulary: some composition tests will surely be sensitive to specific idiosyncrasies of individual words, in which case the choice of vocabulary will be of great importance. However, for the particular semantic role and negation tasks described here, the focus is on identification of structural dependencies between words, which are not in this case sensitive to the specific nouns/verbs used. Consequently, for these tasks—as long as vocabulary words are not out-of-vocabulary for the models (which we confirm below)—the important thing should be not what the words themselves are, but whether dependencies between them have been captured in the sentence embeddings.

6 Surface tasks: word content and order

Though our ultimate interest is in abstract meaning information, part of the goal of these experiments is to get a clear picture of the information currently captured by existing systems. For this reason, we

[2] More sentences can be found in the publicly available classification datasets.

[3] Sentences themselves contain various morphological inflections of these lemmas.

include the content and order experiments as performed by Adi et al. (2016), to see how encoding of these surface variables compares to encoding of meaning information—and to compare with the results of Adi et al. (2016) after the more rigorous controls used in our datasets.

We structure these tasks to be maximally parallel with our meaning tasks. To this end, we have two content tasks: one-probe ("**Content1Probe**") and two-probe ("**Content2Probe**"), with the one-probe task using verb probes as in the negation task, and two-probe using noun-verb probe pairs, as in the semantic role task. Similarly, for the order task ("**Order**") we use only noun-verb pairs. The order task is thus formulated as "Given representation $\mathbf{n}$ of probe noun n, representation $\mathbf{v}$ of probe verb v, and embedding $\mathbf{s}$ of sentence s (with s containing both n and v), does n occur before v in s?". The two-word content task is formulated as "Given representation $\mathbf{n}$ of probe noun n, representation $\mathbf{v}$ of probe verb v, and embedding $\mathbf{s}$ of sentence s, do both n and v occur in s?", and the one-word content task is formulated as "Given representation $\mathbf{v}$ of probe verb v, and embedding $\mathbf{s}$ of sentence s, does v occur in s?"

7 Classification experiments

To demonstrate the utility of our analysis, we use it to test several existing sentence composition models. Following Adi et al. (2016), for our classifier we use a multi-layer perceptron with ReLU activations and a single hidden layer matching the input size. For each of the above tasks we construct train/test sets consisting of 4000 training items and 1000 test items.[4] No tuning is necessary, as the hyperparameters of hidden layer number and size are fixed in accordance with the architecture used by Adi et al. (2016).

It is important to note that the training of the classifier, which uses the 4000 items mentioned above, is to be distinguished from the training of the sentence embedding methods. The sentence embedding models are pre-trained on separate corpora, as described below, such that they map sentence inputs to embeddings. Once these models are trained, they are used to produce the 4000 sentence embeddings that will serve as training input to the classifier (and the 1000 sentence embeddings used for testing).

Our use of a relatively simple classifier with a single hidden layer builds on the precedent not only of Adi et al. (2016), but also of related methods in neuroscience, which in fact typically use linear classifiers (an option that we could not employ due to our use of the variable probes). An important reason for use of simpler classifiers is to test for *straightforward* extractability of information from embeddings—if a complex classifier is necessary in order to extract the information of interest, then this calls into question the extent to which we might consider this information to be "captured" in the embeddings, as opposed to the information being somehow reconstructable from the embeddings' encoding of other information. That said, the question of how the complexity of the classifier relates to the encoding of the target information in these sentence embeddings is an interesting issue for future work.

For each experiment, we also run two corresponding experiments, in which random vectors are used in place of the sentence vectors and the probes, respectively. This serves as an additional check for biases in the datasets, to ensure that neither the sentence vectors nor the probe vectors alone are sufficient to perform above chance on the tasks. For all tasks, these random vectors produce chance performance.

7.1 Sentence encoding models

We test a number of composition models on these classification tasks. These models represent a range of influential current models designed to produce task-general sentence embeddings. They employ a number of different architectures and objectives, and have shown reasonable success on existing metrics (Hill et al., 2016; Conneau et al., 2017).

All sentence embeddings used are of 2400 dimensions. Because our pre-trained models (SDAE, Skip-Thought) are trained on the Toronto Books Corpus (Zhu et al., 2015), we use this as our default training corpus, except when other supervised training data is required (as in the case of InferSent). Before sentence generation, the chosen vocabulary was checked against the training corpora to ensure that no words were out-of-vocabulary (or below a count of 50).

[4] See footnote 1 for link to all classification datasets used in these experiments.

	Accuracy				
	Content1Probe	Content2Probe	Order	SemRole	Negation
BOW	100.0	97.1	55.0	51.3	50.9
SDAE	100.0	79.8	92.9	63.7	99.0
ST-UNI	100.0	88.1	93.2	62.3	96.6
ST-BI	96.6	79.4	88.7	63.2	74.7
InferSent	100.0	70.1	86.4	50.1	97.2

Table 2: Classification results

BOW averaging Our first sentence embedding model (**"BOW"**) is a BOW averaging model, for which we use the skip-gram architecture of the word2vec model (Mikolov et al., 2013) to learn word embeddings. As discussed above, the BOW model serves primarily as a sanity check for our purposes, but it is important to note that this model has had competitive results on various tasks, and is taken seriously as a sentence representation method for many purposes (Wieting et al., 2016; Arora et al., 2016).

Sequential Denoising Autoencoder Our second model (**"SDAE"**) is an autoencoder variant from Hill et al. (2016) for unsupervised learning of sentence embeddings. The model uses an LSTM-based encoder-decoder framework, and is trained to reconstruct input sentences from their vector representations (last hidden state of encoding LSTM) despite noise applied to the input sentence. We use a pre-trained model provided by the authors. This model has the advantage of an unsupervised objective and no need for sequential sentence data, and it shows competitive performance on a number of evaluations.

Skip-Thought Embeddings Our next two models are variants of the Skip-Thought model (Kiros et al., 2015), in which sentences are encoded with gated recurrent units (GRUs), with an objective of using the current sentence representation to predict the immediately preceding and following sentences. Following the model's authors, we use both the uni-skip (**"ST-UNI"**) and bi-skip (**"ST-BI"**) variants: uni-skip consists of an encoding based on a forward pass of the sentence, while bi-skip consists of a concatenation of encodings of the forward and backward passes of the sentence (each of 1200 dimensions, for 2400 total). We use the publicly available pre-trained Skip-Thought model for both of these variants.[5]

Skip-Thought sentence embeddings have been used as pre-trained embeddings for a variety of tasks. They have proven to be generally effective for supervised tasks and passable for unsupervised tasks (Hill et al., 2016; Triantafillou et al., 2016; Wieting et al., 2016). Like the SDAE model, the Skip-Thought model is able to use unsupervised learning, though it requires sequential sentence data. However, more than the SDAE model, the Skip-Thought model uses an objective intended to capture semantic and syntactic properties, under the authors' assumption that prediction of adjacent sentences will encourage more syntactically and semantically similar sentences to map to similar embeddings.

InferSent Our final model is the InferSent model (Conneau et al., 2017), which uses multi-layer BiLSTM encoders with max pooling on the hidden states of the last layer to produce vector representations of the sentences. This model is trained with a natural language inference (NLI) objective, and for this reason we train it on the SNLI dataset (Bowman et al., 2015).

The InferSent model is intended to produce "universal" sentence representations, and has been shown to outperform unsupervised methods like Skip-Thought on a number of tasks (Conneau et al., 2017). More generally, the NLI objective is believed to encourage learning of compositional meaning information, given that inference of entailment relations should require access to meaning information.

7.2 Results and Discussion

Table 2 shows the accuracy of the different models' sentence embeddings on our classification tasks.

The first thing to note is that our BOW control allows us to confirm nearly complete lexical balance in

[5] https://github.com/ryankiros/skip-thoughts

the sentence sets: the averaged word embeddings perform roughly at chance on all but the content tasks.[6] By contrast, BOW performs with near-perfect accuracy on the content tasks, lending support to the intuitive conclusion: the one thing that BOW *does* encode is word content. The quality of performance of the BOW model on this task exceeds that reported by Adi et al. (2016)—we speculate that this may be due to our use of a smaller vocabulary to facilitate the learning of the mapping from one-hot probes.

While BOW has very high performance on two-probe word content, SDAE, ST-UNI, ST-BI and InferSent have much lower accuracy (albeit still far above chance), suggesting that some detail with respect to word content is sacrificed from these representations in favor of other information types. This is exemplified by the order task, on which all non-BOW models show significantly higher accuracy than on the word content tasks, supporting the intuitive conclusion that such sequence-based models retain information about relative word position. This result is generally consistent with the Adi et al. (2016) result, but due to the additional control that brings BOW roughly to chance, we can conclude with greater confidence that the performance on this task pertains to order information in the source sentence itself.

Turning to our meaning information tasks, we see that with the exception of ST-BI, the sequence models perform surprisingly well on the negation task, despite the fact that this task cannot be solved simply by detecting adjacency between negation and the verb (due to our insertion of adverbs). Instead, we speculate that these sequence models may be picking up on the utility of establishing a dependency between negation and the *next* verb, even in the face of intervening words. This is not a complete solution to the problem of representing the meaning and dependencies of negation, but it is a useful step in that direction, and suggests that models may be sensitive to some of the behaviors of negation.

Interestingly, ST-BI shows markedly weaker performance on the negation task. We see two potential reasons for this. First, it may be due to the reduced dimensionality of each of the two concatenated encodings (recall that ST-BI involves concatenating 1200-dimensional encodings of the forward and backward passes). Second, the reduced performance could be influenced by the inclusion of the backward pass: while the forward pass can leverage the strategy of linking negation to the next verb, the backward pass cannot use this strategy because it will encounter the relevant verb before encountering the negation.

Turning to the semantic role task, we see a stark contrast with the high performance for the negation task. InferSent performs squarely at chance, suggesting that it retains as little compositional semantic role information as does BOW. SDAE, ST-UNI and ST-BI perform modestly above chance on the semantic role task at 62-63% accuracy, suggesting that they may provide some amount of abstract role information—but no model shows any substantial ability to capture semantic role systematically.

These results accomplish two things. First, they lend credence to this method as a means of gaining insight into the information captured by current models. Second, they give us a sense of the current capacity of sequence-based models to capture compositional meaning information. The picture that emerges is that sequence models are able to make non-trivial headway in handling negation, presumably based on a sequential strategy of linking negation to the next verb—but that these sequence models fall significantly short when it comes to capturing semantic role compositionally. Another point that emerges from these results is that despite the fairly substantial differences in architecture, objective, and training of these models, capacity to capture the compositional information is fairly similar across models, suggesting that these distinct design decisions are not having a very significant impact on compositional meaning extraction. We plan to test more substantially distinct models, like those with explicit incorporation of syntactic structure (Bowman et al., 2016; Dyer et al., 2016; Socher et al., 2013) in future work.

8 Related work

This work relates closely to a growing effort to increase interpretability of neural network models in NLP—including use of visualization to analyze what neural networks learn (Li et al., 2015; Kádár et al., 2016), efforts to increase interpretability by generating explanations of model predictions (Ribeiro et al., 2016; Lei et al., 2016; Li et al., 2016), and work submitting adversarial examples to systems in order to identify weaknesses (Zhao et al., 2017; Jia and Liang, 2017; Ettinger et al., 2017).

[6] The slightly higher accuracy on the order task is most likely the result of a very slight bias due to our use of only noun-verb order probe pairs for the sake of matching the SemRole task.

Methodologically the most closely related work is that of Adi et al. (2016), which uses classification tasks to probe for information in sentence embeddings. As discussed above, we depart from that work in targeting deeper and more linguistically-motivated aspects of sentence meaning, and we incorporate careful controls of our datasets to ensure elimination of bias in the results.

Our focus on assessing linguistically-motivated information relates to work on evaluations that aim for fine-grained analysis of systems' linguistic capacities (Rimell et al., 2009; Bender et al., 2011; Marelli et al., 2014). The present work contributes to this effort with new tasks that assess composition *per se*, and that do so in a highly targeted manner via careful controls. Our use of synthetically generated data to achieve this level of control relates to work like that of Weston et al. (2015), which introduces synthetic question-answering tasks for evaluating the capacity of systems to reason with natural language input.

Our examination of the capacity of neural sequence models to identify abstract relations in sentence representations also relates to work by Linzen et al. (2016), who explore whether LSTMs can learn syntactic dependencies, as well as Williams et al. (2017), who investigate the extent to which parsers that are learned based on a semantic objective produce conventional syntax.

Finally, importantly related work is that concerned specifically with testing systematic composition. Lake and Baroni (2017) investigate the capacity of RNNs to perform zero-shot generalization using composition, and Dasgupta et al. (2018) construct an entailment dataset with balanced lexical content in order to target composition more effectively. We contribute to this line of inquiry by establishing an analysis method that can take output embeddings from sentence composition models and query them directly for specific types of information to be expected in properly compositional sentence representations.

9 Conclusions and future directions

We have presented an analysis method and accompanying generation system designed to address the problem of assessing compositional meaning content in sentence vector representations. We make the datasets for these tasks, as well as the generation system used to create them, available for public use to facilitate broader testing of composition models. We have also presented the results of applying this method for analysis of a number of current sentence composition models, demonstrating the capacity of the method to derive meaningful information about what is captured in these models' outputs.

Having established a means of analyzing compositional meaning information in sentence embeddings, in future work we plan to apply this system to identify more precisely which design decisions lead to effective capturing of meaning information, in order to guide system improvement. As part of this effort, we will expand to more comprehensive testing of a diverse range of sentence embedding systems (Bowman et al., 2016; Subramanian et al., 2018). We also plan to investigate the potential of our generation system to create not just evaluation data, but training data—given that it allows us to produce large, meaning-annotated corpora. Finally, we plan to expand beyond semantic role and negation in the set of information types targeted by our method, in order to establish more comprehensive coverage of meaning information that can be assessed by this analysis system.

Acknowledgements

Devin Ganey contributed code interfacing the generation system and the XTAG database. The work described in this paper benefited from discussions with Alexander Williams, Marine Carpuat, and Hal Daumé III, and from helpful comments by Ido Dagan, Jason Eisner, Chris Dyer, audience members at RepEval 2016, members of the UMD CLIP and CNL labs, as well as anonymous reviewers. This work was supported in part by an NSF Graduate Research Fellowship to Allyson Ettinger under Grant No. DGE 1322106, and by NSF NRT Grant DGE-1449815. Any opinions, findings, and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the NSF.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2012. Abstract meaning representation (AMR) 1.0 specification. In *Parsing on Freebase from Question-Answer Pairs. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle: ACL*, pages 1533–1544.

Emily M. Bender, Dan Flickinger, Stephan Oepen, and Yi Zhang. 2011. Parser evaluation over local and non-local deep dependencies in a large corpus. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 397–408, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Luisa Bentivogli, Raffaella Bernardi, Marco Marelli, Stefano Menini, Marco Baroni, and Roberto Zamparelli. 2016. SICK through the SemEval glasses. Lesson learned from the evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *Language Resources and Evaluation*, 50(1):95–124.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. *arXiv preprint arXiv:1603.06021*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Ishita Dasgupta, Demi Guo, Andreas Stuhlmüller, Samuel J Gershman, and Noah D Goodman. 2018. Evaluating compositionality in sentence embeddings. *arXiv preprint arXiv:1802.04302*.

Christy Doran, Dania Egedi, Beth Ann Hockey, Bangalore Srinivas, and Martin Zaidel. 1994. XTAG system: a wide coverage grammar for English. In *Proceedings of the 15th conference on Computational linguistics-Volume 2*, pages 922–928. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. *NAACL*.

Allyson Ettinger, Ahmed Elgohary, and Philip Resnik. 2016. Probing for semantic evidence of composition by means of simple classification tasks. *Proceedings of the 1st Workshop on Evaluating Vector Space Representations for NLP*, page 134.

Allyson Ettinger, Sudha Rao, Hal Daumé III, and Emily M Bender. 2017. Towards linguistically generalizable NLP systems: A workshop and shared task. *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. *NAACL*.

James V Haxby, Andrew C Connolly, and J Swaroop Guntupalli. 2014. Decoding neural representational spaces using multivariate pattern analysis. *Annual review of neuroscience*, 37:435–456.

Irene Heim and Angelika Kratzer. 1998. *Semantics in generative grammar*, volume 13. Blackwell Oxford.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *NAACL*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*.

Ákos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2016. Representation of linguistic form and function in recurrent neural networks. *arXiv preprint arXiv:1602.08952*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3276–3284.

Brenden M Lake and Marco Baroni. 2017. Still not systematic after all these years: On the compositional skills of sequence-to-sequence recurrent networks. *arXiv preprint arXiv:1711.00350*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. *arXiv preprint arXiv:1606.04155*.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in NLP. *arXiv preprint arXiv:1506.01066*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Language Resources and Evaluation*, pages 216–223.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

M Palmer, D Gildea, and P Kingsbury. 2005. The Proposition Bank: A corpus annotated with semantic roles. *Computational Linguistics*, 31(1):712105.

Ehud Reiter, Robert Dale, and Zhiwei Feng. 2000. *Building natural language generation systems*, volume 33. MIT Press.

Philip Resnik. 1996. Selectional constraints: An information-theoretic model and its computational realization. *Cognition*, 61(1-2):127–159.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should I trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 813–821, Singapore. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. *ICLR*.

Eleni Triantafillou, Jamie Ryan Kiros, Raquel Urtasun, and Richard Zemel. 2016. Towards generalizable sentence embeddings. In *ACL Workshop on Representation Learning for NLP*, page 239.

Tim Van de Cruys. 2014. A neural network approach to selectional preference acquisition. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 26–35.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In *International Conference on Learning Representations (ICLR)*.

Adina Williams, Andrew Drozdov, and Samuel R Bowman. 2017. Learning to parse from a semantic objective: It works. is it syntax? *arXiv preprint arXiv:1709.01121*.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2017. Generating natural adversarial examples. *arXiv preprint arXiv:1710.11342*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Subword-augmented Embedding for Cloze Reading Comprehension

Zhuosheng Zhang[1,2,*], **Yafang Huang**[1,2,*], **Hai Zhao**[1,2,†]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
{zhangzs, huangyafang}@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn

Abstract

Representation learning is the foundation of machine reading comprehension. In state-of-the-art models, deep learning methods broadly use word and character level representations. However, character is not naturally the minimal linguistic unit. In addition, with a simple concatenation of character and word embedding, previous models actually give suboptimal solution. In this paper, we propose to use subword rather than character for word embedding enhancement. We also empirically explore different augmentation strategies on *subword-augmented embedding* to enhance the cloze-style reading comprehension model (reader). In detail, we present a reader that uses subword-level representation to augment word embedding with a short list to handle rare words effectively. A thorough examination is conducted to evaluate the comprehensive performance and generalization ability of the proposed reader. Experimental results show that the proposed approach helps the reader significantly outperform the state-of-the-art baselines on various public datasets.

1 Introduction

A recent hot challenge is to train machines to read and comprehend human languages. Towards this end, various machine reading comprehension datasets have been released, including cloze-style (Hermann et al., 2015; Hill et al., 2015; Cui et al., 2016) and user-query types (Joshi et al., 2017; Rajpurkar et al., 2016). Meanwhile, a number of deep learning models are designed to take up the challenges, most of which focus on attention mechanism (Wang et al., 2017b; Seo et al., 2017; Cui et al., 2017a; Kadlec et al., 2016; Dhingra et al., 2017; Zhang and Zhao, 2018). However, how to represent word in an effective way remains an open problem for diverse natural language processing tasks, including machine reading comprehension for different languages. Particularly, for a language like Chinese with a large set of characters (typically, thousands of), lots of which are semantically ambiguous, using either word-level or character-level embedding alone to build the word representations would not be accurate enough. This work especially focuses on a cloze-style reading comprehension task over fairy stories, which is highly challenging due to diverse semantic patterns with personified expressions and reference.

In real practice, a reading comprehension model or system which is often called *reader* in literatures easily suffers from out-of-vocabulary (OOV) word issues, especially for the cloze-style reading comprehension tasks when the ground-truth answers tend to include rare words or named entities (NE), which are hardly fully recorded in the vocabulary. This is more challenging in Chinese. There are over 13,000 characters in Chinese[1] while there are only 26 letters in English without regard to punctuation marks. If a reading comprehension system cannot effectively manage the OOV issues, the performance will not be semantically accurate for the task.

*These authors contribute equally. † Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100), National Natural Science Foundation of China (No. 61672343 and No. 61733011), Key Project of National Society Science Foundation of China (No. 15-ZDA041), The Art and Science Interdisciplinary Funds of Shanghai Jiao Tong University (No. 14JCRZ04).

[1]Refer to the statistics of Xinhua Dictionary, version 11, published by The Commercial Press in 2014.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1802–1814
Santa Fe, New Mexico, USA, August 20-26, 2018.

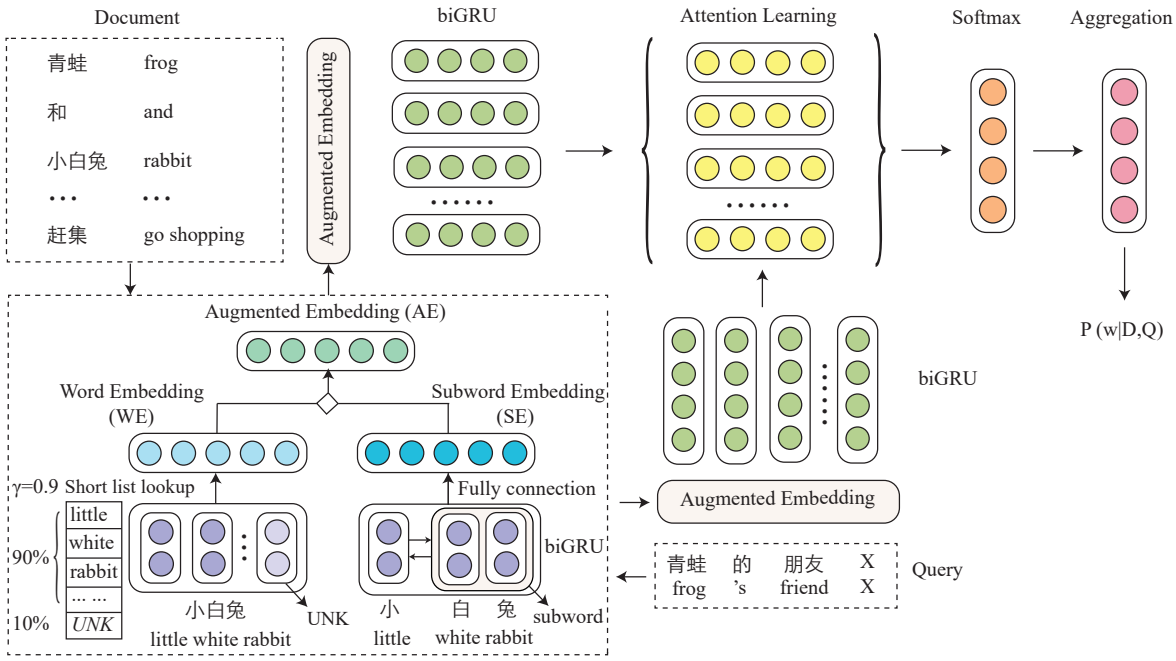


Figure 1: Architecture of the proposed Subword-augmented Embedding Reader (SAW Reader).

Commonly, words are represented as vectors using either word embedding or character embedding. For the former, each word is mapped into low dimensional dense vectors from a lookup table. Character representations are usually obtained by applying neural networks on the character sequence of the word, and their hidden states are obtained to form the representation. Intuitively, word-level representation is good at catching global context and dependency relationships between words, while character embedding helps for dealing with rare word representation.

However, the minimal meaningful unit below word usually is not character, which motivates researchers to explore the potential unit (subword) between character and word to model sub-word morphologies or lexical semantics. In fact, morphological compounding (e.g. *sunshine* or *playground*) is one of the most common and productive methods of word formation across human languages, which inspires us to represent word by meaningful sub-word units. Recently, researchers have started to work on morphologically informed word embeddings (Botha and Blunsom, 2014; Cao and Rei, 2016), aiming at better capturing syntactic, lexical and morphological information. With ready subwords, we do not have to work with characters, and segmentation could be stopped at the subword-level to reach a meaningful representation.

In this paper, we present various simple yet accurate subword-augmented embedding (SAW) strategies and propose SAW Reader as an instance. Specifically, we adopt subword information to enrich word embedding and survey different SAW operations to integrate word-level and subword-level embedding for a fine-grained representation. To ensure adequate training of OOV and low-frequency words, we employ a short list mechanism. Our evaluation will be performed on three public Chinese reading comprehension datasets and one English benchmark dataset for showing our method is also effective in multi-lingual case.

2 The Subword-augmented Word Embedding

The concerned reading comprehension task can be roughly categorized as user-query type and cloze-style according to the answer form. Answers in the former are usually a span of texts while in the cloze-style task, the answers are words or phrases which lets the latter be the harder-hit area of OOV issues, inspiring us to select the cloze-style as our testbed for SAW strategies. Our preliminary study shows even for the advanced word-character based GA reader, OOV answers still account for nearly 1/5 in the error results.

This also motivates us to explore better representations to further performance improvement.

The cloze-style task in this work can be described as a triple $< D, Q, A >$, where D is a document (context), Q is a query over the contents of D, in which a word or phrase is the right answer A. This section will introduce the proposed SAW Reader in the context of cloze-style reading comprehension. Given the triple $< D, Q, A >$, the SAW Reader will be built in the following steps.

2.1 BPE Subword Segmentation

Word in most languages usually can be split into meaningful subword units despite of the writing form. For example, "*indispensable*" could be split into the following subwords: $< in, disp, ens, able >$.

In our implementation, we adopt Byte Pair Encoding (BPE) (Gage and Philip, 1994) which is a simple data compression technique that iteratively replaces the most frequent pair of bytes in a sequence by a single, unused byte. BPE allows for the representation of an open vocabulary through a fixed-size vocabulary of variable-length character sequences, making it a very suitable word segmentation strategy for neural network models.

The generalized framework can be described as follows. Firstly, all the input sequences (strings) are tokenized into a sequence of single-character subwords, then we repeat,

1. Count all bigrams under the current segmentation status of all sequences.
2. Find the bigram with the highest frequency and merge them in all the sequences. Note the segmentation status is updating now.
3. If the merging times do not reach the specified number, go back to 1, otherwise the algorithm ends.

In (Sennrich et al., 2016), BPE is adopted to segment infrequent words into sub-word units for machine translation. However, there is a key difference between the motivations for subword segmentation. We aim to refine the word representations by using subwords, for both frequent and infrequent words, which is more generally motivated. To this end, we adaptively tokenize words in multi-granularity by controlling the merging times.

2.2 Subword-augmented Word Embedding

Our subwords are also formed as character n-grams, do not cross word boundaries. After using unsupervised segmentation methods to split each word into a subword sequence, an augmented embedding (AE) is to straightforwardly integrate word embedding $WE(w)$ and subword embedding $SE(w)$ for a given word w.

$$AE(w) = WE(w) \diamond SE(w)$$

where $\diamond$ denotes the detailed integration operation. In this work, we investigate concatenation (*concat*), element-wise summation (*sum*) and element-wise multiplication (*mul*). Thus, each document D and query Q is represented as $\mathbb{R}^{d \times k}$ matrix where d denotes the dimension of word embedding and k is the number of words in the input.

Subword embedding could be useful to refine the word embedding in a finer-grained way, we also consider improving word representation from itself. For quite a lot of words, especially those rare ones, their word embedding is extremely hard to learn due to the data sparse issue. Actually, if all the words in the dataset are used to build the vocabulary, the OOV words from the test set will not obtain adequate training. If they are initiated inappropriately, either with relatively high or low weights, they will harm the answer prediction. To alleviate the OOV issues, we keep a short list H for specific words.

$$H = \{w_1, w_2, \ldots, w_n\}$$

If w is in H, the immediate word embedding $WE(w)$ is indexed from word lookup table $M^w \in \mathbb{R}^{d \times s}$ where s denotes the size (recorded words) of lookup table. Otherwise, it will be represented as the randomly initialized default word (denoted by a specific mark UNK). Note that, this is intuitively

	CMRC-2017			PD			CFT
	Train	Valid	Test	Train	Valid	Test	human
# Query	354,295	2,000	3,000	870,710	3,000	3,000	1,953
Max # words in docs	486	481	484	618	536	634	414
Max # words in query	184	72	106	502	153	265	92
Avg # words in docs	324	321	307	379	425	410	153
Avg # words in query	27	19	23	38	38	41	20
# Vocabulary	94,352	21,821	38,704	248,160	536	634	414

Table 1: Data statistics of CMRC-2017, PD and CFT.

like "guessing" the possible unknown words (which will appear during test) from the vocabulary during training and only the word embedding of the OOV words will be replaced by UNK while their subword embedding $SE(w)$ will still be processed using the original word. In this way, the OOV words could be tuned sufficiently with expressive meaning after training. During test, the word embedding of unknown words would not severely bias its final representation. Thus, $AE(w)$ can be rewritten as

$$AE(w) = \begin{cases} WE(w) \diamond SE(w) & \text{if } w \in H \\ UNK \diamond SE(w) & \text{otherwise} \end{cases}$$

In our experiments, the short list is determined according to the word frequency. Concretely, we sort the vocabulary according to the word frequency from high to low. A frequency filter ratio γ is set to filter out the low-frequency words (rare words) from the lookup table. For example, γ=0.9 means the least frequent 10% words are replaced with the default UNK notation.

The subword embedding $SE(w)$ is generated by taking the final outputs of a bidirectional gated recurrent unit (GRU) (Cho et al., 2014) applied to the embeddings from a lookup table of subwords. The structure of GRU used in this paper are described as follows.

$$\begin{aligned} r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r), \\ z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z), \\ \tilde{h}_t &= \tanh(W_h x_t + U_h(rt \odot h_{t-1}) + b_h) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \end{aligned}$$

where $\odot$ denotes the element-wise multiplication. r_t and z_t are the reset and update gates respectively, and $\tilde{h}_t$ are the hidden states. A bi-directional GRU (BiGRU) processes the sequence in both forward and backward directions. Subwords of each word are successively fed to forward GRU and backward GRU to obtain the internal features of two directions. The output for each input is the concatenation of the two vectors from both directions: $\overleftrightarrow{h_t} = \overrightarrow{h_t} \parallel \overleftarrow{h_t}$. Then, the output of BiGRUs is passed to a fully connected layer to obtain the final subword embedding $SE(w)$.

$$SE(w) = W\overleftrightarrow{h_t} + b$$

2.3 Attention Module

Our attention module is based on the Gated attention Reader (GA Reader) proposed by (Dhingra et al., 2017). We choose this model due to its simplicity with comparable performance so that we can focus on the effectiveness of SAW strategies. This module can be described in the following two steps. After augmented embedding, we use two BiGRUs to get contextual representations of the document and query respectively, where the representation of each word is formed by concatenating the forward and backward hidden states.

$$\begin{aligned} H_q &= \text{BiGRU}(Q) \\ H_d &= \text{BiGRU}(D) \end{aligned}$$

For each word d_i in H_d, we form a word-specific representation of the query $q_i \in H_q$ using soft attention, and then adopt element-wise product to multiply the query representation with the document word representation.

$$\alpha_i = softmax(H_q^\top d_i)$$
$$\beta_i = Q\alpha_i$$
$$x_i = d_i \odot \beta_i$$

where $\odot$ denotes the multiplication operator to model the interactions between d_i and q_i. Then, the document contextual representation $\tilde{H}_d = \{x_1, x_2, \ldots, x_k\}$ is gated by query representation.

Suppose the network has K layers. At each layer, the document representation $\tilde{H}_d$ is updated through above attention learning. After going through all the layers, our model comes to answer prediction phase. We use all the words in the document to form the candidate set C. Let q_t denote the t-th intermediate output of query representation H_q and H_D represent the full output of document representation $\tilde{H}_d$. The probability of each candidate word $w \in C$ as being the answer is predicted using a softmax layer over the inner-product between q_t and H_D.

$$p = softmax((q_t)^\top H_D)$$

where vector p denotes the probability distribution over all the words in the document. Note that each word may occur several times in the document. Thus, the probabilities of each candidate word occurring in different positions of the document are summed up for final prediction.

$$P(w|D,Q) \propto \sum_{i \in I(w,D)} p_i$$

where $I(w,d)$ denotes the set of positions that a particular word w occurs in the document D. The training objective is to maximize $\log P(A|D,Q)$ where A is the correct answer.

Finally, the candidate word with the highest probability will be chosen as the predicted answer.

$$A^* = \text{argmax}_{w \in C} P(w|D,Q)$$

Different from recent work employing complex attention mechanisms (Wang et al., 2017b; Cui et al., 2017a; Sordoni et al., 2016), our attention mechanism is much more simple with comparable performance so that we can focus on the effectiveness of SAW strategies.

3 Experiments

3.1 Dataset and Settings

To verify the effectiveness of our proposed model, we conduct multiple experiments on three Chinese Machine Reading Comprehension datasets, namely CMRC-2017 (Cui et al., 2017b), People's Daily (PD) and Children Fairy Tales (CFT) (Cui et al., 2016)[2]. In these datasets, a story containing consecutive sentences is formed as the *Document* and one of the sentences is either automatically or manually selected as the *Query* where one token is replaced by a placeholder to indicate the *answer* to fill in. Table 1 gives data statistics. Different from the current cloze-style datasets for English reading comprehension, such as CBT, Daily Mail and CNN (Hermann et al., 2015), the three Chinese datasets do not provide candidate answers. Thus, the model has to find the correct answer from the entire document.

Besides, we also use the Children's Book Test (CBT) dataset (Hill et al., 2015) to test the generalization ability in multi-lingual case. We only focus on subsets where the answer is either a common noun (CN)

[2]Note that the test set of CMRC-2017 and human evaluation test set (Test-human) of CFT are harder for the machine to answer because the questions are further processed manually and may not be accordance with the pattern of automatic questions.

Model	CMRC-2017	
	Valid	Test
Random Guess †	1.65	1.67
Top Frequency †	14.85	14.07
AS Reader †	69.75	71.23
GA Reader	72.90	74.10
SJTU BCMI-NLP †	76.15	77.73
6ESTATES PTE LTD †	75.85	74.73
Xinktech †	77.15	77.53
Ludong University †	74.75	75.07
ECNU †	77.95	77.40
WHU †	78.20	76.53
SAW Reader	**78.95**	**78.80**

Table 2: Accuracy on CMRC-2017 dataset. Results marked with † are from the latest official CMRC-2017 Leaderboard [7]. The best results are in bold face.

or NE which is more challenging since the answer is likely to be rare words. We evaluate all the models in terms of accuracy, which is the standard evaluation metric for this task.

Throughout this paper, we use the same model setting to make fair comparisons. According to our preliminary experiments, we report the results based on the following settings. The default integration strategy is *element-wise product*. Word embeddings were 200d and pre-trained by word2vec (Mikolov et al., 2013) toolkit on *Wikipedia* corpus[3]. Subword embedding were 100d and randomly initialized with the uniformed distribution in the interval [-0:05; 0:05]. Our model was implemented using the Theano[4] and Lasagne Python libraries[5]. We used stochastic gradient descent with ADAM updates for optimization (Kingma and Ba, 2014). The batch size was 64 and the initial learning rate was 0.001 which was halved every epoch after the second epoch. We also used gradient clipping with a threshold of 10 to stabilize GRU training (Pascanu et al., 2013). We use three attention layers for all experiments. The GRU hidden units for both the word and subword representation were 128. The default frequency filter proportion was 0.9 and the default merging times of BPE was 1,000. We also apply dropout between layers with a dropout rate of 0.5 [6].

3.2 Main Results

CMRC-2017 Table 2 shows our results on CMRC-2017 dataset, which shows that our SAW Reader (mul) outperforms all other single models on the test set, with 7.57% improvements compared with Attention Sum Reader (AS Reader) baseline. Although WHU's model achieves the best besides our model on the valid set with only 0.75% below ours, their result on the test set is lower than ours by 2.27%, indicating our model has a satisfactory generalization ability.

We also list different integration operations for word and subword embeddings. Table 3 shows the comparisons. From the results, we can see that Word + BPE outperforms Word + Char which indicates subword embedding works essentially. We also observe that *mul* outperforms the other two operations, *concat* and *sum*. This reveals that *mul* might be more informative than *concat* and *sum* operations. The superiority might be due to element-wise product being capable of modeling the interactions and eliminating distribution differences between word and subword embedding. Intuitively, this is also similar to endow subword-aware "attention" over the word embedding. In contrast, concatenation operation may cause too high dimension, which leads to serious over-fitting issues, and sum operation is too simple to prevent from detailed information losing.

[3] https://dumps.wikimedia.org/
[4] https://github.com/Theano/Theano
[5] https://github.com/Lasagne/Lasagne
[6] Our code is available at: https://github.com/cooelf/subMrc
[7] http://www.hfl-tek.com/cmrc2017/leaderboard.html

Model	Operation	CMRC-2017	
		Valid	Test
Word + Char	concat	74.80	75.13
	sum	75.40	75.53
	mul	77.80	77.93
Word + BPE	concat	75.95	76.43
	sum	76.20	75.83
	mul	**78.95**	**78.80**

Table 3: Case study on CMRC-2017.

Model	PD		CFT
	Valid	Test	Test-human
AS Reader	64.1	67.2	33.1
GA Reader	67.2	69.0	36.9
CAS Reader	65.2	68.1	35.0
SAW Reader	**72.8**	**75.1**	**43.8**

Table 4: Accuracy on PD and CFT datasets. Results of AS Reader and CAS Reader are from (Cui et al., 2016).

PD & CFT Since there is no training set for CFT dataset, our model is trained on PD training set. Note that the CFT dataset is harder for the machine to answer because the test set is further processed by human evaluation, and may not be accordance with the pattern of PD dataset. The results on PD and CFT datasets are listed in Table 4. As we see that, our SAW Reader significantly outperforms the CAS Reader in all types of testing, with improvements of 7.0% on PD and 8.8% on CFT test sets, respectively. Although the domain and topic of PD and CFT datasets are quite different, the results indicate that our model also works effectively for out-of-domain learning.

CBT To verify if our method can only work for Chinese, we also evaluate the effectiveness of the proposed method on benchmark English dataset. We use CBT dataset as our testbed to evaluate the performance. For a fair comparison, we simply set the same parameters as before. Table 5 shows the results. We observe that our model outperforms most of the previously public works, with 2.4 % gains on the CBT-NE test set compared with GA Reader which adopts word and character embedding concatenation. Our SAW Reader also achieves comparable performance with FG Reader who adopts neural gates to combine word-level and character-level representations with assistance of extra features including NE, POS and word frequency while our model is much simpler and faster. This result shows our SAW Reader is not restricted to Chinese reading comprehension, but also for other languages.

4 Analysis

4.1 Merging Times of BPE

The vocabulary size could seriously involve the segmentation granularity. For BPE segmentation, the resulted subword vocabulary size is equal to the merging times plus the number of single-character types. To have an insight of the influence, we adopt merge times from 0 to $20k$, and conduct quantitative study on CMRC-2017 for BPE segmentation. Figure 2 shows the results. We observe that when the vocabulary size is $1k$, the models could obtain the best performance. The results indicate that for a task like reading comprehension the subwords, being a highly flexible grained representation between character and word, tends to be more like characters instead of words. However, when the subwords completely fall into characters, the model performs the worst. This indicates that the balance between word and character is quite critical and an appropriate grain of character-word segmentation could essentially improve the word representation.

Model	CBT-NE		CBT-CN	
	Valid	Test	Valid	Test
Human ‡	-	81.6	-	81.6
LSTMs ‡	51.2	41.8	62.6	56.0
MemNets ‡	70.4	66.6	64.2	63.0
AS Reader ‡	73.8	68.6	68.8	63.4
Iterative Attentive Reader ‡	75.2	68.2	72.1	69.2
EpiReader ‡	75.3	69.7	71.5	67.4
AoA Reader ‡	77.8	72.0	72.2	69.4
NSE ‡	78.2	73.2	74.3	71.9
FG Reader ‡	**79.1**	**75.0**	**75.3**	**72.0**
GA Reader ‡	76.8	72.5	73.1	69.6
SAW Reader	78.5	74.9	75.0	71.6

Table 5: Accuracy on CBT dataset. Results marked with ‡ are of previously published works (Dhingra et al., 2017; Cui et al., 2016; Yang et al., 2017).

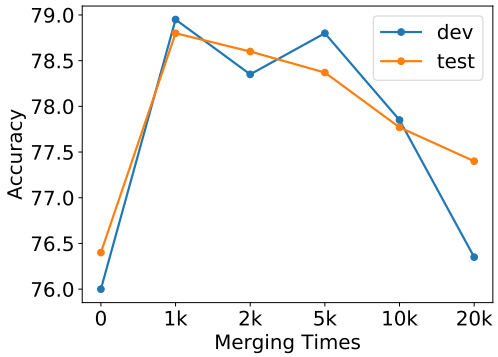


Figure 2: Case study of the subword vocabulary size of BPE.

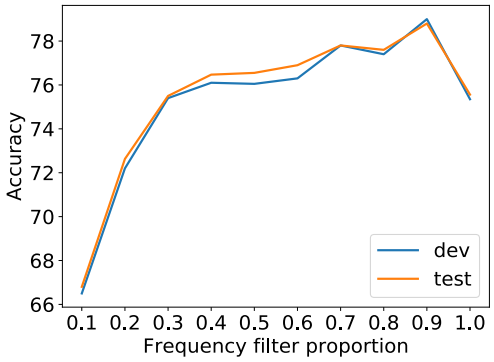


Figure 3: Quantitative study on the influence of the short list.

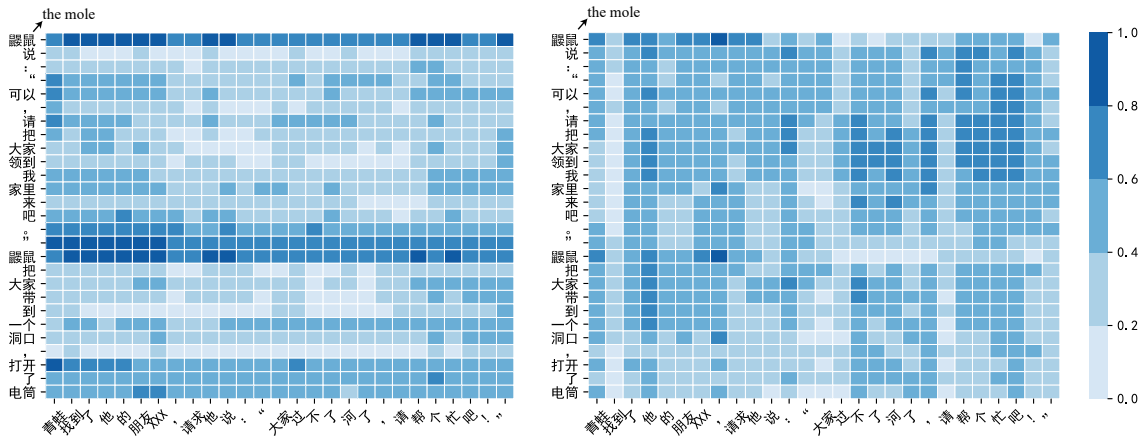
4.2 Filter Mechanism

To investigate the impact of the short list to the model performance, we conduct quantitative study on the filter ratio from [0.1, 0.2, ..., 1]. The results on the CMRC-2017 dataset are depicted in Figure 3. As we can see that when $\gamma = 0.9$ our SAW reader can obtain the best performance, showing that building the vocabulary among all the training set is not optimal and properly reducing the frequency filter ratio can boost the accuracy. This is partially attributed to training the model from the full vocabulary would cause serious over-fitting as the rare words representations can not obtain sufficient tuning. If the rare words are not initialized properly, they would also bias the whole word representations. Thus a model without OOV mechanism will fail to precisely represent those inevitable OOV words from test sets.

4.3 Subword-Augmented Representations

In text understanding tasks, if the ground-truth answer is OOV word or contains OOV word(s), the performance of deep neural networks would severely drop due to the incomplete representation, especially for cloze-style reading comprehension task where the answer is only one word or phrase. In CMRC-2017, we observe questions with OOV answers (denoted as "OOV questions") account for 17.22% in the error results of the best Word + Char embedding based model. With BPE subword embedding, 12.17% of these "OOV questions" could be correctly answered. This shows the subword representations could be essentially useful for modeling rare and unseen words.

To analyze the reading process of SAW Reader, we draw the attention distributions at intermediate



(a) Embedding of Document and query

(b) Final document and query representation

Doc (extract): The mole said, "That's fine, please bring them to my house." The mole took everyone to a hole, turned on the flashlight and asked the little white rabbit, the hedgehog, the big ant and the frog to follow him, saying, "Don't be afraid, just go ahead."
Query: The frog found his friend _____ and told him, We cannot get across the river. Please give us a hand!

Figure 4: Pair-wise attention visualization.

layers as shown in Figure 4. We observe the salient candidates in the document can be focused after the pair-wise matching of document and query and the right answer ("*The mole*") could obtain a high weight at the very beginning. After attention learning, the key evidence of the answer would be collected and irrelevant parts would be ignored. This shows our SAW Reader is effective at selecting the vital points at the fundamental embedding layer, guiding the attention layers to collect more relevant pieces.

5 Related Work

5.1 Machine Reading Comprehension

Recently, many deep learning models have been proposed for reading comprehension (Sordoni et al., 2016; Trischler et al., 2016; Wang and Jiang, 2016; Munkhdalai and Yu, 2017; Wang et al., 2017a; Dhingra et al., 2017; Zhang et al., 2018b; Wang et al., 2018b). Notably, Chen et al. (2016) conducted an in-depth and thoughtful examination on the comprehension task based on an attentive neural network and an entity-centric classifier with a careful analysis based on handful features. Kadlec et al. (2016) proposed the Attention Sum Reader (AS Reader) that uses attention to directly pick the answer from the context, which is motivated by the Pointer Network (Vinyals et al., 2015). Instead of summing the attention of query-to-document, GA Reader (Dhingra et al., 2017) defined an element-wise product to endowing attention on each word of the document using the entire query representation to build query-specific representations of words in the document for accurate answer selection. Wang et al. (2017b) employed gated self-matching networks (R-net) on passage against passage itself to refine passage representation with information from the whole passage. Cui et al. (2017a) introduced an "attended attention" mechanism (AoA) where query-to-document and document-to-query are mutually attentive and interactive to each other.

5.2 Augmented Word Embedding

Distributed word representation plays a fundamental role in neural models (Cai and Zhao, 2016; Qin et al., 2016; Zhao et al., 2017; Peters et al., 2018; He et al., 2018; Wang et al., 2018a; Bai and Zhao, 2018; Zhang et al., 2018a). Recently, character embeddings are widely used to enrich word representations (Kim et al., 2016; Yang et al., 2017; Luong and Manning, 2016; Huang et al., 2018). Yang et al. (2017) explored a fine-grained gating mechanism (FG Reader) to dynamically combine word-level and character-level representations based on properties of the words. However, this method is computationally complex and it is not end-to-end, requiring extra labels such as NE and POS tags. Seo et al. (2017)

concatenated the character and word embedding to feed a two-layer Highway Network.

Not only for machine reading comprehension tasks, character embedding has also benefit other natural language process tasks, such as word segmentation (Cai et al., 2017), machine translation (Luong and Manning, 2016), tagging (Yang et al., 2016; Li et al., 2018) and language modeling (Verwimp et al., 2017; Miyamoto and Cho, 2016). However, character embedding only shows marginal improvement due to a lack internal semantics. Lexical, syntactic and morphological information are also considered to improve word representation (Cao and Rei, 2016; Bergmanis and Goldwater, 2017). Bojanowski et al. (2017) proposed to learn representations for character n-gram vectors and represent words as the sum of the n-gram vectors. Avraham and Goldberg (2017) built a model inspired by (Joulin et al., 2017), who used morphological tags instead of n-grams. They jointly trained their morphological and semantic embeddings, implicitly assuming that morphological and semantic information should live in the same space. However, the linguistic knowledge resulting subwords, typically, morphological suffix, prefix or stem, may not be suitable for different kinds of languages and tasks. Sennrich et al. (2016) introduced the byte pair encoding (BPE) compression algorithm into neural machine translation for being capable of open-vocabulary translation by encoding rare and unknown words as subword units. Instead, we consider refining the word representations for both frequent and infrequent words from a computational perspective. Our proposed subword-augmented embedding approach is more general, which can be adopted to enhance the representation for each word by adaptively altering the segmentation granularity in multiple NLP tasks.

6 Conclusion

This paper presents an effective neural architecture, called subword-augmented word embedding to enhance the model performance for the cloze-style reading comprehension task. The proposed SAW Reader uses subword embedding to enhance the word representation and limit the word frequency spectrum to train rare words efficiently. With the help of the short list, the model size will also be reduced together with training speedup. Unlike most existing works, which introduce either complex attentive architectures or many manual features, our model is much more simple yet effective. Giving state-of-the-art performance on multiple benchmarks, the proposed reader has been proved effective for learning joint representation at both word and subword level and alleviating OOV difficulties.

References

Oded Avraham and Yoav Goldberg. 2017. The interplay of semantics and morphology in word embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 422–426.

Hongxiao Bai and Hai Zhao. 2018. Deep enhanced representation for implicit discourse relation recognition. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*.

Toms Bergmanis and Sharon Goldwater. 2017. From segmentation to analyses: a probabilistic model for unsupervised morphology induction. In *Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 337–346.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistic (TACL)*, 5:135–146.

Jan A. Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*, 32:1899–1907.

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for Chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 409–420.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and accurate neural word segmentation for Chinese. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 608–615.

Kris Cao and Marek Rei. 2016. A joint model for word embedding and word morphology. In *The Workshop on Representation Learning for NLP*, pages 18–26.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the CNN/Daily Mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 2358–2367.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1724–1734.

Yiming Cui, Ting Liu, Zhipeng Chen, Shijin Wang, and Guoping Hu. 2016. Consensus attention-based neural networks for Chinese reading comprehension. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers (COLING 2016)*, pages 1777–1786.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017a. Attention-over-attention neural networks for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1832–1846.

Yiming Cui, Ting Liu, Zhipeng Chen, Wentao Ma, Shijin Wang, and Guoping Hu. 2017b. Dataset for the first evaluation on Chinese machine reading comprehension. *arXiv preprint arXiv:1511.02301*.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1832–1846.

Gage and Philip. 1994. A new algorithm for data compression. *C Users Journal*, 12(2):23–38.

Shexia He, Zuchao Li, Hai Zhao, Hongxiao Bai, and Gongshen Liu. 2018. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.

Yafang Huang, Zuchao Li, Zhuosheng Zhang, and Hai Zhao. 2018. Moon IME: neural-based chinese pinyin aided input method with customizable association. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018), System Demonstration*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1601–1611.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 427–431.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 908–918.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016)*, pages 2741–2749.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Haonan Li, Zhisong Zhang, Yuqi Ju, and Hai Zhao. 2018. Neural character-level dependency parsing for Chinese. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

Minh-Thang Luong and Christopher D Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. *arXiv preprint arXiv:1604.00788*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Yasumasa Miyamoto and Kyunghyun Cho. 2016. Gated word-character recurrent language model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 1992–1997.

Tsendsuren Munkhdalai and Hong Yu. 2017. Reasoning with memory augmented neural networks for language comprehension. *Proceedings of the International Conference on Learning Representations (ICLR 2017)*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on International Conference on Machine Learning*, volume 28, page 13101318.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016. A stacking gated neural architecture for implicit discourse relation classification. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 2263–2270.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 2383–2392.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1715–1725.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of the International Conference on Learning Representations (ICLR 2017)*.

Alessandro Sordoni, Philip Bachman, Adam Trischler, and Yoshua Bengio. 2016. Iterative alternating neural attention for machine reading. *arXiv preprint arXiv:1606.02245*.

Adam Trischler, Zheng Ye, Xingdi Yuan, and Kaheer Suleman. 2016. Natural language comprehension with the epireader. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 128–137.

Lyan Verwimp, Joris Pelemans, Hugo Van Hamme, and Patrick Wambacq. 2017. Character-word LSTM language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 417–427.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. *In Advances in Neural Information Processing Systems*, pages 2692–2700.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using Match-LSTM and answer pointer. *Proceedings of the International Conference on Learning Representations (ICLR 2016)*.

Bingning Wang, Kang Liu, and Jun Zhao. 2017a. Conditional generative adversarial networks for commonsense machine comprehension. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI 2017)*, pages 4123–4129.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017b. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 189–198.

Rui Wang, Hai Zhao, Sabine Ploux, Bao-Liang Lu, Masao Utiyama, and Eiichiro Sumita. 2018a. Graph-based bilingual word embedding for statistical machine translation. *ACM Transaciton on Asian and Low-Resource Language Information Processing*, 17(4).

Yizhong Wang, Kai Liu, Jing Liu, Wei He, Yajuan Lyu, Hua Wu, Sujian Li, and Haifeng Wang. 2018b. Multi-passage machine reading comprehension with cross-passage answer verification.

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*.

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W. Cohen, and Ruslan Salakhutdinov. 2017. Words or characters? fine-grained gating for reading comprehension. In *Proceedings of the International Conference on Learning Representations (ICLR 2017)*.

Zhuosheng Zhang and Hai Zhao. 2018. One-shot learning for question-answering in gaokao history challenge. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*.

Zhuosheng Zhang, Jiangtong Li, Hai Zhao, and Bingjie Tang. 2018a. Sjtu-nlp at semeval-2018 task 9: Neural hypernym discovery with term embeddings. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval 2018), Workshop of NAACL-HLT 2018*.

Zhuosheng Zhang, Jiangtong Li, Pengfei Zhu, and Hai Zhao. 2018b. Modeling multi-turn conversation with deep utterance aggregation. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*.

Hai Zhao, Deng Cai, Yang Xin, Yuzhu Wang, and Zhongye Jia. 2017. A hybrid model for Chinese spelling check. *ACM Transactions on Asian Low-Resource Language Information Process*, pages 1–22.

Enhancing Sentence Embedding with Generalized Pooling

Qian Chen
University of Science and Technology of China
cq1231@mail.ustc.edu.cn

Zhen-Hua Ling
University of Science and Technology of China
zhling@ustc.edu.cn

Xiaodan Zhu
ECE, Queen's University
xiaodan.zhu@queensu.ca

Abstract

Pooling is an essential component of a wide variety of sentence representation and embedding models. This paper explores generalized pooling methods to enhance sentence embedding. We propose vector-based multi-head attention that includes the widely used max pooling, mean pooling, and scalar self-attention as special cases. The model benefits from properly designed penalization terms to reduce redundancy in multi-head attention. We evaluate the proposed model on three different tasks: natural language inference (NLI), author profiling, and sentiment classification. The experiments show that the proposed model achieves significant improvement over strong sentence-encoding-based methods, resulting in state-of-the-art performances on four datasets. The proposed approach can be easily implemented for more problems than we discuss in this paper.

1 Introduction

Distributed representation learned with neural networks has shown to be effective in modeling natural language at different granularities. Learning representation for words (Bengio et al., 2000; Mikolov et al., 2013; Pennington et al., 2014), for example, has achieved notable success. Much remains to be done to model larger spans of text such as sentences or documents. The approaches to computing sentence embedding generally fall into two categories. The first consists of learning sentence embedding with unsupervised learning, e.g., auto-encoder-based models (Socher et al., 2011), Paragraph Vector (Le and Mikolov, 2014), SkipThought vectors (Kiros et al., 2015), FastSent (Hill et al., 2016), among others. The second category consists of models trained with supervised learning, such as convolution neural networks (CNN) (Kim, 2014; Kalchbrenner et al., 2014), recurrent neural networks (RNN) (Conneau et al., 2017; Bowman et al., 2015), and tree-structure recursive networks (Socher et al., 2013; Zhu et al., 2015; Tai et al., 2015), just to name a few.

Pooling is an essential component of a wide variety of sentence representation and embedding models. For example, in recurrent-neural-network-based models, pooling is often used to aggregate hidden states at different time steps (i.e., words in a sentence) to obtain sentence embedding. Convolutional neural networks (CNN) also often uses max or mean pooling to obtain a fixed-size sentence embedding.

In this paper we explore generalized pooling methods to enhance sentence embedding. Specifically, by extending scalar self-attention models such as those proposed in Lin et al. (2017), we propose vector-based multi-head attention, which includes the widely used max pooling, mean pooling, and scalar self-attention itself as special cases. On one hand, the proposed method allows for extracting different aspects of the sentence into multiple vector representations through the multi-head mechanism. On the other, it allows the models to focus on one of many possible interpretations of the words encoded in the context vector through the vector-based attention mechanism. In the proposed model we design penalization terms to reduce redundancy in multi-head attention.

We evaluate the proposed model on three different tasks: natural language inference, author profiling, and sentiment classification. The experiments show that the proposed model achieves significant improvement over strong sentence-encoding-based methods, resulting in state-of-the-art performances on

Proceedings of the 27th International Conference on Computational Linguistics, pages 1815–1826
Santa Fe, New Mexico, USA, August 20-26, 2018.

four datasets. The proposed approach can be easily implemented for more problems than we discuss in this paper.

2 Related Work

There exist in the literature much previous work for sentence embedding with supervised learning, which mostly use RNN and CNN as building blocks. For example, Bowman et al. (2015) used BiLSTMs as sentence embedding for natural language inference task. Kim (2014) used CNN with max pooling for sentence classification. More complicated neural networks were also proposed for sentence embedding. For example, Socher et al. (2013) introduced Recursive Neural Tensor Network (RNTN) over parse trees to compute sentence embedding for sentiment analysis. Zhu et al. (2015) and Tai et al. (2015) proposed tree-LSTM. Yu and Munkhdalai (2017a) proposed a memory augmented neural networks, called Neural Semantic Encoder (NSE), as sentence embedding for natural language understanding tasks.

Some recent research began to explore inner/self-sentence attention mechanism for sentence embedding, which can be classified into two categories: self-attention network and self-attention pooling. Cheng et al. (2016) proposed an intra-sentence level attention mechanism on the base of LSTM, called LSTMN. For each step in LSTMN, it calculated the attention between a certain word and its previous words. Vaswani et al. (2017) proposed a self-attention network for the neural machine translation task. The self-attention network uses multi-head scaled dot-product attention to represent each word by weighted summation of all word in the sentence. Shen et al. (2017) proposed DiSAN, which is composed of a directional self-attention with temporal order encoded. Shen et al. (2018) proposed reinforced self-attention network (ReSAN), which integrate both soft and hard attention into one context fusion with reinforced learning.

Self-attention pooling has also been studied in previous work. Liu et al. (2016) proposed inner-sentence attention based pooling methods for sentence embedding. They calculate scalar attention between the LSTM states and the mean pooling using multi-layer perceptron (MLP) to obtain the vector representation for a sentence. Lin et al. (2017) proposed a scalar structure/multi-head self-attention method for sentence embedding. The multi-head self-attention is calculated by a MLP with only LSTM states as input. There are two main differences from our proposed method; i.e., (1) they used scalar attention instead of vectorial attention, (2) we propose different penalization terms which is suitable for vector-based multi-head self-attention, while their penalization term on attention matrix is only designed for scalar multi-head self-attention. Choi et al. (2018) proposed a fine-grained attention mechanism for neural machine translation, which also extend scalar attention to vectorial attention. Shen et al. (2017) proposes multi-dimensional/vectorial self-attention pooling on the top of self-attention network instead of BiLSTM. However, both of them didn't consider multi-head self-attention.

3 The Model

In this section we describe the proposed models that enhance sentence embedding with generalized pooling approaches. The pooling layer is built on a state-of-the-art sequence encoder layer. Below, we first discuss the sequence encoder, which, when enhanced with the proposed generalized pooling, achieves state-of-the-art performance on three different tasks on four datasets.

3.1 Sequence Encoder

The sequence encoder in our model takes into T word tokens of a sentence $\boldsymbol{S} = (w_1, w_2, \ldots, w_T)$. Each word w_i is from the vocabulary $\boldsymbol{V}$. For each word we concatenate pre-trained word embedding and embedding learned from characters. The character composition model feeds all characters of the word into a convolution neural network (CNN) with max pooling (Kim, 2014). The detailed experiment setup will be discussed in Section 4. The sentence $\boldsymbol{S}$ is represented as a word embedding sequence: $\boldsymbol{X} = (\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_T) \in \mathbb{R}^{T \times d_e}$, where d_e is the dimension of word embedding which concatenates embedding obtained from character composition and pretrained word embedding.

To represent words and their context in sentences, the sentences are fed into stacked bidirectional LSTMs (BiLSTMs). Shortcut connections are applied, which concatenate word embeddings and input

hidden states at each layer in the stacked BiLSTM except for the first (bottom) layer. The formulae are as follows:

$$\overrightarrow{\boldsymbol{h}}_t^l = \text{LSTM}([\boldsymbol{e}_t; \overrightarrow{\boldsymbol{h}}_t^{l-1}], \overrightarrow{\boldsymbol{h}}_{t-1}^l)\,, \tag{1}$$

$$\overleftarrow{\boldsymbol{h}}_t^l = \text{LSTM}([\boldsymbol{e}_t; \overleftarrow{\boldsymbol{h}}_t^{l-1}], \overleftarrow{\boldsymbol{h}}_{t+1}^l)\,, \tag{2}$$

$$\boldsymbol{h}_t^l = [\overrightarrow{\boldsymbol{h}}_t^l; \overleftarrow{\boldsymbol{h}}_t^l]\,. \tag{3}$$

where hidden states $\boldsymbol{h}_t^l$ in layer l concatenate two directional hidden states of LSTM at time t. Then the sequence is represented as the hidden states in the top layer L: $\mathbf{H}^L = (\boldsymbol{h}_1^L, \boldsymbol{h}_2^L, \ldots, \boldsymbol{h}_T^L) \in \mathbb{R}^{T\times 2d}$. For simplicity, we ignore the superscript L in the remainder of the paper.

3.2 Generalized Pooling

3.2.1 Vector-based Multi-head Attention

To transform a variable length sentence into a fixed size vector representation, we propose a generalized pooling method. We achieve that by using a weighted summation of the T LSTM hidden vectors, and the weights are vectors rather than scalars, which can control every element in all hidden vectors:

$$\mathbf{A} = \text{softmax}(\mathbf{W}_2\text{ReLU}(\mathbf{W}_1\mathbf{H}^{\mathrm{T}} + \boldsymbol{b}_1) + \boldsymbol{b}_2)^{\mathrm{T}}\,, \tag{4}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_a\times 2d}$ and $\mathbf{W}_2 \in \mathbb{R}^{2d\times d_a}$ are weight matrices; $\boldsymbol{b}_1 \in \mathbb{R}^{d_a}$ and $\boldsymbol{b}_2 \in \mathbb{R}^{2d}$ are bias, where d_a is the dimension of attention network and d is the dimension of LSTMs. $\mathbf{H} \in \mathbb{R}^{T\times 2d}$ and $\mathbf{A} \in \mathbb{R}^{T\times 2d}$ are the hidden vectors at the top layer and weight matrices, respectively. The softmax ensures that $(\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_T)$ are non-negative and sum up to 1 for every element in vectors. Then we sum up the LSTM hidden states $\mathbf{H}$ according to the weight vectors provided by $\mathbf{A}$ to get a vector representation $\boldsymbol{v}$ of the input sentence.

However, the vector representation usually focuses on a specific component of the sentence, like a special set of related words or phrases. We extend pooling method to a multi-head way:

$$\mathbf{A}^i = \text{softmax}(\mathbf{W}_2^i\text{ReLU}(\mathbf{W}_1^i\mathbf{H}^{\mathrm{T}} + \boldsymbol{b}_1^i) + \boldsymbol{b}_2^i)^{\mathrm{T}}\,, \forall i \in 1, \ldots, I\,, \tag{5}$$

$$\boldsymbol{v}^i = \sum_{t=1}^{T} \boldsymbol{a}_t^i \odot \boldsymbol{h}_t^i\,, \forall i \in 1, \ldots, I\,, \tag{6}$$

where $\boldsymbol{a}_t^i$ indicates the vectorial attention from $\mathbf{A}^i$ for the t-th token in i-th head and $\odot$ is the element-wise product (also called the Hadamard product). Thus the final representation is a concatenated vector $\boldsymbol{v} = [\boldsymbol{v}^1; \boldsymbol{v}^2; \ldots; \boldsymbol{v}^I]$, where each $\boldsymbol{v}^i$ captures different aspects of the sentence. For example, some heads of vectors may represent the predicate of sentence and other heads of vectors represent argument of the sentence, which enhances representation of sentences obtained in single-head attention.

3.2.2 Penalization Terms

To reduce the redundancy of multi-head attention, we design penalization terms for vector-based multi-head attention in order to encourage the diversity of summation weight across different heads of attention. We propose three types of penalization terms.

Penalization Term on Parameter Matrices The first penalization term is applied to parameter matrix $\mathbf{W}_1^i$ in Equation 5, as shown in the following formula:

$$P = \mu \sum_{i=1}^{I} \sum_{j=i+1}^{I} \max(\lambda - \|\mathbf{W}_1^i - \mathbf{W}_1^j\|_{\mathrm{F}}^2, 0)\,. \tag{7}$$

Intuitively, we encourage different heads to have different parameters. We maximum the *Frobenius* norm of the differences between two parameter matrices, resulting in encouraging the diversity of different heads. It has no further bonus when the *Frobenius* norm of the difference of two matrices exceeds the a threshold λ. Similar to adding an L2 regularization term on neural networks, the penalization term P will be added to the original loss with a weight of μ. Hyper-parameters λ and μ need to be tuned on a development set. We can also add constrains on $\mathbf{W}_2^i$ in a similar way, but we did not observe further improvement in our experiments.

Penalization Term on Attention Matrices The second penalization term is added on attention matrices. Instead of using $\|\mathbf{A}\mathbf{A}^{\mathrm{T}} - \mathbf{I}\|_{\mathrm{F}}^2$ to encourage the diversity for scalar attention matrix as in Lin et al. (2017), we propose the following formula to encourage the diversity for vectorial attention matrices. The penalization term on attention matrices is

$$P = \mu \sum_{i=1}^{I} \sum_{j=i+1}^{I} \max(\lambda - \|\mathbf{A}^i - \mathbf{A}^j\|_{\mathrm{F}}^2, 0)\,, \tag{8}$$

where λ and μ are hyper-parameters which need to be tuned based on a development set. Intuitively, we try to encourage the diversity of any two different $\mathbf{A}^i$ under the threshold λ.

Penalization Term on Sentence Embeddings In addition, we propose to add a penalization term on multi-head sentence embedding $\boldsymbol{v}^i$ directly as follows:

$$P = \mu \sum_{i=1}^{I} \sum_{j=i+1}^{I} \max(\lambda - \|\boldsymbol{v}^i - \boldsymbol{v}^j\|_2^2, 0)\,, \tag{9}$$

where λ and μ are hyper-parameters. Here we try to maximize the l^2-norm of any two different heads of sentence embeddings under the threshold λ.

3.3 Top-layer Classifiers

The output of pooling is fed to a top-layer classifier to solve different problems. In this paper we evaluate our sentence embedding models on three different tasks: natural language inference (NLI), author profiling, and sentiment classification, on four datasets. The evaluation covers two typical types of problems. The author profiling and sentiment tasks classify individual sentences into different categories and the two NLI tasks classify sentence pairs.

For the NLI tasks, to enhance the relationship between sentence pairs, we concatenate the embeddings of two sentences with their absolute difference and element-wise product (Mou et al., 2016) as the input to the multilayer perceptron (MLP) classifier:

$$\boldsymbol{v} = [\boldsymbol{v}_a; \boldsymbol{v}_b; |\boldsymbol{v}_a - \boldsymbol{v}_b|; \boldsymbol{v}_a \odot \boldsymbol{v}_b]\,, \tag{10}$$

where $\odot$ is the element-wise product. The MLP has two hidden layers with *ReLU* activation with shortcut connections and a *softmax* output layer. The entire model is trained end-to-end through minimizing the cross-entropy loss. Note that for the two classification tasks on individual sentences (i.e., the author profiling and sentiment classification task), we use the same MLP classifiers described above for sentence pair classification. But instead of concatenating two sentences, we directly feed a sentence embedding into MLP.

4 Experimental Setup

4.1 Data

SNLI The SNLI (Bowman et al., 2015) is a large dataset for natural language inference. The task detects three relationships between a premise and a hypothesis sentence: the premise entails the hypothesis

(*entailment*), they contradict each other (*contradiction*), or they have a neutral relation (*neutral*). We use the same data split as in Bowman et al. (2015), i.e., 549.367 samples for training, 9,842 samples for development and 9,824 samples for testing.

MultiNLI MultiNLI (Williams et al., 2017) is another natural language inference dataset. The data are collected from a broader range of genres such as fiction, letters, telephone speech, and 9/11 reports. Half of these 10 genres are used in training while the rest are not, resulting in-domain and cross-domain development and test sets used to test NLI systems. We use the same data split as in Williams et al. (2017), i.e., 392,702 samples for training, 9,815/9,832 samples for in-domain/cross-domain development, and 9,796/9,847 samples for in-domain/cross-domain testing. Note that, we do not use SNLI as an additional training/development set in our experiments.

Age Dataset To compare our models with that of Lin et al. (2017), we use the same Age dataset in our experiment here, which is an Author Profiling dataset. The dataset are extracted from the Author Profiling dataset[1], which consists of tweets from English Twitter. The task is to predict the age range of authors of input tweets. The age range are split into 5 classes: 18-24, 25-34, 35-49, 50-64, 65+. We use the same data split as in Lin et al. (2017), i.e., 68,485 samples for training, 4,000 for development, and 4,000 for testing.

Yelp Dataset The Yelp dataset[2] is a sentiment analysis task, which takes reviews as input and predicts the level of sentiment in terms of the number of stars, from 1 to 5 stars, where 5-star means the most positive. We use the same data split as in Lin et al. (2017), i.e., 500,000 samples for training, 2,000 for development, and 2,000 for testing.

4.2 Training Details

We implement our algorithm with Theano (Theano Development Team, 2016) framework. We use the development set (in-domain development set for MultiNLI) to select models for testing. To help replicate our results, we publish our code[3], which is developed from our codebase for multiple tasks (Chen et al., 2018; Chen et al., 2017a; Chen et al., 2016; Zhang et al., 2017). Specifically, we use Adam (Kingma and Ba, 2014) for optimization. The initial learning rate is 4e-4 for SNLI and MultiNLI, 2e-3 for Age dataset, 1e-3 for Yelp dataset. For SNLI and MultiNLI dataset, stacked BiLSTMs have 3 layers. For Age and Yelp dataset, stacked BiLSTMs have 1 layer. The hidden states of BiLSTMs for each direction and MLP are 300 dimension, except for SNLI whose dimensions are 600. We clip the norm of gradients to make it smaller than 10 for SNLI and MultiNLI, and 0.5 for Age and Yelp dataset. The character embedding has 15 dimensions, and 1D-CNN filters lengths are 1, 3 and 5, respectively. Each filter has 100 feature maps, resulting in 300 dimensions for character-composition embedding. We initialize word-level embedding with pre-trained *GloVe-840B-300D* embeddings (Pennington et al., 2014) and initialize out-of-vocabulary words randomly with a Gaussian distribution. The word-level embedding is fixed during training for SNLI and MultiNLI dataset, but updated during training for Age and Yelp dataset, which is determined by the performance on development sets. The mini-batch size is 128 for SNLI and 32 for the rest. We use 5 heads generalized pooling for all tasks. And d_a is 600 for SNLI and 300 for the other datasets. For the penalization term, we choose $\lambda = 1$; the penalization weight μ is selected from [1,1e-1,1e-2,1e-3,1e-4] based on performances on the development sets.

5 Experimental Results

5.1 Overall Performance

For the NLI tasks, there are many ways to add cross-sentence (Rocktäschel et al., 2015; Parikh et al., 2016; Chen et al., 2017a) level attention. To ensure the comparison is fair, we only compare methods that use sentence-encoding-based models; i.e., cross-sentence attention is not allowed. Note that this

[1] http://pan.webis.de/clef16/pan16-web/author-profiling.html
[2] https://www.yelp.com/dataset/challenge
[3] https://github.com/lukecq1231/generalized-pooling

Model	Test
100D LSTM (Bowman et al., 2015)	77.6
300D LSTM (Bowman et al., 2016)	80.6
1024D GRU (Vendrov et al., 2015)	81.4
300D Tree CNN (Mou et al., 2016)	82.1
600D SPINN-PI (Bowman et al., 2016)	83.3
600D BiLSTM (Liu et al., 2016)	83.3
300D NTI-SLSTM-LSTM (Yu and Munkhdalai, 2017b)	83.4
600D BiLSTM intra-attention (Liu et al., 2016)	84.2
600D BiLSTM self-attention (Lin et al., 2017)	84.4
4096D BiLSTM max pooling (Conneau et al., 2017)	84.5
300D NSE (Yu and Munkhdalai, 2017a)	84.6
600D BiLSTM gated-pooling (Chen et al., 2017b)	85.5
300D DiSAN (Shen et al., 2017)	85.6
300D Gumbel TreeLSTM (Choi et al., 2017)	85.6
600D Residual stacked BiLSTM (Nie and Bansal, 2017)	85.7
300D CAFE (Tay et al., 2018)	85.9
600D Gumbel TreeLSTM (Choi et al., 2017)	86.0
1200D Residual stacked BiLSTM (Nie and Bansal, 2017)	86.0
300D ReSAN (Shen et al., 2018)	**86.3**
1200D BiLSTM max pooling	85.3
1200D BiLSTM mean pooling	85.0
1200D BiLSTM last pooling	84.9
1200D BiLSTM **generalized pooling**	**86.6**

Table 1: Accuracies of the models on the SNLI dataset.

follows the setup in the RepEval-2017 Shared Task. Table 1 shows the results of different models for NLI, consisting of results of previous work on sentence-encoding-based models, plus the performance of our baselines and that of the model proposed in this paper. We have three additional baseline models: the first uses max pooling on top of BiLSTM, which achieves an accuracy of 85.3%; the second uses mean pooling on top of BiLSTM, which achieves an accuracy of 85.0%; the third uses last pooling, i.e., concatenating the last hidden states of forward and backward LSTMs, which achieves an accuracy of 84.9%. Instead of using heuristic pooling methods, the proposed sentence-encoding-based model with generalized pooling achieves a new state-of-the-art accuracy of 86.6% on the SNLI dataset; the improvement over the baseline with max pooling is statistically significant under the one-tailed paired t-test at the 99.999% significance level. The previous state-of-the-art model ReSAN (Shen et al., 2018) used a hybrid of hard and soft attention model with reinforced learning achieved an accuracy of 86.3%.

Table 2 shows the results of different models on the MultiNLI dataset. The first group is the results of previous sentence-encoding-based models. The proposed model with generalized pooling achieves an accuracy of 73.8% on the in-domain test set and 74.0% on the cross-domain test set; both improve over the baselines using max pooling, mean pooling and last pooling. In addition, the results on cross-domain test set yield a new state of the art at an accuracy of 74.0%, which is better than 73.6% of shortcut-stacked BiLSTM (Nie and Bansal, 2017).

Table 3 shows the results of different models for the Yelp and the Age dataset. The BiLSTM with self-attention proposed by Lin et al. (2017) achieves better result than CNN and BiLSTM with max pooling. One of our baseline models using max pooling on BiLSTM achieves accuracies of 65.00% and 82.30% on the Yelp and the Age dataset respectively, which is already better than the self-attention model proposed by Lin et al. (2017). We also show that the results of baseline with mean pooling and last pooling, in which mean pooling achieves the best result on the Yelp dataset among three baseline models and max pooling achieves the best on the Age dataset among three baselines. Our proposed generalized pooling method obtains further improvement on these already strong baselines, achieving 66.55% on the Yelp dataset and 82.63% on the Age dataset (statistically significant $p < 0.00001$ against best baselines),

Model	In	Cross
CBOW (Williams et al., 2017)	64.8	64.5
BiLSTM (Williams et al., 2017)	66.9	66.9
BiLSTM gated-pooling (Chen et al., 2017b)	73.5	**73.6**
Shortcut stacked BiLSTM (Nie and Bansal, 2017)	**74.6**	**73.6**
BiLSTM max pooling	73.6	73.0
BiLSTM mean pooling	71.5	71.6
BiLSTM last pooling	71.6	71.9
BiLSTM **generalized pooling**	**73.8**	**74.0**

Table 2: Accuracies of the models on the MultiNLI dataset.

Model	Yelp	Age
BiLSTM max pooling (Lin et al., 2017)	61.99	77.30
CNN max pooling (Lin et al., 2017)	62.05	78.15
BiLSTM self-attention (Lin et al., 2017)	**64.21**	**80.45**
BiLSTM max pooling	65.00	82.30
BiLSTM mean pooling	65.30	81.78
BiLSTM last pooling	64.95	81.10
BiLSTM **generalized pooling**	**66.55**	**82.63**

Table 3: Accuracies of the models on the Yelp and Age dataset.

which are also new state of the art performances on these two datasets.

5.2 Detailed Analysis

Effect of Multiple Vectors/Scalars To compare the difference between vector-based attention and scalar attention, we draw the learning curves of different models using different heads on the SNLI development dataset without penalization terms as in Figure 1. The green lines indicate scalar self-attention pooling added on top of the BiLSTMs, same as in Lin et al. (2017), and the blue lines indicate vector-based attention used in our generalized pooling methods. It is obvious that the vector-based attention achieves improvement over scalar attention. Different line styles are used to indicate self-attention using different numbers of multi-head, ranging from 1, 3, 5, 7 to 9. For vector-based attention, the 9-head model achieves the best accuracy of 86.8% on the development set. For scalar attention, the 7-head model achieves the best accuracy of 86.4% on the development set.

Effect of Penalization Terms To analyze the effect of penalization terms, we show the results with/without penalization terms on the four datasets in Table 4. Without using any penalization terms, the proposed generalized pooling achieves an accuracy of 86.4% on the SNLI dataset, which is already slightly better than previous models (compared to accuracy 86.3% in Shen et al. (2018)). When we use penalization on parameter matrices, the proposed model achieves a further improvement with an accuracy of 86.6%. In addition, we also observe a significant improvement on MultiNLI, Yelp and Age dataset after using the penalization terms. For the MultiNLI dataset, the proposed model with penalization on parameter matrices achieves an accuracy of 73.8% and 74.0% on the in-domain and the cross-domain test set, respectively, which outperform the accuracy of 73.7% and 73.4% of the model without penalization, respectively. For the Yelp dataset, the proposed model with penalization on parameter matrices achieves the best results among the three penalization methods, which also improve the accuracy of 65.25% to 66.55% compared to the models without penalization. For the Age dataset, the proposed model with penalization on attention matrices achieves the best accuracy of 82.63%, compared to the 82.18% accuracy of the model without penalization. In general, the penalization on parameter matrices achieves the most effective improvement among most of these tasks, except for the Age dataset.

To verify whether the penalization term P discourages the redundancy in the sentence embedding, we

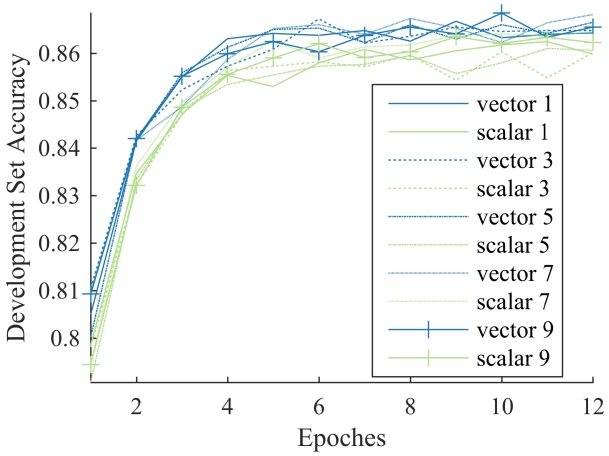


Figure 1: The effect of the number of heads and vectors/scalars in sentence embedding. The vertical axis indicates the development-set accuracy and the horizontal axis indicates training epochs. Numbers in the legend are the number of heads.

Model	**SNLI**	**MultiNLI**		**Yelp**	**Age**
		IN	**Cross**		
w/ Penalization on Parameter Matrices	**86.6**	**73.8**	**74.0**	**66.55**	82.45
w/ Penalization on Attention Matrices	86.2	73.6	73.8	66.15	**82.63**
w/ Penalization on Sentence Embeddings	86.1	73.5	73.6	65.75	82.15
w/o Penalization	86.4	73.7	73.4	65.25	82.18

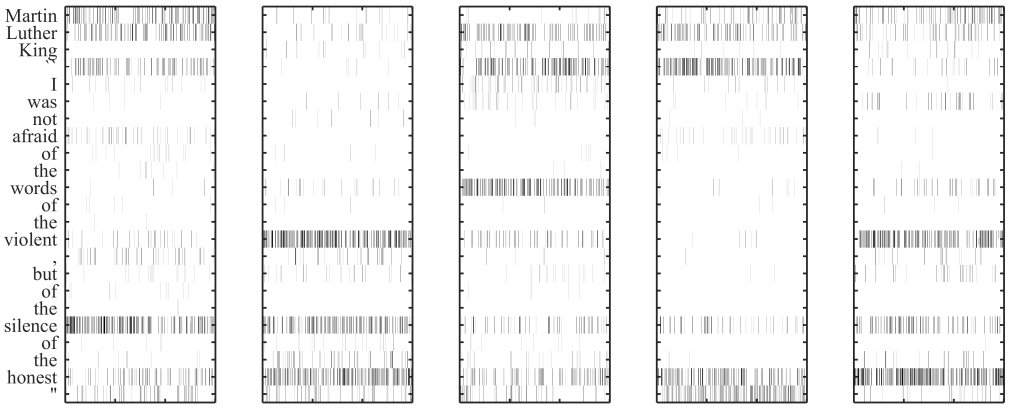
Table 4: Performance with/without the penalization term. The penalization weight is selected from [1,1e-1,1e-2,1e-3,1e-4] on the development sets.

visualize the vectorial multi-head attention according. We compare two models with the same hyper-parameters except that one is with penalization on attention matrices and the other without penalization. We pick a sentence from the development set of the Age data: *Martin Luther King "I was not afraid of the words of the violent, but of the silence of the honest"* , with the gold label being the category of 65+. We plot all 5 heads of attention matrices as in Figure 2. From the figure we can tell that the model trained without the penalization term has much more redundancy between different heads of attention (Figure 3b), resulting in putting significant focus on the word "Martin" in the 1st, 3rd and 5th head, and on the word "violent" in the 2nd and 4th head. However in Figure 3a, the model with penalization shows much more variation between different heads.

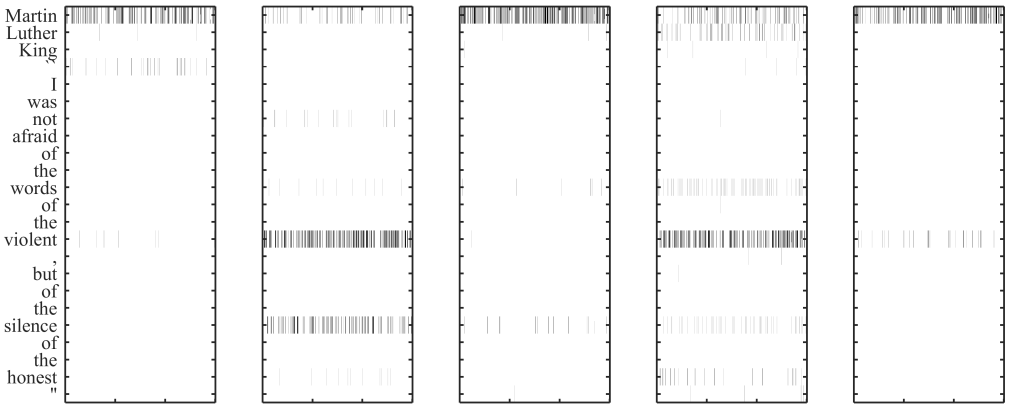
6 Conclusions

In this paper, we propose a generalized pooling method for sentence embedding through vector-based multi-head attention, which includes the widely used max pooling, mean pooling, and scalar self-attention as its special cases. Specifically the proposed model aims to use vectors to enrich the expressiveness of attention mechanism and leverage proper penalty terms to reduce redundancy in multi-head attention. We evaluate the proposed approach on three different tasks: natural language inference, author profiling, and sentiment classification. The experiments show that the proposed model achieves significant improvement over strong sentence-encoding-based methods, resulting in state-of-the-art performances on four datasets. The proposed approach can be easily implemented for more problems than we discuss in this paper.

Our future work includes exploring more effective MLP to use the structures of multi-head vectors, inspired by the idea from Lin et al. (2017). Leveraging structure information from syntactic and semantic parses is another direction interesting to us.



(a) Age dataset with penalization



(b) Age dataset without penalization

Figure 2: Visualization of vectorial multi-head attention. The vertical and horizontal axes indicate the source word tokens and the 600 dimensions of the attention $\mathbf{A}^i$ for different heads.

Acknowledgments

This work was partially funded by the National Natural Science Foundation of China (Grant No. U1636201) and the Key Science and Technology Project of Anhui Province (Grant No. 17030901005).

References

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. In *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 932–938.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 632–642.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling document. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2754–2760.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017a. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1657–1668.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017b. Recurrent neural network-based sentence encoder with gated attention for natural language inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP, RepEval@EMNLP 2017, Copenhagen, Denmark, September 8, 2017*, pages 36–40.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. 2018. Neural natural language inference models enhanced with external knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018*.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 551–561.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2017. Unsupervised learning of task-specific tree structures with tree-lstms. *CoRR*, abs/1707.02786.

Heeyoul Choi, Kyunghyun Cho, and Yoshua Bengio. 2018. Fine-grained attention mechanism for neural machine translation. *Neurocomputing*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 670–680.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1367–1377.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

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

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3294–3302.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1188–1196.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *CoRR*, abs/1703.03130.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional LSTM model and inner-attention. *CoRR*, abs/1605.09090.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*.

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP, RepEval@EMNLP 2017, Copenhagen, Denmark, September 8, 2017*, pages 41–45.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2249–2255.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *CoRR*, abs/1509.06664.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *CoRR*, abs/1709.04696.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Sen Wang, and Chengqi Zhang. 2018. Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling. *CoRR*, abs/1801.10296.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 151–161.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1556–1566.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. A compare-propagate architecture with alignment factorization for natural language inference. *CoRR*, abs/1801.00102.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2015. Order-embeddings of images and language. *CoRR*, abs/1511.06361.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *CoRR*, abs/1704.05426.

Hong Yu and Tsendsuren Munkhdalai. 2017a. Neural semantic encoders. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 397–407.

Hong Yu and Tsendsuren Munkhdalai. 2017b. Neural tree indexers for text understanding. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 11–21.

Junbei Zhang, Xiaodan Zhu, Qian Chen, Lirong Dai, Si Wei, and Hui Jiang. 2017. Exploring question understanding and adaptation in neural-network-based question answering. *CoRR*, abs/arXiv:1703.04617v2.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1604–1612.

Treat us like the sequences we are: Prepositional Paraphrasing of Noun Compounds using LSTM

Girishkumar Ponkiya[†], **Kevin Patel**[†], **Pushpak Bhattacharyya**[†] and **Girish K Palshikar**[‡]
[†]Indian Institute of Technology Bombay, Mumbai
[‡]TCS Research, Tata Consultancy Services, Pune
{girishp,kevin.patel,pb}@cse.iitb.ac.in, gk.palshikar@tcs.com

Abstract

Interpreting noun compounds is a challenging task. It involves uncovering the underlying predicate which is dropped in the formation of the compound. In most cases, this predicate is of the form VERB+PREP. It has been observed that uncovering the preposition is a significant step towards uncovering the predicate.

In this paper, we attempt to paraphrase noun compounds using prepositions. We consider noun compounds and their corresponding prepositional paraphrases as parallelly aligned sequences of words. This enables us to adapt different architectures from cross-lingual embedding literature. We choose the architecture where we create representations of both noun compound (source sequence) and its corresponding prepositional paraphrase (target sequence), such that their similarity is high. We use LSTMs to learn these representations. We use these representations to decide the correct preposition. Our experiments show that this approach performs considerably well on different datasets of noun compounds that are manually annotated with prepositions.

1 Introduction

A noun compound is a sequence of two or more nouns which have a well-defined meaning when written together. For example, *orange juice*, *colon cancer*, *research paper submission*, *paper submission deadline*, *etc.* The fact that "*juice is made from orange*" is hidden in *orange juice*. Noun compound interpretation deals with uncovering such hidden relations.

Noun compounds are usually interpreted in two ways: labelling and paraphrasing. Labelling involves assigning a semantic relation to a noun compound *e.g.*, *student protest*: AGENT, *orange juice*: MADEOF, *etc.*. These relations come from a set of a predefined taxonomy of semantic relations (Lauer, 1995; Warren, 1978; Barker and Szpakowicz, 1998; Girju et al., 2003; Tratz and Hovy, 2010; Ponkiya et al., 2018). Such detailed, fine-grained information can be useful for downstream tasks such as machine translation (Baldwin and Tanaka, 2004; Balyan and Chatterjee, 2015), question answering (Ahn et al., 2005), text entailment (Nakov, 2013), *etc.* Unfortunately, there is a lack of standard taxonomy. There is no consensus on which set of labels should be uniformly used.

On the other hand, paraphrasing involves rewriting the noun compound as a paraphrase which conveys its meaning explicitly, *e.g.*, *orange juice*: "*juice made from orange*" or "*juice with orange flavour*". Generic paraphrasing has been relatively less pursued (Butnariu et al., 2009; Hendrickx et al., 2013).

Prepositional paraphrasing – paraphrasing using only prepositions, *e.g.*, *orange juice*: "*juice of orange*" – is a simpler version of generic paraphrasing. The advantage of prepositional paraphrasing as compared to labelling is that the set of prepositions is finite, limited and pre-defined. However, the shortcoming is that the information is too coarse-grained for downstream tasks.

Prepositional paraphrasing of noun compounds is a useful subtask to solve. Our preliminary analysis reveals a strong correlation between labels of certain taxonomies and prepositions (refer Section 3 for more details). This was also observed by Girju et al. (2005), who concluded that knowledge of preposition can aid labelling. Also, it filters out the verbs that go with the preposition for verb+preposition

Proceedings of the 27th International Conference on Computational Linguistics, pages 1827–1836
Santa Fe, New Mexico, USA, August 20-26, 2018.

(VERB+PREP) paraphrasing. For example, if the preposition used in paraphrase is *at*, the verbs that go along with mainly be of type LOCATION. Thus, it can aid verb+preposition paraphrasing. Moreover, uncovering of a preposition is enough for some NLP application like Hindi-to-English translation (Kulkarni et al., 2012). Thus, prepositional paraphrasing is a useful first step towards noun compound interpretation.

Deep learning has made tremendous progress in various fields. One of the significant contributions of deep learning in NLP is word embeddings. They are dense real-valued representations of words learnt in an unsupervised manner. Their use has advanced the state in many applications (word sense disambiguation - Rothe and Schütze (2015), named entity recognition - Lample et al. (2016), sentiment classification - Tang et al. (2014), sarcasm detection - Joshi et al. (2016), *etc.*). Word embeddings enable words to share statistical strength. For instance, pattern learnt for the word *hotel* could be used to a good extent for the word *motel*, since they are semantically similar (which is elicited by the similarity between their corresponding word embeddings). This motivates us to investigate the use of word embeddings for noun compound interpretation.

Another significant contribution of deep learning to the field of NLP is the introduction of Encoder Decoder architectures (Bahdanau et al., 2015) for different tasks involving sequences. In such models, an input sequence of words is converted to a sequence of corresponding embeddings, which is then encoded into a single embedding via a part of the network known as Encoder. The single embedding is known as sequence embedding, sequence representation, *etc.* The second part of the network, the Decoder, then uses this sequence embedding in tandem with other information to produce the desired output. Such architectures are commonly used in machine translation, where the sequence embeddings are learnt such that the embeddings of a sentence in a source language and its parallelly aligned sentence in target language have high similarity.

As far as prepositional paraphrasing is concerned, one can observe that both the noun compounds as well as their corresponding paraphrases are parallelly aligned sequences of words. Thus, we can also make use of various sequence learning models from deep learning such as RNN (Werbos, 1990; Rumelhart et al., 1986), LSTM (Hochreiter and Schmidhuber, 1997), *etc.* as encoders to learn their representations.

Thus, in this paper, we raise the following question:

Can sequence representations, learned from word embeddings of components of a noun compound, help in its prepositional paraphrasing?

We attempt to answer this question in the following manner: we encode a noun compound and its prepositional paraphrase through two different LSTMs. Then, we train a network such that the encodings of a noun compound and its corresponding prepositional paraphrase have high similarity. Using this core intuition, our approach is able to generate the correct prepositional paraphrase. We evaluated our approach on four different datasets. Our approach performs reasonably well on a relatively small dataset, and outperforms others on the larger datasets.

The rest of the paper is organized as follows: Section 2 furnishes the necessary background. Section 3 motivates the need for prepositional paraphrasing. Section 4 details the sequence learning based architecture for learning representations of noun compounds and prepositional paraphrases, which we use to generate the correct prepositional paraphrase. Section 5 provides the experimental setup: the datasets used and the training/testing procedure used. Section 6 discusses the results of our experiments and some analysis, followed by a conclusion and future work.

2 Background and Related Work

A relation between the components of a noun compound can be represented in either of the following two ways: (1) Labelling: assigning a relation from a predefined set of semantic relations (*e.g.*, *apple juice*: MADE_OF), or (2) Paraphrasing: using a paraphrase to convey the underlying semantic relation (*e.g.*, *apple juice*: *"juice extracted from an apple"* or *"juice with apple flavor"*).

Over the years, many sets of semantic relations have been proposed (Levi, 1978; Warren, 1978; Lauer, 1995; Ó Séaghdha, 2007; Rosario et al., 2002; Barker and Szpakowicz, 1998; Vanderwende, 1994;

Tratz and Hovy, 2010; Ponkiya et al., 2018). Most of these sets have been created with the assumption that most predicates are of the form verb+preposition. For example, The CAUSED-BY relation in Levi (1978), or the USER/RECIPIENT+THINGUSED/RECEIVED relation in Tratz and Hovy (2010) which paraphrases to *thing used by user*. One may observe that these labels are motivated from verb+preposition constructions. We believe that preposition uncovering will help in the verb+preposition paraphrasing of noun compounds.

A system for automatic uncovering of the predicate can be developed in two ways: rule-based or statistical. A rule-based system will involve linguistic analysis of both w_1 and w_2. In this case, we may end up with more exceptions than the rules. Thus statistical approaches are the way to go. Among statistical approaches, supervised approaches rely on annotated data that needs to be sufficiently large and representative enough of the underlying problem. However, such datasets are rare, and the ones that do exist are small and heavily skewed, which makes the learning more difficult. For example, ≈80% of noun compounds in Girju (2007)'s dataset are annotated with the preposition *of*. The problems arising in such a supervised setting have been well studied by Girju et al. (2005).

On the other hand, Lauer (1995) and Lapata and Keller (2004) have proposed unsupervised models for identifying preposition for an interpretation of noun compounds. Lauer (1995) models $P(prep|n_1, n_2)$ (probability of preposition given two components of a noun compound) using frequency of triples $\langle n_1, prep, n_2 \rangle$. In an alternative model to handle sparsity, she used the following:

$$prep^* = \arg\max_{prep} \sum_{t1 \in cats(n_1), t_2 \in cats(n_2)} P(t_1|prep)P(t_2|prep)$$

where t_1 and t_2 represent particular concepts from the sets of concepts $cats(n_1)$ and $cats(n_2)$, respectively. These sets come from Roget's thesaurus. She also used a lexical frequency based approach in the above formula using pattern searching from Grolier's encyclopedia. Lapata and Keller (2004) implemented the same lexical based model, but they used Altavista search engine and BNC corpus for computing the probability factors.

Recently, Dima and Hinrichs (2015) proposed a feed-forward neural network based system for noun compound interpretation via labelling. Their network takes concatenated word-vectors of components of the noun compound as input, and predicts one of the labels from the Tratz and Hovy (2010)'s label set. We also adapt their system to our problem setting for a comparison.

3 Why Prepositional Paraphrasing of Noun Compounds?

Uncovering a hidden relation or ellipsis from a construct is an important problem in NLP. For instance, when a customer searches for *15 inch laptop*, he/she is actually searching for *a laptop with 15-inch display*. But, when a customer searches for *30 inch tyres*, he/she is actually searching for *tyres with 60 inch diameter*.

Noun Compounds are one of the many such constructs where the relation is hidden. For example, *"WHO Geneva headquarter"* is *"(the) headquarter of WHO located in Geneva."* One way of explicitly revealing the hidden relation is to paraphrase the noun compound using a verb+preposition construction. For example, *city hospital*: *"hospital located in a/the city"*, *mango juice*: *"juice extracted from mango"*, *etc*.

Prepositions are ambiguous, and they have their own selectional preferences (Zapirain et al., 2013). Once, a preposition and its corresponding sense are known, the space of verb+preposition constructions that can be used for paraphrasing is severely reduced (illustrated in Table 1), thereby making the task of verb+preposition paraphrasing easy. Thus we believe prepositional paraphrasing to be an important step towards verb+preposition paraphrasing of noun compounds.

Noun compound interpretation via labelling is another avenue where prepositional paraphrasing is useful. We annotated Kim and Baldwin (2005)'s dataset with the correct paraphrasing prepositions. A comparison of preposition vs. the annotated semantic relation (Table 2) shows that knowing preposition for a noun compound helps the system in identifying the finer semantic relations. For instance, 29 samples in the annotated data have *from* preposition. Following is the distribution of those samples:

Correct Preposition	Intended Meaning	Possible Verb+Prep Constructions	Noun Compounds	Example Verb+Prep Paraphrases
from	Source	resulted from, produced from, grown in	forest product farm product bonds yield	product produced from forest product grown in farm yield resulted from the bonds
from	Purpose	provided during	pain relief	relief provided during pain

Table 1: Demonstrating how knowledge of prepositions with intended meaning can restrict the space of verb+preposition constructions

	about	at	for	from	in	of	on	with
AGENT	0	0	0	0	0	3	0	0
BENEFICIARY	0	0	4	1	2	3	0	0
CAUSE	0	0	6	3	15	29	3	0
CONTAINER	0	0	1	0	3	12	0	0
CONTENT	0	0	3	2	3	29	1	1
DESTINATION	0	0	0	0	0	2	0	0
EQUATIVE	0	1	4	0	0	1	0	0
INSTRUMENT	0	0	5	0	0	2	0	1
LOCATED	0	0	3	0	0	8	0	0
LOCATION	0	1	2	0	9	17	0	0
MATERIAL	0	0	0	1	0	11	0	0
OBJECT	1	0	7	0	16	55	1	0
POSSESSOR	0	0	8	1	0	21	0	0
PRODUCT	1	0	3	0	0	28	0	0
PROPERTY	3	1	11	0	5	15	1	0
PURPOSE	1	1	89	3	6	41	2	0
RESULT	0	0	3	0	0	2	0	0
SOURCE	1	0	10	13	23	38	3	0
TIME	0	2	4	0	15	3	0	0
TOPIC	9	5	74	5	45	227	8	3

Table 2: Comparison of semantic relations (SR) and prepositions in Kim and Baldwin (2005)'s dataset manually annotated with prepositions. Row labels are SRs and column labels are prepositions. Each entry indicates the number of examples labelled with corresponding semantic relation and preposition.

SOURCE: 13, TOPIC: 5, CAUSE: 3, PURPOSE: 3, CONTENT: 2, BENEFICIARY: 1, MATERIAL: 1, POSSESSOR: 1, and 0 for remaining 12 relations. The annotation frequencies indicate that, for a noun compound with *from* as preposition, SOURCE is a highly likely *e.g.*, *marketing profit*, *forest product*, *tax revenue*, *etc*, whereas the 12 relations with 0 counts are extremely unlikely. Girju et al. (2005) used true prepositions as an additional feature in their classifier for fine semantic relation prediction and reported significant improvement in their performance. Thus, knowledge of preposition will definitely help in predicting and labelling finer relations.

In some NLP tasks, prepositional paraphrasing of a noun compound yields sufficient information for solving the task. Knowledge of Hindi case markers is sufficient for proper Hindi to English translation of Hindi noun compounds (a subset of *samāsa*)(Kulkarni et al., 2012).

4 Approach

As discussed earlier, we learn representations of noun compounds and their prepositional paraphrases, such that the cosine similarity between the representation of a noun compound and the representation of

the corresponding prepositional paraphrase is high. To do this, we use LSTMs (Hochreiter and Schmidhuber, 1997) based encoders. The architecture of our network is shown in Figure 1.

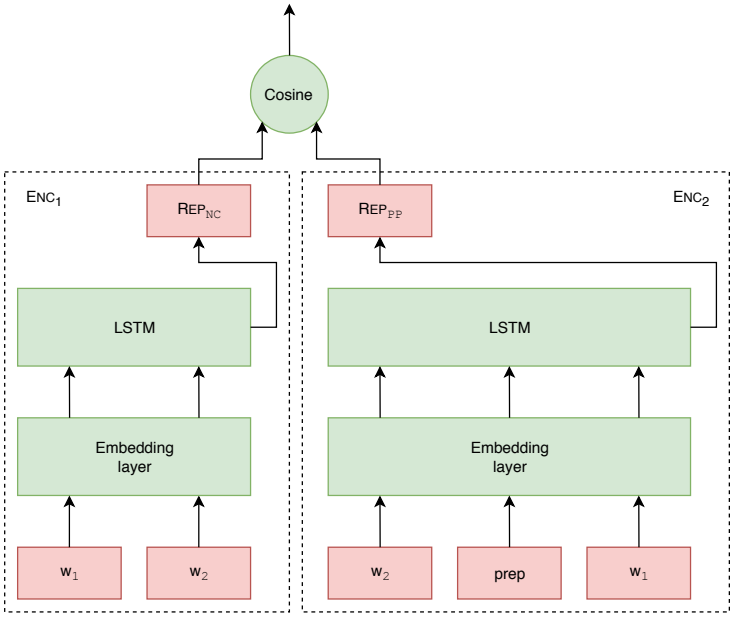


Figure 1: Architecture for learning representations of noun compounds (REP_{NC}) and prepositional paraphrases (REP_{PP}).

The network consists of two encoders: ENC_1 and ENC_2. ENC_1 embeds the constituents of an input noun compound and encodes those embeddings using an LSTM to get a representation REP_{NC} for the noun compound. ENC_2 embeds the words in an input prepositional paraphrase and encodes those embeddings using LSTM to get a representation REP_{PP} for the prepositional paraphrase. The architecture then computes the cosine similarity of these representations, which generates a value in the range $[-1, 1]$. The higher the similarity, the greater is the match between the noun compound and the prepositional paraphrase.

We initialize the embedding layer with Google's pre-trained embeddings[1]. For the 8-prepositions, we use 1-hot representations. We add padding of 0's to make them of same dimensions as that of word embeddings. The embedding layer is not updated during the training.

To calculate the correct prepositional paraphrase for a test noun compound NNC, we use the following procedure:

- Generate representation REP_{NNC} using the encoder ENC_1
- Generate a list of candidate prepositional phrases using the predefined set of prepositions
- For each candidate prepositional paraphrase CPP, generate the corresponding representation REP_{CPP} using the encoder ENC_2.
- Return that prepositional paraphrase PP^* such that cosine similarity between REP_{NNC} and REP_{PP^*} is maximum. Mathematically,

$$\text{PP}^* = \arg\max_{\text{CPP}} \; cosine(\text{REP}_{\text{NNC}}, \text{REP}_{\text{CPP}})$$

[1]Pre-trained embeddings available at `https://code.google.com/archive/p/word2vec/`

5 Experimental Setup

5.1 Datasets

For prepositional paraphrasing of noun compounds, Lauer (1995) and Girju et al. (2005) have proposed datasets. These datasets contain noun compounds annotated with correct preposition.

Lauer (1995)'s dataset is not publicly available. Thus, we extracted test samples from her thesis (Lauer, 1995). We dropped 118 samples out of 400 samples, as they had been marked as non-prepositional, and use the remaining 282 samples for our test experiments. Do note that this dataset is too small for effective learning. We have included it only for the sake of completeness.

Similarly, Girju et al. (2005)'s dataset is not available online[2]. Thus, we use Girju (2007)'s dataset (which has cross-lingual information along with English noun compounds) and extract relevant information to create a dataset. We extracted 832 samples. As our experiments are for Lauer (1995)'s set of 8 prepositions, we dropped examples annotated with prepositions that do not belong to this set. We use the remaining 805 examples for our experiments.

In addition to the above two datasets, we prepared a dataset by manually annotating noun compounds with Lauer (1995)'s prepositions. For annotation, we use 1,779 noun compounds (without labels) from Kim and Baldwin (2005)'s dataset. Each example was annotated by two annotators. The inter-annotator agreement is 51.48% (Cohen's kappa κ =0.36). For experiments, we have used only 1042 examples[3] where both annotators agreed upon. From here on, we refer to this dataset as Inhouse dataset.

In order to make learning more effective, we needed a lot more training samples. We adapted Lapata and Keller (2004)'s approach which use the web to automatically annotate noun compounds. We, however, use Netspeak[4]. We collected a list of noun compounds from existing noun compound interpretation datasets (Kim and Baldwin, 2005; Ó Séaghdha, 2007; Tratz and Hovy, 2010) and query each of them with a set of prepositions (as shown in Figure 2). The preposition with the highest frequency is chosen to form the *correct* prepositional paraphrase. For example, as shown in Figure 2, '*expert in (the) analysis*' has the highest frequency among all candidate paraphrases of *analysis expert*. So, we choose *in* as a correct preposition for the noun compound *analysis expert*. This dataset is used for training all systems.

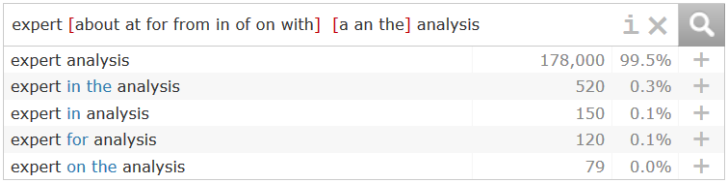


Figure 2: Example query result using Netspeak

5.2 Training

To train our system, we have noun compounds with their preposition paraphrases. Our objective is to learn representations such that similarity of a noun compound representation is higher with correct preposition compared with its similarity with any other prepositional paraphrase. Thus, for each noun compound, we treat the remaining prepositions as negative examples. For example, correct preposition for *analysis expert* is *in*. So, we treat (*analysis expert, 'expert in analysis'*) as a positive example and other pairs – like (*analysis expert, 'expert from analysis'*), (*analysis expert, 'expert about analysis'*), (*analysis expert, 'expert at analysis'*), *etc.* – as negative examples. We create such examples from the training dataset mentioned above.

We perform two different set of experiments. In one kind of experiments, we train our system (explained in Section 4) on the automatically annotated dataset and evaluate performance on the various datasets. In another type, for each dataset, we additionally fine-tune the trained-model using a portion of

[2]We requested the authors for the Girju et al. (2005)'s dataset, but they did not respond positively.

[3]The dataset is available at `http://www.cfilt.iitb.ac.in/nc-dataset`

[4]A search engine on Web 1T 5-gram available at `http://www.netspeak.org`.

the dataset. We use 75% of the dataset for tuning, and rest 25% of examples for testing. In both cases, we use the same test-set for a fair comparison. We discuss the performance of the models in the next section.

5.3 Evaluation metrics

We report Precision, Recall and F-score for our experiments. These values are weighted values in proportion to a number of test-examples for each preposition. For instance, following is a formula for computing (weighted) precision:

$$\text{Precision} = \sum_{prep} P_{prep} * \frac{N_{prep}}{N}; \quad P_{prep} = \frac{TP_{prep}}{TP_{prep} + FP_{prep}}$$

where P_{prep} is the precision score, TP_{prep} is the number of true-positives and FP_p is the number of false-positives for a preposition $prep$. N_{prep} is the number of instances with preposition $prep$ in the test set, and N is the total number of instances in a test-set.

To compare the results with previous work, we report micro-averaged accuracy[5]. The following formula computes micro-accuracy:

$$\text{Accuracy} = \frac{\text{Number of correcly classified instances}}{\text{Total number of instances in test-test}}$$

6 Results and Analysis

We compare the performance of our approach (referred to as NC-LSTM hereafter) with the performance reported by Lauer (1995), Lapata and Keller (2004), and Girju (2007) and performance computed for Dima and Hinrichs (2015)'s approach (referred to as NC-FFN hereafter) on different datasets.

We first compare NC-LSTM with NC-FFN. Table 3 shows that NC-LSTM performs comparably, if not better, than NC-FNN. Also, NC-LSTM easily outperforms NC-FFN by a significant margin when the network parameters are further tuned with a portion of the dataset. Thus, this also shows the importance of fine-tuning. This is easily explained by the fact that the original training data was extracted from the web, and is noisy in nature.

Dataset	Approach	Without Tuning			With Tuning		
		P	R	F	P	R	F
Lauer (1995)	NC-FFN	40.85	38.03	31.15	43.97	40.85	40.09
	NC-LSTM	50.84	45.07	40.66	48.72	46.48	**46.21**
Girju (2007)	NC-FFN	74.72	80.69	77.52	74.20	86.14	79.72
	NC-LSTM	76.86	74.26	75.50	84.74	88.61	**85.13**
Inhouse Dataset	NC-FFN	63.00	66.96	63.97	64.91	67.39	64.40
	NC-LSTM	62.32	65.65	63.09	73.50	72.17	**71.27**

Table 3: Comparison of performance of our LSTM based architecture (NC-LSTM) with Dima and Hinrichs (2015)'s feed-forward neural network based architecture (NC-FNN) on different datasets (P: Precision; R: Recall, F: F-score)

To demonstrate the effect of noisy data, we provide an analysis of the noun compound *apple tree*. The correct paraphrase of this noun compound is *'tree of apple'*. However, the system considers the paraphrase *'tree with apple'* to be correct. The following example sentences, which contributed to the count of *'tree with apple'*, demonstrate some errors made by the system:

[5]Mathematically, micro-averaged accuracy and weighted recall are same. We reported accuracy so that it can be compared with prior work. We reported F-score as it is the standard metric for classification evaluation. We reported recall as it used for computing f-score.

1. *if you combine a pine **tree with an apple** tree you do indeed get a pineapple tree.*
2. *What do you get if you cross a Christmas **tree with an apple?***

Clearly, the implicit assumption that *apple* is attached to the first occurrence of *tree* made by automatic web-based annotation algorithm which created the training dataset in sentence 1 is incorrect. Similarly, the implicit assumption that *apple* is attached to *tree* in sentence 2 is incorrect. Such errors add noise to the training dataset. However, using the tuning set fixes such errors, thereby improving performance considerably.

To check the extent to which such errors can impact effective training of the system, we estimate the accuracy of the automatic annotation process. We do this by testing the accuracy of automatic annotation on the common examples between automatically annotated and each of the three manually annotated datasets. The statistics of the common examples are shown in Table 4.

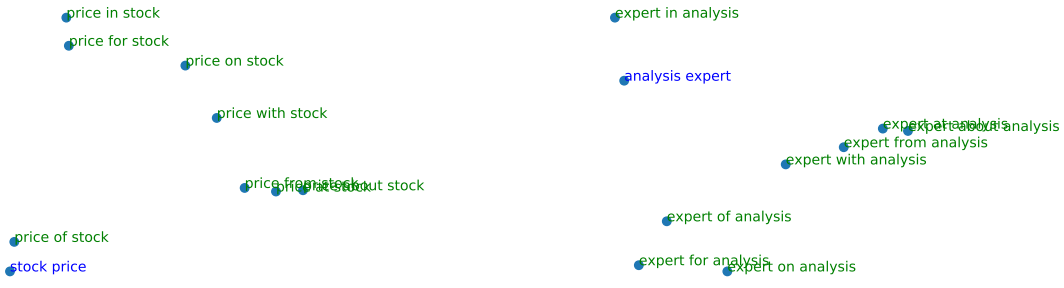
Dataset	**#of Common Examples**	**#of Common Examples with Matching Labels**
Lauer (1995)	31	12
Girju et al. (2005)	9	8
Inhouse Dataset	434	317

Table 4: Statistics of common examples between automatically created dataset and each of the three gold-standard dataset.

Table 5 compares reported results of various approaches with results from our approach. It shows that NC-LSTM outperforms Girju (2007)'s approach on their own dataset. We are unable to perform well on Lauer (1995) dataset. However, as mentioned earlier, given the relatively small size of the data, we refrain from commenting about our system's performance on the basis of this result.

Approach	**L95**	**G05**	**Inhouse**
Lauer (1995)	40.00		
Lapata and Keller (2004)	**55.71**		
Girju et al. (2005)	46.09	56.22	
NC-FFN	40.85	86.14	67.39
NC-LSTM	46.48	**88.61**	**72.17**

Table 5: Accuracy of various system reported in the literature. (L95: Lauer (1995); G05: Girju et al. (2005); Inhouse: Our manually created Inhouse dataset)



(a) *stock price* (b) *analysis expert*

Figure 3: PCA visualizations of noun compounds and their prepositional paraphrases for some examples

Figure 3 shows visualizations of noun compounds and their candidate prepositional paraphrases. One can observe from Figure 3a that among all candidate paraphrases, the embedding of *'price of stock'* is the closest to the embedding of *stock price*. Similarly, Figure 3b shows that among all candidate paraphrases, the embedding of *'expert in analysis'* is the closest to the embedding of *analysis expert*.

7 Conclusion and Future Work

In this paper, we proposed a novel way to perform prepositional paraphrasing of noun compounds. We learn representations for noun compounds and their corresponding prepositional paraphrases using LSTMs, such that cosine similarity of a noun compound and a prepositional paraphrase correlates with their semantic similarity. Our proposed method, when trained on noun compound data automatically annotated using the web, performs reasonably well on existing datasets. In the future, we would investigate adapting this approach for verb+preposition paraphrasing, *i.e.* given a noun compound *orange juice*, generate the verb+preposition paraphrase *'juice made from orange'*.

References

Kisuh Ahn, Johan Bos, David Kor, Malvina Nissim, Bonnie L Webber, and James R Curran. 2005. Question answering with QED at TREC 2005. In *TREC*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR 2015)*.

Timothy Baldwin and Takaaki Tanaka. 2004. Translation by machine of complex nominals: Getting it right. In *Proceedings of the Workshop on Multiword Expressions: Integrating Processing*, pages 24–31.

Renu Balyan and Niladri Chatterjee. 2015. Translating noun compounds using semantic relations. *Computer Speech & Language*, 32(1):91–108.

Ken Barker and Stan Szpakowicz. 1998. Semi-automatic recognition of noun modifier relationships. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 96–102.

Cristina Butnariu, Su Nam Kim, Preslav Nakov, Diarmuid Ó Séaghdha, Stan Szpakowicz, and Tony Veale. 2009. Semeval-2010 task 9: The interpretation of noun compounds using paraphrasing verbs and prepositions. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 100–105.

Corina Dima and Erhard Hinrichs. 2015. Automatic noun compound interpretation using deep neural networks and word embeddings. *IWCS 2015*, pages 173–183.

Roxana Girju, Adriana Badulescu, and Dan Moldovan. 2003. Learning semantic constraints for the automatic discovery of part-whole relations. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 1–8.

Roxana Girju, Dan Moldovan, Marta Tatu, and Daniel Antohe. 2005. On the semantics of noun compounds. *Computer speech & language*, 19(4):479–496.

Roxana Girju. 2007. Improving the interpretation of noun phrases with cross-linguistic information. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, volume 45, page 568.

Iris Hendrickx, Preslav Nakov, Stan Szpakowicz, Zornitsa Kozareva, Diarmuid O Séaghdha, and Tony Veale. 2013. Semeval-2013 task 4: Free paraphrases of noun compounds. *Atlanta, Georgia, USA*, page 138.

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

Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, and Mark Carman. 2016. Are word embedding-based features useful for sarcasm detection? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1006–1011. Association for Computational Linguistics.

Su Nam Kim and Timothy Baldwin. 2005. Automatic interpretation of noun compounds using wordnet similarity. In *Natural Language Processing–IJCNLP 2005*, pages 945–956. Springer.

Amba Kulkarni, Soma Paul, Malhar Kulkarni, Anil Kumar, and Nitesh Surtani. 2012. Semantic processing of compounds in indian languages. In *COLING*, pages 1489–1502.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June. Association for Computational Linguistics.

Mirella Lapata and Frank Keller. 2004. The web as a baseline: Evaluating the performance of unsupervised web-based models for a range of nlp tasks. In *the Proceedings of the Human Language Technology Conference/North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*.

Mark Lauer. 1995. *Designing statistical language learners: Experiments on compound nouns*. Ph.D. thesis, Ph. D. thesis, Macquarie University.

Judith N Levi. 1978. *The syntax and semantics of complex nominals*. Academic Press New York.

Preslav Nakov. 2013. On the interpretation of noun compounds: Syntax, semantics, and entailment. *Natural Language Engineering*, 19(03):291–330.

Diarmuid Ó Séaghdha. 2007. Annotating and learning compound noun semantics. In *Proceedings of the 45th Annual Meeting of the ACL: Student Research Workshop*, pages 73–78.

Girishkumar Ponkiya, Kevin Patel, Pushpak Bhattacharyya, and Girish K Palshikar. 2018. Towards a standardized dataset for noun compound interpretation. In *Language Resources and Evaluation Conference*, Miyazaki, Japan.

Barbara Rosario, Marti A Hearst, and Charles Fillmore. 2002. The descent of hierarchy, and selection in relational semantics. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 247–254.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the ACL*.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533.

Duyu Tang, Furu Wei, Bing Qin, Ting Liu, and Ming Zhou. 2014. Coooolll: A deep learning system for twitter sentiment classification. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 208–212.

Stephen Tratz and Eduard Hovy. 2010. A taxonomy, dataset, and classifier for automatic noun compound interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 678–687.

Lucy Vanderwende. 1994. Algorithm for automatic interpretation of noun sequences. In *Proceedings of the 15th conference on Computational linguistics-Volume 2*, pages 782–788.

Beatrice Warren. 1978. Semantic patterns of noun-noun compounds. *Acta Universitatis Gothoburgensis. Gothenburg Studies in English Goteborg*, 41:1–266.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Beñat Zapirain, Eneko Agirre, Lluís Màrquez, and Mihai Surdeanu. 2013. Selectional preferences for semantic role classification. *Computational Linguistics*, 39(3).

CASCADE: Contextual Sarcasm Detection in Online Discussion Forums

Devamanyu Hazarika
School of Computing,
National University of Singapore
hazarika@comp.nus.edu.sg

Soujanya Poria
Artificial Intelligence Initiative,
A*STAR, Singapore
sporia@ihpc.a-star.edu.sg

Sruthi Gorantla
Computer Science & Automation,
Indian Institute of Science, Bangalore
gorantlas@iisc.ac.in

Erik Cambria
School of Computer Science and
Engineering, NTU, Singapore
cambria@ntu.edu.sg

Roger Zimmermann
School of Computing,
National University of Singapore
rogerz@comp.nus.edu.sg

Rada Mihalcea
Computer Science & Engineering,
University of Michigan, Ann Arbor
mihalcea@umich.edu

Abstract

The literature in automated sarcasm detection has mainly focused on lexical-, syntactic- and semantic-level analysis of text. However, a sarcastic sentence can be expressed with contextual presumptions, background and commonsense knowledge. In this paper, we propose a Context-tuAl SarCasm DEtector (CASCADE), which adopts a hybrid approach of both content- and context-driven modeling for sarcasm detection in online social media discussions. For the latter, CASCADE aims at extracting contextual information from the discourse of a discussion thread. Also, since the sarcastic nature and form of expression can vary from person to person, CASCADE utilizes user embeddings that encode stylometric and personality features of users. When used along with content-based feature extractors such as convolutional neural networks, we see a significant boost in the classification performance on a large Reddit corpus.

1 Introduction

Sarcasm is a linguistic tool that uses irony to express contempt. Its figurative nature poses a great challenge for affective systems performing sentiment analysis (Cambria et al., 2017). Previous research in automated sarcasm detection has primarily focused on lexical and pragmatic cues found in sentences (Kreuz and Caucci, 2007). In the literature, interjections, punctuations, and sentimental shifts have been considered as major indicators of sarcasm (Joshi et al., 2017). When such lexical cues are present in sentences, sarcasm detection can achieve high accuracy. However, sarcasm is also expressed implicitly, i.e., without the presence of such lexical cues. This use of sarcasm also relies on context, which involves the presumption of commonsense and background knowledge of an event. When it comes to detecting sarcasm in a discussion forum, it may not only be required to understand the context of previous comments but also the necessary background knowledge about the topic of discussion. The usage of slangs and informal language also diminishes the reliance on lexical cues (Satapathy et al., 2017). This particular type of sarcasm is tough to detect (Poria et al., 2016).

Contextual dependencies for sarcasm can take many forms. As an example, a sarcastic post from Reddit[1], *"I'm sure Hillary would've done that, lmao."* requires background knowledge about the event, i.e., Hillary Clinton's action at the time the post was made. Similarly, sarcastic posts like *"But atheism, yeah *that's* a religion!"* requires the knowledge that topics like *atheism* often contain argumentative discussions and, hence, they are more prone towards sarcasm.

The main aim of this work is sarcasm detection in online discussion forums. In particular, we propose a hybrid network, named CASCADE, that leverages both the *content* and the *context* required for sarcasm detection. It starts by processing contextual information in two ways. First, it performs user profiling to create user embeddings that capture indicative behavioral traits for sarcasm. Recent findings suggest that such modeling of the user and their preferences is highly effective for the given task (Amir et al.,

[1] http://reddit.com

Proceedings of the 27th International Conference on Computational Linguistics, pages 1837–1848
Santa Fe, New Mexico, USA, August 20-26, 2018.

2016). It makes use of users' historical posts to model their writing style (stylometry) and personality indicators, which are then fused into comprehensive user embeddings using a multi-view fusion approach, termed canonical correlation analysis (CCA) (Hotelling, 1936). Second, it extracts contextual information from the discourse of comments in the discussion forums. This is done by document modeling of these consolidated comments belonging to the same forum. We hypothesize that these discourse features would give the important contextual information, background cues along with topical information required for detecting sarcasm.

After the contextual modeling phase, CASCADE is provided with a comment for sarcasm detection. It performs content-modeling using a convolutional neural network (CNN) to extract its syntactic features. This CNN representation is then concatenated with the relevant user embedding and discourse features to get the final representation which is used for classification. The overall contribution of this work can be summarized as:

- We propose a novel hybrid sarcasm detector, CASCADE, that models both content and contextual information.
- We model stylometric and personality details of users along with discourse features of discussion forums to learn informative contextual representations. Experiments on a large Reddit corpus demonstrate significant performance improvement over state-of-the-art automated sarcasm detectors.

The remainder of the paper is organized as follows: Section 2 lists related works; Section 3 explains the process of learning contextual features comprising user embeddings and discourse features; Section 4 presents experimentation details of the model and result analysis; finally, Section 5 draws conclusions.

2 Related Work

Automated sarcasm detection is a relatively recent field of research. Previous works can be classified into two main categories: content- and context-based sarcasm detection models.

Content-based models: These networks model the problem of sarcasm detection as a standard classification task and try to find lexical and pragmatic indicators to identify sarcasm. Numerous works have taken this path and presented innovative ways to unearth interesting cues for sarcasm. Tepperman et al. (2006) investigate sarcasm detection in spoken dialogue systems using prosodic and spectral cues. Carvalho et al. (2009) use linguistic features like positive predicates, interjections and gestural clues such as emoticons, quotation marks, etc. Davidov et al. (2010), Tsur et al. (2010) use syntactic patterns to construct classifiers. González-Ibánez et al. (2011) also study the use of emoticons, mainly amongst tweets. Riloff et al. (2013) assert sarcasm to be a contrast to positive sentiment words and negative situations. Joshi et al. (2015) use multiple features comprising lexical, pragmatics, implicit and explicit context incongruity. In the explicit case, they include relevant features to detect thwarted sentimental expectations in the sentence. For implicit incongruity, they generalize Riloff et al. (2013) by identifying verb-noun phrases containing contrast in both polarities.

Context-based models: The usage of contextual sarcasm has increased in recent years, especially in online platforms. Texts found in microblogs, discussion forums, and social media are plagued by grammatical inaccuracies and contain information which is highly temporal and contextual. In such scenarios, mining linguistic information becomes relatively inefficient and the need arises for additional clues (Carvalho et al., 2009). Wallace et al. (2014) demonstrate this need by showing how traditional classifiers fail in instances where humans require additional context. They also indicate the importance of speaker and topical information associated to a text to gather such context. Poria et al. (2016) use additional information by sentiment, emotional and personality representations of the input text. Previous works have mainly used historical posts of users to understand sarcastic tendencies (Rajadesingan et al., 2015; Zhang et al., 2016). Khattri et al. (2015) try to discover users' sentiments towards entities in their histories to find contrasting evidence. Wallace et al. (2015) utilize sentiments and noun phrases used within a forum to gather context typical to that forum. Such forum-based modeling simulates user

communities. Our work follows a similar motivation as we explore the context provided by user profiling and the topical knowledge embedded in the discourse of comments in discussion forums (subreddits[2]).

Amir et al. (2016) performed user modeling by learning embeddings that capture homophily. This work is the closest to our approach given the fact that we too learn user embeddings to acquire context. However, we take a different approach that involves stylometric and personality description of the users. Empirical evidence shows that these proposed features are better than previous user modeling approaches. Moreover, we learn discourse features which has not been explored before in the context of this task.

3 Method

3.1 Task Definition

The task involves detection of sarcasm for comments made in online discussion forums, i.e., Reddit. Let us denote the set $U = \{u_1, ..., u_{N_u}\}$ for N_u-users, where each user participates across a subset of N_t-discussion forums (subreddits). For a comment C_{ij} made by the i^{th} user u_i in the j^{th} discussion forum t_j, the objective is to predict whether the comment posted is sarcastic or not.

3.2 Summary of the Proposed Approach

Given the comment C_{ij} to be classified, CASCADE leverages *content*- and *context*-based information from the comment. For content-based modeling of C_{ij}, a CNN is used to generate the representation vector $\vec{\boldsymbol{c}}_{i,j}$ for a comment. CNNs generate abstract representations of text by extracting location-invariant local patterns. This vector $\vec{\boldsymbol{c}}_{i,j}$ captures both syntactic and semantic information useful for the task at hand. For contextual modeling, CASCADE first learns user embeddings and discourse features of all users and discussion forums, respectively (Section 3.3). Following this phase, CASCADE then retrieves the learnt user embedding $\vec{\boldsymbol{u}}_i$ of user u_i and discourse feature vector $\vec{\boldsymbol{t}}_j$ of forum t_j. Finally, all three vectors $\vec{\boldsymbol{c}}_{i,j}$, $\vec{\boldsymbol{u}}_i$, and $\vec{\boldsymbol{t}}_j$ are concatenated and used for the classification (Section 3.6). One might argue that, instead of using one CNN, we could use multiple CNNs as in (Majumder et al., 2017), to get better text representations whenever a comment contains multiple sentences. However, that is out of the scope of this work. Here, we aim to show the effectiveness of user-specific analysis and context-based features extracted from the discourse. Also, the use of a single CNN for text representation helps to consistently compare our model with the state of the art.

3.3 Learning Contextual Features

In this section, we explain in detail the procedures to generate the contextual features, i.e., user embeddings and discourse features. The user embeddings try to capture users' traits that correlate to their sarcastic tendencies. These embeddings are created considering the accumulated historical posts of each user (Section 3.4). Contextual information are also extracted from the discourse of comments within each discussion forum. These extracted features are named as discourse features (Section 3.5). The aim of learning these contextual features is to acquire discriminative information crucial for sarcasm detection.

3.4 User Embeddings

To generate user embeddings, we model their stylometric and personality features and then fuse them using CCA to create a single representation. Below, we explain the generation of user embedding $\vec{\boldsymbol{u}}_i$, for the i^{th} user u_i. Figure 1 also summarizes the overall architecture for this kind of user profiling.

3.4.1 Stylometric features

People possess their own idiolect and authorship styles, which is reflected in their writings. These styles are generally affected by attributes such as gender, diction, syntactic influences, etc. (Cheng et al., 2011; Stamatatos, 2009) and present behavioral patterns which aid sarcasm detection (Rajadesingan et al., 2015).

We use this motivation to learn stylometric features of the users by consolidating their online comments into documents. We first gather all the comments by a user and create a document by appending them using a special delimiter `<END>`. An unsupervised representation learning method *ParagraphVector* (Le

[2] `http://reddit.com/reddits`

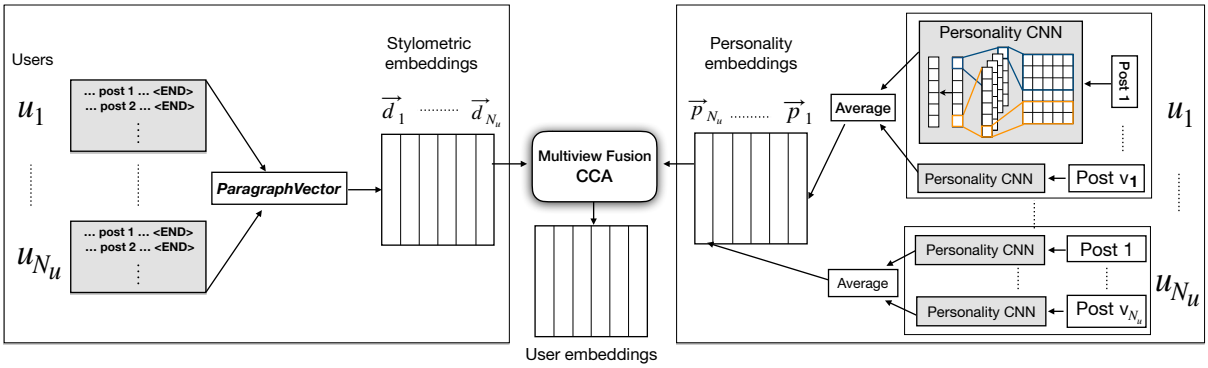


Figure 1: The figure describes the process of user profiling. Stylometric and personality embeddings are generated and then fused in a multi-view setting using CCA to get the user embeddings.

and Mikolov, 2014) is then applied on this document. This method generates a fixed-sized vector for each user by performing the auxiliary task of predicting the words within the documents. The choice of *ParagraphVector* is governed by multiple reasons. Apart from its ability to effectively encode a user's writing style, it has the advantage of applying to variable lengths of text. *ParagraphVector* also has been shown to perform well for sentiment classification tasks. The existence of synergy between sentiment and sarcastic orientation of a sentence also promotes the use of this method.

We now describe the functioning of this method. Every user document and all words within them are first mapped to unique vectors such that each vector is represented by a column in matrix $D \in \mathbb{R}^{d_s \times N_u}$ and $W_s \in \mathbb{R}^{d_s \times |V|}$, respectively. Here, d_s is the embedding size and $|V|$ represents the size of the vocabulary. *Continuous bag-of-words* approach (Mikolov et al., 2013) is then performed where a target word is predicted given the word vectors from its context window. The key idea here is to use the document vector of the associated document as part of the context words. More formally, given a user document d_i for user u_i comprising a sequence of n_i-words $\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_{n_i}$, we calculate the average log probability of predicting each word within a sliding context window of size k_s. This average log probability is:

$$\frac{1}{n_i} \sum_{t=k_s}^{n_i - k_s} \log p(w_t | d_i, w_{t-k_s}, ..., w_{t+k_s}) \tag{1}$$

To predict a word within a window, we take the average of all the neighboring context word vectors along with the document vector $\vec{\boldsymbol{d}}_i$ and use a neural network with softmax prediction:

$$p(w_t | d_i, w_{t-k_s}, ..., w_{t+k_s}) = \frac{e^{\vec{\boldsymbol{y}}_{w_t}}}{\sum_i e^{\vec{\boldsymbol{y}}_i}} \tag{2}$$

Here, $\vec{\boldsymbol{y}} = [y_1, ..., y_{|V|}]$ is the output of the neural network, i.e.,

$$\vec{\boldsymbol{y}} = \mathrm{U}_d h(\vec{\boldsymbol{d}}_i, \vec{\boldsymbol{w}}_{t-k_s}, ..., \vec{\boldsymbol{w}}_{t+k_s}; D, W_s) + \vec{\boldsymbol{b}}_d \tag{3}$$

$\vec{\boldsymbol{b}}_d \in \mathbb{R}^{|V|}, U_d \in \mathbb{R}^{|V| \times d_s}$ are parameters and $h(\cdot)$ represents the average of vectors $\vec{\boldsymbol{d}}_i, \vec{\boldsymbol{w}}_{t-k_s}, ..., \vec{\boldsymbol{w}}_{t+k_s}$ taken from D and W_s. Hierarchical softmax is used for faster training (Morin and Bengio, 2005). Finally, after training, D learns the users' document vectors which represent their stylometric features.

3.4.2 Personality features

Discovering personality from text has numerous natural language processing (NLP) applications such as product recognition, mental health diagnosis, etc. Described as a combination of multiple characteristics, personality detection helps in identifying behavior, thought patterns of an individual. To model the dependencies of users' personality with their sarcastic nature, we include personality features in the user embeddings. Previously, Poria et al. (2016) also utilized personality features in sentences. However, we take a different approach of extracting the personality features of a user instead.

For user u_i, we iterate over all the v_i-comments $\{S^1_{u_i}, ..., S^{v_i}_{u_i}\}$ written by them. For each $S^j_{u_i}$, we provide the comment as an input to a pre-trained CNN which has been trained on a multi-label personality detection task. Specifically, the CNN is pre-trained on a benchmark corpus developed by Matthews and Gilliland (1999) which contains 2400 essays and is labeled with the Big-Five personality traits, i.e., Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism (OCEAN). After the training, this CNN model is used to infer the personality traits present in each comment. This is done by extracting the activations of the CNN's last hidden layer vector, which we call as the personality vector $\vec{\boldsymbol{p}}^j_{u_i}$. The expectation over the personality vectors for all v_i-comments made by the user is then defined as the overall personality feature vector $\vec{\boldsymbol{p}}_i$ of user u_i:

$$\vec{\boldsymbol{p}}_i \;=\; \mathbb{E}_{j\in[v_i]}[\vec{\boldsymbol{p}}^j_{u_i}] \;=\; \frac{1}{v_i}\sum_{j=1}^{v_i}\vec{\boldsymbol{p}}^j_{u_i} \tag{4}$$

CNN: Here, we describe the CNN that generates the personality vectors. Given a user's comment, which is a text $S = [w_1, ..., w_n]$ composed of n words, each word w_i is represented as a word embedding $\vec{\boldsymbol{w}}_i \in \mathbb{R}^{d_{em}}$ using the pre-trained FastText embeddings (Bojanowski et al., 2016). A single-layered CNN is then modeled on this input sequence S (Kim, 2014). First, a convolutional layer is applied having three filters $F_{[1,2,3]} \in \mathbb{R}^{d_{em}\times h_{[1,2,3]}}$ of heights $h_{[1,2,3]}$, respectively. For each $k \in \{1, 2, 3\}$, filter F_k slides across S and extracts h_k-gram features at each instance. This creates a feature map vector $\vec{\boldsymbol{m}}_k$ of size $\mathbb{R}^{|S|-h_k+1}$, whose each entry $m_{k,j}$ is obtained as:

$$m_{k,j} = \alpha(\; F_k \cdot S_{[j:j+h_k-1]} + b_k \;) \tag{5}$$

here, $b_k \in \mathbb{R}$ is the bias and $\alpha(\cdot)$ is a non-linear activation function.

M feature maps are created from each filter F_k giving a total of $3M$ feature maps as output. Following this, a max-pooling operation is performed across the length of each feature map. Thus, for all M feature maps computed from F_k, output $\vec{\boldsymbol{o}}_{\boldsymbol{k}}$ is calculated as, $\vec{\boldsymbol{o}}_{\boldsymbol{k}} = [\; max(\vec{\boldsymbol{m}}^1_1), ..., \; max(\vec{\boldsymbol{m}}^{\mathrm{M}}_1) \;]$. Overall the max-pooling output is calculated by concatenation of each $\vec{\boldsymbol{o}}_{\boldsymbol{k}}$ to get $\vec{\boldsymbol{o}} = [\vec{\boldsymbol{o}}_{\boldsymbol{1}} \oplus \vec{\boldsymbol{o}}_{\boldsymbol{2}} \oplus \vec{\boldsymbol{o}}_{\boldsymbol{3}}] \in \mathbb{R}^{3M}$, where $\oplus$ represents concatenation. Finally, $\vec{\boldsymbol{o}}$ is projected onto a dense layer with d_p neurons followed by the final sigmoid-prediction layer with 5 classes denoting the five personality traits (Matthews et al., 2003). We use sigmoid instead of softmax to facilitate multi-label classification. This is calculated as:

$$\vec{\boldsymbol{q}} = \alpha(\; W_1\vec{\boldsymbol{o}} + \vec{\boldsymbol{b}}_1 \;) \tag{6}$$

$$\hat{\boldsymbol{y}} = \sigma(\; W_2\vec{\boldsymbol{q}} + \vec{\boldsymbol{b}}_2 \;) \tag{7}$$

$W_1 \in \mathbb{R}^{d_p\times 3M}, W_2 \in \mathbb{R}^{5\times d_p}, \vec{\boldsymbol{b}}_1 \in \mathbb{R}^{d_p}$ and $\vec{\boldsymbol{b}}_2 \in \mathbb{R}^5$ are parameters and $\alpha(.)$ represents non-linear activation.

3.4.3 Fusion

We take a multi-view learning approach to combine both stylometric and personality features into a comprehensive embedding for each user. We use CCA to perform this fusion. CCA captures maximal information between two views and creates a combined representation (Hardoon et al., 2004; Benton et al., 2016). In the event of having more than two views, fusion can be performed using an extension of CCA called *Generalized* CCA (see Appendix).

Canonical Correlation Analysis: Let us consider the learnt stylometric embedding matrix $D \in \mathbb{R}^{d_s\times N_u}$ and personality embedding matrix $P \in \mathbb{R}^{d_p\times N_u}$ containing the respective embedding vectors of user u_i in their i^{th} columns. The matrices are then mean-centered and standardized across all user columns. We call these new matrices as X_1 and X_2, respectively. Let the correlation matrix for X_1 be $R_{11} = X_1{X_1}^T \in \mathbb{R}^{d_s\times d_s}$, for X_2 be $R_{22} = X_2{X_2}^T \in \mathbb{R}^{d_p\times d_p}$ and the cross-correlation matrix between them be $R_{12} = X_1{X_2}^T \in \mathbb{R}^{d_s\times d_p}$. For each user u_i, the objective of CCA is to find the linear projections of both embedding vectors that have a maximum correlation. We create K such projections, i.e., K-canonical variate pairs such that each pair of projection is orthogonal with respect to the previous pairs. This is done by constructing:

$$W = X_1^T A_1 \text{ and } Z = X_2^T A_2 \tag{8}$$

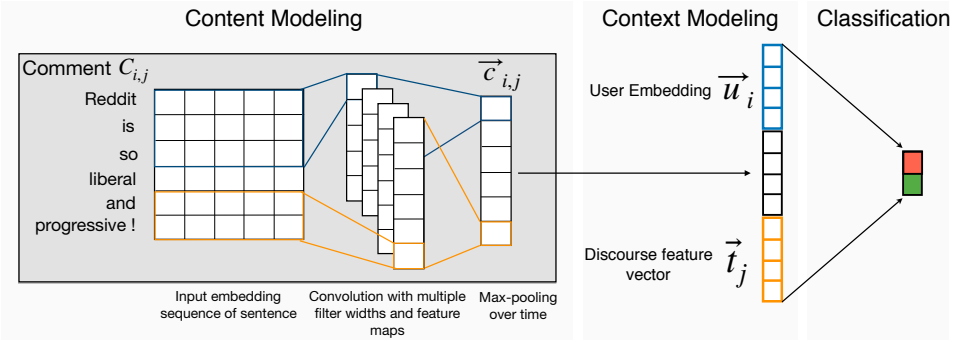


Figure 2: Overall hybrid network of CASCADE. For the comment $C_{i,j}$, its content-based sentential representation $\vec{c}_{i,j}$ is extracted using a CNN and appended with context vectors $\vec{u}_i$ and $\vec{t}_j$.

where, $A_1 \in \mathbb{R}^{d_s \times K}$, $A_2 \in \mathbb{R}^{d_p \times K}$ and $W^T W = Z^T Z = I$. To maximize correlation between W and Z, optimal A_1 and A_2 are calculated by performing singular value decomposition as:

$$R_{11}^{-\frac{1}{2}} R_{12} R_{22}^{-\frac{1}{2}} = A \Lambda B^\top \quad \text{, where} \quad A_1 = R_{11}^{-\frac{1}{2}} A \text{ and } A_2 = R_{22}^{-\frac{1}{2}} B \tag{9}$$

It can be seen that,

$$W^T W = {A_1}^T R_{11} A_1 = A^T A = I \quad \text{and} \quad Z^T Z = {A_2}^T R_{22} A_2 = B^T B = I \tag{10}$$

$$\text{also,} \quad W^T Z = Z^T W = \Lambda \tag{11}$$

Once optimal A_1 and A_2 are calculated, overall user embedding $\vec{u}_i \in \mathbb{R}^K$ of user u_i is generated by fusion of $\vec{d}_i$ and $\vec{p}_i$ as:

$$\vec{u}_i = (\vec{d}_i)^T A_1 + (\vec{p}_i)^T A_2 \tag{12}$$

3.5 Discourse Features

Similarly to how a user influences the degree of sarcasm in a comment, we assume that the discourse of comments belonging to a certain discussion forum contain contextual information relevant to the sarcasm classification. They embed topical information that selectively incur bias towards degree of sarcasm in the comments of a discussion. For example, comments on political leaders or sports matches are generally more susceptible to sarcasm than natural disasters. Contextual information extracted from the discourse of a discussion can also provide background knowledge or cues about the topic of that discussion.

To extract the discourse features, we take a similar approach of document modeling performed for stylometric features (Section 3.4.1). For all N_t-discussion forums, we compose each forum's document by appending the comments within them. As before, *ParagraphVector* is employed to generate discourse representations for each document. We denote the learnt feature vector of j^{th} forum t_j as $\vec{t}_j \in \mathbb{R}^{d_t}$.

3.6 Final Prediction

Following the extraction of text representation $\vec{c}_{i,j}$ for comment $C_{i,j}$ and retrieval of user embedding $\vec{u}_i$ for author u_i and discourse feature vector $\vec{t}_j$ for discussion forum t_j, we concatenate all three vectors to form the unified text representation $\hat{c}_{i,j} = [\vec{c}_{i,j} \oplus \vec{u}_i \oplus \vec{t}_j]$. Here, $\oplus$ refers to concatenation. The CNN used for extraction of $\vec{c}_{i,j}$ has the same design as the CNN we used to extract personality features described in Section 3.4.2. Finally, $\hat{c}_{i,j}$ is projected to the output layer having two neurons with a softmax activation. This gives a softmax-probability over whether a comment is sarcastic or not. This probability estimate is then used to calculate the categorical cross-entropy which is used as the loss function:

$$Loss = \frac{-1}{N} \sum_{i=1}^{N} \sum_{j=1}^{2} \boldsymbol{y}_{i,j} \, log_2(\hat{\boldsymbol{y}}_{i,j}) \quad \text{, where} \quad \hat{\boldsymbol{y}} = softmax(W_o \hat{\boldsymbol{c}}_{i,j} + \vec{\boldsymbol{b}}_o) \tag{13}$$

Here, N is the number of comments in the training set, y_i is the one-hot vector ground truth of the i^{th} comment and $\hat{\boldsymbol{y}}_{i,j}$ is its predicted probability of belonging to class j.

4 Experimental Results

4.1 Dataset

We perform our experiments on a large-scale self-annotated corpus for sarcasm, SARC[3] (Khodak et al., 2017). This dataset contains more than a million examples of sarcastic/non-sarcastic statements made on Reddit. Reddit comprises of topic-specific discussion forums, also known as subreddits, each titled by a post. In each forum, users communicate either by commenting to the titled post or other's comments, resulting in a tree-like conversation structure. This structure can be unraveled to a linear format, thus creating a discourse of the comments by keeping the topological constraints intact. Each comment is accompanied with its author details and parent comments (if any) which is subsequently used for our contextual processing. It is important to note that almost all comments in SARC are composed of a single sentence. We consider three variants of the SARC dataset in our experiments.

- **Main balanced:** This is the primary dataset which contains a balanced distribution of both sarcastic and non-sarcastic comments. The dataset contains comments from 1246058 users (118940 in training and 56118 in testing set) distributed across 6534 forums (3868 in training and 2666 in testing set).
- **Main imbalanced:** To emulate real-world scenarios where the sarcastic comments are typically fewer than non-sarcastic ones, we use an imbalanced version of the Main dataset. Specifically, we maintain a $20:80$ ratio (approx.) between the sarcastic and non-sarcastic comments in both training/testing sets.
- **Pol:** To further test the effectiveness of our user embeddings, we perform experiments on a subset of Main, comprising of forums associated with the topic of politics. Table 1 provides the comment distribution of all the dataset variants mentioned.

		Training set				Testing set			
		no. of comments		avg. no. of words per comment		no. of comments		avg. no. of words per comment	
		non-sarc	*sarc*	*non-sarc*	*sarc*	*non-sarc*	*sarc*	*non-sarc*	*sarc*
Main	balanced	77351	77351	55.13	55.08	32333	32333	55.55	55.01
	imbalanced	77351	25784	55.13	55.21	32333	10778	55.55	55.48
Pol	balanced	6834	6834	64.74	62.36	1703	1703	62.99	62.14

*non-sarc: non-sarcastic, sarc: sarcastic

Table 1: Details of comments in SARC.

The choice of using SARC for our experiments comes with multiple reasons. First, this corpus is the first of its kind that was purposely developed to investigate the necessity of contextual information in sarcasm classification. This characteristic aligns well with the main goal of this paper. Second, the large size of the corpus allows for statistically-relevant analyses. Third, the dataset annotations contain a small false-positive rate for sarcastic labels thus providing reliable annotations. Also, its self-annotation scheme rules out the annotation errors induced by third-party annotators. Finally, the corpus structure provides meta-data (e.g., user information) for its comments, which is useful for contextual modeling.

4.2 Training details

We hold out 10% of the training data for validation. Hyper-parameter tuning is performed using this validation set through RandomSearch (Bergstra and Bengio, 2012). To optimize the parameters, Adam optimizer (Kingma and Ba, 2014) is used, starting with an initial learning rate of $1e^{-4}$. The learnable parameters in the network consists of $\theta = \{U_d, D, W_{[1,2,o,s]}, F_{[1,2,3]}, \vec{\mathbf{b}}_{[1,2,o,d]}, b_{[1,2,3]}\}$. Training termination is decided using early stopping technique with a patience of 12. For the batched-modeling of comments in CNNs, each comment is either restricted or padded to 100 words for uniformity. The optimal hyper-parameters are found to be $\{d_s, d_p, d_t, K\} = 100$, $d_{em} = 300$, $k_s = 2$, $M = 128$, and $\alpha = ReLU$.

We manually analyze the effect in validation performance for different sizes of user-embedding dimension K (Figure 3a) and discourse feature vector size d_t (Figure 3b). In both cases, the performance trend suggests the optimal size to be approximately 100.

[3] http://nlp.cs.princeton.edu/SARC

Models	Main				Pol	
	balanced		imbalanced			
	Accuracy	*F1*	*Accuracy*	*F1*	*Accuracy*	*F1*
Bag-of-words	0.63	0.64	0.68	0.76	0.59	0.60
CNN	0.65	0.66	0.69	0.78	0.62	0.63
CNN-SVM (Poria et al., 2016)	0.68	0.68	0.69	0.79	0.65	0.67
CUE-CNN (Amir et al., 2016)	0.70	0.69	0.73	0.81	0.69	0.70
CASCADE (no personality features)	0.68	0.66	0.71	0.80	0.68	0.70
CASCADE	**0.77**†	**0.77**†	**0.79**†	**0.86**†	**0.74**†	**0.75**†
Δ_{SOTA}	$\uparrow$ 7%	$\uparrow$ 8%	$\uparrow$ 6%	$\uparrow$ 5%	$\uparrow$ 5%	$\uparrow$ 5%

†:significantly better than CUE-CNN (Amir et al., 2016).

Table 2: Comparison of CASCADE with state-of-the-art networks and baselines on multiple versions of the SARC dataset. We assert significance when $p < 0.05$ under paired-t test. Results comprise of 10 runs with different initializations. The bottom row shows the absolute difference with respect to the CUE-CNN system.

For modeling the *ParagraphVector*, we use the open-sourced implementation provided by *Gensim*[4]. The CNNs used in the model are implemented using Tensorflow library[5].

4.3 Baseline Models

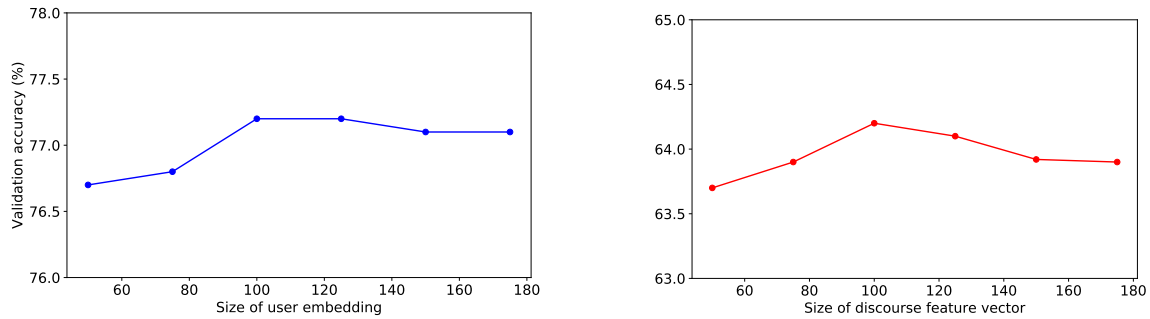
Here, we describe the state-of-the-art methods and baselines that we compare CASCADE with.

- **Bag-of-words:** This model uses an SVM classifier whose input features comprise of a comment's word-counts. The size of the vector is the vocabulary size of the training dataset.
- **CNN:** We compare our model with this individual CNN version. This CNN is capable of modeling only the *content* of a comment. The architecture is similar to the CNN used in CASCADE (see Section 3.2).
- **CNN-SVM:** This model proposed by Poria et al. (2016) consists of a CNN for content modeling and other pre-trained CNNs for extracting sentiment, emotion and personality features from the given comment. All the features are concatenated and fed into an SVM for classification.
- **CUE-CNN:** This method proposed by Amir et al. (2016) also models user embeddings with a method akin to *ParagraphVector*. Their embeddings are then combined with a CNN thus forming the CUE-CNN model. We compare with this model to analyze the efficiency of our embeddings as opposed to theirs. Released software[6] is used to produce results on the SARC dataset.

4.4 Results

Table 2 presents the performance results on SARC. CASCADE manages to achieve major improvement across all datasets with statistical significance. The lowest performance is obtained by the bag-of-words approach whereas all neural architectures outperform it. Amongst the neural networks, the CNN baseline

[4] http://radimrehurek.com/gensim/models/doc2vec.html
[5] http://github.com/dennybritz/cnn-text-classification-tf
[6] http://github.com/samiroid/CUE-CNN



(a) CASCADE with only user embeddings.

(b) CASCADE with only discourse features.

Figure 3: Exploration of dimensions for user embedding and discourse feature vector.

receives the least performance. CASCADE comfortably beats the state-of-the-art neural models CNN-SVM and CUE-CNN. Its improved performance on the Main imbalanced dataset also reflects its robustness towards class imbalance and establishes it as a real-world deployable network.

We further compare our proposed user-profiling method with that of CUE-CNN, with absolute differences shown in the bottom row of Table 2. Since CUE-CNN generates its user embeddings using a method similar to the *ParagraphVector*, we test the importance of personality features being included in our user profiling. As seen in the table, CASCADE without personality features drops in performance to a range similar to CUE-CNN. This suggests that the combination of stylometric and personality features are indeed crucial for the improved performance of CASCADE.

4.5 Ablation Study

We experiment on multiple variants of CASCADE so as to analyze the importance of the various features present in its architecture. Table 3 provides the results of all the combinations. First, we test performance for the *content*-based CNN only (row 1). This setting provides the worst relative performance with almost 10% lower accuracy than optimal. Next, we include contextual features to this network. Here, the effect of discourse features is primarily seen in the Pol dataset getting an increase of 3% in F1 (row 2). A major boost in performance is observed ($8 - 12\%$ accuracy and F1) when user embeddings are introduced (row 5). Visualization of the user embedding cluster (Section 4.6) provides insights for this positive trend. Overall, CASCADE consisting of CNN with user embeddings and contextual discourse features provides the best performance in all three datasets (row 6).

We challenge the use of CCA for the generation of user embeddings and, hence, replace it with simple concatenation. This, however, causes a significant drop in performance (row 3). Improvement is not observed even when discourse features are used with these concatenated user embeddings (row 4). We assume the increase in parameters caused by concatenation for this performance degradation. CCA, on the other hand, creates succinct representations with maximal information, giving better results.

4.6 User Embedding Analysis

We investigate the learnt user embeddings in more detail. In particular, we plot random samples of users on a 2D-plane using t-SNE (Maaten and Hinton, 2008). The users who have greater sarcastic comments (atleast 2 more than the other type) are termed as sarcastic users (colored red). Conversely, the users having fewer sarcastic comments are called non-sarcastic users (colored green). Equal number of users from both the categories are plotted. We aim to analyze the reason behind the performance boost provided by the user embeddings as shown in Table 3. We see in Figure 4 that both the user types belong to similar distributions. However, the sarcastic users have a greater spread than the non-sarcastic ones (red belt around the green region). This is also evident from the variances of the distributions where the sarcastic distribution comprises of 10.92 variance as opposed to 5.20 variance of the non-sarcastic distribution. From this observation, we can infer that the user embeddings belonging to this non-overlapping red-region provide discriminative information regarding the sarcastic tendencies of their users.

	CASCADE			Main				Pol	
	user		dis-	balanced		imbalanced			
	cca	concat.	course	*Acc.*	*F1*	*Acc.*	*F1*	*Acc.*	*F1*
1.	-	-	-	0.65	0.66	0.69	0.78	0.62	0.63
2.	-	-	✓	0.66	0.66	0.68	0.78	0.63	0.66
3.	-	✓	-	0.66	0.66	0.69	0.79	0.62	0.61
4.	-	✓	✓	0.65	0.67	0.71	0.85	0.63	0.66
5.	✓	-	-	0.77	0.76	**0.80**	**0.86**	0.70	0.70
6.	✓	-	✓	**0.78**	**0.77**	0.79	**0.86**	**0.74**	**0.75**

Table 3: Comparison with variants of the proposed CASCADE network. All combinations use content-based CNN

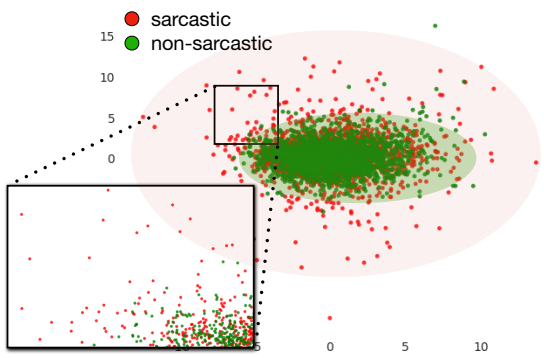


Figure 4: 2D-Scatterplot of the user embeddings visualized using t-SNE (Maaten and Hinton, 2008).

4.7 Case Studies

Results demonstrate that discourse features provide an improvement over baselines, especially on the Pol dataset. This signifies the greater role of the contextual cues for classifying comments in this dataset over the other dataset variants used in our experiment. Below, we present a couple of cases from the Pol dataset where our model correctly identifies the sarcasm which is evident only with the neighboring comments. The previous state-of-the-art CUE-CNN, however, misclassifies them.

- For the comment *Whew, I feel much better now!*, its sarcasm is evident only when its previous comment is seen *So all of the US presidents are terrorists for the last 5 years.*
- The comment *The part where Obama signed it.* doesn't seem to be sarcastic until looked upon as a remark to its previous comment *What part of this would be unconstitutional?*.

Such observations indicate the impact of discourse features. However, sometimes contextual cues from the previous comments are not enough and misclassifications are observed due to lack of necessary commonsense and background knowledge about the topic of discussion. There are also other cases where our model fails despite the presence of contextual information from the previous comments. During exploration, this is primarily observed for contextual comments which are very long. Thus, sequential discourse modeling using RNNs may be better suited for such cases. Also, in the case of user embeddings, misclassifications were common for users with fewer historical posts. In such scenarios, potential solutions would be to create user networks and derive information from similar users within the network, e.g., by means of community embeddings (Cavallari et al., 2017). These are some of the issues which we plan to address in future work.

5 Conclusion

In this paper, we introduced CASCADE, a Contextual Sarcasm Detector, which leverages both content and contextual information for the classification. For contextual details, we perform user profiling along with discourse modeling from comments in discussion threads. When this information is used jointly with a CNN-based textual model, we obtain state-of-the-art performance on a large-scale Reddit corpus. Our results show that discourse features along with user embeddings play a crucial role in the performance of sarcasm detection.

References

Silvio Amir, Byron C Wallace, Hao Lyu, and Paula Carvalho Mário J Silva. 2016. Modelling context with user embeddings for sarcasm detection in social media. *arXiv preprint arXiv:1607.00976*.

Adrian Benton, Raman Arora, and Mark Dredze. 2016. Learning multiview embeddings of twitter users. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 14–19.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Erik Cambria, Soujanya Poria, Alexander Gelbukh, and Mike Thelwall. 2017. Sentiment analysis is a big suitcase. *IEEE Intelligent Systems*, 32(6):74–80.

Paula Carvalho, Luís Sarmento, Mário J Silva, and Eugénio De Oliveira. 2009. Clues for detecting irony in user-generated contents: oh...!! it's so easy;-. In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 53–56. ACM.

Sandro Cavallari, Vincent Zheng, Hongyun Cai, Kevin Chang, and Erik Cambria. 2017. Learning community embedding with community detection and node embedding on graphs. In *CIKM*, pages 377–386.

Na Cheng, Rajarathnam Chandramouli, and KP Subbalakshmi. 2011. Author gender identification from text. *Digital Investigation*, 8(1):78–88.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the fourteenth conference on computational natural language learning*, pages 107–116. Association for Computational Linguistics.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers-Volume 2*, pages 581–586. Association for Computational Linguistics.

David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. 2004. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664.

Harold Hotelling. 1936. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 757–762.

Aditya Joshi, Pushpak Bhattacharyya, and Mark J Carman. 2017. Automatic sarcasm detection: A survey. *ACM Computing Surveys (CSUR)*, 50(5):73.

Anupam Khattri, Aditya Joshi, Pushpak Bhattacharyya, and Mark Carman. 2015. Your sentiment precedes you: Using an author's historical tweets to predict sarcasm. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 25–30.

Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. 2017. A large self-annotated corpus for sarcasm. *arXiv preprint arXiv:1704.05579*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Roger J Kreuz and Gina M Caucci. 2007. Lexical influences on the perception of sarcasm. In *Proceedings of the Workshop on computational approaches to Figurative Language*, pages 1–4. Association for Computational Linguistics.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Navonil Majumder, Soujanya Poria, Alexander Gelbukh, and Erik Cambria. 2017. Deep learning-based document modeling for personality detection from text. *IEEE Intelligent Systems*, 32(2):74–79.

Gerald Matthews and Kirby Gilliland. 1999. The personality theories of hj eysenck and ja gray: A comparative review. *Personality and Individual differences*, 26(4):583–626.

Gerald Matthews, Ian J Deary, and Martha C Whiteman. 2003. *Personality traits*. Cambridge University Press.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252. Citeseer.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, and Prateek Vij. 2016. A deeper look into sarcastic tweets using deep convolutional neural networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1601–1612.

Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. 2015. Sarcasm detection on twitter: A behavioral modeling approach. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 97–106. ACM.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 704–714.

Ranjan Satapathy, Claudia Guerreiro, Iti Chaturvedi, and Erik Cambria. 2017. Phonetic-based microtext normalization for twitter sentiment analysis. In *ICDM*, pages 407–413.

Efstathios Stamatatos. 2009. A survey of modern authorship attribution methods. *Journal of the Association for Information Science and Technology*, 60(3):538–556.

Joseph Tepperman, David Traum, and Shrikanth Narayanan. 2006. " yeah right": Sarcasm recognition for spoken dialogue systems. In *Ninth International Conference on Spoken Language Processing*.

Oren Tsur, Dmitry Davidov, and Ari Rappoport. 2010. Icwsm-a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *ICWSM*, pages 162–169.

Michel van de Velden. 2011. On generalized canonical correlation analysis. In *Proceedings of the 58th World Statistical Congress*.

Byron C Wallace, Laura Kertz, Eugene Charniak, et al. 2014. Humans require context to infer ironic intent (so computers probably do, too). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 512–516.

Byron C Wallace, Eugene Charniak, et al. 2015. Sparse, contextually informed models for irony detection: Exploiting user communities, entities and sentiment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1035–1044.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Tweet sarcasm detection using deep neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2449–2460.

A Generalized Canonical Correlation Analysis

For user profiling with more than two views, we can use *Generalized* CCA (GCCA) as the multiview-fusion approach. In GCCA, the input data consists of I different views, $X_i \in \mathbb{R}^{d_i \times N} \quad \forall\ i \in [1, I]$, where, N is the total number of data points and d_i is the dimension of the ith view. Also, X_i represent the mean centered matrix of the data. We find a common representation $G \in \mathbb{R}^{N \times K}$ for all the input points. The *canonical covariates* $\vec{\boldsymbol{w}}_{\boldsymbol{i}} = X_i^T \vec{\boldsymbol{a}}_{\boldsymbol{i}}$ are chosen in such a way that the sum of the squared correlations between them and the group configuration is maximum:

$$\max R^2 = \sum_{i=1}^{N} r(\vec{\boldsymbol{g}}, X_i^T \vec{\boldsymbol{a}}_{\boldsymbol{i}})^2 \quad \text{s.t. } \vec{\boldsymbol{g}}^T \vec{\boldsymbol{g}} = 1 \tag{14}$$

For K-canonical variate pairs, the GCCA objective function can be formulated as follows:

$$\underset{G, A_i}{\operatorname{argmax}} \left\| G - X_i^T A_i \right\|_F^2 \quad \text{s.t. } G^T G = I \tag{15}$$

where $A_i \in \mathbb{R}^{d_i \times K}$. G can be obtained using the eigen equation:

$$(\sum_{i=1}^{N} P_i) G = G\Gamma \quad \text{, where,} \quad P_i = X_i^T (X_i X_i^T)^{-1} X_i \tag{16}$$

The matrices A_i can then be calculated as:

$$A_i = (X_i X_i^T)^{-1} X_i^T G \tag{17}$$

It is to be noted that GCCA with two views is equivalent to CCA (van de Velden, 2011).

Recognizing Humour using Word Associations and Humour Anchor Extraction

Andrew Cattle **Xiaojuan Ma**
Hong Kong University of Science and Technology
Department of Computer Science and Engineering
Clear Water Bay, Hong Kong
{acattle,mxj}@cse.ust.hk

Abstract

This paper attempts to marry the interpretability of statistical machine learning approaches with the more robust models of joke structure and joke semantics capable of being learned by neural models. Specifically, we explore the use of semantic relatedness features based on word associations, rather than the more common Word2Vec similarity, on a binary humour identification task and identify several factors that make word associations a better fit for humour. We also explore the effects of using joke structure, in the form of humour anchors (Yang et al., 2015), for improving the performance of semantic features and show that, while an intriguing idea, humour anchors contain several pitfalls that can hurt performance.

1 Introduction

Humour is a part of everyday communication. Although telling and understanding jokes comes naturally to most humans, the recognition and interpretation of humour is a very difficult task for computers. Beyond merely expressing a funny idea, great jokes choose evocative words and present them in a surprising structure (Attardo, 2008) often with pleasing phonetics (Mihalcea and Strapparava, 2005). In addition to this linguistic mastery, many jokes require world knowledge either by setting up and then subverting an expectation or by referencing a common piece of culture (Kukovačec et al., 2017). This makes computational humour a challenging yet very intriguing problem for natural language processing (NLP). As such, it is no surprise that computational humour, and humour recognition in particular, has received increased attention from the NLP community with SemEval-2017 devoting two tasks to it: ranking humorous tweets (Potash et al., 2017) and interpreting English puns (Miller et al., 2017).

This recent attention has lead to advancements such as sequence-based neural humour models capable of implicitly learning a joke structure and semantic features (Bertero and Fung, 2016a; Donahue et al., 2017). While these approaches offer good performance, their reliance on complicated neural architectures over explicitly engineered features present a problem for interpretability which may make it difficult to diagnose problems if results go wrong.

Works which take a more interpretable statistical machine learning approach have their own drawbacks. For example, the representation of joke semantics has been fairly basic, typically computing word embedding similarities between all word pairs in a document (Yang et al., 2015), and bear little resemblance to the way humans actually interpret humour. Additionally, many works fail to take advantage of joke structure, treating texts as unordered bags-of-words (Bertero and Fung, 2016b; Mihalcea and Strapparava, 2005; Yan and Pedersen, 2017).

This paper aims to marry the interpretability of statistical machine learning approaches with the more nuanced models of joke structure and joke semantics of neural approaches. Specifically, we explore the effectiveness modelling joke semantics using semantic relatedness features based on word associations, rather than the more common semantic similarity features based on word embeddings. We present evidence not only that relatedness in general is better suited than similarity for computational humour tasks

Proceedings of the 27th International Conference on Computational Linguistics, pages 1849–1858
Santa Fe, New Mexico, USA, August 20-26, 2018.

due to its broader nature, but also that word associations are particularly well suited due to their ability to map more nuanced relationships (De Deyne and Storms, 2008) and asymmetric nature. While such features have been explored in the past (Cattle and Ma, 2016; Cattle and Ma, 2017b), this work presents a more in depth analysis, focusing on a more fundamental task (binary humour classification vs. relative humour ranking) on with a dataset that better represents natural language (oneliners and puns vs. Twitter hashtag games), and is the first work to incorporate interpolated word association strengths. Furthermore, we introduce a novel method for targetting our semantic features using joke structure to help reduce noise and increase reliability. Specifically, we experiment with integrating the extraction of humour anchors, the "meaningful, complete, minimal set of word spans" (Yang et al., 2015) that allow humour to occur, into the humour classification process itself, the first work to do so. We are also making the code used to run these experiments publicly available[1].

2 Related Work

Informally, jokes are generally divided into setups and punchlines, with the setup establishing a context and the punchline delivering the humour. More formally, according to the Semantic Script Theory of Humor (SSTH) introduced in Raskin (1985), humour is derived from the resolution of two overlapping, but incongruent scripts of differing levels of obviousness (Labutov and Lipson, 2012). That is to say, humour comes from causing listeners to make assumptions about a situation (i.e. implying the obvious script) and then subverting them (i.e. forcing a re-evaluation which results in the choosing of the less obvious one). Consider the following joke taken from Raskin (1985):

> "Is the doctor at home?" the patient asked in his bronchial whisper. "No," the doctor's young and pretty wife whispered in reply. "Come right in."

The use of the words *doctor*, *patient*, and *bronchial* in the setup lead the listener to assume that the man is seeking medical advice. However, the punchline, the doctor's wife's response along with her description, reveals the man's true intent.

The General Theory of Verbal Humor (GTVH), introduced in Attardo and Raskin (1991), expanded on SSTH, retaining the notion of script opposition but highlighting other factors which also affect humour. These include "target (i.e., the person or group made fun of in the joke), the situation (i.e., the background assumed by the joke, such as the props for the story), the narrative strategy (the 'genre' of the joke), and the language (i.e., the lexical, syntactic choices of the text)." (Attardo, 2008)

While target, situation, and the overlapping but incongruent nature of the scripts speak to the importance of semantics in humour, narrative strategy, language, and the fact that listeners should pick the more obvious script first speak to the importance of structure. Although these two aspects are not entirely independent, for the purposes of this paper, it is useful to consider them separately.

2.1 Joke Semantics

Following SSTH, we expect that punchlines that do not sufficiently overlap with their setup are unfunny as they do not flow logically from the context. Similarly, punchlines that are not sufficiently incongruent with their setup are unfunny as there is no re-evaluation. As overlap and incongruity are difficult to measure directly, one common approach is instead to use word embeddings, such as Word2Vec (Mikolov et al., 2013), to calculate the cosine similarities between pairs of vectors representing words in a document (Yang et al., 2015; Shahaf et al., 2015; Kukovačec et al., 2017). However, measuring incongruity and overlap in terms of similarity is a rather odd choice. Just because two scripts overlap does not imply they are similar. In the "doctor" example in Section 2, the two scripts, namely *[the patient is seeking medical advice]* and *[the patient is having an affair with the doctor's wife]*, overlap in terms of the people and locations involved but otherwise are quite different. Similarly, incongruent scripts such as *[dog bites man]* and *[man bites dog]* are very similar in all but the assignment of the roles.

Compared with *similarity*, *relatedness* is a much broader concept. It is easy to think of concepts that are related but not similar. For example, "beer" and "glass" are not similar, describing very different

[1] https://github.com/acattle/HumourTools

concepts, but are quite closely related in that beer often comes in glasses (Ma, 2013). However, it is difficult to think of two concepts which are similar but not related. Although similarity is much more common, several semantic relatedness measures do exist. Extended Lesk (Banerjee and Pedersen, 2003), for example, compute the relatedness between two words as a function of the size of the overlap of their glosses. However, such measures do not fully address the short comings of word embedding similarities.

Any measure based on a distributional semantic approach, including both Extended Lesk and Word2Vec, at its core relies on word co-occurrence to extract relationships. However, not all relationships are evidenced by co-occurrence. For example, "yellow" and "banana" are so closely related that "yellow banana" occurs relatively infrequently since bananas are assumed to be yellow unless told otherwise (De Deyne et al., 2016).

Word associations capture relationships between concepts, are not reliant on distributional semantics, and have long been used in psychology and cognitive science dating back to Galton (1879). Furthermore, some word association-based metrics have been shown to perform better than Word2Vec on a series of similarity tasks (De Deyne et al., 2016). Although the collection of word association datasets is much more labour intensive than distributional semantic approaches, several word association datasets do exist, the most popular being the Edinburgh Associative Thesaurus (EAT), collected in Kiss et al. (1973), and the University of South Florida Free Association Norms (USF), collected in Nelson et al. (2004). Both these datasets were compiled by presenting participants with a cue word and asking them to respond with the first word that comes to mind. The proportion of respondents who give a specific response for a specific cue is called the *forward strength* from the cue to the target. This approach does have its drawbacks, forward strengths are relative instead of absolute (Nelson et al., 2004) and accepting only a single response causes weaker relationships to be under-represented (De Deyne and Storms, 2008), but it is a straightforward and widely accepted way of collecting word association data.

This not the first work to propose word associations for humour recognition. Cattle and Ma (2016) and its follow up, Cattle and Ma (2017b), previously explored the effectiveness of word association strengths on a relative humour ranking task with encouraging results. Most notably, they examined the role of word association asymmetry. While the cosine similarity function preferred by vector space models like Word2Vec and TFIDF is symmetrical, word associations are not. Someone shown the word "beer" may be likely to say "glass" but someone shown the word "glass" may be unlikely to say "beer" (Ma, 2013). This allows for finer grained exploration of the relationship between setups and punchlines. In fact, Cattle and Ma (2016) shows evidence that punchlines tend to be funnier if the strength from the setup to the punchline, which they claim represents how obvious a punchline is, is weaker than the strength of the punchline back to the setup, which they claim represents how easy a punchline is to understand.

While our approach is similar to that of Cattle and Ma (2017b), we note several key differences. First, they perform a relative humour ranking task as opposed to our binary humour classification task. As relative ranking is a more difficult task in general (Potash et al., 2017), Cattle and Ma (2017b)'s results were far from convincing. By pursuing the easier binary classification task, we intend to present a clearer evaluation of word association features for humour recognition. Second, Cattle and Ma (2017b) focused on Twitter hashtag games, a type of online game where participants submit their best responses to a central topic or theme. Compared to the oneliners and puns explored in this work, not only are hashtag war entries slightly shorter, but hashtag wars also include a clear and explicitly marked setup in the form of the central hashtag (Cattle and Ma, 2016). Third, Cattle and Ma (2017b) exclusively utilize a graph-based method for extracting association strengths. This led to coverage issues due to the relatively small vocabularies of the word association datasets. We address this issue by using an machine learning-based approach capable of predicting strengths between arbitrary word pairs, similar to the one introduced in (Cattle and Ma, 2017a).

The association strength prediction method introduced in Cattle and Ma (2017a) is in turn very similar to the WordNet Evocation (Boyd-Graber et al., 2006) prediction method introduced in Hayashi (2016). In all cases, association strengths are estimated using a variety of WordNet (Miller and Fellbaum, 1998) and vector space features which are fed into a multi-layer perceptron.

2.2 Joke Structure

Early humour generation systems tended to model jokes' setup/punchline structure explicitly. HAHAcronyms (Stock and Strapparava, 2002) takes well-known acronyms as setups and, using listener's preconceived notions of that acronym's canonical expansion, generates novel expansions as punchlines which present an ironic clash. Petrović and Matthews (2013) generate jokes of the form "I like my *X* like I like my *Y*, *Z*," where the punchline *Z* acts as a link between the setups *X* and *Y*. Labutov and Lipson (2012) invokes SSTH directly, generating setups by mixing compatible elements of two overlapping but incongruent scripts of varying degrees of obviousness, generated using the common-sense knowledge-base ConceptNet (Speer and Havasi, 2012), and punchlines which introduce information compatible with only the second, less obvious script.

The punning riddle generators, such as Binsted and Ritchie (1994), present an rather unique version of setups and punchlines. Considering the example "What do you call a *sour assistant*? A *lemon aide*." (Binsted, 1996), the setup, instead of establishing an expectation to be subverted, actually establishes a context in which the somewhat odd reading of *lemon aide* can be preferred over the much more common and phonetically identical reading *lemonade*. However, it should be noted that clearly defined setups and punchlines still exist.

Given humour generation's firm grasp of punchline/setup structure, it is somewhat surprising that humour recognition has largely ignored it despite humour recognition starting in earnest with Mihalcea and Strapparava (2005), a full decade after early humour generation works such as Binsted and Ritchie (1994). Early humour recognition works starting with Mihalcea and Strapparava (2005) and its follow-up Mihalcea and Pulman (2007), up to more recent works such as Radev et al. (2016) and Yan and Pedersen (2017) employ bag-of-words models. Such systems are unable to capture document-level structure, such as that setups tend to precede punchlines.

As mentioned in Section 1, joke structure and joke semantics are not entirely independent. This means poor models of joke structure can affect the performance of features designed to capture joke semantics. Despite the issues mentioned in Section 2.1, Yang et al. (2015)'s "incongruity" feature set, maximum and minimum word embedding similarities between pairs of words in a document, perform fairly well. Cattle and Ma (2017b) takes a similar approach with their word association features. The problems comes from the fact that both works compute these values across *all* possible pairs of words in a document. This can introduce noise as not all word pairs are meaningful (e.g. pairs of stopwords) and internally-cohesive setups and punchlines can bias maximum similarity scores. While this can be somewhat alleviated by judicial filtering of stopwords, this does not guarantee meaningful word pairs either.

Yang et al. (2015), in addition to their humour classifier, also introduces a method for identifying jokes' humour anchors (HAs), the "meaningful, complete, minimal set of word spans" that allow humour to occur. While this is slightly different from identifying a joke's setup and punchline, focusing only on pairs of HAs would help reducing noise by increasing the precision of meaningful word pairs selection without sacrificing recall. However, Yang et al. (2015) does not use their extracted HAs to improve their humour classification performance. Likely this is due to the fact that their proposed HA extraction method requires a separate, fully trained humour prediction model which is robust to word order. Furthermore, the quality of the extracted HAs is directly tied to the quality of the humour model. However, both issues could have been alleviated using some form of co-training or bootstrapping between the overall humour prediction model and the HA extractor's internal humour prediction model. The HA extractor works by generating a list of HA candidates for each document following a heuristic method. After removing various combinations of HA candidates from the original document, these modified documents are fed into a trained humour prediction model, with the HAs being the combination of HA candidates which causes the largest drop in humour score. Since the humour prediction model is by design robust against word order, words can be freely omitted with few side effects.

It should be noted that sequence-based humour models such as Bertero and Fung (2016b) or Donahue et al. (2017) should theoretically be capable of implicitly learning HAs, especially Bertero and Fung (2016a)'s Long Short-Term Memory-based approach. However, these models are much more complex than Yang et al. (2015)'s approach, require more training data, and suffer from a lack of interpretability.

3 Methodology

3.1 Datasets

We evaluate our classifiers across two separate datasets: Pun of the Day (PotD), collected in Yang et al. (2015), and 16000 One-Liner (OL), collected in Mihalcea and Strapparava (2005). PotD consists of positive examples collected from the Pun of the Day website[2] and negative examples collected from a combination of news sources, question/answer forums, and lists of proverbs (Yang et al., 2015). OL consists of positive examples scraped from humour websites and negative examples taken from a combination of new headlines, sentences from the British National Corpus, and proverbs.

3.2 Baseline

For our baseline we implemented our own version of Yang et al. (2015)'s highest performing classifier. This model was chosen due to its high performance and its use of statistical machine learning techniques which make it a fair point of comparison. Features include, for each document, minimum and maximum Word2Vec similarities between all word pairs, the total number of word sense combinations in each document according to WordNet, minimum and maximum WordNet path similarities between all word pairs, number of words with negative/positive polarity as well as weak/strong subjectivities according to the Wilson et al. (2005) sentiment lexicon, number of and length of longest alliteration and rhyme chains according to the CMU Pronouncing Dictionary[3], labels of the five nearest neighbours in the training set according to word frequencies, and an averaged Word2Vec embedding across all words for a total of 318 feature dimensions. We then train a Random Forest Classifier using the scikit-learn[4] Python library with 100 estimators but otherwise default settings.

All Word2Vec features, including those described below, use Google's pre-trained 300 dimension Word2Vec embeddings[5].

3.3 Semantic Features

Similar to Yang et al. (2015), we compute the minimum, maximum, and average Word2Vec similarity between ordered word pairs. Not only is this a common humour recognition feature (Yang et al., 2015; Shahaf et al., 2015; Kukovačec et al., 2017), but it also acts as a point of comparison for word association strength.

For our word association features we compute the minimum, maximum, and average association strength between ordered word pairs, which we refer to this as the *forward* strength. Since, as described in Section 2.1, word associations are directional, we also compute the minimum, maximum, and average associations strengths between the reverse ordered word pairs, which we refer to as the *backward* strength. Following Cattle and Ma (2016), we also compute the difference between these two values on both a micro (per word) and macro (per document) level. We refer to these sets of features as the *diff* strengths. Forward, backward, and diff strengths are extracted for both the EAT and USF word association datasets.

We also compare two methods for computing our word association strength features. The first method uses the graph-based method described in Cattle and Ma (2016). We refer to this approach as *graph*. The second uses an in-house association strength predictor using an machine learning-based method similar to the one described in Cattle and Ma (2017a). We refer to this approach as *ML*. Unlike Cattle and Ma (2017a), which uses three different types of word embeddings, we use only Word2Vec. In addition to Word2Vec similarity and vector offsets (the difference between the cue and response vectors), we also include LDA similarity (300 dimensions, trained using Gensim[6] on English Wikipedia), AutoExtend (Rothe and Schütze, 2015) similarity, and a variety of WordNet-based features described in Cattle and Ma (2017a).

[2] http://punoftheday.com/
[3] http://www.speech.cs.cmu.edu/cgi-bin/cmudict
[4] http://scikit-learn.org/
[5] https://code.google.com/archive/p/word2vec/
[6] https://radimrehurek.com/gensim/

3.4 Humour Classifier

In addition to the semantic features described above, we also calculate each document's perplexity according to a 3gram language model (LM) trained on the WMT15 English news discussion corpus (Bojar et al., 2015) using KenLM[7]. This feature is included not only because document perplexity has been shown to be useful for humour recognition (Shahaf et al., 2015) but also to act as a baseline for our semantic features. Because both the datasets described in Section 3.1 draw negative examples from news sources, we were concerned that training an LM on a general English corpus might unfairly bias the perplexity scores against those negative examples. Similarly, training an LM on a news corpus might unfairly bias perplexity towards those negative examples. Therefore, the news discussion corpus was chosen as a happy medium between news vocabulary and informal writing styles.

As with our baseline, our extracted features are used to train a Random Forest Classifier using scikit-learn with 100 estimators and default settings using a 10 fold cross validation.

3.5 Humour Anchors

By default, word association and Word2Vec features are computed across all word pairs in a document. However, we also experiment computing these features only across pairs of humour anchor (HA) words. HAs are extracted using the method described in Yang et al. (2015) using the same baseline humour model described in Section 3.2 for anchor candidate evaluation.

HA extraction's requirement of a fully trained anchor candidate evaluator raises the problem of what data that evaluator should be trained on. Given that, as described in Section 3.1, we experiment on two separate humour datasets, we train the anchor candidate evaluator on the opposite dataset from the overall humour classifier. This is to avoid overfitting or biasing the anchor candidate evaluator by training on the test data.

4 Results and Discussion

The results of our experiments are reported in Table 1. In general, our model performs slightly worse than the Yang et al. (2015) baseline. One interesting aspect to note is that our model uses only 28 feature dimensions compared to Yang et al. (2015)'s 318. While this is not exactly a fair comparison in the case of our ML-based word association strengths (our ML strength predictor takes 415 feature dimensions as input), graph-based associations perform similarly and do truly use only 28 dimensions. Overall performance is similar across both datasets with the only notable exception being graph-based USF performing better on OL than PotD. This is likely due to OL being better suited than PotD to USF's relatively smaller set of associations (72,176 pairs and 10,617 unique words versus EAT's 325,588 and 23,218).

4.1 Word Association Strengths

As shown in Table 1, word association strength features outperform Word2Vec similarity. This provides the first clear evidence of their usefulness in humour recognition. This is particularly notable in the case of graph-based associations as they are based on a much smaller dataset and vocabulary.

Despite differences in their individual feature performance on PotD, our model's overall performance is very similar for both graph- and ML-based word association features. This was somewhat unexpected as ML-based associations were included specifically to address coverage issues with graph-based associations noted in Cattle and Ma (2017b). However, graph-based associations seem to exhibit an acceptable level of performance on both datasets, even outperforming ML on OL. One possible explanation is document length. Tweets in Potash et al. (2017)'s dataset averaged 5.5 words (after removing each game's hashtag and the source TV show's Twitter username) versus 13.3 for PotD puns and 10.6 for OL one-liners. As document length increases, the number of word pairs increases exponentially, raising the likelihood of finding at least one word pair with a valid word association strength.

It was expected that ML-based associations would outperform Word2Vec similarity since, as noted in Section 3.3, our association strength prediction model uses Word2Vec similarity as an input. However,

[7] https://kheafield.com/code/kenlm/

	Pun of the Day				16000 One-Liners			
	Acc	P	R	F1	Acc	P	R	F1
Yang et al. (2015)	0.795	0.761	0.862	0.808	0.798	0.801	0.794	0.797
All Features (graph)	0.757	0.755	0.764	0.759	**0.759**	**0.745**	**0.787**	**0.765**
All Features (ML)	**0.763**	**0.757**	**0.780**	**0.768**	0.742	0.727	0.777	0.751
All Features (ML) + HA	0.711	0.700	0.741	0.720	0.679	0.657	0.748	0.700
Perplexity	0.610	0.613	0.608	0.610	0.591	0.591	0.589	0.590
Word2Vec	0.658	0.652	0.685	0.668	0.657	0.655	0.661	0.658
Word Associations (graph)	0.695	0.692	0.706	0.699	**0.747**	**0.735**	**0.773**	**0.753**
Word Associations (ML)	**0.713**	**0.703**	**0.741**	**0.722**	0.720	0.712	0.740	0.726
EAT (graph)	0.670	0.665	0.691	0.678	0.729	0.716	0.761	0.738
EAT (ML)	0.697	0.690	0.722	0.706	0.708	0.701	0.725	0.713
EAT (ML) forward	0.665	0.664	0.674	0.669	0.668	0.667	0.673	0.670
EAT (ML) backward	0.641	0.640	0.651	0.646	0.662	0.663	0.661	0.662
EAT (ML) micro diff	0.563	0.568	0.541	0.555	0.612	0.607	0.637	0.622
EAT (ML) macro diff	0.557	0.558	0.567	0.562	0.606	0.613	0.575	0.594
USF (graph)	0.611	0.618	0.586	0.602	0.733	0.723	0.754	0.738
USF (ML)	0.692	0.686	0.713	0.699	0.709	0.700	0.732	0.715
USF (ML) forward	0.676	0.674	0.685	0.679	0.667	0.664	0.677	0.670
USF (ML) backward	0.637	0.637	0.646	0.641	0.672	0.668	0.681	0.674
USF (ML) micro diff	0.567	0.572	0.544	0.558	0.639	0.631	0.666	0.648
USF (ML) macro diff	0.554	0.554	0.576	0.565	0.603	0.611	0.567	0.588

Table 1: Selected Results of Binary Humour Classification Experiments

it should be noted that, like Cattle and Ma (2017a)'s model, the single highest performing feature was not Word2Vec similarity but vector offset (the difference between two word's Word2Vec embeddings). This further highlights the limitations of Word2Vec similarity described in Section 2.1 as our strength predictor is clearly learning something that cannot be captured by cosine similarity alone.

While Cattle and Ma (2016) provided evidence that the difference between the forward and backward word association strengths was more useful than forward or backward strengths alone, this is not the case here. Both micro and macro difference performed worse than either forward or backward. Furthermore, forward and backward perform similarly. This is likely due to Cattle and Ma (2016)'s use of Twitter hashtag wars, meaning setup and punchline were clearly marked. As such, forward strengths were guaranteed to represent associations from the setup to the punchline and backward strengths were guaranteed to be from the punchline back to the setup. In our approach, setups and punchlines were not explicitly defined. This meant it was difficult to know if a specific word pair represented setup to punchline, punchline to setup, or even setup to setup or punchline to punchline. Even using humour anchors (HAs) does not solve this problem since, as referenced in Section 2.2, identifying HAs is slightly different from identifying setups and punchlines. While labelling a word span as a setup or a punchline gives us some insight into its purpose in the joke (i.e. whether it is meant to establish context or to trigger a reframing, respectively), HAs do not include this information.

4.2 Humour Anchors

As mentioned in Section 2.2, using HAs for humour recognition is an appealing notion and would allow semantic features to be targeted only to meaningful word pairs, potentially increasing their effectiveness. The wonderfully simple extraction method described in Yang et al. (2015) only makes HAs more intriguing. Unfortunately, as can be seen in Table 1, HA targetting actually hurts the performance of our humour model.

One obvious suspect for this drop in performance is the quality of the extracted HAs, a sample of which is shown in Table 2. While the *Honeymoon* and *LASER* examples seem relatively accurate, many

I'll **[Marry]** You Tomorrow But Let's **[Honeymoon Tonight]**.
Caution! Do not look into **[LASER]** with **[remaining eye]**.
Newsflash! Dyslexic Christian **[sells]** soul to **[Santa]**.
[Dark] is faster than light, **[otherwise]** you would see it.
If there **[were]** a single **[word]** to **[describe]** me it would have to be profectionist.

Table 2: Sample extracted humour anchors

extracted anchors were either incomplete, as is the case with *Dark* and *Santa*, or nonsensical, like *profectionist*.

As described in Section 2.2, Yang et al. (2015)'s HA extraction algorithm requires a fully trained humour model, the accuracy of which undoubtedly affects the quality of the extracted HAs. For this reason we also experimented with training our anchor candidate scorer using the test data, to maximize its performance. While this approach is problematic, it does provide an upper bound for our HA extraction performance. While using such HA targetted models did result in increased humour classification performance ($ACC = 0.740$, $P = 0.735$, $R = 0.754$, $F1 = 0.744$ on PotD. $ACC = 0.706$, $P = 0.678$, $R = 0.783$, $F1 = 0.727$ on OL.), it still failed to exceed our non-HA models.

We chose our baseline Yang et al. (2015) humour classifier as our anchor candidate scorer for simplicity but their HA extraction algorithm is able to work with any humour recognition model so long as it is robust to word order and capable of generating a humour score (in our case, we used humour probability). Therefore, using a more accurate humour model may have led to better performance.

Another reason HAs may have hurt humour recognition performance is that it may be make non-humorous documents more humorous. Yang et al. (2015)'s method finds the combination of humour anchor candidates that cause the largest drop humour score, i.e. by design it selects a subset of words which positively effect humour scores. As such, HAs may be more likely to be judged as humorous even if the documents they are extracted from are not.

5 Conclusion and Future Work

In this paper we have presented a humour classification system based on word associations and shown performance, across two datasets, above state-of-the-art non-neural models. Furthermore, we show that word association features outperform Word2Vec similarity on this task, providing the first significant evidence that word associations are particularly well suited to computational humour tasks. Furthermore, we explored the effectiveness of humour anchors for humour recognition and found they actually hurt performance.

Possible next steps include exploring the usefulness of word association features on other computational humour tasks such as relative humour ranking or even humour generation. On a more fundamental level, there are still many aspects of word association features left to explore such as examining different datasets or even different strength metrics (e.g. overlap strength, the proportion of shared associations between two words).

References

Salvatore Attardo and Victor Raskin. 1991. Script theory revis(it)ed: Joke similarity and joke representation model. *Humor-International Journal of Humor Research*, 4(3-4):293–348.

Salvatore Attardo. 2008. Semantics and pragmatics of humor. *Language and Linguistics Compass*, 2(6):1203–1215.

Satanjeev Banerjee and Ted Pedersen. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 805–810, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Dario Bertero and Pascale Fung. 2016a. A long short-term memory framework for predicting humor in dialogues. In *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016-Proceedings of the Conference*, pages 130–135.

Dario Bertero and Pascale Fung. 2016b. Predicting humor response in dialogues from TV sitcoms. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5780–5784.

Kim Binsted and Graeme Ritchie. 1994. An implemented model of punning riddles. In *Proceedings of the Twelfth AAAI National Conference on Artificial Intelligence*, pages 633–638. AAAI Press.

Kim Binsted. 1996. *Machine humour: An implemented model of puns*. Ph.D. thesis, The University of Edinburgh.

Ondřej Bojar, Rajan Chatterjee, Christian Federmann, Barry Haddow, Chris Hokamp, Matthias Huck, Varvara Logacheva, and Pavel Pecina. 2015. Proceedings of the tenth workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*.

Jordan Boyd-Graber, Christiane Fellbaum, Daniel Osherson, and Robert Schapire. 2006. Adding dense, weighted connections to WordNet. In *Proceedings of the Third Global WordNet Meeting*, Jeju Island, Korea.

Andrew Cattle and Xiaojuan Ma. 2016. Effects of semantic relatedness between setups and punchlines in Twitter hashtag games. *Workshop on Computational Modeling of Peoples Opinions, Personality, and Emotions in Social Media (PEOPLES 2016)*, page 70.

Andrew Cattle and Xiaojuan Ma. 2017a. Predicting word association strengths. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1283–1288, Copenhagen, Denmark, September. Association for Computational Linguistics.

Andrew Cattle and Xiaojuan Ma. 2017b. SRHR at SemEval-2017 task 6: Word associations for humour recognition. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 400–405, Vancouver, Canada, August. Association for Computational Linguistics.

Simon De Deyne and Gert Storms. 2008. Word associations: Norms for 1,424 dutch words in a continuous task. *Behavior Research Methods*, 40(1):198–205.

Simon De Deyne, Amy Perfors, and J. Daniel Navarro. 2016. Predicting human similarity judgments with distributional models: The value of word associations. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1861–1870. The COLING 2016 Organizing Committee.

David Donahue, Alexey Romanov, and Anna Rumshisky. 2017. HumorHawk at SemEval-2017 task 6: Mixing meaning and sound for humor recognition. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 98–102, Vancouver, Canada, August. Association for Computational Linguistics.

Francis Galton. 1879. Psychometric experiments. *Brain*, 2(2):149–162.

Yoshihiko Hayashi. 2016. Predicting the evocation relation between lexicalized concepts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1657–1668, Osaka, Japan, December. The COLING 2016 Organizing Committee.

George R Kiss, Christine Armstrong, Robert Milroy, and James Piper. 1973. An associative thesaurus of english and its computer analysis. *The computer and literary studies*, pages 153–165.

Marin Kukovačec, Juraj Malenica, Ivan Mršić, Antonio Šajatović, Domagoj Alagić, and Jan Šnajder. 2017. TakeLab at SemEval-2017 task 6: #rankinghumorin4pages. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 395–399, Vancouver, Canada, August. Association for Computational Linguistics.

Igor Labutov and Hod Lipson. 2012. Humor as circuits in semantic networks. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 150–155. Association for Computational Linguistics.

Xiaojuan Ma. 2013. Evocation: analyzing and propagating a semantic link based on free word association. *Language resources and evaluation*, 47(3):819–837.

Rada Mihalcea and Stephen Pulman. 2007. Characterizing humour: An exploration of features in humorous texts. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 337–347. Springer.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: Investigations in automatic humor recognition. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 531–538. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

George Miller and Christiane Fellbaum. 1998. *WordNet: An electronic lexical database*. MIT Press Cambridge.

Tristan Miller, Christian Hempelmann, and Iryna Gurevych. 2017. SemEval-2017 task 7: Detection and interpretation of english puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 58–68, Vancouver, Canada, August. Association for Computational Linguistics.

Douglas L Nelson, Cathy L McEvoy, and Thomas A Schreiber. 2004. The university of south florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers*, 36(3):402–407.

Saša Petrović and David Matthews. 2013. Unsupervised joke generation from big data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 228–232, Sofia, Bulgaria, August. Association for Computational Linguistics.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2017. SemEval-2017 task 6: #HashtagWars: Learning a sense of humor. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 49–57, Vancouver, Canada, August. Association for Computational Linguistics.

Dragomir Radev, Amanda Stent, Joel Tetreault, Aasish Pappu, Aikaterini Iliakopoulou, Agustin Chanfreau, Paloma de Juan, Jordi Vallmitjana, Alejandro Jaimes, Rahul Jha, and Robert Mankoff. 2016. Humor in collective discourse: Unsupervised funniness detection in the New Yorker cartoon caption contest. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, 05. European Language Resources Association (ELRA).

Victor Raskin. 1985. Semantic theory of humor. In *Semantic Mechanisms of Humor*. Springer.

Sascha Rothe and Hinrich Schütze. 2015. AutoExtend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1793–1803, Beijing, China, July. Association for Computational Linguistics.

Dafna Shahaf, Eric Horvitz, and Robert Mankoff. 2015. Inside jokes: Identifying humorous cartoon captions. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1065–1074, New York, NY, USA. ACM.

Robert Speer and Catherine Havasi. 2012. Representing general relational knowledge in ConceptNet 5. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 3679–3686, Istanbul, Turkey, May. European Language Resources Association (ELRA). ACL Anthology Identifier: L12-1639.

Oliviero Stock and Carlo Strapparava. 2002. HAHAcronym: Humorous agents for humorous acronyms. *Stock, Oliviero, Carlo Strapparava, and Anton Nijholt. Eds*, pages 125–135.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.

Xinru Yan and Ted Pedersen. 2017. Duluth at semeval-2017 task 6: Language models in humor detection. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 384–388, Vancouver, Canada, August. Association for Computational Linguistics.

Diyi Yang, Alon Lavie, Chris Dyer, and Eduard Hovy. 2015. Humor recognition and humor anchor extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2367–2376, Lisbon, Portugal, September. Association for Computational Linguistics.

A Retrospective Analysis of the Fake News Challenge Stance Detection Task

Andreas Hanselowski[†], Avinesh PVS[†], Benjamin Schiller[†], Felix Caspelherr[†], Debanjan Chaudhuri[‡]*, Christian M. Meyer[†], Iryna Gurevych[†]
Research Training Group AIPHES
[†]Computer Science Department, Technische Universität Darmstadt
[‡]Smart Data Analytics, University of Bonn
https://www.aiphes.tu-darmstadt.de

Abstract

The 2017 Fake News Challenge Stage 1 (FNC-1) shared task addressed a stance classification task as a crucial first step towards detecting fake news. To date, there is no in-depth analysis paper to critically discuss FNC-1's experimental setup, reproduce the results, and draw conclusions for next-generation stance classification methods. In this paper, we provide such an in-depth analysis for the three top-performing systems. We first find that FNC-1's proposed evaluation metric favors the majority class, which can be easily classified, and thus overestimates the true discriminative power of the methods. Therefore, we propose a new F1-based metric yielding a changed system ranking. Next, we compare the features and architectures used, which leads to a novel feature-rich stacked LSTM model that performs on par with the best systems, but is superior in predicting minority classes. To understand the methods' ability to generalize, we derive a new dataset and perform both in-domain and cross-domain experiments. Our qualitative and quantitative study helps interpreting the original FNC-1 scores and understand which features help improving performance and why. Our new dataset and all source code used during the reproduction study are publicly available for future research[1].

1 Introduction

Recently, Pomerleau and Rao (2017) organized the first Fake News Challenge[2] (FNC-1) in order to foster the development of AI technology to automatically detect fake news. The challenge received much attention in the NLP community: 50 teams from both academia and industry participated. The goal of the FNC-1 challenge is to determine the perspective (or *stance*) of a news article relative to a given headline. An article's stance can either *agree* or *disagree* with the headline, *discuss* the same topic, or it is completely *unrelated*. Table 1 shows four example documents illustrating these classes.

Stance detection is a crucial building block for a variety of tasks, such as analyzing online debates (Walker et al., 2012; Sridhar et al., 2015; Somasundaran and Wiebe, 2010), determining the veracity of rumors on twitter (Lukasik et al., 2016; Derczynski et al., 2017), or understanding the argumentative structure of persuasive essays (Stab and Gurevych, 2017). While stance detection has been previously focused on individual sentences or phrases, the systems participating in FNC-1 have to detect the stance of an entire document, which raises many new challenges. Although the disagreeing article of Table 1 clearly leans against the headline's claim, the fourth sentence would agree to it if considered in isolation.

To properly learn from a scientific shared task, there are typically overview and analysis papers that compare the architectures, features, and results of the participating systems. To date, there is, however, no such paper for FNC-1, which is why we conduct a reproduction study of the top three participating systems. Our goal is to independently verify the results reported in the challenge, which is an important asset in empirical research, to critically assess the experimental setup of FNC-1, and to learn building

*The work by Debanjan Chaudhuri has been carried out during his internship at the Ubiquitous Knowledge Processing (UKP) Lab / Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) from 01.01.2017 to 30.06.2017

[1] https://github.com/UKPLab/coling2018_fake-news-challenge

[2] http://www.fakenewschallenge.org/

Proceedings of the 27th International Conference on Computational Linguistics, pages 1859–1874
Santa Fe, New Mexico, USA, August 20-26, 2018.

Headline: Hundreds of Palestinians flee floods in Gaza as Israel opens dams	
Agree (AGR)	GAZA CITY (Ma'an) – Hundreds of Palestinians were evacuated from their homes Sunday morning after Israeli authorities opened a number of dams near the border, flooding the Gaza Valley in the wake of a recent severe winter storm. The Gaza Ministry of Interior said in a statement that civil defense services and teams from the Ministry of Public Works had evacuated more than 80 families from both sides of the Gaza Valley (Wadi Gaza) after their homes flooded as water levels reached more than three meters [..]
Discuss (DSC)	Palestinian officials say hundreds of Gazans were forced to evacuate after Israel opened the gates of several dams on the border with the Gaza Strip, and flooded at least 80 households. Israel has denied the claim as "entirely false". [..]
Disagree (DSG)	Israel has rejected allegations by government officials in the Gaza strip that authorities were responsible for released storm waters flooding parts of the besieged area. "The claim is entirely false, and southern Israel does not have any dams," said a statement from the Coordinator of Government Activities in the Territories (COGAT). "Due to the recent rain, streams were flooded throughout the region with no connection to actions taken by the State of Israel." At least 80 Palestinian families have been evacuated after water levels in the Gaza Valley (Wadi Gaza) rose to almost three meters. [..]
Unrelated (UNR)	Apple is continuing to experience 'Hairgate' problems but they may just be a publicity stunt [..]

Table 1: Headline and text snippets from document bodies with respective stances from the FNC dataset

better methods by understanding their merits and drawbacks. Based on our analysis of the shared task data, we first propose a new evaluation metric for FNC-1 and related document-level stance detection tasks, which is less affected by highly imbalanced datasets. To understand the headroom of the state-of-the-art performance, we additionally estimate the upper bound for this task. In a feature ablation study, we then identify which features contribute to solving the stance detection task. On the basis of our analysis, we combine ideas from previous systems and propose a novel architecture that performs on par with the state-of-the-art systems, but is better able to correctly classify difficult cases. Since generalizability is crucial for the method's future impact, we finally introduce a new evaluation dataset and evaluate how well the FNC-1 models generalize to unseen data from a different domain. In addition to in-domain experiments, we also conduct cross-domain experiments in order to analyze the transfer potential of a method.

2 Related Work

Previous works in stance detection mostly considered target-specific stance prediction, whereby the stance of a text entity with respect to a topic or a named entity is determined. Target-specific stance prediction has been performed for tweets (Mohammad et al., 2016; Augenstein et al., 2016; Zarrella and Marsh, 2016) and online debates (Walker et al., 2012; Somasundaran and Wiebe, 2010; Sridhar et al., 2015). Such target-specific approaches are based on structural (Walker et al., 2012), linguistic and lexical features (Somasundaran and Wiebe, 2010) and they jointly model disagreement only and collective stance using probabilistic soft logic (Sridhar et al., 2015) or neural models (Zarrella and Marsh, 2016; Du et al., 2017) with conditional encoding (Augenstein et al., 2016). Stance prediction in tweets (Mohammad et al., 2016; Augenstein et al., 2016; Du et al., 2017) and in online debates (Hasan and Ng, 2013) is different from that of stance detection in a news article, which – while similar – is concerned with stance detection of a news article relative to a statement in natural language.

To the best of our knowledge, there is yet no overview or analysis paper on FNC-1 similar to the shared task on detecting stance in twitter (Mohammad et al., 2016; Derczynski et al., 2017; Taulé et al., 2017). To demonstrate the best scientific practices and achieve research transparency, we close this gap by systematically reviewing the top-ranked systems at FNC-1.

The FNC-1 stance detection task is inspired by Ferreira and Vlachos (2016), who classify the stance of a single sentence of a news headline towards a specific claim. In FNC-1, however, the task is document-level stance detection, which requires the classification of an entire news article relative to a headline. The top performing system in FNC-1 is called *SOLAT in the SWEN* (Sean et al., 2017) by Talos Intelligence (henceforth: Talos). They use a combination of deep convolutional neural networks and gradient-boosted decision trees with lexical features. Team *Athene* (Hanselowski et al., 2017) won the second place with

Dataset	headlines	documents	tokens	instances	AGR	DSG	DSC	UNR
FNC-1	2,587	2,587	372	75,385	7.4%	2.0%	17.7%	72.8%

Table 2: Corpus statistics and label distribution for the FNC-1 dataset

an ensemble of five multi-layer perceptrons (MLP) with six hidden layers each and handcrafted features. For the prediction they used hard voting. Finally, *UCL Machine Reading* (*UCLMR*) (Riedel et al., 2017) were placed third using a multi-layer perceptron with bag-of-words features. Additionally, recently published work on FNC-1 use a two-step logistic regression based classifier (Bourgonje et al., 2017) and a stacked ensemble of five classifiers (Thorne et al., 2017) which achieve 9th and 11th places respectively. Although multiple systems[3] participated at FNC-1, we focus on the top three systems in this paper, due to the availability of source code and our goal of analyzing what contributes most to good performance. In the remaining paper, we introduce and analyze these three systems in detail.

3 Reproduction of the Fake News Challenge FNC-1

In this section, we take a closer look at the challenge. We briefly discuss the task and dataset of FNC-1, describe the three top-ranked systems and reproduce their results.

FNC-1 task and dataset. The task in FNC-1 is learning a classifier $f\colon (d, h) \mapsto s$ that predicts one of four stance labels $s \in S = \{\text{AGR}, \text{DSG}, \text{DSC}, \text{UNR}\}$ for a document d with regard to a headline h. If headline and document cover different topics, the stance is $s = \text{UNR}$ (unrelated). Otherwise, s is AGR if d agrees and DSG if d disagrees with h. If h and d merely discuss the same topic, but d does not take a definite position, s will be DSC.

To evaluate the challenge, the organizers provide a dataset[4] of 300 topics. The topics are represented by claims with 5–20 news article documents each. The dataset is derived from the Emergent project (Silverman, 2017) which addressed rumor debunking. In the project, each news article document was summarized into a headline that reflects the stance of the whole document. Other than for rumor debunking, the FNC-1 organizers match each document with every summarized headline and then label the (d, h) pair with one of the four stance labels S. To generate the unrelated class UNR, headlines and documents belonging to different topics are randomly matched. Document–headline pairs of 200 topics are reserved for training, the remaining document–headline pairs of 100 topics for testing. Topics, headlines, and documents are therefore not shared between the two data splits. To prevent teams from using any unfair means by deriving the labels for the test set from the publicly available Emergent data, the organizers additionally created 266 instances. Table 2 shows the corpus size and label distribution.

Participating systems. For our reproduction study, we consider FNC-1's three top-ranked systems. Talos Intelligence's SOLAT in the SWEN team (Sean et al., 2017) won the FNC-1 using their weighted average model (*TalosComb*) of a deep convolutional neural network (*TalosCNN*) and a gradient-boosted decision trees model (*TalosTree*). *TalosCNN* uses pre-trained word2vec embeddings[5] passed through several convolutional layers followed by three fully-connected and a final softmax layer for classification. *TalosTree* is based on word count, TF-IDF, sentiment, and singular-value decomposition features in combination with the word2vec embeddings.

Team *Athene* (Hanselowski et al., 2017) was ranked second. They propose a multi-layer perceptron (MLP) inspired by the work of Davis and Proctor (2017). They extend the original model structure to six hidden and a softmax layer and they incorporate multiple hand-engineered features: unigrams, the cosine similarity of word embeddings of nouns and verbs between headline and document tokens, and topic models based on non-negative matrix factorization, latent Dirichlet allocation, and latent semantic indexing in addition to the baseline features provided by the FNC-1 organizers. Depending on the feature type, they either form separate feature vectors for document and headline, or a joint feature vector.

[3] e.g., `http://web.stanford.edu/class/cs224n/reports.html`

[4] `https://github.com/FakeNewsChallenge/fnc-1-baseline`

[5] https://code.google.com/archive/p/word2vec/

Systems	FNC-FNC					
	FNC	F_1m	AGR	DSG	DSC	UNR
Majority vote	.394	.210	0.0	0.0	0.0	.839
TalosComb	.820	.582	**.539**	.035	.760	.994
TalosTree	**.830**	.570	.520	.003	.762	.994
TalosCNN	.502	.308	.258	.092	0.0	.882
Athene	.820	.604	.487	.151	**.780**	**.996**
UCLMR	.817	.583	.479	.114	.747	.989
featMLP	.825	.607	.530	.151	.766	.982
stackLSTM	.821	**.609**	.501	**.180**	.757	.995
Upper bound	.859	.754	.588	.667	.765	.997

Table 3: FNC, F_1m, and class-wise F_1 scores for the analyzed models on in-domain experiments

The UCL Machine Reading (*UCLMR*) team propose an MLP as well, but use only a single hidden layer (Riedel et al., 2017). Their system was ranked third. As features, they use term frequency vectors of unigrams of the 5,000 most frequent words for the headlines and the documents. Additionally, they compute the cosine similarity between the TF-IDF vectors of the headline and document. The resulting term frequency feature vectors of headline and document are concatenated along with the cosine similarity of the two TF-IDF vectors.

Reproduction. Following the instructions from the GitHub repositories of the three teams,[6] we could successfully reproduce the results reported in the competition without significant deviations. Table 3 shows these results in the FNC column of the *FNC-FNC* setup, which means that the models were trained and tested on the FNC dataset. Since Talos use a combination of two models, we have also included the results of *TalosCNN* and *TalosTree*. A first interesting finding is that *TalosTree* even outperforms the combined model, since the CNN component performed poorly. To understand the merits and drawbacks of the systems, we analyze the performance metrics and the features used, as discussed in the following sections.

4 Performance evaluation

In this section, we critically assess the FNC-1 evaluation methodology and we determine a human upper bound for this task in order to identify the room for improvement for the document-level stance detection task.

Evaluation metrics. The FNC-1 organizers propose the hierarchical evaluation metric FNC, which first awards .25 points if a document is correctly classified as related (i.e., $s \in \{\text{AGR}, \text{DSG}, \text{DSC}\}$) or UNR to a given headline. If it is related, .75 additional points are assigned if the model correctly classifies the document-headline pair as AGR, DSG, or DSC. The goal of this weighting schema is to balance out the large number of unrelated instances.

Nevertheless, the metric fails to take into account the highly imbalanced class distribution of the three related classes AGR, DSG, and DSC illustrated in Table 2. Thus, models, which perform well on the majority class and poorly on the minority classes are favored. Since it is not difficult to separate related from unrelated instances (the best systems reach about $F_1 = .99$ for the UNR class), a classifier that just randomly predicts one of the three related classes would already achieve a high FNC score. A classifier that always predicts DSC for the related documents even reaches FNC = .833, which is even higher than the top-ranked system.

We therefore argue that the FNC metric is not appropriate for validating the document-level stance detection task. Instead, we propose the class-wise and the macro-averaged F_1 scores (F_1m) as a new metric for this task that is not affected by the large size of the majority class. The class-wise F_1 scores

[6] https://github.com/Cisco-Talos/fnc-1; https://github.com/hanselowski/athene_system; https://github.com/uclmr/fakenewschallenge

are the harmonic means of the precisions and recalls of the four classes, which are then averaged to the F_1m metric. The naïve approach of perfectly classifying UNR and always predicting DSC for the related classes, would achieve only F_1m = .444, which is clearly different from the proposed systems. By averaging over the individual classes' scores, F_1m is also applicable to other datasets, which have a different class distribution than the FNC-1 dataset. While the averaged F_1m objectively reflects the quality of the prediction rather than the class distribution, we can also analyze which classes cannot be properly predicted yet.

As the scores in Table 3 indicate, the performance of the three top-ranked systems reach only about F_1m = .6. Our analysis reveals that *TalosCNN* does not predict the DSC class yielding an F_1 score of zero. Also the overall performance of this model is low and according to the FNC metric, *TalosTree* would even outperform *TalosComb*. In contrast, *TalosTree* returns almost no predictions for the DSG class, although it performs exceptionally well in terms of FNC. This is because it often predicts the majority class DSC for the related documents. Since there are only few DSG instances in the dataset, the overall performance of this model appears high.

Considering the FNC-1 results according to our proposed F_1m metric, the ranking of the three systems changes: The *TalosComb* and *TalosTree* systems are slightly outperformed by *UCLMR* and clearly outperformed by the Athene system. This is because the two Talos models benefit from the FNC metric definition, favoring the prediction of the majority classes UNR and DSC. On smaller classes, such as DSG, they perform much worse than Athene and *UCLMR*. Using F_1m as a metric, the Athene system would be ranked first, as it outperforms *UCLMR* by 2.1 percentage points. In addition to that, Athene also works best on the DSG, DSC, and UNR class.

Human upper bound. In addition to the issues with the evaluation metric, there is also no upper bound reported for the FNC-1 data, although this will help estimating the headroom of the proposed systems with regard to human performance. Therefore, we ask five human raters to manually label 200 instances. The raters reach an overall inter-annotator agreement of Fleiss' $\kappa = .686$ (Fleiss, 1971), which is substantial and allows drawing tentative conclusions (Artstein and Poesio, 2008). However, when ignoring the UNR class, the inter-annotator agreement dramatically drops to $\kappa = .218$. This indicates that differentiating between the three related classes AGR, DSG, and DSC is difficult even for humans.

On the basis of the annotation, we also determine the most probable stance labels according to MACE (Hovy et al., 2013), and compare them to the ground truth from the Emergent project. The agreement of the labels in this case is better, reaching a Fleiss' κ of .807 overall and .552 for the three related classes. The MACE-based most probable label allows us to compute the human upper bound as F_1m = .754, which we include in Table 9 along with the upper bound per class F1 scores UNR = .997 AGR = .588, DSG =.667, and DSC = .765.

5 Analysis of models and features

In this section, we first perform an error analysis in order to be able to find out what the three best performing models are learning and in which cases they fail. In order to address the identified drawbacks, we conduct a systematic feature analysis and derive an alternative model based on our findings.

5.1 Error analysis

Our error analysis for the three analyzed systems shows that the models fail in the following cases: (1) If there is lexical overlap between the headline and the document, the models classify the instance as one of the related classes, even in cases in which the two are unrelated. (2) If the document–headline pair is related, but only contains synonyms rather than the same tokens, the model often misclassifies the case as UNR. (3) If keywords like *reports*, *said*, or *allegedly* are detected, the systems classify the pair as DSC. (4) The DSG class is especially difficult to determine, as only few lexical indicators (e.g., *false*, *hoax*, *fake*) are available as features. The disagreement is often expressed in complex terms which demands more sophisticated machine learning techniques. For example: "If the bizarre story about... sounded outlandish, that's because it was". In appendix A.2, we illustrate these errors with concrete examples.

The analysis shows that the models exploit the similarity between the headline and the document in terms of lexical overlap. Lexical cue words, such as *reports*, *said*, *false*, *hoax* play an important role in classification. However, the systems fail when semantic relations between words need to be taken into account, complex negation instances are encountered, or the understanding of propositional content in general is required. This is not surprising since the three models are based on n-grams, bag-of-words, topic models and lexicon-based features instead of capturing the semantics of the text. In this section, we test these features systematically and we propose new features and a new architecture for FNC-1.

5.2 Feature analysis

Throughout our feature analysis, we use the Athene model, which performed best in terms of F_1m and allows a large number of experiments due to its fast computation. All tests are performed on the FNC-1 development set with 10-fold cross-validation. In the remaining section, we first discuss and evaluate the performance of each feature individually and then conduct an ablation test for groups of similar features. Detailed feature descriptions are included in the supplementary material (section A.1). Figure 1 shows the system performance of the individual features discussed below.

FNC-1 baseline features. The FNC-1 organizers provide a gradient-boosting baseline using the co-occurrence (COOC) of word and character n-grams in the headline and the document as well as two lexicon-based features, which count the number of refuting (REFU) and polarity (POLA) words based on small word lists. Figure 1 indicates that COOC performs well, whereas both lexicon-based features are on par with the majority vote baseline.

Challenge features. The three analyzed FNC-1 systems rely on combinations of the following features: Bag-of-words (BoW) unigram features, topic model features based on non-negative matrix factorization (NMF-300, NMF-cos) (Lin, 2007), Latent Dirichlet Allocation (LDA-cos) (Blei et al., 2001), Latent Semantic Indexing (LSI-300) (Deerwester et al., 1990), two lexicon-based features using NRC Hashtag Sentiment (NRC-Lex) and Sentiment140 (Sent140) (Mohammad et al., 2013), and word similarity features which measure the cosine similarity of pre-trained word2vec embeddings of nouns and verbs in the headlines and the documents (WSim). The topic models use 300 topics. Besides the concatenated topic vectors, we also consider the cosine similarity between the topics of document and headline (NMF-cos, LDA-cos). The BoW features perform best in terms of F_1m. While LSI-300, NMF-300 and NMF-cos topic models yield high scores, LDA-cos and WSim fall behind.

Novel features. We also analyze a number of novel features for the FNC-1 task which have not been used in the challenge. Bag-of-character 3-grams (BoC) represent subword information. They show promising results in our setup. The structural features (STRUC) include the average word lengths of the headline and the document, the number of paragraphs in the document and their average lengths. The low performance of this feature indicates that the structure of the headline and the documents is not indicative of their stance. Furthermore, we test readability features (READ) which estimate the complexity of a text. Less complex texts could be indicative of deficiently written fake news. We tried the following metrics for headline and document as a concatenated feature vector: SMOG grade (Mc Laughlin, 1969), Flesch-Kincaid grade level and Flesch reading ease (Kincaid et al., 1975), Gunning fog index (Štajner et al., 2012), Coleman-Liau index (Mari and Ta Lin, 1975), automated readability index (Senter and Smith, 1967), LIX and RIX (Jonathan, 1983), McAlpine EFLAW Readability Score (McAlpine, 1997), and Strain Index (Solomon, 2006). However, in the present problem setting these features show only a low performance. The same is true for the lexical diversity (LexDiv) metrics, type-token ratio, and the measure of textual diversity (MTLD) (McCarthy, 2005). We finally analyze the performance of features based on the following lexicons: MPQA (Wilson et al., 2005), MaxDiff (Kiritchenko et al., 2014), and EmoLex (Mohammad and Turney, 2010). These features are based on the sentiment, polarity, and emotion expressed by headlines and documents, which might be good indicators of an author's opinion. However, our results show that these lexicon-based features are not helpful. Even though the considered lexicons are important for fake-news detection (Shu et al. (2017), Horne and Adali (2017)), for stance detection, the properties captured by the lexicon-based features are not very useful.

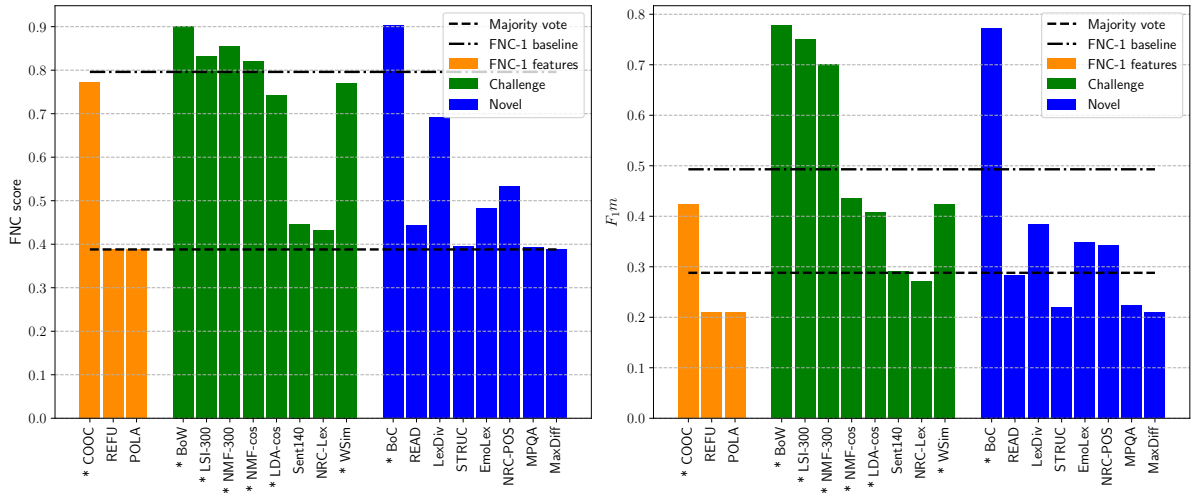


Figure 1: Performance of the system based on individual features

Feature ablation test. We first remove all features that are more than 10% below the FNC-1 baseline, since they mostly predict the majority class and thus harm the F_1m score. In Figure 1, we mark these features with an asterisk (*). To quantify their contribution, we perform an ablation test across three groups of related features: (1) BoW and BoC (BoW/C), (2) LSI-topic, NMF-topic, NMF-cos, LDA-cos (Topic), and (3) NRC-POS and WSim (Oth).

Table 4 show the results of our ablation test. The BoW and BoC features have the biggest impact on the performance. While the topic models yield further improvements, the NRC-POS and WSim features are not helpful. Hence, we suggest BoW, BoC, and the four topic model based features as the most promising feature set.

We evaluate this feature set on the FNC-1 test dataset. The results are included in the *featMLP* row of Table 3 for the *FNC-FNC* setting. Although *featMLP* with the revised feature selection outperforms the best performing FNC-1 system Athene in terms of F_1m and FNC score, the margin is not significant. Similar to the three FNC-1 systems, we observe a .2 performance drop between the development and test dataset. This is most likely because of the 100 new topics in the test dataset, which have not been seen during training. Thus, the evaluation on the test set can be considered as an out-of-domain prediction.

5.3 Model analysis

In order to increase the overall performance, we conduct additional experiments with an ensemble of the three models *featMLP*, *TalosComb*, and *UCLMR* using hard voting. However, we could not significantly improve the results. Since all models struggle with the DSG class, we have applied different under- and over-sampling techniques to balance the class distribution, but also this technique did not yield improved results.

In the error analysis, we observed that the feature-based systems lack semantic understanding. Therefore, we combine a feature-based system with a model that is better able to capture the semantics based

	Baselines		**Only**			**All without**				
	Maj. vote	FNC-1	BoW/C	Topic	Oth	-BoW/C	-Topic	-Oth	All*	All
AGR	0.0	.241	**.772**	.637	0.0	.665	.714	.722	.713	.675
DSG	0.0	.047	.601	.571	0.0	.530	.598	**.616**	.573	.455
DSC	0.0	.738	.874	.838	.731	.841	.863	**.876**	.870	.835
UNR	.835	.970	.991	.983	.964	.982	.989	**.995**	.993	.989
F_1m	.209	.499	.796	.757	.425	.754	.791	**.802**	.787	.738

Table 4: Results of the feature ablation test. Baseline FNC-1 uses gradient boosting classifier with all FNC-1 baseline features. * states that only the preselected features are used (see Figure 1).

Headline: NHL expansion ahead? No, says Gary Bettman
Article body: It wasn't very long ago that NHL commissioner Gary Bettman was treating talk of expansion as though he was being asked if he'd like an epidemic of Ebola. But recently the nature of the rhetoric has changed so much that the question is becoming not if, but when. ...

Table 5: A correctly classified DSG instance by the *stackLSTM*

on word embeddings and sequential encoding. Sequential processing of information is important in order to get the meaning of the whole sentence, e.g. "It wasn't long ago that Gary Bettman was ready to expand NHL." VS. "It was long ago that Gary Bettman wasn't ready to expand NHL." In Figure 2, we introduce this *stackLSTM* model, which combines the best feature set found in the ablation test with a stacked long short-term memory (LSTM) network (Hermans and Schrauwen, 2013). We use 50-dimensional GloVe word embeddings[7] (Pennington et al., 2014) in order to generate sequences of word vectors of a headline–document pair. For this, we concatenate a maximum of 100 tokens of the headline and the document. These embedded word sequences $v_1, v_2, \ldots, v_n$ are fed through two stacked LSTMs with a hidden state size of 100 with a dropout of 0.2 each. The last hidden state of the second LSTM is concatenated with the feature set and fed into a 3-layer neural network with 600 neurons each. Finally, we add a dense layer with four neurons and softmax activation function in order to retrieve the class probabilities.

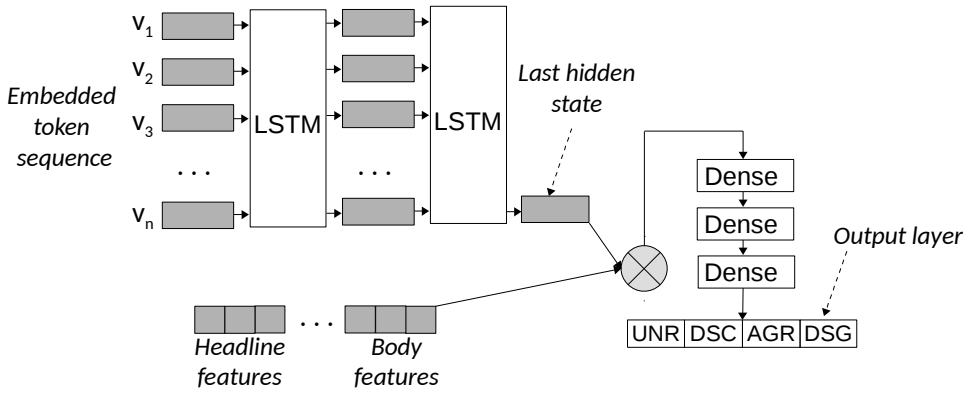


Figure 2: Model Architecture of the feature-rich stackLSTM

Table 3 shows the performance of *stackLSTM* for the *FNC-FNC* setup. Our model outperforms all other methods in terms of F_1m. The difference to Athene and *featMLP* is, however, not significant. An important advantage of *stackLSTM* is its improved performance for the DSG class, which is the most difficult one to predict due to the low number of instances. This means that *stackLSTM* correctly classifies a larger number of complex negation instances. The difference on this difficult DSG class between *stackLSTM* and all other methods is statistically significant (using Student's t-test). The model predicts more often for the DSG class and gets more of these examples correct without compromising the overall performance. One challenging DSG example, which was correctly classified by the *stackLSTM*, is given in Table 5.

6 Analysis of the generalizability of the models

To test the robustness of the models (i.e. how well they generalize to new datasets), we introduce novel test data for document-level stance detection based on the Argument Reasoning Comprehension (ARC) task proposed by Habernal et al. (2018). In this section, we describe the dataset, analyze the models' performance, and perform cross-domain experiments.

ARC dataset. Habernal et al. (2018) manually select 188 debate topics with popular questions from the user debate section of the New York Times. For each topic, they collect user posts, which are highly ranked by other users, and create two claims representing two opposing views on the topic. Then, they

[7] `http://nlp.stanford.edu/data/glove.twitter.27B.zip`

Dataset	headlines	documents	tokens	instances	AGR	DSG	DSC	UNR
ARC	4,448	4,448	99	17,792	8.9%	10.0%	6.1%	75.0%

Table 6: Corpus statistics and label distribution for the ARC dataset

Example from the original ARC dataset		
Topic	Do same-sex colleges play an important role in education, or are they outdated?	
User post	Only 40 women's colleges are left in the U.S. And, while there are a variety of opinions on their value, to the women who have attended ... them, they have been ... tremendously valuable. ...	
Claims	**1.** Same-sex colleges are outdated **2.** Same-sex colleges are still relevant	
Label	Same-sex colleges are still relevant	
Generated instance in alignment with the FNC problem setting		
Stance	**Headline**	**Document**
AGR	Same-sex colleges are still relevant	Only 40 women's colleges are left in the U.S. ...

Table 7: Example of the original ARC dataset and the generated instance to align with FNC dataset

ask crowd workers to decide whether a user post supports either of the two opposing claims or does not express a stance at all. This Argument Reasoning Comprehension (ARC) dataset consists of typical controversial topics from the news domain, such as *immigration*, *schooling issues*, or *international affairs*. While this is similar to the FNC-1 dataset, there are significant differences, as a user post is typically a multi-sentence statement representing one viewpoint on the topic. In contrast, the news articles of FNC-1 are longer and usually provide more balanced and detailed perspective on an issue.

To allow using the ARC data for our FNC stance detection setup, we consider each user post as a document and randomly select one of the two claims as the headline. We label the claim–document pair as AGR if the claim has also been chosen by the workers, as DSG if the workers chose the opposite claim, and as DSC if the workers selected neither claim. Table 7 shows an example of our revised ARC corpus structure. In order to generate the unrelated instances, we randomly match the user posts with claims, but avoid that a user post is assigned to a claim from the same topic. Table 6 provides basic corpus statistics. For training and testing, we split the corpus into 80% training/validation set and 20% testing set.

As for the FNC-1 corpus, we have also determined a human upper bound for the ARC dataset. Five subjects annotate 200 samples using the four classes. Even though the overall Fleiss' $\kappa = .614$ is slightly lower compared to the FNC-1 corpus, the agreement for the three related classes AGR, DSG, and DSC is higher ($\kappa = .383$) than for FNC-1. The human upper bound based on MACE is F_1m = .773. Table 9 contains also the class-wise F_1 scores.

In-domain experiments ARC-ARC: The in-domain results for the ARC corpus listed in Table 8 show that the overall performance of all models decreases. Since the models have been constructed to perform well on the FNC-1 dataset, this is not surprising. Nevertheless, for the ARC corpus, the models are better able to distinguish between AGR and DSG instances. We assume this is because the number of DSG instances is substantially larger and is similar to the number AGR instances. The classification of the DSC instances, on the other hand, turns out to be more challenging on the ARC corpus. This is because even if a user post is related to the claim, it often does not explicitly refer to it. With *TalosComb* being best, the Talos models were better able to generalize to the new data. Even though the *stackLSTM* is again better on the more difficult minority class (in this case DSC), the structure and features of *TalosComb* seem to be more appropriate for this problem setting.

Cross-domain experiments: In the cross-domain setting we train on the training data of one corpus and evaluate on the test data of the other corpus. The experiments in Table 9 show that the performance of the models is substantially better than the majority vote baseline. We therefore conclude that the two problem settings are related and exhibit a common structure. The results suggest that *TalosComb* is best able to learn from the ARC corpus, as it is also superior in the *ARC-FNC* setting. The *stackLSTM*, on the

Systems	ARC-ARC					
	FNC	F_1m	AGR	DSG	DSC	UNR
Majority vote	.430	.214	0.0	0.0	0.0	.857
TalosComb	**.725**	**.573**	**.593**	**.598**	.160	**.944**
Athene	.680	.548	.516	.482	.190	.933
UCLMR	.667	.519	.517	.503	.121	.932
featMLP	.690	.526	.526	.506	.144	.934
stackLSTM	.685	.524	.451	.518	**.194**	.935
Upper bound	.796	.773	.710	.857	.571	.954

Table 8: FNC, F_1m, and class-wise F_1 scores for the analyzed models on in-domain experiments

Systems	FNC-ARC						ARC-FNC					
	FNC	F_1m	AGR	DSG	DSC	UNR	FNC	F_1m	AGR	DSG	DSC	UNR
Majority vote	.430	.214	0.0	0.0	0.0	.857	.394	.210	0.0	0.0	0.0	.839
TalosComb	.584	.365	.336	0.0	.195	**.929**	.607	**.388**	.279	**.183**	**.113**	**.977**
Athene	.523	.340	**.340**	**.244**	.138	.894	.548	.321	.277	.097	.028	.882
UCLMR	.557	.358	.271	.064	**.201**	.896	.482	.288	.234	.109	.080	.728
featMLP	.586	.389	.321	.159	.171	.906	.585	.351	.322	.111	.033	.939
stackLSTM	**.591**	**.401**	.321	.191	.182	.910	**.613**	.373	**.343**	.116	.082	.950
Upper bound	.796	.773	.710	.857	.571	.954	.859	.754	.588	.667	.765	.997

Table 9: FNC, F_1m and class-wise F_1 scores(F_1m) based on cross-domain experiments

other hand, yields best results when trained on the FNC corpus as the *FNC-ARC* setting suggests.

7 Discussion and conclusion

In this paper, we conducted a thorough analysis of the Fake News Challenge stage one stance detection task. Although this is common for shared tasks, there is yet no analysis or reproduction study of this task, which is why we close this gap. Given that the challenge has attracted much attention in the NLP community with 50 participating teams, a detailed analysis is valuable as it provides insights into the problem setting and lessons learned for upcoming competitions. In our investigation, we evaluated the performance of the three top-scoring systems, critically assessed the experimental setup, and performed a detailed feature analysis, in which we identify high-performing features for the task yielding a new model. We conducted an error analysis and found that the models mostly rely on the lexical overlap for classification. To assess how well the models generalize to a similar problem setting, we experimented with a second, newly derived corpus. We also propose a new evaluation metric based on F_1 scores, since the challenge's metric is highly affected by the imbalanced class distribution of the test data. Using this evaluation setup, the ranking of the top three systems changes. Based on our analysis, we conclude that the investigated stance detection problem is challenging, since the best performing features are not yet able to resolve the difficult cases. Thus, more sophisticated machine learning techniques are needed, which have a deeper semantic understanding, and are able to determine the stance on the basis of propositional content instead of relying on lexical features.

8 Acknowledgements

This work has been supported by the German Research Foundation as part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) at the Technische Universität Darmstadt under grant No. GRK 1994/1.

References

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics* 34(4):555–596.

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Austin, TX, USA, pages 876–885.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2001. Latent dirichlet allocation. In *Advances in Neural Information Processing Systems 14 (NIPS)*. Vancouver, BC, Canada, pages 601–608.

Peter Bourgonje, Julian Moreno Schneider, and Georg Rehm. 2017. From Clickbait to Fake News Detection: An Approach based on Detecting the Stance of Headlines to Articles. In *Proceedings of the EMNLP 2017 Workshop 'Natural Language Processing meets Journalism'*. Copenhagen, Denmark, pages 84–89.

Sanjiv R. Das and Mike Y. Chen. 2007. Yahoo! for Amazon: Sentiment Extraction from Small Talk on the Web. *Management Science* 53(9):1375–1388.

Richard Davis and Chris Proctor. 2017. Fake News, Real Consequences: Recruiting Neural Networks for the Fight Against Fake News. Online: `http://web.stanford.edu/class/cs224n/reports/2761239.pdf`. Accessed: 2018-03-16.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41(6):391.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. SemEval-2017 Task 8: RumourEval: Determining rumour veracity and support for rumours. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval)*. Vancouver, BC, Canada, pages 69–76.

Jiachen Du, Ruifeng Xu, Yulan He, and Lin Gui. 2017. Stance classification with target-specific neural attention. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI)*. Melbourne, Australia, pages 3988–3994.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL/HLT)*. San Diego, CA, USA, pages 1163–1168.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin* 76(5):378.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. 2018. The argument reasoning comprehension task: Identification and reconstruction of implicit warrants. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL/HLT)*. New Orleans, LA, USA, pages 1930–1940.

Andreas Hanselowski, Avinesh PVS, Benjamin Schiller, and Felix Caspelherr. 2017. Description of the system developed by team Athene in the FNC-1, 2017. Online: `https://github.com/hanselowski/athene_system/blob/master/system_description_athene.pdf`. Accessed: 2018-03-13.

Kazi Saidul Hasan and Vincent Ng. 2013. Stance Classification of Ideological Debates: Data, Models, Features, and Constraints. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing (IJCNLP)*. Nagoya, Japan, pages 1348–1356.

Michiel Hermans and Benjamin Schrauwen. 2013. Training and analysing deep recurrent neural networks. In *Advances in neural information processing systems 26 (NIPS)*. Stateline, NV, USA, pages 190–198.

Benjamin D. Horne and Sibel Adali. 2017. This Just In: Fake News Packs a Lot in Title, Uses Simpler, Repetitive Content in Text Body, More Similar to Satire than Real News. In *Proceedings of the ICWSM 2017 Workshop on News and Public Opinion*. Montréal, QC, Canada, pages 759–766.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning Whom to Trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL/HLT)*. Atlanta, GA, USA, pages 1120–1130.

Anderson Jonathan. 1983. Lix and Rix: Variations on a Little-known Readability Index. *Journal of Reading* 26(6):490–496.

J. Peter Kincaid, Robert P. Fishburne Jr, Richard L. Rogers, and Brad S. Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Research Branch Report 8-75, Naval Technical Training Command, Millington, TN, USA.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M. Mohammad. 2014. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research* 50:723–762.

Chih-Jen Lin. 2007. Projected gradient methods for nonnegative matrix factorization. *Neural Computation* 19(10):2756–2779.

Michal Lukasik, P.K. Srijith, Duy Vu, Kalina Bontcheva, Arkaitz Zubiaga, and Trevor Cohn. 2016. Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter. In *Proceedings of 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany, pages 393–398.

Coleman Mari and Liau Ta Lin. 1975. A computer readability formula designed for machine scoring. *Journal of Applied Psychology* 60(2):283.

G. Harry Mc Laughlin. 1969. SMOG grading—a new readability formula. *Journal of reading* 12(8):639–646.

Rachel McAlpine. 1997. *Global English for global business*. Longman.

Philip M. McCarthy. 2005. An assessment of the range and usefulness of lexical diversity measures and the potential of the measure of textual, lexical diversity (MTLD). *Dissertation Abstracts International* 66:12.

Saif M. Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. SemEval-2016 Task 6: Detecting Stance in Tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval)*. San Diego, CA, USA, pages 31–41.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets. In *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval)*. Atlanta, GA, USA, pages 321–327.

Saif M. Mohammad and Peter D. Turney. 2010. Emotions Evoked by Common Words and Phrases: Using Mechanical Turk to Create an Emotion Lexicon. In *Proceedings of the NAACL/HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*. Los Angeles, CA, USA, pages 26–34.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a word-emotion association lexicon 29(3):436–465.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar, pages 1532–1543.

Dean Pomerleau and Delip Rao. 2017. The Fake News Challenge: Exploring how artificial intelligence technologies could be leveraged to combat fake news. `http://www.fakenewschallenge.org/`. Accessed: 2017-10-20.

Benjamin Riedel, Isabelle Augenstein, Georgios P Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the fake news challenge stance detection task. *arXiv preprint arXiv:1707.03264* .

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. SemEval-2015 Task 10: Sentiment Analysis in Twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval)*. Denver, CO, USA, pages 451–463.

Baird Sean, Sibley Doug, and Pan Yuxi. 2017. Talos Targets Disinformation with Fake News Challenge Victory. `http://blog.talosintelligence.com/2017/06/talos-fake-news-challenge.html`. Accessed: 2017-12-02.

R.J. Senter and Edgar A. Smith. 1967. Automated readability index. Technical Report AMRL-TR-66-220, University of Cincinnati.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter* 19(1):22–36.

Craig Silverman. 2017. Emergent: A real-time rumor tracker. Online: `http://www.emergent.info/`. Accessed: 2017-12-13.

N. Watson Solomon. 2006. *Strain Index: A New Readability Formula*. Master thesis, Madurai Kamaraj University.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL/HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*. Los Angeles, CA, USA, pages 116–124.

Dhanya Sridhar, James R. Foulds, Bert Huang, Lise Getoor, and Marilyn A. Walker. 2015. Joint models of disagreement and stance in online debate. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL/IJCNLP)*. Beijing, China, pages 116–125.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics* 43(3):619–659.

Sanja Štajner, Richard Evans, Constantin Orăsan, and Ruslan Mitkov. 2012. What can readability measures really tell us about text complexity. In *Proceedings of the LREC 2012 Workshop on Natural Language Processing for Improving Textual Accessibility (NLP4ITA)*. Istanbul, Turkey, pages 14–21.

Mariona Taulé, Maria Antònia Martí, Francisco M. Rangel Pardo, Paolo Rosso, Cristina Bosco, and Viviana Patti. 2017. Overview of the Task on Stance and Gender Detection in Tweets on Catalan Independence. In *Proceedings of the Second Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval) co-located with 33th Conference of the Spanish Society for Natural Language Processing (SEPLN)*. Murcia, Spain, pages 157–177.

James Thorne, Mingjie Chen, Giorgos Myrianthous, Jiashu Pu, Xiaoxuan Wang, and Andreas Vlachos. 2017. Fake news stance detection using stacked ensemble of classifiers. In *Proceedings of the EMNLP 2017 Workshop 'Natural Language Processing meets Journalism'*. Copenhagen, Denmark, pages 80–83.

Marilyn A. Walker, Pranav Anand, Robert Abbott, and Ricky Grant. 2012. Stance classification using dialogic properties of persuasion. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL/HLT)*. Montréal, QC, Canada, pages 592–596.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*. Vancouver, BC, Canada, pages 347–354.

Guido Zarrella and Amy Marsh. 2016. MITRE at SemEval-2016 Task 6: Transfer Learning for Stance Detection. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, CA, USA, pages 458–463.

Xiaodan Zhu, Svetlana Kiritchenko, and Saif M. Mohammad. 2014. NRC-Canada-2014: Recent Improvements in the Sentiment Analysis of Tweets. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval)*. Dublin, Ireland, pages 443–447.

A Supplemental Material

A.1 Features: Detailed description

BoW/BoC features We use bag-of-words (BoW) 1- and 2-grams with 5,000 tokens vocabulary for the headline as well as the document. For the BoW feature, based on a technique by Das and Chen (2007), we add a negation tag "_NEG" as prefix to every word between special negation keywords (e.g. "not", "never", "no") until the next punctuation mark appears. For the bag-of-characters (BoC) 3-grams are chosen with 5,000 tokens vocabulary, too. For the BoW/BoC feature we use the TF to extract the vocabulary and to build the feature vectors of headline and document. The resulting TF vectors of headline and document get concatenated afterwards. Feature *co-occurrence* (FNC-1 baseline feature) counts how many times word 1-/2-/4-grams, character 2-/4-/8-/16-grams, and stop words of the headline appear in the first 100, first 255 characters of the document, and how often they appear in the document overall.

Topic models We use non-negative matrix factorization (NMF) (Lin, 2007), latent semantic indexing (LSI) (Deerwester et al., 1990), and latent Dirichlet allocation (LDA) (Blei et al., 2001) to create topic models out of which we create independent features. For each topic model, we extract 300 topics out of the headline and document texts. Afterwards, we compute the similarity of headlines and bodies to the found topics separately and either concatenate the feature vectors (NMF, LSI) or calculate the cosine distance between them as a single valued feature (NMF, LDA).

Lexicon-based features These features are based on the NRC Hashtag Sentiment and Sentiment140 lexicon (Kiritchenko et al., 2014; Mohammad et al., 2013; Zhu et al., 2014), as well as for the MPQA lexicon (Wilson et al., 2005) and MaxDiff Twitter lexicon (Rosenthal et al., 2015; Kiritchenko et al., 2014). All named lexicons hold values that signal the sentiment/polarity for each word. The features are computed separately for headline and document, and constructed as proposed by Mohammad et al. (2013): First, we count how many words with positive, negative, and without polarity are found in the text. Two features sum up the positive and negative polarity values of the words in the texts and another two features are set by finding the word with the maximum positive and negative polarity value in the text. Finally, the last word in the text with negative or positive polarity is taken as a feature. Since the MaxDiff Twitter lexicon also contains 2-grams, we decide to take them into account as well, whereas for the other lexicons only 1-grams incorporated. Additionally, we base features on the EmoLex lexicon (Mohammad and Turney, 2010, 2013). For all its words, it holds up to eight emotions (anger, fear, anticipation, trust, surprise, sadness, joy, disgust), based on the context they frequently appear in. For headline and document respectively, the emotions for all words are counted as a feature vector. The resulting vectors for headline and document are then concatenated. Lastly, the baseline features *polarity words* and *refuting words* are added. The first one counts refuting words (e.g. "fake", "hoax"), divides the sum by two, and takes the remainder as a feature signaling the polarity of headline or document. The latter one sets a binary feature for each refuting word (e.g. "fraud", "deny") appearing in the headline or document.

Readability features We measure the readability of headline and document with SMOG grade (only document), Flesch-Kincaid grade level, Flesch reading ease, and Gunning fog index (Štajner et al., 2012), Coleman-Liau index (Mari and Ta Lin, 1975), automated readability index (Senter and Smith, 1967), LIX and RIX (Jonathan, 1983), McAlpine EFLAW Readability Score (McAlpine, 1997), Strain Index (Solomon, 2006). The SMOG grade is only valid if a text has at least 30 sentences, and thus is only implemented for the bodies.

Lexical features As lexical features we implement the type-token-ratio (TTR) and the measure of textual lexical diversity (MTLD) (McCarthy, 2005) for the document, and only type-token-ratio for the headline, since MTLD needs at least 50 tokens to be valid. Also, the baseline feature *word overlap* belongs to this group. It divides the cardinality of the intersection of unique words in headline and document by the cardinality of the union of unique words in headline and document.

POS features The POS features amongst others include counters for nouns, personal pronouns, verbs and verbs in past tense, adverbs, nouns and proper nouns, cardinal numbers, punctuations, the ratio of quoted words, and also the frequency of the three least common words in the text. The headline feature also contains a value for the percentage of stop words and the number of verb phrases, which showed good results in the work of Horne and Adali (2017). For the *word-similarity* feature, [which are mainly based on Ferreira and Vlachos (2016) we calculated average word embeddings (pre-trained word2vec model[8]) for all verbs (retrieved with Stanford Core NLP toolkit[9]) of headline/document separately. The cosine similarity between the averaged embeddings of headline and document is taken as a feature, as well as the hungarian distance between verbs of headline and document based on the paraphrase database[10]. The same computation is done for all nouns of headline and document. Additionally the average sentiment of the headline and the average sentiment of the document is used as a feature. A count of negating words of the headline and the document is added to the feature vector as well as the distance from the negated word to the root of the sentence. The number of average words per sentence of headline and document is another feature. The aforementioned features are improved by only selecting a predefined number of sentences of document and headline. Therefore the sentences are ordered by TF-IDF score.

Structural features The structural features contain the average word length of the headline and document, and the number of paragraphs and average paragraph length of the document.

A.2 Misclassified examples identified in the error analysis

Example 1.
(ground truth: "unrelated", system predicts: "agree")
Headline: CNN: Doctor Took Mid-Surgery Selfie with Unconscious Joan Rivers

Document: "A TEENAGER woke up during brain surgery to ask doctors how it was going. Iga Jasica, 19, was having an op to remove a tumour at when the anaesthetic wore off and she struck up a conversation with the medics still working on her."

Example 2.
(ground truth: "agree", system predicts: "unrelated")
Headline: Three Boobs Are Most Likely Two Boobs and a Lie

Document: The woman who claimed she had a third breast has been proved a hoax.

Example 3.
(ground truth: "disagree", system predicts: "discuss")
Headline: Woman pays 20,000 for third breast to make herself LESS attractive to men

Document: The woman who reported that she added a third breast was most likely lying.

Example 4.
(ground truth: "disagree", system predicts: "agree")

[8] https://code.google.com/archive/p/word2vec/
[9] https://stanfordnlp.github.io/CoreNLP/
[10] http://www.cis.upenn.edu/ ccb/ppdb/

Headline: Disgusting! Joan Rivers Doc Gwen Korovin’s Sick Selfie EXPOSED — Last Photo Of Comic Icon, When She Was Under Anesthesia

Document: If the bizarre story about Joan Rivers’ doctor pausing to take a "selfie" in the operating room minutes before the 81-year-old comedienne went into cardiac arrest on August 29 sounded outlandish, that’s because it was.

Exploiting Syntactic Structures for Humor Recognition

Lizhen Liu
Information Engineering
Capital Normal University
Beijing, China
liz_liu7480@cnu.edu.cn

Donghai Zhang
Information Engineering
Capital Normal University
Beijing, China
dhzhang@cnu.edu.cn

Wei Song*
Information Engineering
Capital Normal University
Beijing, China
wsong@cnu.edu.cn

Abstract

Humor recognition is an interesting and challenging task in natural language processing. This paper proposes to exploit syntactic structure features to enhance humor recognition. Our method achieves significant improvements compared with humor theory driven baselines. We found that some syntactic structure features consistently correlate with humor, which indicate interesting linguistic phenomena. Both the experimental results and the analysis demonstrate that humor can be viewed as a kind of style and content independent syntactic structures can help identify humor and have good interpretability.

1 Introduction

Humor, as a human-specific attribute, plays an important role in human communication. In addition to the tremendous help for human social success, humor also has positive and far-reaching effects on human psychology and physical health (Martineau, 1972; Anderson and Arnoult, 1989; Lefcourt and Martin, 1986). In recent years, the development of artificial intelligence has reinforced the human requirement for the intelligence of machines. As one of the qualities that embodies human wisdom, humor has attracted wide interests and attention (Mihalcea and Strapparava, 2005; Friedland and Allan, 2008; Zhang and Liu, 2014; Yang et al., 2015). The establishment of humor understanding mechanism promotes the development of language intelligence.

With the encouragement of exploring the essence of humor, great progress has been made in the research of humor theories in recent decades. Well recognized theories including superiority theory (Gruner, 1997), relief theory (Rutter, 1997) and incongruity theory (Suls, 1972) have been successively put forward, which explain the origin and essence of humorous feelings. Inspired by these theories, many computational methods are designed to model humor and make some achievements (Mihalcea and Strapparava, 2005; Friedland and Allan, 2008; Zhang and Liu, 2014; Yang et al., 2015).

Although many linguistic cue features have been studied, one aspect is often ignored that humor is a kind of style as well. An interesting question is whether we can explain humor from the perspective of styles. In this paper, we attempt to answer this question by exploring the syntactic structures, which are content independent and style related, for recognizing humor in text. The main contributions can be summarized as below:

(1) We extract production rules, dependency relations and statistics of structural elements as features to model humor. The syntactic structures significantly improve the performance of humor recognition.

(2) We demonstrate that some syntactic structures, which are differently distributed in humorous and non-humorous texts, indicate some interesting linguistic phenomena about humor.

2 Humor Recognition

The main goal of humor recognition is to judge whether the given text expresses humor (Mihalcea and Strapparava, 2005). It can be viewed as a typical classification task.

*corresponding author

Proceedings of the 27th International Conference on Computational Linguistics, pages 1875–1883
Santa Fe, New Mexico, USA, August 20-26, 2018.

Inspired by highly-recognized theories, many studies put forward interpretable features to model humor. We build the baseline features by following the work of (Yang et al., 2015). Next, we explain the features in detail.

Incongruity Structure. Incongruity theory (Suls, 1972) is a widely accepted theory. It believes that the core of humor is inconsistency or conflict. Following the work of (Yang et al., 2015), we describe incongruity through the following two features: (1) the largest semantic distance between word pairs in a sentence. (2) the smallest semantic distance between word pairs in a sentence.

Ambiguity. Another important way to create humor is semantic ambiguity (Miller and Gurevych, 2015), in which misunderstandings of author's intentions often produce unexpected feelings (Bekinschtein et al., 2011). In order to model the semantic ambiguity, we use WordNet (Fellbaum and Miller, 1998) to obtain all senses of word w in current text t. Then the probability of producing semantic ambiguity will be calculated through $\log \prod_{w \in t} sense(w)$, where $sense(w)$ indicates the number of the meanings that word w has. In addition, we also compute the sense farmost and sense closest features which are described in (Yang et al., 2015) to measure ambiguity.

Interpersonal Effect. Interpersonal effect is considered to be closely related to humor (Zhang and Liu, 2014). Some studies suggest that the occurrence of emotions and subjectivity increases the possibility of humor, so we use the resources in (Wilson et al., 2005) to build the following features: (1) the number of words with positive and negative polarity. (2) the number of subjective words.

Phonetic Style. According to the description in (Mihalcea and Strapparava, 2005), phonetic is also an important factor to create comedy effects. In this paper, we follow the work of (Mihalcea and Strapparava, 2005) and (Yang et al., 2015) to extract alliteration chains and rhyme chains in text by using CMU speech dictionary[1]. The alliteration chain is a set of words which have the same first phonemes and the words in rhyme chain have the same last syllable. The corresponding features are as follows: (1) the number of alliteration chains and rhyme chains. (2) the length of the longest alliteration chain and rhyme chain.

KNN features. In addition to humor-related features, we also use the semantic similarity as an indicator to build a KNN feature set that contains the labels of the top 5 sentences in the training data, which are closest to the target instance.

As described, most commonly used features are inspired by humor theories and linguistic knowledge. Few work researches syntactic styles of humorous expressions.

3 Exploiting Syntactic Structures for Humor Recognition

Syntactic information has been successfully used for analyzing text styles. We are going to exploit syntactic structures to reveal stylistic characteristics of humor. We use both constituent parsing and dependency parsing and derive features from parsed trees.

Consider the following sentence:

Never go to a doctor whose office plants have died.

The constituent tree is shown in Figure 1(a), while the dependency tree is shown in Figure 1(b). Both are provided by the Stanford parser (Klein and Manning, 2003) .

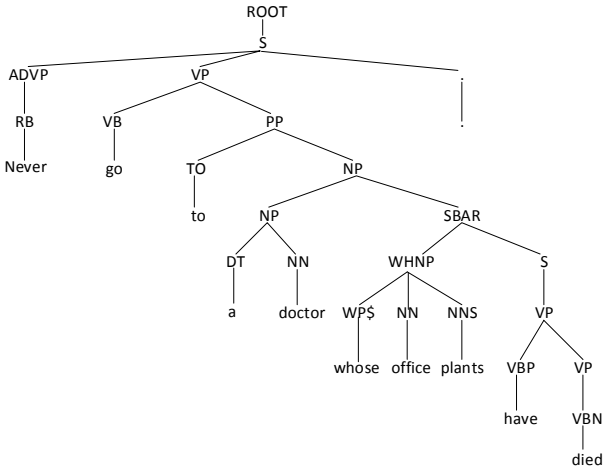
We mainly construct 3 types of features. Such features indicate statistics of basic structural elements and their relations.

3.1 Statistics of Structural Elements

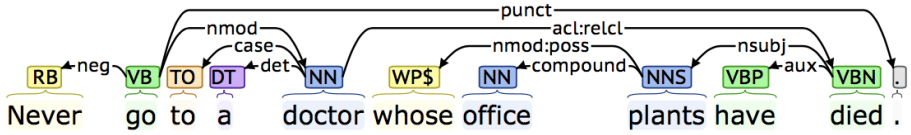
Statistics of structural elements on constituent have been proven to be effective in evaluating the linguistic quality of text (Nenkova et al., 2010). We borrow some features for human recognition.

- Complexity Metrics: Humorous texts and non-humorous texts may differ in the way they express intentions. We expect to measure the differences from the perspective of sentence complexity. Therefore, the number of noun phrases($NPcount$), verb phrases($VPcount$), prepositional

[1] http://www.speech.cs.cmu.edu/cgi-bin/cmudict



(a) a constituent tree.



(b) a dependency tree.

Figure 1: Examples of syntactic parsing.

phrases($PPcount$) and subordinating conjunctions($SBARcount$) are counted respectively as features.

- Phrase Length Ratio: The length ratio(LR) of PP, NP, VP is computed respectively. For each phrase type, the value of its length ratio equals the number of words in the phrase divided by the length of the sentence.

- Average Phrase Length: The average length of a phrase is calculated by dividing the number of words per phrase by the number of corresponding phrase types. It is worth noting that there are two different ways to calculate, one (APL_1) is to consider nested phrases. Consider a VP phrase (VP_1...(...VP_2...)), the length of VP is equal to *length*(VP_1)+ *length*(VP_2). Another way (APL_2) is to only consider the maximum length of the phrase, so the phrase length is *length*(VP_1).

- Normalized Phrase Length: This feature (NPL) just is considered for PP, NP and VP. The value is equal to the average length of each phrase type divided by the length of the sentence. The nested phrases are not considered in this situation.

- Phrase Ratio: Phrase ratio (PR), for each phrase type, is calculated by dividing the number of phrases that appear in the sentence by the length of the sentence.

- Ratio of PP or NP within a VP ($RPNV$): If a VP contains NP or PP, then this value is equal to the average length of the NP or PP divided by the length of the VP.

- Modifiers Change: There are several ways to modify the head noun. This feature is designed to model the modification of the head noun. Finally, two types of feature values are considered. One (MN) is the length of all the modified structures corresponding to the head noun. The other (NMN) is the normalization of the length of the modified structures.

3.2 Production Rules

Production rules describe the composition of phrases at all levels to form a complete sentence. We extract all production rules except the specific words at leaves of each instance as features.

The resulting features, corresponding to the example, include: *ROOT→S*, *S→ADVP VP .*, *ADVP→RB*, *VP→VB PP*, *PP→TO NP*, *NP→NP SBAR*, *NP→DT NN*, *SBAR→WHNP S*, *WHNP→*WP$ NN NNS, *S→*VP, *VP→VBP VP*, *VP→VBN*.

For a given production rule a, we set its feature value as $v(a) := count(a)/count(*)$, where $count(a)$ denotes the number of production rule a, and $count(*)$ represents the number of all production rules extracted from the current instance.

3.3 Dependency Relations

Compared with constituent parsing, dependency parsing indicates relation types between words. Thus, we directly design features for dependency relations. We build a feature for a dependency relation and normalize the count of the dependency relation by the number of all dependency relations in the sentence as the feature value.

We also attempt to combine part of speech (POS) tags and dependency relations. For the example shown in Figure 1(b), there is a dependency from *go* to *never* with type *neg*, therefore we construct a feature $VB \circ neg \circ RB$, where VB and RB are the POS tags of *go* and *never* respectively. But such features don't perform as well as using dependency relations only.

4 Experiment and Analysis

4.1 Data

The data includes humorous sentences as positive instances and non-humorous sentences as negative instances. The humorous sentences are from the work by (Mihalcea and Strapparava, 2005).

One-Liner includes10200 short humorous texts, which are crawled from the Web sites that containing the tags of $humor$, $humour$, $oneliner$, $one\text{-}liner$, $funny$ and $joke$.

They also provide a set of non-humorous short texts.

RPBN contains about 10000 sentences from Reuters titles, Proverbs and British National Corpus.

We found that the average number of tokens in a sentence is 9 in PRBN and 12 in One-Liner. Since we used statistics of structural elements and relations as features, such bias may affect the evaluation, although we have normalized most features. To reduce artifacts and increase the data diversity, we also extracted sentences from the newswire dataset provided by CoNLL-2012 for the task of coreference resolution (Pradhan et al., 2012). These sentences are samples from the OntoNotes corpus (Hovy et al., 2006). We filtered out the texts shorter than 5 words to reduce data noise. We finally sampled 10000 sentences to form a set of negative instances. Their average sentence length is 16.

We conducted experiments on three datasets. All three datasets use One-Liner as the positive instances. The first data set uses RPBN as negative instances, noted as RPBN as well; the second dataset uses sentences extracted from OntoNotes as negative instances, noted as CoNLL and the third dataset was built by sampling negative instances from both RPBN and CoNLL, half from RPBN and half from CoNLL, to keep a balanced positive and negative instance distribution, noted as Mixed.

Random Forest in Scikit-learn (Pedregosa et al., 2011) is used as the classifier. We ran 10-fold cross-validation on the datasets and the average performance would be reported.

4.2 Baselines

- Humor Theory driven Features (HTF). This method uses all features described in Section 2 except the KNN features. Thus all the used features are motivated by the humor theories. These features don't depend on specific content. We expect to use these features to capture the nature of humor.

- Human Centric Features (HCF). The method is a complete re-implementation of the proposed method in (Yang et al., 2015), which uses all features described in Section 2, i.e., HTF plus the KNN features.

Feature	RPBN				CoNLL				Mixed				
	Acc.	P	R	F_1	Acc.	P	R	F_1	Acc.	P	R	F_1	Average ΔF_1
HTF	0.709	0.702	0.749	0.725	0.801	0.771	0.862	0.814	0.707	0.683	0.793	0.734	—
HCF	0.786	0.777	0.816	0.796	0.908	0.889	0.935	0.911	0.821	0.797	0.87	0.832	—
Word2Vec	0.772	0.777	0.778	0.777	0.894	0.881	0.915	0.897	0.799	0.79	0.825	0.807	—
HCF+Word2Vec	**0.81**	0.81	0.821	**0.815**	**0.915**	0.90	0.937	**0.918**	**0.825**	0.801	0.872	**0.835**	—
HTF+Syntactic	0.787	0.768	0.834	0.800	0.871	0.846	0.912	0.878	0.798	0.777	0.847	0.810	+7.2%
HCF+Syntactic	0.814	0.794	0.858	**0.825**	**0.922**	0.91	0.939	**0.925**	**0.850**	0.827	0.891	**0.858**	+2.3%
Word2Vec+Syntactic	0.797	0.798	0.807	0.803	0.902	0.886	0.926	0.906	0.806	0.801	0.822	0.811	+ 0.4%
HCF+Word2Vec+Syntactic	**0.817**	0.813	0.832	0.823	0.915	0.90	0.936	0.918	0.837	0.822	0.866	0.844	+0.6%

Table 1: Humor recognition performance of four baselines and their combinations with syntactic structure features. HTF: Humor Theory driven Features, HCF: HTF plus KNN features.

- Word2Vec. The method makes use of the semantic representation of the sentences by using the averaged word embeddings as features. Similar to KNN features, it is also sensitive to content. We used the pre-trained word embeddings that were learned using the Word2Vec toolkit (Mikolov et al., 2013) on Google News dataset.[2]
- HCF+Word2Vec. The method combines HCF and Word2Vec. Since the combination of two strong methods, it achieved best evaluation scores in (Yang et al., 2015). It is more dependent on content, since both KNN features and Word2Vec features are used.

4.3 Results

The results are shown in Table 1 with accuracy ($Acc.$), precision (P), recall (R) and F_1 score. To verify the effectiveness of proposed features, we add syntactic structure features (Syntactic) to the four baselines respectively. The performance can be improved to varying degrees.

We can see that after adding syntactic features, HTF achieves significant improvements on all datasets, with an average improvement of 7.9% in accuracy and 7.2% in F_1 score. This indicates that syntactic structure features can complement humor theory driven features and benefit humor recognition. Syntactic structure features don't depend on specific content of texts and their performance is consistent on different datasets, whose sentences have different distribution of the number of tokens. It means that syntactic features can capture some key properties of humor.

Other baselines all consider content features and their performance is greatly superior to HTF. However, such great improvements may come from the artifacts in the datasets rather than capture the nature of humor. The non-humorous samples in our experiments contain news titles, which may involve different vocabularies compared with humorous samples. It is highly probable that these models match specific content and topics better. Even so, after adding syntactic features, all these baselines still achieve improvements, although the margins become small.

Feature	RPBN				CoNLL				Mixed			
	Acc.	P	R	F_1	Acc.	P	R	F_1	Acc.	P	R	F_1
HTF	0.709	0.702	0.749	0.725	0.801	0.771	0.862	0.814	0.707	0.683	0.793	0.734
HTF+DR	0.774	0.756	0.823	0.788	0.868	0.837	0.917	0.875	0.783	0.755	0.848	0.799
HTF+PR	0.783	0.775	0.81	0.793	**0.877**	0.856	0.91	**0.882**	0.797	0.784	0.83	0.806
HTF+SE	0.772	0.75	0.832	0.789	0.869	0.850	0.901	0.875	0.79	0.761	0.855	0.805
HTF+Syntactic	**0.787**	0.768	0.834	**0.800**	0.871	0.846	0.912	0.878	**0.798**	0.777	0.847	**0.810**

Table 2: Contributions of individual syntactic structure feature types. HTF: Humor theory driven features, DR:dependency relations, PR: production rules , SE: statistics of syntactic elements, Syntactic: DR+PR+SE.

In addition to reflecting the effects of the syntactic structure features as a whole, Table 2 shows the results of adding individual syntactic structure features on the basis of HTF. We can see that all three

[2] https://code.google.com/archive/p/word2vec/

Production Rules	Pearson	One-Liner	RPBN
S→NP VP	0.236	51%	22%
ROOT→S	-0.229	75%	74%
NP→PRP	0.209	59%	29%
ROOT→NP	-0.188	10%	18%
SBAR→IN S	0.173	20%	6%
WHADVP→WRB	0.150	13%	3%
SBAR→S	0.138	16%	6%
ROOT→SBARQ	0.136	7%	0.8%
NP→NP SBAR	0.120	17%	7%
SBAR→WHADVP S	0.112	10%	3%

(a) Discriminative production rules in One-Liner and RPBN non-humorous dataset

Production Rules	Pearson	One-Liner	CoNLL
NP→PRP	0.352	59%	25%
S→NP VP	0.207	51%	37%
WHADVP→WRB	0.193	13%	3%
NP→NNP NNP	-0.179	4%	8%
ROOT→SBARQ	0.166	7%	0.4%
NP→NP SBAR	0.163	17%	8%
SBAR→WHADVP S	0.150	10%	3%
WHNP→WP	0.148	11%	3%
PP→IN NP	-0.136	55%	81%
SBAR→IN S	0.097	20%	15%

(b) Discriminative production rules in One-Liner and CoNLL non-humorous dataset

Table 3: The top discriminative production rules and their distributions in corresponding humorous and non-humorous datasets, sorted by Pearson correlation coefficient. Rules in bold have the same correlation trend in Table 3(a) and Table 3(b).

types of syntactic structure features improve the performance on three datasets. Generally, their contributions are close. Production rule features perform slightly better. This may mean that different kinds of syntactic representations have similar effect. Next, we analyze the three types of syntactic structure features respectively.

Statistics of Structural Elements. The performance of features related to the statistics of structural elements is fairly good on three datasets. We calculate the Pearson correlation coefficient between the specific features and humor on RPBN and CoNLL datasets. We found that not all structural elements have the same correlation on two datasets but some are consistently discriminative. The features that have best positive correlations are the phrase ratio of verb phrases (PR_VP) and the number of subordinating conjunctions (SBAR_Count). In contrast, the features that have consistent negative correlations are the length ratio (LR_NP) and the average length of noun phrases (APL_2_NP).

Production Rules. As shown in Table 2, production rule features can lead to great improvements. We also compute the Pearson correlation coefficient between production rules and humor and the ratio of each production rule in humorous instances or in non-humorous instances of two datasets. Table 3 shows the results. The production rules are ranked according to the correlation coefficient. We can see that some rules are discriminative on both datasets such as *S→NP VP*, *SBAR→IN S* and *WHADVP → WRB*. Their distributions in humorous and non-humorous texts are quite different. This analysis demonstrates that syntactic structures can provide useful information for distinguishing humorous texts from non-humorous texts.

Dependency Relations. Table 4 shows the discriminative power of dependency relations with Pearson correlation coefficient. We can see that the most discriminative features are related to the dependency relation *nsubj*, *compound*, *case*, *aux*. The relation *case* and *compound* are less in humorous instances. The relation *nsubj* together with pronouns (*PRP*) occurs more in humorous instances, while the relation *nsubj* together with noun phrases are less in humorous instances.

Generally, three types of features describe syntactic features from different angles so that their contributions have overlap but also supplement each other in certain degree. One interesting question is that do these features indicate some linguistic interpretation? We attempt to discuss it in next part.

4.4 Linguistic Interpretation

The experimental results show that our proposed syntactic structures do help to identify humor. In this section, we explain how these syntactic structures relate to linguistic phenomena. We found that humorous texts have the following characteristics that can be explained by the syntactic features.

1) Humorous texts use simpler words but more complex syntactic structures.

We can see that compound phrases appear much less in humorous texts. This can be revealed by the average length of noun phrases and the dependency relation *compound*, which have negative correlations

Dependency Relations	Pearson	One-Liner	RPBN
NNP*∘ *compound*∘ *NNP	-0.167	8%	17%
VBP*∘ *nsubj*∘ *PRP	0.163	18%	5%
VBZ∘ *nsubj*∘ *NNP*	-0.138	2%	7%
NNS∘ *compound*∘ *NNP*	-0.127	2%	6%
VB*∘ *nsubj*∘ *PRP	0.121	16%	6%
VB*∘ *aux*∘ *VBP	0.114	7%	2%
VBZ∘ *dobj*∘ *NN*	-0.112	5%	8%
VBP∘ *mark*∘ *IN*	0.105	6%	2%
NNP*∘ *case*∘ *IN	-0.104	5%	9%
VBP∘ *advmod*∘ *WRB*	-0.104	4%	0.5%

(a) Discriminative dependency relations in One-Liner and RPBN

Dependency Relations	Pearson	One-Liner	CoNLL
VBP*∘ *nsubj*∘ *PRP	0.229	18%	3%
VB*∘ *nsubj*∘ *PRP	0.220	16%	3%
NNP*∘ *case*∘ *IN	-0.203	5%	25%
NNP*∘ *compound*∘ *NNP	-0.190	8%	34%
VB∘ *neg*∘ *RB*	0.174	12%	4%
NNS∘ *amod*∘ *JJ*	-0.165	10%	32%
CD∘ *compound*∘ *CD*	-0.161	0.1%	7%
VB*∘ *aux*∘ *VBP	0.160	7%	0.8%
NNS ∘ *case*∘ *IN*	-0.151	15%	36%
NN∘ *nmod:poss*∘ *PRP$*	0.142	15%	8%

(b) Discriminative dependency relations in One-Liner and CoNLL

Table 4: The top discriminative dependency relations and their distributions in corresponding humorous and non-humorous datasets, sort by Pearson correlation coefficient. Dependency relations in bold have the same correlation trend in both datasets.

with humor. The reason may be that many jokes are about events in life so that common words are more often used, while complex and professional words are less used. In contrast, humorous texts often have subordinate conjunctions, which means that subordinate clauses are often used. This can be seen in features involving $SBAR$, such as *NP→NP SBAR*, *SBAR→WHADVP S* and *SBAR→IN S*, all have positive correlations with humorous instances. Considering the following sentence,

I've learned that we are responsible for what we do, $unless$ we are celebrities.

Here, subordinated conjunction $unless$ is used to bring an $unexpected$ feeling to the readers, which results in comedy effect. To break the expectation, complex syntactic structures are often utilized.

2) Humorous texts are more vivid and specific.

We can see that aux relation appears much more in humorous instances as shown in Table 4. This is because humorous texts usually describe some details to let reader imagine the situation, so that auxiliary words are used more to enhance such descriptions.

In addition, aux often related to negation. For example, the feature $VB \circ aux \circ VBP$ mostly describes "do/don't + verb" pattern. The use of negation is usually related to an attitude change or contrast effect, which may lead to humor.

We also see that *WHADVP→WRB* and *SBAR→WHADVP S* have a positive correlation. This is because interrogative adverb why are often used to produce a satirical effect, such as

If ignorance is bliss, why aren't more people happy?

Such rhetoric questions are useful to enhance the expressiveness of a sentence.

3) Humorous texts are more like conversations.

We can see that personal pronouns often appear in humorous texts, including first-person, second-person and third-person pronouns. The corresponding features include the production rule *NP→PRP* and the dependency relation $VBP \circ nsubj \circ PRP$. These words appear more often in conversations. Besides, the rhetoric questions as we discussed earlier also build an effect like conversation.

This phenomenon can be explained by superiority theory (Keith-Spiegel, 1972; Gruner, 1997) that humor is the result of comparing with others. As a result, it is unavoidable to mention people with pronouns. Also, many jokes involve dialogues between people so that they are naturally a kind of conversation.

5 Related Work

Much existing work models humor according to psychological theories. For example, in early studies, Taylor and Mazlack (2004) proposed a computational approach to identify humor based on the humor theory of Raskin (1984). This method took into account the constraint set of all possible jokes, which

took wordplay as a component. The algorithm used in their method learned the statistical patterns of text in N-grams, which provided a heuristic focus for the location of wordplay. Purandare and Litman (2006) identified humor by modeling the distinction of prosodic characteristics between humorous and non-humorous speech by utilizing acoustic-prosodic and linguistic features. Mihalcea and Pulman (2009) presented and analyzed the most frequent features including human-centeredness and negative polarity, which solved two questions related to humor recognition: One is whether humorous and non-humorous texts are separable. Another is whether the characteristics of humor are special.

Recently, Zhang and Liu (2014) designed features according to influential humor theories, linguistic rules and affective dimensions. These features can be applied to distinguish humorous tweets from non-humorous tweets. Motivated by humor theories, Yang et al. (2015) modeled humor by using the semantic structures including incongruity structure, ambiguity, interpersonal effect and phonetic styles. We used their method as one of our baselines. A recent work (Liu et al., 2018) studied sentiment association in discourse for humor recognition.

Some work considers content features. Mihalcea and Strapparava (2005) reported that using content-based features could achieve big improvements. However, content features may overfit the dataset, because in most existing work, non-humorous instances are sampled according to some assumptions which may bring in artifacts. Our work describes humorous expressions by exploring syntactic structures and treats humor as a kind of style. The syntactic structure features don't depend on specific content or topics.

6 Conclusion

In this paper, we have presented a method for humor recognition by exploiting syntactic structure features. We derive features from parsing trees and demonstrate that such features can improve the performance of humor recognition. Moreover, they are not dependent on specific content, which reduces the risk of satisfying artifacts during dataset construction and helps capture the essence of humor.

We analyzed the linguistic insights behind the features and found that humorous texts tend to 1) use simple words but with complex sentence structures; 2) be more vivid with auxiliary adverbs, negations and rhetoric questions; 3) be like conversations, involving more personal pronouns and questions. These observations indicate stylistic characteristics of humor and provide an opportunity to humor computation in a new perspective.

Acknowledgements

The research work is funded by the National Natural Science Foundation of China (No.61402304), Beijing Municipal Education Commission (KM201610028015, Connotation Development) and Beijing Advanced Innovation Center for Imaging Technology.

References

Craig A. Anderson and Lynn H. Arnoult. 1989. An examination of perceived control, humor, irrational beliefs, and positive stress as moderators of the relation between negative stress and health. *Basic and Applied Social Psychology*, 10(2):101–117.

T. A. Bekinschtein, M. H. Davis, J. M. Rodd, and A. M. Owen. 2011. Why clowns taste funny: the relationship between humor and semantic ambiguity. *Journal of Neuroscience the Official Journal of the Society for Neuroscience*, 31(26):9665.

C Fellbaum and G Miller. 1998. *WordNet:An Electronic Lexical Database*. MIT Press.

Lisa Friedland and James Allan. 2008. Joke retrieval: recognizing the same joke told differently. In *ACM Conference on Information and Knowledge Management*, pages 883–892.

Charles R Gruner. 1997. *The game of humor: A comprehensive theory of why we laugh.* Transaction PUblishers.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 57–60.

Patricia Keith-Spiegel. 1972. Early conceptions of humor: Varieties and Issues. *Psychology of Humor*, pages 3–39.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Meeting of the Association for Computational Linguistics*, pages 423–430.

Herbert M. Lefcourt and Rod A. Martin. 1986. *Humor and Life Stress*. Springer New York.

Lizhen Liu, Donghai Zhang, and Wei Song. 2018. Modeling sentiment association in discourse for humor recognition. *Association of Computational Linguistics*.

William H Martineau. 1972. *A Model of the Social Functions of Humor*. The Psychology of Humor.

Rada Mihalcea and Stephen Pulman. 2009. Characterizing humour: An exploration of features in humorous texts. In *International Conference on Computational Linguistics and Intelligent Text Processing*, pages 337–347.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: investigations in automatic humor recognition. In *Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 531–538.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tristan Miller and Iryna Gurevych. 2015. Automatic disambiguation of english puns. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 1:719–729.

Ani Nenkova, Jieun Chae, Annie Louis, and Emily Pitler. 2010. Structural features for predicting the linguistic quality of text - applications to machine translation, automatic summarization and human-authored text. In *Empirical methods in natural language generation*, pages 222–241.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40.

Amruta Purandare and Diane Litman. 2006. Humor: prosody analysis and automatic recognition for f*r*i*e*n*d*s*. In *Conference on Empirical Methods in Natural Language Processing*, pages 208–215.

Victor Raskin. 1984. *Semantic Mechanisms of Humor*. Springer Science & Business Media.

Jason. Rutter. 1997. Stand-up as interaction : performance and audience in comedy venues. *University of Salford*, 33(4):1 – 2.

Jerry M. Suls. 1972. A two-stage model for the appreciation of jokes and cartoons: An information-processing analysis. *The Psychology of Humor*, 331(6019):81–100.

Julia M. Taylor and Lawrence J. Mazlack. 2004. Computationally recognizing wordplay in jokes. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 26.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.

Diyi Yang, Alon Lavie, Chris Dyer, and Eduard Hovy. 2015. Humor recognition and humor anchor extraction. In *Conference on Empirical Methods in Natural Language Processing*, pages 2367–2376.

Renxian Zhang and Naishi Liu. 2014. Recognizing humor on twitter. In *ACM International Conference on Conference on Information and Knowledge Management*, pages 889–898.

An Attribute Enhanced Domain Adaptive Model for Cold-Start Spam Review Detection

Zhenni You [1], **Tieyun Qian** [1*], **Bing Liu** [2]
[1] School of Computer Science, Wuhan University, Wuhan, China
[2] Department of Computer Science, University of Illinois at Chicago, IL, USA
[*] Contact author
{znyou,qty}@whu.edu.cn, liub@uic.edu

Abstract

Spam detection has long been a research topic in both academic and industry due to its wide applications. Previous studies are mainly focused on extracting linguistic or behavior features to distinguish the spam and legitimate reviews. Such features are either ineffective or take long time to collect and thus are hard to be applied to cold-start spam review detection tasks. Recent advance leveraged the neural network to encode the textual and behavior features for the cold-start problem. However, the abundant attribute information are largely neglected by the existing framework.

In this paper, we propose a novel deep learning architecture for incorporating entities and their inherent attributes from various domains into a unified framework. Specifically, our model not only encodes the entities of reviewer, item, and review, but also their attributes such as location, date, price ranges. Furthermore, we present a domain classifier to adapt the knowledge from one domain to the other. With the abundant attributes in existing entities and knowledge in other domains, we successfully solve the problem of data scarcity in the cold-start settings. Experimental results on two Yelp datasets prove that our proposed framework significantly outperforms the state-of-the-art methods.

1 Introduction

Online reviews and ratings are playing more and more critical roles in E-Commerce places. The Mintel flagship report showed that 69 percent of Americans seek out others' advice online before making a purchase [1]. This gives strong incentives for imposters to game the system. As a result, fraudulent reviews flood the e-market websites. Forbes news [2] reported that Amazon's fake review problem is now worse than ever in 2017. Fake reviews are perceived as threats to the ecosystem of the e-business sites, companies, and users, and thus it becomes an urgent task to detect fake or spam reviews.

Existing approaches in spam review detection mainly focused on exacting linguistic features and behavioral features. However, linguistic features are ineffective when they are used to detect the real-life fake reviews (Mukherjee et al., 2013b; Wang et al., 2017b), and it usually requires a large number of samples to make the observations on behavior features. When dealing with the cold-start problem, i.e., *a review is just posted by a new reviewer*, it is hard to construct effective behavioral features for the new reviewer. More recently, Wang et al. (2017b) proposed a neural network model to jointly encode the textual and behavioral information into the review embeddings. This was a good try. However, due to the lack of sufficient information, the results reported in (Wang et al., 2017b) are not promising enough (with an accuracy less than 65%).

[1] http://store.mintel.com/american-lifestyles-2015-the-connected-consumer-seeking-validation-from-the-online-collective-us-april-2015

[2] https://www.forbes.com/sites/emmawoollacott/2017/09/09/exclusive-amazons-fake-review-problem-is-now-worse-than-ever/#501eccb87c0f

Proceedings of the 27th International Conference on Computational Linguistics, pages 1884–1895
Santa Fe, New Mexico, USA, August 20-26, 2018.

Our study is inspired by the work (Wang et al., 2017b) but moves one step further, i.e., we incorporate not only the entities, but also the attributes into a unified framework. The rationale is that similar reviewers' comments on items with similar attributes will result in similar review contexts. For example, the review for luxury restaurants may focus on their atmosphere and personal services while for ordinary restaurants one cares about their tastes. Here the luxury or ordinary is the key property of an restaurant, and might be reflected by the *price range* attribute.

An attribute is the inherent characteristic of an entity and is available for all users (reviewers), items or reviews no matter whether it is in a cold-start setting. For example, when a new user joins a website, the location is automatically recorded as his/her attribute. Such a property makes attributes extremely suitable for our cold-start spam detection problem. Furthermore, while an entity may be new, its attributes may not be new. For example, the attributes for a hotel like "price range" or "having wifi or not" must have been appeared in many other hotels' attributes. Hence we can piece together the profile of new users, items, and reviews by using attributes from others.

To further alleviate the data scarcity in cold-start spam review detection, we turn to seek more available information from other domains. The basic idea is that the entities in different domains may have same attributes. For example, both a hotel and a restaurant have an attribute of "price range". Hence we aim to leverage the shared attributes across different domains. The challenge of this task arises from the discrepancy between two domains. For example, a price of 200 USD may be high for a restaurant but it is low for a hotel. Inspired by prior work on domain adaption (Ben-David et al., 2007; Ben-David et al., 2010; Tzeng et al., 2015), we extend the idea of knowledge transferring between domains to learn a generalized representation for the entity.

Based on the above analysis, we propose an attribute-enhanced domain adaptive (AEDA) embedding model in this paper. Our model not only exploits the inherent attributes of reviewers, items and reviews, but also captures the domain correlations. In summary, the main contributions of our work are as follows.

- We propose to leverage the attributes of entities and their relations to enhance the representations of entities. These attributes are inherent for the entities and thus do not need any experts' experience or time-consuming procedure to collect.
- we present a novel neural network to jointly encode the attributes, entities, and their relations, as well as to adapt the knowledge from one domain to the other. The abundant information greatly alleviate data scarcity problem in the cold-start scenario of spam review detection.
- Extensive experiments demonstrate that our model significantly outperforms the state-of-the-art baseline methods.

The rest of the paper is structured as follows. In Section 2, we present the related work. In Section 3, we introduce our methodology. In Section 4, we show the experimental evaluation. We conclude the paper in Section 5.

2 Related Work

2.1 Review Spam Detection

The problem of spam review detection has aroused great research interests in recent years. This problem was first introduced in (Jindal and Liu, 2008), and the focus of this work was to find effective features to represent the fake and real reviews. Later, a variety of linguistic features were introduced in the literature (Ott et al., 2011; Xu and Zhao, 2013; Harris, 2012; Feng et al., 2012a; Kim et al., 2015; Li et al., 2013; Li et al., 2014b; Fornaciari and Poesio, 2014; Li et al., 2014a; Hovy, 2016). However, an in-depth study (Mukherjee et al., 2013b) found that the linguistic features were insufficient for detecting fake reviews in real business website. Therefore, the features besides review contents were exploited, and the behavioral features of reviewers were intensively studied in (Lim et al., 2010; Jindal et al., 2010; Feng et al., 2012b; Mukherjee et al., 2012; Xie et al., 2012; Fei et al., 2013; Li et al., 2015; KC and Mukherjee, 2016; Wang et al., 2011; Akoglu et al., 2013; Mukherjee et al., 2013a; Mukherjee et al., 2013b). The intuition is that the reviewers with spammer-like behaviors are more likely to post fake reviews. Several methods (Li

et al., 2011; Rayana and Akoglu, 2015) were proposed to utilize multiple information mentioned above. Overall, the traditional features are manually constructed and depend heavily on the experts' knowledge.

More recently, deep learning techniques are applied to fake review detection. Ren and Zhang (Ren and Zhang, 2016) compared several neural networks and found that CNN was more effective than RNN on review text encoding in spam review detection task. Hai et al. (2016) implemented a semi-supervised multi-task learning method, which introduced a covariance matrix to capture the relation between tasks and a Laplacian regularizer to leverage the unlabeled data. Wang et al. (2016) employed a tensor decomposition based on the global behavioral information to learn the representation of the reviewer and item.

The cold-start problem in spam review detection was first introduced in (Wang et al., 2017b), where the authors proposed an embedding learning model to jointly utilize the behavioral information of reviewers and the textual information. While we aim to solve the same cold start problem as that in (Wang et al., 2017b), we propose a totally different framework which leverage both the new attribute and domain knowledge information.

2.2 Domain Adaption

Domain adaption has been applied to addressing the scarcity of labeled data in a variety of real-world applications. It has been well studied in many tasks in the area of computer vision, such as the image classification and object recognization. These methods mainly differ in network structure or objectives, including parameter sharing (Yosinski et al., 2014; Liu et al., 2017), minimizing the distance between source and target distributions (Baktashmotlagh et al., 2013; Tzeng et al., 2014; Long et al., 2015), maximizing of the loss of domain classifier (Ganin and Lempitsky, 2015; Tzeng et al., 2015; Tzeng et al., 2017; Gopalan et al., 2011), and applying semi-supervised or selective learning techniques (Ao et al., 2017; Tan et al., 2017).

Recent years also witnessed the applications of domain adaption technique in many other areas, especially in natural language processing tasks. To name a few, typical applications include question answering (Min et al., 2017), machine translation (Chu et al., 2017; Johnson et al., 2016; Wang et al., 2017a), sentiment analysis (Li et al., 2017), recommendation system (Man et al., 2017) and image-text retrieval (Huang et al., 2017).

While domain adaption technique was introduced in the above research fields, it was rarely adopted in spam review detection. We aim to exploit this technique to help alleviate the problem of data scarcity in the cold-start scenario of spam detection.

3 Methodology

In this section, we present the details of our attribute-enhanced domain adaptive embedding model (AEDA). The key idea is to leverage the relations between entities and attributes and adapt knowledge from one domain to the other. Motivated by recent advances in deep learning which has been proven to be powerful in learning nonlinear representations, we design a novel deep neural network to jointly encode the rich information in entities, attributes, and domains.

3.1 Attribute Enhanced Objective

When a reviewer comments on an item (or product), a review is posted along with a rating score. This process involves three kinds of entities: reviewers, items, and reviews. All these entities consist in a number of attributes. For example, when an item, e.g., a newly opened restaurant, first registers on Yelp, its location and other attributes are recorded in the website. The attributes are essential for legitimate users to learn about the entity. For example, a user on a business trip may care about whether the hotel is close to a metro station.

The attributes provide additional information of the entity and form the background of the comment action. Such information and background are particularly useful when dealing with the cold start spam review detection problem. Since these attributes are inherent components of an entity, we can simply piece together the profile for a new entity using the attributes in existing entities of the same type. Hence

we collect all the attributes and aim to incorporate them into our framework. Below is a brief introduction to these attributes.

- **Reviewer attributes** are recorded when the reviewer registers on the review system. The *YelpJoinDate* is the date when the reviewer joins the system, while the *Location* indicates where the reviewer comes from, which is filled by him/herself.
- **Item attributes** reflects the basic information of each item, such as its *Location*, *Average Rating*, *Price Ranges* (from 1 to 5). The *AcceptCreditCards*, *WiFi*, *WebSite*, and *PhoneNumber* are the attributes about whether the item provides the specific service or not. We convert them into boolean values like "hasWifi" or "ifAcc".
- **Review attributes** include the *Date* when the review is posted and the *Rating* score the reviewer rates on the item.

Overall, there are three kinds of entities and eight kinds of attributes, which further form three types of relations: entity-attribute, attribute-attribute, and entity-entity. We aim to leverage these relations between the attribute and the entity. We will give details below.

Relation 1: entity-attribute relation In order to exploit the rich information contained in the entity-attribute relation, we propose an attribute-enhanced objective L_{ea} to maximize the conditional probability $P(v|e)$ of an entity e's attribute a with a value of v. The intuition is that the attributes can be treated as the contexts of the entities, and the attribute information can be encoded into the representations of entities. We formalize it as follows:

$$\begin{aligned} L_{ea} &= P(v|e) \\ P(v|e) &= \frac{exp(V_v \cdot V_e)}{\sum_{u \in A} exp(V_u \cdot V_e)} \end{aligned} \tag{1}$$

where V_i indicates the embedding of i, including entities and attributes, and A is the corresponding value set of attribute a. Note that we employ negative sampling (Mikolov et al., 2013) to transform the objective. By maximizing the attribute-enhanced objective L_{ea}, the entities with similar contexts (attributes) tend to be similar and the attributes of similar entities become associated. In this way, the representations of attributes become informative and we obtain the attribute-enhanced entity embeddings.

Relation 2: attribute-attribute relation Several attributes, i.e., date, location, and rating, are shared by different entities. In order to capture the relation between these attributes, we take their difference as a new feature. Specifically, we subtract *Date* (review) from *YelpJoinDate* (reviewer) as the new *dateDif* feature, and subtract *Rating* (review) from *Average Rating* (item) as the new *ratingDif* feature, and treat the new *locDif* feature as 1 when the reviewer and item have same location otherwise 0.

Relation 3: entity-entity relation While the entity-attribute and attribute-attribute relations are proposed by us, the entity-entity relation has been examined before (Wang et al., 2017b). For a fair comparison, we follow the work of (Wang et al., 2017b) by applying the TransE model on our attribute-enhanced entity embedding. More formally, we propose an entity-interacted objective L_{ee} as follows:

$$\begin{aligned} L_{ee} = \sum_{(i,r,t) \in S} \sum_{(i',r,t') \in S'} & max\{0, 1 + d(i+r,t) - d(i'+r,t')\} \\ d(i+t,c) =& ||V_i + V_r - V_t||_2^2, \\ s.t.\ ||V_i||_2^2 =& ||V_r||_2^2 = ||V_t||_2^2 = 1 \end{aligned} \tag{2}$$

where S is a set of triples (i, r, t) in the training set of reviews, including the item $i \in I$ (item set) which the review talk about, the reviewer $r \in R$ (reviewer set) who posts this review, and the review text $t \in T$ (review set). S' denotes the corrupted set which is constructed by replacing the item *i* (or text *t*) with a random chosen i' (or t').

The entity-entity and entity-attribute relation can be encoded into the embedding of the entities and attributes in our model by the proposed attribute-enhanced and entity-interacted objective. To make full of them, we concatenate the review text embedding generated by CNN and all the attribute embeddings

of item, reviewer and review into a long vector V_{rcon} to represent the review. This is our basic attribute enhanced (AE) model. The representations of reviews can then be used as the features to train a spam review classifier and make a classification of the new reviews.

3.2 Domain Adaptive Objective

Our AE model is trained in the single domain, i.e., all the embeddings are trained using data merely from one hotel or restaurant domain. As proved in (Hai et al., 2016), the detection of spam review in different domains may have strong correlations. We are curious about whether the cold-start problem can be alleviated by borrowing knowledge from other domains. This is reasonable because entities in different domains may have same attributes. For example, both hotel and restaurant have an attribute of "price range". If we can confuse the domain difference and focus on the relations of attributes from different domains, we may complement the attribute embeddings and subsequently enhance entity embeddings.

Motivated by the recent advances in domain adaption (Ben-David et al., 2007; Ben-David et al., 2010; Tzeng et al., 2015), we add a domain classifier to perform domain classification of reviews in the training set based on the review embedding V_{rcon} mentioned before. Then an adversarial loss is adopted to intensify the domain confusion.

Domain Classifier The domain classifier is implemented by adding a dense layer to identify which domain a review belongs to according to its representation V_{rcon}. We aim to minimize the domain label prediction loss L_{dcla} (defined as the cross-entropy between the predicted label and the true label) by updating the parameters of the domain classifier using a softmax function.

$$q = softmax(W \cdot V_{rcon} + b) \tag{3}$$

$$L_{dcla} = -\sum_{d} \mathbf{1}[y_{dtrue} = d] \, log \, q_d \tag{4}$$

where q is the predicted domain distribution, d is the index of the corresponding domain, y_{dtrue} is the true domain where the review comes from, and W and b are the parameters of the domain classifier.

Adversarial Loss The target of the domain confusion is to adjust the representation to become too domain-invariant to distinguish its domain label. To this end, we minimize the domain confusion loss L_{dcon}, which is defined as the cross entropy between domain label distribution predicted by the domain classifier and a uniform distribution, by updating all parameters in our model except those of domain classifier. We formalize it as follows:

$$L_{dcon} = -\sum_{d} \frac{1}{D} \, log \, q_d \tag{5}$$

where D is the number of the domain in our training set.

After the adversarial training, we can reach a point at which the review representation can not be classified into the true domain label by the best domain classifier. This is what our attribute enhanced domain adaptive (AEDA) model does for training the representations.

Although the domain correlation problem was introduced in (Hai et al., 2016), our task is completely different from that in (Hai et al., 2016). We deal with the domain adaption problem while Hai et al. (Hai et al., 2016) address the multi-task learning problem. In addition, their method was heavily dependent on the labeled data, while our method is totally unsupervised when adapting the knowledge from other domains.

3.3 Model Architecture and Learning Procedure

We present the architecture of our AEDA model in Figure 1. It consists of three layers. The bottom layer of the architecture integrates various attributes into the model. The middle layer aims to capture three relations, i.e., entity-entity, entity-attribute, and attribute-attribute relations. The top layer is used to implement a domain classifier to capture the domain correlation.

We now discuss the learning procedure of our AEDA model in Algorithm 1. It works as follows. Line 1 initializes the parameters. Most parameters of model are randomly initialized, with the exception that the review texts E_t which are generated from the CNN based on the pre-trained word embedding by CBOW

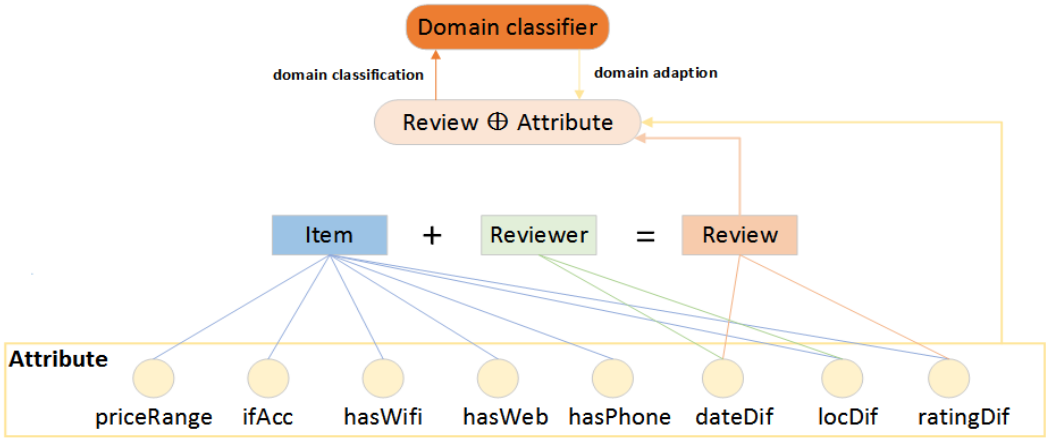


Figure 1: Architecture of AEDA model

(Mikolov et al., 2013). Then for each batch in the training set, we sequentially train entity-attribute relation in reviewer, item, and review, as shown in lines 4 to 6. Next, we use Eq.(2) to capture the entity-entity relation in line 7, which updates the parameters for the entities. Finally we minimize the Eq.(4) to train an accurate domain classifier, and minimize the Eq.(5) to confuse the review representations from different domains. Lines 8, 9 show this adversarial training procedure. After the model is properly trained, through the input of training sets and testing sets, we can obtain the representations of reviews in both sets. These embeddings of reviews can be fed into the traditional classification model like SVM to train the classifier and evaluate the performance of our model.

Algorithm 1 Attribute-enhanced domain adaptive (AEDA) model

Input: The combined training set of different domains, $X_{train} = \{x_i, y_{dtrue,i}\}_{i=1}^{n_{train}}$. Each x_i is composed of several parts including reviewer, item, review text, and their attributes, and $y_{dtrue,i}$ is the domain label of review i;

Output: The trained AEDA model

1: Initialize the parameters of the model, including embeddings of reviews E_t and their attributes E_{ta}, embeddings of reviewers E_r and their attributes E_{ra}, embeddings of items E_i and their attributes E_{ia}, the parameters of CNN Ψ_{tc} and those of domain classifier W, b
2: **repeat**
3: **for** each batch in X_{train} **do**
4: update E_r, E_{ra} by maximizing Eq.(1), where e is the reviewer, v is the value of reviewers' attributes.
5: update E_i, E_{ia} by maximizing Eq.(1), where e is the item, v is the value of items' attributes.
6: update E_t, Ψ_{tc}, E_{ta} by maximize Eq.(1), where e is the review generated from CNN, v is the value of reviews' attributes.
7: update E_r, E_i, E_t, Ψ_{tc} by minimizing Eq.(2)
8: update W, b by minimizing Eq.(4)
9: keep W, b fixed and update all other parameters in the model by minimizing Eq.(5)
10: **end for**
11: **until** converge or reach the predetermined number of epoches
12: **return** the trained AEDA model

4 Experiments

4.1 Experimental Settings

We verify the effectiveness of our proposed model on two cold-start datasets (Wang et al., 2017b), which are the subset of the Yelp datasets used in a number of previous studies (Mukherjee et al., 2013b; Rayana and Akoglu, 2015; Mukherjee et al., 2013a). To deal with the cold-start problem, the original datasets are divided into two parts (Wang et al., 2017b). The reviews posted before January 1, 2012 are used as the training data, and the first new reviews posted by the new reviewers after January 1, 2012 are used as the test data.

We use the SVM method to train the classifier on the training data and test it on the test data. We choose precision (P), recall (R), F1-Score (F1), and accuracy (Acc) as the evaluation metrics. Both the SVM method and the metrics are same as as those in (Wang et al., 2017b; Mukherjee et al., 2013b; Rayana and Akoglu, 2015; Mukherjee et al., 2013a).

4.2 Baseline Methods

We compare our model with eight state-of-the-art methods based on linguistic features and behavioral features, which are listed as follows.

LF (Mukherjee et al., 2013a) captures the linguistic features by extracting bigrams on the labeled review data.

Supervised-CNN uses the same textual information as LF but its features are trained in a supervised convolutional neural network.

LF+BF (Mukherjee et al., 2013a) is a concatenation of linguistic features (LF) and behavioral features (BF).

BF_EditSim+LF (Wang et al., 2017b) first calculates the edit distance between the current review and existing reviews and find the most similar one, then uses its reviewer's behavioral features as the approximation of the new reviewer.

BF_W2VSim+W2V (Wang et al., 2017b) is similar to the BF_EditSim+LF, but uses the similarity of averaged word embedding (pretrained by Word2vec (Mikolov et al., 2013)) to find the most similar review, and concatenates the behavioral features with the average word embedding instead of bigram.

RE* (Wang et al., 2017b) jointly utilizes the TransE (Bordes et al., 2013) to model the behavioral information of reviewers, CNN with same parameter settings of supervised-CNN to encode the textual information, and a constraint to preserve semantics of the sentiment polarity in rating.

RE+RRE+PRE* (Wang et al., 2017b) is an improved version of RE, which further concatenates the review embedding, the review's rating embedding and the item's average rating embedding into a long vector as the feature of the review.

ATT+LF uses the all the attribute values used in our model as the features and then directly concatenates them with bigrams.

For the baselines, we report the results in (Wang et al., 2017b) if they have been implemented, since we conduct experiments on the exactly same datasets and training/testing splits. In addition, we use the same hyper-parameters for our model as those in (Wang et al., 2017b) for a fair comparison.

4.3 Comparison with Baselines

We conduct the comparison experiments on two Yelp datasets. The results are shown in Table 1. AEDA denotes our proposed attribute-enhanced domain adaptive model, and AE refers to a variation of AEDA, which only trains in the single domain without domain-adaption.

It is clear that the proposed AEDA model and its variant AE significantly consistently outperform all the baseline methods on both hotel and restaurant datasets in terms of precision, F1, and accuracy. The slight decreases in recall can be due to the trade off between the precision and recall. The significant improvements of F1 values over baselines clearly demonstrate the effectiveness of our attribute-enhanced method (AE) plus domain adaption technique (AEDA) in cold-start spam review detection task. We have more observations for Table 1.

Table 1: Comparison with baselines.

Row	Features	Hotel				Restaurant			
		P	**R**	**F1**	**Acc**	**P**	**R**	**F1**	**Acc**
1	LF	54.5	71.1	61.7	55.9	53.8	80.8	64.6	55.8
2	Supervised-CNN	61.2	51.7	56.1	59.5	56.9	58.8	57.8	57.1
3	LF+BF	63.4	52.6	57.5	61.1	58.1	61.2	59.6	58.5
4	BF_EditSim+LF	55.3	69.7	61.6	56.6	53.9	82.2	65.1	56.0
5	BF_W2Vsim+W2V	58.4	65.9	61.9	59.5	56.3	73.4	63.7	58.2
6	RE*	62.1	68.3	65.1	63.3	58.4	75.1	65.7	60.8
7	RE+RRE+PRE*	63.6	71.2	67.2	65.3	59.0	78.8	67.5	62.0
8	ATT+LF	71.1	74.7	72.8	72.1	64.0	73.2	68.3	66.0
9	AE (ours)	**76.7**	74.2	**75.4**	**75.8**	**80.3**	66.2	**72.6**	**75.0**
10	AEDA (ours)	**83.9**	74.2	**78.7**	**80.0**	**82.4**	65.1	**72.8**	**75.6**

(1) The LF and Supervised-CNN methods (rows 1-2) which only use linguistic features of reviews are the worst in terms of F1 or accuracy.

(2) Adding the behavioral features (rows 3-6) can improve the performance to some extents, but it is important to choose a good combination method. BF_EditSim+LF and BF_W2Vsim+W2V are both based on the idea of approximating the behavioral features of new reviewer with the existing one with the most similar text. RE* encodes the correlated behavioral information with a neural network framework and thus performs better than simple approximation.

(3) With the additional attribute information (rows 7-10), all the remaining methods outperform those without attributes. For example, our AE shows an increase of 10.3% in F1 and 12.5% in accuracy over RE(*) in hotel domain, and 6.9% in F1 and 14.2% in accuracy in restaurant domain.

(4) Among the methods with additional attributes (rows 7-10), our basic model AE is already significantly better than RE+RRE+PRE*. The improvements may be due to the fact that AE captures the relationship of the common attributes between two entities like the dateDif in Figure 1. Also note that both RE* and RE+RRE+PRE* take all the existing reviews posted before January 1, 2012 to train their models while we only use a small subset. More details about the effects of data size will be discussed in the next section. Furthermore, compared with ATT+LF, our AE leads to an improvement of 9.0% and 3.7% of accuracy, and 4.3% and 2.6% of F1 in restaurant and hotel domain, respectively. This shows that our neural framework can well captures the joint effects of attribute and behavioral information.

(5) Between our AEDA and its simple variant AE (rows 9-10), AEDA is better. The reason is that the attribute-attribute relation in different domains is captured by domain-adaptive objective in AEDA and thus knowledge in the restaurant domain can complement that in the hotel. We find that the performance of AEDA in restaurant does not improve too much. This is because the training set of hotel domain is too small to provide much information for restaurant domain.

4.4 With or Without Unlabeled Data

In the Yelp datasets, 99.18% and 92.58% reviews are unlabeled in hotel and restaurant domain. Wang et al. (2017b) exploited the large-scale unlabeled reviews into their neural network to encode the global behavioral information into the embeddings. Results shows that their model RE* benefits a lot from unlabeled data. However, this is at the expense of efficiency since the training set size increases dramatically when the unlabeled data are added. Our model is trained on the small subset consisting of labeled data. So the problem is how about the performance of RE* without unlabeled data and that of our AE

model with unlabeled data.

Table 2: Impacts of unlabeled data. * indicates that unlabeled data are included

Methods	Hotel				Restaurant			
	P	**R**	**F1**	**Acc**	**P**	**R**	**F1**	**Acc**
RE	53.2	57.6	55.3	53.5	59.8	62.2	61.0	60.2
AE (ours)	76.7	74.2	75.4	75.8	80.3	66.2	72.6	75.0
RE*	62.1	68.3	65.1	63.3	58.4	75.1	65.7	60.8
AE* (ours)	82.4	71.4	76.5	78.1	80.0	67.7	73.3	75.4

We investigate the impacts of unlabeled data and the results are shown in Table 2. It can be seen that whether we use unlabeled data or not, our model is significantly better than that in (Wang et al., 2017b). On one hand, our AE model trained on the small labeled data significantly outperforms the RE* model trained on the whole dataset, let alone our enhanced AE* model. On the other hand, when RE* is reduced to RE, its performance drops a lot. For example, the F1 value decreases from 65.1% to 55.3%. This suggests that the performance of RE* is highly dependent on the size of training data. In contrast, our model is less sensitive to the training size than RE. We believe the reason is that we make good use of attributes. Our model can get discriminative representations even with a very small number of data with the help of attribute information.

4.5 Attribute Effects

To evaluate the effectiveness of each attribute, we remove one attribute from our model at a time, and the results are listed in Table 3.

Table 3: Attribute effects

Removed attribute	Hotel		Restaurant	
	F1	**Acc**	**F1**	**Acc**
location (locDiff)	-0.9	-0.9	0.0	-0.2
date (dateDif)	-22.5	-19.8	-12.1	-12.1
rating (ratingDif)	-1.2	-0.9	-1.0	-0.6
price range	0.5	0.7	0.0	-0.3
ifAcc	0.1	0.0	-0.3	-0.4
hasWifi	1.2	1.2	-0.1	-0.2
hasWeb	-0.3	-0.5	-0.2	-0.4
hasPhone	-0.5	-0.7	0.4	0.1

We find that three attributes (location, date, and rating) are the most important features in both domain. This is reasonable because they are not only the components of the entity itself, but also they are shared by two entities. In particular, we find that the date (dateDif) feature plays a critical role in spam review detection. This is an interesting finding which indicates that spammers may write reviews right after they register at the website. The location difference is more important in hotel than in restaurant. The reason can be that a legitimate user needs an accommodation usually when he/she is out for travelling or business, while a spammer just makes comments and does not care about the location of the hotel.

Among the single attributes, removing "hasWeb" results in the decrease of performance in both hotel and restaurant domains. We suppose that users prefer to search the website of the hotel or restaurant for further information before they make a decision to put it into the schedule. However, these information do not make any sense for spammers. Hence the attribute of "hasWeb" is a good indicator of spam review detection.

5 Conclusion

We introduce an attribute-enhanced domain adaptive embedding (AEDA) model to cope with the cold-start problem in spam review detection. With a carefully designed neural network, our model can jointly encode the inherent attributes of entities, behavioral information, and domain correlations into the review representations. The learnt embeddings are then fed into the traditional SVM machine learning framework to train a classifier and to detect the spam reviews. Experimental results demonstrate the superiority of our proposed AEDA model over all the baseline methods. In the future, we will exploit more available information to enhance the review embedding.

Acknowledgments

The work described in this paper has been supported in part by the NSFC projects (61572376, 91646206), and the 111 project(B07037).

References

Leman Akoglu, Rishi Chandy, and Christos Faloutsos. 2013. Opinion fraud detection in online reviews by network effects. *ICWSM*, 13:2–11.

Shuang Ao, Xiang Li, and Charles X Ling. 2017. Fast generalized distillation for semi-supervised domain adaptation. In *AAAI*, pages 1719–1725.

Mahsa Baktashmotlagh, Mehrtash T Harandi, Brian C Lovell, and Mathieu Salzmann. 2013. Unsupervised domain adaptation by domain invariant projection. In *ICCV*, pages 769–776.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2007. Analysis of representations for domain adaptation. In *NIPS*, pages 137–144.

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

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of simple domain adaptation methods for neural machine translation. *arXiv preprint arXiv:1701.03214*.

Geli Fei, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013. Exploiting burstiness in reviews for review spammer detection. *ICWSM*, 13:175–184.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012a. Syntactic stylometry for deception detection. In *ACL*, pages 171–175.

Song Feng, Longfei Xing, Anupam Gogar, and Yejin Choi. 2012b. Distributional footprints of deceptive product reviews. *ICWSM*, 12:98–105.

Tommaso Fornaciari and Massimo Poesio. 2014. Identifying fake amazon reviews as learning from crowds. In *ACL*, pages 279–287.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *ICML*, pages 1180–1189.

Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. 2011. Domain adaptation for object recognition: An unsupervised approach. In *ICCV*, pages 999–1006.

Zhen Hai, Peilin Zhao, Peng Cheng, Peng Yang, Xiao Li Li, and Guangxia Li. 2016. Deceptive review spam detection via exploiting task relatedness and unlabeled data. In *EMNLP*, pages 1817–1826.

Christopher Harris. 2012. Detecting deceptive opinion spam using human computation. In *Workshops at AAAI*, pages 87–93.

Dirk Hovy. 2016. The enemy in your own camp: How well can we detect statistically-generated fake reviews–an adversarial study. In *ACL*, pages 351–356.

Xin Huang, Yuxin Peng, and Mingkuan Yuan. 2017. Cross-modal common representation learning by hybrid transfer network. *arXiv preprint arXiv:1706.00153*.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *WSDM*, pages 219–230.

Nitin Jindal, Bing Liu, and Ee-Peng Lim. 2010. Finding unusual review patterns using unexpected rules. In *CIKM*, pages 1549–1552.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2016. Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*.

Santosh KC and Arjun Mukherjee. 2016. On the temporal dynamics of opinion spamming: Case studies on yelp. In *WWW*, pages 369–379.

Seongsoon Kim, Hyeokyoon Chang, Seongwoon Lee, Minhwan Yu, and Jaewoo Kang. 2015. Deep semantic frame-based deceptive opinion spam analysis. In *CIKM*, pages 1131–1140.

Fangtao Li, Minlie Huang, Yi Yang, and Xiaoyan Zhu. 2011. Learning to identify review spam. In *IJCAI*, page 2488.

Jiwei Li, Claire Cardie, and Sujian Li. 2013. Topicspam: a topic-model based approach for spam detection. In *ACL*, volume 2, pages 217–221.

Huayi Li, Bing Liu, Arjun Mukherjee, and Jidong Shao. 2014a. Spotting fake reviews using positive-unlabeled learning. *Computación y Sistemas*, 18(3):467–475.

Jiwei Li, Myle Ott, Claire Cardie, and Eduard Hovy. 2014b. Towards a general rule for identifying deceptive opinion spam. In *ACL*, pages 1566–1576.

Huayi Li, Zhiyuan Chen, Arjun Mukherjee, Bing Liu, and Jidong Shao. 2015. Analyzing and detecting opinion spam on a large-scale dataset via temporal and spatial patterns. In *ICWSM*, pages 634–637.

Zheng Li, Yu Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017. End-to-end adversarial memory network for cross-domain sentiment classification. In *IJCAI*, page 2237.

Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. 2010. Detecting product review spammers using rating behaviors. In *CIKM*, pages 939–948.

Jiaming Liu, Yali Wang, and Yu Qiao. 2017. Sparse deep transfer learning for convolutional neural network. In *AAAI*, pages 2245–2251.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. 2015. Learning transferable features with deep adaptation networks. In *ICML*, pages 97–105.

Tong Man, Huawei Shen, Xiaolong Jin, and Xueqi Cheng. 2017. Cross-domain recommendation: an embedding and mapping approach. In *IJCAI*, pages 2464–2470.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. 2017. Question answering through transfer learning from large fine-grained supervision data. *arXiv preprint arXiv:1702.02171*.

Arjun Mukherjee, Bing Liu, and Natalie Glance. 2012. Spotting fake reviewer groups in consumer reviews. In *WWW*, pages 191–200.

Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie Glance. 2013a. Fake review detection: Classification and analysis of real and pseudo reviews. *Technical Report UIC-CS-2013–03, University of Illinois at Chicago, Tech. Rep.*

Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie S Glance. 2013b. What yelp fake review filter might be doing? In *ICWSM*.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *ACL*, pages 309–319.

Shebuti Rayana and Leman Akoglu. 2015. Collective opinion spam detection: Bridging review networks and metadata. In *SIGKDD*, pages 985–994.

Yafeng Ren and Yue Zhang. 2016. Deceptive opinion spam detection using neural network. In *COLING*, pages 140–150.

Ben Tan, Yu Zhang, Sinno Jialin Pan, and Qiang Yang. 2017. Distant domain transfer learning. In *AAAI*, pages 2604–2610.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*.

Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. 2015. Simultaneous deep transfer across domains and tasks. In *ICCV*, pages 4068–4076.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *CVPR*, page 4.

Guan Wang, Sihong Xie, Bing Liu, and S Yu Philip. 2011. Review graph based online store review spammer detection. In *ICDM*, pages 1242–1247.

Xuepeng Wang, Kang Liu, Shizhu He, and Jun Zhao. 2016. Learning to represent review with tensor decomposition for spam detection. In *EMNLP*, pages 866–875.

Rui Wang, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. 2017a. Sentence embedding for neural machine translation domain adaptation. In *ACL*, pages 560–566.

Xuepeng Wang, Kang Liu, and Jun Zhao. 2017b. Handling cold-start problem in review spam detection by jointly embedding texts and behaviors. In *ACL*, pages 366–376.

Sihong Xie, Guan Wang, Shuyang Lin, and Philip S Yu. 2012. Review spam detection via temporal pattern discovery. In *SIGKDD*, pages 823–831.

Qiongkai Xu and Hai Zhao. 2013. Using deep linguistic features for finding deceptive opinion spam. In *COLING*, pages 1341–1350.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *NIPS*, pages 3320–3328.

Robust Lexical Features for Improved Neural Network Named-Entity Recognition

Abbas Ghaddar
RALI-DIRO
Université de Montréal
Montréal, Canada
abbas.ghaddar@umontreal.ca

Philippe Langlais
RALI-DIRO
Université de Montréal
Montréal, Canada
felipe@iro.umontreal.ca

Abstract

Neural network approaches to Named-Entity Recognition reduce the need for carefully hand-crafted features. While some features do remain in state-of-the-art systems, lexical features have been mostly discarded, with the exception of gazetteers. In this work, we show that this is unfair: lexical features are actually quite useful. We propose to embed words and entity types into a low-dimensional vector space we train from annotated data produced by distant supervision thanks to Wikipedia. From this, we compute — offline — a feature vector representing each word. When used with a vanilla recurrent neural network model, this representation yields substantial improvements. We establish a new state-of-the-art F1 score of 87.95 on ONTONOTES 5.0, while matching state-of-the-art performance with a F1 score of 91.73 on the over-studied CONLL-2003 dataset.

1 Introduction

Named-Entity Recognition (NER) is the task of identifying textual mentions and classifying them into a predefined set of types. Various approaches have been proposed to tackle the task, from hand-crafted feature-based machine learning models like conditional random fields (Finkel et al., 2005) and perceptron (Ratinov and Roth, 2009), to deep neural models (Collobert et al., 2011; Ma and Hovy, 2016; Strubell et al., 2017).

Word representations (Turian et al., 2010; Mikolov et al., 2013), also known as word embeddings, are a key element for multiple NLP tasks including NER (Collobert et al., 2011). Due to the small amount of named-entity annotated data, embeddings are used to extend, rather than replace, hand-crafted features in order to obtain state-of-the-art performance (Lample et al., 2016). Recent studies (Yang et al., 2017; Søgaard and Goldberg, 2016) have explored methods for supplying deep sequential taggers with complementary features to standard embeddings. Peters et al. (2017) and Tran et al. (2017) tested special embeddings extracted from a neural language model (LM) trained on a large corpus. LM embeddings capture context-dependent aspects of word meaning using future (forward LM) and previous (backward LM) context words. When this information is added to standard features, it leads to significant improvements in NER. Also, Chiu and Nichols (2016) showed that external knowledge resources (namely gazetteers) are crucial to NER performance. Gazetteer features encode the presence of word n-grams in predefined lists of NEs.

In this work, we discuss some of the limitations of gazetteer features and propose an alternative lexical representation which is trained offline and that can be added to any neural NER system. In a nutshell, we embed words and entity types into a joint vector space by leveraging WiFiNE (Ghaddar and Langlais, 2018), a ressource which automatically annotates mentions in Wikipedia with 120 entity types. From this vector space, we compute for each word a 120-dimensional vector, where each dimension encodes the similarity of the word with an entity type. We call this vector an LS representation, for Lexical Similarity. When included in a vanilla LSTM-CRF NER model, LS representations lead to significant gains. We

Proceedings of the 27th International Conference on Computational Linguistics, pages 1896–1907
Santa Fe, New Mexico, USA, August 20-26, 2018.

establish a new state-of-the-art F1 score of 87.95 on ONTONOTES 5.0, while matching state-of-the-art performance on the over-studied CONLL-2003 dataset.

In the rest of this paper, we motivate our work in Section 2. We describe how we compute LS vectors in Section 3. We present our system in Section 4 and report results in Section 5. In Section 6, we discuss related works before concluding in Section 7.

2 Motivation

Gazetteers are lists of entities that are associated with specific NE categories. They are widely used as a feature source in NER, and have been successfully included in feature-based (Ratinov and Roth, 2009) and neural (Chiu and Nichols, 2016) models. Typically, lists of entities are compiled from structured data sources such as DBpedia (Auer et al., 2007) or Freebase (Bollacker et al., 2008). The surface form of the title of a Wikipedia article, as well as aliases and redirects are mapped to an entity type using the `object_type` attribute of the related DBpedia (or Freebase) page. Ratinov and Roth (2009) use this methodology to compile 30 lists of fine-grained entity types extracted from Wikipedia, while Chiu and Nichols (2016) create 4 gazetteers that map to CoNLL categories (PER, LOC, ORG and MISC). Despite their importance, gazetteer-based features suffer from a number of limitations.

- **Binary representation.** Gazetteer features encode only the presence of an n-gram in each list and omit its relative frequency. For example, the word "France" can be used as a person, an organization, or a location, while it likely refers to the country most of the time. Binary features cannot capture this preference.

- **Generation.** At test time, we need to match every n-gram (up to the length of the longest lexicon entry) in a sentence against entries in the lexicons, which is time consuming. In their work, Chiu and Nichols (2016) use 4 lists that count over 2.3M entries.

- **Non-entity words.** Gazetteer features do not capture signal from non-entity words, while earlier feature-based models strived to encode that some words (or n-grams) trigger specific entity types. For instance, words such as "eat", "directed" or "born" are words that typically appear after a mention of type PER.

To overcome those limitations, we propose an alternative approach where we embed annotations mined from Wikipedia into a vector space from which we compute a feature vector that represent words. This vector compactly and efficiently encodes both gazetteer and lexical information. Note that at test time, we only have to feed our model with this feature vector, which is efficient.

3 Our Method

3.1 Embedding Words and Entity Types

Turning Wikipedia into a corpus of named-entities annotated with types is a task that received continuous attention over the years (Nothman et al., 2008; Al-Rfou et al., 2015; Ghaddar and Langlais, 2017). It consists mainly in exploiting the hyperlink structure of Wikipedia in order to detect entity mentions. Then, structured data from a knowledge base (for instance Freebase) are used to map hyperlinks to entity types. Because the number of anchored strings in Wikipedia is no more than 3% of the text tokens, Ghaddar and Langlais (2017) proposed to augment Wikipedia articles with mentions unmarked in Wikipedia, thanks to a mix of heuristics that benefit the Wikipedia structure (Ghaddar and Langlais, 2016a), as well as a coreference resolution system adapted specifically to Wikipedia (Ghaddar and Langlais, 2016b).

The authors applied their approach on English Wikipedia and produce coarse (4 classes) and fine-grained (120 labels) named-entity annotations, leading to WiNER (Ghaddar and Langlais, 2017) and WiFiNE (Ghaddar and Langlais, 2018). In this work, we adopt WiFiNE which is publicly available at `http://rali.iro.umontreal.ca/rali/en/wifiner-wikipedia-for-et` as our source of annotations. Each entity mention is mapped (via its Freebase `object_type` attribute) to a pre-defined set of 120 entity types. Types are stored in a 2-level hierarchical structure

(e.g. `/person` and `/person/musician`). The corpus consist of 3.2M Wikipedia articles, comprising 1.3G tokens that we annotated with 157.4M named-entity mentions and their types. We used this very large quantity of automatically annotated data for jointly embedding words and entity types into the same low-dimensional space. The key idea consists in learning an embedding for each entity type using its surrounding words. For instance, the embedding for `/product/software` will be trained using context words that surround all entities that were (automatically) labelled as `/product/software` in Wikipedia. In practice, we found that simply concatenating a sentence (v1) with its annotated version (v2), as illustrated in Figure 1, offers a simple but efficient way of combining words and entity types so that embeddings can make good use of them.

(v1) On **October 9, 2009**, the **Norwegian Nobel Committee** announced that **Obama** had won the **2009 Nobel Peace Prize**.
(v2) On `/date`, the `/organization/government_agency` announced that `/person/politician` had won the `/award`.

Figure 1: Example of the two variants of a given sentence.

We use the FastText toolkit (Bojanowski et al., 2016) to learn the uncased embeddings for both words and entity types. We train a skipgram model to learn 100-dimensional vectors with a minimum word frequency cutoff of 5, and a window size of 5. This configuration (recommended by the authors) performs the best in the experiments described in Section 5. Since FastText learns representations of character *n*-grams, it has the ability to produce vectors for unknown words.

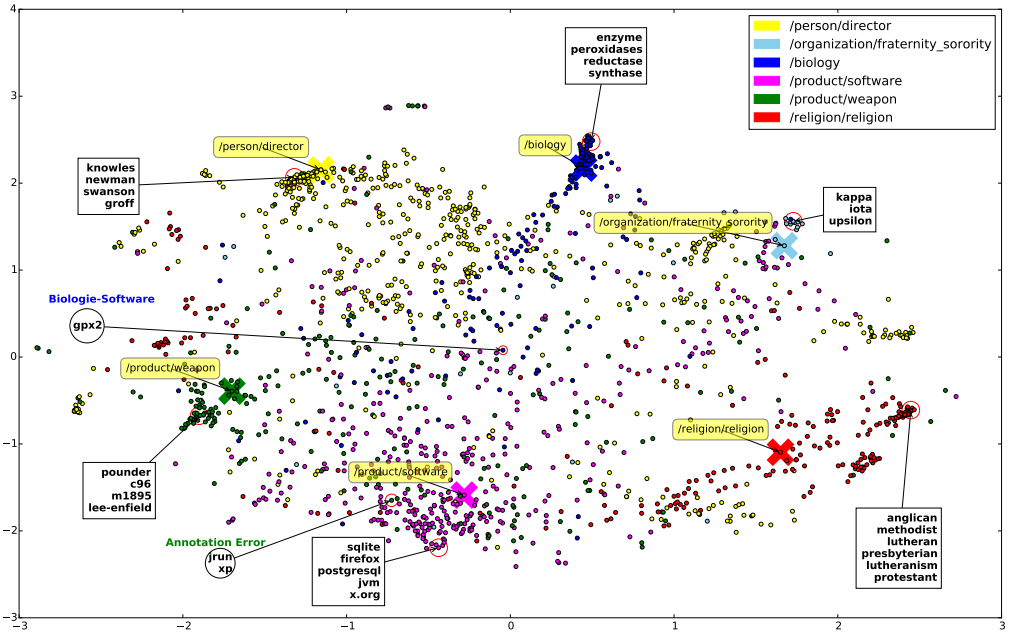


Figure 2: Two-dimensional representation of the vector space which embeds both words and entity types. Big Xs indicate entity types, while circles refer to words (i.e. named-entities, here).

Figure 2 illustrates a T-SNE (van der Maaten, 2014) two-dimensional projection of the embedding of 6 entity types and a sample of 1500 words. Entity type embeddings are marked by big Xs, while circles indicate words. For visualization proposes, we only plot single-word mentions that were annotated in WiFiNE with one of those 6 types. Words were randomly and proportionally sampled according to the frequency of each entity type. In addition, words have the color associated with the most frequent type they were annotated with in WiFiNE.

We observe that mentions often annotated by a given type in our resource tend to cluster around this entity type. For instance, "firefox" is close to the type /product/software, while "enzyme" is close to the /biology entity type. We also notice that words that are labelled with different types tend to appear between types they were annotated with. For instance, "gpx2", which is used both as a software and as a gene, has its embedding in between /product/software and /biology. We inspected some of the words plotted in Figure 2, and found that "jrun" and "xp" are incorrectly labelled as /product/weapon in WiFiNE. But since these words are seen in a *software* context, their embeddings are closer to the /product/software embedding than the /product/weapon one. We feel this tolerance to noise is a desirable feature, one that hopefully allows a more efficient use of distant supervision. Last, we also observe the tendency of rare words to cluster around their entity type. For instance, "iota" and "x.org" are embedded near their respective types, despite the fact that they appear less than 30 times in the version of Wikipedia used to compile WiFiNE.

3.2 LS Representation

This joint vector space only serves the purpose of associating to each word a LS representation, that is, a 120-dimensional vector where the ith coefficient is a value in the $[-1, +1]$ interval, equal to the cosine similarity[1] between the word embedding and the embedding of the ith entity type (we have 120 types).

Word	Entity Type	Sim	Word	Entity Type	Sim
hilton	/building/hotel	0.58	located	/location	0.47
	/building/restaurant	0.46		/location/city	0.44
	/person/actor	0.37		/building	0.40
gpx2	/biology	0.69	directed	/person/director	0.60
	/product/software	0.56		/art/film	0.55
jrun	/product/software	0.64	in	/date	0.58
	/product/weapon	0.23		/location/city	0.54
dammstadt	/location/city	0.45	won	/award	0.53
	/location/railway	0.44		/event/sports_event	0.53

Table 1: Topmost similar entity types to a few single-word mentions (left table) and non-entity words (right table).

Table 1 shows the topmost similar entity types for proper names (left column) and common words (right column). We observe that ambiguous mentions (those annotated with several types) are adequately handled. For instance, the LS representation of the word "hilton" encodes that it more often refers to a hotel or a restaurant than to an actress. Also, we observe that entity words that are either not or rarely annotated in WiFiNE are still adequately associated with their right type. For instance, "dammstadt", which appears only 5 times in WiFiNE, and which refers to the Damm city in Germany, is most similar to /location/city and /location/railway. Interestingly, this mention does not have its page in English Wikipedia. Furthermore, we observe that non-entity context words have a strong similarity to types they precede or succeed. For instance the verb "directed" is very close to /person/director, an entity type that usually precedes it, and to /art/film, that usually follows it. Likewise, the preposition "in" is near /date and /location/city, which frequently follow "in".

3.3 Strength of the LS Representation

To summarize, we propose a compact lexical representation which is computed offline, therefore incurring no computation burden at test time This representation encodes the preference of an entity-mention word for a given type, an information out of reach of binary gazetteer features. It also lends itself nicely to the inclusion of lexical features that have been successfully used in earlier feature-based systems (Ratinov and Roth, 2009; Luo et al., 2015). Also, because entity types are well represented in WiFiNE, their

[1]The cosine similarity outperforms other metrics in our experiments.

embeddings are robust: Our representation does accommodate unfrequent words and seems tolerant to the inherent noise of distant supervision.

4 Our NER System

In order to test the efficiency of our lexical feature representation, we implemented a state-of-the-art NER system we now describe.

4.1 Bi-LSTM-CRF Model

We adopt the popular Bi-LSTM-CRF architecture (Figure 3), a *de facto* baseline in many sequential tagging tasks (Lample et al., 2016; Søgaard and Goldberg, 2016; Chiu and Nichols, 2016).

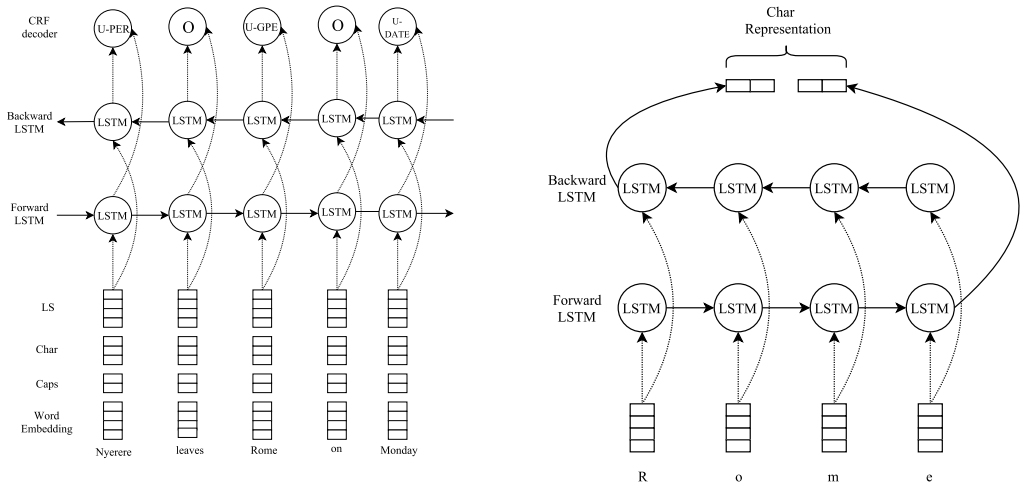


Figure 3: **Left Figure:** Main architecture of our NER system. **Right Figure:** Character representation of the word "Roma" given to the word-level bi-LSTM.

4.2 Features

In addition to the LS vector, we incorporate publicly available pre-trained embeddings, as well as character-level, and capitalization features. Those features have been shown to be crucial for state-of-the-art performance.

4.2.1 Word Embeddings

We experimented with several publicly available word embeddings, such as Senna (Collobert et al., 2011), Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and SSKIP (Yulia et al., 2015). We find that the latter performs the best in our experiments. SSKIP embeddings are 100-dimensional case sensitive vectors that where trained using a *n*-skip-gram model (Yulia et al., 2015) on 42B tokens. These embeddings were previously used by (Lample et al., 2016; Strubell et al., 2017), who report good performance on CONLL, and state-of-the-art results on ONTONOTES respectively. Note that these pre-trained embeddings are adjusted during training.

4.2.2 Character Embeddings

Following (Lample et al., 2016), we use a forward and a backward LSTM to derive a representation of each word from its characters (right part of Figure 3). A character lookup table is randomly initialized, then trained at the same time as the Bi-LSTM model sketched in Section 4.1.

4.2.3 Capitalization Features

Similarly to previous works, we use capitalization features for characterizing certain categories of capitalization patterns: `allUpper`, `allLower`, `upperFirst`, `upperNotFirst` , `numeric` or

noAlphaNum. We define a random lookup table for these features, and learn its parameters during training.

4.2.4 LS Vectors

Contrarily to previous features, lexical vectors are computed offline and are not adjusted during training. We found useful in practice to apply a MinMax scaler in the range $[-1, +1]$ to each LS vector we computed; thus, $[.., 0.095, .., 0.20, .., 0.76, ..]$ becomes $[.., -1, .., -0.67, .., 1, ..]$.

5 Experiments

5.1 Data and Evaluation

We consider two well-established NER benchmarks: CoNLL-2003 and OntoNotes 5.0. Table 2 provides an overview of the two datasets. As we can see, OntoNotes is much larger. For both datasets, we convert the IOB encoding to BILOU, since Ratinov and Roth (2009) found the latter to perform better. In keeping with others, we report mention-level F1 score using the conlleval script[2].

The CoNLL-2003 NER dataset (Tjong Kim Sang and De Meulder, 2003) is a well known collection of Reuters newswire articles that contains a large portion of sports news. It is annotated with four entity types: *Person (*PER*), Location (*LOC*), Organization (*ORG*) and Miscellaneous (*MISC*)*. The four entity types are fairly evenly distributed, and the train/dev/test datasets present a similar type distribution.

Dataset		Train	Dev	Test
CoNLL-2003	*#tok*	204,567	51,578	46,666
	#ent	23,499	5,942	5,648
OntoNotes 5.0	*#tok*	1,088,503	147,724	152,728
	#ent	81,828	11,066	11,257

Table 2: Statistics of the CoNLL-2003 and OntoNotes 5.0 datasets. *#tok* stands for the number of tokens, and *#ent* indicates the number of named-entities gold annotated.

The OntoNotes 5.0 dataset (Hovy et al., 2006; Pradhan et al., 2013) includes texts from five different genres: broadcast conversation (200k), broadcast news (200k), magazine (120k), newswire (625k), and web data (300k). This dataset is annotated with 18 entity types, and is much larger than CoNLL. Following previous researches (Chiu and Nichols, 2016; Strubell et al., 2017), we use the official train/dev/test split of the CoNLL-2012 shared task (Pradhan et al., 2012). Also, we exclude (both during training and testing) the New Testaments portion as it does not contain gold NE annotations.

5.2 Training and Implementation

Training is carried out by mini-batch stochastic gradient descent (SGD) with a momentum of 0.9 and a gradient clipping of 5.0. The mini-batch is 10 for both datasets, and learning rates are 0.009 and 0.013 for CoNLL and OntoNotes respectively. More sophisticated optimization algorithms such as AdaDelta (Zeiler, 2012) or Adam (Kingma and Ba, 2014) converge faster, but none outperformed SGD with exponential learning rate decay in our experiments.

Our system uses a single Bi-LSTM layer at the word level whose hidden dimensions are set to 128 and 256 for CoNLL and OntoNotes respectively. For both models, the character embedding size was set to 25, and the hidden dimension of the forward and backward character LSTMs are set to 50. To mitigate overfitting, we apply a dropout mask (Srivastava et al., 2014) with a probability of 0.5 on the input and output vectors of the Bi-LSTM layer. For both datasets, we set the dimension of capitalization embeddings to 25 and trained the models up to 50 epochs.

We tuned the hyper-parameters by grid search, and used early stopping based on the performance on the development set. We varied dropout ($[0.25, 0.5, 0.65]$), hidden units ($[50, 128, 256, 300]$), capitalization ($[10, 20, 30]$) and char ($[25, 50, 100]$) embedding dimensions, learning rate ($[0.001, 0.015]$ by step

[2] http://www.cnts.ua.ac.be/conll2000/chunking/conlleval.txt

0.002), and optimization algorithms and fixed the other hyper-parameters. We implemented our system using the Tensorflow (Abadi et al., 2016) library, and ran our models on a GeForce GTX TITAN Xp GPU. Training requires about 2.5 hours for CoNLL and 8 hours for OntoNotes.

5.3 Results on the Development Set

Table 3 shows the development set performance of our final models on each dataset compared to the work of Chiu and Nichols (2016). The authors use an architecture similar to ours, but use a binary gazetteer feature set, while we use our LS representation. Since our systems involve random initialization, we report the mean as well as the standard deviation over five runs. The improvements yielded by our model on the CoNLL dataset are significant although modest, while those observed on OntoNotes are more substantial. We also observe a lower variance of our system over the 5 runs.

	CoNLL	**OntoNotes**
(Chiu and Nichols, 2016)	94.03 ($\pm$ 0.23)	84.57 ($\pm$ 0.27)
Our model	**94.80 ($\pm$ 0.10)**	**86.44 ($\pm$ 0.14)**

Table 3: Development set F1 scores of our best hyper-parameter setting compared to the results reported in (Chiu and Nichols, 2016).

5.4 Results on CoNLL

Table 4 reports our model's performance[3] on the CoNLL test set, as well as the performance of systems previously tested on this test set (the figures are those published by the authors). Because of the small size of the training set, some authors (Chiu and Nichols, 2016; Yang et al., 2017; Peters et al., 2017; Peters et al., 2018) incorporated the development set as a part of training data after tuning the hyper-parameters. Consequently, their results are not directly comparable, so we do not report them.

Model	LEX	GAZ	CAP	EMB	CHE	LME	LS	**F1**
(Finkel et al., 2005)	+	+	+	•	•	•	•	86.86
(Ratinov and Roth, 2009)	+	+	+	•	•	•	•	90.88
(Lin and Wu, 2009)	+	+	+	•	•	•	•	90.90
(Luo et al., 2015)	+	+	+	•	•	•	•	91.20
(Collobert et al., 2011)	•	+	+	+	•	•	•	89.56
(Huang et al., 2015)	•	•	+	+	+	•	•	90.10
(Lample et al., 2016)	•	•	+	+	+	•	•	90.94
(Ma and Hovy, 2016)	•	•	+	+	+	•	•	91.21
(Shen et al., 2017)	•	•	+	+	•	•	•	90.89
(Strubell et al., 2017)	•	•	+	+	•	•	•	90.54
(Tran et al., 2017)	•	•	+	+	+	+	•	91.69
(Liu et al., 2017)	•	•	+	+	+	+	•	91.71
This work	•	+	+	+	+	•	+	**91.73**

Table 4: F1 scores on the CoNLL test set. The first four systems are feature-based, the others are neuronal. The feature configuration of each system is encoded with: LEX which stands for LEXical feature, GAZ for GAZetteers, CAP for CAPitalization, EMB for pre-trained EMBeddings, CHE for CHaracter Embeddings, LME for Language Model Embeddings, and LS for the proposed LS feature representation. + indicates that the model uses the feature set.

First, we observe that our model significantly outperforms models that use extensive sets of hand-crafted features (Ratinov and Roth, 2009; Lin and Wu, 2009) as well as the system of (Luo et al.,

[3] Standard deviation on the test set is reported in Table 7

2015) that uses NE and Entity Linking annotations to jointly optimize the performance on both tasks. Second, our model outperforms as well other NN models that only use standard word embeddings, which indicates that our lexical feature vector is complementary to standard word embeddings. Third, our system matches state-of-the-art performances of models that use either more complex architectures or more elaborate features. Tran et al. (2017) use three layers of stacked residual RNN (Bi-LSTM) with bias decoding. Our model is much simpler and faster. They report a performance of 90.43 when using an architecture similar to ours. The two systems that have slightly higher F1 scores on the CoNLL dataset both use embeddings obtained from a forward and a backward Language Model trained on the One Billion Word Benchmark (Chelba et al., 2013). They report gains between 0.8 and 1.2 points by using such LM embeddings, which suggests that LS vectors are indeed efficient. Unfortunately, due to time and resource constraints,[4] we were not able to measure whether both features complement each other. This is left for future investigations.

5.5 Results on OntoNotes

Table 5 reports the F1 score of our system compared to the performance reported by others on the OntoNotes test set. To the best of our knowledge, we surpass previously reported F1 scores on this dataset. In particular, our system significantly outperforms the Bi-LSTM-CNN-CRF models of (Chiu and Nichols, 2016) and (Strubell et al., 2017) by an absolute gain of 1.68 and 0.96 points respectively. Less surprisingly, it surpasses systems with hand-crafted features, including Ratinov and Roth (2009) that use gazetteers, and the system of Durrett and Klein (2014) which uses coreference annotation in OntoNotes to jointly model NER, entity linking, and coreference resolution tasks.

Model	LEX	GAZ	CAP	EMB	CHE	LME	LS	**F1**
(Finkel and Manning, 2009)	+	+	+	•	•	•	•	82.42
(Ratinov and Roth, 2009)	+	+	+	•	•	•	•	84.88
(Passos et al., 2014)	+	+	+	•	•	•	•	82.24
(Durrett and Klein, 2014)	+	+	+	•	•	•	•	84.04
(Chiu and Nichols, 2016)	•	+	+	+	+	•	•	86.28
(Shen et al., 2017)	•	•	+	+	+	•	•	86.52
(Strubell et al., 2017)	•	•	+	+	+	•	•	86.99
This work	•	+	+	+	+	•	+	**87.95**

Table 5: F1 scores on the OntoNotes test set. The first four systems are feature-based, the following ones are neuronal. See Table 4 for an explanation of the column of features.

The OntoNotes benchmark is annotated with 18 types (e.g. LAW, PRODUCT) and contains many rare words, especially in the Web data collection. Chiu and Nichols (2016) note that the 4-class gazetteer they used yielded marginal improvements on OntoNotes, contrarily to CoNLL. In particular, they observe that mentions that match LOC entries in their gazetteer often match GPE, NORP and FAC lists.

Model	**BC**	**BN**	**MZ**	**NW**	**TC**	**WB**
(Finkel and Manning, 2009)	78.66	87.29	82.45	85.50	67.27	72.56
(Durrett and Klein, 2014)	78.88	87.39	82.46	87.60	72.68	76.17
(Chiu and Nichols, 2016)	85.23	89.93	84.45	88.39	72.39	78.38
This work	**86.33**	**90.46**	**85.91**	**89.75**	**75.41**	**80.39**

Table 6: Per-genre F1 scores on OntoNotes (numbers taken from Chiu and Nichols (2016)). BC = broadcast conversation, BN = broadcast news, MZ = magazine, NW = newswire, TC = telephone conversation, WB = blogs and newsgroups.

[4] LM embeddings are not publicly available, and according to Jozefowicz et al. (2016), they require three weeks to train on 32 GPUs.

They suggest that a finer-grained gazetteer could improve the performance of their system on ONTONOTES. Our results confirm this, since we use 120 types. We further detail the gains we observed for each sub-collection of texts in the test set. Table 6 reveals that major improvements over the model of (Chiu and Nichols, 2016) are on noisier collections such as telephone conversations (+3 points) and blogs or newsgroups (+2 points). Those type of texts are characterized by a large set of infrequent words, for which classical embeddings are typically poorly trained. Our approach does not seem to suffer from this problem as severely, as discussed in Section 2.

5.6 Ablation Results

In this experiment, we directly compare the LS representation with the SSKIP word-embedding feature set. In order to maintain a high level of performance, both character and capitalization features are used in all configurations. We want to point out that LS vectors are not adapted during training, contrarily to the SSKIP embeddings. Similarly to Section 5.3, we report in Table 7, for each feature configuration, the average F1 score as well as the standard deviation over five runs.

Model	CONLL	ONTONOTES
SSKIP	90.52 ($\pm$ 0.18)	86.57 ($\pm$ 0.10)
LS	89.94 ($\pm$ 0.16)	85.92 ($\pm$ 0.12)
all	**91.73 ($\pm$ 0.10)**	**87.95 ($\pm$ 0.13)**

Table 7: F1 scores of differently trained systems on CONLL and ONTONOTES 5.0 datasets. Capitalization (Section 4.2.3) and character features (Section 4.2.2) are used by default by all models.

We observe that on both CONLL and ONTONOTES, the SSKIP model outperforms our feature vector approach by 0.65 F1 points on average. The difference is not has high as we first expected, especially since the SSKIP model is adjusted during training, while our representation is not. Still, LS vectors seem to encode a large portion of the information needed to model the NER task. Also, it is worth mentioning that our embeddings are trained on 1.3B words compared to 42B for SSKIP.

We also observe that models that use both feature sets significantly outperform other configurations. To confirm that the gains came from our feature vector and not from increasing the number of hidden units, we tested several SSKIP models by increasing the LSTM hidden layer dimension so that number of parameters is the same as the model with LS vectors. We observed a degradation of performance on both datasets, mostly due to overfitting on the training set. From those results, we conclude that our lexical representation and the SSKIP one are complementary.

6 Related Works

Traditional approaches to NER, like CRF-based (Finkel et al., 2005) and Perceptron-based systems (Ratinov and Roth, 2009) have dominated the field for over a decade. They rely heavily on hand-engineered features (Luo et al., 2015) and external resources such as gazetteers. One major drawback of such an approach is its weak generalization power (Lample et al., 2016). Current state-of-the art systems (Chiu and Nichols, 2016; Strubell et al., 2017) use a combination of Convolutional Neural Networks (CNNs), Bi-LSTMs, along with a CRF decoder. CNNs are used to encode character-level features (prefix and suffix), while LSTM is used to encode word-level features. Finally, a CRF is placed on top of those models in order to decode the best tag sequence. Pre-trained embeddings obtained by unsupervised learning are core features of those models. In this work, we show that deep NN architectures can also benefit from lexical features, at least when encoded in the compact form we propose.

Tran et al. (2017) and Peters et al. (2017) propose an alternative approach different from ours. They incorporate LM embeddings that were pre-trained on a large unlabelled corpus as features for NER. These embeddings allow to generate a representation for a word depending on its context. For instance, the LM embeddings of the word *France* in "*France is a developed country*" is different than that in "*Anatole France began his literary career*". Such embeddings are trained on very large amount of texts.

Our feature set is crafted from distant supervision applied to Wikipedia, a much less time-consuming process which we showed to be nevertheless adapted to rare words. Chiu and Nichols (2016) used gazetteer features in order to establish state-of-the-art performance on both CoNLL and OntoNotes. They mined DBPedia in order to compile 4 lists of named-entities that contain over 2.3M entries. We show that LS representations outperform their gazetteer features.

7 Conclusion and Future Work

We have explored the idea of generating lexical features for NER out of Wikipedia data automatically annotated with fine-grained entity types. We used WiFiNE (Ghaddar and Langlais, 2018), a Wikipedia dump annotated with fine entity type mentions, for training a vector space that jointly embeds words and named-entities. This vector space is used to compute a 120 dimensional vector per word, which encodes the similarity of the word to each of the entity types. Our results show that our proposed lexical representation, even though it is not adjusted at training time, matches state-of-the-art results compared to more complex approaches on the well-studied CoNLL dataset, and delivers a new state-of-the-art F1 score of 87.95 on the more diversified OntoNotes dataset. We further observe larger gains on collections with more unfrequent words.

The source code and the data we used in this work are publicly available at `http://rali.iro.umontreal.ca/rali/en/wikipedia-lex-sim`, with the hope that other researchers will report gains, when using our lexical representation. As a future work, we want to investigate the usefulness of our LS feature representation on other NER tasks, including NER in tweets where out-of-vocabulary and low-frequency words represent a challenge; as well as finer-grained NER which suffers from the lack of manually annotated training data.

Acknowledgements

This work has been partly funded by the TRIBE Natural Sciences and Engineering Research Council of Canada CREATE program and Nuance Foundation. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. We thank the anonymous reviewers for their insightful comments.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Rami Al-Rfou, Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. 2015. Polyglot-NER: Massive multilingual named entity recognition. In *Proceedings of the 2015 SIAM International Conference on Data Mining, Vancouver, British Columbia, Canada*. SIAM.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. In *Proceedings of the 54st Annual Meeting of the Association for Computational Linguistics*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Greg Durrett and Dan Klein. 2014. A Joint Model for Entity Analysis: Coreference, Typing, and Linking. *Transactions of the Association for Computational Linguistics*, 2:477–490.

Jenny Rose Finkel and Christopher D Manning. 2009. Joint Parsing and Named Entity Recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 326–334. Association for Computational Linguistics.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Abbas Ghaddar and Philippe Langlais. 2016a. Coreference in Wikipedia: Main Concept Resolution. In *CoNLL*, pages 229–238.

Abbas Ghaddar and Philippe Langlais. 2016b. WikiCoref: An English Coreference-annotated Corpus of Wikipedia Articles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia, 05/2016.

Abbas Ghaddar and Phillippe Langlais. 2017. WiNER: A Wikipedia Annotated Corpus for Named Entity Recognition. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 413–422.

Abbas Ghaddar and Phillippe Langlais. 2018. Transforming Wikipedia into a Large-Scale Fine-Grained Entity Type Corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France, may. European Language Resources Association (ELRA), European Language Resources Association (ELRA).

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for Named Entity Recognition. *arXiv preprint arXiv:1603.01360*.

Dekang Lin and Xiaoyun Wu. 2009. Phrase Clustering for Discriminative Learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1030–1038. Association for Computational Linguistics.

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2017. Empower Sequence Labeling with Task-Aware Neural Language Model. *arXiv preprint arXiv:1709.04109*.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint Entity Recognition and Disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 879–888, Lisbon, Portugal, September. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Joel Nothman, James R Curran, and Tara Murphy. 2008. Transforming Wikipedia into named entity training data. In *Proceedings of the Australian Language Technology Workshop*, pages 124–132.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for Named Entity Resolution. *arXiv preprint arXiv:1404.5367*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*, volume 14, pages 1532–1543.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards Robust Linguistic Analysis using OntoNotes. In *CoNLL*, pages 143–152.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Tom Redman, Mark Sammons, and Dan Roth. 2016. Illinois Named Entity Recognizer: Addendum to Ratinov and Roth '09 reporting improved results.

Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep Active Learning for Named Entity Recognition. *arXiv preprint arXiv:1707.05928*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 231–235.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and Accurate Entity Recognition with Iterated Dilated Convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2660–2670.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Quan Tran, Andrew MacKinlay, and Antonio Jimeno Yepes. 2017. Named entity recognition with stack residual lstm and trainable bias decoding. *arXiv preprint arXiv:1706.07598*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

L.J.P. van der Maaten. 2014. Accelerating t-SNE using Tree-Based Algorithms. *Journal of Machine Learning Research*, 15:3221–3245.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural Reranking for Named Entity Recognition. *arXiv preprint arXiv:1707.05127*.

Wang Ling Lin Chu-Cheng Yulia, Tsvetkov Silvio Amir, Ramón Fernandez Astudillo Chris Dyer Alan, and W Black Isabel Trancoso. 2015. Not all contexts are created equal: Better word representations with variable attention.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

A Pseudo Label based Dataless Naive Bayes Algorithm for Text Classification with Seed Words

Ximing Li, Bo Yang*

College of Computer Science and Technology, Jilin University, China
Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, China
liximing86@gmail.com; ybo@jlu.edu.cn

Abstract

Traditional supervised text classifiers require a large number of manually labeled documents, which are often expensive to obtain. Recently, dataless text classification has attracted more attention, since it only requires very few seed words of categories that are much cheaper. In this paper, we develop a pseudo-label based dataless Naive Bayes (PL-DNB) classifier with seed words. We initialize pseudo-labels for each document using seed word occurrences, and employ the expectation maximization algorithm to train PL-DNB in a semi-supervised manner. The pseudo-labels are iteratively updated using a mixture of seed word occurrences and estimations of label posteriors. To avoid noisy pseudo-labels, we also consider the information of nearest neighboring documents in the pseudo-label update step, i.e., preserving local neighborhood structure of documents. We empirically show that PL-DNB outperforms traditional dataless text classification algorithms with seed words. Especially, PL-DNB performs well on the imbalanced dataset.

1 Introduction

Automatic text classification is one of the most popular directions in the machine learning community. A typical procedure for creating a classifier in supervised learning consists of two steps: (1) Given a collection of text documents, human experts define a number of category labels, and then manually assign these pre-defined labels to documents. We refer to these labeled documents as training dataset; (2) A supervised learning algorithm, e.g., support vector machines (SVMs), is trained on the training dataset, outputting a classifier for predicting future documents.

Manually labeling documents, i.e., step (1), is very expensive and time-consuming. Unfortunately, supervised learning algorithms often require massive labeled documents to avoid learning issues such as overfitting (Cawley and Talbot, 2010). A common way to reduce the labeling effort is developing semi-supervised learning algorithms, where one trains text classifiers on a mixture collection of a few labeled documents and a larger number of unlabeled documents (Nigam et al., 2000; Hu et al., 2017). However, semi-supervised learning still requires labeled documents, and manually labeling a small number of documents remains very expensive in many real world applications.

Recently, researchers have proposed a number of dataless text classification algorithms (Liu et al., 2004; Chang et al., 2008; Downey and Etzioni, 2008; Druck et al., 2008; Hingmire et al., 2013; Hingmire and Chakraborti, 2014; Chen et al., 2015; Li et al., 2016), which do not require labeled documents as training instances. Instead, they train text classifiers using unlabeled documents with seed words, i.e., the selected representative words for categories. Actually, manually choosing seed words is significantly cheaper than labeling documents (Raghavan et al., 2006; Druck et al., 2008), saving many human efforts. This kind of dataless algorithm has empirically achieved promising classification results, and it has become a practical alternative to supervised learning algorithms, especially when the labeled documents are extremely expensive to obtain.

* corresponding author

Proceedings of the 27th International Conference on Computational Linguistics, pages 1908–1917
Santa Fe, New Mexico, USA, August 20-26, 2018.

A straightforward way of dataless classification is to construct a pseudo training dataset using the supervision information provided by seed words, before applying a traditional supervised classification algorithm (Liu et al., 2004; Druck et al., 2008). However, this methodology suffers from two problems. First, the documents containing no seed words can not be marked with any category label, resulting in a waste of instances to some extent. Unfortunately, there may exist a large number of such "unlabeled" documents, especially when the seed words are scarce. Second, the seed word based pseudo-labels are often quite noisy. That is because many documents may not contain the seed words from the true categories, but only contain the ones from unassociated categories.

To address the problems mentioned above, we develop a novel extension of the Naive Bayes classifier for dateless text classification, named pseudo-label based dataless Naive Bayes (PL-DNB). First, following (Nigam et al., 2000) we employ the expectation maximization (EM) algorithm to train PL-DNB in a semi-supervised manner, so that all the documents with and without pseudo-labels can be used for creating a classifier. Second, we iteratively update the pseudo-labels with highly acceptable confidence. The confidence values of pseudo-labels are measured by a mixture of seed word occurrences and estimations of label posteriors. Additionally, we aim to further avoid noisy pseudo-labels by preserving local neighborhood structure of documents. That is, for each document we update its pseudo-label by also considering the information of its nearest neighboring documents. We empirically compare the proposed PL-DNB algorithm against the dataless classification algorithms and traditional supervised algorithms. Experimental results indicate that our PL-DNB outperforms the existing dataless algorithms with seed words, and especially performs well on the imbalanced dataset

The rest of this paper is organized as follows: In Section 2, we introduce recent related works. In Section 3, we describe the proposed PL-DNB algorithm for dataless text classification. Section 4 presents the empirical results, and the conclusion is shown in Section 5.

2 Related Work

In this section, we review recent related works on dataless text classification and semi-supervised naive Bayes.

Dataless text classification There are some previous dataless text classifiers based on pseudo-labels. For example, the seed word naive Bayes (SNB) (Liu et al., 2004) computes information gain values of words over document clusters computed by k-means, and manually selects some top-ranked words as seed words to represent each category. It assigns pseudo-labels or probabilistic pseudo-labels to documents using these selected seed words. A naive Bayes classifier is then trained by the EM algorithm in a semi-supervised manner. Another dataless classification algorithm (Ko and Seo, 2004) constructs context-clusters as the basic unit using sliding windows, instead of documents. These context-clusters are associated with pseudo-labels using representative words occurring in category labels and titles. A naive Bayes classifier is also trained over these bootstrapped context-clusters. Compared with these previous algorithms, our PL-DNB creates pseudo-labels by further considering the information of nearest neighboring documents.This can lead to more accurate pseudo-labels than only using seed words.

Specifically, other dataless text classifiers aims to accurately categorize text documents by understanding the labels, i.e., embedding documents and category labels into a same semantic space (Chang et al., 2008; Palatucci et al., 2009; Elhoseiny et al., 2013; Song and Roth, 2014). The work of (Chang et al., 2008) builds a semantic space by exploiting the concepts of Wikipedia, and uses explicit semantic analysis to measure the distance between documents and labels. In some sense, this algorithm can be considered as a text classifier based on distant supervision (Mintz et al., 2008). Recently, the authors of (Song and Roth, 2014) further investigate dataless hierarchical text classification tasks. These dataless algorithms empirically performed well, however, they require auxiliary knowledge bases that are not always available.

Additionally, researchers have investigated some dataless text classifiers based on topic models, such as latent Dirichlet allocation (LDA) (Blei et al., 2003). The dataless classification algorithm proposed in (Hingmire et al., 2013), named classifyLDA, involves three steps: 1) learn a set of original topics by inferring the unsupervised LDA model over the concerned dataset; 2) manually assign a category

label to each topic, and then combine the topics that are associated with a same label into a new single topic; 3) classify test documents using the document-level posterior of those combined topics learned by unsupervised LDA. An extension algorithm (Hingmire and Chakraborti, 2014) of classifyLDA allows the original topics to be assigned to more than one category label in step 2, making the algorithm more flexible. Recently, the authors of (Li et al., 2016) propose a seed-guided topic model (STM), which simultaneously considers category word probability and initial document category distribution based on seed words. In STM, each category label is associated with a category-topic, and it further incorporates a new type of topic, i.e., general-topic, to filter out the noise words during Gibbs sampling. Empirical study in (Li et al., 2016) shows that STM consistently outperforms many existing dataless text classification algorithms. We present comparison results of this state-of-the-art STM in the experiment section.

Semi-supervised naive Bayes To the best of our knowledge, an early semi-supervised naive Bayes proposed in (Nigam et al., 2000) uses the EM algorithm to train a classifier over labeled and unlabeled documents, named NB-EM. The pooling multinomials algorithm (Yao et al., 2009) incorporates training documents with auxiliary knowledge, building a composite naive Bayes classifier for semantic analysis tasks. Recently, (Zhao et al., 2016) extends NB-EM by leveraging the word-level statistical constraint. These semi-supervised naive Bayes algorithms can be directly applied to text classification applications with a few labeled documents. However, manually collecting a small number of labeled documents remains expensive in many text analysis applications. In contrast, our PL-DNB only requires a much cheaper set of seed words, making it more practical.

3 Algorithm

We briefly review the framework of semi-supervised naive Bayes, and then present the proposed pseudo-label based dataless naive Bayes (PL-DNB) algorithm for dataless text classification with seed words.

3.1 Semi-supervised Naive Bayes

Suppose that there exists a collection of text documents $\mathcal{D}$, consisting of a subset of labeled documents $\mathcal{D}_L = \{(d_i, y_i)\}_{i=1}^{i=|\mathcal{D}_L|}$ and a subset of unlabeled ones $\mathcal{D}_U = \{d_i\}_{i=1}^{i=|\mathcal{D}_U|}$. Let $\mathcal{C} = \{c_i\}_{i=1}^{i=|\mathcal{C}|}$ and $\mathcal{W} = \{w_i\}_{i=1}^{i=|\mathcal{W}|}$ denote the set of category labels and the vocabulary of words, respectively.

The goal of semi-supervised text classification is to train a classifier over $\mathcal{D}$, which can automatically assign a correct category label for any test document. Semi-supervised naive Bayes is built on the Bayesian rule. Thanks to the bag-of-words assumption, it can classify a document *d* by a fully factored posterior distribution of labels:

$$y = \max_{c_i \in \mathcal{C}} \Pr(y = c_i|d), \qquad \Pr(y = c_i|d) \propto \Pr(y = c_i) \prod_{j=1}^{|\mathcal{W}|} \Pr(w_j|y = c_i)^{N_{d,w_j}} \tag{1}$$

where N_{d,w_j} is the occurrence number of word w_j in document *d*.

To achieve the parameter $\theta = \{\Pr(y), \Pr(w_j|y)\}$, semi-supervised naive Bayes employs the EM algorithm to maximize the following objective $\mathcal{L}(\theta)$, a mixture of the joint likelihood of labeled documents and marginal likelihood of unlabeled ones:

$$\mathcal{L}(\theta) = \sum\nolimits_{d \in \mathcal{D}_L} \log p(d, y) + \lambda \sum\nolimits_{d \in \mathcal{D}_U} \log p(d) \tag{2}$$

where $\lambda \in [0, 1]$ is a tuning parameter, controlling the importance of unlabeled documents.

3.2 Pseudo-label based Dataless Naive Bayes

Algorithm outline Overall, our PL-DNB is built on semi-supervised naive Bayes.

In the context of dataless classification, there are no labeled documents available. To address this, we initialize pseudo-labels $\widehat{y}^{(0)}$ for documents using seed word occurrences, leading to a set of documents with pseudo-labels $\mathcal{D}_L^{(0)}$ and a set of unlabeled documents that contain no seed words $\mathcal{D}_U^{(0)}$. After this initialization, PL-DNB iteratively performs the following two steps until the maximum iterative number is reached.

- **Step 1**: At each iteration *t*, use the EM algorithm to estimate the naive Bayes parameter $\theta^{(t)}$ by maximizing the following semi-supervised objective:

$$\mathcal{L}^{(t)}(\theta) = \sum\nolimits_{d \in \mathcal{D}_L^{(t-1)}} \log p(d, \widehat{y}^{(t-1)}) + \lambda \sum\nolimits_{d \in \mathcal{D}_U^{(t-1)}} \log p(d) \tag{3}$$

- **Step 2**: Given the optimum of $\theta^{(t)}$, compute the label posterior distributions for each document. We update pseudo-labels $\widehat{y}^{(t)}$ using these label posteriors and seed word occurrences. Only the pseudo-labels with acceptable confidence are left, leading to new document sets of $\mathcal{D}_L^{(t)}$ and $\mathcal{D}_U^{(t)}$.

Given the final optimum of θ, we can classify test documents using the label posterior distributions computed by Eq.1. For clarity, PL-DNB is summarized in *Algorithm 1*. We then introduce details of pseudo-label initialization, naive Bayes parameter θ update and pseudo-label $\widehat{y}$ update .

Algorithm 1 PL-DNB outline

1: **Initialize** pseudo-labels $\widehat{y}^{(0)}$ by seed word occurrences, obtaining $\mathcal{D}_L^{(0)}$ and $\mathcal{D}_U^{(0)}$
2: **For** $t = 1, 2, \ldots,$ MaxIter
3: **Update** the naive Bayes parameter $\theta^{(t)}$ using EM. Details are outlined in *Algorithm 2*
4: **Set** $\mathcal{D}_L^{(t)} = \mathcal{D}_U^{(t)} = \emptyset$
5: **For** $i = 1, 2, \ldots, |\mathcal{D}|$
6: **Assign** d_i a pseudo-label $\widehat{y}_i^{(t)}$ with acceptable confidence and **add** $(d_i, \widehat{y}_i^{(t)})$ into $\mathcal{D}_L^{(t)}$, otherwise
7: **add** d_i into $\mathcal{D}_U^{(t)}$
8: **End For**
9: **End for**

Pseudo-label initialization To incorporate the supervision information provided by seed words, we initialize pseudo-labels for each document using seed word occurrences. Since the selected seed words are representatives for categories, we suppose that a document containing more seed words of a category is more likely to be associated with this category. Following this, for each document *d* we compute the normalized seed word occurrence vector π_d as follows:

$$\pi_d(c_i) = \frac{SF_d(c_i) + \gamma}{\sum_{j=1}^{|\mathcal{C}|} SF_d(c_j) + |\mathcal{C}|\gamma} \tag{4}$$

where $SF_d(c_i)$ denotes the number of times that the seed words of category c_i have occurred in document *d*, and γ is a smoothing parameter used to avoid dividing by zero.[1] Then, the pseudo-label is initialized by the following rule:

$$\widehat{y}_d^{(0)} = \begin{cases} \infty & \text{if } \pi_d(c_1) = \pi_d(c_2), \cdots, = \pi_d(c_{|\mathcal{C}|}) \\ \underset{c_i \in \mathcal{C}}{\operatorname{argmax}}\, \pi_d(c_i) & \text{otherwise} \end{cases} \tag{5}$$

We do not assign a pseudo-label to document *d* without any seed word occurrence, denoted by ∞. We can obtain $\mathcal{D}_L^{(0)}$ and $\mathcal{D}_U^{(0)}$ by applying this initialization rule to all documents in $\mathcal{D}$.

Naive Bayes parameter θ update Given $\mathcal{D}_L^{(t-1)}$ and $\mathcal{D}_U^{(t-1)}$, the optimization of Eq.3 becomes a standard semi-supervised naive Bayes. We can use the EM algorithm to find the (local) optimum of $\theta^{(t)}$. This is achieved by iterating the following E-step and M-step. Due to the space limitation, we directly present the update equations without derivations.

In the **E-step**, we estimate the label posterior for each unlabeled document *d* using the Bayesian rule:

$$\Pr(y = c_i|d) = \frac{\Pr(y = c_i) \prod_{j=1}^{|\mathcal{W}|} \Pr(w_j|y = c_i)^{N_{d,w_j}}}{\sum_{c_h \in \mathcal{C}} \Pr(y = c_h) \prod_{j=1}^{|\mathcal{W}|} \Pr(w_j|y = c_h)^{N_{d,w_j}}} \tag{6}$$

[1] We empirically set γ to 0.01 in this work.

In the **M-step**, we update the naive Bayes parameter $\theta = \{\Pr(y), \Pr(w_j|y)\}$ with Laplace smoothing:

$$\Pr(y = c_i) = \frac{1 + \widehat{N}_{c_i} + \lambda \sum_{d \in \mathcal{D}_U} \Pr(y = c_i|d)}{|\mathcal{C}| + |\mathcal{D}_L| + \lambda |\mathcal{D}_U|} \tag{7}$$

$$\Pr(w_j|y = c_i) = \frac{1 + N_{c_i,w_j} + \lambda \sum_{d \in \mathcal{D}_U} N_{d,w_j} \Pr(y = c_i|d)}{|\mathcal{W}| + N_{c_i} + \lambda \sum_{d \in \mathcal{D}_U} \sum_{j=1}^{|\mathcal{W}|} N_{d,w_j} \Pr(y = c_i|d)} \tag{8}$$

where $\widehat{N}_{c_i}$ is the number of labeled documents marked with label c_i; N_{c_i,w_j} and N_{c_i} are the number of word w_j occurring and total number of words occurring in documents marked with label c_i, respectively. For clarity, we outline this EM procedure in *Algorithm 2*.

Algorithm 2 EM procedure outline for θ update

1: **Repeat**
2: **E-step**: Estimate the label posteriors for unlabeled documents using Eq.6
3: **M-step**: Update θ using Eqs.7 and 8
4: **Until convergence**

Pseudo-label $\widehat{y}$ update To update pseudo-labels $\widehat{y}^{(t)}$, for each document d we compute a label weight vector $\widehat{\pi}_d$ as follows[2]:

$$\widehat{\pi}_d(c_i) = \frac{\Pr^{(t)}(y = c_i|d) + \pi_d(c_i)}{2} \tag{9}$$

where $\Pr^{(t)}(y|d)$ is the label posterior distribution of document d estimated using the current $\theta^{(t)}$, and π_d the normalized seed word occurrence vector that has been shown in Eq.4. The pseudo-label is updated by the following rule:

$$\widehat{y}_d^{(t)} = \begin{cases} \underset{c_i \in \mathcal{C}}{\operatorname{argmax}}\, \widehat{\pi}_d(c_i) & \text{if } \max\limits_{c_i \in \mathcal{C}} \widehat{\pi}_d(c_i) > \delta \\ \infty & \text{otherwise} \end{cases} \tag{10}$$

where δ denotes an acceptable confidence threshold. This means that the dominate label in $\widehat{\pi}_d$ becomes the new pseudo-label for document d, if its weight in $\widehat{\pi}_d$ is larger than δ.

To avoid noisy pseudo-labels, we further consider the information of nearest neighboring documents. Following the intuition that the neighboring documents are more likely from a same category, we re-write the equation of $\widehat{\pi}_d$ by:

$$\widehat{\pi}_d(c_i) = \frac{\Pr^{(t)}(y = c_i|d) + \pi_d(c_i) + \sum_{j \in \Omega_d} \left(\Pr^{(t)}(y = c_i|j) + \pi_j(c_i) \right)}{2 \times (1 + |\Omega_d|)} \tag{11}$$

where Ω_d denotes the set of nearest neighboring documents for document d. In this work, we find Ω using the cosine similarity of TF-IDF representations, and empirically set $|\Omega_d|$ to 5.

4 Experiment

In this section, we empirically compare the proposed PL-DNB against both the existing dataless classification algorithms and traditional supervised algorithms.

4.1 Experimental Setup

Dataset and seed word set We employed two datasets of *Reuters*[3] and *Newsgroup*[4], which are usually used in text classification evaluations. In terms of *Reuters*, we left the 10 largest categories in the original

[2]Note that $\sum_{i=1}^{|\mathcal{C}|} \widehat{\pi}_d(c_i) = 1$

[3]http://kdd.ics.uci.edu/database/reuters21578/reuters21578.html

[4]http://qwone.com/∼jason/20Newsgroups/

Table 1: Statistics of datasets and sets of seed words. *#Train* / *#Test*: number of training/test documents; *#AvgDoc*: average length of documents; *#Label*: number of categories; *#Word*: number of unique words; S^L/S^D: average number of seed words in S^L/S^D.

Dataset	*#Train*	*#Test*	*#AvgDoc*	*#Label*	*#Word*	S^L	S^D
Reuters	5228	2057	70.8	10	7419	1	6
Newsgroup	11314	7532	152.1	20	52761	1	4.75

dataset, obtaining a dataset of 7285 documents in total. The training and test sets consist of 5228 and 2057 documents, respectively. Besides, the *Reuters* dataset is very imbalanced, where the largest category has 3713 documents but the smallest one has only 90 documents. In terms of *Newsgroup*, we used the "bydate" version, which contains totally 18846 documents divided into 20 categories. The training and test sets consist of 11314 and 7532 documents, respectively. The *Newsgroup* dataset is extremely balanced, where each category has about 900 documents. We apply traditional pre-processing methods to both datasets, i.e., removal of the standard stopwords, and the words that are shorter than 2 characters or occur in less than 5 documents.

Following (Chen et al., 2015; Li et al., 2016), we used two sets of seed words, denoted by S^L and S^D respectively. The seed words of S^L are directly selected from the category labels. We only left a single seed word for each category. The seed words of S^D are manually selected with the domain knowledge from additional candidate sets obtained by unsupervised learning algorithms. In contrast to S^L, the set of S^D contains more seed words.

For clarity, we show the statistics of the datasets and sets of seed words in Table 1.

Baseline algorithm To evaluate the effectiveness of PL-DNB, we compare it against four baseline algorithms, described as follows:

- **Seed word based naive Bayes (SNB)** (Liu et al., 2004): SNB is an early dataless naive Bayes algorithm that directly trains a classifier over the training documents with pseudo-labels in a semi-supervised manner. For a fair comparison, we omit the representative word selection step of SNB, but use the seed words of S^L and S^D just as other dataless algorithms

- **Seed-guided topic model (STM)** (Li et al., 2016): STM is a topic model based dataless text classification algorithm.[5] All crucial parameters of STM are tuned following the suggestions provided by its authors.

- **Naive Bayes (NB)**: We implement an in-house code of the supervised version of NB. The TF-IDF representation is used for documents.

- **Support vector machines (SVMs)**: We employ the *sklearn* tool[6] of SVMs with default parameter settings. The TF-IDF representation is also used.

Following the settings of (Li et al., 2016), we run dataless algorithms (i.e., SNB, STM and PL-DNB) over all documents, and only evaluate the classification results of test documents. For supervised algorithms (i.e., NB and SVMs), we train them over the training set, and also evaluate the results of test documents.

For the proposed PL-DNB, the numbers of the outer iteration and inner EM iteration are set to 10 and 5, respectively. The acceptable confidence threshold δ is set to 0.3. Besides, we empirically set the tuning parameter λ to 0 for *Reuters* and 0.3 for *Newsgroup*, respectively.

[5]This code is available at https://github.com/ly233/Seed-Guided-Topic-Model

[6]This tool is available at http://scikit-learn.org/stable/

Table 2: Evaluation results of Micro-F1. The superscript "‡" denotes that the best result of PL-DNB is statistically significant at 0.01 level.

Dataset	S^L			S^D			NB	SVMs
	PL-DNB	SNB	STM	PL-DNB	SNB	STM		
Reuters	0.847	0.633‡	0.819‡	0.895	0.826‡	**0.900**	0.921	0.973
Newsgroup	0.710	0.562‡	0.702‡	0.756	0.709‡	**0.766**	0.764	0.823

Table 3: Evaluation results of Macro-F1. The superscript "‡" denotes that the best result of PL-DNB is statistically significant at 0.01 level.

Dataset	S^L			S^D			NB	SVMs
	PL-DNB	SNB	STM	PL-DNB	SNB	STM		
Reuters	0.758	0.539‡	0.697‡	**0.832**	0.769‡	0.815‡	0.907	0.973
Newsgroup	0.670	0.508‡	0.668‡	**0.728**	0.661‡	0.724	0.753	0.823

4.2 Comparison of Baseline Algorithms

In the experiment, we use Micro-F1 and Macro-F1 to evaluate the classification performance. For each algorithm, we independently run it 10 times, and show the average results in Tables 2 and 3.

Comparison of dataless algorithms In contrast to SNB, we can observe that the proposed PL-DNB algorithm performs better in all settings. Overall, the performance gain of Micro-F1 is about 0.05∼0.21, and that of Macro-F1 is about 0.06∼0.22. More importantly, PL-DNB significantly outperforms SNB with the seed word set of S^L. The result indicates that PL-DNB can achieve high classification performance with very few seed words. This makes PL-DNB more practical for the text analysis tasks, where even seed words are expensive to obtain. Additionally, since the main difference between SNB and PL-DNB is that PL-DNB exploits nearest neighboring documents to improve the pseudo-label update, the performance gain over SNB indicates that this nearest neighbor scheme is effective.

In contrast to STM, we can observe that PL-DNB gets higher scores in most settings. In terms of S^L, the Micro-F1 and Macro-F1 scores of PL-DNB are higher than those of STM across both datasets. For example, the improvements of Micro-F1 and Macro-F1 over STM are about 0.03 and 0.06 on *Reuters*, respectively. This further supports the above observation that PL-DNB can perform well with a few seed words. In terms of S^D, PL-DNB is competitive with STM, where PL-DNB gets higher Macro-F1 scores but slightly lower Micro-F1 scores.

Comparing between the two dataless baselines, STM gains better performance in all settings. This result is consistent with the empirical results reported in (Li et al., 2016). The possible reason is that STM incorporates the category word probability to exploit the discriminative power of words, which can effectively improve classification results.

Comparison of supervised algorithms First, we can observe that PL-DNB is a little worse than NB, where the performance gap is not very obvious. For example, the Micro-F1 and Macro-F1 scores of PL-DNB+S^D are only about 0.01 and 0.025 lower than those of NB on *Newsgroup*. That is, PL-DNB may approach the supervised version of NB given sufficient seed words. Second, PL-DNB performs worse than SVMs, however it uses very limited supervision during classifier training. In some sense, our PL-DNB can be considered as an important candidate algorithm for text classification, especially when collecting labeled documents for training is extremely expensive.

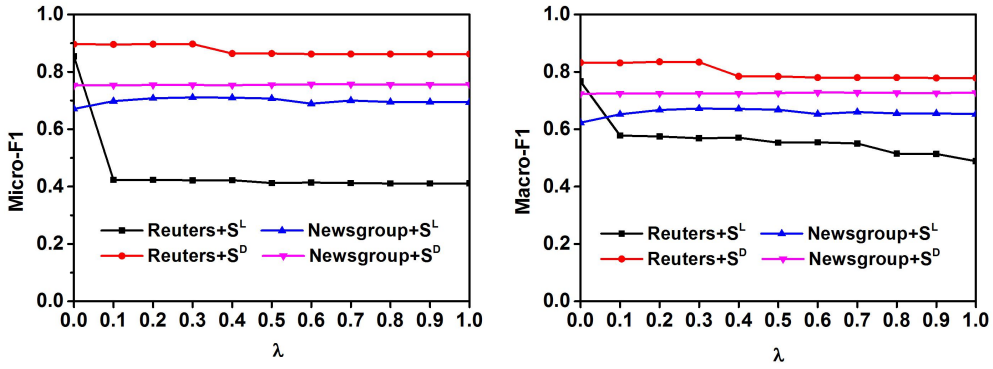


Figure 1: Evaluation on different values of λ

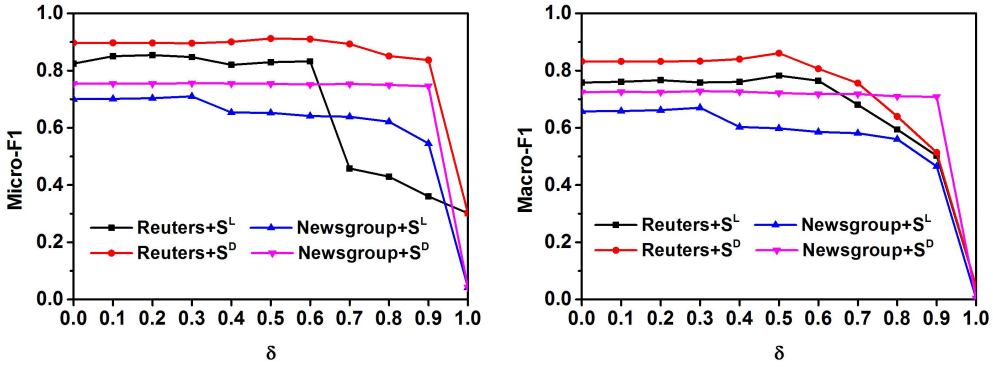


Figure 2: Evaluation on different values of δ

4.3 Evaluation of Parameters

In this section, we empirically evaluate two crucial parameters of PL-DNB, i.e., λ and δ.

Analysis of λ The parameter λ is used to control the importance of unlabeled documents during semi-supervised learning (ref. Eq.3). Note that $\lambda = 1$ means that unlabeled documents are equally important with labeled ones, and $\lambda = 0$ means that unlabeled documents are not utilized.

To evaluate the impact of λ, we examine the classification results of different values of $\lambda \in \{0, 0.1, 0.2 \cdots, 1\}$ when δ is fixed to 0.3. The experimental results of Micro-F1 and Macro-F1 are shown in Figure 1.

In terms of *Reuters*, we can observe that PL-DNB achieves higher scores when the value of λ is relatively smaller. Especially when the seed word set of S^L is utilized, the best scores are achieved at $\lambda = 0$, and significantly higher than scores of other λ values. The Micro-F1 score of $\lambda = 0$ is over 0.85, but those of other λ values are only about 0.4. Besides, the gap of Macro-F1 between $\lambda = 0$ and other values is about 0.2. Since S^L provides only one seed words for each category, a great majority of documents can not be initialized with pseudo-labels. At the beginning of optimization iterations, these unlabeled documents that are without any supervision prefer equally contributing to all categories, resulting in worse estimations of naive Bayes parameters. This problem grows worse for imbalanced datasets, because the smaller categories are much more sensitive to the noise. Additionally, smaller values of λ still perform better than larger ones when using the seed word set of S^D, but the gap is not obvious. The reason is that S^D can assign pseudo-labels to most of the documents. In summary, this evaluation result implies that for the imbalanced dataset PL-DNB prefers smaller λ values. We thus set $\lambda = 0$ for *Reuters* in our experiment.

In terms of *Newsgroup*, we can roughly observe that PL-DNB performs stable as λ varies. Overall, the

reason may be that *Newsgroup* is very balanced, so that PL-DNB is insensitive to unlabeled documents at the beginning of optimization iterations. For the setting of *Newsgroup*+S^D, both Micro-F1 and Macro-F1 scores are mostly the same with different values of λ. This is because only a small number of documents can not be initialized with pseudo-labels[7] by S^D. That is to say, λ only affects a few documents during the semi-supervised optimization step. Additionally, we see that the scores of PL-DNB are a bit higher when λ is in the interval $[0.1, 0.5]$ when using S^L. We empirically set $\lambda = 0.3$ for *Newsgroup* in our experiment.

Analysis of δ The parameter δ is a threshold used to determine whether a document can be marked with a pseudo-label in the pseudo-label update step (ref. Eq.10). On one hand, $\delta = 0$ means that all documents will be associated with pseudo-labels; On the other hand, a higher δ value implies that the updated pseudo-labels must be more "accurate".

We evaluate the classification results of different values of $\delta \in \{0, 0.1, 0.2 \cdots, 1\}$ by holding λ fixed to 0 for *Reuters* and 0.3 for *Newsgroup* respectively. The experimental results of Micro-F1 and Macro-F1 are shown in Figure 2.

Overall, we can observe that the classification performance is fast dropped as δ becomes larger. For example, the Micro-F1 score drops from 0.8 to 0.4 across *Reuters*+S^L when δ becomes larger; and especially all scores of $\delta = 1$ are very poor. The main reason of this trend is that the number of documents with pseudo-labels decreases as δ becomes larger. That is, less labeled documents can be used for training naive Bayes (ref. Eq.3). This result indicates that in the context of dataless classification, the number of labeled documents may be a bit more important then the quality of pseudo-labels.

In terms of *Reuters*, PL-DNB performs relatively stable as $\delta \in [0.1, 0.6]$, but worse when δ becomes larger than 0.6. In contrast, the performance of *Newsgroup* gets stable before δ achieves 0.9, especially when the set of S^D is utilized. This comparison indicates that the balanced dataset may be less sensitive to the value of δ than the imbalanced dataset. Additionally, the best scores of *Reuters* are achieved at $\delta = 0.5$ in most settings, but it is only a bit better than $\delta = 0.1, 0.2, 0.3, 0.4, 0.6$. The best scores of *Newsgroup* are mostly at $\delta = 0.3$. Empirically, we set δ to 0.3 for both datasets in our experiment.

5 Conclusion

In this paper, we develop a novel dataless PL-DNB classification algorithm, which can be directly trained on unlabeled documents with seed words, instead of labeled documents. In PL-DNB, we create pseudo-labels for documents and train a naive Bayes classifier over the pseudo training set in semi-supervised learning. The pseudo-labels are iteratively updated, and the information of nearest neighboring documents is considered to avoid noisy pseudo-labels. We examine the effective of PL-DNB on two popular datasets in classification evaluations, where one is imbalanced and the other is balanced. Empirical results indicate that PL-DNB performs better than the state-of-the-art dataless text classifiers, especially for the imbalanced dataset. Specially, PL-DNB achieves competitive performance with supervised naive Bayes in some settings.

Acknowledgements

We would like to acknowledge support for this project from the National Natural Science Foundation of China (NSFC) (grant numbers 61602204, 61572226 and 61472157).

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Gavin C. Cawley and Nicola L. C. Talbot. 2010. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11:2079–2107.

[7] In the current setting, only about 23% documents have no initialized pseudo-labels.

Ming-Wei Chang, Lev Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of semantic representation: dataless classification. In *AAAI Conference on Artificial Intelligence*, pages 830–835.

Xingyuan Chen, Yunqing Xia, Peng Jin1, and John Carroll. 2015. Dataless text classification with descriptive LDA. In *AAAI Conference on Artificial Intelligence*, pages 2224–2231.

Doug Downey and Oren Etzioni. 2008. Look ma, no hands: analyzing the monotonic feature abstraction for text classification. In *Neural Information Processing Systems*, pages 393–400.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 595–602.

Mohamed Elhoseiny, Babak Saleh, and Ahmed M. Elgammal. 2013. Write a classifier: Zero-shot learning using purely textual descriptions. In *IEEE International Conference on Computer Vision*.

Swapnil Hingmire and Sutanu Chakraborti. 2014. Topic labeled text classification: A weakly supervised approach. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 385–394.

Swapnil Hingmire, Sandeep Chougule, and Girish K. Palshikar. 2013. Document classification by topic labeling. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 877–880.

Wenbo Hu, Jun Zhu, Hang Su, Jingwei Zhuo, and Bo Zhang. 2017. Semi-supervised max-margin topic model with manifold posterior regularization. In *International Joint Conference on Artificial Intelligence*, pages 1865–1871.

Youngjoong Ko and Jungyun Seo. 2004. Learning with unlabeled data for text categorization using bootstrapping and feature projection techniques. In *Annual Meeting on Association for Computational Linguistics*.

Chenliang Li, Jian Xing, Aixin Sun, and Zongyang Ma. 2016. Effective document labeling with very few seed words: a topic modeling approach. In *ACM International on Conference on Information and Knowledge Management*, pages 85–94.

Bing Liu, Xiaoli Li, Wee Sun Lee, , and Philip S. Yu. 2004. Text classification by labeling words. In *AAAI Conference on Artificial Intelligence*, pages 425–430.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2008. Distant supervision for relation extraction without labeled data. In *Annual Meeting of the Association for Computational Linguistics*, pages 1003–1011.

Kamal Nigam, Andrew Kachites Mccallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2-3):103–134.

Mark Palatucci, Dean Pomerleau, Geoffrey Hinton, and Tom M. Mitchell. 2009. Zero-shot learning with semantic output codes. In *Neural Information Processing Systems*, pages 1410–1418.

Hema Raghavan, Omid Madani, and Rosie Jones. 2006. Active learning with feedback on features and instances. *Journal of Machine Learning Research*, 7:1655–1686.

Yangqiu Song and Dan Roth. 2014. On dataless hierarchical text classification. In *AAAI Conference on Artificial Intelligence*.

Limin Yao, David Mimno, and Andrew McCallum. 2009. Efficient methods for topic model inference on streaming document collections. In *International Conference on Knowledge Discovery and Data Mining*.

Li Zhao, Minlie Huang, Ziyu Yao, Rongwei Su, Yingying Jiang, and Xiaoyan Zhu. 2016. Semi-supervised multinomial naive bayes for text classification by leveragingword-level statistical constraint. In *AAAI Conference on Artificial Intelligence*.

Visual Question Answering Dataset for Bilingual Image Understanding: A Study of Cross-Lingual Transfer Using Attention Maps

Nobuyuki Shimizu[1], **Na Rong**[*2] , **Takashi Miyazaki**[1]
[1] Yahoo Japan Corporation, Tokyo, Japan
[2] Tokyo Institute of Technology, Tokyo, Japan
nobushim@yahoo-corp.jp, na@ks.cs.titech.ac.jp, takmiyaz@yahoo-corp.jp

Abstract

Visual question answering (VQA) is a challenging task that requires a computer system to understand both a question and an image. While there is much research on VQA in English, there is a lack of datasets for other languages, and English annotation is not directly applicable in those languages. To deal with this, we have created a Japanese VQA dataset by using crowdsourced annotation with images from the Visual Genome dataset. This is the first such dataset in Japanese. As another contribution, we propose a cross-lingual method for making use of English annotation to improve a Japanese VQA system. The proposed method is based on a popular VQA method that uses an attention mechanism. We use attention maps generated from English questions to help improve the Japanese VQA task. The proposed method experimentally performed better than simply using a monolingual corpus, which demonstrates the effectiveness of using attention maps to transfer cross-lingual information.

1 Introduction

Visual question answering (VQA) is the automated task of answering questions about a given image, which is a difficult task because the computer system must understand both the question (natural language processing, or NLP) and the image (computer vision). Recently there has been much research on VQA but the existing literature focuses only on English. Because each language differs in its grammar and resources, there is a pressing need to develop VQA systems for different languages. The effort to develop such systems is hindered by a lack of language resources. In this paper, we discuss two ways to address the lack of datasets for languages besides English. First, by using images from the Visual Genome dataset (Krishna et al., 2016) and annotation through crowdsourcing, we have created the first VQA dataset for Japanese. We call it the Japanese Visual Genome VQA dataset. It consists of 99,208 images with a total of 793,664 Japanese question answering (QA) pairs, for an average of eight QA pairs per image. The examples of obtained Japanese questions are shown along with the English version of Visual Genome questions in Figure 1. Second, we propose a cross-lingual method of exploiting the information in an existing English dataset to help improve the performance of our Japanese system in a VQA setting.

The difficulty of applying a cross-lingual method for VQA is the diversity of questions that one can ask about an image. For image captioning (Miyazaki and Shimizu, 2016), English and Japanese captions on the same image are generally considered as comparable corpora. Questions for VQA, however, can be quite diverse as compared to image captions, because they are not necessarily specific to the image. For example, questions about the time of day can be asked for almost any image available. As such questions might appear in one language but not in another, it is much tougher to exploit questions across languages.

On the other hand, striking features of an image are usually picked up by several questions. The motivation of our approach is based on experiences as a bilingual and a second language learner. Sometimes

[*] This study was conducted during an internship at Yahoo! JAPAN Research, Tokyo, Japan

Proceedings of the 27th International Conference on Computational Linguistics, pages 1918–1928
Santa Fe, New Mexico, USA, August 20-26, 2018.

(a)
J1: kono norimono ha nandesuka
J2: hikouki no kitai ha naniiro desuka – E2
J3: kitai no moji ha naniiro desuka
J4: hikouki ha nani wo shiteimasuka
J5: hikouki ha doko ni imasuka – E14
J6: kokoha doko desuka – E1
J7: hikouki ha dochira ni hashitte imasuka
J8: enjin ha doko ni tsuite imasuka – E15

(b)
E1: Where was this picture taken?
E2: What color is the plane?
E3: When was this picture taken?
E4: How many planes are in the picture?
E5: What color is the sky?
E6: What color is the grass?
E7: Where is the body?
E8: What besides smoke could be seen behind plane?
E9: How could the sky be described?
E10: What is seen immediately behind runway?
E11: What is seen in the foreground of photo?
E12: How could the hills at skyline be described?
E13: What appears to be in the field on far right of photo?
E14: Where is the plane?
E15: Where is the turbine engine?
E16: What is red and white on the runway?
E17: What is in the distance?
E18: What website is on the bottom of the photo?
E19: What is under the plane?

Figure 1: Examples of (a) Japanese questions from our dataset and (b) English questions from the Visual Genome dataset. A Japanese question with an equivalent English question is matched, for example J2 – E2. The translations of J1, J3, J4, and J7 are as follows. J1: What form of transport is it? J3: What is the color of the letters on the body? J4: What is the airplane doing? J7: In which direction is the airplane going?

we can infer what a foreign language speaker is saying simply from the situation we are in. If we happen to be paying attention to the same object, we share the contexts and are likely to have similar perceptions.

We thus propose the first-ever method for cross-lingual VQA, enabling us to exploit questions in English to improve the performance of our VQA system in Japanese. In our method, we first generate attention maps for the English questions about an image. We then input these attention maps to the Japanese VQA system. We use the method given in Lu et al. (2016) both to generate the attention maps and to provide an experimental baseline. Our proposed cross-lingual method is simple yet effective, as we simply replace the input image feature from Lu et al. (2016) in our Japanese system with the attention maps generated from English questions. In our experiment, the proposed method using cross-lingual information produced better results than did the method using only a monolingual corpus. Because the attention maps generated from English questions tended to focus on salient objects in the images, our results also suggest the possibility of improving VQA through saliency modeling.

Our contribution is thus twofold. First, we have created a Japanese VQA dataset, which we believe is the first such dataset to be released[1]. Second, by using cross-lingual information to perform the VQA task, we obtain better results than by simply using a monolingual corpus, demonstrating the effectiveness of cross-lingual information.

The remainder of the paper is organized as follows. In section 2, we discuss English VQA datasets and related work using attention maps for the VQA task. In section 3 we introduce our dataset, including its construction and statistics. We then explain both the method used to generate attention maps from English questions and our proposed cross-lingual method in section 4. Experimental results are given in section 5 and we conclude the paper with a brief summary in section 6.

2 Related Work

In this section we discuss three related topics: first, four English datasets that have been generally used in VQA; second, some recent work on VQA with attention mechanisms, which have become a popular approach for VQA tasks; and last, cross-lingual models used in NLP. The attention-based approach involves generating image regions, called "attention maps," that are relevant to answering questions and then using the attention maps to generate answers.

[1]The dataset is to be released at `https://research-lab.yahoo.co.jp/en/software/`.

2.1 Datasets

A typical VQA dataset contains at least an image, a question for the image and the answer. Sometimes additional annotations, such as image regions relevant to the answers, or image captions, are provided as well. A number of datasets have been proposed for the VQA task in recent years; however, most of them are in English. In the following paragraphs we list the VQA datasets that have been generally used in this field and one multi-lingual VQA dataset with Chinese original VQA pairs and their English translations. The details are presented below.

The VQA dataset (Antol et al., 2015) contains 614,163 questions and 7,984,119 answers (including answers provided by humans either looking or not looking at the corresponding images) for 204,721 images from Microsofts COCO dataset (Lin et al., 2014), along with 150,000 questions and 1,950,000 answers for 50,000 abstract scenes. This dataset has two modalities for answering questions: open-ended and multiple-choice; in contrast, our dataset focuses on open-ended answers.

The Visual Genome (Krishna et al., 2016) dataset contains 1,773,258 QA pairs. On average, an image has 17 QA pairs. This dataset has two types of QA pairs (freeform QAs, which are based on the entire image, and region-based QAs, which are based on selected regions of the image) and six types of questions (what, where, when, who, why, and how). A subset of the freeform QA portion of Visual Genome is released as Visual 7W (Zhu et al., 2016). Because the release date of Visual 7W precedes that of Visual Genome, evaluations of VQA systems on freeform QA pairs are often conducted in Visual 7W instead of Visual Genome.

The COCO-QA (Ren et al., 2015) dataset was automatically generated from captions in the COCO dataset. It contains 78,736 training questions and 38,948 test questions, with the questions divided into four types: object questions, number questions, color questions, and location questions. Each answer consists of one word.

The DAQUAR (Malinowski and Fritz, 2014) dataset was built on top of the NYU-Depth V2 dataset (Silberman et al., 2012).The answers are mostly single-word answers. The complete dataset has 6,795 training QA pairs and 5,674 test pairs. On average, an image has nine QA pairs and the answers encompass 894 categories. The Reduced DAQUAR dataset is a subset of the complete DAQUAR dataset and contains 3,876 training samples, 297 test samples, and answers in 37 categories.

The Chinese Baidu dataset FM-IQA (Gao et al., 2015) uses 123,287 images sourced from the COCO dataset. The difference from COCO-QA is that the questions and answers were provided by human annotators through a crowdsourcing platform operated by Baidu. The annotators were free to add any type of question if it related to the content of the given image. This led to a much greater diversity of questions than in previously available datasets. Answering such questions typically requires both understanding the visual content of the image and incorporating prior "common sense" information. The dataset contains 120,360 images and 250,560 QA pairs, which were originally provided in Chinese and then converted to English by human translators.

Besides FM-IQA, the lack of datasets in languages other than English is striking. This situation hinders VQA research in other languages, such as Japanese. We intend our dataset to remedy this situation by providing resources in a morphologically rich language for the first time.

2.2 Attention-Based Methods for VQA

Previous studies have used information from whole images, but many questions and answers relate specifically to local regions in images. Many recent studies have thus focused on attention models, which select image regions relevant to answering questions, to deal with the VQA task (Xu and Saenko, 2016; Xiong et al., 2016; Shih et al., 2016; Yang et al., 2016; Lu et al., 2016). Shih et al. (2016) developed an approach for learning to answer visual questions by selecting image regions relevant to a text-based query. The approach maps textual queries and visual features from various regions into a shared space in which they are compared for relevance by applying an inner product. Yang et al. (2016) presented stacked attention networks (SANs), which account for the fact that VQA often requires multiple reasoning steps. The stacked attention model locates image regions relevant to the question for answer prediction via multi-step reasoning. While the above two approaches only use attention maps based on image regions,

Lu et al. (2016) proposed a co-attention model for VQA, which jointly considers attention to both the image and the question. Attention to the question (which should be valued equally with attention to the image) represents the importance, in terms of probability, of each word in the question for answering the question. Because Lu et al. (2016) released work describing the implementation of this co-attention model and their results are easy to reproduce, we adopted their implementation as the foundation of our proposed method. We also used their work as a monolingual baseline.

2.3 Cross-Lingual Model

While resources in English have been quickly developed and are abundant by now, other languages have fallen behind in terms of the size and variety of their resources. To remedy this, some research leverages the existing knowledge in English to help process other languages. These multilingual resources capture valuable information that can be used in many fields and is especially useful for languages lacking resources. Many cross-lingual models have been proposed in NLP. These models are trained on a parallel corpus and find ways to connect between two languages, usually through supervised or semi-supervised learning. Cross-lingual information retrieval is another task that requires cross-lingual modeling (Jagarlamudi and Kumaran, 2007; Ballesteros and Croft, 1996), in which the query and results are written in different languages. In word embedding research, some studies have tried to transfer linguistic knowledge from one language to another, especially from English to low-resource languages, through distributed representations at the word level (Hermann and Blunsom, 2013; Haghighi et al., 2008; Klementiev et al., 2012).For image captions, Miyazaki and Shimizu (2016) proposed a caption generation model that transfers cross-lingual information from English to Japanese through pretraining of the model. Our proposed method shows that image attention maps can be used to transfer information cross-lingually. While the transfer in Miyazaki and Shimizu (2016) occurs at a fully connected layer after fc7 in the VGG16 model, our information transfer occurs at the attention-map level and is spatially interpretable.

Cross-lingual information is very useful, especially when dealing with sparse language resources. Although many cross-lingual learning models have been proposed, to our knowledge there has been no such prior research for the VQA task. This paper, we believe, reports the first work done on using cross-lingual information in the VQA field.

3 Statistics for Japanese Dataset

To create a Japanese version of a VQA corpus that would be comparable to the English version, we chose the Visual Genome dataset as a starting point. As noted previously, Visual Genome has two types of QA pairs: freeform and region-based. Our dataset is meant to be comparable to the freeform QA part of Visual Genome, which is similar to other VQA corpora, except for its focus on six types of questions (what, where, when, who, why, and how). Krishna et al. (2016) stated the benefits of focusing on those six question types as follows:

> First, they offer a considerable coverage of question types, ranging from basic perceptual tasks (e.g. recognizing objects and scenes) to complex common sense reasoning (e.g. inferring motivations of people and causality of events). Second, these categories present a natural and consistent stratification of task difficulty ... For instance, why questions that involve complex reasoning lead to the poorest performance ... of the six categories. This enables us to obtain a better understanding of the strengths and weaknesses of todays computer vision models, which sheds light on future directions in which to proceed.

3.1 Crowdsourcing Procedure

Visual Genome was created with Amazon Mechanical Turk (AMT), a well-known crowdsourcing platform for microtasks. To create a comparable Japanese dataset, we used a similar platform called Yahoo! Crowdsourcing, operated by Yahoo Japan Corporation. While AMT participants can be from anywhere in the world and have any mother language, participants in Yahoo! Crowdsourcing can be safely assumed to be proficient in Japanese, since such proficiency is required for signing up, navigating the user interface, and participating in the microtask market.

For collecting the freeform QA data in Visual Genome, consider a procedure with the following conditions: (1) Crowdsourcing participants are asked to view an image and write eight QA pairs related to it. (2) The participants are instructed that each question should start with one of the six question words - what, where, when, who, why, and how. (3) To encourage diversity, the participants are asked to write questions using at least three different question words out of the six.

While the above conditions are simple for English, we encountered several problems in creating a Japanese procedure with similar conditions. The second condition was especially troublesome. First, in the Japanese language, creating an interrogative sentence out of a declarative one does not require changing the word order. Since Japanese questions typically do not start with a question word, asking participants to write a question starting with such a word did not make sense. Second, just as English allows the phrase "what time" instead of "when," Japanese allows "what place" for "where," "what reason" for "why," "what number" for "how many," and so on. We thus attempted to list Japanese interrogative words similar to the six English question words used in Visual Genome: *nani* (what), *dare* (who), *doko* (where), *donna* (what kind), *dorekurai* (how much), *dou* (how), *itsu* (when), *ikutsu* (how many), and *naze* (why). We recognized, however, that the Japanese equivalent *nani* for "what" could become a catch-all category used for questions that would otherwise be asked with when/where/how/why and so on in English.

Once we came up with the Japanese equivalents of the six question words, we posted a pilot task that asked participants to view an image and write eight QA pairs. We found that some participants reused the same QA pairs over and over, on the basis of common knowledge. Consider, for example, the QA pair of "What color is the sky?" and "Blue." Such a pair could be applicable to any outdoor image in the daytime. Given this experience with the pilot task, we modified the instructions to disallow such repeated QA pairs. All instructions and examples are provided in Japanese. The following instructions (translated into English) are the modified version:

> Please enter eight pairs consisting of a question and its answer in Japanese about the image linked by the URL.
>
> Please follow the following four rules in writing the question/answer pairs.
>
> Rule 1: Every question must use one of the following question words: *nani* (what), *dare* (who), *doko* (where), *donna* (what kind), *dorekurai* (how much), *dou* (how), *itsu* (when), *ikutsu* (how many), or *naze* (why). Example of a rule violation: "Question: Which is it, raining or sunny?" Reason for the violation: The word "which" is not in the list of question words.
>
> Rule 2: Every question must be distinct and ask something different. Furthermore, the eight questions must include at least three different question words from the list. Example of a rule violation: The eight questions use only the question words *nani* (what) and *dare* (who). Reason for the violation: The eight questions do not use at least three of the question words from the list.
>
> Rule 3: All eight questions must have a commonly agreed answer. Example of a rule violation: "Question: What is the man in the image thinking?" Reason for the violation: The answer is not commonly agreed.
>
> Rule 4: All eight questions must require the image content for the correct answer. Example of a rule violation: "Question: What is the color of the sunset? Answer: Red." Reason for the violation: This question is answerable with common knowledge, so it does not require the image content.

After conducting the pilot task, we examined the results and selected promising participants (comprising a whitelist) for future task requests, so that only participants on the whitelist could perform the next task. We repeated this selection process until the final whitelist included about 1,500 participants. About 200-250 of them regularly participated in the actual VQA collection task. We posted tasks in small batches over the course of six months to prevent participants from working long hours. Despite

	Nine question types									other
	nani what	*dare* who	*doko* where	*donna* what kind	*dorekurai* how much	*dou* how	*itsu* when	*ikutsu* how many	*naze* why	
Whole set	386,113	21,294	78,568	146,493	21,195	24,826	85,893	15,748	3,135	10,883
Test set	29,765	1,392	6,076	11,167	1,586	1,743	6,408	1,066	215	1,124

Table 1: Numbers of QA pairs in each category.

these measures, some participants eventually started to enter short, meaningless characters. To remove such noise, we removed sets of eight QA pairs whose question lengths averaged less than four characters.

3.2 Dataset Statistics and Analysis

The resulting Japanese VQA dataset consists of 99,208 images from Visual Genome, together with 793,664 QA pairs in Japanese, since every image has eight QA pairs.

Data quality. To ensure the consistency and integrity of the corpus, we randomly sampled and manually checked 800 QA pairs. Among those pairs, we found that six contained a minor typo, ten had a wrong answer, and two had an ambiguous question. In addition, four had questions answerable with common knowledge, while 29 had questions that were not based on one of the six question words and thus answerable with "yes" or "no." Thus, we found that 93.6% of the QA pairs correctly conformed to the specification. If we include the four pairs related to common knowledge, the figure increases to 94.1%, and if we also include the 29 pairs with a yes-no question and the six questions with a minor typo, it increases to 98.5%. Overall, we found that the quality of the QA pairs was very good.

Question type distribution. Unlike with English questions, which are easily classified according to the six question words (what, where, when, who, why, how), with Japanese questions grammar makes such classification more difficult. As mentioned in section 3.1, the questions in the Japanese dataset were classified into nine question types by nine words: *nani* (what), *dare* (who), *doko* (where), *donna* (what kind), *dorekurai* (how much), *dou* (how), *itsu* (when), *ikutsu* (how many), and *naze* (why). These nine words cover most Japanese questions. In addition to checking 800 QA pairs, we also examined 100 sets of eight QA pairs each and found twelve sets of questions that did not have at least three different question words. While the crowdsourcing participants sometimes did not follow the instructions exactly, we believe that this did not significantly decrease the diversity of question types. As shown by the statistics listed in Table 1, the question types in the corpus varied considerably.

Question and answer length distributions. We used MeCab to tokenize the Japanese questions and answers. Figure 2 shows the resulting distribution of average lengths for each category.

Comparison with English version. Manual examination of our corpus revealed that questions were very similar to the original English version of Visual Genome. An image with Japanese and English samples is shown in Figure 1. Generally, there are two to three times more QA pairs per image for the English version than for the Japanese version and questions often overlap. In Figure 1, J2 and E2, J5 and E14, J5 and E14, J6 and E1, and J8 and E15 are paired because they are essentially the same question.

4 Methodology

The method introduced in section 4.1 was used both to generate attention maps from English questions and to provide an experimental baseline. Section 4.2 describes how we used cross-lingual information (that is, the attention maps generated from the English data) to improve the performance of the Japanese VQA system.

4.1 Baseline

In this section, we briefly introduce material from Lu et al. (2016). That work proposed a VQA co-attention model, which jointly considers attention to both the image and the question. We used this approach for two purposes: first, to generate visual attention maps for English; and second, to provide a baseline for our experiment.

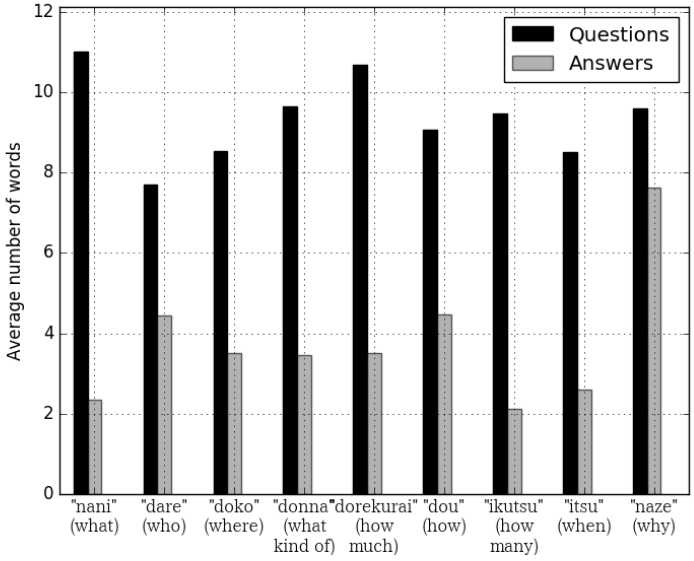


Figure 2: Average lengths of questions and answers for each question type.

A question consisting of T words is represented by $\mathbf{Q} = \{\mathbf{q}_1, ..., \mathbf{q}_T\}$, where $\mathbf{q}_t$ is the feature vector for the t-th word. Then, $\mathbf{q}_t^w$, $\mathbf{q}_t^p$, and $\mathbf{q}_t^s$ represent respectively the word embedding, phrase embedding, and question embedding at position t. The feature vectors are extracted the same way as in Lu et al. (2016) using LSTM (Hochreiter and Schmidhuber, 1997). The image feature is represented by

$$\mathbf{V} = \{\mathbf{v}_1, ..., \mathbf{v}_i, ..., \mathbf{v}_N\}, \tag{1}$$

where $\mathbf{v}_i$ is the feature vector for the i-th spatial location. N is the number of grids in an image, which in our case is 96 (14 by 14). The co-attention features of the question and image at each level in the hierarchy are respectively denoted as $\hat{\mathbf{q}}^r$ and $\hat{\mathbf{v}}^r$, where $r \in \{w, p, s\}$ (i.e., the level of a word, phrase, or question).

As noted in section 2, there are two attention maps in this model, for attention to the image and to the question. Depending on the order in which the image and question attention maps are generated, there are two co-attention mechanisms in Lu et al. (2016): parallel co-attention, and alternating co-attention. Because Lu et al. showed that parallel co-attention outperforms alternating co-attention, we chose the former as a baseline model and building block of our proposed model. We forgo explaining the mechanism of the latter here.

Parallel Co-Attention. The parallel co-attention mechanism generates the image and question attention maps simultaneously. Given an image feature map $\mathbf{V} \in R^{d \times N}$ and a question representation $\mathbf{Q} \in R^{d \times T}$, the mechanism calculates the similarity between the image and question features for all pairs of image locations and question positions:

$$\mathbf{C} = \tanh(\mathbf{Q}^\top \mathbf{W}_b \mathbf{V}), \tag{2}$$

where $\mathbf{C} \in R^{T \times N}$ is the resulting affinity matrix, and $\mathbf{W}_b \in R^{d \times d}$ contains weights. Then, the mechanism uses the affinity matrix as a feature for learning to predict the image and question attention maps:

$$\begin{aligned}
\mathbf{H}^v &= \tanh(\mathbf{W}_v \mathbf{V} + (\mathbf{W}_q \mathbf{Q})\mathbf{C}) \\
\mathbf{H}^q &= \tanh(\mathbf{W}_q \mathbf{Q} + (\mathbf{W}_v \mathbf{V})\mathbf{C}^\top) \\
\mathbf{a}^v &= softmax(\mathbf{w}_{hv}^\top \mathbf{H}^v) \\
\mathbf{a}^q &= softmax(\mathbf{w}_{hq}^\top \mathbf{H}^q),
\end{aligned} \tag{3}$$

where $\mathbf{W}_v, \mathbf{W}_q \in R^{k \times d}$, $\mathbf{w}_{hv}$, and $\mathbf{w}_{hq} \in R^k$ are weight parameters. Then, the image attention map is $\mathbf{a}^v \in R^N$ and the question attention map is $\mathbf{a}^q \in R^T$. These represent respectively the attention probabilities for each image region $\mathbf{v_n}$ and word $\mathbf{q_t}$. Finally, the mechanism calculates the image and question attention vectors:

$$\hat{\mathbf{v}} = \sum_{n=1}^{N} a_n^v \mathbf{v}_n, \ \hat{\mathbf{q}} = \sum_{t=1}^{T} a_t^q \mathbf{q}_t. \tag{4}$$

The parallel co-attention mechanism is applied at each level in the hierarchy, leading to $\hat{\mathbf{v}}^r$ and $\hat{\mathbf{q}}^r$, where $r \in \{w, p, s\}$.

4.2 Proposed Method

The goal of our method is to use the information learned from the English dataset to help improve the performance of the Japanese dataset. First, we use the method introduced in the previous section to generate attention maps for each image by using English questions. Then, for the Japanese Visual Genome QA dataset, we define an image feature $\mathbf{V}_{new}$ created from the attention maps and the image feature defined by eq. (1). Finally, we replace the image feature $\mathbf{V}$ in eqs. (2) and (3) with $\mathbf{V}_{new}$, and the method then proceeds as in Lu et al. (2016). We discuss the details below.

First, our method learns information from the English dataset. For each English question and image pair in Visual Genome, we generate one visual attention map $\mathbf{a}^v = \{\mathbf{a}_1^v, ..., \mathbf{a}_i^v, ..., \mathbf{a}_N^v\}$, where $\mathbf{a}_i^v$ contains the attention probabilities for image region $\mathbf{v}_n$. An image with $\mathbf{M}$ English questions thus has $\mathbf{M}$ attention maps $\mathbf{a}^v$, which we average to obtain a final attention map:

$$\mathbf{a} = \frac{1}{M} \sum_{m=1}^{M} \mathbf{a}_m^v, \ where \ \mathbf{a} \in R^N. \tag{5}$$

Then, the information learned from the English dataset (i.e., the image attention maps) is used for Japanese VQA prediction. The motivation for using these attention maps is that although the English and Japanese questions for the same image are usually different, the foci of attention most likely overlap. Letting $\mathbf{a} = (a_1, ..., a_N)$, we represent the image feature as $\mathbf{V}_{new} = \mathbf{Va} = (a_1\mathbf{v}_1, .., a_n\mathbf{v}_n)$. Finally, after replacing $\mathbf{V}$ with $\mathbf{V}_{new}$, our proposed method proceeds as in section 4.1 to learn the Japanese VQA model and predict Japanese answers. When the method is not considering information learned from the English dataset, it sets $\mathbf{a} = I$ (a vector with all elements set to 1).

5 Experiment

5.1 Setup

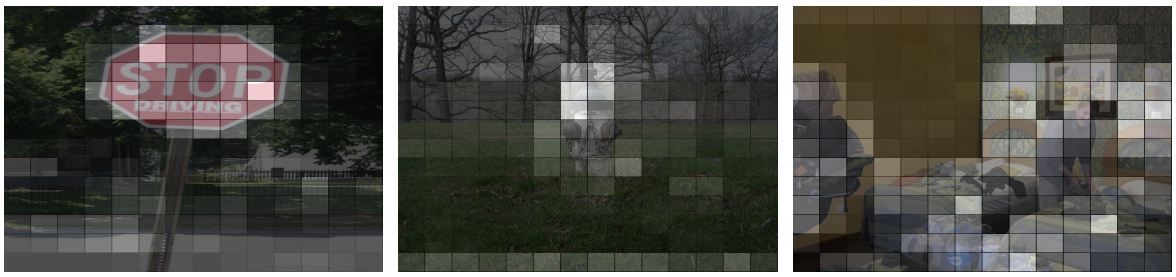
Our experimental baseline was the parallel co-attention model trained using the Japanese corpus. The proposed method uses the same model but with the attention maps initialized with the English corpus and then trained with the Japanese corpus. We divided our dataset into two parts for training and testing. The numbers of images in the training and test sets are respectively 91,609 and 7,599. The test set has 60,525 QA pairs. There are 135,740 unique answers in our dataset. We used the top 1,000 most frequent answers as the possible outputs, which covers 66.7% of the answers found in our dataset.

We used the RMSProp optimizer with a base learning rate of 4e-4, momentum 0.99, and weight decay 1e-8 and set the batch size to 20. We used VGG-19 (Simonyan and Zisserman, 2014), which is based on CNN (Fukushima, 1980; LeCun et al., 1989), to extract image features and MeCab (Kudo, 2005) to tokenize Japanese sentences. We performed 250,000 iterations.

5.2 Results and Analysis

For evaluation, we used the following definition of ***accuracy***:

$$\text{accuracy} = \frac{\text{No. of correctly classified questions}}{\text{No. of questions}} \tag{6}$$



(a) "mannaka ni utsutteiru hyoushiki ha naniiro desuka?" (b) "mannaka ni utsutteiru mono ha nan desuka?" (c) "ningen no mae ni donna kagu ga arimasu ka?"

Figure 3: Examples of attention maps for which the proposed method predicted the correct answer but the baseline method without cross-lingual attention maps did not. Questions: (a) What color is the road sign? (b) What is the object in the middle of the picture? (c) What is the furniture in front of the man?

	# images in training set		
Method	30,536	61,072	91,609
baseline	17.1%	17.7%	18.3%
proposed	**18.0%**	**18.8%**	**19.2%**

Table 2: Average accuracy with varying training set sizes. Each cell contains the average accuracy over four runs. All differences are statistically significant according to McNemar's test.

Figure 3 shows examples of attention maps that were generated from English questions and correctly predicted the answers for the Japanese dataset. For the same images, the baseline system without cross-lingual attention maps predicted wrong answers. We found that the attention maps usually focused on foreground objects and that accuracy tended to improve for images with clear foreground objects. Table 3 lists the accuracy by question type. For all question types except "why" questions, the cross-lingual attention maps improved the performance.

Because we trained and evaluated both our model and the baseline three times, Table 2 lists the average accuracies for each case. Our proposed model achieves 19.2% accuracy with 91,609 images in the training set. Note that our task is much tougher than in prior work (Zhu et al., 2016; Lu et al., 2016). In Zhu et al. (2016), the outputs are chosen from four multiple choices. While in Lu et al. (2016) the outputs are chosen from the same top 1,000 frequently occurring answers, the coverage of this set for their VQA corpus is 86.54%, unlike our 66.7%. The table also illustrates how the accuracy increased with the number of training images. Our method consistently performed around 1% better than the baseline method in all cases. As the performance increase was consistent across systems, we believe that using cross-lingual information should also improve performance in other situations. We can also see from Table 2 that the performance difference between the proposed method and the baseline did not decrease as the number of training images increased, which shows the value of our method for both larger and smaller datasets.

6 Conclusion

We have created a Japanese visual question answering (VQA) dataset comparable to the freeform question answering portion of the Visual Genome dataset (Krishna et al., 2016). This dataset is the first such dataset in Japanese. To show the utility of our corpus, we proposed a cross-lingual method for making use of English annotation to improve the Japanese VQA system. The proposed method experimentally

	Accuracy (%)								
Method	*nani* what	*dare* who	*doko* where	*donna* what kind	*dorekurai* how much	*dou* how	*itsu* when	*ikutsu* how many	*naze* why
baseline	19.9	26.1	14.4	20.3	15.5	24.4	53.1	18.9	**5.1**
proposed	**21.1**	**27.5**	**15.2**	**21.9**	**17.6**	**25.8**	**55.2**	**19.1**	**5.1**

Table 3: Accuracy for each of the nine Japanese question types.

performed better than simply using a monolingual corpus, which demonstrates the effectiveness of using attention maps to transfer cross-lingual information.

While VQA is mainly a testbed for monolingual image understanding, our data together with the original English Visual Genome allows modeling how a bilingual person understands images and two languages, which we call bilingual image understanding. We believe the release of our dataset will add significant resources to the research in this direction.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

Lisa Ballesteros and Bruce Croft. 1996. Dictionary methods for cross-lingual information retrieval. In *International Conference on Database and Expert Systems Applications*, pages 791–801. Springer.

Kunihiko Fukushima. 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, Apr.

Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei Xu. 2015. Are you talking to a machine? dataset and methods for multilingual image question. In *Advances in Neural Information Processing Systems*, pages 2296–2304.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *ACL*, volume 2008, pages 771–779.

Karl Moritz Hermann and Phil Blunsom. 2013. Multilingual distributed representations without word alignment. *arXiv preprint arXiv:1312.6173*.

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

Jagadeesh Jagarlamudi and A Kumaran. 2007. Cross-lingual information retrieval system for indian languages. In *Workshop of the Cross-Language Evaluation Forum for European Languages*, pages 80–87. Springer.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2016. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *arXiv preprint arXiv:1602.07332*.

Taku Kudo. 2005. Mecab: Yet another part-of-speech and morphological analyzer. *http://mecab. sourceforge. net/*.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551, December.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297.

Mateusz Malinowski and Mario Fritz. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in Neural Information Processing Systems*, pages 1682–1690.

Takashi Miyazaki and Nobuyuki Shimizu. 2016. Cross-lingual image caption generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1780–1790.

Mengye Ren, Ryan Kiros, and Richard Zemel. 2015. Exploring models and data for image question answering. In *Advances in Neural Information Processing Systems*, pages 2953–2961.

Kevin J Shih, Saurabh Singh, and Derek Hoiem. 2016. Where to look: Focus regions for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4613–4621.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. 2012. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2397–2406, New York, New York, USA, 20–22 Jun. PMLR.

Huijuan Xu and Kate Saenko. 2016. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 451–466, Cham. Springer International Publishing.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2016. Stacked attention networks for image question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 21–29.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June.

Style Detection for Free Verse Poetry from Text and Speech

Timo Baumann
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, USA
tbaumann@cs.cmu.edu

Hussein Hussein and **Burkhard Meyer-Sickendiek**
Department of Literary Studies
Free University of Berlin
Berlin, Germany
{hussein,bumesi}@zedat.fu-berlin.de

Abstract

Modern and post-modern *free verse poems* feature a large and complex variety in their poetic prosodies that falls along a continuum from a more fluent to a more disfluent and choppy style. As the poets of modernism overcame rhyme and meter, they oriented themselves in these two opposing directions, creating a *free verse spectrum* that calls for new analyses of prosodic forms. We present a method, grounded in philological analysis and current research on cognitive (dis)fluency, for automatically analyzing this spectrum. We define and relate six classes of poetic styles (ranging from *parlando* to *lettristic decomposition*) by their gradual differentiation. Based on this discussion, we present a model for automatic prosodic classification of spoken free verse poetry that uses deep hierarchical attention networks to integrate the source text and audio and predict the assigned class. We evaluate our model on a large corpus of German author-read post-modern poetry and find that classes can reliably be differentiated, reaching a weighted f-measure of 0.73, when combining textual and phonetic evidence. In our further analyses, we validate the model's decision-making process, the philologically hypothesized continuum of fluency and investigate the relative importance of various features.

1 Introduction

One of the most important explanations for modern art is the theory of aesthetic pleasure, which claims that the fluency of cognitive processing is the cause for the positive effect of aesthetic experience (Topolinski and Strack, 2009). Similarly, cognitive research on fluency showed that people rate stimuli that are processed more easily higher (Belke et al., 2010). On the other hand, many modern artists like Picasso or Schönberg complicated the processability of their works using processes of abstraction in order to prevent such automated or fluid forms of art comprehensibility. Regarding this development, Bullot and Reber introduced the term *disfluency* as an artistic strategy to bring more analytical forms of art experience to the fore (Bullot and Reber, 2013). Other researchers suggested that disfluency prompts people to process information more carefully, deeply, and on a higher level of abstraction (Smith and Smith, 2006).

We present a computational analysis that tests these two trends in the current discussion on art experience by focusing on the prosodic feature of (dis)fluency in modern and post-modern poetry. We assume that the rhythmical quality of modern poetry is "not properly measurable by the rules of traditional verse" (Wesling, 1996), since modern poetry often rejects the traditional metrical verse. Most modern and post-modern poetry uses rhythmical features beyond the metrical forms, preferring the imitation of naturalness of everyday language and its very fluent speech prosody. At the same time, modern poetry uses disfluent processes to introduce the kinds of obstructions to ease consumption as outlined above. Since Dadaism, many avantgarde poets developed certain kinds of disfluencies like line-breaks as well as segments and "micro-particles" known from sound poetry. In this paper, we offer a classification of modern and post-modern poetry along the continuum of fluency, going back to a similar idea developed in Eleanor Berrys theory of a 'Free Verse Spectrum' (Berry, 1997).

Proceedings of the 27th International Conference on Computational Linguistics, pages 1929–1940
Santa Fe, New Mexico, USA, August 20-26, 2018.

We develop a method to identify poetic features that relate to literary prosodic classes and with a special regard to the modelling of prosodic (dis)fluency, hence a form of style detection. Style detection has been a long-running topic in literary study: Tools for style analysis like *Metricalizer* (Bobenhausen, 2011) analyze metrical patterns given in a poem's text (see also (Agirrezabal et al., 2016) for more recent work on English metrical poems) and *Sparsar* (Delmonte and Prati, 2014) have used similar analyses to aid speech synthesis for metrical poems. Style modeling has also been used to automatically generate poems such as limericks (Manurung et al., 2000) or, more recently, traditional Chinese poetry (Zhang and Lapata, 2014; Wang et al., 2016). The above approaches feature poetic styles with very restrictive patterns far from the breadth of free verse. Focusing on contemporary American poetry, Kaplan and Blei (2007) analyze and visualize (textual) features of poems with respect to their poetic properties and Kao and Jurafsky (2015) estimate whether poems were written by professionals or amateurs. All the above works have used only the textual form of the poem. In contrast, music genre classification frequently relies on both audio and lyrics. Particularly close to our work is Tsaptsinos (2017) who uses a hierarchical attention network (Yang et al., 2016), which, however, is using just the lyrics. In our work, we extend the hierarchical network to comprise both speech and text in order to differentiate poetic styles that are primarily differentiated by their recitation.

In our own prior work, we have tried to differentiate classes of post-modern poetry using conventional feature extraction-based classification approaches. In (Hussein et al., 2018a), we havedifferentiated two enjambment-dominated styles using simple features from pausing and POS tagging and achieved an f-measure of 0.69; in (Hussein et al., 2018b), we differentiate *parlando* and *variable foot* styles with similar features, again reaching an f-measure of 0.69.

2 Classifying the 'Free Verse Spectrum'

At least 80 per cent of modern and post-modern poems have neither rhyme nor metrical schemes such as iambic or trochaic meter. Does this, however, mean that they lack any rhythmical features? According to the theory of free verse prosody, the opposite is true. Modern poets like the Imagists (Silkin, 1997; Cooper, 1998; Beyers, 2001), the Black Mountain poets (Golding, 1981; Steele, 1990; Silkin, 1997; Berry, 1997; Finch, 2000), as well as European poets before and after the second world war (Meyer-Sickendiek, 2012; Lüdtke et al., 2014) developed a post-metrical idea of prosody that employs rhythmical features of everyday language, prose, and musical styles including jazz and hip hop. In parts, they intended to create a fluent, in parts a disfluent prosody, for example in Dadaistic poetry. Donald Wesling (1971) offered a number of examples to illustrate the stylistic range of free verse poetry, focusing on the typical line arrangements in modern poems: (1) "Whitmanic", referring to Walt Whitman's adaptation of "the biblical verset and syntax" in "end-stopped lines . . . with boundaries so often equivalent to those of larger units of grammar," which Wesling sees as "constitut[ing] the precomposition or matrix of free verse in English"; (2) "line-sentences," as developed by Ezra Pound in *Cathay* on the basis of Ernest Fenollosa's theories of the sentence, in turn derived from the study of Chinese; (3) dismemberment of the line, whereby the line becomes "ground to the figures of its smaller units," and, as a sub-category, spatial dismemberment of the line by indentation, as William Carlos Williams does in his triadic line verse; (4) systematic enjambment (breaking a sentence or phrase into two lines), whereby the lines are "figures on the ground of the larger unit, the stanza"; (5) dismemberment with enjambment of the line, such that "the middle units on the rank scale engage in a protean series of identity shifts as between figure and ground" (Wesling, 1971). As can be seen, these five classes imply a continuous development from more to less fluent styles using an increase in dismemberments and enjambments.

Based on these examples, Eleanor Berry (1997) called for the investigation of poetry with regards to their features in 'the multidimensional space of free verse' on the basis of five 'axes' of form: (1) line-length, including extent of variability in length; (2) line-integrity, as determined by intralinear features as well as line-divisions; (3) line-grouping, whether stichic (ungrouped), in verse paragraphs, in stanzas, or dispersed on the page; (4) sensory basis of the verse form, whether aural, visual, or both; and (5) semantic function, that is, the relation of the verse form to the semantic aspect of the text, characterized in such terms as organic, iconic, and abstract (Berry, 1997). Berry used these five axes in order to classify

the spectrum of free verse in modern and post-modern poetry.

In this paper, we add prosody to this aforementioned free verse spectrum and establish a gradual one-dimensional continuum, whose two poles are denoted by the terms fluent and disfluent. We illustrate this prosodic spectrum by ranking six different poetic styles respectively prosodic patterns within the free verse spectrum, starting with the most fluent one: (a) The **parlando** pattern was coined by the German poet Gottfried Benn, who created a colon (word group)-based line grouping as in (3) above, but ignored the gap to the run-on-line – i.e. the part after the enjambment – by a fluent reading not emphasizing the enjambment. This fluency was typical for his conversational idiom. The second poetic pattern, (b) the **variable foot** is is identical to the "triadic line verse" (3 above) invented by W. C. Williams (Cushman, 1985). Like the parlando, the variable foot uses a "soft enjambment", but the poet now emphasizes the gap to the run-on-line. As long as this run-on-line also occurs between each singular colon of the poem, i.e. the noun and verbal phrases, this gap does not really affect the flow of the stanza and the poem still sounds quite natural.

In the third pattern, the (c) **unemphasized enjambment**, the poet now creates a more disfluent, choppy style by using the so-called "hard enjambments" that interrupts the reading flow of the poem. This occurs when the enjambment runs across stanzas; separates articles or adjectives from their nouns or splits a word across a line. Finally, the (d) **gestic rhythm** even emphasizes these hard enjambments, which makes the poem sound way more disfluent than in the two previous patterns. Bertolt Brecht coined this technique by calling it a "gestic rhythm", preventing the ear from gliding past the message. Gestic rhythm is any rhythm that causes some difficulty in listening to it.

Even more radical kinds of poetic disfluency have been developed in modern "sound poetry" by dadaistic poets like Hugo Ball and Kurt Schwitters or concrete poets like Ernst Jandl and Oskar Pastior. Within the genre of sound poetry, there are two main patterns: the (e) **syllabic decomposition** and the (f) **lettristic decomposition**, the last and most disfluent pattern. A typical example for syllabic decomposition in sound poetry is the *Ursonate* [The Sonata in Primal Speech] by Dadaist Kurt Schwitters, which begins with "Fümms bö wö tää zää Uu." A typical example for the most disfluent lettristic decomposition can be found in Ernst Jandl's *schtzngrmm*, which is presented in Figure 2 (b) below.

Given this spectrum of free verse poetry, we can add a simple hierarchy of linguistic units to clarify the range from fluency to disfluency. In a grammatical hierarchy, letters are the smallest units and they combine to form morphemes, which combine to form words, to groups, to clauses and finally sentences. Dominating units differ between the different styles along the fluency continuum, from lettristic decompositions to parlandos which are dominated by the largest units. In addition, the recitation style itself can be less or more fluent, and the formation and emphasis of the enjambments at the end of each line is indicative of the fluency of the style.

3 Modeling the Prosody of Spoken Free Verse Poetry

In this section, we describe our model, which is inspired by Yang et al. (2016), as well as our high-level decisions for modeling. Poetry, in particular post-modern poetry, is challenging material for computational modeling and statistical natural language processing. The very purpose of art (and post-modern poetry in particular) is to stand out and to defy or re-define rules, making generalization difficult. Poems, as compared to normal language use, contain unusual words (or no words at all in the case of decompositions, see above), and there is generally only very little data available as compared to most other domains. The automatic alignment of text and audio in spoken poetry is non-trivial (in particular for decompositions in more abstract poetry) and important clues may be contained not only in how the textual material is spoken, but also in the gaps between textual material, such as extra white-space or the pausing between the lines of a poem.

Given the broad variety of the poems in combination with their relatively small number (see below), our model must deal well with *data sparsity*, i.e. use as few free parameters as possible that need to be optimized during training. For this reason, we decide to focus our textual processing on character-by-character encoding of the lines in the poem (and using character embeddings). We use a bidirectional recurrent neural network (RNN, using gated recurrent unit (GRU) cells (Cho et al., 2014)) which encodes

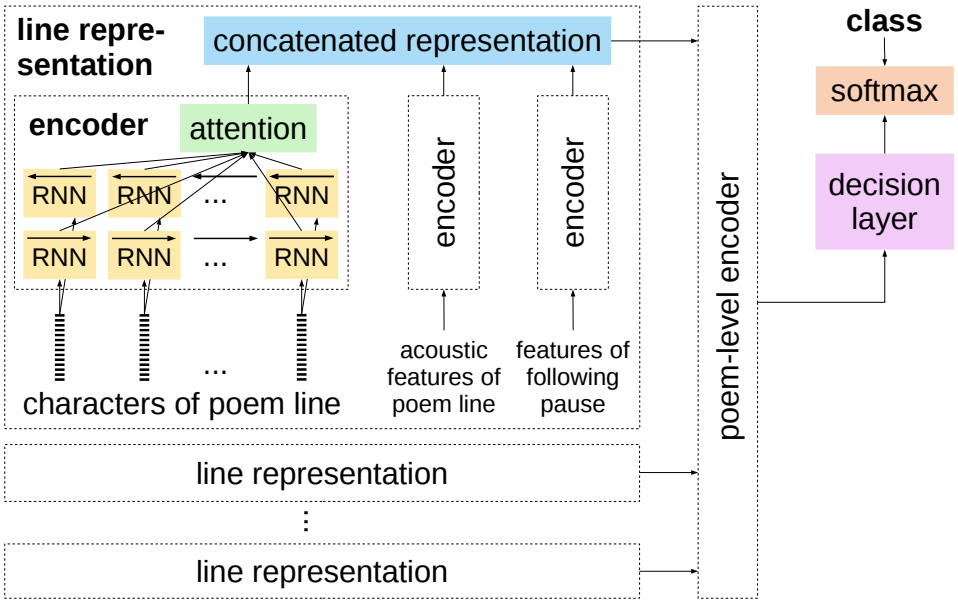


Figure 1: Full model for poetry style detection: each line is encoded character-by-character by a recurrent neural network (using GRU cells) with attention. Acoustic features of each line, as well as of the pause following up to the next line, are encoded similarly. Per-line representations are concatenated and passed to a poem-level encoder. The final decision layer optimizes for the poem's class.

the sequence of characters into a multi-dimensional representation that, although it is not directly applicable to human interpretation, is trained to be optimal towards differentiating the prosodic classes.

Our model is not trained using an explicit notion of words. Instead, it may implicitly encode word-level information (such as parts of speech) via the constituting sequences of characters. This is in line with recent work on end-to-end learning, e.g. for speech recognition (Hannun et al., 2014; Graves and Jaitly, 2014), which no more explicitly models phonemes nor words, but directly transfers audio features to character streams. While processing on the word level might allow our model to build a better higher-level understanding of the poem's meaning, this semantic information would likely not help in style differentiation. In addition, word representations would not capture the usage of whitespace, e.g. for indentation, to create justified paragraphs, or other uses, nor special characters.

We encode the speech (after feature extraction) in a similar way in order to capture the notion of fluency of speech delivery in the author's recitation. As for the text, we use speech line-by-line so that the model may synchronize what it 'hears' and what it 'reads'. A more fine-grained alignment of text and speech would be much harder to produce and might plausibly fail for sound poetry (although it would certainly be desirable). Finally, in order to differentiate the reading of enjambments, we also encode the pauses in-between lines, which may also contain important information about breathing, pausing, in-breath, etc.

For a model to be a suitable and acceptable tool for (digital) humanistic research, it should provide insight into its decision making process, as our primary goal is not so much the automatic classification of poetry but to learn about and better understand poetic styles. To satisfy this requirement of inspectability of the decision making process (at least to some extent), we implement a notion of *inner attention* (Liu et al., 2016) that is to learn how to combine the sequential states of each line's encodings (text, audio, and final pause) to a representation that is best suited towards our training objective. Attention (a) may improve the model's representations and hence yield better performance (although some initial testing did not show a large impact), and (b) can be observed during the application of the model and gives an indication of what the model pays attention to, and can be discussed wrt. its philological plausibility.

We combine the line-by-line representations using a poem-level encoder which is fed to a decision layer and a final softmax to determine the poem's class, yielding the hierarchical attention network as shown in Figure 1. While our network is similar to those of Yang et al. (2016) and Tsaptsinos (2017), we base ours on characters instead of words as textual input and include the audio stream into the analysis via

Table 1: Descriptive characteristics of the data used in the experiments.

		poems	lines	characters	audio
lyrikline: German subcorpus		2392	61849	2025484	52 h
	parlando	34	1435	44323	67 min
	variable foot	34	878	23684	39 min
	unemphasized enjambment	36	1090	33178	48 min
	gestic rhythm	33	897	27741	44 min
	syllabic decomposition	21	540	12390	26 min
	lettristic decomposition	17	684	10007	31 min
deutschestextarchiv.de		—	34291	996714	—

additional encoders.

Our model is implemented in *dyNet* (Neubig et al., 2017) and the software to replicate our experiments is available at `https://bitbucket.org/timobaumann/deeplyrik` in order to foster research on free verse poetry. Our implementation is flexible towards the number of classes to distinguish and can be configured to ignore some of the features (cmp. Table 3). We make use of this flexibility and perform a range of classification experiments in Section 5, once we have described our data in more detail.

4 Data and Setup

We collected German poems available on the webpage *lyrikline*[1] and classified 175 of a total of ~2400 German poems into the six prosodic classes defined above, with a special focus on poets known for the use of such prosodic patterns. Manual classification of the data was carried out by the third author, a philologist and literary scholar and following criteria from the literature as outlined above. We also collected the corresponding audio recording of each poem as spoken by the original author, yielding 52 h of audio. Checking manually, we found some poems tagged as German that actually were not (<1 %) and discarded these from further processing. For pre-training (see below), we also downloaded from the German Text Archive (Geyken et al., 2011)[2] additional text-only poetry published between 1800 and 1930 (and hence most likely not post-modern; more recent poems are hard to find as they are typically still copyrighted).

Some key descriptive statistics of the poems as assigned to their classes are reported in Table 1. In order to check whether poetic classes can already be singled out based on their length (in lines, characters, or audio duration) alone, we checked for significant deviations from the overall corpus. For none of the classes, the poems' durations, or number of lines significantly differ from the average poem in the corpus (two-tailed t-tests, $p > .05$ for all tests). However, variable foot poems, as well as syllabic and lettristic decompositions have significantly fewer characters than average poems.

4.1 Preprocessing

We perform forced-alignment of text and speech for the poems in our six classes using the text-speech aligner published by Baumann et al. (2018) which uses a variation of the SailAlign algorithm (Katsamanis et al., 2011) implemented via Sphinx-4 (Walker et al., 2004). The alignments are stored in a format that guarantees the original text to remain unchanged which is important to be able to recreate the exact white-spacing in the poem and would be helpful when adding further annotations (e.g. parts of speech, syntax or semantics) to the poem in the future.

We extract the line-by-line timing (start of first and end of last word of the line) for each line. Forced alignment of poetry is far from trivial and often individual words cannot be aligned. Lettristic and syllabic decompositions, being a form of sound poetry, are notoriously hard to align automatically and we resorted to manual alignment of those lines that could not be aligned automatically.

[1] `http://www.lyrikline.org`
[2] `http://www.deutschestextarchiv.de`

We extract Mel-frequency cepstral coefficients (MFCC) for every 10 milliseconds of the audio signal as well as fundamental frequency variation (Laskowski et al., 2008, FFV) vectors, which are a continuous representation of the speaker's pitch. We z-normalize all feature dimensions. In order to not overwhelm the model with acoustic sequence information, and given that relevant speech phenomena are typically much longer than 10 milliseconds, we compute the mean and standard deviation of 10 consecutive frames for every feature.

4.2 Pre-training

The manually classified corpus is small and hence the quality of intermediate representations is limited by data sparsity. As an example: a strongly distinctive characteristic of syllabic and lettristic decomposition is the presence of repetitive consonant-vowel sequences in the poem (which occur frequently in syllabic decompositions). Yet, in the two-class problem of distinguishing the sub-types of decompositions, it is hard to infer the differentiation of characters into consonants and vowels from only 38 example poems (which feature a wealth of other characteristics). It would be similarly unreasonable to ask a student of literary studies to learn to differentiate poetic styles based on just a few examples in a language and writing script unknown to them. We mitigate this problem by using pre-training methods (Erhan et al., 2010) to help bootstrap the generation of reasonable intermediate representations (i.e., we teach the model some notion of poetic language as a foundation before teaching it to differentiate styles).

We pre-train the character embeddings and the line encoder using a recurrent autoencoder that aims to build a representation of the line that best allows it to re-create the original line (using combined costs of both forward and backward decoding as training objective); in other words: we ask our model to memorize poetic lines but given its limited memory it has to learn an abstraction of each line that helps it to remember the line. We train this on the whole poetry corpus including poems collected from deutschestextarchiv.de.

We pre-train our representations for the acoustic features of each line similarly to the textual pre-training in that we train a recurrent autoencoder that aims to re-create the original line-by-line features, as well as the length of the acoustic stimulus. Re-creating the length of the original stimulus is particularly important as this feature is directly relevant for measuring the pause between two lines and is otherwise only a very indirect objective in pre-training. Given that line-by-line alignments are only available for the 175 poems that were manually classified, we pre-train the acoustic representations on inter-pausal units for the line representations (pausal units for between-line representations) detected using voice activity detection.

It would also be desirable to pre-train the model's poem-level encoding (i.e. not just teach it about lines in a poem by having it memorize lines, but also teach how lines are combined into a poem). Unfortunately, line-by-line text-audio alignments are not available for the full corpus and hence we are limited to either pre-train based on textual information only (reported as 'text-only' in Section 5) or to use audio information but not use pre-trained poem-level information.

4.2.1 Training Procedure

Even when using pre-trained internal representations, only 175 training instances are too few for training the deep model towards the classification objective. However, poems typically display their structural properties on the vast majority of the lines they are composed of. We hence split training into two steps by first training a decision network that learns to classify individual lines of the poem in order to adapt the pre-trained network. While we here ignore the run-on-line in the case of enjambements, we do include the pausing information to model enjambments at least partially. Coming back to Figure 1, we first leave out the poem-level encoding and directly pass each line representation to a line-by-line decision layer.

Afterwards, we replace the line-by-line decision layer with the poem-level encoder and final decision layer and train towards the per-poem decisions based on the parameters estimated before. Thus, the final model is able to steer its attention mechanism towards the important lines and can learn to sacrifice the initially trained per-line optimization for the overall per-poem optimization.

For all classification experiments reported below, we perform 15 training epochs and use a dropout probability of 0.3 (Srivastava et al., 2014) to reduce overfitting. Each encoder is two layers deep and

Table 2: Per-class f-measures as well as the confusion matrix for the six-class classifier.

	f-measure	parlando	var. foot	unemph. enj.	gestic	syll. dec.	lettr. dec.
parlando	0.83	30	2	2			
variable foot	0.60	3	20	6	5		
unemph. enj.	0.71	2	4	27	3		
gestic rhythm	0.68		6	5	21		1
syllabic dec.	0.81	2	1			17	1
lettristic dec.	0.77	1				4	12

Table 3: Results (weighted f-measure) for reduced feature sets; $^{\text{NS}}$ indicates that the classifier does not perform significantly above chance level.

	all features	no pause	text-only
all six classes	0.73	0.66	0.47
parlando vs. variable foot	0.85	0.85	0.65
unemphasized enjambment vs. gestic rhythm	0.78	0.66	0.57^{NS}
syllabic vs. lettristic dec.	0.82	0.92	0.82

has a 20-dimensional state. Our character embeddings are 20-dimensional as well as are the attention representations.

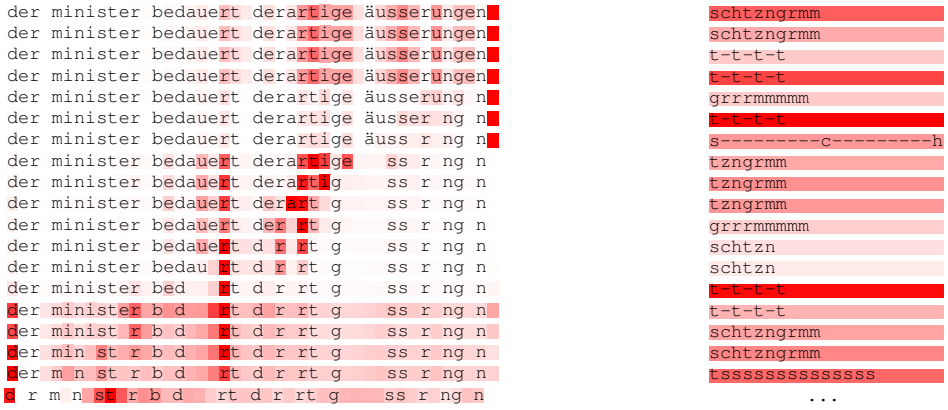
5 Classification Experiments

We train a classifier to distinguish the six classes of poetic style with all features (text, speech, and pause) using pre-processing and pre-training as described in the previous section; given the little available data, we use 25-fold cross-validation (5 poems per test fold). We report the per-class f-measures and confusion matrix in Table 2. The average f-measure (weighted by class size) is 0.73, indicating that it is indeed possible to distinguish the postulated prosodic classes based on text, speech, and pauses and using a deep neural model.

Analyzing the confusion matrix (in which we ordered the classes by their postulated fluency), we furthermore find that most misclassifications are clustered near the central diagonal. We take this as an indicator that classes close to each other on the fluency continuum are more easily misclassified with each other. In contrast, none of the very fluent classes is taken for a decomposition and only rarely are decompositions misclassified as member of the fluent classes. The picture is less clear for the three classes of variable foot, unemphasized enjambment and gestic rhythm. Given that all of these styles feature enjambments of some sort, they might be somewhat harder to differentiate, or the limited performance might point to the importance of higher-level information in the manual classification which our model is unable to pick up.

In Table 3 we show the weighted f-measure (as the most descriptive single performance metric) for the full six-class classifiers, as well as for binary classifiers that we train to differentiate (a) the two most fluent styles which are still regarded as relatively fluent to read and listen to; (b) the two styles that build on enjambments and mostly differ by whether these enjambments are unemphasized or emphasized as in gestic rhythm, and (c) the two types of decomposition which differ quite strongly in their textual appearance. We train each of these classifiers for the full feature set, we leave out pause information and only use audio during speech, and finally use on the textual information.

The results in Table 3 indicate that indeed, (a) parlando and variable foot can hardly be differentiated based on textual features alone but only with access to the recitation style (yet do not require pausing information). (b) Pause information is crucial to differentiate the enjambment-dominated styles which differ mostly between the lines. In fact, these styles cannot be differentiated at all based on textual information alone, in which case the classifier does not significantly outperform the chance level. Finally,



(a) Jaap Blonk: Der Minister I

(b) Ernst Jandl: schtzngrmm

Figure 2: Visualization of attention in two lettristic poems: (a) attention to characters within the line, (b) attention to lines (including the audio) in the poem.

the (c) decomposition styles can already be differentiated based on textual information. Access to the recitation style improves classification performance but adding the pausal features shrinks this advantage, presumably because the pausing information is irrelevant to distinguish the classes and confuses the classifier during training. We also find that the neural model substantially outperforms our previous feature-based approaches (Hussein et al., 2018a; Hussein et al., 2018b), at least when using the full feature sets.

6 Model Analysis

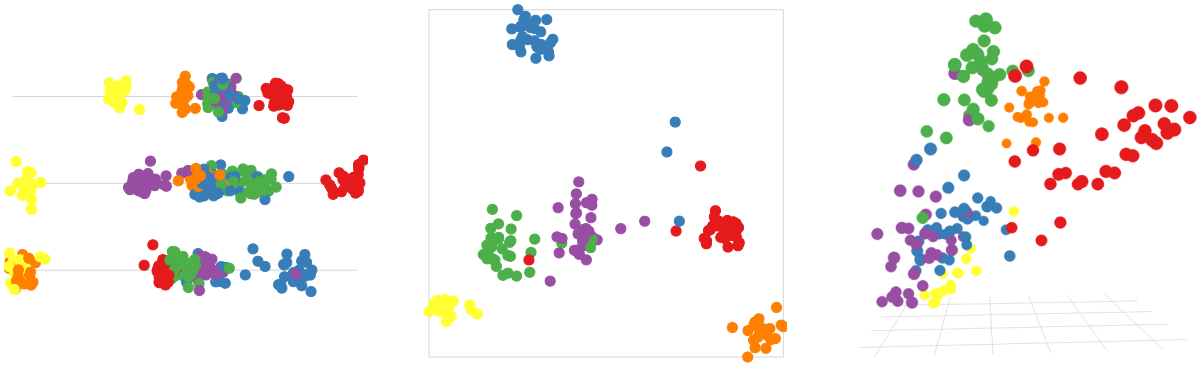
In order to gain additional insight into the classification performance and to investigate whether the model is observing the philologically 'correct' features of poems, we perform further analyses on the attention sub-model, as well as by analyzing the poem-level representations in a lower dimensional space.

Attention models can provide insight into the decision making of the model. In particular, they allow to analyze the weighing of parts of the input in the decision making process. This enables us to test whether the model has actually learned to classify similarly to how a human might.

We plot attention results for two exemplary poems in Figure 2: As can be seen, the poem in Figure 2 (a) repeats the same line over and over which gradually decomposes from the right. This leads to a diagonal frontier in the poem and the model attends to this frontier, at least to some extent. Another poem by the same author (which decomposes from the left) shows a similar pattern. The poem in Figure 2 (b) is about trench[3] warfare during World War II and onomatopoeically mimics the noises of war (leaving out all vowels in doing so). The model singles out those lines in the poem that deviate most from fluent language – both textually and acoustically. The decision layer trained in line-by-line training can be used for further analysis. For example, in the poem in Figure 2 (a), the line-by-line classification is wrong for some of the first few lines, mirroring the fact that decomposition only gradually sets in.

We have postulated in Section 2 that the six classes of poetic style can be ordered along a (dis)fluency continuum, with *parlando* being the most fluent (and similar to normal spoken language) and *lettristic decomposition* the least. While we have provided external evidence that the classifier does differentiate the fluency classes with the confusion matrix presented in Section 5, we wonder if the internal poem representation also reflects the ordering in this continuum. In order to visualize the internal representations of the poetic model, we train another classifier which injects a low-dimensional *bottleneck layer* immediately before the final decision layer, thus forcing the model to represent every poem as a point in low-dimensional space (we use 1-3 dimensions). We then plot each poem's representation as a point and color-code classes by color. Resulting plots are shown in Figure 3: on the left, we show the results of three 1D mappings resulting from different random initializations of the model. It can be seen that the first model represents the classes in the order of fluency but fails to differentiate the enjambment-based

[3]German: 'Schützengraben' which is reduced to 'schtzngrmm' by Jandl.



(a) three trials of linear mapping (b) 2D-mapping (c) Projection of 3D-mapping[4]

Figure 3: Visualization of low-dimensional mapping of the poems contained in the six classes (parlando: red, variable foot: blue, unemphasized enjambment: green, gestic rhythm: purple, syllabic decomposition: orange, lettristic decomposition: yellow).

classes. The second mapping better differentiates the classes but more strongly deviates from the fluency continuum, which the third mapping ignores alltogether.

The center and right of the figure show 2D and 3D mappings[4]. As can be seen, the classes are mapped into separate areas of the representation space and cluster nicely (except for a few outliers). However, their ordering only partially corresponds to the fluency continuum. For the 3D mapping[4] we find differentiations in the representations that could be interpreted as aspects of fluency modeling: (a) the naturalness of the text (with both types of decomposition and their generally non-word content set apart from the other poems in one dimension) and (b) auditory vs. textual fluency, with enjambment and syllabic decomposition being auditorily similarly fluent to *parlando* but textually not.

7 Conclusion

With our analysis, we have captured the spectrum of free verse prosody in modern poetry, using computational techniques, along the fluency/disfluency continuum which also plays a central role in the discussion on the cognitive processing of aesthetic artifacts; to the best of our knowledge, we are the first to do so. We have trained classifiers that integrate for each line the textual information, the spoken recitation, as well as the pausing information, and integrate information across the lines within the poem. We deem the overall classification performance as high (although classification is not our primary goal). In addition, we have trained classifiers for sub-problems and using reduced feature sets and the results obtained in these experiments support the expectations based on philological understanding.

Our classifiers provide some insight into the decision making process via the attention mechanism and the possibility to map the internal state into lower-dimensional spaces for visualization. We find that our model indeed seems to internally re-create some notion of fluency. However, we also find that the mappings to lower dimensions do not fully support the claim of one single dimension of fluency. In particular in 3D-mapping, a more complex picture evolves. This may be due to shortcomings in the model, the underlying data and annotated classes. It may also question our initial hypothesis calling for a further refinement of the fluency theory.

In our future work, we intend to analyze the whole *lyrikline* corpus (and beyond) in order to gain insights about this broad sample of post-modern poetry. We hope to semi-automatically find additional poems that belong into one of our classes (and could be added as further training material after manual validation). We also intend to analyze the corpus for clusters of outliers from our current classification in order to determine further and refine the existing classes using an iterative "human in the loop" appoach (Baumann and Meyer-Sickendiek, 2016). Our goal is the mutual benefit of the model (which requires human input) and the philological expert who will be able to quickly scan, analyze and browse vastly larger collections of poetry than has been possible in the past.

[4]An interactive version of the 3-D plot is available at `https://timobaumann.bitbucket.io/colingfreeversepoetry/`.

Our model so far is still far from optimal (beyond the fact that we have not searched meta-parameters that yield best classification results) and we want to point out some aspects of future work: enjambments are linked to syntactic characteristics and we could integrate syntactic information (such as part-of-speech (POS) tags) or try to pre-train our character encoding to decode the POS tag sequence. It would be helpful if we could inject philological insight about what to pay attention to into the training process (and extract it out of the application process) in order to increase the philological validity of our model. With regards to the neural network, we could link the text and speech streams using connectionist temporal classification (Graves et al., 2006) for it to relate auditory to textual information in more detail, and we could connect the (sequential) line and pause encodings of audio in order for the model to better normalize out speaker specific (but not style-specific) characteristics.

Acknowledgements

This work is funded by Volkswagen Foundation in the programme 'Mixed Methods in the Humanities.' We thank Vasu Sharma for valuable suggestions on neural network processing, and Arne Köhn for valuable comments on a draft of the paper, as well as the anonymous reviewers for their insightful comments.

References

Manex Agirrezabal, Iñaki Alegria, and Mans Hulden. 2016. Machine learning for metrical analysis of english poetry. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 772–781. The COLING 2016 Organizing Committee.

Timo Baumann and Burkhard Meyer-Sickendiek. 2016. Large-scale analysis of spoken free-verse poetry. In *Proceedings of Language Technology Resources and Tools for Digital Humanities (LT4DH)*, Osaka, Japan, December.

Timo Baumann, Arne Köhn, and Felix Hennig. 2018. The Spoken Wikipedia Corpus Collection: Harvesting, Alignment and an Application to Hyperlistening. *Language Resources and Evaluation*.

Benno Belke, Helmut Leder, Tilo Strohbach, and Claus Christian Carbon. 2010. Cognitive fluency: High-level processing dynamics in art appreciation. *Psychology of Aesthetics, Creativity, and the Arts*, 4(4):214–222.

E. Berry. 1997. The Free Verse Spectrum. *College English*, 59(8):873–897.

C. Beyers. 2001. *A History of Free Verse*. University of Arkansas Press.

Klemens Bobenhausen. 2011. The Metricalizer – automated metrical markup of German poetry. *Current Trends in Metrical Analysis, Bern: Peter Lang*, pages 119–131.

NJ Bullot and R. Reber. 2013. The Artful Mind Meets Art History: Toward a Psycho-Historical Framework for the Science of Art Appreciation. *Behavioral and Brain Sciences*, 36(2):123–137.

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

G. B. Cooper. 1998. *Mysterious Music: Rhythm and Free Verse*. Stanford University Press.

S. Cushman. 1985. *William Carlos Williams and the Meaning of Measure*. Yale Studies in English. New Haven and London.

Rodolfo Delmonte and Anton Maria Prati. 2014. Sparsar: An expressive poetry reader. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 73–76, Gothenburg, Sweden, April. ACL.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660.

A. Finch. 2000. *The Ghost of Meter: Culture and Prosody in American Free Verse*. University of Michigan Press.

Alexander Geyken, Susanne Haaf, Bryan Jurish, Matthias Schulz, Jakob Steinmann, Christian Thomas, and Frank Wiegand. 2011. Das deutsche textarchiv: Vom historischen korpus zum aktiven archiv. *Digitale Wissenschaft*, page 157.

A. Golding. 1981. Charles Olson's Metrical Thicket: Toward a Theory of Free-Verse Prosody. *Language and Style*, 14:64–78.

Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning*, pages 1764–1772.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

Hussein Hussein, Burkhard Meyer-Sickendiek, and Timo Baumann. 2018a. Automatic detection of enjambment in german readout poetry. In *Proceedings of Speech Prosody*, Poznán, Poland, June.

Hussein Hussein, Burkhard Meyer-Sickendiek, and Timo Baumann. 2018b. Tonality in language: The "generative theory of tonal music" as a framework for prosodic analysis of poetry. In *6th International Symposium on Tonal Aspects of Language (TAL)*, Berlin, Germany, June.

Justine T Kao and Dan Jurafsky. 2015. A computational analysis of poetic style. *LiLT (Linguistic Issues in Language Technology)*, 12.

David M Kaplan and David M Blei. 2007. A computational approach to style in american poetry. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 553–558. IEEE.

Athanasios Katsamanis, Matthew Black, Panayiotis G Georgiou, Louis Goldstein, and S Narayanan. 2011. SailAlign: Robust long speech-text alignment. In *Proc. of Workshop on New Tools and Methods for Very-Large Scale Phonetics Research*.

Kornel Laskowski, Mattias Heldner, and Jens Edlund. 2008. The fundamental frequency variation spectrum. In *Proceedings of FONETIK 2008*.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional LSTM model and inner-attention. *CoRR*, abs/1605.09090.

J. Lüdtke, B. Meyer-Sickendieck, and A. M. Jacobs. 2014. Immersing in the stillness of an early morning: Test ing the mood empathy hypothesis of poetry reception. *Psychology of Aesthetics, Creativity, and the Arts*, 8(3):363–377.

Hisar Manurung, Graeme Ritchie, and Henry Thompson. 2000. Towards a computational model of poetry generation. In *Proceedings of the AISB'00 Symposium on Creative and Cultural Aspects and Applications of AI and Cognitive Science*.

B. Meyer-Sickendiek. 2012. *Lyrisches Gespür: Vom geheimen Sensorium moderner Poesie*. Fink, Wilhelm.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

J. Silkin. 1997. *The Life of Metrical and Free Verse in Twentieth-Century Poetry*. Palgrave Macmillan UK.

L. F. Smith and J. K. Smith. 2006. The nature and growth of aesthetic fluency. In *New directions in aesthetics, creativity, and the psychology of art, ed. P. Locher, C. Martindale, L. Dorfman, V. Petrov and D. Leontiev*, page 47–58.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

T. Steele. 1990. *Missing Measures: Modern Poetry and the Revolt Against Meter*. University of Arkansas Press.

S. Topolinski and F Strack. 2009. The architecture of intuition: Fluency and affect determine intuitive judgments of semantic and visual coherence, and of grammaticality in artificial grammar learning. *Journal of Experimental Psychology*, 1(138):39–63.

Alexandros Tsaptsinos. 2017. Lyrics-based music genre classification using a hierarchical attention network. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, pages 694–701.

W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf, and J. Woelfel. 2004. Sphinx-4: A Flexible Open Source Framework for Speech Recognition. Technical report, Mountain View, CA, USA, November.

Zhe Wang, Wei He, Hua Wu, Haiyang Wu, Wei Li, Haifeng Wang, and Enhong Chen. 2016. Chinese poetry generation with planning based neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1051–1060. The COLING 2016 Organizing Committee.

D. Wesling. 1971. The Prosodies of Free Verse. In Reuben A. Brower, editor, *Twentieth-Century Literature in Retrospect*, pages 155–187. Cambridge, MA: Harvard University Press.

D. Wesling. 1996. *The Scissors of Meter: Grammetrics and Reading*. University of Michigan Press.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680, Doha, Qatar, October. Association for Computational Linguistics.

A Neural Question Answering Model Based on Semi-Structured Tables

Hao Wang[1], **Xiaodong Zhang**[1], **Shuming Ma**[1], **Xu Sun**[1], **Houfeng Wang**[1,2], **Mengxiang Wang**[3]
[1] MOE Key Lab of Computational Linguistics, Peking University, Beijing, 100871, China
[2] Collaborative Innovation Center for Language Ability, Xuzhou, Jiangsu, 221009, China
[3] Teachers' College of Beijing Union University, Beijing, 100011, China
{hhhwang, zxdcs, shumingma, xusun, wanghf}@pku.edu.cn
41326498@qq.com

Abstract

Most question answering (QA) systems are based on raw text and structured knowledge graph. However, raw text corpora are hard for QA system to understand, and structured knowledge graph needs intensive manual work, while it is relatively easy to obtain semi-structured tables from many sources directly, or build them automatically. In this paper, we build an end-to-end system to answer multiple choice questions with semi-structured tables as its knowledge. Our system answers queries by two steps. First, it finds the most similar tables. Then the system measures the relevance between each question and candidate table cells, and choose the most related cell as the source of answer. The system is evaluated with TabMCQ dataset, and gets a huge improvement compared to the state of the art.

1 Introduction

Question answering is a practical task, and there are many related challenges. Due to the complexity of natural language, those challenges are hard to solve. A QA model requires knowledge, which contains facts, and fetches the answer from the knowledge. For most QA systems, raw text and structured knowledge graph are used as their knowledge. However, raw text corpora are hard to understand for machine. Besides, most knowledge graphs contain too much noise and still need manual intervention. In contrast, the semi-structured data are more flexible to be comprehended than raw text corpora, and much easier to build from text automatically than knowledge graphs. Besides, there are many documents which can be easily converted to semi-structured tables, such as announcements published by institutions and knowledge in science books, etc. We would like to utilize these kinds of knowledge in QA systems.

There are many works on QA based on other knowledge. Some of them are based on raw text corpora, such as Miller et al. (2016), Min et al. (2017), Yin et al. (2016). Besides, QA models based on knowledge graphs are improving rapidly. Freebase (Bollacker et al., 2008) and Wikidata (Vrandecic and Krötzsch, 2014) are well-known structured knowledge bases, which are built almost manually by their users. Although there are some studies on automatic knowledge graph construction, like Sateli and Witte (2015), it is not good enough. There are also many studies on those structured data, like Yih et al. (2015), Yao and Durme (2014). Some of them focused on text description of each item, while others, such as Zhang et al. (2016), try to obtain embeddings of the items.

Although there is limited work on semi-structured tables, there has already been research of QA based on simple tables. HILDB (Dua et al., 2013) is a QA system which converts questions in natural language to SQL queries. Vakulenko and Savenkov (2017) offer another data structure of tables, and introduce a QA system based on tabular knowledge. However, those methods are limited to the specific structure of tables, and cannot take advantage of semi-structured data.

In previous work, candidate table selection is based on manually selected features, such as TF-IDF, which makes it hard to transfer to other environment. Besides, all columns in each row are considered equally, but for some questions, some of columns in the table are completely unrelated to question, and those cells can mislead the model to give a wrong answer.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1941–1951
Santa Fe, New Mexico, USA, August 20-26, 2018.

Resource Type Organism

RESOURCE for organisms		RESOURCE TYPE	RESOURCE SOURCE		DIRECT OR INDIRECT?		ORGANISM CLASS	
sunlight	is	light energy	from the Sun	that is	directly	required by	plants	to survive
sunlight	is	light energy	from the Sun	that is	indirectly	required by	all living things	to survive
water	is	matter	available on Earth	that is	directly	required by	all living things	to survive
air	is	matter	available on Earth	that is	indirectly	required by	all living things	to survive
air	is	matter	available on Earth	that is	directly	required by	animals	to survive
air	is	matter	available on Earth	that is	directly	required by	plants	to survive
food	is	matter	created using water, air, and sunlight	that is	directly	required by	animals	to survive
food	is	chemical energy	created using water, air, and sunlight	that is	directly	required by	animals	to survive
habitat	is	an environment	on Earth	that is	directly	required by	animals	to survive

What is an example of light energy indirectly required by all living things?
A. Air B. habitat C. food D. sunlight

Figure 1: The structure of semi-structured table knowledge and one of its corresponding MCQs.

In this paper, we will propose a neural system for answering multiple choice questions (MCQs) based on semi-structured tabular data. Our model contains two parts: 1) *candidate table selection*: for each query, tables are ranked according to relevance to the question, and the model fetches candidate cells from the most relevant tables, 2) *cell scoring and answer selection*: the model measures the relevance of query and each candidate cell, and finally the model obtains an answer based on the most relevant cells to the query.

Our first contribution is offering an end-to-end model to select candidate tables before scoring and ranking the answers, instead of using hand-crafted features in previous work. According to our evaluation, recall of the model on the top three tables is more than 98.9%, which leads the system to be efficient and accurate.

Another contribution of our work is a new relevance scorer between table cells and text. We use an attention layer to make the model focus on useful information of the table. With effort of the new scorer, our system has better performance than state-of-the-art on the task. Besides, the model has fewer parameters, which makes it easier to be trained.

Base on our evaluation, our system gets a high performance on the answer scoring task of TabMCQ. Our system achieves an accuracy of 79.0% on test set, while the state-of-the-art system TabNN (Jauhar, 2017) only answers 56.8% of questions correctly.

2 Task Description

The knowledge contains some semi-structured tables $\mathbb{T} = \{T_i | i \in 1...n\}$. And each case contains a MCQ query. There is an example of knowledge and queries in Figure 1. There are several semi-structured tables in its knowledge base.

Each semi-structured table has a title, like *Resource Type Organism* in Figure 1. Different from structured tables, semi-structured tables have some columns used to link other cells, which makes each row is a complete sentence, and those columns are not tagged. Except for those columns, each column in the tables has a tag.

Each query contains a question and three or four candidate choices, i.e. $Q = (q, c_1, c_2, c_3, c_4)$. Our task is to predict the correct choice C of query Q based on $\mathbb{T}$. We assume that each case has one correct answer. In training set, each case contains a MCQ query Q, its correct choice C, and source cells of answer $\left\{T_i(j,k) | i \in 1...n, j \in 1...d_r^{T_i}, k \in 1...d_c^{T_i}\right\}$, where $T_i(j,k)$ is the cell at i-th row and j-th column of T_i, and $d_r^{T_i}$ and $d_c^{T_i}$ are the numbers of rows and columns of T_i. The system is evaluated by accuracy of correct selections.

3 System Architecture and Training

To narrow down the range of candidate tables, our system uses an end-to-end model which has high recall to filter out unrelated tables. After that, our model searches in candidate tables, and obtains a list

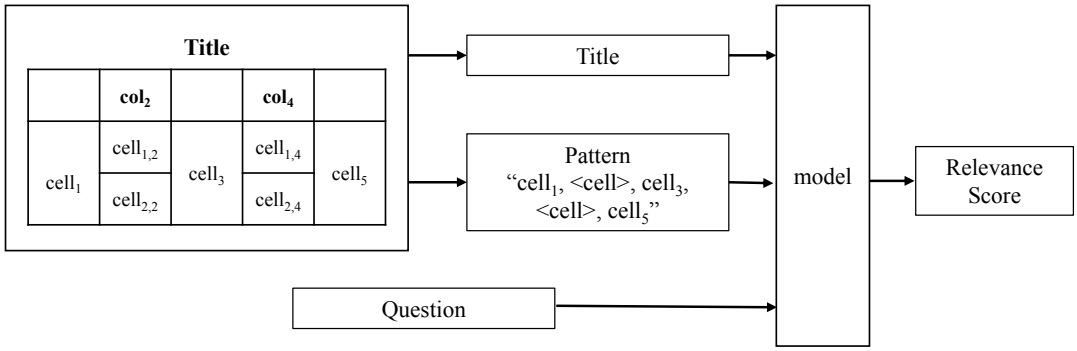


Figure 2: The system structure to select candidate tables.

of candidate cells. Then an attentive recurrent model will measure the relevance of all these cells with the query.

3.1 Candidate Table Selection

In this section, we will introduce the model to select related tables of a given question. The structure is shown in Figure 2. We use DiSAN (Shen et al., 2017) to encode the tables and questions separately, and then measure relevance. The tables with highest relevance scores are selected as input of answer selection model.

Directional Self-Attention Network (DiSAN): DiSAN is a network to learn sentence representations, instead of convolution neural networks (CNN) and recurrent neural networks (RNN), it is based on the attention mechanism. DiSAN is made up by directional self-attention (DiSA) blocks. For each block, assume the input sequence is $x = [x_1, x_2, ..., x_n]$, we have

$$\boldsymbol{h} = \sigma_h(W^h \boldsymbol{x} + b^h), \tag{1}$$

where σ_h is an activation function. Then given masks M, we can use masks to encode temporal order information into the output. In DiSA, the masks are the forward mask and the backward mask, i.e.

$$M_{ij}^{fw} = \begin{cases} 0 & i < j, \\ -\infty & \text{otherwise.} \end{cases}, \tag{2}$$

$$M_{ij}^{bw} = \begin{cases} 0 & i > j, \\ -\infty & \text{otherwise.} \end{cases}, \tag{3}$$

Define $p(z_k = i|x, q)$ as the probability of the k-th feature of token x_i contributes important information to q, and $\mathbf{1}$ represents all-one vector. In the multi-dimensional attention layer, we have a feature-wise score vector instead of a single scalar score, that is

$$f(h_i, h_j) = c \cdot \tanh\left([W^{(1)} h_i + W^{(2)} h_j + b^{(1)}]/c\right) + M_{ij}\mathbf{1}, \tag{4}$$

$W^{(1)}, W^{(2)} \in \mathbb{R}^{d_h \times d_h}$ and $b^{(1)} \in \mathbb{R}^{d_h}$ are parameters. When $M = M^{fw}$, only the relation of later j and earlier i will be in attention, and when $M = M^{bw}$, only earlier j and later i will be focused. Now, we can obtain the importance weight of each feature k in each token i, i.e.

$$P(z_k = i|\boldsymbol{h}, h_j) = \text{softmax}(f(h_i, h_j)). \tag{5}$$

Then, the output of x_j can be written as

$$\boldsymbol{s} = [\mathbb{E}_{i \sim p(z_k|\boldsymbol{h}, h_j)} x_{ki}]_{k=1}^{d_e} = \sum_{i=1}^{n} p(z.|\boldsymbol{h}, h_j) \odot x_i. \tag{6}$$

	CHARACTERISTIC Physical characteristic of an animal or human		INHERITED? Is the characteristic inherited, learned, or acquired?
An	facial scar	is	acquired
	blue eyes	are	inherited
	long hair	is	acquired
A	broken leg	is	acquired
	strong muscles	are	acquired
	telling a story	is	learned

Figure 3: An example of the tables with a few link words.

The final output u of a DiSA block is gotten by input h and output s of the block by a fusion gate.

$$F = sigmoid(W^{(f1)}h_i + W^{(f2)}h_j + b^{(f)}), \tag{7}$$

$$\boldsymbol{u} = F \odot \boldsymbol{h} + (1 - F) \odot \boldsymbol{s}, \tag{8}$$

$W^{(f1)}, W^{(f2)} \in \mathbb{R}^{d_h \times d_h}$ and $b^{(f)} \in \mathbb{R}^{d_h}$ are parameters to be learned. DiSAN firstly applies forward blocks and backward blocks, and their outputs $\boldsymbol{u}^{fw}, \boldsymbol{u}^{bw} \in \mathbb{R}^{d_h \times n}$ are concatenated and sent to another self-attention block called source2token block, which compresses $[\boldsymbol{u}^{fw}, \boldsymbol{u}^{bw}]$ into a single vector, i.e.

$$f(u_i) = W^T(W^{(1)}u_i + b^{(1)}) + b, \tag{9}$$

$$s_{\text{disan}} = [\mathbb{E}_{i \sim p(z_k|u)} u_{ki}]_{k=1}^{d_e} = \sum_{i=1}^{n} p(z.|\boldsymbol{u}) \odot u_i \tag{10}$$

Compared to other methods, DiSAN contains fewer parameters and takes less time to train, and it works well on encoding tables and questions separately. In table selection, there are many query-table pairs, and we need to score them accurately and time-efficiently. Inspired by this method, when tables and questions are encoded, they are self-attentioned without extra information, which can reduce computation complexity of this part.

Candidate Table Selection Model: For each (query, table) pair, we will score the relevance between them. Consider a table $[x, T]$ and a question $q = \{w_{i,q}\}_{i=1}^{n}$, where x is the title of a table, T is content of the table, and $w_{i,q}$ is the i-th word in q.

We first convert words in questions into representations $I_q = \mathbb{R}^{d_w \times n}$, which are concatenation of the word-level embeddings of the words $\{e_{w_{i,q}}\}_{i=1}^{n}$ and their POS-tag embeddings $\{e_{p_{i,q}}\}_{i=1}^{n}$. The embeddings of POS tags are randomly initialized and are trained together with parameters, and the word-level embeddings are initialized by GloVe (Pennington et al., 2014) vectors. We then use a DiSAN layer described above to encode each question q:

$$E(q) = \text{DiSAN}(I_q). \tag{11}$$

Then we convert tables into their vector representations. A table contains a title and some structured data, and we encode them respectively. In some tables, there are some columns filled with link words, and each row in the table is a complete sentence, while others have few or even no link words (like Figure 3). We put those columns, which link the cells, and tags of other columns into a DiSAN encoder, and then, we measure the relevance between each table and the question by their representations, i.e.

$$E([x, T]) = W[\text{DiSAN}(I_x), \text{DiSAN}(T)] + b, \tag{12}$$

$$o(q, [x, T]) = \cos\left(E(q), E([x, T])\right). \tag{13}$$

For each question, we select the l most relevant tables. Then, rows in those tables are selected by their intersection of text with the query. Those rows are sent into the cell scoring model.

Question: Which period of daylight is the summer solstice related?

	ORBITAL EVENT		PERIOD OF DAYLIGHT		PERIOD OF NIGHT	
The	summer solstice	is the day with the	longest	period of daylight and the	shortest	period of night
The	winter solstice	is the day with the	shortest	period of daylight and the	longest	period of night
The	spring equinox	is the day with the	midrange	period of daylight and the	midrange	period of night
The	fall equinox	is the day with the	midrange	period of daylight and the	midrange	period of night

Figure 4: The table contains the answer of "Which period of daylight is the summer solstice related?", relevant rows and columns, and their intersection are marked.

3.2 Cell Scoring and Answer Selection

After finding candidate rows, we introduce a model that focuses on useful parts of tables and tries to score each answer by its relevance with the query. For most queries, answers can be selected based on cells from candidate tables, so we score each cell directly. As Figure 4 shows, for question "Which period of daylight is the summer solstice related?", we can find a row in table, which tells us summer solstice has longest period of daylight and shortest period of night, but "shortest period of night" has nothing to do with answering the query, and may mislead model to make a wrong decision. Therefore, to focus on useful information of the table, a soft attention layer is on the top of table encoder. The structure of our answer selection model is shown in Figure 5.

Soft Attention: The r-th row of table T contains several cells $\{T(r,i)\}_{i=1}^{d_c}$. The attention layer needs to find out useful cells in the candidate row (like those marked in Figure 4). When we obtain attention weights, rows in the table will be transformed to the unstructured sentences. Suppose we have m words in q and n words in $T(c)$, for each query Q and candidate row, we build a matrix $S_{q,T(r,\cdot)} = \{\text{sim}(i,j)\}_{m\times n}$, $\text{sim}(i,j)$ is cosine similarity between the vector representation of the i-th word of q and j-th word of sent$(T(r,\cdot))$. Then, attention weights W_a are calculated by a CNN, which have S as its input. Convolution layers are used to measure the similarity of n-grams between q and sent$(T(r,\cdot))$, and a pooling layer on the dimension with variable length is applied before we obtain W_a, i.e.

$$W_a = \tanh(\text{pool}(\text{conv}(S))). \tag{14}$$

Here, we add an activation layer to make sure the weights are in a small range.

Cell Scoring: In most cases, the answer candidate cells in each row are not components of a question. Therefore, for each $(q, T(r,c))$, input of the encoder does not contain the candidate answer cell $T(r,c)$, and it is replaced by a tag "⟨SPACE⟩". For some rows in same table, if we remove the answer cell, they are in same pattern (such as, in the table "Country Hemispheres", both "China" and "Japan" are in the north hemisphere, if we remove "country" cells, those two rows are both in pattern "⟨SPACE⟩ / is in / north hemisphere"). For these cells, we merge them into one group. Also, a row in the table is hard to be encoded directly, we convert each row into a word sequence, in which different cells in a row are separated by sign "||".

Suppose we have a question q with n words, and a row $T(r)$ with m words, let $I_q \in \mathbb{R}^{d_w\times n}$ and $I_{T(r,c)} \in \mathbb{R}^{d_w\times m}$ denote the embeddings of words in question q and row $T(r)$, where d_w is the dimension of word representations. Question encoding $E(q)$ and $E(q,T(r,c))$ can be obtained through a bidirectional Gated Recurrent Unit (Bi-GRU) Network:

$$E(q) = \text{Bi-GRU}(I_q), \tag{15}$$

$$E(q,T(r,c)) = \text{Bi-GRU}(W_a \cdot I_{T(r,c)}). \tag{16}$$

We concatenate the final states of two directions of Bi-GRU and take it as the representation of the question and the selected cells. Then for each $(q,T(r,c))$, we put $E(q)$ and $E(q,T(r,c))$ into a fully-connected layer and then obtain the score of each pair, i.e.

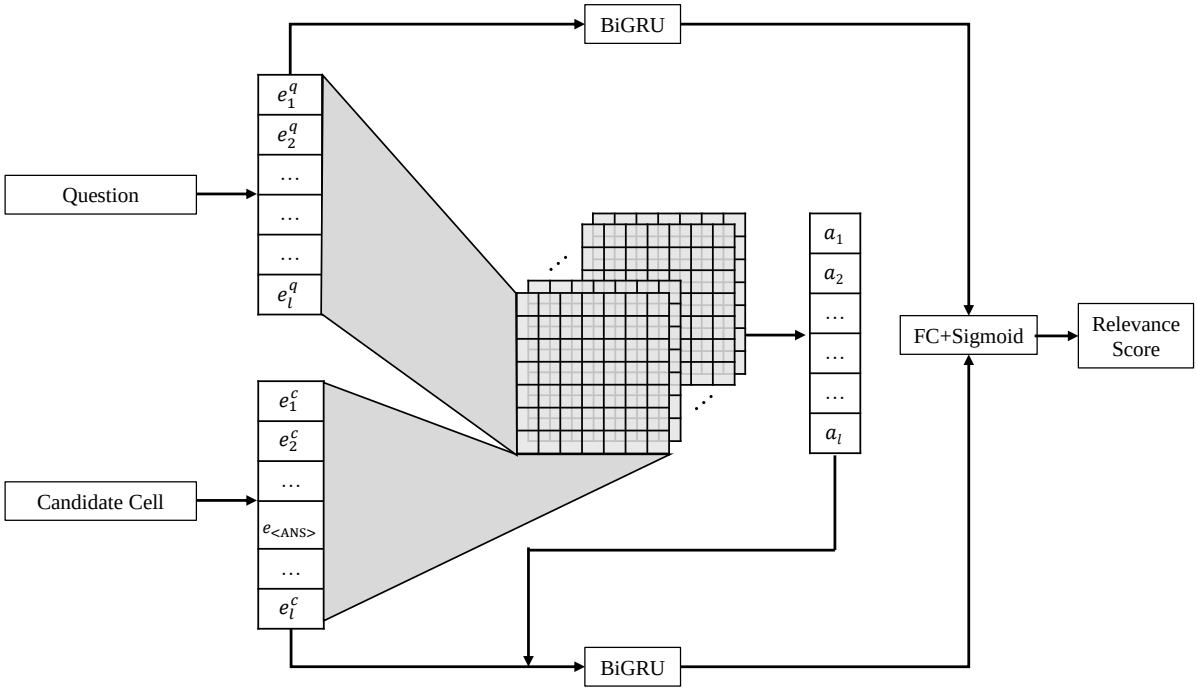


Figure 5: The structure of the answer scoring model.

$$s = rel(q, T(r, c)) = \text{sigmoid}(W \cdot \text{FC}(E(q), E(q, T(r, c))) + b). \tag{17}$$

Answer Selection from the candidate cells: After we score each rows, we need to select the best answer from the three or four candidates. Because of some queries have a choice like "none of above", we should also determine whether there are true answers in all choices. For each query Q, we order all patterns by their scores, and for each pattern, we will find the matched words between selected cells and the choices. For those queries can not be determined by the most relevant patterns, the next pattern will be used to select choices and get its intersection with the answer set. The process finishes when only one choice is remained, or no answer is left in the answer set.

Our model is a scorer model, which can answer MCQs with multiple answers. There are two different kinds of those queries. In some of those queries, the answers are in same pattern set, and it is easy to find them out. For those cases that answers are in different set, a straightforward solution is selecting candidates of which the score is above a threshold as answers. However, there is no such case in both the training and testing set, we did not explore the way to measure the threshold.

3.3 Training

Instead of measuring the system by an absolute score for relevant and irrelevant cells, we train the model to score the best cell with the highest score. For each query, we select some wrong cells from the candidate tables of each query randomly, and train the model based on triples $(q, T(r_+, c_+), T(r_-, c_-))$. We measure the loss of each triple by:

$$L(q, T(r_+, c_+), T(r_-, c_-)) = \max(0, \alpha - s_+ + s_-) \tag{18}$$

In (18), s_+ and s_- are relevance scores between question and $T(r_+, c_+)$, $T(r_-, c_-)$, and α is a hyper-parameter to control the gap between the two scores.

4 Experiment

In this section, we will evaluate our QA system based on semi-structured knowledge. Firstly, we will introduce our dataset, evaluation metrics and setup. Then, we compare our system with other work.

Finally, we analysis the result of evaluation.

4.1 Dataset

We use the TabMCQ (Jauhar et al., 2016) dataset to evaluate our system. TabMCQ contains 9092 manually annotated multiple choice questions (MCQs) with their answers, and 63 tables as its knowledge. Tables in this task are semi-structured tables, rows in each table are sentences with well-defined recurring filler patterns. For those tables built by sentences, some of the tables contain some link words to make the sentence complete, while in other tables, all cells are meaningful. Same as simple tables, analogies between rows of tables also exist.

The target domain for the tables is the *4th grade science exam*, and most tables are constructed based on the topic. MCQs in the dataset are created by information of the row containing the target cell, and the other choices of the question should be built according to other cells of the same table. Therefore, for all MCQs in the dataset, we know the source cell to answer the MCQs, which makes it easier to train based on the dataset.

Jauhar (2017) also evaluates his model on Elementary School Science Questions (ESSQ) dataset. However, the structure of those two datasets are quite different. Length of questions in TabMCQ is much shorter than ESSQ, and in ESSQ, many questions can not be solved by searching, but analogy and reasoning. Our system is not aimed at working on these questions. Besides, ESSQ contains only 108 MCQs in its training set, which is hard to train a model. Therefore, ESSQ is not in the sources of evaluation.

4.2 Evaluation Metric

For each query, there are three or four choices. We measure the systems by their accuracy on answering questions. The cases will be set to wrong when the correct group is selected, but the model returns the wrong choice. On table selection task, the goal is to get more correct recall in limited return, so we evaluate the models by their mean average precision (acc@n), which means the proportion of correct table of the query in n most relevant tables returned by the model.

4.3 Experiment Setup

When preprocessing the corpus, we use tokenizer from Natural Language Toolkit (NLTK) (Bird, 2006). The representations of words are pre-trained by GloVe (Pennington et al., 2014), and all these embeddings are fine-tuned in the training process. The dimension of word representations in all components is 300.

Two parts of the system are trained separately. For the DiSAN layer, we have a dropout layer (Srivastava et al., 2014) with dropout rate being 0.2, and the activation function is ELU (Clevert et al., 2015). The model is trained as a classifier, whose loss is measured by cross entropy with softmax, and we minimize the loss by using Adagrad (Duchi et al., 2011) with learning rate set to 0.05. When getting attention weights, we apply padding and masking to make input into fix length. Candidate cells of the answer scoring model are from table selection model we describe in section 3.1. We use the three tables with the highest scores as its source. The size of the GRU hidden layer is 100, and input and output of each GRU layer have a dropout rate of 0.2. We train the model on batches of 32 queries, and for each query, a correct answer and a wrong answer are used to construct a training sample. Attention layer is trained together with the whole model, and the loss of the answer scoring model is minimized by Adadelta (Clevert et al., 2015) with learning rate set to 0.08. Representation of words in our system is trainable, and we evaluate our model on both pre-trained word embeddings and randomly initialized word embeddings.

To compare with other work, we split queries from TabMCQ into three parts, 10% of queries are in testing set, 10% of the rest queries are in validation set, and the remaining queries are in training set. We train our system on the training set, and finish training when accuracy on the validation set stops increasing. Besides, we also evaluate our system without pre-trained vector and attention.

4.4 Baselines

To evaluate our system, we compare our system with other work. We take some experiments to show the influence of each components of our system. For systems that need to be trained (Bi-LSTM, TabNN, and our model), models are learned by using training set and validation set, and evaluations on all models are based on test set.

Random: The answer is selected from all choices randomly. It's used to measure whether the model gains the result on this task. Because there are a few questions with only three choices, the accuracy of random selection is a little higher than 0.25.

Bi-LSTM: Each row is converted into a sequence, and put into a bidirectional LSTM network. The hidden size of the LSTM layer is 100 or 200. The model is trained by Adadelta. The model's output is the relevance score of query and row. Candidate rows of the query is obtained from the model in 3.1, and the final choice is selected by matched words.

Bi-GRU: Structure of the system is similar with Bi-LSTM, but the recurrent layer is changed to bidirectional GRU. The hidden size of the model is 120 or 200. The model is trained by Adadelta.

Lucene: Lucene is a tool for retrieving unstructured information, and it is well known as a search engine. We convert semi-structured tables into unstructured sentences, which can be indexed and searched by Lucene. For each query, the top hit returned by Lucene is regarded as the answer, and the sentence is compared with the choices, and the choice with most coincidence is selected.

TabNN: TabNN, proposed by (Jauhar, 2017), represents the state-of-the-art on the task. It is composed of two scorers: a scorer between questions and rows and the other one between choices and columns. In each component, the basic unit of table encoder is cell-LSTM, which is an LSTM layer to encode cells. Output of cell-LSTM can be used by the following row and column encoders as their input. In the evaluation, TabNN-fixed is the model with fix word representations, while TabNN-learned is the model with trainable word representations.

4.5 Results

For the whole system, accuracy on the test set of TabMCQ is shown in Table 1. Count of parameters does not include those from word representations. As we can see from Table 1, our system performs better than existing work by a large margin. Because our training is based on (query, relevant cell, improper cell) triples, and does not train by all wrong cells directly, test accuracy of our system is very close to train accuracy. Training accuracy of our model is 79.27%, while TabNN has accuracy of 68.0% on the training set, and 56.8% on the test set.

Besides, our model is less complicated than TabNN. As we can see from Table 1, our scorer has only half of the parameters, compared to the number of TabNN. In most deep learning system trained by a small dataset, model with fewer parameters can be trained more easily than complicated ones, and is less likely to overfit.

For different components of our system, soft attention is significant. As we can see from the Table 1, the system with Bi-GRU as its scorer works poor in the task, although it has similar settings in other parts with our model. That is because ordinary models are hard to classify whether the information in the tables is useless or not, but a soft attention layer can help filter out the cells, which helps ordinary models work much better. Our system benefits from using pre-trained word embedding to initialize our system.

4.6 Analysis of Candidate Selection

In section 3.1, we offer a table selection model. To evaluate the effort of the model, we compare it with some baselines, which is shown on Table 2. In our experiment, LSTM with attention has a hidden size of 120, and CNN has filters in size $(n, d_w)(n = 2, 3, 5)$ and a fully-connected layer with activation. Each of them is trained by AdaGrad optimizer.

As we can see from the result, LSTM takes a poor performance on selecting the most relevant table of the question. On acc@1, both our model and CNN work well, but the correct tables achieve higher score

Systems	Hidden Size	# Parameters	Test Accuracy
Random	-	-	25.1%
Lucene	-	-	52.2%
Bi-LSTM	100	807,005	32.9%
Bi-LSTM	200	1,930,205	32.2%
Bi-GRU	120	755,005	35.4%
Bi-GRU	200	1,445,405	34.7%
TabNN-fixed (Jauhar, 2017)	160	1,131,843	31.9%
	320	3,902,083	30.9%
TabNN-learned (Jauhar, 2017)	160	1,131,843	56.8%
	320	3,902,083	50.8%
Ours w/o pre-trained	100	**605,846**	75.1%
Ours			**79.0%**

Table 1: Comparison between systems.

Model	Acc@1	Acc@2	Acc@3
LSTM with attention	89.9%	95.7%	96.3%
CNN	94.2%	96.0%	97.1%
Ours (based on DiSAN)	96.4%	98.4%	99.0%

Table 2: Comparison between models on table selection.

on our model, which means our model selects fewer tables on the same recall of the accurate tables, and the system can obtain the correct answer more easily than using CNN.

Besides, our model runs faster than LSTM, and has almost the same running time with CNN for each batch of data. Therefore, DiSAN works well on this task in our model. There are much more relevant tables when using the three most relevant tables than the top two, and the proportion of relevant tables is not selected is small (acc@4 of our model is 99.0%, and acc@5 is 99.2%, the gap between acc@3 and acc@5 is smaller than those shown in table 2, and the improvement is little on using more candidate tables. Therefore, for each query, we have three candidate tables sent into answer scoring.

4.7 Error Analysis

Although our system is much better than existing work, there is also a big room to improve. We randomly analysis some queries that our system can not select the correct choice, including those from the training set and the test set.

Some of the errors are due to candidate selection. Although we choose the three most relevant tables as candidates, there are still a few queries matched with wrong tables. Most of those queries' corresponding tables contain only few columns, and has little feature on their patterns. An example is shown on Figure 3. Besides, in some queries, words in different patterns cannot be recognized by the system. For example, the correct choice is "USA", while the representation in the relevant cell is "the united states". It's hard to recognize by our model, and it can be solved by using large knowledge to get better representations of those words. Otherwise, for some question, we have obtained the correct group with several cells, but our answer selection method is not accurate enough, the wrong choice is selected based on other cells in the same group.

There are also some faults of tagging in the dataset. Some queries in the training set are labeled with wrong cells as their "relevant cell", such as the question "Yucatan is a state located in ?", is asking for the country of Yucatan, the "relevant cell" should be "country", but the question is marked as "subdivision". Besides, there are a few questions with two correct choices. For question "Which of the following is a carnivore?", both "Andean Cat" and "Arctic fox" are in choices, and only one of them is marked as correct.

5 Conclusion

In this paper, we present a question answering system based on semi-structured tabular data. Our system is composed of two parts, which are used to select candidate tables and give out answers. Then we take some experiments with different settings on the system, and our model is compared with previous work by their performance. Besides, we evaluate the effect of the components of the model. Due to the effect of attention layer and efficient model, our system performs better than state-of-the-art work on TabMCQ dataset.

In further research, there are still much work to be done. MCQs with multiple correct choices can be added to the dataset, and then the system can solve answer MCQs with multiple answers. Besides, recall of table selection model still can be improved. Making decision on choice selection more accurate is another important point. Answering questions with induction and reasoning is also a point to improve. Otherwise, representation of cells are also important. An embedding with both text and structure information can be helpful to improve the model. Otherwise, the manners to convert raw text and structured-tables to semi-structured tables can also be explored.

Acknowledgements

We would like to thank for anonymous reviewers for their helpful feedback. Our work is supported by National Natural Science Foundation of China under Grant No.61333018 and the National Key Research and Development Program of China under Grant No.2017YFB1002101. The corresponding author of this paper is Houfeng Wang.

References

Steven Bird. 2006. NLTK: the natural language toolkit. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 1247–1250.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289.

Mohit Dua, Sandeep Kumar, and Zorawar Singh Virk. 2013. Hindi language graphical user interface to database management system. In *12th International Conference on Machine Learning and Applications, ICMLA 2013, Miami, FL, USA, December 4-7, 2013, Volume 2*, pages 555–559.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Sujay Kumar Jauhar, Peter D. Turney, and Eduard H. Hovy. 2016. Tables as semi-structured knowledge for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Sujay Kumar Jauhar. 2017. *A Relation-Centric View of Semantic Representation Learning*. Ph.D. thesis, Carnegie Mellon University.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1400–1409.

Sewon Min, Min Joon Seo, and Hannaneh Hajishirzi. 2017. Question answering through transfer learning from large fine-grained supervision data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 510–517.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Bahar Sateli and René Witte. 2015. Automatic construction of a semantic knowledge base from ceur workshop proceedings. In *Semantic Web Evaluation Challenge*, pages 129–141. Springer.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *CoRR*, abs/1709.04696.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Svitlana Vakulenko and Vadim Savenkov. 2017. Tableqa: Question answering on tabular data. *CoRR*, abs/1705.06504.

Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 956–966.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1321–1331.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: attention-based convolutional neural network for modeling sentence pairs. *TACL*, 4:259–272.

Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362. ACM.

LCQMC: A Large-scale Chinese Question Matching Corpus

Xin Liu[†], **Qingcai Chen**[†]*, **Chong Deng**[‡],
Huajun Zeng[#], **Jing Chen**[†], **Dongfang Li**[†], **Buzhou Tang**[†]
†Shenzhen Calligraphy Digital Simulation Technology Lab, Harbin Institute of Technology, China
‡Machine Intelligence Technology, Alibaba Group
†{hit.liuxin, qingcai.chen, mcdh.chenjing, crazyofapple, tangbuzhou}@gmail.com,
‡dengchong.d@alibaba-inc.com, #aaahchi@hotmail.com

Abstract

The lack of large-scale question matching corpora greatly limits the development of matching methods in question answering (QA) system, especially for non-English languages. To ameliorate this situation, in this paper, we introduce a large-scale Chinese question matching corpus (named LCQMC), which is released to the public[1]. LCQMC is more general than paraphrase corpus as it focuses on intent matching rather than paraphrase. How to collect a large number of question pairs in variant linguistic forms, which may present the same intent, is the key point for such corpus construction. In this paper, we first use a search engine to collect large-scale question pairs related to high-frequency words from various domains, then filter irrelevant pairs by the Wasserstein distance, and finally recruit three annotators to manually check the left pairs. After this process, a question matching corpus that contains 260,068 question pairs is constructed. In order to verify the LCQMC corpus, we split it into three parts, i.e., a training set containing 238,766 question pairs, a development set with 8,802 question pairs, and a test set with 12,500 question pairs, and test several well-known sentence matching methods on it. The experimental results not only demonstrate the good quality of LCQMC, but also provide solid baseline performance for further researches on this corpus.

1 Introduction

Question matching is a fundamental task of QA, which is usually recognized as a semantic matching task, sometimes a paraphrase identification task. The goal of the task is to search questions that have similar intent as the input question from an existing database. QA system learns from a large and noisy question matching corpus, and can span a diverse set of topics (Fader et al., 2013). Semantic matching is often regarded as a binary classification problem. Given a pair of sentences, the system is asked to judge whether two sentences express the same meaning. Semantic matching algorithms are widely used in NLP applications, such as information retrieval, machine translation and knowledge-based question answering. Emerging research also shows that first story detection (Petrovic et al., 2012) and text normalization (Xu et al., 2013; Ling et al., 2013) can also benefit from semantic matching techniques.

Chinese is one of the most widely used languages in the world, and the proportion of Chinese using Internet is increasing rapidly (Qiu et al., 2013). Just the same as the English sentence matching task, many Chinese applications can also benefit from Chinese sentence matching. Though some English paraphrase corpora were published for sentence-level matching tasks, the scales are still not enough to meet the data requirements of deep learning techniques. The problem is even worse in Chinese. Researchers can rarely find large-scale Chinese semantic matching corpora (Hu et al., 2015).

One of the critical challenges in constructing sentence-level matching corpus is how to collect sentence pairs that may have the same meaning. The works in paraphrase corpora can provide some references. Some existing paraphrase corpora over the last decade try to overcome this issue with various machine

*Corresponding author

[1]The copurs can be found at http://icrc.hitsz.edu.cn/Article/show/171.html.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1952–1962
Santa Fe, New Mexico, USA, August 20-26, 2018.

learning techniques. The Quora question corpus (Iyer et al., 2017) contains sentences from the online question answering forum where the sentences are mainly questions. In the Microsoft Research Paraphrase corpus (MSRP) (Dolan et al., 2004; B. Dolan and Brockett, 2005) the sentences are distilled from a corpus of news articles gathered from thousands of news sources over an extended period (Rus et al., 2014). Multilingual corpus has also been used to construct the Paraphrase Database(PPDB, (Ganitkevitch et al., 2013; Ganitkevitch and Callison-Burch, 2014)) with machine translation methods. Twitter is also used as a source of paraphrase sentences (Xu et al., 2015).

The collection of general purpose sentence pairs with the same intent is quite difficult, but there are plenty of duplicate questions but with different forms on community question answering websites like Baidu Knows(a popular Chinese community question answering website). Some researchers (Zhang et al., 2016; Zhou and Huang, 2017) had proposed to use the resources on BaiduKnows for question retrieval repectively. However, the main difference is that these researches focused on the judgment of relevance between questions or between the candidate questions and queries instead of semantic equivalence. It naturally inspires a question that could we construct a large-scale question matching corpus based on such websites? To answer this question, in this paper, we tried to construct an open-domain large-scale Chinese question matching corpus.

The main contributions of this paper are listed as follows: 1) Based on publicly available community QA website, we construct and manually annotate an open-domain large-scale Chinese question matching corpus that contains 260,068 pairs; 2) We verify the high quality of the corpus through experiments and detailed sample analysis, which prove the feasibility of the proposed corpus constructing method. 3) We present solid baseline performances of several well-known sentences matching methods on the proposed corpus, which is very helpful for its further using in future research works.

This paper is organized as follows. Section 2 reports related works. Section 3 introduces the procedure of constructing and the annotating of the corpus in detail. Section 4 presents evaluation methods and experimental results. Section 5 gives the detailed analysis of the properties and qualities of the corpus. Section 6 makes the conclusion.

2 Related Works

Iyer et al. (2017) released the Quora question corpus, collected from the Quora forum. The corpus contains over 400,000 question pairs with binary labels corresponding to semantic equivalence or not. As we known, the Quora question corpus is by far the largest manually annotated corpus.

Dolan et al. (2004) released Microsoft Research Paraphrase Corpus. MSRP investigate unsupervised techniques to acquire monolingual sentence-level paraphrases from a corpus of temporally and topically clustered news articles collected from thousands of web-based news sources (Dolan et al., 2004). Levenshtein distance and a heuristic strategy are employed on the construction of the corpus. MSRP consists of 5,801 sentence pairs, 3,900 of which were annotated as paraphrases by human annotators with a binary judgment as to whether the pair constitutes a paraphrase. The MSRP corpus is divided into a training set (4,076 pairs) and a test set (1,725 pairs), and paraphrases in both sets are about twice more than that are not (Rus et al., 2014).

The User Language Paraphrase corpus (ULPC, (McCarthy and McNamara, 2011)) comprises about two thousand target-sentence/student response text-pairs, or protocols. Unlike existing paraphrase corpora, these pairs in ULPC have been evaluated by expert human raters along 10 dimensions of paraphrase characteristics, with a six-point scale rather than traditional binary value evaluating (Rus et al., 2014). From a total of 1,998 pairs, 1,436 (71%) are classified by experts as being paraphrases.

The Question Paraphrase corpus (Bernhard and Gurevych, 2008) contains 1,000 questions along with their paraphrases(totally 7,434 question paraphrases). These questions are randomly selected from 100 FAQ files in the education category of the WikiAnswers website. The question paraphrase corpus constitutes paraphrases by retrieving question paraphrases with the input questions from social Q&A sites.

Rus et al. (2012) developed the SIMILAR corpus to foster a deeper and qualitative understanding of word-to-word semantic similarity metrics. They focus on the more general problem of text-to-text semantic similarity (Rus et al., 2012). They reuse the sentences from MSRP with word-to-word semantic

Seeds	Returned sentences
学英语，快点 English learning, Efficient	S_1:怎样学习英语才能又快又好? En:How to be fast and efficient when learning English?
	S_2:怎样快速提高英语的学习能力 En:How to improve the ability of learning English quickly
	S_3: 英语零基础怎么学习才可以很快进步? En:How to improve on learning English without any foundations?
	S_4:希望你快点学好中文英语怎么说 En:How to say "I hope you learn Chinese fast" in English
	S_5:有什么软件可以快点学英语 En:Is there any software to help to learn English fast

Table 1: Simplified example of returned questions by Baidu Knows with the seed query "English learning, Efficient". Each "En" labelled sentence is the English translation of corresponding Chinese sentence.

similarity information. There are 700 pairs in the SIMILAR corpus, and the creators relabelled the semantic equivalence of the selected pairs. 63%(442) TRUE paraphrases are yielded for an overall agreement rate with the MSRP annotations.

There are two versions of the Paraphrase database(PPDB, (Ganitkevitch et al., 2013; Ganitkevitch and Callison-Burch, 2014)), either of which contains millions of multilingual sentence pairs. The available Chinese part tends to concentrate on lexical, phrasal and syntactic forms, rather than complete sentences. Since these paraphrase pairs are automatically constructed, there is no manual annotation.

The other paraphrase corpora mainly come from SemEval task, e.g. the semantic textual similarity corpus(SRA, (Dzikovska et al., 2013)), semantic textual similarity (Agirre et al., 2013), and Twitter paraphrase corpus (Xu et al., 2015; Lan et al., 2017).

As can be seen, there is a myriad of data sets with a large variety of distributions, annotation styles, data sources, etc. To the best of our knowledge, there are no more than 50 thousand manually annotated pairs contained in existing corpora except the Quora corpus.

3 Constructing LCQMC Corpus

In this paper, we use Baidu Knows as the original data source to collect the large-scale sentence pairs. On Baidu Knows, a registered user puts a question and motivates other members to answer it. Each time we feed a query into Baidu Knows, the search engine returns a list of ranked links that point to the question pages with relevant results. Question pages are made up of question sentences, question descriptions, and their corresponding answers. We do this for two reasons: 1) there are nearly one billion real Chinese questions asked by substantial users; 2) there are plenty of duplicate questions with the great variant of expressions, which provide the possibility of constructing large-scale sentence-level pairs for question matching.

3.1 Data Collection

Table 1 illustrates a simplified example of returned questions for a given seed query. It shows that the returned sentences are most relevant, and some sentence pairs that have the same intent can be treated as matching pairs (e.g. the sentences that have the same intent as each other, i.e., S_1, S_2 and S_3). Meanwhile, we should also note that many relevant sentences imply different intents and should be non-matching. (e.g. the pairs S_4 and S_5). So it requires us to identify the matching pairs among these returned sentences, either manually or automatically.

We collect high-frequency words from a wide range of domains, including daily life, education, entertainment, computer games, social, natural science, and sports etc. About 50 unique words of each domain are selected as initial seeds. These initial seeds are feed into Baidu Knows and the top 100 returned pages of each seed are collected. Each page contains a certain number of sentences(e.g. question sentences, question descriptions and their corresponding answers). The purpose of this step is to generate

more seeds. We apply term frequency and inverse document frequency(tf.idf) weight on the sentences retrieved by initial seeds. The words in each sentence are calculated and ranked by tf.idf in descending order. Words with higher ranks are selected as the additional seeds and feed into Baidu Knows again. Top 50 pages of each seed are retained, and the question sentences are extracted to compose candidate pairs. The pairs that are identical or differ only by punctuation were discarded. By this way, more than five million question pairs are created.

3.2 Wasserstein Distance based Filtering

Candidate pairs created in the above step are extremely redundant. It's a waste of efforts to annotate all the pairs. To further filter the candidate question pairs, a Wasserstein distance based algorithm called word mover distance(WMD) (Kusner et al., 2015) is employed.

WMD can be cast as an instance of the Wasserstein distance, a well studied transportation problem for which several highly efficient solvers have been developed. WMD relies on word embeddings. It measures the dissimilarity between two sentences as the minimum amount of distance that the embedded words of one sentence need to "travel" to reach the embedded words of another sentence. The more similar the pair is, the smaller the distance value between two sentences is.

We calculate the WMD of each pair and find that almost all the pairs with the distance falling into the interval between 0 and 0.15 are matching, but their expressions do not show big differences. At the same time, all the pairs falling into the interval between 0.45 and 1 are obviously not matching. The rest pairs with WMD values between 0.15 and 0.45 contain both matching and non-matching pairs. It is very hard for WMD to distinguish these pairs. Then we randomly select 10% pairs of this part (that is 260,068 pairs) for manually annotation.

3.3 Annotating Corpus

The main difference between annotating question pairs and paraphrases is the definition of "matching". A paraphrase is a restatement of a text, passage giving the meaning in another form. Though the definition of matching in LCQMC is to some extent similar, it takes the intent of questions into consideration. So, LCQMC will not only focus on the form of words or phrases, which means that even some pairs are different on the semantic meaning, they may still be regarded as matching pairs. The pair " S_6:*After a tense stand-off, the battlewagon turned back.*" and " S_7:*After the French threatened to open fire, the battlewagon turned back*" from MSRP is a typical case in paraphrase definition, sentence S_6 is a restatement of sentence S_7 with words or phrases in another form. While the pair "S_8:我手机丢了，我想换个手机(*My cell phone was lost, now I will have to change another one.*)" and "S_9:我想买个新手机，求推荐(*I want to buy a new cell phone, any advice?*)" is the case that satisfies the definition of question matching with the same intent. The meaning in a single sentence is clear and seems to be different with the other, but if we consider the intent in the questions, we know that both sentences are asked for the same answer: *suggestion for cell phone*.

The annotation is conducted via crowdsourcing. The selected pairs are scored by three annotators. All annotators have professional Chinese background and have been trained on making the judgment for ambiguous pairs. The annotators follow the 3-point Likert scale to measure the degree of semantic similarity between sentences with the score "1", "0" and "0.5". The measurement is the same as defined by Agirre et al. (2012). According to our statistics, in our constructing process, there are about 15% pairs completely inconsistent among three annotators and about 20% pairs uncertain annotations by the annotators. For these pairs, we make a second annotation by other annotators until at least 2/3 annotators give the consistent and certain decisions. Besides, after annotating all pairs, the proportion of positive examples and negative examples is about 7:3, in order to balance the distribution, we first discard some positive pairs that show low quality and supplement some random negative pairs, then we select positive and negative pairs to construct LCQMC. Finally, the annotation pairs are divided into three parts: the training set, the validation set, and the test set. The validation set and test set are further reviewed by expert annotations. The distribution of LCQMC corpus is given in Table 2.

Data	Total	Positive	Negative
training	238,766	138,574	100,192
validation	8,802	4,402	4,400
test	12,500	6,250	6,250

Table 2: The distribution of different data sets in LCQMC corpus.

4 Evaluation and Experimental Results

We evaluate the LCQMC corpus with unsupervised and supervised methods respectively. These methods have been proven effective on semantic matching. Unsupervised methods simply make the matching judgment by using two types of similarity computing methods: string similarity methods and vector space methods. Supervised methods, those algorithms have been proven effective on sentence matching tasks conducted on English corpus(e.g. the Quora question corpus, MSRP, SemEval, PPDB etc.).

For each evaluation data set, we compute the precision(P), recall(R) and F1 score of matching. We also compute the main summary metric (Dzikovska et al., 2013): accuracy(Acc). Accuracy is the overall percentage of correctly classified examples.

4.1 Evaluation Methods

4.1.1 Unsupervised Question Matching

To measure the difficulties of the corpus, we firstly use unsupervised matching methods based on word overlap, n-gram overlap, edit distance and cosine similarity respectively. Word overlap coefficient is the average of simple word overlap, that is the number of common words divided by the average length of the two sentences (Dolan et al., 2004). It is computed by the following formula:

$$C_{wo} = \frac{|S_1 \cap S_2|}{avg(|S_1|, |S_2|)} \tag{1}$$

Similarly, the n-gram overlap coefficient is computed by the following formula:

$$C_{ngram} = \frac{1}{N} \sum_{n=1}^{N} \frac{|G_n(S_1) \cap G_n(S_2)|}{avg(|G_n(S_1)|, |G_n(S_2)|)} \tag{2}$$

where $G_n(S)$ is the set of n-grams in sentence S, and N is usually set to 4 (Barzilay and Lee, 2003).

The edit distance(D_{edt}) of two sentences is the number of words that need to be substituted, inserted, or deleted, to transform one sentence into the other (Bernhard and Gurevych, 2008).

The cosine similarity S_{cos} is computed based on the tf.idf coefficient.

4.1.2 Supervised Question Matching

For supervised question matching, we compare the continuous bag of words and deep neural network methods that have been proven useful for paraphrase task. To provide baseline performance on LCQMC, we run several methods on this corpus in the following sections. The training set is used to train each model, validation set is for parameters selecting and the results are reported on the test set. These baseline methods are introduced below.

- Continuous bag of words(CBOW): first we represent each character or word in one sentence with embeddings orderly. The embeddings are pre-trained with the original sentences. Second, each sentence is represented as the sum of the embedding representations. Third, the output is predicted by feeding concatenated representation of both sentence into a softmax classifier (Blacoe and Lapata, 2012; Yin and Schütze, 2015).

Methods	Emb	P	R	F1	Acc
1.baseline	c	67.0	81.2	73.4	70.6
(WMD)	w	64.4	78.6	70.8	60.0
2.C_{wo}	-	61.1	83.6	70.6	70.7
3.C_{ngram}	-	52.3	89.3	66.0	61.2
4.D_{edt}	-	46.5	86.4	60.5	52.3
5.S_{cos}	-	60.1	88.7	71.6	70.3
6.CBOW	c	66.5	82.8	73.8	70.6
	w	67.9	89.9	77.4	73.7
7.CNN	c	67.1	85.6	75.2	71.8
	w	68.4	84.6	75.7	72.8
8.BiLSTM	c	67.4	91.0	77.5	73.5
	w	70.6	89.3	78.92	76.1
9.BiMPM	c	77.6	**93.9**	**85.0**	**83.4**
	w	**77.7**	93.5	84.9	83.3

Table 3: The performance of different methods on LCQMC test set. 'c' means embeddings are character-based and 'w' means word-based.

- Convolutional neural network(CNN): each sentence is represented as an embedding matrix, and the matrix goes through a convolutional neural network (Hu et al., 2014). In this experiment, we make two sentence matrices sharing the same weight of convolution layers, and the convolutional operations follow the convolution in Kim (2014).
- Bi-directional Long Short Term Memory(BiLSTM): first, two sentences go through the same LSTM unit and are encoded into sentence vectors with LSTM encoder in forward and backward direction. Second, we concatenate the representations of both sentences and use softmax to make a classification (Mueller and Thyagarajan, 2016; Tomar et al., 2017).
- Bilateral Multi-Perspective Matching(BiMPM (Wang et al., 2017)): BiMPM uses a character-based LSTM at its input representation layer, a layer of BiLSTMs for computing context information, four different types of multi-perspective matching layers, an additional BiLSTM aggregation layer, followed by a two-layer feedforward network for prediction. BiMPM model has shown state-of-art performance on several NLP task, e.g. paraphrase identification, natural language inference and answer sentence selection.

4.2 Experimental Results

4.2.1 Experimental details

The tool for Chinese word segmentation is jieba[2] and toolkit for computing distance and tf.idf is sklearn[3]. In unsupervised methods, we take the validation set out to choose the threshold for unsupervised methods. The thresholds used to distinguish matching pairs for word overlap coefficient, n-gram overlap coefficient, edit distance, and cosine similarity are 0.65, 0.2, 0.2, 0.7 respectively, the value over thresholds means the pair is matching. In supervised methods, the methods are based on word2vec. We apply gensim[4] to calculate word embeddings on LCQMC. The dimension of embeddings is 200.

4.2.2 Results

Table 3 shows the results of baseline(line 1), unsupervised methods(line 2-5) and supervised paraphrase methods(line 6-9) on the test set. We list WMD performance as the baseline just because it is used to filter the original question pairs. Unsupervised methods try to evaluate the quality of the corpus. The

[2]https://pypi.python.org/pypi/jieba/
[3]http://scikit-learn.org/
[4]http://radimrehurek.com/gensim/index.html

Example	Matching?	Pairs
1	No	S_{10}:飞行员没钱买房怎么办? En: What can pilots do since they could not **afford** a **house**? S_{11}:父母没钱买房子 En: Parents can not **afford** the **house**.
2	Yes	S_{12}: 聊天室都有哪些好的 En: Are there any better **chat rooms**? S_{13}:聊天室哪个好 En: Which **chat room** is better?
3	Yes	S_{14}:不锈钢上贴的模怎么去除 En: How to wipe off the **film** on **stainless steel** S_{15}:不锈钢上的胶怎么去除 En: How to wipe off the **glue** on **stainless steel**
4	No	S_{16}:动漫人物的口头禅 En: The pet phrase of **character in animation** S_{17}:白羊座的动漫人物。 En: The **characters in animation** who are Aries

Table 4: Examples from the same keywords.

results of each method are listed from line 2 to line 5 in Table 3. All methods show a high recall but with very low precision. This is because the matching pairs in LCQMC indeed share some overlaps, but at the same time, the non-matching pairs also satisfy the overlaps. Obviously, the best performance of unsupervised methods stays around 70% either F1 or Acc. In supervised methods, BiMPM shows the best performance on F1 measure is 85.0% reached by char-based model, which outperforms the char-based WMD 11.6% of F1. The other methods also show significant improvements after being trained with the training set. The performance of CNN model is slightly lower than BiLSTM model, this may be that in CNN model we follow the TextCNN structure while it is not fit for sentence matching, which is actually proposed for classification.

Comparing the experimental results for unsupervised and supervised methods(shown in line 2-5 and line 6-9 of Table 3 respectively), we can see that there is about the average of 10% absolute improvement on both F1 and Acc score, and up to about average of 15% improvement on precision. It not only shows the effectiveness of supervised methods but also proves the effectiveness of the proposed LCQMC corpus. Though through training on LCQMC, we get a big performance gain for the questions matching task, compared to other natural language processing tasks, there is still a big margin left for further research, especially the precision of matching on this corpus is far from satisfaction.

5 Discussion

In this section, we discuss three key issues about the principle of our corpus construction methodology, i.e., the keyword based methodology, the overlap in sentence pairs and the distribution of matching types in the corpus. We make the discussion based on a random selection of 1000 samples form LCQMC.

5.1 Keyword based Methodology

In this paper, to overcome the bottleneck of collecting large-scale real questions that may have the same intent, the keyword-based method is proposed. In general, keyword matching is not a requirement of paraphrase construction and solely relying on keyword matching may limit the quality of the corpus. Taking this issue into consideration, in this paper, the keyword-based searching technology just provides a source for matching instead of the final decision. Keywords give the specific domain information, but the intent of the sentences with the same keywords may differ a lot.

Example 1 in Table 4 is a pair of sentences selected from the search results of the same keywords (bold words). We can clearly see that the sentences S_{10} and S_{11} in the pair are expressing different meaning.

Example	Matching?	Categories	Pairs	Proportion(%)
5	No	High lexical overlap but not paraphrase	S_{18}:从广州到长沙在哪里定高铁票 En: Where to buy high-speed rail tickets from Guangzhou to Changsha S_{19}:在长沙那里坐高铁回广州？ En: Where to take high-speed rail to Guangzhou in Changsha?	14.4%
6	Yes	Low lexical overlap but paraphrase	S_{20}:请问现在最好用的听音乐软件是什么啊 En: Please tell me what is the best software for listening music now. S_{21}:听歌用什么软件比较好 En: Which software is better for listening popular songs?	1.1%
7	Yes	Intent-based	S_{22}:谁有吃过完美的产品吗？如何？ En: Has anyone ate the product of company "PERFECT"? How? S_{23}:完美产品好不好 En: How about the product from company "PERFECT"?	2.8%
8	Yes	Elaboration	S_{24}:朱熹是哪个朝代的诗人 En: What dynasty is poet Zhuxi in? S_{25}:朱熹是宋明理学的集大成者,他生活在哪个朝代 En: Zhuxi is the integration of science in Song and Ming Dynasty? Which one does he live in?	2.4%
9	Yes	Phrasal	S_{26}:这是哪个奥特曼？ En: Which Ultraman is this? S_{27}:这是什么奥特曼… En: What Ultraman is this...	5.3%
10	Yes	Synonymy	S_{28}:网上找工作可靠吗 En: Is job hunting online available ? S_{29}:网上找工作靠谱吗 En: Is job hunting online accessible?	4.7%
11	Yes	Reordering	S_{30}:你们都喜欢火影忍者里的谁啊 En: Who do you like in Naruto? S_{31}:火影忍者里你最喜欢谁 En: In Naruto, who do you like most?	11.7%

Table 5: Categories and the corresponding pairs in LCQMC.

Example 2 in Table 4 are from the search results of the keyword "chat room", where the meanings of the sentences S_{12} and S_{13} are not determined by the keyword, but by the rest parts in the sentences. Though the lexicons are little different, we can still make the judgment that these two sentences asked for the recommendation of chat rooms and thus are matching. This kind of examples are quite common in LCQMC, we only list some in Table 4. Here Example 3 are matching while Example 4 is not.

5.2 Lexical Overlap of Questions

Another issue that we are concerning about is the high lexical overlaps between pairs of sentences. We have mentioned it in Section 4.1.1, e.g. the results by unsupervised methods. In fact, the high lexical overlaps have been common in several existing paraphrase data sets, especially those studying sentence-level paraphrases. In MSRP, the average lexical overlap equals about 70% while in the ULPC corpus the average lexical overlap is about 60%. In LCQMC, the average lexical overlap is about 75%. Of course, the problems are brought by the building method, but one main reason is that it is too hard to get such cases with low overlaps but matching in the natural background, especially for manual annotation.

The other reason may be the commonness of users to express the same thing when asking questions in the search engine. Matching pairs with low lexical overlaps are ideal instances. We cannot make the corpus cover all these cases, but the cases indeed contribute to the diversity of matching pairs. Luckily, since we annotate plenty of pairs, there is also a small part falling into the low lexical overlap area. According to our statics, the cases that are matching with less than 50% overlaps are about 2.5% of the

whole corpus. Example 6 in Table 5 is an example of matching pairs with low lexical overlaps. In this kind of pairs, the overlaps are far less than 50%.

Besides, non-matching pairs are another aspect to prove the quality of the corpus. The lexical overlaps in non-matching pairs have a great influence. Just opposite to the matching pairs, non-matching pairs need to be lexical overlaps as many as possible. The more lexical overlaps there are, the harder it is for algorithms to identify. In LCQMC, the non-matching pairs satisfy the requirement very well, because the non-matching pairs are selected by WMD and only kept with smaller Wassertein distance. In most cases, the smaller distance shares the views with higher lexical overlaps except those are expressed with synonyms. Example 5 in Table 5 is a non-matching pairs. The differences between the two sentences are only a few words, which is extremely hard for algorithms to identify.

5.3 Question Matching Types

To explore types of matching pairs in LCMQC, we manually examined the random 1000 samples of sentence pairs from the corpus. The categories of intent-based, elaboration, phrasal, synonymy and reordering often appear in matching pairs.

- Intent-based: Sentence pairs are matching because of the intent instead of the semantic meaning.
- Elaboration: Sentence pairs are different on text content, with an extra word, phrase or clause in one sentence that has no counterpart in the other.
- Phrasal: A phrase in one sentence is changed with another words or phrases in the other.
- Synonymy: Sentences share a little difference but the differences are synonymy.
- Reordering: Words, phrases or other constituents occur in different orders between two sentences.

Table 5 list some categories and the corresponding pairs in LCQMC, the last column is the proportion of each category in the random 1000 samples. The sum of the proportion listed in the Table is about 44%, this is because that some common cases are not listed, e.g. matching pairs with some unnecessary words and non-matching pairs.

The categories in matching pairs are far from comprehensive, there are still some other types that a matching corpus is expected to contain. Some possible categories(e.g. Requiring world knowledge, Metaphoric, Named entity, etc.) referred from (Bernhard and Gurevych, 2008) are what semantic matching algorithms expect and need to handle.

6 Conclusion

In this paper, a large-scale Chinese question matching corpus is constructed from Baidu Knows and is manually annotated. The evaluation is conducted through unsupervised and supervised sentence matching methods. Experimental results not only show that the proposed corpus is helpful for further research on Chinese question matching and other related tasks, they also present solid baseline performance on the proposed corpus. Additionally, our further discussions show the feasibility of the proposed corpus construction methodology. Different from the word-level paraphrase corpus, the LCQMC is mainly focused on the variance of expressions for the same intent, rather than the variance of vocabularies for the same meaning.

In addition to current work, the application of more bootstrapping techniques could be one of the most fruitful research direction for further increasing the scale of the corpus. Seeking for different types of available resources is also a key work for further improving the quality of the corpus.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful feedback. This work is supported by Natural Science Foundation of China (Grant No. 61473101, 61573118), Strategic Emerging Industry Development Special Funds of Shenzhen(Grant No. JCYJ20170307150528934, JCYJ20160531192358466) and the joint project foundation of Alibaba Group.

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, Montréal, Canada, 7-8 June. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Delphine Bernhard and Iryna Gurevych, 2008. *Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, chapter Answering Learners' Questions by Retrieving Question Paraphrases from Social Q&A Sites, pages 44–52. Association for Computational Linguistics.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 546–556. Association for Computational Linguistics.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*.

Myroslava Dzikovska, Rodney Nielsen, Chris Brew, Claudia Leacock, Danilo Giampiccolo, Luisa Bentivogli, Peter Clark, Ido Dagan, and Trang Hoa Dang. 2013. Semeval-2013 task 7: The joint student response analysis and 8th recognizing textual entailment challenge. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 263–274. Association for Computational Linguistics.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1608–1618. Association for Computational Linguistics.

Juri Ganitkevitch and Chris Callison-Burch. 2014. The multilingual paraphrase database. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA).

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764. Association for Computational Linguistics.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2042–2050. Curran Associates, Inc.

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. Lcsts: A large scale chinese short text summarization dataset. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1967—1972. Association for Computational Linguistics.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2017. First quora dataset release: Question pairs.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning(ICML 2015)*, pages 957–966.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of The 2017 Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pages 1235–1245. Association for Computational Linguistics.

Wang Ling, Chris Dyer, W. Alan Black, and Isabel Trancoso. 2013. Paraphrasing 4 microblog normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 73–84. Association for Computational Linguistics.

Philip M. McCarthy and Danielle S. McNamara, 2011. *The user-language paraphrase corpus*, pages 73–89. IGI Global.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, pages 2786–2792.

Sasa Petrovic, Miles Osborne, and Victor Lavrenko. 2012. Using paraphrases for improving first story detection in news and twitter. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 338–346. Association for Computational Linguistics.

Xipeng Qiu, Qi Zhang, and Xuanjing Huang. 2013. Fudannlp: A toolkit for chinese natural language processing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 49–54. Association for Computational Linguistics.

Vasile Rus, Mihai Lintean, Cristian Moldovan, William Baggett, Nobal Niraula, and Brent Morgan. 2012. The similar corpus: A resource to foster the qualitative understanding of semantic similarity of texts. In *In Semantic Relations II: Enhancing Resources and Applications, The 8th Language Resources and Evaluation Conference (LREC 2012)*.

Vasile Rus, Rajendra Banjade, and Mihai Lintean. 2014. On paraphrase identification corpora. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA).

Gaurav Singh Tomar, Thyago Duque, Oscar Täckström, Jakob Uszkoreit, and Dipanjan Das. 2017. Neural paraphrase identification of questions with noisy pretraining. *EMNLP 2017*, page 142.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4144–4150. AAAI Press.

Wei Xu, Alan Ritter, and Ralph Grishman. 2013. Gathering and generating paraphrases from twitter with application to normalization. In *Proceedings of the Sixth Workshop on Building and Using Comparable Corpora*, pages 121–128. Association for Computational Linguistics.

Wei Xu, Chris Callison-Burch, and Bill Dolan. 2015. Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 1–11. Association for Computational Linguistics.

Wenpeng Yin and Hinrich Schütze. 2015. Discriminative phrase embedding for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1368–1373. Association for Computational Linguistics.

Kai Zhang, Wei Wu, Fang Wang, Ming Zhou, and Zhoujun Li. 2016. Learning distributed representations of data in community question answering for question retrieval. pages 533–542.

Guangyou Zhou and Jimmy Xiangji Huang. 2017. Modeling and learning distributed word representation with metadata for question retrieval. pages 1226–1239.

Genre Identification and the Compositional Effect of Genre in Literature

Joseph Worsham
Department of Computer Science
University of Colorado Colorado Springs
1420 Austin Bluffs Prkwy
Colorado Springs, CO 80918, USA
jworsha2@uccs.edu

Jugal Kalita
Department of Computer Science
University of Colorado Colorado Springs
1420 Austin Bluffs Prkwy
Colorado Springs, CO 80918, USA
jkalita@uccs.edu

Abstract

Recent advances in Natural Language Processing are finding ways to place an emphasis on the hierarchical nature of text instead of representing language as a flat sequence or unordered collection of words or letters. A human reader must capture multiple levels of abstraction and meaning in order to formulate an understanding of a document. In this paper, we address the problem of developing approaches which are capable of working with extremely large and complex literary documents to perform Genre Identification. The task is to assign the literary classification to a full-length book belonging to a corpus of literature, where the works on average are well over 200,000 words long and genre is an abstract thematic concept. We introduce the Gutenberg Dataset for Genre Identification. Additionally, we present a study on how current deep learning models compare to traditional methods for this task. The results are presented as a baseline along with findings on how using an ensemble of chapters can significantly improve results in deep learning methods. The motivation behind the ensemble of chapters method is discussed as the compositionality of subtexts which make up a larger work and contribute to the overall genre.

1 Introduction

Literary computing poses unique challenges for Natural Language Processing (NLP) and Information Retrieval (IR). Literature itself is not structured like the usual expository corpora commonly made up of Wikipedia articles or short online reviews of products and services. Rather than communicating succinct and directed information, literature is artistic and conveys complicated themes over the course of very long narratives. Given our research which shows these documents on average contain well over 200,000 words, compared to the 100-1000 words in traditional document modeling datasets, it is apparent that new techniques should be considered to effectively model documents with a much higher fidelity.

This work offers a modern take on the classification problem of Genre Identification (Kessler et al., 1997). The goal of this problem is to assign the correct literary genre, such as *Adventure stories*, *Love stories*, etc., to a piece of literature using the title and body of the work. A solution to this problem would be beneficial to all organizations which supply literary services, such as libraries, online bookstores and search engines. Specifically, approaches to solve the Genre Identification problem would provide decision making aids to library scientists, but could also help with recommendation systems and information retrieval. Additionally, this work builds further insight into the study of literature, themes and genre. The problem of Genre Identification also works toward a greater topic of full length literature understanding and comprehension with complex and evolving themes throughout the course of an entire narrative.

Given the complicated and expansive nature of these documents, traditional approaches such as bag-of-word and bag-of-n-gram models are naturally unable to capture the enduring information which can persist across paragraphs and chapters. The themes which contribute to the assignment of genre to a book are not based purely on the frequencies of words used, but the abstract concepts presented over the course of the entire work. For example, it is not uncommon for Romance novels to dive deep into dark stories of murder and deceit, while still maintaining the overall theme of Romance. With this in mind,

Proceedings of the 27th International Conference on Computational Linguistics, pages 1963–1973
Santa Fe, New Mexico, USA, August 20-26, 2018.

neural modeling approaches which learn fixed-length semantic representations of text are considered here, along with traditional bag-of-word approaches, as solutions when working on pieces of literature.

Most Neural Language Models which have been proposed are designed to capture text in a sequential fashion. These models are aimed at encoding the meanings of words or sentences based on previous items in a sequence. These works have begun to explore the application of successful architectures, such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory Networks (LSTMs) or combination thereof to tasks such as sentiment classification and topic modeling (Kim, 2014; Tang et al., 2015).

However, when working with extremely long pieces of text, many of these Neural Networks quickly become impractical. Some of the models, such as the CNN, grow increasingly larger as the inputs grow, leading to unfeasibly large network implementations. The sequential models, such as the RNNs and LSTMs, are sensitive to extremely long sequences as gradients are likely to vanish during training and they require more and more time to train as the input sequence grows, eventually becoming too slow to train. Due to these limitations, it is evident that neural language models which are capable of processing an entire work of literature end to end would require very powerful systems and specialized machinery. In an effort to apply these neural network solutions to literature datasets, this work researches the effect of modeling smaller portions of text which are extracted from the greater work. These subtexts are used to train models, and results are presented which show the compositionality of the literature; that is, how the individual subtexts, such as chapters, contribute to the whole. The following questions on compositionality are addressed through this research: how do themes evolve over the course of a book and how do these evolving themes contribute to the overall genre of a book?

2 Related Work

This research builds on current deep learning approaches to text classification and sentiment analysis. CNN-Kim has been generally credited as one of the first neural network implementations for text classification (Kim, 2014). This model incorporated a simple CNN whose convolutions would capture neighborhoods of word embeddings. It was shown that on small text classification tasks, CNN-Kim could achieve competitive results when compared to other state-of-the-art models. This work laid a foundation for neural text classification, and while it focused on classifying sentences with an average length of 10-20 words, many researchers have built on this to work with longer documents (Glorot et al., 2011; Xiao and Cho, 2016; Zhang et al., 2015).

Since this time, there have been many more complex network architectures proposed for document modeling and classification. One such work proposes a novel architecture which models documents for sentiment classification using a hierarchy of LSTMs and bidirectional Gated Neural Networks (GNNs) (Tang et al., 2015). This hierarchy was designed to capture the natural structure of language where word representations are composed into sentence representations, which are in turn used to compose a document representation. This model produced competitive results on short text classification datasets with an average word count of 150. Other hierarchical approaches have been proposed that operate on a similar structure. Another proposed technique uses an autoencoder capable of capturing an unsupervised document representation with hierarchies of LSTMs (Li et al., 2015). The training goal is to reconstruct the text from the document representation using a hierarchical decoder. Two separate works have proposed alternative techniques to learning entire document representations in a hierarchical unsupervised fashion (Le and Mikolov, 2014; Kiros et al., 2015). All of these works, while exploring very different techniques, work on short documents where a document is composed of a list of sentences.

It has been shown that many NLP architectures can be significantly improved with the introduction of one or more Attention Mechanisms (Bahdanau et al., 2014). These Attention Mechanisms work to determine how informative individual elements of text are in regards to the task being trained over. These mechanisms employ a technique that learns a set of weights used to capture relevant meaning in a sequence through a weighted average. Using multiple levels of this mechanism, a Hierarchical Attention Network has been proposed for the task of document classification (Yang et al., 2016). This technique showed sweeping improvements over all other models on the same common text classification datasets.

In parallel to the research of deep learning models capable of performing complex text classification, significant efforts have been made towards finding optimal autoencoding solutions which are capable of compressing and representing variable-length text into fixed-length semantic vectors. The most notable of these is the word2vec embeddings which use a neural autoencoder in the skip-gram fashion to build fixed-length vectors which capture the contextual meaning of individual words (Mikolov et al., 2013b). Following this, Le and Mikolov (2014) and Kiros et al. (2015) both proposed the concept of embedding entire paragraphs and documents into fixed length vectors. While there has been reasonable discussion on the effectiveness of these embeddings, it is an important aspect to consider when building deep learning approaches to NLP.

Along with the extensive work that has been produced in applying deep learning techniques to text classification problems, there is also a wealth of solutions which apply traditional machine learning techniques and NLP representations such as bag of words (BOW). Liu et al. (2009) provide an in-depth study into models and weighted term representations to be leveraged when working with imbalanced text classification datasets. Their work compared BOW, term frequency - inverse document frequency (tf-idf) and custom weighting schemes while employing naive Bayes and support vector machines (SVMs) for their classification tasks. Other works have studied the impact that models such as random forests and k-nearest neighbors (kNN) have on text classification (Xu et al., 2012). While deep learning approaches have stolen the spotlight, these techniques are still highly relevant and are strong contenders for solving the Genre Identification problem.

3 Problem Definition and Hypothesis

This paper takes an approach where a literary genre is considered to be a writing style family where texts that contain similar themes are grouped together. The idea of genre used here is the same as the poetic genre defined by Miller and Greenberg (1981): "The term GENRE refers to a mode of writing that follows certain literary rules or conventions that have come down to the poet through custom and use... just as a word has connotations, a particular genre has a wealth of associations the poet may use." With this in mind, a classification task is envisioned that will learn these literary rules and conventions in a supervised environment.

The Genre Identification problem can formally be defined as: Given a book $d \in D$, referred to as a document from here out, with word length l_w where l_w has an order of magnitude of 4 or higher, determine the genre label $g \in G$ which represents the overall theme and categorization of the document. D is the set of documents in a corpus and G is the set of unique genre labels which the books may belong to. Both D and G are supplied by the literary dataset proposed in Section 3.2. In this work only one genre label is assigned to each document and $|G| = 6$. Given the extreme lengths and literary composition of these documents, the hierarchical nature which comprises them should be noted. Each document d is composed of l_c^d chapters or sections, and each chapter c in d is composed of $l_w^{d,c}$ terms. This notation will be used to reflect the hierarchical nature of the documents.

3.1 Text Composition Hypothesis

This work hypothesizes that genre, as an overarching theme of a piece of literature, is influenced over the course of the novel by individual subtexts, such as paragraphs and chapters. This hypothesis, if true, would mean that models which are used to assign a genre to a document could leverage an ensemble of subtexts or take a hierarchical approach in order to better classify the document. The experiments and evaluations presented here will evaluate how different subtexts of these long documents can be leveraged to improve the performance of the presented models.

3.2 Gutenberg Dataset

The Project Gutenberg is an extensive web catalog containing over 56,000 e-books. Along with providing the text for all of these books, Project Gutenberg also reports a detailed index for each book which contains the title, author, publication date and Library of Congress Subject Headers (LCSHs). All of

Genre	Count	Length	Mean Chapter Count	Mean Chapter Length
Science Fiction	1,186	165,874	8.2	21,309.2
Adventure stories	595	459,556	24.5	22,288.5
Love stories	508	480,265	24.2	25,684.9
Detective and mystery stories	485	475,883	25.1	25,301.2
Historical fiction	410	558,343	27.8	26,046.5
Western stories	393	463,569	23.1	26,013.3
Total	3,577	379,101.5	19.4	23,694.6

Table 1: Gutenberg Dataset Statistics

these e-books, along with the related index file, is publicly available on the Project Gutenberg website[1]. This work proposes the Gutenberg Dataset, a subset of the full Project Gutenberg catalog, which includes works with the following LCSHs: *Science fiction*, *Adventure stories*, *Historical fiction*, *Love stories*, *Detective and mystery stories* and *Western stories*. These 6 subject headers are the genres which we attempt to classify under the Genre Identification problem. Details of the Gutenberg Dataset can be found in Table 1. Scripts to download and assemble the Gutenberg Dataset as defined here can be found in the *ai-lit* project repository[2]. All of the authors' work is also available at this repository. 30% of the documents were held out in a test set which was used to evaluate performance.

4 Approaches and Architectures

Experiments are performed using both deep learning and standard text classification models. Additionally, multiple strategies are evaluated for representing the data during training in order to compensate for the extreme lengths of the documents. The objective is to establish a baseline evaluation to the Genre Identification task as well as to research how individual subtexts, such as chapters, contribute to the overall genre of a book.

4.1 Data Preprocessing

A minimal amount of preprocessing was performed on the dataset in order to prepare it for experimentation. First, the vocabulary of the entire dataset was extracted to build a distribution of all the unique terms found in the corpus. This vocabulary was captured and used to index the terms found in each book. In order to perform the indexing, the 5,000 most common terms from the vocabulary, along with the extra term $\langle unk \rangle$, used to indicate words which are outside of the 5,000 most common, were assembled to form the term index. This term index is used to convert each document, initially represented as a list of terms, into a list of integers which index words found in the term index. It is important to note that no additional text processing was performed on the data. Traditional NLP solutions continue to remove stop words and translate each word into its linguistic stem. However, in pursuit of a complete end-to-end deep learning architecture that does not require complicated preprocessing, these steps are omitted, which will force each model to learn the significance of stop word patterns and grammatical modifiers such as tense and plurality.

The neural network architectures proposed in this work utilize the concept of word-embeddings which represents each term found in a dataset as a N-dimensional vector. These vectors capture semantic and relational meaning of terms in regards to the goal on which they are trained (Mikolov et al., 2013b). These architectures can learn their own embeddings during training time, in which each vector is learned and optimized under the constraint of genre classification (Kim, 2014). This embedding layer works like a mapping where each term index corresponds to a meaningful vector, similar to the popular *word2vec* embeddings (Mikolov et al., 2013a).

[1] https://www.gutenberg.org
[2] https://github.com/joeworsh/ai-lit

4.2 Input Methodology

Due to the length of the records in the dataset, it is very prohibitive to design neural networks which can operate on an entire record all at once. Where traditional methods, such as bag of words, will sacrifice the structure and order of terms in a record in order to reduce dimensionality, here we propose alternative methods which allows the neural network models to capture the order and structure while still maintaining reasonable model sizes. These input methodologies follow the pattern proposed by Kolcz et al. (2001) in which smaller segments of a larger work are used for classification. Input methods which are utilized for both deep learning and traditional approaches are described below.

First 5,000: The first 5,000 input methodology both trains and evaluates on only the first 5,000 words in each record. In this case, the remainder of each book is discarded. There are no records in the dataset which are shorter than 5,000 terms, thus no padding is needed with this methodology.

Last 5,000: The last 5,000 input methodology, like the first 5,000, trains and evaluates on only the last 5,000 terms found in the dataset. Again, there is no need for padding as every document is longer than 5,000 terms.

Random 5,000: The random 5,000 method will randomly extract a 5,000 term window from a record on every training batch and evaluation batch. This method effectively expands the dataset as each epoch will see a different subtext for each book. While there is no guarantee that a book will be entirely processed, this random sampling process significantly grows the content which is trained and evaluated.

All Chapters: This input methodology applies a simple pattern matching algorithm to each book in order to split the books up into chapters. In this case, entire books, both in the training set and testing set, will be processed. However, chapter lengths vary significantly from 158 terms up to 169,588 terms, and we must select a fixed length for each (given the nature of the applied models). For this method, 2,500 terms are kept from each chapter. For the chapters which are shorter than this length, a padding token, $\langle pad \rangle$, is used to bring it up to the appropriate length. When using this method, results are computed across the individual chapters, and across full books by applying a voting scheme across all the chapters in each book.

Bag-of-words: BOW is a simple vector representation of an entire document where each dimension is representative of a unique word in the corpus. Thus, the input dimensionality is equivalent to the number of terms which are maintained when building the BOW representation. This method is a very simple and useful technique for representing documents as the frequency of terms which it is made of, while sacrificing the order of the words. In these experiments, all BOW representations are normalized so that the length of a document is not favored by any learning model.

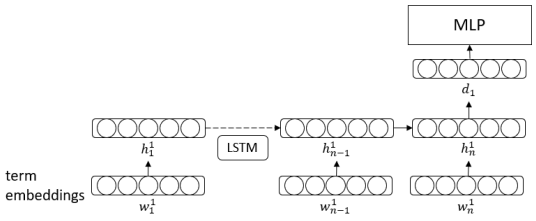
The models which are employed herein require both flat representations of the data and hierarchical representations. Hierarchical inputs will be labeled with an "*h2*" indicator to show that it is broken up into two hierarchical levels. In this case, each record, either a chapter or 5,000 term slice, is further parsed into paragraphs. Then a fixed size representation of 50 paragraphs each made up of 50 terms are captured in order to structure the records in the dataset. Thus, for all $d \in D$, $l_c^d = 50$ and $l_w^{d,c} = 50$. These numbers were selected in an effort to reduce the amount of padding included in the hierarchical models.

4.3 Models

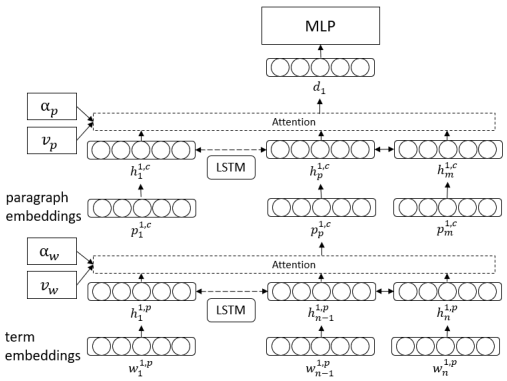
The models which are applied to the Genre Identification problem are detailed in this section. This work presents results with the CNN-Kim architecture (Kim, 2014), a flat LSTM model (Hochreiter and Schmidhuber, 1997), the Hierarchical Attention Network (HAN) (Yang et al., 2016) and a host of traditional machine learning algorithms. The authors also experimented with LSTM models for classification (Hochreiter and Schmidhuber, 1997), however due to the size of the data, they were prohibitively slow to train. Future research is needed to apply more optimized and hierarchical LSTMs to this data given the length of each text.

4.3.1 CNN-Kim

The deep learning model applied to the Genre Identification problem is the convolutional neural network (CNN) proposed by Kim (2014). This model begins with an embedding layer that learns a dense k-



(a) LSTM Architecture



(b) HAN Architecture

dimensional vector representation for each word during the training process. These experiments set $k = 100$ dimensions. Then, these word embeddings are processed in a convolutional layer by a set of filters $w \in W$ where $w \in \mathbb{R}^{hk}$. Here h is the size of the neighborhood around each word which is processed in the convolutional filter. This model employs four different h values, [3, 4, 5, 6], and 128 filters per filter shape. Following the convolutional layers, a max pooling layer is employed in order reduce the dimensionality of the previous layer and select the most informative dimensions. Finally, a fully connected layer, referred to as a multi-layer perceptron (MLP), is used to produce the network outputs. During training a dropout probability of 0.5 is applied following the convolutional layer as regularizer.

The problem is a multi-class classification task with high dependencies across very long sequences of terms. As such, this model will produce a softmax distribution of the confidence that a record belongs to each of the available classes. The softmax unit is defined as:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{1}$$

for network output vector z and unit j. In order to optimize each model for solving the classification problem, the cross-entropy between the softmax distribution p and the true genre distribution q of document x is minimized:

$$H(p, q) = -\sum_{x} p(x) \log q(x). \tag{2}$$

After the the loss of the CNN is computed, the Adam optimizer is used to train all of the network weights (Kingma and Ba, 2014) in order to minimize the loss with a learning rate of 10^{-4}.

4.3.2 LSTM

The next model applied to the Genre Identification problem is a text based Long Short-term Memory (LSTM) network. This model follows the standard LSTM architecture with a single layer where a document is represented as a sequence of term vectors (Hochreiter and Schmidhuber, 1997). Preceding the LSTM, the term index is embedded into a vector space through the embedding layer. As each term-embedded vector t is passed into the network, an LSTM cell c_t produces output h_t. The architecture of this model can be found in Figure 1a. While multiple values for the term and document embeddings were experimented with, the highest performing values were 100 and 500 respectively. The output of the LSTM is the hidden unit which represents the network output at timestep t. At each LSTM cell a dropout probability of 0.5 is applied as a regularizer.

The final LSTM hidden vector of a document sequence of length d, h_d, is used as the input to a softmax layer for performing classification over the set of possible genres and the loss is the same log loss. This loss value is minimized using the RMSProp optimzier (Tieleman and Hinton, 2012) with a learning rate of 10^{-3}.

4.4 HAN

The hierarchical nature of the HAN, plus the addition of the popular attention mechanisms make HAN a good fit for the Genre Identification problem. The encoding layers of the HAN will be made up of LSTM layers designed to capture the forward context of the text elements while focusing primarily on the element encoded at time step t (Li et al., 2015; Yang et al., 2016). The HAN architecture is detailed in Figure 1b. The attention layers of the HAN are designed to learn which encoded elements from the previous layer contribute to the classification task and which elements do not. This layer is made up of three components. The first is a simple MLP trained to build annotations over each element. The annotations learn to represent the contribution made by the element to the classification task. The measurement of the importance of the element is then computed using a softmax function over all elements at this layer. Finally a weighted sum of the encoding layer using the importance measurement is produced as a representation for the element at the next hierarchical level (Bahdanau et al., 2014). The attention mechanism at each layer is represented as:

$$u_t = \tanh(Wh_t + b) \tag{3}$$

$$\alpha_t = \frac{\exp(u_t^T u)}{\sum_t \exp(u_t^T u)} \tag{4}$$

$$s_t = \sum_t \alpha_t h_t \tag{5}$$

where h_t is the hidden state of cell t. W and b are the trainable components of the attention MLP and s_t is the weighted average output of the attention mechanism. The hierarchical nature of this model leads to three different embedding sizes, term, paragraph and document. After experimentation, the highest performing values were found to be 50, 100 and 150 respectively.

Again the architecture is finished off with a softmax layer and the loss is computed. This model employs a dropout probability of 0.5 at both the term layer and the sentence layer. Again, the Adam optimizer is employed (Kingma and Ba, 2014) with a learning rate of 10^{-3}.

4.4.1 Comparison Models

Along with the detailed deep learning models, we employed traditional machine learning models for classification in order to establish a performance baseline for the Genre Identification task. Naive Bayes (Rish, 2001), k-Nearest Neighbor (*kNN*) (Cover and Hart, 1967), Random Forest (Breiman, 2001) and XGBoost (Chen and Guestrin, 2016) are all evaluated with the BOW input method. *kNN* is different than the other models in that it simply employs a distance metric in order to identify which other documents in the dataset a new record is found to be close to. The set of nearest neighbors then vote on what the label of the new record should be. In these experiments, *k* is set to 1, 5 and 10 and the standard Euclidean distance is used (Huang, 2008).

The Random Forest model is made up of an asynchronous collection of small decision trees which are trained on random selections of features from the dataset. In this case the features are the unique tokens within the BOW input method. It has been shown that Random Forest is an effective model for classification and regression and is not prone to overfitting (Breiman, 2001). XGBoost is a highly optimized, award winning Gradient Boosting solution which is made up of a boosted set of sequential trees learned from the gradients of some differentiable loss function (Chen and Guestrin, 2016). This model often produces state-of-the-art results and is trained on the BOW form of the Gutenberg Dataset in this work.

5 Results

After training, each model was used to assign genres to the records reserved in the test set. Accuracy and the F1-Measure are used as a measure of a model's performance. The F1-Measure is defined in terms of precision and recall (Powers, 2011). The results of each experiment can be found in Table 2. Our results show that each of the models were able to generalize and capture elements that pertain

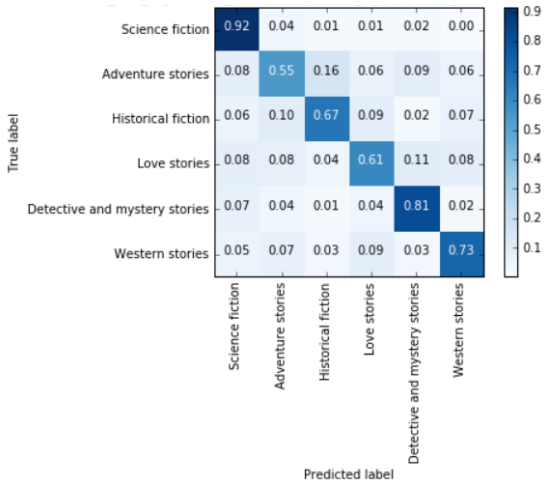


Figure 2: All Chapters CNN Confusion Matrix

Input	Model	Acc.	F1
First 5,000	CNN	0.62	0.55
Last 5,000	CNN	0.56	0.42
Random 100	LSTM	0.42	0.35
Random 5,000	LSTM	0.34	0.20
Random 5,000	CNN	0.65	0.61
All Chapters	CNN	0.75	0.71
All Chap. (h2)	HAN	0.52	0.53
BOW	Naive Bayes	0.63	0.59
BOW	kNN k=1	0.64	0.58
BOW	kNN k=5	0.61	0.55
BOW	kNN k=10	0.61	0.55
BOW	Rand Forest	0.82	0.79
BOW	XGBoost	**0.84**	**0.81**

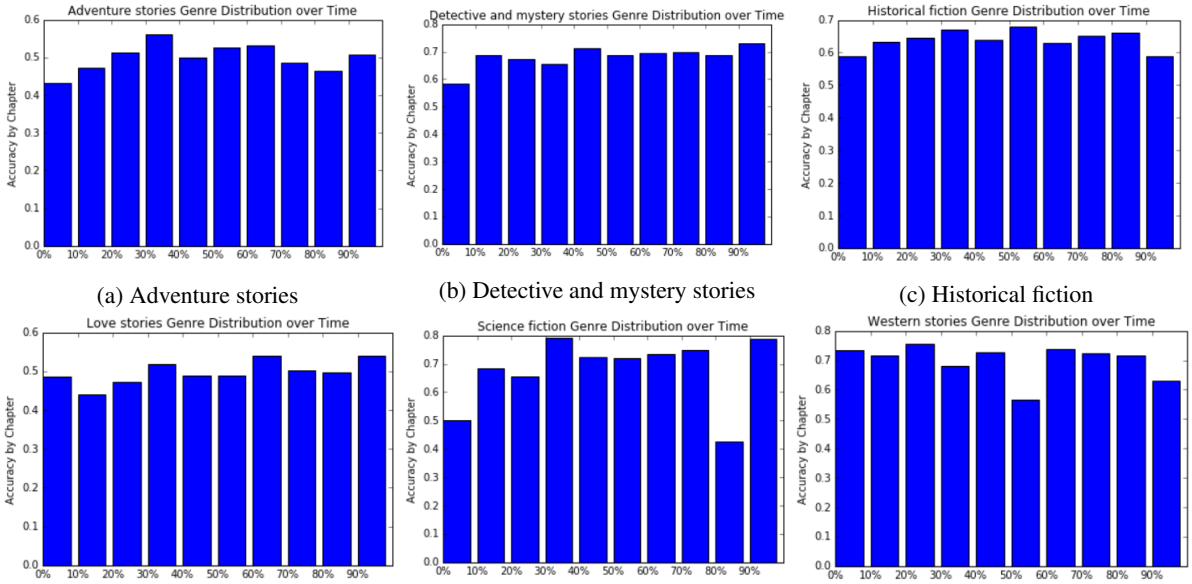
Table 2: Genre Identification Results

to genre across the different works in literature. However each model had weaknesses when working with this dataset. CNN-Kim had the most reliable performance of the deep learning models, scoring a total of 75% accuracy on the "All Chapters" input. Surprisingly, the HAN model had a difficult time converging on this dataset, frequently being fooled by the more represented classes such as *Adventure Stories* and *Science Fiction*. The LSTM, the lowest scoring neural network architecture, collapsed and was unable to learn relevant features over the long sequences. Further experiments showed that an LSTM trained on random 100 term slices from each document actually outperformed the 5,000 term model by 8% accuracy. XGBoost outperformed all models to have the highest accuracy at 84% and F1-Measure at 81%. This shows that while the deep learning models have garnered a lot of attention, tree-based methods still come out on top, and took minutes, instead of days, to train.

These results fall short of similar experiments performed by Kim (2014) using CNNs, however, the difference in data can be credited for this deviation. On a simple question classification task CNN-Kim received an accuracy rating between 91.2% and 93.6%. Lai et al. (2015) report results of 47.47% for another multi-label classification task using a CNN. Similar results have also been published using RNNs and LSTMs for sentiment classification tasks with evaluations ranging between 40% and 71% (Tang et al., 2015; Yang et al., 2016). Additionally, Liu et al. (2009) have shown the common distribution of text classification results given traditional BOW and tf-idf methods. The recorded Naive Bayes F1-Measures from this research range between 45% and 80% across many different experiments and datasets (Liu et al., 2009).

Additionally, this research has shown that across the corpus, there was a 6% gain in accuracy when training and evaluating on the first 5,000 words of each book over training and evaluating on the last 5,000. However, there was an additional 2% increase in accuracy when training and evaluating on the random 5,000 word subtexts extracted from each book. This gain in accuracy provided the insight that training our model on every chapter, and then setting up a voting algorithm where all the chapters of a book get to vote on the genre of the book could prove beneficial. Intuitively, each subtext is unique and can be leveraged in a way that expands the dataset during training allowing the models to train on new subtexts each epoch. As such, the All Chapters CNN-Kim experiment received a large 10% gain in accuracy over the best performing deep learning model with an accuracy of 75% and an F1-Measure of 71%. Figure 2 shows the confusion matrix over all classes in the test set.

Following the voting scheme method employed in the "All Chapters" technique, which yielded a high increase in performance, we digitized each chapter within their respective books into one of ten bins based on the chapter index relative to the number of chapters. That is, chapter $c \in C$ was placed in bin $b = \lfloor \frac{c}{|C|} * 10 \rfloor$. Following this, the accuracy for each bin was computed, relative to the actual genre label, to show the accuracy over the course of each book in the genre. These accuracy over time compositions



(a) Adventure stories (b) Detective and mystery stories (c) Historical fiction

(d) Love stories (e) Science fiction (f) Western stories

Figure 3: Genre composition over the course of the narrative - All Chapters CNN-Kim

for the CNN-Kim experiment are shown in Figure 3. While it is interesting to note the slight pattern variations that have emerged across the books in each genre, such as"Adventure story" novels being more difficult to classify in the first third, it is important to show that accuracy is fairly consistent across all chapters. There is no strong emerging patterns which indicate that some genres would consistently look like others at certain points in the narrative. However, this composition also provides certainty to the voting method as all chapters can be considered relatively reliable when assembling a genre classifier.

6 Future Work

Following the proposal of the Genre Identification problem and the research presented within, which shows the benefits of taking a compositional approach when working with text of this magnitude, future research will focus on building and refining hierarchical models for gains in performance. Applying a hierarchical model, like HAN, on the chapter encodings to learn classification from chapters could yield improved results over the voting scheme employed here, as a model will have the capability to learn patterns over the courses of each chapter. Given the limitations of traditional LSTMs in this context, there is a wealth of future research opportunities to apply specialized models to this domain.

Additionally, we propose future research into the compositionality of literature in the forms of comparing chapter and book similarity across the genres. Future research will study how similar chapters are to each other, and how books of the same genre may vary given additional parameters such as the title, region of origin and publication date. A study on how the title of a book contributes to the assigned genre is the next step in working with the Gutenberg dataset. There is still a lot to be learned when analyzing literature.

7 Conclusion

This work has presented the Genre Identification problem, which is a very long text classification task that requires both syntactic and thematic analysis in order to assign a literary genre to a book from a corpus. Not only is genre a challenge because it is comprised of long running themes which can span paragraphs and chapters, but these documents are significantly longer than the usual text classification datasets.

Along with the Genre Identification problem, different machine learning approaches are presented and evaluated as solutions for assigning genre. Convolutional Neural Networks, LSTMs, HAN, Naive

Bayes, k-Nearest Neighbors, Random Forests and XGBoost are considered along with different data representation schemes to help manage the extreme lengths of the documents. We showed that results were improved when training on random slices of 5,000 words from each document. Following this intuition, an ensemble method was assembled in which each chapter can vote towards the total genre of the book. This ensemble yielded a 10% increase in accuracy. Finally, the genre composition of chapters belonging to each genre has been evaluated to show how chapters contribute to the genre as a whole. There will continue to remain a significant amount of work to be done in researching new deep learning models and gaining more insight into genre and literature.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM.

Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27.

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

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

Anna Huang. 2008. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, pages 49–56.

Brett Kessler, Geoffrey Numberg, and Hinrich Schütze. 1997. Automatic detection of text genre. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 32–38. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Aleksander Kolcz, Vidya Prabakarmurthi, and Jugal Kalita. 2001. Summarization as feature selection for text categorization. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 365–370. ACM.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Ying Liu, Han Tong Loh, and Aixin Sun. 2009. Imbalanced text classification: A term weighting approach. *Expert systems with Applications*, 36(1):690–701.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Ruth Miller and Robert A Greenberg. 1981. Genre. In *Poetry*, pages 158–202. Springer.

David Martin Powers. 2011. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation.

Irina Rish. 2001. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, pages 1422–1432.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367*.

Baoxun Xu, Xiufeng Guo, Yunming Ye, and Jiefeng Cheng. 2012. An improved random forest classifier for text categorization. *JCP*, 7(12):2913–2920.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. 2016. Hierarchical attention networks for document classification. In *HLT-NAACL*, pages 1480–1489.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Transfer Learning for Entity Recognition of Novel Classes

Juan Diego Rodriguez, Adam Caldwell and **Alexander Liu**
Applied Research Laboratories
The University of Texas at Austin
Austin, Texas 78713
{juan.rodriguez, adam.caldwell, aliu}@arlut.utexas.edu

Abstract

In this reproduction paper, we replicate and extend several past studies on transfer learning for entity recognition. In particular, we are interested in entity recognition problems where the class labels in the source and target domains are different. Our work is the first direct comparison of these previously published approaches in this problem setting. In addition, we perform experiments on seven new source/target corpus pairs, nearly doubling the total number of corpus pairs that have been studied in all past work combined. Our results empirically demonstrate when each of the published approaches tends to do well. In particular, simpler approaches often work best when there is very little labeled target data, while neural transfer approaches tend to do better when there is more labeled target data.

1 Introduction

Transfer learning methods can play an important role in improving the performance of machine learning algorithms in scenarios where in-domain data is scarce, such as entity extraction of protected health information in electronic health records. These methods make use of data in a *source* domain to improve performance in a *target* domain. Many transfer learning techniques have been developed for tasks such as named entity recognition (Daumé III, 2007) and sentiment classification (Blitzer et al., 2007). A broad taxonomy of transfer learning scenarios in machine learning (only some of which are applied to NLP tasks) is given in (Zhang et al., 2017).

In this paper we focus on the entity recognition task[1] in which both the source and target domains have labeled data, but the class label sets are different. As is common in the literature, we frame entity recognition as a sequential classification problem, and restrict ourselves to local (sentence-level) supervised techniques.

Most transfer learning methods have been developed under the assumption that the source and target domains have the same class labels (e.g., (Jiang and Zhai, 2007; Daumé III, 2007)), and many NER corpora only annotate a small number of categories (e.g., 4 labels for the CoNLL 2003 corpus (Sang and Meulder, 2003) and 7 labels for the MUC 6 corpus (Grishman and Sundheim, 1996)). In many situations one would like to recognize categories that were not labeled in a given corpus, but that may be semantically related. This includes, for instance, recognizing companies or sports teams (Ritter et al., 2011), rather than organizations in general, or recognizing actors (Liu et al., 2013b), rather than people in general.

In addition, techniques for transfer learning with different class labels may be useful even when the class label set is the same across domains. As Florian et al. (2004) point out, the same labels may be used in very different ways across corpora due to different annotation guidelines. For example, determiners may or may not be annotated as part of an entity, and annotations may or may not include nominal as well as named entities.

[1] Although some researchers use the terms *entity recognition* and *named entity recognition* (NER) interchangeably, we shall refer to the task of recognizing named and nominal entities as entity recognition.

Proceedings of the 27th International Conference on Computational Linguistics, pages 1974–1985
Santa Fe, New Mexico, USA, August 20-26, 2018.

To the best of our knowledge, a direct comparison of existing transfer learning techniques for entity recognition with different labels does not exist. The results in the literature are hard to compare directly for several reasons: in some cases the datasets used are not publicly available, different evaluation measures are used for evaluation (macro-averaged vs micro-averaged scores, for instance), and in some cases, crucial implementation details are missing.

In this paper we replicate and extend several transfer learning methods for recognizing novel entities, using both standard corpora and datasets that have not been used for this task before. To the best of our knowledge, this is the first direct comparison of these transfer learning methods. In addition, we perform experiments on seven new source/target domain corpus pairs, nearly doubling the total number of corpus pairs that have been studied in all past work combined. We also share our code[2] so that others may verify our results and compare future transfer learning techniques against our benchmarks.

2 Transfer Learning for Novel Classes

2.1 Transfer learning scenarios

It is helpful to distinguish between three different scenarios for the transfer learning problem with different class labels:

1. The source domain entities are more fine-grained than the target domain entities and source domain entities can be mapped to target domain entities.

2. The source domain entities are generally coarser than the target domain entities, yet it may not be possible to map every target entity to a source entity.

3. There is no overlap in the labels in the source and target domains, and entities they correspond to have little in common.

While these three scenarios do not exhaust the possibilities that may be encountered in practice, they are helpful to categorize previous transfer learning experiments in the literature.

Under scenario 1, it is possible to map source domain labels to target domain labels without losing much information, to insure that both domains have the same label set. For example, if transferring from ACE 2005 (Walker et al., 2006) to CoNLL 2003, one could map the GEOPOLITICAL and FACILITY entities to LOCATION. This is often done manually (Daumé III, 2007; Augenstein et al., 2017), though Kim et al. (2015) introduced a method to automate this mapping: labels from both domains are embedded in a common vector space using Canonical Correlation Analysis (CCA), and k-nearest neighbors are used to find the target label closest to each source label. In their experiments the appropriate label mappings were not obvious and the CCA label embedding approach outperformed manual mappings. CCA is a dimensionality reduction technique (Hotelling, 1936) that will be described in Section 2.2.

One expects that transfer learning would be hardest in scenario 3, given that the source and target entities have little in common. For example, Qu et al. (2016) show that transfer from CoNLL 2003 (news) to CADEC (a biomedical corpus) is very difficult.

In this paper we focus on scenario 2, which is more challenging than scenario 1 and also of great practical importance. Many corpora have been published with coarse-grained entities, and it would be advantageous to harness this data to recognize other kinds of entities in other domains.

Figure 1 shows the combinations of datasets that have been used in previous transfer learning experiments relevant to the problem of disparate label sets, together with the additional experiments in this paper. We were unable to access TAC-KBP 2015 (Ji et al., 2015), i2b2 2016 (Stubbs et al., 2017) or MIMIC (Dernoncourt et al., 2017b). The TAC-KBP 2015 corpus consists mainly of news text, so we used MUC 6 and NIST IE-ER (NIST, 1999) as substitutes to see if we could obtain results consistent with those of (Obeidat et al., 2016). MIMIC is a manually corrected subset of the MIMIC-III dataset (Johnson et al., 2016). It was unclear how to replicate the manual corrections, so we leave experiments with

[2] https://github.com/ciads-ut/transfer-learning-ner

MIMIC to future work. We decided not to use ACE 2005; the annotated entities in this corpus are nested, and it was unclear which entities should be selected to fit the sequential classification framework.[3]

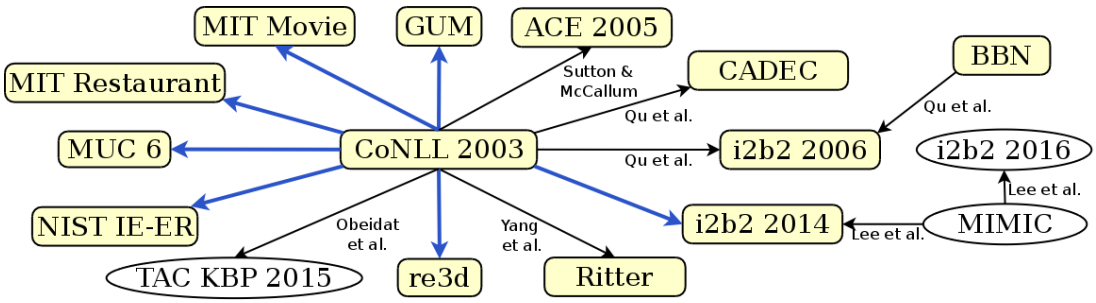


Figure 1: Source and target domains used in transfer learning experiments

Of the datasets shown in Figure 1, only CoNLL 2003, MIMIC and BBN (Weischedel and Brunstein, 2005) have been used as the source domain. Most past experiments use CoNLL 2003 as the source corpus. We replicate most of these past experiments, and extend experiments using CoNLL 2003 as a source corpus to seven additional target corpora that have not previously been used in transfer learning with novel classes for entity recognition. In future work, we plan to extend our study to additional source/target corpus pairs, particularly those that do not use CoNLL 2003 as the source corpus. Details on the datasets that we used for experiments are provided in Section 3.1.

2.2 Transfer learning frameworks

Here we briefly describe the transfer learning frameworks which are able to handle the problem of novel entity types. We use the term *framework*, rather than *algorithm*, because each of these can be implemented with a variety of machine learning models. It is important to emphasize that most existing transfer learning methods assume the label sets are the same across domains (Day and Khoshgoftaar, 2017; Weiss et al., 2016; Zhang et al., 2017). In addition, most of the methods which can handle disparate labels are designed for image and text classification rather than sequence classification problems (Day and Khoshgoftaar, 2017); some examples include (Shi et al., 2009; Shi et al., 2010; Quadrianto et al., 2010; Tommasi et al., 2010; Jie et al., 2011; Qi et al., 2011; Xiang et al., 2011; Patricia and Caputo, 2014; Feuz and Cook, 2015; Bhatt et al., 2016; Moon and Carbonell, 2016). To the best of our knowledge, the only two transfer learning approaches that have been proposed for entity recognition with different class labels are:

- **PRED:** train one classifier on the source domain, and use its predictions on the target domain as features for a second classifier.
- **Pre-training:** train a neural network on the source domain, use the trained weights to initialize a new neural network, and fine-tune the weights on the target domain.

In this paper we shall compare two variants of PRED against the Pre-training approach.

We note that transfer learning is closely related to work in *multi-task learning*. Multi-task learning consists in simultaneously training a classifier (e.g., a neural network with shared weights) to solve more than one task. In multi-task learning, however, the goal is to improve performance across all domains, rather than to optimize performance on the target domain. (Yang et al., 2017) is the only work we are aware of that uses multi-task learning on NER tasks with different label sets. For the sake of restricting the scope of this paper, in our experiments we restrict ourselves to approaches that train on the source and target data separately. A comparison to multi-task learning techniques that train simultaneously on source and target data is planned in future work.

[3]From a practical perspective, using the largest entity span is not always desirable.

PRED and PRED-CCA

It is common in NLP to use the outputs of one classifier as inputs to another one, but usually this is done for different tasks (e.g., using part-of-speech tags as features for NER), rather than for the same task with different class labels. Sutton and McCallum (2005) refer to this as "cascaded transfer"; we shall denote it PRED, following (Daumé III, 2007). A few researchers have used this approach for the entity recognition task with different class labels. Florian et al. (2004) use the output of entity taggers trained on corpora with different label types as features for a second classifier to tag entities in the ACE 2003 evaluation. They report an increase of 2% in F1 score over a baseline using lexical features with a Maximum Entropy classifier. Sutton and McCallum (2005) use PRED to transfer from CoNLL 2003 to ACE 2005, and show an improvement in F1 scores for several classes (e.g., an increase of 5.9% for person names and an increase of 1.2% for person nominals). Daumé III (2007) uses PRED to transfer from ACE 2005 to CoNLL 2003 (though source entities are mapped to target entities before the transfer[4]), and obtain a 0.82% increase in accuracy.

PRED-CCA[5] was introduced by Obeidat et al. (2016) as a variant of PRED. In PRED-CCA label embeddings rather than the predicted entities themselves are used as features for the second classifier. Label embeddings are obtained through CCA as in (Kim et al., 2015). CCA is a dimensionality reduction technique for paired data. In this case, every word is paired with its label. CCA is used to obtain label embeddings by finding a sequence of projections of the words and labels such that the correlations between the projections are maximized.

Obeidat et al. (2016) shows an improvement of 9% in micro-averaged F1 score of PRED-CCA over CCA when transferring from CoNLL 2003 to TAC-KBP 2015.

Pre-training

Fine tuning pre-trained neural networks has been used successfully in computer vision tasks, but seems comparatively less common in NLP. Mou et al. (2016) use pre-training to improve the performance of sentence and sentence-pair classification; two of their experiments transfer between corpora with different label sets. Kim et al. (2015) also experiment with pre-training for slot tagging, but only after the source domain labels have been mapped to target domain labels.

The only works we are aware of that apply pre-training to the entity recognition task are (Lee et al., 2017) and (Qu et al., 2016). Lee et al. (2017) use pre-trained neural networks to improve the performance of de-identification of protected health information in medical records, though they use two datasets with the same label set.[6] Qu et al. (2016) use pre-training in experiments with novel entities, using two conditional random fields (CRFs) rather than two neural networks. Unfortunately, we were not able to obtain the code used in Qu et al.'s experiments or to replicate their approach using new code.

Lee et al. (2017) use a bidirectional LSTM CRF (BiLSTM-CRF) neural network for their transfer learning experiments. Their network architecture is described in (Dernoncourt et al., 2017b), and is similar to the BiLSTM-CRF in (Lample et al., 2016). They experiment with transferring successive layers of the network: character embeddings, word embeddings, the character BiLSTM layer, the word BiLSTM layer, the fully connected layer, and the CRF layer. They experiment with transferring from MIMIC to i2b2 2014 and from MIMIC to i2b2 2016, and show that transferring weights of a pre-trained network boosts performance, with better results as more layers are transferred. While transferring the last (most task-specific) layers does not help as much as transferring the lower layers, it also does not hurt performance.

The results in (Lee et al., 2017) were produced with the NeuroNER program[7] (Dernoncourt et al., 2017a). Unfortunately, NeuroNER requires both source and target corpora to have the same label set, so we were not able to use it for our experiments. To run experiments similar to those in (Lee et al., 2017),

[4](Daumé III, 2007) does not explicitly describe this preprocessing step, but we infer it based on the other methods that PRED is compared against.

[5]Obeidat et al. (2016) refer to it as "AugmntTr"; we call it PRED-CCA in order to emphasize its relation to PRED.

[6]The label sets were slightly different, but in the few cases where entities were different they were mapped to a common type.

[7]Available at `https://github.com/Franck-Dernoncourt/NeuroNER/`

but with different label sets, we implemented a variant of their neural network architecture. Since our goal is to replicate the transfer learning approach, not the exact network architecture, we made one simplification: we replaced the character-level embeddings with casing embeddings following (Reimers and Gurevych, 2017). This has two advantages: it makes the neural network experiments more comparable to the CRF experiments with PRED and PRED-CCA (which use simple token features), and it reduces the number of layers that can be transferred. In our experiments the only layers that can be transferred are the word embedding layer and the word BiLSTM layer, since the last two layers depend on the target domain's label set. Details of our implementation and our choice of hyperparameters are listed in Section 3.2.

3 Experimental Setup

Our experiments compare the performance of the transfer learning methods in different settings: we vary both the target corpus and the size of the target training set. We run every experiment 5 times, using different subsets of the target data to train with, and average results.

3.1 Datasets

We used CoNLL 2003 with the standard train/test split as the source corpus for our experiments . CoNLL 2003 is annotated with Person, Location, Organization and Miscellaneous entities. Table 1 shows the corpora that we used as the target, together with their entity types. The entities which overlap with the CoNLL 2003 entities are italicized; we refer to the others as "novel entities" [8].

Corpus	**Domain**	**Annotated entities**
Ritter	Twitter	*Person*, *geo-loc*, facility, company, sportsteam, band, product, tv-show, movie, other
i2b2 2006	Medical	Date, doctor, hospital, ID, *location*, patient, phone
i2b2 2014	Medical	Age, username, ID, patient, doctor, profession, hospital, street, state, zip, city, country, *organization*, date, medical record, phone
CADEC	Medical	Adverse reaction, disease, drug, finding, symptom
MUC 6	News	*Person*, *location*, *organization*, date, percent, money
NIST IE-ER	News	*Person*, *location*, *organization*, date, duration, percent, money, cardinal
GUM	Wikinews wikihow wikivoyage	Abstract, animal, event, object, *organization*, *person*, *place*, plant, quantity, substance, time
MIT Movie	Spoken queries	Actor, character, director, genre, plot, year, soundtrack, opinion, award, origin, quote, relationship
MIT Restaurant	Spoken queries	Amenity, cuisine, dish, hours, *location*, price, rating, restaurant name
re3d	Defense and security	Document reference, *location*, military platform, money, nationality, *organization*, *person*, quantity, temporal, weapon

Table 1: Corpora used as the target domain in our experiments

Entities which occured less than 20 times in a corpus were removed[9]. We used the standard train/test splits for the i2b2 2006 (Uzuner et al., 2007), i2b2 2014 (Stubbs and Uzuner, 2015), MIT Movie (Liu et al., 2013b) and MIT Restaurant (Liu et al., 2013a) corpora. For the Ritter corpus (Ritter et al., 2011)

[8]We consider subtypes of CoNLL 2003 entities such as "doctor" and "hospital" to be novel. Although "geo-loc" and "place" are used in slightly different ways, we identify them with "location", so do not consider them to be novel.

[9]In particular, "time" was removed from both MUC 6 and NIST IE-ER; "age" was removed from i2b2 2006; "healthplan", "URL", "fax", "email", "device", "location-other" and "bioID" were removed from i2b2 2014; "commsIdentifier", "frequency" and "vehicle" were removed from re3d.

we used the same train/test split as in (Yang et al., 2017)[10]. We used stratified random sampling at the sentence level to create suitable test sets for CADEC (Karimi et al., 2015), GUM (Zeldes, 2017), NIST IE-ER (NIST, 1999), re3d (DSTL, 2017) and MUC 6 (Grishman and Sundheim, 1996)[11]. Unlike i2b2 2006, the i2b2 2014 corpus is not tokenized. To stay consistent with (Lee et al., 2017), we used the scripts included with NeuroNER with the *spacy* tokenizer to convert i2b2 2014 to the CoNLL 2003 format.

CoNLL 2003, Ritter, and i2b2 2014 also have default train/test/dev splits. In order to have more control over the sizes of the training and development sets (in particular, to have the same train/dev ratio over increasing sizes of the training set), we decided not to use the standard train/dev splits.

The available target training data is randomly shuffled, and for every shuffle we use increasing subsets of the target corpus (with 20, 50, 100, 250, 500, 1000, 1500 and 2000 sentences) as target training data. In order not to give the neural network methods an unfair advantage over PRED and PRED-CCA, we do not use a hold-out development set for them, and instead use 20% of the available training data for validation.

3.2 Implementation Details

We used the IOB2 tagging scheme for all our experiments. For every experiment we compute the micro and macro-averaged chunk-based precision, recall and F1 scores, as well as micro and macro-averaged scores over novel classes, as was done in (Qu et al., 2016). Due to space limitations we only include the micro-averaged scores in this paper, but our full results (including scores for specific labels) will be available online.

PRED and PRED-CCA

Obeidat et al.'s experiments comparing PRED and PRED-CCA were performed using *Stanford NER*, which is an implementation of a second-order linear CRF. Since we are replicating the PRED and PRED-CCA approaches rather than Obeidat's exact setup, we perform our experiments with *CRFSuite*. Although Stanford NER has an extensive set of built-in features, it is harder to modify them than it is for CRFSuite. We use the following features when evaluating PRED and PRED-CCA: (1) the current token x_i and tokens in a window of size 2, (2) initial capitalization of tokens in a window of size 2, (3) word shape information for x_i (uppercase, alphanumeric, or all digits), and (4) token prefixes of lengths 3 and 4, and token suffixes of lengths 1 through 4.

These features were used in (Zhang and Johnson, 2003)[12], and were also used as baseline features in (Turian et al., 2010). Augenstein et al. (2017) also evaluate CRFSuite on NER across a variety of datasets with features similar to these. Due to limitations of CRFSuite, we did not include second-order label transitions or features of the form $\mathbb{I}_{tag_i \wedge x_i}$. We also opted against using gazeteer, word embedding, part-of-speech or chunking features, since the goal of this paper is to compare transfer learning approaches and not achieve state-of-the-art results.

While Kim et al. (2015) and Obeidat et al. (2016) use the predicted entities without the IOB prefix as features, we experimented with using the IOB-prefixed entities for both PRED and PRED-CCA. Preliminary experiments suggested the differences were minimal, so we ran all our experiments with entity labels without the IOB prefix as features. In addition, Obeidat et al. (2016) do not specify whether they included the "O" entity label when computing the CCA label embeddings. If "O" is not included in the calculation, one must find a suitable label embedding for "O" separately. We experimented with excluding "O" from the computation and mapping "O" to the zero vector. The difference was minimal, so we ran our full experiments with the "O" label included in the CCA computation.

We use the same embedding dimension as (Obeidat et al., 2016), k=5, for all our experiments.

[10] Available at `http://kimi.ml.cmu.edu/transfer/data.tar.gz`

[11] We used 1000 sentences for the test sets of CADEC, GUM, and MUC6; for NIST IE-ER and re3d we used 690 and 200 sentences, respectively.

[12] Specifically the combination B+D+E+F, listed in their Table 1.

Pre-training

We used the same network architecture and hyperparameters as in (Lee et al., 2017) [13], with the following exceptions: (1) we replaced the character embedding layer with a casing layer as in (Reimers and Gurevych, 2017), (2) we used the IOB2 tagging scheme rather than the BIOES tagging scheme, and (3) we used RMSprop rather than stochastic gradient descent (SGD) for fine-tuning on the target dataset. Preliminary experiments showed that SGD with a learning rate of 0.005 and 100 epochs (the defaults in (Lee et al., 2017)) failed to converge for many of our datasets. We experimented with learning rates of 0.05 and 0.5, as well as with optimizers with adaptive learning rates (Adam, Adagrad, and RMSprop). Since RMSprop tended to have the highest F1 scores, and in order not to have to tune the learning rate of SGD for each dataset, we ran our final fine-tuning experiments with RMSprop, with a learning rate of 0.001.

We ran experiments with three settings: no transfer (training the network from scratch on the target corpus), transferring the word embedding layer, and transferring both the word embedding layer and the BiLSTM layer. We observed that the difference between not transferring any layers and transferring one layer was often small; this was also noted in (Lee et al., 2017). We therefore only include the results with transferring both layers and when discussing pretraining below we are referring to this case.

4 Results

We are particularly interested in comparing the behavior of the transfer learning methods as the size of the target dataset varies. This can help practitioners decide which approach is best suited to their situation. Plots of micro-averaged precision, recall and F1 scores with error bars for a few target domain datasets are shown in Figures 2, 3 and 4 [14]. We denote the no-transfer (training on the target only) baselines for the CRF and the BiLSTM-CRF as CRF-TGT and BiLSTM-TGT, respectively.

4.1 Comparing PRED/PRED-CCA and neural network approaches

The PRED/PRED-CCA approaches have greater precision than the neural network approaches in most cases (the only exceptions being GUM, with 1500 and 2000 sentences), with the greatest difference occurring when the number of target training sentences is small. For recall, the situation is not so clear. For MIT Restaurant, MIT Movie, CADEC, i2b2 2014 and i2b2 2006 (Figure 2), the neural network approaches have better recall than PRED and PRED-CCA across all dataset sizes; this results in the neural network approaches having a greater F1 score for these datasets in most cases (the exceptions being MIT Restaurant with under 500 sentences, and CADEC with 500 sentences).

On the other hand, for GUM, re3d, NIST IE-ER, MUC 6 (Figure 4) and Ritter (Figure 3), we observe that initially PRED and PRED-CCA have higher recall than the neural network approaches, but the neural networks' recall surpasses them once the target domain dataset is large enough. This occurs at 500 sentences for Ritter, NIST IE-ER and MUC 6, and at 250 sentences for re3d and GUM; it leads to a similar pattern in F1 scores: for Ritter, GUM, MUC 6 and NIST IE-ER, PRED and PRED-CCA have higher F1 scores than either of the neural network approaches for smaller target training sets, but the situation is reversed once the target domain dataset is large enough.

4.2 Comparing PRED and PRED-CCA

For most corpora, the PRED/PRED-CCA approaches improved the F1 score over the "target-only" (CRF-TGT) baseline. MIT Restaurant and MIT Movie were the only corpora where the F1 scores for CRF-TGT, PRED, and PRED-CCA were about the same. For CADEC the F1 scores for the 20 and 50 sentence settings decreased.

PRED-CCA outperformed PRED in most cases, but not always by very much. MUC 6, NIST IE-ER and Ritter show the greatest increase in F1 score, due to an increase in recall. The improvements in F1

[13] The hyperparameters can be found in the `parameters.ini` file in NeuroNER; they include using dropout of 0.5 after the word embedding layer, using the 100-dimensional Glove 6B word embeddings, and using 100 LSTM hidden units. Training on the source corpus is done using SGD with 100 epochs and patience of 10, with a learning rate of 0.005 and a gradient clipping value of 5.0.

[14] We only include three due to space limitations; the plots for the other transfer learning experiments will be available online.

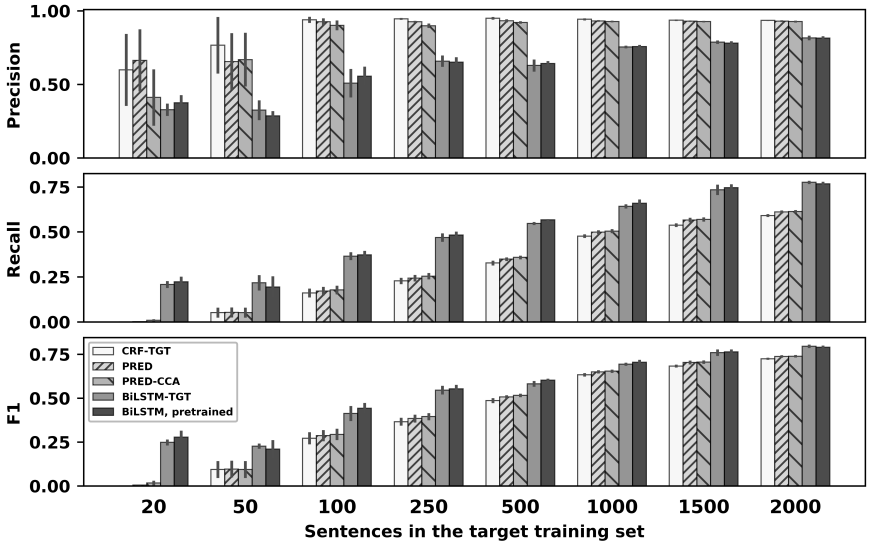


Figure 2: Micro-averaged scores for the i2b2 2006 corpus

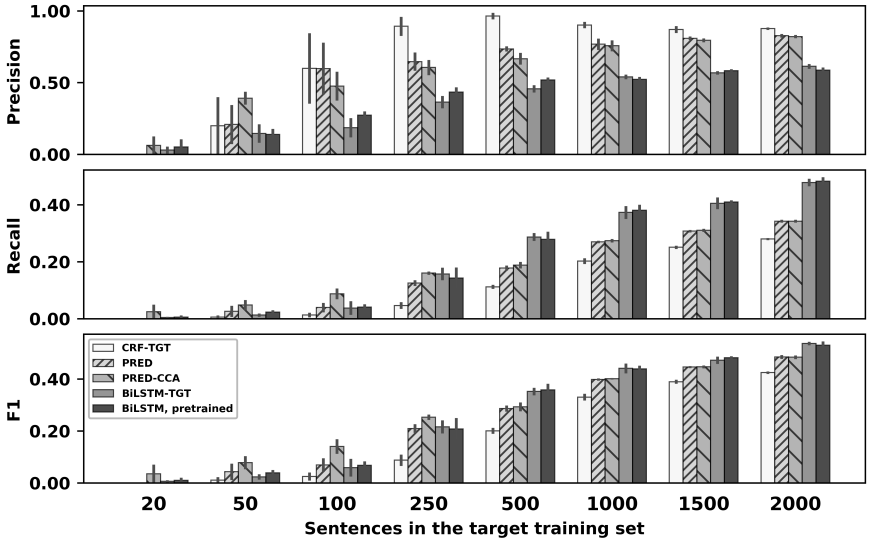


Figure 3: Micro-averaged scores for the Ritter corpus

score are highest for smaller target training datasets, and rapidly decrease with increasing dataset size. Figure 5 shows the change in F1 score between PRED and PRED-CCA for the MUC 6 and Ritter corpora. NIST IE-ER displayed a pattern similar to MUC 6, so it was omitted. re3d showed an improvement of 2% under the 20 sentence setting; the improvement for the other datasets was under 1%.

We note that in most cases, using PRED-CCA rather than PRED results in a small drop in precision; this is consistent with the results in (Obeidat et al., 2016), who observe a drop of 1% between PRED and PRED-CCA for the TAC-KBP 2015 corpus. The large increase in F1 score for MUC 6 and NIST IE-ER are consistent with the 9% increase for TAC-KBP 2015 reported in (Obeidat et al., 2016). Although the label sets are different in each case, MUC 6, NIST IE-ER and TAC-KBP 2015 are all news corpora, as is CoNLL 2003. We hypothesize that the reason PRED-CCA does so much better than PRED in these cases is that there is more of an overlap in the source and target domains' vocabularies. This would lead to more semantically similar source and target labels being mapped closer together.[15]

[15]We did not lowercase the tokens before computing the label embeddings. We expect to obtain higher F1 scores in this case, particularly for the MIT Movie and MIT Restaurant corpora, but leave this for future work.

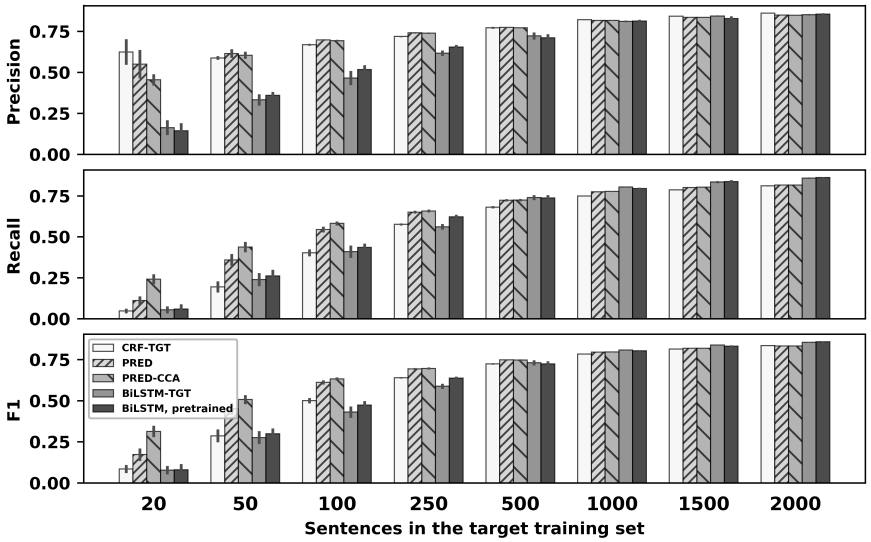


Figure 4: Micro-averaged scores for the MUC 6 corpus

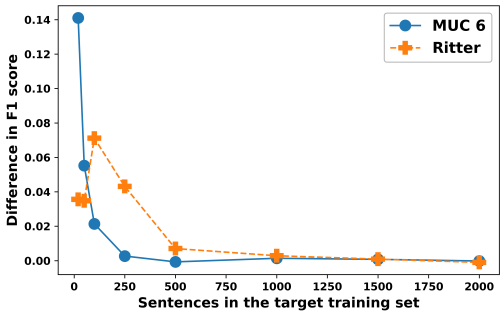


Figure 5: Difference in F1 score between PRED and PRED-CCA

4.3 Comparing the BiLSTM-CRFs with and without pre-training

The results for the pre-trained neural network were less clear. On i2b2 2006, i2b2 2014, and MIT Movie, using pre-training generally led to an improvement in F1 scores over the BiLSTM-TGT baseline. When these datasets had over 500 sentences the improvement was minimal (under 1%); for less than 500 sentences the improvement varied, but was often between 2% and 3%. In these scenarios pre-training also performed better than PRED and PRED-CCA as well. The improvement in F1 score of i2b2 2014 with 1500 sentences was 0.7%. For comparison, Lee et al. (2017) report an increase of 3% when transferring from MIMIC to i2b2 2014, when using 5% of their target set (roughly 1600 sentences) and reusing all layers up to the token layer.

For MUC 6, NIST IE-ER, re3d, Ritter, MIT Restaurant, and GUM, pre-training often achieved a higher F1 score than training from scratch; this mostly occurred when there were 500 target training sentences or less. The increase in F1 score was highest for MUC 6 (2%-5%) and for re3d (2-3%). However, in nearly all of these cases pre-training was outperformed by PRED-CCA.

5 Summary and Conclusion

In this reproduction paper, we have compared three existing methods that can be applied to the setting of transfer learning with novel entities in the target domain. These methods have not been compared against each other before in the literature.

In summary, our results show that in some situations (particularly when there is less labeled target data), PRED and PRED-CCA outperform pre-training neural networks for the entity recognition task.[16]

[16]This may not generalize to other neural network architectures, hyperparameter choices, or domain pairs.

With sufficient labeled data, the neural transfer approaches tend to do well, but of course, these correspond to cases with less data scarcity.

Thus, we encourage researchers to compare neural network approaches with simpler baselines such as PRED and PRED-CCA, rather than only comparing against the "no tranfer" option, and to report not only F1 scores, but also precision and recall, which may be useful when deciding which transfer learning approach to use.

A number of extensions to our study are possible in future work. For example, we hope to run similar comparisons with multi-task and few-shot learning approaches. In particular, we believe methods for few-shot learning in image classification could potentially be adapted to sequential classification problems such as entity recognition. A more thorough study of hyperparameter tuning (versus replicating previously published hyperparameters) would also be interesting future work, particularly for the neural network approaches, which are known to be sensitive to choices in hyperparameters.

References

Isabelle Augenstein, Leon Derczynski, and Kalina Bontcheva. 2017. Generalisation in named entity recognition: A quantitative analysis. *Computer Speech & Language*, 44:61–83.

Himanshu Sharad Bhatt, Manjira Sinha, and Shourya Roy. 2016. Cross-domain text classification with multiple domains and disparate label sets. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1641–1650.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, Prague, Czech Republic, June. Association for Computational Linguistics.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June. Association for Computational Linguistics.

Oscar Day and Taghi M. Khoshgoftaar. 2017. A survey on heterogeneous transfer learning. *Journal of Big Data*, 4(1):29.

Defence Science and Technology Laboratory. 2017. Relationship and Entity Extraction Evaluation Dataset. `https://github.com/dstl/re3d`. Accessed: January 2018.

Franck Dernoncourt, Ji Young Lee, and Peter Szolovits. 2017a. NeuroNER: an easy-to-use program for named-entity recognition based on neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 97–102, Copenhagen, Denmark, September. Association for Computational Linguistics.

Franck Dernoncourt, Ji Young Lee, Özlem Uzuner, and Peter Szolovits. 2017b. De-identification of patient notes with recurrent neural networks. *Journal of the American Medical Informatics Association*, 24(3):596–606.

Kyle D. Feuz and Diane J. Cook. 2015. Transfer learning across feature-rich heterogeneous feature spaces via feature-space remapping (FSR). *ACM Transactions on Intelligent Systems and Technology*, 6(1):3:1–3:27.

Radu Florian, Hany Hassan, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, Nicolas Nicolov, and Salim Roukos. 2004. A statistical model for multilingual entity detection and tracking. In Susan Dumais, Daniel Marcu, and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 1–8, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference- 6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Harold Hotelling. 1936. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

Heng Ji, Joel Nothman, Ben Hachey, and Radu Florian. 2015. Overview of TAC-KBP 2015 tri-lingual entity discovery and linking. In *Proceedings of the Eighth Text Analysis Conference (TAC 2015)*.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271, Prague, Czech Republic, June. Association for Computational Linguistics.

Luo Jie, Tatiana Tommasi, and Barbara Caputo. 2011. Multiclass transfer learning from unconstrained priors. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1863–1870. IEEE.

Alistair E.W. Johnson, Tom J. Pollard, Lu Shen, Li-wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data*, 3:160035.

Sarvnaz Karimi, Alejandro Metke-Jimenez, Madonna Kemp, and Chen Wang. 2015. Cadec: A corpus of adverse drug event annotations. *Journal of biomedical informatics*, 55:73–81. Available at `https://data.csiro.au` Accessed: November 2017.

Young-Bum Kim, Karl Stratos, Ruhi Sarikaya, and Minwoo Jeong. 2015. New transfer learning techniques for disparate label sets. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 473–482, Beijing, China, July. Association for Computational Linguistics.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Ji Young Lee, Franck Dernoncourt, and Peter Szolovits. 2017. Transfer Learning for Named-Entity Recognition with Neural Networks. *arXiv preprint arXiv:1705.06273*.

Jingjing Liu, Panupong Pasupat, Scott Cyphers, and Jim Glass. 2013a. Asgard: A portable architecture for multilingual dialogue systems. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8386–8390. IEEE. Available at `https://groups.csail.mit.edu/sls/downloads/restaurant/` Accessed: January 2018.

Jingjing Liu, Panupong Pasupat, Yining Wang, Scott Cyphers, and Jim Glass. 2013b. Query understanding enhanced by hierarchical parsing structures. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 72–77. IEEE. Available at `https://groups.csail.mit.edu/sls/downloads/movie/` We used the trivia10k13 portion. Accessed: January 2018.

Seungwhan Moon and Jaime Carbonell. 2016. Proactive transfer learning for heterogeneous feature and label spaces. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 706–721. Springer.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in NLP applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 479–489, Austin, Texas, November. Association for Computational Linguistics.

NIST. 1999. Information Extraction - Entity Recognition Evaluation. `http://www.nist.gov/speech/tests/ieer/er_99/er_99.htm`. We used the newswire development test data included in the NLTK package.

Rasha Obeidat, Xiaoli Z. Fern, and Prasad Tadepalli. 2016. Label embedding approach for transfer learning. In *Proceedings of the Joint International Conference on Biological Ontology and BioCreative, Corvallis, Oregon, United States, August 1-4, 2016.*

Novi Patricia and Barbara Caputo. 2014. Learning to learn, from transfer learning to domain adaptation: A unifying perspective. In *Proceedings of the Computer Vision and Pattern Recognition*.

Guo-Jun Qi, Charu Aggarwal, Yong Rui, Qi Tian, Shiyu Chang, and Thomas Huang. 2011. Towards cross-category knowledge propagation for learning visual concepts. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 897–904. IEEE.

Lizhen Qu, Gabriela Ferraro, Liyuan Zhou, Weiwei Hou, and Timothy Baldwin. 2016. Named entity recognition for novel types by transfer learning. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 899–905, Austin, Texas, November. Association for Computational Linguistics.

Novi Quadrianto, James Petterson, Tibério S. Caetano, Alex J. Smola, and S.V.N. Vishwanathan. 2010. Multitask learning without label correspondences. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1957–1965. Curran Associates, Inc.

Nils Reimers and Iryna Gurevych. 2017. Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks. *arXiv preprint arXiv:1707.06799*.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*.

Xiaoxiao Shi, Wei Fan, Qiang Yang, and Jiangtao Ren. 2009. Relaxed transfer of different classes via spectral partition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 366–381. Springer.

Xiaoxiao Shi, Qi Liu, Wei Fan, Philip S. Yu, and Ruixin Zhu. 2010. Transfer learning on heterogenous feature spaces via spectral transformation. In *2010 IEEE International Conference on Data Mining*, pages 1049–1054.

Amber Stubbs and Özlem Uzuner. 2015. Annotating longitudinal clinical narratives for de-identification: The 2014 i2b2/UTHealth corpus. *Journal of biomedical informatics*, 58:S20–S29. Available at `https://www.i2b2.org/NLP/DataSets/` Accessed: February 2018.

Amber Stubbs, Michele Filannino, and Özlem Uzuner. 2017. De-identification of psychiatric intake records: Overview of 2016 cegs n-grid shared tasks track 1. *Journal of Biomedical Informatics*, 75:S4 – S18. A Natural Language Processing Challenge for Clinical Records: Research Domains Criteria (RDoC) for Psychiatry.

Charles Sutton and Andrew McCallum. 2005. Composition of conditional random fields for transfer learning. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 748–754, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Tatiana Tommasi, Francesco Orabona, and Barbara Caputo. 2010. Safety in numbers: Learning categories from few examples with multi model knowledge transfer. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3081–3088. IEEE.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden, July. Association for Computational Linguistics.

Özlem Uzuner, Yuan Luo, and Peter Szolovits. 2007. Evaluating the state-of-the-art in automatic de-identification. *Journal of the American Medical Informatics Association*, 14(5):550–563. Available at `https://www.i2b2.org/NLP/DataSets/` Accessed: February 2018.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 Multilingual Training Corpus. *Linguistic Data Consortium, Philadelphia*.

Ralph Weischedel and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. *Linguistic Data Consortium, Philadelphia*.

Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. 2016. A survey of transfer learning. *Journal of Big Data*, 3(1):9.

Evan Wei Xiang, Sinno Jialin Pan, Weike Pan, Jian Su, and Qiang Yang. 2011. Source-selection-free transfer learning. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2355–2360. AAAI Press.

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2017. Transfer Learning for Sequence Tagging with Hierarchical Recurrent Networks. *arXiv preprint arXiv:1703.06345*.

Amir Zeldes. 2017. The GUM corpus: creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612. Available at `https://github.com/amir-zeldes/gum/tree/master/coref/tsv/` Accessed: November 2017.

Tong Zhang and David Johnson. 2003. A robust risk minimization based named entity recognition system. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 204–207.

Jing Zhang, Wanqing Li, and Philip Ogunbona. 2017. Transfer Learning for Cross-Dataset Recognition: A Survey. *arXiv preprint arXiv:1705.04396*.

Location Name Extraction from Targeted Text Streams using Gazetteer-based Statistical Language Models

Hussein S. Al-Olimat, Krishnaprasad Thirunarayan, Valerie L. Shalin and Amit Sheth
Kno.e.sis Center, Wright State University, Dayton, OH
{hussein;tkprasad;valerie;amit}@knoesis.org

Abstract

Extracting location names from informal and unstructured social media data requires the identification of referent boundaries and partitioning compound names. Variability, particularly *systematic* variability in location names (Carroll, 1983), challenges the identification task. Some of this variability can be anticipated as operations within a statistical language model, in this case drawn from gazetteers such as OpenStreetMap (OSM), Geonames, and DBpedia. This permits evaluation of an observed n-gram in Twitter targeted text as a legitimate location name variant from the same location-context. Using n-gram statistics and location-related dictionaries, our Location Name Extraction tool (LNEx) handles abbreviations and automatically filters and augments the location names in gazetteers (handling name contractions and auxiliary contents) to help detect the boundaries of multi-word location names and thereby delimit them in texts.

We evaluated our approach on 4,500 event-specific tweets from three targeted streams to compare the performance of LNEx against that of ten state-of-the-art taggers that rely on standard semantic, syntactic and/or orthographic features. LNEx improved the average F-Score by 33-179%, outperforming all taggers. Further, LNEx is capable of stream processing.[1]

1 Introduction

In context-aware computing, location is a fundamental component that supports a wide-range of applications (Hazas et al., 2004; Licht et al., 2017). During natural disasters, location is crucial for situational awareness during disaster response (Son et al., 2008). When available, targeted streams of social media data are therefore particularly valuable for disaster response (Munro, 2011)[2]. For example, the tweet "water level in Ganapathy Colony is around 2 m" refers to a location. But, unless we know where "Ganapathy Colony" is, the water level data cannot enhance situational awareness and inform disaster response applications such as storm surge modeling or forecasting.

However, pragmatic influences on writing style shorten names to reduce redundant content in social media. We call this the *location name contraction problem*. For example, "Balalok School", appears in the Chennai flood tweets in contrast to the full gazetteer name–"Balalok Matriculation Higher Secondary School". Carroll (1983) examined the complex phenomenon of alternate name forms (called Nameheads). He distinguishes between four shortening processes: (1) Appellation Formation, (2) Explicit Metonomy, (3) Category Ellipsis, and (4) Location Ellipsis.

Appellation Formation occurs when, for example, the author refers to the location name "The Erie Canal" as "The Canal". People may also refer to the only airport in the affected area as just "The Airport". Referring to "University of Michigan" as "Michigan" is an example of Explicit Metonomy. Common ground or shared understanding between the author and the recipient establishes the referent (Resnick et al., 1991). Both Appellation formation and Metonomy pose *disambiguation* problems, and require context such as the author's location to resolve.

[1] Data and the tool is available at https://github.com/halolimat/LNEx

[2] We define a *targeted stream* as a set of tweets that has the potential to satisfy an event-related information need (Piskorski and Ehrmann, 2013) crawled using keywords and hashtags, to contextualize the event (e.g., "#HarveyFlood").

Proceedings of the 27th International Conference on Computational Linguistics, pages 1986–1997
Santa Fe, New Mexico, USA, August 20-26, 2018.

In contrast, Category Ellipsis and Location Ellipsis pose *delimitation* problems that can be resolved with a statistical language model. Category Ellipsis occurs when the author strips words related to the location category (e.g., "City" from "Houston City" to become "Houston"). Location Ellipsis occurs when an author drops the specific location reference in the location name (e.g., when "New York Yankee Stadium" becomes "Yankee Stadium" or "Cars India - Adyar" becomes "Cars India").

This distinction between delimitation and disambigution is important in the location extraction literature. Entity delimitation is typically the first step of location extraction, to identify the boundaries of a location mention in the text. To address this problem, previous research (Liu et al., 2014; Malmasi and Dras, 2015; Hoang and Mothe, 2018) has applied both syntactic heuristics (using lexical cues, e.g., "*in* New Orleans") as well as semantic heuristics (i.e., content-based, for different types of locations such as *buildings* and *streets*). These heuristics have serious limitations, such as failing to delimit metonyms and location names that begin sentences (i.e., outside locative expressions, e.g., "*New Orleans* is flooded") and they cannot assist in hashtag segmentation (needed to extract locations from hashtags). Moreover, simply identifying a location name still leaves open the problem of linking the entity to a corresponding gazetteer record for geocoding. Simple fixed phrase matching with gazetteers entries, as in (Middleton et al., 2014; Malmasi and Dras, 2015), solves the linking problem, but remains vulnerable in two respects. With simple fixed phrase matching, the tendency for authors to shorten names while the gazetteers extend names, creates conflicting conditions causing poor recall. On the other hand, simply relaxing matching criteria exacerbates the disambiguation problem.

To address delimitation, we treat location names as a sequence of ordered words known as *collocations* (Manning and Schütze, 1999). Collocations are neither strictly compositional nor always atomic. We cannot identify them with grammatical rules, and fixed phrase matching is not reliable for longer names. Fortunately, the gazetteer provides a resource to establish region-specific naming regularities. Given a region-specific gazetteer, which retains the same location-context as the text, we can construct a statistical model of the *token sequences* it contains. However, current gazetteers are overly specific in two respects. First, consistent with (Carroll, 1983), they do not always represent Category Ellipsis and Location Ellipsis. To mimic these processes, we judiciously apply a skip-gram method to token sequences in the gazetteers, thereby including, for example, "Balalok School" as a variant of the complete name (which is a special case of Category Ellipsis where we retain only the last category token and drop the other ones). Second, we eliminate auxiliary or ambiguous gazetteer content (e.g., "George, Washington") that would otherwise threaten recall.

Here we answer the research question: ***Can we accurately and rapidly spot location mentions in text solely relying on a statistical language model synthesized from augmented and filtered region-specific gazetteers?*** Although LNEx works on a targeted Twitter stream collected using event-specific keywords, it does not rely on rarely available tweets geo-coordinates and it does not need any supervision (i.e., training data). It is well-suited for stream processing, and needs only freely available data. Our contributions include:

1. A method for preparing high-quality gazetteers from online open data, such as OSM, Geonames, and DBpedia, and deriving a language model from them; and a comprehensive analysis of the contribution of gazetteer quality to overall performance.

2. A referent corpus representing the full scope of location name extraction challenges and a challenge-based categorization of place names found in the corpora resulting from targeted streams. We annotate three different Twitter streams from flooding events in three different locations: the 2015 Chennai flood, the 2016 Louisiana flood, and the 2016 Houston flood, for our own evaluation and also for use by others.

3. A demonstration that LNEx convincingly outperforms commercial-grade NER and Twitter-specific tools with at least a 33% improvement on average F-Score. Examples reveal the true challenges of location name extraction and the locus of tool failure in the face of these challenges.

LNEx provides the foundation for localizing information, and with increased availability of open data, we expect our approach based on region-specific knowledge to be widely applicable in practice.

2 The LNEx Method

We discuss the details of LNEx in four subsections. First, we present the general idea of statistical inference via n-gram models; the core of LNEx is a statistical language model consisting of a probability distribution over sequences of words (collocations) that represent location names in preexisting, region-specific gazetteers. Then we separately discuss several modifications to both gazetteers and text samples, including gazetteer augmentation and filtering, and tweet preprocessing. Finally, we illustrate the full location analysis and matching process that reliably spots location names.

2.1 Statistical Inference via n-gram Models

LNEx constructs an n-gram model from the collocations that exist *in the gazetteer* to determine the valid location names (LNs) that might appear in tweets. Given tweet content such as "texas ave is closed", the model can then check the validity of one to n-grams. From the gazetteer, "texas" and "ave" are valid gazetteer unigrams but "is" and "closed" are not. Similarly, "texas ave" is a valid and preferred bigram (over two unigrams).

Specifically, as shown in Algorithm 1, we first tokenize all location names in the gazetteer to construct the n-gram model and then save the resulting lists of unigrams, bigrams, and trigrams (Lines 2-5). Next, for bigrams and trigrams, in Lines 6-9 we create conditional frequency distributions (CFD) to count the collocations (i.e., $c(\cdot)$ in equations 1-2). Conditional probability distributions (CPD) are then constructed from the recorded n-grams using maximum likelihood estimation (MLE). We make the assumption that only the previous two words determine the probability of the next word (Markovian assumption of order two)[3]. MLE assumes zero probability values to tokens missing from the gazetteers. *This data sparsity problem is mitigated by augmenting the gazetteers with location name variants* (see Section 2.2). In lines 11-13, we determine the validity of an n-gram (the string s) using the boolean function `VALID-N-GRAM` with the help of equations 1-4, where $c(w_x^y) \equiv c(w_x w_{x+1} \dots w_y)$, w_x^y is the collocation count (i.e., the occurrences of the consecutive words, w_x to w_y), $P(w_z|w_x^y)$ is the conditional probability of a word w_z given previous collocation w_x^y, and the chain of probabilities P_1 (for unigrams), P_2 (for bigrams), and P_3 (for tri or larger grams).

$$P(w_z|w_x^y) = \frac{c(w_x^z)}{c(w_x^y)} \quad (1)$$

$$P_1 = P(w_1^1) = \frac{c(w_1)}{\sum_{i=1}^{|\text{unigrams}|} c(w_i)} \quad (2)$$

$$P_2 = P(w_1^2) = P_1 \times p(w_2 \mid w_1^1) \quad (3)$$

$$P(w_1^n) = P_2 \times \prod_{i=3}^{n} P(w_i \mid w_{i-2}^{i-1}), n \geq 3 \quad (4)$$

Algorithm 1 Language Model Generation

```
1:  procedure COMPUTE-MODEL(Gazetteer)
2:      for ln ∈ Gazetteer do
3:          unigrams ← tokenize(ln);
4:          bigrams, trigrams ← generate from unigrams;
5:      end for
6:      for n-grams ∈ {bigrams} ∪ {trigrams} do
7:          CFD ← create CFD using n-grams;
8:          CPD ← create CPD using CFD;
9:      end for
10: end procedure

11: procedure VALID-N-GRAM(string = s): boolean
12:     w_1^n = (w_1, ..., w_n) ← tokenize(s);
        return P(w_1^n) > 0          ▷ calculated using the equations (1-4);
13: end procedure
```

2.2 Gazetteer Augmentation and Filtering

We faced two primary challenges when building our language model using raw gazetteers, which are not adequately explored in (Middleton et al., 2014; Weissenbacher et al., 2015):

1. **Conditional Collocation Contractions**: Some atomic n-gram location names (collocations) cannot be shortened, e.g., "New York". However, contraction does preserve the meaning of some longer names, especially when the first and the last words denote a specific part and a generic part respectively, such as in "Balalok School".

2. **Auxiliary and Spurious Content**: Gazetteer entries may contain extraneous content that can cause location matching to fail. Cleaning such entries can improve matching reliability (see Table 1).

[3] We found in our dataset that roughly 98% of location mentions in tweets have less than three words.

To address these challenges, we took inspiration from Category Ellipsis (for collocation contraction) and Location Ellipsis (for filtering the auxiliary content) as follows:

1. **Skip-grams**: Given a location name $t_1 \dots t_n$, we retain t_1 and t_n while varying $t_2 \dots t_{n-1}$. To avoid adding "City York" as a legitimate variant of the location name "City College of New York", we require t_n to be a location category name (e.g., building, road). Therefore, "Balalok Matriculation Higher Secondary School" generates {`Balalok School, Balalok Secondary School,` ...}. This technique results in a small number of contractions that are either useful collocations or are too random to cause many false positives (Guthrie et al., 2006).

2. **Filtering**: To address bracketed auxiliary content, we compiled a generic list of phrases to remove specific words on a case-by-case basis (e.g., 1-2 in Table 1). The remaining bracketed names are deemed legitimate alternatives (e.g., 3-4 in Table 1). We treat hyphenated location names as Location Ellipsis and split them on the hyphen and add the two splits (e.g., 5 in Table 1). We expect that the majority of these location names represent a partonomy relationship where the hyphen may be read as a "part of" relation between split tokens. We do not add the second token as a variant when it already exists in the gazetteer on its own as location name entity (e.g., "Hammond" in "Pilot - Hammond").

	Content Description	Example Gazetteer Record
1	Descriptive Tags	(Private Road)
2	Life-cycle/Status Tags	Little Rock School (historical)
3	Alternative/Old Names	Scenic Road (Frontage Road)
4	Acronyms	International House of Pancakes (IHOP)
5	Hyphenations	Cars India - Adyar, Pilot - Hammond

Table 1: Extraneous text in raw gazetteers

These two methods augment and filter partial OSM, Geonames, and DBpedia gazetteers sliced from the original sources using a bounding box. Further, we can attach the metadata of the original location name to the generated variants. Moreover, by treating derived names as synonyms for existing names, we avoid creating additional demands on disambiguation or equivalencing. We add the derived, variant location name to the gazetteer as long as it does not collide with an existing location name. Additional filtering of proposed variants is required to prevent false alarms. Similar to the use case in (Weissenbacher et al., 2015; Gelernter and Balaji, 2013), we compiled a list of 11,203 words including 678 inseparable bigrams, such as "Building A", as gazetteer stop words. This list also includes unusual location names (e.g., "Boring" in Maryland and "Why" in Arizona) and proper nouns (e.g., "James" in Mississippi) that could appear as non-location tokens. We then eliminate from all gazetteers the location names that overlap with our gazetteer stop words to reduce false positives.

2.3 Tweet Preprocessing

To complement the gazetteer preprocessing, we also require potentially non-trivial tweet preprocessing. We start by removing the retweet handles, URLs, non-ASCII characters, and all user mentions. Then, we tokenize tweets using TweetMotif's Twokenizer (O'Connor et al., 2010), which treats hashtags, mentions, and emoticons as a single token. We do not tokenize on periods (e.g., "U.S.").

Hashtag Segmentation: In our datasets, on average, around 29% of the hashtags include location names. Excluding hashtags used to crawl the data, around 17% of the unique hashtags contain locations. As the number of locations in hashtags is significant, similar to (Malmasi and Dras, 2015), we adopted a statistical word segmentation algorithm to break hashtags for location spotting (Norvig, 2009).

Spelling Correction: We consider a tweet token as misspelled if it is an out-of-vocabulary token, where the vocabulary is gazetteer words and a large English vocabulary word list[4]. LNEx corrects all misspelled tokens using the Symmetric Delete Spelling Correction algorithm (SymSpell)[5] that is six orders of magnitude faster than Norvig's spelling corrector (Norvig, 2009), which was used by (Gelernter and Zhang, 2013) in their location extraction tool. As we shall see, spelling correction has only a small influence on system accuracy.

[4] `https://github.com/norrissoftware/words3`
[5] `https://github.com/wolfgarbe/symspell`

2.4 Extracting Location Names using LNEx

After the modifications to gazetteers and texts, LNEx extracts locations as illustrated in Fig. 1. In ⓐ, LNEx reads the raw tweet text, preprocesses it (as in Section 2.3) starting with case-folding. After tokenizing the tweet, the hashtag segmenter breaks hashtags into tokens. Later, stop words are used to split a tweet into consecutive word fragments where each tweet split of size n can have zero to n potential location names. We custom build the tweet stop list starting with around 890 words[6] excluding the gazetteer unigrams.

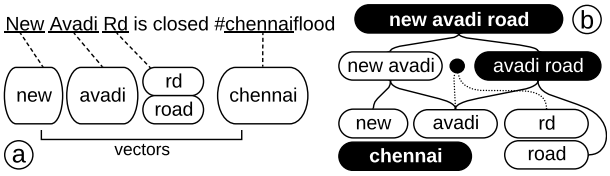


Figure 1: Extracting Locations using LNEx

LNEx now takes each tweet split and converts its tokens into a vector of tokens v using two dictionaries: the USPS street suffixes dictionary[7] and the English OSM abbreviations dictionary[8]. This adds possible expansions and abbreviations of a token (e.g., "Rd" to "Road", and vice versa). This overcomes the lexical variations between location mentions in tweets and their corresponding gazetteer entries.

In ⓑ, the language model is used to find the valid n-grams from the Cartesian product of the consecutive vectors. It builds a bottom up tree for each tweet split starting from 1 to n-grams by gluing the consecutive tokens together if they represent a valid segment in the gazetteer. We improve the speed of the algorithm significantly by splitting the tweet and eliminating invalid n-grams (i.e., n-grams with zero probability values). LNEx then selects a subset of valid n-grams from the tree; for the overlapping n-grams, we prefer the longest full mentions (e.g, "New Avadi Road" over "Avadi Road") and keep both if they are of the same length. When full location names appear inside partial ones, we keep only full names (e.g., extracting "Louisiana" from "The Louisiana").

Time and Space Complexities: LNEx extracts and links a full location mention to its corresponding gazetteer entry through a simple dictionary lookup that takes constant time $\mathcal{O}(1)$. The location extraction time is bounded by the time for creating the bottom up tree of tokens which takes $\mathcal{O}(|v|^s)$ where $|v|$ is the length of the longest vector of token synonyms (i.e., all the expansions and abbreviations of a token) and s is the largest number of tokens with synonyms in a location name. Luckily, splitting the tweet into smaller fragments using stop words significantly lowers the asymptotic growth of the algorithm, thereby enabling stream processing. In practice, for our dictionaries and gazetteers, $|v| \leq 4$ and $s \leq 3$. So, a pessimistic upper bound on the number of candidates for each location (though rarely realized) is 4^3. The space complexity of the method is bounded by the product of number of gazetteer entries, L, and the number of variants of a location name (Skip-gram method 1), that is, 2^{m-2}, where m is the number of tokens in a location name. Effectively, the space complexity is $\mathcal{O}(L.2^{m-2})$ where typically, $2 \leq m \leq 5$. Further, according to our tests, LNEx needed only up to 650 MB of memory to initialize and is able to process, on average, 200 tweets per second.

3 Experimental Results

To demonstrate the effectiveness of our context-aware location extractor, we used a set of event-specific hashtags and keywords to collect 4,500 geographically limited, disaster-related tweets from three different targeted streams corresponding to floods in Chennai, Louisiana, and Houston. Below, we categorize and annotate these tweets used for benchmarking each component of LNEx, and for comparing LNEx with other state-of-the-art tools for the location extraction task.

3.1 Benchmarking and Annotations

Consistent with the problem of determining whether a location mention is inside the area of interest, our benchmark categorization scheme is not content based as in (Matsuda et al., 2015; Gelernter and Balaji, 2013). To better identify and characterize the challenges in extracting location names accurately, our

[6] http://www.ranks.nl/stopwords
[7] http://pe.usps.gov/text/pub28/28apc_002.htm
[8] wiki.openstreetmap.org/wiki/Name_finder:Abbreviations

annotation scheme is based on where these locations lie in relation to the area of interest. For example, with respect to Chicago, IL, USA:

1. in**LOC**: Locations inside the area of interest, (e.g., Millennium Park or Burlington Ave.)
2. out**LOC**: Locations outside the area of interest, (e.g., Central Park, 5th Ave, New York.)
3. amb**LOC**: Ambiguous locations that need context for identification, (e.g., "our house")

In contrast to (Matsuda et al., 2015; Gelernter and Balaji, 2013), our categorization is not based on location types (e.g., buildings, facilities, schools) but on the relative position (i.e., inLOC or outLOC) and the nature of the location mention (i.e., inLOC or ambLOC). This approach identifies the true scope of challenges in extracting location names. Other schemes that annotate for a limited set of location types, such as "Geoparse Twitter Benchmark Dataset" (Middleton et al., 2014), miss obvious location mentions in tweets, such as "New Zealand" and "Christchurch", making the dataset incompatible for testing the tools mentioned in this paper including LNEx.[9]

Tweet Annotations Figure 2 shows an example of manual annotation from the Louisiana flood tweets using the BRAT tool (Stenetorp et al., 2012). It allows us to define search functionalities and additional resources for the annotators to use such as Google Maps.

Figure 2: Example Annotations using BRAT

The annotators annotated three datasets: the 2015 Chennai flood, the 2016 Louisiana flood, and the 2016 Houston flood. In Chennai, they spotted 4,589 location names (75% inLOC, 4% outLOC, and 21% ambLOC); in Louisiana, 2,918 (66% inLOC, 13% outLOC, and 22% ambLOC); and in Houston, 4,177 (66% inLOC, 7% outLOC, and 27% ambLOC). We randomly selected 1,000 tweets (500 each from Chennai and Louisiana) as a development set and the remaining 3,500 as the test set for evaluation.

3.2 Evaluation Strategy

Because BRAT records the start and the end character offsets of the annotated LNs, we evaluate the extraction task by checking the character offsets of the spotted location name in comparison with the annotated data. We used the standard comparison metrics: Precision, Recall, and the balanced F-Score. In the case of overlapping or partial matches, we penalize all tools by adding $\frac{1}{2}FP$ (False Positive) and $\frac{1}{2}FN$ (False Negative) to the precision and recall equations (e.g., if the tool spots "The Louisiana" instead of "Louisiana").

We evaluate all tools based on the category of the extracted location in our annotation scheme. For the inLOC mentions, we count all hits and misses of a tool and ignore all hits when the category of the extracted location is outLOC or ambLOC. However, we take a particularly conservative approach and additionally penalize LNEx for extracting location names of outLOC and ambLOC categories, counting them as false positives (FPs) as our tool is not supposed to extract these.

Spell Checking: This led to 1% increase in recall but the F-Score decreased by 2% on average due to the influence of increased false positives on precision. In the final system, we excluded the spelling corrector component.

Hashtag Breaking: We evaluated the performance of the hashtag breaking component only on the hashtags that contain locations. The accuracies were 97%, 87%, and 93% for Chennai, Louisiana, and Houston respectively, reduced due to examples such as "#lawx", which was broken into "law" and "x".

Picking a Gazetteer: The augmentation and filtering of gazetteers improved the F-Scores (see Figure 3-a). After this process, combinations of gazetteer sources had similar performance where the difference between the worst and the best was around 0.02 F-Score units (see Figure 3-b). In the final system, we relied on OSM, which performed the best. Moreover, DBpedia is not focused on geographical information; therefore, it does not contain the metadata useful for the system's future use (e.g., extents and full addresses). Also, OSM has more fine-grained locations and more accurate geo-coordinates than Geonames (Gelernter et al., 2013).

[9]Our dataset can be used to test any location extraction tool by ignoring the optional additional expressivity, which makes our dataset more compatible than other available ones.

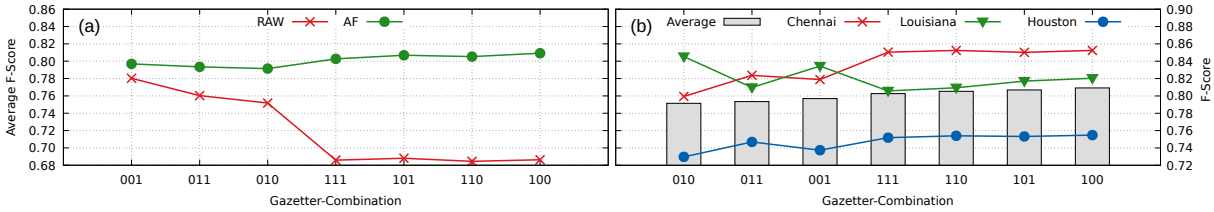


Figure 3: (a) Raw vs. augmented and filtered (AF) gazetteers combinations. (b) Combinations Performance. Each of the seven combinations is a subset of {OSM Geonames DBpedia}.

Google NLP	Location, Organization
OpenCalais	City, Company, Continent, Country, Facility, Organization, ProvinceOrState, Region, TVStation
DBpedia Spotlight	Place, Organization
OSU TwitterNLP	Geo-Location, Company, Facility
TwitIE-Gate	Location, Organization

Table 2: Types considered as Locations per tool

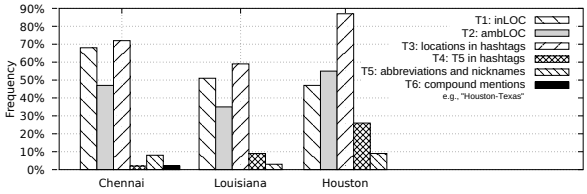


Figure 4: Random Sample Evaluation.

Comparing LNEx with other tools: We compared LNEx with the following tools:

1. **Commercial Grade**: Google NL[10], OpenCalais[11], and Yahoo! BOSS PlaceFinder[12]. All of these tools have REST APIs and are black box tools that use Machine Learning.
2. **General Purpose NER**: Stanford NER and OpenNLP Name Finder. Stanford NER learns a linear chain Conditional Random Field (CRF) sequence model (Finkel et al., 2005), while OpenNLP uses the maximum entropy (ME) framework (Bender et al., 2003). We trained both tools interchangeably on our annotated datasets in addition to all the data from W-NUT '16[13] while retaining the annotations and unifying the classes we consider as locations (i.e., geo-loc, company, facility) into one type. We used LNEx-OSM gazetteer's features while training Stanford NER. Additionally, we used DBpedia Spotlight (Mendes et al., 2011).
3. **Twitter NLP**: OSU Twitter NLP (Ritter et al., 2011) and TwitIE-Gate (Bontcheva et al., 2013). Both are pipelined systems of POS-tagging followed by NER. TwitIE-GATE also supports normalization, gazetteer lookup, and regular expression-based tagging. For fair comparison, we also augmented them with LNEx OSM gazetteers.
4. **Twitter Location Extraction**: Geolocator 3.0 (Gelernter and Zhang, 2013) and Geoparsepy (Middleton et al., 2014). Geolocator 3.0 uses a tweet-trained CRF classifier and other rule-based models to extract street names, building names, business names, and unnamed locations (i.e., location names containing a category such as "School").

All tools have been evaluated using the same metrics and on the same annotated data. In the case of hashtags, we count all hits for all tools and when a tool missed, we penalized only the ones that were designed to break hashtags (namely, TwitIE-Gate and LNEx). Additionally, we consider all spotted mentions from PlaceFinder, Geolocator 3.0, Geoparsepy, Stanford NER, and OpenNLP as location names but, for other tools, we consider only the entity types in Table 2 as location entities.

Fig. 4 shows that the prevalence of various challenges differ in the three corpora. Neverthe-

	Chennai			Louisiana			Houston			AVG
	P	R	F	P	R	F	P	R	F	F
Google NLP	0.40	0.49	0.44	0.55	0.75	0.64	0.39	0.51	0.44	0.51
OpenCalais	0.43	0.10	0.17	0.81	0.77	0.78	0.62	0.35	0.45	0.47
DBpedia Spotlight	0.31	0.44	0.36	0.57	**0.88**	0.70	0.35	0.53	0.42	0.50
Yahoo! PLaceFinder	0.67	0.39	0.49	0.83	0.80	0.81	0.64	0.42	0.50	0.61
Stanford NER	0.72	0.29	0.41	0.78	0.42	0.55	0.74	0.32	0.45	0.47
OpenNLP	0.55	0.15	0.24	0.62	0.19	0.29	0.60	0.23	0.34	0.29
OSU TwitterNLP	0.74	0.40	0.52	**0.84**	0.69	0.76	0.66	0.39	0.49	0.59
TwitIE-Gate	0.51	0.36	0.43	0.66	0.84	0.74	0.35	0.39	0.37	0.52
Geolocator 3.0	0.43	0.54	0.48	0.32	0.71	0.44	0.38	0.58	0.46	0.46
Geoparsepy	0.41	0.28	0.33	0.45	0.72	0.55	0.44	0.46	0.45	0.45
LNEx-RawGaz	0.80	0.78	0.79	0.51	0.80	0.62	0.63	0.66	0.64	0.69
LNEx-AFGaz	**0.91**	**0.80**	**0.85**	0.83	0.81	**0.82**	**0.87**	**0.67**	**0.76**	**0.81**

Table 3: Tools vs. LNEx with a raw (RawGaz) or augmented and filtered gazetteer (AFGaz).

[10] https://cloud.google.com/natural-language/
[11] http://www.opencalais.com/
[12] https://developer.yahoo.com/boss/geo/
[13] http://noisy-text.github.io/2016

less, LNEx outperformed all other tools on all datasets in terms of F-Score, and the average F-Score (see Table 3). LNEx showed stability on the test and development sets from Louisiana with only a 0.2% F-Score reduction and around a 2.6% reduction on the test set from Chennai.

The augmentation and filtering method significantly improved the average F-Score from 0.69 to 0.81 (See Table 3). However, limitations of the gazetteer augmentation and filtering methods did contribute to lowering precision. For example, on average, around 5% of the extracted location names were outLOC and ambLOC, mistakenly extracted from Chennai, Louisiana, and Houston tweets. Example errors include the augmentation of location names such as "The x Apartments" to "The Apartments", causing LNEx to extract the phrase "The Apartments" as an actual full location name. Fixing such limitations should contribute to around 2% F-Score improvement on average.

We trained Stanford NER and OpenNLP to emulate their use in other studies mentioned in Section 4. Their performances were calculated by interchangeably training them using three datasets at a time and testing on the fourth one (the gazetteer of the area of the test data was also used in training the Stanford NER models). We always used the W-NUT '16 dataset to train the models with more than 10,000 tweets each time.

We observed that the ill-formatted text of tweets with ungrammatical text and missing orthographic features impact the F-Score of tools we compared with LNEx. While the performance of each tool differs, we observed that Google heavily relies on orthographic features and expects grammatical texts (even though it scored a 0.38 average F-Score). Additionally, TwitIE-GATE was not always successful in extracting location names from hashtags or text even if these location names were part of the tool's gazetteers. Finally, OpenCalais extracts only well-known location names of coarser granularity than street and building levels unless a location has an attached location category (e.g., school or street).

Original Text	sou th kr koil street near Oxford school.west mambalam..
	We r lucky where I am in New Iberia. #PrayForLouisiana #lawx
	Didn't Houston have a bad flood last year now again poor htown
Manual Annotations & Types	misspelling T6 (**sou th** **kr** **koil street**) near (**Oxford school**).(**west mambalam**).. T6
	T1 T3 T4 We r lucky where I am in (**New Iberia**). #PrayFor(**Louisiana**) #(**la**)wx T5
	Didn't **(Houston)** have a bad flood last year now again poor **(htown)** T1 T5
Google NLP	sou th kr (**koil street**) near (**Oxford**) school.west (**mambalam**)..
	We r lucky where I am in (**New Iberia**). #PrayForLouisiana #lawx
	Didn't (**Houston**) have a bad flood last year now again poor htown
OpenCalais	sou th kr koil street near (**Oxford school**).west mambalam..
	We r lucky where I am in New Iberia. #PrayForLouisiana #lawx
	Didn't (**Houston**) have a bad flood last year now again poor htown
DBpedia Spotlight	sou th kr koil street near (**Oxford**) school.west (**mambalam**)..
	We r lucky where I am in (**New Iberia**). #PrayForLouisiana #lawx
	Didn't (**Houston**) have a bad flood last year now again poor htown
Yahoo! PlaceFinder	sou (**th**) kr koil street near (**Oxford**) school.west mambalam..
	We r lucky where I am in (**New Iberia**). #PrayForLouisiana #lawx
	Didn't Houston have a bad flood last year now again poor htown
Stanford NER	sou th kr koil street near Oxford school.west mambalam..
	We r lucky where I am in (**New Iberia**). #PrayForLouisiana #lawx
	Didn't Houston have a bad flood last year now again poor htown
OpenNLP	sou th kr (**koil street**) near (**Oxford**) school.west mambalam..
	We r lucky where I am in (**New Iberia.**) #PrayForLouisiana #lawx
	Didn't Houston have a bad flood last year now again poor htown
OSU TwitterNLP	sou th kr koil street near (**Oxford**) school.west mambalam..
	We r lucky where I am in (**New Iberia**). #PrayForLouisiana #lawx
	Didn't Houston have a bad flood last year now again poor htown
TwitIE-Gate	sou th kr koil (**street**) near (**Oxford**) school.(**west mambalam**)..
	We r lucky where I am in New Iberia. #PrayForLouisiana #lawx
	Didn't Houston have a bad flood last year now again poor htown
Geolocator 3.0	(**sou th**) (**kr**) (**koil**) street near (**Oxford school**).(**west mambalam**)..
	We r lucky where I am in (**New Iberia**). #PrayForLouisiana #lawx
	Didn't (**Houston**) have a bad flood last year now again poor (**htown**)
Geoparsepy	sou (**th**) (**kr**) koil street near (**Oxford**) school.west mambalam..
	We r lucky where I am in (**New** (**Iberia**)). #PrayForLouisiana #lawx
	Didn't (**Houston**) have a bad flood last year now again poor htown
LNEx	sou th kr koil street near (**Oxford school**).(**west mambalam**)..
	We r lucky where I am in (**New Iberia**). #PrayFor(**Louisiana**) #lawx
	Didn't (**Houston**) have a bad flood last year now again poor htown

Table 4: Example tool outputs: bracketed bold text are the identified LNs and braces highlights the types from Fig. 4.

Illustrative Examples: Table 4 shows the comparative handling of three tweets one each from Chennai, Louisiana, and Houston datasets, covering most challenging cases for all the tools. The location name "Oxford school" allowed us to examine if a tool relies on capitalization for delimitation. Only OpenCalais, Geolocator and LNEx were able to extract the name correctly while the rest either partially extracted it or missed it. For example, PlaceFinder extracted "Oxford" and geocoded it with the geocodes of Oxford city in England. Although Stanford NER and OpenNLP were trained on the same datasets, OpenNLP extracted Oxford while Stanford NER did not, which suggests that the cue word "near" was insufficient evidence for Stanford NER to spot at least Oxford. Correspondingly, since "New Iberia" is a correctly capitalized full location name, almost all tools were able to

extract it. However, TwitIE-Gate missed it although it is part of the gazetteer we added to the tool, and Geoparsepy extracted Iberia in addition to the full mention, not favoring the longest mention as LNEx. OpenCalais is a black box so we don't know why it failed.

Regarding T3-T5 annotations, LNEx and TwitIE-Gate are designed to break hashtags but TwitIE-Gate was not able to extract any locations from the hashtags in the table. LNEx extracted "Louisiana" but was not able to extract "la" from "#lawx" due to the statistical method which broke the hashtag into "law" and "x" since this combination is more probable. Only Geolocator was able to extract the Houston nickname "htown". In the future, a dictionary of region-specific acronyms, abbreviations, and nicknames can augment LNEx's region-specific gazetteers.

Google NLP does not handle T6. Adding space between the dot and "west" to create "... school. west ...", results in the extraction of "west mambalam" but omits "Oxford school". Google NLP relies on capitalization, so changing the case of "s" to create "Oxford School" does help. OpenCalais cannot extract "west mambalam" despite fixing all grammatical mistakes, normalizing the orthographic features, and even introducing cue words. The tool only extracts well-known location names of coarser granularity than street and building levels unless they have an attached location category (e.g., school or street). PlaceFinder, on the other hand, tries to find geocodable location names in text. Therefore, the tool extracts "th" as the country code of Thailand and "Oxford" as the city in England. Hence, geocoding is influencing some of the mistakes of the tool.

4 Related Work

Twitter messages (tweets) lack features exploited by main stream NLP tools. Informality, ill-formed words, irregular syntax and non-standard orthographic features of tweets challenge such tools (Kaufmann and Kalita, 2010). We agree with (Baldwin et al., 2013) that some issues might be exaggerated. Indeed we found that spelling corrections only contributed to 1% recall improvement. Nevertheless, text normalization alone is insufficient for NER (Derczynski et al., 2015). Specially designed tools such as (Ritter et al., 2011; Gelernter and Zhang, 2013) use pipelined systems of POS tagging followed by NER. The latter also perform Regex tagging, normalization, and gazetteer lookup.

Relying on the orthographic features for POS tagging or Regex tagging, previous methods extract locations from the text chunks and phrases of sentences using the following techniques:

1. **Gazetteer search or n-gram matching**: Li et al. (2014) and Gelernter and Zhang (2013) use a gazetteer matching technique that relies on a segment-based inverted index. Sultanik and Fink (2012) use an exhaustive n-gram technique. Middleton et al. (2014) use location-specific gazetteers for matching phrases from tweets. TwitIE-GATE uses a gazetteer lookup component. All of these techniques do not deal with the important issue of the gazetteers' auxiliary content and noise.

2. **Handcrafted rules**: Weissenbacher et al. (2015) and Malmasi and Dras (2015) use pattern and Regex matching which rely on cue words or orthographic features for POS-tagging. TwitIE-GATE adapts rules from ANNIE (Cunningham et al., 2002) for extraction.

3. **Supervised Methods**: *Tweet-trained models*: The majority of the methods trained Stanford NER on tweets (Gelernter and Zhang, 2013; Yin et al., 2014) or retrained OpenNLP (Lingad et al., 2013). *News-trained models*: Malmasi and Dras (2015) use tools like Stanford NER and OpenNLP.

4. **Semi-supervised methods**: Ji et al. (2016) use beam search and structured perceptron for extraction and linking to Foursquare entities. However, they did not address the noise that is prevalent in such sources (e.g., "my sofa" or "our house") (Dalvi et al., 2014).

The closest works to ours are TwiNER (Li et al., 2012) and LEX (Downey et al., 2007). Both use Microsoft Web n-grams (which capture language statistics) for chunking but the former uses DBpedia for entity linking. However, our method exploits a region-specific gazetteer for delimitation and linking. Moreover, LEX worked with web data and relies heavily on capitalization.

Finally, few other methods extract locations from hashtags. Malmasi and Dras (2015) uses a statistical hashtag breaker technique similar to ours. Ji et al. (2016) removes only the # symbol and treats the hashtag as a unigram. Yin et al. (2014) uses a greedy maximal matching method for breaking. TwitIE-GATE

uses two methods for hashtag breaking: a dynamic programming-based method for finding subsequences and a camel-case-based method for tokenization.

5 Conclusions and Future Work

LNEx accurately spots locations in text relying solely on statistical language models synthesized from augmented and filtered region-specific gazetteers. It outperforms state-of-the-art techniques and mainstream location name extractors. By exploiting the knowledge in the gazetteer, we extract (delimit) location mentions and then materialize them as location metadata for future/further processing. LNEx does not employ any training and does not depend on syntactic analysis or orthographic conventions. We compensate for limitations in fixed phrase matching with gazetteer augmentation and filtering. Although we do not solve the disambiguation problem here, still the geo/geo ambiguity is reduced by preserving the spatial context through location-specific gazetteers. Furthermore, systematic gazetteer augmentation ties legitimate variants to known locations, minimizing potential ambiguity.

Certainly, LNEx does not solve all location extraction problems. It actually presents an effective precision-recall trade-off apparent in the F-Score. But as the method is driven by the linking procedure, it does not extract location names missing from gazetteers (e.g., "our house"). However, in the future, we might consider extracting them at a reduced cost by integrating LNEx with an incremental learning method (Al-Olimat et al., 2018). Additionally, a more sophisticated name model that ignores the generic parts and retains the specific parts when augmenting a location name (e.g., adding "Sam's" as a variant of "Sam's Club") can be used (Dalvi et al., 2014).

Acknowledgments

This research was partially supported by the NSF award EAR-1520870 "Hazards SEES: Social and Physical Sensing Enabled Decision Support for Disaster Management and Response". We would like to thank Jibril Ikharo for introducing us to the Nameheads work and our other colleagues from Kno.e.sis for helping us in data annotation.

References

Hussein S. Al-Olimat, Steven Gustafson, Jason Mackay, Krishnaprasad Thirunarayan, and Amit Sheth. 2018. A practical incremental learning framework for sparse entity extraction. In *The 27th International Conference on Computational Linguistics (COLING 2018)*.

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. 2013. How noisy social media text, how diffrnt social media sources? In *International Joint Conference on Natural Language Processing*, pages 356–364.

Oliver Bender, Franz Josef Och, and Hermann Ney. 2003. Maximum entropy models for named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 148–151, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kalina Bontcheva, Leon Derczynski, Adam Funk, Mark A Greenwood, Diana Maynard, and Niraj Aswani. 2013. Twitie: An open-source information extraction pipeline for microblog text. In *RANLP*, pages 83–90.

John M Carroll. 1983. Nameheads. *Cognitive science*, 7(2):121–153.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. Gate: an architecture for development of robust hlt applications. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 168–175. Association for Computational Linguistics.

Nilesh Dalvi, Marian Olteanu, Manish Raghavan, and Philip Bohannon. 2014. Deduplicating a places database. In *Proceedings of the 23rd international conference on World wide web*, pages 409–418. Association for Computing Machinery (ACM).

Leon Derczynski, Diana Maynard, Giuseppe Rizzo, Marieke van Erp, Genevieve Gorrell, Raphaël Troncy, Johann Petrak, and Kalina Bontcheva. 2015. Analysis of named entity recognition and linking for tweets. *Information Processing & Management*, 51(2):32–49.

Doug Downey, Matthew Broadhead, and Oren Etzioni. 2007. Locating complex named entities in web text. In *IJCAI*, volume 7, pages 2733–2739.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA. Association for Computational Linguistics.

Judith Gelernter and Shilpa Balaji. 2013. An algorithm for local geoparsing of microtext. *GeoInformatica*, 17(4):635–667.

Judith Gelernter and Wei Zhang. 2013. Cross-lingual geo-parsing for non-structured data. In *Proceedings of the 7th Workshop on Geographic Information Retrieval*, pages 64–71. Association for Computing Machinery (ACM).

Judith Gelernter, Gautam Ganesh, Hamsini Krishnakumar, and Wei Zhang. 2013. Automatic gazetteer enrichment with user-geocoded data. In *Proceedings of the Second ACM SIGSPATIAL International Workshop on Crowdsourced and Volunteered Geographic Information*, pages 87–94. Association for Computing Machinery (ACM).

David Guthrie, Ben Allison, Wei Liu, Louise Guthrie, and Yorick Wilks. 2006. A closer look at skip-gram modelling. In *Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC-2006)*, pages 1–4.

Mike Hazas, James Scott, and John Krumm. 2004. Location-aware computing comes of age. *Computer*, 37(2):95–97.

Thi Bich Ngoc Hoang and Josiane Mothe. 2018. Location extraction from tweets. *Information Processing & Management*, 54(2):129–144.

Zongcheng Ji, Aixin Sun, Gao Cong, and Jialong Han. 2016. Joint recognition and linking of fine-grained locations from tweets. In *Proceedings of the 25th International Conference on World Wide Web*, pages 1271–1281. International World Wide Web Conferences Steering Committee.

Max Kaufmann and Jugal Kalita. 2010. Syntactic normalization of twitter messages. In *International conference on natural language processing, Kharagpur, India*.

Chenliang Li, Jianshu Weng, Qi He, Yuxia Yao, Anwitaman Datta, Aixin Sun, and Bu-Sung Lee. 2012. Twiner: named entity recognition in targeted twitter stream. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 721–730. Association for Computing Machinery (ACM).

Guoliang Li, Jun Hu, Jianhua Feng, and Kian-lee Tan. 2014. Effective location identification from microblogs. In *Data Engineering (ICDE), 2014 The Institute of Electrical and Electronics Engineers (IEEE) 30th International Conference on*, pages 880–891. The Institute of Electrical and Electronics Engineers (IEEE).

Yehoshua Zvi Licht, David Allen Turner, and Joseph Arnold White. 2017. Location context aware computing, November 30. US Patent App. 15/166,740.

John Lingad, Sarvnaz Karimi, and Jie Yin. 2013. Location extraction from disaster-related microblogs. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1017–1020. Association for Computing Machinery (ACM).

Fei Liu, Maria Vasardani, and Timothy Baldwin. 2014. Automatic identification of locative expressions from social media text: A comparative analysis. In *Proceedings of the 4th International Workshop on Location and the Web*, pages 9–16. Association for Computing Machinery (ACM).

Shervin Malmasi and Mark Dras. 2015. Location mention detection in tweets and microblogs. In *International Conference of the Pacific Association for Computational Linguistics (PACL)*, pages 123–134. Springer.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA.

Koji Matsuda, Akira Sasaki, Naoaki Okazaki, and Kentaro Inui. 2015. Annotating geographical entities on microblog text. In *The 9th Linguistic Annotation Workshop held in conjuncion with North American Chapter of the Association for Computational Linguistics (NAACL) 2015*, page 85.

Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8. ACM.

Stuart E Middleton, Lee Middleton, and Stefano Modafferi. 2014. Real-time crisis mapping of natural disasters using social media. *The Institute of Electrical and Electronics Engineers (IEEE) Intelligent Systems*, 29(2):9–17.

Robert Munro. 2011. Subword and spatiotemporal models for identifying actionable information in haitian kreyol. In *Proceedings of the fifteenth conference on computational natural language learning*, pages 68–77. Association for Computational Linguistics (ACL).

Peter Norvig. 2009. Natural language corpus data. In T. Segaran and J. Hammerbacher, editors, *Beautiful Data*, chapter 14, pages 219–242. O'Reilly Media.

Brendan O'Connor, Michel Krieger, and David Ahn. 2010. Tweetmotif: Exploratory search and topic summarization for twitter. In *ICWSM*, pages 384–385.

Jakub Piskorski and Maud Ehrmann. 2013. On named entity recognition in targeted twitter streams in polish. In *The 4th Biennial International Workshop on Balto-Slavic Natural Language Processing: ACL*, pages 84–93. Citeseer.

L.B. Resnick, J.M. Levine, and S.D. Teasley. 1991. *Perspectives on Socially Shared Cognition*. American Psychological Association.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics (ACL).

Jeongwook Son, Zeeshan Aziz, and Feniosky Pena-Mora. 2008. Supporting disaster response and recovery through improved situation awareness. *Structural survey*, 26(5):411–425.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. BRAT: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations Session at EACL 2012*, Avignon, France, April. Association for Computational Linguistics (ACL).

Evan A Sultanik and Clayton Fink. 2012. Rapid geotagging and disambiguation of social media text via an indexed gazetteer. *Proceedings of International conference on Information Systems for Crisis Response and Management (ISCRAM)*, 12:1–10.

Davy Weissenbacher, Tasnia Tahsin, Rachel Beard, Mari Figaro, Robert Rivera, Matthew Scotch, and Graciela Gonzalez. 2015. Knowledge-driven geospatial location resolution for phylogeographic models of virus migration. *Bioinformatics*, 31(12):i348–i356.

Jie Yin, Sarvnaz Karimi, and John Lingad. 2014. Pinpointing locational focus in microblogs. In *Proceedings of the 2014 Australasian Document Computing Symposium*, page 66. Association for Computing Machinery (ACM).

The APVA-TURBO Approach To Question Answering in Knowledge Base

Yue Wang[†], Richong Zhang[†*], Cheng Xu[†] and Yongyi Mao[‡]
[†]BDBC and SKLSDE, School of Computer Science and Engineering, Beihang University, China
[‡]School of Electrical Engineering and Computer Science, University of Ottawa, Canada
[†]{wangyue16, zhangrc, xucheng}@act.buaa.edu.cn
[‡]yymao@site.uottawa.ca

Abstract

In this paper, we study the problem of question answering over knowledge base. We identify that the primary bottleneck in this problem is the difficulty in accurately predicting the relations connecting the subject entity to the object entities. We advocate a new model architecture, APVA, which includes a verification mechanism responsible for checking the correctness of predicted relations. The APVA framework naturally supports a well-principled iterative training procedure, which we call turbo training. We demonstrate via experiments that the APVA-TUBRO approach drastically improves the question answering performance.

Title and Abstract in Chinese

面向知识库问答的 APVA-TURBO 方法

本文主要围绕目前基于知识库的问答方法中的存在的问题展开研究。经过对现有各类问答方法的调查与分析，我们发现目前知识库问答的瓶颈主要体现在关系预测上，即如何准确地预测出问题中的实体与答案实体之间的关联，是需要面临的最大挑战。因此本文在现有问答框架的基础上，加入了负责对预测关系的可靠性进行评价检验的验证机制，用以增强关系预测效果，并提出了一种新的知识库问答框架“APVA”。另外，文中为 APVA 设计了一种具有良好理论基础的迭代训练过程，我们称之为“涡轮式(turbo)训练”。通过实验证明，APVA-TUBRO 方法可以在问答数据集上取得优异效果，大大提升了目前问答方法的准确性。

1 Introduction

Knowledge bases (KBs), such as YAGO, DBPedia, and Freebase contain an enormous volume of facts, or knowledge, about the real world. Each of these facts is represented as a triple (s, r, o), where s is a *subject entity*, r is a *relation*, and o is an *object entity*. This representation allows the KB to be interpreted as a graph in which entities are vertices, triples are edges, and relations are edge labels. Such a structured representation has enabled the development of query languages, such as RQL(Karvounarakis et al., 2002), Versa(Ogbuji, 2005) and SPARQL(Prud et al., 2006), for retrieving knowledge stored in the KB. Despite the usefulness of these languages, it is highly desirable that the query-processing engine of a KB is capable of processing queries presented in human language directly. This stimulates intense research in question answering over KB.

Briefly, question answering over KB, or QA-KB, aims at taking as input a question presented in a natural language and outputs the answer in terms of a fact triple or a collection of fact triples in the KB. In a general setting, the answer is a set of *paths*, where each path connects the subject entity of the question to the object entity via a sequence of relations. We refer to this relation sequence as a *path label*.

There are two main lines of solutions to QA-KB. The first centres around semantic parsing, which translates the natural-language question into a logical form and uses it to query the KB. This line of

*Corresponding Author: Richong Zhang

Proceedings of the 27th International Conference on Computational Linguistics, pages 1998–2009
Santa Fe, New Mexico, USA, August 20-26, 2018.

solutions includes, for example, (Berant et al., 2013; Dong et al., 2017; Yih et al., 2015; Xu et al., 2016; Reddy et al., 2017).

The second line of solutions centres around learning with neural network models, which follows what we call an APA architecture or its variants. Specifically, the APA architecture includes three components: an *entity alignment* component, which identifies the subject entity implied in the question, a *path label prediction* component, which predicts a path label, and an *object answering* component, which finds the object entitie(s). This line of solutions includes, for example, (Dong et al., 2015; Hsiao et al., 2017; Yin et al., 2016; Yu et al., 2017; Lukovnikov et al., 2017; Jain, 2016; Hao et al., 2017; Bordes et al., 2014).

Despite the success of the existing approaches to QA-KB, particularly that of the APA architecture, we observe that the primary bottleneck for further improving the model performance lies in the model's limited accuracy in label path prediction. Motivated by this observation, this work extends the APA methodology into an "APVA" framework, which incorporates a verification mechanism in the model. Such a mechanism is responsible for checking if the predicted relation or path label is correct and hence improving the answering performance. Unifying APVA's components in a probabilistic modelling set-up, we show that such an APVA architecture allows the joint training of its model components. More specifically, the joint training can exploit coordinate descent in the optimization of the loss function. This naturally gives rise to an iterative training procedure, which we call turbo training. We develop a concrete model and its training algorithm based on this APVA-TURBO approach. Experimental study demonstrates that the proposed APVA-TURBO model outperforms the existing models by a large margin, thereby establishing itself as a new state of the art in QA-KB.

2 Problem Statement

For a given KB $\mathcal{K}$, we will denote its set of entities by $\mathcal{E}$ and its set of relations by $\mathcal{R}$. The KB $\mathcal{K}$ contains a set of factual triples $\mathcal{F}$ in which each triple is in the form of (s, r, o), with $s, o \in \mathcal{E}$ and $r \in \mathcal{R}$. For example, (CarlosGomez, baseball.baseballPlayer.position, CenterFielder) is one such triple taken from Freebase. It specifies the fact that Carlos Gomez plays the Center Fielder position in baseball. It is customary to interpret the triple set $\mathcal{F}$ in $\mathcal{K}$ as an edge-labelled graph in which vertices are entities in $\mathcal{K}$, edges are triples in $\mathcal{F}$, and on each edge (s, r, o), r is the edge label.

Additionally the KB $\mathcal{K}$ usually also prescribes a set $\mathcal{T}$ of entity types. Specifically, for each entity $e \in \mathcal{E}$, its type $t(e)$ is a token in $\mathcal{T}$. Note that in KBs like Freebase, an entity e may be associated with multiple types. In this work, for simplicity, we take $t(e)$ as the *primary type* of e (which is referred to as the "notable type" of e in Freebase).

We will denote by q the question presented in natural language. Specifically q is a sequence $(w_1, w_2, \ldots, w_n)$ of words, where each w_i is a word in a vocabulary $\mathcal{V}$, and n depends on q. The objective of the *question answering over KB*, or simply QA-KB, is then to develop an answering algorithm which returns the correct answer triple (s, r, o) for any given question q. For example, when a user asks a question "What position does Carlos Gomez play", the answering algorithm is expected to retrieve the triple (CarlosGomez, baseball.baseballPlayer.position, CenterFielder) from the KB.

The above form of QA-KB is in fact in its most basic form. The problem can be made more general in at least the following two ways. First, the answer to q may be multiple triples which form a path in the KB graph. For example, the answer could involve two triples (a, r_1, b) and (b, r_2, c) concatenated in tandem. This corresponds to the path (a, r_1, b, r_2, c) in the KB graph. We will call such an answer a *multi-hop answer*, and call the pair (r_1, r_2) of edge labels the *path label*. Furthermore, there can be multiple answers for a given question q. For example, if the question is "what are the provinces of Canada", then more than one answer is expected.

This paper in fact deals with a general setting of QA-KB, namely, a question may have *multiple multi-hop* answers. We do impose a restriction that we only consider the case in which all answers of any given question have *the same* path label. We note that this simplification does not imply that our proposed framework and model do not extend further. It is solely due to that this restriction sufficiently well describes the datasets we work with.

Denote by $\mathcal{R}^*$ the set of all finite-length sequences consisting of relations in $\mathcal{R}$. We still denote an

answer, multi-hop or single-hop, as a triple (s, r, o) while considering $r \in \mathcal{R}^*$. For any given question q, let $\mathcal{A}(q)$ be the set of all answers, for q, where each answer is in the form of (s, r, o) with $r \in \mathcal{R}^*$.

A machine learning model for QA-KB relies on a training set $\mathcal{D}$ consisting of many pairs $(q, \mathcal{A}(q))$. We will use $\mathcal{D}_{\text{q}}$ to denote the set of all questions in $\mathcal{D}$. After the model is trained on $\mathcal{D}$, one expects it to output the answers $\mathcal{A}(q)$ for an arbitrary question q, unseen in $\mathcal{D}_{\text{q}}$.

3 APVA Modelling Framework

3.1 The Conventional APA Architecture

Conventionally, a QA-KB model, if put into a probabilistic framework, can be regarded as learning a conditional distribution $p(s, r, o|q)$, where for each configuration of (s, r, o, q), the value $p(s, r, o|q)$ is probability that the triple (s, r, o) is an answer in $\mathcal{A}(q)$. Note that in this formulation, we have suppressed the dependency of (s, r, o) on the given KB $\mathcal{K}$.

By the chain rule of probability,

$$p(s, r, o|q) = p(s|q)p(r|q, s)p(o|q, s, r).$$

A QA-KB model factorizing this way essentially assumes three components: an *entity alignment* component $p(s|q)$ to identify the subject entity s in the question q, a *path label prediction* component $p(r|q, s)$ to predict the path label r that leads the subject entity to the object entities, and an *object answering* component to provide the list of object entities o's each forming an answer (s, r, o). We call such a model an *APA* model.

The existing neural network models for QA-KB largely fall in the APA architecture, or its variants, in which instead of having the three components separable, the model may combine some of the components or have parameters shared between them. We now summarize these models.

In (Yin et al., 2016; Lukovnikov et al., 2017), a word/character-level encoder is used for question encoding and to match the subject entity and the relation in a fact candidate with the question. In (Yu et al., 2017; Hsiao et al., 2017) a bi-directional LSTM is used to score and rank the question and relation pairs. In (Ture and Jojic, 2017), only simple RNN is used both for entity alignment and for relation prediction.The approach of (Jain, 2016) extracts candidate triples and finds a path to the object entity using multi-hop reasoning and refinement. In (Dong et al., 2015; Hao et al., 2017), a multi-column CNN or a cross-attention mechanism is introduced to match the question with the answer path, the answer context and the answer type. In (Bordes et al., 2014), questions and their answer subgraphs from KB are both embedded into a lower dimensional to be matched.

3.2 The APVA Framework

Despite the successes with the APA approaches, we observe that the bottleneck in the existing models lies primarily in the accuracy of predicting the path labels. In fact our experiments suggest that, if the correct (s, r) can be obtained, a good object answering accuracy can be easily achieved. For example, on the WebQuestion dataset(Berant et al., 2013), we see that a simple SVM classifier followed by a search of the KB (see section 5.2) achieves an answering precision of 89.68% and a recall of 93.20%.

The fact that path label prediction forms the primary bottleneck can be attributed to the enormous number of entities and the extremely high dimension of the question space. Relative to the space of (q, s), the dataset $\mathcal{D}$ is in fact very sparse. This makes path label prediction prone to error and limits the performance of a QA-KB model.

To increase the accuracy of path label prediction, this paper augments the APA models with an additional component, which we call *subject-path verification*. The basic idea here is to use a verification mechanism to check if a predicted pair (s, r) is correct. We call this architecture the *APVA* architecture.

3.2.1 The APVA Architecture

In APVA framework, for each question q, we assume that there is another variable y, taking values in $\{0, 1\}$ and indicating whether the obtained (s, r) is correct (1 for correct). The overall objective of

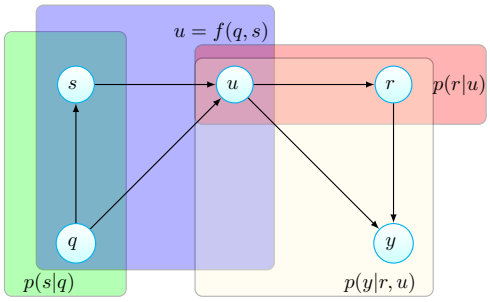


Figure 1: The $p(s, r, y|q)$ part of APVA

learning is then to learn a distribution $p(s, r, o, y|q)$, which factors as

$$p(s, r, o, y|q) = p(s, r, y|q)p(o|q, s, r, y). \tag{1}$$

The term $p(o|q, s, r, y)$ in (1) essentially characterizes the *object answering* component (where we are only interested in the case of $y = 1$). The term $p(s, r, y|q)$ further factors as

$$p(s, r, y|q) = p(s|q)p(r|q, s)p(y|q, s, r) \tag{2}$$

In (2), the terms $p(s|q)$ and $p(r|q, s)$ respectively correspond to an *entity alignment* component and a *path label prediction* component. The term $p(y|q, s, r)$ on the other hand is the new *subject-path verification* component.

For the path label prediction component, APVA further expresses $p(r|q, s)$ by

$$u := f(q, s) \tag{3}$$
$$p(r|q, s) := p(r|u) \tag{4}$$

where f is an appropriate *encoder* (ENC) function, and (4) is referred to as the *decoder* (DEC). Here f serves to compress the high-dimensional representation of (q, s) to a low dimensional space so as to deal with the sparsity of (q, s).

Furthermore APVA expresses $p(y|q, s, r)$ as

$$p(y|q, s, r) := p(y|u, r). \tag{5}$$

Since (5) essentially defines a binary classifier, we call it the *verification classifier* (CLS).

In summary, the APVA framework consists of the following components: entity alignment, path label prediction (ENC-DEC), subject-path verification (CLS) and object answering. Figure 1 shows the overall model architecture of APVA, with the object answering component excluded.

3.2.2 Loss Functions and Training

For each of the components, we can derive an appropriate loss function: the (entity) alignment loss ℓ_{al}, the (path label) prediction loss ℓ_{pr}, the (subject-path) verification loss ℓ_{ve}, and the (object) answering loss ℓ_{an}. It is worth noting that to properly define the verification loss ℓ_{ve}, we need to introduce "negative examples" for CLS.To that end, denote

$$\mathcal{D}_{\text{qsr}} := \{(q, s, r) : q \in \mathcal{D}_{\text{q}}, (s, r, o) \in \mathcal{A}(q)\}$$

That is, $\mathcal{D}_{\text{qsr}}$ is the set of all training examples for path label prediction (ENC-DEC). It is also the set of all positive training examples for the verification classifier CLS.

For each positive example $(q, s, r) \in \mathcal{D}_{\text{qsr}}$, we can create several negative examples by replacing the path label r in (q, s, r) with a random wrong path label r'. This gives rise to a negative training set $\mathcal{D}^{-}_{\text{qsr}}$.

If the four loss functions do not share parameters, then one can separately minimize each loss and learn its parameters. If there is a sharing of parameters among some of these losses , one can take a

joint approach to minimize the (possibly weighted) sum of these losses. A generic approach for this joint minimization is coordinate descent, namely, we hold some parameter fixed and optimize the sum loss over other parameters, and keep iterating this process. We will give a concrete example of this training method in a later section.

Although APVA is derived from probabilistic modelling, its overall methodology extends to non-probabilistic models. Indeed, all we need are those well-defined losses, whether or not they result from probabilistic modelling.

4 The APVA-TURBO Model

We now describe a concrete model in the proposed APVA framework, which we refer to as the APVA-TURBO model. The model can be divided into three models which are trained separately. They are the Entity Alignment Model, the Prediction-Verification Model, and the Object Answering Model. Note that the Prediction-Verification Model is a joint model containing both path label prediction and subject-path verification.

The key innovation in this model architecture is the Prediction-Verification Model, where the Entity Alignment Model is a modest extension of a previous model and the Object Answering Model is a standard application of SVM on a selected choice of feature. To stay focused, details of the Entity Alignment Model and Object Answering Model are presented in section 5.2.

We now present the central piece of APVA-TURBO, the Prediction-Verification Model. It decomposes further into an encoder-decoder (or ENC-DEC) model for *path label prediction* and a classifier model (CLS) for subject-path verification.

4.1 ENC-DEC Model

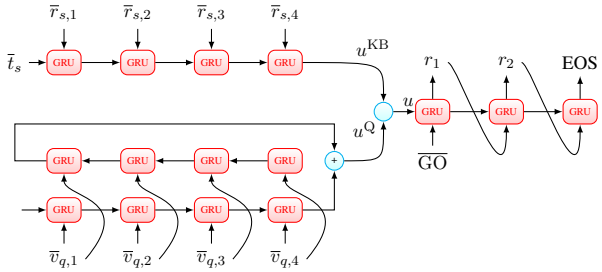


Figure 2: ENC-DEC Model

Recall that in APVA, the ENC component, Equation (3), encodes (q, s) into a vector u and the DEC component, Equation (4), maps u to a desired distribution on the path labels in $\mathcal{R}^*$. The structure of ENC-DEC in APVA-TURBO is shown in Figure 2, which we describe below.

For the identified subject entity s, let $\mathcal{R}(s) := (r_{s,1}, r_{s,2}, \ldots, r_{s,L})$ be the list of all relations in $\mathcal{R}$ that are labels of at least one edge in the KB graph connecting to entity s. Let $\overline{\mathcal{R}}(s) := (\overline{r}_{s,1}, \overline{r}_{s,2}, \ldots \overline{r}_{s,L})$ be the embeddings of $\mathcal{R}(s)$ and $\overline{t}_s$ be the embedding of the type $t(s)$ of s , both of which are obtained from a KB schema embedding model, called KSE. KSE is an separately pre-trained model to embed the relations and types in KB in an Euclidean space, by using an adaptation of the TransH (Wang et al., 2014) to the "schema graph", which is a graphical representation of the KB schema set $\mathcal{S}$ of all "schema triples" $(t, r, t') \in \mathcal{T} \times \mathcal{R} \times \mathcal{T}$ for which knowledge triples (s, r, o) with $t(s) = t$ and $t(o) = t'$ are allowed. We take $\overline{t}_s$ as the initial state of a GRU network, and take $\overline{\mathcal{R}}(s)$ as the input sequence to the network. We denote the final state of the GRU network by u^{KB}. We note that u^{KB} encodes the subject entity s and its relative position in $\mathcal{K}$. We refer to u^{KB} as the *KB-based encoding*. In this encoding step, although order of the elements in $\overline{\mathcal{R}}(s)$ is irrelevant, we still feed them to the GRU network sequentially due to its superiority in processing variable-length input and automatically selecting relevant feature.

On the other hand, for the question $q = (w_1, w_2, \ldots, w_n)$, its subsequence $\mu := (w_m, w_{m+1}, \ldots, w_{m+k})$ of words that corresponds to the subject entity s in the KB $\mathcal{K}$ is referred to

as the *subject mention*. we remove the subject mention μ from the word sequence and replace it with a placeholder token "$\langle e \rangle$" representing the subject entity s. Let $v := (v_{q,1}, v_{q,2}, \ldots, v_{q,n'})$ denote the modified word sequence from q, and let $\overline{v} := (\overline{v}_{q,1}, \overline{v}_{q,2}, \ldots, \overline{v}_{q,n'})$ be its word2vec(Mikolov et al., 2013) embeddings. We simply create a random embedding for "$\langle e \rangle$" and include it in the dictionary.

The sequence $\overline{v}$ is then entered to a Bi-GRU network and the sum of the two final states from the network is denoted by u^{Q}. We then set the output u of the ENC model as the concatenation of u^{Q} and u^{KB}. This completes the ENC block.

The DEC block is a simple GRU network following the usual Seq2Seq(Sutskever et al., 2014) decoder structure. The network takes vector u as its initial state. The input to the network at $t = 1$ is the embedding of a contrived special relation label "GO", signalling the beginning of the input. The input at any other time $t > 1$ is the embedding of the $(t-1)^{\mathrm{th}}$ relation r_{t-1} in the path label r. The embedding dictionary of the relations here is a learnable parameter. Each output of the GRU network at time t is passed to a soft-max function (with learnable parameters) to generate the predictive distribution for the t^{th} relation r_t in r.

Under the maximum-likelihood principle, the loss function $\ell_{\mathrm{pr}}(q, s, r)$ of ENC-DEC is the sum of cross-entropy losses, each between a generated distribution at t and a observed relation r_t in r. We note that, as is standard, a special "EOS" token is included in r to indicate the termination of r.

4.2 CLS Model

The verification classifier CLS is constructed as a Multilayer Perceptron (MLP). The input of the MLP is the concatenation of u with the KSE embedding of a relation r. The MLP outputs the predictive distribution of y. Its loss function $\ell_{\mathrm{ve}}(q, s, r, y)$ for training CLS is defined as the cross entropy between the predictive distribution and the observed class label y.

4.3 Turbo Training of ENC-DEC and CLS

Let θ_{enc}, θ_{dec}, and θ_{cls} denote respectively all parameters in ENC, in DEC and in CLS. The total prediction loss is then

$$\mathcal{L}_{\mathrm{pr}}(\theta_{\mathrm{enc}}, \theta_{\mathrm{dec}}) := \sum_{(q,s,r) \in \mathcal{D}_{\mathrm{QSR}}} \ell_{\mathrm{pr}}(q, s, r; \theta_{\mathrm{enc}}, \theta_{\mathrm{dec}}) \tag{6}$$

and the total verification loss is

$$\begin{aligned} \mathcal{L}_{\mathrm{ve}}(\theta_{\mathrm{enc}}, \theta_{\mathrm{cls}}) := & \sum_{(q,s,r) \in \mathcal{D}_{\mathrm{QSR}}} \ell_{\mathrm{ve}}(q, s, r, 1; \theta_{\mathrm{enc}}, \theta_{\mathrm{cls}}) \\ & + \sum_{(q,s,r) \in \mathcal{D}^{-}_{\mathrm{QSR}}} \ell_{\mathrm{ve}}(q, s, r, 0; \theta_{\mathrm{enc}}, \theta_{\mathrm{cls}}) \end{aligned} \tag{7}$$

It is worth noting that the loss ℓ_{ve} does not depend on θ_{dec}, since r is observed in the CLS model. It however depends on θ_{enc} through u.

We define the overall loss for training the Prediction-Verification Model as

$$\mathcal{L} := \mathcal{L}_{\mathrm{pr}}(\theta_{\mathrm{enc}}, \theta_{\mathrm{dec}}) + \lambda \mathcal{L}_{\mathrm{ve}}(\theta_{\mathrm{enc}}, \theta_{\mathrm{cls}}). \tag{8}$$

Here λ is chosen larger than 1 by a decent margin. This is because when using the trained model for answering questions, we will primarily rely on CLS rather than ENC-DEC.

We now present a training algorithm, which we call "turbo training", to minimize the loss function $\mathcal{L}$. Turbo training consists of two phases.

Base Phase.Train ENC-DEC, namely minimizing $\mathcal{L}_{\mathrm{pr}}$ over $(\theta_{\mathrm{enc}}, \theta_{\mathrm{dec}})$.

Turbo Phase. In this phase, we iterate over the following three steps in order.

A.Train CLS, namely, minimizing $\mathcal{L}_{\mathrm{ve}}$ over θ_{cls} for the current setting of $(\theta_{\mathrm{enc}}, \theta_{\mathrm{dec}})$.

B.Train ENC, namely, minimizing $\mathcal{L}_{\mathrm{pr}} + \lambda \mathcal{L}_{\mathrm{ve}}$ over θ_{enc} for the current setting of $(\theta_{\mathrm{dec}}, \theta_{\mathrm{cls}})$.

C.Train DEC, namely, minimizing $\mathcal{L}_{\mathrm{pr}}$ over θ_{dec} for the current setting of $(\theta_{\mathrm{enc}}, \theta_{\mathrm{cls}})$.

A diagram showing the turbo training algorithm is given in Figure 3. It can be proved that the three

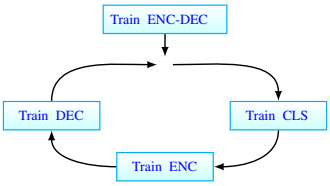


Figure 3: Turbo training

steps in the turbo phase correspond precisely to coordinate-descent minimization of $\mathcal{L}$. Therefore iterations in the turbo phase *provably* decreases the loss $\mathcal{L}$ monotonically and are guaranteed to converge.

In the implementation of turbo training, for each phase and the each step within, the minimization can be carried out via mini-batched SGD over a number of mini-batches. Additionally, when the learning rate of SGD is small, the updates of θ_{enc} in the turbo phase (step B) are primarily dominated by gradient signal of $\mathcal{L}_{\text{ve}}$, due to the large value of λ. Thus step B of the turbo phase can be simplified to minimizing only the loss $\mathcal{L}_{\text{ve}}$.

4.4 Question Answering with APVA-TURBO

At this end, we have completed explaining APVA-TURBO and the training of each component within. After it is trained, APVA-TURBO can be used to solve a QA-KB task according to the following steps.

To answer a question q, the Entity Alignment model is first applied to q and the KB $\mathcal{K}$. A list of M_{s} subject entities are generated as candidates according to their ranking under the model.

Each candidate entity is used as a hypothetic subject entity s, and passed to ENC-DEC together with q and $\mathcal{K}$. In this step, a list of M_{r} candidate path labels are generated. Specifically these candidate path labels are the highest ranked M_{r} path labels among all path labels r for which there is a path with label r in the KB that leaves from s. The ranking of the path labels is according to the learned prediction loss ℓ_{pr}. At the end of this step, a total of $M_{\text{s}}M_{\text{r}}$ (s, r) pairs are generated for question q.

Each of these subject-path pairs (s, r) is then passed to CLS, and the highest ranked (s, r) pair is declared as the correct subject-path pair. Here the ranking is according to the verification loss ℓ_{ve}.

Finally, submitting the declared (s, r) (together with q if needed) to the Object Answering model gives a list $\widehat{\mathcal{A}}(q)$ of answer paths, which APVA-TURBO declares as the answers to question q.

5 Experiments

5.1 Datasets and Pre-Processing

Two popular datasets, SimpleQuestions (Bordes et al., 2015) and WebQuestions (Berant et al., 2013) are used in our experiments.

SimpleQuestions (SQ) consists of 108,442 questions written by human English-speaking annotators. The dataset is partitioned randomly into 75,910 training questions, 10,845 validation questions and 21,687 test questions. The questions in SQ are all single-hop single-answer.

WebQuestions (WQ) consists of 5,810 questions. The dataset is partitioned randomly into 3000 training questions, 778 validation questions and 2032 test questions. The questions in WQ can be multi-hop multi-answer.

The answer triples or paths in SQ and WQ are all in Freebase. A subset of Freebase, FB2M (Bordes et al., 2015), is also used in our study. It includes 2,150,604 entities, 6701 relations and 14,180,937 triples. Since FB2M does not contain the entity type information, we retrieve this information from Freebase to annotate the type for each entity. Specifically, Freebase contains two kinds of types, "common.topic.notable_types" (or "notable type") and "type.object.type". Each entity in Freebase is designed to have a unique notable type while allowed to have many object types. We use notable types as the entity types. For an entity with the notable type missing in Freebase, we annotate it as "NO TYPE".

The answer triples in SQ can all be found in FB2M, whereas not all answer paths in WQ are within FB2M. Thus, for the QA-KB tasks, the KB associated with SQ is chosen as FB2M, and that with WQ is chosen as the entire Freebase.

For both SQ and WQ, the subject mention μ in each question q is also annotated by matching the surface name of the subject entity with the words in q. Each answer triple (s, r, o) in SQ is presented directly in terms of the unique ID's specifying the involved entities and the unique string specifying the relation r. In WQ, however, for each answer path, the path label r is not given, and the entities s and o are only given in terms of their names. We therefore perform a simple search and screening to reconstruct each answer path in its unique (s, r, o) representation.

5.2 Implementations

Given the subject mention μ, the subject entity s in $\mathcal{K}$ can be found easily, as long as the question is well formed and the subject entity s exists in $\mathcal{K}$ unambiguously. Determining the subject entity s from subject mention μ usually only involves matching the words in u with the surface name string of the entities in $\mathcal{K}$. So the problem of entity alignment reduces to determining the subject mention μ in q. Conventionally, such a problem is treated as sequence labelling problem. In this work, our entity alignment model combines the recent BiGRU-CRF model (Dai et al., 2016) with joint character-level word encoding (Zhang et al., 2015). The entity alignment model is trained separately. In the model, the state/output dimension of the GRU network, the dimension of word embeddings and that of character encoding vectors are all set to 50.

The KSE model is also trained separately. In this model, FB2M is used to construct the schema graph for both SQ and WQ datasets, and the dimension of the embedding space is chosen as 100.

For ENC-DEC, the state dimension for the GRU network encoding u^{KB} and the input dimension are both chosen as 100. The state dimension for each of the two GRU networks for encoding u^{Q} is set to 200. Each input to these networks is a 200-dimensional word2vec vector (pre-trained on a large corpus WikiAnswers (Fader et al., 2013)) concatenated with a 10-dimensional position vector indicating the relative location of the word in the question. The state dimension for the GRU network in DEC is set to 300.

In CLS, the MLP has two ReLU activated latent layers, with dimensions 200 and 100 respectively.

Turbo training with mini-batch SGD is used to jointly train ENC-DEC and CLS. The batch size is set to 128. In the BASE phase, the ENC-DEC model is trained for 8000 batches on SQ, and 1500 batches on WQ. In the first round of TURBO phase, the number of batches in Steps A, B, and C are respectively 2000, 1000, 100 on SQ, and 1000, 500, 50 on WQ. In the remaining rounds, every step uses 100 batches on SQ, and 50 batches on WQ.

For the given question q, we denote the set of entities by $\mathcal{O}(s, r)$, such that there is a path in the KB graph that connect entity s to entity o via a path with label r. For a multiple-answer question q, not necessarily all entities in $\mathcal{O}(s, r)$ give rise to a correct answer. A usual approach to deal with this is to learn a separate binary classifier to screen the entities in $\mathcal{O}(s, r)$. In this work, we use a simple SVM classifier as the object answering model. The input feature vector for the SVM is the concatenation of three vectors: the average word2vec vectors of all words in q, the KSE embedding vector $\overline{r}_T$ of the final relation r_T in the path label r (assuming the path is T-hop), and the KSE embedding of $\bar{t}_o$ is the type $t(o)$ of the candidate object entity $o \in \mathcal{O}(s, r)$.

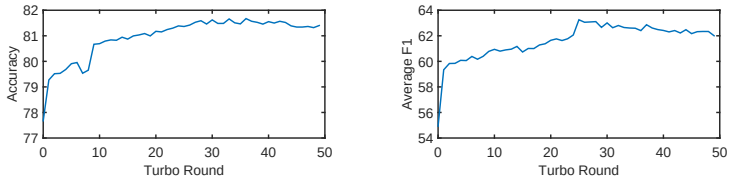
5.3 Question Answering Performance

Extensive experiments are performed to evaluate the proposed APVA-TURBO model in QA-KB tasks, relative to other existing models and various reductions of APVA-TURBO.

In experiments on SQ, the single answer triple (s, r, o) in $\widehat{A}(q)$ produced by a model is regarded as correct, if it matches the *unique* correct answer triple. The *accuracy* of a model is defined as the percentage of test questions for which the model produces the correct answer. On SQ, we use this accuracy to measure model performance.

On WQ, a given question q may have multiple answer paths (s, r, o) in the ground-truth answer set $\mathcal{A}(q)$. The model's *precision* for answering a question q is defined as the fraction of answers in $\widehat{\mathcal{A}}(q)$ that are in $\mathcal{A}(q)$, and the model's *recall* for answering q is the fraction of answers in $\mathcal{A}(q)$ that are in $\widehat{\mathcal{A}}(q)$. In such a setting, the overall performance is usually measured by the harmonic mean of precision and

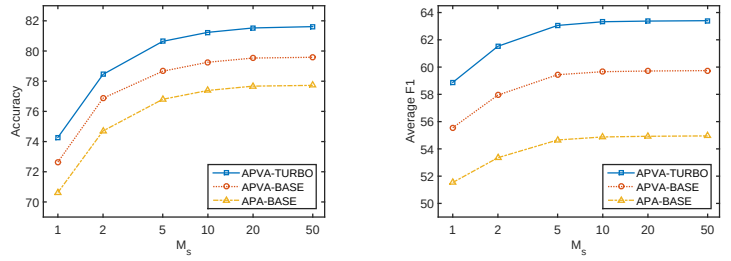
recall, namely, F1. We then use the average F1 value of a model over all test questions to measure the model's performance on WQ.



(a) SimpleQuestions (b) WebQuestions

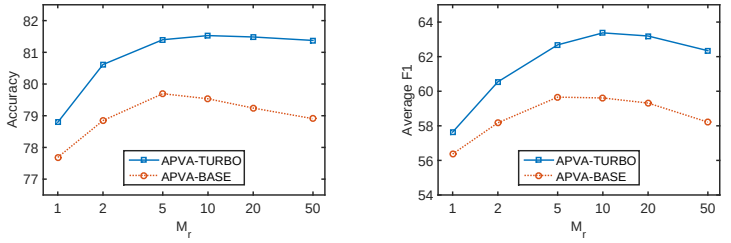
Figure 4: The performance of APVA-TURBO over TURBO iterations. $M_{\mathrm{s}} = 20$, $M_{\mathrm{r}} = 10$.

Analysis of APVA-TURBO The performance of APVA-TURBO over turbo iterations is shown in Figure 4. It can be seen that the performance increases with turbo iterations until 25 to 30 rounds. This demonstrates the effectiveness of turbo training. The performance starts to take a slight decreasing trend at around 30 to 40 rounds, due to overfitting, a phenomenon commonly arising when a model is overtrained. In the rest of the experiments, the number of Turbo rounds is taken as 30 for SQ and 25 for WQ.



(a) SimpleQuestions (b) WebQuestions

Figure 5: Performances of APVA-TURBO and its reductions vs M_{s}. $M_{\mathrm{r}} = 10$.



(a) SimpleQuestions (b) WebQuestions

Figure 6: Performances of APVA-TURBO and its reductions vs M_{r}. $M_{\mathrm{s}} = 20$.

Figure 5 plots the performances of APVA-TURBO and its various reductions as functions of the number M_{s} of candidate subject entities. APVA-BASE is the APVA-TURBO model without full turbo training. Specifically, after training in the Base Phase, it only enters the Turbo Phase once and stops right after the training of CLS. APA-BASE is a reduction of APVA-BASE by completely ignoring CLS. Including the verification classifier CLS appears to give a much significant boost of the performance (seen in APVA-BASE vs APA-BASE). Yet another leap in performance is achieved by including turbo training (seen in APVA-TURBO vs APVA-BASE).

It is also seen in Figure 5 that as M_{s} increases, the performances of the compared models all improve and eventually saturate after $M_{\mathrm{s}} = 20$. This performance improvement is expected since when more candidate subject entities are included, it increases the chance that the correct (s, r) pairs are selected.

In addition, although a high value of $M_{\rm s}$ also increases the chance of selecting a spurious (s, r) pair, we see no drop in performance. This is because the training of ENC-DEC has always been using *only the correct* subject entities. When a wrong entity is submitted to ENC-DEC, the path labels output from ENC-DEC tend to be a random draw, which are not competitive against the correct ones.

Figure 6 plots the performances of APVA-BASE and APVA-TURBO models against the number $M_{\rm r}$ of candidate path labels. Comparing the performance gap between the two models, it is apparent that turbo training brings in significantly benefit.

Figure 6 also shows that as $M_{\rm r}$ increases, the performances of the compared models all first increase and then gradually drop. This is because increasing $M_{\rm r}$ has two effects. The first is increasing the chance of selecting the correct path labels. The second is increasing the number of spurious path labels. It is sensible that a small increment of $M_{\rm r}$ makes the first effect dominate while a large increment makes the second dominate. The curves in the figure also show that the stronger a model is, at a higher $M_{\rm r}$ the performance trend reverts.

Table 1: Performance comparison on WQ

Model	F1(%)
(Yih et al., 2015)	52.5
(Xu et al., 2016)	53.3
(Yavuz et al., 2016)	52.6
(Jain, 2016)	**55.7**
(Reddy et al., 2017)	49.5
(Hao et al., 2017)	42.9
(Dong et al., 2017)	50.7
(Yue et al., 2017)	53.9
(Cheng et al., 2017)	50.1
APVA-BASE	**59.7**
APVA-TURBO	**63.4**

Table 2: Performance comparison on SQ

Model	Accuracy (%)
(Yin et al., 2016)	76.4
(Lukovnikov et al., 2017)	71.2
(Hsiao et al., 2017)	76.7
(Yu et al., 2017)	**78.7**
(Ture and Jojic, 2017)	**88.3**
APVA-BASE	**79.5**
APVA-TURBO	**81.5**

Performance Comparison with Existing Models The performance of APVA-TURBO is compared with the published performances of the existing models in Tables 1 and 2.

On WQ, the best known previous performance achieves an F1 value of 55.7%, due to (Jain, 2016). But this state of the art has been surpassed by APVA-BASE, with a F1 score 59.7%. This demonstrates the effectiveness of augmenting a model with a verification mechanism. With turbo training, APVA-TURBO further boosts its performance and presents a new F1 record of 63.4%. This performance, exceeding the current art by a stunning margin of nearly 8%, demonstrates the power of the APVA-TURBO approach developed in this paper.

On SQ, the best performance except that of (Ture and Jojic, 2017) achieves an accuracy of 78.7%. This performance has been surpassed by APVA-BASE, and further exceeded by APVA-TURBO at an accuracy of 81.5%. However that the current reported state of art is still that of (Ture and Jojic, 2017),

which achieves an accuracy of 88.3%. But that paper did not report sufficient details for us to understand in what respect APVA-TURBO under-performs. Neither does it contain all necessary details for us to implement the model and reproduce its reported performance.

6 Conclusion

This work demonstrates that incorporating a verification mechanism in a QA-KB model can be a powerful means of improving the model. Built on such a mechanism, the proposed APVA framework is also naturally facilitated with an iterative learning procedure, which we call turbo training. Turbo training is shown capable of significantly boosting the model performance. This makes APVA-TURBO a new state of art in QA-KB.

Acknowledgments

This work is supported partly by China 973 program (Nos.2015CB358700, 2014CB340305), by the National Natural Science Foundation of China (Nos.61772059, 61421003). This paper is also supported by the State Key Laboratory of Software Development Environment of China and Beijing Advanced Innovation Center for Big Data and Brain Computing.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 615–620.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *Computer Science*.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2017. Learning an executable neural semantic parser. *arXiv preprint arXiv:1711.05066*.

Zihang Dai, Lei Li, and Wei Xu. 2016. Cfo: Conditional focused neural question answering with large-scale knowledge bases. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 800–810.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 260–269.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 875–886.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1608–1618. Association for Computational Linguistics.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 221–231.

Wei-Chuan Hsiao, Hen-Hsen Huang, and Hsin-Hsi Chen. 2017. Integrating subject, type, and property identification for simple question answering over knowledge base. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 976–985.

Sarthak Jain. 2016. Question answering over knowledge base using factual memory networks. In *Proceedings of the NAACL Student Research Workshop*, pages 109–115.

Gregory Karvounarakis, Sofia Alexaki, Vassilis Christophides, Dimitris Plexousakis, and Michel Scholl. 2002. Rql: a declarative query language for rdf. In *Proceedings of the 11th international conference on World Wide Web*, pages 592–603. ACM.

Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. 2017. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th international conference on World Wide Web*, pages 1211–1220. International World Wide Web Conferences Steering Committee.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Chimezie Ogbuji. 2005. Versa: Path-based rdf query language. *O'Reilly XML. com*.

Eric Prud, Andy Seaborne, et al. 2006. Sparql query language for rdf.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 89–101.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pages 3104–3112. MIT Press.

Ferhan Ture and Oliver Jojic. 2017. No need to pay attention: Simple recurrent neural networks work! In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2866–2872.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2326–2336.

Semih Yavuz, Izzeddin Gur, Yu Su, Mudhakar Srivatsa, and Xifeng Yan. 2016. Improving semantic parsing via answer type inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 149–159.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1321–1331.

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple question answering by attentive convolutional neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1746–1756.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 571–581.

Bin Yue, Min Gui, Jiahui Guo, Zhenglu Yang, Jin-Mao Wei, and Shaodi You. 2017. An effective framework for question answering over freebase via reconstructing natural sequences. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 865–866. International World Wide Web Conferences Steering Committee.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 649–657, Cambridge, MA, USA. MIT Press.

An Interpretable Reasoning Network for Multi-Relation Question Answering

Mantong Zhou Minlie Huang* Xiaoyan Zhu
State Key Lab. of Intelligent Technology and Systems,
National Lab. for Information Science and Technology,
Dept. of Computer Science and Technology, Tsinghua University, Beijing, PR China
zmt.keke@gmail.com, aihuang@tsinghua.edu.cn,
zxy-dcs@tsinghua.edu.cn

Abstract

Multi-relation Question Answering is a challenging task, due to the requirement of elaborated analysis on questions and reasoning over multiple fact triples in knowledge base. In this paper, we present a novel model called **Interpretable Reasoning Network** that employs an interpretable, hop-by-hop reasoning process for question answering. The model dynamically decides which part of an input question should be analyzed at each hop; predicts a relation that corresponds to the current parsed results; utilizes the predicted relation to update the question representation and the state of the reasoning process; and then drives the next-hop reasoning. Experiments show that our model yields state-of-the-art results on two datasets. More interestingly, the model can offer traceable and observable intermediate predictions for reasoning analysis and failure diagnosis, thereby allowing manual manipulation in predicting the final answer.

1 Introduction

Open-domain Question Answering (QA) has always been a hot topic in AI and this task has recently been facilitated by large-scale Knowledge Bases (KBs) such as Freebase (Bollacker et al., 2008). However, due to the variety and complexity of language and knowledge, open-domain question answering over knowledge bases (KBQA) is still a challenging task.

Question answering over knowledge bases falls into two types, namely single-relation QA and multi-relation QA, as argued by Yin et al. (2016). Single-relation questions, such as *"How old is Obama?"*, can be answered by finding one fact triple in KB, and this task has been widely studied (Bordes et al., 2015; Xu et al., 2016; Savenkov and Agichtein, 2017). In comparison, reasoning over multiple fact triples is required to answer multi-relation questions such as *"Name a soccer player who plays at forward position at the club Borussia Dortmund."* where more than one entity and relation are mentioned. Compared to single-relation QA, multi-relation QA is yet to be addressed.

Previous studies on QA over knowledge bases can be roughly categorized into two lines: semantic parsing and embedding-based models. Semantic parsing models (Yih et al., 2014; Yih et al., 2016) obtain competitive performance at the cost of hand-crafted features and manual annotations, but lack the ability to generalize to other domains. In contrast, embedding-based models (Bordes et al., 2014b; Hao et al., 2017; Yavuz et al., 2017) can be trained end-to-end with weak supervision, but existing methods are not adequate to handle multi-relation QA due to the lack of reasoning ability.

Recent reasoning models (Miller et al., 2016; Wang et al., 2017) mainly concentrate on Reading Comprehension (RC) which requires to answer questions according to a given document. However, transferring existing RC methods to KBQA is not trivial. For one reason, the focus of reasoning in RC is usually on understanding the document rather than parsing questions. For another reason, existing reasoning networks are usually designed in a black-box style, making the models less interpretable. While in multi-relation question answering, we believe that an interpretable reasoning process is essential.

In this paper, we propose a novel Interpretable Reasoning Network (IRN) to equip QA systems with the reasoning ability to answer multi-relation questions. Our central idea is to design an interpretable

*Corresponding author: Minlie Huang (aihuang@tsinghua.edu.cn).

Proceedings of the 27th International Conference on Computational Linguistics, pages 2010–2022
Santa Fe, New Mexico, USA, August 20-26, 2018.

reasoning process for a complex question: the reasoning module decides which part of an input question should be analyzed at each hop, and predicted a KB relation that corresponds to the current parsed results. The predicted relation will be used to update the question representation as well as the state of the reasoning module, and helps the model to make the next-hop reasoning. At each hop, an entity will be predicted based on the current state of the reasoning module.

Different from previous models, our model is ***interpretable*** in that the predicted relation and entity at each hop are ***traceable and observable***. At each hop our model has a specific aim to find an appropriate relation based on the iterative analysis of a question, and intermediate output at each hop can be interpreted by the corresponding linked entity. In this manner, IRN offers the ability of visualizing ***a complete reasoning path*** for a complex question, which facilitates reasoning analysis and failure diagnosis, thereby allowing manual manipulation in answer prediction as detailed in our experiments.

The contributions of this paper are in two folds:

1. We design an Interpretable Reasoning Network which can make reasoning on multi-relation questions with multiple triples in KB. Results show that our model obtains state-of-the-art performance.

2. Our model is more interpretable than existing reasoning networks in that the intermediate entities and relations predicted by the hop-by-hop reasoning process construct traceable reasoning paths to clearly reveal how the answer is derived.

2 Related Works

Recent works on QA can be roughly classified into two types: one is semantic-parsing-based and the other is embedding-based. Semantic parsing approaches map questions to logical form queries (Pasupat and Liang, 2015; Yih et al., 2016; Abujabal et al., 2017). These systems are effective but at the cost of heavy data annotation and pattern/grammar engineering. What's more, parsing systems are often constrained on a specific domain and broken down when executing logical queries on incomplete KBs.

Our work follows the line of Embedding-based models (Bordes et al., 2014b; Dong et al., 2015; Xu et al., 2016; Hao et al., 2017; Yavuz et al., 2017) which are recently introduced into the QA community where questions and KB entities are represented by distributed vectors, and QA is formulated as a problem of matching between vectors of questions and answer entities. These models need less grammars as well as annotated data, and are more flexible to deal with incomplete KBs. To make better matching, subgraphs of an entity in KB (Bordes et al., 2014a), answer aspects (Dong et al., 2015; Hao et al., 2017) and external contexts (Xu et al., 2016) can be used to enrich the representation of an answer entity. Though these methods are successful to handle simple questions, answering multi-relation questions or other complex questions is far from solved, since such a task requires reasoning or other elaborated processes.

Our work is also related to recent reasoning models which focus on Reading Comprehension where memory modules are designed to comprehend documents. State-of-the-art memory-based Reading Comprehension models (Sukhbaatar et al., 2015; Kumar et al., 2015; Shen et al., 2016; Wang et al., 2017; Celikyilmaz et al., 2017) make interactions between a question and the corresponding document in a multi-hop manner during reasoning. MemNN (Weston et al., 2015), KVMemN2N (Miller et al., 2016) and EviNet (Savenkov and Agichtein, 2017) transferred the reading comprehension framework to QA where a set of triples is treated as a document and a similar reasoning process can be applied. However, reading comprehension makes reasoning over documents instead of parsing the questions.

Other studies applying hop-by-hop inference into QA can be seen in Neural Programmer (Neelakantan et al., 2015; Neelakantan et al., 2016) and Neural Enquirer (Yin et al., 2015), where deep networks are proposed to parse a question and execute a query on tables. However, Neural Programmer needs to predefine symbolic operations, while Neural Enquirer lacks explicit interpretation. Mou et al. (2017) proposed a model coupling distributed and symbolic execution with REINFORCE algorithm, however, training such a model is challenging. Neural Module Network (Andreas et al., 2015; Andreas et al., 2016) customized network architectures for different patterns of reasoning, making the reasoning network interpretable. However, a dependency parser and the REINFORCE algorithm are required.

3 Interpretable Reasoning Network

3.1 Task Definition

Our goal is to offer an interpretable reasoning network to answer multi-relation questions. Given a question q and its topic entity or subject e_s which can be annotated by some NER tools, the task is to find an entity a in KB as the answer.

In this work, we consider two typical categories of multi-relation questions, a path question (Guu et al., 2015) and a conjunctive question (Zhang et al., 2016), while the former is our major focus.

A path question contains only one topic entity (subject e_s) and its answer (object a) can be found by walking down an answer path consisting of a few relations and the corresponding intermediate entities. We define an **answer path** as a sequence of entities and relations in KB which starts from the subject and ends with the answer like $e_s \xrightarrow{r_1} e_1 \xrightarrow{r_2} ... \xrightarrow{r_n} a$. Relations ($r_i$) are observable (in various natural language forms) in the question, however, the intermediate entities ($e_1 \cdots e_H$) are not. For example, for question *"How old is Obama's daughter?"*, the subject is *Barack_Obama* and the answer path is *Barack_Obama* $\xrightarrow{Children}$ *Malia_Obama* $\xrightarrow{Age}$ *18*. Note that since there are 1-to-many relations[1], the range of the intermediate entities can be large, resulting in more than one answer path for a question.

A conjunctive question is a question that contains more than one subject entity and the answer can be obtained by the intersection of results from multiple path queries. For instance, the question *"Name a soccer player who plays at forward position at the club Borussia Dortmund."* has a possible answer as the intersection of results from two path queries[2] $FORWARD \xrightarrow{plays\_position^{-1}} Marco\_Reus$ and $Borussia\_Dortmund \xrightarrow{plays\_in\_club^{-1}} Marco\_Reus$. The details for dealing with conjunctive questions are shown in Fig 2.

3.2 Overview

The reasoning network has three modules: input module, reasoning module, and answer module. The input module encodes the question into a distributed representation and updates the representation hop-by-hop according to the inference results of the reasoning module. The reasoning module initializes its state by the topic entity of a question and predicts a relation on which the model should focus at the current hop, conditioned on the present question and reasoning state. The predicted relation is utilized to update the state vector and the question representation hop-by-hop. The answer module predicts an entity conditioned on the state of the reasoning module.

The process can be illustrated by the example as shown in Figure 1. For question *"How old is Obama's daughter?"*, the subject entity *Barack_Obama* is utilized to initialize the state vector. IRN predicts the first relation *"Children"* at the first hop. The *"Children"* relation is added to the state vector to encode the updated parsing result, and the corresponding natural language form of this relation in the question (here is *"daughter"*) is subtracted from the question to avoid repeatedly analyzing the relation-relevant word *"daughter"*. This procedure is repeated until the *Terminal* relation is predicted.

3.3 Input Module

The input module encodes a question to a vector representation and updates the representation of the question at each hop of the reasoning process: the predicted relation will be subtracted from the current representation to compel the reasoning process to attend to other words that should be analyzed.

Formally, the question $\boldsymbol{X} = x_1, x_2, ..., x_n$ can be initialized by the sum of the word embeddings and updated by subtracting the relation predicted by the reasoning module at the previous hop:

$$\boldsymbol{q}^0 = \sum_{i=1}^{n} \boldsymbol{x}_i \quad (1)$$

$$\boldsymbol{q}^h = \boldsymbol{q}^{h-1} - \boldsymbol{M}_{rq}\hat{\boldsymbol{r}}^h \quad (2)$$

[1]For instance, relation *Children* is one-to-many, where a person may have more than one child.

[2]Superscript -1 stands for the inverse relation.

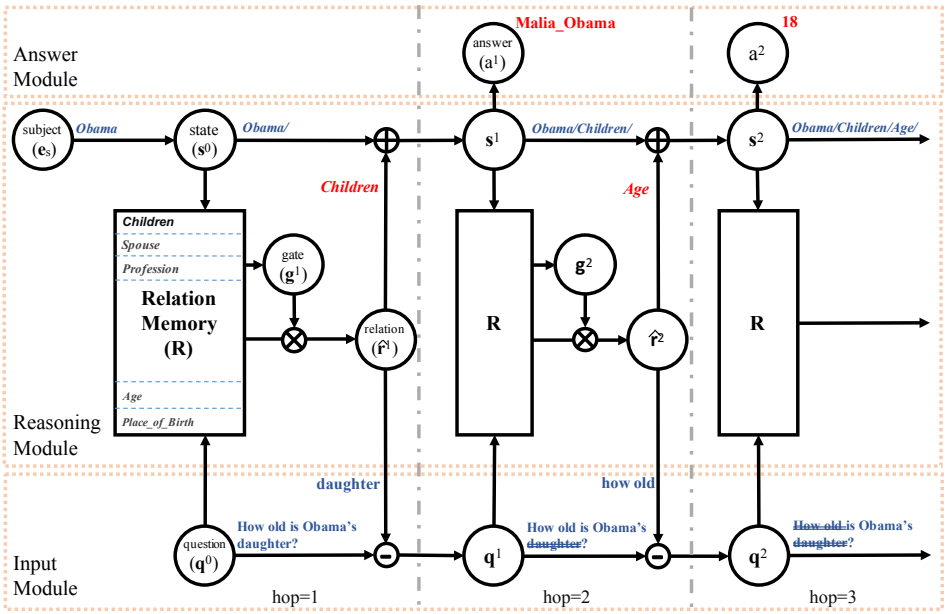


Figure 1: Interpretable Reasoning Network. At each hop IRN computes the probability of selecting the next relation as $\boldsymbol{g}^h$ and obtains a predicted relation $\hat{\boldsymbol{r}}^h$. The predicted relation $\hat{\boldsymbol{r}}^h$ is used to update the question $\boldsymbol{q}^h$ and the state $\boldsymbol{s}^h$ with different projections. The state is initialized by subject as $\boldsymbol{s}^0 = \boldsymbol{e}_s$. The answer path ($Obama \xrightarrow{Children} Malia\_Obama \xrightarrow{Age} 18$) is composed of the predicted relations and entities (in red).

where $\boldsymbol{M}_{rq}$ is a matrix projecting the KB relation space to the natural language question space, $\boldsymbol{q}^{h-1}$ is the question representation at hop $h-1$, and $\hat{\boldsymbol{r}}^h$ defined by Eq. 4 is the predicted relation at hop h. The intuition of such update is that the already analysed part of the question should not be parsed again.

Representing a question as a bag of words might be too simple. However, this method works well in our setting. Future work would consider other sophisticated encoders such as CNN or LSTM.

3.4 Reasoning Module

The reasoning module aims to attend to a particular part of the question at each hop, predict an associated relation in knowledge base, and then update its state.

The reasoning module takes as input the previous state vector ($\boldsymbol{s}^{h-1}$) and the previous question vector ($\boldsymbol{q}^{h-1}$), and then predicts a relation ($\hat{\boldsymbol{r}}^h$) based on the analysis at the current hop. Once the predicted relation ($\hat{\boldsymbol{r}}^h$) is obtained, the relation will be used to update the next question representation ($\boldsymbol{q}^h$) and the state of the reasoning module ($\boldsymbol{s}^h$). In this manner, the reasoning network is traceable and interpretable.

The process can be formally described by the following equations[3]:

$$g_j^h = P(r^h = r_j | \boldsymbol{q}^{h-1}, \boldsymbol{s}^{h-1}) = \text{softmax}((\boldsymbol{M}_{rq}\boldsymbol{r}_j)^T \boldsymbol{q}^{h-1} + (\boldsymbol{M}_{rs}\boldsymbol{r}_j)^T \boldsymbol{s}^{h-1}) \tag{3}$$

$$\hat{\boldsymbol{r}}^h = \sum_j g_j^h * \boldsymbol{r}_j \tag{4}$$

$$\boldsymbol{s}^h = \boldsymbol{s}^{h-1} + \boldsymbol{M}_{rs}\hat{\boldsymbol{r}}^h \tag{5}$$

where $\boldsymbol{r}_j$ is the embedding of a relation in KB and all the relation embeddings are stored in a static memory R, and $\boldsymbol{s}^h$ is the state of the reasoning module at hop h. g_j^h is the probability of selecting the j^{th} relation in KB and $\boldsymbol{M}_{rs}$ is the projection matrix mapping $\boldsymbol{r}$ from the relation space to the state space. $\boldsymbol{M}_{rq}$ is the same projection matrix used in Eq. 2 to map $\boldsymbol{r}$ from the relation space to the question space.

We initialize the state vector with the topic entity (subject) $\boldsymbol{s}^0 = \boldsymbol{e}_s$. IRN will learn to enrich the state representation hop by hop, for instance, at the first hop $\boldsymbol{s}^1 \approx \boldsymbol{e}_s + \boldsymbol{r}_1$, and at the second hop $\boldsymbol{s}^2 \approx \boldsymbol{e}_s + \boldsymbol{r}_1 + \boldsymbol{r}_2$, intuitively. In this manner, the state vector encodes historical information.

[3] $\boldsymbol{a}^T\boldsymbol{b}$ is the inner-product of vector $\boldsymbol{a}$ and $\boldsymbol{b}$.

In order to signify when the reasoning process should stop, we augment the relation set with the *Terminal* relation. Once the reasoning module predicts the *Terminal* relation, the reasoning process will stop, and the final answer will be the output when the last non-terminal relation is added to the state $\boldsymbol{s}$.

3.5 Answer Module

The answer module chooses the corresponding entity from KB at each hop (denoted as a^h). At the last hop, the selected entity is chosen as the final answer, while at the intermediate hops, the predictions of these entities can be inspected to help reasoning analysis and failure diagnosis.

More formally, an entity at each hop can be predicted as follows:

$$\boldsymbol{e}^h = \boldsymbol{M}_{se}\boldsymbol{s}^h \tag{6}$$

$$o_j^h = P(a^h = e_j|\boldsymbol{s}^h) = \text{softmax}(\boldsymbol{e}_j^T\boldsymbol{e}^h) \tag{7}$$

$\boldsymbol{M}_{se}$ is used to transfer from the state space ($\boldsymbol{s}^h$) to the entity space ($\boldsymbol{e}^h$) to bridge the representation gap between the two spaces. $\boldsymbol{e}_j$ is the embedding vector of the j^{th} entity in KB.

3.6 Loss Function

We adopt cross entropy to define the loss function. The first loss term is defined on the intermediate prediction of relations, while the second term on the prediction of entities.

The loss on one instance is defined as follows:

$$\mathcal{L} = \sum_h \mathcal{C}_r(h) + \lambda\mathcal{C}_a(h) \tag{8}$$

$$\mathcal{C}_r(h) = -\sum_{j=1}^{n_r}[\hat{g}_j^h \ln g_j^h]\ ,\ \mathcal{C}_a(h) = -\sum_{i=1}^{n_e}[\hat{o}_i^h \ln o_i^h]$$

where n_r/n_e is the number of relations/entities in KB respectively, $\hat{\boldsymbol{g}}^h$ is the gold distribution (one-hot) over relations at hop h, $\boldsymbol{g}^h$ is the predicted distribution defined by Eq. 3, $\hat{\boldsymbol{o}}$ is the gold distribution over entities, which is also one-hot representation, and $\boldsymbol{o}$ is defined by Eq. 7. λ is a hyper parameter to balance the two terms.

Note that the training data is in the form of $(q, < e_s, r_1, e_1, ..., a >)$, which indicates that the model can incorporate supervision not only from the final answer (referred to as IRN-weak), but also from the intermediate relations and entities along the answer path (referred to as IRN).

3.7 Multitask Training for KB Representation

In order to incorporate more constraints from KB[4], we learn the embeddings of entities and relations as well as the space transition matrix with a multitask training schema. For a given fact triple in KB, (e_s, r, e_o), the representations of the entities and the relation apply the following constraint:

$$\boldsymbol{M}_{se}(\boldsymbol{e}_s + \boldsymbol{r}) = \boldsymbol{e}_o \tag{9}$$

where $\boldsymbol{e}_s, r, \boldsymbol{e}_o$ are embeddings of the subject (or head) entity, relation, and the object (or tail) entity. This idea is inspired by TransE (Bordes et al., 2013), but we adopt $\boldsymbol{M}_{se}$ (see Eq. 6) as a transfer matrix to bridge the representation gap between the state space (here $\boldsymbol{e}_s + \boldsymbol{r} = \boldsymbol{s}$) and the entity space (here $\boldsymbol{e}_o = \boldsymbol{e}$).

The parameters are updated with a multi-task training schema. We first learn the KB embeddings $\boldsymbol{e}$/$\boldsymbol{r}$ and the transformation matrix $\boldsymbol{M}_{se}$ to fit Eq. 9 with several epoches. This is the task of KB embedding training. Then, we update all the parameters of IRN under supervision from the QA task with one epoch, which is the task of QA training. We run the two tasks iteratively.

With the help of the auxiliary KB embedding training, IRN not only utilizes the additional information from KB to make better inferences, but also has an ability to deal with incomplete answer paths. For example, even if the connection between *Barack_Obama* and *Malia_Obama* is not present in KB, our model can still make correct prediction thanks to $\boldsymbol{M}_{se}(\boldsymbol{e}_{Barack\_Obama} + \boldsymbol{r}_{Children}) \approx \boldsymbol{e}_{Malia\_Obama}$.

[4]Constraint from KB means that two entities and a relation form a triple in KB, as (e_s, r, e_o).

3.8 Dealing with Conjunctive Questions

IRN is not limited to only path questions. For a conjunctive question that contains more than one topic entity, the answer can be found by executing multiple IRNs with the same parameters in parallel and then obtaining the intersection of individual results.

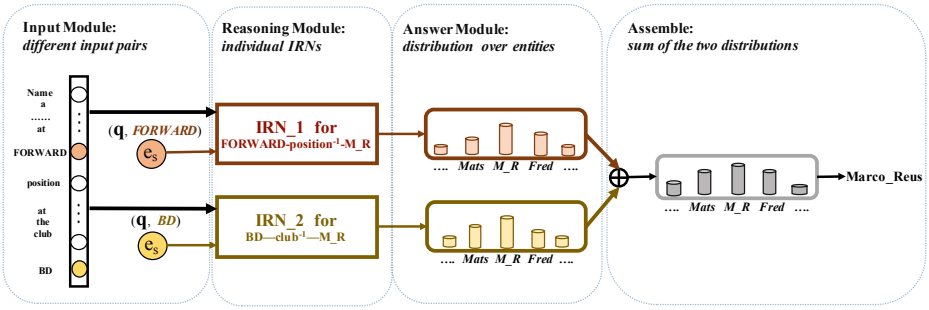


Figure 2: An assembly of two IRNs to handle a conjunctive question with two subjects. Different IRNs take as input the same question but different subjects and output the distribution over the candidate answers. The final answer is selected after summing the two distributions.

This process is exemplified by Figure 2. The input question *"Name a soccer player who plays at forward position at the club Borussia Dortmund"* has two subject entities, *"FORWARD"* and *"Borussia_Dortmund(BD)"*. One IRN (IRN_1) takes the original question and *"FORWARD"* as input, and then predicts possible objects for path query *"FORWARD* $\xrightarrow{plays\_position^{-1}}$ *?(Marco_Reus)"*.[5] The output is a distribution over entities. Similarly, another IRN (IRN_2) tackles the path query *"BD* $\xrightarrow{plays\_in\_club^{-1}}$ *?(Marco_Reus)"* where the input is the same question but another subject entity *"Borussia Dortmund(BD)"*. After summing the two output distributions, the answer *"Marco_Reus"* is chosen with the largest probability.

4 Data Preparation

We prepared two KBQA datasets to evaluate our Interpretable Reasoning Network: one is PathQuestion[6], constructed by ourselves, and the other is WorldCup2014, adopted from (Zhang et al., 2016).

4.1 PathQuestion

We adopted two subsets of Freebase (Bollacker et al., 2008) as Knowledge Bases to construct the PathQuestion (PQ) and the PathQuestion-Large (PQL) datasets. We extracted paths between two entities which span two hops ($e_s \xrightarrow{r_1} e_1 \xrightarrow{r_2} a$, denoted by -2H) or three hops ($e_s \xrightarrow{r_1} e_1 \xrightarrow{r_2} e_2 \xrightarrow{r_3} a$, denoted by -3H) and then generated natural language questions with templates. To make the generated questions analogical to real-world questions, we included paraphrasing templates and synonyms for relations by searching the Internet and two real-world datasets, WebQuestions (Berant et al., 2013) and WikiAnswers (Fader et al., 2013). In this way, the syntactic structure and surface wording of the generated questions have been greatly enriched.

PQL is more challenging than PQ in that PQL utilizes larger KB and provides less training instances. The statistics are shown in Table 1 and more details are described in the Appendix 6.

4.2 WorldCup2014

We also evaluated our model on the WorldCup2014 (WC2014) dataset constructed by (Zhang et al., 2016). The dataset contains single-relation questions (denoted by WC-1H), two-hop path questions (WC-2H), and conjunctive questions (WC-C). WC-M is the mixture of WC-1H and WC-2H. The statistics of *WorldCup2014* are listed in Table 1.

[5]The question mark indicates the entity to be predicted and the entity in bracket is the expected answer.

[6]This dataset is available at `https://github.com/zmtkeke/IRN`

Dataset	#Entity	#Relation	#Question	Exemplar Question
PQ-2H / 3H	2,215	14	1,908 / 5,198	What does the son of princess_Sophia's mom do for a living?
PQL-2H / 3H	5,035	364	1,594 / 1,031	What is the notable type of Jody_Harris's profession?
WC-1H / 2H / M / C	1,127	6	6,482 / 1,472 / 7,954 / 2,208	Name a player who plays at Forward position from Mexico?

Table 1: Statistics and exemplar questions of PathQuestion (PQ), PathQuestion-Large (PQL) and WorldCup2014 (WC).

5 Experiment and Evaluation

5.1 Implementation Details

ADAM optimizer (Kingma and Ba, 2015) was used for parameter optimization. The dimension of all the embeddings (words in question, entities and relations in KB) was set as $d_x = d_e = d_r = 50$. The hyper-parameter λ (see Eq. 8) is set to 1 . We partitioned the entire dataset into the train/valid/test subset with a proportion of $8:1:1$ and set the batch size as 50.

5.2 Performance of Question Answering

In this section, we evaluated the performance of multi-relation question answering on **PathQuestion** and **WorldCup2014** respectively. To further show that IRN is able to handle more challenging datasets, we evaluated the model with two configurations:

Incomplete KB To simulate the real KBs which are often far from complete, we removed half of the triples (entities and relations are retained but the connections between entities were removed) from the KB of the PQ-2H dataset.

Unseen KB To simulate a real QA scenario where out-of-vocabulary(OOV) words is one of the major challenges, we removed questions whose answer path includes relation *"Cause_of_Death"*, *"Gender"* or *"Profession"* from the PQ-2H training set. The models need to cope with questions related to these three OOV relations during the test.

Several baselines are included here:

Embed (Bordes et al., 2014b) deals with factoid QA over KB by matching a question with an answer in the embedding spaces.

Subgraph (Bordes et al., 2014a) improves the *Embed* model by enriching the representation of an answer entity with the entity's subgraph.

Seq2Seq (Sutskever et al., 2014) is a simplified seq2seq semantic parsing model, which adopts an LSTM to encode the input question sequence and another LSTM to decode the answer path.

MemN2N (Sukhbaatar et al., 2015) is an end-to-end memory network that can be used for reading comprehension and question answering. The memory units consist of the related triples in a local subgraph of the corresponding answer path, where the settings are the same as (Bordes et al., 2015).

KVMemN2N (Miller et al., 2016) improves the MemN2N for KBQA as it divides the memory into two parts: the key memory stores the head entity and relation while the value memory stores the tail entity.

IRN-weak is our model that employs only supervision from the final answer entity rather than the complete answer path. This can be implemented by simply ignoring the loss from the intermediate hops except the final entity in Eq. 8.

The performance is measured by accuracy[7]: correct if a predicted entity is in the answer set of input question. Since there are many 1-to-many relations in Freebase and WC2014, a question may have several possible answer paths, resulting in multiple answers. For example, given the question *"How old is Obama's daughter?"*, the original path can be *"Barack_Obama* $\xrightarrow{Children}$ *Malia_Obama* $\xrightarrow{Age}$*18"* or *"Barack_Obama* $\xrightarrow{Children}$*Sasha_Obama* $\xrightarrow{Age}$ *14"*, thus the answer can be either *"18"* or *"14"*. For this question, either answer is correct.

The results in Table 2 demonstrate that our system outperforms the baselines on single-relation questions (WC-1H), 2-hop-relation questions (PQ-2H/PQL-2H/WC-2H) as well as 3-hop-relation questions (PQ-3H/PQL-3H). Furthermore, assembled IRNs obtain strong performance when dealing with conjunctive questions in WC-C .

[7]Reported accuracy is the average accuracy of five repeated runs.

	PathQuestion		PathQuestion Large		WorldCup2014				Challenging PQ-2H	
	PQ-2H	PQ-3H	PQL-2H	PQL-3H	WC-1H	WC-2H	WC-M	WC-C	Incomplete	Unseen
Random	0.151	0.104	0.021	0.015	0.085	0.064	0.053	0.073	-	-
Embed	0.787	0.483	0.425	0.225	0.448	0.588	0.518	0.642	-	-
Subgraph	0.744	0.506	0.500	0.213	0.448	0.507	0.513	0.692	-	-
IRN-weak(*Ours*)	0.919	0.833	0.630	0.618	0.749	0.921	0.786	0.837	-	-
Seq2Seq	0.899	0.770	0.719	0.647	0.537	0.548	0.538	0.577	-	-
MemN2N	0.930	0.845	0.690	0.617	0.854	0.915	**0.907**	0.733	0.899($\downarrow 3.3\%$)	0.558
KVMemN2N	0.937	**0.879**	0.722	0.674	**0.870**	0.928	0.905	0.788	0.902($\downarrow 3.7\%$)	0.554
IRN(*Ours*)	**0.960**	0.877	**0.725**	**0.710**	0.843	**0.981**	**0.907**	**0.910**	0.937($\downarrow 2.3\%$)	0.550

Table 2: Accuracy on different QA datasets. WC-C is for conjunctive questions while other datasets for path questions. Challenging PQ-2H are two more difficult configurations of PQ-2H. The models in the second block utilize the answer path information but those in the first block do not.

We have further observations as follows:

• *IRN-weak* outperforms *Embed* and *Subgraph*, indicating that multi-hop reasoning indeed helps to answer complex questions even when our model is trained end-to-end in the same configuration of weak supervision[8].

• The *Seq2Seq* baseline performs worse than *IRN*. Though they are both interpretable, *IRN* is more powerful when dealing with complicated KBs and questions.

• *IRN* is better than *MemN2N* and *KVMemN2N* on most of the datasets, and both models are much better than other baselines using the path information. Note that the memory in *(KV)MemN2N* consists of fact triples which are distilled from KB using answer path. In this sense, *(KV)MemN2N* indirectly employs strong supervision from answer path. In contrast, *IRN* has a better (or easier) mechanism to supervise the reasoning process thanks to its interpretable framework.

• The highest accuracy on PQL-2H/3H reveals that *IRN* performs better when faced with larger datasets. IRN deals with relations and entities separately, where the number of entities and relations is much less than that of triples. However, *(KV)MemN2N* has to handle much more triples in its memory.

• *IRN* is more robust than the baselines ($\downarrow 3.7\%$ vs. $\downarrow 2.3\%$) when dealing with incomplete KB, which is probably because auxiliary KB embedding training facilitates the prediction of missing triples. While the baselines are more sensitive to the incomplete information stored in the memory units.

• Both *IRN* and the baselines degrade remarkably (0.9→0.5) in the unseen setting because wrong distributed representations are influential in embedding-based QA models. In addition, the size of the training set is much smaller than that of the original PQ-2H, which also leads to worse performance.

• *IRN* is more interpretable compared with *(KV)MemN2N* , attributed to the structure of *IRN*. The relation/entity predicted at each hop is a part of the answer path. The intermediate outputs offer the possibility to trace the complete reasoning process, diagnose failures, and manipulate answer prediction through intermediate interactions (see Section 5.3).

5.3 Interpretable Path Reasoning

In this section, we demonstrated how IRN is interpretable by both quantitative and qualitative analysis. For the quantitative analysis, we can measure how it performs during the reasoning process by investigating the prediction accuracy of intermediate relations and entities. In this task, we collected all the relations and entities with largest probabilities (see Eq. 3 and Eq. 7) at each hop $\{r^1, a^1, r^2, .., a^H\}$ and compared these intermediate outputs with the ground truth $\{r_1, e_1, r_2, .., a\}$. As for KVMemN2N, we also fetched an entity at each hop by an answer distribution, similar to that at the final hop.

According to the structure of IRN, the relation/entity predicted at each hop constitutes an answer path. Results[9] in Table 3 indicate that IRN can predict intermediate entities more accurately than final answers, due to the cascading errors in the consecutive prediction.

Though KVMemN2N (KVM) predicts the exact answers well, it lacks interpretability. On the one hand, KVM can not predict relations to trace the answer path. On the other hand, the hops in KVM all

[8]Only supervision from question-answer pairs, but without answer path information from KB.

[9]Note here that only if an output matches the labeled entity exactly, the prediction will be judged as correct.Thus, the accuracy here has a different definition from that in Table 2.

	r_1		e_1		r_2		e_2		r_3		a	
	IRN	KVM	IRN	KVM	IRN	KVM	IRN	KVM	IRN	KVM	IRN	KVM
PQ-2H	1.000	NA	0.957	0.016	1.000	NA	NA	NA	NA	NA	0.934	0.916
PQL-2H	0.968	NA	0.722	0.083	0.836	NA	NA	NA	NA	NA	0.673	0.676
WC-2H	1.000	NA	0.531	0.000	1.000	NA	NA	NA	NA	NA	0.528	0.382
PQ-3H	1.000	NA	0.883	0.003	1.000	NA	0.772	0.001	1.000	NA	0.738	0.774
PQL-3H	0.808	NA	0.721	0.019	0.702	NA	0.721	0.007	0.683	NA	0.608	0.600

Table 3: Accuracy at different hops along the answer path from IRN and KVMemN2N (KVM). r_i indicates the relation at hop i, e_i indicates the entity at hop i. a indicates the final answer. NA means not applicable.

aim at finding the answer entity rather than the intermediate entities along the answer path.

To illustrate how our model parses a question and predicts relations hop-by-hop, we studied the distributions over all the relations ($\boldsymbol{g}^h$, see Eq. 3) and chose an example from PathQuestion as shown in Figure 3. It is clear that IRN is able to derive the relations in correct order. For question *"What does john_hays_hammond's kid do for a living?"*, IRN first detects relation *Children* (the corresponding word/phrase in the question is *kid*) and then *Profession* (*what does..do*). When detecting the *Terminal* relation, IRN will stop the analysis process.

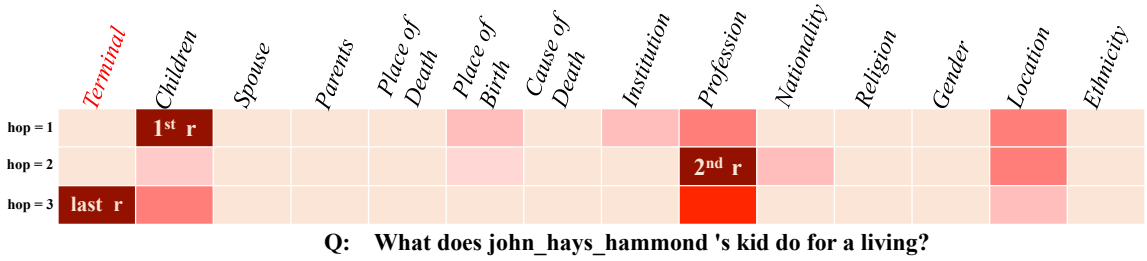


Figure 3: The predicted relations at each hop. Each row represents a probability distribution over relations. Darker color indicates larger probability. The terminal relation is highlighted in red.

Our model can map relations in KB to words in question. We sampled some relations in KB and projected them to the question space by $\boldsymbol{r}^q = \boldsymbol{M}_{rq}\boldsymbol{r}$ (see Eq. 2). We then obtained words whose embeddings are most close to $\boldsymbol{r}^q$ measured by cosine similarity $cos(\boldsymbol{r}^q, \boldsymbol{x}_i)$. The result in Table 4 indicates that IRN can establish reasonable mapping between KB relation and natural language, such as linking *Profession* to words like *working, profession*, and *occupation* . Besides mapping to single words, relation in KB can be associated with some complicate templates, such as *Profession* $\rightarrow$ *"what does ... do"* .

Relation	Similar words in natural language questions
Profession	profession, do, working, occupation
Insititution	institution, organization, work, where
Religion	faith, religion, what, belief
Cause_of_Death	died, killed, how, death
Place_of_Birth	hometown, born, city, birthplace

Table 4: Most similar words in questions for some exemplar relations.

The above analysis demonstrates that our model is interpretable. Specifically, IRN has merits at:
•Providing a traceable reasoning path for question answering. With the aid of these intermediate entities and relations, we can obtain not only the final answer but also the complete path that infers the answer.

•Facilitating failure diagnosis. For instance, IRN fails to answer the question *"Where did the child of Joseph_P_Kennedy_Sr die ?"*. The true answer path should be *"Joseph_P_Kennedy_Sr* $\xrightarrow{Children}$ *Patricia_kennedy_Lawford* $\xrightarrow{Place\_of\_Death}$ *New_York_County"*. However, the middle entity decided by IRN is *"Rosemary_Kennedy"* who is also a child of *"Joseph_P_Kennedy_Sr"*, but her death is not included in KB.

•Allowing manual manipulation in answer prediction. We updated the state (Eq. 5) and the question (Eq. 2) in IRN with the ground-truth relation vectors and compared the performance. The higher

accuracy in Table 5 implies that we can improve the final prediction by correcting intermediate predictions.

Dataset	PQ-2H	PQ-3H	PQL-2H	PQL-3H
Acc	0.980 (↑ 2.0%)	0.900 (↑ 2.3%)	0.755 (↑ 3.0%)	0.744 (↑ 3.4%)

Table 5: Accuracy when the intermediate predictions are replaced by ground truth.

6 Conclusion

We present a novel Interpretable Reasoning Network which is able to make reasoning hop-by-hop and then answer multi-relation questions. Our model is interpretable in that the intermediate predictions of entities and relations are traceable and the complete reasoning path is observable. This property enables our model to facilitate reasoning analysis, failure diagnosis, and manual manipulation in answer prediction. Results on two QA datasets demonstrate the effectiveness of the model on multi-relation question answering.

As future work, there is much room for complex question answering. For instance, answering *"How old is Obama's younger daughter?"* needs to handle arithmetic operation. Furthermore, multi-constraint questions will also be considered in this framework.

Acknowledgements

This work was partly supported by the National Science Foundation of China under grant No.61272227/61332007 and the National Basic Research Program (973 Program) under grant No. 2013CB329403.

References

Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum. 2017. Quint: Interpretable question answering over knowledge bases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 61–66. Association for Computational Linguistics.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Klein Dan. 2015. Neural module networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. *NAACL*, pages 1545–1554.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on freebase from question-answer pairs. *Empirical Methods in Natural Language Processing*, pages 1533–1544.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, Bc, Canada, June*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Annual Conference on Neural Information Processing Systems*, pages 2787–2795.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014a. Question answering with subgraph embeddings. *Empirical Methods in Natural Language Processing*, pages 615–620.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014b. Open question answering with weakly supervised embedding models. *In Proceedings of ECML-PKDD*, pages 165–180.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075.

Asli Celikyilmaz, Li Deng, Lihong Li, and Chong Wang. 2017. Scaffolding networks for teaching and learning to comprehend.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*, pages 260–269.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Meeting of the Association for Computational Linguistics*, pages 1608–1618.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. *Empirical Methods in Natural Language Processing*, pages 318–327.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 221–231. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *international conference on learning representations*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *International Conference on Machine Learning*, pages 1378–1387.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Empirical Methods in Natural Language Processing*, pages 1400–1409.

Lili Mou, Zhengdong Lu, Hang Li, and Zhi Jin. 2017. Coupling distributed and symbolic execution for natural language queries. In *Proceedings of the 34th International Conference on Machine Learning, Sydney, NSW, Australia, 6-11 August*, pages 2518–2526.

Arvind Neelakantan, Quoc V Le, and Ilya Sutskever. 2015. Neural programmer: Inducing latent programs with gradient descent. *International Conference on Learning Representations*.

Arvind Neelakantan, Quoc V Le, and Ilya Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. *international conference on learning representations*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *Association for Computational Linguistics*, pages 1470–1480.

Denis Savenkov and Eugene Agichtein. 2017. Evinets: Neural networks for combining evidence signals for factoid question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 299–304. Association for Computational Linguistics.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2016. Reasonet: Learning to stop reading in machine comprehension. *SIGKDD*, pages 1047–1055.

R. Socher, D. Chen, C. D. Manning, and A. Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *International Conference on Intelligent Control & Information Processing*, pages 926–934.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. *Annual Conference on Neural Information Processing Systems*, pages 2440–2448.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Annual Conference on Neural Information Processing Systems*, 4:3104–3112.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198. Association for Computational Linguistics.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. *International Conference on Learning Representations*.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2326–2336.

Semih Yavuz, Izzeddin Gur, Yu Su, and Xifeng Yan. 2017. Recovering question answering errors via query revision. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 903–909. Association for Computational Linguistics.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of Association for Computational Linguistics*, pages 643–648.

Wen Tau Yih, Matthew Richardson, Chris Meek, Ming Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Meeting of the Association for Computational Linguistics*, pages 201–206.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2015. Neural enquirer: Learning to query tables with natural language. *Association for Computational Linguistics*, pages 2308–2314.

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schtze. 2016. Simple question answering by attentive convolutional neural network. *International Conference on Computational Linguistics*, pages 1746–1756.

Liwen Zhang, John Winn, and Ryota Tomioka. 2016. Gaussian attention model and its application to knowledge base embedding and question answering. *CoRR*, abs/1611.02266.

Appendix A. PathQuestion Construction and Question Templates

We constructed a synthesis dataset by generating questions with templates. The knowledge base for PathQuestion has more than 60,000 triples which are adopted from FB13 (Socher et al., 2013) with 13 relations and thousands of entities. As for PathQuestion-Large, we adopted another more complex subset of Freebase (Bollacker et al., 2008). **First**, we extracted all the paths with two hops ($< e_s, r_1, e_1, r_2, a >$), or three hops ($< e_s, r_1, e_1, r_2, e_2, r_3, a >$) among these triples. **Second**, we crafted templates to generate natural language questions from these paths. **Last**, we collected question and answer path pairs $(q, < e_s, r_1, e_1, ..., a >)$ to construct the *PathQuestion (PQ)* dataset.

We crafted templates to transfer an answer path extracted from KB to natural language questions. To make the generated questions analogical to real-world questions, those templates are firstly written manually, and then enriched by replacing synonyms. Besides, we searched for different syntactical structures and paraphrases in real-world datasets including WebQuestions (Berant et al., 2013) and WikiAnswers (Fader et al., 2013) as well as on the Internet. In this manner, the templates have been greatly diversified and are much closer to real questions.

Synonyms used in templates for PathQuestion are shown in Table 6 and templates for 2-hop paths (PQ-2H) are shown in Table 7. The datasets are available at `https://github.com/zmtkeke/IRN`.

Relation	**Synonyms**
Spouse	couple, wife, husband
	other half, darling
Children	child, offspring, kid
	daughter, son, heir
Parents	parent, father, mother
	dad, mom
Profession	job, occupation, work
Institution	organization
	educational institution
Ethnicity	race
Gender	sex
Nationality	nation, country
Location	address
Religion	faith, religious belief
	type of religion

Table 6: Natural language synonyms for relations appearing in questions in PathQuestion.

Path Pattern	**Question-templates**
Universal	"What is the r_2 of e_s's r_1?"
	"What is the e_s's r_1's r_2?"
	"What is the r_2 of r_1 of e_s?"
	"The r_2 of e_s's r_1?"
	"The e_s's r_1's r_2?"
	"The r_2 of r_1 of e_s?"
Ask about a person	"Who is the r_2 of e_s's r_1?"
	"What is the name of the r_2 of e_s's r_1?"
r_2=r_1=Parents/ r_2=r_1=Children	"Who is the grand-r_1 of e_s?"
	"What is the name of the grand-r_1 of e_s?"
r_2=Ethnicity	"What r_2 is e_s's r_1"
	"What is e_s's r_1's r_2 like?"
	"What is e_s's r_1's r_2 about?"
r_2=Institution	"Where does e_s's r_1 work?"
	"Where does e_s's r_1 work for?"
	"Which r_2 does e_s's r_1 work for?"
r_2=Nationality	"Which nationality is e_s's r_1?"
	"Where does e_s's r_1 come from?"
r_2=Religion	"What r_2 does e_s's r_1 follow?"
	"What r_2 is e_s's r_1?"
	"What r_2 does e_s's r_1 have?"
	"What r_2 is e_s's r_1 practice?"
r_2=Gender	"What r_2 is e_s's r_1?"
	"Is e_s's r_1 a man or a woman?"
r_2=Location	"Where is e_s's r_1 living?"
	"Where is e_s's r_1 staying?"
	"Please tell me e_s's r_1 present address."
r_2=Profession	"What does e_s's r_1 do?"
	"What is e_s's r_1 working on?"
	"What is e_s's r_1?"
	"What line of business is e_s's r_1 in?"
	"What does e_s's r_1 do for a living?"
r_2=Cause_of_Death	"Why e_s's r_1 died?"
	"How e_s died?"
	"What's the reason of e_s's r_1's death?"
	"What caused the death of e_s's r_1?"
	"What killed the e_s's r_1?"
	"What made the e_s's r_1 dead?"
	"What did e_s's r_1 die from?"
r_2=Place_of_Death	"Where did e_s's r_1 die?"
	"Where did the r_1 of e_s die?"
	"What city did e_s's r_1 die?"
r_2=Place_of_Birth	"Where did e_s's r_1 born?"
	"What city did e_s's r_1 born?"
	"What is the hometown of e_s's r_1?"
	"What is e_s's r_1's birthplace?"

Table 7: Templates for generating natural language questions from answer paths in PQ-2H.

Task-oriented Word Embedding for Text Classification

Qian Liu[1,2,3], **Heyan Huang**[1,2]*, **Yang Gao**[1,2], **Xiaochi Wei**[1,2], **Yuxin Tian**[1,2], **Luyang Liu**[1,2]
1. Department of Computer Science, Beijing Institute of Technology, China
2. Beijing Engineering Research Center of High Volume Language Information Processing and Cloud Computing Applications, China
3. Centre for Artificial Intelligence, University of Technology Sydney, Australia

Abstract

Distributed word representation plays a pivotal role in various natural language processing tasks. In spite of its success, most existing methods only consider contextual information, which is suboptimal when used in various tasks due to a lack of task-specific features. The rational word embeddings should have the ability to capture both the semantic features and task-specific features of words. In this paper, we propose a task-oriented word embedding method and apply it to the text classification task. With the function-aware component, our method regularizes the distribution of words to enable the embedding space to have a clear classification boundary. We evaluate our method using five text classification datasets. The experiment results show that our method significantly outperforms the state-of-the-art methods.

1 Introduction

Learning word representation is a fundamental step in various natural language processing tasks. Tremendous advances have been made by distributed representations (also known as word embeddings) which learn a transformation of each word from raw text data to a dense, lower-dimensional vector space. Most existing methods leverage contextual information from the corpus (Mikolov et al., 2013; Pennington et al., 2014) and other complementary information, such as subword information (Cao and Lu, 2017), implicitly syntactic dependencies (Shen et al., 2018a; Shen et al., 2018b), and semantic relations (Bollegala et al., 2016; Liu et al., 2018).

In traditional evaluations such as word similarity and word analogy, the aforementioned context-aware word embeddings work well since semantic information plays a vital role in these tasks, and this information is naturally addressed by word contexts. However, in real-world applications, such as text classification and information retrieval, word contexts alone are insufficient to achieve success in the absence of task-specific features. Figure 1 illustrates this problem with the classification task as an example. Several sentences from different categories are given at the far left of the figure where the words in bold are salient words for the category distinction. We also illustrated expected word distribution of these salient words in the embedding space. To obtain a good classification performance, the expected word distribution should have a clear classification boundary: words within the same category are close to each other and far away from words in other categories as illustrated in Figure 1. However, the actual distribution obtained from Word2Vec at the far right of Figure 1 is normally not satisfactory because Word2Vec only focuses on context similarity. For example, although *learning* and *educational* are with similar context as recognized by Word2Vec, they are salient words to distinguish categories of *AI* and *Sociology*, so they should be far away from each other. Apparently, using word embedding directly from Word2Vec would not obtain good performance on the text classification task due to the fact that words' functional features in the real tasks are ignored in the training process.

In this paper, we propose a task-oriented word embedding method (denoted as ToWE) to solve the aforementioned problem. It learns the distributed representation of words according to the given specific

*Corresponding author

Proceedings of the 27th International Conference on Computational Linguistics, pages 2023–2032
Santa Fe, New Mexico, USA, August 20-26, 2018.

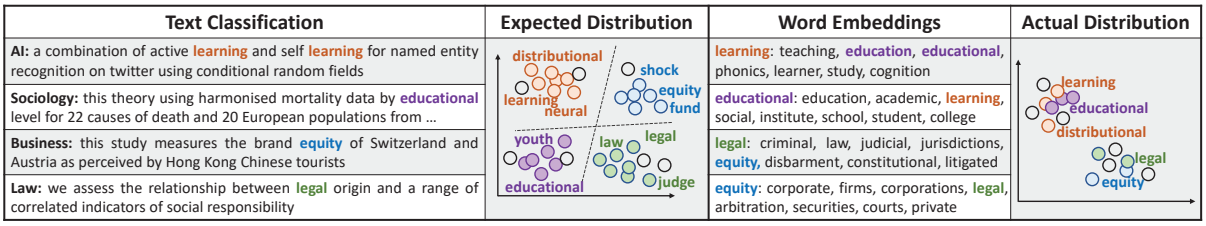


Figure 1: The example sentences from the text classification dataset. Words in bold are salient words to distinguish the sentence category. Their most similar words in the Word2Vec space are shown in the right-hand column. The word color indicates the category, and the words in black are general words for the task.

NLP task. Specifically, we focus on text classification. In our method, the words' contextual information and task information are inherently jointed to construct the word embeddings. In the joint learning framework, the contextual information is captured following the context prediction task introduced by (Mikolov et al., 2013). To model the task information, we regularize the distribution of the salient words to have a clear classification boundary, and then adjust the distribution of the other words in the embedding space correspondingly. To give an intuitive understanding on how our method works from the classification perspective, we design a 5AbstractsGroup dataset (detailed in Section 4.1) and conduct a qualitative analysis. Experiments show qualitative improvements of our method over context-based Skip-gram method on word neighbors for classification. We also perform empirical comparisons on five text classification datasets, which demonstrate the effectiveness of our method over the other state-of-the-art methods.

The contributions of this paper can be summarized as following:

- We propose a task-oriented word embedding method that is specially designed for text classification. It introduces the function-aware component and highlights word's functional attributes in the embedding space by regularizing the distribution of words to have a clear classification boundary.
- We design a 5AbstractsGroup dataset and present a qualitative analysis, giving an intuitive understanding on how our method works from the classification perspective. Experimental results on five text classification datasets also show that the proposed method is more optimal for classification on account of revealing functional attributes of words.

2 Related Work

Word embeddings that provide continuous low-dimensional vector representations of words have been widely studied by NLP communities (Yu et al., 2017; Liu et al., 2017; Li et al., 2017b; Chih et al., 2017). The last few years have seen the development of word embedding methods purely based on the co-occurrence information in a corpus (Bengio et al., 2003; Mnih and Hinton, 2008; Collobert et al., 2011; Mikolov et al., 2013; Mnih and Kavukcuoglu, 2013; Lebret and Collobert, 2014; Pennington et al., 2014; Cao and Lu, 2017; Bollegala et al., 2018). Some studies also pay attention to the semantic knowledge stored in the knowledge bases (Nie et al., 2015). For example, Faruqui et al. (2015) refine word representations using relational information from semantic lexicons, Liu et al. (2015b) represent semantic knowledge as a number of ordinal similarity inequalities of related word pairs to learn semantic word embeddings.

Recent works have thrown light on the problems associated with directly applying word embeddings into real-world applications. Diaz et al. (2016) demonstrated that the globally trained word embedding underperform corpus and query-specific embeddings for retrieval tasks. They proposed locally training word embeddings in a query-specific manner for the query expansion task. Zamani and Croft (2017) indicated that the underlying assumption in typical word embedding methods is not equal to the need of IR tasks, and they proposed relevance-based models to learn word representations based on query-document relevance information, which is the primary objective of most IR task. For the sentiment

analysis task, Yu et al.(2017) refined word embedding to avoid generating similar vector representations for sentimentally opposite words. For the contradiction detection task, Li et al. (2017a) developed contradiction-specific word embedding to recognize contradiction relations between a pair of sentences. These studies show that general trained word embeddings cannot be optimized for a specific task, thus, they are likely to be suboptimal. To meet the needs of real-world applications, rational word embeddings should have the ability to capture both the semantics of words and the task-specific features of words.

In this work, we focus on task-oriented word embedding for the text classification task. Several attempts have shown that revised word embeddings can boost the performance of classification. For example, topical information is shown to be effective in generating high quality word embeddings (Liu et al., 2015c; Liu et al., 2015a), which can enhance the performance of text classification. On the other hand, to enhance the performance of sentiment classification, Chih et al.(2017) proposed a word embedding refinement model to refine existing semantically oriented word vectors using sentiment lexicons in order to distinguish words with similar vector representations but opposite sentiment polarities. Our work departs from previous work in that it directly models task-specific features to construct the embedding space with a clear boundary for classification.

3 Method

Given the unlabeled corpus C and the labeled training set $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_g\}$ with g text categories, our method aims to train the task-oriented d-dimensional word embedding $\mathbf{w_i} \in \mathcal{R}^d$ for the i-th word w_i in vocabulary $\mathcal{V}$. Formally, the document collection belonging to the k-th category is denoted as $\mathcal{D}_k$. Our proposed joint learning framework contains two components, i.e., the context-aware component and function-aware component. The context-aware part models the co-occurrence in corpus C and captures the word semantic features. The function-aware part reveals the word's functional attributes following the task-specific features observed in $\mathcal{D}$. We next describe these two parts respectively.

3.1 Context-aware Component

Our method uses the Word2Vec method to model the context information and uses log-linear models to produce word embeddings. It applies a sliding window moving on the corpus. The word in the center of the window is the target word and the others are context words. Word2Vec has two versions, i.e., CBOW and Skip-gram. The CBOW model uses the average/sum of context words as input to predict the target, and the Skip-gram model uses the target word as input to predict each context word. To simplify, we represent the objective of each prediction as

$$\mathcal{L}_{context} = Pr(w|\mathbf{c}) = \frac{\exp(\mathbf{w} \cdot \mathbf{c})}{\sum_{w' \in \mathcal{V}} \exp(\mathbf{w'} \cdot \mathbf{c})}. \tag{1}$$

In CBOW, w is the target word, and $\mathbf{c}$ is the vector of the context words, and in Skip-gram, w is each word in the context, and $\mathbf{c}$ is the vector of the target word.

3.2 Function-aware Component

In the function-aware component, we define salient words as those words with the ability to distinguish the document category. These salient words are first extracted from the labeled training set $\mathcal{D}$ in an offline process. Then the correlations among these words are used to model the functional features in the embedding space.

Each salient word w of the k-th category is offline extracted according to the following the two principles: (1) The term frequency of the word w in this category (i.e., $\mathcal{D}_k$) is much higher than that in other categories; (2) w is common in other categories, expressed as a small variance of term frequencies in other categories. Formally, we design the following formula to measure the importance of word w to the k-th category as a salient word:

$$Score(w, k) = \frac{t_k - \frac{1}{g}\sum_{1 \le i \le g} t_i}{var(T_{-k}(w))}, \tag{2}$$

where t_i is the term frequency in the i-th category, $T_{-k}(w)$ is the collection of term frequencies except the k-th category (i.e., $T_{-k}(w) = \{t_j | 1 \leq j \leq g, j \neq k\}$), and $var(\cdot)$ is the variance.

According to this importance score, we generate a salience words set by selecting the top N words for each category, denoted as $S_k = \{w_j | 1 \leq j \leq N\}, k \in [1, g]$. Then, for the task, the words in the salient words set $S = \{S_1, S_2, \cdots, S_g\}$ have the ability to distinguish different categories.

The salient words are next utilized to capture the functional relations between words in the embedding space. In the learning framework, if the predicted word w is in S, the function-aware component will be activated. As to modeling the correlations of function-salient words, we expect to constrain w to be close to the words in the same category and far away from the words in different categories. According to this idea, we construct a set $P(w)$ with n word-pairs for each salient word w. Each word-pair contains a positive word u and a negative word v. The positive words are randomly selected from S which belong to the same category with w, and the negative words are randomly sampled from other categories. We maximize a margin-based ranking criterion over the training set S:

$$\mathcal{L}_{function} = argmax \sum_{<u,v> \in P(w)}^{n} [\gamma + s(\mathbf{w}, \mathbf{u}) - s(\mathbf{w}, \mathbf{v})], \tag{3}$$

where γ is a margin hyper parameter, n is the size of sample set $P(w)$, and $s(\cdot, \cdot)$ is similarity measure. Following the recommendations in prior work on word similarity measurement, we apply the cosine similarity of a pair of words by computing $s(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}| \cdot |\mathbf{b}|}$. The objective function favors higher values of the similarity for positive word-pairs than for negative word-pairs, and is thus a natural implementation of the intended criterion.

3.3 Joint Learning

The context-aware component and the function-aware component are jointly optimized, so we then obtain the following object function:

$$\mathcal{L} = \underset{\Theta}{argmax} \quad \lambda \mathcal{L}_{context} + (1 - \lambda) \mathcal{L}_{function}, \tag{4}$$

where Θ is a set of all parameters in $\mathcal{L}_{context}$ and $\mathcal{L}_{function}$, and λ is the combination parameter which balances the contribution of each component in the training process.

The goal of the training objective is to maximize $\mathcal{L}$ with respect to the model parameters. The optimization process is conducted via Stochastic Gradient Descent (SGD). The optimization of $\mathcal{L}_{context}$ follows the negative sampling introduced in (Mikolov et al., 2013). If the predicted word w is in the salient words set S, the corresponding optimization process for $\mathcal{L}_{function}$ will be activated, and the parameters are updated as $\mathbf{w} \leftarrow \mathbf{w} + \eta \frac{\partial \mathcal{L}}{\partial \mathbf{w}}$, $\mathbf{u} \leftarrow \mathbf{u} + \eta \frac{\partial \mathcal{L}}{\partial \mathbf{u}}$, and $\mathbf{v} \leftarrow \mathbf{v} + \eta \frac{\partial \mathcal{L}}{\partial \mathbf{v}}$, where η is the learning rate, and the gradients are calculated as follows:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{w}} &= \lambda \sum_{<u,v> \in P(w)}^{n} \frac{\partial s(\mathbf{w}, \mathbf{u})}{\partial \mathbf{w}} - \frac{\partial s(\mathbf{w}, \mathbf{v})}{\partial \mathbf{w}}, \\
\frac{\partial \mathcal{L}}{\partial \mathbf{u}} &= \lambda \sum_{<u,v> \in P(w)}^{n} \frac{\partial s(\mathbf{w}, \mathbf{u})}{\partial \mathbf{u}}, \\
\frac{\partial \mathcal{L}}{\partial \mathbf{v}} &= \lambda \sum_{<u,v> \in P(w)}^{n} - \frac{\partial s(\mathbf{w}, \mathbf{v})}{\partial \mathbf{v}},
\end{aligned} \tag{5}$$

where w is the predicted word, u is its positive word and v is its negative word. Since we apply cosine distance to compute the similarity between two words, the optimization can be derived as follows:

$$\frac{\partial s(\mathbf{a}, \mathbf{b})}{\partial \mathbf{a}} = -\frac{S_{a,b} \cdot \mathbf{a}}{|\mathbf{a}|^2} + \frac{\mathbf{b}}{|\mathbf{a}| \cdot |\mathbf{b}|}, \tag{6}$$

Algorithm 1 Task-oriented Word Embedding Method.

Input: Corpus C, the labeled training set $\mathcal{D}$ with g categories, dimensionality d, sampling times n, and word vocabulary $\mathcal{V}$

Output: Embeddings $\mathbf{w} \in \mathcal{R}^d$ of all words in the vocabulary $\mathcal{V}$.

Initialization: randomly set $\mathbf{w} \in \mathcal{R}^d$ for all words in $\mathcal{V}$; generate the salient words set S; constructing T prediction tasks using a sliding window.

for t = 1, 2, . . . , T **do**
  optimize $\mathcal{L}_{context}$ using negative sample method introduced in (Mikolov et al., 2013)
  if w in S **then**
    for n **do**
      sampling the positive word u and the negative word v.
      optimize $\mathcal{L}_{function}$ using Eq.(5) to update $\mathbf{w}$, $\mathbf{u}$, $\mathbf{v}$.
    end for
  end if
end for
return $\mathbf{w}$ for all words in $\mathcal{V}$.

where $S_{a,b} = \frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}|\cdot|\mathbf{b}|}$. The pseudo code for our word embedding method is shown in Algorithm 1, and the source code is available on the Github[1].

4 Experiments

4.1 Datasets

To undertake an extensive evaluation, we investigate the empirical performances of our proposed method on five text classification datasets. The detailed statistics of all the datasets are listed in Table 1. Each dataset is briefly described as follows:

Datasets	Type	Train Size	Test Size	#Classes	Avg.L	Vocab Size	#Tokens
20NewsGroup	Doc.	11,314	7,532	20	315	179,092	6,555,230
5AbstractsGroup	Doc.	2,500	3,756	5	223	38,103	1,203,022
IMDB	Doc.	25,000	25,000	2	126	170,543	6,141,136
MR	Sen.	32,361	32,359	2	21	47,568	974,626
SST	Sen.	5,928	5,927	2	12	19,362	152,474

Table 1: Statistics of the five mainstream datasets for text classification.

(1) The **20NewsGroup**[2] is a popular text classification dataset which contains 18,846 documents from 20 different newsgroups. Each document contains several sentences. The dataset is separated into a training set of 11,314 documents and a test set of 7,532 documents. (2) The **5AbstractsGroup** dataset is academic papers from five different domains collected from the Web of Science namely, business, artificial intelligence, sociology, transport and law. We extracted the abstract and title fields of each paper as a document. The dataset contains 6,256 documents, and we randomly selected 500 papers in each category as the training set, and the others as the test set. The dataset is published on the Github[3]. (3) The **IMDB**[4] contains movie reviews with binary classes (i.e., positive and negative). It consists of 50,000 movie reviews (Maas et al., 2011), and each movie review has several sentences. (4) The **MR**[5] dataset consists of movie reviews from Rotten Tomato website with two classes labeled by (Pang and Lee, 2005). Each review contains only one sentence. (5) The **SST**[6] dataset contains the movie reviews

[1] https://github.com/qianliu0708/ToWE
[2] http://qwone.com/ jason/20Newsgroups/.
[3] https://github.com/qianliu0708/5AbstractsGroup
[4] https://www.cs.jhu.edu/mdredze/datasets/sentiment/unprocessed.tar.gz
[5] https://www.cs.cornell.edu/people/pabo/movie-review-data/
[6] http://nlp.stanford.edu/sentiment

Business	AI	Law	Sociology	Transport
employee	distributional	jurisdiction	educational	driver
entrepreneur	predefine	interpreted	sport	departure
stakeholder	inference	law	food	urban
trait	recognition	dispute	farmer	intersection
consumer	variant	qualified	sociology	accident
marketplace	analytic	congress	experience	incident
asset	learn	interfere	poverty	route
bond	aggregate	contract	religious	transferring
manager	object	victim	youth	passenger
markets	uncertain	permit	ethnicity	vehicle

Figure 2: Top ten salient words for each category in the 5AbstractsGroup dataset.

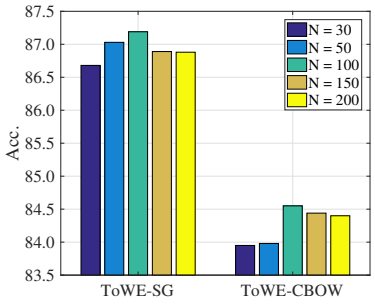
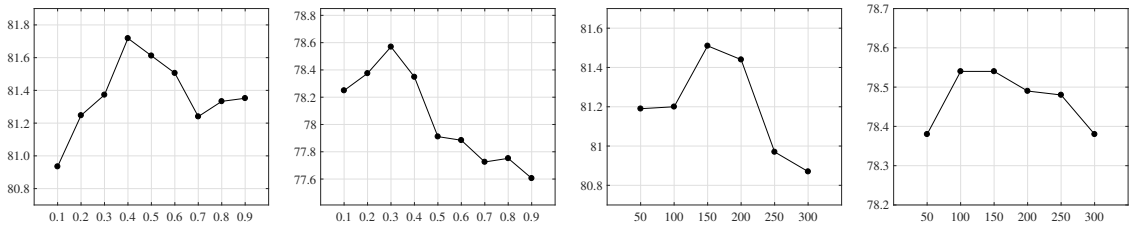


Figure 3: Performance of the ToWE method with varying N on the 5AbstractsGroup dataset.



(a) λ on the ToWE-SG (b) λ on the ToWE-CBOW (c) n on the ToWE-SG (d) n on the ToWE-CBOW

Figure 4: Performance of the ToWE method with the varying parameter λ and the size of sampling n. The Y-axis represents the accuracy (%) on the 20NewsGroup dataset.

in the Stanford Sentiment Treebank labeled by (Socher et al., 2013) comprising one sentence for each review. 50% of the MR and SST datasets are partitioned randomly into the training set and 50% into the test set.

4.2 Baseline Methods

To evaluate our method, we consider the following baselines: (1) the **BOW** method is employed as a basic baseline. It represents each document as a bag of words and the weighting scheme is TFIDF. We select the top 2,000 words according to the TFIDF scores as features; (2) the Word2Vec method is a neural network language method which learns word embeddings by maximizing the conditional probability leveraging contextual information. It comprises two models, i.e., **CBOW** which predicts the target word using context information, and the Skip-gram (denoted as **SG**) which predicts each context word using the target word; (3) the **GloVe** (Pennington et al., 2014) method is a state-of-the-art matrix factorization method. It leverages global count information aggregated from the entire corpus as word-word occurrence matrix to learn word embeddings; (4) the Topical Word Embedding method (denoted as **TWE**) (Liu et al., 2015c) learns a topic model from the training set, then generates word embeddings by jointly considering words and topics in a neural network; (5) the **Retrofit** method (Faruqui et al., 2015) is a popular method that refines pre-trained word embeddings using relational information from the knowledge base (e.g., WordNet used in our experiments).

4.3 Experimental Settings

In this paper, we use the text classification task to evaluate the performance of word embeddings. Word embeddings are used to construct the document embeddings $\mathbf{d}$ by simply averaging all word embeddings in the given document, i.e., $\mathbf{d} = \frac{1}{|d|}\sum_{w \in d} \mathbf{w}$, where w is a word in document d. We regard document embedding as a document feature and trained a linear classifier using Liblinear[7] (Fan et al., 2008), since the feature size is large, and Liblinear can quickly train a linear classifier with high dimension features. The classifier is then used to predict the class labels of documents in the test set. The multi-group

[7]https://www.csie.ntu.edu.tw/ cjlin/liblinear/.

Methods	20NewsGroup				5AbstractsGroup				IMDB	MR	SST
	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Acc.	Acc.
BOW	73.6	73.6	72.8	73.0	77.1	76.6	77.2	76.5	85.3	59.3	73.4
GloVe	62.3	61.2	61.1	60.5	79.6	78.4	79.4	79.4	87.4	58.7	75.5
CBOW	74.5	73.6	73.5	73.4	79.4	78.6	78.8	78.8	87.1	61.8	77.9
SG	76.7	75.9	75.6	75.4	85.2	84.0	85.0	84.4	89.1	63.5	77.3
TWE	81.5	81.2	80.6	80.6	81.5	80.5	81.2	80.7	87.1	56.0	76.9
Retrofit-CBOW	75.6	75.9	73.5	72.1	78.2	77.4	77.6	77.3	86.6	61.9	78.0
Retrofit-SG	77.4	77.9	75.5	74.3	83.3	82.3	83.0	82.6	88.8	63.7	77.9
ToWE-CBOW	80.9	80.2	79.9	79.9	84.7	84.0	84.4	84.4	90.1	64.5	**78.8**
ToWE-SG	**86.0**	**85.5**	**85.0**	**85.0**	**87.2**	**86.2**	**87.1**	**87.1**	**90.8**	**65.1**	78.4

Table 2: Performance of our methods on five datasets against the baselines. Bold scores are the best overall.

classification performance was evaluated in terms of four measures: accuracy (Acc.), precision (Prec.), recall (Rec.) and F-measure (F1), and the binary classification performance was evaluated by accuracy (Acc.). All the measures are computed by averaging the metrics of each class and are weighted by the number of true instances for each class.

For each dataset, all documents are joined together as a corpus for embedding training. We tokenized the corpus with the Stanford Tokenizer[8] and converted it to lower case, then removed the stop words. For a fair comparison, all word embeddings adhere to the following settings: the dimensionality of vectors is 300, the size of the context window is 5, the number of negative samples is 25.

In our method, an offline process is used to extract salient word set S from labeled training set $\mathcal{D}$. To obtain an intuitive understanding of these salient words, we list the top ten words for each category in the 5AbstractsGroup dataset. The result is displayed in Table 2. We vary parameter N (detailed in section 3.2) in the range between 30 and 200, and show the performance in Figure 3. Our method achieves the best performance when N is set to 150 for the 5AbstractsGroup dataset. If the value of N is too large, this may hinder the performance because too much noise will be involved. The recommended N is 150 with the constraint that the total size of S is under 1200 based on practical experience.

There are two hyper-parameters in our method, i.e., the combination parameter λ in Eq.(4) and the size n of sample set $P(w)$ in Eq.(3). We carefully tune these parameters by fixing one and varying the other. The parameters corresponding to the best accuracy in 20NewsGroup are used to report the final settings. As shown in Figure 4, the optimal values for λ were tuned from 0 to 1, with a step size of 0.1. The proposed method based on Skip-gram and CBOW reaches optimal performance when $\lambda = 0.4$ and $\lambda = 0.3$, respectively. We tuned the value for n from 50 to 300, and the methods achieve the best performance when $n = 150$. We follow the optimal settings in this work, with recommended settings of $\lambda \in (0.3, 0.4)$ and $n \in (100, 150)$.

4.4 Overall Performance

We compared our proposed method with the baseline methods. Table 2 shows the evaluation results. Based on the experiment results, we make several observations:

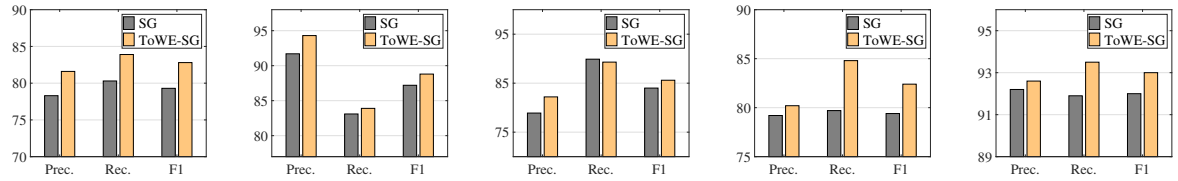
(1) Our method performs better than the other methods, and are proved to be highly reliable for the text classification task. In particular, the ToWE-SG method significantly outperforms the other baselines on the 20NewsGroup, 5AbstractsGroup, and MR. This is mainly attributed to the task-specific modeling mechanism, which enables our models to capture functional features among words, therefore, it can more accurately distinguish classes.

(2) The word embedding methods outperform the basic bag-of-words methods in most cases, indicating the superiority of distributed word representation over the one-hot representation. Moreover, the

[8]https://nlp.stanford.edu/software/tokenizer.shtml

manager (*Business*)		**layer** (*AI*)		**congress** (*Law*)		**poverty** (*Sociology*)		**accident** (*Transport*)	
ToWE-SG	SG	ToWE-SG	SG	ToWE-SG	SG	ToWE-SG	SG	ToWE-SG	SG
managerial	innovate	appearance	form	federal	chapter	deprivations	urbanization	accidents	crash
extant	pursuing	recurrent	forgetting	permit	authorized	belonging	projections	drivers	severity
executives	incentives	architecture	symbolic	administrative	secrecy	homeless	urbanization	severity	injury
stakeholder	subfield	automatic	space	enforcement	prohibiting	inequality	auto	red	rtc
investors	accord	collecting	encoding	earned	bureaucrats	affordability	commuting	mobility	crashes
bond	strategically	cognitive	involves	regulating	dockets	malnutrition	deforestation	road	fatal
moderates	helps	proposed	structures	exception	wrongful	adulthood	anthropogenic	elasticity	rollover
innovation	strategic	learning	polarization	submitted	defense	ethnicity	co-benefits	safety	single-vehicle
marketing	tailor	algorithms	activation	regulate	hear	religious	modal	estimated	taz
asymmetry	create	neural	discontinuous	defense	he	discursive	ownership	delay	crash-related

Table 3: Ten most similar words to the salient words using the ToWE-SG method and SG method. The bold words are salient words, and their category is marked in italic.



(a) Business (b) AI (c) Law (d) Sociology (e) Transport

Figure 5: Performance of SG and ToWE-SG methods for each category in the 5AbstractsGroup dataset.

methods which integrate the abundant information discovered from the datasets (i.e., TWE and ToWE) achieve better performance compared to those that only consider contextual information, such as GloVe, CBOW, and SG. This demonstrates the effectiveness of refining context-aware word embeddings with task information.

(3) The Retrofit method is the knowledge-base enhanced word embedding method. Our method achieves better performance over Retrofit method, indicating that the task-specific features could be more effective compared with general semantic relations constructed by humans in the knowledge bases.

(4) In sentence classification, such as the MR and SST datasets, it is obvious that TWE achieves a relatively lower performance. This observation shows that topical information enhanced word embedding does not accurately represent a short text. Our method outperforms the TWE method on both the document-level and sentence-level tasks, which shows the stability and reliability of modeling task-specific features in real-world applications.

4.5 Case Study

A case study was conducted to qualitatively analyze in-depth why task-oriented word embedding methods surpass typical context-aware word embedding methods. We selected several salient words from different categories in the 5AbstractsGroup dataset, and then compared the top ten similar words obtained by ToWE-SG and SG, respectively. The results are displayed in Table 3. We observe that the similar words selected by the ToWE-SG method belong to the same category, while the SG method may select words from different categories. Taking the word *manager* as an example, the most similar words selected by ToWE-SG all belong to the *Business* category, whereas the SG method selects *helps*, *create* which can hardly be regarded as being in the *Business* category. This demonstrates that our method is capable of capturing a clear boundary in the embedding space. For further investigation, we compared the classification performance of these two word embeddings in each category. As shown in Figure 5, ToWE-SG outperforms SG in all these categories. This indicates that by forcing words in the same category to have similar representations, the classifier achieves better performance.

5 Conclusion

In this paper, we proposed a novel approach for learning task-oriented word embedding, especially for the text classification task. Instead of learning embedding vectors merely based on context information, we incorporate task-specific features into the training process in order to reveal the words functional

attributes in the embedding space. The results of the experiments with different datasets show that the proposed method outperforms the existing state-of-the-art word embedding learning methods on text classification tasks. In the future, we will study how to effectively construct the task-oriented word embeddings with the help of transferable task-features across domains.

6 Acknowledgment

The research work is supported by the National Key Research and Development Program of China under Grant No.2017YFB0803302, National Natural Science Foundation of China (Key Program) under Grant No.61751201 and National Nature Science Foundation of China under Grant No.61602036.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Danushka Bollegala, Mohammed Alsuhaibani, Takanori Maehara, and Ken-ichi Kawarabayashi. 2016. Joint word representation learning using a corpus and a semantic lexicon. In *Proceedings of AAAI*, pages 2690–2696.

Danushka Bollegala, Yuichi Yoshida, and Ken-ichi Kawarabayashi. 2018. Using k-way co-occurrences for learning word embeddings. In *Proceedings of AAAI*, pages 5038–5044.

Shaosheng Cao and Wei Lu. 2017. Improving word embeddings with convolutional feature learning and subword information. In *Proceedings of AAAI*, pages 3144–3151.

Yu Liang Chih, Wang Jin, Lai K. Robert, and Xuejie Zhang. 2017. Refining word embeddings for sentiment analysis. In *Processings of EMNLP*, pages 534–539.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query expansion with locally-trained word embeddings. In *Proceedings of ACL*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL-HLT*, pages 1606–1615.

Rémi Lebret and Ronan Collobert. 2014. Word embeddings through hellinger PCA. In *Proceedings of EACL*, pages 482–490.

Luyang Li, Bing Qin, and Ting Liu. 2017a. Contradiction detection with contradiction-specific word embedding. *Algorithms*, 10(2):59.

Zheng Li, Yu Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017b. End-to-end adversarial memory network for cross-domain sentiment classification. In *Proceedings of IJCAI*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2015a. Learning context-sensitive word embeddings with neural tensor skip-gram model. In *Proceedings of IJCAI*, pages 1284–1290.

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015b. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of ACL*, pages 1501–1511.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015c. Topical word embeddings. In *Proceedings of AAAI*, pages 2418–2424.

Qian Liu, Heyan Huang, Jie Lu, Yang Gao, and Guangquan Zhang. 2017. Enhanced word embedding similarity measures using fuzzy rules for query expansion. In *Proceedings of FUZZ-IEEE*, pages 1–6.

Qian Liu, Heyan Huang, Guangquan Zhang, Yang Gao, Junyu Xuan, and Jie Lu. 2018. Semantic structure-based word embedding by incorporating concept convergence and word divergence. In *Proceedings of AAAI*, pages 5261–5268.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Andriy Mnih and Geoffrey E. Hinton. 2008. A scalable hierarchical distributed language model. In *Proceedings of NIPS*, pages 1081–1088.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Proceedings of NIPS*, pages 2265–2273.

Liqiang Nie, Yi-Liang Zhao, Mohammad Akbari, Jialie Shen, and Tat-Seng Chua. 2015. Bridging the vocabulary gap between health seekers and healthcare knowledge. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):396–409.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*, pages 115–124.

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

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018a. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Proceedings of AAAI*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. 2018b. Bi-directional block self-attention for fast and memory-efficient sequence modeling. In *Proceedings of ICLR*.

R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*.

Liang-Chih Yu, Jin Wang, K. Robert Lai, and Xue-Jie Zhang. 2017. Refining word embeddings for sentiment analysis. In *Proceedings of EMNLP*, pages 534–539.

Hamed Zamani and W. Bruce Croft. 2017. Relevance-based word embedding. In *Proceedings of SIGIR*, pages 505–514.

Adaptive Learning of Local Semantic and Global Structure Representations for Text Classification

Jianyu Zhao[1,2,*], Zhiqiang Zhan[1,2,*], Qichuan Yang[3], Yang Zhang[4], Changjian Hu[4], Zhensheng Li[4], Liuxin Zhang[4], and **Zhiqiang He[4]**
[1]University of Chinese Academy of Sciences, Beijing 100049, China
[2]Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[3]Beihang University, Beijing 100103, China
[4]Lenovo Research, Beijing 100085, China
zhaojy7ict@gmail.com

Abstract

Representation learning is a key issue for most Natural Language Processing (NLP) tasks. Most existing representation models either learn little structure information or just rely on pre-defined structures, leading to degradation of performance and generalization capability. This paper focuses on learning both local semantic and global structure representations for text classification. In detail, we propose a novel Sandwich Neural Network (SNN) to learn semantic and structure representations automatically without relying on parsers. More importantly, semantic and structure information contribute unequally to the text representation at corpus and instance level. To solve the fusion problem, we propose two strategies: Adaptive Learning Sandwich Neural Network (AL-SNN) and Self-Attention Sandwich Neural Network (SA-SNN). The former learns the weights at corpus level, and the latter further combines attention mechanism to assign the weights at instance level. Experimental results demonstrate that our approach achieves competitive performance on several text classification tasks, including sentiment analysis, question type classification and subjectivity classification. Specifically, the accuracies are MR (82.1%), SST-5 (50.4%), TREC (96%) and SUBJ (93.9%).

1 Introduction

Representation learning plays an important role in various NLP tasks, especially in text classification (Bengio et al., 2013; Le and Mikolov, 2014). Existing representation models for text classification can be categorized into four types: *bag-of-words representation models*, *sequence representation models*, *structure representation models* and *attention-based models*. *Bag-of-words representation models* (Salton et al., 1975) are able to represent the sentence accounting for the different words, but fail to encode word order and syntactic structures. *Sequence representation models* (Kim, 2014; Kalchbrenner et al., 2014) consider word order without using structure information. *Structure representation models*, such as Tree-LSTM (Tai et al., 2015) and DSCNN (Zhang et al., 2016), take the structure information into account. *Attention-based models* (Yang et al., 2017; Lin et al., 2017) use attention mechanism to build representations by scoring input words respectively.

However, most structure representation methods either learn little structure information or rely heavily on parsers, leading to relatively low performance or gradient vanishing problem. Moreover, to our best knowledge, no one has considered effective fusion of semantic and structure information. These hinder the performance of sentence representation for text classification.

To address these issues, we propose a novel model named Sandwich Neural Network, which consists of a Convolutional Neural Network (CNN) layer in the middle of two Long Short-Term Memory (LSTM) layers like a sandwich. We define local semantic representation as n-grams feature and global structure representation as the dependency relationship between words or phrases in the sentence. SNN can generate both local semantic representation and global structure representation without relying on parsers. Then two dynamic fusion methods are developed to make full use of these information.

*They contribute equally to this paper.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2033–2043
Santa Fe, New Mexico, USA, August 20-26, 2018.

The contributions of this paper can be summarized into two parts:

1) We propose a novel SNN to extract semantic and structure representations, which enriches information of sentence representation. Thus SNN improves the performance of text classification.

2) In order to make full use of these two representations, we design and implement two fusion methods to learn the weights of semantic and structure representations at corpus and instance level respectively. The former employs adaptive learning strategy to automatically learn a pair of weights for the whole corpus. The latter integrates attention mechanism to adjust the weights subtly for each single instance.

The rest of our paper is organized as follows: Section 2 elaborates the related work about structure representation and attention mechanism. Section 3 details the proposed SNN and two fusion methods. Section 4 conducts experiments on four datasets to verify the effectiveness of our model. Section 5 draws a conclusion and discusses the future work.

2 Related Work

Text classification develops from rule-based method, statistical machine learning method to deep learning method (Aggarwal and Zhai, 2012). Many researches on text classification focus on the sentence representation (Liu et al., 2018). Structure representation and attention mechanism are important for sentence representation.

2.1 Sentence Structure Representation

Current researches about sentence representation pay much attention to structure representation. Structure representation models can be divided into three types: *hierarchical models*, *tree-based models* and *reinforcement learning (RL) models*. *Hierarchical models* combine CNN and RNN to learn structure representation, such as C-LSTM (Zhou et al., 2015) and DSCNN (Zhang et al., 2016). C-LSTM utilizes CNN to extract a sequence of higher-level phrase representations. Then the representations are fed into an LSTM to obtain sentence representation with long dependency structure. DSCNN utilizes CNN to extract features from hidden states of an LSTM layer, which is able to catch long dependency structure at a certain level. *Tree-based models*, such as Tree-LSTM (Tai et al., 2015) and TBCNN (Mou et al., 2015), rely on existing parsers. Tree-LSTM utilizes LSTM along the tree structure and treats root as sentence representation. TBCNN combines CNN with tree structure and uses pooling results as sentence representation. More recently, *RL models* are attempted in structure representation, such as ID-LSTM (Zhang et al., 2018) and HS-LSTM (Zhang et al., 2018), which are integrated seamlessly with a policy network and a classification network. These works contribute significantly to sentence structure representation.

However, *hierarchical models* are not able to learn adequate structure information and lack effective fusion of semantic and structure representations; *tree-based models* are time-consuming and rely on parsers which are unable to guarantee an accurate syntactic structure; *RL models* require much experience to initialize and design reward function, which yields moderate performance.

2.2 Attention Mechanism

Attention mechanism for NLP is first proposed by Bahdanau et al. (2014) to improve machine translation task. The neural machine translation model consists of two RNNs: an encoder and a decoder. When decoding the hidden state s_i, the decoder calculates attention distribution $\boldsymbol{\alpha}$ on the whole source sentence. The attention mechanism is defined as follows:

$$\begin{aligned} \boldsymbol{\alpha}_{ij} &= exp(\boldsymbol{e}_{ij}) / \sum_{k=1}^{T_x} exp(\boldsymbol{e}_{ik}) \\ \boldsymbol{e}_{ij} &= \sigma(\boldsymbol{s}_{i-1}, \boldsymbol{h}_j) \end{aligned} \tag{1}$$

where x is the source sentence, T_x is the length of source sentence, σ is an activation function, $\boldsymbol{s}_{i-1}$ is the hidden state of the decoder RNN at time step $i-1$, $\boldsymbol{h}_j$ is the hidden state of the encoder RNN at time step j.

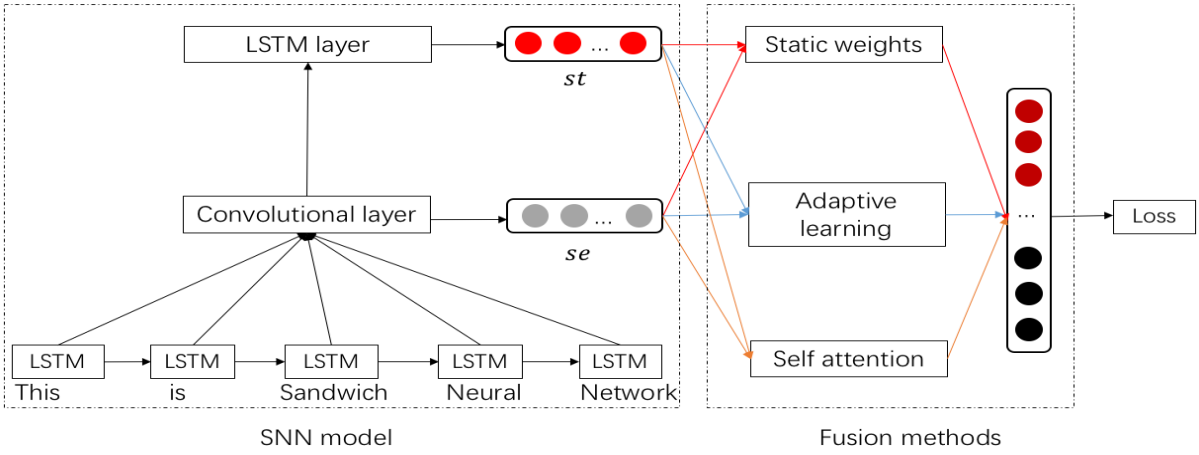


Figure 1: SNN and fusion methods.

Because of its promising performance, attention mechanism attracts lots of attention. For example, Rush et al. (2015) apply it to solve sentence summarization. Xu et al. (2015) utilize it to generate caption of images. Tan et al. (2017) employ it to handle reading comprehension task.

3 Proposed Model

In order to address problems as discussed in Section 2.1, we are inspired by Section 2.2 to propose our novel methods. As shown in Figure 1, there exists three major parts: (1) SNN model: build the semantic and structure representations through SNN; (2) Fusion methods: fuse the semantic and structure representations with different methods; (3) Loss function: train the model with cross-entropy objective function. The core of our methods can be formulated as $\boldsymbol{M} = \phi(\boldsymbol{se}, \boldsymbol{st})$, where ϕ is a fusion function which combines the semantic representation $\boldsymbol{se}$ with the structure representation $\boldsymbol{st}$.

3.1 Obtaining the semantic and structure representations

As shown in Figure 1, SNN consists of three parts: First LSTM layer, CNN layer and Second LSTM layer. The first LSTM layer utilizes the hidden states of LSTM as tuned word representations to take the context into account and the tuned representations are the input of CNN layer. The CNN layer is designed to learn local n-gram high-level semantic representations like standard CNN (Kim, 2014) through convolution and max pooling. The second LSTM layer feeds results from the convolution into LSTM and use the last hidden state of LSTM to obtain global structure representations.

First LSTM layer

Let the input of our model be a sentence of length $s : [w_1, w_2, ..., w_s]$, c be the total number of word embedding versions and $\boldsymbol{x}_i^{(j)}$ is the i^{th} word's embedding of the j^{th} version. The common word embedding versions are Word2vec (Mikolov et al., 2013) and Glove (Pennington et al., 2014).

The first layer of our model consists of LSTM networks processing multiple versions of word embeddings. For each version of word embedding, we construct an LSTM network where the input $\boldsymbol{x}_t \in \mathbb{R}^d$ is the d-dimensional word embedding for w_t. The LSTM layer will produce a hidden state $\boldsymbol{h}_t \in \mathbb{R}^d$ at each time step. We collect hidden states as the output of LSTM layers:

$$\boldsymbol{h}^{(i)} = [\boldsymbol{h}_1^{(i)}, \boldsymbol{h}_2^{(i)}, ..., \boldsymbol{h}_t^{(i)}, ..., \boldsymbol{h}_s^{(i)}] \tag{2}$$

for $i = 1, 2, ..., c$.

Then these hidden states are fed into CNN layer as input.

CNN layer

To utilize multiple kinds of word embeddings, we apply a filter $\boldsymbol{F} \in \mathbb{R}^{c \times d \times l}$, where l is the window size. Hidden state sequence $\boldsymbol{h}^{(i)}$ produced by the i^{th} version of word embedding forms one channel of the feature maps. Then these feature maps are stacked c-channel feature maps $\boldsymbol{X} \in \mathbb{R}^{c \times d \times s}$. Afterwards,

Sentence	Ground truth	Reason
What is the conversion rate between dollars and pounds ?	numeric	key word "rate"
What does target heart rate mean ?	describe	key structure "what does ... mean" and word "rate" is a disturbance

Table 1: Examples of TREC classification

filter F convolves with the window vectors (l-gram) at each position to generate a feature map $\boldsymbol{c} \in \mathbb{R}^{s-l+1}$; c_k is the element of the feature map $\boldsymbol{c}$ for window vector $\boldsymbol{X}_{k:k+l-1}$ at position k and it is produced as follows:

$$c_k = f(\sum_{i,j,r}(\boldsymbol{F} \odot \boldsymbol{X}_{k:k+l-1})_{i,j,r}) \tag{3}$$

where $\odot$ denotes element-wise multiplication.

The n feature maps generated from n filters can be rearranged through column vector concatenation method to form a new representation,

$$\boldsymbol{W} = [\boldsymbol{c}_1; \boldsymbol{c}_2; ...; \boldsymbol{c}_n] \tag{4}$$

Each row $\boldsymbol{W}_j$ of $\boldsymbol{W} \in \mathbb{R}^{(s-l+1)\times n}$ is the feature map generated from n filters for the window vector at position j. The new successive higher-level representations are then fed into the second LSTM layer.

Here, a max-over-time pooling layer is added after the convolution neural network. The pooling result of the feature map $\boldsymbol{c}$ is :

$$p = max(c_1, c_2, ..., c_{s-l+1}) \tag{5}$$

These pooling results are used as our local semantic representation $\boldsymbol{se} \in \mathbb{R}^n$:

$$\boldsymbol{se} = [p_1, p_2, ..., p_n] \tag{6}$$

Second LSTM layer

This layer feeds the high-level phrase representation $\boldsymbol{W}$ generated from CNN into LSTM. We use the same number of filters n to denote the dimension in this LSTM layer for simple and fair fusion and use the last hidden state of LSTM as global structure representation $\boldsymbol{st} \in \mathbb{R}^n$.

Thus, we get the local semantic representation $\boldsymbol{se}$ and the global structure representation $\boldsymbol{st}$. Then we combine $\boldsymbol{se}$ and $\boldsymbol{st}$ to get the concatenated weighted sentence representation.

3.2 Fusion methods

To make better use of semantic and structure representations and address the fusion problems discussed in Section 1, we explore three different fusion ways to learn the weights of semantic and structure representations.

Static weights

Combine semantic and structure representations to get the sentence representation. That is to say, this method gives equal weights to these two kinds of representations.

Adaptive learning at corpus level

Different properties (i.e. semantic and structure property) of a sentence contribute unequally according to the specific corpus. Table 1 shows examples of this phenomenon. To take the above observation into account, we design an adaptive learning method to learn the semantic weight g_{se} and the structure weight g_{st} at corpus level. To reflect the difference of these two properties, we use a simple but elegant and effective method: set g_{se} to be α and g_{st} to be $1 - \alpha$. They are parameters that can be learned automatically during the training process. Each element in the semantic representation $\boldsymbol{se}$ will multiply α to get the weighted semantic representation. It is the same with structure representation $\boldsymbol{st}$ and $1 - \alpha$. Figure 2 shows the adaptive learning method.

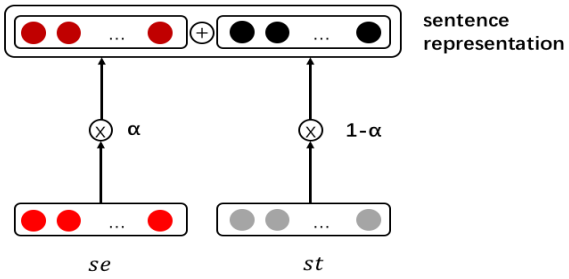


Figure 2: Adaptive learning at corpus level.

Self-attention at instance level

With the adaptive learning strategy, the weights are learned based on the whole corpus. However, the weights of semantic and structure information vary according to the sentence itself. For instance, "*The dog the stick the fire burned beat bit the cat*" is rich in structure information, and we should pay more attention to its structure. Motivated by the attention mechanism, we propose the self-attention strategy to learn weights at instance level. First, we average the word embeddings of the sentence to get a simple but useful sentence representation $\boldsymbol{S}$. Second, linear transformations are used to transform the representations into semantic space representation $\boldsymbol{S}_{se}$ and structure space representation $\boldsymbol{S}_{st}$. The transformed dimension is d. Third, we compute the similarity of representations through inner product. Last, softmax is used to normalize weights. Figure 3 shows the whole process. The computational procedure is as follows:

$$
\begin{aligned}
(att_{se}, att_{st}) &= softmax(p_{se}, p_{st}) \\
p_{se} &= \rho(\boldsymbol{S}_{se}, \boldsymbol{se}) \\
p_{st} &= \rho(\boldsymbol{S}_{st}, \boldsymbol{st}) \\
\boldsymbol{S}_{se} &= \boldsymbol{W}_{se} \times \boldsymbol{S} + \boldsymbol{b}_{se} \\
\boldsymbol{S}_{st} &= \boldsymbol{W}_{st} \times \boldsymbol{S} + \boldsymbol{b}_{st} \\
\boldsymbol{S} &= (\boldsymbol{x}_1 + \boldsymbol{x}_2 + ... + \boldsymbol{x}_s)/s
\end{aligned}
\tag{7}
$$

where $\boldsymbol{x}_i$ is the word embedding, size of d; $\boldsymbol{S}$ is a representation of the sentence, calculated by averaging the word embeddings, size of d; $\boldsymbol{S}_{se}$ and $\boldsymbol{S}_{st}$ are the transformed semantic and structure representations, size of n; $\boldsymbol{W}_{se}$ and $\boldsymbol{W}_{st}$ are the projection matrixes, size of $n \times d$; $\boldsymbol{b}_{se}$ and $\boldsymbol{b}_{st}$ are the projection biases, size of d; ρ is an average inner product operator; att_{se} and att_{st} are the attention weights.

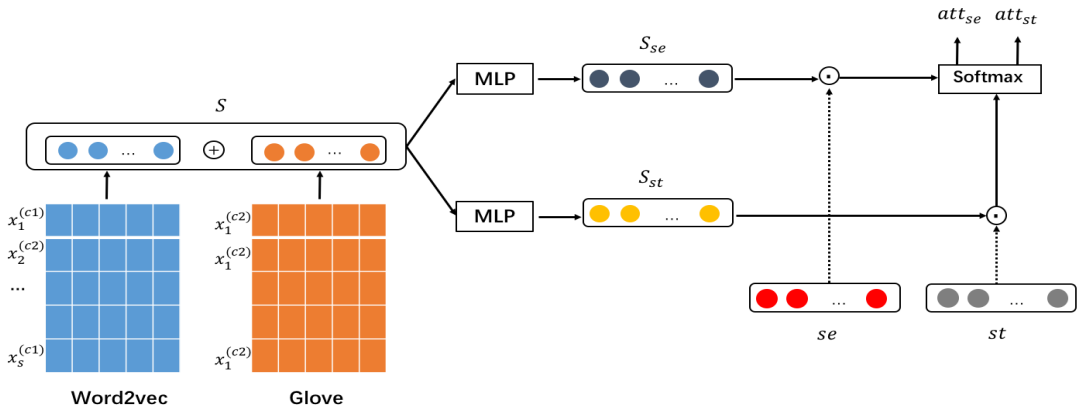


Figure 3: Self-attention at instance level.

In the above fusion methods, the adaptive learning strategy and self-attention strategy can learn weights on the two representations. Then, we concatenate weighted semantic and structure represen-

Data	Class	Len	Max Len	Size	Test
MR	2	20	56	10662	CV
SST-5	5	18	53	11855	2210
TREC	6	10	37	9592	500
SUBJ	2	23	120	10000	CV

Table 2: Statistics for experiment datasets. Class: number of classes. Len: average length of sentence. Max: max length of sentence. Size: Dataset size. Test: test set size (CV means no standard train/test split and thus 10-fold CV is used).

tations to get the final sentence representation.

$$\boldsymbol{M} = \begin{cases} [\boldsymbol{se}; \boldsymbol{st}], \text{for fixed weights} \\ [g_{se} * \boldsymbol{se}; g_{st} * \boldsymbol{st}], \text{for adaptive weights} \\ [att_{se} * \boldsymbol{se}; att_{st} * \boldsymbol{st}], \text{for self-attention} \end{cases} \tag{8}$$

where $\boldsymbol{M}$ is the sentence representation, and operator $[\boldsymbol{v1}; \boldsymbol{v2}]$ denotes concatenation of vector $\boldsymbol{v1}$ and $\boldsymbol{v2}$.

3.3 Training Models

After getting both semantic and structure representations, a fully-connected layer follows. To learn the model parameters, we minimize a cross-entropy objective function as follows:

$$Loss = -\frac{1}{m}\sum_{i=1}^{m} y^i \times log(\hat{y}_i) \tag{9}$$

where y^i is the one-hot vector representation of real label of the i-th sentence, $\hat{y}_i$ is the predicted probability representation in the neural network, and m is the number of samples.

4 Experiments

4.1 Datasets

We test our model on several datasets. The detailed information is listed in Table 2.

• **Movie Review (MR)** proposed by Pang and Lee (2005) is a dataset for sentiment analysis of movie reviews. It contains two classes: positive and negative.

• **Stanford Sentiment Treebank with Five Labels (SST-5)** is another popular sentiment classification dataset proposed by Socher et al. (2013). The sentences are labeled in a fine-grained way: very negative, negative, neutral, positive, very positive.

• **Text REtrieval Conference (TREC)** dataset proposed by Li and Roth (2002) contains sentences that are questions in the following 6 classes: abbreviation, entity, description, location, numeric, human.

• **SUBJectivity (SUBJ)** classification dataset released by Pang and Lee (2004) contains two classes: subjective and objective.

4.2 Experimental Configuration

Literature usually adopts 100-300 as the dimension of word embedding. Here, we use two versions of word embeddings, Word2vec and Glove, with dimension 300. We use 100 convolution filters each for window sizes of 3,4,5. Rectified Linear Units (RELU) is chosen as the nonlinear function in the convolutional layer. For the second LSTM, we set the dimension to be 100. For regularization, both word-embedding and the penultimate layer use dropout (Hinton et al., 2012) with rate 0.5. We don't impose L2 regularization at each layer. We use the gradient-based optimizer Adam (Kingma and Ba, 2014) to minimize the loss between the predicted and true distributions, and the training is early stopped when the accuracy on validation set starts to drop. Additionally, we use the same configuration for all datasets.

Method	MR	SST-5	TREC	SUBJ
Standard-RNN (Socher et al., 2013)	–	43.2	–	–
Standard-LSTM (Tai et al., 2015)	77.4	46.4	–	92.2*
bi-LSTM (Tai et al., 2015)	79.7	49.1	–	92.8*
CNN (Kim, 2014)	81.5	48	93.6	93.4
DCNN (Kalchbrenner et al., 2014)	–	48.5	93	–
MVCNN (Yin and Schütze, 2016)	–	49.6	–	**93.9**
Tree-LSTM (Tai et al., 2015)	80.7	50.1	–	93.2*
TBCNN (Mou et al., 2015)	–	**51.4**	**96**	–
Dep-CNN (Ma et al., 2015)	81.9	49.5	95.4	–
DSCNN (Zhang et al., 2016)	81.5	49.7	95.4	93.2
C-LSTM (Zhou et al., 2015)	–	49.2	94.6	–
ID-LSTM (Zhang et al., 2018)	81.6	50.0	–	93.5
HS-LSTM (Zhang et al., 2018)	**82.1**	49.8	–	93.7
Self-Attentive (Lin et al., 2017)	80.1	47.2	–	92.5
SNN	81.7	49.8	95.6	93.8
AL-SNN	81.9	50	95.8	**93.9**
SA-SNN	**82.1**	50.4	**96**	**93.9**

Table 3: Experiment results of our methods compared with other models. Performance is measured in accuracy(%). The results with * is from Zhang et al. (2018). Models are categorized into 7 classes. The first block is RNN and its variants. The second is CNN and its variants. The third is tree-based model. The forth is the hierarchical model. The fifth is RL model. The sixth is attention-based model. And the last is our models. **Standard-RNN**: Standard Recursive Neural Network (Socher et al., 2013). **Standard-LSTM**: Standard Long Short-Term Memory Network (Tai et al., 2015). **bi-LSTM**: Bidirectional LSTM (Tai et al., 2015). **CNN**: Convolutional Neural Network (Kim, 2014) . **DCNN**: Dynamic Convolutional Neural Network with k-max pooling (Kalchbrenner et al., 2014). **MVCNN**: Multichannel Variable-Size Convolution Neural Network (Yin and Schütze, 2016). **Tree-LSTM**: Tree-Structured LSTM (Tai et al., 2015). **TBCNN**: Tree-Based Convolutional Neural Network (Mou et al., 2015). **Dep-CNN**: Dependency-based Convolutional Neural Network (Ma et al., 2015). **DSCNN**: Dependency Sensitive Convolutional Neural Network (Zhang et al., 2016). **C-LSTM** :Convolutional LSTM (Zhou et al., 2015). **ID-LSTM**: Information Distilled LSTM (Zhang et al., 2018). **HS-LSTM**: Hierarchical Structured LSTM (Zhang et al., 2018). **Self-Attentive**: Structured Self-Attentive model (Lin et al., 2017).

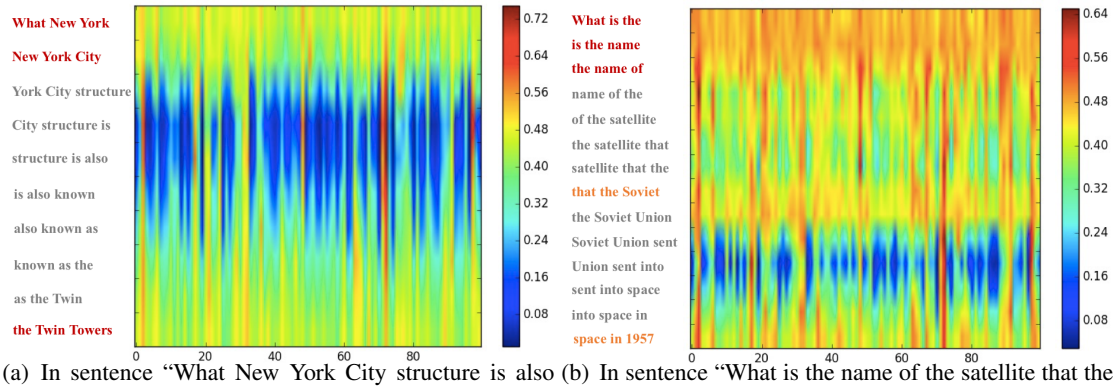
4.3 Results and Discussion

As shown in Table 3, the results demonstrate effectiveness of our model in comparison with other state-of-the-art methods. Among the models without relying on parsers, the other models get the highest accuracy in one certain task (e.g. HS-LSTM in MR and MVCNN in SUBJ), but our models get the highest accuracy on all datasets: MR, SST-5, TREC and SUBJ. It shows strong generalization capability. Additionally, compared with TBCNN, which relies heavily on parsers, our models get the same highest accuracy on TREC and are on a par with TBCNN's performance on SST-5.

The benefit of SNN is attributed to its ability to capture both semantic and structure representations. The benefit of adaptive learning is attributed to its ability to learn the weights of both representations according to the specific corpus. The benefit of self-attention is attributed to its ability to adjust the weights of both representations at instance level. In the following, we first give visualization of our models to show the learned structure. Then, we give some examples to show how our models make progress from SNN, adaptive learning to self-attention. At last, we make quantitative analysis to show the correlation between structure attention value and structure complexity.

4.3.1 Visualization of learned structure

The input gate value of LSTM is able to show how much a word representation contributes to the final sentence representation (Palangi et al., 2016). We draw a heat map of input gates of the second LSTM layer to show the learned structure. Figure 4 gives two examples from TREC. The color represents the values (from 0 to 1) of input gate. The higher means the n-gram part contributes more to the final sentence representation.



(a) In sentence "What New York City structure is also known as the Twin Towers", structure "What New York City" – "the Twin Towers" is learned. It is very related to sentence class "location".

(b) In sentence "What is the name of the satellite that the Soviet Union sent into space in 1957", structure "What is the name of" is important and related to sentence class "entity". More amazing, attributive clause structure "that" is learned. The content before "that" contributes more than the content after "that", which makes the representation focus more on what is related to the classification.

Figure 4: Visualization of learned structure.

Sentence	Dataset	Ground truth	DSCNN	SNN
What does your spleen do ?	TREC	describe	abbreviation	describe
Some actors have so much charisma that you'd be happy to listen to them reading the phone book.	SST-5	positive	negative	positive

Table 4: Examples that SNN makes right while DSCNN fails.

4.3.2 SNN analysis

SNN can learn both semantic and structure representations. As shown in Table 4: in the first sentence, word "spleen" does not appear in the train data, but its topically related words in medical domain like "AIDS", "HIV", "BPH", "SIDS" appear in abbreviation class frequently. Traditional models (e.g. DSCNN) ignore the structure "what does ... do" and misclassifies it into abbreviation class. In contrast, our model captures the structure and correctly classifies it into entity class.

4.3.3 AL-SNN analysis

AL-SNN can learn the weights of semantic and structure representations at corpus level. As shown in Table 5: g_{se} denotes semantic weight and g_{st} denotes structure weight as mentioned in Section 3. In the first sentence, the structure "what is ... the name of" appears with almost the same frequency (67 and 66 times) in human and entity class and appears very little in the other classes. Thus, more weights should be added on semantic representation. Through adaptive learning at corpus level, AL-SNN achieves it.

Sentence	Dataset	Ground truth	SNN	AL-SNN	g_{se}	g_{st}
What was the name of the plane Lindbergh flew solo across the Atlantic?	TREC	entity	human	entity	0.94	0.06
The acting is stiff, the story lacks all trace of wit, the sets look like they were borrowed from gilligan's inland, and – the cgi scooby might well be the worst special-effects creation of the year.	SST-5	very negative	negative	very negative	0.56	0.44

Table 5: Examples that AL-SNN makes right while SNN fails.

Sentence	Dataset	Ground truth	AL-SNN	SA-SNN	g_{st}	att_{st}
What is Hawaii 's state flower ?	TREC	entity	location	entity	0.06	0.33
The director byler may yet have a great movie in him, but charlotte sometimes is only half of one .	SST-5	negative	positive	negative	0.44	0.57

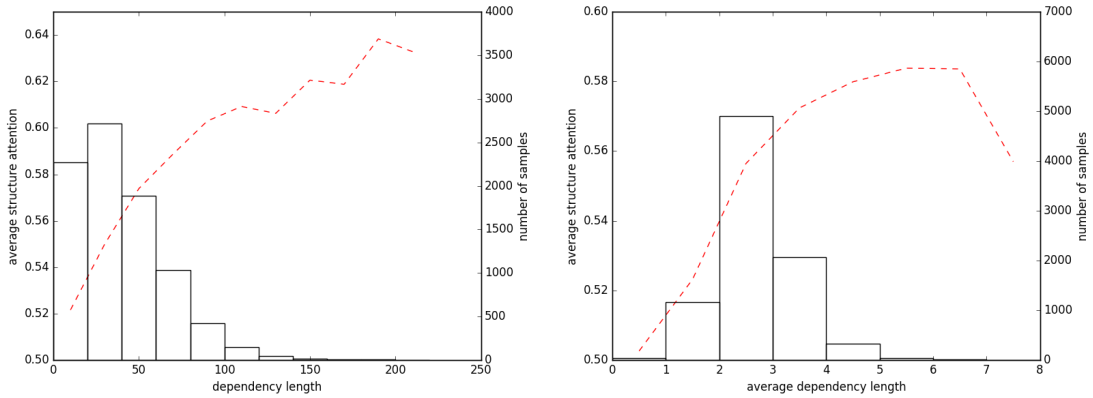
Table 6: Examples that SA-SNN makes right while AL-SNN fails.

4.3.4 SA-SNN analysis

SA-SNN can adjust weights at instance level. As shown in Table 6: g_{st} denotes structure weight and att_{st} denotes structure attention as mentioned in Section 3. In the first sentence, AL-SNN gives more weight to semantic representation at corpus level, focuses on word "Hawaii" and misclassifies the sentence into location class. Through self attention, SA-SNN gives more weight to structure, focuses on structure "what is ... ?" and correctly classifies the sentence into entity class.

4.3.5 Quantitative analysis

It is obvious that more complicated structure means higher structure weight. We use four statistics to represent structure complexity: dependency length (dl), average dependency length (adl), max dependency length (mdl) and sentence length (sl). Among them, average dependency length is dependency length divided by the number of dependency relationships of the sentence. Figure 5 depicts the correlation between the average structure attention (asa) of SA-SNN and the four statistics on SST-5. It shows a strongly positive correlation apart from a few small random variations caused by a limited sample size. Thus, it proves that our attention mechanism is very reasonable.



(a) the correlation between asa and dl

(b) the correlation between asa and adl

(c) the correlation between asa and mdl

(d) the correlation between asa and sl

Figure 5: The correlation between structure attention of SA-SNN and structure complexity on SST-5.

5 Conclusion

In this paper, we propose a novel SNN model and two effective fusion methods, i.e. AL-SNN and SA-SNN. By capturing and fusing both local semantic information and global structure information effectively, our model achieves state-of-the-art in several text classification tasks among the methods without relying on parsers, which verifies its great performance and strong generalization capability. Additionally, we give examples, visualization and quantitative analysis to make explanations in detail. In the future, we will pay more attention to knowledge-embedded methods to speed up training and improve performance.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments and suggestions, which are helpful in improving the quality of the paper.

References

Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text classification algorithms. In *Mining text data*, pages 163–222. Springer.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Y Bengio, A Courville, and P Vincent. 2013. Representation learning: a review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computer Science*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

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

Li Liu, Jie Chen, Paul Fieguth, Guoying Zhao, Rama Chellappa, and Matti Pietikainen. 2018. A survey of recent advances in texture representation. *arXiv preprint arXiv:1801.10324*.

Mingbo Ma, Liang Huang, Bing Xiang, and Bowen Zhou. 2015. Dependency-based convolutional neural networks for sentence embedding. *arXiv preprint arXiv:1507.01839*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. *arXiv preprint arXiv:1504.01106*.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(4):694–707.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

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

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Chuanqi Tan, Furu Wei, Nan Yang, Weifeng Lv, and Ming Zhou. 2017. S-net: From answer extraction to answer generation for machine reading comprehension. *arXiv preprint arXiv:1706.04815*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2017. Hierarchical attention networks for document classification. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Wenpeng Yin and Hinrich Schütze. 2016. Multichannel variable-size convolution for sentence classification. *arXiv preprint arXiv:1603.04513*.

Rui Zhang, Honglak Lee, and Dragomir Radev. 2016. Dependency sensitive convolutional neural networks for modeling sentences and documents. *arXiv preprint arXiv:1611.02361*.

Tianyang Zhang, Minlie Huang, and Li Zhao. 2018. Learning structured representation for text classification via reinforcement learning.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

Lyrics Segmentation: Textual Macrostructure Detection using Convolutions

Michael Fell[♣], Yaroslav Nechaev[♦], Elena Cabrio[♥], Fabien Gandon[♠]
Université Côte d'Azur, CNRS, Inria, I3S, France[♣♥♠]
Fondazione Bruno Kessler, University of Trento[♦]
michael.fell@unice.fr[♣], nechaev@fbk.eu[♦]
elena.cabrio@unice.fr[♥], fabien.gandon@inria.fr[♠]

Abstract

Lyrics contain repeated patterns that are correlated with the repetitions found in the music they accompany. Repetitions in song texts have been shown to enable lyrics segmentation – a fundamental prerequisite of automatically detecting the building blocks (e.g. chorus, verse) of a song text. In this article we improve on the state-of-the-art in lyrics segmentation by applying a convolutional neural network to the task, and experiment with novel features as a step towards deeper macrostructure detection of lyrics.

Title and Abstract in French

Segmenter les paroles de chansons:
détection par réseau de neurones convolutif d'une macrostructure textuelle

Les paroles de chansons contiennent des passages qui se répètent et sont corrélés aux répétitions trouvé dans la musique qui les accompagne. Ces répétitions dans les textes de chansons ont montré leur utilité pour la segmentation des paroles qui est une étape préalable fondamentale dans la détection automatique des blocs de construction d'une chanson (ex. le refrain, les couplets). Dans cet article, nous améliorons l'état de l'art de la segmentation des paroles en concevant un réseau de neurones convolutif pour cette tâche et expérimentons de nouvelles caractéristiques pour aller vers une détection plus profonde de la macrostructure des paroles.

1 Introduction

Among the seas of textual resources available online are the lyrics of songs that are very popular resources for professional, cultural and social applications. To support intelligent interactions with these resources (e.g. browsing, visualizing, synchronizing) one of the first needs is to access the structure of the lyrics (e.g. intro, verse, chorus). However, lyrics are essentially provided as flat text files without structural markup. In this article, we address the problem of textual macrostructure detection in lyrics.

Lyrics encode an important part of the semantics of a song and a motivating scenario for this work is to improve structural clues that can be used by search engines handling large collections of lyrics. Ideally we would like to be able to detect these structures reliably in different music genres to support new search criteria such as "find all the songs where the chorus talks about freedom".

As a first step, structure segmentation could serve as a front end processor for music content analysis, since it enables a local description of each section rather than a coarse, global representation of the whole song (Cheng et al., 2009; Casey et al., 2008). Music structure discovery is a research field in Music Information Retrieval where the goal is to automatically estimate the temporal structure of a music track by analyzing the characteristics of its audio signal over time. However only a few works have addressed such task lyrics-wise (P. G. Mahedero et al., 2005; Watanabe et al., 2016; Baratè et al., 2013). Given that lyrics contain rich information about the semantic structure of a song, relying on textual features could

Proceedings of the 27th International Conference on Computational Linguistics, pages 2044–2054
Santa Fe, New Mexico, USA, August 20-26, 2018.

help in overcoming the existing difficulties associated with large acoustic variation in music. Moreover, a complete music search engine should support search criteria exploiting both the audio and the textual dimensions of a song.

In this direction, this paper focuses on the following research question: *given the text of the lyrics, can we learn to detect the lines delimiting segments in the song?* This question is broken down into two sub questions: *which features are the most relevant to detect that a line delimits a segment?* and *which classification method is the most efficient to detect that a line delimits a segment?*

In order to infer such lyrics structure, we introduce a neural network-based model that *i)* efficiently exploits the Self-Similarity Matrix representations (SSM) used in the state-of-the-art (Watanabe et al., 2016), *ii)* can utilize traditional features alongside the SSMs, and *iii)* jointly infers the structure of the entire text. An evaluation on two standard datasets of English lyrics (Music Lyrics Database V.1.2.7[1], and the WASABI corpus (Meseguer-Brocal et al., 2017)), shows that our proposed method can effectively detect the boundaries of music segments outperforming the state of the art, and is portable across collections of song lyrics of heterogeneous musical genre.

Generally speaking, structure segmentation consists of two stages: a text segmentation stage that divides lyrics into segments, and a semantic labeling stage that labels each segment with a structure type (e.g. intro, verse, chorus). Although a few works have addressed the task of finding chorus or repeated parts in music (P. G. Mahedero et al., 2005; Baratè et al., 2013), full song text segmentation remains challenging unless some complexity reduction strategies are applied (as selecting a subset of songs belonging to musical genres characterized by repeating patterns, e.g. Country or Pop songs). As the accuracy of lyrics segmentation in state of the art is not fully satisfying yet (Watanabe et al., 2016), and given the variability in the set of structure types provided in the literature according to different genres (Tagg, 1982; Brackett, 1995), rare attempts have been made to achieve semantic labeling. Therefore, in our work, we focus on improving the state of the art in lyrics segmentation, leaving the task of semantic labeling for future work.

Earlier in this section, we presented our research questions and motivation. In the remainder of the paper, in Section 2 we define the task of classifying lines as segment borders, and in Section 3 we detail the features used to represent the lines and the classification methods we selected for the task. Section 4 presents the experiment we conducted and compares the results obtained by the different models and configurations to perform the segment border classification. In Section 5 we position our work in the current state of the art, and in Section 6 we conclude with future research directions to provide ever more metadata to music information retrieval systems.

2 Segmentation task definition

Figure 1 shows a song text and its segmentation into the segments A, B, C, D, E, as given by the annotation of the text. As a Pop song, this example lyrics has a fairly common structure which can be described as: *Verse 1-Chorus-Verse 2-Chorus-Outro*. The reasoning behind this structure analysis is that perfectly repeating parts usually correspond to the chorus. Hence, B and D should both be a chorus. A and C are verses, as they lead to the chorus and there is no visible bridge. The last segment, E, repeats the end of the chorus and is very short, so it can be classified as an outro. While the previous analysis appears plausible, it relies on world knowledge, such that the chorus is the most repeated part, a verse usually leads into a chorus (optionally via a bridge), and an outro ends a song text, but is optional.[2]

Labelling the segments of a song text is a non-trivial task that requires diverse knowledge. As explained in the introduction, in this work we focus on the first task of segmenting the text. This first step is fundamental to segment labelling when segment borders are not known. Even when segment borders are "indicated" by line breaks in lyrics available online, those line breaks have usually been annotated by users and neither are they necessarily identical to those intended by the songwriter, nor do users in general agree on where to put them. Thus, a method to automatically segment unsegmented song texts is needed to automate that first step. Many heuristics can be imagined to find the segment borders. For

[1] `http://www.odditysoftware.com/page-datasales1.htm` (retrieved 11/24/17)

[2] For more details on the set of structure types, we refer the reader to (Tagg, 1982; Brackett, 1995).

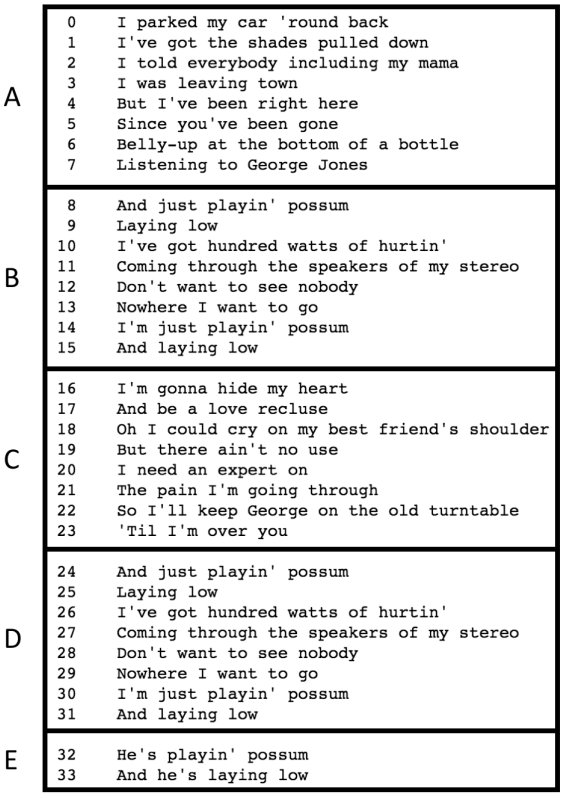


Figure 1: Segment structure of a Pop song ("Don't Rock The Jukebox" by A. Jackson, MLDB-ID: 2954)

example, separating the lines into segments of consistent lengths, which in our example gives 8-8-8-8-2 lines. Another heuristic could be to never start a segment with a conjunction. But as we can see, this rule does not hold for our example, as the chorus starts with the conjunction "And". This is to say that enumerating heuristic rules can be an open-ended task. Among previous works in the literature on lyrics structure analysis, (Watanabe et al., 2016) heavily exploited repeated patterns present in the lyrics to address this task, and it shows that this general class of pattern is very helpful with segment border detection.

For this reason in this paper we follow (Watanabe et al., 2016) by casting the segmentation task as binary classification task. Each line of a song text is defined as either ending a segment or not. Let L_s be the lyrics of a song s composed of n lines of text: $L_s = \{l_1, l_2, ..., l_n\}$. Let $seg \subseteq (L_s, \mathbb{B})$ be a function that returns for each line $l_i \in L_s$ if it is the end of a segment. We define the segmentation task as learning a classifier that approximates seg. At the learning stage, the segment borders are observed from segmented text as double line breaks. At the testing stage the classifier has to predict the now hidden segment borders.

3 Modelling segments in song texts

In order to infer the lyrics structure, we propose a neural network-based model that *i)* efficiently relies on repeated patterns in a song text that are conveyed by the self-similarity matrix representations (SSM) used in the state-of-the-art (Watanabe et al., 2016), *ii)* can utilize traditional features alongside the SSMs (e.g., n-grams and characters count) and *iii)* jointly infers the structure of the entire text.

In the following, Section 3.1 describes the similarity measures that we have tested to produce the SSM representations, and Section 3.2 explains how such SSMs are produced. Section 3.3 lists the set of features used by the neural network-based model, and Sections 3.4 and 3.5 describe the model itself.

3.1 Similarity measures

The choice of the similarity measure is particularly important. Different measures and their combinations have been explored, for example, in musicology (Cohen-Hadria and Peeters, 2017). For our model, we produce SSMs based on three line-based similarity measures:

1. *String similarity* ($\text{sim}_{\text{string}}$): a normalized Levenshtein string edit distance between the characters of two lines (Levenshtein, 1966).

2. *Phonetic similarity* (sim_{phon}): a simplified phonetic representation of the lines computed using the "Double Metaphone Search Algorithm" (Philips, 2000). When applied to "i love you very much" and "i'l off you vary match" it returns the same result: "ALFFRMX". This algorithm was developed to capture the similarity of similar sounding words even with possibly very dissimilar orthography. After translating the words into this "phonetic language", we compute the character-wise edit distance like with string similarity.

3. *Lexico-structural similarity* ($\text{sim}_{\text{lex-struct}}$): this measure, initially proposed in (Fell, 2014), combines lexical with syntactical similarity. $\text{sim}_{\text{lex-struct}}$ captures the similarity between text lines such as "Look into my eyes" and "I look into your eyes": these are partially similar on a lexical level and partially similar on a syntactical level. Given two lines x, y lexico-structural similarity is defined as: $sim_{lex-struct}(x, y) = sim^2_{lex}(x, y) + (1 - sim_{lex}) \cdot sim_{struct}(\hat{x}, \hat{y})$, where sim_{lex} is the overlap of the bigrams of words in x and y, and sim_{struct} is the overlap of the bigrams of pos tags in $\hat{x}, \hat{y}$, the remaining tokens that did not overlap on a word level.

3.2 Self-similarity Matrices

We start by constructing features for each target lyrics. First, the line-based self-similarity matrices are produced to capture repeated patterns in a text. Such representations have been previously used in the literature to estimate the structure of music (Foote, 2000; Cohen-Hadria and Peeters, 2017) and lyrics (Watanabe et al., 2016). Given a text with n lines, a self-similarity matrix $SSM_M \in \mathbb{R}^{n \times n}$ is constructed, where each element is set by computing a similarity measure between the two corresponding lines $(SSM_M)_{ij} = sim_M(l_i, l_j)$, where sim_M is one of the similarity methods we defined. As a result, SSMs highlight distinct blocks of the target text revealing the underlying structure.

Figure 2 shows an example of lyrics consisting of five segments (left) along with its $\text{SSM}_{\text{string}}$ encoding (right). The segment borders are indicated by green lines. The repeated patterns in the song text are revealed by its SSM. This is illustrated as the song text (left) is nicely aligned to the SSM (right).

There are two common patterns that were investigated in the literature: *diagonals* and *rectangles*. Diagonals parallel to the main diagonal indicate text sequences that repeat and are typically found in a chorus. Rectangles, on the other hand, indicate text sequences in which all the lines are highly similar to one another. Both of these patterns were found to be indicators of segment borders.

3.3 Final features

After the SSMs were produced, each possible segment border was represented by the surrounding lines in a context window. Formally, given a fixed window size w_{size}, for each line l_i the submatrix (or patch) of the SSM was selected: $P_i = SSM_M[i - w_{\text{size}} + 1, ..., i + w_{\text{size}}; *] \in \mathbb{R}^{2w_{\text{size}} \times d}$. The resulting submatrices are the SSM features used for classification ($\text{SSM}_{\text{string}}$, SSM_{phon}, and $\text{SSM}_{\text{lex-struct}}$).

In addition to similarity matrices, we extracted the character count from each line, a simple proxy of the orthographic shape of the song text. Intuitively, segments that belong together tend to have similar shapes. Finally, we extracted n-grams (similar to (Watanabe et al., 2016)'s term features) from each line that were most indicative for segment borders, using the tf-idf weighting. We extracted n-grams that are typically found left or right from the border, varied n-gram lengths and also included indicative PoS tag n-grams. This resulted in 240 term features in total. The most indicative words at the start of a segment were: {ok, lately, okay, yo, excuse, dear, well, hey}. As segment-initial phrases we found: {Been a long, I've been, There's a, Won't you, Na na na, Hey, hey}. Typical words ending a segment were: {..., ..,

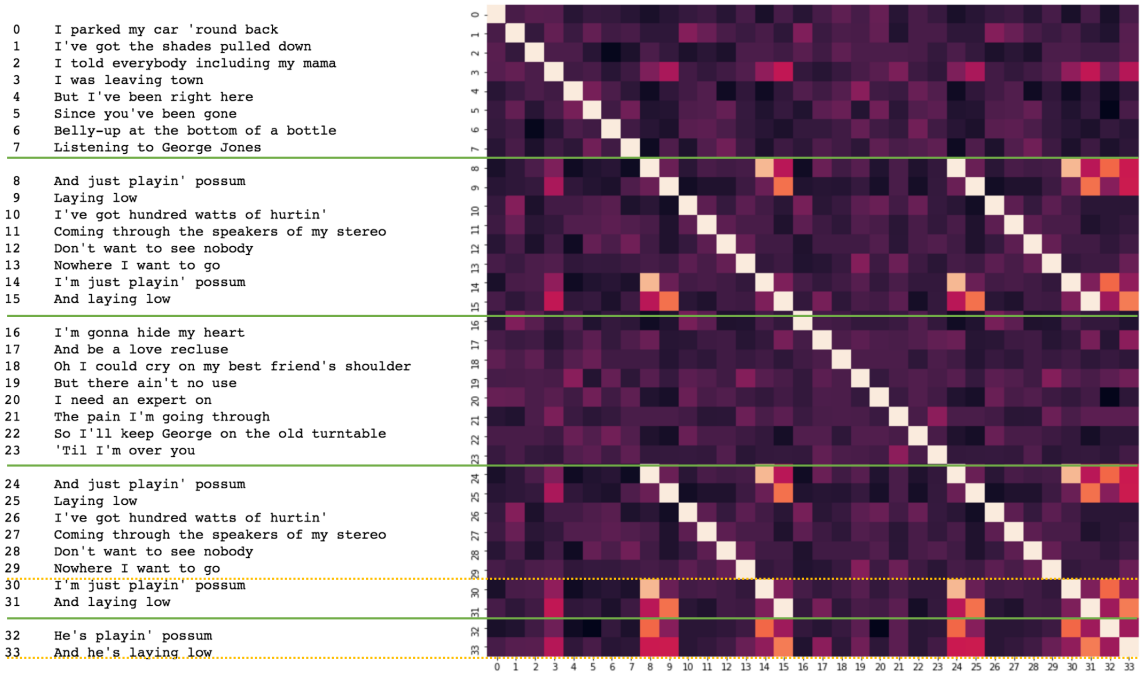


Figure 2: Green lines indicate the true segment borders. Diagonal stripe patterns from (8,24) to (15,31) co-occur well with the borders of the true segment from line 8 to line 15. Note, there is also a large square pattern from (30,30) to (33,33) indicated by the yellow lines - this is a highly repetitive musical state, but not a true segment. This is a typical false positive segment. ("Don't Rock The Jukebox" by Alan Jackson, MLDB-ID: 2954)

!, ., yeah, ohh, woah. c'mon, wonderland}. And as segment-final phrases we found as most indicative: {yeah!, come on!, love you., !!!, to you., with you., check it out, at all., let's go, ...}

3.4 Convolutional Neural Network-based Model

Figure 3 outlines our approach. The goal of the model is to predict if the segment border should be placed after the given line in a text. So, for each line, the model receives patches extracted from the precomputed SSMs: $\text{input}_i = \{P_i^1, P_i^2, ..., P_i^c\} \in \mathbb{R}^{2w_{\text{size}} \times d \times c}$, where c is the number of SSMs or number of channels.

Then, the input goes through a series of convolutions. Convolutional layers (Goodfellow et al., 2016) have been extensively used in image processing to allow the neural network to extract translation, scaling, and rotation invariant features anywhere on the input image. We employ the same motivation here: segment borders manifest themselves in the form of distinct patterns in the SSM. Thus convolutions allow the network to capture those patterns efficiently regardless of their location and relative size.

The first convolution uses a filter of size $(w_{\text{size}}+1) \times (w_{\text{size}}+1)$ so that each feature extracted captures a prospective segment border. The following max pooling layer downsamples the resulting tensor by w_{size} on both dimensions. The second convolution has a filter of size $1 \times w_{\text{size}}$ and the second max pooling layer further downsamples each feature to a scalar. Convolutions employ the ReLU function for activation. After the convolutions, the resulting feature vector is concatenated with the line-based features described above and goes through a series of densely connected layers with $tanh$ as the activation function. Then the $softmax$ is applied to produce probabilities for each class (border/not border). The model is trained using Adam (Kingma and Ba, 2014) with the cross-entropy as an objective function.

Finally, after repeating the procedure for each line in the text and picking the most probable class, we acquire the inferred structure for the entire text. Note that the usage of the self-similarity matrix as input allows us to make predictions based on the entire text: each input patch contains similarity scores between the lines surrounding the provisional segment border and all others. As shown in Section 4, CNN-based models significantly outperform the state-of-the-art.

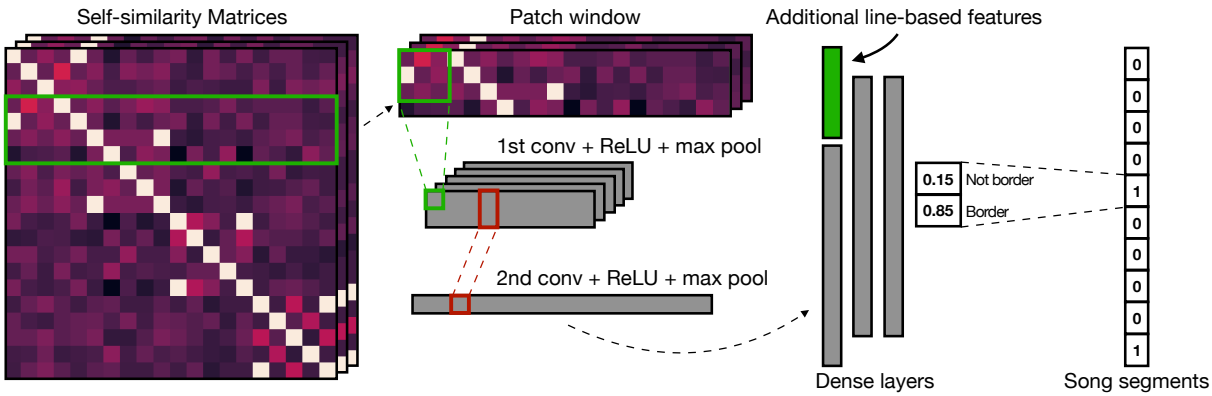


Figure 3: Convolutional Neural Network-based model inferring text structure.

3.5 Joint Predictions via a Recurrent Layer

Following the CNN-based model, we investigate the possibility of making segment border predictions jointly for the entire input text instead of producing labels for a single line at the time. The idea is to refine predictions for the target line using previous predictions. With this in mind, instead of producing the label directly, dense layers are wired through an additional recurrent layer.

Recurrent layers (Goodfellow et al., 2016) consist of cells containing a state that is able to carry information from the previous predictions for the same song and combine it with the current input to produce a refined output. We use LSTM cells and $tanh$ as an activation function. Each line is fed consecutively through the convolutional layers into the RNN cell, which outputs class probabilities.

4 Experimental setting

This section describes the experiments carried out to perform the segment border classification. First, we describe the different models and configurations that we have investigated (Section 4.1), the datasets on which we have run our method evaluation (Section 4.2), and then we discuss the obtained results (Section 4.3).

4.1 Models and configurations

We compare to the state-of-the-art (Watanabe et al., 2016) and successfully reproduce their best features to validate their approach. Two groups of features are used in the replication: repeated pattern features (RPF) extracted from SSMs and n-grams extracted from text lines.

Then, our own models are neural networks (CNNs and RNNs) that use as features SSMs, n-grams (as described in Section 3.3), and character count. We index the SSM features according to the similarity they embed (see Section 3.1): $\text{SSM}_{\text{string}}$ embeds the string similarity, SSM_{phon} the phonetic similarity and $\text{SSM}_{\text{lex-struct}}$ the lexico-structural similarity. The feature set SSM_{all} means that all SSMs are used in parallel. For convolutional layers we empirically set $w_{\text{size}} = 2$ and the amount of features extracted after each convolution to 128. Dense layers have 512 hidden units, the size of the LSTM hidden state is 25. We have also tuned the learning rate (negative degrees of 10), the dropout probability with increments of 0.1. The batch size was selected from the beginning to be 256 to better saturate our GPU. Both the CNN and RNN models were implemented using Tensorflow.

For comparison, we implement two baselines, i.e. the random baseline (as defined in (Watanabe et al., 2016)), and a logistic regression that uses the character count of each line as feature. For the simple character count baseline we used sklearn.LogisticRegression with the option class_weight='balanced' to account for highly imbalanced data.

4.2 Datasets

Song texts are available widely across the Web in the form of user-generated content. Unfortunately for research purposes, there is no comprehensive publicly available online resource that would allow a more standardized evaluation of research results. This is mostly attributable to copyright limitations and has been criticized before in (Mayer and Rauber, 2011). Research therefore is usually undertaken on corpora that were created using standard web-crawling techniques by the respective researchers. Due to the user-generated nature of song texts on the Web, such crawled data is potentially noisy and heterogeneous, e.g. the way in which line repetitions are annotated can range from verbatim duplication to something like *Chorus (4x)* to indicate a triple repetition of a chorus.

To compare our approach with the state of the art, as a first corpus we selected the English part of the Music Lyrics Database V.1.2.7, a proprietary lyrics corpus previously used in (Watanabe et al., 2016). We call this corpus henceforth MLDB. Like (Watanabe et al., 2016) we only consider those song texts that have five or more segments. We use the same training, development and test indices, which is a 60%-20%-20% split. In total we have 102802 song texts with at least 5 segments. 92% of the remaining song texts count between 6 and 12 segments. Evaluation metrics are precision, recall, and f-score.

To test the system portability to bigger and more heterogeneous data sources, we selected the WASABI corpus[3] (Meseguer-Brocal et al., 2017), which is a larger corpus of song texts (henceforth called WASABI). This dataset contains 743939 English song texts with at least 5 segments, and for each song it provides the following information: its lyrics[4], the synchronized lyrics when available[5], DBpedia abstracts and categories the song belongs to, genre, label, writer, release date, awards, producers, artist and/or band members, the stereo audio track from Deezer, when available, the unmixed audio tracks of the song, its ISRC, bpm, and duration. Moreover, we aligned MLDB to WASABI as the latter provides genre information. Song texts that had the exact same title and artist names (ignoring case) in both data sets were aligned. This rather strict filter resulted in an amount of 58567 (57%) song texts with genre information in MLDB. Table 2 shows the distribution of the genres in MLDB song texts. Thanks to the alignment with WASABI, we were able to group MLDB lyrics according to their genre. We then tested our method on each subset separately, to verify our intuition that classification is harder for some genres in which almost no repeated patterns can be detected (as Rap songs). To the best of our knowledge, previous work did not report on genre-specific results.

In this work we did not normalize the lyrics in order to rigorously compare our results to (Watanabe et al., 2016). We estimate the proportion of lyrics containing tags such as *Chorus* to be marginal (0.1-0.5%) in the MLDB corpus. When applying our methods for lyrics segmentation to lyrics found online, an appropriate normalization method should be applied as a pre-processing step. For details on such a normalization procedure we refer the reader to (Fell, 2014), Section 2.1.

4.3 Results and discussion

Table 1 shows the results of our experiments on the MLDB dataset. We start by measuring the performance of our replication of (Watanabe et al., 2016)'s approach. This reimplementation exhibits 56.3% F_1, similar to the results reported in the original paper. The divergence could be attributed to a different choice of hyperparameters and feature extraction code. Much weaker baselines were explored as well. The random baseline resulted in 13.3% F_1, while the usage of simple line-based features, such as character count, improves this to 25.4%.

The best CNN-based model, SSM_{all}+n-grams, outperforms all our baselines reaching 67.4% F_1, 8.2% better than the results reported in (Watanabe et al., 2016). We perform an approximate randomization test of this model against all other models reported below. In every case, the performance difference is statistically significant ($p < .05$). Subsequent feature analysis revealed that the $\text{SSM}_{\text{string}}$ is by far the best individual SSM. The $\text{SSM}_{\text{lex-struct}}$ feature exhibits much lower performance, despite being a

[3] https://wasabi.i3s.unice.fr/

[4] Extracted from http://lyrics.wikia.com/

[5] From http://usdb.animux.de

Model	*Configuration*	*Precision[%]*	*Recall[%]*	*F_1[%]*
Baselines	Random	14.3	12.5	13.3
	Char count (CC)	16.7	52.8	25.4
(Watanabe et al., 2016)	RPF (replication)	48.2	67.8	56.3
	RPF	56.1	59.4	57.7
	RPF + n-grams	57.4	61.2	59.2
CNN	SSM_{string} + CC	70.4	63.0	66.5
	SSM_{phon} + CC	75.9	55.6	64.2
	$SSM_{lex\text{-}struct}$ + CC	74.8	50.0	59.9
	SSM_{all} + CC	74.1	60.5	66.6
	SSM_{all} + n-grams	72.1	63.3	**67.4**
RNN	SSM_{string} + CC	69.8	59.6	64.3
	SSM_{phon} + CC	65.3	63.1	64.2
	$SSM_{lex\text{-}struct}$ + CC	72.8	51.4	60.3
	SSM_{all} + CC	68.3	63.2	65.6
	SSM_{all} + n-grams	71.7	61.8	66.4

Table 1: Performances obtained by the different models and configurations on MLDB test set.

much more complex feature. We believe the lexico-structural similarity is much noisier as it relies on n-grams and PoS tags, and thus propagates error from the tokenizers and PoS taggers. The SSM_{phon} exhibits a small but measurable performance decrease from SSM_{string}, possibly due to phonetic features capturing similar regularities, while also depending on the quality of preprocessing tools and the rule-based phonetic algorithm being relevant for our song-based dataset. However, despite lower individual performance, SSMs are still able to complement each other with the SSM_{all} model yielding the best performance.

In addition, we test the performance of several line-based features on our dataset. Most notably, the n-grams feature provides a significant performance improvement producing the best model. The CC feature shows marginal improvement.

Finally, the best RNN-based model exhibits 66.4% F_1. As mentioned above, the motivation to use this model was to improve predictions on the subsequent lines in the text by leveraging previous predictions. We did not observe an improvement over the CNN-based model most likely due to SSMs already accommodating enough global information so that convolutional layers are able to utilize it for prediction efficiently. As a result, the introduction of an additional recurrent layer increased the complexity of the model without providing much of a benefit. However, we believe that the addition of a more substantial amount of local features could make this additional complexity relevant.

Results differ significantly based on genre. In Table 2 we report the performances of the SSM_{string} on subsets of songs divided by their musical genre. The model is trained on the whole MLDB dataset and separately tested on each subset. Songs belonging to genres such as Country, Rock or Pop, contain recurrent structures with repeating patterns, which are more easily detectable by the CNN-based algorithm. Therefore, they show significantly better performance. On the other hand, the performance on genres such as Hip Hop or Rap, is much worse. An SSM for a Rap song is depicted in Figure 4. As texts in this genre are less repetitive, the SSM-based features are becoming much less reliable to determine a song's structure.

To show the portability of our method to bigger and more heterogeneous datasets, we ran the CNN model on the WASABI dataset, obtaining the following results (very close to the ones obtained on MLDB dataset): precision: 67.4%, recall: 67.3%, f-score: 67.4% using the SSM_{string} features.

5 Related Work

Besides the work of (Watanabe et al., 2016) that we have discussed in detail in Section 3, only a few papers in the literature have focused on the automated detection of the structure of lyrics. (Mahedero et al., 2005) report experiments on the use of standard NLP tools for the analysis of music lyrics. Among the tasks they address, for structure extraction they focus on lyrics having a clearly recognizable structure (which is not always the case) divided into segments. Such segments are weighted following the results given by descriptors used (as full length text, relative position of a segment in the song, segment simi-

Genre	*Lyrics[#]*	*Precision[%]*	*Recall[%]*	F_1*[%]*
Rock	6011	73.8	57.7	64.8
Hip Hop	5493	71.7	43.6	54.2
Pop	4764	73.1	61.5	66.6
RnB	4565	71.8	60.3	65.6
Alternative Rock	4325	76.8	60.9	67.9
Country	3780	74.5	66.4	**70.2**
Hard Rock	2286	76.2	61.4	67.7
Pop Rock	2150	73.3	59.6	65.8
Indie Rock	1568	80.6	55.5	65.6
Heavy Metal	1364	79.1	52.1	63.0
Southern Hip Hop	940	73.6	34.8	47.0
Punk Rock	939	80.7	63.2	**70.9**
Alternative Metal	872	77.3	61.3	68.5
Pop Punk	739	77.3	68.7	**72.7**
Soul	603	70.9	57.0	63.0
Gangsta Rap	435	73.6	35.2	47.7

Table 2: SSM_{string} model performances across musical genres in the MLDB dataset. Underlined are the performances on genres with less repetitive text. Genres with highly repetitive structure are in bold.

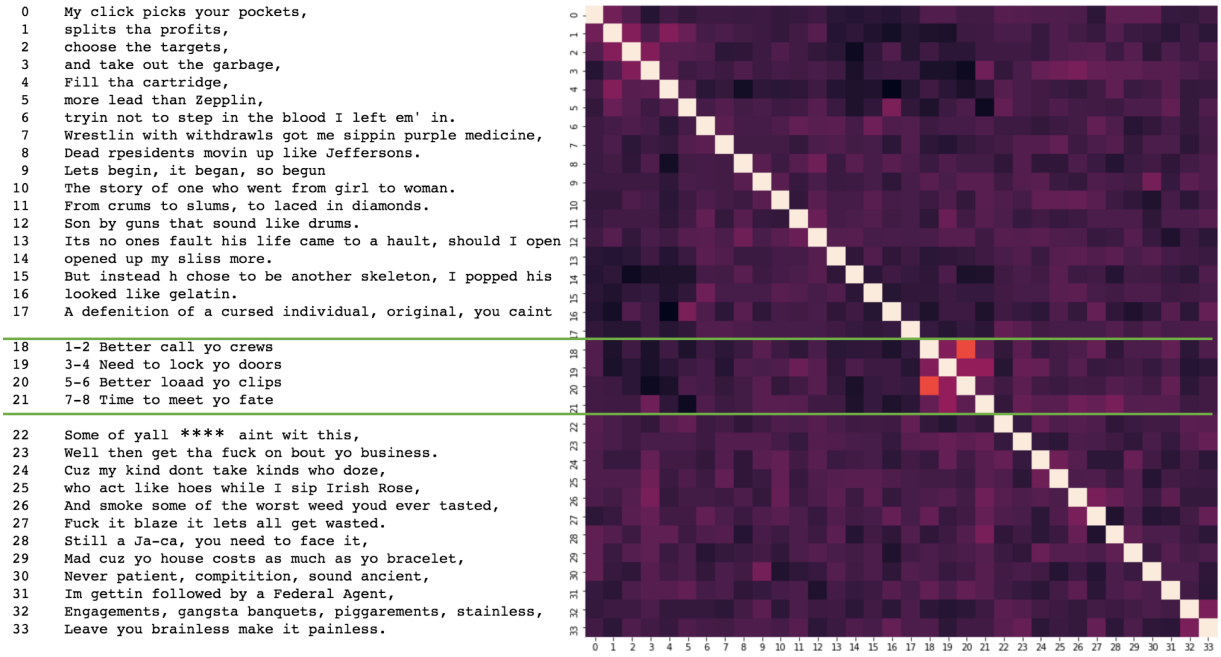


Figure 4: As common for Rap song texts, this example does not have a chorus (diagonal stripes). However, there is a highly repetitive musical state from line 18 to 21 which can be recognized by inspecting the SSM the from line 1 to line 4 is indicated by the corresponding rectangle in the SSM spanning from (18,18) to (21,21). ("Meet Your Fate" by Southpark Mexican, MLDB-ID: 125521)

larity), and then tagged with a label describing them (e.g. chorus, verses). They test the segmentation algorithm on a small dataset of 30 lyrics, 6 for each language (English, French, German, Spanish and Italian), which had previously been manually segmented.

More recently, (Baratè et al., 2013) describe a semantics-driven approach to the automatic segmentation of song lyrics, and mainly focus on pop/rock music. Their goal is not to label a set of lines in a given way (e.g. verse, chorus), but rather identifying recurrent as well as non-recurrent groups of lines. They propose a rule-based method to estimate such structure labels of segmented lyrics, while in our approach we apply ML methods to unsegmented lyrics.

To enhance the accuracy of audio segmentation, (Cheng et al., 2009) propose a new framework utiliz-

ing both audio and lyrics information for structure segmentation. For audio segmentation, the proposed constrained clustering algorithm improves the accuracy of boundary detection by introducing constraints on neighboring and global information. For semantic labeling, they derive the semantic structure of songs by lyrics processing to improve structure labeling.

With the goal of identifying repeated musical parts in music audio signals to estimate music structure boundaries (lyrics are not considered), (Cohen-Hadria and Peeters, 2017) propose to feed Convolutional Neural Network with the square-sub-matrices centered on the main diagonals of several self-similarity-matrices (SSM), each one representing a different audio descriptors, building their work on (Foote, 2000).

Lastly, the discourse segmentation literature heavily relies on segments to reflect topics and on segment borders to indicate topic changes. However, the segments in lyrics tend to be short and lyrics tend not to change topic a lot compared to longer prose texts. We experimented with word embeddings to measure topical similarity, but did not find it useful in our experiments. Still, integrating a domain-independent segmentation algorithm into our approach - such as the divisive clustering found in (Choi, 2000) - might be beneficial.

6 Conclusion and Future Work

Lyrics encode an important part of the semantics of a song and relying on their textual features could help us in overcoming the difficulties in audio processing for music management systems. Moreover the support for search criteria exploiting both the audio and the textual dimensions of a song would improve the usefulness of music search engine and retrieval systems by providing more and combinable selection and filtering dimensions.

In this paper, we focused on improving the state of the art in lyrics segmentation which is a prerequisite before considering the semantic labeling of these segments. We defined the task of classifying lyrics lines as segment borders or not, and we detailed the features we considered to feed to the classification methods. Having identified several feature configurations and state of the art classification methods we conducted an experiment and compared performance on the task of segment border classification. The best result (67.4 % f-score) was obtained using the Convolutional Neural Network with a self-similarity-matrix with layers combining all the similarity metrics and, in addition, the n-gram features. Moreover, at this stage, the CNN models outperformed the state of the art models using hand-crafted features[6].

For future work, we plan to complement this analysis with acoustic and cultural metadata (coupling our work for instance with (Cohen-Hadria and Peeters, 2017)), in the direction of developing a comprehensive music information retrieval systems over a real-world song collection.

Acknowledgements

The work described in this paper is carried out in the framework of the WASABi project, funded by the French National Research Agency (ANR-16-CE23-0017-01). Moreover, we want to thank the anonymous reviewers for their insightful comments.

References

A. Baratè, L. A. Ludovico, and E. Santucci. 2013. A semantics-driven approach to lyrics segmentation. In *2013 8th International Workshop on Semantic and Social Media Adaptation and Personalization*, pages 73–79, Dec.

D. Brackett. 1995. *Interpreting Popular Music*. Cambridge University Press.

M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. 2008. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, April.

H. T. Cheng, Y. H. Yang, Y. C. Lin, and H. H. Chen. 2009. Multimodal structure segmentation and analysis of music using audio and textual information. In *2009 IEEE International Symposium on Circuits and Systems*, pages 1677–1680, May.

[6]To allow for experimental reproducibility, the code is available at `https://github.com/TuringTrain/lyrics_segmentation`

Freddy YY Choi. 2000. Advances in domain independent linear text segmentation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 26–33. Association for Computational Linguistics.

Alice Cohen-Hadria and Geoffroy Peeters. 2017. Music Structure Boundaries Estimation Using Multiple Self-Similarity Matrices as Input Depth of Convolutional Neural Networks. In *AES International Conference Semantic Audio 2017*, Erlangen, Germany, June.

Michael Fell. 2014. Lyrics classification. Master's thesis, Saarland University, Germany.

Jonathan Foote. 2000. Automatic audio segmentation using a measure of audio novelty. In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, volume 1, pages 452–455. IEEE.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. 12.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Jose P. G. Mahedero, Álvaro MartÍnez, Pedro Cano, Markus Koppenberger, and Fabien Gouyon. 2005. Natural language processing of lyrics. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, MULTIMEDIA '05, pages 475–478, New York, NY, USA. ACM.

Rudolf Mayer and Andreas Rauber. 2011. Musical genre classification by ensembles of audio and lyrics features. In *Proceedings of the 12th International Conference on Music Information Retrieval*, pages 675–680.

Gabriel Meseguer-Brocal, Geoffroy Peeters, Guillaume Pellerin, Michel Buffa, Elena Cabrio, Catherine Faron Zucker, Alain Giboin, Isabelle Mirbel, Romain Hennequin, Manuel Moussallam, Francesco Piccoli, and Thomas Fillon. 2017. WASABI: a Two Million Song Database Project with Audio and Cultural Metadata plus WebAudio enhanced Client Applications. In *Web Audio Conference 2017 – Collaborative Audio #WAC2017*, London, United Kingdom, August. Queen Mary University of London.

Jose P. G. Mahedero, Alvaro Martinez, Pedro Cano, Markus Koppenberger, and Fabien Gouyon. 2005. Natural language processing of lyrics. pages 475–478, 01.

Lawrence Philips. 2000. The double metaphone search algorithm. 18:38–43, 06.

Philip Tagg. 1982. Analysing popular music: theory, method and practice. *Popular Music*, 2:37–67.

Kento Watanabe, Yuichiroh Matsubayashi, Naho Orita, Naoaki Okazaki, Kentaro Inui, Satoru Fukayama, Tomoyasu Nakano, Jordan Smith, and Masataka Goto. 2016. Modeling discourse segments in lyrics using repeated patterns. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1959–1969.

Learning What to Share: Leaky Multi-Task Network for Text Classification

Liqiang Xiao[1,2], **Honglun Zhang**[1,2], **Wenqing Chen**[1,2], **Yongkun Wang**[3], **Yaohui Jin**[1,2]
[1] State Key Lab of Advanced Optical Communication System and Network,
Shanghai Jiao Tong University
[2] Artificial Intelligence Institute, Shanghai Jiao Tong University
[3] Network and Information Center, Shanghai Jiao Tong University
{jinyh}@sjtu.edu.cn

Abstract

Neural network based multi-task learning has achieved great success on many NLP problems, which focuses on sharing knowledge among tasks by linking some layers to enhance the performance. However, most existing approaches suffer from the interference between tasks because they lack of selection mechanism for feature sharing. In this way, the feature spaces of tasks may be easily contaminated by useless features borrowed from others, which will confuse the models for making correct prediction. In this paper, we propose a multi-task convolutional neural network with the Leaky Unit, which has memory and forgetting mechanism to filter the feature flows between tasks. Experiments on five different datasets for text classification validate the benefits of our approach.

1 Introduction

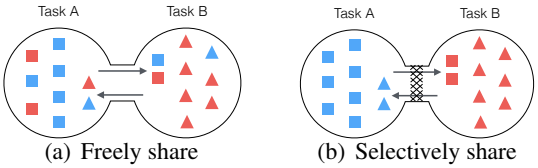
Convolutional neural network (CNN) models have achieved impressive results on many natural language processing (NLP) tasks, which use convolving filters to extract local features and have been successfully applied in computer vision field. In recent years, increasing works transfer CNNs to natural language processing tasks, such as representation learning (Liu et al., 2015), information retrieval (Shen et al., 2014), text classification (Kalchbrenner et al., 2014), etc. In particular, feed-forward CNNs with word embedding have been proven to be a relatively simple yet powerful kind of models for text classification (Kim, 2014).

However, the reliance on large-scale corpus has been a formidable constraint for deep neural networks (DNNs) based methods due to their numerous parameters. It costs a lot for a large-scale dataset to train the huge volume of parameters, because constructing a large-scale labeled dataset is extremely labor-intensive. To solve this problem, these models usually employ a pre-trained phase to map words into vectors with semantic implication (Collobert et al., 2011), which just introduces extra knowledge and does not directly optimize the target task. The problem of insufficiency for annotated resources is not intrinsically solved either.

Multi-task learning (MTL) can implicitly increase the corpus size and create a synergy effect among datasets (Caruana, 1997). By learning related tasks in parallel, MTL has the ability to exploit the relations between similar tasks to benefit each other, and eventually improves the performance for classification. In addition, models handling multiple tasks also benefit from a regularization effect to decrease the overfitting. It can help the models to learn a more universal representation for text sequences.

Inspired by this, more DNN-based models (Collobert and Weston, 2008; Liu et al., 2015; Liu et al., 2016) utilize multi-task learning to improve their performance. Traditionally, multi-task learning only shares a few shallow layers among tasks, so only low-level knowledge is exchanged to benefit each other (Collobert and Weston, 2008). But recently more works prefer to share deeper layers to share more high-level knowledge, which has been proven effective to enhance the performance (Liu et al., 2015; Liu et al., 2016).

Proceedings of the 27th International Conference on Computational Linguistics, pages 2055–2065
Santa Fe, New Mexico, USA, August 20-26, 2018.



(a) Freely share (b) Selectively share

Figure 1: Two schemes for sharing knowledge among tasks. Boxes and triangles denote the features for Task A and B respectively. The red boxes and blue triangles represent the features can be shared to benefit the other task.

The scheme for information sharing is the linchpin for designing a practical multi-task network. Most existing work attempts to find an appropriate proportion to sharing the layers between tasks, despite they entirely reuse some layers among tasks (Liu et al., 2015; Caruana, 1997) or add the tasks' layers up at a proportion (Fang et al., 2017). And recently, the latter architecture shows its advantages for controlling the relation intensity among tasks and becomes prevailing. More models adopt this thought to enhance the performance (Liu et al., 2015; Liu et al., 2016).

However, under the scheme of proportional addition (Ruder et al., 2017; Misra et al., 2016), all the features, between every pair of tasks, are shared in the same weight without selection. Helpless or harmful features if freely transported between tasks with the same importance as helpful ones, namely, the interference is generated just like Figure 1(a) shows. This would burden the network for distinguishing the helpful features and even mislead the predictions.

To resolve above problems, we propose a new CNN-based model *LK-MTL* for multi-task learning, which shares the features with a selective mechanism (Figure.1(b)). Our model employs a "split structure" that every task owns a private subnet and shares their features through a well-designed module—*Leaky Unit*, which has memory and forgetting mechanisms to control the feature flows, deciding what features should be shared or discarded. By that the feature flows are purified and the interference would be restrained.

We conduct extensive experiments on five benchmark datasets for text classification. And the result shows that our multi-task model gains great improvement over the single-task CNNs and other multi-task competitors.

The contributions of this paper can be briefly summarized as follows:

- Proposed Leaky Unit has the ability to remember and forget, which is effective for filtering the feature flows and can help multi-task learning dispel the interference among tasks.
- The architecture of our multi-task model combines the advantages of GRU and CNN. The performance is enhanced by both memory mechanism from gated recurrent unit (GRU) and the feature extraction ability from convolutional neural network.
- Our model achieves strong results on several benchmark classification datasets and outperforms the state-of-the-art baselines on four datasets.

2 CNN Models for Text Classification

The main capability of DNNs for text classification is to represent word sequences into fix-length vectors. There are many frameworks can be used for sentence modeling, involving Neural Bag-of-Words (NBOW) model, recursive neural network (RecNN) (Socher et al., 2012; Socher et al., 2013), recurrent neural network (RNN) (Chung et al., 2014) and convolutional neural network (CNN) (Collobert et al., 2011).

In recent years, CNN has shown its advantages in the NLP field. CNN models can not only handle input sentences of varying length but also capture short and long range relations through the feature graph over the sentence (Kalchbrenner et al., 2014). Most of the CNN models for text sequences are

constructed by alternating the convolution layers and pooling layers. In this paper, this kind of CNN architectures is defined as the subnet in our multi-task network, which can be detailed as follows.

2.1 CNN for Text Representation

Given a text sequence $x_{1:l} = x_1 x_2 \cdots x_l$, we first use a lookup table to get the embedding results (word vector) $\mathbf{x}_i$ for each word x_i. Then the input matrix can be represented as $\mathbf{x} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \cdots \oplus \mathbf{x}_l$ by concatenating each word vector, where $\oplus$ denotes the operation of concatenation. CNNs produce the representation of the input sequence through stacking the layer of convolution, pooling in order, which can be briefly formalized as:

$$\boldsymbol{F} = \mathbf{H} * \mathbf{x} \tag{1}$$

$$\hat{\boldsymbol{F}} = pooling(\boldsymbol{F}) \tag{2}$$

$$\mathbf{y} = \boldsymbol{w}\hat{\boldsymbol{F}} + \boldsymbol{b}, \tag{3}$$

where $\boldsymbol{H}$ is the filter for convolutional operation $*$; $pooling$ denotes the pooling operation; $\boldsymbol{F}$ and $\hat{\boldsymbol{F}}$ represent the feature maps; $\boldsymbol{w}$ and $\boldsymbol{b}$ denote the weight and bias respectively in fully connected layer. Usually, drop out mechanism is used as a kind of regularization for output layer and the Eq.3 can be rewrote as

$$\mathbf{y} = \boldsymbol{w}\hat{\boldsymbol{F}} \circ \boldsymbol{r} + \boldsymbol{b}. \tag{4}$$

Here $\circ$ refers to the element-wise multiplication operator, and $\mathbf{r} \in \mathbb{R}^d$ is a "masking" vector comprising Bernoulli random variables with probability p to be 1.

2.2 Output Layer for Text Classification

Following the pooling layer, a fully connected layer with dropout and the softmax layer was used, which transforms the vector representation into the probability distribution over classes.

During backpropagation, the parameters in the network are updated by gradient of the loss between the predicted and true distributions. Such as cross-entropy function

$$L(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{i=1}^{N}\sum_{j=1}^{C} \mathbf{y}_i^j log(\hat{\mathbf{y}}_i^j), \tag{5}$$

where $\hat{\mathbf{y}}$ is the predicted probability distribution; $\mathbf{y}$ is the ground-truth label; N and C are the batch size and the number of classes respectively.

3 Multi-Task CNN for Text Classification

The goal of multi-task learning is to utilize the connection among these related tasks to improve classification when learning tasks in parallel. Recently, "split architecture" that provides each task a private subnet is widely accepted by researchers, since it has ability to make a balance between task specific features and shared features. Hence, the key factor of multi-task learning is the fusion mechanism for the information flow between tasks. In CNN models, the information lies in the feature maps. Therefore, the biggest difference of the multi-task architecture is the method for fusing the feature maps. Here we introduce a typical architecture for CNN based multi-task models: PA-MTL.

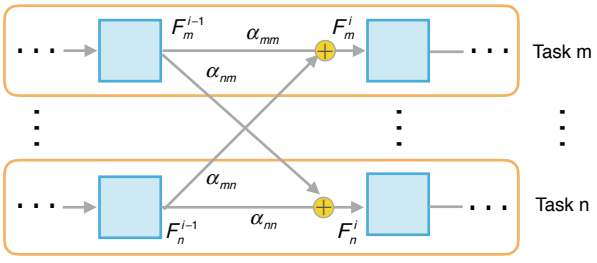


Figure 2: Illustration of proportional addition model.

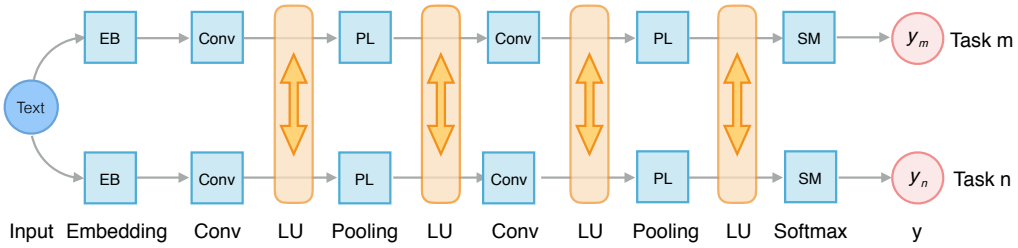


Figure 3: Illustration of the architecture of Leaky Multi-Task Network. LU denotes the Leaky Unit.

Proportional Addition Model (PA-MTL) Generally, most existing multi-task CNN models share the information between tasks by proportional addition (Misra et al., 2016; Fang et al., 2017). As Figure 2 shows, they share the information by adding the feature maps between tasks m, n with scalar weights α^{i-1}. α^{i-1} is updated by back-propagation, which reflects the strength of association between tasks but has no selection for the features. Between the $i-1$-th and i-th layers, the whole process for fusing the feature maps F^{i-1} from M tasks can be formulated as:

$$\begin{bmatrix} \mathbf{F}_1^i \\ \vdots \\ \mathbf{F}_M^i \end{bmatrix} = \begin{bmatrix} \alpha_{11} & \cdots & \alpha_{1M} \\ \vdots & \ddots & \vdots \\ \alpha_{M1} & \cdots & \alpha_{MM} \end{bmatrix} \begin{bmatrix} \mathbf{F}_1^{i-1} \\ \vdots \\ \mathbf{F}_M^{i-1} \end{bmatrix}, \tag{6}$$

4 Leaky Multi-Task CNN

The key factor for multi-task learning is the scheme for sharing the features. So an elaborate architecture that can well control the feature sharing among tasks is very crucial for multi-task learning. A good architecture is supposed to not only fully exchange the features to help extend tasks' feature space, but also select the helpful features to avoid the contamination. So, in this section, we propose a selective architecture to optimize the sharing scheme.

4.1 Model Architecture Choice

Multi-task model with deeper layers shared can fuse high-level knowledge and greatly increase the feature space. But undesirable interference is inevitable and simultaneously comes with the benefits, especially between the less-related tasks. This would burden the models with the overhead on distinguishing helpful features. To overcome above problem, we explore another structure— *Leaky Multi-Task CNN (LK-MTL)*, in which the tasks have ability to selectively borrow helpful knowledge from others.

As shown in Figure 3, in order to reduce the interference, we assign each task a private subnet, under which circumstance the information is shared indirectly and easy to control. These relatively separated tasks can only borrow information from others through the bridge: *Leaky Unit*. Memory mechanism is employed in this unit to control the feature sharing. The parameters are updated through backpropagation to optimize the selection without needing for extra supervision. This is an end-to-end and easy-training method and its thought is easy to be employed by other works. The convolutional neural network can be easily replaced by multi-layer perceptrons or recurrent neural networks for other applications.

4.2 Leaky Unit

In this separate architecture, we design a module named *Leaky Unit* to selectively share the information between tasks, which is inspirited by the hidden unit of Gated Recurrent Unit (GRU) (Cho et al., 2014), a variant of Leaky Integration Unit proposed by (Bengio et al., 2013). The hidden state of GRU fuses the cell and hidden state of standard LSTM to make the structure easier to compute and implement, which has a sophisticated mechanism for remembering and forgetting features. Thus we make some revision and use it as a tool to filter the feature flows between tasks. Leaky Unit addresses two problems in the process of sharing: what features could be helpfully borrowed from other tasks and How many features should be preserved for current task?

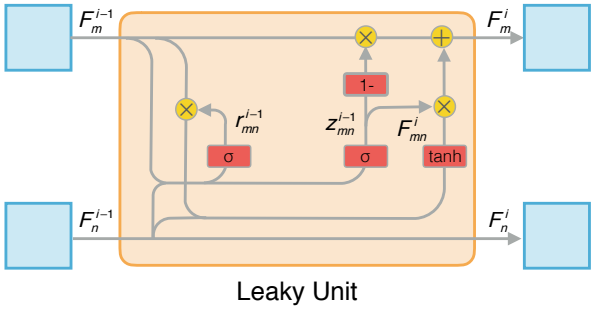


Figure 4: Illustration of the details for Leaky Unit

As shown in Figure 4, a *Leaky Gate* r_{mn} is first generated from the feature maps in $(i-1)$-th layers

$$\mathbf{r}_{mn}^{i-1} = \sigma(\mathbf{W}_r^{i-1} \cdot [\mathbf{F}_m^{i-1}, \mathbf{F}_n^{i-1}]), \tag{7}$$

where mn means the direction of feature flow is from task n to task m; σ denotes the logistic sigmoid function, which limits the values into $[0, 1]$; $\mathbf{F}_m^{i-1}$, $\mathbf{F}_n^{i-1}$ is the feature maps from task m, n respectively. Leaky gate decides what features will be leaked in from other tasks. Hence, a new feature map is calculated by

$$\tilde{\mathbf{F}}_{mn}^i = tanh(\mathbf{U}^{i-1} \cdot \mathbf{F}_m^{i-1} + \mathbf{W}^{i-1} \cdot (\mathbf{r}_{mn}^{i-1} \odot \mathbf{F}_n^{i-1})). \tag{8}$$

When the j-th element in vector $\mathbf{r}_{mn}^{i-1}$ is close to 0, the corresponding feature $[\mathbf{F}_n^{i-1}]_j$ in task n will be discarded. On the contrary, feature $[\mathbf{F}_n^{i-1}]_j$ will be kept and passed to task m.

In addition, we employ another *Update Gate* z_{mn}^{i-1} to determine how much information should be maintained from current task m into the next layer, which is emitted by

$$\mathbf{z}_{mn}^{i-1} = \sigma(\mathbf{W}_z^{i-1} \cdot [\mathbf{F}_m^{i-1}, \mathbf{F}_n^{i-1}]). \tag{9}$$

And final output for current task m is calculated by

$$\mathbf{F}_m^i = \mathbf{z}_{mn}^{i-1} \cdot \mathbf{F}_m^{i-1} + (1 - \mathbf{z}_{mn}^{i-1}) \cdot \tilde{\mathbf{F}}_{mn}^i. \tag{10}$$

If we consider all the directions of information flows and extend above formulations into all M tasks, the output of Leaky Unit is the sum of each row in

$$\begin{bmatrix} \sum_{k=1}^M \mathbf{z}_{1k}^{i-1} & (1-\mathbf{z}_{12}^{i-1}) & \cdots & (1-\mathbf{z}_{1M}^{i-1}) \\ (1-\mathbf{z}_{21}^{i-1}) & \sum_{k=1}^M \mathbf{z}_{2k}^{i-1} & \cdot & 1-\mathbf{z}_{2M}^{i-1}) \\ \vdots & \vdots & \ddots & \vdots \\ (1-\mathbf{z}_{M1}^{i-1}) & (1-\mathbf{z}_{M2}^{i-1}) & \cdots & \sum_{k=1}^M \mathbf{z}_{Mk}^{i-1} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{F}_1^{i-1} & \mathbf{F}_{12}^{i-1} & \cdots & \mathbf{F}_{1M}^{i-1} \\ \tilde{\mathbf{F}}_{21}^i & \mathbf{F}_2^{i-1} & \cdots & \tilde{\mathbf{F}}_{2M}^i \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{\mathbf{F}}_{M1}^i & \tilde{\mathbf{F}}_{M2}^i & \cdots & \tilde{\mathbf{F}}_M^i \end{bmatrix} /M. \tag{11}$$

Leaky units adjust the states of two gates according to the helpfulness of other tasks' features. The units tend to borrow more features from other tasks will more frequently activate the leaky gate. On the contrary, the update gate will be more active to preserve more information in current task.

4.3 Training

In the last layer of model, the vector representations $\hat{\mathbf{F}}_m$ of input sequences are fed into different output layers to fit the number of class, which emits the prediction of probability distribution for task m

$$\hat{y}_m = softmax(\mathbf{W}_m \hat{\mathbf{F}}_m + b_m), \tag{12}$$

where $\hat{y}_m$ is predictive result, $\mathbf{W}_m$ is the weight of the full-connected layer, and $\mathbf{b}_m$ is the bias term.

Given the prediction of all tasks, a global loss function forces models to take every task into account.

$$\Phi = \sum_{m=1}^{M} \lambda_m L(\hat{y}_m, y_m), \tag{13}$$

Dataset	Target	Type	Train Size	Dev. Size	Test Size	Class	Avg. Length
SST-1	Sentiment	Sentence	8544	1101	2210	5	19
SST-2		Sentence	6920	872	1821	2	19
IMDB		Document	25000	-	25000	2	279
SUBJ	Subjectivity	Sentence	9000	-	1000	2	21
QC	Question Types	Sentence	5153	-	489	6	10

Table 1: Statistics of the five text classification datasets. Dev. and Avg. are the abbreviations of development and average respectively.

where λ_m is the weight for the task m. In this paper, we simply set λ_m to $1/M$ for all M tasks to make a balance.

In order to train the parameters with different datasets, following (Collobert and Weston, 2008), each task is trained by turn in a stochastic manner. The steps can be described as follows:

1. Pick up a task m randomly;
2. Select an arbitrary sample s from the task m;
3. Feed the sample s into the model and update the parameters;
4. Go back to 1.

Following the training phase, we employ fine tuning strategy (Liu et al., 2016) to further optimize the parameters for each task alone.

5 Experiments

In this section, we investigate the empirical performances of our models on five related benchmark tasks for text classification. And the results are compared with the state-of-the-art models. Furthermore, we also intuitively show the operation mechanism of leaky unit via visualization.

5.1 Datasets

As Table 1 shows, we collect five benchmark datasets for text classification that are related to each other to different degree. These datasets contain sentiment, subjectiveness and question type classification, which belong to different class of target and can be briefly introduced as follows:

- **SST-1** Stanford Sentiment Treebank[1] (Socher et al., 2013), a movie review dataset with five classes of labels (very positive, positive, neutral, negative, very negative), and split into train/dev/test three parts.

- **SST-2** Also from the Stanford Sentiment Treebank, but with neutral reviews removed and binary labels.

- **SUBJ** Subjectivity dataset[2] that the task is to classify a sentence level text as being subjective or objective (Pang et al., 2004).

- **IMDB** Binary class dataset[3] (Maas et al., 2011) consisting of 100,000 document-level movie reviews that are classified to be positive/negative.

- **QC** Question dataset classifying a given question into six types (whether the question subjects to person, location, numeric information, etc.) (Li and Roth, 2002).

[1] http://nlp.stanford.edu/sentiment.
[2] http://www.cs.cornell.edu/people/pabo/movie-review-data/
[3] http://ai.stanford.edu/ amaas/data/ sentiment/

Hyperparameter	Setting
Embedding size	300
Dropout rate	0.5
Mini-batch size	30
Learning rate	0.001
l_2 constraint	0.1
Filter size	3,4,5
Filter number	50×6

Table 2: Hyperparameter settings

Model		SST-1	SST-2	IMDB	SUBJ	QC
Single-Task	RNTN	45.7	85.4	-	-	-
	DCNN	48.5	86.8	-	-	-
	MGNC-CNN	48.7	88.3	-	94.1	-
	PV	44.6	82.7	**91.7**	90.5	91.8
Multi-Task[4]	MT-GRNN	49.2	87.7	91.6	-	92.3
	MT-RNN	49.6	87.9	91.3	94.1	-
	MT-CNN	49.0	86.9	91.5	94.1	92.0
	MT-DNN	48.1	87.3	91.4	94.0	92.1
	PA-MTL	48.2	87.1	90.7	94.0	91.5
Our Models	LK-MTL (hidden unit)	49.4	**88.6**	91.1	94.3	92.9
	LK-MTL (hidden unit + peephole)	49.6	88.2	91.3	94.1	**93.8**
	LK-MTL (leaky unit)	**49.7**	88.5	91.3	**94.5**	**93.8**

Table 3: Results of LK-MTL against other state-of-the-art models.

5.2 Hyperparameters and Training

Initializing word vectors with the dataset trained by an unsupervised neural network model is an efficient method to enhance the performance without a large-scale supervised training data (Collobert et al., 2011; Socher et al., 2011; Iyyer et al., 2014). In all of our experiments, our models contain a lookup table by employing Word2Vec (Mikolov et al., 2013) trained on Google News, which comprises more than 100B words with a vocabulary size of around 3M. Word2Vec derives from a continuous bag-of-words architecture and each vector has 300 dimensions.

Word vectors are further fine-tuned during training to get a more optimized embedding result that fits the datasets. The whole network is trained through stochastic gradient decent using Adadelta update rule (Zeiler, 2012). For datasets without development set, we randomly select 10% of the training data as the dev set. The key hyperparameters are chosen via a small grid search, and final setting is illustrated in Table 2.

5.3 Performance of Multi-task CNN

We simultaneously train our model LK-MTL on five datasets and compare it with single-task scenario in Table 3. We can see that our model improves the performance with a large margin over the single task models, which demonstrates the positive synergy of our multi-task architecture. This also testified our assumption that separate structure for MTL is conducive to the information sharing between tasks, helping making better representations for text sequences.

We also test several variants of LK-MTL. The first one replaces Leaky Unit by the Hidden Unit [5] of standard LSTM, and the second one further adds the peephole connections. The results reported in Table

[4]We use reported data for MT-GRNN and MT-RNN and implement MT-CNN, MT-DNN and PA-MTL since the authors do not release the code.

[5]To suit the multi-task learning architecture, hidden state h_t is removed from the formulations of hidden unit.

3 shows that our LK-MTL with Leaky Unit outperforms its variants, which proves the mechanism of Leaky Unit is more suitable for multi-task CNNs.

Comparisons with the State-of-the-art Models
We compare our proposed model against the following models:

- **RNTN** Recursive Neural Tensor Network with tensor-based feature function and parse trees (Socher et al., 2013).
- **DCNN** Convolutional Neural Network with a novel dynamic k-max pooling (Kalchbrenner et al., 2014)
- **MGNC-CNN** Multi-group norm constraint CNN. We use the result of MGNC-CNN(w2v+Syn+Glv) (Zhang et al., 2016).
- **PV** Paragraph vectors based logistic regression (Le and Mikolov, 2014).
- **MT-GRNN** A general multi-task learning architecture with Recurrent Neural Network (Zhang et al., 2017).
- **MT-RNN** Multi-task learning with Recurrent Neural Networks by a shared-layer architecture (Liu et al., 2016)
- **MT-DNN** Multi-Task learning with Deep Neural Networks that uses bag-of-word representations and a hidden shared layer (Liu et al., 2015).
- **MT-CNN** Multi-Task learning with Convolutional Neural Network that partially shares a lookup table (Collobert and Weston, 2008).

As shown in Table 3, LK-MTL also outperforms the state-of-the-art multi-task learning works in four datasets and show competitive results in the other one. Competitors slightly outperform our models in IMDB, since its unique character that the sequences are much longer than other datasets. Specifically, LK-MTL surpasses the PA-MTL, which result demonstrates that the memory and forgetting mechanisms in leaky unit indeed have advantages over other sharing approaches, which help distinguish the helpfulness of the features and making a balance between features in current task and borrowed features. In another word, task-specific feature and shared feature is organically fused by leaky unit in a better way.

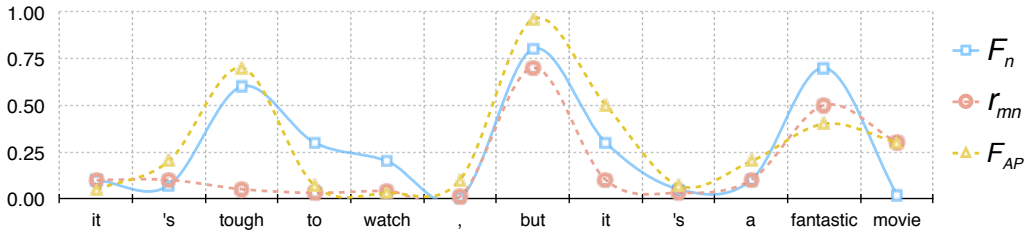


Figure 5: Lines illustrate the feature weights of $\mathbf{F}^1_{SUBJ}$ in SUBJ subnet. And red line shows the value of $\mathbf{r}^1_{SST-1 \leftarrow SUBJ}$ that filters the features from SUBJ subnet to SST-1 subnet. F_{AP} line visualizes the feature weights in the first layer of PA-MTL.

5.4 Visualization

To intuitively show the selection process in leaky unit, we design an experiment to show the values of gate and how they block the useless features. For the first convolutional layer and leaky unit, we visualize the activations $\mathbf{F}^1_m$ of the filters with normalized values and show their corresponding leaky gate $\mathbf{r}^1_{mn}$ in the gate units. By that we can easily find what kind of features are discarded as interference.

Figure 5 illustrates the behavior of leaky unit on a random selected sentence from test set of SST-2 task. We visualize the results of the first feature map for SUBJ subnet and the leak gate $\mathbf{r}^1_{mn}$ that filters the features from SUBJ to SST-2 task. For the positive sentence "it's tough to watch, but it's a fantastic movie", we can see that subnet for SUBJ task focuses on two critical positions "tough" and "fantastic". The word "tough" is subjective for SUBJ task, so the subnet focuses on that world, but actually it is useless and may mislead the prediction of SST-2 task. Successfully, our leaky gate lowers the intensity of that interference "tough", making a correct prediction. However, PA-MTL wrongly makes a negative prediction for lacking resistance to interference. This indicates the effectiveness of our memory and forgetting mechanism in the leaky unit.

6 Related Work

In the NLP field, NN-based multi-task learning has been proven to be effective (Collobert and Weston, 2008; Liu et al., 2015; Liu et al., 2016). The synergy between multi-task learning and neural network is obvious. The earliest idea can be traced back to (Caruana, 1997).

(Collobert and Weston, 2008) develops a multi-task learning model based on CNN. It shares only the partial lookup table to train a better word embedding. And (Liu et al., 2015) proposes a DNN based model for multi-task learning, which shares some low layers to represent the text. These works allow the tasks to reuse low-level layers but separate the high-level layers.

Some models are proposed to sharing deeper layer of networks, which can exchange high level knowledge among tasks and gain better performance. (Liu et al., 2016) and (Zhang et al., 2017) introduce some RNN architectures and design different schemes for knowledge sharing among tasks. These trials promote the performance of models, but they give no consideration to the interference in multi-task learning.

Thus, more approaches are proposed to reduce the interference among tasks. One is to strengthen connection between the strong-related tasks and weaken the less-related ones. (Misra et al., 2016; Fang et al., 2017) proposes a model for images, trying to reduce the interference through weighting the connections between tasks and finding an appropriate proportion. Though this kind of methods weakens the noise from less-related tasks, the processing is coarse-grained and cannot distinguish the useful information in smaller feature level.

Different from these models, our model control the information flows between tasks in feature level, which uses the mechanism of remembering and forgetting, passing only the useful features for the current task. It solves the problems of what should be maintained in current task and what should be borrowed from other tasks. By that our model is capable of reducing the interference without extra supervision.

7 Conclusion and Future Work

In this paper, we propose a CNN based framework for multi-task learning, which has the structure to control the passing of information in feature level. By remembering the helpful features and forgetting the useless ones, LK-MTL can reduce the interference between the tasks. And we also visualize the property of our models and intuitively show the selection mechanism of Leaky Unit. Experiment result on five datasets for text classification demonstrates the effectiveness of our model for text representation.

In the feature, we would like to investigate deeper into the interference problem, testing more kinds of gate mechanisms and find better one to purify the feature flows. We also want to equip our unit on other neural networks for other applications, such as multi-layer perceptrons and RNNs.

8 Acknowledge

We appreciate the constructive advices from Naoki Yoshinaga and the valuable comments from anonymous reviewers. We also thank Xuan Luo for building and maintaining the GPU platforms. This research was funded by National Natural Science Foundation of China under Grant No. 61371048.

References

Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. 2013. Advances in optimizing recurrent networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8624–8628.

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

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Computer Science*.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pages 160–167.

Ronan Collobert, Jason Weston, L Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(1):2493–2537.

Yuchun Fang, Zhengyan Ma, Zhaoxiang Zhang, Xu-Yao Zhang, and Xiang Bai. 2017. Dynamic multi-task learning with convolutional neural network. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 1668–1674.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Meeting of the Association for Computational Linguistics*, pages 1113–1122.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *Computer Science*, 4:1188–1196.

Xin Li and Dan Roth. 2002. Learning question classifiers. *Coling*, 12(24):556–562.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 912–921.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26:3111–3119.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June.

Pang, Bo, Lee, and Lillian. 2004. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. *Proceedings of Acl*, pages 271–278.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Sgaard. 2017. Sluice networks: Learning what to share between loosely related tasks.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 101–110.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John Mcintyre Conference Centre, Edinburgh, Uk, A Meeting of Sigdat, A Special Interest Group of the ACL*, pages 151–161.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *Computer Science*.

Ye Zhang, Stephen Roller, and Byron C. Wallace. 2016. MGNC-CNN: A simple approach to exploiting multiple word embeddings for sentence classification. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1522–1527.

Honglun Zhang, Liqiang Xiao, Yongkun Wang, and Yaohui Jin. 2017. A generalized recurrent neural architecture for text classification with multi-task learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 3385–3391.

Towards an argumentative content search engine using weak supervision

Ran Levy,* **Ben Bogin,*** **Shai Gretz,*** **Ranit Aharonov, Noam Slonim**
IBM Research
{ranl,boginb,avishaig,ranita,noams}@il.ibm.com

Abstract

Searching for sentences containing claims in a large text corpus is a key component in developing an argumentative content search engine. Previous works focused on detecting claims in a small set of documents or within documents enriched with argumentative content. However, pinpointing relevant claims in massive unstructured corpora, received little attention. A step in this direction was taken in (Levy et al., 2017), where the authors suggested using a weak signal to develop a relatively strict query for claim–sentence detection. Here, we leverage this work to define weak signals for training DNNs to obtain significantly greater performance. This approach allows to relax the query and increase the potential coverage. Our results clearly indicate that the system is able to successfully generalize from the weak signal, outperforming previously reported results in terms of both precision and coverage. Finally, we adapt our system to solve a recent argument mining task of identifying argumentative sentences in Web texts retrieved from heterogeneous sources, and obtain F_1 scores comparable to the supervised baseline.

1 Introduction

The arguments raised during a decision making process, will often determine its outcome. A common component in all argument models (e.g., (Toulmin, 2003)) is the *claim*, i.e. the assertion the argument aims to prove. The problem of automatically detecting claims supporting or contesting a given controversial topic [1] (Levy et al., 2014) is considered a fundamental task in the emerging field of computational argumentation (Lippi and Torroni, 2016; Palau and Moens, 2009). We refer to their definition of a Topic and a Claim; **Topic** - a short phrase that frames the discussion and **Context Dependent Claim** - a general, concise statement that directly supports or contests the given Topic (we henceforth use the term claim instead of Context Dependent Claim).

Previous works have focused on detecting claims within a small set of documents related to the topic (Levy et al., 2014), or within documents enriched with argumentative content (Stab and Gurevych, 2014). However, pinpointing relevant claims within massive unstructured corpora, received relatively little attention. While this problem is obviously more challenging, its potential value is also much higher. For a widely discussed topic, one should expect many relevant claims to be mentioned across a widespread set of articles in the given corpus. The remaining issue is to develop a technology to swiftly detect these claims and present the results to potential users, similarly to search engines that retrieve information in response to a query.

A step in this direction was taken in (Levy et al., 2017). They suggested a relatively strict sentence–level query (strict in the sense that it considerably limits the set of potential answers hence reduces the coverage). Their query combines three query parts that must appear in order, with possible gaps between them. The first part requires the sentence to contain the token *'that'* as it is often a precursor for a claim (e.g. <someone> argued that <claim>). The second query part requires some restriction on the scope of topics the system can handle, and assumes that each topic deals with exactly one concept (denoted

*First three authors contributed equally.

[1] We will henceforth refer to claims supporting or contesting a given controversial topic as *relevant claims*.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2066–2081
Santa Fe, New Mexico, USA, August 20-26, 2018.

MC – for main concept) and that this concept has a Wikipedia title (e.g. Affirmative Action). The second part of the query thus restricts the returned sentences to those in which the MC follows the word *'that'* (possibly with a gap). The third and final query part requires a token from a pre–specified *claim lexicon* (CL) to appear after the MC (possibly with a gap). The CL lexicon aims to characterize *claim sentences* (CS) and the process of its creation did not involve any labeled data. Table 1 shows the fifty most indicative tokens from the lexicon. Relying on this formulation led to promising precision results in the challenging task of corpus wide claim detection, albeit with low recall . Specifically, while each of the sentences in Table 2 contains a valid and relevant claim, only $S1$ satisfies their query; In contrast, $S2$ satisfies only the first part of the query ('that' preceding the MC); $S3$ satisfies the second part of the query (CL token following the MC); and $S4$ only mentions the MC. By construction, these latter three sentences, are out of the radar of Levy et al. (2017).

Claim Lexicon - partial
should, tuned, could, unconstitutional, violate, might, violated, violates, wrong, rather, valid, invalid, irrelevant, inherently, necessarily, cannot, prevail, justify, flawed, merely, corpus, ought, inevitably, cause, justifiable, unacceptable, untrue, abhorrent, unless, harmful, punished, liable, incompatible, beneficial, justifying, undecided, skimmed, indefensible, impossible, undermine, necessary, flourish, meaningless, outweigh, substantiated, refute, jeopardized, incapable, irrational, heterosexual

Table 1: Fifty most indicative words in the Claim Lexicon (starting from the most indicative)

Example Id	Sentence
S1	*He believed* **that nuclear power** *would become* **obsolete**, *to be replaced by clean energy sources.*
S2	*The author concludes* **that wind energy** *has the greatest potential for near-term expansion.*
S3	*As Buckley writes, "If* **atheism** *was* **unacceptable**, *superstition and fanaticism were even more so".*
S4	*Any form of* **corporal punishment** *is barbaric and has no place in a civilized polity.*

Table 2: CS examples for the topics 'We should further exploit nuclear power', 'We should further exploit wind power', 'Atheism is the only way' and 'We should prohibit corporal punishment'. The query items 'that', MC and CL are highlighted in boldface.

The main contribution of the current work is to propose a more flexible approach for corpus wide claim detection, that significantly outperforms previous work, in terms of both precision and of coverage. We also release two data sets, one of $\approx 1.5M$ sentences matching the topics in this study, and one of $2,500$ sentences predicted by our method, annotated for whether they contain a relevant claim or not [2].

We use Deep Neural Networks (DNN) trained with weak supervision that stems from different parts of the aforementioned query. Considering the list of 100 MC used by Levy et al. (2017), we first construct two weakly supervised labeled data sets, each composed of two classes. In the first, the weakly–positive class includes all sentences that mention the MC preceded by 'that'; while the weakly–negative class contains a similar number of sentences that mention the MC *without* a preceding 'that'. Our underlying assumption is that the former set will be more enriched with CS (this we first noted in (Levy et al., 2014)). However, since these two classes are trivially distinguished via the (non) presence of 'that', we train the DNN on the *suffixes* of the sentences in these data, where the suffix of a sentence is defined as the sentence part immediately following the MC.

Similarly, we construct another data set, in which the weakly–positive class includes all sentences that mention the MC followed by a token from CL; while the weakly–negative class contains a similar number of sentences that mention the MC *without* a following token from CL. Here as well, to avoid the trivial signal, we train the DNN on the *prefixes* of the sentences in these data, where the prefix of each sentence is defined as the part preceding the MC.

[2]The data sets can be downloaded from `http://www.research.ibm.com/haifa/dept/vst/debating_data.shtml`

The priors for the two positive classes as well as the strict query were estimated in (Levy et al., 2017) by performing a small labeling experiment. We present their results in table 3 in order to demonstrate that the assumptions indeed hold.

Query Name	Query	Estimated Prior
q_{MC}	MC	2.4%
q_{that}	*that* $\rightarrow$ MC	4.8%
q_{CL}	MC $\rightarrow$ CL	Not Estimated
q_{strict}	that $\rightarrow$ MC $\rightarrow$ CL	9.8%

Table 3: Estimated priors for different queries from (Levy et al., 2017).

Finally, we restrict both datasets to sentences in which the number of suffix (prefix) words is greater than 3. We assume this restriction mostly removes negative examples, and in any case will not convey a lot of information to the DNN in the learning process.

We test the performance of these DNNs over a distinct test set of 50 topics, also from (Levy et al., 2017). However, in contrast to this previous work, we consider a much more relaxed query that only requires the MC to be mentioned in the sentence. Our results clearly indicate that both DNNs were able to generalize and obtain promising precision results, that are further improved when their scores are averaged. That is, combining the predictions of a DNN trained over prefixes of sentences enriched with claims, with those by a DNN trained over suffixes of such sentences, results in a pincer–movement like approach, that successfully pinpoints a wide range of CS in a massive unstructured corpus, while using only weak supervision for training.

2 Related Work

Recently, Wachsmuth et al. (2017) suggested an argument search framework and a corresponding search engine prototype. However, the proposed system relies on arguments crawled from dedicated resources that suggest pre–written arguments for various topics, and hence, is only relevant for topics covered in these resources, and cannot be used directly over unstructured textual data. Stab et al. (2018) tackled the argument mining task in heterogeneous texts retrieved by Google search when queried with a controversial topic. They show that it is feasible to annotate the retrieved documents via crowd-sourcing and to use these labels in order to build a supervised learning system that finds arguments in the given documents. Similar to our work, sentences are treated in isolation (ignoring the document context). The only work we are aware of that tackles corpus wide claim detection, is the work by (Levy et al., 2017). Here, we demonstrate how this work can be leveraged to define weak signals for training DNNs to obtain significantly greater performance.

Several works used DNN to tackle a variety of computational argumentation tasks, such as argument mining (Eger et al., 2017), predicting argument convincingness (Habernal and Gurevych, 2016), detecting context dependent claims and evidence (Laha and Raykar, 2016) and attack and support relations between arguments (Cocarascu and Toni, 2017). However, these works used the fully–supervised learning paradigm, which is inherently demanding, especially in the context of argument mining where obtaining labeled data is notoriously difficult (Aharoni et al., 2014). In addition, Al-Khatib et al. (2016) used a distant supervision approach trained over debate portals' data, to develop a classifier for argumentative texts stored in these portals. To the best of our knowledge, the present work is the first to demonstrate the value of DNN trained solely with weak supervision (Hearst, 1992) in this challenging field.

For a good exposition on the field of argument mining refer to (Lippi and Torroni, 2016). Some notable works include (Palau and Moens, 2009) who first suggested the argument mining task, (Levy et al., 2014; Rinott et al., 2015) who focused on mining claims/evidence in the context of a user given controversial topic and several works related to specific text genres such as student essays (Stab and Gurevych, 2014), legal documents (Wyner et al., 2010; Moens et al., 2007; Grabmair et al., 2015), user comments on proposed regulations (Park and Cardie, 2014) and newspaper articles (Feng and Hirst, 2011).

3 Method

3.1 Setup and pre-processing

We follow the setup and pre-processing described in (Levy et al., 2017) – see appendix for details. We consider the same train [3] and test sets, consisting of 100 and 50 topics respectively. Next, we prepared a sentence–level index from the Wikipedia May 2017 dump, and used a simple Wikification tool (to be described in a separate publication)[4] to focus our attention on sentences that mention the MC. Filtering out sentences that mention a location/person named entity using Stanford NER (Finkel et al., 2005), after the MC, results in an average of $\approx 10K$ sentences per MC.

3.2 Claim sentence queries and weak labels

The basic query we start with, denoted q_{MC}, only requires that the MC will appear in the sentence. For the 150 topics of this study, we retrieve a total of $\approx 1.5M$ sentences matching q_{MC} (Table 4), which we release as a data set to enhance future research. Next, we consider the query q_{that}, which retrieves all sentences in which the token 'that' precedes the MC (cf. $S1$ and $S2$ in table 2). There are $\approx 1,100$ such sentences per topic (Table 4). Aiming to increase the prior of CS in the weak–positive set, for training the network, we focus on the subset of these sentences in which the token 'that' immediately precedes the MC. As a weak–negative set we consider a similar number of sentences, with similar length distribution, selected at random from the q_{MC} sentences with the additional requirement of *not* having 'that' before the MC. As explained in section 1, the corresponding DNN, termed DNN_{suff}, is trained only on the sentence suffixes.

Query Name	Query	# Sentences Per Topic
q_{MC}	MC	9,947
q_{that}	*that* → MC	1,073
q_{CL}	MC → CL	793
q_{strict}	that → MC → CL	164

Table 4: Queries used to construct weak labels. # Sentences Per Topic is averaged over the 150 topics used in this study. q_{strict} is added for reference and was not used in training the networks.

Similarly, we consider the query q_{CL}, which retrieves all sentences in which the MC is followed by a token from CL, e.g., sentences $S1$ and $S3$ in table 2. Again, these sentences as well are expected to be relatively enriched with claims. In Table 4 we see that on average we have ≈ 790 such sentences per topic. As a weak–negative set we consider a similar number of sentences, with similar length distribution, selected at random from the q_{MC} sentences with the additional requirement of *not* having a CL token after the MC. Again, the corresponding DNN, denoted by DNN_{pref} is trained only on the sentence prefixes.

Table 5 lists examples of sentences in the weak–positive and weak–negative sets used to train the networks. The part "seen" by the relevant network appears in bold, where by an anecdotal examination it is indeed possible to identify a signal in the positive sets. Table 6 summarizes the characteristics of the two datasets used to train the networks.

3.3 DNN System

For both DNN_{suff} and DNN_{pref}, we use a Bi-LSTM architecture with self-attention (Yang et al., 2016). The networks were trained on sentences retrieved for 70 of the 100 train–set topics, where sentences retrieved from the other 30 train–set topics (heldout set) were used to optimize hyper-parameters. We used Adam optimizer (Kingma and Ba, 2014) over the cross-entropy loss. The best model was

[3] The set of 100 topics was termed *dev set* in their work because there was no training involved.

[4] A Wikification tool allows retrieving sentences that mention the topic explicitly, as well as sentences which use a different surface form, as in S2 in Table 2 (wind energy surface-form linked to the wind power concept).

Network	Positive/Negative	Sentence
DNN_{suff}	Positive	*There is no good evidence that organic food* **tastes better than its non-organic counterparts**.
DNN_{suff}	Negative	*Today it is known for its remoteness, its somewhat "alternative" atmosphere, organic food* **production, and its pioneering use of wind power**.
DNN_{pref}	Positive	**Fermi did not believe that** *atomic bombs would deter nations from starting wars, nor did he think that the time was ripe for world government.*
DNN_{pref}	Negative	**In particular, fission products do not themselves undergo fission, and therefore cannot be used for** *nuclear weapons.*

Table 5: Examples from the positive and negative sets of DNN_{suff} and DNN_{pref} for the topics "Organic Food" and "Nuclear weapon". The respective prefix/suffix appears in bold.

Network	Positive sentences	Negative sentences	Part of sentence used by the network	Size of data
DNN_{suff}	*that* $\rightarrow$ MC	MC without preceding 'that'	following the MC	11,624
DNN_{pref}	MC $\rightarrow$ CL	MC without a following CL token	preceding the MC	132,856

Table 6: Characteristics of the two datasets used to train the networks. Note that the data for the suffix network is much smaller because of the restriction to sentences in which the token 'that' immediately precedes the MC.

trained with a dropout of 0.15, using a single dropout mask across all time-steps as proposed by (Gal and Ghahramani, 2016), one LSTM layer with a cell size of 128, and an attention layer of size 100. Words are represented using the 300 dimensional GloVe embeddings (Pennington et al., 2014). Inference is performed for any q_{MC} sentence by averaging the DNN_{suff} score of its suffix with the DNN_{pref} score of its prefix.

We used the heldout set to determine early stopping and to optimize the following hyper–parameters (each parameter was optimized independently): Number of layers $(1/2)$, LSTM cell size $(64/128/256/512)$, attention FF size $(50/100/200)$ and dropout rate $(0/0.05/0.1/0.15/0.2/0.25/0.3/0.35)$.

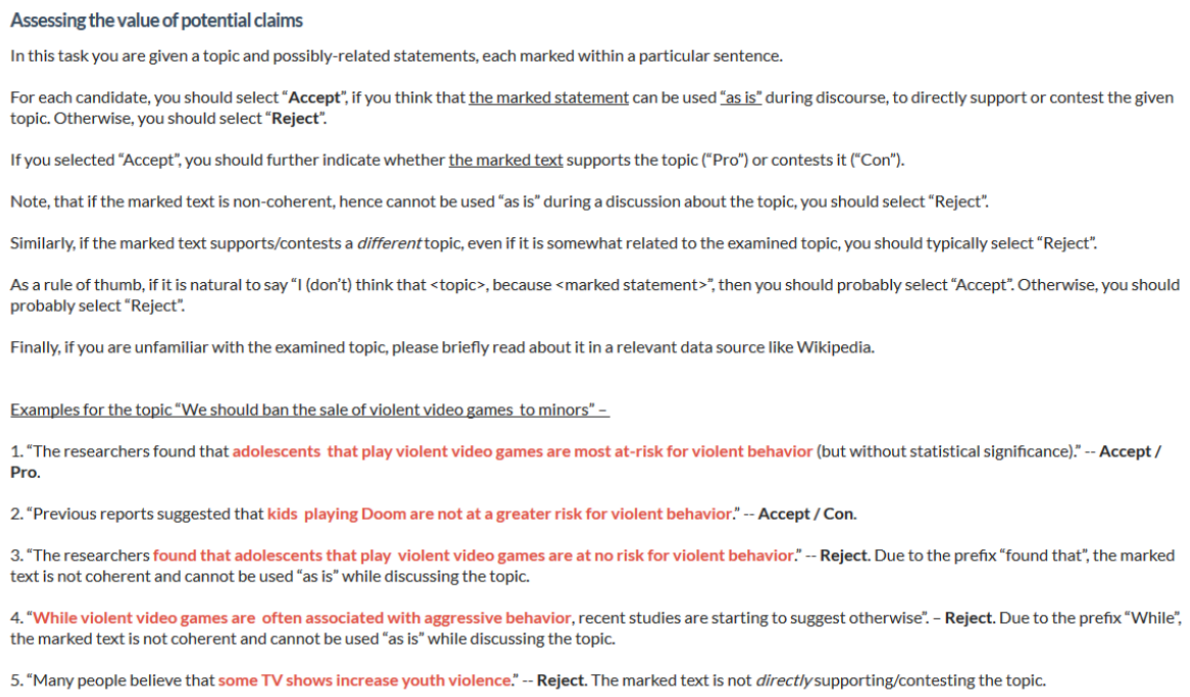
4 Data for evaluation

We labeled via crowd the top 50 predicted sentences for each of the 50 test-set topics, taking the majority vote of at least 10 workers. The guidelines are presented in figure 1. The inference is applied to all sentences containing the MC (matching q_{MC}), and hence there are always 50 predictions, that are all released along with their manual evaluation. We also label in the same manner the predictions of the system described in (Levy et al., 2017) [5]. There, since all predictions must match q_{strict}, for some topics there are less than 50 predictions. In those cases, we label all predictions.

This paper focuses on retrieving claim sentences, however, we have found that it is easier for the crowd workers to label a sentence if the phrase suggested to be the claim is highlighted. For this reason, we used an internal boundary detection component and applied it to all system versions (including the re-implementation baseline of (Levy et al., 2017)). The rest of the labeling process was done similarly to (Levy et al., 2017). Each sentence was labeled by 10-15 crowd workers per row via the Figure–Eight platform [6]. We used the MACE de–noising tool (Hovy et al., 2013) to filter labels before computing Cohen's Kappa coefficient. We averaged the Kappa coefficient across all worker pairs with at least 50 joint labeled instances. Using a threshold of 0.9 (i.e. keeping 90% of the labels) the Kappa was 0.58.

[5] We re-implemented their system since since we used a more recent Wikipedia dump and a different Wikification tool. The results we obtained are very close to the reported results.

[6] https://www.figure-eight.com/ (previously known as CrowdFlower)



Assessing the value of potential claims

In this task you are given a topic and possibly-related statements, each marked within a particular sentence.

For each candidate, you should select "**Accept**", if you think that the marked statement can be used "as is" during discourse, to directly support or contest the given topic. Otherwise, you should select "**Reject**".

If you selected "Accept", you should further indicate whether the marked text supports the topic ("Pro") or contests it ("Con").

Note, that if the marked text is non-coherent, hence cannot be used "as is" during a discussion about the topic, you should select "Reject".

Similarly, if the marked text supports/contests a *different* topic, even if it is somewhat related to the examined topic, you should typically select "Reject".

As a rule of thumb, if it is natural to say "I (don't) think that <topic>, because <marked statement>", then you should probably select "Accept". Otherwise, you should probably select "Reject".

Finally, if you are unfamiliar with the examined topic, please briefly read about it in a relevant data source like Wikipedia.

Examples for the topic "We should ban the sale of violent video games to minors" –

1. "The researchers found that adolescents that play violent video games are most at-risk for violent behavior (but without statistical significance)." -- **Accept / Pro.**

2. "Previous reports suggested that kids playing Doom are not at a greater risk for violent behavior." -- **Accept / Con.**

3. "The researchers found that adolescents that play violent video games are at no risk for violent behavior." -- **Reject.** Due to the prefix "found that", the marked text is not coherent and cannot be used "as is" while discussing the topic.

4. "While violent video games are often associated with aggressive behavior, recent studies are starting to suggest otherwise". – **Reject.** Due to the prefix "While", the marked text is not coherent and cannot be used "as is" while discussing the topic.

5. "Many people believe that some TV shows increase youth violence." -- **Reject.** The marked text is not *directly* supporting/contesting the topic.

Figure 1: Labeling guidelines exactly as they appeared in the Figure–Eight platform. Note that for the sake of this work we ignore the stance labeling (pro/con answers).

5 Results

To evaluate the performance of the network, we employ two sets of experiments. In the first we use the test-set topics in a manner similar to (Levy et al., 2017). In the second, we test our network on the UKP Sentential Argument Mining Corpus released in (Stab et al., 2018). Note that the UKP data is more inline with our goal than other argument mining tasks as it separates between sentences that support/contest a given topic from sentences that don't. A major difference between the UKP data and our test set is the source from which the sentences were taken – while we used Wikipedia, the UKP data comes from various sources, and hence it would test how well our approach generalizes to other text genres. Another important difference is in the definition of positive examples - we consider sentences containing relevant claims as positive, whereas they require that a sentence contain some supporting evidence or reasoning. The results on our test set are presented in subsection 5.1 and the results on the UKP data are presented in subsection 5.2.

5.1 Results on the Test Set

Figure 2 depicts the average number of CS (i.e., true positives) retrieved per the top $K = 10, 20, 50$ predictions. Both DNN_{pref} and DNN_{suff} seem to generalize well from the weak signal and provide comparable results to (Levy et al., 2017). More importantly, using the average score (DNN) yields the best performance, consistently outperforming (Levy et al., 2017) (with p-value < 0.005 for $K = 20, 50$ based on a two-tailed Wilcoxon test). The gap is most prominent for $K = 50$, where the DNN yields $\approx 30\%$ more CS compared to the the non–learning system that used a strict query with limited recall.

A major question is whether the learned system is able to generalize from the weak labels and identify CS that do not match the weak queries we started with. By construction, all sentences retrieved by the (Levy et al., 2017) system, match q_{strict}. From Table 7, we see that although the DNN system trained on sentences matching q_{that} or q_{CL} or both, 28% of the 2500 sentences predicted by the system, *do not match either*. Sentence $S4$ in Table 2 is an example of such a predicted sentence. The precision on those sentences, that are only known to contain the MC, is still considerably high – 0.22, and in fact comparable

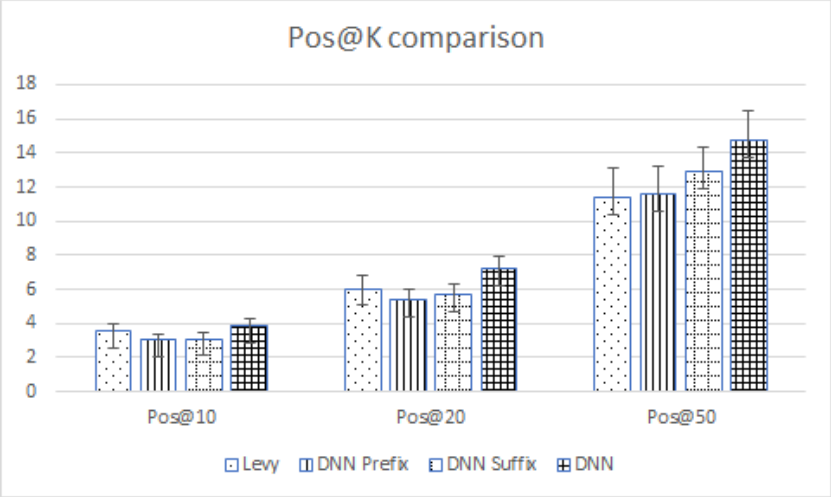


Figure 2: Results for the 50 test-set topics. $Pos@K$: The number of claim sentences (CS) out of the top K predicted sentences per topic, averaged over all test topics. Bars denote the standard error. Levy – Results reproduced on current index using the system described in (Levy et al., 2017); DNN – using the average score of DNN_{pref} and DNN_{suff}.

to the precision achieved in the restricted, low–recall system of (Levy et al., 2017). These results suggest that the DNN captures some general characteristics of CS, that are not limited to sentences that satisfy the two weak–signal queries we started with, q_{that} and q_{CL}. In addition, the precision on sentences containing one or both of the weak signals is even higher. Specifically, the precision on the subset of sentences matching q_{strict} is 0.42, a factor of two compared to the precision of (Levy et al., 2017) on this set of sentences. Thus, overall we were able to increase the potential recall from the restricted set of sentences matching q_{strict} to the full set of q_{MC} sentences, while also increasing the precision of the predictions from 0.23 to 0.3 (see the column q_{MC}).

Measure	System	q_{strict}	q^*_{that}	q^*_{CL}	q^*_{MC}	q_{MC}
Percent	Levy	100	0	0	0	100
	DNN	30	26	16	28	100
Precision	Levy	0.23	NA	NA	NA	0.23
	DNN	0.42	0.32	0.22	0.22	0.30

Table 7: Distribution and precision of predictions. q^*_{that}: matches q_{that} but not q_{CL}; q^*_{CL}: matches q_{CL} but not q_{that}; q^*_{MC}: matches q_{MC}, but not any of the other queries. Percent: out of the top 50 (or all if less available) predicted sentences matching the query. Precision: Precision of the corresponding candidates, calculated per topic, and averaged over test topics.

5.2 Results on the UKP Sentential Argument Mining Corpus

The UKP Sentential Argument Mining Corpus (Stab et al., 2018) contains a total of 25,492 labeled sentences (11,139 argumentative, 14,353 non-argumentative), divided to train (70%), validation (10%), and test sets (20%). The sentences are associated with one of 8 controversial topics – *abortion*, *cloning*, *death penalty*, *gun control*, *marijuana legalization*, *minimum wage*, *nuclear energy* and *school uniforms* – and were derived from the top 50 results of a Google query for the topic name, thus representing various genres and text types.

Adapting to the UKP dataset

In order to evaluate our method on the UKP dataset we had to adapt it to sentences that do not necessarily contain the MC. In our formulation, the MC was used as a natural point to divide the sentence into its prefix and suffix, which were then used by the appropriate networks. To overcome this difference, we applied DNN_{suff} (DNN_{pref}) to all possible suffixes (prefixes) and used the maximal score.

For a sentence S comprised of n words, $w_1, w_2, ..., w_n$, we define,

$$DNN'_{suff}(S) = max_{\{i:1..n\}} DNN_{suff}(w_i, ..., w_n)$$
$$DNN'_{pref}(S) = max_{\{i:0..n-1\}} DNN_{pref}(w_1, ..., w_{n-i})$$

The adapted scores may still be at a disadvantage because without the MC we don't have a way to select sentences that are more related to the topic. For this reason we add a similarity score $Score_{w2v}$ which is computed by taking the maximal word2vec (Mikolov et al., 2013) similarity of a word in the topic against all words in the sentence and then averaging across the words of the topic. More formally, for a topic T comprised of k words, $t_1, ..., t_k$,

$$Score_{w2v}(S) = avg_{\{i:1..k\}} max_{\{j:1..n\}} word2vec(t_i, w_j)$$

Finally, we define,

$$DNN'_{suff,w2v} = avg(DNN'_{suff}, Score_{w2v})$$
$$DNN'_{pref,w2v} = avg(DNN'_{pref}, Score_{w2v})$$

We employ the same setup that was used by (Stab et al., 2018) for the cross topic evaluation, in which the train set is comprised of the train part of all topics except for the tested topic. We use the training set only to tune the threshold from which we predict the positive class. To do so, we run the different DNN methods on the train set, compute the F_1 over all sentences, and choose the score that maximizes the F_1. This score is then used in the test set as the threshold that determines whether the network predicts a positive or not. Overall, this tuning was done 8 times, one for each train set induced by the left-out test topic.

Evaluation

The results are shown in table 8. Interestingly, our system achieves comparable results to the state of the art in the Accuracy and F_1 measures but without using human labels for training and without training on multiple text genres. These results also demonstrate the ability of the proposed method to generalize to topics that are not characterized by a single MC or that such a concept was not provided by the user. Note, the results reflect that our system and the baseline operate at different points on the precision/recall curve, choosing a different compromise between precision and recall. This is not surprising, given the choice of tuning the F_1 measure on the train set, however, it makes the comparison less obvious.

Method	Accuracy	F_1	Precision	Recall
UKP	0.69	0.66	0.75	0.52
DNN'_{suff}	0.57	0.65	0.51	0.90
$DNN'_{suff,w2v}$	0.67	0.69	0.59	0.83

Table 8: Results of the cross topic evaluation on the UKP dataset (averaged across the 8 topics). UKP method stands for the best supervised results reported in (Stab et al., 2018). From the networks combined with $w2v$ the $DNN'_{suff,w2v}$ performed best and is the one presented here.

It should be noted that the lower precision of our method may be explained by the different assumption on what an argumentative sentence is. Whereas Stab et al. (2018) reject sentences that contain claims but provide no evidence or reasoning, our network was designed to identify claims regardless of the

existence of a surrounding argument. Indeed, as mentioned in section 6.2, by sampling 50 false-positives we found that in 25% of the cases they contained relevant claims but with no evidence or reasoning.

6 Error Analysis

6.1 Test Set - 50 Topics

We analyzed the top 50 labeled predictions over three test topics for which the performance was above/near/below average (table 9).

Topic Text	Main Concept	Pos@50
We should further exploit wind power	Wind power	29
Private education brings more good than harm	Private school	13
We should protect whistleblowers	Whistleblower	9

Table 9: Test topics chosen for error analysis.

Each sentence rejected by the labelers was assigned one of the following types: **Factual** – a sentence with no argumentative content, that merely states a fact; **Different Topic** – a sentence that contains a claim for a different topic; **Other** – an assortment of problems such as bad sentence split, missing context, etc; and finally **Accept** – a sentence that should have been accepted by the labelers. The two main types of errors were Factual and Different topic, each accounting for 35% of the analyzed errors. The Accept type accounted for 18% of the rejected sentences, though this high number was mostly due to the Whistleblowers topic. We suspect that many such sentences were rejected because of bad claim boundary choices by the system [7]. Table 10 shows examples from the topic *"Private education brings more good than harm"*.

Error Type	Sentence
Different Topic	*Changes in private school enrollment is not a likely contributor to any changes in schools segregation patterns during that time.*
	In 2014 Hunt proposed that private schools should be required to form "partnerships" with local state schools if they wanted to keep their charitable status.
Factual	*Before enrolling the children, however, Mr. Brar ensured that the total cost of private school tuition would not exceed* $\$10,000$.
	The IRS announced in 1970 that private schools with racially discriminatory admissions policies would no longer receive tax exemptions
Accept	*Coaches were concerned that the private schools were winning a disproportionate amount of conference titles and had several unfair advantages.*
Other*	*It is clear that affording private education is a mere fantasy for these families.*

Table 10: Examples of sentences from the topic 'Private education brings more good than harm'. The sentences are split according to their assigned error type. * The example for the Other type was rejected because of a missing context – it is hard to judge this example without resolving the reference to "these families"

6.2 Test Set - UKP Dataset

We analyzed 50 random sentences from the UKP test set labeled as non-argumentative, on which the score of the $DNN'_{suff,w2v}$ network was higher than 0.9 (the average threshold obtained by tuning F_1 was 0.65). We add the following error type to the list above: **No Reasoning** – a sentence containing a claim with no supporting evidence or reasoning. The most frequent type of error was Factual, accounting

[7] We used a claim boundary component (Levy et al., 2014) on top of all systems in order to simplify the labeling task. This came at a cost of some CS being rejected due to errors in the boundary component.

for about 33% of the errors. The No Reasoning type accounted for about 25% of the errors, similar to the Different Topic type. Table 11 shows examples of No Reasoning sentences. These sentences contain text boundaries that are relevant claims, e.g., the boundary *the life in the womb is not human* in the first sentence, and thus are typical to sentences that our network was trained to find.

Topic	Sentence
abortion	*A question for those who believe in abortion, and that* **the life in the womb is not human**.
death penalty	**We need stricter laws and swift death penalty**.
minimum wage	*Myth:* **Raising the minimum wage will only benefit teens**.
marijuana legalization	*A small share of opponents (7%) say that while* **the recreational use of marijuana should be illegal**, *they do not object to legalizing medical marijuana.*

Table 11: Examples of sentences marked as No Reasoning from the UKP test set. The phrases marked in boldface are the suggested claim boundaries according to our analysis.

7 Discussion and Future Work

This work aims at making the first steps towards a search engine for argumentative content, by focusing on the problem of corpus wide claim detection. A variety of argument theories have been proposed throughout the years, which all agree on the importance of one argument component – the claim. Thus, properly addressing the problem of corpus wide claim detection seems like a key component in developing a full fledged argument search engine. Such an engine could add massive amounts of data to argument networks such as the world wide argument web (Rahwan et al., 2007), and further enhance decision processes in various ways. Using a similar methodology for evidence detection would be a natural way to push the boundary of existing work, e.g., (Rinott et al., 2015) from considering a pre-selected list of articles to searching full corpora. To the best of our knowledge this is the first work using weak supervision to train DNNs for argument mining, demonstrating the potential of this coupling in the field. Two directions for future work could increase the precision and coverage of our system. For increasing precision, we intend to employ a supervised approach, using labels on top of predictions from the weak–supervision approach, as it may help reach a reasonable prior of positive examples before starting the labeling effort. For the coverage, we intend to explore the same approach on top of sentences which do not necessarily contain the MC. This direction is challenging since it requires integrating a method for identifying whether a sentence is related to the topic, and would need to score sentences in which the prior for a claim is even lower.

During the error analysis on the 'Wind power' topic, we encountered the following high-scoring sentence – *"When Scratchy suggests that wind power is cheap and safe, Itchy chops Scratchy's head off with the blades of a wind turbine."*. On the one hand, Scratchy raises a legitimate claim, and on the other hand, Scratchy is a fictional character from the TV show The Simpsons. The example demonstrates a phenomenon that may be exasperated when moving from argument mining on pre-selected high-quality documents to mining large (possibly heterogeneous) text corpora – the phenomenon of claims made by unreliable sources. In extreme cases the claims made by such parties may be ridiculous or offensive and a practical search engine would need to detect and remove such claims.

Appendix A Index and Preprocessing

We processed the Wikipedia dump from May 1st, 2017. We applied text cleaning and sentence splitting using OpenNlp Sentence Detector [8] and an internal Wikification tool to wikify each sentence [9]. Starting from 5.4M articles, the sentence level index contains approximately 102M sentences. The inverted index along with the support for queries that mix surface form tokens with Wiki concepts was implemented as in (Levy et al., 2017).

We annotated the sentences retrieved by all queries using Stanford NER (Finkel et al., 2005), and removed sentences with a person/location entity after the MC (e.g., the sentence *"Yan warned Li that the Nationalist cause was doomed unless Li went to Guangdong"* for the topic *"Nationalism does more harm than good"* would be removed). This filter is motivated by our goal of retrieving general claim sentences for the topic, assuming that claims about specific entities are less interesting for potential users.

Appendix B Topics and Folds

The list of 150 topics is taken from (Levy et al., 2017) and split to dev/test in the same manner. Since here we use a learning system, we further split the dev set into a train set of 70 topics and a heldout set of 30 topics which was used to decide when to stop the learning. Tables 12 and 13 show the train topics and tables 14 and 15 show the topics of the heldout and test sets respectively.

Appendix C Released Data

We release two datasets, one containing $\approx 1.5M$ sentences matching the topics in this study based on the q_{MC} query, and one containing $2,500$ sentences predicted by our network and annotated for whether they contain a relevant claim or not (top 50 predictions across the 50 topics in the test set)[10]. The q_{MC} dataset can be found in the attached `q_mc_train.csv`, `q_mc_heldout.csv` and `q_mc_test.csv` files, according to the topics split used in the learning/evaluation process. A detailed description of this dataset appears in the `readme_mc_queries.txt` file. The system prediction dataset is in the `test_set.csv` file with a corresponding description in the `readme_test_set.txt` file.

[8] http://opennlp.apache.org/

[9] To be described in a separate publication.

[10] The datasets can be downloaded from `http://www.research.ibm.com/haifa/dept/vst/debating_data.shtml`

#	Id	Topic Text	Main Concept
1	1	We should ban the sale of violent video games to minors	Video game controversies
2	2	We should legalize doping in sport	Doping in sport
3	3	We should ban boxing	Boxing
4	4	We should abolish intellectual property rights	Intellectual property
5	5	We should protect endangered species	Endangered species
6	6	Operation Cast Lead was justified	Gaza War (2008-09)
7	7	Tower blocks are advantageous	Tower block
8	8	Private universities bring more good than harm	Private university
9	9	We should disband ASEAN	Association of Southeast Asian Nations
10	10	The free market brings more good than harm	Free market
11	11	We should ban child actors	Child actor
12	12	Religion does more harm than good	Religion
13	13	We should ban cosmetic surgery	Plastic surgery
14	14	Same sex marriage brings more good than harm	Same-sex marriage
15	15	Reality television does more harm than good	Reality television
16	16	Internet censorship brings more good than harm	Internet censorship
17	17	Socialism brings more harm than good	Socialism
18	18	We should ban beauty contests	Beauty pageant
19	19	We should adopt vegetarianism	Vegetarianism
20	20	We should adopt libertarianism	Libertarianism
21	21	The internet brings more harm than good	Internet
22	22	Science is a major threat	Science
23	23	Suicide should be a criminal offence	Suicide
24	24	Nationalism does more harm than good	Nationalism
25	25	The atomic bombings of Hiroshima and Nagasaki were justified	Atomic bombings of Hiroshima and Nagasaki
26	26	Casinos bring more harm than good	Casino
27	27	We should lower the age of consent	Age of consent
28	28	We should abolish standardized tests	Standardized test
29	29	We should ban extreme sports	Extreme sport
30	30	The alternative vote is advantageous	Instant-runoff voting
31	31	Illegal immigration brings more harm than good	Illegal immigration
32	32	We should subsidize renewable energy	Renewable energy
33	33	We should end daylight saving times	Daylight saving time
34	34	We should further exploit geothermal energy	Geothermal energy
35	35	Assisted suicide should be legalized	Assisted suicide
36	36	Security hackers do more harm than good	Hacker (computer security)
37	37	We should disband the United Nations	United Nations
38	38	We should ban hate sites	Hate speech
39	39	We should privatize future energy production	Energy development
40	40	Child labor should be legalized	Child labour
41	41	The paralympic games bring more good than harm	Paralympic Games
42	42	Chain stores bring more harm than good	Chain store
43	43	We should subsidize Habitat for Humanity International	Habitat for Humanity
44	44	We should subsidize public art	Public art
45	45	IKEA brings more harm than good	IKEA
46	46	We should ban online advertising	Online advertising
47	47	Mixed-use development is beneficial	Mixed-use development
48	48	We should ban Greyhound racing	Greyhound racing
49	49	The Israeli disengagement from Gaza brought more harm than good	Israeli disengagement from Gaza
50	50	We should not subsidize single parents	Single parent

Table 12: Train topics 1-50

#	Id	Topic Text	Main Concept
51	51	We should ban private military companies	Private military company
52	52	Coaching brings more harm than good	Coaching
53	53	We should abandon disposable diapers	Diaper
54	54	PayPal brings more good than harm	PayPal
55	55	The Internet archive brings more harm than good	Internet Archive
56	56	The 2003 invasion of Iraq was justified	2003 invasion of Iraq
57	57	Virtual reality brings more harm than good	Virtual reality
58	58	Internet cookies bring more harm than good	HTTP cookie
59	59	Magnet schools bring more harm than good	Magnet school
60	60	The right to strike brings more harm than good	Strike action
61	61	We should subsidize student loans	Student loan
62	62	We should abandon Youtube	YouTube
63	63	Ecotourism brings more harm than good	Ecotourism
64	64	Academic freedom is not absolute	Academic freedom
65	65	Homeschooling should be banned	Homeschooling
66	66	We should abolish the US Electoral College	Electoral College (United States)
67	67	Generic drugs should be banned	Generic drug
68	68	We should fight global warming	Global warming
69	69	We should fight for Quebecan Independence	Quebec sovereignty movement
70	70	We should subsidize newspapers	Newspaper

Table 13: Train topics 51-70

#	Id	Topic Text	Main Concept
1	71	The freedom of speech is not absolute	Freedom of speech
2	72	We should criminalize blasphemy	Blasphemy
3	73	Holocaust denial should be a criminal offence	Holocaust denial
4	74	Television does more harm than good	Television
5	75	We should subsidize higher education	Higher education
6	76	We should ban organic food	Organic food
7	77	Urbanization does more harm than good	Urbanization
8	78	We should adopt direct democracy	Direct democracy
9	79	We should ban lotteries	Lottery
10	80	We should close the Guantanamo Bay detention camp	Guantanamo Bay detention camp
11	81	We should abandon the insanity plea	Insanity defense
12	82	We should protect coral reefs	Coral reef
13	83	We should disband NASA	NASA
14	84	We should abolish nuclear weapons	Nuclear weapon
15	85	We should cancel the speed limit	Speed limit
16	86	Randomized controlled trials bring more harm than good	Randomized controlled trial
17	87	Anarchism brings more good than harm	Anarchism
18	88	We should subsidize public service broadcasters	Public broadcasting
19	89	We should ban labor organizations	Trade union
20	90	Pride parades bring more harm than good	Pride parade
21	91	Paternity leave brings more harm than good	Parental leave
22	92	Tabloid journalism brings more harm than good	Tabloid journalism
23	93	We should disband UNESCO	UNESCO
24	94	We should disband the National Rifle Association	National Rifle Association
25	95	Second Life brought more harm than good	Second Life
26	96	Economic sanctions bring more harm than good	Economic sanctions
27	97	Vietnam War was justified	Vietnam War
28	98	Animal slaughter is not justified	Animal slaughter
29	99	We should raise the corporate tax	Corporate tax
30	100	Division of labor is a major threat	Division of labour

Table 14: Heldout topics

#	Id	Topic Text	Main Concept
1	101	Affirmative action brings more good than harm	Affirmative action
2	102	We should ban gambling	Gambling
3	103	We should abolish the monarchy	Monarchy
4	104	Atheism is the only way	Atheism
5	105	We should further exploit wind power	Wind power
6	106	We should legalize polygamy	Polygamy
7	107	We should further exploit hydroelectric dams	Hydroelectricity
8	108	We should privatize water supply	Water supply
9	109	We should legalize prostitution	Prostitution
10	110	Zoos bring more harm than good	Zoo
11	111	Private education brings more good than harm	Private school
12	112	Recall elections are beneficial	Recall election
13	113	We should further exploit nuclear power	Nuclear power
14	114	We should abolish temporary employment	Temporary work
15	115	Surrogacy should be banned	Surrogacy
16	116	Progressive tax is beneficial	Progressive tax
17	117	We should ban alcoholic beverages	Alcoholic drink
18	118	We should ban abortions	Abortion
19	119	Astrology brings more harm than good	Astrology
20	120	Embryonic stem cell research brings more good than harm	Embryonic stem cell
21	121	We should abolish the Olympic Games	Olympic Games
22	122	We should end athletic scholarships	Athletic scholarship
23	123	Social media does more harm than good	Social media
24	124	We should disband the United Nations Security Council	United Nations Security Council
25	125	We should legalize insider trading	Insider trading
26	126	We should prohibit hydraulic fracturing	Hydraulic fracturing
27	127	We should prohibit corporal punishment	Corporal punishment
28	128	We should disband NATO	NATO
29	129	We should abolish the two-party system	Two-party system
30	130	Capital punishment brings more harm than good	Capital punishment
31	131	We should abolish term limits	Term limit
32	132	We should protect whistleblowers	Whistleblower
33	133	Twitter brings more harm than good	Twitter
34	134	ISO brings more harm than good	International Organization for Standardization
35	135	Conscientious objectors are justified	Conscientious objector
36	136	The American Bar Association brings more harm than good	American Bar Association
37	137	Digital rights management brings more harm than good	Digital rights management
38	138	We should ban the Church of Scientology	Church of Scientology
39	139	eBay brings more good than harm	EBay
40	140	We should abolish the caste system in India	Caste system in India
41	141	We should abolish infant baptism	Infant baptism
42	142	EHRs bring more harm than good	Electronic health record
43	143	Wildlife management brings more good than harm	Wildlife management
44	144	We should tax plastic bags	Plastic bag
45	145	The energy industry should be nationalized	Energy industry
46	146	We should fight protectionism	Protectionism
47	147	We should limit genetic testing	Genetic testing
48	148	We should end manned spaceflights	Human spaceflight
49	149	Extra-curricular activity should be mandatory	Extracurricular activity
50	150	We should abolish homework	Homework

Table 15: Test topics

References

Ehud Aharoni, Anatoly Polnarov, Tamar Lavee, Daniel Hershcovich, Ran Levy, Ruty Rinott, Dan Gutfreund, and Noam Slonim. 2014. A benchmark dataset for automatic detection of claims and evidence in the context of controversial topics. In *Proceedings of the First Workshop on Argumentation Mining*, pages 64–68.

Khalid Al-Khatib, Henning Wachsmuth, Matthias Hagen, Jonas Köhler, and Benno Stein. 2016. Cross-domain mining of argumentative text through distant supervision. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1395–1404, San Diego, California, June. Association for Computational Linguistics.

Oana Cocarascu and Francesca Toni. 2017. Identifying attack and support argumentative relations using deep learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1374–1379, Copenhagen, Denmark, September. Association for Computational Linguistics.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural end-to-end learning for computational argumentation mining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11–22, Vancouver, Canada, July. Association for Computational Linguistics.

Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 987–996. Association for Computational Linguistics.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Matthias Grabmair, Kevin D Ashley, Ran Chen, Preethi Sureshkumar, Chen Wang, Eric Nyberg, and Vern R Walker. 2015. Introducing luima: an experiment in legal conceptual retrieval of vaccine injury decisions using a uima type system and tools. In *Proceedings of the 15th International Conference on Artificial Intelligence and Law*, pages 69–78. ACM.

Ivan Habernal and Iryna Gurevych. 2016. Which argument is more convincing? analyzing and predicting convincingness of web arguments using bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1589–1599, Berlin, Germany, August. Association for Computational Linguistics.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with mace. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Anirban Laha and Vikas Raykar. 2016. An empirical evaluation of various deep learning architectures for bi-sequence classification tasks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2762–2773, Osaka, Japan, December. The COLING 2016 Organizing Committee.

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

Ran Levy, Shai Gretz, Benjamin Sznajder, Shay Hummel, Ranit Aharonov, and Noam Slonim. 2017. Unsupervised corpus–wide claim detection. In *Proceedings of the 4th Workshop on Argument Mining*, pages 79–84, Copenhagen, Denmark, September. Association for Computational Linguistics.

Marco Lippi and Paolo Torroni. 2016. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology (TOIT)*, 16(2):10.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Marie-Francine Moens, Erik Boiy, Raquel Mochales Palau, and Chris Reed. 2007. Automatic detection of arguments in legal texts. In *Artificial intelligence and law*, pages 225–230. ACM.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: the detection, classification and structure of arguments in text. In *Proceedings of the 12th international conference on artificial intelligence and law*, pages 98–107. ACM.

Joonsuk Park and Claire Cardie. 2014. Identifying appropriate support for propositions in online user comments. In *Workshop on Argumentation Mining*, pages 29–38, Baltimore, Maryland, June. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.

Iyad Rahwan, Fouad Zablith, and Chris Reed. 2007. Laying the foundations for a world wide argument web. *Artificial intelligence*, 171(10-15):897–921.

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

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56, Doha, Qatar, October. Association for Computational Linguistics.

Christian Stab, Tristan Miller, and Iryna Gurevych. 2018. Cross-topic argument mining from heterogeneous sources using attention-based neural networks. *arXiv preprint arXiv:1802.05758*.

Stephen E Toulmin. 2003. *The uses of argument*. Cambridge university press.

Henning Wachsmuth, Martin Potthast, Khalid Al Khatib, Yamen Ajjour, Jana Puschmann, Jiani Qu, Jonas Dorsch, Viorel Morari, Janek Bevendorff, and Benno Stein. 2017. Building an argument search engine for the web. In *Proceedings of the 4th Workshop on Argument Mining*, pages 49–59, Copenhagen, Denmark, September. Association for Computational Linguistics.

Adam Wyner, Raquel Mochales-Palau, Marie-Francine Moens, and David Milward. 2010. Semantic processing of legal texts. chapter Approaches to Text Mining Arguments from Legal Cases, pages 60–79. Springer.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *HLT-NAACL*.

Improving Named Entity Recognition by Jointly Learning to Disambiguate Morphological Tags

Onur Güngör
Boğaziçi University, Istanbul, Turkey
onurgu@boun.edu.tr

Suzan Üsküdarlı
Boğaziçi University, Istanbul, Turkey
suzan.uskudarli@boun.edu.tr

Tunga Güngör
Boğaziçi University, Istanbul, Turkey
gungort@boun.edu.tr

Abstract

Previous studies have shown that linguistic features of a word such as possession, genitive or other grammatical cases can be employed in word representations of a named entity recognition (NER) tagger to improve the performance for morphologically rich languages. However, these taggers require external morphological disambiguation (MD) tools to function which are hard to obtain or non-existent for many languages. In this work, we propose a model which alleviates the need for such disambiguators by jointly learning NER and MD taggers in languages for which one can provide a list of candidate morphological analyses. We show that this can be done independent of the morphological annotation schemes, which differ among languages. Our experiments employing three different model architectures that join these two tasks show that joint learning improves NER performance. Furthermore, the morphological disambiguator's performance is shown to be competitive.

Title and Abstract in Turkish

Biçimbilimsel Etiketleri Ayrıştırmayı Birlikte Öğrenerek Varlık İsmi Tanıma Başarısını Artırmak

Daha önceki çalışmalar, biçimbilimsel olarak zengin dillerdeki varlık ismi tanıma (VAT) başarısını artırmak için sözcüklerin iyelik, genitif ve benzeri hâllerinin kullanılabileceğini göstermiştir. Ancak, bu türden varlık ismi tanıma işaretleyicilerinin çalışabilmesi için elde edilmesi zor veya bazı diller için imkansız olan dışsal biçimbilimsel ayrıştırıcılara (BA) ihtiyaç vardır. Bu çalışmada, bu tür ayrıştırıcılara olan ihtiyacı ortadan kaldırmak için VAT ve BA görevlerini aynı anda çözen ve aday biçimbilimsel çözümlemelerin sunulabildiği dillere uygulanabilen bir model önerilmektedir. Bunun dillere göre değişen biçimbilimsel işaretleme şemalarından bağımsız olarak yapılabildiği gösterilmiştir. Bu iki görevi aynı anda gerçekleştiren üç farklı model mimarisi kullanarak yaptığımız deneyler birlikte öğrenmenin VAT başarısını artırdığını göstermiştir. Buna ek olarak, biçimbilimsel ayrıştırıcının başarısının önceki çalışmalarla karşılaştırılabilir olduğu görülmüştür.

1 Introduction

Named entity recognition (NER) is the task of selecting the portions of text which refer to an entity that designate a person, location or organization. This makes it a basic natural language processing (NLP) task closely related to relation extraction, knowledge base population, and entity linking.

Works that represent the current state of the art in NER generally start by representing words with pretrained word embeddings, embeddings which rely on surface form characters (Lample et al., 2016; Ma and Hovy, 2016). These architectures feed the word representations to a bidirectional long short-term memory layer (Bi-LSTM) to represent the context where the disambiguation between the possible entities is undertaken by decoding on trellis provided by a conditional random field (CRF) model.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2082–2092
Santa Fe, New Mexico, USA, August 20-26, 2018.

When these models are trained and evaluated for morphologically rich languages (MRLs), it has been shown that using embeddings based on characters or linguistic properties of the word such as morphological features indicating a grammatical case improves the performance compared to using only pretrained word embeddings (Gungor et al., 2017). Even though they provide a better approach for MRLs, they require an external morphological disambiguator for every language of interest, a requirement which can be hard or even impossible for some languages to satisfy. This is especially true for agglutinative languages where there can be many roots and morphological tag sequences for a single word. Although there is an effort to provide a tool for POS tagging and lemmatization for many languages in a single format (Straka and Straková, 2017), it has been shown that there is a better approach for morphological tagging in terms of performance which can utilize the information in the context of the target word (Shen et al., 2016).

In this paper, we propose a model to jointly learn the NER and morphological disambiguation (MD) tasks to offer a solution to this problem. We design our model so that any language with a mechanism which can provide a number of candidate morphological analyses for a word can utilize our joint model. This is easier compared to providing disambiguated morphological analyses because systems that disambiguate morphological analyses are harder to build. Furthermore, we do not require the labels of each task to be present in the same dataset. One can easily train the part of the model which is responsible for the MD task in another -preferably larger- dataset and start with the pretrained model. Our main contribution is to show that jointly disambiguating morphological tags and predicting the NER tags results in an equivalent level of performance compared to using externally provided morphological tags.

We give a survey of related work on the subject in Section 2. We explain our basic models and the proposed joint models in Section 3. In Section 4, we describe our dataset which is derived from a frequently used database in the literature. After running the experiments described in Section 5, we observe that jointly training our model for NER and MD results in an increase in the NER performance.

2 Related Work

Early approaches to NER typically use several hand-crafted features such as capitalization, word length, gazetteer based features, and syntactic features (part-of-speech tags, chunk tags, etc.) (McCallum and Li, 2003; Finkel et al., 2005; Humphreys et al., 1998; Appelt et al., 1995). Some of them are data-driven approaches such as conditional random fields (CRF) (McCallum and Li, 2003; Finkel et al., 2005), maximum entropy (Borthwick, 1999), bootstrapping (Jiang and Zhai, 2007; Wu et al., 2009), latent semantic association (Guo et al., 2009), and decision trees (Szarvas et al., 2006).

Recently, RNN based sequence taggers have dominated the state of the art in NER (Lample et al., 2016; Ma and Hovy, 2016; Huang et al., 2015; Yang et al., 2016). These approaches model the words as fixed length vectors and employ Bi-LSTM or GRU layers to obtain a characterization of the relevant context of the word to be labeled. These context vectors are then transferred to a CRF module after transforming into score vectors. However, in these studies, the morphological information present in the surface form of the word is handled only through the use of character based embeddings. Although this is not a limiting factor for languages which are not morphologically rich, it has been shown that employing morphologically disambiguated tags when representing words in a neural architecture improves the NER performance (Gungor et al., 2017; Straková et al., 2016).

There has been other approaches to the NER task for morphologically rich languages (Demir and Özgür, 2014; Seker and Eryiğit, 2012; Yeniterzi, 2011; Tür et al., 2003; Hasan et al., 2009). A study which can be considered as one of the first attempts in tackling NER for morphologically rich languages uses a hidden Markov model and takes the morphological tag sequence as input along with others like the surface form, capitalization features and similar features (Tür et al., 2003). In a study which basically depends on handcrafted features given to a CRF-based sequence tagger system, the word morphology was captured using the first and last three characters of the word as a feature resulting in an improvement in the NER tagging performance for Bengali (Hasan et al., 2009). In another study (Yeniterzi, 2011), a similar approach is taken with features generated using the output of an external morphological disambiguator and also shown to improve the performance. Another study (Seker and Eryiğit, 2012) uses the

same method but with a different approach for extracting morphological information, where they show an improvement over the previous state of the art results of Yeniterzi (2011). The first study focusing on morphologically rich languages to employ neural networks (Demir and Özgür, 2014) contains a regularized averaged perceptron (Freund and Schapire, 1999) and relies on handcrafted rules along with pretrained word embeddings. However, they refrain from using output from external morphological disambiguators and only rely on the first and last few characters of a word as features. Our work in this paper differs from these studies as it does not rely on handcrafted features. We represent words as fixed length vectors, employ morphological information to disambiguate the correct morphological analysis, and then combine them in such a way to obtain a context vector to label with NER tags.

In a recent study on morphological disambiguation (Yildiz et al., 2016), the authors propose a two-layer network for prediction. In the first layer, they process the candidate morphological analyses along with the correctly predicted analyses of previous words and obtain a vector to be processed in the second layer. The second layer takes all vectors propagated from the previous words and computes a `softmax` function over positive and negative classes. They predict the correct morphological analysis starting from the first word and use this prediction in the next word positions. The model is evaluated on a dataset manually labeled by the authors and considered as the state of the art for Turkish and competitive for French and German. Our proposed model for morphological disambiguation relies on scoring the candidate morphological analyses to predict the correct one for a word in a sentence. We borrow this idea from Shen et al. (2016). In their study, they feed the word representations to a Bi-LSTM and obtain context embeddings for each position. Using these embeddings, they score each morphological analysis by calculating a similarity function reaching the state of the art or competitive results for Turkish, Russian and Arabic.

Most of the work in morphological disambiguation or tagging strictly depend on their chosen specific output format for morphological analysis. This is due to the fragmented nature of computational approaches to morphological analysis for every language in the literature. However, we argue that our approach is immune to this problem as all of these output formats can be treated as a sequence. An example from Finnish is '`raha+[POS=NOUN]+[NUM=SG]+[CASE=ADE]`' (Silfverberg et al., 2016), another from Turkish is '`Ankara+Noun+Prop+A3sg+Pnon+Loc`' (Oflazer, 1994), and one for Hungarian is '`hír+NOUN+Case=Nom+Number=Plur`' (Trón et al., 2005). All of these can be split by the '+' symbol and transformed into a root and tag sequence. Moreover, there is an attempt in the area to unify the morphological annotation along with syntax annotation across many languages which will contribute more towards a solution (Nivre et al., 2016).

Many models targeting NLP tasks are designed to work independently although they usually employ linguistic information related with other tasks. Given that there are state of the art models which are similar in the sense that they all employ a sentence level Bi-LSTM, it is reasonable to hypothesize that jointly learning several tasks will improve the performance as shown in the literature (Hashimoto et al., 2017; Luong et al., 2015). In a recent study, it has been suggested that using separate layers for separate tasks is better rather than using the same (or usually top) layer for all the tasks (Søgaard and Goldberg, 2016).

3 Models

We test our hypothesis by training a number of models where we choose to enable or disable the selected components and features[1]. We start by explaining two basic models for each task: (i) a Bi-LSTM based sequence tagger where we predict the correct NER tags with a CRF (Section 3.1), (ii) a Bi-LSTM tagger which is used to represent the context for selecting the correct morphological analysis at the given position (Section 3.2). The joint models are combinations of these two basic models in various ways (Section 3.3).

[1]The code to replicate the experiment environment and the actual source code is published at `https://github.com/onurgu/joint-ner-and-md-tagger`

3.1 NER Model

We formally define an input sentence as $X = (x_1, x_2, \ldots, x_n)$ where each x_i is a vector of size l and the corresponding NER labels as $y_{\text{NER}} = (y_{\text{NER},1}, y_{\text{NER},2}, \ldots, y_{\text{NER},n})$. x_i are then fed to a Bi-LSTM which is composed of two LSTMs (Hochreiter and Schmidhuber, 1997) treating the input forwards and backwards. The output of this Bi-LSTM at position i, h_i, is a vector of size $2p$ where p is the size of the LSTM cell. Further, we transform h_i through a fully connected layer $\texttt{FC}_{last}$ with $\texttt{tanh}$ activations at the output to combine the left and right contexts into a vector of size p. This is followed by another fully connected layer to obtain a vector s_i of size K, where K is the number of the NER tags.

We follow a conditional random field (CRF) based approach to model the dependencies between the consequent tokens (Lafferty et al., 2001). To do this, we take the vector s_i at each position i as the score vector of the corresponding word and aim to minimize the following loss function $\texttt{loss}_{\text{NER}}(X, y_{\text{NER}})$ for a single sample sentence X:

$$-\sum_{i=0}^{n} A_{y_i, y_{i+1}} - \sum_{i=1}^{n} s_{i,y_i} + logZ(X)$$

where $A_{i,j}$ represents the score of a transition from tag i to j, $Z(X) = \sum_{y' \in \mathbb{Y}} \texttt{exp}\left(\sum_{i=0}^{n} A_{y_i, y'_{i+1}} + \sum_{i=1}^{n} s_{i,y'_i}\right)$ where $\mathbb{Y}$ is the set of all possible label sequences. Using this model, we decode the most probable tagging sequence y^*_{NER} as $\text{argmax}_{\tilde{y}_{\text{NER}}} \texttt{loss}_{\text{NER}}(X, \tilde{y}_{\text{NER}})$. We call this basic model the NER model (Lample et al., 2016) (see Figure 2).

In the remaining part of the section, we explain the details of the word representations used in this study.

Representing words. As the default setting, we obtain word and character based embeddings as described below and combine them by concatenation. For the first component, we allocate a word embedding vector of size w_d for every word in our dataset. This can be loaded from a pretrained word embeddings database as is done frequently in the literature, but we chose to learn the word embeddings during training. The second component is generated from the surface forms. We feed the character sequence of the word into a Bi-LSTM as described at the beginning of this section. However, instead of using the outputs of LSTM cells at each position, we just take the last and the first cell outputs of the forward and backward LSTMs and concatenate them (Figure 1). The resulting representation is two times the length of one character embedding length, $2\texttt{ch}_d$. This second component is in turn concatenated with the first component to obtain a word representation vector x_i of size $w_d + 2\texttt{ch}_d$.

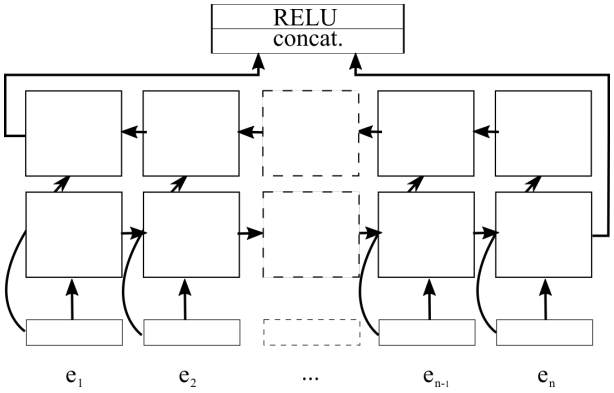


Figure 1: The basic model to generate representations for surface forms, roots, and morphological tag sequences. The input sequence $(e_1, e_2, \cdots, e_{n-1}, e_n)$ can either be the characters of the surface form, the characters of the root of the word, or the tags in the morphological tag sequence. RELU unit is active only for root and morphological tag sequences.

External morphological features. In order to compare our models with a previous method (Gungor et al., 2017), we utilize the golden morphological analysis provided with the dataset in addi-

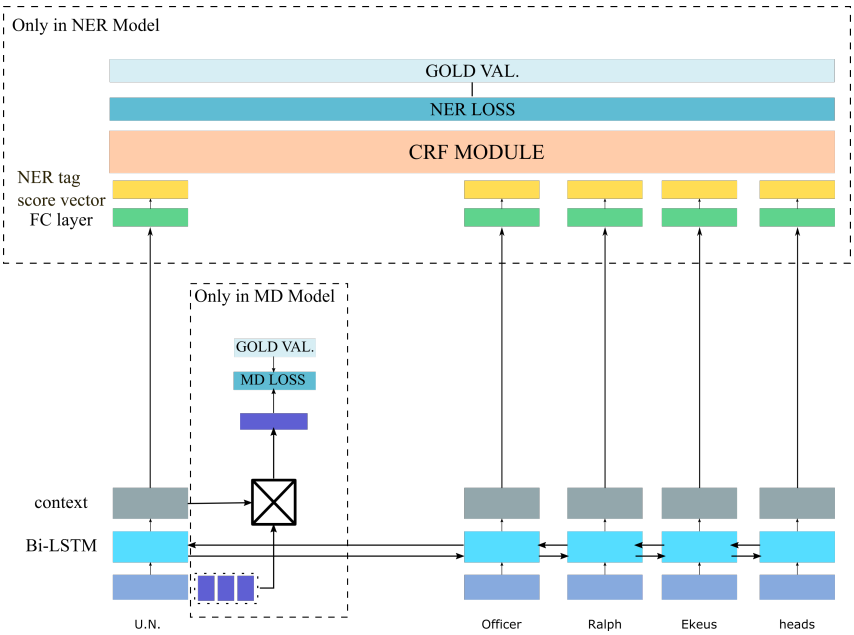


Figure 2: Our basic models: NER and MD. The portions of the model which are only active either for NER or MD models are indicated with dashed lines. The symbol ⊠ represents the selection of $\texttt{ma}_{ij^*}$.

tion to the word and character based embeddings and call this model EXT_M_FEAT. The best approach reported by Gungor et al. (2017) treats the string form of a morphological analysis as a sequence of characters and apply the process depicted in Figure 1. For example, a morphological analysis in Hungarian is '`Magyar+PROPN+Case=Nom+Number=Sing`' in string form and can be split into a list of characters as `(M,a,g,y,a,r,+,P,R,O,P,N,+,C,a,s,e,=,N,o,m,+,N,u,m,b,e,r,=,S,i,n,g)`. Using the sequence of characters of the morphological analysis instead of the sequence of morphemes might seem counterintuitive at first glance. However it has been argued that a benefit of treating morphological analyses as sequences of characters is the information conveyed by the characters within the tags. For example, in Turkish, the tags '`A3sg`' and '`A3pl`' indicate third person singular and third person plural where the leading two characters '`A3`' indicate third person agreement. This allows the model to represent the fragments of the tags which may improve the training performance. In this case, '`A3`' would represent the third person agreement independent of the singular or plural case. The resulting vector representation is thus of length $2\texttt{mt}_d$ which is added to word and character based embeddings to obtain a word representation of $w_d + 2\texttt{ch}_d + 2\texttt{mt}_d$.

3.2 MD Model

In this section, we describe our model for morphological disambiguation which is based on (Shen et al., 2016). In this model, we are given a sentence X in the same form as in the NER task, however we optimize the model to predict y_{MD} where $y_{\text{MD},i}$ represents the correct morphological analysis out of the candidate morphological analyses for word i. Like in the NER model, the MD model also employs a Bi-LSTM layer to obtain context representations when fed with the word representations x_i (Figure 2). We define the candidate morphological analyses for word i as $\texttt{ma}_i = \{\texttt{ma}_{i,1}, \texttt{ma}_{i,2}, \cdots, \texttt{ma}_{i,j}, \cdots, \texttt{ma}_{i,K}\}$. To determine the correct morphological analysis, we examine each morphological analysis output form to extract the root surface form and the morpheme sequence and generate the representation $\texttt{ma}_{ij}$ which we explain below.

We design this approach to be generalizable to many morphological analysis output forms described in Section 2. We give an example from Turkish here: the unique analysis of the Turkish word "Moda'da" is "`Moda+Noun+Prop+A3sg+Pnon+Loc`". The word literally means 'in Moda' (which is a district in Istanbul) and a common morpheme naming convention is used (Oflazer, 1994). So, we determine the root as 'Moda' and the morpheme sequence as '`(Noun, Prop, A3sg, Pnon, Loc)`'. The root

and the morpheme sequence are used to generate a representation as depicted in Figure 1. Except in this case the RELU activation function (Nair and Hinton, 2010) is also applied to the concatenation of the root and morpheme sequence representations. We choose the resulting representations r_{ij} and $\texttt{ms}_{ij}$ to be of two times the length of a morpheme embedding $\texttt{mt}_d$. Furthermore, we add the root representation vector r_{ij} and the morpheme sequence representation vector $\texttt{ms}_{ij}$ and apply hyperbolic tangent function ($\texttt{tanh}$), thus the morphological analysis representation $\texttt{ma}_{ij}$ is defined as follows $\texttt{tanh}(r_{ij} + \texttt{ms}_{ij})$.

We then select the morphological analysis ma_{ij^*} by performing a dot product with the context vector h_i: $\texttt{ma}_{ij^*} = \text{argmax}_j\, h_i \cdot \texttt{ma}_{ij}$ when decoding. During training, the loss $\texttt{loss}_{\text{MD}}(X, y_{\text{MD}})$ is calculated as

$$-\sum_{i=1}^{n} \texttt{log softmax}(\texttt{mscore}_i)$$

over a score vector $\texttt{mscore}_i$ such that $\texttt{mscore}_{ij} = \{h_i \cdot \texttt{ma}_{ij}\}$.

3.3 Joint model for NER and MD

We have experimented with three approaches for jointly learning NER and MD tasks. In this section, we explain the details of each approach.

Integration mode 1 - In this scheme, we employ a Bi-LSTM layer which is fed with word representations as in the basic models, NER and MD. We then use the same context h_i to calculate the losses separately for NER and MD as explained in Sections 3.1 and 3.2. We call this joint model JOINT1 and show in Figure 3a. We then learn the model parameters to optimize $\texttt{loss}_{\text{JOINT1}}$

$$\texttt{loss}_{\text{NER}}(X, y_{\text{NER}}) + \texttt{loss}_{\text{MD}}(X, y_{\text{MD}}).$$

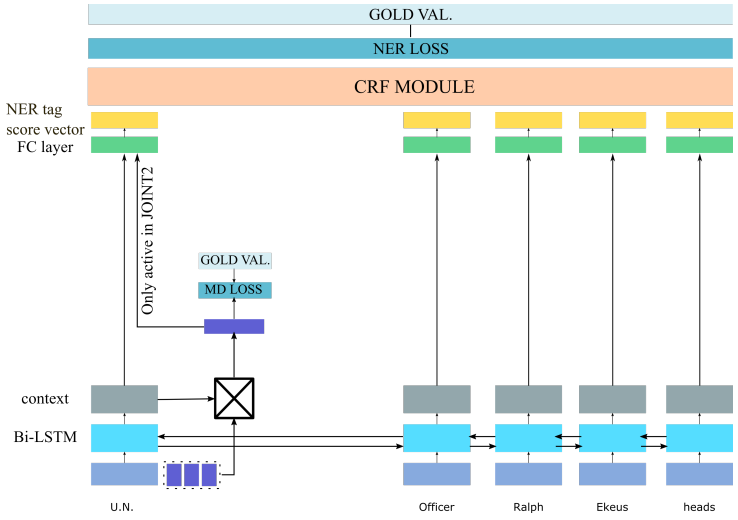
Integration mode 2 - As in the JOINT1 model, this model also calculates separate losses for each task and sums them to obtain a single loss to optimize. However, we additionally concatenate the selected morphological analysis representation $\texttt{ma}_{ij^*}$ to h_i before feeding it into the fully connected network with $\texttt{tanh}$ outputs as described in Section 3.1. The model is shown in Figure 3a. The rationale of this concatenation is to facilitate information flow from the disambiguated morphological analysis. We call this model JOINT2. The loss function $\texttt{loss}_{\text{JOINT2}}$ of this model is then calculated similar to $\texttt{loss}_{\text{JOINT1}}$.

Multilayer and Shortcut Connections. Our most complicated model is employing three Bi-LSTM layers instead of only one. We basically feed the output of the first layer h_i^1 to layer 2, the output of the second layer h_i^2 to layer 3. In addition to this, we transfer the word representation x_i to all layer inputs and concatenate with h_i^{level} to obtain $\overline{h}_i^{level}$. When processing to obtain the third layer's output $\overline{h}_i^3$, we also concatenate the selected morphological analysis representation $\texttt{ma}_{ij^*}$ to h_i^3 in addition to x_i. This is done to propagate the information gained from the disambiguated morphological analysis to the last layer of the network. We use the first layer's output h_i^1 when calculating $\texttt{mscore}_i$ as shown to be better for a variety of tasks (Hashimoto et al., 2017). We call this model J_MULTI and depict in Figure 3b.

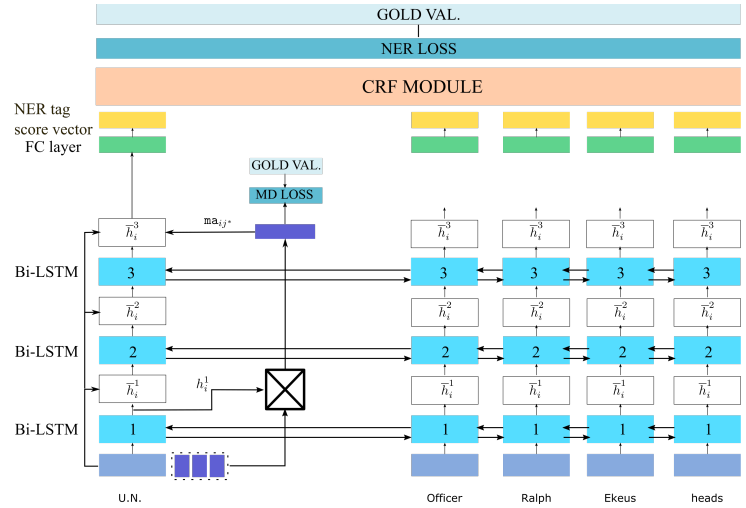
4 Data

To test our proposed model, we derived a new dataset based on a dataset commonly used in the literature for the NER task for Turkish (Tür et al., 2003). This dataset contains sentences from the online edition of a Turkish national newspaper with NER labels. The creators of the dataset also provide a golden morphological analysis along with each word. However, golden morphological analyses in this dataset are sometimes erroneous. For example, words which are inflections of foreign words are usually problematic. An example is "Hillary'nin" which is the genitive case for the word "Hillary". It has been incorrectly labeled as if it is in nominal case. Also, when the surface form is a number in some noun case, like "98'e" which is the dative case for the number ninety eight, the morphological analysis is almost always nominal. We believe the reason for this is the incorrect handling of the quote character when preparing the original version.

In our study, we have first divided the training portion of the original dataset into training and development sets. We then augmented these portions using candidate morphological analyses for each word



(a) (i) Model JOINT1: two losses for two tasks sharing a Bi-LSTM. (ii) Model JOINT2: We concatenate the selected morphological analysis' vector representation to the last layer's context vector.



(b) Model J_MULTI: We employ shortcut connections and two more Bi-LSTM layers.

Figure 3: Our joint models: (a) JOINT1 and JOINT2 models (b) J_MULTI model. The symbol ⊠ represents the selection of $\mathtt{ma}_{ij^*}$.

with a commonly used morphological analyzer (Oflazer, 1994). Unfortunately, the golden morphological analyses in about 5% of the word tokens were not found in these candidate analyses. To mitigate this issue, we listed the most frequent contexts where a specific mismatch happens, selected the most suitable morphological analysis out of the candidates for each context, thus providing a solution to the mismatch. We then automatically corrected all contexts with a mismatch which has a solution in our solution database. Although we tried to give the utmost attention to selecting the best solution, some of our solutions might be problematic. Thus, we share the data, the scripts and the tool which helps the user to select a solution as described for academic use and examination[2]. Unfortunately, there were still left a few hundred mismatches. As providing a solution for them required a lot of manual work and would only save 1-2 sentences for each, we just removed any sentence that contains any of these mismatches. This way, we have retained 25511 out of 28835 sentences in the original dataset for training, 2953 of 3336 for development and 2913 of 3328 for test. By this process, despite losing some of the sentences,

[2]The data can be found at `https://github.com/onurgu/joint-ner-and-md-tagger`

we have built a new dataset with both the NER labels and the candidate morphological analyses which have correct golden labels.

4.1 Training

We implemented the model using the DyNet Neural Network Toolkit in Python. The model parameters are basically the word embeddings, the parameters of Bi-LSTMs, the weights of the fully connected layer $\texttt{FC}_{last}$, and the CRF transition matrix A. We trained by calculating the gradients of the loss for a batch of five sentences consisting of surface forms and its associated NER and/or MD labels and updated the parameters with Adam (Kingma and Ba, 2014) for 50 epochs and reported the performance on test set of the model with the highest development set performance. We applied dropout (Srivastava et al., 2014) with probability 0.5 to the word representations x_i. To facilitate the reproducibility of our work, we also provide our system as a virtual environment[3] that provides the same environment on which we evaluated our system in an open manner.

5 Results

To test our approach, we train and evaluate every model for 10 times and report the mean F1-measure value for named entity recognition and accuracy for morphological disambiguation. This is done to decrease the potential negative effects of random initialization of model parameters as shown in the literature (Reimers and Gurevych, 2017). To accomplish this given our limited computing resources, we set the parameter dimension sizes to 10 and do not employ pretrained word embeddings.

The results are shown in Table 1. We see that the mean NER performance increases in joint models. We see that the JOINT2 model is performing better than just calculating two losses at the last layer as we did in the JOINT1 model. However, applying the Welch's t-test between the JOINT1 and JOINT2 runs does not strongly imply this difference ($p = .24$). Adding multiple Bi-LSTM layers to JOINT2 and obtaining J_MULTI also helped and achieved the best score among our joint models[4]. Employing Welch's t-test confirms the significance of this difference with other joint models, $p < .05$ for each pair.

This work	
Model	Mean F1-measure
NER	81.07
JOINT1	81.28
JOINT2	81.84
J_MULTI	**83.21**
Previous work	
EXT_M_FEAT	**83.47**

Table 1: Evaluation of our models for NER performance with our dataset. We report F1-measure results over the test portion of our dataset averaged over 10 replications of the training with the same hyper parameters.

To make a comparison with a previous method (Gungor et al., 2017), we also evaluated a model where the golden morphological analysis in the corpus is represented as a vector and included in the word representation x_i, namely EXT_M_FEAT (see Section 3.1). As one can see from the table, it achieved the best results compared to our joint models. However, we cannot confirm the difference between EXT_M_FEAT and J_MULTI models as the calculated p is well above .05. Thus our best performing model J_MULTI is performing at a competitive level with an additional advantage of disambiguating the morphological tags while predicting the NER tags. This also serves as another confirmation to the

[3] You can obtain our implementation and find more information about how to use our virtual environment at `https://github.com/onurgu/joint-ner-and-md-tagger`.

[4] One can wonder whether this performance improvement could be due to an increase in the total number of parameters of the model. We saw that the increase is negligible as it only accounted for a 2% increase.

hypothesis that employing linguistic information such as morphological features leads to an increase in the NER performance.

This work	
Model	Mean Accuracy
MD	88.61
JOINT1	88.17
JOINT2	86.86
J_MULTI	88.05
Previous work	
Yuret and Türe (2006)	89.55
Shen et al. (2016)	**91.03**

Table 2: Evaluation of our models for MD performance. As in the NER evaluation, we report accuracies over the test dataset averaged over 10 replications of the training.

To evaluate the performance of morphological disambiguation, we have tested the MD performance of our models, which are trained with the training portion of our dataset, on the test portion of a frequently used dataset (Yuret and Türe, 2006). As can be seen from Table 2, we are very close to the state of the art MD performance even if we only trained with a low number of parameters as stated in the beginning of this section. We have to also note that in contrast with the NER task, the MD task did not enjoy a performance increase from joint learning. The mean morphological disambiguation accuracies for the test portion of our dataset also suggests the same, all hovering around 77% without much change. This might be due to the fact that NER can utilize the disambiguated morphological analysis of a word to predict the correct label, however a correctly predicted NER label does not contribute to the disambiguation of the word's morphology.

6 Conclusions

In this work, we propose a joint model of NER and MD tasks that removes the need for external morphological disambiguators. The method is applicable to every language given that one can provide the candidate morphological analyses for a word, making this approach portable to many languages. We have also shown that joint learning leads to an increase in the NER tagging performance. However, there is more work to do as we are still bound to language specific tools in obtaining the list of candidate morphological analyses. Generating the list of candidate analyses within the model, testing our hypothesis on other morphologically rich languages, and testing with models which have higher number of parameters are left for future work.

Acknowledgements

This study was supported by the Boğaziçi University Research Fund (BAP 13083).

References

Douglas E Appelt, Jerry R Hobbs, John Bear, David Israel, Megumi Kameyama, David Martin, Karen Myers, and Mabry Tyson. 1995. SRI International FASTUS system: MUC-6 test results and analysis. In *Proceedings of the 6th Conference on Message Understanding*, pages 237–248. ACL.

Andrew Eliot Borthwick. 1999. *A Maximum Entropy Approach to Named Entity Recognition*. Ph.D. thesis, New York University, New York, NY, USA.

Hakan Demir and Arzucan Özgür. 2014. Improving named entity recognition for morphologically rich languages using word embeddings. In *Machine Learning and Applications (ICMLA), 2014 13th International Conference on*, pages 117–122. IEEE.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of ACL*, pages 363–370. ACL.

Yoav Freund and Robert E Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296.

Onur Gungor, Eray Yildiz, Suzan Uskudarli, and Tunga Gungor. 2017. Morphological embeddings for named entity recognition in morphologically rich languages. *arXiv preprint arXiv:1706.00506*.

Honglei Guo, Huijia Zhu, Zhili Guo, Xiaoxun Zhang, Xian Wu, and Zhong Su. 2009. Domain adaptation with latent semantic association for named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the NAACL*, pages 281–289. ACL.

Kazi Saidul Hasan, Vincent Ng, et al. 2009. Learning-based named entity recognition for morphologically-rich, resource-scarce languages. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 354–362. Association for Computational Linguistics.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *EMNLP*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8).

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Kevin Humphreys, Robert Gaizauskas, Saliha Azzam, Chris Huyck, Brian Mitchell, Hamish Cunningham, and Yorick Wilks. 1998. University of Sheffield: Description of the LaSIE-II system as used for MUC-7. In *Proceedings of the Seventh Message Understanding Conferences (MUC-7)*. ACL.

Jing Jiang and Cheng Xiang Zhai. 2007. Instance weighting for domain adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, volume 7, pages 264–271, Prague. ACL.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

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

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. *arXiv preprint arXiv:1603.01354*.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the 7th Conference on Natural Language Learning at HLT-NAACL*, volume 4, pages 188–191. Association for Computational Linguistics.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *LREC*.

Kemal Oflazer. 1994. Two-level description of Turkish morphology. *Literary and Linguistic Computing*, 9(2):137–148.

Nils Reimers and Iryna Gurevych. 2017. Optimal hyperparameters for deep LSTM-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799*.

Gökhan Akın Seker and Gülşen Eryiğit. 2012. Initial explorations on using CRFs for Turkish named entity recognition. In *Proceedings of COLING 2012*, pages 2459–2474, Mumbai, India, December. The COLING 2012 Organizing Committee.

Qinlan Shen, Daniel Clothiaux, Emily Tagtow, Patrick Littell, and Chris Dyer. 2016. The role of context in neural morphological disambiguation. In *COLING*, pages 181–191.

Miikka Silfverberg, Teemu Ruokolainen, Krister Lindén, and Mikko Kurimo. 2016. FinnPos: an open-source morphological tagging and lemmatization toolkit for Finnish. *Language Resources and Evaluation*, 50(4):863–878.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 231–235.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Milan Straka and Jana Straková. 2017. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99.

Jana Straková, Milan Straka, and Jan Hajič. 2016. Neural networks for featureless named entity recognition in Czech. In *International Conference on Text, Speech, and Dialogue*, pages 173–181. Springer.

György Szarvas, Richárd Farkas, and András Kocsor. 2006. A multilingual named entity recognition system using boosting and C4.5 decision tree learning algorithms. In *International Conference on Discovery Science*, pages 267–278. Springer.

Viktor Trón, András Kornai, György Gyepesi, László Németh, Péter Halácsy, and Dániel Varga. 2005. Hunmorph: open source word analysis. In *Proceedings of the Workshop on Software*, pages 77–85. Association for Computational Linguistics.

Gökhan Tür, Dilek Hakkani-Tür, and Kemal Oflazer. 2003. A statistical information extraction system for Turkish. *Natural Language Engineering*, 9(2):181–210.

Dan Wu, Wee Sun Lee, Nan Ye, and Hai Leong Chieu. 2009. Domain adaptive bootstrapping for named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 3, pages 1523–1532. ACL.

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

Reyyan Yeniterzi. 2011. Exploiting morphology in Turkish named entity recognition system. In *Proceedings of the ACL 2011 Student Session*, HLT-SS '11, pages 105–110, Stroudsburg, PA, USA. Association for Computational Linguistics.

Eray Yildiz, Caglar Tirkaz, H Bahadir Sahin, Mustafa Tolga Eren, and Omer Ozan Sonmez. 2016. A morphology-aware network for morphological disambiguation. In *30th AAAI Conference on Artificial Intelligence*.

Deniz Yuret and Ferhan Türe. 2006. Learning morphological disambiguation rules for Turkish. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 328–334. Association for Computational Linguistics.

Farewell Freebase: Migrating the SimpleQuestions Dataset to DBpedia

Michael Azmy*, Peng Shi*, Jimmy Lin, and Ihab F. Ilyas
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
{mwazmy,p8shi,jimmylin,ilyas}@uwaterloo.ca

Abstract

Question answering over knowledge graphs is an important problem of interest both commercially and academically. There is substantial interest in the class of natural language questions that can be answered via the lookup of a single fact, driven by the availability of the popular SIMPLEQUESTIONS dataset. The problem with this dataset, however, is that answer triples are provided from Freebase, which has been defunct for several years. As a result, it is difficult to build "real-world" question answering systems that are operationally deployable. Furthermore, a defunct knowledge graph means that much of the infrastructure for querying, browsing, and manipulating triples no longer exists. To address this problem, we present SIMPLEDBPEDIAQA, a new benchmark dataset for simple question answering over knowledge graphs that was created by mapping SIMPLEQUESTIONS entities and predicates from Freebase to DBpedia. Although this mapping is conceptually straightforward, there are a number of nuances that make the task non-trivial, owing to the different conceptual organizations of the two knowledge graphs. To lay the foundation for future research using this dataset, we leverage recent work to provide simple yet strong baselines with and without neural networks.

1 Introduction

Question answering over knowledge graphs is an important problem at the intersection of multiple research communities, with many commercial deployments. To ensure continued progress, it is important that open and relevant benchmarks are available to support the comparison of various techniques. In this paper, we focus on the class of questions that can be answered by a single triple (i.e., fact) from a knowledge graph. For example, the question "What type of music is on the album Phenomenon?" can be answered via the lookup of a simple fact—in this case, the "genre" property of the entity "Phenomenon". Analysis of an existing benchmark dataset (Yao, 2015) and real-world user questions (Dai et al., 2016; Ture and Jojic, 2017) show that such questions cover a broad range of users' needs.

The SIMPLEQUESTIONS dataset (Bordes et al., 2015) has emerged as the de facto benchmark for evaluating these simple questions over knowledge graphs. However, there is one major deficiency with this resource: the answers draw from Freebase. Unfortunately, Freebase is defunct and no longer maintained. This creates a number of insurmountable challenges: First, because the knowledge graph is stale, it is no longer possible to build a "real-world" operational QA system using models trained on SIMPLEQUESTIONS. Second, a defunct knowledge graph means that researchers must develop custom infrastructure for querying, browsing, and manipulating the graph. Thus, we are not able to leverage multiple cooperative and interchangeable service APIs that are deployed and maintained by different parties—which is the strength of the broader "open linked data" ecosystem. While it may be the case that one can apply transfer learning so that models trained on SIMPLEQUESTIONS can be re-targeted to another "live" knowledge graph, we are not aware of research along these lines.

*These authors contributed equally.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2093–2103
Santa Fe, New Mexico, USA, August 20-26, 2018.

Dataset	Training	Validation	Test	Total
SIMPLEQUESTIONS	75,910	10,845	21,687	**108,442**
SIMPLEDBPEDIAQA	30,186	4,305	8,595	**43,086**

Table 1: Statistics of SIMPLEQUESTIONS and SIMPLEDBPEDIAQA.

To address these issues, we present SIMPLEDBPEDIAQA, a new dataset that we have created by mapping entities and predicates that comprise the answers to SIMPLEQUESTIONS from Freebase to DBpedia. Unlike Freebase, DBpedia is actively maintained by a dedicated community. We describe how this dataset migration is accomplished via high-quality alignments between entities in the two different knowledge graphs, and explain many of the nuances that make the creation of this dataset non-trivial. Our new dataset includes a total of 43,086 questions and corresponding answers that cover 40% of the original dataset. Summary statistics of SIMPLEDBPEDIAQA and SIMPLEQUESTIONS are shown in Table 1. The complete dataset is available at https://github.com/castorini/SimpleDBpediaQA.

In addition to the contribution of providing the community with a new evaluation resource, we provide a series of simple yet strong baselines to lay the foundation for future work. These baselines include neural network models and other techniques that do not take advantage of neural networks, building on recently-published work (Mohammed et al., 2018). An additional contribution of this paper is that having two parallel datasets allows us to examine the effects of different conceptual organizations and knowledge graph structures: For example, we notice that many single-fact triples in Freebase require two-hop traversals in the DBpedia knowledge graph, which makes them no longer "simple" questions. Finally, evaluation resources targeting different conceptual organizations of knowledge help "keep researchers honest" in guarding against model overfitting on a single dataset.

2 Background and Related Work

The development and continual advance of question answering techniques over knowledge graphs require benchmark datasets that cover different aspects of the task. Quite obviously, each dataset has to target one (or more) knowledge graphs, which means that the structure of the answers are dictated by the conceptual organization of the particular knowledge graph.

Over the years, researchers have built a number of datasets based on Freebase (Bollacker et al., 2008). For instance, FREE917 (Cai and Yates, 2013) contains 917 questions involving 635 distinct Freebase predicates. WEBQUESTIONS (Berant et al., 2013) contains 5,810 question-answer pairs collected using the Google Suggest API and manually answered using Amazon Mechanical Turk (AMT). Both contain answers that require complex, multi-hop traversals of the knowledge graph. In contrast, the SIMPLEQUESTIONS dataset focuses on questions that can be answered via the lookup of a single fact (i.e., triple). Due to its much larger size and thus support for data-hungry machine learning techniques, this dataset has gained great popularity with researchers. Unfortunately, Google shut down Freebase in 2015; a final snapshot of the knowledge graph is still available online for download, but the associated APIs are no longer available.

Like Freebase, DBpedia (Bizer et al., 2009) has also been used as the target knowledge graph for multiple question answering datasets. For example, QALD[1] (Question Answering over Linked Data) is a series of evaluation campaigns focused on question answering over linked data. LC-QUAD (Trivedi et al., 2017) is another recent dataset that comprises 5,000 questions with answers in the form of SPARQL queries over DBpedia. These questions are relatively complex and require the integration of evidence from multiple triples. However, a more recent analysis by Singh et al. (2018) found that only 3,252 of the questions returned answers using the provided queries.

We are not the first to attempt to migrate SIMPLEQUESTIONS to another knowledge graph. Diefenbach et al. (2017) mapped the dataset from Freebase to Wikidata.[2] However, our migrated SIMPLE-

[1] https://qald.sebastianwalter.org/

[2] https://www.wikidata.org/wiki/Wikidata:Main_Page

DBPEDIAQA dataset has roughly twice the number of mapped questions. DBpedia is generally considered to be more mature than Wikidata due to its longer history, and thus we believe targeting DBpedia will ultimately yield higher-impact applications.

3 Problem Definition

We begin with a formal definition of our problem. Some preliminaries: Let $E = \{e_1, e_2, \cdots, e_r\}$ be a set of entities, where e_i is a Uniform Resource Identifier (URI) uniquely identifying each entity. Let $P = \{p_1, p_2, \cdots, p_s\}$ be a set of predicates. Let $S \subseteq E$ be a set of subjects and $O \subseteq (L \cup E)$ be a set of objects, where L is a set of literals. In this context, $t = (s, p, o)$ denotes a Resource Description Framework (RDF) triple, comprised of a subject $s \in S$, a predicate $p \in P$, and an object $o \in O$.

Given this formalism, Freebase (Bollacker et al., 2008) represents a specific knowledge graph T^b, where $T^b = \{t_1^b, \cdots, t_m^b\}$ (i.e., a set of Freebase triples). Each Freebase entity is uniquely identified by a MID (Machine ID). Similarly, DBpedia (Bizer et al., 2009) represents another knowledge graph T^d, where $T^d = \{t_1^d, \cdots, t_n^d\}$.

The SIMPLEQUESTIONS dataset is a collection of natural language questions and answers based on Freebase. Formally, $Q^b = \{q_1^b, \cdots, q_l^b\}$, where $q_i^b = (\mathcal{Q}_i, t_j^b)$; $\mathcal{Q}_i$ is a natural language question and $t_j^b \in T^b$ is a Freebase triple that supplies the answer to that question.

For example, the question "Who wrote The New Canada?" has the following answer triple:

(fb:m/02qtvzv, fb:book/written_work/author, fb:m/01hxz2)

where fb stands for the prefix http://www.freebase.com/. The subject of the above answer triple is referred to as the topic entity, and the object of the triple is referred to as the answer entity. To answer the natural language question, a system must correctly identify the topic entity and the predicate, and then consult the knowledge graph to look up the answer entity.

Given Freebase T^b, DBpedia T^d, and SIMPLEQUESTIONS Q^b, our problem can be formally defined as follows: for each $q_i^b = (\mathcal{Q}_i, t_j^b) \in Q^b$, find $q_i^d = (\mathcal{Q}_i, t_k^d)$, where $t_k^d \in T^d$ is a DBpedia triple, such that t_j^b is semantically equivalent to t_k^d. The result $Q^d = \{q_1^d, \cdots, q_l^d\}$ is our SIMPLEDBPEDIAQA dataset. Although this characterizes the basic structure of the problem, there are a number of nuances that deviate from this formalism, which we describe in the following sections.

4 Dataset Migration

Our overall strategy for dataset migration breaks down into the following steps: entity mapping, predicate mapping, and candidate refinement. At a high level, we begin by first mapping the topic and answer entities from Freebase to DBpedia; these then serve as anchors from which we can project the Freebase predicates to DBpedia. To assist in the process, we ingest the knowledge graphs into an RDF store to facilitate querying via SPARQL. For this effort, we use the latest version of DBpedia released in 2017.[3]

4.1 Entity Mapping

The first step is to map Freebase entities from SIMPLEQUESTIONS to entities in DBpedia. Freebase MIDs and DBpedia URIs are linked through the predicate http://www.w3.org/2002/07/owl#sameAs; these official mappings are released as part of DBpedia.[4] For each Freebase entity MID (topic entity or answer entity), we issue a SPARQL query to retrieve the corresponding DBpedia URI. For example, Justin Trudeau, the current Prime Minister of Canada, is mapped via the triple:

(dbr:Justin_Trudeau, http://www.w3.org/2002/07/owl#sameAs, fb:m/02b5jh).

Here and throughout the paper we use dbr as the DBpedia prefix for http://dbpedia.org/resource/. For approximately 56% of questions in SIMPLEQUESTIONS, we can map both the topic entity and the answer entity from Freebase to DBpedia. For the remaining questions, we are only able to map the topic entity, the answer entity, or neither. The detailed breakdowns are shown in Table 2.

[3] https://wiki.dbpedia.org/develop/datasets/dbpedia-version-2016-10

[4] http://downloads.dbpedia.org/2016-10/core-i18n/en/freebase_links_en.ttl.bz2

Note that the URI for Justin Trudeau can be used to uniquely identify this entity within the broader open linked data ecosystem. For example, a human-readable version of facts associated with this individual is located at http://dbpedia.org/page/Justin_Trudeau. This, as well as a variety of other libraries, toolkits, APIs, etc. provide infrastructure that simplifies the development of operational question answering systems. The existence of these resources illustrates one of the major benefits of migrating SIMPLEQUESTIONS over to a knowledge graph that is actively maintained by a dedicated community.

4.2 Predicate Mapping: One-Hop Predicates

Let us consider the case where we are able to map both the topic entity and the answer entity from Freebase to DBpedia. We can then issue a SPARQL query over DBpedia to enumerate the paths (sequence of one or more predicates) connecting those entities. In the simplest case, there is a single predicate connecting the topic entity to the answer entity, which yields a straightforward mapping of the triple from Freebase to DBpedia. This occurs for approximately half of the questions with successfully mapped topic and answer entities; see detailed statistics in Table 2.

Consider the question "Which city is McCormick Field in?" The Freebase topic entity fb:m/05_xgn is mapped to DBpedia as dbr:McCormick_Field and the answer entity is mapped from fb:m/0ydpd to dbr:Asheville,_North_Carolina. The DBpedia predicate dbo:location connects those two entities, which provides a valid and correct mapping for the Freebase predicate fb:location/location/containedby. Here and throughout the paper we use dbo as the DBpedia prefix for http://dbpedia.org/ontology/.

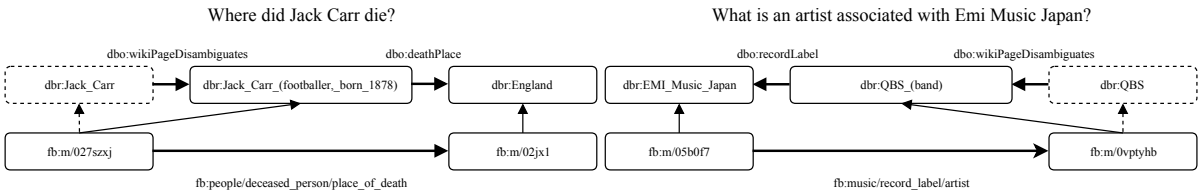
Due to differences in the conceptual organization of the two knowledge graphs, the directionality of equivalent predicates in Freebase and DBpedia may differ. For example, the DBpedia predicate dbo:birthPlace takes a person as the subject and a location as the object, whereas the equivalent predicate in Freebase fb:location/location/people_born_here inverts the subject and object. Therefore, for a question such as "Who was born in Aguascalientes?", the subject in the Freebase triple becomes the object in the DBpedia triple.

During the migration from Freebase to DBpedia, if the directionality of the mapped predicate is the same, we refer to the result as a forward predicate; if the directionality is reversed, we refer to the result as a backward predicate. We explicitly keep track of this metadata, which is necessary for the actual question answering task.

4.3 Predicate Mapping: Two-Hop Predicates

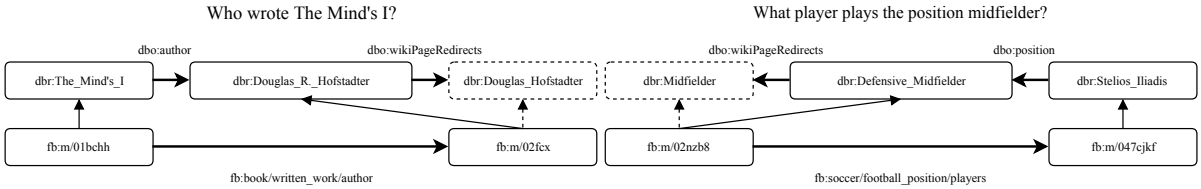
Next, we consider the more complex case where the topic entity and the answer entity are *not* directly connected by a single predicate in DBpedia. That is, the results of our SPARQL query over DBpedia to enumerate the paths connecting the mapped entities contain multiple hops. In this work, we only consider two-hop traversals, as even longer paths are generally rare and spurious. These two-hop predicates can be categorized into *disambiguation predicates*, *redirection predicates*, *complex predicates*, and *missing predicates*, detailed as follows:

- *Disambiguation Predicates*: DBpedia uses wikiPageDisambiguates predicates to disambiguate different entities with the same name. The DBpedia sameAs links, however, might map a Freebase MID to an ambiguous URI, thus yielding a two-hop traversal from the topic entity to the answer entity. In these cases, we can "compress" the path back into a single predicate by changing the original topic entity to the disambiguated entity. Note that this disambiguation process can occur with forward predicates, as in Figure 1a, where dbr:Jack_Carr is disambiguated to dbr:Jack_Carr_(footballer,_born_1878), as well as backward predicates, as in Figure 1b, where dbr:QBS is disambiguated to dbr:QBS_(band).
- *Redirections Predicates*: Similar to disambiguation links, DBpedia uses wikiPageRedirects predicates to redirect an entity to another entity (typically, the canonical variant of that entity). As with disambiguation predicates above, these two-hop redirection predicates can also be compressed into a single triple. For example, dbr:Douglas_Hofstadter is redirected back to dbr:Douglas_R._Hofstadter and dbr:Midfielder is redirected back to dbr:Defensive_Midfielder, as shown in Figure 2a and Figure 2b, respectively. Once again, this can occur with both forward and backward predicates.



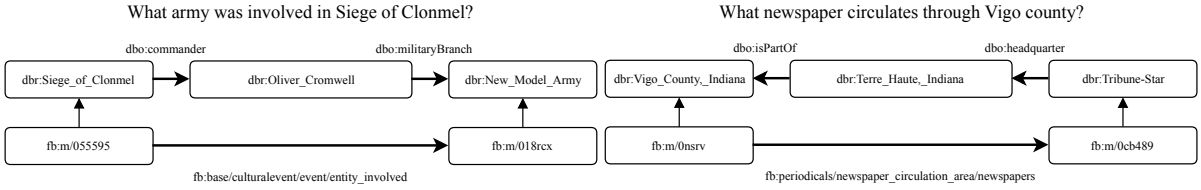
(a) Forward Predicate (b) Backward Predicate

Figure 1: Examples of mapping disambiguation predicates from Freebase to DBpedia.



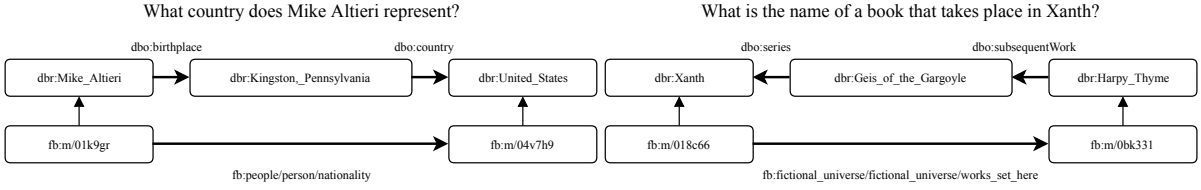
(a) Forward Predicate (b) Backward Predicate

Figure 2: Examples of mapping redirection predicates from Freebase to DBpedia.



(a) Forward Predicate (b) Backward Predicate

Figure 3: Examples of mapping complex predicates from Freebase to DBpedia.



(a) Forward Predicate (b) Backward Predicate

Figure 4: Examples of missing predicates in DBpedia.

- *Complex Predicates*: Due to differences in the conceptual organization of Freebase and DBpedia, there is no direct equivalent in DBpedia for some Freebase predicates. Instead, a chain of two predicates is necessary to capture the relationship between the topic and answer entities. An example is shown in Figure 3a: the question "What army was involved in Siege of Clonmel?" can be answered using the Freebase predicate fb:base/culturalevent/event/entity_involved, but in DBpedia the same fact requires a chain of two predicates, dbo:commander and dbo:militaryBranch. Note that this can also occur with backward predicates, as shown in Figure 3b.
- *Missing Predicates*: Some questions in DBpedia are answered using two-hop predicates even though there exists a one-hop predicate in the knowledge graph that represents a better mapping; this situation arises due to the incompleteness of DBpedia. Note that missing predicates actually represent a special case of complex predicates, which we only discovered by manual examination of the predicate mapping results. Nevertheless, it seems appropriate to separately categorize this particular type of predicate mismatch between Freebase and DBpedia. An example is shown in Figure 4a: The entity dbr:Mike_Altieri should have a predicate dbo:nationality that directly links to dbr:United_States, as is typical of person entities. However, since this predicate is missing, our SPARQL query discovered a roundabout path via dbo:birthplace then dbo:country. Figure 4b shows a similar case involving a backward predicate, where the entity dbr:Harpy_Thyme should have a predicate dbo:series that directly links to dbr:Xanth; instead, the answer entity is discovered via the extra hop dbo:subsequentWork. We believe that DBpedia can be enhanced by inserting these missing links, but augmenting DBpedia is beyond the scope of this work.

Detailed statistics of these two-hop predicate matches are shown in Table 2. As described above, there is no automatic way to differentiate between complex and missing predicates, and thus we provide the sum of the two categories. For questions that have both topic and answer entity mappings, we are not able to find any predicate mappings for approximately 34% of them.

4.4 Candidates Refinement

The output of the initial entity and predicate mapping process (as described above) is then refined to produce the final SIMPLEDBPEDIAQA dataset; detailed statistics are shown in Table 3. In this section, we detail the candidate refinement process.

The need for post-processing candidate results from the output of the processes described above is apparent from manual examination. While the entity mappings are generally of high quality, some of the mapped predicates are invalid, primarily due to two reasons:

- *Semantic drift:* Some candidate predicates are not semantically correct even though the answer may be factually correct. For example, consider the question "From where does Anjali Devi claim nationality?" The predicate mapping produces dbo:deathPlace instead of the correct predicate, dbo:nationality. This is because the correct predicate is missing for this entity, and by coincidence, this person's nationality is the same as her death place.
- *Predicate constraints:* In some cases, we observe mismatches between the domains of the subjects or objects of a Freebase predicate and its corresponding DBpedia predicate. For example, the DBpedia predicate dbo:author can take as subject books, movies, etc. However, the Freebase predicate fb:book/author/works_written can only be mapped to the DBpedia predicate dbo:author (in the backward direction) if the DBpedia subject has the type dbo:WrittenWork. More generally, a predicate mapping is valid only under certain type constraints.

To tackle these challenges with minimal manual effort, we construct manual rules that map high-frequency Freebase predicates in the initial mappings to all potentially correct (at the semantic level) DBpedia predicates. Each rule includes a Freebase predicate and a list of corresponding DBpedia predicates, an associated directionality (forward or backward), and an optional type constraint. A few examples are shown in Table 4. The interannotator agreement of these rules based on three human annotators

				Training	Validation	Test	Total
Mapped Entities	Mapped Predicates	One Hop		22,158	3,193	6,304	**31,655**
		Two Hop	Disambiguation	173	24	59	**256**
			Redirection	1,992	247	553	**2,792**
			Complex + Missing	3,504	501	985	**4,990**
	Not Mapped Predicates			14,875	2,059	4,373	**21,307**
Sub-Total				**42,702**	**6,024**	**12,274**	**61,000**
Not Mapped Entities	Only Answer Entity Mapped			18,040	2,635	5,174	**25,849**
	Only Topic Entity Mapped			9,611	1,415	2,648	**13,674**
	Both Not Mapped			5,557	771	1,591	**7,919**
Sub-Total				**33,208**	**4,821**	**9,413**	**47,442**
SIMPLEQUESTIONS **Total**				**75,910**	**10,845**	**21,687**	**108,442**

Table 2: Statistics from the initial mapping of entities and predicates in SIMPLEQUESTIONS.

				Training	Validation	Test	Total
Mapped Entities	Mapped Predicates	One Hop		19,271	2,773	5,467	**27,511**
		Two Hop	Disambiguation	84	17	32	**133**
			Redirection	1,547	191	429	**2,167**
			Complex + Missing	1,365	191	377	**1,933**
	Not Mapped Predicates			3,940	531	1,183	**5,654**
Sub-Total				**26,207**	**3,703**	**7,488**	**37,398**
Not Mapped Entities	Only Answer Entity Mapped			0	0	0	**0**
	Only Topic Entity Mapped			3,979	602	1,107	**5,688**
	Both Not Mapped			0	0	0	**0**
Sub-Total				**3,979**	**602**	**1,107**	**5,688**
SIMPLEDBPEDIAQA **Total**				**30,186**	**4,305**	**8,595**	**43,086**

Table 3: Final statistics of SIMPLEDBPEDIAQA following candidates refinement.

is 97%, where agreement is computed as the number of predicates that were identically labeled by all the annotators, divided by the count of all predicates.

Using these mapping rules, we can filter and discard spurious one-hop mappings (including the disambiguation and redirection cases) where both the topic and answer entities are correctly mapped. Furthermore, we can expand the dataset by applying these rules to a few additional cases. Consider the case of complex and missing predicate: since these questions have two-hop predicates, making them no longer "simple questions", they would have been discarded from our dataset. However, we can issue a SPARQL query using the topic entity and the mapped DBpedia predicates from our mapping rules to search for valid answers (ignoring the answer entity). If the query returns a result, we can add the question to our dataset. Heuristically, this means that the question *does* have an answer in DBpedia, just not the same as the one provided in Freebase.

The same process can be applied to cases where we have successfully mapped the entities but not the predicates, and even to cases where we have only successfully mapped the topic entity. As a concrete example, for the question "What is a song by John Rutter?", only the topic entity is mapped. Based on our rules, the Freebase predicate fb:music/artist/track is mapped to the DBpedia predicate dbo:artist with a constraint of dbo:MusicalWork in the backward direction. Using the topic entity as an anchor, a SPARQL query returns a valid result.

Detailed statistics from the refinement process are shown in Table 3. The final output of entity mapping, predicate mapping, and candidate refinement is our SIMPLEDBPEDIAQA dataset, which successfully migrates SIMPLEQUESTIONS from Freebase over to DBpedia.

Freebase Predicate	DBpedia Predicate	Directionality	Type Constraint
fb:architecture/structure/architect	dbo:architect	forward	-
fb:location/location/contains	dbo:country	backward	-
fb:baseball/baseball_position/players	dbo:position	backward	dbo:BaseballPlayer
fb:music/album_release_type/albums	dbo:type	backward	dbo:Album
fb:book/author/works_written	dbo:author	backward	dbo:WrittenWork

Table 4: Examples of predicate mapping rules.

5 Question Answering Baseline

To lay the foundation for future work on our new dataset, we provide simple yet strong baselines using recent work by Mohammed et al. (2018), who applied techniques with and without neural networks to SIMPLEQUESTIONS. In this paper, we used their open-source code[5] to generate the experimental results reported here. We briefly describe their approach, which decomposes into four tasks:

- **Entity Detection**: Given a question, the task is to identify the topic entity of the question. For this task, we examined bidirectional LSTMs and Conditional Random Fields (CRFs).
- **Entity Linking**: Detected entities (text strings) need to be linked to entities in the knowledge graph (e.g., URI from DBpedia in our case). This is formulated as a string matching problem: Levenshtein Distance is used along with a few heuristics for ranking candidate entities.
- **Predicate Prediction:** Given a question, the task is to identify the predicate being queried. We examined three models: bidirectional GRU, convolutional neural network (CNN), and logistic regression (LR). The first two are standard neural network models; for logistic regression we used as input the average of the word embeddings of each word. BiGRU was selected over BiLSTM based on the experiments of Mohammed et al. (2018), where it was found to be slightly more accurate.
- **Evidence Integration:** With *m* candidate entities and *r* candidate predicates from the previous components, the evidence integration model selects the best (entity, predicate) pair based on the product of each component score as well as a number of heuristics.

One additional detail is necessary to understand our experimental methodology for entity detection. In SIMPLEQUESTIONS, the topic entity is not explicitly tagged in the natural language question at the token level; as a result, SIMPLEDBPEDIAQA does not have token-level annotations either. This presents a problem, as our formulation of entity detection as sequence labeling requires per-token labels. The solution adopted by Mohammed et al. (2018) with SIMPLEQUESTIONS was to "back-project" the entities onto the natural language questions to automatically derive token labels, either ENTITY or NOTENTITY. We performed exactly the same back-projection in this work. If the entity text can be matched exactly in the question, the corresponding tokens are tagged appropriately. If there is no exact match, n-grams are generated from the question (from length one up to the length of the question) and the Levenshtein Distances between these n-grams and the entity text are computed. The n-gram with the highest score is selected and the corresponding tokens are tagged appropriately. We find that 94.1% of questions have exact matches with entity strings.

6 Experiment Results and Error Analysis

We evaluated the quality of our models in the same way as Mohammed et al. (2018): For entity detection, we compute F1 in terms of the entity labels. For both entity linking and predicate prediction, we evaluate recall at N (R@N). For the final end-to-end evaluation, we use accuracy (or equivalently, R@1). For evidence integration, our model considers 20 entity candidates and 5 predicate candidates. All hyperparameters and other settings follow the original paper; we have not specifically fine-tuned parameters for this dataset.

[5] http://buboqa.io/

Entity Linking	R@1	R@5
BiLSTM	78.0	88.2
CRF	75.1	85.4
Predicate Prediction	**R@1**	**R@5**
CNN	89.2	99.1
BiGRU	88.1	99.0
LR	84.2	97.6

(a) Component accuracy on validation set.

Entity Detection	Relation Prediction	Accuracy
BiLSTM	CNN	78.5
BiLSTM	BiGRU	78.2
BiLSTM	LR	75.8
CRF	CNN	76.1
CRF	BiGRU	76.0
CRF	LR	73.5

(b) End-to-end accuracy on test set.

Figure 5: Experiment results applying the models of Mohammed et al. (2018) to SIMPLEDBPEDIAQA.

Error Type	# Errors	Prevalence
Hard ambiguity	42	21.0%
Soft ambiguity	21	10.5%
Entity detection error	19	9.5%
Predicate prediction error	28	14.0%
Error in both	90	45.0%
Total	200	100.0%

Figure 6: Results of error analysis.

For entity detection, on the validation set, the BiLSTM achieves 90.3 F1, compared to the CRF at 88.1. The top of Table 5a shows the entity linking results for the BiLSTM and the CRF. These results are consistent with the findings of Mohammed et al. (2018): the BiLSTM achieves a higher F1 score than the CRF, which translates into higher recall in entity linking (both R@1 and R@5). Predicate prediction results are shown on the bottom of Table 5a: the CNN slightly outperforms the BiGRU on R@1, but in terms of R@5 the accuracy of both are quite similar. The neural network models appear to be more effective than logistic regression.

Finally, Table 5b shows end-to-end accuracy on the test set. The best model combination uses the BiLSTM for entity detection and the CNN for predicate prediction, achieving 78.5% accuracy. By swapping the BiLSTM with the CRF for entity detection, we observe a 2.4% absolute decrease in end-to-end accuracy. Results from other combinations are also shown in Table 5b. Note that using the CRF for entity detection and logistic regression (LR) for predicate prediction, which is a baseline that does not use neural networks (with the exception of word embeddings), is also reasonably accurate. This finding is also consistent with Mohammed et al. (2018), who advocate that NLP researchers examine baselines that do not involve neural networks as a sort of "sanity check".

Following Lukovnikov et al. (2017), we sampled 200 examples of errors on the test set from the Bi-LSTM + CNN model to analyze their causes. We manually classified them into the following categories, summarized in Table 6 and described below:

- *Hard ambiguity:* The context provided by the question is insufficient, even for a human, to disambiguate between two or more entities with the same name. In these cases, our model correctly identified the entity string, but linked it to an incorrect entity in the knowledge graph. For example, in the question "What is the place of birth of Sam Edwards?", it is unclear if Sam Edwards refers to the actor dbr:Sam_Edwards or the physicist dbr:Sam_Edwards_(physicist).
- *Soft ambiguity:* The context provided by the question is sufficient to disambiguate the entity, but our model fails to identify the correct entity in the knowledge graph. In these cases, our model correctly identified the entity string, so the error is isolated to the entity linking component. For example, in the question "What kind of show is All In?", the model predicted dbr:All_In_(song) instead of dbr:All_In_(TV_series) (the correct entity). Note that in this case, it is clear to a human based on context that the question refers to a show and not a song.

- *Entity detection error:* The extracted entity string is incorrect.
- *Predicate prediction error:* The predicted predicate is incorrect.
- *Error in both:* Both the extracted entity and the predicted predicate are incorrect.

From the above analysis, we find that there is still substantial room to improve on the effectiveness of our baselines. However, these results also suggest that there is an upper bound on accuracy that lies substantially below 100%, as the cases of hard ambiguity are difficult to resolve, even for humans. In those cases, correct entity linking is more a matter of luck and other idiosyncratic characteristics of the dataset rather than signals that can be reliably extracted to understand the true question intent.

7 Conclusion

This paper presents SIMPLEDBPEDIAQA, a new benchmark dataset for simple question answering over knowledge graphs created by migrating the SIMPLEQUESTIONS dataset from Freebase to DBpedia. Although this mapping process is conceptually straightforward, there are a number of nuances and complexities we had to overcome with a combination of special-case handling and heuristics. The result is a dataset targeting a knowledge graph that is actively maintained by a dedicated community. We hope that our efforts better connect existing research communities, in particular, NLP researchers with the open linked data community, and spur additional work in question answering over knowledge graphs.

Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question–answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pages 1533–1544, Seattle, Washington.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. DBpedia — A crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1249, Vancouver, British Columbia, Canada.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv:1506.02075v1*.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 423–433, Sofia, Bulgaria.

Zihang Dai, Lei Li, and Wei Xu. 2016. CFO: Conditional focused neural question answering with large-scale knowledge bases. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 800–810, Berlin, Germany.

Dennis Diefenbach, Thomas Pellissier Tanon, Kamal Deep Singh, and Pierre Maret. 2017. Question answering benchmarks for Wikidata. In *Proceedings of the 2017 International Semantic Web Conference (Posters, Demos & Industry Tracks)*.

Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. 2017. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1211–1220, Perth, Australia.

Salman Mohammed, Peng Shi, and Jimmy Lin. 2018. Strong baselines for simple question answering over knowledge graphs with and without neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 291–296, New Orleans, Louisiana.

Kuldeep Singh, Arun Sethupat Radhakrishna, Andreas Both, Saeedeh Shekarpour, Ioanna Lytra, Ricardo Usbeck, Akhilesh Vyas, Akmal Khikmatullaev, Dharmen Punjani, Christoph Lange, Maria Esther Vidal, Jens Lehmann, and Sören Auer. 2018. Why reinvent the wheel: Let's build question answering systems together. In *Proceedings of the 2018 World Wide Web Conference*, pages 1247–1256, Lyon, France.

Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. 2017. LC-QuAD: A corpus for complex question answering over knowledge graphs. In *Proceedings of the 16th International Semantic Web Conference*, pages 210–218, Vienna, Austria.

Ferhan Ture and Oliver Jojic. 2017. No need to pay attention: Simple recurrent neural networks work! (for answering "simple" questions). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 2856–2862, Copenhagen, Denmark.

Xuchen Yao. 2015. Lean question answering over Freebase from scratch. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations (NAACL/HLT 2015)*, pages 66–70, Denver, Colorado.

An Analysis of Annotated Corpora for Emotion Classification in Text

Laura-Ana-Maria Bostan and **Roman Klinger**
Institut für Maschinelle Sprachverarbeitung
University of Stuttgart, Pfaffenwaldring 5b, 70569 Stuttgart, Germany
laura.bostan@ims.uni-stuttgart.de
roman.klinger@ims.uni-stuttgart.de

Abstract

Several datasets have been annotated and published for classification of emotions. They differ in several ways: (1) the use of different annotation schemata (*e. g.*, discrete label sets, including *joy*, *anger*, *fear*, or *sadness* or continuous values including *valence*, or *arousal*), (2) the domain, and, (3) the file formats. This leads to several research gaps: supervised models often only use a limited set of available resources. Additionally, no previous work has compared emotion corpora in a systematic manner. We aim at contributing to this situation with a survey of the datasets, and aggregate them in a common file format with a common annotation schema. Based on this aggregation, we perform the first cross-corpus classification experiments in the spirit of future research enabled by this paper, in order to gain insight and a better understanding of differences of models inferred from the data. This work also simplifies the choice of the most appropriate resources for developing a model for a novel domain. One result from our analysis is that a subset of corpora is better classified with models trained on a different corpus. For none of the corpora, training on all data altogether is better than using a subselection of the resources. Our unified corpus is available at http://www.ims.uni-stuttgart.de/data/unifyemotion.

Title and Abstract in German

Eine Analyse von annotierten Korpora zur Emotionsklassifizierung in Text

Es existieren bereits verschiedene Textkorpora, welche zur Erstellung von Modellen für die automatische Emotionsklassifikation erstellt wurden. Sie unterscheiden sich (1) in den unterschiedlichen Annotationsschemata (z.B. diskrete Klassen wie *Freude*, *Wut*, *Angst*, *Trauer* oder kontinuierliche Werte wie *Valenz* und *Aktivierung*), (2) in der Domäne, und, auf einer technischen Ebene, (3) in den Dateiformaten. Dies führt dazu, dass überwacht erstellte Modelle typischerweise nur einen Teil der verfügbaren Ressourcen nutzen sowie kein systematischer Vergleich der Korpora existiert. Hier setzt unsere Arbeit mit einem Überblick der verfügbaren Datensätze an, welche wir in ein gemeinsames Format mit einem einheitlichen Annotationsschema konvertieren. Darauf aufbauend führen wir erste Experimente durch, in dem wir auf Teilmengen der Korpora trainieren und auf anderen testen. Dies steht im Sinne zukünftiger, durch unsere Arbeit ermöglichten Analysen, die Unterschiede zwischen den Annotationen besser zu verstehen. Des Weiteren vereinfacht dies die Wahl einer angemessenen Ressource für die Erstellung von Modellen für eine neue Domäne. Wir zeigen unter anderem, dass die Vorhersagen für einige Korpora besser funktioniert, wenn ein Modell auf einer anderen Ressource trainiert wird. Weiterhin ist für kein Korpus die Vorhersage am besten, wenn alle Daten vereint werden. Unser aggregiertes Korpus ist verfügbar unter http://www.ims.uni-stuttgart.de/data/unifyemotion.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2104–2119
Santa Fe, New Mexico, USA, August 20-26, 2018.

1 Introduction

Emotion detection and classification in text focuses on mapping words, sentences, and documents to a set of emotions following a psychological model such as those proposed by Ekman (1992), Plutchik (1980) or Russell (1980), *inter alia*. The task has emerged from a purely research oriented topic to playing a role in a variety of applications, which include dialog systems (chatbots, tutoring systems), intelligent agents, clinical diagnoses of mental disorders (Calvo et al., 2017), or social media mining.

As the variety of applications is large, the set of domains and differences in text is large. An early work, motivated by the goal to develop an empathic storyteller for children stories, is the corpus creation and modelling of emotions in tales by Alm et al. (2005). Afterwards, the idea has been transferred to the Web, namely blogs (Aman and Szpakowicz, 2007), and microblogs (Schuff et al., 2017; Mohammad, 2012; Wang et al., 2012). A different domain under consideration are news articles: Strapparava and Mihalcea (2007) focus on emotions in headlines. It can be doubted that emotions are expressed in a comparable way in these different domains: Journalists ideally tend to be objective when writing articles, authors of microblog posts need to focus on brevity, and one might assume that emotion expressions in tales are more subtle and implicit than, for instance, in blogs. Therefore, the transfer across emotion recognition models is, presumably, challenging. The most straight-forward alternative is, however, to build resources from scratch, a costly process. Given this situation, it remains unclear, given a novel domain for which an emotion recognition system should be developed, how to start. Next to the methodological issues we just discussed, another challenge is to acquire and compare available corpora.

With this paper, we aim at contributing to all these challenges. We support future research by comparing existing datasets, exploring how they complement each other, and mapping them to a common format such that future emotion detection can benefit from being developed on different domains and annotation schemata. In addition, we perform cross-corpus experiments by training classifiers on each dataset and evaluating them on others. One challenge here is that corpora are from different domains and are annotated following different guidelines and schemata. We aim at helping to make decisions which support the best model development for a future domain by selecting appropriate corpora. Parameters to be taken into account are the source of the text (*e. g.*, blogs, news, social media), the annotation schema (*e. g.*, Plutchik, Ekman, subsets of them), and the annotation procedure (*e. g.* crowdsourcing, self-reporting, expert-based, distant labeling). Our main contributions are therefore (1) to describe existing resources, (2) to evaluate which state-of-the-art classifiers perform well when trained on one dataset and applied on another, (3) to evaluate which datasets generalize best to other domains, and (4) to compare the datasets qualitatively and quantitatively. To achieve our goals and as additional support for future research, we unify all available datasets in a common file format. We provide a script that downloads and converts the datasets, and instructs on how to obtain datasets where the license requires no redistributions. Our resources are available via `http://www.ims.uni-stuttgart.de/data/unifyemotion`. We aim at keeping the unified corpus up to date in the future.

2 Background & Related work

In the following, we discuss differences in psychological models and annotation schemata (Section 2.1), annotation procedures (Section 2.2), different domains and topics (Section 2.3), and different prediction methods (Section 2.4). An overview on the resources and previous work is shown in Table 1. In addition to this, we recommend the surveys by Munezero et al. (2014) and Santos and Maia (2018).

2.1 Emotion Models in Psychology & Annotation Schemata

Emotion models are still debated in psychology (Barrett et al., 2018; Cowen and Keltner, 2018). We do not contribute to these debates but focus on the main theories in psychology and natural language processing (NLP): discrete and finite sets of emotions (categorical models) and combinations of different continuous dimensions (dimensional models).

Early work on emotion detection (Alm et al., 2005; Strapparava and Mihalcea, 2007) focused on conceptualizing emotions by following Ekman's model which assumes the following six basic emotions: *anger, disgust, fear, joy, sadness* and *surprise* (Ekman, 1992). Suttles and Ide (2013), Meo and Sulis

(2017), and Abdul-Mageed and Ungar (2017) follow the *Wheel of Emotion* (Plutchik, 1980; Plutchik, 2001) which also considers emotions as a discrete set of eight basic emotions in four opposing pairs: *joy–sadness*, *anger–fear*, *trust–disgust*, and *anticipation–surprise*, together with emotion mixtures.

Dimensional models were more recently adopted in NLP (Preoţiuc-Pietro et al., 2016; Buechel and Hahn, 2017a; Buechel and Hahn, 2017b): The circumplex model (Russell and Mehrabian, 1977) puts affective states into a vector space of valence (corresponding to sentiment/polarity), arousal (corresponding to a degree of calmness or excitement), and dominance (perceived degree of control over a given situation). Any emotion is a linear combination of these.

2.2 Annotation Procedures

A standard way to create annotated datasets is via *expert annotation* (Aman and Szpakowicz, 2007; Strapparava and Mihalcea, 2007; Ghazi et al., 2015; Li et al., 2017; Schuff et al., 2017; Li et al., 2017). However, having an expert annotate a statement means that they must estimate the private state of the author. Therefore, the creators of the ISEAR dataset follow a different, but similar, route making use of *self reporting*: subjects are asked to describe situations associated with a specific emotion (Scherer and Wallbott, 1994). This approach can be considered an annotation by experts in their own right.

Crowdsourcing, for instance using the platforms Amazon's Mechanical Turk[1] or CrowdFlower[2], is another way to acquire human judgments. Crowdsourcing often lacks sufficient quality control but some popular datasets have been successfully acquired with this approach, *e. g.*, the dataset released by Crowdflower for Cortana[3] or the datasets constructed by Milnea et al. (2015) and Lapitan et al. (2016). Another example is the dataset by Mohammad (2012), who design two detailed online questionnaires and annotate tweets by crowdsourcing.

Lastly, social networks play a central role in data acquisition with *distant supervision* (also called *self-labeling* in this context), because they provide a quick and cheap way to get large amounts of noisy data annotated by writers or readers (Mohammad and Kiritchenko, 2015; Abdul-Mageed and Ungar, 2017; De Choudhury et al., 2012; Liu et al., 2017). For example, on Twitter one could add a "#joy" hashtag to a happy tweet or on Facebook one could tag personal posts with a "feeling" and people can show an emotional "surprised reaction". In this last example, two levels of annotation are provided that are relevant to emotion analysis, namely both the reader's and the writer's emotional state. Accessing this information is comparably straight-forward in these social network platforms. More challenging is to acquire such data for other domains. Buechel and Hahn (2017a) and Buechel and Hahn (2017b) look specifically into distinguishing between writers' and readers' emotion expressions.

It should be noted that some of these approaches exist in parallel to previous research in assessing emotion states of people, despite the fact that standardized psychological instruments exist (Bradley and Lang, 1994).

2.3 Domains and Topics

Previous work on emotion detection focuses on different domains and topics, *e. g.*, descriptions of *self reported emotional events* (Scherer and Wallbott, 1994), *news* (Lei et al., 2014; Buechel et al., 2016), *news headlines* (Strapparava and Mihalcea, 2007), *blogs* (Aman and Szpakowicz, 2007), *tales* (Alm et al., 2005), *micro-blog posts* (*i. e.*, tweets) (Wang et al., 2012) to different domains, such as *health*, *politics* (Mohammad, 2012), and *stock markets* (Bollen et al., 2011).

An early example and one of the first initiatives of emotion classification is the work by Aman and Szpakowicz (2007), who use *blog posts*, sampled without taking a specific topic into account. They identify the emotion, category, intensity and cue words and phrases. Mishne and de Rijke (2005), Balog et al. (2006) and Nguyen et al. (2014) works on LiveJournal[4] data to develop predictive models for moods.

Similarly, *user-generated data in social media* has been a subject of research. Mohammad et al. (2015) and Mohammad and Bravo-Marquez (2017b) annotate electoral *tweets* for sentiment, intensity, semantic

[1] https://www.mturk.com/

[2] https://www.crowdflower.com/

[3] https://www.crowdflower.com/data/sentiment-analysis-emotion-text/

[4] https://www.livejournal.com

roles, style, purpose and emotions. De Choudhury et al. (2012) identify more than 200 moods frequent on Twitter. Mohammad (2012), Mohammad et al. (2015), Wang et al. (2012), Volkova and Bachrach (2016) make use of Twitter distantly labeled data. Recently, Liu et al. (2017) analyzed the role of context that grounds sentiment in *tweets*, and looked into whether the effect of weather and news events relate to the emotion expressed in a given tweet. EmoNet is claimed to be the largest dataset constructed of *tweets* (Abdul-Mageed and Ungar, 2017) .

Twitter is often the preferred subject of research as it is easy to use and has a well-documented API. However, Facebook is also used, *e. g.*, Preoţiuc-Pietro et al. (2016) create a dataset of *Facebook posts* and train prediction models for valence and arousal. Pool and Nissim (2016) and Krebs et al. (2017) make use of the reaction feature in Facebook to collect labeled data for distant supervision of a classifier. A different approach within the same domain was used by Polignano et al. (2017) who labeled posts with emoticons mapped to Ekman's model.

Data in social media can be in the form of *dialogues*. Li et al. (2017) manually label a dataset of conversations. Wang et al. (2016) introduce EmotionPush, a system that automatically conveys the emotion of received text on mobile devices, deployed on Facebook's messenger app. From the same domain, but on a different topic is the study of patients' emotional states dynamics expressed by their Facebook posts (Lombardo et al., 2017).

Motivated by the work of literary scholars is the creation of datasets to study emotion in *literature*. One of the first datasets is the tales corpus by Alm et al. (2005). Kim et al. (2017) investigate the relationship between literary genres and emotions.

2.4 Methods used in Emotion Identification

Emotion classification is commonly phrased as text classification. As text classification in general, the array of methods seen for emotion classification can be divided into rule-based methods and machine learning, which we discuss in the following.

2.4.1 Rule-based Algorithms

Rule-based text classification typically builds on top of lexical resources of emotionally charged words. These dictionaries can originate from crowdsourcing or expert curation. Examples include WordNet-Affect (Strapparava et al., 2004) and SentiWordNet (Esuli and Sebastiani, 2007), both of which stem from expert annotation. Partly built on top of them is the NRC Word-Emotion Association Lexicon (Mohammad et al., 2013), which uses the eight basic emotions (Plutchik, 1980). Warriner et al. (2013) use crowdsourcing to assign values of valence, arousal, and dominance (Russell, 1980).

Another related category of lexical resources which has been used for emotion analysis is *concreteness* and *abstractness* (Köper et al., 2017). Brysbaert et al. (2014) publish a lexicon based on crowdsourcing, where the task was to assign a rating from 1 to 5 of the concreteness of 40,000 words. Similarly, Köper and Schulte im Walde (2016) automatically generate affective norms of abstractness, arousal, imageability, and valence for 350,000 lemmas in German. Lastly, the Linguistic Inquiry and Word Count (LIWC) is a set of 73 lexicons (Pennebaker et al., 2001), built to gather aspects of lexical meaning regarding psychological tasks. Dictionary and rule-based approaches are particularly common in the field of digital humanities due to their transparency and straightforward use.

2.4.2 Machine Learning

A performance improvement over dictionary lookup has been observed with supervised feature-based learning. Common features include word n-grams, character n-grams, word embeddings, affect lexicons, negation, punctuation, emoticons, or hashtags (Mohammad, 2012) . This feature representation is then usually used as input to feed classifiers such as naive Bayes, SVM (Mohammad, 2012), MaxEnt and others to predict the relevant emotion category (Aman and Szpakowicz, 2007; Alm et al., 2005). Similarly to the paradigm shift in sentiment analysis, from feature-based modelling to deep learning, state-of-the-art models for emotion classification are often based on neural networks (Barnes et al., 2017). Schuff et al. (2017) applied models from the classes of CNN, BiLSTM (Schuster and Paliwal, 1997), and LSTM (Hochreiter and Schmidhuber, 1997) and compare them to linear classifiers (SVM and MaxEnt), where the

Dataset	Granularity	Annotation	Size	Topic	Source	Avail.
AffectiveText	headlines	E + V	1,250	news	Strapparava (2007)	D-U
Blogs	sentences	E + ne + me	5,025	blogs	Aman (2007)	R
CrowdFlower	tweets	E + CF	40,000	general	Crowdflower (2016)	D-U
DailyDialogs	dialogues	E	13,118	multiple	Li et al. (2017)	D-RO
Electoral-Tweets	tweets	P	4,058	elections	Mohammad (2015)	D-RO
EmoBank	sentences	V+A+D	10,548	multiple	Buechel (2017a)	CC-by4
EmoInt	tweets	E − DS	7,097	general	Mohammad (2017b)	D-RO
Emotion-Stimulus	sentences	E + shame	2,414	general	Ghazi et al. (2015)	D-U
fb-valence-arousal	faceb. posts	V+A	2,895	questionnaire	Preoţiuc (2016)	D-U
Grounded-Emotions	tweets	HS	2,585	weather/events	Liu et al. (2017)	D-U
ISEAR	descriptions	E + SG	7,665	events	Scherer (1994)	GPLv3
Tales	sentences	E	15,302	fairytales	Alm et al. (2005)	GPLv3
SSEC	tweets	P	4,868	general	Schuff et al. (2017)	D-RO
TEC	tweets	E ±S	21,051	general	Mohammad (2012)	D-RO

Table 1: Selection of resources for emotions analysis. Ann. refers to the following annotation schemata: [E] *Ekman: anger, disgust, fear, joy, sadness, surprise*, [P] *Plutchik: anger, disgust, fear, joy, sadness, surprise, trust, anticipation*, [CF] *enthusiasm, fun, hate, neutral, love, boredom, relief, empty*, [DS] *disgust, surprise*, [JS] happy, sad, [V] *valence*, [A] *arousal*, [D] *dominance*, [SG] *shame, guilt*, [±S] *positive surprise, negative surprise*, [ne] *no emotion* [me] *mixed emotion* and Availability refers to the following [D-RO] *Available to download, research only*, [D-U] *Available to download, unknown licensing*, [R] *Available upon request*, [GPLv3] *GNU Public License version 3*, [CC-by 4] *Creative Commons Attribution version 4.0*

BiLSTM show best results with the most balanced precision and recall. Abdul-Mageed and Ungar (2017) claim the highest F_1 following Plutchik's emotion model with gated recurrent unit networks (Chung et al., 2015).

One approach to tackle sparsity of datasets is transfer learning; to make use of similar resources and then transfer the model to the actual task. A recent successful example for this procedure is Felbo et al. (2017) who present a neural network model trained on emoticons which is then transfered to different downstream tasks, namely the prediction of sentiment, sarcasm, and emotions.

3 Unified Dataset of Emotion Annotations

In this section, we describe each dataset we aggregate in our unified corpus. We provide a brief description and then show how the different schemata are merged. Please note that our interpretation might differ from the author's original description (though we aimed at avoiding that).

3.1 Datasets Overview

AffectiveText The dataset AffectiveText published by Strapparava and Mihalcea (2007) is built on news headlines and consists of 1,250 instances. The main goal of this resource is the classification of emotions and valence in news headlines. The annotation schema follows Ekman's basic emotions, complemented by valence. It is multi-label annotated via expert annotation and can be freely downloaded, the license is not specified. The emotion categories are assigned a score from 0 to 100. Training/developing data amounts to 250 annotated headlines, while systems are evaluated on another 1,000 instances.

Blogs This dataset, published by Aman and Szpakowicz (2007) consists of 5,205 sentences from 173 blogs. Instances are annotated with one emotion label each, emotion intensity and emotion indicators. The annotation schema for the emotion category corresponds to the six fundamental Ekman emotions to which *no emotion* is added. This resource can be obtained through contacting the authors.

CrowdFlower The dataset "The Emotion in Text, published by CrowdFlower" consists of 39,740 tweets. Part of this data has been used in Microsoft's Cortana Intelligence Gallery. The set of labels is non-standard (see Table 4). It is annotated via crowdsourcing with one label per tweet and can be freely downloaded, the license is not specified. The data is comparably noisy.

DailyDialogs The dataset, published by Li et al. (2017), is built on conversations and consists of 13,118 sentences. The annotation schema follows Ekman, complemented by "no emotion". It is single label annotated via expert annotation and can be freely downloaded for research purposes. This dataset has additional annotations for communication intention and topic.

Electoral-Tweets The dataset, published by Mohammad et al. (2015), targets the domain of elections. It consists of over 100,000 responses to two detailed online questionnaires (the questions targeted emotions, purpose, and style in electoral tweets). The tweets are annotated via crowdsourcing. The set of labels for emotion is non-standard (see Table 1). In addition to document-level annotations, tweets are annotated with emotion words. It can be freely downloaded for research purposes.

EmoBank The dataset published by Buechel and Hahn (2017a) builds on multiple genres and domains. It consists of 10,548 sentences where each sentence was manually annotated according to both the emotion which is expressed by the writer, and the emotion which is perceived by the readers. The annotations are according to the valence-arousal-dominance model. A subset of the corpus is AffectiveText, which makes this dataset a good resource to design models that map between both discrete or dimensional representations.

EmoInt The EmoInt published by Mohammad and Bravo-Marquez (2017a) builds on social media content that amounts to 7,097 tweets altogether. The main goal of this resource is to associate text with various intensities of emotion. The tweets are annotated via crowdsourcing with intensities of *anger*, *joy*, *sadness*, and *fear*, while most tweets are only annotated with one emotion. It can be freely downloaded for research purposes.

Emotion-Stimulus The Emotion-Stimulus dataset published by Ghazi et al. (2015) consists of 820 sentences which are annotated both with emotions and their causes, and 1,549 sentences which are marked only with their emotion. The set of labels used for annotation consists of Ekman's basic emotions to which *shame* is added. The main goal of this resource is to predict the cause of an emotion in the text. It is annotated using FrameNet's emotions-directed frame with one emotion label per sentence. It is available for download for research purposes.

fb-valence-arousal The fb-valence-arousal dataset published by Preoţiuc-Pietro et al. (2016) is built on Facebook posts. It consists of 2,895 posts stratified by age and gender. The main goal of this resource is to train prediction models for *valence* and *arousal*. Each message is written by a distinct user and all messages are from the same time interval. The posts are annotated with valence and arousal on a nine point scale via expert annotation. It is available for download.

Grounded-Emotions The dataset published by Liu et al. (2017) is built on social media and consists of 2,557 single labeled instances published by 1,369 unique users. The main goal of this resource is to put emotions into context of other factors including weather, news events, social network, user predisposition, and timing. The set of labels is *happy* and *sad*. The tweets are annotated by the authors. The information is used in experiments aiming at showing the role played by external context in predicting emotions.

ISEAR The "International Survey on Emotion Antecedents and Reactions" dataset published by Scherer and Wallbott (1994) is built by collecting questionnaires answered by people with different cultural backgrounds. These people report on their own emotional events. The final dataset contains reports by approximately 3,000 respondents, for a total of 7,665 sentences labeled with single emotions. The labels are *joy, fear, anger, sadness, disgust, shame*, and *guilt*. It is available for download.

SSEC The Stance Sentiment Emotion Corpus published by Schuff et al. (2017) is an annotation of the SemEval 2016 Twitter stance and sentiment dataset (Mohammad et al., 2017). It consists of 4,868 tweets. The main goal of this resource is to enable further research on the relations between emotions and other factors. It is annotated via expert annotation with multiple emotion labels per tweet following Plutchik's fundamental emotions. An additional feature of this resource is that they not only provide a majority annotation but publish the individual information for all annotators.

Tales The Tales corpus published by Alm et al. (2005) is built on literature and consists of 15,302 sentences from 185 fairytales by B. Potter, H.C. Andersen and the brothers Grimm. Out of these 15,302 sentence, all annotators agree only on 1,280. The main goal of this resource is to build

	joy		anger		sadness		disgust		fear		surprise	
Dataset	o	m	o	m	o	m	o	m	o	m	o	m
AffectiveText	619	619	374	374	650	650	253	253	537	537	787	787
Blogs	848	536	884	179	883	173	882	172	861	115	847	115
CrowdFlower	5,209	9,220	110	1,421	5,165	5,123	—	179	8,459	8,430	2,187	2,177
DailyDialogs	12,885	12,885	1,022	1,022	1,150	1,150	353	353	74	174	1,823	1,823
Electoral-Tweets	267	349	300	569	34	31	1,937	1,638	80	91	259	251
EmoInt	1,616	1,616	1,701	1,701	1,533	1,533	—	—	2,252	2,252	—	–
Emotion-Stimulus	479	479	483	483	575	575	95	95	423	423	213	213
Grounded-Emotions	1,530	1,530	—	—	1,055	1,055	—	—	—	—	—	—
ISEAR	1,094	1,094	1,096	1,096	3,285	3,285	1,096	1,096	1,095	1,095		—
SSEC	2,067	2,067	2,902	2,902	2,644	2,644	2,183	2,183	1,840	1,840	1,108	1,108
Tales	107	107	195	195	116	116	14	14	111	111	90	90
TEC	8,240	8,237	1,555	1,555	3,830	3,830	761	761	2,816	2,816	3,849	3,849
Total	34,961	38,739	10,622	11,497	21,920	20,165	7,574	6,744	16,708	17,884	11,162	10,413

Table 2: The distribution of categories across the datasets, limited to *anger*, *disgust*, *fear*, *joy*, *sadness* and *surprise*. Non-availability of a class in a set is marked with —. [o]: original distributions, without taking high agreement annotations; [m]: counts after mapping to our labels (see Table 4).

emotion classifiers for literature. The annotation schema consists of Ekman's six basic emotions. In the final data the labels *angry* and *disgust* are merged. It can be freely downloaded for research purposes.

TEC The Twitter Emotion Corpus published by Mohammad (2012) is built on social media. It consists of 21,051 tweets. The main goal of this resource is answering the question if emotion-word hashtags can successfully be used as emotion labels. The annotation schema corresponds to Ekman's model of basic emotions. They collected tweets with hashtags corresponding to the six Ekman emotions: *#anger*, *#disgust*, *#fear*, *#happy*, *#sadness*, and *#surprise*, therefore it is distantly single-label annotated. It can be freely downloaded for research purposes.

3.2 Analysis

The majority of resources consists of user generated data. The biggest dataset is CrowdFlower with 39,740 annotated tweets, followed by TEC with 21,051 tweets, and Blogs with 15,000 annotated sentences from blog posts. Out of all 14 resources, 9 are annotated with the six fundamental emotions defined by Ekman, with small variations. SSEC and Electoral-Tweets follow Plutchik's model. Electoral-Tweets is annotated with 19 emotions and the authors provide a mapping to Plutchik's set (which we follow in the aggregation). Non-fundamental emotions are annotated in CrowdFlower (*fun, worry, enthusiasm, and love*).

Not only the size, also the distribution of labels is different in the corpora. Table 2 shows the distribution for Ekman's emotions before and after having applied the mapping to a unique set of emotion labels (see Table 4 in the Appendix A). In many corpora, *joy* is the dominating emotion, followed by *sadness*, *surprise*, and *anger*. Exceptions are SSEC, Electoral-Tweets, and EmoInt, in which negative emotions are more frequent. In SSEC, this is because of its origin as a stance dataset. Similarly, Electoral-Tweets shows a polarizing nature of political debates with *disgust* and *anger* being more common.

Figure 1 shows a quantitative similarity comparison of the data. We represent each dataset by its term distribution, taking the top 5,000 most common words from each dataset and calculating the cosine similarity across corpora (inspired by Ruder and Plank (2017) and Plank and Van Noord (2011)). Twitter corpora are more similar to each other (EmoInt and CrowdFlower are the most similar to TEC) than to other domains with the exception of SSEC, which is the most dissimilar to the other tweet datasets. DailyDialogs is more similar to the tweets than to ISEAR and Blogs.

The column *All* stands for the union of all datasets except the one that is being compared to. In this context, the most dissimilar towards the respective aggregated set is AffectiveText. The reason is that this is a small dataset compared to the tweet-based corpora and that it covers a specific topic, *headlines*. Grounded-Emotions is also notably dissimilar. Most similar to *All* is EmoInt, followed closely by TEC and Blogs, which covers blog posts and not tweets.

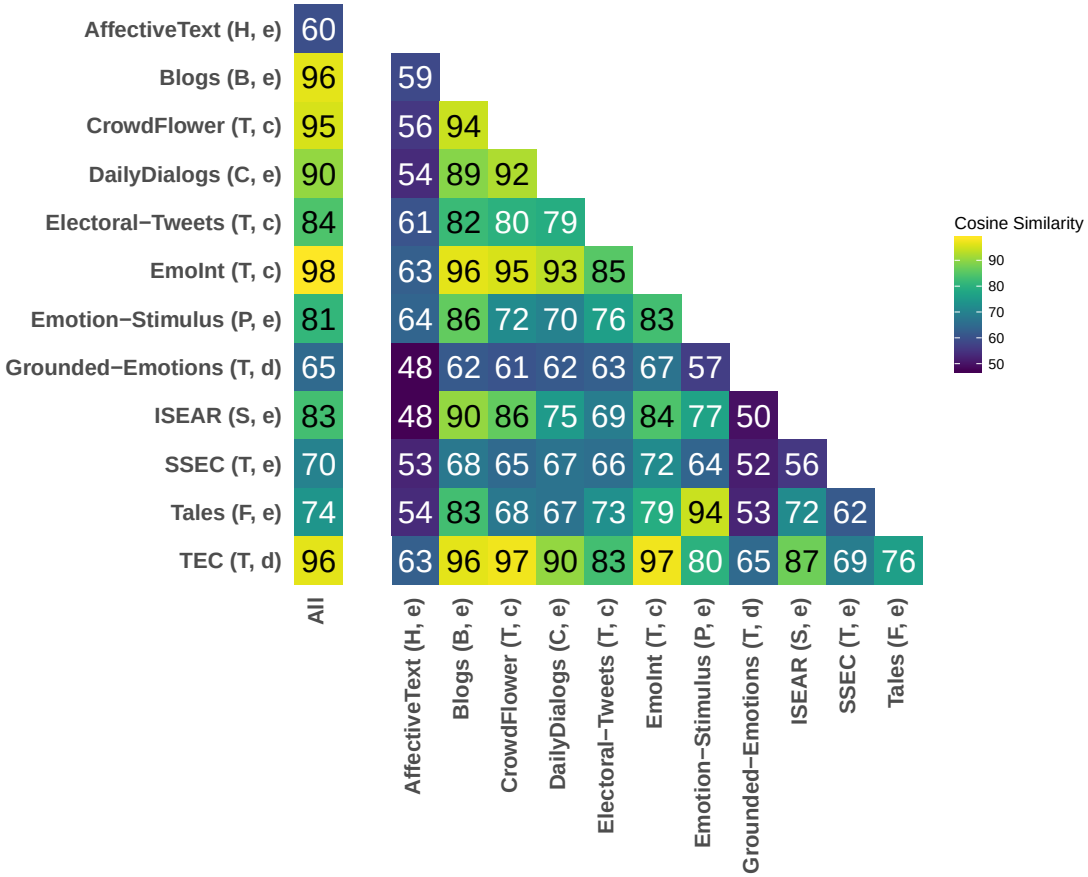


Figure 1: Similarity of corpora with cosine measure. T: tweets, F: tales, P: paragraphs, S: descriptions, H: headlines, B: blogpost, C: conversations; e: expert annotation, d: distant supervision, c: crowdsourcing.

3.3 Aggregation

To provide a standardized access to the datasets, we define *joy, anger, sadness, disgust, fear, trust, surprise, love, confusion, anticipation* and *noemo* as our common label set. The original resources additionally include other 44 labels that come from Electoral-Tweets and CrowdFlower. Where availble from the original publication, we follow proposed mappings (*e. g.*, Electoral-Tweets with 19 emotions and a mapping to Plutchik's model). Table 4 in Appendix A summarizes the mapping. We include *valence, arousal*, and *dominance* where annotated, however, we currently do not map the categorical emotion models onto the dimensional ones.

Each instance in the unified dataset contains, in addition, a unique id, the source corpus name, the text, and an assignment of a real number to each of the 11 emotion variables. In most datasets, each instance is only annotated discretely with single labels (exceptions are SSEC and AffectiveText). Therefore, most instances have exactly one label marked with a 1.0. For the multi-labeled datasets, several emotions can be marked. For datasets with annotated emotion intensity, values can range between 0 and 1. For datasets with multiple annotator information, we follow the recommendations by the original authors. For SSEC, that means, accepting a label if at least one annotator assigned it. For Tales, the authors provide a gold annotation, in which *angry* and *disgust* are merged. We handle them separately, and accept a label if and only if all annotators agree. For Blogs, we take the examples of the dataset with the high agreement annotations.

Next to these fundamental attributes, we provide the domain and annotation style information, as well as additional, dataset specific attributes (*e. g.*, the story from which an instance originates, in the Tales corpus). Our unified data includes all datasets for which the licenses are available; as some datasets are not redistributable, but free to download, we provide a script that interactively downloads and converts the existing resources into our unified format. An excerpt of the data is depicted in Appendix B.

4 Experiments

We perform experiments in the following settings: Firstly, we perform a within-corpus emotion classification (training on one corpus and testing on the same, using cross-validation[5]). Secondly, we do pairwise-corpus evaluation: training on the entire data of one corpus and evaluating on all the data of a different one, for all corpus pairs. This includes the use of the aggregated corpus, but for this experiment excluding the test corpus – this corresponds to a cross validation setting in which the subsets correspond to corpora.

4.1 Experimental settings

Previous methods have shown that linear classifiers are nearly *en par* with neural methods (Schuff et al., 2017). We therefore use maximum entropy classifiers as implemented in scikit-learn (Pedregosa et al., 2011) with bag-of-words (BOW) features for these experiments for simplicity and easy reproducibility. We use L2 regularization and balance the classes in the training data with instance weights. Training/test splits are 80%/20%. Cross-validation is 10-folds stratified.

For datasets in which the labels do not align following our mapping, we use the intersection of labels in the train and test data. We do not discard any instances. For datasets that are designed for other tasks than emotion classification such as EmoInt and Emotion-Stimulus, we do not change the setting of our classification task.

For experiments in which we move from a *multi-label setting* (more than one emotion can hold per instance) to a *single-label setting* (only one emotion holds), we train multiple binary classifiers from which we only accept, in the prediction phase, the highest scoring emotion. For experiments in which we move from a *single-label setting* to a *multi-label setting*, we as well train separate binary classifiers and accept all emotions for which the binary classifiers output one. The case *multi-label* to *multi-label* works analogously. For *single-label* to *single-label*, we use one multi-class MaxEnt model.

4.2 Results

Within-corpus emotion classification. The results for our first experiment are shown per emotion in Table 3 (where we restrict the results to the six fundamental emotions defined by Ekman) and in the diagonal of Figure 2. These should be interpreted in context to the similarity analysis in Figure 1. We see that some datasets and domains are more difficult to be modeled than others. The "easiest" dataset seems to be Emotion-Stimulus, followed by EmoInt. The reason for the high scores lies in the fact that both datasets are constructed for different tasks (stimulus and intensity prediction). As such, our task does not suit these two very well.

Next datasets with comparably high performance measures are Blogs and DailyDialogs. In contrast, CrowdFlower and Electoral-Tweets seem to be the most challenging in the within-corpus setting. For CrowdFlower, the results are due to the larger label set, which makes the task more difficult. Mostly, the emotions that occur less frequently (like surprise) show lower results than the ones occurring frequently (like *joy* and *sadness*). In addition, manual inspection shows that this data is comparably noisy. This is also backed by the general observation that our model performs in general worse on Twitter data than on most other domains.

In terms of annotation procedures, these experiments allow almost for no judgement, since most of the datasets use expert annotation and we only have few examples for the other two ways of annotation (crowdsourcing and distant supervision) being used. However, we could observe that the crowdsourced datasets are more difficult which might be due to a more noisy annotation.

Cross-corpus emotion classification The in-corpus results (the diagonal in Figure 2) shows higher F_1-scores than the cross-corpus results. The exception is Electoral-Tweets, where the same performance is observed by training on a different corpus, Blogs. Models trained on Twitter data perform slightly

[5]Some datasets came with designated train and test parts, but in order to treat all datasets equal, we chose to ignore that. However, in the aggregated dataset, the information about which part of the corpus an example is from is preserved in a special field.

	Joy			Sadness			Fear			Anger			Disgust			Surprise		
Dataset	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
AffectiveText	68	64	66	72	64	68	85	55	67	37	76	50	44	55	49	69	74	71
Blogs	63	70	67	42	32	36	67	43	53	61	31	41	73	51	60	24	28	25
CrowdFlower	42	35	38	26	28	27	32	30	31	21	29	24	6	23	9	9	9	9
DailyDialogs	43	63	51	13	47	20	10	33	16	12	44	18	8	37	13	16	56	24
Electoral-Tweets	37	23	28	23	35	27	26	42	32	36	36	36	52	35	42	15	25	18
EmoInt	94	92	93	83	81	82	85	90	88	90	86	88	0	0	0	0	0	0
Emotion-Stimulus	98	99	98	95	100	97	99	98	98	99	98	98	100	90	95	97	97	97
Grounded-Emotions	61	53	57	41	48	44	0	0	0	0	0	0	0	0	0	0	0	0
ISEAR	61	74	67	73	66	69	69	69	69	45	48	47	58	61	59	0	0	0
SSEC	67	67	67	59	89	71	45	72	55	79	74	76	64	62	63	44	55	49
Tales	34	35	34	24	30	26	27	28	27	28	34	31	16	21	18	19	28	23
TEC	67	69	68	46	44	45	58	56	57	41	40	40	27	26	26	54	50	52

Table 3: Results obtained via 10-fold crossvalidation in precision, recall, and F_1 micro-averaged of the MaxEnt classifier with BOW as features, reported per Ekman's fundamental emotion. Zeros denote that the respective class is not annotated in the respective data set. Note that a subset of datasets has more classes annotated than the provided Ekman's emotions.

better on other Twitter sets, with an exception of Electoral-Tweets, for which the distribution of labels is different, with *disgust* dominating the set.

It is notable that EmoInt, Emotion-Stimulus, Grounded-Emotions ISEAR, and SSEC are easier to classify (high performance when used for testing) while DailyDialogs, Blogs, CrowdFlower, and Tales are more informative: training on them and classifying other datasets leads to better results. Models trained on ISEAR and SSEC perform comparably well. DailyDialogs is classified best not by a classifier trained on itself, but by a classifier trained on Blogs.

We cannot recommend to train on Emotion-Stimulus and Grounded-Emotions as long as the specific properties of these datasets are not required. The models estimated on these data do not perform well on other sets. Note that this is not a quality judgement, Grounded-Emotions has different labels and Emotion-Stimulus was designed for a different purpose.

Note that the similarity measure is an approximation of mediocre quality for model performance, with a Pearson correlation of $r = 0.32$.

All vs. one cross-corpus emotion classification. The results for this setting can be seen in the *All* column of Figure 2. These results show which datasets are easier to classify, namely DailyDialogs, Blogs, and EmoInt. It might seem intuitive that adding more and diverse training data could be helpful in classifying almost all datasets. However, we can see in the results that this is not the case. Especially the multi-labeled datasets AffectiveText and SSEC together with the datasets that were the most transformed (*i. e.*, many labels unified) during the aggregation process, such as Electoral-Tweets and CrowdFlower are more difficult to classify while training on all the other datasets.

5 Conclusion & Future Work

Datasets annotated for emotion classification are important in emotion analysis research as they are used in many downstream tasks as well; having these datasets all in place reduces drastically the amount of work needed in preprocessing and transferring the needed data. Yet, it is a diverse collection of datasets driven by different psychological emotion models, on different domains, and approaches used in annotation. With this paper, we present the first survey on emotion datasets on text.

In addition to this literature review, we provide a unified, aggregated corpus to support future research on standardized data. The existence of such a benchmark opens up the possibility of other experiments, such as transfer learning and domain adaptation work, with focus on different domains and on different label sets. From the collected unified datasets one could learn how to select the most suitable dataset for a given new domain and evaluate it across different classification models, domains, and annotation procedures, easier than it was possible until now. Also having this open will help the emotion detection

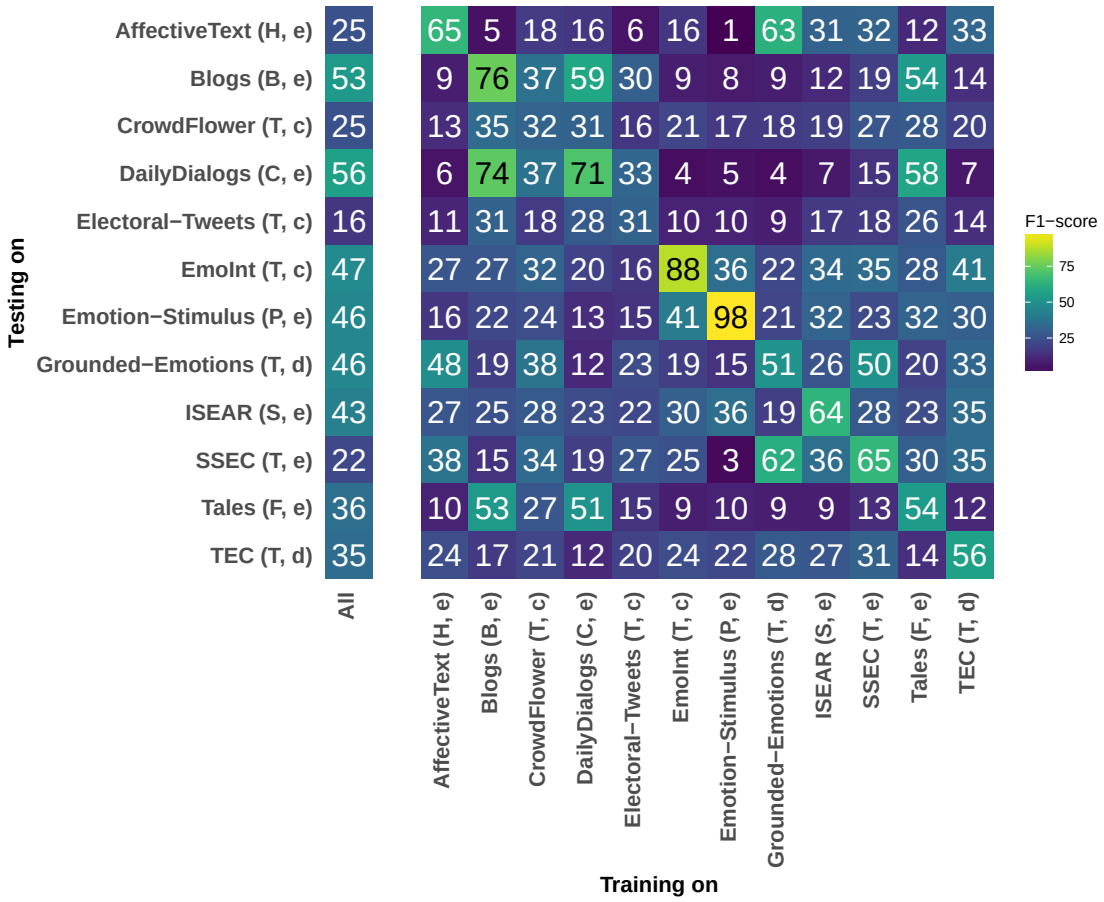


Figure 2: Results of MaxEnt in F_1 measure (micro-averaged) for all cross-corpus experiments. T: tweets, C: conversations, F: tales, P: paragraphs, S: descriptions, H: headlines, B: blogsposts; e: expert annotation, d: distant supervision, c: crowdsourcing.

task to become a standard task on which state-of-the-art methods used in general classification are tested upon, similarly to other tasks like sentiment classification, which plays this role already.

In addition, this work can be used by anyone who wants to explore the current state of the emotion analysis field. As future work we plan to release another version of the dataset in which the conversion between the different emotion models are added and to perform transfer learning experiments between datasets, domains, and annotation procedures. Furthermore, we propose to use the resource to qualitatively analyze the different realizations of emotions across annotation schemata and domains.

Acknowledgements

This research has been funded by the German Research Council (DFG), project SEAT (Structured Multi-Domain Emotion Analysis from Text, KL 2869/1-1). We thank Evgeny Kim and Jeremy Barnes for fruitful discussions.

References

Muhammad Abdul-Mageed and Lyle Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 718–728, Vancouver, Canada, July. Association for Computational Linguistics.

Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. 2005. Emotions from text: Machine learning for text-based emotion prediction. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 579–586, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Saima Aman and Stan Szpakowicz. 2007. Identifying expressions of emotion in text. In Václav Matoušek and Pavel Mautner, editors, *Text, Speech and Dialogue*, pages 196–205, Berlin, Heidelberg. Springer Berlin Heidelberg.

Krisztian Balog, Gilad Mishne, and Maarten De Rijke. 2006. Why are they excited?: identifying and explaining spikes in blog mood levels. In *11th Conference of the European Chapter of the Association for Computational Linguistics: Demostrations*, pages 207–210. Association for Computational Linguistics.

Jeremy Barnes, Roman Klinger, and Sabine Schulte im Walde. 2017. Assessing state-of-the-art sentiment models on state-of-the-art sentiment datasets. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Copenhagen, Denmark. Workshop at Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics.

Lisa Feldman Barrett, Zulqarnain Khan, Jennifer Dy, and Dana Brooks. 2018. Nature of emotion categories: Comment on cowen and keltner. *Trends in Cognitive Sciences*, 22(2):97 – 99.

Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1 – 8.

Margaret M. Bradley and Peter J. Lang. 1994. Measuring emotion: The self-assessment manikin and the semantic differential. *Journal of Behavior Therapy and Experimental Psychiatry*, 25(1):49–59.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known English word lemmas. *Behavior research methods*, 46(3):904–911.

Sven Buechel and Udo Hahn. 2017a. Emobank: Studying the impact of annotation perspective and representation format on dimensional emotion analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 578–585, Valencia, Spain, April. Association for Computational Linguistics.

Sven Buechel and Udo Hahn. 2017b. Readers vs. writers vs. texts: Coping with different perspectives of text understanding in emotion annotation. In *Proceedings of the 11th Linguistic Annotation Workshop*, pages 1–12, Valencia, Spain, April. Association for Computational Linguistics.

Sven Buechel, Udo Hahn, Jan Goldenstein, Sebastian G. M. Händschke, and Peter Walgenbach. 2016. Do enterprises have emotions? In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 147–153, San Diego, California, June. Association for Computational Linguistics.

Rafael A Calvo, David N Milne, M Sazzad Hussain, and Helen Christensen. 2017. Natural language processing in mental health applications using non-clinical texts. *Natural Language Engineering*, 23(5):649–685.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2067–2075, Lille, France, 07–09 Jul. PMLR.

Alan S. Cowen and Dacher Keltner. 2018. Clarifying the conceptualization, dimensionality, and structure of emotion: Response to barrett and colleagues. *Trends in Cognitive Sciences*, 22(4):274–276.

Munmun De Choudhury, Scott Counts, and Michael Gamon. 2012. Not all moods are created equal! exploring human emotional states in social media. In *Sixth international AAAI conference on weblogs and social media*, pages 66–73.

Paul Ekman. 1992. An argument for basic emotions. *Cognition & emotion*, 6(3-4):169–200.

Andrea Esuli and Fabrizio Sebastiani. 2007. Sentiwordnet: a high-coverage lexical resource for opinion mining. *Evaluation*, pages 1–26.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625, Copenhagen, Denmark, September. Association for Computational Linguistics.

Diman Ghazi, Diana Inkpen, and Stan Szpakowicz. 2015. Detecting emotion stimuli in emotion-bearing sentences. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, pages 152–165, Cham. Springer International Publishing.

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

Evgeny Kim, Sebastian Padó, and Roman Klinger. 2017. Investigating the relationship between literary genres and emotional plot development. In *Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 17–26, Vancouver, Canada, August. Association for Computational Linguistics.

Maximilian Köper and Sabine Schulte im Walde. 2016. Automatically generated affective norms of abstractness, arousal, imageability and valence for 350 000 german lemmas. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Maximilian Köper, Evgeny Kim, and Roman Klinger. 2017. IMS at EmoInt-2017: Emotion intensity prediction with affective norms, automatically extended resources and deep learning. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Copenhagen, Denmark. Workshop at Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics.

Florian Krebs, Bruno Lubascher, Tobias Moers, Pieter Schaap, and Gerasimos Spanakis. 2017. Social emotion mining techniques for facebook posts reaction prediction. *arXiv preprint arXiv:1712.03249*.

Fermin Roberto Lapitan, Riza Theresa Batista-Navarro, and Eliezer Albacea. 2016. Crowdsourcing-based annotation of emotions in filipino and english tweets. In *Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016)*, pages 74–82, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Jingsheng Lei, Yanghui Rao, Qing Li, Xiaojun Quan, and Liu Wenyin. 2014. Towards building a social emotion detection system for online news. *Future Generation Computer Systems*, 37:438–448.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. *arXiv preprint arXiv:1710.03957*.

Vicki Liu, Carmen Banea, and Rada Mihalcea. 2017. Grounded emotions. In *2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII)*, pages 477–483, San Antonio, Texas, Oct.

Gianfranco Lombardo, Alberto Ferrari, Paolo Fornacciari, Monica Mordonini, Laura Sani, and Michele Tomaiuolo. 2017. Dynamics of emotions and relations in a facebook group of patients with hidradenitis suppurativa. In *International Conference on Smart Objects and Technologies for Social Good*, pages 269–278. Springer.

Rosa Meo and Emilio Sulis. 2017. Processing Affect in Social Media: A Comparison of Methods to Distinguish Emotions in Tweets. *ACM Transactions on Internet Technology*, 17(1):7:1–7:25.

David Milnea, Cecile Parisb, Helen Christensenc, Philip Batterhamc, and Bridianne ODeac. 2015. We feel: Taking the emotional pulse of the world. In *Proceedings 19th Triennial Congress of the IEA*, volume 9.

Gilad Mishne and Maarten de Rijke. 2005. Boosting web retrieval through query operations. In David E. Losada and Juan M. Fernández-Luna, editors, *Advances in Information Retrieval*, pages 502–516, Berlin, Heidelberg. Springer Berlin Heidelberg.

Saif Mohammad and Felipe Bravo-Marquez. 2017a. Emotion intensities in tweets. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 65–77, Vancouver, Canada, August. Association for Computational Linguistics.

Saif Mohammad and Felipe Bravo-Marquez. 2017b. Wassa-2017 shared task on emotion intensity. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 34–49, Copenhagen, Denmark, September. Association for Computational Linguistics.

Saif M Mohammad and Svetlana Kiritchenko. 2015. Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence*, 31(2):301–326.

Saif Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 321–327, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Saif M Mohammad, Xiaodan Zhu, Svetlana Kiritchenko, and Joel Martin. 2015. Sentiment, emotion, purpose, and style in electoral tweets. *Information Processing & Management*, 51(4):480–499.

Saif M. Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. 2017. Stance and sentiment in tweets. *ACM Trans. Internet Technol.*, 17(3):26:1–26:23, June.

Saif Mohammad. 2012. #emotional tweets. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 246–255, Montréal, Canada, 7-8 June. Association for Computational Linguistics.

Myriam D Munezero, Calkin Suero Montero, Erkki Sutinen, and John Pajunen. 2014. Are they different? affect, feeling, emotion, sentiment, and opinion detection in text. *IEEE transactions on affective computing*, 5(2):101–111.

Thin Nguyen, Dinh Phung, Brett Adams, and Svetha Venkatesh. 2014. Mood sensing from social media texts and its applications. *Knowledge and Information Systems*, 39(3):667–702, June. 00011.

W. Gerrod Parrott. 2000. *Emotions in Social Psychology*. Key Readings in Social Psychology. Psychology Press.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: LIWC 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001):2001.

Barbara Plank and Gertjan Van Noord. 2011. Effective measures of domain similarity for parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1566–1576. Association for Computational Linguistics.

Robert Plutchik. 1980. A general psychoevolutionary theory of emotion. *Theories of emotion*, 1(3-31):4.

Robert Plutchik. 2001. The nature of emotions human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice. *American Scientist*, 89(4):344–350.

Marco Polignano, Marco de Gemmis, Fedelucio Narducci, and Giovanni Semeraro. 2017. Do you feel blue? detection of negative feeling from social media. In Floriana Esposito, Roberto Basili, Stefano Ferilli, and Francesca A. Lisi, editors, *AI*IA 2017 Advances in Artificial Intelligence*, pages 321–333, Cham. Springer International Publishing.

Chris Pool and Malvina Nissim. 2016. Distant supervision for emotion detection using facebook reactions. In *Proceedings of the Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media (PEOPLES)*, pages 30–39. The COLING 2016 Organizing Committee.

Daniel Preoţiuc-Pietro, H. Andrew Schwartz, Gregory Park, Johannes Eichstaedt, Margaret Kern, Lyle Ungar, and Elisabeth Shulman. 2016. Modelling valence and arousal in facebook posts. In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 9–15. Association for Computational Linguistics.

Sebastian Ruder and Barbara Plank. 2017. Learning to select data for transfer learning with bayesian optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 372–382, Copenhagen, Denmark, September. Association for Computational Linguistics.

James A Russell and Albert Mehrabian. 1977. Evidence for a three-factor theory of emotions. *Journal of research in Personality*, 11(3):273–294.

James A Russell. 1980. A circumplex model of affect. *Journal of personality and social psychology*, 39(6):1161.

Diana Santos and Belinda Maia. 2018. Language, emotion, and the emotions: A computational introduction. *Language and Linguistics Compass*, 12(6):e12279.

Klaus R Scherer and Harald G Wallbott. 1994. Evidence for universality and cultural variation of differential emotion response patterning. *Journal of personality and social psychology*, 66(2):310.

Hendrik Schuff, Jeremy Barnes, Julian Mohme, Sebastian Padó, and Roman Klinger. 2017. Annotation, modelling and analysis of fine-grained emotions on a stance and sentiment detection corpus. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Copenhagen, Denmark. Workshop at Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 70–74, Prague, Czech Republic, June. Association for Computational Linguistics.

Carlo Strapparava, Alessandro Valitutti, et al. 2004. Wordnet affect: an affective extension of wordnet. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 1083–1086.

Jared Suttles and Nancy Ide. 2013. Distant Supervision for Emotion Classification with Discrete Binary Values. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, number 7817 in Lecture Notes in Computer Science, pages 121–136. Springer Berlin Heidelberg, March. 00029.

Svitlana Volkova and Yoram Bachrach. 2016. Inferring perceived demographics from user emotional tone and user-environment emotional contrast. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1567–1578.

Wenbo Wang, Lu Chen, Krishnaprasad Thirunarayan, and Amit P. Sheth. 2012. Harnessing twitter "big data" for automatic emotion identification. In *SocialCom/PASSAT*, pages 587–592. IEEE.

Shih-Ming Wang, Chun-Hui Scott Lee, Yu-Chun Lo, Ting-Hao Huang, and Lun-Wei Ku. 2016. Sensing emotions in text messages: An application and deployment study of emotionpush. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 141–145, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior research methods*, 45(4):1191–1207.

A Mapping of Labels to Unified Labels

Unified Label	Original Labels
anger	anger, angry, annoyance, fury, hostility, ag, hate
anticipation	anticipation, vigilence, interest, expectancy
confusion	confusion, indecision
disgust	disgust, dg, dislike, boredom, hate, disappointment, indifference
fear	fear, panic, terror, apprehension, fr, worry
joy	joy, happy, happiness, joyful, elation, hp, fun, enthusiasm, relief, serenity, calmness
love	love
noemo	noemo, neutral, ne, BLANK
sadness	sadness, sad, gloominess, grief, sorrow, sd, shame, guilt
surprise	surprise, uncertainty, amazement , su+, su−
trust	trust, acceptance, admiration, like

Table 4: Mapping of original labels to labels in unified dataset. We roughly follow the models by Plutchik (2001) and Parrott (2000)

B Example Excerpt from the Aggregated Corpus


```
{
  "id": 210599,
  "VAD": {
    "valence": null,
    "arousal": null,
    "dominance": null
  },
  "source": "ssec",
  "text": "He who exalts himself shall be humbled; and he who humbles himself shall
     be exalted. Matt 23:12. #SemST"
  "emotions": {
    "joy": 1,
    "anger": 1,
    "sadness": 0,
    "disgust": 1,
    "fear": 0,
    "trust": 1,
    "surprise": 0,
    "love": null,
    "noemo": null,
    "confusion": null,
    "anticipation": 0
     },
  "original_split": "test",
  "emotion_model": "Plutchik"
  "domain": "social-media/tweets",
  "labeled": "multilabeled"
}
```


Figure 3: Example excerpt from our aggregated and unified corpus.

Investigating the Working of Text Classifiers

Devendra Singh Sachan[♠,△], **Manzil Zaheer**[♠], **Ruslan Salakhutdinov**[♠]
[♠]School of Computer Science, Carnegie Mellon University
[△]Petuum, Inc
Pittsburgh, PA, USA
{dsachan, manzilz, rsalakhu}@andrew.cmu.edu

Abstract

Text classification is one of the most widely studied tasks in natural language processing. Motivated by the principle of compositionality, large multilayer neural network models have been employed for this task in an attempt to effectively utilize the constituent expressions. Almost all of the reported work train large networks using discriminative approaches, which come with a caveat of no proper capacity control, as they tend to latch on to any signal that may not generalize. Using various recent state-of-the-art approaches for text classification, we explore whether these models actually learn to compose the meaning of the sentences or still just focus on some keywords or lexicons for classifying the document. To test our hypothesis, we carefully construct datasets where the training and test splits have no direct overlap of such lexicons, but overall language structure would be similar. We study various text classifiers and observe that there is a big performance drop on these datasets. Finally, we show that even simple models with our proposed regularization techniques, which disincentivize focusing on key lexicons, can substantially improve classification accuracy.

1 Introduction

Text classification is one of the fundamental tasks in natural language processing (NLP) in which the objective is to categorize text documents into one of the predefined classes. This task has a lot of applications such as topic classification of news articles, sentiment analysis of reviews, email filtering, etc. Text classification still remains a significant challenge in language understanding as we need to encode the intrinsic grammatical relations between sentences in the semantic meaning of a document. This is especially crucial for sentiment classification because relations like "contrast" and "cause" can have great influence on determining the meaning and the overall polarity of a document.

Traditionally, for text classification, bag-of-words model (Harris, 1954) was used to represent a document. This simple approach uses the term frequency as features followed by a classifier such as Naïve Bayes (McCallum and Nigam, 1998) or Support Vector Machine (SVM) (Joachims, 1998). One drawback of this approach is that it ignores the word order and grammatical structure. It also suffers from data sparsity problem when the training set's size is small but has shown to give good results when size is not an issue (Wang and Manning, 2012). Before applying these methods, feature selection is typically carried out to reduce the effective vocabulary size by removing the noisy features (Manning et al., 2008). A key property of these linear classifiers is that they assign high weights to some class label specific keywords, which are also known as lexicons.

The next generation of approaches includes neural networks that have shown to outperform bag-of-words models in text classification tasks (Kim, 2014; Johnson and Zhang, 2016). These methods typically use multiple layers of Convolutional Neural Network (CNN) (Lecun et al., 1998) and/or Long Short Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997). The motivation of using these complex neural network approaches for classification tasks comes from the principle of compositionality (Frege, 1948), which states that the meaning of a longer expression (e.g. a sentence or a document) depends on

Proceedings of the 27th International Conference on Computational Linguistics, pages 2120–2131
Santa Fe, New Mexico, USA, August 20-26, 2018.

the meaning of its constituents. It is believed that lower layers of the network learn representations of words or phrases, and as we move up the layers more complex expressions are represented (Peters et al., 2018).

However, we believe that these state-of-the-art text classification techniques do not actually follow this mechanism, despite the motivations. Like any discriminative approach which can pick up any signal, it appears that these techniques most likely still learn and rely heavily on key lexical items or phrases and just use these lexicons to classify the document, which might not generalize to new documents.

To test our hypothesis, we first construct datasets (Section 3) where the training and test splits have no direct overlap of such key lexicons while taking care of class imbalance, although remaining language structure remains the same. Such dataset split also occurs in many real-world classification problems. For example, in the case of scientific documents, with the discovery of a new phenomenon/law/theorem, a new keyword is born. The document containing the new word would still belong to one of the existing classes, such as optics/biochemistry/discrete math, and a text classifier should be able to correctly classify the given document as the language structure would remain the same.

We study the performance of various text classifiers on this new training-test split and compare performance results with the commonly used random training-test split of such a dataset (Section 5). We observe that there is a big performance gap in current approaches between the two splits. We further show that even simple regularization techniques of replacing key lexicons with random embeddings can improve performance on the training-test split where there is no overlap of such keywords. We also present a novel approach based on the gradient of word embeddings that leads to further improvement on such dataset split (Section 4). We also provide two large-scale text classification datasets which contain both the random split and lexicon based split.

We report additional experimental results by modeling the above problem as that of unsupervised domain adaptation where the data distribution of training domain and test domain are different (Ganin et al., 2016; Zhang et al., 2017). Even such domain adaptation approaches do not yield better results than our proposed regularization techniques (Section 5).

2 Background and Related Work

In this section, we first discuss work related to neural network approaches for text classification followed by a discussion on recent domain adaptation techniques which can be thought of as an alternative solution for our problem.

2.1 Neural Network Approaches

Recent research on text classification tasks makes use of neural networks which has shown to outperform methods based on the bag-of-words model. These approaches take distributed representation of words as input which is also known as word vectors (Mikolov et al., 2013). These word vectors can be learned either using skip-gram (Mikolov et al., 2013) or Glove (Pennington et al., 2014) methods. One remarkable property of these vectors is that they learn the semantic relationships between words such that in the embedding space, semantically similar words are closer together. To learn document embeddings from these word vectors, Le and Mikolov (2014) use distributed bag-of-words (DBOW) approach also known as paragraph vectors while Joulin et al. (2017) computes the average of the word and subword vectors of a document to train a linear classifier.

To extract sentence level features for text classification task, Kim (2014) uses shallow CNN with max-pooling on top of pre-trained word vectors. They also observe that learning task-specific vectors through fine-tuning leads to better classification accuracy. Similarly, LSTM network pre-trained using language model parameters or sequence autoencoder parameters is used by Dai and Le (2015) for various text classification tasks. Recently, it was shown by Johnson and Zhang (2015a) that a CNN with dynamic max pooling layer can effectively use the word order structure when trained using one-hot encoding representation of input words. They also learn multi-view region embeddings through semi-supervised learning and incorporate them as additional features to further improve the model's performance (Johnson and Zhang, 2015b). Similarly, they also perform semi-supervised experiments using a simplified LSTM

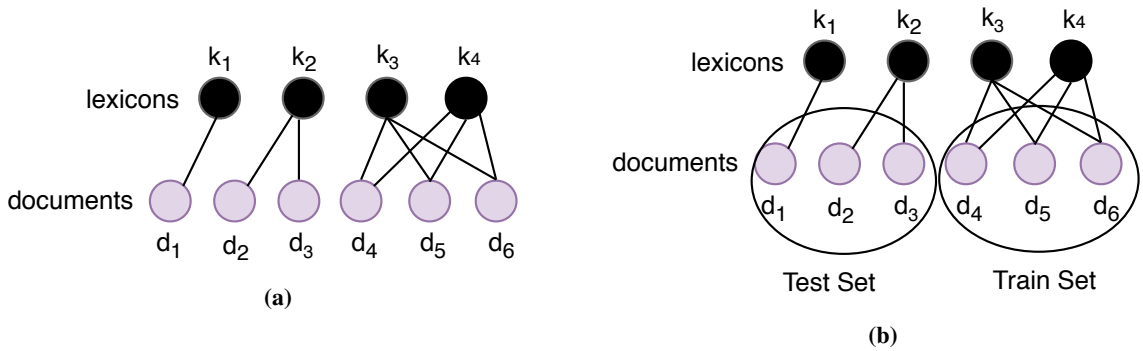


Figure 1: Schematic diagram illustrating the methodology of lexicon dataset construction. In (a), we construct a bipartite, undirected graph between the identified keywords ($k_1, \ldots, k_4$) and all the documents ($d_1, \ldots, d_6$). We identify all the connected components in this graph. In (b), all the documents belonging to the largest connected component (d_4, d_5, d_6) are selected as the training set while those documents belonging to all the other connected components (d_1, d_2, d_3) are selected as the test set.

model which also takes one-hot encoding of words as input (Johnson and Zhang, 2016).

2.2 Domain Adaptation

Next, we discuss the recent works applicable to domain adaptation of text classifiers. In unsupervised domain adaptation, the labeled instances from training data are considered as the source domain while unlabeled instances from test data are considered as the target domain. In this paper, it is assumed that there exists covariate shift between the two domains such that the conditional distribution of class label remains the same in both the domains while the marginal distribution of instances may differ (Shimodaira, 2000; Bickel et al., 2007). To address the problem of covariate shift, attempts have been made by adversarially training the encoder so as to make document representation domain invariant (Ganin et al., 2016).

Another approach to domain adaptation is multi-task learning as it can help improve the performance of text classifiers on lexicon dataset (Caruana, 1997). It is assumed that by learning related tasks in parallel while using a shared representation of document encoder can improve generalization by inducing proper inductive biases.

3 Lexicon Dataset Construction

As mentioned previously, to test our hypothesis whether the text classifiers just focus on key lexicons, we consider three text corpora: IMDB reviews, its standard subset (Maas et al., 2011), and Arxiv abstracts. For each dataset, we construct two versions: random split version and lexicon split version. In the random split version, selection of training and test examples is done by random sampling. The ratio of test to training examples is kept approximately the same for both the random and lexicon version of each dataset. In the lexicon split version, we make sure that key lexicons in training and test examples do not overlap while taking care of class imbalance. Below, we provide stepwise details of the approach used for the construction of lexicon version of the datasets.

1. **Identification of important label specific lexicons:** We begin by extracting *tf-idf* weighted, unigram to five-gram word features for all the documents. Using thus obtained feature vectors as input, we train a multinomial Naïve Bayes classifier for predicting the class on the entire dataset (both training and test examples). From the trained Naïve Bayes classifier, we extract the feature weights. Next, we rescale the feature weights by dividing all the feature values by the corresponding minimum value for that feature across all the classes. Lastly, to identify lexicons, we select the top-*k* features with the maximum weight for every class. We set the value of *k* to be 1500 for IMDB datasets and 150 for Arxiv abstracts dataset. We also experiment with Logistic Regression and SVM classifiers but observe that multinomial Naïve Bayes selects the most diverse set of features.

	cs.AI	cs.IR	cs.CV	cs.RO
Train	reinforcement learning logic programming markov decision processes probabilistic inference	information retrieval recommender systems collaborative filtering recommendation systems	computer vision image segmentation deep convolutional neural super resolution	motion planning humanoid robot dynamic environments aerial vehicle
Test	bayesian networks graphical models constraint satisfaction	matrix factorization social networks search engines	pose estimation optical flow sparse representation	multi robot simultaneous localization extended kalman filter

(a) Arxiv abstracts

	most-positive	positive	neutral	negative	most-negative
Train	absolutely amazing awesome experience fantastic movie	completely satisfied good experience great movie	average movie is just ok moderate movie	disappointed scope for improvement movie sucks	a huge let down movie was awful very frustrating
Test	outstanding performance very impressive	nice experience movie is good	satisfactory movie avg performance	not a good no plot in movie	worst experience pretty bad

(b) IMDB reviews

Table 1: Example lexicons for training and test set of Arxiv abstracts and IMDB reviews datasets. The column headers in both the tables indicate the names of various classes in these datasets. For Arxiv abstracts dataset, the details of all the class names can be found in the URL: `https://arxiv.org`

2. **Creation of lexicons-documents graph:** Since each label specific lexicon term can occur in multiple documents, and one document can contain many such lexicons, in order to create a disjoint partition of lexicons and documents into training/test sets, it is natural to represent documents and lexicons using a graph-based structure. Therefore, for each class, we create an undirected graph whose nodes are lexicons and documents respectively. If a lexicon occurs in a document, we create an edge in the graph between the two corresponding nodes. The resulting graph is bipartite, as edges only exist between the lexicon nodes and the document nodes but not among each other (Figure 1a).

3. **Graph partitioning to generate training/test splits:** We compute all the connected components of the lexicons-documents graph. We then identify the largest connected component, i.e. one which contains the maximum number of nodes and select all the documents in it as the training set.[1] The remaining documents from all the other connected components are selected as the test set (Figure 1b). If the ratio of the number of test to training documents is below a cutoff threshold, we repeat the above steps by gradually selecting fewer top-*k* features (i.e. smaller set of lexicons). Some example lexicons for training/test splits of Arxiv abstracts and IMDB reviews datasets is shown in Table 1.

We want to emphasize that after the creation of training and test sets, the majority of the words appear in both the sets for the lexicon version.

4 Methods

In this section, we will first briefly describe a simple neural network for text classification on which our proposed regularization methods are based.

Let the vocabulary size be V and embedding dimension be D. Our network consists of an embedding layer ($E \in \mathbb{R}^{V \times D}$), single layer bidirectional LSTM (BiLSTM) (Schuster and Paliwal, 1997), a pooling layer, and finally a linear layer with softmax function for classification. The sequence of word ids ($w_t, t \in [1, T]$) are given as input to the embedding layer which maps them to dense vectors. The forward LSTM and backward LSTM of a BiLSTM processes these input sequence of word vectors in forward and reverse directions respectively. The hidden state of forward LSTM and backward LSTM is concatenated at every time-step and are passed to the pooling layer which computes the maximum value over time

[1] We also tried spectral partitioning of lexicons-documents graph but it didn't generate balanced graph partitions.

dimension to obtain the representation of the input sequence (Conneau et al., 2017). We train the model by minimizing the cross-entropy loss.

While training various neural network models on both random and lexicon splits of ACL IMDB dataset (Maas et al., 2011), we observed that on the lexicon split, the network is easily able to learn the training data distribution in 1-2 epochs. These models also overfit on the lexicon split as the accuracy gap between the training and test set widens up. We hypothesize that this happens because, during training step, the model is able to memorize common label-specific keywords occurring in the training set. During evaluation, when the model is not able to spot such keywords in the test set, as it contains non-overlapping partition of keywords from the training set, the performance degrades quite rapidly as compared to the random split version. Therefore, motivated by these observations, we propose two methods which attempt to prevent neural networks from memorizing the word order structure in keywords by introducing more randomness in them.

4.1 Keyword Anonymization

In order to prevent the model from memorizing keyword specific rules and thus learning degenerate representations, we introduce more randomness in our training data split (Hermann et al., 2015). In the first step, we identify some keywords with high scores using a supervised classifier trained using bag-of-words as features. The first step is similar to the first step of data construction, with the important distinction that now we only operate on training set as test set is assumed to be unknown during the training phase. In the second step, we anonymize the corpus by replacing a single word selected at random from the occurrences of such keyword phrases with a placeholder word '`ANON`'. In the third step, we assign a random word embedding during model training for every occurrence of the placeholder word '`ANON`'. These modified word embeddings are given as input to the neural network encoder.

In this way, we corrupt the information present in keywords by introducing some form of random noise in the data. This is one of the ways to regularize model training. We later show that this regularization forces the network to learn context-aware text representations which gives a significant gain in classification accuracy. Motivated by the effectiveness of this method, we now propose an end-to-end version as the next approach.

4.2 Adaptive Word Dropout

This approach is based on embedding layer's gradient which is computed at every optimization step. The motivation of this approach is based on the observation that key lexicon terms have a higher norm of the embedding gradient. In this method, we first compute the word embedding gradient ($\nabla E \in \mathbb{R}^{V \times D}$) during the backward step. In the second step, for every word we maintain the running average ($A(w)$) of the L2-norm of this gradient. This average is computed with respect to the number of completed optimization steps.

$$\tilde{A}(w) = \tilde{A}(w) + \sqrt{\sum_{j=1}^{D} \nabla E_{ij}^2}$$

$$A(w) = \frac{1}{\text{optim\_steps}} \tilde{A}(w)$$

In the third step, we compute the dropout probability of every word as:

$$P_d(w) = 1 - \sqrt{\frac{t}{A(w)}}$$

In the above equation, t is a hyperparameter (typical values of t are 10^{-3}, 10^{-4}). We apply dropout on the words before the word embedding layer with probability $P_d(w)$ (Srivastava et al., 2014). We also apply variational dropout (Gal and Ghahramani, 2016) on the output of word embedding layer.

	Arxiv abstracts				IMDB reviews				ACL IMDB			
	lexicons		random		lexicons		random		lexicons		random	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
c	127	127	127	127	5	5	5	5	2	2	2	2
l	162	140	153	153	299	252	278	281	290	254	234	228
N	668K	438K	664K	443K	1.48M	1.07M	1.49M	1.07M	23K	26K	25K	25K
V	297K	238K	301K	244K	590K	509K	601K	505K	75K	78K	75K	74K

Table 2: This table shows the dataset summary statistics. c: Number of classes, l: Average length of a sentence, N: Dataset size, V: Vocabulary size

5 Experiments

We now present empirical studies in order to establish that (i) many text classifiers resort to just identifying key lexicons and thus poorly performing on specially crafted dataset splits (Section 5.3), and (ii) simple regularization techniques which disincentivize focusing on key lexicons can significantly boost performance (Section 5.4).

5.1 Dataset Description

To illustrate aforementioned claims, we evaluate on an existing dataset for binary sentiment classification and also collected two more large-scale text classification datasets. The details of these three datasets are described below:

- **ACL IMDB**: This is a popular benchmark dataset (Maas et al., 2011) of IMDB movie reviews for coarse-grained sentiment analysis in which the task is to determine if a review belongs to the positive class or to the negative class. The original random split version of this dataset contains an equal number of highly positive and highly negative reviews. To construct its lexicon based version, we apply our approach to the combined training and test splits of this dataset.

- **IMDB reviews**: This is a much bigger version of the IMDB movie reviews dataset in which the task is to do fine-grained sentiment analysis. We collect more than 2.5 million reviews from IMDB website and partition them into five classes based on their ratings out of 10. These classes are *most-negative*, *negative*, *neutral*, *positive*, and *most-positive*.

- **Arxiv abstracts**: In this, the task is to do multiclass topic classification. We construct this dataset by collecting more than 1 million abstracts of scientific papers from "*arxiv.org*". Each paper has one primary category such as cs.AI, stat.ML, etc. which we use as its class label. We selected those primary categories which have at least 500 papers. In order to extract text data, we use the title and abstract of each paper. In both the Arxiv abstracts and IMDB reviews dataset, we keep the ratio of the number of instances in the test set to that of training set as 0.6.

We pre-process the data by converting all the text to lowercase followed by Penn Treebank style tokenization. An overall summary of the training/test sets of lexicon and randomized splits is provided in Table 2.

5.2 Experimental Setup

In this section, we will discuss our experimental setup. For simple baseline comparisons, we conduct experiments with methods which take bag-of-words as input feature representations. Specifically, we compute *tf-idf* weighted *n-gram* features for each document to train multinomial Naïve Bayes and Logistic Regression classifiers. For neural network models, we have a common experimental setup based on Pytorch framework (Paszke et al., 2017) for all the datasets unless specified otherwise. Due

to computational constraints, we did not perform extensive hyperparameter tuning for the methods considered.

We select most common 80K words for model training. We initialize the embedding layer parameters for all the models using 300-dimensional pre-trained embeddings. These embeddings are trained using skip-gram approach (Mikolov et al., 2013) on the combined training and test set.[2] The embeddings of words which are not present in these vectors are uniformly initialized. The hidden state of LSTM and BiLSTM has 1024 and 512 dimensions respectively. For training of CNN models, we follow the settings mentioned in Kim (2014). We perform model training using mini-batch stochastic gradient descent with a batch size of 150. Optimization is performed using Adam Optimizer (Kingma and Ba, 2014) with default parameter settings. To train LSTM models, we perform backpropagation for 250 timesteps on IMDB datasets and 150 timesteps on Arxiv abstracts dataset. For model regularization, we apply dropout (Srivastava et al., 2014) with probability 0.5 to the input and output of LSTM. In order to prevent gradient explosion problem, we perform gradient clipping (Pascanu et al., 2013) by constraining the norm of the gradient to be less than 5.

5.3 Results

In order to estimate the difficulty level on both the lexicon and random splits version of a dataset, we do experiments with a wide range of popular approaches and show their results in Table 3. It can be observed from the results that for each dataset, there is a big performance gap between the random split and lexicon split for all the models. In this section, we will analyze these performance results. During our analysis, we will mostly focus on classifier's performance on the lexicon version of a dataset. For a detailed analysis of the performance of a method on the random version, we encourage the reader to read the associated paper.

5.3.1 Bag-of-Words Model

Here, we analyze the performance of methods which takes bag-of-words representation as input. We observe that the simple generative classifier of Naïve Bayes performs poorly as compared to other approaches on both Arxiv abstracts and IMDB reviews dataset, while its results are competitive on ACL IMDB dataset. Because of its relatively low scores, we don't experiment with more complex generative models. In contrast, a simple discriminative classifier such as Logisitic Regression performs significantly better than Naïve Bayes on the random version for all the datasets. However, on lexicon version, it's performance is quite low when compared to neural network methods. Due to this, the accuracy gap for this method is largest between both the dataset versions. These low scores of above classifiers on lexicon version can be explained owing to the conditional independence assumption among the input features. During training step, bag-of-words based models assign high weights to class-specific keywords. When this trained model is not able to spot such discriminative keywords in the test set due to strict no overlap condition, the performance of these methods degrades.

5.3.2 Simple Word Embedding based Methods

Next, we analyze the results of classifiers which are based on simple linear or nonlinear transformation of word embeddings. We train 300-dimensional document vectors (Le and Mikolov, 2014) using standard training settings for DBOW model.[3] Similarly, we use the recommended training setting for FastText method (Joulin et al., 2017). These methods further improve the lexicon results in both the IMDB datasets. Although DBOW and FastText methods don't strictly make use of word order in a sentence, improved performance suggests that they leverage the semantic properties of the embedding space as the document vectors are also learned in the same geometrical space as the word vectors.

5.3.3 Convolutional and Recurrent Networks

As discussed previously (Section 2), it has been shown that both CNN and LSTM models can learn effective text representations. We observe that these networks show much better performance on both the random and lexicon version as compared to the above-discussed methods. We also note that the use of a

[2]We use *word2vec* tool from `https://code.google.com/p/word2vec`

[3]We use the existing implementation from the open-source gensim toolkit.

Model	Arxiv abstracts			IMDB reviews			ACL IMDB		
	random	lexicon	Δ	random	lexicon	Δ	random	lexicon	Δ
Naïve Bayes	57.6	40.4	17.2	54.9	41.6	13.3	89.8	79.3	10.5
Logistic Regression	65.2	40.6	24.6	60.0	41.9	18.1	90.5	80.6	9.9
DocVec	62.6	38.9	23.7	55.9	44.4	11.5	88.6	82.8	5.8
FastText	66.8	43.8	23.0	57.9	45.1	12.8	88.5	80.5	8.0
Deep Sets	55.7	37.9	17.8	52.0	46.8	5.2	88.2	79.6	8.6
LSTM	65.0	45.3	19.7	64.6	48.6	16.0	90.6	81.9	8.7
BiLSTM	**67.8**	46.5	21.3	**64.8**	48.9	18.9	**91.4**	83.1	8.3
CNN-MaxPool	65.8	46.0	19.8	50.5	44.1	6.4	89.6	80.5	9.1
CNN-DynMaxPool	66.6	45.5	21.1	52.1	41.5	10.6	90.0	80.9	9.1
Adv-Training	–	45.1	–	–	47.3	–	–	82.2	–
Multi-task Learning	–	43.5	–	–	44.8	–	–	78.7	–
LSTM-Anon	67.2	48.2	19.0	62.6	50.3	12.3	89.0	83.1	5.9
BiLSTM-Anon	67.5	48.7	18.8	62.8	51.4	11.4	89.7	84.1	5.6
Adaptive Dropout	67.2	**49.1**	18.1	64.0	**52.6**	11.4	91.2	**86.0**	5.2

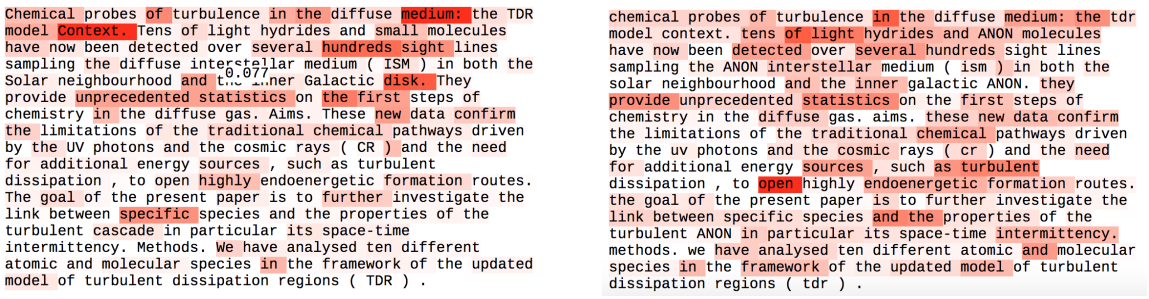
Table 3: Classification accuracy of various models on random and lexicon based version of each dataset. We also show the accuracy difference (Δ) between these two results. **Naïve Bayes**: *n-gram* feature extraction using *tf-idf* weighting followed by Naïve Bayes classifier. **Logistic Regression**: *n-gram* feature extraction using *tf-idf* weighting followed by Logistic Regression classifier. **DocVec**: Document vectors trained using DBOW model (Le and Mikolov, 2014). **FastText**: Average of the word and subword embeddings (Joulin et al., 2017). **Deep Sets** (Zaheer et al., 2017): two layer MLP on top of word embeddings with max pooling layer. **LSTM**: Word level LSTM as a document encoder in which hidden state of the last time-step is used for classification. **BiLSTM**: Word level bidirectional LSTM as a document encoder in which forward LSTM and backward LSTM hidden states are concatenated followed by max pooling layer (Conneau et al., 2017). **CNN-MaxPool**: CNN with max pooling layer (Kim, 2014). **CNN-DynMaxPool**: CNN with dynamic max pooling layer. For details on dynamic max pooling, we request the reader to refer to Johnson and Zhang (2015a). **Adv-Training**: Adversarial training of encoder is done to fool the discriminator and make the representation of training and test instances domain invariant. **Multi-task Learning**: A shared BiLSTM encoder is used to perform joint training of text classifier, denoising autoencoder, and adversarial training. **LSTM-Anon**: LSTM encoder applied to the anonymized training data. **BiLSTM-Anon**: BiLSTM encoder applied to the anonymized training data. **Adaptive Dropout**: BiLSTM encoder applied after embedding gradient-based adaptive word dropout.

more complex model such as BiLSTM gives performance improvement in both random and lexicon splits for all the datasets.

Although there is an improvement in classification accuracy, still there is a large drop in performance from random to lexicon split. For the best performing approach of BiLSTM model, accuracy difference between random and lexicon split of ACL IMDB, IMDB reviews, and Arxiv abstracts dataset is 8.3%, 18.9%, and 21.3% respectively. In order to narrow down this performance gap, there is a need for approaches which can learn more robust context-based representations so that the performance on documents containing new unseen key phrases can be improved.

5.3.4 Domain Adaptation Methods

For the lexicon split, one can consider that the training and test data distributions are different, and thus model this as an instance of unsupervised domain adaptation. We evaluate two different approaches for such domain adaptation: adversarial training and multi-task learning.



(a) without word anonymization

(b) with word anonymization

Figure 2: This attention heat map shows the change in hidden state activations of BiLSTM encoder when some label specific keywords are anonymized for an example instance from Arxiv abstracts dataset whose label is *astro-ph.GA*. In Figure 2a, the trained classifier makes an incorrect prediction as it can be seen that the model is performing a keyword-based classification. In Figure 2b with word anonymization, the attention span also involves other context words and the prediction is correct.

To address lexical/covariance shift in lexicon dataset split, we adversarially train BiLSTM model with the objective that the resulting document representation will be discriminative for the text classification task while indiscriminative to the training/test domain classification. To perform such domain-invariant adversarial training, we include a discriminator module which consists of three feed-forward layers. To learn the model parameters, we follow the training method of Ganin et al. (2016), but instead of gradient reversal, we feed the interchanged labels of the domains to fool the discriminator. From the results, we observe that adversarial training of encoder hurts the performance on lexicon datasets.

In multi-task learning, the various tasks which were trained jointly are text classification, denoising autoencoder (Hill et al., 2016), and adversarial training. In all these tasks, we share the embedding layer and BiLSTM encoder layer. Denoising autoencoder was trained on the entire dataset using the strategy described in Lample et al. (2017). From the results, we observe that multi-task learning doesn't help and it further hurts the overall classification accuracy.

We didn't experiment with these domain adaptation approaches on the random split of the dataset, as it is assumed that it contains the same distribution of lexicons in the training and test set. Also, as the initial results were not promising on lexicon dataset, we didn't try more complex approaches.

5.4 Our Methods

BiLSTM encoder applied to anonymized training data with random embedding substitution performs around 2% better on Arxiv abstracts dataset, 2.5% better on IMDB reviews dataset and gives around 1% improvement on ACL IMDB lexicon dataset. This shows that keyword anonymization with random embedding substitution can be a good strategy for model regularization in case of lexicon-based split. From a qualitative perspective, we show in Figure 2 the change of hidden state activations for the BiLSTM encoder after the data is anonymized.

Our approach of adaptive word dropout performs 2.6% better on Arxiv abstracts, 3.6% better on IMDB reviews and gives around 2.8% improvement on ACL IMDB lexicon datasets. We also note that this approach leads to an only marginal drop in accuracy for the random split version of the datasets. One of the reasons why this method is more effective is that word dropout partially masks some lexical terms in the training set thereby lowering the variance of the fitted model.

6 Conclusion

Recently, multilayer neural network models have gained wide popularity for text classification tasks due to their much better performance than traditional bag-of-words based approaches. It is widely believed that this happens as neural networks can effectively utilize the word order structure present in documents. But, a potential drawback is that since all neural network approaches are discriminative, they tend to identify key signals in the training data which may not generalize to test data. In this work, we investigate

whether these neural network models actually learn to compose the meaning of sentences or just use discriminative keywords.

To test the generalization ability of different state-of-the-art text classifiers, we construct hard datasets where the training and test splits have no direct overlap of lexicons. Our experiments with popular text classifiers show that there is a large drop in test classification accuracy between random and lexicon splits of these datasets. We show that simple regularization techniques such as keyword anonymization can substantially improve the performance of text classifiers. We also observe that adaptive word dropout method which is based on embedding layer's gradient can further improve accuracy and thus reduce the gap between the two dataset splits.

Acknowledgments

This work was supported by a generous research funding from CMU, MCDS students grant. We would also like to thank the anonymous reviewers for giving us their valuable feedback which helped to improve the paper.

References

[Bickel et al.2007] Steffen Bickel, Michael Brückner, and Tobias Scheffer. 2007. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 81–88, New York, NY, USA. ACM.

[Caruana1997] Rich Caruana. 1997. Multitask learning. *Mach. Learn.*, 28(1):41–75, July.

[Conneau et al.2017] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680. Association for Computational Linguistics.

[Dai and Le2015] Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 3079–3087, Cambridge, MA, USA. MIT Press.

[Frege1948] Gottlob Frege. 1948. Sense and reference. *The philosophical review*, 57(3):209–230.

[Gal and Ghahramani2016] Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1019–1027. Curran Associates, Inc.

[Ganin et al.2016] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030, January.

[Harris1954] Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

[Hermann et al.2015] Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*.

[Hill et al.2016] Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377. Association for Computational Linguistics.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November.

[Joachims1998] Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML'98, pages 137–142, Berlin, Heidelberg. Springer-Verlag.

[Johnson and Zhang2015a] Rie Johnson and Tong Zhang. 2015a. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103–112, Denver, Colorado, May–June. Association for Computational Linguistics.

[Johnson and Zhang2015b] Rie Johnson and Tong Zhang. 2015b. Semi-supervised convolutional neural networks for text categorization via region embedding. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 919–927, Cambridge, MA, USA. MIT Press.

[Johnson and Zhang2016] Rie Johnson and Tong Zhang. 2016. Supervised and semi-supervised text categorization using lstm for region embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 526–534. JMLR.org.

[Joulin et al.2017] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain, April. Association for Computational Linguistics.

[Kim2014] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.

[Kingma and Ba2014] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

[Lample et al.2017] Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

[Le and Mikolov2014] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org.

[Lecun et al.1998] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov.

[Maas et al.2011] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

[Manning et al.2008] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

[McCallum and Nigam1998] Andrew McCallum and Kamal Nigam. 1998. A comparison of event models for naive bayes text classification. In *Proceedings of AAAI-98, Workshop on Learning for Text Categorization*, pages 41–48. AAAI Press.

[Mikolov et al.2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

[Pascanu et al.2013] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1310–III–1318. JMLR.org.

[Paszke et al.2017] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.

[Peters et al.2018] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

[Schuster and Paliwal1997] M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681, November.

[Shimodaira2000] Hidetoshi Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227 – 244.

[Srivastava et al.2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January.

[Wang and Manning2012] Sida Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 90–94, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Zaheer et al.2017] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. 2017. Deep sets. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3391–3401. Curran Associates, Inc.

[Zhang et al.2017] Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *Transactions of the Association for Computational Linguistics*, 5:515–528.

A Review on Deep Learning Techniques Applied to Answer Selection

Tuan Manh Lai
Adobe Research
tlai@adobe.com

Trung Bui
Adobe Research
bui@adobe.com

Sheng Li
Adobe Research
sheli@adobe.com

Abstract

Given a question and a set of candidate answers, answer selection is the task of identifying which of the candidates answers the question correctly. It is an important problem in natural language processing, with applications in many areas. Recently, many deep learning based methods have been proposed for the task. They produce impressive performance without relying on any feature engineering or expensive external resources. In this paper, we aim to provide a comprehensive review on deep learning methods applied to answer selection.

1 Introduction

Answer selection is an active research field and has drawn a lot of attention from the natural language processing community. Given a question and a set of candidate answers, the task is to identify which of the candidates contains the correct answer to the question. For example, given the question "Who established the Nobel Prize?" and the following candidate answers:

1. The Nobel Prize was established more than 100 years ago.
2. The Fields Medal, established in 1936, is often described as the Nobel Prize of mathematics.
3. The Nobel Prize was established in the will of Alfred Nobel.

The third answer should be selected. This example shows that simple word matching is not enough. Even though, all of the sentences contain the keywords "established" and "Nobel Prize", only the third sentence answers the question. From this point, we assume that each candidate answer is a sentence although the discussion is also applicable to more general case such as each candidate answer is a passage.

Answer selection is an important problem in its own right as well as in the context of open domain question answering. Although details vary from system to system, a typical open domain question answering system can be considered as consisting of: (a) question analysis (b) retrieval of potentially relevant documents (c) ranking and selecting of the most promising sentences (or more generally, passages) within the retrieved documents; and optionally d) extracting the exact natural language phrase that answers the question (Prager, 2006; Ferrucci, 2012; Sequiera et al., 2017). Figure 1 depicts a typical question answering pipeline. In this setup, answer selection can be applied to identify the sentences that are most relevant to the question within the retrieved documents. Besides its application in open domain question answering, the techniques developed for answer selection can be potentially used to predict answer quality in community question answering (cQA) sites (Nakov et al., 2015).

Previous work on answer selection typically relies on feature engineering, linguistic tools, or external resources (Wang et al., 2007; Wang and Manning, 2010; Heilman and Smith, 2010; Yih et al., 2013; Yao et al., 2013). Recently, many deep learning based methods have been proposed for the task (Bian et al., 2017; Shen et al., 2017; Tran et al., 2018). They outperform traditional techniques. In addition, they do not need any feature-engineering effort or hand-coded resources beyond some large unlabeled corpus on which to learn the initial word embeddings, such as word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014).

Proceedings of the 27th International Conference on Computational Linguistics, pages 2132–2144
Santa Fe, New Mexico, USA, August 20-26, 2018.

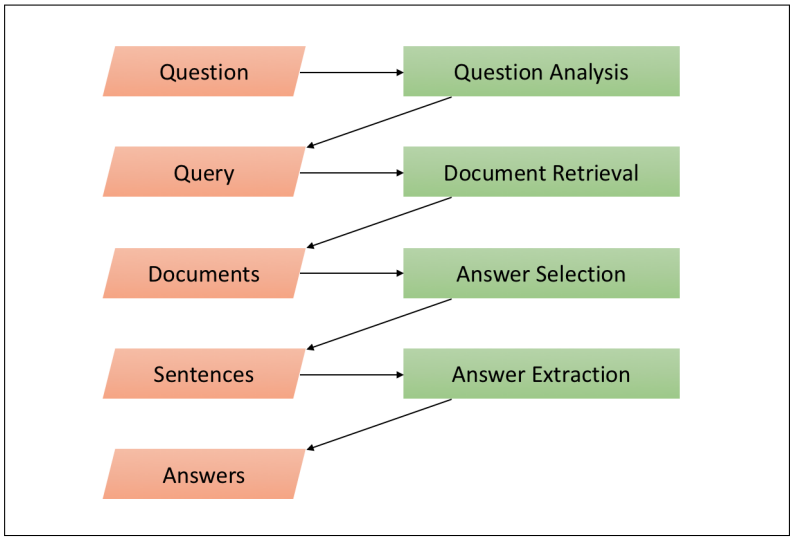


Figure 1: A typical question answering pipeline architecture, adapted from (Sequiera et al., 2017)

While previous work has recognized the increasing use of deep learning techniques in natural language processing (Young et al., 2017), no systematic survey of deep learning methods for answer selection to date has been published. Therefore, in this paper, we aim to give a comprehensive review of various deep learning methods that have been used to tackle the answer selection task.

In Section 2, we present a review of deep learning methods for answer selection. In Section 3, we examine the most popular datasets and the evaluation metrics for answer selection. We conclude this paper in Section 4 by discussing the potential future research directions.

2 Methods

Existing deep learning methods for answer selection can be examined along two dimensions: (i) learning approaches (ii) neural network architectures (Table 1).

2.1 Learning Approaches

Given a question and a set of candidate sentences, the task is to identify candidate sentences that contain the correct answer to the question. From the definition, the problem can be formulated as a ranking problem, where the goal is to give better rank to the candidate sentences that are relevant to the question. There are three most common approaches to learn the ranking function h_θ, namely, *pointwise*, *pairwise* and *listwise* (Liu, 2011).

In the *pointwise* approach, the ranking problem is transformed to a binary classification problem. More specifically, the training instances are triples (q_i, c_{ij}, y_{ij}), where q_i is a question in the dataset, c_{ij} is a candidate answer sentence for q_i, and y_{ij} is a binary value indicating whether c_{ij} contains the correct answer to q_i. It is enough to train a binary classifier: $h_\theta(q_i, c_{ij}) \rightarrow \hat{y_{ij}}$, where $0 \leq \hat{y_{ij}} \leq 1$. For example, in (Yu et al., 2014), the training objective is to minimize the cross entropy of all labelled question-candidate pairs in the training set. During inference, given a question, the trained classifier h_θ is used to rank every candidate sentence, and the top-ranked candidate is selected (i.e., $\text{argmax}_{c_{ij}} h_\theta(q_i, c_{ij})$ should be selected as the answer to q_i). Many work adopted this approach (Yu et al., 2014; Severyn and Moschitti, 2015; Wang et al., 2016b; Shen et al., 2017).

The second approach to ranking is the *pairwise* approach, where the ranking function h_θ is explicitly trained to score correct candidate sentences higher than incorrect sentences. Given a question, the approach takes a pair of candidate answer sentences and explicitly learns to predict which sentence is more relevant to the question. For example, in (Feng et al., 2015), the training instances are triples (q_i, c_i^+, c_i^-), where q_i is a question, c_i^+ is a correct sentence for q_i, and c_i^- is an incorrect sentence sampled from the

Method	Learning Approach	Model Architecture	MAP (Raw TrecQA)	MAP (Clean TrecQA)
TRAIN-ALL unigram+count (Yu et al., 2014)	Pointwise	Siamese	0.693	-
TRAIN-ALL bigram+count (Yu et al., 2014)	Pointwise	Siamese	0.711	-
QA-LSTM (Tan et al., 2015)	Pairwise	Siamese	-	0.682
QA-LSTM with attention (Tan et al., 2015)	Pairwise	Attentive	-	0.690
QA-LSTM/CNN (Tan et al., 2015)	Pairwise	Siamese	-	0.706
Attentive Pooling CNN (dos Santos et al., 2016)	Pairwise	Attentive	-	0.753
(Severyn and Moschitti, 2015)	Pointwise	Siamese	0.746	-
L.D.C Model (Wang et al., 2016b)	Pointwise	Compare-Aggregate	-	0.771
Pairwise Word Interaction Modelling (He and Lin, 2016)	Pointwise	Compare-Aggregate	0.758	-
Multi-Perspective CNN (He et al., 2015)	Pointwise	Siamese	0.762	0.777
HyperQA (Hyperbolic Embeddings) (Tay et al., 2018a)	Pairwise	Siamese	0.770	0.784
PairwiseRank+Multi-Perspective CNN (Rao et al., 2016)	Pairwise	Siamese	0.780	0.801
BiMPM (Shen et al., 2017)	Pointwise	Compare-Aggregate	-	0.802
Dynamic-Clip Attention (Bian et al., 2017)	Listwise	Compare-Aggregate	-	0.821
IWAN (Shen et al., 2017)	Pointwise	Compare-Aggregate	-	0.822
IWAN+CARNN (Tran et al., 2018)	Pointwise	Compare-Aggregate	-	0.829
MCAN (Tay et al., 2018b)	Pointwise	Compare-Aggregate	-	0.838

Table 1: Overview of existing deep learning methods to answer selection

whole candidate sentence space. And the hinge loss function is defined as follows:

$$L = max\{0, m - h_\theta(q_i, c_i^+) + h_\theta(q_i, c_i^-)\} \quad (1)$$

where m is the margin. If $h_\theta(q_i, c_i^+) - h_\theta(q_i, c_i^-) < m$ then L is positive. When this condition is satisfied, the implication is that the system ranks the positive sentence below the negative sentence, or does not sufficiently rank the positive answer above the negative answer. On the other hand, if the correct sentence has a score higher than the incorrect sentence by at least a margin m (i.e., $h_\theta(q_i, c_i^+) - h_\theta(q_i, c_i^-) \geq m$), and then the above expression has zero loss. In summary, the loss function is designed to encourage the correct answer to have a higher score than the incorrect answer by a certain margin. Similar to the *pointwise* approach, during testing, the candidate answer with the largest score is selected. (Tan et al., 2015; Yang et al., 2016; dos Santos et al., 2016; Tay et al., 2018a) also used the pairwise hinge loss function above.

The third method is the *listwise* approach (Cao et al., 2007). The *pointwise* approach and the *pairwise* approach ignore the fact that answer selection is a prediction task on list of candidate sentences. In

the *listwise* approach, a single instance consists of a question and its list of candidates. For example, (Bian et al., 2017) adopted the approach. Concretely, during training, given a question q_i and its list of candidate sentences $\mathbf{C}$ $\{c_{i1}, c_{i2}, ..., c_{im}\}$ and the ground truth labels $\mathbf{Y}$ $\{y_{i1}, y_{i2}, ..., y_{im}\}$, the normalized score vector $\mathbf{S}$ is calculated as follows:

$$\begin{aligned} Score_j &= h_\theta(q_i, c_{ij}) \\ \mathbf{S} &= softmax([Score_1, Score_2, ..., Score_m]) \end{aligned} \tag{2}$$

Target labels also need to be normalized:

$$\mathbf{Y} = \frac{\mathbf{Y}}{\sum_{j=1}^{m} y_{ij}} \tag{3}$$

And the objective is to minimize the KL-divergence of $\mathbf{S}$ and $\mathbf{Y}$.

Even though many work (Yu et al., 2014; Severyn and Moschitti, 2015; Wang et al., 2016b; Shen et al., 2017) adopted the *pointwise* approach, this approach is not close to the nature of ranking. The *pairwise* approach and the *listwise* approach exploit more information about the ground truth ordering of candidate sentences. (Rao et al., 2016) proposed a *pairwise* ranking approach that can directly exploit existing *pointwise* neural network models as base components. The approach outperforms many competitive *pointwise* baselines. (Bian et al., 2017) showed that the *listwise* approach performs better than the *pointwise* approach on public datasets such as TrecQA (Wang et al., 2007) and WikiQA (Yang et al., 2015). In the next section, we describe various neural network architectures for modeling the ranking function h_θ, which takes a question-candidate pair and returns a score indicating whether the candidate is relevant to the question.

2.2 Neural Network Architectures

There are three main types of general architectures for measuring the relevance of a candidate sentence to a question.

- **Siamese Architecture.** In a Siamese architecture (Bromley et al., 1993), the same encoder (e.g., a CNN or a RNN) is used to build the vector representations for the input sentences (i.e., the candidate answer and the question) individually. After that, the relevance score is determined solely based on the encoded vectors. There is no explicit interaction between the input sentences during the encoding process.
- **Attentive Architecture.** Instead of generating representations for the candidate answer and the question independently, attention mechanisms (Bahdanau et al., 2014; Hermann et al., 2015; Luong et al., 2015) can be used to allow the information from an input sentence to influence the computation of the other's representation (Tan et al., 2015; dos Santos et al., 2016). Even though the weakness of the Siamese models is alleviated, the interaction between the input sentences during the encoding process is still minimal in most Attentive architectures.
- **Compare-Aggregate Architecture.** The Compare-Aggregate architectures can capture more interactive features between input sentences than the Siamese architectures and the Attentive architectures, therefore typically have better performance when evaluated on public datasets such as TrecQA (Wang et al., 2007) and WikiQA (Yang et al., 2015). In a Compare-Aggregate architecture, vector representations of small units such as words of the sentences are first compared. After that, these comparison results are aggregated to calculate the final relevance score.

It is worth mentioning that the boundaries between the architecture types are not always crystal clear. For example, while many Siamese architectures typically capture less interactive features between the input sentences than the Attentive architectures, few recently proposed Siamese architectures have sophisticated comparison layer after the encoding layer (He et al., 2015; Rao et al., 2016). As a result, they even outperform some Attentive architectures. Even though the boundaries are not crystal clear, separating the existing different neural architectures into the three categories can provide the big picture more easily.

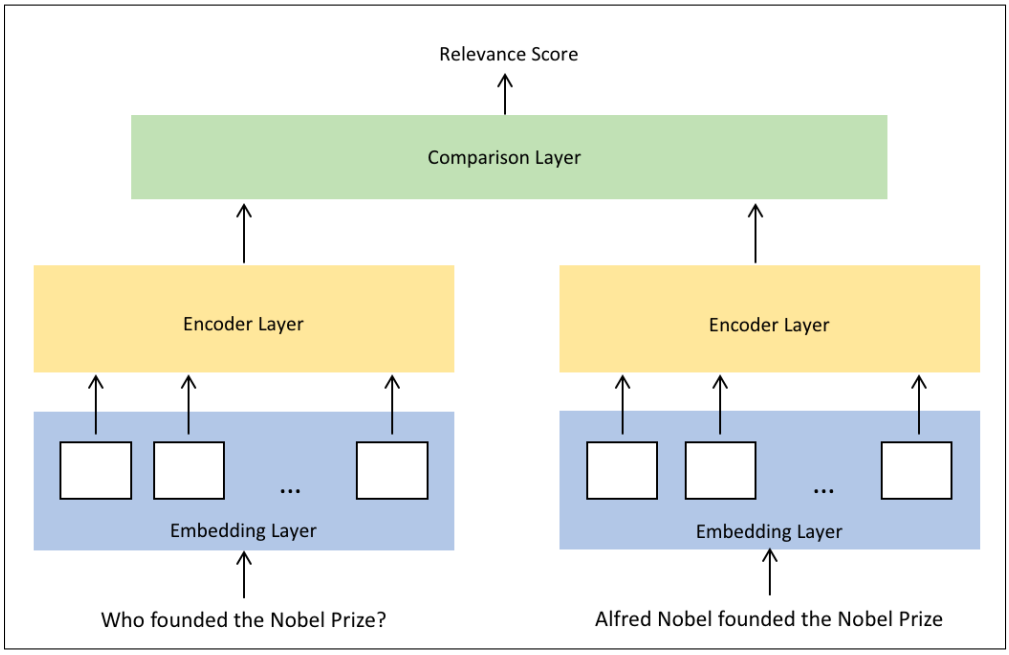


Figure 2: The general architecture of a Siamese model. The same encoder is used to generate the vector representations for the input sentences.

2.2.1 Siamese Architecture

Siamese neural networks have been proposed for a number of sentence pair modeling tasks, including semantic similarity (Mueller and Thyagarajan, 2016), paraphrase identification (Hu et al., 2014), and natural language inference (Conneau et al., 2017). Figure 2 shows the general architecture of a Siamese model. The vector representations of the input sentences are built separately by the encoder. Two input sentences have no influence on the computation of each other's representation. After that, the encoded vectors are compared using measures such as cosine similarity (Feng et al., 2015; Yang et al., 2015), element-wise operations (Tai et al., 2015), or neural network-based combination (Bowman et al., 2015). An advantage of this architecture is that applying the same encoder to each input sentence makes the model smaller. In addition, the sentence vectors can be used for visualization, sentence clustering and many other purposes (Wang et al., 2016a).

One of the first attempts at applying deep learning to answer selection was the bag-of-words model proposed by (Yu et al., 2014). The model generates the vector representation of a sentence by simply taking the average of all the word vectors in the sentence - having previously removed all the stop words. Integrating additional overlapping word count features with the model achieves performance better than many traditional techniques that require large numbers of hand-crafted features or external resources. Tan et al. (2015) proposed the QA-LSTM model that employs a bidirectional long short-term memory (biLSTM) network (Hochreiter and Schmidhuber, 1997) and a pooling layer to construct distributed vector representations of the input sentences independently. Then the model utilizes cosine similarity to measure the distance of the sentence representations. Severyn and Moschitti (2015) proposed a model that employs a convolutional neural network (CNN) to generate the representations of the input sentences. The CNN is based on an architecture that has previously been applied to many sentence classification tasks (Kalchbrenner et al., 2014; Kim, 2014). In (He et al., 2015), each input sentence is modeled using a CNN that extracts features at multiple levels of granularity and uses multiple types of pooling. The representations of the input sentences are then compared at several granularities using multiple similarity metrics. Finally, the comparison results are fed into a fully connected layer to obtain the final relevance score. The proposed model outperforms many other Siamese models. Tay et al. (2018a) proposed a simple but novel deep learning architecture that models the relationship between a question and a candidate sentence in Hyperbolic space instead of Euclidean space. It achieves highly competitive performance

without employing any sophisticated neural encoder such as LSTM or CNN. Vector representations of the input sentences are generated independently in a bag-of-words manner.

2.2.2 Attentive Architecture

In a Siamese architecture, the input sentences are first encoded into fixed-length vector representations separately, and these representations are then compared. Despite its conceptual simplicity, a disadvantage is the absence of explicit interaction between the input sentences during the encoding process. A question is always mapped to the same vector regardless of the candidate answer in consideration, and vice versa. Attention mechanisms have been applied to alleviate the weakness. Tan et al. (2015) extended the basic QA-LSTM model with attention (Figure 3). The model employs a biLSTM network and a pooling layer to generate the question representation $\mathbf{o}_q$. The candidate representation $\mathbf{o}_c$ is calculated similarly, except that prior to the pooling layer, each biLSTM output vector will be multiplied by a weight, which is determined by the question representation $\mathbf{o}_q$. Conceptually, the attention mechanism gives more weights on certain words in the candidate answer, and the weights are computed according to the question information. In this case, the attention mechanism is performed only in a single direction. dos Santos et al. (2016) proposed a two-way attention mechanism called Attentive Pooling (AP). AP allows the information from the two input sentences to influence the computation of each other's representation.

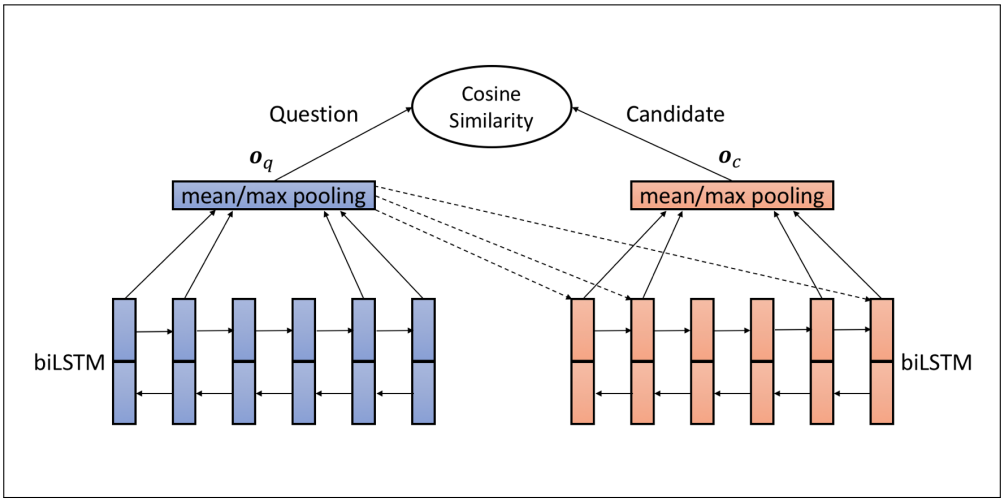


Figure 3: QA-LSTM with attention (figure adapted from (Tan et al., 2015))

2.2.3 Compare-Aggregate Architecture

A common trait of a number of recent state-of-the-art methods for answer selection is the use of the Compare-Aggregate architecture (Wang and Jiang, 2016; Wang et al., 2017; Bian et al., 2017; Shen et al., 2017; Tran et al., 2018). Under this architecture, smaller units (such as words) of the input sentences are compared. And then the comparison results are aggregated (e.g., by a CNN or a RNN) to make the final decision. Figure 4 shows the architecture of BiMPM, a Compare-Aggregate model proposed in (Wang et al., 2017). The model consists of five layers as follows.

Word Representation Layer. The goal of this layer is to represent each word in the input sentences with a d-dimensional vector. BiMPM constructs the d-dimensional vector with two components: a character-composed embedding and a word embedding pre-trained with GloVe (Pennington et al., 2014) or word2vec (Mikolov et al., 2013).

Context Representation Layer. The goal of this layer is to obtain a new representation for each position in the input sentences that captures some contextual information in addition to the word at the position. BiMPM employs a biLSTM to generate the contextual representations.

Matching Layer. The goal of this layer is to compare each contextual representation of one sentence against all contextual representations of the other sentence. The output of this layer are two sequences of

matching vectors, where each matching vector corresponds to the comparison result of one position of a sentence against all the positions of the other sentence.

Aggregation Layer. The goal of this layer is to aggregate the comparison results from the previous layer. BiMPM employs another BiLSTM to aggregate the two sequences of matching vectors into fixed-length vectors.

Prediction Layer. The goal of this layer is to make the final prediction. BiMPM uses a two layer feed-forward neural network to consume the fixed-length vectors from the previous layer, and apply the softmax function to get the final score.

Even though specific details vary from model to model, other Compare-Aggregate models for answer selection (Bian et al., 2017; Shen et al., 2017; Tran et al., 2018) have a similar structure to the BiMPM model. The Compare-Aggregate architectures can capture more interactive features between input sentences than the Siamese architectures and the Attentive architectures, therefore typically have better performance when evaluated on public datasets such as TrecQA (Table 1)

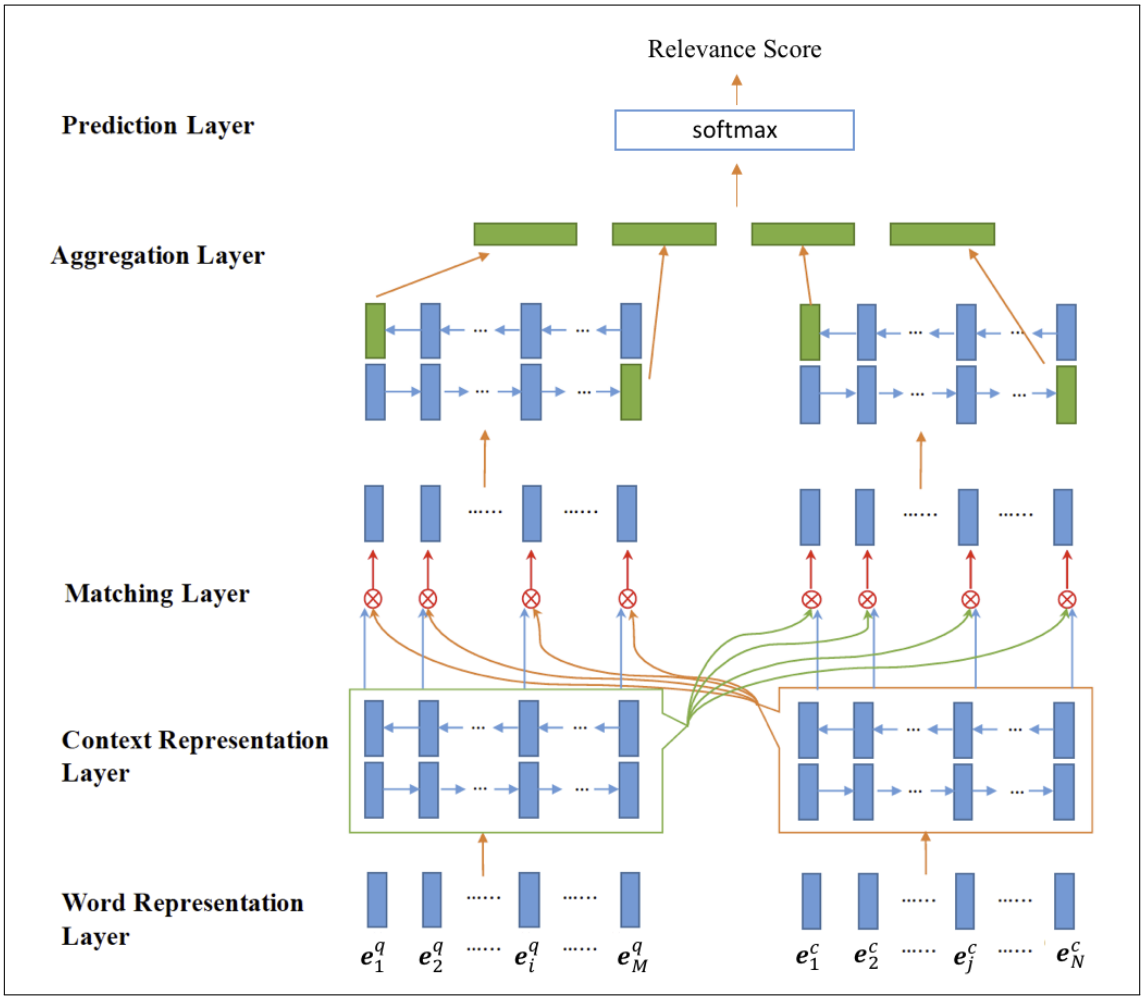


Figure 4: The architecture of the BiMPM model (figure adapted from (Wang et al., 2017))

3 Datasets and Evaluation Metrics

TrecQA, WikiQA, and InsuranceQA datasets have been widely used for benchmarking answer selection systems.

- The TrecQA dataset (Wang et al., 2007) was created from the TREC Question Answering tracks. In the literature (Yih et al., 2013; Yu et al., 2014; Severyn and Moschitti, 2015; Wang et al., 2016b;

dos Santos et al., 2016), we observe two versions of TrecQA: both have the same training set but their development and test sets differ. The Raw version has 82 development questions and 100 test questions. The Clean version removed questions in development and test sets with no answers or only positive/negative answers, reducing the development and test set's sizes to 65 and 68 questions, respectively. Relevant statistics are shown in Table 2. The data provided for training come as two sets: a small set of 94 questions (TRAIN) that were manually judged and a more noisy set of 1229 questions (TRAIN-ALL) that comes with automatic judgements.

Split	# Questions	# QA pairs	% Correct
TRAIN	94	4718	7.4%
TRAIN-ALL	1229	53417	12.0%
Raw DEV	82	1148	19.3%
Raw TEST	100	1517	18.7%
Clean DEV	65	1117	18.4%
Clean TEST	68	1442	17.2%

Table 2: Statistics of the TrecQA dataset

- The WikiQA dataset (Yang et al., 2015) is an open-domain question answering dataset that was constructed from real queries of Bing and Wikipedia. Relevant statistics are shown in Table 3. There are 3047 questions and 29258 candidate answer sentences in the dataset, and 1473 sentences were labeled as correct answer sentences to their corresponding questions. In the WikiQA dataset, there are questions with only incorrect answers. All questions with no correct answers are usually removed when the dataset is used to train and evaluate answer selection systems (Yang et al., 2015; Bian et al., 2017; Shen et al., 2017). The excluded WikiQA has 873/126/243 questions and 8627/1130/2351 question-answer pairs for train/dev/test split.

	Train	Dev	Test	Total
# Questions	2118	296	633	3047
# Candidate Answers	20360	2733	6165	29258
# Correct Answers	1040	140	293	1473
# Questions w/o Correct Answers	1245	170	390	1805

Table 3: Statistics of the WikiQA dataset

- The InsuranceQA dataset (Feng et al., 2015) is a large-scale domain specific answer selection dataset in which all question and candidate pairs are in the insurance domain. The original dataset consists of four parts: train, development, test1 and test2. Relevant statistics are shown in Table 4. There could be multiple correct answers for some questions so that the number of correct answers is larger than the number of questions. The released dataset contains 24981 unique answers in total. For each question, the candidate answer pool size is set to be 500. These candidate pools were constructed by including the correct answer(s) and randomly selecting candidates from the complete set of unique answers.

Split	# Questions	# Correct Answers
TRAIN	12887	18540
DEV	1000	1454
TEST1	1800	2616
TEST2	1800	2593

Table 4: Statistics of the original InsuranceQA dataset

Community Question Answering (cQA) platforms such as Stack Overflow [1] and Yahoo Answers [2] have become an important resource of information for many Web users. A person posts a question on

[1] http://stackoverflow.com
[2] https://answers.yahoo.com

a specific topic and other users post their answers. The techniques developed for answer selection can be potentially used to improve various aspects of a cQA platform. For example, in a cQA platform, it is not unusual for a question to have hundreds of answers, the vast majority of which would not satisfy a user's information needs. Therefore, using answer selection techniques to find relevant answers can be very beneficial. In the literature, the SemEval-2016 cQA dataset (Nakov et al., 2016) has been widely used to test different answer selection systems.

- In the SemEval-2016 cQA challenge (Nakov et al., 2016), there are three subtasks for English. Subtask A (*Question-Comment Similarity*): Given a question and its first ten comments in the question thread, the goal is to rank these ten comments according to their relevance with respect to the question. Subtask B (*Question-Question Similarity*): Given a new question and the set of the first ten related questions from the forum retrieved by a search engine, the goal is to rank the related questions according to their similarity with respect to the new question. Subtask C (*Question-External Comment Similarity*): Given a new question and the set of the first ten related questions from the forum retrieved by a search engine, each associated with its first ten comments appearing in its thread, the goal is to rank the 100 comments according to their relevance with respect to the new question. The data of the three subtasks was extracted from the community-created Qatar Living Forums [3]. Answer selection techniques can be directly applied to each of the subtask. All the information related to the dataset can be found on the SemEval-2016 Task 3 website[4].

Table 5 shows examples from the datasets. Each example consists of a question with a positive answer and a negative answer. A major difference between the SemEval-2016 cQA dataset and the other answer selection datasets is the average length of a question. In WikiQA (Yang et al., 2015) and TrecQA (Wang et al., 2007), a question is typically a short sentence, while in the cQA dataset, the question body typically contains quite a few sentences.

Dataset	Example
TrecQA	**Question**: Who established the Nobel prize awards? **Positive Answer**: The Nobel Prize was established in the will of Alfred Nobel, a Swede who invented dynamite and died in 1896. **Negative Answer**: The awards aren't given in specific categories.
WikiQA	**Question**: How many albums has Eminem sold in his career? **Positive Answer**: He has sold more than 100 million records worldwide, including 42 million tracks and 49.1 million albums in the United States. **Negative Answer**: Eminem is one of the best-selling artists in the world and is the best-selling artist of the 2000s.
InsuranceQA	**Question**: Does Medicare cover my spouse? **Positive Answer**: If your spouse has worked and paid Medicare taxes for the entire required 40 quarters, or is eligible for Medicare by virtue of being disabled or some other reason, your spouse can receive his/her own medicare benefits. If your spouse has not met those qualifications, if you have met them, and if your spouse is age 65, he/she can receive Medicare based on your eligibility. **Negative Answer**: If you were married to a Medicare eligible spouse for at least 10 years, you may qualify for Medicare. If you are widowed, and have not remarried, and you were married to your spouse at least 9 months before your spouses death, you may be eligible for Medicare benefits under a spouse provision.
SemEval-2016 cQA	**Question**:Hi;Can any one tell me a place where i can have a good massage drom philipinies????? yesterday i had a massage in Bio-Bil they charged me 300qr for 01 hour bt it is totally waste... pls advice me if theres any philipinos.... **Positive Answer**: Try Magic Touch in Abu Hamour (beside Abu Hamour Petrol Stn)it will just cost you 60QR per hour and I've seen a lot of Qataris as their customers. **Negative Answer**: I dont know the name; you can call them. Do it fast; they have sooooo many reservations ;)

Table 5: Examples from different datasets

Typically, the performance of an answer selection system is measured in Mean Reciprocal Rank (MRR) and Mean Average Precision (MAP), which are standard metrics in Information Retrieval and

[3] http://www.qatarliving.com/forum

[4] http://alt.qcri.org/semeval2016/task3

Question Answering. Given a set of questions Q, MRR is calculated as follows:

$$\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\text{rank}_i} \tag{4}$$

where rank_i refers to the rank position of the first correct candidate answer for the i^{th} question. In other words, MRR is the average of the reciprocal ranks of results for the questions in Q. On the other hand, if the set of correct candidate answers for a question $q_j \in Q$ is $\{d_1, d_2, ..., d_{m_j}\}$ and R_{jk} is the set of ranked retrieval results from the top result until you get to the answer d_k, then MAP is calculated as follows:

$$\text{MAP} = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{|m_j|}\text{Precision}(R_{jk}) \tag{5}$$

When a relevant answer is not retrieved at all for a question, the precision value for that question in the above equation is taken to be 0.

Whereas MRR measures the rank of any correct answer, MAP examines the ranks of all the correct answers. In general, MRR is higher than MAP on the same list of ranked outputs, except that they are the same in the case where each question has exactly one correct answer. Table 1 shows the performance of several deep learning methods when trained and evaluated on the Raw TrecQA dataset or the Clean TrecQA dataset.

4 Discussion and Future Directions

Answer selection is an important problem in natural language processing, and many deep learning methods have been proposed for the task. In this paper, we give a comprehensive and systematic review of various deep learning methods for answer selection along two dimensions: (i) learning approaches (*pointwise*, *pairwise*, and *listwise*) (ii) neural network architectures (*Siamese architecture*, *Attentive architecture*, and *Compare-Aggregate architecture*). In addition, we examine the most popular datasets and the evaluation metrics for answer selection. Below we discuss several promising future research directions.

Transfer learning (Pan and Yang, 2010) has achieved success in domains such as speech recognition (Huang et al., 2013), computer vision (Razavian et al., 2014), and natural language processing (Zhang et al., 2017). Its applicability to question answering and answer selection has recently been studied (Min et al., 2017; Chung et al., 2017). Min et al. (2017) created SQuAD-T, a modification of the original large-scale SQuAD dataset (Rajpurkar et al., 2016) to allow for directly training and evaluating answer selection systems. Through a basic transfer learning technique from SQuAD-T, the state-of-the-art result in the WikiQA dataset can be improved. This demonstrates the potential of developing novel transfer learning techniques for the answer selection task.

Many deep learning methods for answer selection are applicable to other sentence pair modeling tasks such as natural language inference (He and Lin, 2016; Wang et al., 2017), paraphrase identification (He and Lin, 2016; Wang et al., 2017), or measuring semantic relatedness (Shen et al., 2017). Extending existing methods for answer selection to achieve state-of-the-art results on other sentence pair modeling tasks can be an interesting research direction, and vice versa.

In addition to its applications in open domain question answering and community question answering, answer selection has many other applications. Lai et al. (2018) formulated the task of answering questions related to product facts and specifications as an answer selection problem. Given a question and a set of candidate product specifications, an answer selection technique was used to identify the specification that is most relevant to the question. Applying answer selection techniques to real-world problems can be an interesting research direction. For example, answer selection techniques can be potentially used to enhance truth discovery methods (Li et al., 2017; Zhang et al., 2018).

References

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

Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A compare-aggregate model with dynamic-clip attention for answer selection. In *CIKM*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Jane Bromley, James W. Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. *IJPRAI*, 7:669–688.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: From pairwise approach to listwise approach. Technical report, April.

Yu-An Chung, Hung-yi Lee, and James R. Glass. 2017. Supervised and unsupervised transfer learning for question answering. *CoRR*, abs/1711.05345.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR*, abs/1602.03609.

Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 813–820.

David A. Ferrucci. 2012. Introduction to "this is watson". *IBM Journal of Research and Development*, 56(3):1.

Hua He and Jimmy J. Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *HLT-NAACL*.

Hua He, Kevin Gimpel, and Jimmy J. Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *EMNLP*.

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 1011–1019, Stroudsburg, PA, USA. Association for Computational Linguistics.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9 8:1735–80.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*.

J. T. Huang, J. Li, D. Yu, L. Deng, and Y. Gong. 2013. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7304–7308, May.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Tuan Lai, Trung Bui, Sheng Li, and Nedim Lipka. 2018. A simple end-to-end question answering model for product information. In *ECONLP@ACL*.

Yaliang Li, Nan Du, Chaochun Liu, Yusheng Xie, Wei Fan, Qi Li, Jing Gao, and Huan Sun. 2017. Reliable medical diagnosis from crowdsourcing: Discover trustworthy answers from non-experts. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 253–261. ACM.

Tie-Yan Liu. 2011. *Learning to Rank for Information Retrieval*. 01.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Sewon Min, Min Joon Seo, and Hannaneh Hajishirzi. 2017. Question answering through transfer learning from large fine-grained supervision data. In *ACL*.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI*.

Preslav Nakov, Lluís Màrquez i Villodre, Walid Magdy, Alessandro Moschitti, Jim Glass, and Bilal Randeree. 2015. Semeval-2015 task 3: Answer selection in community question answering. In *SemEval@NAACL-HLT*.

Preslav Nakov, Lluís Màrquez i Villodre, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. Semeval-2016 task 3: Community question answering. In *SemEval@NAACL-HLT*.

S. J. Pan and Q. Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct.

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

John M. Prager. 2006. Open-domain question-answering. *Foundations and Trends in Information Retrieval*, 1(2):91–231.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*.

Jinfeng Rao, Hua He, and Julie Qiaojin Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *CIKM*.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. Cnn features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW '14, pages 512–519, Washington, DC, USA. IEEE Computer Society.

Royal Sequiera, Gaurav Baruah, Zhucheng Tu, Salman Mohammed, Jinfeng Rao, Haotian Zhang, and Jimmy J. Lin. 2017. Exploring the effectiveness of convolutional neural networks for answer selection in end-to-end question answering. *CoRR*, abs/1707.07804.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017. Inter-weighted alignment network for sentence pair modeling. In *EMNLP*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*.

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *CoRR*, abs/1511.04108.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018a. Hyperbolic representation learning for fast and efficient neural question answering. In *WSDM*.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018b. Multi-cast attention networks. In *KDD*.

Quan H. Tran, Tuan Lai, Ingrid Zukerman, Gholamreza Haffari, Trung Bui, and Hung Bui. 2018. The context-dependent additive recurrent neural net. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Shuohang Wang and Jing Jiang. 2016. A compare-aggregate model for matching text sequences. *CoRR*, abs/1611.01747.

Mengqiu Wang and Christopher D. Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1164–1172, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016a. Semi-supervised clustering for short text via deep representation learning. In *CoNLL*.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016b. Sentence similarity learning by lexical decomposition and composition. In *COLING*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *IJCAI*.

Yi Yang, Wen tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*.

Liu Yang, Qingyao Ai, Jiafeng Guo, and W. Bruce Croft. 2016. anmm: Ranking short answer texts with attention-based neural matching model. In *CIKM*.

Xuchen Yao, Benjamin Van Durme, Chris Callison-burch, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Scott Wen-tau Yih, Ming-Wei Chang, Chris Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. ACL Association for Computational Linguistics, August.

Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2017. Recent trends in deep learning based natural language processing. *CoRR*, abs/1708.02709.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen G. Pulman. 2014. Deep learning for answer sentence selection. *CoRR*, abs/1412.1632.

Yuan Zhang, Regina Barzilay, and Tommi S. Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *CoRR*, abs/1701.00188.

Hengtong Zhang, Yaliang Li, Fenglong Ma, Jing Gao, and Lu Su. 2018. Texttruth: An unsupervised approach to discover trustworthy information from multi-sourced text data. In *Proceedings of the 24rd SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM.

A Survey on Recent Advances in Named Entity Recognition from Deep Learning models

Vikas Yadav
University of Arizona
vikasy@email.arizona.edu

Steven Bethard
University of Arizona
bethard@email.arizona.edu

Abstract

Named Entity Recognition (NER) is a key component in NLP systems for question answering, information retrieval, relation extraction, etc. NER systems have been studied and developed widely for decades, but accurate systems using deep neural networks (NN) have only been introduced in the last few years. We present a comprehensive survey of deep neural network architectures for NER, and contrast them with previous approaches to NER based on feature engineering and other supervised or semi-supervised learning algorithms. Our results highlight the improvements achieved by neural networks, and show how incorporating some of the lessons learned from past work on feature-based NER systems can yield further improvements.

1 Introduction

Named entity recognition is the task of identifying named entities like person, location, organization, drug, time, clinical procedure, biological protein, etc. in text. NER systems are often used as the first step in question answering, information retrieval, co-reference resolution, topic modeling, etc. Thus it is important to highlight recent advances in named entity recognition, especially recent neural NER architectures which have achieved state of the art performance with minimal feature engineering.

The first NER task was organized by Grishman and Sundheim (1996) in the Sixth Message Understanding Conference. Since then, there have been numerous NER tasks (Tjong Kim Sang and De Meulder, 2003; Tjong Kim Sang, 2002; Piskorski et al., 2017; Segura Bedmar et al., 2013; Bossy et al., 2013; Uzuner et al., 2011). Early NER systems were based on handcrafted rules, lexicons, orthographic features and ontologies. These systems were followed by NER systems based on feature-engineering and machine learning (Nadeau and Sekine, 2007). Starting with Collobert et al. (2011), neural network NER systems with minimal feature engineering have become popular. Such models are appealing because they typically do not require domain specific resources like lexicons or ontologies, and are thus poised to be more domain independent. Various neural architectures have been proposed, mostly based on some form of recurrent neural networks (RNN) over characters, sub-words and/or word embeddings.

We present a comprehensive survey of recent advances in named entity recognition. We describe knowledge-based and feature-engineered NER systems that combine in-domain knowledge, gazetteers, orthographic and other features with supervised or semi-supervised learning. We contrast these systems with neural network architectures for NER based on minimal feature engineering, and compare amongst the neural models with different representations of words and sub-word units. We show in Table 1 and Table 2 and discuss in Section 7 how neural NER systems have improved performance over past works including supervised, semi-supervised, and knowledge based NER systems. For example, NN models on news corpora improved the previous state-of-the-art by 1.59% in Spanish, 2.34% in German, 0.36% in English, and 0.14%, in Dutch, without any external resources or feature engineering. We provide resources, including links to shared tasks on NER, and links to the code for each category of NER system. To the best of our knowledge, this is the first survey focusing on neural architectures for NER, and comparing to previous feature-based systems.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2145–2158
Santa Fe, New Mexico, USA, August 20-26, 2018.

We first discuss previous summary research on NER in section 2. Then we explain our selection criterion and methodology for selecting which systems to review in section 3. We highlight standard, past and recent NER datasets (from shared tasks and other research) in section 4 and evaluation metrics in section 5. We then describe NER systems in section 6 categorized into knowledge-based (section 6.1), bootstrapped (section 6.2), feature-engineered (section 6.3) and neural networks (section 6.4).

2 Previous surveys

The first comprehensive NER survey was Nadeau and Sekine (2007), which covered a variety of supervised, semi-supervised and unsupervised NER systems, highlighted common features used by NER systems during that time, and explained NER evaluation metrics that are still in use today. Sharnagat (2014) presented a more recent NER survey that also included supervised, semi-supervised, and unsupervised NER systems, and included a few introductory neural network NER systems. There have also been surveys focused on NER systems for specific domains and languages, including biomedical NER, (Leaman and Gonzalez, 2008), Chinese clinical NER (Lei et al., 2013), Arabic NER (Shaalan, 2014; Etaiwi et al., 2017), and NER for Indian languages (Patil et al., 2016).

The existing surveys primarily cover feature-engineered machine learning models (including supervised, semi-supervised, and unsupervised systems), and mostly focus on a single language or a single domain. There is not yet, to our knowledge, a comprehensive survey of modern neural network NER systems, nor is there a survey that compares feature engineered and neural network systems in both multi-lingual (CoNLL 2002 and CoNLL 2003) and multi-domain (e.g., news and medical) settings.

3 Methodology

To identify articles for this survey, we searched Google, Google Scholar, and Semantic Scholar. Our query terms included *named entity recognition*, *neural architectures for named entity recognition*, *neural network based named entity recognition models*, *deep learning models for named entity recognition*, etc. We sorted the papers returned from each query by citation count and read at least the top three, considering a paper for our survey if it either introduced a neural architecture for named entity recognition, or represented a top-performing model on an NER dataset. We included an article presenting a neural architecture only if it was the first article to introduce the architecture; otherwise, we traced citations back until we found the original source of the architecture. We followed the same approach for feature-engineering NER systems. We also included articles that implemented these systems for different languages or domain. In total, 154 articles were reviewed and 83 articles were selected for the survey.

4 NER datasets

Since the first shared task on NER (Grishman and Sundheim, 1996)[1], many shared tasks and datasets for NER have been created. CoNLL 2002 (Tjong Kim Sang, 2002)[2] and CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003)[3] were created from newswire articles in four different languages (Spanish, Dutch, English, and German) and focused on 4 entities - PER (person), LOC (location), ORG (organization) and MISC (miscellaneous including all other types of entities).

NER shared tasks have also been organized for a variety of other languages, including Indian languages (Rajeev Sangal and Singh, 2008), Arabic (Shaalan, 2014), German (Benikova et al., 2014), and slavic languages (Piskorski et al., 2017). The named entity types vary widely by source of dataset and language. For example, Rajeev Sangal and Singh (2008)'s southeast Asian language data has named entity types person, designation, temporal expressions, abbreviations, object number, brand, etc. Benikova et al. (2014)'s data, which is based on German wikipedia and online news, has named entity types similar to that of CoNLL 2002 and 2003: PERson, ORGanization, LOCation and OTHer. The shared task[4] or-

[1]Shared task: https://www-nlpir.nist.gov/related_projects/muc/
[2]Shared task: https://www.clips.uantwerpen.be/conll2002/ner/
[3]Shared task: https://www.clips.uantwerpen.be/conll2003/ner/
[4]Shared task: http://bsnlp.cs.helsinki.fi/shared_task.html

ganized by Piskorski et al. (2017) covering 7 slavic languages (Croatian, Czech, Polish, Russian, Slovak, Slovene, Ukrainian) also has person, location, organization and miscellaneous as named entity types.

In the biomedical domain, Kim et al. (2004) organized a BioNER task on MedLine abstracts, focusing on protien, DNA, RNA and cell attribute entity types. Uzuner et al. (2007) presented a clinical note de-identification task that required NER to locate personal patient data phrases to be anonymized. The 2010 I2B2 NER task[5] (Uzuner et al., 2011), which considered clinical data, focused on clinical problem, test and treatment entity types. Segura Bedmar et al. (2013) organized a Drug NER shared task[6] as part of SemEval 2013 Task 9, which focused on drug, brand, group and drug_n (unapproved or new drugs) entity types. (Krallinger et al., 2015) introduced the similar CHEMDNER task [7] focusing more on chemical and drug entities like trivial, systematic, abbreviation, formula, family, identifier, etc. Biology and microbiology NER datasets[8] (Hirschman et al., 2005; Bossy et al., 2013; Delėger et al., 2016) have been collected from PubMed and biology websites, and focus mostly on bacteria, habitat and geolocation entities. In biomedical NER systems, segmentation of clinical and drug entities is considered to be a difficult task because of complex orthographic structures of named entities (Liu et al., 2015).

NER tasks have also been organized on social media data, e.g., Twitter, where the performance of classic NER systems degrades due to issues like variability in orthography and presence of grammatically incomplete sentences (Baldwin et al., 2015). Entity types on Twitter are also more variable (person, company, facility, band, sportsteam, movie, TV show, etc.) as they are based on user behavior on Twitter.

Though most named entity annotations are flat, some datasets include more complex structures. Ohta et al. (2002) constructed a dataset of nested named entities, where one named entity can contain another. Strassel et al. (2003) highlighted both entity and entity head phrases. And discontinuous entities are common in chemical and clinical NER datasets (Krallinger et al., 2015). Eltyeb and Salim (2014) presented an survey of various NER systems developed for such NER datasets with a focus on chemical NER.

5 NER evaluation metrics

Grishman and Sundheim (1996) scored NER performance based on *type*, whether the predicted label was correct regardless of entity boundaries, and *text*, whether the predicted entity boundaries were correct regardless of the label. For each score category, *precision* was defined as the number of entities a system predicted correctly divided by the number that the system predicted, recall was defined as the number of entities a system predicted correctly divided by the number that were identified by the human annotators, and (micro) F-score was defined as the harmonic mean of precision and recall from both type and text.

The *exact match* metrics introduced by CoNLL (Tjong Kim Sang and De Meulder, 2003; Tjong Kim Sang, 2002) considers a prediction to be correct only when the predicted label for the complete entity is matched to exactly the same words as the gold label of that entity. CoNLL also used (micro) F-score, taking the harmonic mean of the exact match precision and recall.

The *relaxed F1* and *strict F1* metrics have been used in many NER shared tasks (Segura Bedmar et al., 2013; Krallinger et al., 2015; Bossy et al., 2013; Delėger et al., 2016). Relaxed F1 considers a prediction to be correct as long as part of the named entity is identified correctly. Strict F1 requires the character offsets of a prediction and the human annotation to match exactly. In these data, unlike CoNLL, word offsets are not given, so relaxed F1 is intended to allow comparison despite different systems having different word boundaries due to different segmentation techniques (Liu et al., 2015).

6 NER systems

6.1 Knowledge-based systems

Knowledge-based NER systems do not require annotated training data as they rely on lexicon resources and domain specific knowledge. These work well when the lexicon is exhaustive, but fail, for example, on every example of the drug_n class in the DrugNER dataset (Segura Bedmar et al., 2013), since drug_n

[5]Shared task: `https://www.i2b2.org/NLP/Relations/`
[6]Shared task: `https://www.cs.york.ac.uk/semeval-2013/task9/index.html`
[7]Similar datasets can be found here: `http://www.biocreative.org`
[8]Shared task: `http://2016.bionlp-st.org/tasks/bb2`

is defined as unapproved or new drugs, which are by definition not in the DrugBank dictionaries (Knox et al., 2010). Precision is generally high for knowledge-based NER systems because of the lexicons, but recall is often low due to domain and language-specific rules and incomplete dictionaries. Another drawback of knowledge based NER systems is the need of domain experts for constructing and maintaining the knowledge resources.

6.2 Unsupervised and bootstrapped systems

Some of the earliest systems required very minimal training data. Collins and Singer (1999) used only labeled seeds, and 7 features including orthography (e.g., capitalization), context of the entity, words contained within named entities, etc. for classifying and extracting named entities. Etzioni et al. (2005) proposed an unsupervised system to improve the recall of NER systems applying 8 generic pattern extractors to open web text, e.g., *NP is a <class1>*, *NP1 such as NPList2*. Nadeau et al. (2006) presented an unsupervised system for gazetteer building and named entity ambiguity resolution based on Etzioni et al. (2005) and Collins and Singer (1999) that combined an extracted gazetteer with commonly available gazetteers to achieve F-scores of 88%, 61%, and 59% on MUC-7 (Chinchor and Robinson, 1997) location, person, and organization entities, respectively.

Zhang and Elhadad (2013) used shallow syntactic knowledge and inverse document frequency (IDF) for an unsupervised NER system on biology (Kim et al., 2004) and medical (Uzuner et al., 2011) data, achieving 53.8% and 69.5% accuracy, respectively. Their model uses seeds to discover text having potential named entities, detects noun phrases and filters any with low IDF values, and feeds the filtered list to a classifier (Alfonseca and Manandhar, 2002) to predict named entity tags.

6.3 Feature-engineered supervised systems

Supervised machine learning models learn to make predictions by training on example inputs and their expected outputs, and can be used to replace human curated rules. Hidden Markov Models (HMM), Support Vector Machines (SVM), Conditional Random Fields (CRF), and decision trees were common machine learning systems for NER.

Zhou and Su (2002) used HMM (Rabiner and Juang, 1986; Bikel et al., 1997) an NER system on MUC-6 and MUC-7 data, achieving 96.6% and 94.1% F score, respectively. They included 11 orthographic features (1 numeral, 2 numeral, 4 numeral, all caps, numerals and alphabets, contains underscore or not, etc.) a list of trigger words for the named entities (e.g., 36 trigger words and affixes, like *river*, for the location entity class), and a list of words (10000 for the person entity class) from various gazetteers.

Malouf (2002) compared the HMM with Maximum Entropy (ME) by adding multiple features. Their best model included capitalization, whether a word was the first in a sentence, whether a word had appeared before with a known last name, and 13281 first names collected from various dictionaries. The model achieved 73.66%, 68.08% Fscore on Spanish and Dutch CoNLL 2002 dataset respectively.

The winner of CoNLL 2002 (Carreras et al., 2002) used binary AdaBoost classifiers, a boosting algorithm that combines small fixed-depth decision trees (Schapire, 2013). They used features like capitalization, trigger words, previous tag prediction, bag of words, gazetteers, etc. to represent simple binary relations and these relations were used in conjunction with previously predicted labels. They achieved 81.39% and 77.05% F scores on the Spanish and Dutch CoNLL 2002 datasets, respectively.

Li et al. (2005) implemented a SVM model on the CoNLL 2003 dataset and CMU seminar documents. They experimented with multiple window sizes, features (orthographic, prefixes suffixes, labels, etc.) from neighboring words, weighting neighboring word features according to their position, and class weights to balance positive and negative class. They used two SVM classifiers, one for detecting named entity starts and one for detecting ends. They achieved 88.3% F score on the English CoNLL 2003 data.

On the MUC6 data, Takeuchi and Collier (2002) used part-of-speech (POS) tags, orthographic features, a window of 3 words to the left and to the right of the central word, and tags of the last 3 words as features to the SVM. The final tag was decided by the voting of multiple one-vs-one SVM outputs.

Ando and Zhang (2005a) implemented structural learning (Ando and Zhang, 2005b) to divide the main task into many auxiliary tasks, for example, predicting labels by looking just at the context and masking

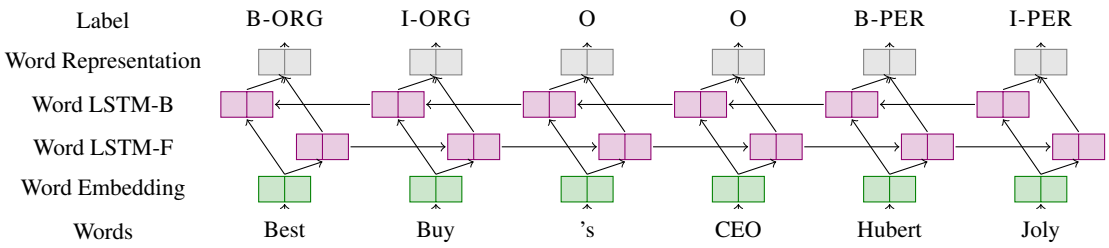


Figure 1: Word level NN architecture for NER

the current word. The best classifier for each auxiliary task was selected based on its confidence. This model had achieved 89.31% and 75.27% F score on English and German, respectively.

Agerri and Rigau (2016) developed a semi-supervised system[9] by presenting NER classifiers with features including orthography, character n-grams, lexicons, prefixes, suffixes, bigrams, trigrams, and unsupervised cluster features from the Brown corpus, Clark corpus and k-means clustering of open text using word embeddings (Mikolov et al., 2013). They achieved near state of the art performance on CoNLL datasets: 84.16%, 85.04%, 91.36%, 76.42% on Spanish, Dutch, English, and German, respectively.

In DrugNER (Segura Bedmar et al., 2013), Liu et al. (2015) achieved state-of-the-art results by using a CRF with features like lexicon resources from Food and Drug Administration (FDA), DrugBank, Jochem (Hettne et al., 2009) and word embeddings (trained on a MedLine corpus). For the same task, Rocktäschel et al. (2013) used a CRF with features constructed from dictionaries (e.g., Jochem (Hettne et al., 2009)), ontologies (ChEBI ontologies), prefixes-suffixes from chemical entities, etc.

6.4 Feature-inferring neural network systems

Collobert and Weston (2008) proposed one of the first neural network architectures for NER, with feature vectors constructed from orthographic features (e.g., capitalization of the first character), dictionaries and lexicons. Later work replaced these manually constructed feature vectors with word embeddings (Collobert et al., 2011), which are representations of words in n-dimensional space, typically learned over large collections of unlabeled data through an unsupervised process such as the skip-gram model (Mikolov et al., 2013). Studies have shown the importance of such pre-trained word embeddings for neural network based NER systems (Habibi et al., 2017), and similarly for pre-trained character embeddings in character-based languages like Chinese (Li et al., 2015; Yin et al., 2016).

Modern neural architectures for NER can be broadly classified into categories depending upon their representation of the words in a sentence. For example, representations may be based on words, characters, other sub-word units or any combination of these.

6.4.1 Word level architectures

In this architecture, the words of a sentence are given as input to Recurrent Neural Networks (RNN) and each word is represented by its word embedding, as shown in Figure 1.

The first word-level NN model was proposed by Collobert et al. (2011)[10]. The architecture was similar to the one shown in Figure 1, but a convolution layer was used instead of the Bi-LSTM layer and the output of the convolution layer was given to a CRF layer for the final prediction. The authors achieved 89.59% F1 score on English CoNLL 2003 dataset by including gazetteers and SENNA embeddings.

Huang et al. (2015) presented a word LSTM model (Figure 1) and showed that adding a CRF layer to the top of the word LSTM improved performance, achieving 84.26% F1 score on English CoNLL 2003 dataset. Similar systems were applied to other domains: DrugNER by Chalapathy et al. (2016) achieving 85.19% F1 score (under an unofficial evaluation) on MedLine test data (Segura Bedmar et al., 2013), and medical NER by Xu et al. (2017) achieving 80.22% F1 on disease NER corpus using this architecture. In similar tasks, Plank et al. (2016) implemented the same model for multilingual POS tagging.

[9]Code: `https://github.com/ixa-ehu/ixa-pipe-nerc`
[10]Code: `https://ronan.collobert.com/senna/`

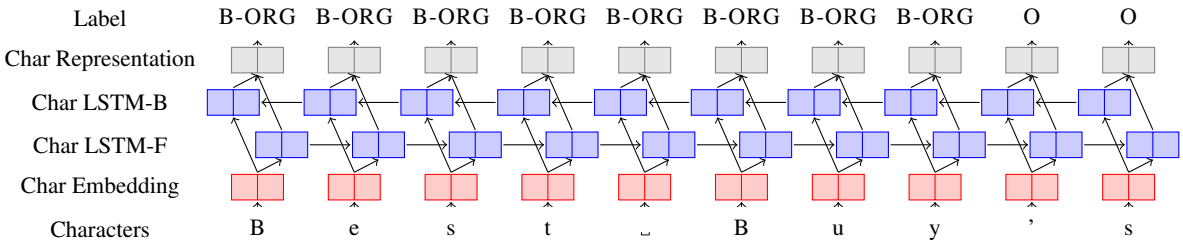


Figure 2: Character level NN architecture for NER

With slight variations, Yan et al. (2016) implemented word level feed forward NN, bi-directional LSTM (bi-LSTM) and window bi-LSTM for NER of English, German and Arabic. They also highlighted the performance improvement after adding various features like CRF, case, POS, word embeddings and achieved 88.91% F1 score on English and 76.12% on German.

6.4.2 Character level architectures

In this model, a sentence is taken to be a sequence of characters. This sequence is passed through an RNN, predicting labels for each character (Figure 2). Character labels transformed into word labels via post processing. The potential of character NER neural models was first highlighted by Kim et al. (2016) using highway networks over convolution neural networks (CNN) on character sequences of words and then using another layer of LSTM + softmax for the final predictions.

This model was implemented by Pham and Le-Hong (2017) for Vietnamese NER and achieved 80.23% F-score on Nguyen et al. (2016)'s Vietnamese test data. Character models were also used in various other languages like Chinese (Dong et al., 2016) where it has achieved near state of the art performance.

Kuru et al. (2016) proposed CharNER [11] which implemented the character RNN model for NER on 7 different languages. In this character model, tag prediction over characters were converted to word tags using Viterbi decoder(Forney, 1973) achieving 82.18% on Spanish, 79.36% on Dutch, 84.52% on English and 70.12% on German CoNLL datasets. They also achieved 78.72 on Arabic, 72.19 on Czech and 91.30 on Turkish. Ling et al. (2015) proposed word representation using RNN (Bi-LSTM) over characters of the word and achieved state of the art results on POS task using this representation in multiple languages including 97.78% accuracy on English PTB(Marcus et al., 1993).

Gillick et al. (2015) implemented sequence to sequence model (Byte to Span- BTS) using encoder decoder architecture over sequence of characters of words in a window of 60 characters. Each character was encoded in bytes and BTS achieved high performance on CoNLL 2002 and 2003 dataset without any feature engineering. BTS achieved 82.95%, 82.84%,86.50%,76.22% Fscore on Spanish, Dutch, English and German CoNLL datasets respectively.

6.4.3 Character+Word level architectures

Systems combining word context and the characters of a word have proved to be strong NER systems that need little domain specific knowledge or resources. There are two base models in this category. The **first type of model** represents words as a combination of a word embedding and a convolution over the characters of the word, follows this with a Bi-LSTM layer over the word representations of a sentence, and finally uses a softmax or CRF layer over the Bi-LSTM to generate labels. The architecture diagram for this model is same as Figure 3 but with the character Bi-LSTM replaced with a CNN[12].

Ma and Hovy (2016) implemented this model to achieve 91.21% F1 score on the CoNLL 2003 English dataset and 97.55% POS-tagging accuracy on the WSJ portion of PTB (Marcus et al., 1993). They also showed lower performance by this model for out of vocabulary words.

Chiu and Nichols (2015) achieved 91.62% F1 score on the CoNLL 2003 English dataset and 86.28% F score on Onto notes 5.0 dataset (Pradhan et al., 2013) by adding lexicons and capitalization features to

[11]Code: `https://github.com/ozanarkancan/char-ner`
[12]Code: `https://github.com/LopezGG/NN_NER_tensorFlow`

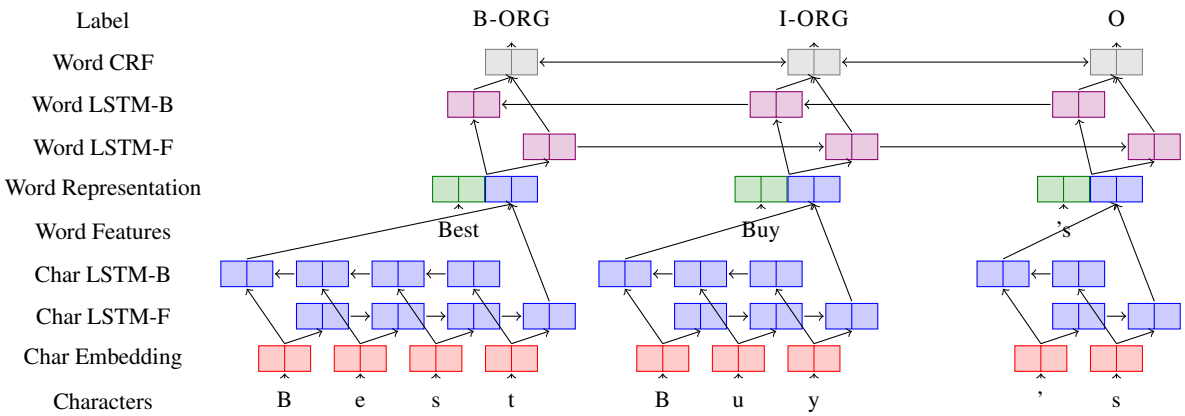


Figure 3: Word+character level NN architecture for NER

this model. Lexicon feature were encoded in the form or B(begin), I(inside) or E(end) PER, LOC, ORG and MISC depending upon the match from the dictionary.

This model has also been utilized for NER in languages like Japanese where Misawa et al. (2017) showed that this architecture outperformed other neural architectures on the *organization* entity class.

Limsopatham and Collier (2016) implemented a character+word level NER model for Twitter NER (Baldwin et al., 2015) by concatenating a CNN over characters, a CNN over orthographic features of characters, a word embedding, and a word orthographic feature embedding. This concatenated representation is passed through another Bi-LSTM layer and the output is given to CRF for predicting. This model achieved 65.89% F score on segmentation alone and 52.41% F score on segmentation and categorization.

Santos and Guimaraes (2015) implemented a model with a CNN over the characters of word, concatenated with word embeddings of the central word and its neighbors, fed to a feed forward network, and followed by the Viterbi algorithm to predict labels for each word. The model achieved 82.21% F score on Spanish CoNLL 2002 data and 71.23% F score on Portuguese NER data (Santos and Cardoso, 2007).

The **second type of model** concatenates word embeddings with LSTMs (sometimes bi-directional) over the characters of a word, passing this representation through another sentence-level Bi-LSTM, and predicting the final tags using a final softmax or CRF layer (Figure 3). Lample et al. (2016)[13] introduced this architecture and achieved 85.75%, 81.74%, 90.94%, 78.76% Fscores on Spanish, Dutch, English and German NER dataset respectively from CoNLL 2002 and 2003.

Dernoncourt et al. (2017) implemented this model in the NeuroNER toolkit[14] with the main goal of providing easy usability and allowing easy plotting of real time performance and learning statistics of the model. The BRAT annotation tool[15] is also integrated with NeuroNER to ease the development of NN NER models in new domains. NeuroNER achieved 90.50% F score on the English CoNLL 2003 data.

Habibi et al. (2017) implemented the model for various biomedical NER tasks and achieved higher performance than the majority of other participants. For example, they achieved 83.71 F-score on the CHEMDNER data (Krallinger et al., 2015).

Bharadwaj et al. (2016)[16] utilized phonemes (from Epitran) for NER in addition to characters and words. They also utilize attention knowledge over sequence of characters in word which is concatenated with the word embedding and character representation of word. This model achieved state of the art performance (85.81% F score) on Spanish CoNLL 2002 dataset.

A slightly improved system focusing on multi-task and multi-lingual joint learning was proposed by Yang et al. (2016) where word representation given by GRU (Gated Recurrent Unit) cell over characters plus word embedding was passed through another RNN layer and the output was given to CRF models trained for different tasks like POS, chunking and NER. Yang et al. (2017) further proposed transfer

[13]Code: `https://github.com/glample/tagger`
[14]Code: `http://neuroner.com`
[15]Code: `http://brat.nlplab.org/`
[16]Code: `https://github.com/dmort27/epitran`

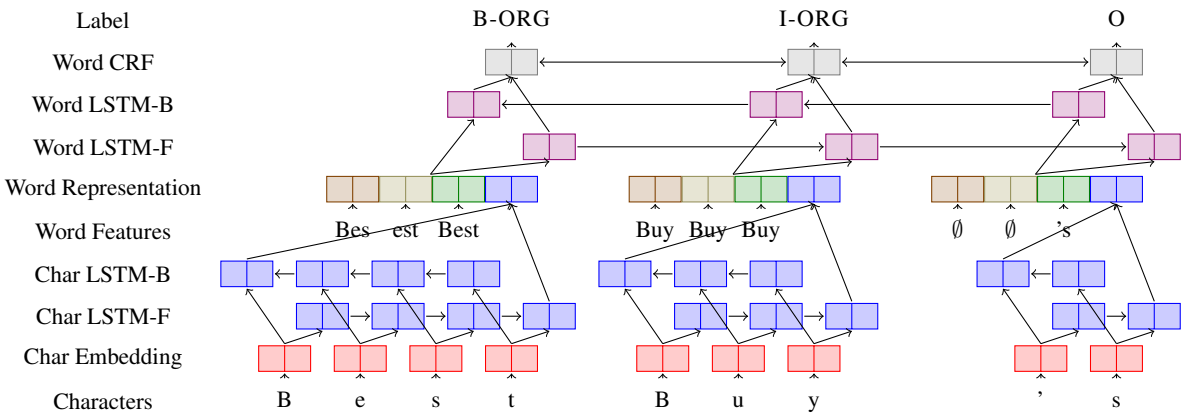


Figure 4: Word+character+affix level NN architecture for NER

learning for multi-task and multi-learning, and showed small improvements on CoNLL 2002 and 2003 NER data, achieving 85.77%, 85.19%, 91.26% F scores on Spanish, Dutch and English, respectively.

6.4.4 Character + Word + affix model

Yadav et al. (2018) implemented a model that augments the character+word NN architecture with one of the most successful features from feature-engineering approaches: affixes. Affix features were used in early NER systems for CoNLL 2002 (Tjong Kim Sang, 2002; Cucerzan and Yarowsky, 2002) and 2003 (Tjong Kim Sang and De Meulder, 2003) and for biomedical NER (Saha et al., 2009), but had not been used in neural NER systems. They extended the Lample et al. (2016) character+word model to learn affix embeddings[17] alongside the word embeddings and character RNNs (Figure 4). They considered all n-gram prefixes and suffixes of words in the training corpus, and selected only those whose frequency was above a threshold, T. Their word+character+affix model achieved 87.26%, 87.54%, 90.86%, 79.01% on Spanish, Dutch, English and German CoNLL datasets respectively. Yadav et al. (2018) also showed that affix embeddings capture complementary information to that captured by RNNs over the characters of a word, that selecting only high frequency (realistic) affixes was important, and that embedding affixes was better than simply expanding the other embeddings to reach a similar number of hyper-parameters.

7 Discussion

Table 1 shows the results of all the different categories of systems discussed in section 6 on the CoNLL 2002 and 2003 datasets. The table also indicates, for each model, whether it makes use of external knowledge like a dictionary or gazetteer. Table 2 presents a similar analysis on the DrugNER dataset from SemEval 2013 task 9 (Segura Bedmar et al., 2013).

Our first finding from the survey is that feature-inferring NN systems outperform feature-engineered systems, despite the latter's access to domain specific rules, knowledge, features, and lexicons. For example, the best feature-engineered system for Spanish, Agerri and Rigau (2016), is 1.59% below the best feature-inferring neural network system, (Lample et al., 2016), and 1.65% below the best neural network system that incorporates lexical resources (Bharadwaj et al., 2016). Similarly, the best feature-engineered system for German, Agerri and Rigau (2016), is 2.34% below the best feature-inferring neural network system, Lample et al. (2016). The differences are smaller for Dutch and English, but in neither case is the best feature-engineered model better than the best neural network model. In DrugNER, the word+character NN model outperforms the feature engineered system by 8.90% on MedLine test data and 3.50% on the overall dataset.

Our next finding is that word+character hybrid models are generally better than both word-based and character-based models. For example, the best hybrid NN model for English, Chiu and Nichols (2015), is 0.52% better than the best word-based model, Huang et al. (2015), and 5.12% better than the best character-based model, (Kuru et al., 2016). Similarly, the best hybrid NN model for German, Lample et

[17]Code: `https://github.com/vikas95/Pref_Suff_Span_NN`

Feature-engineered machine learning systems	Dict	SP	DU	EN	GE
Carreras et al. (2002) binary AdaBoost classifiers	Yes	81.39	77.05	-	-
Malouf (2002) - Maximum Entropy (ME) + features	Yes	73.66	68.08	-	-
Li et al. (2005) SVM with class weights	Yes	-	-	88.3	-
Passos et al. (2014) CRF	Yes	-	-	90.90	-
Ando and Zhang (2005a) Semi-supervised state of the art	No	-	-	89.31	75.27
Agerri and Rigau (2016)	Yes	**84.16**	**85.04**	**91.36**	**76.42**
Feature-inferring neural network word models					
Collobert et al. (2011) Vanilla NN +SLL / Conv-CRF	No	-	-	81.47	-
Huang et al. (2015) Bi-LSTM+CRF	No	-	-	84.26	-
Yan et al. (2016) Win-BiLSTM (English), FF (German) (Many fets)	Yes	-	-	88.91	**76.12**
Collobert et al. (2011) Conv-CRF (SENNA+Gazetteer)	Yes	-	-	89.59	-
Huang et al. (2015) Bi-LSTM+CRF+ (SENNA+Gazetteer)	Yes	-	-	**90.10**	-
Feature-inferring neural network character models					
Gillick et al. (2015) – BTS	No	**82.95**	**82.84**	**86.50**	**76.22**
Kuru et al. (2016) CharNER	No	82.18	79.36	84.52	70.12
Feature-inferring neural network word + character models					
Yang et al. (2017)	Yes	85.77	**85.19**	91.26	-
Luo (2015)	Yes	-	-	91.20	-
Chiu and Nichols (2015)	Yes	-	-	**91.62**	-
Ma and Hovy (2016)	No	-	-	91.21	-
Santos and Guimaraes (2015)	No	82.21	-	-	-
Lample et al. (2016)	No	85.75	81.74	90.94	**78.76**
Bharadwaj et al. (2016)	Yes	**85.81**	-	-	-
Dernoncourt et al. (2017)	No	-	-	90.5	-
Feature-inferring neural network word + character + affix models					
Re-implementation of Lample et al. (2016) (100 Epochs)	No	85.34	85.27	90.24	78.44
Yadav et al. (2018)(100 Epochs)	No	86.92	87.50	90.69	78.56
Yadav et al. (2018) (150 Epochs)	No	**87.26**	**87.54**	90.86	**79.01**

Table 1: Comparison of NER systemsin four languages: CoNLL 2002 Spanish (SP), CoNLL 2002 Dutch (DU), CoNLL 2003 English (EN), and CoNLL 2003 German (GE). Dict indicates whether or not the approach makes use of dictionary lookups. Best performance in each category is highlighted in bold.

		MedLine (80.10%)			DrugBank (19.90%)			Complete dataset		
	Dict	P	R	F1	P	R	F1	P	R	F1
Feature-engineered machine learning systems										
Rocktäschel et al. (2013)	Yes	60.70	55.80	58.10	88.10	87.50	87.80	73.40	69.80	71.50
Liu et al. (2015) (baseline)	No	-	-	-	-	-	-	78.41	67.78	72.71
Liu et al. (2015) (MED. emb.)	No	-	-	-	-	-	-	82.70	69.68	75.63
Liu et al. (2015) (state of the art)	Yes	78.77	60.21	68.25	90.60	88.82	**89.70**	84.75	72.89	**78.37**
NN word model										
Chalapathy et al. (2016) (relaxed performance)	No	52.93	52.57	52.75	87.07	83.39	85.19	-	-	-
NN word + character model										
Yadav et al. (2018)	No	73	62	67	87	86	87	79	72	75
NN word + character + affix model										
Yadav et al. (2018)	No	74	64	**69**	89	86	87	81	74	77

Table 2: DrugNER results on the MedLine and DrugBank test data (80.10% and 19.90% of the test data, respectively). The Yadav et al. (2018) experiments report no decimal places because they were run after the end of shared task, and the official evaluation script outputs no decimal places.

al. (2016), is 2.64% better than the best word-based model, Yan et al. (2016), and 2.54% better than the best character-based model, (Kuru et al., 2016). In DrugNER, the word+character hybrid model is better than the word model by 14.25% on MedLine test data and 1.81% on DrugBank test data.

Our final finding is that there is still interesting progress to be made by incorporating key features of past feature-engineered models into modern NN architectures. Yadav et al. (2018)'s simple extension

of Lample et al. (2016) to incorporate affix features yields a very strong new model, achieving a new state-of-the-art in Spanish, Dutch, and German, and performing within 1% of the best model for English.

8 Conclusion

Our survey of models for named entity recognition, covering both classic feature-engineered machine learning models, and modern feature-inferring neural network models has yielded several important insights. Neural network models generally outperform feature-engineered models, character+word hybrid neural networks generally outperform other representational choices, and further improvements are available by applying past insights to current neural network models, as shown by the state-of-the-art performance of our proposed affix-based extension of character+word hybrid models.

References

Rodrigo Agerri and German Rigau. 2016. Robust multilingual named entity recognition with shallow semi-supervised features. *Artificial Intelligence*, 238:63–82.

Enrique Alfonseca and Suresh Manandhar. 2002. An unsupervised method for general named entity recognition and automated concept discovery. In *Proceedings of the 1st international conference on general WordNet, Mysore, India*, pages 34–43.

Rie Kubota Ando and Tong Zhang. 2005a. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853.

Rie Kubota Ando and Tong Zhang. 2005b. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853.

Timothy Baldwin, Marie-Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 126–135.

Darina Benikova, Chris Biemann, and Marc Reznicek. 2014. Nosta-d named entity annotation for german: Guidelines and dataset. In *LREC*, pages 2524–2531.

Akash Bharadwaj, David R. Mortensen, Chris Dyer, and Carlos de Juan Carbonell. 2016. Phonologically aware neural model for named entity recognition in low resource transfer settings. In *EMNLP*.

Daniel M Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. 1997. Nymble: a high-performance learning name-finder. In *Proceedings of the fifth conference on Applied natural language processing*, pages 194–201. Association for Computational Linguistics.

Robert Bossy, Wiktoria Golik, Zorana Ratkovic, Philippe Bessières, and Claire Nédellec. 2013. Bionlp shared task 2013–an overview of the bacteria biotope task. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 161–169.

Xavier Carreras, Lluís Màrquez, and Lluís Padró. 2002. Named entity extraction using adaboost, proceedings of the 6th conference on natural language learning. *August*, 31:1–4.

Raghavendra Chalapathy, Ehsan Zare Borzeshi, and Massimo Piccardi. 2016. An investigation of recurrent neural architectures for drug name recognition. *arXiv preprint arXiv:1609.07585*.

Nancy Chinchor and Patricia Robinson. 1997. Muc-7 named entity task definition. In *Proceedings of the 7th Conference on Message Understanding*, volume 29.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

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

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Silviu Cucerzan and David Yarowsky. 2002. Language independent ner using a unified model of internal and contextual evidence. In *proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–4. Association for Computational Linguistics.

Louise Delėger, Robert Bossy, Estelle Chaix, Mouhamadou Ba, Arnaud Ferrė, Philippe Bessieres, and Claire Nėdellec. 2016. Overview of the bacteria biotope task at bionlp shared task 2016. In *Proceedings of the 4th BioNLP Shared Task Workshop*, pages 12–22.

Franck Dernoncourt, Ji Young Lee, and Peter Szolovits. 2017. Neuroner: an easy-to-use program for named-entity recognition based on neural networks. *arXiv preprint arXiv:1705.05487*.

Chuanhai Dong, Jiajun Zhang, Chengqing Zong, Masanori Hattori, and Hui Di. 2016. Character-based lstm-crf with radical-level features for chinese named entity recognition. In *Natural Language Understanding and Intelligent Applications*, pages 239–250. Springer.

Safaa Eltyeb and Naomie Salim. 2014. Chemical named entities recognition: a review on approaches and applications. *Journal of cheminformatics*, 6(1):17.

Wael Etaiwi, Arafat Awajan, and Dima Suleiman. 2017. Statistical arabic name entity recognition approaches: A survey. *Procedia Computer Science*, 113:57–64.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134.

G David Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2015. Multilingual language processing from bytes. *arXiv preprint arXiv:1512.00103*.

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference-6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, volume 1.

Maryam Habibi, Leon Weber, Mariana Neves, David Luis Wiegandt, and Ulf Leser. 2017. Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics*, 33(14):i37–i48.

Kristina M Hettne, Rob H Stierum, Martijn J Schuemie, Peter JM Hendriksen, Bob JA Schijvenaars, Erik M van Mulligen, Jos Kleinjans, and Jan A Kors. 2009. A dictionary to identify small molecules and drugs in free text. *Bioinformatics*, 25(22):2983–2991.

Lynette Hirschman, Alexander Yeh, Christian Blaschke, and Alfonso Valencia. 2005. Overview of biocreative: critical assessment of information extraction for biology.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the bio-entity recognition task at jnlpba. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*, pages 70–75. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

Craig Knox, Vivian Law, Timothy Jewison, Philip Liu, Son Ly, Alex Frolkis, Allison Pon, Kelly Banco, Christine Mak, Vanessa Neveu, et al. 2010. Drugbank 3.0: a comprehensive resource for omics research on drugs. *Nucleic acids research*, 39(suppl_1):D1035–D1041.

Martin Krallinger, Obdulia Rabal, Florian Leitner, Miguel Vazquez, David Salgado, Zhiyong Lu, Robert Leaman, Yanan Lu, Donghong Ji, Daniel M Lowe, et al. 2015. The chemdner corpus of chemicals and drugs and its annotation principles. *Journal of cheminformatics*, 7(S1):S2.

Onur Kuru, Ozan Arkan Can, and Deniz Yuret. 2016. Charner: Character-level named entity recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 911–921.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Robert Leaman and Graciela Gonzalez. 2008. Banner: an executable survey of advances in biomedical named entity recognition. In *Biocomputing 2008*, pages 652–663. World Scientific.

Jianbo Lei, Buzhou Tang, Xueqin Lu, Kaihua Gao, Min Jiang, and Hua Xu. 2013. A comprehensive study of named entity recognition in chinese clinical text. *Journal of the American Medical Informatics Association*, 21(5):808–814.

Yaoyong Li, Kalina Bontcheva, and Hamish Cunningham. 2005. Svm based learning system for information extraction. In *Deterministic and statistical methods in machine learning*, pages 319–339. Springer.

Yanran Li, Wenjie Li, Fei Sun, and Sujian Li. 2015. Component-enhanced chinese character embeddings. *arXiv preprint arXiv:1508.06669*.

Nut Limsopatham and Nigel Henry Collier. 2016. Bidirectional lstm for named entity recognition in twitter messages.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

Shengyu Liu, Buzhou Tang, Qingcai Chen, and Xiaolong Wang. 2015. Effects of semantic features on machine learning-based drug name recognition systems: word embeddings vs. manually constructed dictionaries. *Information*, 6(4):848–865.

2015. *Joint Named Entity Recognition and Disambiguation*, September.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Robert Malouf. 2002. Markov models for language-independent named entity recognition, proceedings of the 6th conference on natural language learning. *August*, 31:1–4.

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

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Shotaro Misawa, Motoki Taniguchi, Yasuhide Miura, and Tomoko Ohkuma. 2017. Character-based bidirectional lstm-crf with words and characters for japanese named entity recognition. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, pages 97–102.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

David Nadeau, Peter D Turney, and Stan Matwin. 2006. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 266–277. Springer.

TS Nguyen, LM Nguyen, and XC Tran. 2016. Vietnamese named entity recognition at vlsp 2016 evaluation campaign. In *Proceedings of The Fourth International Workshop on Vietnamese Language and Speech Processing*.

Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. 2002. The genia corpus: An annotated research abstract corpus in molecular biology domain. In *Proceedings of the second international conference on Human Language Technology Research*, pages 82–86. Morgan Kaufmann Publishers Inc.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *arXiv preprint arXiv:1404.5367*.

Nita Patil, Ajay S Patil, and BV Pawar. 2016. Survey of named entity recognition systems with respect to indian and foreign languages. *International Journal of Computer Applications*, 134(16).

Thai-Hoang Pham and Phuong Le-Hong. 2017. End-to-end recurrent neural network models for vietnamese named entity recognition: Word-level vs. character-level. *arXiv preprint arXiv:1705.04044*.

Jakub Piskorski, Lidia Pivovarova, Jan Šnajder, Josef Steinberger, Roman Yangarber, et al. 2017. The first cross-lingual challenge on recognition, normalization and matching of named entities in slavic languages. In *Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing*. Association for Computational Linguistics.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv preprint arXiv:1604.05529*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152.

Lawrence Rabiner and B Juang. 1986. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16.

Dipti Misra Sharma Rajeev Sangal and Anil Kumar Singh, editors. 2008. *Proceedings of the IJCNLP-08 Workshop on Named Entity Recognition for South and South East Asian Languages*. Asian Federation of Natural Language Processing, Hyderabad, India, January.

Tim Rocktäschel, Torsten Huber, Michael Weidlich, and Ulf Leser. 2013. Wbi-ner: The impact of domain-specific features on the performance of identifying and classifying mentions of drugs. In *SemEval@ NAACL-HLT*, pages 356–363.

Sujan Kumar Saha, Sudeshna Sarkar, and Pabitra Mitra. 2009. Feature selection techniques for maximum entropy based biomedical named entity recognition. *Journal of Biomedical Informatics*, 42(5):905 – 911. Biomedical Natural Language Processing.

Diana Santos and Nuno Cardoso. 2007. Reconhecimento de entidades mencionadas em português: Documentação e actas do harem, a primeira avaliação conjunta na área.

Cicero Nogueira dos Santos and Victor Guimaraes. 2015. Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv:1505.05008*.

Robert E Schapire. 2013. Explaining adaboost. In *Empirical inference*, pages 37–52. Springer.

Isabel Segura Bedmar, Paloma Martínez, and María Herrero Zazo. 2013. Semeval-2013 task 9: Extraction of drug-drug interactions from biomedical texts (ddiextraction 2013). Association for Computational Linguistics.

Khaled Shaalan. 2014. A survey of arabic named entity recognition and classification. *Computational Linguistics*, 40(2):469–510.

Rahul Sharnagat. 2014. Named entity recognition: A literature survey. *Center For Indian Language Technology*.

Stephanie Strassel, Alexis Mitchell, and Shudong Huang. 2003. Multilingual resources for entity extraction. In *Proceedings of the ACL 2003 workshop on Multilingual and mixed-language named entity recognition-Volume 15*, pages 49–56. Association for Computational Linguistics.

Koichi Takeuchi and Nigel Collier. 2002. Use of support vector machines in extended named entity recognition. In *proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–7. Association for Computational Linguistics.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Erik F Tjong Kim Sang. 2002. Introduction to the conll-2002 shared task: language-independent named entity recognition, proceedings of the 6th conference on natural language learning. *August*, 31:1–4.

Özlem Uzuner, Yuan Luo, and Peter Szolovits. 2007. Evaluating the state-of-the-art in automatic de-identification. *Journal of the American Medical Informatics Association*, 14(5):550–563.

Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2011. 2010 i2b2/va challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association*, 18(5):552–556.

Kai Xu, Zhanfan Zhou, Tianyong Hao, and Wenyin Liu. 2017. A bidirectional lstm and conditional random fields approach to medical named entity recognition. In *International Conference on Advanced Intelligent Systems and Informatics*, pages 355–365. Springer.

Vikas Yadav, Rebecca Sharp, and Steven Bethard. 2018. Deep affix features improve neural named entity recognizers. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 167–172.

Shao Yan, Christian Hardmeier, and Joakim Nivre. 2016. Multilingual named entity recognition using hybrid neural networks. In *The Sixth Swedish Language Technology Conference (SLTC)*.

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. *arXiv preprint arXiv:1703.06345*.

Rongchao Yin, Quan Wang, Peng Li, Rui Li, and Bin Wang. 2016. Multi-granularity chinese word embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 981–986.

Shaodian Zhang and Noémie Elhadad. 2013. Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts. *Journal of biomedical informatics*, 46(6):1088–1098.

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an hmm-based chunk tagger. In *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 473–480. Association for Computational Linguistics.

Distantly Supervised NER with Partial Annotation Learning and Reinforcement Learning

Yaosheng Yang, Wenliang Chen, Zhenghua Li, Zhengqiu He, Min Zhang
Institute of Artificial Intelligence
School of Computer Science and Technology, Soochow University, China
{ysyang, zqhe}@stu.suda.edu.cn
{wlchen, zhli13, minzhang}@suda.edu.cn

Abstract

A bottleneck problem with Chinese named entity recognition (NER) in new domains is the lack of annotated data. One solution is to utilize the method of distant supervision, which has been widely used in relation extraction, to automatically populate annotated training data without human-cost. The distant supervision assumption here is that if a string in text is included in a predefined dictionary of entities, the string might be an entity. However, this kind of auto-generated data suffers from two main problems: incomplete and noisy annotations, which affect the performance of NER models. In this paper, we propose a novel approach which can partially solve the above problems of distant supervision for NER. In our approach, to handle the incomplete problem, we apply partial annotation learning to reduce the effect of unknown labels of characters. As for noisy annotation, we design an instance selector based on reinforcement learning to distinguish positive sentences from auto-generated annotations. In experiments, we create two datasets for Chinese named entity recognition in two domains with the help of distant supervision. The experimental results show that the proposed approach obtains better performance than the comparison systems on both two datasets.

1 Introduction

In recent years, deep learning approaches have achieved great progress in the task of named entity recognition (NER) (Collobert et al., 2011; Chiu and Nichols, 2015). The standardized approach is that using BiLSTMs for encoding and then applying CRF for jointly label decoding (Huang et al., 2015; Lample et al., 2016). In addition, BiLSTMs and CNNs are employed to model character- or word-level representations (Ma and Hovy, 2016).

Most previous studies on NER focus on a certain set of predefined NER types, such as organization, location, person, date, and so on, where a certain amount of labeled data is provided to train the models. However, different applications require particular entity types, such as "Brand" and "Product" in E-commerce domain, and "Company" for finance industry. Considering the high cost of human annotation, it may not be feasible to annotate large amounts of labeled data for each new NER type, but small-scale data is available at some time.

As an alternative solution, distant supervision can automatically generate large-scale labeled data for new-type NER without human-cost. The idea of distant supervision has widely used in the task of relation extraction (Mintz et al., 2009; Riedel et al., 2010; Zeng et al., 2015). For relation extraction, at first we have a knowledge base. If two entities e_1 and e_2 have relation r according to the knowledge base, then we populate this knowledge and assume the relation between e_1 and e_2 is r in the sentences that contain the both entities. In this way, we can produce a lot of labeled data for model training.

Similarly, in our task, we first acquire a dictionary containing a list of the new-type entities. Then, we automatically generate large-scale labeled data by assuming that each entity mention in a sentence is a positive instance of the corresponding type according to the dictionary. Figure 1(a) shows an example

The corresponding author is Wenliang Chen.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2159–2169
Santa Fe, New Mexico, USA, August 20-26, 2018.

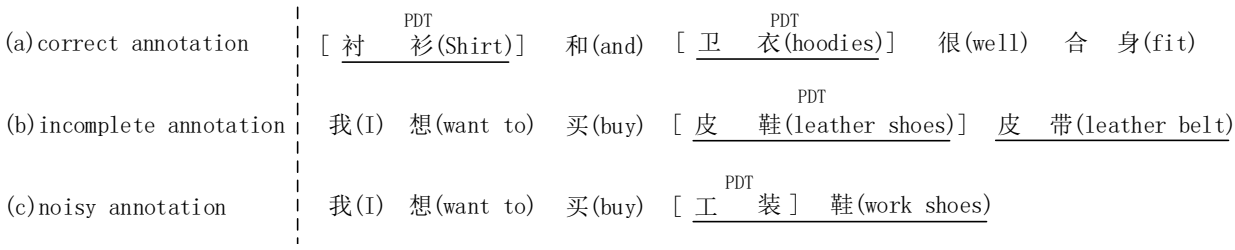


Figure 1: Examples generated by distant supervision, where "PDT" means a product name tagged by the distant supervision and one string with underline should be an entity of product.

that is regarded as a positive instance with two "Product" names are correctly matched by the distant supervision method.

However, in practice we find that the automatically labeled NER data suffers from two problems, i.e., *incomplete annotation* and *noisy annotation*, which negatively affect the performance of NER systems. The incomplete annotation problem means that not every entity has been labeled in the way of distant supervision. For example, "皮鞋(leather shoes)" is included in the dictionary, while "皮带(leather belt)" is not included. Thus in Figure 1(b), "皮鞋(leather shoes)" is annotated as a PDT, but "皮带(leather belt)" is not annotated. The noisy annotation problem means that the matched entity does not correspond to entity definition, such as Figure 1(c), where "工装鞋(work shoes)" is a product, but only the first two characters "工装(fatigue clothes)" are matched by the dictionary because "工装鞋(work shoes)" is not included in the dictionary. Obviously, such false labeled examples certainly provide wrong supervision during model training if we directly use the automatically generated data.

In this paper, we propose an approach to handle the two problems of distantly supervised NER data. As for the incomplete annotation problem, we treat the data as partially annotated data based on the extended CRF-PA model that can directly learn from partial annotations (PA) (Tsuboi et al., 2008). The noisy annotation problem is also ubiquitous in distantly supervised for relation extraction, and researchers try to address this issue by using reinforcement learning (RL) technology to select positive instances (Feng et al., 2018). Inspired of their work, we design an instance selector to obtain clean instances from distantly supervised NER data.

In summary, we make the following contributions:

- We propose a novel approach for new-type named entity recognition, which firstly combines the advantages of both partial annotation learning and reinforcement learning, to handle the problems of incomplete annotation and noisy annotation brought by distant supervision.
- We create two datasets for Chinese named entity recognition with the help of distant supervision in e-commerce and news domains. The experimental results on the newly created datasets show that the proposed approach performs better than the comparison systems.

2 Basic Settings

2.1 Distantly Supervised NER Data

Here we focus mainly on the Chinese NER, which is more difficult than NER for other languages such as English for the lack of morphological variations such as capitalization and in particular the uncertainty in word segmentation. To get a good tagger for new entity types in new domains, we perform distant supervision to acquire labeled data for Chinese NER.

Initially, we have a small set of labeled seed data $\mathcal{H}$ for new entity types, and large-scale unlabeled data pool $\mathcal{U}$. We collect named entities to construct dictionary $\mathcal{D}$, and use the entries of $\mathcal{D}$ to match the strings of the sentences in $\mathcal{U}$ by the method of distant supervision. Then we obtain a set of sentences containing at least one matched strings, and the set is denoted as $\mathcal{A}$. The purpose in this paper is that we make full use of $\mathcal{H}$ and $\mathcal{A}$ to build a NER system.

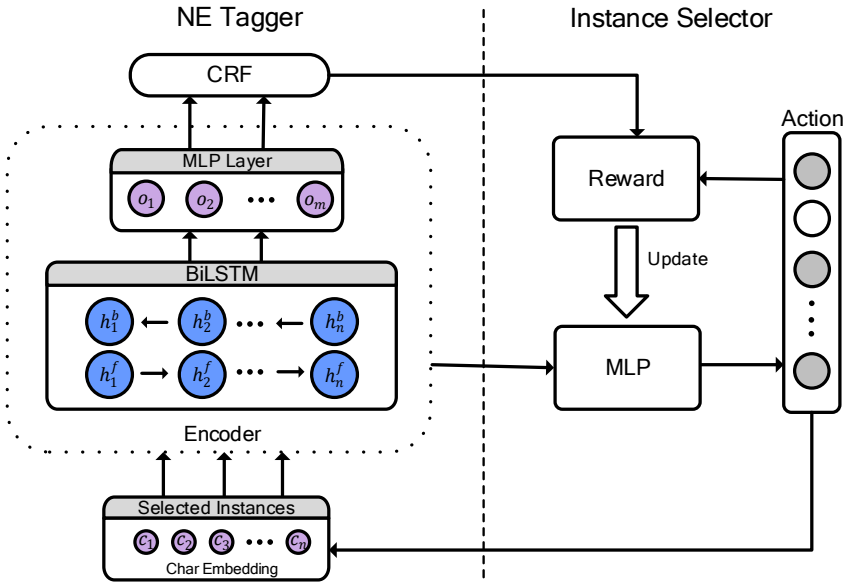


Figure 2: The framework of the proposed model, which consists of two parts. The right instance selector is a policy network, which chooses sentences from candidate dataset to expand training data to improve the left NE Tagger. The instance selector is trained based on the reward provided by NE Tagger.

In this paper, we treat Chinese NER task as a sequence labeling problem. We exploit the traditional BIO schema to represent the tags of sentences. Concretely, we tag the beginning character of an entity by "B-XX", the other characters of this entity by "I-XX", and the character as "O" if it is not inside an entity, where "XX" is the type of entities.

2.2 The Baseline LSTM-CRF

Given a sentence $x = c_1 c_2 \cdots c_n$, the goal is to assign an unique tag y_i for Chinese character c_i in the sentence. In general, the model predicts the entities in the sentence x by estimating the probability $p(y|x)$, where y is a possible label sequence for sentence x. The final output y_{max} of the system for one sentence is the label sequence with the maximum probability.

Here, we present a new NE tagger based on the LSTM-CRF model of Lample et al. (2016), which achieves the state-of-the-art performance in the NER task. Following Peng and Dredze (2015a), we represent Chinese characters as vectors and feed them into BiLSTM layer in the Chinese NER task. The left part of Figure 2 shows the framework of our baseline LSTM-CRF based NE tagger.

The input layer. For each input sentence $x = c_1 c_2 \cdots c_n$, we map serialized characters into a list of vectors $\mathbf{x}_1 \mathbf{x}_2 \cdots \mathbf{x}_n$ with an embedding layer including a lookup table as its key parameter. Following Lample et al. (2016), the lookup table is initialized with embeddings pre-trained on a large-scale raw corpus and is further fine-tuned during our training process.

The BiLSTM layer. With vector sequence $\mathbf{x}_1 \mathbf{x}_2 \cdots \mathbf{x}_n$ from input layer, we apply a bidirectional LSTM layer to encode the semantic dependency which provides high level features. The LSTM incorporates a memory-cell to solve the exploding and diminishing gradients of basic RNNs (Graves and Schmidhuber, 2005). For each character c_t in the sentence, we can obtain the output features $\mathbf{h}_t$ by concatenating $\overrightarrow{\mathbf{h}}_t$ and $\overleftarrow{\mathbf{h}}_t$, which computed in turn and in reverse to represent its left and right context. What's more, dropout (Srivastava et al., 2014) is applied to the outputs of the BiLSTM layer to avoid overfitting.

The MLP layer. Then, we employ a multi-layer perceptron (MLP), also known as feed-forward neural networks, to compute the scores of all labels at each position t, denoted as $\mathbf{o}_t$,

$$\begin{aligned} \mathbf{h}_t^{\text{mlp1}} &= \mathbf{W}^{\text{mlp1}} \mathbf{h}_t + \mathbf{b}^{\text{mlp1}} \\ \mathbf{o}_t &= \mathbf{W}^{\text{mlp2}} \mathbf{h}_t^{\text{mlp1}} + \mathbf{b}^{\text{mlp2}} \quad t \in [1, n] \end{aligned} \tag{1}$$

where $\mathbf{W}^{\text{mlp1/mlp2}}$ and $\mathbf{b}^{\text{mlp1/mlp2}}$ are the parameters.

The CRF layer. Finally, we adopt a CRF layer to derive the conditional probability of a label sequence y as follows:

$$\begin{aligned} p(y|x) &= \frac{e^{\text{score}(x,y)}}{\sum_{\bar{y}\in\mathbf{Y}_x} e^{\text{score}(x,\bar{y})}} \\ \text{score}(x,y) &= \sum_{t=1}^{n}(o_{t,y_t} + T_{y_{t-1},y_t}) \end{aligned} \tag{2}$$

where T_{y_{t-1},y_t} represents the transmission score from y_{t-1} to y_t, and $\mathbf{Y}_x$ are all candidate label sequences of input sentence x.

During evaluation, we adopt the minimum Bayes risk (MBR) decoding to find the optimal label sequence as follows:

$$\begin{aligned} y^* &= \arg\max_{y} \big(\sum_{t=1}^{n} p(y_t|x,t)\big) \\ p(y_t|x,t) &= \sum_{\bar{y}\in\mathbf{Y}_x,\bar{y}_t=y_t} p(\bar{y}|x) \end{aligned} \tag{3}$$

where $p(y_t|x,t)$ is the marginal probability of tagging the t-th position in x as y_t.

Training objective. In order to maximize the probability of the correct label sequence, we exploit a negative log-likelihood objective as the loss function:

$$\text{loss}(\Theta, x, y) = -\log p(y|x) \tag{4}$$

where Θ is the set of baseline model parameters.

3 Our Approach

This section describes our approach for new-type NER via distant supervision. To handle the problem of incomplete and noisy annotations, we propose a novel model for NER task. As shown in Figure 2, the framework of our model consists of two modules: the NE Tagger built on the idea of partial annotation learning to reduce the effect of unknown-type characters, the instance selector which chooses positive sentences from a candidate set and provides them to the NE Tagger.

3.1 LSTM-CRF-PA for Incomplete Annotation

It is inappropriate to regard those characters as non-entities, although they can not be matched according to the dictionary. It is a common problem known as false negative instance which may misguide model if we arbitrarily label them as "O". Therefore, we consider that each non-matched character can be annotated as any proper label. For example in Figure 3, except that "皮鞋 (leather shoes)" have definite labels, all of the remaining characters can be labeled as "B-PDT", "I-PDT" and so on. In other words, we denote a set of label sequences $\mathbf{z}$ for every distantly supervised sentence, whose probability is naturally the sum of probability of each possible label sequence $\tilde{y}$ in $\mathbf{z}$. We expand the original model for this situation and apply softmax over all candidate output label sequences, thus the probability of a distantly supervised instance is computed as follows:

$$p(\mathbf{z}|x) = \sum_{\tilde{y}\in\mathbf{z}} p(\tilde{y}|x) = \frac{\sum_{\tilde{y}\in\mathbf{z}} e^{\text{score}(x,\tilde{y})}}{\sum_{\bar{y}\in\mathbf{Y}_x} e^{\text{score}(x,\bar{y})}} \tag{5}$$

We exploit a negative log-likelihood objective as the loss function. Therefore, the loss function of our model with CRF-PA can be computed as follows:

$$\text{loss}(\Theta, x, \mathbf{z}) = -\log p(\mathbf{z}|x) \tag{6}$$

Figure 3: An example instance which is partially annotated. "PDT" means product name and "SPC" means specification name.

where Θ is the set of all NE tagger parameters.

In particular, if the sentence is annotated by hand and each character has definite label, the set $\mathbf{z}$ includes only one label sequence. Thus the above objective function is also suitable for supervised instances. We use standard back-propagation method to minimize the loss function of the NE tagger.

3.2 Instance Selector for Noisy Annotation

Our goal is to train an agent as an instance selector with reinforcement learning (RL) technology. Following Feng et al. (2018), the agent interacts with the environment and makes a decision at the sentence-level. We merge the initial hand-tagged seed set $\mathcal{H}$ and the distantly supervised set $\mathcal{A}$ into a candidate dataset $\mathcal{C}$. At each episode, we collect a random-size bag of instances $\mathcal{B}$ from $\mathcal{C}$. All the supervised instances in the current bag are default to be selected without decisions of agent. For each distantly supervised instance in the current bag, the agent makes an action from set of $\{1, 0\}$ to decide whether to select this instance or not. The agent receives the reward when all the actions have been completed. The reward represents the feedback of actions on this bag and will be used to update agent. The goal of agent is to decide actions that enable to maximize the reward.

State representation. In our view, the state s_t represents the current instance along with its label sequences. We represent the state as a vector $\mathbf{S}_t$, which consists of the following information: (1) The serialized vector representations of current instance, which are observed from the BiLSTM layer of baseline model. (2) The label scores calculated with output of the MLP layer from the shared encoder (denoted as $\mathbf{o}_t$ in Formula 1) and annotation of this instance, which means the tag-conditional noise of the distantly supervised annotation. More specifically, if a character is a part of an entity and annotated as a definite label (such as "皮" and "鞋" in Figure 3), the score of this position is the corresponding value in $\mathbf{o}_t$. Otherwise, we compute it by averaging the scores of all labels in $\mathbf{o}_t$. In this way, the dimension of the label scores vector is equal to the uniform length of sentences and will be concatenated with the first part.

Policy network. The agent decides an action $a_t \in \{0, 1\}$ to indicate whether the selector will select the t-th distantly supervised instance. The action value is sampled by the selector as $A_\Theta(s_t, a_t)$ where Θ is a multi-layer perceptron (MLP) with the parameter $\{\mathbf{W}, \mathbf{b}\}$. We adopt a logistic function as the policy function:

$$A_\Theta(s_t, a_t) = a_t\sigma(\mathbf{W} * \mathbf{S}_t + \mathbf{b}) + (1 - a_t)(1 - \sigma(\mathbf{W} * \mathbf{S}_t + \mathbf{b})) \tag{7}$$

where $\mathbf{S}_t$ is the state vector, and $\sigma(.)$ is the sigmoid function.

Reward. The reward is used to evaluate the ability of current NE tagger to predict labels of each character. The model receives a delayed average reward when it finishes all the selections in current bag, and before that the reward of each action is zero. The current bag $\mathcal{B}$ consists of two subsets: hand-tagged sentences $\tilde{\mathcal{H}}$ and distantly supervised instances $\tilde{\mathcal{A}}$. Now that the NE tagger calculates conditional probability for every sentence of bag $\mathcal{B}$, the reward can be computed on the set of selected distantly

supervised instances $\tilde{\mathcal{A}}_s$ and all the hand-tagged sentences:

$$r = \frac{1}{|\tilde{\mathcal{A}}_s| + |\tilde{\mathcal{H}}|}\Big(\sum_{x_j, \mathbf{z} \in \tilde{\mathcal{A}}_s} \log p(\mathbf{z}|x_j) + \sum_{x_k, y \in \tilde{\mathcal{H}}} \log p(y|x_k)\Big) \tag{8}$$

Different from the work of Feng et al. (2018), we have a set of supervised data. Our selector can be trained along the guidance of these prior knowledge about which sentences are labeled correctly. Therefore, the reward will become dependable and oriented, and it can guide the selector to maximize likelihood of all the instances in training dataset.

Selector Training. We use policy gradient method (Sutton et al., 2000) to optimize the policy network to maximize the reward of selections. For each random-sized bag $\mathcal{B}$, the feedback of every action $r(a_t)$ is same as average reward r. We compute the gradient and update the selector as follows:

$$\Theta = \Theta + \alpha \sum_{t=1}^{|\tilde{\mathcal{A}}|} r(a_t) \nabla_\Theta \log A_\Theta(s_t, a_t) \tag{9}$$

3.3 Joint Training

The parameters of the NE tagger and instance selector are learned iteratively. In each round, the selector first selects $\mathcal{A}_s$ from $\mathcal{A}$ and merges them with supervised sentences for the tagger. Meanwhile, the parameters of the NE tagger are learned from the newly training data and the tagger provides feedback reward to the selector to optimize its policy function.

4 Experiment

4.1 Datasets

We use two datasets in the experiments: one is from e-commerce domain and another is from news domain.

EC: In e-commerce domain (EC), we have five types of entities: Brand, Product, Model, Material, and Specification on user queries. This data contains 2,400 sentences tagged by annotators. We split the data into three sets: 1,200 sentences for training, 400 for dev, and 800 for testing. We collect a list of entities to construct dictionary from the training data. To reduce the effect of ambiguities, we remove the entry that belongs to more than one type, or it is a number or single character. Finally, the dictionary has 927 entries (included as EC.dic in supplementary materials). We perform distant supervision on raw data to obtain 2,500 sentences.

NEWS: For news domain, we use a NER data from MSRA, which was used in Sighan-bakeoff (Levow, 2006). We only test our systems on the type of PERSON. We randomly select 3,000 sentences as training dataset, 3,328 as dev data, and 3,186 as testing data. The rest set is used as raw data, having 36,602 sentences. We collect a list of person names from the training data. To increase the coverage, we add an additional names to the list. Finally, the list has 71,664 entries (included as NEWS.dic in supplementary materials). We perform distant supervision on raw data to obtain 3,722 sentences.

Embedding: In our approach, we need to map Chinese characters into vector representations by the lookup table, which can be initialized either by random or pre-trained. Many previous works (Lample et al., 2016; Peng and Dredze, 2015b) have shown that pre-trained embeddings on large-scale unlabeled corpus enable to initialize the table and observe efficiently improvements. Therefore, we collect one million sentences from the user-generated text in Internet, and pre-trained embeddings with the tool `word2vec`[1]. We set the embedding dimension as 100, the minimum frequency of occurrence as 5, and the window size of 5.

[1]https://code.google.com/archive/p/word2vec

Model	Training Data	Dev			Test		
		P	R	F1	P	R	F1
Dict-based	/	**74.21**	29.68	42.41	**75.60**	31.05	44.02
LSTM-CRF	$\mathcal{H}$	63.78	61.26	62.49	59.93	58.46	59.19
LSTM-CRF	$\mathcal{H}$ + $\mathcal{A}$	67.75	52.91	59.42	62.36	48.54	54.59
LSTM-CRF+SL	$\mathcal{H}$ + $\mathcal{A}$	67.56	55.04	60.66	62.86	50.87	56.23
LSTM-CRF-PA	$\mathcal{H}$ + $\mathcal{A}$	60.34	**64.49**	62.35	59.36	60.82	60.08
LSTM-CRF-PA+SL	$\mathcal{H}$ + $\mathcal{A}$	62.31	63.79	**63.04**	61.57	**61.33**	**61.45**

Table 1: Main results on the EC dataset.

Model	Training Data	Dev			Test		
		P	R	F1	P	R	F1
Dict-based	/	**95.40**	35.87	52.14	**96.08**	31.77	47.75
LSTM-CRF	$\mathcal{H}$	85.21	78.91	81.94	78.50	74.50	76.45
LSTM-CRF	$\mathcal{H}$ + $\mathcal{A}$	87.00	65.20	74.54	83.41	58.96	69.09
LSTM-CRF+SL	$\mathcal{H}$ + $\mathcal{A}$	86.33	72.34	78.72	81.99	66.10	73.19
LSTM-CRF-PA	$\mathcal{H}$ + $\mathcal{A}$	83.78	**81.79**	82.77	79.19	**77.59**	78.38
LSTM-CRF-PA+SL	$\mathcal{H}$ + $\mathcal{A}$	86.94	80.12	**83.40**	81.63	76.95	**79.22**

Table 2: Main results on the NEWS dataset.

4.2 Settings

For evaluation, we use the entity-level metrics of Precision (P), Recall (R), and their F1 values in our experiments, treating one tagged entity as correct only when it matches the gold entity exactly.

There are several hyper-parameters in our models. We set them empirically by the development performances. The instance selector is a multi-layer perceptron with 100 units in each hidden layer. We use Adam (Kingma and Ba, 2014) to train the instance selector with the learning rate 0.001. For the parameters of the tagger, we set the character embedding dimension as 100, the dimension sizes of hidden features as 200. We exploit online training with a mini-batch size 128 to learn model parameters. The max-epoch iteration is set by 800, and the best-epoch model is chosen according to the development performances. We use RMSprop (Tieleman and Hinton, 2012) with a learning rate 0.001 to update model parameters. We adopt the dropout technique to avoid overfitting by a drop value of 0.2 at the training stage.

4.3 Baselines

We build four systems based on our proposed approach and a dictionary-based baseline system, listed as follows:

- Dict-based: The collected entity dictionary is directly used to match the strings in the testing data.
- LSTM-CRF: the baseline model described in Section 3.2.
- LSTM-CRF-PA: the system trained on $\mathcal{H}$ and $\mathcal{A}$ with CRF-PA learning, but without instance selector.
- LSTM-CRF+SL: the system trained on $\mathcal{H}$ and $\mathcal{A}$ with instance selector, but without CRF-PA learning.
- LSTM-CRF-PA+SL: our final system trained on $\mathcal{H}$ and $\mathcal{A}$ with CRF-PA learning and instance selector.

Model	Training Data	Dev			Test		
		P	R	F1	P	R	F1
LSTM-CRF	$25\%\mathcal{H}$	43.35	46.18	44.72	40.36	40.78	40.57
LSTM-CRF-PA+SL	$25\%\mathcal{H} + 25\%\mathcal{A}$	45.63	50.95	48.14	44.08	46.57	45.29
LSTM-CRF	$50\%\mathcal{H}$	54.73	52.77	53.73	49.88	47.64	48.73
LSTM-CRF-PA+SL	$50\%\mathcal{H} + 50\%\mathcal{A}$	53.25	56.84	54.99	50.48	51.96	51.21
LSTM-CRF	$100\%\mathcal{H}$	63.78	61.26	62.49	59.93	58.46	59.19
LSTM-CRF-PA+SL	$100\%\mathcal{H} + 100\%\mathcal{A}$	62.31	63.79	63.04	61.57	61.33	61.45

Table 3: Results for varing percent of training data on the EC dataset.

4.4 Results

In this section, we show the model performances of our proposed systems and the other systems mentioned above. Table 1 shows the experimental results on the EC data and Table 2 shows the results on the NEWS data, respectively.

The low recall scores of Dict-based systems show that the coverage of the dictionaries is low even we have more than 70K person-names for the NEWS data. Compared with LSTM-CRF trained on $\mathcal{H}$, LSTM-CRF system trained on $\mathcal{H}$ and $\mathcal{A}$ provides much lower performance on two datasets. These facts indicate that the data generated by distant supervision contains many noises that affect the performance of the models. LSTM-CRF-PA yields better performance than LSTM-CRF trained on $\mathcal{H}$, showing +0.89 F1 improvement on EC and +1.93 F1 on NEWS. This indicates that CRF-PA learning can reduce the effect of incomplete annotation.

From the tables, we find that compared with LSTM-CRF-PA, LSTM-CRF-PA+SL obtains absolute improvements of +1.37 and +0.84 F1 points on EC and NEWS, respectively. Overall, our final system (LSTM-CRF-PA+SL) achieves better improvement with +2.26 and +2.77 F1 on EC and NEWS respectively over our baseline system LSTM-CRF. These facts show that the RL-based instance selector can provide additional help to CRF-PA learning.

We further investigate the effect of different sizes of human-annotated data. We randomly select 25% and 50% sentences from human-annotated data $\mathcal{H}$ as training data and build new dictionaries of entities based on them respectively. The new dictionaries are used to generate distantly supervised annotated data. Table 3 shows the results on the EC dataset, where the first two lines are for 25%, the third and fourth lines are for 50%, and the last two are for 100%. From the table, LSTM-CRF-PA+SL performs better performance than the baseline system, showing +4.72 F1 improvement on 25% and +2.48 improvement on 50%, respectively. This indicates that with smaller human-annotated data, our proposed approach can provide relatively larger improvement.

5 Related Work

Our approach is to utilize partial annotation learning and reinforcement learning to perform new-type named entity recognition in new domains. Several previous studies relevant to our approach have been conducted.

NER. Most early studies treated NER task as the sequence labeling problem based on a large annotated corpus with supervised methods, such as HMM, MEMM (Hai and Ng, 2002) and CRF (Lafferty et al., 2001). Recently, neural networks have been explored by researchers (Collobert et al., 2011; Lample et al., 2016), and applied to reduce the weakness of feature sparsity problem and heavy feature engineering (Cai and Zhao, 2016). Those models have the similar architecture for decoding and feature extraction, which is chosen as our baseline model. In order to overcome the challenge of data deficient, some approaches based on weakly supervised learning (Nadeau et al., 2006; Riloff and Jones, 1999) have been proposed and successfully expand training data and feature space. However, it is difficult to implement these methods on Chinese tasks because of the lack of morphological variations such as capitalization and in particular the uncertainty in word segmentation, and it may cause large number of matching errors.

Under reasonable assumptions, OOV features should not be forced into certain tag. What's more, joint models have also obtained great performance (Qian and Liu, 2013; Finkel and Manning, 2009).

Learning from PAs. Learning from PAs has always been an attractive idea, since it usually requires much less or even none human annotation effort to obtain partially annotated data than fully annotated data, especially for complex tasks like sequence labeling. Li et al. (2012) propose to only manually annotate the most uncertain word boundaries in a sentence for Chinese word segmentation in order to reduce annotation cost. Tsuboi et al. (2008) extend the standard CRF to directly learn from incomplete annotations for sequence labeling tasks. This work refers to their model as *CRF-PA*. Jiang et al. (2013) propose to derive segmentation boundaries from implicit information encoded in web texts, such as anchor texts and punctuation marks, and use them as partially labeled training data in Chinese word segmentation. Liu et al. (2014) and Yang and Vozila (2014) further improve the work of Jiang et al. (2013) by employing the more sophisticated CRF-PA model. Marcheggiani and Artières (2014) systematically compare a dozen uncertainty metrics in token-wise active learning with CRF-PA for several sequence labeling tasks. Li et al. (2016b) propose a coupled sequence labeling approach for exploiting heterogeneous data by treating the single-sided annotations as PAs for the task of joint word segmentation and POS tagging. In this work, we for the first time apply the CRF-PA model to NER, and employ distance supervision to produce partially annotated NE data.

Reinforcement Learning. In recent years, reinforcement learning has become an issue in research, and applied successfully to many tasks. In text generation community, a deep Q-learning is served by Guo (2015) as generative model to improve the seq2seq model, which completes the process of decoding by Iteration. Li et al. (2016a) show how to apply deep reinforcement learning to model future reward in chatbot dialogue and capture the impact of this conversation in the future. Dethlefs (2010) aim to optimize the integration of NLG tasks that are inherently different in nature by learning a generation policy with reinforcement learning. For computer vision, Yeung et al. (2017) propose a reinforcement learning-based formulation select the right examples for training a classifier from noisy web search results. To more fine-grainedly select high-quality training sentences from noisy data, Feng et al. (2018) train an instance selector based on a policy function with reinforcement learning, which is inspirational to our model.

6 Conclusion

This paper presents a new approach to utilize the data generated by distant supervision to perform new-type named entity recognition in new domains. We adopt partial annotation learning to address the problem of incomplete annotation and design the instance selector to choose positive sentences to reduce the effect of noisy annotation. The instance selector is based on reinforcement learning and obtains the feedback reward from the NE tagger. When tested on two newly created datasets, our systems provide better performance than the comparison systems. The data and code is available at `https://github.com/rainarch/DSNER`.

7 Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 61572338 and 61525205). Wenliang is also partially supported by the Natural Science Foundation of the Jiangsu Higher Education Institutions(Grant No. 16KJA520001). We would also thank the anonymous reviewers for their detailed comments, which have helped us to improve the quality of this work.

References

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 409–420.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

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

Nina Dethlefs. 2010. Hierarchical reinforcement learning for adaptive text generation. In *International Natural Language Generation Conference*, pages 37–45.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. pages 5779–5786.

Jenny Rose Finkel and Christopher D Manning. 2009. Joint parsing and named entity recognition. In *NAACL*, pages 326–334.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Hongyu Guo. 2015. Generating text with deep reinforcement learning. *Computer Science*, 40(4):1–5.

Leong Chieu Hai and Hwee Tou Ng. 2002. Named entity recognition: a maximum entropy approach using global information. In *International Conference on Computational Linguistics*, pages 1–7.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *CoRR*, abs/1508.01991.

Wenbin Jiang, Meng Sun, Yajuan Lü, Yating Yang, and Qun Liu. 2013. Discriminative learning with natural annotations: Word segmentation as a case study. In *Proceedings of ACL*, pages 761–769.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computer Science*.

John Lafferty, Andrew McCallum, Fernando Pereira, et al. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, volume 1, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL*, pages 260–270, June.

Gina-Anne Levow. 2006. The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117, Sydney, Australia, July. Association for Computational Linguistics.

Shoushan Li, Guodong Zhou, and Chu-Ren Huang. 2012. Active learning for Chinese word segmentation. In *Proceedings of COLING 2012: Posters*, pages 683–692.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016a. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Austin, Texas, November. Association for Computational Linguistics.

Zhenghua Li, Jiayuan Chao, Min Zhang, and Jiwen Yang. 2016b. Fast coupled sequence labeling on heterogeneous annotations via context-aware pruning. In *Proceedings of EMNLP*, pages 753–762.

Yijia Liu, Yue Zhang, Wanxiang Che, Ting Liu, and Fan Wu. 2014. Domain adaptation for CRF-based Chinese word segmentation using free annotations. In *Proceedings of EMNLP*, pages 864–874.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Meeting of the Association for Computational Linguistics*, pages 1064–1074.

Diego Marcheggiani and Thierry Artières. 2014. An experimental comparison of active learning strategies for partially labeled sequences. In *Proceedings of EMNLP*, pages 898–906.

Mintz, Mike, Steven, Jurafsky, and Dan. 2009. Distant supervision for relation extraction without labeled data. In *Joint Conference of the Meeting of the ACL and the International Joint Conference on Natural Language Processing of the Afnlp: Volume*, pages 1003–1011.

David Nadeau, Peter D. Turney, and Stan Matwin. 2006. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 266–277.

Nanyun Peng and Mark Dredze. 2015a. Named entity recognition for chinese social media with jointly trained embeddings. In *Proceedings of the EMNLP*, pages 548–554, Lisbon, Portugal.

Nanyun Peng and Mark Dredze. 2015b. Named entity recognition for chinese social media with jointly trained embeddings. In *Conference on Empirical Methods in Natural Language Processing*, pages 548–554.

Xian Qian and Yang Liu. 2013. Joint chinese word segmentation, pos tagging and parsing. In *Proceedings of EMNLP*, pages 501–511.

Sebastian Riedel, Limin Yao, and Andrew Mccallum. 2010. Modeling relations and their mentions without labeled text. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, pages 474–479.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Yuta Tsuboi, Hisashi Kashima, Hiroki Oda, Shinsuke Mori, and Yuji Matsumoto. 2008. Training conditional random fields using incomplete annotations. In *Proceedings of COLING*, pages 897–904.

Fan Yang and Paul Vozila. 2014. Semi-supervised Chinese word segmentation using partial-label learning with conditional random fields. In *Proceedings of EMNLP*, pages 90–98.

Serena Yeung, Vignesh Ramanathan, Olga Russakovsky, Liyue Shen, Greg Mori, and Li Fei-Fei. 2017. Learning to learn from noisy web videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5154–5162, July.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Conference on Empirical Methods in Natural Language Processing*, pages 1753–1762.

Joint Neural Entity Disambiguation with Output Space Search

Hamed Shahbazi, Xiaoli Z. Fern, Reza Ghaeini, Chao Ma,
Rasha Obeidat, Prasad Tadepalli
Oregon State University, Corvallis, OR, USA
{shahbazh, xfern, ghaeinim, machao, obeidatr, tadepall}@eecs.oregonstate.edu

Abstract

In this paper, we present a novel model for entity disambiguation that combines both local contextual information and global evidences through Limited Discrepancy Search (LDS). Given an input document, we start from a complete solution constructed by a local model and conduct a search in the space of possible corrections to improve the local solution from a global view point. Our search utilizes a heuristic function to focus more on the least confident local decisions and a pruning function to score the global solutions based on their local fitness and the global coherences among the predicted entities. Experimental results on CoNLL 2003 and TAC 2010 benchmarks verify the effectiveness of our model.

1 Introduction

The goal of entity disambiguation is to link a set of given query mentions in a document to their referent entities in a Knowledge Base (KB). As an essential and challenging task in Knowledge-Base Population (KBP) for text Analysis (Ji et al., 2014; Heng et al., 2015), entity disambiguation has attracted many research efforts from the NLP community. Recently, deep learning based approaches have demonstrated strong performances on this task (Ganea and Hofmann, 2017; Sil et al., 2018).

A main challenge for entity disambiguation is to best identify and represent the appropriate context, which can be local or global. Different methods have been proposed to capture and represent different types of contexts. Textual context has been heavily investigated for local models that score each query's candidates independently. Representations of the textual contexts range from weighted combination of the word embeddings based on attention (Ganea and Hofmann, 2017), to more fine-grained contextual representations using recurrent neural networks (Sil et al., 2018). The global context of other entities in the document has also been studied for a more global and joint prediction view on the problem. Ganea and Hofmann (2017) use a Conditional Random Field (CRF) based model to capture the interrelationship among entities in the same document, whereas Globerson et al. (2016) introduce a soft k-max attention model to weigh the importance of other entities in the document in making prediction for any given query.

Working with the recently proposed models, we observe that local models that employ an appropriate attention mechanism often have a solid linking performance. In a single document, there are often a small number of hard queries for which the local model fails to make a correct decision. We conjecture that if some of these mistakes can be corrected, a global model that enforces coherence among entities will be able to propagate these corrections to improve the overall solution quite effectively.

This inspires us to consider the Limited Discrepancy Search (LDS) framework (Doppa et al., 2014), which conducts a search over possible corrections on a complete output with the goal of improving the final output. Critically, LDS works well for cases where only a small number of local corrections are needed to reach a good global solution. This nicely matches up with our observation of the behavior of entity disambiguating models.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2170–2180
Santa Fe, New Mexico, USA, August 20-26, 2018.

In this paper, we propose a LDS based global entity disambiguating system. Given a document and its query mentions, our system first applies a local disambiguation model to produce an initial solution. We then use LDS to conduct a shallow search in the space of possible corrections (with the focus on hard/least confident mentions) to find a better solution. Evaluation on CoNLL 2003 and TAC 2010 shows that our method outperforms the current state-of-the-art models. We also conduct an extensive ablation study on different variants of our model, providing insight into the strength of our method.

2 Proposed Approach

We are given a document D containing n mentions $[x_1, ...x_n]$. We assume that all mentions are linkable to a Knowledge Base (KB) by excluding the NILL mentions. We are interested in finding a joint assignment of all mentions to $\mathbf{Y} = [y_1, ...y_n]$ of referent entities to maximize the following score:

$$s(\mathbf{Y}) = \sum_{i=1}^{n} \psi(x_i, y_i) + \sum_{i=1}^{n} \sum_{j=1:j\neq i}^{n} \phi(y_i, y_j) \quad (1)$$

where function $\psi(x_i, y_i)$ gives the local compatibility score between the mention x_i and its candidate y_i and $\phi(y_i, y_j)$ indicates the amount of relatedness between the assigned candidates y_i and y_j. Optimizing this objective, however, is NP-hard. In this work, we develop a LDS-based search strategy for optimizing this objective.

2.1 Overview of the Approach

Given a document with n mentions, we initialize the search with a solution acquired based solely on the local scoring function $\psi(\cdot, \cdot)$. We then conduct a greedy beam search in the space of possible discrepancies (changes/corrections) to this initial solution while focusing on mentions with least confident local scores in the hope of finding a better solution. Fig 1 shows the overview of our search framework

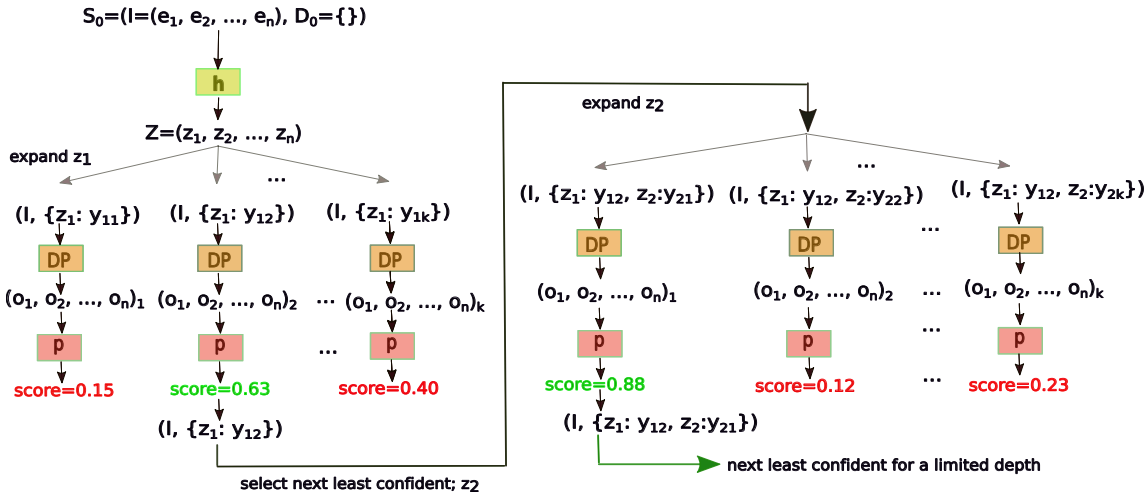


Figure 1: The general framework of our proposed model with beam size b=1.

for beam size $b = 1$. Each state S_i is a pair (I, D_i) where $I = (e_1, e_2, ..., e_n)$ is the initial solution given by the local model and D_i is the discrepancy set for state S_i. For example, a discrepancy set $\{x_{k_1} : y_{k_1}, x_{k_2} : y_{k_2}\}$ contains two discrepancies, changing the assignment for mentions x_{k_1} and x_{k_2} from e_{k_1} and e_{k_2} in I to y_{k_1} and y_{k_2} respectively. Starting with initial state (I,), we utilize a heuristic function h (Section 2.3.2) to sort the mentions in the increasing order of their local confidence. Let $(z_1, ..., z_n)$ be the ordered mentions where z_1 is the least confident mention. Each iteration of the greedy beam search explores and prunes the space of discrepancy sets as follows:

We select the least confident mention z_1 and expand state $(I, \{\})$ to k new states $(I, \{z_1 : y_{1j}\})$ for $j = 1 \cdots k$ where y_{1j} is the j^{th} candidate for mention z_1 (we consider top k most probable candidates). Each

expanded state $(I, \{z_1 : y_{1j}\}), j = 1, \ldots, k$ is given to a Discrepancy Propagator (DP) (Section 2.3.1) to get its discrepancy set propagated throughout the document and produce an updated complete solution $(o_1, ...o_n)_j$, for $j = 1, \ldots, k$. Utilizing a trained pruning function p (Section 2.3.3) we then rank the k complete solutions and prune them to top b states (b is beam size).

The search continues with the next least confident mention z_2 as shown in Fig 1. Each iteration increases the size of the discrepancy set by one. Note that a mention is not repeated in a discrepancy set. We consider two different strategies for terminating the search depending on the used heuristic function h (Section 2.3.3). Each strategy causes the search to terminate at different depth (length) of the discrepancy set. The output of the search is selected by the pruning function p from the last set of complete solutions.

In the following, we will first explain our **local model** ($\psi(\cdot, \cdot)$) for producing the initial solution. We will then introduce our LDS search framework, and its key components including the **Discrepancy Propagator** to propagate a set of discrepancies to other mentions in the document; the **Heuristic Function** to compute the local confidence of the mentions in the document and identify the least confident ones; and finally the **Pruning Function** to guide the search and select the final solution.

2.2 Local Model

Our local model utilizes contextual, lexical and prior evidences to compute the compatibility score $\psi(x_i, y_i)$ for assigning candidate y_i to mention x_i. These evidences are extracted and used as follows:

2.2.1 Contextual Evidence

Given a query mention, a key challenge for the local model is to identify a minimum but sufficient contextual evidence to disambiguate the query. We first extract all sentences in the document relevant to the query mention. This is achieved by applying CoreNLP (Manning and McClosky, 2014) to perform coreference resolution to all mentions in the document and extract all sentences containing a mention in the query's coreference chain.

The set of sentences are then concatenated to form the context for the query; $w_i = [w_{i1}, ...w_{im}]$, where $w_{ij} \in R^d$ is the embedding of the j-th word in the context. We then use an attention model introduced by (Ganea and Hofmann, 2017) to compress the context into a single embedding. Specifically, we define the contextual representation $c_i \in R^d$ for mention x_i with candidate set $\{y_{i1}, y_{i2}, ...y_{ik}\}$ over context w_i as

$$c_i = \sum_{l=1}^{m} \alpha_{il} w_{il} \tag{2}$$

The weight vector α is computed using the following attention considering all k entity candidates:

$$\alpha_i = \textbf{softmax}([\max_{j \in 1..k} y_{ij}^T A w_{i1}, ..., \max_{j \in 1..k} y_{ij}^T A w_{im}]) \tag{3}$$

where A is a learned matrix that scores the relatedness between word and entity. This attention model computes the relatedness of each word with all of the entity candidates and takes the max as the score for each word. The scores of all context words are then passed through softmax to compute their weight. Under this model, if a word is strongly related to one of the candidates, it will be given a high weight. Subsequently, using c_i we define the following contextual features for candidate y_{ij} as $f_{ij}^{(c)} = [y_{ij}^T B c_i; y_{ij}^T B; c_i]$, where B is a learned $R^d \times R^d$ matrix. Note that $f_{ij}^{(c)} \in R^{2d+1}$.

2.2.2 Lexical Evidence

The contextual features extracted above ignores any lexical/surface information between query mention and entity title, which can be useful. To this end we include some lexical features to our local model including variants of the edit distance between the surface strings of the mention and the candidate title, whether their surface strings follow an acronym pattern and etc. Detailed list can be found in the Table 1(a). The extracted features are scalar real values. We use RBF binning (Sil et al., 2018) to transform each scalar to a 10-d vector. Hence for each query x_i and candidate y_{ij} we have lexical vector feature $f_{ij}^{(l)} \in R^{10 \times |f|}$ where $|f|$ is the number of features listed in Table 1(a).

(a)

mention: m = [m_1, ... m_a], **entity title:** e = [t_1, ... t_b]
f_1: mention length = a
f_2: entity title length = b
f_3: $\sum_{i=1}^{b}$ (occurrence counts of t_i in the document)
f_4: m is acronym
f_5: e is acronym
f_6: m and e acronym patterns are exact match
f_7: m and e are non-acronyms and exact match
f_8: min-edit(m, e)
f_{10}: sum of partial min edits: $\sum_{i=1}^{a}(\min_{j=1}^{b}(\text{min-edit}(m_i, t_j)))$

(b)

local score for mention m: l = **sotmax**([$\psi(x_i, y_{i1})$, ... $\psi(x_i, y_{ik})$])
f_1: max(l)
f_2: second-max(l)
f_3: entropy(l)
f_4: m is an acronym
f_5: length(m)

Table 1: (a) list of lexical features in the local model (b) list of the features to learn heuristic function h_2

2.2.3 Prior Evidence

We consider $p(e|m)$, the prior probability that an entity e is linked to a mention string m as prior evidence. This is computed using hyper-link statistics from Wikipedia and aliases mapping from (Hoffart et al., 2011) obtained by extending the means tables of YAGO (Hoffart et al., 2013). The $p(e|m)$ is also transformed to a 10-d vector using RBF binning (Sil et al., 2018) to create prior feature $f_{ij}^{(p)} \in R^{10}$.

2.2.4 Overall local model

The contextual, lexical and prior features are concatenated and fed through a Multi Layer Perceptron (MLP) with 2 hidden layers (relu, 200-d and 50-d respectively) to produce a final local score for all candidates as follows:

$$\psi(x_i, y_{ij}) = \sigma(W_s[f_{ij}^{(c)}; f_{ij}^{(l)}; f_{ij}^{(p)}] + b_s) \tag{4}$$

Where W_s and b_s are weight and bias parameters for the MLP. The learning of the model is two tiered. We first pre-train the matrices A and B using the cross-entropy loss by using **softmax**($[y_{ij}^T B c_i]_{j \in 1..k}$) as the predicted probability for each candidate for mention x_i, Keeping A and B fixed, we then train the MLP weights with a drop-out rate of 0.7 minimizing again the cross-entropy loss. Note that we use the entity embeddings produced by (Ganea and Hofmann, 2017) and the word embeddings produced by (Mikolov et al., 2013) using skip-gram model.

2.3 Global Model via LDS

Our local model primarily focuses on the textual context of each mention in making its decisions and ignores the relationship between mentions. Our global model takes as input the local predictions for all the mentions in a single document and constructs a globally coherent final solution using Limited Discrepancy Search. Before we introduce our LDS search procedure, we will first describe the discrepancy propagator which is critical toward achieving an efficient search space.

2.3.1 Discrepancy Propagator

The purpose of the discrepancy propagator is to allow the influence of the local changes to propagate to other parts of the solution, thus reducing the necessary number of discrepancies needed. Our discrepancy propagator is essentially a new scoring function that evaluates each candidate for a given mention not only based on the local compatibility but also the global coherence among the predictions.

Given that not all mentions in a document need to be related to one another, for each mention x_i we construct an entity context E_i considering a window of 30 mentions centered at query mention x_i. Given the current entity assignment $e_1, e_2, ...e_n$ to the mentions in the document, the entity context for mention x_i will be $E_i = e_{i-15}..., e_{i-1}, e_{i+1}, ...e_{i+15}$.
For a given mention x_i and its entity context E_i, the new score of a candidate y_{ij} is defined as:

$$g(x_i, y_{ij}) = \psi(x_i, y_{ij}) + \frac{1}{|E_i|} \sum_{e \in E_i} \phi(e, y_{ij}) \tag{5}$$

In equation 5, the score of a candidate now considers both its local score ψ as well as its average coherence with the other predictions in its entity context. The coherence score ϕ between two entities e and y_{ij} is defined as:

$$\phi(e, y_{ij}) = e^T C y_{ij} + w^T r(e, y_{ij}) \tag{6}$$

where e and y_{ij} are entity embeddings and $r(e, y_{ij})$ is a vector of three pairwise features between the two entities: log transformed counts of co-ocurrences, shared in-links and shared out-links; $C \in R^{d \times d}$ and $w \in R^3$ are learned weights.

The score given by equation 5 is fed through a softmax activation to assign probabilities to each candidate of x_i. We train W and c in a mention-wise procedure using cross entropy loss. Specifically for each mention x_i we use ground truth entities for the other mentions in E_i and update weights of C and w such that the true candidate of x_i gets higher score than its false candidates.

Note that we do not intend to use this scoring function to do joint inference on all mentions. Instead, it is used with LDS to propagate the discrepancies to the output of all related mentions as follows. Given the original solution $e_1, e_2, ...e_n$ and a set of discrepancies $\{x_k : y_k\}$, we first update e_k to y_k. Then we re-evaluate equation 5 for all mentions that have y_k in their entity context to produce a complete solution based on new entity context containing y_k.

2.3.2 Heuristic Function

The heuristic function takes the initial solution and sort all the mentions based on their prediction confidence. In this work we consider two heuristic functions for computing the prediction confidence of the local model. For a given query x_i, our first heuristic function h_1 computes its confidence simply by taking the max of the local scores normalized by a softmax funtion:

$$h_1(x_i) = \textbf{max}(\textbf{softmax}([\psi(x_i, y_{i1}), ...\psi(x_i, y_{ik})])) \tag{7}$$

where $\psi(x_i, y_{ij})$ is the local score for candidate y_{ij}.

For our second heuristic function h_2, we learn a binary classifier (a two layer MLP) to produce local confidence for each mention x_i. The binary classifier utilizes the features listed in Table 1(b). Each feature value is transformed into a 10-d vector using RBF binning (Sil et al., 2018). The training samples for the binary classifier are generated from the prediction of the local model on the training set. One training example is generated for each local prediction — if the local model is correct about its decision then the label for the sample is 1. In order to balance the positive and negative training samples, we randomly sub-sample positive local predictions. For each local prediction, the learned classifier outs a probability of its being correct, which is used as the confidence score.

2.3.3 Pruning Function

At each iteration of the search, each beam is expanded to k new states. Following the expansion, the DP is applied to the $b \times k$ expanded nodes and re-evaluates equation 5 for all mentions with updated entity contexts due to the discrepancies. It hence propagates the discrepancies and produces a complete solution. Subsequently, the pruning function evaluates each of the $b \times k$ complete solutions and reduces the expansion list to the beam size b.

The pruning function is similar to equation 5 but scores all the mentions collectively. Given all the mentions in the document $x_1, ..., x_n$ and their predicted entities denoted as $o = o_1, ..., o_n$, our pruning function scores the solution o as follows:

$$s(o) = \sum_i \psi(x_i, o_i) + \sum_{(o_i \in E_j \text{ or } o_j \in E_i)} \phi_g(o_i, o_j) \tag{8}$$

Where constraint ($o_i \in E_j$ or $o_j \in E_i$) indicates that we consider relation among pair of entities (o_i, o_j) if they are in the entity context of one another. Here ϕ_g takes the same form as equation 6 but uses a different set of weights C_g and w_g.

Despite the similarity in forms, the pruning function serves a different purpose from that of equation 6 and requires different training. We train the pruning function by reducing it to a rank learning problem.

Specifically, in training, we collect all the complete solutions that are considered in each pruning step, and compute their hamming losses from the ground truth. Given a set of solutions in a pruning step, we will create ranking pairs to require the solution with the least hamming loss to score higher than all the others. Given a ranking pair, o^t and o^f, assuming o^t has lower hamming loss than o^f, we use the following ranking loss for training:

$$max\left(0, \Delta(o^t, o^f) - s(o^t) + s(o^f)\right) \tag{9}$$

where $\Delta(o^{(t)}, o^{(f)})$ is the absolute difference of the hamming-loss between $o^{(t)}$ and $o^{(f)}$. This loss function penalizes the scoring function if it fails to score o^t higher than o^f by a margin specified by $\Delta(o^{(t)}, o^{(f)})$.

Terminating the search. We consider two different strategies for terminating the search depending on the heuristic function in use. When using h_1 as the heuristic function, search terminates when we reach a depth limit τ (a maximum τ discrepancies). The strategy with h_2 uses a flexible depth. For this strategy, we terminate the search when discrepancies have been introduced for all queries that are predicted to be incorrect by h_2.

3 Experiments

3.1 Data Sets

We use two datasets CoNLL 2003 (Hoffart et al., 2011) and TAC 2010 (Ji et al., 2010) for evaluation. The CoNLL dataset is partitioned into train, test-a and test-b with 946, 216 and 231 documents respectively. Following our baselines we only use 27816 mentions with valid links to the KB. The TAC 2010 dataset is yet another popular NED dataset released by the Text Analysis Conference (TAC). The dataset contains training and test set with 1043 and 1013 documents respectively. Similar to CoNLL we only consider linkable mentions in TAC and report our performance on 1020 query mentions in the test set.

To learn and tune the parameters of the local models for CoNLL and TAC we use their own training and development splits. However, to learn and tune the parameters of the global model (the discrepancy propagator, the heuristic and pruning functions) we only use the CoNLL training and dev-sets.

The number of queries per document in the test sets of the CoNLL and TAC are approximately 20 and 1 respectively. In order to have a global setup for TAC we apply CoreNLP (Manning and McClosky, 2014) mention extractor to the test documents of TAC and perform joint disambiguation of the extracted mentions together with the query mentions, increasing the number of mentions to approximately 4 per document. We only report the performance on the query mentions with the given standard ground truth by TAC.

3.2 Hyper-parameters and Dimensions

Our model settings for the hyper-parameters and dimensionality of the embeddings and weights are as follows: We use entity/word embeddings of 300-d. We learned the embeddings using the mechanism proposed by (Ganea and Hofmann, 2017). We use 2-layers MLPs for the local model and the heuristic h_2 with hidden layer sizes 200×50 and 100×20 for each respectively. The RBF binning always transfers a scalar to 10-d. Although we analyze and compare different configurations for beam-size and depth limit in section 3.4, we use beam size 5 and flexible depth limit τ with heuristic h_2 in our reported results.

3.3 Results

To evaluate the model performance we use the standard micro-average accuracies of the top-ranked candidate entities. We use different alias mappings for TAC and CoNLL. Specifically, for TAC we only use anchor-title alias mappings constructed from hyper-links in the Wikipedia. For CoNLL, in order to follow the experimental setup reported by (Sil et al., 2018), in addition to the alias mappings of Wikipedia anchor-titles, we use the mappings produced by (Pershina et al., 2015) and (Hoffart et al., 2011).

Tables 2(a) and (b) show our performance on the CoNLL and TAC datasets for our local and global models along with other competitive systems respectively. The prior state of the art performances on

these two datasets are achieved by (Sil et al., 2018). The results show that our global model outperforms all the competitors for both CoNLL 2003 and TAC 2010. It is interesting to note that our local model is solid, but is noticeably inferior to the stat-of-the-art local model. With an even stronger local model like that of (Sil et al., 2018), one can potentially expect our LDS-based global model to push the state-of-the-art even further.

models	In-KB acc%
local	
(He et al., 2013)	85.6
(Francis-Landau et al., 2016)	85.5
(Sil and Florian, 2016)	86.2
(Nevena Lazic and Pereira, 2015)	86.4
(Yamada et al., 2016)	90.9
(Sil et al., 2018)	94.0
global	
(Hoffart et al., 2011)	82.5
(Pershina et al., 2015)	91.8
(Globerson et al., 2016)	92.7
(Yamada et al., 2016)	93.1
our model	
local	90.89
global	**94.44**

(a) CoNLL 2003

models	In-KB acc%
local	
(Sil and Florian, 2016)	78.6
(He et al., 2013)	81.0
(Sun et al., 2015)	83.9
(Yamada et al., 2016)	84.6
(Sil et al., 2018)	87.4
global	
(Yamada et al., 2016)	85.2
(Globerson et al., 2016)	87.2
our model	
local	85.73
global	**87.9**

(b) TAC-2010

Table 2: Evaluation on CoNLL 2003 Test-b and TAC-2010

3.4 Ablation Study and Performance Analysis

For ablation we analyze the impact of the features and search. We also analyze the behavior of our model based on the rarity of the entities.

3.4.1 Feature Analysis

We study the impact of features on the local and global models. For the local model, starting with only considering the contextual evidence as described in Section 2.2.1, we see that the performance steadily increases as we add the prior and lexical features. As shown in tables 3(a) and (b) the prior and lexical features have very strong impact on TAC. The binning technique that projects the prior and lexical features to 10 dimensions gives an average of 0.94% and 0.61% percentage points for TAC and CoNLL respectively.

For the global model, entity pair compatibility is computed using both entity embeddings and log transformed counts of co-occurences, shared in-links and shared out-links. Using these features leads to a gain of 1.1% and 0.43% in accuracy for CoNLL and TAC respectively compared to the global model using only the embeddings in equation 6.

models	In-KB acc%
local	
Context only	85.37
Context + Prior + Lexical	90.28
Context + Prior + Lexical + Bin	90.89
global	
Our global	94.44
- log count features	93.34
- LDS + 1-step global prop.	93.14
- LDS + conv. global prop.	93.63

(a) CoNLL 2003

models	In-KB acc%
local	
Context	70.86
Context + Prior + Lexical	84.79
Context + Prior + Lexical + Bin	85.73
global	
Our global	87.9
- log count features	87.47
- LDS + 1 step global prop.	86.21
- LDS + conv. global prop.	86.29

(b) TAC-2010

Table 3: The performance of variants of our Local/Global models on CoNLL 2003 and TAC-2010

3.4.2 Search Analysis

In the last two rows of Table 3, we list variants of our model where LDS search is removed. Specifically, given the initial local solution, our discrepancy propagator (equation 5) can be applied without any discrepancy to re-evaluate the candidates for all mentions and obtain a more globally compatible solution. Naturally, one can repeat this for multiple iterations till convergence, which gives us an alternative global model that does not rely on search. We consider this model and show its single iteration performance (-LDS + 1-step global prop.) as well as its converged performance (-LDS + conv. global prop.) in Table 3.

From the results we can see that a single iteration of the global propagation was able to significantly improve the initial local solution, but later iterations lead to very mild gain. This is because after the first propagation the convergence is almost immediate and there is a limited improvement after the first iteration. In contrast, our search based global model is able to achieve significant further performance gain 1.3% for CoNLL and 1.68% for TAC. The reason is that with local discrepancies introduced by LDS, we force the solution to escape the current local optimum and search for better ones.

Additionally, several parameters/choices in the search can potentially impact the performance: the beam size b, the depth of search, and the different heuristics for prioritizing the discrepancy locations. The following set of experiments explore these choices of parameters:

Beam size. We compare model with beam size $b = 5$ and a simple greedy model with $b = 1$. For CoNLL 2003 and TAC-2010, the model with $b = 5$ gives a gain of 0.24% and 0.13% compared to the greedy. Increasing the size of the beam further beyond 5 did not show significant gain.

Depth of search. For heuristic h_1, we use a fixed depth strategy. In particular, given a document with n mentions, we consider two different depth limits: $\tau = 25\%n$ and $\tau = 50\%n$, which lead to an average depth of 5 and 10 per document respectively. For heuristic h_2, the depth is flexible and determined by the number of mentions that are predicted to be incorrect by h_2. This strategy leads to an average depth of 4. In Tables 4 (a) and (b), we report the confusion matrices given by h_1 applied to the local prediction on test-b of CoNLL 2003 with $\tau = 25\%n$ and $\tau = 50\%n$ respectively. In these tables correct/incorrect mean whether the local prediction is correct or not (comparing to the ground truth). Therefore the first cell with value 1134 indicates that there are 1134 mentions in test-b that are correctly predicted by the local model, but deemed as among the top 25% least confident mentions (aka the hard queries) by h_1. The confusion matrix for a good heuristic will have small diagonal values and large anti-diagonal values.

In Table 4 (c), we apply heuristic h_2 to the same data with flexible depth. These results show that heuristic h_2 gives the best precision as well as recall of the real mistakes made by the local model.

$\tau = 25\%n$	r $\leq 25\%$	r $> 25\%$
correct	1134	2937
incorrect	276	136

(a)

$\tau = 50\%n$	r $\leq 50\%$	r $> 50\%$
correct	2071	2000
incorrect	331	81

(b)

τ=flexible	label=0	label=1
correct	805	3266
incorrect	333	79

(c)

Table 4: Confusion Matrices for (a) using heuristic h_1 with $\tau = 25\%n$, (b) heuristic h_1 with $\tau = 50\%n$ and (c) heuristic h_2 with flexible τ.

Models	Heuristic	Depth (τ)	Beam Size (b)	In-KB acc %
Global + LDS	h_1	$\tau = 25\%n$	1	93.88
Global + LDS	h_1	$\tau = 25\%n$	5	94.12
Global + LDS	h_1	$\tau = 50\%n$	5	94.23
Global + LDS	h_2	flexible	5	94.44

Table 5: CoNLL 2003 Test-b

In Table 5, we report the performance of our model on CoNLL 2003 with different choices for the heuristic, search depth, and beam size. The results show that using a beam of size 5 improves upon single greedy search, and using heuristic h_2 with flexible depth gives the best performance in terms of both prediction accuracy and efficiency (due to smaller search trees). When h_1 is used, doubling the depth of the search tree brings about only a marginal improvement in accuracy at the cost of doubling the search depth and thus the prediction time.

3.4.3 Performance Analysis Based on Entity Rarity

We also analyze the behavior of the coherent global models based on the rarity of the entities for the given mention. Specifically, we measure the rarity of e for given query mention m by $p(e|m)$, as defined in 2.2.3, scaled to (0, 100). To this end we quantize the value of rarity measure into different bins. For each bin we compute the difference of accuracy between a coherent global model and the local model. Two coherent models are considered here; a global model without LDS (-LDS + conv.global prop in Table 3) and global model with LDS using h_2. Figure 2 shows the difference of accuracy between the global and local models per bin. As shown in this figure both global models achieve gains mostly on bins with small $p(e|m)$, which are related to the mentions with rare true entity. Additionally, the impact of

the global model when LDS is used is more significant, especially for the mentions whose true entities are most rare according to $p(e|m)$.

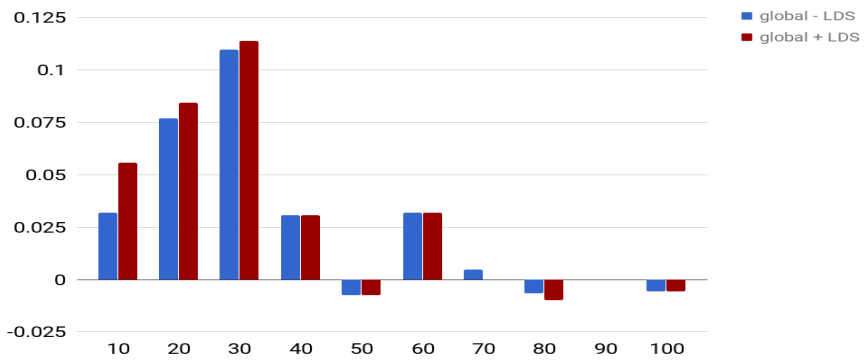


Figure 2: Global accuracy minus local accuracy per bin of rarity measure $p(e|m)$ which is scaled to (0, 100) for two coherent global models; model without LDS and model with LDS.

4 Related Work

Deep learning has been leveraged in recent local and global models. In local models Sun et al. (2015) use neural tensor networks to model mention, context and entity embeddings. Ganea and Hofmann (2017) develop an attention based model to weigh the importance of the words in the query document. The model by Sil et al. (2018) utilizes neural tensor network, multi-perspective cosine similarity and lexical composition and decomposition. In global models, the early models (Milne and Witten, 2008; Ferragina and Scaiella, 2010) decomposes the problem over mentions. Hoffart et al. (2011) use an iterative heuristic to prune edges among mention and entity. Cheng and Roth (2013) use an integer linear program solver and Lev-Arie Ratinov and Anderson (2011) apply SVM to use relation scores as ranking features.

In recent global models, Personalized PageRank (PPR) (Jeh and Widom, 2003) is adopted by several studies (Han and Sun, 2011; He et al., 2013; Alhelbawy and Gaizauskas, 2014; Pershina et al., 2015). Yamada et al. (2016) extend the skip-gram model (Mikolov et al., 2013) to learn the relatedness of entities using the linking structure of the KB. Ganea and Hofmann (2017) use a Conditional Random Field (CRF) based model to capture the interrelationship among entities in the same document whereas Globerson et al. (2016) introduce a soft k-max attention model to weight the importance of other entities in the document in making prediction for any given query.

In our proposed global model we address the intractable global optimization in a search framework. Specifically we use Limited Discrepancy Search (Doppa et al., 2014). Initialized with a local solution, LDS only explores a small number corrections to the hard queries. The propagation of the corrections enables the correction of other mistakes (even those with high local confidence) and allows us to reach high quality solutions with shallow searches. Moreover, the heuristic to prioritize the discrepancies can noticeably reduce the time without hurting the performance.

5 Conclusions

In this paper we study the problem of entity disambiguation. We are inspired by the observation that local models in this task tend to produce reasonable solutions, such that with only a small number of corrections and a proper propagation of these corrections throughout the document, one can quickly find superior solutions that are globally more coherent.

Based on this observation, we propose a search based approach that starts from an initial solution from a local model, and uses Limited Discrepancy Search (LDS) to search through the space of possible corrections with the goal of improving the linking performance. The experimental results show that our global model improves the state of the art on both the CoNLL 2003 and TAC 2010 benchmarks. For future research, we are interested in further understanding of the strengths and weaknesses of the LDS-based approach for different types of queries and entities. We are also interested in applying different local models to initialize the LDS search.

References

Ayman Alhelbawy and Robert Gaizauskas. 2014. Graph ranking for collective named entity disambiguation. *In Proc. 52nd Annual Meeting of the Association for Computational Linguistics, ACL.*

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. *In Proc. of EMNLP*.

Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. 2014. Structured prediction via output space search. *Journal of Machine Learning Research*, 15:1317–1350.

Paolo Ferragina and Ugo Scaiella. 2010. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). *In Proc. of the 19th ACM International Conference on Information Knowledge and Management, CIKM*.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. semantic similarity for entity linking with convolutional neural networks. *Annual Conference of the North American Chapter of the Association for Computational Linguistics, NAACL.*

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. *In Proc. of Empirical Methods in Natural Language Processing*.

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. *In Proc. of Association for Computational Linguistics, ACL.*

Xianpei Han and Le Sun. 2011. A generative entity-mention model for linking entities with knowledge base. *In Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, ACLHLT*.

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013. Learning entity representation for entity disambiguation. *Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL.*

Ji Heng, Nothman Joel, and Hachey Ben. 2015. Overview of tac-kbp2015 tri-lingual entity discovery and linking. *In Proc. Text Analysis Conference, TAC*.

Johannes Hoffart, Klaus Berberich, and Gerhard Weikum. 2013. A spatially and temporally enhanced knowledge base from wikipedia; yago2. *Artificial Intelligence*.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, and etc. 2011. Robust disambiguation of named entities in text. *In Proc. of Empirical Methods in Natural Language Processing, EMNLP*.

Glen Jeh and Jennifer Widom. 2003. Scaling personalized web search. *In Proceedings of the 12th international conference on World Wide Web, pages 271279. ACM*.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the tac 2010 knowledge base population track. *In Proc. of the 3rd Text Analysis Conference, TAC*.

Heng Ji, Nothman Joel, and Hachey Ben. 2014. Overview of tac-kbp2014 entity discovery and linking tasks. *In Proc. Text Analysis Conference, TAC*.

Doug Downey Lev-Arie Ratinov, Dan Roth and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. *In Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, ACL HLT*.

Surdeanu Mihai Bauer John Finkel Jenny Bethard Steven J. Manning, Christopher D. and David McClosky. 2014. The stanford corenlp natural language processing toolkit. *Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL*, pages 55–60.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *In Advances in Neural Information Processing Systems, NIPS*.

David N. Milne and Ian H. Witten. 2008. Learning to link with wikipedia. *In Proc. of the 17th ACM Conference on Information and Knowledge Management, CIKM*.

Michael Ringgaard Nevena Lazic, Amarnag Subramanya and Fernando Pereira. 2015. Plato: A selective context model for entity resolution. *Transactions of the Association for Computational Linguistics*.

Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized page rank for named entity disambiguation. *Annual Conference of the North American Chapter of the Association for Computational Linguistics, NAACL.*

Avirup Sil and Radu Florian. 2016. Towards language independent named entity linking. *Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL.*

Avirup Sil, Gourab Kundu, Radu Florian, and Wael Hamza. 2018. Neural cross-lingual entity linking. *Thirty-Second AAAI Conference on Artificial Intelligence, AAAI.*

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. *International Joint Conference on Artificial Intelligence, IJCAI.*

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. *International Conference on Computational Linguistics, COLING.*

Learning to Progressively Recognize New Named Entities with Sequence to Sequence Models

Lingzhen Chen
University of Trento
Povo, Italy
lingzhen.chen@unitn.it

Alessandro Moschitti*
Amazon
Manhattan Beach, CA, USA
amosch@amazon.com

Abstract

In this paper, we propose to use a sequence to sequence model for Named Entity Recognition (NER) and we explore the effectiveness of such model in a progressive NER setting – a Transfer Learning (TL) setting. We train an initial model on source data and transfer it to a model that can recognize new NE categories in the target data during a subsequent step, when the source data is no longer available. Our solution consists in: (i) to reshape and re-parametrize the output layer of the first learned model to enable the recognition of new NEs; (ii) to leave the rest of the architecture unchanged, such that it is initialized with parameters transferred from the initial model; and (iii) to fine tune the network on the target data. Most importantly, we design a new NER approach based on sequence to sequence (Seq2Seq) models, which can intuitively work better in our progressive setting. We compare our approach with a Bidirectional LSTM, which is a strong neural NER model. Our experiments show that the Seq2Seq model performs very well on the standard NER setting and it is more robust in the progressive setting. Our approach can recognize previously unseen NE categories while preserving the knowledge of the seen data.

1 Introduction

Standard NER models are trained and tested on data with the same NE label set. In this paper, we explore a progressive setting, where (i) in the initial step, the model is trained on a dataset D_{S} with certain NE categories; and (ii) in the subsequent step, we further train the model on a dataset D_{T} containing all categories from D_{S} along with the new ones, without using the examples from D_{S} since we assume they are no longer available. Our goal is to learn to recognize the new NE categories in D_{T} while keeping the knowledge of previously seen categories.

This study is motivated by the application of NER model in industry scenarios, where different companies may aim to recognize different types of NEs in the text. For example, users interested in finance would probably target entities such as companies or banks while other users interested in politics would target entities such as senators, bills, ministries, etc. Nevertheless, there are certain types of commonly-seen NE categories such as locations or dates that most users would like to recognize. These common NEs are typically provided with more available labeled data. Hence, methods that can utilize such data in new domains are rather useful. Additionally, the possibility to only use the models trained on such data is very important as it avoids the NER designer to deliver most training data to the user for adaptation in new domains. The setting above describes a TL problem, where the domain remains unchanged but the output space of the task changes: a graphical illustration of our progressive task is shown in Fig. 1.

Since the current state-of-the-art of NER is provided by neural models (Lample et al., 2016; Chiu and Nichols, 2016), we focus on them. In particular, we implement a baseline model with a bidirectional LSTM (BLSTM) as it is effective and does not require a heavy feature engineering for the task. BLSTM achieves competitive performance in comparisons with many other more complex state-of-the-art models, being much less complex. A drawback of such model is that, although its prediction takes the

* The main part of this work was carried out when the author was at the Unviversity of Trento.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2181–2191
Santa Fe, New Mexico, USA, August 20-26, 2018.

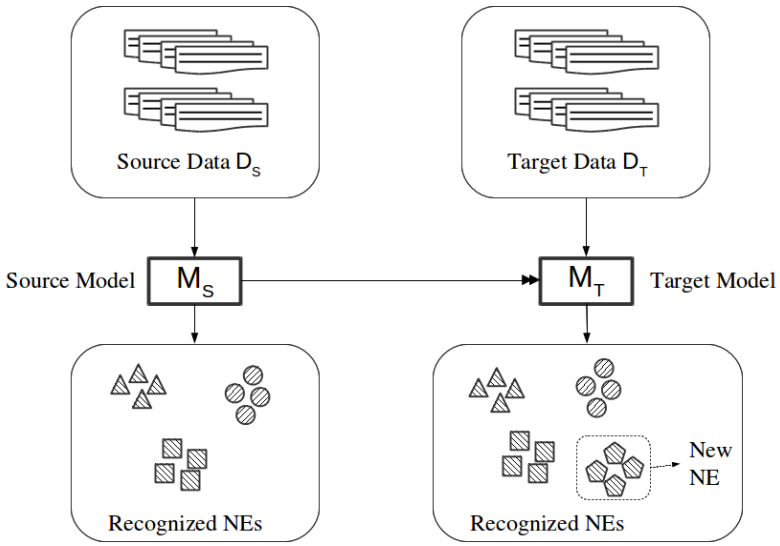


Figure 1: Progressive NER task setting.

surrounding context of the input word into account, it does not explicitly model the dependencies of the label sequence. To better capture these dependencies, we propose the use of Seq2Seq models for NER.

We speculate that Seq2Seq models can be suitable for our NER task for the following reasons: (i) they can be used to translate from a source language (the word sequence) to the target language (the label sequence); and (ii) on the decoder side, the current prediction is able to take the previous prediction into account, by feeding the previous output together with current input. This is an important property to build NERs, since the output sequence has some dependency assumptions among them (e.g., B-LOC cannot be followed by I-PER). Intuitively, Seq2Seq models can separate the mapping from input to output into two steps: encoding the information from input to an intermediate hidden space, and decoding the intermediate information to the output space. When the output space changes, as in our experiment, the model ought to better leverage the use of previously learned knowledge since the encoding of the input is still similar to the previous task.

Our solution to the task consists in (i) modifying the output layer of the model by adding neurons for the new NE category; (ii) transferring the parameters in the source model trained on data D_S to the target model that is trained on data D_T; and (iii) fine-tuning the obtained network on the target data. As for the proposed Seq2Seq model, we modify the output layer only on the decoder side. We conduct extensive experiments on progressive NER, analyzing the performance of both models. Our main contribution is twofold, we show that the Seq2Seq model is: (i) effective for the NER task, and (ii) more robust in the progressive NER setting as it does not forget previously learned knowledge about seen NE categories.

2 Related Work

In this section, we report related works on current state-of-the-art models for standard NER task, as well as the related works in the Transfer Learning and Progressive Learning areas.

2.1 Neural Models for NER

Early works on NER focused on engineering useful linguistic features for the task (Chieu and Ng, 2003; Carreras et al., 2003; Florian et al., 2003). The recent advances in the deep learning brought about new methods and produced a higher performance for the task. Current state-of-the-art models on standard NER are mostly neural models, in particular, Recurrent Neural Networks. These models incorporate word (sometimes also character) level embeddings and/or additional morphological word features for recognizing NE. Examples include CRF (Conditional Random Fields) over BLSTM (Bidirectional LSTM) model proposed by Lample et al. (2016) and BLSTM with Convolutional character feature extractor proposed by Chiu and Nichols (2016). We implement a BLSTM as the baseline model for pro-

gressive NER since it is an effective model that does not require a heavy feature engineering for the task. The performance of BLSTM model are reported in different works (Lample et al., 2016; Chiu and Nichols, 2016): this also makes it easier for us to conduct a comprehensive comparison with other works.

2.2 Seq2Seq Models

Although not previously used for NER task, Seq2Seq models have seen a great success in various applications, such as Neural Machine Translation by (Bahdanau et al., 2014; Luong et al., 2015), Speech Recognition (Lu et al., 2016; Zhang et al., 2017), Question Answering (Yin et al., 2016; Zhou et al., 2015) and so on. The idea of Seq2Seq model is to map a variable length source sequence to a fixed length vector representation by the encoder, and then remap it back to a variable length target sequence by the decoder. The Seq2Seq model was proposed by Cho et al. (2014) to learn phrase representations to be used in statistical Machine Translation. They reported empirical improvement of log-linear model performance while incorporating conditional probability of phrase pairs computed by the Seq2Seq model as additional features. Sutskever et al. (2014) also proposed a more general end-to-end approach to learn to map source sentences in one language into the translated sentences in the other.

To mitigate the bottleneck caused by mapping into a fixed-length vector, attention mechanism was proposed by Bahdanau et al. (2014). The attention mechanism works as a soft-alignment for the related parts in the source and target sequence. It is achieved by conditioning the prediction on a context vector that is calculated by the weighted sum of the encoder hidden states. Luong et al. (2015) further proposed global attention and local attention, to explore the effectiveness of different attention mechanisms. Our proposed model for NER is the same Seq2Seq architecture as proposed by Bahdanau et al. (2014), except that we use LSTM units for the encoder and decoder instead of simple RNN units. We also enforce an explicit alignment between the input word sequence and the output label sequence because the source and target sequences have a one-to-one mapping in the NER task.

2.3 Transfer Learning and Progressive Learning

Most machine learning methods assume the data for training and testing has the same feature space and distribution. TF is a technique that emerges to relax this assumption, allowing previously-learned knowledge to be used for a new task or in a new domain (Pan and Yang, 2010). Neural networks-based TL has proven to be very effective for image recognition (Donahue et al., 2014; Razavian et al., 2014). As for NLP, Mou et al. (2016) showed that TL can also be applied successfully on semantically equivalent NLP tasks. TL was applied on NER too, to transfer model and features on NER dataset with different NE entities (Qu et al., 2016; Kim et al., 2015).

Our learning paradigm falls in a more specific TL category – Progressive Learning. The Progressive Learning technique is adopted in multi-class classification by Venkatesan and Er (2016). They remodeled a network architecture (a single layer feed-forward network) by increasing the number of new neurons and interconnections while encountering unseen class labels in the dataset. The effectiveness of this technique is shown on several classic classification datasets. Our task is a sequence labeling task, where the independence assumption of output labels in a sequence does not hold. Hence, it cannot be treated as a set of classification tasks. We implement more sophisticated models and prove that the effectiveness of progressive learning techniques still holds.

3 Problem Formulation

In standard NER tasks, the model learns to maximize the conditional probability $P(Y|X)$ where $X = x_1, x_2, ..., x_n$ ($x_i \in \mathcal{X}$) is the input word sequence and $Y = y_1, y_2, ..., y_n$ ($y_i \in \mathcal{Y}$) is the output label sequence. $\mathcal{X}$ and $\mathcal{Y}$ represent the input and output space respectively. In our setting, in the initial step, the NER system is trained on the source dataset, D_{S}, which has C NE classes. Then, in the subsequent step, the model is presented with only the target dataset, D_{T}, which contains new examples annotated with the NE classes from the initial step and the E new classes. The initial model needs to adapt to be able to recognize the new E classes.

4 Models

We implemented (1) a BLSTM model, and (2) an attention-based Seq2Seq model for progressive NER. The input and output representation of both models is the expected to be the same.

4.1 Word & Character Embeddings

A word in the input sequence is represented by both its word-level and character-level embeddings. For the word-level representation, we use pretrained word embeddings to initialize an embedding lookup table to map the input word x (represented by an integer index) to a vector $\boldsymbol{w}$.

A character-level representation is typically used because the NER task is sensitive to the morphological traits of a word such as capitalization. They were shown to provide useful information for NER (Lample et al., 2016). We use a randomly initialized character embedding lookup table and then pass the embeddings to a character-level BLSTM (the details are described in Sec. 4.2) to obtain character level embedding $\boldsymbol{e}$ for x.

The complete representation of the t-th word $\boldsymbol{x}_t$ in the input sequence is the concatenation of its word-level embedding $\boldsymbol{w}_t$ and character-level embedding $\boldsymbol{e}_t$.

4.2 Bidirectional LSTM

BLSTM model is composed of a forward LSTM ($\overrightarrow{\text{LSTM}}$) and a backward LSTM ($\overleftarrow{\text{LSTM}}$), which read the input sequence (represented as word vectors described in the previous subsection) in both left-to-right and reverse order. The output of the BLSTM $\boldsymbol{h}_t$ is obtained by the concatenation of forward and backward output: $\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}_t}; \overleftarrow{\boldsymbol{h}_t}]$, where

$$\overrightarrow{\boldsymbol{h}_t} = \overrightarrow{\text{LSTM}}(\boldsymbol{x}_t, \overrightarrow{\boldsymbol{h}}_{t-1})$$

and

$$\overleftarrow{\boldsymbol{h}_t} = \overleftarrow{\text{LSTM}}(\boldsymbol{x}_t, \overleftarrow{\boldsymbol{h}}_{t+1})$$

$\boldsymbol{h}_t$ captures the left and right context for $\boldsymbol{x}_t$ and is then fed into a fully-connected layer. The output of the fully connected layer is $\boldsymbol{p}_t$. The final prediction is obtained with a softmax over all possible tags as

$$\text{P}(y_t = m|\boldsymbol{p}_t) = \frac{e^{W_{o,m}\boldsymbol{p}_t}}{\sum_{m' \in M} e^{W_{o,m'}\boldsymbol{p}_t}},$$

where W_o are parameters to be learned and M represents the set of all the possible output labels.

In the case of character-level BSLTM, the forward and backward LSTMs take the sequence of character vectors $[\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_k]$ as input, where k is the number of characters in a word. The final character level embedding $\boldsymbol{e}_t$ for a word at time t is obtained by concatenating $\overrightarrow{\boldsymbol{e}_t}$ and $\overleftarrow{\boldsymbol{e}_t}$.

4.3 Attention-based Encoder-Decoder

LSTM units are used in both encoder and decoder. On the encoder side, we use a BLSTM to capture the left and right context information of a word in the input sequence. The input to the encoder is the concatenation of word and character level embedding for each word, as the same as that for the BLSTM described above. The output $\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}_t}; \overleftarrow{\boldsymbol{h}_t}]$ of the BLSTM at each time step t is later used for attention calculation. On the decoder side, we use a single layer LSTM that generates label predictions step by step from the start to the end of the sentence. The last hidden state of the forward encoder RNN ($\overrightarrow{\boldsymbol{h}_t}$) is used as the initial decoder hidden state. At each decoding time step, the decoder hidden state $\boldsymbol{s}_t$ is calculated by $\boldsymbol{s}_t = \sigma(\boldsymbol{s}_{t-1}, \boldsymbol{c}_t, y_{t-1}, \boldsymbol{h}_t)$, where $\boldsymbol{s}_{t-1}$ is the previous decoder hidden state, $\boldsymbol{c}_t$ is the context vector and y_{t-1} is the previous prediction. $\boldsymbol{c}_t$ is calculated as weighted sum of the encoder hidden state sequence as follows: $\boldsymbol{c}_t = \sum_{j=1}^{T} \alpha_{ij} \boldsymbol{h}_j$, where T is the total length of a sentence. The weight α_{ij} is obtained by applying a scoring function on the previous decoder hidden state $\boldsymbol{s}_{t-1}$ and the aligned encoder hidden state $\boldsymbol{h}_j$. More details on how attention is calculated can be found in the work of Bahdanau et al. (2014).

Since we have a specific sequence to sequence task, where each element of the input at a position t maps to an element of the output at the same position t, we define an explicit alignment between encoder and decoder by feeding the encoder hidden state $\boldsymbol{h}_t$ at time t to the decoder. We also restrict the decoder to output exactly T label predictions by restricting the recurrent step of decoder LSTM as T. This ensures the one-to-one mapping between the input words and the output labels. The final prediction of y_t is made by passing s_t to a fully-connected layer and then applying a softmax function over the output of this layer (the same as described in Sec. 4.2).

4.4 Progressive Adaption

Our proposed method for progressive NER is through pre-training a source model, transferring parameters to the target model and then fine-tuning such model. In the initial step, the model is trained on labeled source data until the optimal parameters $\hat{\theta}^{\mathrm{S}}$ are obtained and saved. $\hat{\theta}^{\mathrm{S}}$ include the parameters regarding the output layer $\hat{\theta}^{\mathrm{S}}_{output}$ and those regarding all the other layers $\hat{\theta}^{\mathrm{S}}_{-output}$.

To recognize the new NE categories, the major modification is made on the fully-connected layer after the BLSTM, i.e., the output layer of the network. In the BLSTM architecture, the fully-connected layer maps the output $\boldsymbol{h}$ of BLSTM to a vector $\boldsymbol{p}$ of size nC. n is a factor depending on the tagging format of the dataset (e.g., $n = 4$ if the dataset is in the BIOES format [1], since for each NE category, there would be 4 output labels *B-NE*, *I-NE*, *E-NE* and *S-NE*). To recognize the target NEs, we build the same model architecture for all layers before the output layer and initialize all their parameters by $\hat{\theta}^{\mathrm{S}}_{-output}$ obtained in the initial step. We extend the output layer with nE nodes, where E is the number of new NE categories. These are initialized with weights drawn from the random distribution $X \sim \mathcal{N}(\mu, \sigma^2)$, where μ and σ are the mean and standard deviation of the pre-trained weights in the same layer. [2] This way, the associated weight matrix of the fully-connected layer $\mathbf{W}^s_o$ also updates from the original shape $nC \times p$ to a new matrix $\mathbf{W}^{s'}_o$ of shape $(nC + nE) \times p$. Note that $\hat{\theta}^{\mathrm{S}}_{output}$ and $\hat{\theta}^{\mathrm{S}}_{-output}$ are essentially the weights in the matrix $\mathbf{W}^s_o$ and $\mathbf{W}^s_{-o}$. The new model parameters are updated in the same way as in the initial step.

5 Experiments

5.1 Data

We used the CONLL 2003 NER dataset[3] by modifying it to suit our experiment settings. The original dataset includes news articles with four types of named entities – organization, person, location and miscellaneous (represented by `ORG`, `PER`, `LOC`, and `MISC`, respectively). The `O` label indicates tokens that do not belong to any of these four named entities. The entity labels are annotated in BIOES format.

For the purpose of our experiment, we shuffle and divide the training set into 80%/20% partitions as D_{S} and D_{T} for the initial and the subsequent steps, as we assume that the size of the target data would be less than that of the source data. The validation and the test sets are the same in both steps in terms of the sentence instances, but different in label annotation. In particular, we create four datasets by (i) selecting one of the four categories as the new NE category to be recognized in the subsequent step; and (ii) substituting the labels of such category with `O` (not an entity) in D_{train_S}, D_{valid_S} and D_{test_S}. For example, when we use `LOC` as the new NE, we replace all the `LOC` label annotations with `O`, where they appear in D_{train_S}, D_{valid_S} and D_{test_S}. Instead, in D_{train_T}, D_{valid_T} and D_{test_T}, we keep `LOC` label annotation as is. A summary of label statistics for each case is shown in Table 1.

Apart from dividing the dataset into two components D_{S} and D_{T}, we do not perform any specific pre-processing of the dataset, except for replacing all digits with 0 to reduce the size of the vocabulary.

[1] The BIOES format (short for Begin, Inside, Outside, End and Single) is a common tagging format for tagging tokens.

[2] We initialize the weights using the approach producing the higher accuracy on the validation set. Our initizialization options are (i) set all parameters to zeros; (ii) using the random distribution $X \sim \mathcal{N}(\mu, \sigma^2)$, where $\mu = 0.0$ and $\sigma = 1.0$; and (iii) using the random distribution $X \sim \mathcal{N}(\mu, \sigma^2)$, where μ and σ are the mean and standard deviation of the pre-trained weights in the same layer.

[3] https://www.clips.uantwerpen.be/conll2003/ner/

Set		LOC	PER	ORG	MISC
Train	Init.	0	8948	8100	3686
	Sub.	1637	2180	1925	907
Valid (Init./Sub.)		0/2094	3146	2092	1268
Test (Init./Sub.)		0/1925	2773	2496	918

(a) LOC as new NE

Set		PER	LOC	ORG	MISC
Train	Init.	0	6630	8106	3706
	Sub.	2169	1667	1919	887
Valid (Init./Sub.)		0/3149	2094	2092	1268
Test (Init./Sub.)		0/2773	1925	2496	918

(b) PER as new NE

Set		ORG	PER	LOC	MISC
Train	Init.	0	8999	6671	3631
	Sub.	1992	2129	1626	962
Valid (Init./Sub.)		0/2092	3146	2094	1268
Test (Init./Sub.)		0/2496	2773	1925	918

(c) ORG as new NE

Set		MISC	PER	ORG	LOC
Train	Init.	0	8782	7992	6662
	Sub.	920	2346	2033	1635
Valid (Init./Sub.)		0/1268	3146	2092	2094
Test (Init./Sub.)		0/918	2773	2496	1925

(d) MISC as new NE

Table 1: Number of labels after customizing CONLL dataset for the progressive NER experiments. Init. and Sub. represent the initial and the subsequent steps, respectively.

Parameter	Value	Range
$batch\_size$	128	[16, 32, 64, 128, 256]
$clip\_value$	5.0	([5.0, 50.0], 1.0)
$char\_hidden\_size$	25	([5, 25], 5)
$word\_hidden\_size$	128	[64, 72, 128, 256, 512]
$dropout\_rate$	0.5	([0.1, 0.8], 0.1)
$learning\_rate$	0.005	([0.0025, 0.01], 0.0025)

(a) BLSTM model

Parameter	Value	Range
$batch\_size$	16	[16, 32, 64, 128, 256]
$clip\_value$	50.0	([5.0, 50.0], 1.0)
$char\_hidden\_size$	15	([5, 25], 5)
$word\_hidden\_size$	72	[64, 72, 128, 256, 512]
$dropout\_rate$	0.4	([0.1, 0.8], 0.1)
$learning\_rate$	0.02	([0.0025, 0.01], 0.0025)

(b) Seq2Seq model

Table 2: Value range and final value of hyperparameters for the BLSTM and the Seq2Seq models.

5.2 Models

We use 100 dimension GLOVE pretrained embedding[4] to initialize the word embedding lookup table. Since we do not lowercase the tokens in the input sequences, the words without direct mappings to the pretrained word embeddings are initialized as their lowercase counterparts, if found in the pretrained word embeddings. We replace the infrequent words, i.e., appearing less than two in the training set or in the test set, with *<UNK>*. The word embedding for *<UNK>* is randomly initialized by drawing from a uniform distribution between $[-0.25, 0.25]$.

We perform hyperparameter optimization by a Random Search (Bergstra and Bengio, 2012) on different intervals for each hyperparameter for both models, including the size for the character and word level embeddings of the BLSTMs, the batch size, the learning rate, the dropout rate and the maximum value for gradient clipping. Tab. 2 shows the intervals and the final values for each hyperparameter, where the final values are the ones giving the best performance on the validation set (D_{valid_S}) in each experiment setting.

5.3 Training

Both models are implemented in PyTorch (Paszke et al., 2017). [5] We use Stochastic Gradient Descent with a momentum of 0.9 to minimize the multi-class cross entropy. We apply dropout layer in both models to control overfitting. During each step, we allow a maximum epoch number of 200. The early stopping of the training is enforced if the F1 score on the validation set does not improve for 30 epochs. From our preliminary experiments, we found out that keeping the transferred parameters fixed during the subsequent step usually produces worse results for both models. Thus, in the subsequent step, the transferred parameters are not fixed but updated together with the rest of the network. We evaluate the F1 score of the models using the CONLL script.

	Init.	*Sub.*	*Rand.*
MISC	78.53	75.96	75.15
ORG	87.67	86.44	84.01
PER	96.31	96.59	95.46
LOC	-	88.58	87.55
Overall	90.03	89.00	87.51

(a) LOC as new NE

	Init.	*Sub.*	*Rand.*
LOC	90.62	90.32	86.83
MISC	77.41	75.72	72.96
ORG	83.88	83.22	79.68
PER	-	91.69	91.56
Overall	85.27	86.56	83.92

(b) PER as new NE

	Init.	*Sub.*	*Rand.*
MISC	74.80	74.01	73.36
LOC	89.99	88.64	87.66
PER	93.46	93.31	92.32
ORG	-	77.61	76.42
Overall	88.35	84.39	82.57

(c) ORG as new NE

	Init.	*Sub.*	*Rand.*
LOC	91.12	90.03	88.41
ORG	83.75	82.96	75.91
PER	94.36	94.10	91.86
MISC	-	70.48	70.70
Overall	90.42	86.20	82.13

(d) MISC as new NE

Table 3: Test F1 of the BLSTM model on each category.

	Init.	*Sub.*	*Rand.*
MISC	80.76	78.73	75.11
ORG	88.43	87.36	83.24
PER	95.10	95.03	92.90
LOC	-	91.33	90.52
Overall	91.44	90.82	88.20

(a) LOC as new NE

	Init.	*Sub.*	*Rand.*
LOC	92.20	92.29	89.84
MISC	78.91	78.98	74.98
ORG	87.55	86.84	84.03
PER	-	93.30	92.44
Overall	87.82	88.96	86.46

(b) PER as new NE

	Init.	*Sub.*	*Rand.*
MISC	78.31	79.19	75.29
LOC	91.97	92.00	90.04
PER	94.79	94.67	92.92
ORG	-	84.11	82.12
Overall	90.51	88.47	87.14

(c) ORG as new NE

	Init.	*Sub.*	*Rand.*
LOC	92.89	92.72	90.36
ORG	88.47	87.68	83.01
PER	95.07	94.96	93.35
MISC	-	74.28	77.19
Overall	92.00	89.28	86.95

(d) MISC as new NE

Table 4: Test F1 of the Seq2Seq model on each category.

5.4 Results

Our baseline BLSTM model performs in-line with the BLSTM models reported by Lample et al. (2016) or by Chiu and Nichols (2016). We obtained an overall F1 score of 89.74 on traditional setting of NER. The Seq2Seq model outperformed the baseline model by 1.24 points in F1, reaching 90.98. The latter is comparable to some of the state-of-the-art results reported on the CONLL dataset.

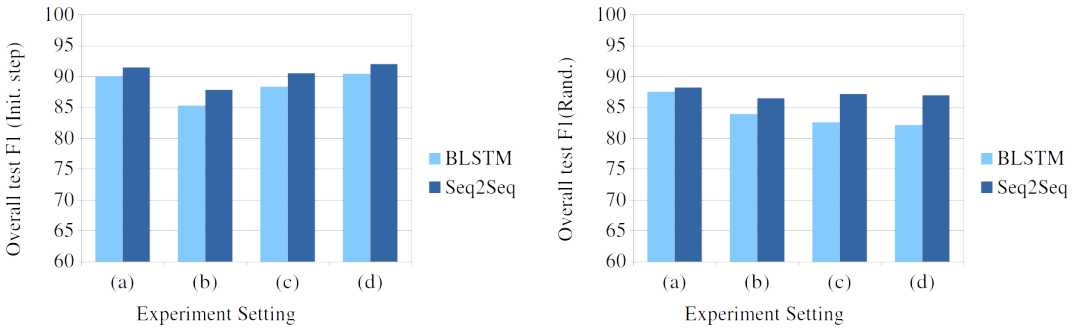
5.4.1 Performance on Source Data

The performance of the BLSTM and Seq2Seq models in our progressive learning setting is shown in tables 3 and 4, respectively. Sub-level tables (a), (b), (c) and (d) report the test F1 in different experiment settings – recognizing different target new NE in the subsequent step. *Init.* indicates the F1 scores in the initial step, while *Sub.* indicates the scores in the subsequent step. *Rand.* reports the test F1 scores obtained by the same model architecture but with randomly initialized parameters and trained with all the target data in the subsequent step.

The training in the initial step and for the model with randomly initialized parameter in the subsequent step can be seen as a standard NER task. We further report a comparison of the performance of both models in the initial step as well as with the randomly initialized parameters in Figure 2. We can see that the Seq2Seq model outperforms the BLSTM model in all experiment settings. This supports our argument that the Seq2Seq model can be a better approach for the NER task than BLSTM due to its ability to exploit the previous predicted labels.

[4]http://nlp.stanford.edu/data/glove.6B.zip

[5]The source code is available at https://github.com/liah-chan/sequence2sequenceNER



(a) Overall test F1 in the initial step

(b) Overall test F1 in the subsequent step (Rand.)

Figure 2: A comparison of between the BLSTM and Seq2Seq models on standard NER task.

5.4.2 Transferred Parameters vs. Randomly Initialized Parameters

By construction, there is always a label disagreement between the data of the initial and of the subsequent steps for the new NE, as it is annotated as `O` in the data for the initial step. We can derive some interesting observations from the comparison between the F1 of the model using randomly initialized parameters (Column *Rand.* in each table) and the F1 of the model using transferred parameters in the subsequent step (Column *Sub.* in each table). In almost all cases, using transferred parameters produces better performance, by around 2 to 3 points in F1. In terms of the target new NE category, the models using transferred parameters are able to perform better than the models with only randomly-initialized parameters. This is because we allow the transferred parameters to be updated together with the other model parameters in the subsequent step. Such approach can recover from the adverse knowledge learned in the initial step with regard to treating the target new label as `O`. The only exception is when using `MISC` as target new NE. The F1 obtained by randomly initialized parameters for both BLSTM and Seq2Seq model is higher than the one obtained by the models using transferred parameters. This may be explained by the fact that the `MISC` category is a collection of NEs that do not fit into other categories. Thus, the context of such NEs may vary a lot from one to another, making the recovery from the previously-learned adverse knowledge on this NE category more challenging, e.g., its new context and information in the subsequent step are sparser.

We allow a maximum of 200 epochs in the subsequent step while forcing the training to stop when the F1 score on the validation set does not improve for more than 30 epochs. Even if we increase the training epochs to 60 for the model using randomly initialized parameters, transferred parameters still perform better. We also observed that the model with transferred parameters converges faster during the subsequent training, in around 20 epochs. This suggests that our progressive learning technique enables the models to more accurately predict the new categories and in a shorter time. That is, the transferred parameters not only help the model to reach a superior performance, but also provide a better initialization for the model to converge faster.

5.4.3 Original NE Classes vs. New Target NE Class

The improvement or decrease of the F1 for the original NE categories is an indicator of how well the model is able to preserve the learned knowledge transferred from the initial step. In Table 5, we report the difference of test F1 between the initial and the subsequent steps, for each of the original NE categories, using the BLSTM and Seq2Seq models. In this table, lower value means a lower loss in the F1 score, i.e., the model is better in preserving the performance on the NE categories that are seen in the initial step. In almost all cases, the Seq2Seq model suffers less degradation in F1. In some cases, the Seq2Seq model is able to further learn from the subsequent data and achieve better performance on these original NE categories. For example, when using `PER` as the new NE category, the Seq2Seq model is able to improve the performance on `LOC` and `MISC`, while the BLSTM model decreases its F1.

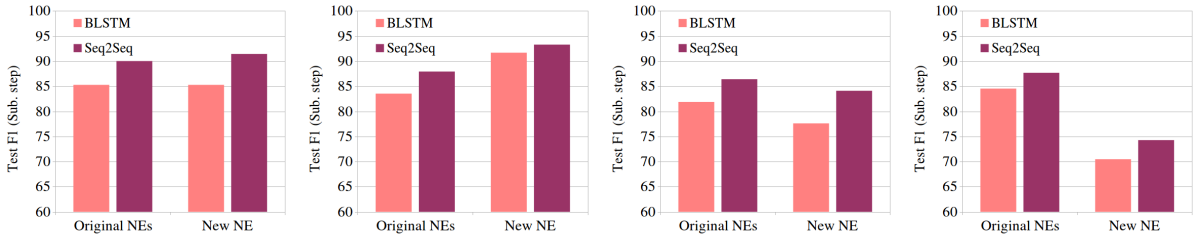
The ability of the model in terms of learning to recognize the new NE category is directly reflected by the test F1 score in the subsequent step of the target NE category of different settings, as shown in

	LOC as new NE		PER as new NE		ORG as new NE		MISC as new NE	
	BLSTM	Seq2Seq	BLSTM	Seq2Seq	BLSTM	Seq2Seq	BLSTM	Seq2Seq
MISC	2.51	2.03	1.69	-0.07	0.79	-0.88	-	-
ORG	1.23	1.07	0.66	0.71	-	-	0.79	0.79
PER	-0.28	-0.07	-	-	0.15	0.12	0.26	0.11
LOC	-	-	0.30	-0.09	1.35	-0.03	1.09	0.17

Table 5: Difference of the test F1 score from the initial step to the subsequent step (lower value is better, while a negative value means that the model is able to further improve).

	BLSTM	Seq2Seq
LOC	88.58	91.33
PER	91.96	93.30
ORG	77.61	84.11
MISC	70.48	74.28

Table 6: Test F1 on the new target NE in the subsequent step.



(a) LOC as new NE (b) PER as new NE (c) ORG as new NE (d) MISC as new NE

Figure 3: Test F1 scores obtained on the original NEs and the target new NE by the BLSTM and the Seq2Seq model.

Table 6. As it is evident from the table, in all cases, the Seq2Seq model produces better test F1 scores by 2 to 3 points at least. In the case of ORG label, the performance of the Seq2Seq model surpasses that of the BLSTM model by nearly 7 points. By comparing both the performance on the original labels and on the target new label, as shown in Figure 3, we see that the performance of Seq2Seq model is better than the baseline BLSTM in both the original and the new NE classes. It can be concluded that Seq2Seq model is more robust in the subsequent step regarding recognizing the new target NE category while it does not degrade the performance of the original NE categories.

6 Conclusions

In this work, we propose to use attention-based Seq2Seq models for both standard NER task as well as the progressive NER task, which is a more practical scenario of industry level applications of NER. We argue that the Seq2Seq model is suitable for both settings. Our experimental results support our claim as the Seq2Seq model outperforms BLSTM in both standard and progressive NER settings. Further analyses show that the Seq2Seq model is better in both preserving the knowledge learned from the source data, and in learning the new knowledge from the target data. We also demonstrated the effectiveness of the proposed progressive learning technique in the NER task. In future work, we would like to experiment with other datasets as well as extend our methods to target data that is only annotated with the new NE category. It would be also interesting to apply our model and technique to other areas such as domain adaptation or few-shot learning.

Acknowledgements

This research was partially supported by Almawave S.r.l. We would like to thank Giuseppe Castellucci, Andrea Favalli, and Raniero Romagnoli for inspiring this work with useful discussions on neural models for applications to real-world problems in the industrial world.

References

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

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305.

Xavier Carreras, Lluís Màrquez, and Lluís Padró. 2003. Learning a perceptron-based named entity chunker via online recognition feedback. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 156–159.

Hai Leong Chieu and Hwee Tou Ng. 2003. Named entity recognition with a maximum entropy approach. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 160–163.

Jason P. C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *TACL*, 4:357–370.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 647–655.

Radu Florian, Abraham Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 168–171.

Young-Bum Kim, Karl Stratos, Ruhi Sarikaya, and Minwoo Jeong. 2015. New transfer learning techniques for disparate label sets. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 473–482.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 260–270.

Liang Lu, Xingxing Zhang, and Steve Renals. 2016. On training the recurrent neural network encoder-decoder for large vocabulary end-to-end speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, March 20-25, 2016*, pages 5060–5064.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in NLP applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 479–489.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Lizhen Qu, Gabriela Ferraro, Liyuan Zhou, Weiwei Hou, and Timothy Baldwin. 2016. Named entity recognition for novel types by transfer learning. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 899–905.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. CNN features off-the-shelf: An astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2014, Columbus, OH, USA, June 23-28, 2014*, pages 512–519.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Rajasekar Venkatesan and Meng Joo Er. 2016. A novel progressive learning technique for multi-class classification. *Neurocomputing*, 207:310–321.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2972–2978.

Yu Zhang, William Chan, and Navdeep Jaitly. 2017. Very deep convolutional networks for end-to-end speech recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*, pages 4845–4849.

Xiaoqiang Zhou, Baotian Hu, Qingcai Chen, Buzhou Tang, and Xiaolong Wang. 2015. Answer sequence learning with neural networks for answer selection in community question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 713–718.

1

Responding E-commerce Product Questions via Exploiting QA Collections and Reviews

Qian Yu, Wai Lam, Zihao Wang
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
{yuqian, wlam, zhwang}@se.cuhk.edu.hk

Abstract

Providing instant responses for product questions in E-commerce sites can significantly improve satisfaction of potential consumers. We propose a new framework for automatically responding product questions newly posed by users via exploiting existing QA collections and review collections in a coordinated manner. Our framework can return a ranked list of snippets serving as the automated response for a given question, where each snippet can be a sentence from reviews or an existing question-answer pair. One major subtask in our framework is question-based response review ranking. Learning for response review ranking is challenging since there is no labeled response review available. The collection of existing QA pairs are exploited as distant supervision for learning to rank responses. With proposed distant supervision paradigm, the learned response ranking model makes use of the knowledge in the QA pairs and the corresponding retrieved review lists. Extensive experiments on datasets collected from a real-world commercial E-commerce site demonstrate the effectiveness of our proposed framework.

1 Introduction

Many E-commerce sites provide product information on product pages. Under a particular product page, users can ask questions about the corresponding product. Other experienced or qualified users can voluntarily provide answers. Thus, for each product, a product-specific question-answer (QA) collection is commonly available in E-commerce sites. However, the amount of QA pairs in the QA collection associated with a product is small since it is product-specific. In addition to the QA collection, users may write reviews about the product and a product-specific review collection is also commonly available. Figure 1 depicts an example, from a real-world E-commerce site, of a product page of a bluetooth car device. In this product page, the middle region contains the question-answer collection. The bottom region contains the review collection. Users may browse those two information sources for obtaining useful or insightful information about the corresponding product.

Both information sources mentioned above would be helpful for providing instant responses to questions newly posed by users about the corresponding product. Consider a new question "Is this device compatible with 2004 VW Jetta SportWagen?", a semantically similar question can be found in the existing QA collection of this product. Specifically the first question is a good match with the new question. Thus, the user posed answer for that question can be extracted as the response. On the other hand, consider another new question "Can I use iPhone 4s with it?". This new question cannot match any existing question well. In spite of that, the review sentence "My iPhone 4S connected with the unit and uploaded my contacts with no issues." found in the review collection of the corresponding product can provide informative content which can be retrieved as a response.

In this paper, we investigate the problem of responding E-commerce product questions newly posed by users. We aim at providing responses before any answer is posed by other users. Such kind of instant responses can improve user satisfaction. As aforementioned, two commonly available information

The work described in this paper is substantially supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Code: 14203414).

Proceedings of the 27th International Conference on Computational Linguistics, pages 2192–2203
Santa Fe, New Mexico, USA, August 20-26, 2018.

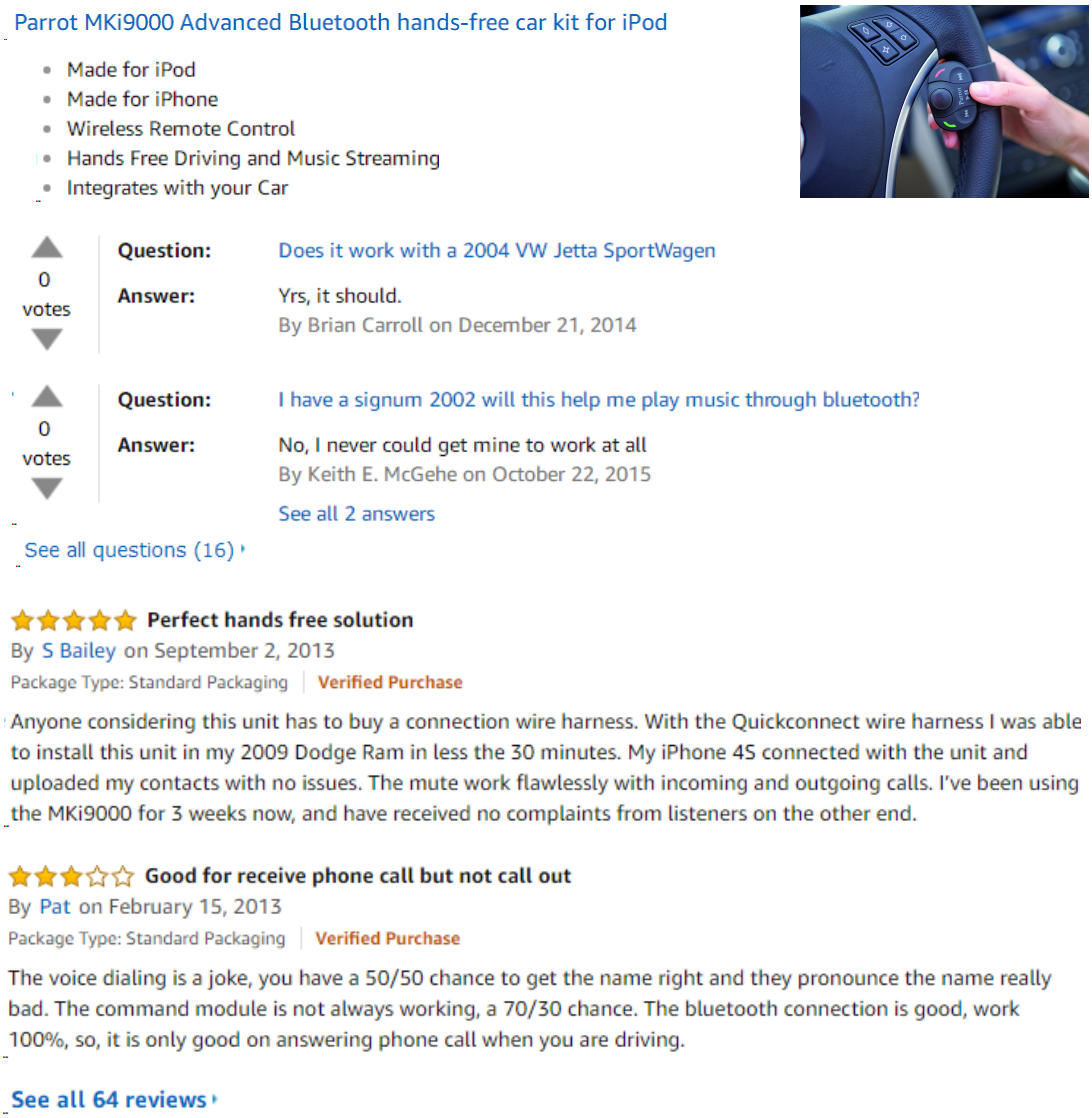


Figure 1: A product page of a bluetooth car device.

sources, namely, question-answer collections and review collections contain valuable information for generating instant responses for product questions. Consequently, given a question, our framework can return a ranked list of snippets serving as the automated response, where each snippet can be a sentence from reviews or an existing question-answer pair.

Community Question Answering (CQA) approaches can be employed, but they can only make use of the QA collection of the corresponding product, which typically contains just a small amount of QA pairs. Furthermore, similar questions associated with other products are not helpful due to different product specifications. Another limitation of CQA methods is that they cannot make use of the review collection which is another important information source for generating responses. Another possible approach is to employ a question answering learning approach such as QA-LSTM (Tan et al., 2016). But these QA models typically cannot effectively exploit reviews due to the heterogeneous nature of answer collections and review collections. Recently a chatbot for E-commerce sites known as SuperAgent has been developed (Cui et al., 2017). SuperAgent considers both QA collections and reviews when answering questions. However, it employs separated modules for each of the information sources without mutual coordination. Moreover, it requires external knowledge and a large volume of annotated data. Some models based on Mixture of Expert (MoE) (McAuley and Yang, 2016; Wan and McAuley, 2016) are proposed for handling product questions. Although the learning procedures of these models involve

QA collections and reviews, these models assume that the candidate answer set only comes from existing QA pairs containing the correct answers. Such setting is not practical for instant response generation.

We propose a new framework which is able to automatically respond product questions newly posed by users via exploiting question-answer collections and review collections in a coordinated manner. One major subtask in our framework is question-based response review ranking which aims at extracting reviews that are suitable for providing the response to the given question. Learning for response review ranking in this problem setting is challenging since no labeled review sample is available. The existing QA pairs in the corresponding QA collection contains knowledge for response ranking. These QA pairs and the corresponding retrieved review lists can be exploited as distant supervision for question-based response ranking. The relationship between such available QA pairs and reviews can be modeled for facilitating the learning of question-based review ranking.

2 Related Work

Product question answering is an emerging topic. Yu et al. (2012) extract hierarchical structure from the product review collection, and then select sentences from reviews based on the structure. Their model only focuses on review collections. Community Question Answering (CQA) (Zhou et al., 2011) approaches can be adopted to tackle this problem. For example, Zhou et al. (2015) propose to learn continuous word embeddings based on the QA corpus incorporating metadata such as category information, and the learned word embedding can be applied for question retrieval in CQA platform. Chen et al. (2018) encode the question text together with users' social interactions for handling the lexical gap among questions. A random walk based learning method is designed to facilitate the similarities evaluation via the recurrent neural network. One shortcoming of these CQA methods for tackling the task of E-commerce product questions is that they can only make use of the typically small QA collection of the corresponding product, and similar questions associated with other products are not helpful due to different specifications.

QA models (Shen et al., 2017; Yang et al., 2016) try to capture the relation between questions and answers. QA-LSTM (Tan et al., 2016) is developed for question answer matching via bidirectional LSTM with max pooling. The QA model proposed in (Wang and Jiang, 2016) also utilizes LSTM. Learning-to-rank model has also been adopted in some method for question-answer matching, such as in (Surdeanu et al., 2008). Generally existing QA models cannot handle the heterogeneous nature of answer collections and review collections in the problem setting investigated in this paper. McAuley and Yang (2016) propose to exploit product reviews for answer prediction via a Mixture of Expert (MoE) model. This MoE model makes use of a review relevance function and an answer prediction function. One restriction of this model is that it can only be used for answer selection given a candidate answer set. Although the QA collections and review collections are involved in the learning procedures, one assumption is that a candidate answer set containing the correct answers is available for answer selection. Such setting is not practical for instant response generation. In addition, both the above mentioned models only make use of information from reviews, while existing question-answer pairs of the corresponding product are ignored. A chatbot for E-commerce sites known as SuperAgent has been developed (Cui et al., 2017). SuperAgent considers both QA collections and reviews when answering questions. However, there is no mutual coordination for the response generation from different information sources, and the response from SuperAgent cannot be presented as a ranked list of snippets. Moreover, the training procedures of some modules in this system require external knowledge and a large volume of annotated data, including a set of similar question pairs.

3 Our Proposed Framework

3.1 Framework Overview

Our proposed framework exploits QA collections and review collections for instant responses of E-commerce product questions newly posed by users. The output is a ranked list of snippets providing relevant information related to the response of the question. Each snippet can be a sentence from reviews

or an existing QA pair. There are two main components in our framework, namely, response review ranking component and response generation component.

The response review ranking component tackles a major subtask whose aim is to extract response reviews that are suitable for providing the response to the given question. Learning for response review ranking in this problem setting is challenging and it cannot be handled by supervised learning. The reason is that there is no ground truth question-based reviews which can be used for training. The existing QA pairs and the corresponding retrieved review lists contains knowledge for response review ranking. To learn a question-based review ranking model, the relationship between the available QA pairs and reviews needs to be modeled. To this end, we develop a distant supervision paradigm for incorporating the knowledge contained in QA collections into response review ranking. An adapted Learning to Rank (LTR) method is designed in this distant supervision paradigm.

The aim of the response generation component is to generate a list of snippets which serve as the response of a given question newly posed by a user. Both the existing question-answer pairs in the QA collection of the corresponding product and question-based ranked reviews from the review collection of the corresponding product are considered in a coordinated manner. Semantically similar questions are extracted, if exist, via processing the question-answer pairs in the QA collection. The retrieved question-answer pairs are reformulated based on the corresponding review collection, and then integrated into the response review ranking model.

3.2 Response Review Ranking Model

As mentioned above, the aim of response review ranking is to extract reviews that can provide the response to a given question. Consider a E-commerce product t with the corresponding review collection R_t. Given a new question q_0 associated with this product, we wish to obtain the score $S(q_0, r)$ for each review $r \in R_t$. Based on this score, a ranked list of response reviews $L_{r|q_0} = (r_{1|q_0}, r_{2|q_0}, ...)$ can be obtained. A learning model is designed for tackling this subtask. Different from answer selection in CQA, response review ranking cannot be tackled via supervised learning. Suppose we have an existing question q in the question collections, the only available data is the question-answer pair (q, a) while no ground truth question-review pair (q, r) is available for the learning of this component. This setting makes this subtask challenging. Thus, we model the relationship between (q, a) pairs of the corresponding product t and the review set R_t. Such modeling can be regarded as a distant supervision paradigm based on the knowledge in the existing QA pairs for the learning of question-based response review ranking. In this way, we exploit the knowledge in QA pairs in the corresponding QA collection and learn the response review ranking model $p(r|q) \triangleq S(q, r)$, where $S(q, r)$ is the ranking score of the review r given the question q. An adapted Learning to Rank (LTR) method is designed in this distant supervision paradigm, with various types of features are designed.

3.2.1 Distant Learning Paradigm

Training Instance Preparation for Question-based Response Review Ranking. From the QA collections, we can obtain question-answer pairs. Each question-answer pair can provide knowledge about the corresponding product, and neither the question nor the answer alone is adequate for conveying the same semantics as the question-answer pair. We use question-answer pairs as queries for retrieving and ranking reviews, and the top ranked reviews are utilized as the training data for the question-based LTR model.

Given a question-answer pair (q, a), the review collection of the corresponding product is processed. We employ the Positional Language Model (PLM) (Lv and Zhai, 2009) to rank the reviews using (q, a) as the query. Formally, given a review r and a term position k, the PLM of r at this position is:

$$p(w|r, k) = \frac{c'(w, k)}{\sum_{w' \in V} c'(w', k)} \quad (1)$$

where $c'(w,k)$ is the pseudo word count of the word w at the position k:

$$c'(w,k) = \sum_{j=1}^{|r|} c(w,j) f(k,j) \tag{2}$$

and $f(k,j)$ is a weight given to another position. Basically, the closer are the position k and j, the larger value is assigned by $f(k,j)$. In our implementation, we use the Gaussian kernel for $f(k,j)$ as mentioned in (Lv and Zhai, 2009). To obtain an available relevant review ranking list, we design a modified relevance score of PLM:

$$S((q,a),r) = \max_{i\in[1,|r|]} \{S(q,r,i) + S(a,r,i)\} \tag{3}$$

$$= \max_{i\in[1,|r|]} \left\{ -\sum_{w\in V} p(w|q) \log \frac{p(w|q)}{p(w|r,i)} - \sum_{w\in V} p(w|a) \log \frac{p(w|a)}{p(w|r,i)} \right\} \tag{4}$$

We denote the review ranking list based on (q,a) as

$$L_{r|q,a} = (r_{1|q,a}, r_{2|q,a}, ..., r_{n|q,a}) \tag{5}$$

where $r_{i|q,a}$ is the i-th review sentence ranked by the question-answer pair (q,a). The top ranked reviews in this QA-based review ranking list $L_{r|q,a}$ are more relevant to (q,a) and are useful for capturing the knowledge of response review ranking. Thus, we truncate the review ranking list via retaining only the top M ranked reviews as the response reviews.

Since our goal is question-based review ranking, each $L_{r|q,a}$ is integrated with the question q and a training instance is packaged as $(q, L_{r|q,a})$. Here the answer is removed from the training instance even though it participates in the generation of $L_{r|q,a}$. The reason is that there is no available answer for questions newly posed by users during operation or testing. The response review ranking ability of the question-answer pair (q,a) is captured in the response review ranking list $L_{r|q,a}$, and this training instance can be utilized to guide the review ranking based on only the question text. Another issue about the training instance preparation is that different answers have different quality in regard to the question, and the instance based on answers with high quality should have more influence on the learning of the response review ranking model. To achieve this goal, for each question q, we sample training instances from its related response review ranking set $\{L_{r|q,a_1}, L_{r|q,a_2}, ...\}$. The sampling probability of the ranking list $L_{r|q,a_i}$ is the probability of the answer a given the question q, namely $p(a|q)$. We utilize a learning-to-rank model trained with the QA collections to generate $p(a|q)$, and this model makes use of the features as described in Section 3.2.2.

Question-based Learning to Rank. Based on the sampled $L_{r|q,a}$ instances, we utilize the pair-wise learning to rank model to make use of these response review ranking lists to the maximum extent. For each question-review pair, we make use of the features presented in Section 3.2.2. These features are concatenated as a vector denoted as $f^q(r)$. The details about this designed features will be described in Section 3.2.2.

Suppose for two response review r_i and r_j, we obtain the feature vectors $f^q(r_i)$ and $f^q(r_j)$. The probability that the review r_i is ranked higher than the review r_j is formulated as:

$$P^q_{ij} = P(r_i \rhd r_j) = \frac{1}{1 + e^{-\beta[f^q(r_i) - f^q(r_j)]}} \tag{6}$$

To learn a probability model for question-base review ranking, we minimize the cross entropy of the ranking probability generated from the model and the ranking probability from the training instances. Formally, the objective function can be written as:

$$O = -\bar{P_{ij}} \log P_{ij} - (1 - \bar{P_{ij}}) \log(1 - P_{ij}) \tag{7}$$

In our implementation, we utilize the training method with boosting in LambdaMART (Wu et al., 2010) to update the parameters in our response review ranking model.

3.2.2 Feature Design

We describe the features utilized in our LTR models.

- **Features based on Likelihood.** Generation likelihood (Jin et al., 2002) plays an important role in information retrieval. Two related features are designed, namely, the question likelihood given review and the review likelihood given question. For a question q and a review r, the question likelihood can be computed by multiplying the likelihood of query terms w given r, denoted as $p_\lambda(w|r)$. The likelihood is smoothed by Jelinek-Mercer method:

$$p_\lambda(w|r) = (1-\lambda)p(w|r) + \lambda p(w|C_r) \tag{8}$$

  where $p(w|C_r)$ is the probability of the term w in the review collection. Similarly, we can formulate the second feature about the likelihood of the reviews given the question q.

- **Features based on Aspect-based Similarity.** For text data from E-commerce platforms, aspects are an important concept which captures features or attributes about products. Two features related to aspects are designed. The first feature is derived from the aspect discovery model based on Latent Dirichlet Allocation (LDA) (Zhao et al., 2010). Trained with the text collection containing reviews, questions and answers, this model can transform each given text into an aspect representation. Let the representations for a question q and a review r be denoted as $v(q)$ and $v(r)$. We employ the cosine similarity score of these two representations to model the aspect-based similarity between the question and the review. The second feature is based on the Restricted Boltzmann Machine (RBM) (Wang et al., 2015) trained as an aspect modeling. The learned hidden representation from RBM contains aspect information and can be obtained efficiently. Similar to the aspect-based similarity in the first feature, we can also compute a similarity score based on the RBM as the second feature.

- **Features based on Word Embedding** Word embedding techniques map each term to a distributed representation capturing semantics of text. Three features are designed according to different types of word embedding. For the first two features, we adopt a set of pre-trained word embedding, known as Global Vectors for word representation (GloVe)(Pennington et al., 2014). Given a question $q = (t_1^q, t_2^q, ...)$, each term t_i^q in q can be represented as a word embedding denoted as $v(t_i^q)$. Then the question q can be represented as the average of each term representation. The review sentence is also represented in the same way. Then the first feature is obtained by computing the inner product of these two representations reflecting the semantic similarity between the question q and the review r. Another representation of question is to use the largest value in each dimension among all the term vectors as the value of the corresponding dimension in the question representation. Then the second feature can be obtained in a similar manner. Besides the pre-trained word embedding, we also implement an autoencoder with the reconstruction function as the objective function. This autoencoder is trained via the collection of reviews, questions and answers, for capturing the intrinsic elements of text useful for reconstruction. In this way, we obtain a new embedding of questions and reviews, and the corresponding similarity score can be obtained as the third embedding-based feature.

- **Features based on Word Count** Some simple features also contribute to some extent. These features include the word count of a review, the ratio of the review word count and the question word count, and the number of matched words between the question and the review.

3.3 Response Generation

As mentioned above, the aim of the response generation component is to generate a ranked list of snippets treated as the response to the given question. The snippets are extracted from the existing QA collection and the review collection of the corresponding product. To obtain this snippet ranking list, we integrate the question-answer pair result and the review ranking result in a unified ranking model so that the heterogeneous nature between these two collections can be handled in a coordinated manner.

Consider an E-commerce product t with the QA collection Q_t and the review collection R_t. Given a question q_0 associated with this product newly posed by a user, we retrieve semantically similar questions from the QA collection Q_t via searching the question-answer pairs. Specifically, we utilize the Positional Language Model (PLM) (Lv and Zhai, 2009) for question retrieval and obtain a ranked list of retrieved existing questions denoted by $L_{q^*,a^*|q}$. The next step is to utilize the trained LTR model described in Section 3.2 to reformulate each retrieved QA pair $(q_i, a_i) \in L_{q^*,a^*|q}$. The rationale is to narrow the syntactic gap between the QA pairs and the review collection so that QA pairs and reviews can be ranked in a unified model. Suppose that given the new question, we retrieve a similar QA pair (q_1, a_1). We then use (q_1, a_1) as the query to retrieve and rank the reviews in the review collection R_t, and obtain the top ranked reviews $\{r_{1|q_1,a_1}, r_{2|q_1,a_1}, ..., r_{M|q_1,a_1}\}$ in $L_{r|q_1,a_1}$. The reformulation method is designed as follows:

$$r_{q_i,a_i} =< q_i, a_i, \lambda \cdot r_{1|q_i,a_i}, \lambda \cdot r_{2|q_i,a_i}, ... > \tag{9}$$

where $< a, \lambda \cdot b >$ stands for concatenating the bag-of-word model of the two text a and b, and λ is the weight of b for the concatenation. Then we use the trained response review ranking model described in Section 3.2 to rank all the snippets in the set:

$$T = \{r_{q_1,a_1}, r_{q_2,a_2}, ..., r_{q_N,a_N}, \{r\}\} \tag{10}$$

where r_{q_i,a_i} is the pseudo review based on the reformulated QA pairs.

4 Experiments

4.1 Data Collections

We conduct our experiments using two datasets from a real-world E-commerce site, namely *Amazon.com*[1]. The first dataset is from the product category "Cell Phones & Accessories" and the second dataset is from the category "Clothing, Shoes & Jewelry". These two datasets were originally collected by McAuley et al. (McAuley and Yang, 2016; Wan and McAuley, 2016) containing a large number of products, question-answer pairs [2], and reviews [3]. For each product category, we randomly select around 50 questions to form the testing set, and conduct annotations so that these questions can be used for evaluation purpose. The remaining questions are retained as the training set. Regarding the annotation of the testing questions, consider each testing question, we collect the existing QA collection of the corresponding product. Then we examine each QA pair and annotate whether it is semantically similar to a given testing question. Similarly we collect the existing review collection of the corresponding product. We examine each review and annotate whether it can be used as a response for a given testing question. The detailed statistics of the datasets is depicted in Table 1.

Table 1: Statistics of the Datasets

Category	Cell Phones & Accessories	Clothing, Shoes & Jewelry
# of product	10320	2871
# of question	60791	17233
# of review	3447249	5748920
Ave. length of question	12.4	15.2
Ave. length of review sentence	13.9	14.1
# of question for testing	53	55
# of question w/ response reviews	53	55
# of question w/ similar QA Pairs	12	20

[1] https://www.amazon.com

[2] http://jmcauley.ucsd.edu/data/amazon/qa/

[3] http://jmcauley.ucsd.edu/data/amazon/

4.2 Experiment Settings

We conduct sentence segmentation for the reviews. Terms in text such as review sentences, questions, and answer are lemmatized. Stopwords and punctuations are removed. The parameter value of M in Section 3.2 is set to 50. In Jelinek-Mercer smoothing method for LM, the interpolation parameter λ is set to 0.8. For both LDA and RBM in Section 3.2.2, the dimension of hidden layer is set to 10. For pre-trained word embedding, 300-dimension GloVe embeddings are utilized. In the autoencoder for word embedding, the dimension of hidden layer is set to 10.

We implement several baseline and state-of-the-art models to compare with our proposed framework.

- **Language Model (LM)**(Jin et al., 2002). We regard the given new question as the query, and calculate the query likelihood from each review and each question-answer pair via a language model. The list of output snippets are ranked by the question likelihood.

- **Positional Language Model (PLM)** (Lv and Zhai, 2009). Similar to LM, PLM models each document and provides query likelihood considering positional information. The list of output snippets are ranked by the positional question likelihood.

- **QA-LTR** (Surdeanu et al., 2008). QA-LTR is a supervised learning-to-rank model for answer selection, making use of a large volume labeled question-answer pairs. The original QA-LTR model is designed for online QA collections. Hence some features are not suitable. We implement QA-LTR with features replaced by our designed features described in Section 3.2.2. We expand the existing collection of question-answer pairs by including some relevant reviews to form the positive instance training set. Specifically, given a question q in the training set with its answer set A_q, we add similar reviews as candidate answers into A_q. For each $a \in A_q$, we retrieve k review sentences that have the largest similarity to a, denoted as $\{r_1^a, r_2^a, ..., r_k^a\}$. These retrieved reviews are added into the answer set A_q of the question q. We set k to 2 which yields the best performance.

- **QA-LSTM** (Tan et al., 2016). QA-LSTM is one of the state-of-the-art representation learning models for question answering matching. Equipped with bidirectional LSTM and maximum pooling, QA-LSTM has been applied to answer selection. In our experiment, we utilize this model to obtain semantic representation of questions and reviews, and to estimate the matching score given a question q_0 and a review r_0. Similar as in the training of QA-LSTM, the positive instances also include reviews via expanding the existing answer set for each question.

- **Ours.** It denotes our proposed framework.

The evaluation metrics are Normalized Discounted Cumulative Gain (NDCG), Mean Average Precision (MAP), and Mean Reciprocal Rank (MRR). We conduct the t-test for the results to compare the performance improvement over QA-LSTM and QA-LTR, with the significance level being 0.05.

4.3 Product Question Response Performance

Table 2: Response Performance. †, ‡ indicate that Ours is statistical significant at the significance level of 0.05 over QA-LSTM and QA-LTR respectively.

	Cell Phones & Accessories			Clothing, Shoes & Jewelry		
Model	NDCG	MAP	MRR	NDCG	MAP	MRR
LM	0.298	0.137	0.247	0.305	0.146	0.261
PLM	0.307	0.148	0.260	0.321	0.178	0.294
QA-LTR	0.369	0.177	0.403	0.392	0.192	0.320
QA-LSTM	0.397	0.184	0.390	0.407	0.211	0.367
Ours	$0.493^{\dagger\ddagger}$	$0.236^{\dagger\ddagger}$	$0.532^{\dagger\ddagger}$	$0.424^{\ddagger}$	$0.247^{\dagger\ddagger}$	$0.492^{\dagger\ddagger}$

The result is shown in Table 2 for the two product categories. We can observe that our proposed model "Ours" outperforms all the other models. It is statistical significant that our proposed framework

is better than QA-LSTM and QA-LTR. The results indicate that traditional information retrieval method cannot effectively handle the product question response. The reason is that our target is to find the review sentences or existing question-answer pairs that can answer the given question, instead of just retrieving relevant text for the question. The semantic relationship between questions and reviews cannot be modeled via information retrieval models. QA models, namely, QA-LTR and QA-LSTM are not effective as compared to our proposed framework despite the fact that QA models have been previously utilized for review ranking. It reveals that existing QA models have limitation when handling the heterogeneous nature of the review collections and the QA collections. In contrast, the distant supervision paradigm in our framework is more suitable. One main advantage of our proposed paradigm is that it can exploit both information sources in a coordinated manner.

Table 3: Response Performance with different amount of training data.

	Cell Phones & Accessories			Clothing, Shoes & Jewelry		
Model	NDCG	MAP	MRR	NDCG	MAP	MRR
Ours w/ all training data	0.493	0.236	0.532	0.424	0.247	0.492
Ours w/ 1/2 training data	0.483	0.219	0.521	0.421	0.230	0.479
Ours w/ 1/3 training data	0.470	0.201	0.492	0.401	0.215	0.440
Ours w/ 1/4 training data	0.403	0.189	0.453	0.365	0.157	0.398

In addition, we evaluate the performance of our framework under different amount of training data, and the result is presented in Table 3. "Ours w/ 1/n training data" stands for our proposed framework trained with only 1/n of all the training data. We can observe that even when we use a partial amount of the training data, the performance of the response generation does not drop significantly. It demonstrates that our proposed framework can effectively make use of the available training data including QA pairs and QA-based review ranking.

4.4 Case Study

We present a sample case study to gain more insights. Table 4 contains a response snippet list produced from our framework given a new question. The product is a cable and the new question is about how to use the cable for photo transfer to a Mac. The snippets in the response are compposed of review sentences and an existing QA pair. Specifically, the first ranked snippet is a question-answer pair talking about transferring photos from a mobile phone to a Mac. The following snippets come from reviews and they mention that the cable is not for data transferring. Thus, the user can receive informative response extracted from both the review collection and the QA collection associated with the corresponding product.

Table 4: A case study.

Newly posed question
Does anyone know if this cord will work to transfer photos by USB to a Mac? or is it just PC? Thanks!
Response Snippet List
Q: What do I need to get to transfer photos from phone to a Mac if this cable won't work? A: Not quite sure to be honest but maybe if the phone supports like sending emails then you can send your photos through email and download the photos onto your Mac from your email if the cable doesn't work but it should.
It would be nice if it could be used to transfer pictures, but if it can they sure do not make it easy.
Read all descriptions carefully because some of the cables are USB charging cables and do not transfer data!
This is not a data cable.

5 Conclusions

We propose a framework for instant product question response considering two common information sources, namely, existing QA collections and review collections. Our framework exploits both information sources in a coordinated manner. The distant supervision paradigm is adopted for handling a major subtask of question-based response review ranking. Extensive experiments demonstrate the effectiveness and stability of the proposed framework.

References

Zheqian Chen, Chi Zhang, Zhou Zhao, Chengwei Yao, and Deng Cai. 2018. Question retrieval for community-based question answering via heterogeneous social influential network. *Neurocomputing*, 285:117–124.

Lei Cui, Shaohan Huang, Furu Wei, Chuanqi Tan, Chaoqun Duan, and Ming Zhou. 2017. Superagent: A customer service chatbot for e-commerce websites. *Proceedings of ACL 2017, System Demonstrations*, pages 97–102.

Rong Jin, Alex G Hauptmann, and ChengXiang Zhai. 2002. Language model for information retrieval. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 42–48. ACM.

Yuanhua Lv and ChengXiang Zhai. 2009. Positional language models for information retrieval. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 299–306. ACM.

Julian McAuley and Alex Yang. 2016. Addressing complex and subjective product-related queries with customer reviews. In *Proceedings of the 25th International Conference on World Wide Web*, pages 625–635. International World Wide Web Conferences Steering Committee.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Yikang Shen, Wenge Rong, Nan Jiang, Baolin Peng, Jie Tang, and Zhang Xiong. 2017. Word embedding based correlation model for question/answer matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3511–3517.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2008. Learning to rank answers on large online qa collections. *Proceedings of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 719–727.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 464–473.

Mengting Wan and Julian McAuley. 2016. Modeling ambiguity, subjectivity, and diverging viewpoints in opinion question answering systems. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 489–498. IEEE.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. In *Proceedings of The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1442–1451.

Linlin Wang, Kang Liu, Zhu Cao, Jun Zhao, and Gerard de Melo. 2015. Sentiment-aspect extraction based on restricted boltzmann machines. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 616–625.

Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270.

Liu Yang, Qingyao Ai, Jiafeng Guo, and W Bruce Croft. 2016. anmm: Ranking short answer texts with attention-based neural matching model. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 287–296. ACM.

Jianxing Yu, Zheng-Jun Zha, and Tat-Seng Chua. 2012. Answering opinion questions on products by exploiting hierarchical organization of consumer reviews. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 391–401. Association for Computational Linguistics.

Wayne Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. 2010. Jointly modeling aspects and opinions with a maxent-lda hybrid. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 56–65. Association for Computational Linguistics.

Guangyou Zhou, Li Cai, Jun Zhao, and Kang Liu. 2011. Phrase-based translation model for question retrieval in community question answer archives. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 653–662. Association for Computational Linguistics.

Guangyou Zhou, Tingting He, Jun Zhao, and Po Hu. 2015. Learning continuous word embedding with metadata for question retrieval in community question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 250–259.

Aff2Vec: Affect–Enriched Distributional Word Representations

Sopan Khosla, Niyati Chhaya, and Kushal Chawla
Big Data Experience Lab
Adobe Research
Bangalore, India
{skhosla,nchhaya,kchawla}@adobe.com

Abstract

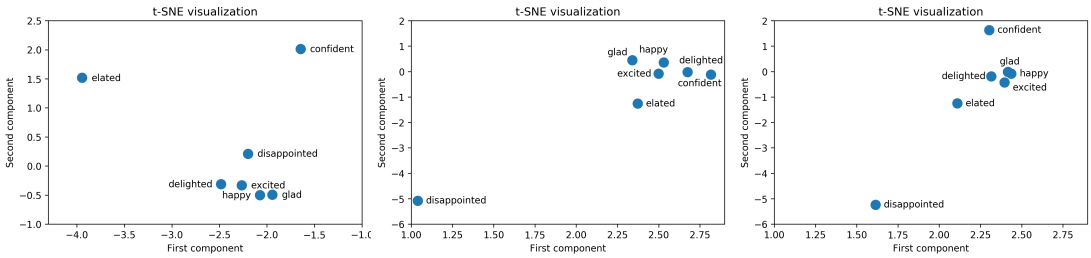
Human communication includes information, opinions and reactions. Reactions are often captured by the affective-messages in written as well as verbal communications. While there has been work in affect modeling and to some extent affective content generation, the area of affective word distributions is not well studied. Synsets and lexica capture semantic relationships across words. These models, however, lack in encoding affective or emotional word interpretations. Our proposed model, Aff2Vec, provides a method for enriched word embeddings that are representative of affective interpretations of words. Aff2Vec outperforms the state-of-the-art in intrinsic word-similarity tasks. Further, the use of Aff2Vec representations outperforms baseline embeddings in downstream natural language understanding tasks including sentiment analysis, personality detection, and frustration prediction.

1 Introduction

Affect refers to the experience of a feeling or an emotion (Scherer et al., 2010; Picard, 1997). This definition includes emotions, sentiments, personality, and moods. The importance of affect analysis in human communication and interactions has been discussed by Picard (1997). Historically, affective computing has focused on studying human communication and reactions through multi-modal data gathered via various sensors. The study of human affect from text and other published content is an important topic in language understanding. Word correlation with social and psychological processes is discussed by Pennebaker (2011). Preotiuc-Pietro et al. (2017) studied personality and psycho-demographic preferences through Facebook and Twitter content. Sentiment analysis in Twitter, with a detailed discussion on human affect (Rosenthal et al., 2017) and affect analysis in poetry (Kao and Jurafsky, 2012) have also been explored. Human communication not only contains semantic and syntactic information but also reflects the psychological and emotional states. Examples include the use of opinion and emotion words (Ghosh et al., 2017). The analysis of affect in interpersonal communication such as emails, chats, and longer articles is necessary for various applications including the study of consumer behavior and psychology, understanding audiences and opinions in computational social science, and more recently for dialogue systems and conversational agents. This is an open research space today.

Traditional natural language understanding systems rely on statistical language modeling and semantic word distributions such as WORDNET (Miller, 1995) to understand relationships across different words. There has been a resurgence of research efforts towards creating word distributions that capture multi-dimensional word semantics (Mikolov et al., 2013a; Pennington et al., 2014). Sedoc et al. (2017b) introduce the notion of affect features in word distributions but their approach is limited to creating enriched representations and no comments on the utility of the new word distribution is presented. Beyond word-semantics, deep learning research in natural language understanding is focused towards sentence representations using encoder-decoder models (Ahn et al., 2016), integration of symbolic knowledge to language models (Vinyals et al., 2015), and some recent works in augmenting neural language modeling with affective information to emotive text generation (Ghosh et al., 2017). These works, however, do

Proceedings of the 27th International Conference on Computational Linguistics, pages 2204–2218
Santa Fe, New Mexico, USA, August 20-26, 2018.



(a) GloVe (b) GloVe⊕Affect (c) GloVe + Counterfitting ⊕ Affect

Figure 1: t-SNE for significant affect words: The graphs show the distribution of sample words from Sedoc et al (2017b). The variance in the visualization illustrates the perturbation introduced by distributional schemes discussed in this paper. Vanilla GloVe embeddings show 'disappointed' near 'delighted', while these are separated in the ⊕Affect representations.

not introduce distributional affective word representations that not only reflect affective content but are also superior for related downstream natural language tasks such as sentiment analysis and personality detection.

We introduce Aff2Vec, affect-enriched word distributions trained on lexical resources coupled with semantic word distributions. Aff2Vec captures opinions and affect information in the representation using post-processing approaches. Figure 1 illustrates how Aff2Vec captures affective relationships using a t-SNE visualization of the word space. Even though Aff2Vec is trained on the Valence-Arousal-Dominance dimensions, our approach is generalizable to any other affect spaces. Our experiments show that Aff2Vec outperforms vanilla embedding spaces on both intrinsic word-similarity tasks as well as extrinsic natural language applications. The main contributions of this paper include:

- **Aff2Vec:** Affect-enriched word representations using post-processing techniques. We show that Aff2Vec outperforms the state-of-the-art in both intrinsic word similarity metrics as well as downstream natural language tasks including Sentiment analysis, Personality detection, and Frustration detection in interpersonal communication.

- **ENRON-FFP Dataset:** We introduce the ENRON-FFP Email dataset with Frustration, Formality, and Politeness tags gathered using a crowd-sourced human perception study.

The remainder of the paper is organized as follows. The prior art for enriched word distributions is discussed in Section 2. Aff2Vec is introduced in Section 3. We present a crowd-sourcing study for the ENRON-FFP Dataset in Section 4 and Section 5 discusses the experimental setup. Section 6 presents the evaluation of Aff2Vec for various intrinsic and extrinsic tasks. A discussion on the distributional word representations is presented in Section 7 before the conclusion in Section 8.

2 Related Work

The use of lexical semantic information (lexical resources) to improve distributional representations is recent. Methods like (Yu and Dredze, 2014; Xu et al., 2014; Bian et al., 2014; Kiela et al., 2015) achieve improved representations by using word similarity and relational knowledge to modify the prior or add a regularization term. We call such methods 'pre-training methods', as they alter the training process for word representations. Such methods require a change in the loss function while training the embeddings, hence are computationally expensive.

The other set of word distribution enhancements are done post-training. These methods aim to include external information using normalizations and modifications to the vanilla word distributions. Methods such as Retrofitting (Faruqui et al., 2015), which try to drag similar words closer together (where notion of similarity is taken from word relation knowledge found in semantic lexica (e.g. WordNet)) fall in this category. Counterfitting (Mrkšic et al., 2016) on the other hand, initiates from SimLex-999 tuned

embeddings, injects antonym and synonym constraints to improve word representations. This paper introduces post-training techniques on vanilla, retrofitted and counterfitted embeddings to include affective information in the distributions. Our work falls in the post-training category, hence no direct comparison with the pre-trained approaches is presented in this paper.

Recent work has explored approaches to adapt general-purpose lexica for specific contexts and affects. Studies have recognized the limited applicability of general purpose lexica such as ANEW (Bradley and Lang, 1999) to identify affect in verbs and adverbs, as they focus heavily on adjectives. Recognizing that general-purpose lexica often detect sentiment which is incongruous with context, Ribeiro et al. (2016) proposed a sentiment-damping method which utilizes the average sentiment strength over a document to damp any abnormality in the derived sentiment strength. Similarly, Blitzer et al. (2007) argued that words like 'predictable' induced a negative connotation to book reviews, while 'must-read' implied a highly positive sentiment. This paper doesn't focus on building yet another affect lexicon but studies the consequences of including affect information in distributional word representations that aim at defining relational relationships across all words in large contexts and vocabularies.

Automatic expansion of affect rating has been approached with the intuition that words closer in the distributional space would have similar ratings (Recchia and Louwerse, 2015; Palogiannidi et al., 2015; Vankrunkelsven et al., 2015; Köper and Im Walde, 2016). Recent work by Sedoc et al. (2017b) uses Signed Spectral Clustering to differentiate between words which are contextually similar but display opposite affect. Wang et al. (2016) use a graph-based method inspired by label propagation. While our approach follows the nature of the task defined in Sedoc et al. (2017b), we propose a generalized method to enrich content with affective information. Instead of only focusing on distinguishing the polarities, our method incorporates both semantic and affect information. Hence, creating embeddings that can also be used for semantic similarity tasks. Note that Sedoc et al. do not include any semantic information in their modeling.

3 Aff2Vec: Affect–enriched Word Distributions

Aff2Vec aims at incorporating affective information in word representations. We leverage the Warriner's lexicon (Warriner et al., 2013) in the Valence-Arousal-Dominance space for this work. The proposed work is generalizable to other affect spaces (Refer Appendix A for experiments with different dimensions.). This section presents two approaches for affect–enrichment of word distributions.

Warriner's lexicon: We use the Warriner's lexicon (Warriner et al., 2013) in this work. This is a affect lexicon with $13,915$ English words. It contains real-valued scores for valence, arousal, and dominance (VAD) on a scale of $1-9$ each. 1, 5, and 9 correspond to the low, moderate (i.e. neutral), and high values for each dimension respectively. For out-of-dictionary words, such as stop words or proper nouns, we assume a neutral affect vector $\vec{a} = [5, 5, 5]$.

3.1 Affect-APPEND ($\oplus$ Affect)

Consider word embeddings W, the aim is to introduce affective information to this space using the affect embedding space, A. The word vectors W, each with dimension D, are concatenated with affect vectors A with dimension F, thus resulting in a $D + F$ dimensional enriched representation. The process for this concatenation is described here:

1. Normalize word vector W and affect vector A using their L2-Norms (Equation 1). This reduces the individual vectors to unit-length.

$$x_i = \frac{x_i}{\sqrt{\sum_{k=1}^{D} x_{ik}^2}} \ \forall x_i \in W, \quad a_i = \frac{a_i}{\sqrt{\sum_{k=1}^{F} a_{ik}^2}} \ \forall a_i \in A \tag{1}$$

2. Concatenate the regularized word vectors x_i with regularized affect vectors a_i.

$$WA(w) = W(w) \oplus A(w) \tag{2}$$

3. Standardize (variance 1 , mean 0) the $D + F$ dimensional embeddings to achieve standard normal distribution.

$$y_i = \frac{y_i - \mu}{\sigma} \quad \forall y_i \in WA \tag{3}$$

where, μ and σ represent the mean and standard deviation respectively.

4. The enriched space WA is then reduced to original D dimensional vector. We use Principal Component Analysis for the dimensionality reduction.

3.2 Affect-STRENGTH

In this approach, the strength in the antonym-synonym relationships of the words is incorporated to the word distribution space. Hence, we leverage the retrofitting algorithm (Faruqui et al., 2015) as shown below. [1]

Retrofitting: Let $V = \{w_1, w_2, w_3, ..., w_n\}$ be a vocabulary and Ω be an ontology which encodes semantic relations between words present in V (e.g. WORDNET). This ontology Ω is represented as an undirected graph (V, E) with words as vertices and (w_i, w_j) as edges indicating the semantic relationship of interest. Each word $w_i \in V$ is represented as a vector representation $\hat{q}_i \in R^d$ learnt using a data-driven approach (e.g. Word2Vec or GloVe) where d is the length of the word vectors.

Let $\hat{Q}$ be the matrix collection of these vector representations. The objective is to learn the matrix $Q = (q_1, ..., q_n)$ such that the word vectors (q_i) are both close to their counterparts in $\hat{Q}$ and to adjacent vertices in Ω. The distance between a pair of vectors is defined to be Euclidean, hence the objective function for minimization is

$$\Psi(Q) = \sum_{i=1}^{n} \left[\alpha_i \|q_i - \hat{q}_i\|^2 + \sum_{(i,j)\in E} \beta_{ij} \|q_i - q_j\|^2 \right] \tag{4}$$

where, α and β are hyper parameters and control the relative strengths of the two associations. Ψ is a convex function in Q and its global optimal solution can be found by using an iterative update method. By setting $\frac{\partial \Psi(Q)}{\partial q_i} = 0$, the online updates are as follows:

$$q_i = \frac{\sum_{j:(i,j)\in E} \beta_{ij} q_j + \alpha_i \hat{q}_i}{\sum_{j:(i,j)\in E} \beta_{ij} + \alpha_i} \tag{5}$$

We propose two ways to modify β_{ij} in equation 4 in order to incorporate affective strength in the edge weights connecting two retrofitted vectors to each other.

Affect-cStrength ($*$ cStrength): In this approach, the affective strength is considered as a function of all F affect dimensions.

$$S(w_i, w_j) = 1 - \frac{\|a_i - a_j\|}{\sqrt{\sum_{f=1}^{F} max\_dist_f^2}} \tag{6}$$

where, a_i and a_j are F dimensional vectors in A and max_dist_f is defined as the maximum possible distance between two vectors in f^{th} dimension ($= 9.0 - 1.0 = 8.0$ for VAD dimensions).

Affect-iStrength ($*$ iStrength): Here, each dimension is treated individually. For every dimension f in A, we add an edge between neighbors in the Ontology Ω where the strength of that edge is given by $S_f(w_i, w_j)$:

$$S_f(w_i, w_j) = 1 - \frac{|a_{if} - a_{jf}|}{max\_dist_f}, \quad S(w_i, w_j) = \sum_{f=1}^{F} S_f(w_i, w_j) \tag{7}$$

β_{ij} from equation 5 is normalized with this strength function as $\beta_{ij} = \beta_{ij} * S(w_i, w_j)$, where $S(w_i, w_j)$ is defined by either Affect-cStrength or Affect-iStrength.

[1] https://github.com/mfaruqui/retrofitting

4 Dataset: ENRON-FFP

Table 1: Enron-FFP Dataset Description

Property	Value
Total number of emails (Main Experiment)	960
Total number of emails (Pilot Experiment)	90
Min. sentences per email	1
Max. sentences per email	17
Average email size (no. of sentences)	4.22
Average number of words per email	77.5

Table 2: Datasets for Intrinsic Evaluation

Dataset	# Word-Pairs
Word Similarity (**WS**) (Finkelstein et al., 2001)	353
RG-65 (Rubenstein and Goodenough, 1965)	65
MEN (Bruni et al., 2012)	3000
Miller-Charles (**MC**) (Miller and Charles, 1991)	30
RW (Luong et al., 2013)	2034
SCWS (Huang et al., 2012)	2023
SimLex-999 (**SL**) (Hill et al., 2016)	999
SimVerb-3500 (**SV**) (Gerz et al., 2016)	3500

Table 3: Example emails with varying inter-annotator agreements.

Affect Dimension	Example	Annotations
Frustration: Low Agreement	See highlighted portion. We should throw this back at Davis next time he points the finger.	(-1, -1, 0, 0, -2, -2, 0, 0, -2, 0)
Frustration: High Agreement	Please see announcement below. Pilar, Linda, India and Deb, please forward to all of your people. Thanks in advance, adr	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
Formality: Low Agreement	I talked with the same reporters yesterday (with Palmer and Shapro). Any other information that you can supply Gary would be appreciated. Steve, did Gary A. get your original as the CAISO turns email? GAC	(0, 0, -1, 1, 1, 1, 0, -1, -2, -1)
Politeness: High Agreement	John, This looks fine from a legal perspective. Everything in it is either already in the public domain or otherwise non-proprietary. Kind regards, Dan	(1, 1, 1, 1, 1, 1, 1, 1, 2, 1)

We introduce an email dataset, a subset of the ENRON data (Cohen, 2009), with tags about interpersonal communication traits: Formality, Politeness, and Frustration. Along with the content of the emails, the dataset also provides user and network level information for email exchanges between Enron employees.

Human Perceptions and Definitions: *Tone* or affects such as frustration and politeness are highly subjective. In this work, we do not attempt to introduce or standardize an accurate definition for frustration (or formality and politeness). Instead, we assume that these are defined by human perception and each individual may differ in their understanding of these metrics. This approach of using untrained human judgments has been used in prior studies of pragmatics in text data (Pavlick and Tetreault, 2016; Danescu-Niculescu-Mizil et al., 2013) and is a recommended way of gathering gold-standard annotations (Sigley, 1997). The tagged data is then used to predict the formality, frustration, and politeness tags using Aff2Vec embeddings.

Dataset Annotation: We conducted a crowd-sourced experiment using Amazon's Mechanical Turk[2]. The analysis presented in this section is based on $1,050$ emails that were tagged across multiple experiments[3]. Table 1 provides the statistics of the annotated data. We follow the annotation protocol of the Likert Scale (Allen and Seaman, 2007) for all three dimensions. Each email is considered as a single data point and only the text in the email body is provided for tagging. Frustration is tagged on a 3 point scale with neutral being equated to 'not frustrated'; 'frustrated' and 'very frustrated' are marked with -1 and -2 respectively. Formality and politeness follow a 5 point scale from -2 to $+2$, where both extremes mark the higher degree of presence and absence of the respective dimension. Table 3 shows example emails from the dataset.

Inter-annotator Agreement: To measure whether an individual's intuition of the affect dimensions is consistent with other annotators' judgment, we use inter-class correlation[4] to quantify the

[2]https://www.mturk.com/mturk/welcome

[3]Link to the annotated ENRON-FFP dataset: https://bit.ly/2IAxPab

[4]We report the average raters absolute agreement (ICC1k) using the psych package in R.

Table 4: Intrinsic Evaluation: Word Similarity–We report the Spearman's correlation coefficient (ρ). The results show that Aff2Vec variants improve performance consistently.

Model	**Word Similarity**							
	SL	**SV**	**WS**	**RG**	**RW**	**SCWS**	**MC**	**MEN**
GloVe	0.41	0.28	0.74	0.77	0.54	0.64	0.80	0.80
$\oplus$ Affect	0.49	0.39	**0.77**	0.79	0.59	0.67	0.80	0.84
+ Retrofitting	0.53	0.37	0.73	0.81	0.52	0.66	0.82	0.82
+ Retrofitting $*$ cStrength	0.53	0.36	0.74	0.81	0.52	0.66	0.82	0.82
+ Retrofitting $*$ iStrength	0.56	0.38	0.64	0.80	0.44	0.62	0.80	0.78
+ Retrofitting $\oplus$ Affect	0.60	0.46	0.76	0.81	**0.61**	**0.69**	0.81	**0.85**
+ Counterfitting	0.58	0.47	0.65	0.80	0.56	0.61	0.78	0.77
+ Counterfitting $\oplus$ Affect	**0.62**	**0.53**	0.70	**0.84**	**0.61**	0.64	**0.84**	0.80
Word2Vec	0.45	0.36	0.70	0.76	0.59	0.67	0.80	0.78
$\oplus$ Affect	0.49	0.42	0.67	0.81	0.59	0.66	0.85	0.79
+ Retrofitting	0.55	0.45	**0.74**	0.82	**0.62**	**0.70**	0.83	0.80
+ Retrofitting $*$ cStrength	0.55	0.44	0.73	0.82	0.62	**0.70**	0.83	0.80
+ Retrofitting $*$ iStrength	0.58	0.47	0.71	0.83	0.57	0.69	0.85	0.80
+ Retrofitting $\oplus$ Affect	0.59	0.49	0.71	**0.84**	**0.62**	**0.70**	**0.86**	**0.82**
+ Counterfitting	0.56	0.51	0.66	0.75	0.61	0.64	0.75	0.73
+ Counterfitting $\oplus$ Affect	**0.60**	**0.54**	0.64	0.82	0.60	0.64	0.82	0.76
Paragram	0.69	0.54	**0.73**	0.78	0.59	0.68	0.80	0.78
$\oplus$ Affect	0.71	0.59	0.70	0.77	**0.60**	0.67	0.76	**0.79**
+ Retrofitting	0.68	0.55	**0.73**	0.79	0.59	0.68	0.81	0.78
+ Retrofitting $*$ cStrength	0.69	0.55	**0.73**	0.79	0.59	**0.69**	0.81	0.78
+ Retrofitting $*$ iStrength	0.68	0.56	0.71	0.80	0.58	0.68	**0.84**	0.77
+ Retrofitting $\oplus$ Affect	0.71	0.58	0.70	0.80	0.59	0.67	0.78	**0.79**
+ Counterfitting	0.74	0.63	0.69	**0.81**	**0.60**	0.66	0.82	0.74
+ Counterfitting $\oplus$ Affect	**0.75**	**0.66**	0.68	**0.81**	**0.60**	0.65	0.82	0.76

ordinal ratings. This measure accounts for the fact that we may have different groups of annotators for each data point. Each data point has 10 distinct annotations. Agreements reported are 0.506 ± 0.05(for 3 class), 0.73 ± 0.02(for 5 class), and 0.64 ± 0.03(for 5 class) for frustration, formality, and politeness respectively. The agreement measures are similar to those reported for other such psycholinguistic tagging tasks.

5 Experiments

Two sets of experiments are presented to evaluate Aff2Vec embeddings[5] - Intrinsic evaluation using word similarity tasks and extrinsic evaluation using multiple NLP applications. We focus on 3 vanilla word embeddings: GloVe (Pennington et al., 2014), Word2Vec-SkipGram[6] (Mikolov et al., 2013b) and Paragram-SL999 (Wieting et al., 2015); and their retrofitted (Faruqui et al., 2015) and counterfitted (Mrkšic et al., 2016) versions. The vocabulary and embeddings used in our experiments resonate with the experimental setup by Mrkšić et al.(2016) ($76,427$ words).

5.1 Intrinsic Evaluation

Word similarity is a standard task used to evaluate embeddings (Mrkšic et al., 2016; Faruqui et al., 2015; Bollegala et al., 2016). In this paper, we evaluate the embeddings on benchmark datasets given in Table 2. We report the Spearman's rank correlation coefficient between rankings produced by our model (based on cosine similarity of the pair of words) against the benchmark human rankings for each dataset.

5.2 Extrinsic Evaluation

Although intrinsic tasks are popular, performance of word embeddings on these benchmarks does not reflect directly into the downstream nlp tasks (Chiu et al., 2016). Gladkova and Drozd (2016) and

[5]Link to the Aff2Vec word embeddings: https://bit.ly/2HGohsO

[6]https://code.google.com/archive/p/word2vec/

Table 5: Extrinsic Evaluation: Results for FFP-Prediction, Personality Detection, Sentiment Analysis, and WASSA Emotional Intensity task for Aff2Vec variants for GloVe and Word2Vec embeddings. We report the Mean Squared Error (MSE) for FFP-Prediction, Accuracy (% ACC) for Personality Detection, and Sentiment Analysis (SA) and Person's ρ for the WASSA Emo-Int Task (EMO-INT)

Model	**FFP-Prediction**			**Personality Detection**					**SA**	**EMO-INT**			
	MSE ($X10^{-3}$)			**Acc.** (%)					**Acc.** (%)	**Pearson's** ρ ($X10^{-2}$)			
	FOR	**FRU**	**POL**	**EXT**	**NEU**	**AGR**	**CON**	**OPEN**	**DAN**	**ANG**	**FEA**	**JOY**	**SAD**
GloVe	27.59	32.40	21.89	**56.08**	55.25	56.06	**57.32**	59.14	83.1	70.98	71.19	65.85	73.30
$\oplus$ Affect	27.72	28.76	22.02	51.47	57.41	56.09	55.06	**62.08**	84.3	70.91	71.72	66.26	**73.58**
+ Retrofitting	27.44	29.35	21.75	55.79	59.67	55.59	56.89	59.67	82.7	72.10	71.86	**67.11**	73.14
+ Retrofitting $\oplus$ Affect	28.33	**27.91**	22.24	55.01	56.43	**57.48**	53.04	61.12	83.7	**72.38**	**72.53**	66.29	72.76
+ Counterfitting	**25.66**	29.20	22.90	55.11	58.32	55.41	53.89	60.36	84.2	70.45	68.95	65.27	72.63
+ Counterfitting $\oplus$ Affect	28.89	32.46	**21.64**	52.12	**60.03**	56.53	54.93	59.51	**84.4**	70.20	70.43	65.81	72.37
Word2Vec	25.86	27.88	21.56	**56.08**	58.19	56.59	55.18	61.41	83.3	68.86	71.24	65.23	72.60
$\oplus$ Affect	25.39	28.16	22.99	53.54	57.97	55.17	54.12	59.31	83.4	69.29	**71.92**	64.49	**72.63**
+ Retrofitting	27.81	29.05	21.85	54.33	56.65	**57.39**	54.65	60.03	82.5	70.12	71.42	**67.96**	72.02
+ Retrofitting $\oplus$ Affect	**25.08**	**27.08**	21.64	53.74	**59.61**	56.34	**56.93**	59.7	83.3	**70.65**	71.90	66.36	72.20
+ Counterfitting	28.28	27.12	22.95	54.55	57.61	57.09	54.1	58.5	83.3	68.64	70.13	63.36	70.67
+ Counterfitting $\oplus$ Affect	27.73	29.67	**21.52**	51.28	58.86	56.66	53.22	**61.62**	**83.5**	69.38	70.31	64.94	71.37
Baselines													
(Majumder et al., 2017)	–	–	–	**58.09**	59.38	56.71	57.30	**62.68**	–	–	–	–	–
ENRON Trainable	31.61	43.90	26.27	–	–	–	–	–	–	–	–	–	–
Re(Glove)(Yu et al., 2017)	–	–	–	–	–	–	–	–	82.2	–	–	–	–
Re(w2v)(Yu et al., 2017)	–	–	–	–	–	–	–	–	82.4	–	–	–	–

Batchkarov et al. (2016) suggest that intrinsic tasks should not be considered as gold standards but as a tool to improve the model. Therefore, we test the utility of Aff2Vec on 4 distinct natural language understanding tasks:

Affect Prediction (FFP-Prediction): The experiment is to predict the formality, politeness, and frustration in email. We introduce the ENRON-FFP dataset for this task in section 4. A basic CNN model is used for the prediction (Refer to Appendix B.4 for hyper-parameters and model details). The purpose of this experiment is to evaluate the quality of the embeddings and not necessarily the model architecture. The CNN is hence not optimized for this task. Embeddings trained on the ENRON dataset (ENRON-Trainable) are used as a baseline.

Personality Detection: This task is to predict human personality from text. The big five personality dimensions (Digman, 1990) are used for this experiment. The 5 personality dimensions include Extroversion (EXT), Neuroticism (NEU), Agreeableness (AGR), Conscientiousness (CON), and Openness (OPEN). Stream-of-consciousness essay dataset by Pennebaker et al. (1999) contains $2,468$ anonymous essays tagged with personality traits of the author. We use this dataset for the experiment. Majumder et al (2017) propose a CNN model for this prediction. We use their best results as baseline and report the performance of Aff2Vec on their default implementation[7].

Sentiment Analysis: The Stanford Sentiment Treebank (SST) (Socher et al., 2013) contains sentiment labels on sentences from movie reviews. This dataset in its binary form is split into $6,920$ training, 872 validation, and $1,821$ test set samples. We report the performance on a Deep Averaging Network (DAN)(Iyyer et al., 2015)[8] with default parameters on the SST dataset and compare against refined embeddings specifically created for sentiment analysis. Implementation by Yu et al (2017) is used for the refined embeddings[9].

Emotion Intensity Task (WASSA): WASSA shared task on emotion intensity (Mohammad and Bravo-Marquez, 2017) requires to determine the intensity of a particular emotion (anger, fear, joy, or

[7]https://github.com/SenticNet/personality-detection

[8]https://github.com/miyyer/dan

[9]Implementation provided by the authors is used for this experiment.

Table 6: Polarity-Noise@k (PN@10) and Granularity-Noise@k (GN@10) where $k = 10$ for GloVe and Word2Vec variants. Note that lower the number, better this qualitative metric.

Model	**PN@10** (%)			**GN@10** ($X10^{-2}$)		
	V	**A**	**D**	**V**	**A**	**D**
GloVe	23.21	22.15	27.07	83.91	79.19	74.19
⊕ Affect	**16.46**	19.65	**19.42**	**72.56**	**69.00**	64.02
+ Retrofitting	22.55	21.82	26.5	82.15	78.68	72.53
+ Retrofitting ∗ cStrength	22.07	21.63	26.14	80.85	78.12	71.86
+ Retrofitting ∗ iStrength	23.05	21.77	26.66	83.14	78.76	72.65
+ Retrofitting ⊕ Affect	19.68	**18.16**	22.88	73.45	71.56	66.55
+ Counterfitting	22.68	22.2	26.46	83.31	78.78	72.54
+ Counterfitting ⊕ Affect	16.75	19.99	19.99	73.89	69.55	**63.93**
Word2Vec	24.66	22.19	27.41	85.81	79.23	74.25
⊕ Affect	20.62	**17.83**	23.19	**74.78**	**71.64**	67.32
+ Retrofitting	23.75	22.25	26.94	84.65	79.36	73.00
+ Retrofitting ∗ cStrength	23.33	22.01	26.58	83.39	78.71	72.24
+ Retrofitting ∗ iStrength	23.90	22.30	27.13	85.34	79.46	73.12
+ Retrofitting ⊕ Affect	20.61	18.54	23.6	75.71	72.47	67.61
+ Counterfitting	23.47	22.48	26.72	84.62	79.14	72.29
+ Counterfitting ⊕ Affect	**20.34**	18.17	**23.01**	74.83	71.94	**66.62**
Paragram	25.16	22.55	28.05	88.34	80.73	75.49
⊕ Affect	20.81	21.29	23.45	81.83	75.27	69.79
+ Retrofitting	25.69	22.8	28.48	89.67	81.25	76.05
+ Retrofitting ∗ cStrength	25.46	22.64	28.22	89.06	80.95	75.58
+ Retrofitting ∗ iStrength	25.69	22.84	28.43	89.85	81.26	75.93
+ Retrofitting ⊕ Affect	23.38	**20.34**	25.99	83.17	76.51	71.83
+ Counterfitting	24.86	22.76	27.88	88.27	80.68	75.18
+ Counterfitting ⊕ Affect	**20.31**	21.50	**23.03**	**81.40**	**75.05**	**69.10**

sadness) in a tweet. This intensity score can be seen as an approximation of the emotion intensity of the author or as felt by the reader. We train a BiLSTM-CNN–based model for this regression task with embedding dimensions as features (Refer to Appendix B.3 for model details). Vanilla embeddings are used as a baseline for this experiment.

5.3 Qualitative Evaluation: Noise@k

Affect-enriched embeddings perform better as they move semantically similar but affectively dissimilar words away from each other in the vector space. We demonstrate this effect through two measures that capture noise in the neighborhood of a word.

Polarity-Noise@k (PN@k) (Yu et al., 2017) calculates the number of top k nearest neighbors of a word with opposite polarity for the affect dimension under consideration.
Granular-Noise@k (GN@k) captures the average difference between a word and its top k nearest neighbors for a particular affect dimension (f).

$$GN_i@k = \frac{\sum_{j \in kNN_i} |a_i f - a_j f|}{k} \tag{8}$$

where, a_i, a_j are F-dimensional vectors in A and kNN_i denotes the top k nearest neighbors of word i. This is done for each word in the affect lexicon.

6 Results

All experiments are compared against the vanilla word embeddings, embeddings with counterfitting, and embeddings with retrofitting.

Table A.1 summarizes the results of the **Intrinsic word–similarity tasks**. For the pre-trained word embeddings, Paragram-SL999 outperformed GloVe and Word2Vec on most metrics. Both retrofitting and counterfitting procedures show better or at par performance on all datasets except for WordSim-353. Addition of affect information to different versions of GloVe consistently improves performance

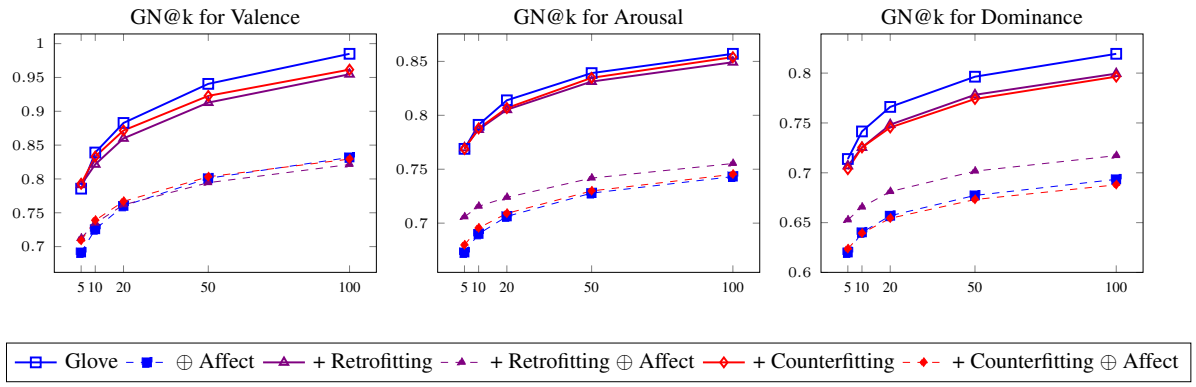


Figure 2: Variation of Granular Noise with different k values for GloVe and Affect-APPEND variants

whereas the only significant improvement for Paragram-SL999 variants is observed on the SimLex-999 and SimVerb-3500 datasets. To the best of our knowledge, $\rho = 0.74$ reported by Mrkšić et al. (2016) represents the current state-of-the-art for SimLex-999 and inclusion of affect information to these embeddings yields higher performance ($\rho = 0.75$). Similarly, for the SimVerb-3500 dataset, Paragram+Counterfitting⊕Affect embeddings beat the state-of-the-art scores[10]. Amongst Affect-APPEND and Affect-STRENGTH, Affect-APPEND out performs the rest in most cases for GloVe and Word2Vec. However, Affect-STRENGTH variations perform slightly better for the Paragram embeddings.

The results for the **Extrinsic tasks** are reported in Table 5. We report the performance for GloVe and Word2Vec with Affect-APPEND variants[11]. For FFP-Prediction, Affect-APPEND gives lowest Mean Squared Error for Frustration and Politeness. However, in the case of Formality, the counterfitting variant reports the lowest error. For the personality detection task, Affect-APPEND variants report best performance for NEU, AGR, and OPEN classes. For CON, Glove beats the best results in (Majumder et al., 2017). Evaluation against the Sentiment Analysis(SA) task shows that Affect-APPEND variants report highest accuracies. The final experiment reported here is the WASSA-EmoInt task. Affect-APPEND and retrofit variants out-perform the vanilla embeddings.

To summarize, the extrinsic evaluation supports the hypothesis that affect–enriched embeddings improve performance for all NLP tasks. Further, the word similarity metrics show that Aff2Vec is not specific to sentiment or affect-related tasks but is at par with accepted embedding quality metrics.

Qualitative Evaluation: Table 6 reports the average *Polarity-Noise@10* and *Granular-Noise@10* for GloVe, Word2Vec, and Paragram-SL999 variants. Note that lower the noise better the performance. Affect-APPEND reports the lowest noise for all cases. This shows that the introduction of affect dimensions in the word distributions intuitively captures psycholinguistic and in particular polarity properties in the vocabulary space. The rate of change of noise with varying k provides insights into (1) how similar are the embedding spaces and (2) how robust are the new representations to the noise i.e. how well is the affect captured in the new embeddings. Figure 2 shows the granular noise@k for valence, arousal, and sominance respectively. Noise@k for the Aff2Vec i.e. the Affect-APPEND variants, specifically, ⊕Affect and Couterfitting⊕Affect has lower noise even for a higher k. The growth rate for all variants is similar and reduces with an increase in the value of k. A similar behavior is observed for Polarity-Noise@k.

7 Discussion

Experiments give an empirical evaluation of the proposed embeddings, none of these provide an insight about the change in the distributional representations of the associated words. Semantic relationship

[10]Mentioned at `http://people.ds.cam.ac.uk/dsg40/simverb.html`

[11]Results for Paragram are reported in the supplement.

Table 7: Top-5 NN for 'Good' and 'Bad' for variants of GloVe, SentiWordNet and Aff2Vec

Model	Good	Bad
GloVe	[*great*, *nice*, *excellent*, *decent*, *bad*]	[*terrible*, *awful*, *horrible*, *wrong*, *thing*]
⊕ Affect	[*great*, *nice*, *excellent*, *decent*, *pretty*]	[*awful*, *terrible*, *horrible*, *wrong*, *crappy*]
+ Retrofitting	[*great*, *decent*, *nice*, *excellent*, *pretty*]	[*wrong*, *awful*, *terrible*, *horrible*, *nasty*]
+ Retrofitting ⊕ Affect	[*nice*, *great*, *decent*, *excellent*, *pretty*]	[*awful*, *wrong*, *nasty*, *terrible*, *horrible*]
+ Counterfitting	[*decent*, *nice*, *optimum*, *presentable*, *exemplary*]	[*rotten*, *shitty*, *horrid*, *naughty*, *lousy*]
+ Counterfitting ⊕ Affect	[*nice*, *decent*, *optimum*, *presentable*, *dignified*]	[*rotten*, *shitty*, *horrid*, *lousy*, *naughty*]
Senti-WordNet[12]	[*commodity*, *full*, *estimable*, *beneficial*, *adept*]	[*regretful*, *badly*]
Warriner's Lexicon	[*grandmother*, *healing*, *cheesecake*, *play*, *blissful*]	[*jittery*, *fuss*, *incessant*, *tramp*, *belligerent*]

capture the synonym like information. We study how the **neighborhood** of a certain word changes based on the different word distribution techniques used to create the corresponding representations. Table 7 shows the top five nearest neighbors based on the representations used. While SENTI-Wordnet represents synonyms more than affectively similar words, the affect–enriched embeddings provide a combination of both affective and semantic similarity. The variance in the ranking of words depicts how different schemes capture the intuition of word distributions. Such an analysis can be used to build automated natural language generation and text modification systems with varying objectives.

8 Conclusion

We present a novel, simple yet effective method to create affect–enriched word embeddings using affect and semantic lexica. The proposed embeddings outperform the state-of-the-art in benchmark intrinsic evaluations as well as extrinsic applications including sentiment analysis, personality detection, and affect prediction. We introduce a new human-annotated dataset with formality, politeness, and frustration tags on a subset of the publicly available ENRON email data. We are currently exploring the effect of dimension size on the performance of the enriched embeddings as well as the use of Aff2Vec for complex tasks such as text generation.

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *CoRR*, abs/1608.00318.

I Elaine Allen and Christopher A Seaman. 2007. Likert scales and data analyses. *Quality progress*, 40(7):64.

Miroslav Batchkarov, Thomas Kober, Jeremy Reffin, Julie Weeds, and David Weir. 2016. A critique of word similarity as a method for evaluating distributional semantic models.

Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Knowledge-powered deep learning for word embedding. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 132–148. Springer.

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

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*, 5(1):135–146.

Danushka Bollegala, Mohammed Alsuhaibani, Takanori Maehara, and Ken-ichi Kawarabayashi. 2016. Joint word representation learning using a corpus and a semantic lexicon. In *AAAI*, pages 2690–2696.

Margaret M Bradley and Peter J Lang. 1999. Affective norms for english words (anew): Instruction manual and affective ratings. Technical report, Technical report C-1, the center for research in psychophysiology, University of Florida.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

[12]http://sentiwordnet.isti.cnr.it/

Billy Chiu, Anna Korhonen, and Sampo Pyysalo. 2016. Intrinsic evaluation of word vectors fails to predict extrinsic performance. In *Proceedings of the 1st Workshop on Evaluating Vector Space Representations for NLP*, pages 1–6.

William W Cohen. 2009. Enron email dataset.

Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In *51st Annual Meeting of the Association for Computational Linguistics*, pages 250–259. ACL.

John M Digman. 1990. Personality structure: Emergence of the five-factor model. *Annual review of psychology*, 41(1):417–440.

Paul Ekman. 1992. An argument for basic emotions. *Cognition & Emotion*, 6(3-4):169–200.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simverb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2173–2182.

Sayan Ghosh, Mathieu Chollet, Eugene Laksana, Louis-Philippe Morency, and Stefan Scherer. 2017. Affect-lm: A neural language model for customizable affective text generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 634–642. Association for Computational Linguistics.

Anna Gladkova and Aleksandr Drozd. 2016. Intrinsic evaluations of word embeddings: What can we do better? In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 36–42.

Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1681–1691.

Justine Kao and Daniel Jurafsky. 2012. A computational analysis of style, affect, and imagery in contemporary poetry. In *CLfL@NAACL-HLT*.

Douwe Kiela, Felix Hill, and Stephen Clark. 2015. Specializing word embeddings for similarity or relatedness. In *EMNLP*, pages 2044–2048.

Maximilian Köper and Sabine Schulte Im Walde. 2016. Automatically generated affective norms of abstractness, arousal, imageability and valence for 350 000 german lemmas. In *LREC*.

Maximilian Köper, Evgeny Kim, and Roman Klinger. 2017. Ims at emoint-2017: emotion intensity prediction with affective norms, automatically extended resources and deep learning. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 50–57.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113.

Navonil Majumder, Soujanya Poria, Alexander Gelbukh, and Erik Cambria. 2017. Deep learning-based document modeling for personality detection from text. *IEEE Intelligent Systems*, 32(2):74–79.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Saif Mohammad and Felipe Bravo-Marquez. 2017. Wassa-2017 shared task on emotion intensity. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 34–49. Association for Computational Linguistics.

Saif Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 321–327.

Nikola Mrkšic, Diarmuid OSéaghdha, Blaise Thomson, Milica Gašic, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of NAACL-HLT*, pages 142–148.

Elisavet Palogiannidi, E Losif, Polychronis Koutsakis, and Alexandros Potamianos. 2015. Valence, arousal and dominance estimation for english, german, greek, portuguese and spanish lexica using semantic models.

Ellie Pavlick and Joel Tetreault. 2016. An empirical analysis of formality in online communication. *Transactions of the Association for Computational Linguistics*, 4:61–74.

James W Pennebaker and Laura A King. 1999. Linguistic styles: Language use as an individual difference. *Journal of personality and social psychology*, 77(6):1296.

J.W. Pennebaker. 2011. *The Secret Life of Pronouns: What Our Words Say About Us*. Bloomsbury USA.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Rosalind W. Picard. 1997. *Affective Computing*. MIT Press, Cambridge, MA, USA.

Daniel Preotiuc-Pietro, Ye Liu, Daniel J. Hopkins, and Lyle Ungar. 2017. Personality Driven Differences in Paraphrase Preference. In *Proceedings of the Workshop on Natural Language Processing and Computational Social Science (NLP+CSS)*, ACL.

Gabriel Recchia and Max M Louwerse. 2015. Reproducing affective norms with lexical co-occurrence statistics: Predicting valence, arousal, and dominance. *The Quarterly Journal of Experimental Psychology*, 68(8):1584–1598.

Filipe N Ribeiro, Matheus Araújo, Pollyanna Gonçalves, Marcos André Gonçalves, and Fabrício Benevenuto. 2016. Sentibench-a benchmark comparison of state-of-the-practice sentiment analysis methods. *EPJ Data Science*, 5(1):1–29.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense word embeddings by orthogonal transformation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 767–777.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Klaus R. Scherer, Tanja Banziger, and Etienne Roesch. 2010. *A Blueprint for Affective Computing: A Sourcebook and Manual*. Oxford University Press, Inc., New York, NY, USA, 1st edition.

Joao Sedoc, Jean Gallier, Dean Foster, and Lyle Ungar. 2017a. Semantic word clusters using signed spectral clustering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 939–949, Vancouver, Canada, July. Association for Computational Linguistics.

Joao Sedoc, Daniel Preotiuc-Pietro, and Lyle Ungar. 2017b. Predicting Emotional Word Ratings using Distributional Representations and Signed Clustering. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, EACL.

Robert J Sigley. 1997. Text categories and where you can stick them: a crude formality index. *International Journal of Corpus Linguistics*, 2(2):199–237.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Hendrik Vankrunkelsven, Steven Verheyen, Simon De Deyne, and Gerrit Storms. 2015. Predicting lexical norms using a word association corpus. In *Proceedings of the 37th Annual Conference of the Cognitive Science Society*, pages 2463–2468. Cognitive Science Society.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164.

Ivan Vulić, Nikola Mrkšić, Roi Reichart, Diarmuid Ó Séaghdha, Steve Young, and Anna Korhonen. 2017. Morph-fitting: Fine-tuning word vector spaces with simple language-specific rules. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 56–68.

Jin Wang, Liang-Chih Yu, K Robert Lai, and Xuejie Zhang. 2016. Community-based weighted graph model for valence-arousal prediction of affective words. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):1957–1968.

Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior Research Methods*, 45(4):1191–1207.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 3:345–358.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1219–1228. ACM.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *ACL (2)*, pages 545–550.

Liang-Chih Yu, Jin Wang, K Robert Lai, and Xuejie Zhang. 2017. Refining word embeddings for sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 534–539.

Appendix A Generalization of Affect and Emotion Dimensions

Apart from the Warriner's lexicon (Warriner et al., 2013) which is in the VAD space, we experimented with the NRC Affect Intensity lexicon by Mohammad (2013) and Ekman's Six emotions from IMS EmoInt Norms dataset[13](Köper et al., 2017; Ekman, 1992). The NRC lexicon has 4 dimensions whereas the Ekman has 6 dimensions. Table A.1 shows the intrinsic word similarity measures for both these affect spaces. The enriched embeddings reported here are based on Affect-APPEND. We report variants of the lexica with vanilla embeddings and their retrofitted and counterfitted versions. The intrinsic metrics improve over the vanilla embeddings with the Aff2Vec versions. Note that the improvement achieved here is slightly lower than those achieved with the VAD space.

This analysis supports the hypothesis that Aff2Vec is generalizable to other dimension spaces and not restricted to a specific affect distribution.

Table A.1: Intrinsic Evaluation: Word Similarity–We report the Spearman's correlation coefficient (ρ) against NRC as well as Ekman's dimensions. The results show that Aff2Vec variants improve performance consistently

Model	**Word Similarity**							
	SL	**SV**	**WS**	**RG**	**RW**	**SCWS**	**MC**	**MEN**
GloVe	0.41	0.28	0.74	0.77	0.54	0.64	0.80	0.80
$\oplus$ Ekman's Six	0.46	0.34	0.77	0.78	0.58	0.67	0.80	0.83
+ Retrofitting $\oplus$ Ekman's Six	0.57	0.42	**0.79**	0.83	**0.61**	**0.70**	**0.82**	**0.85**
+ Counterfitting $\oplus$ Ekman's Six	**0.61**	**0.50**	0.60	0.83	**0.61**	0.64	0.80	0.80
$\oplus$ NRC Affect	0.47	0.34	0.78	0.78	0.58	0.67	0.81	0.84
+ Retrofitting $\oplus$ NRC Affect	0.58	0.43	**0.79**	0.82	**0.61**	**0.70**	0.81	**0.85**
+ Counterfitting $\oplus$ NRC Affect	**0.61**	**0.50**	0.70	**0.84**	**0.61**	0.64	0.81	0.80
Word2Vec	0.45	0.36	0.70	0.76	0.59	0.67	0.80	0.78
$\oplus$ Ekman's Six	0.46	0.38	0.69	0.80	0.59	0.67	0.82	0.79
+ Retrofitting $\oplus$ Ekman's Six	0.57	0.45	**0.73**	**0.85**	0.61	0.70	**0.85**	**0.82**
+ Counterfitting $\oplus$ Ekman's Six	**0.59**	**0.52**	0.65	0.80	0.60	0.65	0.77	0.76
$\oplus$ NRC Affect	0.47	0.38	0.69	0.79	0.59	0.67	0.82	0.79
+ Retrofitting $\oplus$ NRC Affect	0.58	0.46	0.72	0.84	**0.62**	**0.71**	**0.85**	**0.82**
+ Counterfitting $\oplus$ NRC Affect	**0.59**	**0.52**	0.65	0.79	0.60	0.65	0.77	0.76
Paragram	0.69	0.54	**0.73**	0.78	0.59	**0.68**	0.80	0.78
$\oplus$ Ekman's Six	0.69	0.55	0.72	0.79	0.59	**0.68**	0.77	**0.79**
+ Retrofitting $\oplus$ Ekman's Six	0.69	0.56	0.72	0.81	0.59	**0.68**	0.80	**0.79**
+ Counterfitting $\oplus$ Ekman's Six	**0.74**	**0.63**	0.69	**0.83**	**0.60**	0.65	**0.85**	0.76
$\oplus$ NRC Affect	0.70	0.55	0.72	0.78	0.59	**0.68**	0.77	**0.79**
+ Retrofitting $\oplus$ NRC Affect	0.69	0.55	0.72	0.80	0.59	**0.68**	0.80	**0.79**
+ Counterfitting $\oplus$ NRC Affect	**0.74**	**0.63**	0.70	0.82	**0.60**	0.65	0.84	0.76

Appendix B Model and Architecture Details

The architecture and hyperparameter details of various models used in the extrinsic evaluation tasks are presented here.

B.1 Sentiment Analysis

We use a Deep Average Network (`https://github.com/miyyer/dan`) with default parameters on binary Stanford Sentiment Treebank (Manning et al., 2014) for this task. Results reported in the paper are accuracies averaged across 5 independent runs.

[13]`http://www.ims.uni-stuttgart.de/forschung/ressourcen/experiment-daten/IMS_emoint_norms.tar.gz`

B.2 Personality Detection

A Convolutional Neural Network(CNN)-based model proposed by Mujumder et al (2017) is used in this paper to evaluate the performance of Aff2Vec embeddigns on the Personality Detection task. We use their Github implementation (`https://github.com/SenticNet/personality-detection`) with *Filter = True, Classifier = MLP, Convolution-Filter = [1,2,3] and Cross-Validation = 10*.

B.3 Emotion Intensity Task (WASSA)

For the WASSA EmoInt-2017, we train a BiLSTM-CNN model with word embeddings as input features. The model is trained separately for each emotion. The network architecture is explained in Table B.1. We use Adam as the optimizer and report Pearson's correlation coefficient(ρ), averaged across 5 independent runs.

Table B.1: BiLSTM-CNN Architecture for WASSA EmoInt-2017

Layer	**Properties**
1D Convolution	$filters = 200, kernel\_size = 3$
Activation	$relu$
1D Max_pooling	$pool\_size = 2$
Dropout	0.3
BiLSTM	$units = 150$
Activation	$relu$
Dropout	0.2
BiLSTM	$units = 80$
Dense	$size = 50$
Activation	$relu$
Dropout	0.3
Dense	$size = 1$

Table B.2: CNN architecture for FFP-Prediction

Layer	**Properties**
2D Convolution	$filters = 5, kernel\_size = 10X5$
Activation	$relu$
2D Max_pooling	$pool\_size = 5X5, sride = 5$
Dense	$size = 50$
Activation	$relu$
Dropout	0.2
Dense	$size = 1$

B.4 Affect Prediction (FFP-Prediction)

We implement a basic CNN-based model for Formality, Frustration and Politeness prediction on Enron-FFP dataset introduced in the paper. The network architecture is shown in Table B.2. We use Rectified Linear Unit (ReLU) as the activation throughout and Stochastic Gradient Descent(SGD) as the optimizer. A mean squared loss is used for regression. We report mean square error averaged across 10 independent runs. Standard deviation for the reported MSE values is of the order of 10^{-3}.

Aspect-based summarization of pros and cons in unstructured product reviews

Florian Kunneman
Tilburg University
PO Box 90153
5000LE Tilburg, The Netherlands
f.a.kunneman@uvt.nl

Sander Wubben
Tilburg University
PO Box 90153
5000LE Tilburg, The Netherlands
s.wubben@uvt.nl

Emiel Krahmer
Tilburg University
PO Box 90153
5000LE Tilburg, The Netherlands
e.j.krahmer@uvt.nl

Antal van den Bosch
KNAW Meertens Institute
Oudezijds Achterburgwal 185
1024DK Amsterdam, The Netherlands
antal.van.den.bosch@meertens.knaw.nl

Abstract

We developed three systems for generating pros and cons summaries of product reviews. Automating this task eases the writing of product reviews, and offers readers quick access to the most important information. We compared SynPat, a system based on syntactic phrases selected on the basis of valence scores, against a neural-network-based system trained to map bag-of-words representations of reviews directly to pros and cons, and the same neural system trained on clusters of word-embedding encodings of similar pros and cons. We evaluated the systems in two ways: first on held-out reviews with gold-standard pros and cons, and second by asking human annotators to rate the systems' output on relevance and completeness. In the second evaluation, the gold-standard pros and cons were assessed along with the system output. We find that the human-generated summaries are not deemed as significantly more relevant or complete than the SynPat systems; the latter are scored higher than the human-generated summaries on a precision metric. The neural approaches yield a lower performance in the human assessment, and are outperformed by the baseline.

1 Introduction

Reviews posted on Web-based consumer platforms such as Amazon[1] and Trustpilot[2] form a valuable source of information prior to buying a product or hiring a service. Especially when a multitude of reviews of a specific product are gathered, the different experiences enable the consumer to obtain a well-informed conception of the qualities and deficiencies of the respective article. However, achieving a coherent view of a product tends to be challenging when a high volume of reviews diverge in focus and sentiment. A system that can summarize the reviews of a product would therefore be a valuable asset to these platforms. This work describes the implementation and evaluation of both a supervised and an unsupervised approach to aspect-based sentiment analysis, with a focus on Dutch product reviews of electric devices.

We create summaries that zoom in on the pros and cons of the product that are mentioned in a review. Platforms that include product reviews commonly employ a pros and cons template to enable the authors of reviews themselves to summarize their review text. Although such a procedure could lead to

[1] https://www.amazon.com/

[2] https://www.trustpilot.com/

Proceedings of the 27th International Conference on Computational Linguistics, pages 2219–2229
Santa Fe, New Mexico, USA, August 20-26, 2018.

appropriate summaries, only a minority of users make the effort to fill in the pros and cons.[3] On the one hand, this behavior underlines the need for a tool to generate a summary when the writer chooses not to. On the other hand, the review texts that do include pros and cons as explicated by the writer offers the train and test labels to employ a supervised learning framework. We will study the potential of using these user-generated summaries by training a neural-network implementation to generate the appropriate summary as output to the review text as input. This approach will be compared to a knowledge-driven approach based on syntactic patterns.

Aspect-based sentiment analysis from product reviews, of which the detection and summarization of pros and cons is an instance, is a challenging task due to the freedom that the user has when writing a review. In this case, the freedom applies to both the pros and cons the user chose to mention, and to the phrasings used to express these pros and cons. In addition, different types of products are reviewed with a focus on different aspects. For this reason, aspect-based sentiment analysis is commonly approached as a bounded task, with a focus on a specific product or service with predefined aspects. In contrast, we leverage the pros and cons that writers attach to their review, resulting in a more dynamic approach that can cope with any set of reviews, irrespective of the product or service, as long as example summaries are available. While the pros and cons as put forward by the users offers a gold standard for evaluation, we also include a human evaluation in which the qualities of both the system-generated summaries and the human-generated summaries are assessed.

This study offers the following contributions:

- We provide new insights into the utility of author-contributed pros and cons for automatically summarizing product reviews;
- We compare a neural learning approach with a linguistic pattern-driven approach;
- A novel evaluation is proposed that uses the author-assigned pros and cons as gold standard;
- The quality of the author gold standard is compared to system output through human evaluation.

After discussing related work, this paper will proceed with a description of the methods. Subsequently, the experimental set-up will be outlined, followed by the results. The paper ends with a conclusion and discussion.

2 Related Work

Studies in aspect-based sentiment analysis can roughly be divided into rule-based and machine learning-based approaches. Rule-based approaches implement searching for combinations of aspects and evaluations on the sentence level. Such combinations are commonly identified based on syntactic patterns (Poria et al., 2014; Zhang et al., 2014; Chinsha and Joseph, 2015; Rana and Cheah, 2015; Yan et al., 2015) while the evaluation is scored based on a sentiment lexicon (Chinsha and Joseph, 2015; Rana and Cheah, 2015; Poria et al., 2014), a PageRank-based procedure (Yan et al., 2015), or optimization based on review labels (Zhang et al., 2014). We apply a knowledge-driven approach to aspect-based sentiment analysis, comparable to the approach by Rana and Cheah (2015). Candidate phrases are identified based on predefined syntactic patterns and a sentiment lexicon is used to identify patterns that likely signify a pro or con.

Utilizing the pros and cons in product reviews as training data, we also apply a machine-learning based approach, using neural networks. Neural models have become popular for text classification tasks (Socher et al., 2013; Zhang et al., 2015; Conneau et al., 2016) in recent years. These models have the advantage that they generally require little feature engineering. The downside is that they are slow to train and test and require large amounts of training data. More recently, it has been shown that shallow neural models obtain performance on par with deep learning methods, while being much faster and less

[3] Of the reviews that are used in our study, only 30% include pros and/or cons. A full overview of the reviews is given in Table 2

data hungry. The higher representational power of these deep methods is not required for relatively simple tasks such as sentiment classification (Joulin et al., 2016).

Few studies on aspect-based sentiment analysis make use of the pros and cons that are given by the writers themselves to summarize their review. Closest to our work in this respect is the contribution by Kim and Hovy (2006), who align pros and cons with the most likely related sentence in their review text by a maximum entropy model. The output is used as training data to predict the pros and cons for reviews that were not summarized as such by the writer. In comparison to Kim and Hovy (2006), we leverage author-based pros and cons both as training labels and as test labels during evaluation. In addition, we evaluate these pros and cons along with system-generated pros and cons, thereby providing insight into the quality of these kinds of labels for training and evaluating aspect-based sentiment analysis of product reviews.

3 Method

3.1 System 1: Syntactic patterns of subjective phrases (SynPat)

The knowledge-driven SynPat system[4] tracks and matches syntactic cues towards aspect evaluations in reviews (Rana and Cheah, 2015). The system consists of two stages. First, a review is scanned for particular syntactic patterns based on a set of rules. Second, each candidate that is found during the first stage is matched with a lexicon of subjective words to confirm that the phrase indicates valence.

Part-of-speech tags and grammatical phrases are extracted from each review text by means of Frog[5], an off-the-shelf NLP system for Dutch (Van den Bosch et al., 2007). We matched the automatic analyses with a set of syntactic patterns of which the lexical realizations (the words associated with the PoS-tags) may reflect statements on the valence of an aspect. Examples of patterns and example lexical realizations are given in Table 1.

To assess the presence and direction of the valence in the lexical realizations (henceforth referred to as phrases) found with matching syntactic patterns, we matched them to the Duoman subjectivity lexicon (Jijkoun and Hofmann, 2009). This a lexicon of nouns and adjectives rated by human annotators as 'Very negative', 'Negative', 'Neutral', 'Positive' and 'Very positive'. We labeled each phrase with a positive or very positive word as 'pro', and each phrase with a negative or very negative word as 'con'. Phrases in which no word has a positive or negative valence in Duoman lexicon were discarded, as well as phrases that matched an equal amount of positive and negative words. A phrase that carries a combination of a 'positive' and 'very negative' is labeled as 'con', while phrases with a 'very positive' and 'negative' word are labeled as 'pro'.

Syntactic pattern	Example (Dutch)	Example English	Valence word	Pro/con
ADJ	fantastisch	fantastic	fantastic	Pro
ADJ-N	klein zuigmondje	small nozzle	small	Con
ADJ-N-N	heerlijk kopje espresso	delicious cup of espresso	delicious	Pro
ADJ-ADJ-N	onhandige extra handeling	inconvenient additional action	inconvenient	Con
ADJ-V	mooi uitgevoerd	nicely rendered	nicely	Pro
ADJ-Prep-N	goed van smaak	good flavor	good	Pro

Table 1: Examples of syntactic patterns extracted in the SynPat system (ADJ=Adjective, N=Noun, V=Verb, Prep=Preposition), their lexical realization (phrase)

3.2 System 2: Shallow neural classification

To explore the possibility of training a classification system end-to-end without feature engineering we choose to train a shallow neural network on the aspect summarization task. We treat reviews as bags of words and feed word embeddings to the neural classifier. Given the constructed sentence vector, the

[4]The implementation of the system can be found on github: `https://github.com/fkunneman/Product_review_summary/blob/master/synpat.py`

[5]`http://applejack.science.ru.nl/languagemachines/software/frog/`

classifier tries to predict the correct labels. Here, the labels are the human generated aspect sentences. We treat each aspect as a potential class to predict. The chosen architecture is similar to Joulin et al. (2016) which is a modification of the cbow model by Mikolov et al. (2013), but instead of predicting a hidden word, a hidden class is predicted. We use softmax to compute the probability distribution over the aspect classes. The model is trained using stochastic gradient descent and a linearly decaying learning rate. To pretrain the model we use 320-dimensional word embeddings derived from the corpora from the web (COW) introduced in Tulkens et al. (2016).

3.3 System 3: Shallow neural classification with clustering

Because users can state positive and negative aspects in any way they want, we end up with a large number of classes in a long-tail distribution. From inspection of the development data we observe that many of the different classes are in fact paraphrases of the same semantic content ('great coffee', 'coffee tastes great', etc.) To try to alleviate this problem, we add a preprocessing step where we cluster the aspect summaries based on averaging over word embedding vectors. Using k-means clustering we then cluster the summaries into distinct clusters and take the centroid as the label. We then try to predict these labels using the same shallow neural classifier.

4 Experimental Set-up

4.1 Data

We crawled product reviews from the Dutch platform 'kieskeurig.nl', which features consumer reviews for a diversity of electric devices. A screen shot of a typical consumer review is presented in Figure 1. Typically, such a review consists of a written text, a summary of the written text in the form of pros and cons, a review score and, occasionally, scores on predefined aspects such as design and ease of use. We are interested in the review text and the accompanying pros and cons.

Figure 1: Example of a review placed on kieskeurig.nl. 'Pluspunten' are pros, 'Minpunten' cons.

We crawled the reviews for the four most frequently reviewed product types: smartphones, vacuum cleaners, deep fryers and espresso machines.[6] The statistics of the resulting dataset are presented in Table 2. We observe that a significant part of the reviews are written without summing up explicit pros and cons. In addition, pros are more common than cons. About 35% of the reviews are accompanied by one or more explicated pros, while cons are present for 22% of the reviews.

4.2 Procedure

The review texts summarized with pros and cons were utilized as instances to train our systems with, and to test them intrinsically on held-out data from the same source. In order to tune the neural networks, we

[6]Due to privacy regulations, we are not allowed to publicly share this data.

Device	# reviews	# reviews with at least one pro	# reviews with at least one con	# reviews with both pros and cons	# reviews with either a pro or a con
Smartphone	5,133	1,007	703	697	1,012
Deep fryer	4,040	1,526	1,039	1,019	1,536
Vacuum cleaner	2,623	1,026	730	715	1,041
Espresso machine	2,282	979	682	666	986
Total	14,078	4,538	3,154	3,097	4,575

Table 2: Overview of the review data set

Device	# train	# dev	# test	# total
Smartphone	809	101	102	1,012
Deep fryer	1,228	154	154	1,536
Vacuum cleaner	832	104	105	1,041
Espresso machine	793	99	100	986

Table 3: Overview of the data used in the experimental set-up

split the reviews into training (80%), development (10%) and test (10%) instances. The resulting dataset is presented in Table 3.

The unsupervised SynPat system was applied directly to the test set. The hyperparameters of the two shallow neural systems were set using the development set. We trained for 30 epochs using a learning rate of 0.5 and the softmax loss function. For k-means clustering we set $k = 100$.

We compared the performance of the three systems to the performance of a baseline system based on string matching. This works as follows. All pros and cons contained in the training reviews of a specific device are saved as possible descriptions of unseen reviews. For a given unseen review, all known pros and cons from the training reviews are matched with the text. The baseline selects as summary the pros and cons of which an above-threshold proportion of words is seen in the text. The threshold value is empirically decided based on the development set: the threshold that leads to the pros and cons that are closest to the actual given pros and cons is selected and used for the predictions on the test set.[7]

4.3 Evaluation

We evaluated the systems in two ways: a gold standard assessment based on the pros and cons given by the writers of the reviews and a human evaluation in which the pros and cons given by the systems as well as the writers themselves are assessed by human annotators.

4.3.1 Evaluation 1: Gold standard

For the gold standard evaluation, we assessed the pros and cons extracted by a system as their closeness in characters to the author-based pros and cons.[8] Due to the variance in these pros and cons, which are given by the author in a free-text input field, it is infeasible to make direct matches between two sets of pros and cons. Slight grammatical differences between target and prediction would be assessed as a false prediction in such a procedure. Alternatively, we made use of the FuzzyWuzzy string matching library for python.[9] At the basis of this implementation is a computation of the edit distance between two strings using the Ratcliff–Obershelp pattern recognition algorithm (Ratcliff and Metzener, 1988). In this algorithm, the edit distance between two strings is computed by dividing the number of matching characters with the total number of characters. The FuzzyWuzzy library is particularly adequate for

[7]The implementation of the baseline can be found on github: `https://github.com/fkunneman/Product_review_summary/blob/master/baseline.py`

[8]The implementation of evaluation 1 can be found on github: `https://github.com/fkunneman/Product_review_summary/blob/master/evaluation1.py`

[9]`https://github.com/seatgeek/fuzzywuzzy`

assessing the distance between longer phrases, as it expands the core pattern recognition algorithm with two heuristics. The first is to overrule the strong penalization of two strings with a differing length by assigning a more positive score if one of the two strings is a substring of the other. The second heuristic assures that the order of word tokens is not of influence to the edit distance, by separately assessing the intersection of tokens between two phrases as two unordered sets.

Using the FuzzyWuzzy string matching procedure, the quality of the systems are assessed in the following way, for each review in the test set:

1. The pros and cons suggested by the system and the ones given by the user are aligned such that a predicted pro is matched with only one gold standard pro, the most similar one, and a predicted con is matched with only one gold standard con. Two predicted pros or cons can not be aligned with the same gold standard pro or con; the pair with the highest similarity is aligned.

2. Apart from string matching, we employed a heuristic for matching values of pros and cons that refer to 'Nothing'. This is the most common value (especially to the cons field), and is often given by not filling in anything or using words like 'geen' ('none') and 'nog niets gevonden' ('nothing found yet'). The heuristic assures that such values were matched. Such matches were given the optimal similarity score.

3. Aligned pairs with a similarity that surpasses a given threshold are scored as 'true positives'. Other pairs are scored as false positives (for system-generated pros and cons) and false negatives (for user-generated pros and cons). Pros and cons that are not aligned are also scored as 'false positives' and 'false negatives'

4. A precision, recall and F1-score for the summary of each single review is calculated based on the counts of true positives, false positives and false negatives. After all reviews in the test set are processed this way, an aggregate precision, recall and F1-score of the system is calculated.

In this evaluation procedure, true positives are assigned when the similarity between a target pro or con and a predicted pro or con surpasses a given threshold. To obtain insight into the influence of the threshold value, we will run the evaluation with different thresholds that cover the total spectrum of possible edit distance scores. The range of thresholds spans from 0 (no similarity at all) to 100 (complete similarity). Precision, recall and F1-score is calculated separately per review rather than adding up all true positives, false negatives and false positives, so as to avoid the influence of imbalance in the number of pros and cons per review.

4.3.2 Evaluation 2: Human assessment

In addition to a gold standard evaluation, we asked human annotators to assess the quality of review summaries. Apart from the pros and cons generated by the baseline system, SynPat, Neural and Neural_Clust, we included the pros and cons given by the authors themselves (the gold standard pros and cons) in this evaluation. This can give insight into the absolute quality of the gold standard, and its quality with respect to the systems. This kind of experimentally collected (human) judgments is often used in the evaluation of automatically generated text (Gatt and Krahmer, 2018).

We randomly selected twenty reviews from the test set, equally distributed over the four devices 'vacuum cleaner', 'deep fryer', 'espresso machine' and 'smartphone' (e.g.: five reviews per device). Each review was included five times in the evaluation set: linked with the pros and cons generated by each of the four systems as well as the pros and cons given by the author of the review. This led to a total of 100 instances. We divided them into five chunks of twenty instances to be presented to human annotators, which would include all twenty unique review texts presented with the pros and cons of one of the five sources (equally distributed over the twenty reviews). Each chunk of twenty review-system combinations was assessed by the same ten human annotators, enabling us to calculate the agreement between each annotator pair.

Annotators were recruited through the Radboud Research Participation System[10], a university-based platform that helps researchers finding student participants for their studies. For the assessment of twenty review texts, participants could choose to receive one student credit or a voucher worth ten Euros. In total, fifty human annotators assessed the quality of the pros and cons of twenty review texts. For each of these instances, the annotator was asked to assess both the relevance and the completeness of the given pros and cons with respect to the review text. Relevance was assessed by deciding for each of the given pros and cons whether they are actually discussed (literally or paraphrased) in the review text. Completeness was assessed by indicating, on a scale from 1 to 7, the extent to which the given summary covers all pros and cons that are mentioned in the review text. Relevance can be seen as an estimator of precision; completeness is a human-assessed approximation of recall.

A completeness and relevance score per review-system combination was computed as the average of the ten human assessments. The score per system is calculated as the average of these averages.

5 Results

An overview of the number of (aligned) predictions by system is presented in Table 4. Recall that the pros and cons in the predicted and gold standard summaries are aligned based on string matching and a heuristic for the most common phrases to express that there is no pro or con. In the latter case, the highest similarity score of 100 is assigned to the alignment. We therefore present the average similarity of aligned pros and cons with and without the ones aligned based on this 'empty' heuristic.

The baseline is characterized by a high number of predictions and alignments. The proportion of predictions that are aligned is, however, the lowest. Most of these alignments are based on string matching, which has the highest average similarity (50.5) of all systems. The SynPat system makes the least predictions, but yields a relatively high proportion of alignments and a high number of reviews with an aligned prediction. The average similarity of the alignments is slightly lower than the baseline. The Neural system yields the lowest number of aligned pros and cons based on string matching (468), but the highest number of alignments based on the 'empty' heuristic. The percentage of reviews with an alignment is low at 81%, which is due to the high amount of falsely predicted 'empties'. This seems to be the default option for Neural when certainty for other pros or cons is low. The Neural_clust system makes aligned predictions for most of the reviews, at an average string similarity of 44.43.

System	Predicted pros and cons	Aligned pros and cons (string matching)	Aligned pros and cons ('empty' heuristic)	# reviews aligned (% of total)	Avg. similarity	Avg. similarity (without 'empty')
Baseline	3,877	1,469	79	432 (94%)	52.50	50.05
SynPat	1,712	1,101	83	442 (96%)	51.41	48.42
Neural	1,844	468	159	375 (81%)	58.58	44.99
Neural_clust	1,844	1,228	109	456 (99%)	48.96	44.43

Table 4: Overview of predictions and alignments by system for all reviews (# target reviews: 461, # target pros and cons: 2060, similarity scaled 0-100).

5.1 Gold standard

A similarity threshold decides which pros and cons predicted by a system are assessed as matches (true positives) with a gold standard pro or con. The resulting F1-scores by threshold value are presented in Figure 2 (vacuum cleaner), Figure 3 (smartphone), Figure 4 (deep fryer) and Figure 5 (espresso machine).

The curves for most systems are showing a steep drop in performance around a threshold of 30, after which the scores are more stable between 50 and 70 and steadily decrease towards an F1-score of about 0.1 at 100. The SynPat and Neural_clust systems yield the highest performance between a threshold value of 0 and 30–40 for all systems, indicating that these systems make the most predictions that are

[10] https://radboud.sona-systems.com

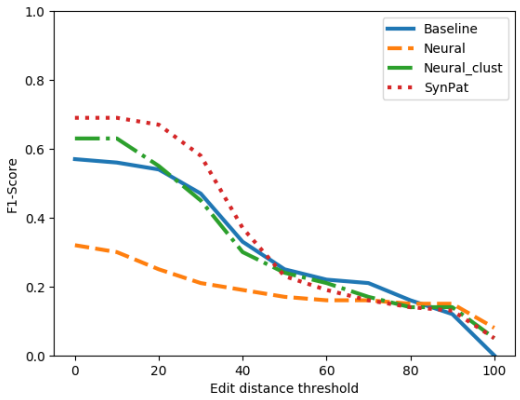


Figure 2: Performance by similarity threshold on vacuum cleaner reviews

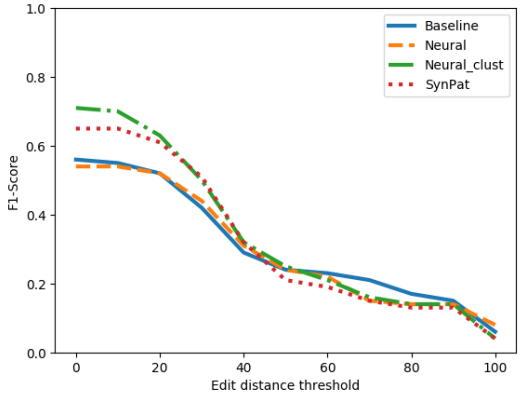


Figure 3: Performance by similarity threshold on smartphone reviews

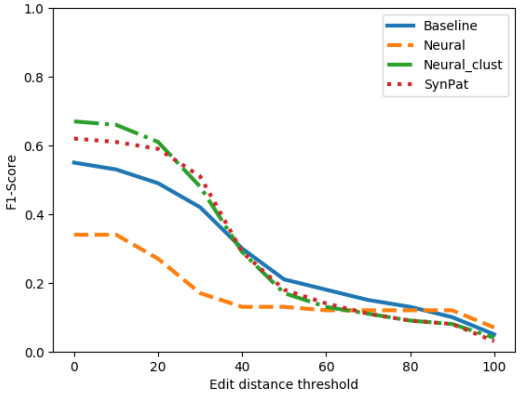


Figure 4: Performance by similarity threshold on deep fryer reviews

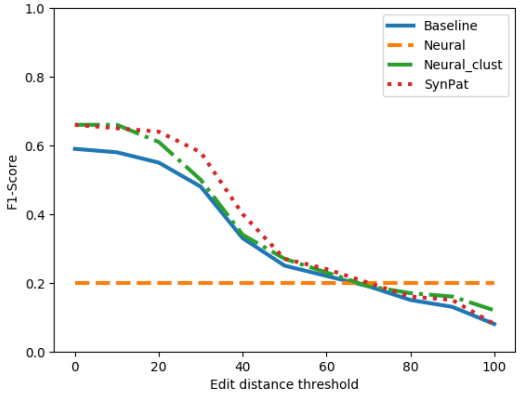


Figure 5: Performance by similarity threshold on espresso machine reviews

somewhat similar to one of the gold standard pros or cons. The F1-scores of all systems are closer to each other at distance thresholds of over 40. The baseline system outperforms the other systems between the threshold values of 60 and 80. The Neural system is generally outperformed at threshold values below 60 (except for the smartphone reviews). The clustering of pros and cons seems to be useful for this approach. Upon closer inspection, the flat line for the Neural system on the espresso machine reviews (Figure 5) is due to the exclusive prediction of 'Nothing' for cons.

5.2 Human assessment

A report of the human assessments of the pros and cons given by each system, as well as the writers of the reviews themselves, is given in Table 5. A Cohen's Kappa (Cohen, 1960) agreement was calculated between the relevance assessments of pros and cons of each pair of annotators. The average agreement on relevance was $\kappa = 0.56$. The agreement over the completeness assessment of pros and cons between each annotator pair was calculated as a weighted Cohen's Kappa. The average agreement for completeness was $\kappa = 0.30$. These Kappa-values suggest a moderate agreement between human judges.

Inspection of Table 5 reveals that the reviewers' summaries were judged to be the most complete at an average score of 4.60, although the differences with the average score of SynPat (4.06) and the Baseline (3.90) appear to be small. Apparently, reviewers themselves often do not cover all pros and cons that they mention in their review text. The relevance of their pros and cons (0.61) is lower than the best system, SynPat (0.67), although this difference is again small. The baseline system lags behind in terms of relevance. The scores of the two Neural systems are lower than the scores of the other systems for both completeness and relevance, and are close to each other.

We conducted two one-way Analyses of Variance (ANOVAs) to test for statistical significance of the

	Completeness	Relevance
Baseline	3.90 (0.90)	0.44 (0.17)
SynPat	4.06 (1.15)	0.67 (0.25)
Neural	2.74 (0.86)	0.25 (0.14)
Neural_clust	2.37 (0.62)	0.18 (0.10)
Reviewers summary	4.60 (1.13)	0.61 (0.25)

Table 5: Outcomes of the human assessment of the pros and cons generated by each system and the writers of the reviews themselves for 20 reviews. Completeness is rated on a scale from 1 to 7, and is reported as the average completeness of the 20 reviews (standard deviation between brackets). Relevance is rated as the proportion of pros and cons that are deemed relevant to a review text, on a scale from 0 to 1, reported as the average relevance of the 20 reviews (standard deviation between brackets).

completeness and relevance assessments. The ANOVA for completeness confirmed that the systems were not rated equally on this dimension, $F(4, 95) = 23.98, p < .001$. All pairwise comparisons (using the Tukey HSD method) were statistically significant at $p < .05$, except the comparisons between the human gold standard and SynPat, between human gold standard and baseline, and between SynPat and baseline (confirming that the three highest rated approaches performed statistically equally well) and between the two neural systems (confirming that neither neural method outperformed the other in terms of completeness). The ANOVA for relevance similarly showed that not all systems performed equally well, $F(4, 95) = 18.61, p < .001$, with all pairwise comparisons (Tukey HSD) statistically signifcant at $p < .05$, except the ones between the human gold standard and SynPat (confirming that SynPat performs on a par with human reviewers), and between the two neural methods.

5.3 Analysis

We set out to investigate the strength of each system by counting the overlap of their predictions for each of the test reviews. To focus on the most sensible predictions, we only selected the predictions with a similarity of > 50 with one of the gold standard pros and cons. An overview of the overlap is presented in a Venn diagram in Figure 6.

The diagram shows a predominantly disjoint pattern, with only 3.6% of the correctly predicted pros and cons generated by all four systems. The combination of three systems with most correct predictions that a fourth system has not generated (5.8%) is formed by the baseline, SynPat and Neural_clust. In most of these cases, the Neural system has falsely predicted 'empty' as con. The most overlapping combination of systems is the combination baseline–SynPat, with 120 predictions (11.6%). Both systems are based on matching phrases, while the Neural systems operate at the sentence level. Apparently, a good part of the pros and cons can best be detected on the phrase level.

6 Conclusion and Discussion

We presented three systems for generating pros and cons summaries of product reviews: a system based on syntactic phrases and a valence lexicon (SynPat), a neural-network-based system trained on bag-of-words in the review text as input and the pros and cons listed by the writer as output (Neural), and the same system trained on clusters of similar pros and cons based on word embeddings (Neural_clust). These systems were applied on reviews of vacuum cleaners, smartphones, deep fryers and espresso machines. We find that the pros and cons assigned by the authors themselves are not deemed by human annotators as more relevant or complete than the summaries extracted by the SynPat system. In fact, the SynPat system is assessed as offering more relevant pros and cons than the human summaries (though not significantly more). The neural-network approaches yield a lower performance in the human assessment, and are also outperformed by the baseline.

The lower performance of the neural-network approaches is likely due to the formulation of the task as a classification task of a text (either encoded as a bag-of-words or as centroids of word embedding clusters) into many different output classes. Moving from this discriminatory approach to a generative

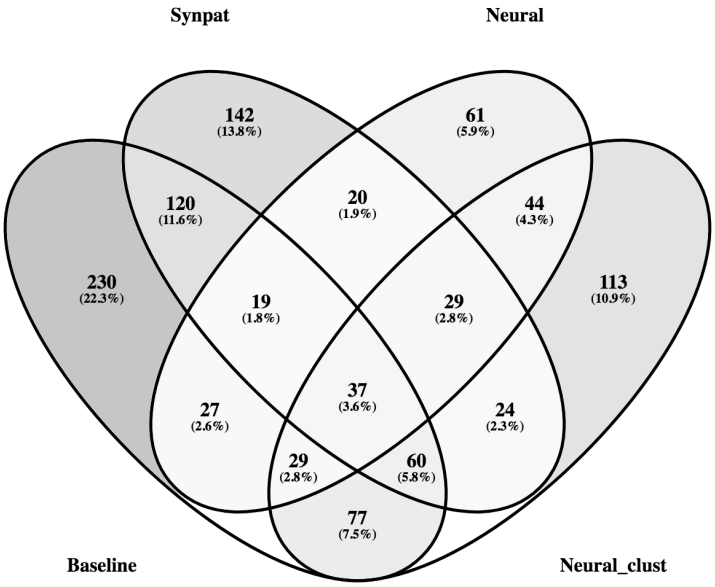


Figure 6: Venn diagram of the overlap between predicted pros and cons by each system which match the gold standard pros and cons at a similarity score of > 50.

approach may be a next step to attain better performances.

Our results show that human performance is imperfect as well. The SynPat system is assessed as not significantly worse than humans, and is even slightly more precise. The upper bound that the best systems appear to touch in Figures 2 to 5, an F1-score of about 0.7, must therefore be bounded by the variance and imperfect completeness (recall) and relevance (precision) of the human summaries we took as gold standard. In line with this finding, the moderate Cohen's Kappa scores that describe the agreement of human annotators on assessing the link between pros and cons and a review text ($\kappa = 0.56$ for relevance and $\kappa = 0.30$ for completeness) underlines the subjective nature of this task. In future work, subjectivity can be reduced (and thereby human agreement increased) by employing a stricter definition of when elements in a review text reflect a pro or con.

Finally, in future work we intend to move beyond extractive summarization of pattern-matched phrases. A strong assumption of the SynPat system is that pros and cons are consecutive word sequences that can be extracted literally from the full review, while pros and cons could be expressed in ways that transcend mere word n-grams. This calls for a richer encoding of the full review text, likely between the level of n-grams and the full bag-of-words. The sentence may be a good domain to target.

Acknowledgements

This work is part of the research programmes Dis- cussion Thread Summarization for Mobile Devices and The Automated Newsroom, which are financed by the Netherlands Organisation for Scientific Research (NWO). We thank the reviewers for their valuable comments.

References

TC Chinsha and Shibily Joseph. 2015. A syntactic approach for aspect based opinion mining. In *Semantic Computing (ICSC), 2015 IEEE International Conference on*, pages 24–31. IEEE.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for natural language processing. *arXiv preprint arXiv:1606.01781*.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170.

Valentin Jijkoun and Katja Hofmann. 2009. Generating a non-english subjectivity lexicon: Relations that matter. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 398–405. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Soo-Min Kim and Eduard Hovy. 2006. Automatic identification of pro and con reasons in online reviews. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 483–490. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Soujanya Poria, Erik Cambria, Lun-Wei Ku, Chen Gui, and Alexander Gelbukh. 2014. A rule-based approach to aspect extraction from product reviews. In *Proceedings of the second workshop on natural language processing for social media (SocialNLP)*, pages 28–37.

Toqir Ahmad Rana and Yu-N Cheah. 2015. Hybrid rule-based approach for aspect extraction and categorization from customer reviews. In *9th International Conference on IT in Asia (CITA)*. IEEE.

John W Ratcliff and David E Metzener. 1988. Pattern-matching-the gestalt approach. *Dr Dobbs Journal*, 13(7):46.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Stephan Tulkens, Chris Emmery, and Walter Daelemans. 2016. Evaluating unsupervised dutch word embeddings as a linguistic resource. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Antal Van den Bosch, Bertjan Busser, Sander Canisius, and Walter Daelemans. 2007. An efficient memory-based morpho-syntactic tagger and parser for Dutch. In P. Dirix, I. Schuurman, V. Vandeghinste, and F. Van Eynde, editors, *Computational Linguistics in the Netherlands: Selected Papers from the Seventeenth CLIN Meeting*, pages 99–114, Leuven, Belgium.

Zhijun Yan, Meiming Xing, Dongsong Zhang, and Baizhang Ma. 2015. Exprs: An extended pagerank method for product feature extraction from online consumer reviews. *Information & Management*, 52(7):850–858.

Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 83–92. ACM.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Learning Sentiment Composition from Sentiment Lexicons

Orith Toledo-Ronen Roy Bar-Haim* Alon Halfon Charles Jochim
Amir Menczel Ranit Aharonov Noam Slonim
IBM Research
{oritht,roybar,alonhal,amir.menczel,ranita,noams}@il.ibm.com
charlesj@ie.ibm.com

Abstract

Sentiment composition is a fundamental sentiment analysis problem. Previous work relied on manual rules and manually-created lexical resources such as negator lists, or learned a composition function from sentiment-annotated phrases or sentences. We propose a new approach for learning sentiment composition from a large, unlabeled corpus, which only requires a word-level sentiment lexicon for supervision. We automatically generate large sentiment lexicons of bigrams and unigrams, from which we induce a set of lexicons for a variety of sentiment composition processes. The effectiveness of our approach is confirmed through manual annotation, as well as sentiment classification experiments with both phrase-level and sentence-level benchmarks.

1 Introduction

Many sentiment analysis systems rely primarily on the sentiment of individual words. However, precise sentiment analysis often requires lexical-semantic knowledge that goes beyond word-level sentiment, even when dealing with short phrases such as bigrams. Phrases may be idiomatic, in which case their polarity cannot be inferred from the sentiment of their constituent words, e.g., *black market*, *on track*, and *let down*. The sentiment of most phrases, however, is *compositional*, namely, it can be determined from the interaction between their constituents.

Sentiment composition is a fundamental problem in sentiment analysis. It involves a variety of semantic phenomena, the most studied of which are *valence shifters* (Polanyi and Zaenen, 2004) that reverse sentiment polarity, e.g., ***not*** *happy*, ***hardly*** *helpful*, ***decrease*** *unemployment*, or change its intensity (***deeply/somewhat*** *disappointed*). Sentiment composition also needs to resolve the polarity of phrases containing opposing polarities, such as *cure cancer*, *sophisticated crime*, and *fake success*. Another, less studied type of sentiment composition is expressions such as *high income*, *long queue*, and *fast learner*, that include gradable adjectives. The sentiment of these expressions cannot be derived from the sentiment of their individual words (none of which have clear sentiment in the above examples). While they can be treated as fixed expressions, similar to idioms, it is beneficial to exploit their semantic compositionality. For example, for a given word *w*, the polarity of *high w*, *higher w*, and *increasing w* is likely to be the same, and opposite to the polarity of *low/lower w*. This allows more robust sentiment learning, as well as sentiment inference for such expressions.

Previous approaches for sentiment composition relied on hand-written lists of negation words and composition rules. Such knowledge is hard to acquire and is typically incomplete. Other approaches aim to learn sentiment composition from sentiment-labeled texts. However, sentiment-labeled texts might not be available for certain domains or languages.

In this work we propose a new approach for learning sentiment composition. We define a set of *sentiment composition classes* that cover a variety of sentiment composition processes, and propose a novel method for automatic acquisition of lexicons for each of these classes from a large corpus. For

* First two authors contributed equally.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2230–2241
Santa Fe, New Mexico, USA, August 20-26, 2018.

example, our method learns that words like *reduce* and *preventing*, when applied to a negative expression, change its polarity from negative to positive (e.g., *preventing violence*). It also learns that the positive sentiment of words like *strong* and *incredibly* is overridden when composed with a negative expression (e.g., *strong disapproval*). In addition, we learn composition lexicons for specific gradable adjectives, e.g., words *w* such that *high w* is positive (*morale, standard, competitiveness*), or negative (*cost ,anxiety, noise*).

Unlike previous approaches, our method does not require sentiment-labeled texts, or manually-created sentiment composition lexicons. The only annotated input required is a sentiment lexicon for individual words. We train an n-gram sentiment classifier on the input sentiment lexicon, using a novel sentiment-oriented phrase embedding that is very efficient to compute and scales well for very large corpora. The classifier is then applied to a large lexicon of bigrams, as well as to the unigrams they are made of. Finally, we induce from the automatically generated sentiment lexicon of bigrams and unigrams a set of fine-grained lexicons for sentiment composition.

The accuracy of the resulting sentiment composition lexicons is confirmed via manual assessment. We also show their contribution to both phrase-level and sentence level sentiment analysis. Our results illustrate the value of our automatically-generated sentiment lexicons for investigating sentiment composition phenomena. We made both these sentiment lexicons and the resulting composition lexicons publicly available, to facilitate further research on sentiment composition.[1]

2 Related Work

Many of the previous works on sentiment analysis include some treatment of valence shifters, based on manual lists of negators, intensifiers, etc. (Wilson et al., 2005; Kennedy and Inkpen, 2006; Taboada et al., 2011). Moilanen and Pulman (2007) apply manually-composed syntactic rules for sentiment composition over the syntactic parse. Neviarouskaya et al. (2010) manually created a lexicon with fine-grained categories for sentiment composition such as *propagation* and *domination* which may resolve conflicting sentiments. Schulder et al. (2017) applied an SVM classifier with linguistic features to bootstrap manual construction of a verbal polarity shifters lexicon. Some researchers, like Choi and Cardie (2008), combined manual composition rules with machine learning.

Other works aimed to learn a sentiment composition function from sentiment-labeled phrases or sentences by employing conditional random fields (Nakagawa et al., 2010), compositional matrix-space models (Yessenalina and Cardie, 2011), statistical parsing (Dong et al., 2015) and recursive neural networks (Socher et al., 2013; Tai et al., 2015). Several researchers aimed to learn sentiment shifters from sentiment-labeled texts, e.g. Ikeda et al. (2008), Noferesti and Shamsfard (2016).

In contrast to previous methods, our method does not require sentiment-labeled texts or feature engineering. It also does not rely on manually-composed lexicons of negators, propagators or dominators, but rather learns such lexicons automatically.

Kiritchenko and Mohammad (2016) wrote that *"lexicons that include sentiment associations for multi-word phrases as well as their constituent words can be very useful in studying sentiment composition"*. They manually developed such a lexicon for a few hundreds of bigrams and trigrams with opposing sentiments (each phrase has both negative and positive words), and experimented with both supervised and unsupervised methods for classifying and ranking the sentiment of these phrases. Their work has shown that rules based on part-of-speech patterns and unigram sentiment strengths are not sufficient for determining the sentiment of phrases with opposing sentiments. This motivated the current work, which is focused on learning lexical knowledge for sentiment composition.

Following Kiritchenko and Mohammad, we also build a sentiment lexicon for phrases and their constituent words, but our lexicon is built automatically, and is three orders of magnitude larger (over 250,000 bigrams). The size of the lexicon makes it suitable for learning lexical knowledge for sentiment composition.

[1] Available at `http://www.research.ibm.com/haifa/dept/vst/debating_data.shtml`

3 Method

Our method for learning sentiment composition lexicons comprises the following steps:

1. Train an n-gram sentiment classifier on a given sentiment lexicon for unigrams.
2. Use the sentiment classifier to automatically generate large sentiment lexicons of bigrams and unigrams.
3. Extract sentiment composition lexicons based on statistics from the bigram and unigram sentiment lexicons.

The rest of the section describes each of the above steps.

3.1 The Sentiment Classifier

Following previous work (Amir et al., 2015; Rothe et al., 2016; Bar-Haim et al., 2017b), we train a sentiment classifier on a sentiment lexicon, where the word's polarity is taken to be the label, and the word embedding is the feature vector. We use the publicly-available sentiment lexicon of Hu and Liu (2004) (hereafter, HL). After removing 224 multi-word expressions, the lexicon contains 6,565 words.

The above previous work focused on learning sentiment of unigrams, and optionally of multi-word expressions (conflated into single tokens), and used word2vec embeddings (Mikolov et al., 2013). In contrast, we are interested in learning sentiment composition from bigrams, and therefore we are mostly interested in compositional bigrams, and aim to construct a sentiment lexicon that contains hundreds of thousands of them. Due to bigram sparsity, learning their embeddings requires a very large corpus.

We use in this work a proprietary English corpus containing news articles and other types of publications from over 10,000 sources. The size of this corpus is in the order of 10^{11} tokens. While word2vec is known as a scalable method for deriving word embeddings, applying it to a corpus of this size is still computationally expensive.

We therefore opted for a more lightweight method for computing sentiment-oriented embeddings. Our method is inspired by the classical work of Turney and Littman (2003), who learned word sentiments based on their pointwise mutual information (PMI) with seed sentiment words. Our method represents each n-gram (unigram or bigram) as a 6,565 dimensional vector of its *Positive PMI (PPMI)* (Levy et al., 2015) with all the words in the HL lexicon. PPMI is defined for phrases u,v as

$$PPMI(u,v) = max(0, log\frac{f(u,v)\cdot N}{f(u)\cdot f(v)}), \quad (1)$$

where $f(u)$ and $f(v)$ are the number of sentences that contain u and v in the corpus, respectively, $f(u,v)$ is the number of sentences in which u and v co-occur, and N is the number of sentences in the corpus. u and v are said to co-occur if they are found in the same sentence, within a maximum distance of 10 tokens from each other, and have no overlap. We define $PPMI(u,u) = 0$ for any u.

We convert these sparse word vectors into dense embeddings as follows. First, we train a linear SVM[2] on the PPMI vectors of the words in HL, and apply it to a large lexicon of unigrams (to be described in the next section). We then select the 2,500 most positive words and 2,500 most negative words as predicted by the classifier (words in the HL lexicon were ignored). Based on the PPMI vectors of these 5,000 selected words, we learn a projection from the 6,565 dimensional space into a reduced space of 100 dimensions. Let M be a 5,000 $\times$ 6,565 matrix whose rows are the representations of the selected words. We compute the truncated Singular Value Decomposition (Deerwester et al., 1990) for M, $M_d = U_d \cdot \Sigma_d \cdot V_d^T$, with $d = 100$, following the notation of Levy et al. (2015). A PPMI row vector x for any new n-gram u (not in HL) can then be projected into the reduced space by taking the product $x \cdot V_d$. The resulting dense representation is much more compact, and is expected to work better for lower-frequency bigrams with sparse PPMI vectors.

[2]We used LIBSVM, (Fan et al., 2008).

#	Class	Condition UG1	Condition UG2	Predicted Bigram Polarity	
	Composition Classes				Sample match
1	REV$\oplus$	w	$\ominus$	$\oplus$	**combat** loneliness
2	REV$\ominus$	w	$\oplus$	$\ominus$	**lacked** courage
3	PROP$\oplus$	$w \wedge \ominus$	$\oplus$	$\oplus$	**overwhelming** success
4	PROP$\ominus$	$w \wedge \oplus$	$\ominus$	$\ominus$	**fresh** trouble
5	DOM$\oplus$	w		$\oplus$	**improving** nutrition
6	DOM$\ominus$	w		$\ominus$	**reckless** decision
	Adjective Classes				Sample match for (high,low)
7	ADJ(a_1,a_2)$\oplus\ominus$	$E(a_1)$	w	$\oplus$	high **morale**
8	ADJ(a_1,a_2)$\oplus\ominus$	$E(a_2)$	w	$\ominus$	low **morale**
9	ADJ(a_1,a_2)$\ominus\oplus$	$E(a_1)$	w	$\ominus$	higher **inflation**
10	ADJ(a_1,a_2)$\ominus\oplus$	$E(a_2)$	w	$\oplus$	lower **inflation**

Table 1: Definition of composition and adjective classes. Words in bold in the sample matches belong to the class.

The resulting 100-dimensional embeddings of the words in HL were used to retrain the SVM classifier. In order to assess the accuracy of the classifier, we performed a 100-fold cross validation experiment over HL words. For each fold, we removed from the training set words that have the same stem or lemma as any word in the test set. The classifier achieved high accuracy of 94.5%, confirming the effectiveness of our method.

3.2 Generating Sentiment Lexicons

Next, we created lexicons of unigrams and bigrams, computed their embeddings and applied the classifier to predict their sentiments. For the unigrams lexicon we selected words whose length is between 2 and 20 characters, are alphabetic and in WordNet (Miller, 1995), and their frequency in the corpus is at least 2000. This resulted in about 66,000 unigrams. For the bigram lexicon we selected bigrams that contain only unigrams from the unigram lexicon, and have minimum frequency of 250. The resulting lexicon of 2.8 million bigrams was still very noisy. In order to obtain a bigram lexicon of coherent phrases, we filtered out bigrams that the PMI between their unigrams is below 3.0. Finally, we only kept bigrams whose part-of-speech tag sequence form grammatical bigrams, e.g., *Adjective-Noun*, *Noun-Noun*, *Verb-Noun*, *Adverb-Adjective* etc. The resulting lexicon contains 262,555 bigrams. The classifier's real-valued predictions were normalized to the range of $[-1, 1]$. Unigrams that had sentiment in HL kept their original sentiment ($+1$ or -1).

3.3 Extracting Composition Lexicons

Finally, we define two types of classes that cover a variety of sentiment composition processes: *composition classes* model sentiment reversals and conflicting (opposing) sentiments; *adjective classes* determine the sentiment of phrases that contain specific gradable adjectives or their expansions. Our method learns a lexicon for each class, based on the unigram and bigram sentiment lexicons acquired in the previous step.

The classes are formally defined in Table 1. Each class definition has two parts: the *condition* for matching the class in a bigram, and the *predicted bigram polarity* (positive $\oplus$ or negative $\ominus$), in case the class is matched. The condition specifies in which unigram a word w from the class should be matched (the first unigram UG1 or the second unigram UG2), and optionally additional constraints on the other unigram. The table also shows a sample match for each class. For example, REV$\oplus$ (row 1 in the table) is a *Reverser*. Words in this class are matched in UG1, and if in addition UG2 is negative, the resulting bigram would be positive, e.g., ***combat*** *loneliness*. Similarly, REV$\ominus$ (row 2) flips the polarity from

positive to negative.

The other two types of composition classes are *Propagators (*PROP*)* and *Dominators (*DOM*)*. Propagators (rows 3-4) are sentiment words that, when followed by a word with opposite sentiment, "propagate" the sentiment of the other word to the bigram level, e.g., ***pure*** *vandalism*. Finally, dominators (rows 5-6) dictate the sentiment of the bigram, overriding any conflicting sentiment of the other unigram, if any.

Composition classes are extracted as follows. Let $C(c, w)$ be the set of bigrams in the bigram lexicon that satisfy the condition of class c for word w, and let $S(c)$ be the set of bigrams whose polarity is the one predicted by c. The precision of w with respect to c can be defined as:

$$P(c, w) = \frac{|S(c) \cap C(c, w)|}{|C(c, w)|} \tag{2}$$

The above formula gives uniform weights for all the bigrams in $C(c, w)$. However, we found that it is beneficial to also take into account the uncertainty stemming from the automatic sentiment prediction of the unigrams and bigrams. We do so by weighting each bigram according to strength of the predicted bigram sentiment, and for classes where the sentiment of UG2 is part of the match (REV and PROP), also according to the sentiment strength of UG2. Let $s(x)$ be the sentiment score of an n-gram x in the lexicon, and let wu be a bigram where UG1=w and UG2=u. The weighted formula is:

$$P(c, w) = \frac{\sum_{wu \in S(c) \cap C(c,w)} |s(u) \times s(wu)|}{\sum_{wu \in C(c,w)} |s(u) \times s(wu)|} \tag{3}$$

For DOM classes, we take $s(u) = 1$, since the sentiment of u is not part of the match. Weak sentiment of bigrams and unigrams (between -0.1 and 0.1 for unigrams, and between -0.05 and 0.08 for bigrams) is considered neutral sentiment.[3]

We include in c words w such that $P(c, w) > \alpha$ and $|C(c, w)| \geq k$. We use $|C(c, w)|$ as our confidence measure, by which we rank the words in each class. We took $(\alpha = 0.8, k = 10)$ for reversers and propagators, and $(\alpha = 0.95, k = 20)$ for dominators.[4]

Adjective classes are defined for specific pairs of opposing gradable adjectives (a_1, a_2) (rows 7-10). In this work we considered the pairs *(high, low)* and *(fast, slow)*. Since a_1 and a_2 are antonyms, we assume that if the bigram $a_1 w$ is positive, then the bigram $a_2 w$ is likely to be negative, and vice versa. We therefore learn two classes for each pair: ADJ(a_1,a_2)$\oplus\ominus$ contains words that are positive with a_1 and negative with a_2. ADJ(a_1,a_2)$\ominus\oplus$ contains words that are negative with a_1 and positive with a_2.

Adjective classes are learned as follows. We manually define a set of expansions $E(a)$ for each adjective a ($E(a)$ also includes a itself), e.g. *higher*, and *increasing* for *high*.[5] Let $B(a, w)$ be the set of all the bigrams in the lexicon such that UG1 is in $E(a)$ and UG2 is w. The score of a given word w with respect to the adjective pair (a_1,a_2) is defined as:

$$S_{(a_1,a_2)}(w) = \frac{\sum_{x \in B(a_1,w)} s(x)}{|B(a_1, w)|} - \frac{\sum_{x \in B(a_2,w)} s(x)}{|B(a_2, w)|} \tag{4}$$

We select for ADJ(a_1,a_2)$\oplus\ominus$ and ADJ(a_1,a_2)$\ominus\oplus$ words w with $S_{(a_1,a_2)}(w)$ above 0.1 or below -0.1, respectively, and use the absolute value of this score to rank the predictions.

4 Results

In this section we briefly describe the lexicons that were learned for each class by our method. Table 2 shows the number of words in each class. 2,783 words were learned in total. The biggest classes are the dominators, in spite of the higher thresholds that are applied to their results, compared to the reversers and the propagators. This happens because their matching condition is the weakest.

[3]The thresholds were determined heuristically by observing the resulting lexicons.

[4]α and k were selected heuristically by observing the resulting lexicons.

[5]The expansions we used are listed as part of the released dataset.

Class	Word Count
REV⊕	54
REV⊖	106
PROP⊕	48
PROP⊖	152
DOM⊕	666
DOM⊖	662
ADJ(*high,low*)⊕⊖	316
ADJ(*high,low*)⊖⊕	416
ADJ(*fast,slow*)⊕⊖	294
ADJ(*fast,slow*)⊖⊕	70
Total	2,783

Table 2: Number of words per class.

REV⊕		PROP⊕		DOM⊕	
less	wasteful	**critical**	acclaim	**unique**	culture
reduce	stress	**cloud**	solution	**beautiful**	gardens
reducing	inflammation	**proprietary**	insights	**wonderful**	atmosphere
preventing	violence	**complex**	ideas	**innovative**	approach
fewer	dropouts	**challenging**	endeavor	**digital**	capabilities
combat	inequality	**overwhelming**	victory	**excellent**	service
reduces	fraud	**invasive**	therapy	**strategic**	thinker
healthy	ageing	**intense**	commitment	**creative**	spirit
eliminate	racism	**strict**	compliance	**diverse**	environment
reduced	bureaucracy	**disruptive**	innovation	**good**	deeds
REV⊖		PROP⊖		DOM⊖	
poor	reputation	**pretty**	awful	**too**	often
too	powerful	**significant**	concern	**illegal**	actions
not	happy	**incredibly**	boring	**serious**	consequences
sexual	liberation	**strong**	disapproval	**severe**	problems
inadequate	protection	**powerful**	adversary	**alleged**	plot
lacked	commitment	**sharp**	pain	**allegedly**	abusive
bad	luck	**great**	danger	**poor**	sanitation
otherwise	decent	**fresh**	injury	**heavy**	debt
negative	contribution	**hot**	mess	**dangerous**	situation
insufficient	protection	**clearly**	drunk	**violent**	outbursts

Table 3: Top-ranked words in each composition class.

Table 3 shows the top-ranked words in each composition class. Each class word (in bold), is followed by a sample UG2 match for that word and the class in the bigram lexicon. For example, ***critical*** *acclaim* is a bigram in $C(\text{PROP}\oplus, critical)$. For the propagator classes, we only show words in HL. These words have the strongest sentiment and therefore are the most challenging for opposing sentiment resolution. In addition, in the experiment to be described in Section 6.2, we use HL as our sentiment lexicon, so only propagators in HL will be matched. Interestingly, the classes learned for REV⊕ and for REV⊖ are disjoint, indicating that the reversers operating on positive words are rather different from those operating on negative words.

Table 4 shows the top-ranked words for adjective classes. For example, *high morale* is positive, and *low morale* is negative, while for *unemployment* it is the other way around.

ADJ(*high,low*)⊕⊖	ADJ(*high,low*)⊖⊕	ADJ(*fast,slow*)⊕⊖	ADJ(*fast,slow*)⊖⊕
morale	unemployment	economic	turns
productivity	poverty	broadband	temper
ceilings	crime	implementation	spiral
literacy	costs	loading	escalation
standards	anxiety	response	buck
quality	noise	progress	spreading
ethical	violence	wit	spread
profitability	debt	learner	breathing
visibility	inequality	reaction	escalates
competitiveness	handedness	growth	population

Table 4: Top-ranked words in adjective classes.

5 Manual Asssessment

We conducted manual assessment of the precision of each class, by measuring the fraction of bigrams that have the polarity predicted by the class, out of the bigrams matched by the class. To this end, we selected from the bigram lexicon a random sample of bigrams matched by each class. Recall that some of the classes require positive or negative sentiment in UG1 and/or UG2. Our goal is to estimate the precision of the predicted bigram polarity, given that the matching is correct. In order to avoid incorrect sentiment matches caused by errors in our automatically-learned unigram sentiment lexicon, we restricted here the sentiment matches to words from the HL lexicon. We sampled 100 bigrams for each composition class, except for PROP⊕, which only had 59 matching bigrams[6]. In addition, we sampled 100 bigrams for each adjective pair (conflating the positive and negative classes for each pair).

Each of the resulting 759 bigrams was annotated by three in-house human annotators, who were asked to assess the bigram sentiment (positive/negative/neutral). They also had the option to indicate that the bigram is an incomplete phrase. Fleiss' kappa (Fleiss, 1971) for inter-annotator agreement was 0.61, indicating *substantial agreement* (Landis and Koch, 1977). The gold label for each bigram was obtained by taking the majority annotation. Ties were adjudicated by one of the authors (34 in total). We then removed the 59 bigrams assessed as incomplete, and computed precision for each class over the remaining 700 bigrams.

The results are summarized in Table 5. The precision achieved by the various classes seems very promising, in particular considering that the only sentiment-annotated input for learning these classes was a unigram sentiment lexicon. Error analysis reveals that the vast majority of misclassifications (84.6%) were labeled as neutral by the majority of the annotators, and in 25.1% of them, one annotator agreed with the class prediction, and the other two were neutral.

The learned classes provide a good starting point for semi-automatic construction of composition lexicons. We believe that their quality can be easily improved with manual filtering of our results, which can be done rather quickly.

6 Experiments

In this section we demonstrate the contribution of the learned composition and adjective classes to both phrase-level and sentence-level sentiment classification tasks. In line with our method for learning the classes, we focus in this section on sentiment analysis methods that do not rely on sentiment-labeled phrases or sentences for training.

6.1 Phrase Polarity Classification

In this experiment we test the extracted lexicons for composition and adjective classes and the bigram sentiment lexicon on the Opposing Polarity Phrases (OPP) lexicon released by Kiritchenko and Mo-

[6]Note that for this class we require here that UG1 is negative in HL and UG2 is positive in HL.

Class	Precision		
	⊕	⊖	Total
REV	0.79	0.78	0.78
PROP	0.75	0.81	0.79
DOM	0.71	0.79	0.75
ADJ(*high,low*)			0.68
ADJ(*fast,slow*)			0.75
All			0.76

Table 5: Precision assessment.

Exp	Sentiment Score	Accuracy (All)	Coverage	Accuracy (Covered)
c0	Most polar unigram	0.668		
c1	Composition	0.713	0.355	0.761
c2	Bigram score	0.726	0.309	0.863
c3	Bigram score or Composition	0.746	0.541	0.807

Table 6: OPP bigram binary classification accuracy and coverage results.

hammad (2016), which consists of phrases with words of opposing polarity. The dataset contains the sentiment score of each n-gram, the part-of-speech tags of the constituent unigrams, and the sentiment of all the unigrams included in the phrases, with a total of 307 bigram and 262 trigram phrases with a non-zero sentiment score. The phrases were collected from Twitter and the sentiment scores were obtained by manual annotations. The purpose of the OPP lexicon is to train and test sentiment composition models for opposing polarity phrases and therefore it is suitable for testing our classes and bigram sentiment prediction methods. Kiritchenko and Mohammad tested both supervised and unsupervised methods for sentiment classification and ranking. Here, we focus on unsupervised classification of bigrams.

Table 6 shows our experimental results. The baseline, c0, is the best-performing unsupervised method reported by Kiritchenko and Mohammad. This method determines the bigram polarity according to the most polar unigram score. Then, in experiment c1 (*Composition*), we predict the sentiment of the bigram by trying to match each of the classes in the bigram according to the following order: ADJ, REV, PROP, DOM. The order is based on the match specificity of the other word in the bigram (the non-class word): ADJ requires a match in a small set of adjective expansions, REV and PROP only require a certain polarity (positive/negative), and DOM does not restrict the other word. The first class that is applicable for the bigram is selected. For the matched class, we predict the bigram sentiment according to the rules in Table 1. In our second experiment c2, we use the predicted bigram score, and in c3 we use the two methods in cascade: if the bigram score is not available, we try the composition method of c1. In all three experiments, we back off to the most polar unigram score (c0) in cases that the test bigram is not covered by our predictions, so the accuracy results under the *Accuracy (All)* column are reported over the complete dataset. The *coverage* of each method is defined as the fraction of predicted bigrams out of all the bigrams in the dataset. Finally, *Accuracy (Covered)* is the accuracy only for the bigrams covered by each of the methods c1-c3.

The best performance is achieved by configuration c3, which utilizes both the bigram sentiment scores and the composition and adjective classes. By combining the two methods, we are able to address both idiomatic phrases (e.g. *top gun*) and compositional phrases (e.g. *great loss*) in the OPP dataset. c3 achieves an absolute improvement of 7.8% in accuracy with respect to the baseline c0, over the whole dataset. The differences between c1-c3 and the baseline c0 are all statistically significant, with $p < 0.05$ for c1 and $p < 0.01$ for c2 and c3, according to McNemar's test (Everitt, 1992). The results show that our sentiment and composition lexicons are effective even in a domain such as Twitter, which is rather different from the corpus we used for learning.

	Positive			Negative				
	Precision	Reccall	F_1	Precision	Recall	F_1	Accuracy	Coverage
Baseline	0.703	0.462	0.558	0.810	0.525	0.637	0.761	0.653
+ADJ	0.703	**0.480**	**0.571**	**0.815**	**0.530**	**0.642**	0.764	**0.665**
+REV	**0.724**	**0.487**	**0.582**	**0.822**	**0.541**	**0.653**	**0.778**	**0.665**
+PROP	**0.724**	**0.488**	**0.583**	**0.823**	**0.545**	**0.656**	**0.778**	**0.667**
+DOM	**0.720**	**0.547**	**0.622**	0.820	**0.546**	**0.655**	0.772	**0.708**

Table 7: Results from sentence polarity assessment. Bold indicates significant difference ($p < 0.05$) from baseline using approximate randomization test (Noreen, 1989).

	Sentence				Bigram		
	Match	$\uparrow$	$\downarrow$	$\leftrightarrow$	Match	$=$	$\neq$
+ADJ	59	27	9	23	63	52	11
+REV	40	31	4	5	40	35	5
+PROP	10	6	0	4	10	8	2
+DOM	903	164	132	607	1160	808	352

Table 8: Impact of class matches.

6.2 Sentence Polarity

We further assess the contribution of the composition and adjective classes to a lexicon-based sentiment classification of sentences.

The baseline system matches terms from the HL lexicon in the sentence. If the number of positive matches is greater than negative matches, we predict positive; if there are more negative matches, we predict negative; otherwise we predict neutral.

We then add the classes incrementally. If a class is matched in a bigram, it may modify the sentiment of that bigram. For example, suppose that the sentence contains the bigram *preventing cancer*. The baseline system will match *cancer* from the HL lexicon in the bigram, giving it negative sentiment of -1. Then the word *preventing* from the class REV$\oplus$ will be matched (line 1 in Table 1), since it is followed by a negative unigram. As a result, the sentiment of the bigram will become positive, and will get the score 1. The classes were added in the same order as in the previous experiment: adjectives, reversers, propagators, and dominators. We do not allow mutiple classes to match the same bigram.

For testing these lexicons we use the Claim Stance Dataset introduced by Bar-Haim et al. (2017a). This dataset contains Pro and Con claims found in Wikipedia for 55 controversial topics, and also includes labels for sentiment. We chose this dataset due to its diversity - it covers a variety of topics, and since it was also approached with a lexicon-based sentiment analysis by Bar-Haim et al. This dataset has 2,252 claim sentences that contain sentiment annotations (1012 positive and 1240 negative). We ignore sentiment found in the sentiment target that is given by the Claim Stance Dataset.

Table 7 shows results for the sentiment prediction with the baseline and after the addition of each class. We use precision, recall, and F_1 to measure performance for positive and negative classes and *accuracy* and *coverage* to measure overall performance, following previous work on this dataset. Bar-Haim et al. define *accuracy* as $\frac{\#correct}{\#predicted}$ and *coverage* as $\frac{\#predicted}{\#sentences}$. If the system returns a neutral label it is not counted as a prediction. There is consistent improvement over the baseline and each of the classes contributes to some improvement. F_1 for the positive class continues to improve with each additional lexicon. The case is similar for negative F_1 and *accuracy*, which improve up to DOM.

In Table 8 we look closer at how the classes are being matched and the impact this has on sentiment prediction. The left side of the table reports the number of sentences with class matches and breaks down how often they help ($\uparrow$), hurt ($\downarrow$), or stay the same ($\leftrightarrow$) compared to the previous lexicon (e.g., ADJ vs the baseline). We see that for the first three classes we match fewer bigrams but consistently help sentiment prediction. Adding the DOM lexicon helps more than it hurts but the numbers are more

balanced. We also see more cases where the prediction remains the same. This is because much of the dominant sentiment learned is consistent with the original lexicon. For example, matching the dominator *good* in the bigram *good decision*, while a correct match, will have no effect on the bigram polarity, since *good* was already matched from the HL lexicon as a unigram.

Without manual labels for all bigrams, another option for roughly checking the quality of the bigram matches is to check if the sentiment of the bigram is consistent with that of the sentence. This is shown in the right side of Table 8 with a report of the bigram matches. Multiple bigrams can match in a sentence so there may be more bigram matches than sentence matches. If the bigram is given the same sentiment as the sentence we count it as consistent ($=$), otherwise inconsistent ($\neq$). This does not strictly mean that the bigram polarity is correct or not, but it does give us a rough idea if the sentiment is correct and if it helps improve the sentence sentiment prediction. The results show that all the classes are consistent with the sentence sentiment in most of the cases. The dominators have far more matches but they are also much noisier than the other classes.

7 Conclusion

We presented a new approach for learning sentiment composition from a large, unlabeled corpus, which only requires a word-level sentiment lexicon for supervision. Our method learns lexicons that can address important sentiment composition phenomena such as sentiment reversals and mixed sentiment expressions. Manual assessment of the predictions made by these classes showed promising results. The classes were also shown to improve both phrase-level and sentence-level sentiment classification.

The automatically learned sentiment lexicons of bigrams and unigrams proved to be very useful for research on sentiment composition. We made them publicly available, and hope it will promote further research in this area. It would also be interesting to apply our method to languages other than English. Finally, we would like to investigate how much our results can be further improved in a semi-supervised setting, where the automatically-acquired classes are filtered by human annotators.

References

Silvio Amir, Ramón Astudillo, Wang Ling, Bruno Martins, Mario J. Silva, and Isabel Trancoso. 2015. Inesc-id: A regression model for large scale twitter sentiment lexicon induction. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 613–618, Denver, Colorado, June. Association for Computational Linguistics.

Roy Bar-Haim, Indrajit Bhattacharya, Francesco Dinuzzo, Amrita Saha, and Noam Slonim. 2017a. Stance classification of context-dependent claims. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 251–261, Valencia, Spain, April. Association for Computational Linguistics.

Roy Bar-Haim, Lilach Edelstein, Charles Jochim, and Noam Slonim. 2017b. Improving claim stance classification with lexical knowledge expansion and context utilization. In *Proceedings of the 4th Workshop on Argument Mining*, pages 32–38, Copenhagen, Denmark, September. Association for Computational Linguistics.

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 793–801, Honolulu, Hawaii, October. Association for Computational Linguistics.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

Li Dong, Furu Wei, Shujie Liu, Ming Zhou, and Ke Xu. 2015. A statistical parsing framework for sentiment classification. *Computational Linguistics*, 41(2):293–336, June.

Brian S. Everitt. 1992. *The analysis of contingency tables*. Chapman & Hall/CRC.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Daisuke Ikeda, Hiroya Takamura, Lev-Arie Ratinov, and Manabu Okumura. 2008. Learning to shift the polarity of words for sentiment classification. In *In Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP)*.

Alistair Kennedy and Diana Inkpen. 2006. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22:2006.

Svetlana Kiritchenko and Saif M. Mohammad. 2016. Sentiment composition of words with opposing polarities. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1102–1108, San Diego, California, June. Association for Computational Linguistics.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November.

Karo Moilanen and Stephen Pulman. 2007. Sentiment composition. In *Proceedings of RANLP 2007*, Borovets, Bulgaria.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 786–794, Los Angeles, California, June. Association for Computational Linguistics.

Alena Neviarouskaya, Helmut Prendinger, and Mitsuru Ishizuka. 2010. Recognition of affect, judgment, and appreciation in text. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 806–814, Beijing, China, August. Coling 2010 Organizing Committee.

Samira Noferesti and Mehrnoush Shamsfard. 2016. Using data mining techniques for sentiment shifter identification. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Eric W. Noreen. 1989. *Computer-intensive methods for testing hypotheses: An introduction*. Wiley.

Livia Polanyi and Annie Zaenen. 2004. Contextual valence shifters. In *Working Notes — Exploring Attitude and Affect in Text: Theories and Applications (AAAI Spring Symposium Series)*.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense word embeddings by orthogonal transformation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 767–777, San Diego, California, June. Association for Computational Linguistics.

Marc Schulder, Michael Wiegand, Josef Ruppenhofer, and Benjamin Roth. 2017. Towards bootstrapping a polarity shifter lexicon using linguistic features. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 624–633, Taipei, Taiwan, November. Asian Federation of Natural Language Processing.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October. Association for Computational Linguistics.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Comput. Linguist.*, 37(2):267–307, June.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July. Association for Computational Linguistics.

Peter D. Turney and Michael L. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Trans. Inf. Syst.*, 21(4):315–346, October.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 347–354, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Ainur Yessenalina and Claire Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 172–182, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Representations and Architectures in Neural Sentiment Analysis for Morphologically Rich Languages: A Case Study from Modern Hebrew

Adam Amram and **Anat Ben David** and **Reut Tsarfaty**
The Open University of Israel, University Road 1, Ra'anana
adam.amram@gmail.com, {anatbd,reutts}@openu.ac.il

Abstract

This paper empirically studies the effects of representation choices on *neural sentiment analysis* for Modern Hebrew, a *morphologically rich language* (MRL) for which no sentiment analyzer currently exists. We study two dimensions of representational choices: (i) the granularity of the input signal (token-based vs. morpheme-based), and (ii) the level of encoding of vocabulary items (string-based vs. character-based). We hypothesise that for MRLs, languages where multiple meaning-bearing elements may be carried by a single space-delimited token, these choices will have measurable effects on task perfromance, and that these effects may vary for different architectural designs: fully-connected, convolutional or recurrent. Specifically, we hypothesize that morpheme-based representations will have advantages in terms of their generalization capacity and task accuracy, due to their better OOV coverage. To empirically study these effects, we develop a new sentiment analysis benchmark for Hebrew, based on 12K social media comments, and provide two instances thereof: *token-based* and *morpheme-based*. Our experiments show that the effect of representational choices vary with architectural types. While fully-connected and convolutional networks slightly prefer token-based settings, RNNs benefit from a morpheme-based representation, in accord with the hypothesis that explicit morphological information may help generalize. Our endeavor also delivers the first state-of-the-art broad-coverage sentiment analyzer for Hebrew, with over 89% accuracy, alongside an established benchmark to further study the effects of linguistic representation choices on neural networks' task performance.

1 Introduction

Deep learning (Goodfellow et al., 2016) has seen a surge of interest in recent years, transforming all application domains of machine learning. In particular, *neural network* (NN) architectures currently dominate the development of models and applications in NLP, including syntactic parsing (Chen and Manning, 2014; Dyer et al., 2015; Durrett and Klein, 2015), sequence labeling (Grave, 2008), information extraction (Chen et al., 2015; dos Santos et al., 2015), text classification (Lai et al., 2015; Zhang et al., 2015) and sentiment analysis (Dos Santos and Gatti, 2014; Dong et al., 2014).

Much NN work in NLP has been conducted on English, which in turn raises the question whether these methods will be equally effective when applied to languages with different linguistic characteristics than English, and in particular, *Morphologically rich languages* (MRLs) (Tsarfaty et al., 2010). MRLs are languages where word structure holds substantial information, corresponding to multiple meaning-bearing units (morphemes) per space-delimited token. Furthermore, in MRLs word-order may be flexible. These properties may pose challenges to NN models which rely heavily on the *distributional* characteristics of the input tokens. While it is clear that *any* application of NN architectures to NLP may be non-trivial to design, and task performance may crucially depend on non-trivial architectural and modelling choices (Goldberg, 2015), we further ask: should these choices also be affected by the structure and properties of the language?

Proceedings of the 27th International Conference on Computational Linguistics, pages 2242–2252
Santa Fe, New Mexico, USA, August 20-26, 2018.

In this paper we attend to this question by empirically studying the effects of two dimensions of representational and modeling choices on *neural sentiment analysis* for Modern Hebrew — an MRL with interesting word-internal complexities and surface level ambiguity — and for which no sentiment analyzer currently exists.

We study two dimensions of representation choices: the first concerns the representation of the morphologically-rich input signal, *token-based* vs. *morpheme-based*, and the second concerns the level of encoding of the vocabulary items, which could be embedded as complete *strings* or as sequences of *characters*. We study these modeling choices for five architectures (a linear baseline and four NNs: fully-connected, convolutional, recurrent, and bi-directironal recurrent) to observe and analyze emerging trends in performance. In order to empirically evaluate the different models we develop a new and novel benchmark dataset for sentiment analysis in Hebrew, consisting of 12K user-generated comments on official Facebook pages of political figures. The sentiment of each comment has been manually coded by two trained human annotators, and the data themselves have been morphologically analysed and disambiguated, providing us with the opportunity to represent the input at different granularities.

Our experiments show that representation choices affect task performance, and these effects vary with the architectural type. While simple fully-connected and convolutional networks show advantages with the *token-based* representation, RNNs, in contrast, prefer *morpheme-based* settings and token-based bi-RNNs close much of the gap with the morphemes. This is the case for both the *string-based* and *char-based* encoding. We further show that, as in English, CNNs perform particularly well on Hebrew neural sentiment analysis, and conjecture that this is due to the tendency of CNNs to capture relatively large window sequences that are strong predictors, and these windows are, in turn, compatible with token-level granularity. Based on this investigation, we deliver the first sentiment analyser for Hebrew, with over 89% accuracy on analysing sentiment of user-generated comments, defining a new state-of-the-art for Hebrew NLP. A qualitative analysis of a sample of our best model's prediction errors shows that around a half of the remaining errors are also difficult for humans to classify, and require further work on integrating finer-grained aspects of the data.

Our contribution is then manifold. Firstly, we present empirical evidence to the varied effects that morphological information may have on different NN architectures. Secondly, we deliver a neural sentiment analyser for Hebrew, approaching 90% accuracy, on social media content. Finally, we provide a new manually annotated benchmark for Hebrew sentiment analysis, and a strong baseline for further investigation of the *task-representation-architecture* interplay and neural sentiment analysis in particular.

2 Task Description

Sentiment is a subjective attitude towards, for or against, a certain topic. The term *sentiment analysis* refers to the use of natural language processing and machine learning methods for the purpose of identifying or characterizing the sentiment expressed in a given piece of textual data (Pang and Lee, 2008). In this work we focus on sentiment analysis of social media content.

Due to their use by wide audiences, data extracted from social media are valuable sources for studying social, political and economic phenomena. One of the main motivations to analyze sentiment on the social web is *opinion mining*, an approach that challenges the traditional and dominant paradigms in political science, such as opinion polling, surveys or focus groups (Liu, 2012). Affect responses on social media have also been applied to study political behaviour (Ceron et al., 2014), determine organizational legitimacy (Etter et al., 2017), and even predict results of election campaigns (Tumasjan et al., 2010). While Twitter is a social media platform that lends itself naturally to sentiment analysis due to the short, informal character of tweets (Kouloumpis et al., 2011), various sentiment models analysed online conversations on other platforms (Paltoglou and Thelwall, 2012).

Open-source tools and sentiment analysis algorithms are available in several languages, including Arabic (Abdulla et al., 2013; Abdul-Mageed et al., 2014; Al Sallab et al., 2015), but as of yet no sentiment analysis tool is available for Hebrew. This represents a serious gap in the NLP technology available for Hebrew, and moreover, it places a significant barrier on the investigation of political situations through the unfolding conversations in the social web. Here we aim to leverage NN architectures for developing

a sentiment analysis tool for Modern Hebrew, in order to facilitate opinion mining in Israeli social media, and to expand the language technology available for the Hebrew-speaking research community. Since Twitter is not widely used for political discussion in Israel, we develop a new benchmark based on user comments to politicians' pages on Facebook, which is widely used in Israel by politicians and the public.

3 Data Representation

Previous work on *sentiment analysis* in general and on *neural sentiment analysis* in particular focused mainly on English data. Here, we consider Modern Hebrew, a Semitic language with very different characteristics than English. In particular, Hebrew is a *morphologically rich language* (MRL), of which word structure is internally complex and word order is quite flexible. How might these properties affect our modeling decisions when developing a NN-based sentiment analyzer for Hebrew?

Let us begin by theoretically considering the notion of an input token in Hebrew. Because of its rich morphology, any single space-delimited token in Hebrew may contain, in addition to its lexical content, functional clitics (prefixes and suffixes) that correspond to independent stand-alone words in English. To illustrate, the word "*wkftmkti*"[1] may be morphologically segmented into "*w*" (and) "*kf*" (when) "*tmkti*" (supported+1person), and be translated to "and when I supported". Without segmenting the raw tokens into these morphological units, the positive affinity of "*tmkti*" (supported+1person) may be lost on the model, which may have observed "*tmkti*" as a standalone token elsewhere, but not in this composition.

This situation is further complicated by the fact that Hebrew surface tokens are highly ambiguous. Due to the rich morphology and the lack of diacritics in standard textual data, a Hebrew space-delimited token may admit multiple different morphological analyses, only one of which is relevant in context (Tsarfaty, 2006). For example, the token "*frch*" may be a-priory analyzed as the simple verb "*frch*" (infested+3person) or as the sequences "*f*" (that) + "*rch*" (ran+3person+feminine) or "*f*" (that) + "*rch*" (wanted+3person+masculine). The latter analysis has a stronger affinity with positive sentiment than the others. This means that improper morphological disambiguation, or lack thereof, may undermine sentiment accuracy scores.

When constructing a vocabulary and defining the alphabet on which the NN will operate, we need to make representation choices that would be suitable for these properties of MRLs. In this paper we consider two dimensions of representational choices:

- **Input Items:** *Token-Based vs. Morpheme-Based.* First, we compare NN models trained on a signal consisting of the *raw tokens* with ones trained on sequences of *morphological segments* that represent standalone lexical and functional units. The hypothesis is that models trained on morphologically segmented input will provide better generalization capacity for the network and will thus improve the prediction accuracy.

- **Vocabulary Encoding:** *String-Based vs. Char-Based.* It has been proposed in previous work (Ling et al., 2015; Ballesteros et al., 2015) that combating the complexity of word structure in NN architectures may benefit from encoding the vocabulary using sub-words or character sequences. We thus contrast models that encode vocabulary items as complete strings to ones that encode a vocabulary of characters that are used to construct each of the strings.

The empirical investigation we conduct aims to empirically answer two questions: (i) would the choice of representation affect prediction accuracy? and, (ii) would the different representation choices affect different neural network architectures *in different ways*?

4 Data Preparation

In order to empirically evaluate task performance for the different representational choices and architectural design, we develop a new Hebrew benchmark for sentiment analysis and provide these annotated data in two different representational forms: a *token-based* and a *morpheme-based* version.

[1] We assume the transliteration of Sima'an et al. (2001).

Positive Sentiment	Negative Sentiment
"Ruvi, well done! Keep following your predecessor. Talk less and do more. You are the right person in the right timing."	"We should revenge the entire village of the kidnappers. Without fear. A government of cowards."
"Mr. President, you radiate peacefulness, humanism and security, and I hope this will soon be the case. Bless you. Mr. President, your vigorous actions create waves of connection and healing. I thank you for being my president."	"Shame on you! We don't care about your opinion! Constitutionally, the president should be a-political and your entire stupid post is wrapped with righteous politics. Shame on you and may this soon happen to your granddaughters".

Table 1: Inter-Annotator Agreement on the Rivlin's Annotated Corpus

Coder 1, Coder 2	N cases	Example	Explanation
Positive, Negative	26	"Terrible. May he rest in peace."	Mixed sentiment
Negative, Positive	14	"I am dreaming about peace and you are dreaming about football! A wasted dream!"	Potentially sarcastic
Neutral, Negative	2	"Daniel, this is not the time to generalize people despite of what you think of him. He's a Jew just like you and me."	Internal discussion among commenters
Neutral, Positive	2	"Mr. President, how can I arrange an appointment with you? I have been cherishing you for many years, since eleventh grade."	Off-topic
Positive, Neutral	4	"Mr. President, I would like to ask you to help the communities surrounding the Gaza Strip and to provide them with adequate housing and educational services for their children in safe places until genuine calm. This is not a matter of abandonment and Hamas propaganda is baseless."	Off-topic
Negative, Neutral	2	"Why not the length and breadth of it, Mr. President"	Potentially sarcastic

Table 2: Qualitative analysis of a sample of 50 comments for which there was no inter-rater agreement

Our data set consists of 12,804 user comments to posts on the official Facebook page of Israel's president, Mr. Reuven Rivlin. In October 2015, we used the open software application Netvizz (Rieder, 2013) to scrape all the comments to all of the president's posts in the period of June – August 2014, the first three months of Rivlin's presidency.[2] While the president's posts aimed at reconciling tensions and called for tolerance and empathy, the sentiment expressed in the comments to the president's posts was polarized between citizens who warmly thanked the president, and citizens that fiercely critiqued his policy. Of the 12,804 comments, 370 are neutral; 8,512 are positive, 3,922 negative.

We use a morphosyntactic parser called *yap* (*yet another parser*) (More and Tsarfaty, 2016) to morphologically analyze, disambiguate and segment all comments in our dataset. In the token-based representation, there is an average of 23 tokens per comment, and in the morpheme-based representation an average of 31 morphemes per comment. In terms of *out-of-vocabulary* (OOV) items, we observe a higher OOV rate in the token-based test set, 8.865% (2552 out of 28787), compared to the OOV rate 7.624% (1442 out of 18912) with the respective morpheme-based representation of the same set, as expected. So, at least in theory, a morpheme-based representation may better generalize from training instances, at least in cases where the composition of seen morphemes yields unseen tokens.

Data Annotation: A trained researcher examined each comment and determined its sentiment value, where comments with an overall positive sentiment were assigned the value 1, comments with an overall negative sentiment were assigned the value -1, and comments that are off-topic to the post's content were assigned the value 0. We validated the coding scheme by asking a second trained researcher to code the same data. There was substantial agreement between raters (N of agreements: 10623, N of disagreements: 2105, Coehn's Kappa = 0.697, p = 0).

Table 1 shows examples of positive and negative sentiment in clear cases of agreement between human annotators. We further examined the source of disagreements between the annotators. In a qualitative analysis of a sample of 50 comments for which there was disagreement between raters, summarized in

[2]It should be noted that the period of data collection was politically and socially charged. This has been due to a series of violent events that escalated fragile tensions with regards to Jewish-Arab relations in the region, including the kidnapping and murder of three teenagers living in the settlements of the West Bank, a revenge murder of an Arab teenager in East Jerusalem, a two-months war in Gaza, and a controversial wedding between a Jewish woman and an Arab man which sparked demonstrations led by extremist Jewish groups.

Table 2, we observe that the majority of the disagreement cases originated in comments that contained mixed sentiments, such as in the following:

- "Mr. President, every person has the right to decide on their lives, but converting to Islam in these crazy days, which is what her partner wanted, means that their offsprings will not be Jewish. You are talking about racism but they want to destroy us because we are Jews. Mr. President I wish you success, health and joy".

- "The hotheads, the enemy inside. The real enemy 'thanks' to whom we were scattered in all directions in the era of the Temple, the real enemy that might put us all in the grave".

In the first example, the commenter expresses a positive and respectful sentiment towards the president, but displays a negative sentiment towards conversion to Islam. In the second example, the sentiment can be interpreted in both directions. The content of the comment expresses negative sentiment towards the 'hot heads', but reading in also the context on the president's post, it actually expresses positive sentiment towards the idea that tolerance is better than hatred. It appears then that the unrestricted length and form of expression in Facebook comments makes the overall sentiment classification task more challenging.

Data Normalization and Two-Level Representation: We excluded from the dataset comments that do not contain any word in Hebrew. We also added spaces to separate Hebrew (or English) tokens, numbers, and punctuation symbols from one another in the text sequence. We then fed the raw tokens in our dataset to a morphological analysis and disambiguation parser (More and Tsarfaty, 2016), to build the respective morpheme-based representation. We split our dataset into a *train set* (80%) and a *test set* (20%) of sizes 10,244 and 2,560 comments, respectively. Out of the comments in the *train set*, 20% are reserved as *dev set* for evaluation and optimization after each epoch.

5 Neural Network Architectures for Sentiment Analysis

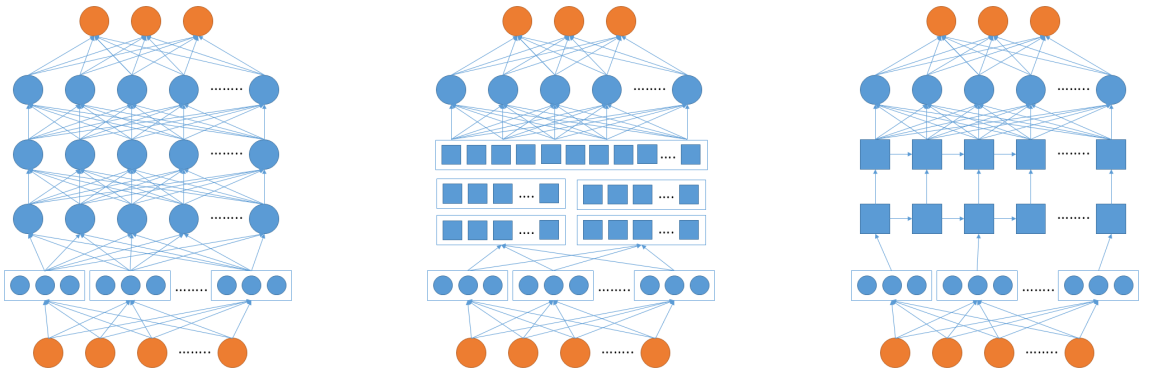
For each of the representation choices outlined above, we set out to compare and contrast the performance of different architectures. In all cases, we learn from our data a sentiment analysis function $f : x \rightarrow y$, where $x \in \mathcal{X}$ is a textual sequence which may be represented and encoded in the different ways discussed in Section 3, and $y \in \{$*positive, neutral, negative*$\}$ is a three-way classification of sentiment polarity.

We compare traditional linear models with non-linear models, contrasting different choices of NN architectures. Simple feed-forward fully-connected networks, such as the *multi-layer perceptron* (MLP), accept a sequence of items as input, which are then treated as features of the model (a *bag of words* (BOW) approach). They are simple to conceptualise and construct, however, the BOW representation, while highly sensitive to lexical content, is insensitive to any sequencing or ordering preferences. So, "The soup was not good, it was bad" may score the same as "The soup was not bad, it was good".

To capture some ordering information, it is customary to employ a feed-forward convolutional neural network (CNN) with a pooling layer. CNN architectures apply a non-linear function on a sliding window over words in the comment, and transform it to channel size vector. Then, a "pooling" operation combines the different vectors into a single channel size vector, that is in turn used for prediction. CNNs were previously shown to provide excellent performance on sentiment analysis of English (dos Santos et al., 2015). Would it be possible to replicate these achievements for Hebrew?

Ordering information may alternatively be captured by changing the network architecture to explicitely encode a sequence: this can be done using *recursive* or *recurrant neural networks* (RNNs). Indeed, various English sentiment analyzers assume RNNs at their backbone (Dong et al., 2014). For sentiment analysis we assume an RNN architecture, where the intermediate states throughout the sequence are used for prediction, loss calculation, and back propagation to update previous states.

In this work we consider a concrete implementation of RNNs — *Long Short-Term Memory* (LSTM) networks (Hochreiter and Schmidhuber, 1997) — which preserves gradients over time using functions that simulate logical gates (memory cells). At each input state, a gate is used to decide how much of the new input should be written to the memory cell, and how much of the current content of the memory



(a) The MLP Architecture (b) The CNN Architecture (c) The RNN Architecture

Figure 1: The Neural Network architectures in our experiments. All architectures started with an input vector of word/char index followed by an embedding layer with embedding size 300. In (a) we used 3 fully-connected layers, the first with 256 units, the second with 128 units and the third with 64 units. Each fully-connected layer has a dropout rate of 0.5. (b) uses 2 parallel convolution layers with 128 filters and kernel size of 3, 8 for *string-based* and 10, 30 for *char-based*. After the convolution, we apply max pooling with pooling size of 2. We then concatenate the results from previous layers and feed it to a fully-connected layer with 128 units and 0.5 dropout rate. (c) uses 2 layers of specific RNN implementations, LSTM or BiLSTM, followed by a fully-connected layer with 128 units and 0.5 dropout.

should be forgotten. Three gates are controlling for input, forget and output. Gate values are computed based on linear combinations of the current input and the previous state.

We also consider *Bidirectional-RNNs*, and in particular, a *Bidirectional-LSTM* (BiLSTM), which is a variant of LSTMs that has been shown to perform well on various sequence labeling and transduction tasks. BiLSTMs maintain two separate states for each input position, one forward and one backward. The forward and backward states are generated by two different LSTMs. The first LSTM is fed the input sequence as is, while the second LSTM is fed the input sequence in reverse. For morphologically rich sequences as discussed here, BiLSTMs may potentially capture the influence of prefixes and suffixes on upcoming content as well content that precedes them. At least in theory, such effects may help leverage morphological phenomena for classification.

6 Experiments

We set out to evaluate the effects of different input representation choices (Sections 3) and different architectural choices (Section 5) on neural sentiment analysis for Modern Hebrew. All of our models aim to learn the same objective function $f : x \rightarrow y$ where $x \in \Sigma^*$ is a sequence of vocabulary items over an alphabet Σ, encoded in the different ways detailed in Section 3. The output is a sentiment value $y \in \{positive, neutral, negative\}$.

We compare a traditional **Linear** basedline model to four different NN-based models: **MLP, CNN, LSTM** and **BiLSTM**. We implemented our models using Keras (Chollet, 2015) a high-level API with TensorFlow (Abadi et al., 2015) running as the backend engine. All models start with input represented by a vector of word/char index which is fed into an embedding layer, followed by the specific model layers (e.g. convolutional, RNN, etc.), and conclude with a softmax activation for the output. All of our NN models' architectures are depicted in Figure 1.

The **Linear** model contains a single fully-connected layer with 100 units and without activation. This is the only model that did not include an embedding layer and dropout regularization. The **MLP** model contains 3 fully-connected feed-forward layers, the first with 256 units, the second with 128 units and the third with 64 units. We apply dropout with rate of 0.5 on each fully-connected layer. Our **CNN** (ConvNet) contains two convolutional layers, one with kernel size 3 and the other with kernel size 8. After each convolutional layer there is a max pooling layer. These layers are followed by a fully-connected layer

(a) *String-based Vocabulary*					
Architecture: *Input Representation:*	**Linear**	**MLP**	**CNN**	**LSTM**	**BiLSTM**
Token-Based	68.20	**86.80**	**89.20**	85.20	87.40
Morpheme-Based	66.13	86.40	89.06	**86.20**	**87.50**
(b) *Char-based Vocabulary*					
Architecture: *Input Representation:*	**Linear**	**MLP**	**CNN**	**LSTM**	**BiLSTM**
Token-Based	**69.38**	74.60	82.40	69.50	73.70
Morpheme-Based	68.71	74.50	78.55	70.60	73.10

Table 3: Accuracy results (percentage of correct label predictions) for all architecture and representation choices on our test set; for (a) the *string-based* vocabulary, and (b) the *char-based* vocabulary (b).

with 256 units. For *char-based* we changed the kernel sizes to 10 and 30.[3] Our **RNN/BiRNN** Recurrent Neural Network architectures contain two specific RNN implementations (LSTM or BiLSTM) with 93 units followed by 1 fully connected layer with 256 units and 0.5 dropout rate.

All models use the cross-entropy loss for training and Adam optimizer (Kingma and Ba, 2014) to update models weights and biases. We set a learning rate of $1e-4$, and use batch size of 50 for training.

We evaluate two alternatives for representing the input: a *token-based* representation, where raw tokens are passed on as they are, and a *morpheme-based* representation, where the input signal is first morphologically disambiguated and is passed on as a sequence of morphological segments. From the train dataset we created a vocabulary Σ with 5K top items (tokens/morphemes). Then, we considered two alternatives of encoding the vocabulary items. A *string-based* encoding, where each of the tokens/morphemes is represented as a single item in the vocabulary, and a *character-based* representation assuming a narrower vocabulary, consisting of alphanumeric characters and special signals, and where each item (token or morpheme) is represented as a sequence of characters. The character set used in the vocabulary of our char-based models consists of 220 characters, including Hebrew letters, digits, and other characters such as white space and a new line character.

We set the comment length limit to 100 items (tokens or morphemes) in the *string-based* encoding and to 300 for *char-based* encoding. If a sequence was less than the limit, we padded it with the padding value, and if it is greater we trimmed it.

Results: Table 3(a) presents the results of the empirical comparison between our five architectures, in *token-based* vs. *morpheme-based* settings, for the *string-based* encoding of vocabulary items. When contrasting *token-based* and *morpheme-based* NN architectures in *string-based encoding*, the most striking outcome is that representation choices do have an effect on task performance, and that the difference in task performance varies across architectures.

First of all, we see that linear models, assuming a naïve classification architecture based on unigram features, perform roughly at the same level as a "majority vote" baseline, i.e., a classifier that would simply predict "positive" for all cases. In our data, "positive" constitutes 66.6% of the gold judgments. Yet, we observe a clear empirical advantage to the *token-based* representation over *morpheme-based* in this setting, perhaps since tokens allow more context to enter the classification — these tokens may essentially be seen as n-grams of morphological segments. Furthermore, in feed-forward networks, both fully-connected and convolutional, there is still an advantage to the token-based representation. We conjecture that this is due to the inherent order-insensitivity of these models. Morphological segments out of context may be less informative, and tokens capture more context of particular morphemes, where these larger "chunks" may exhibit particular ngrams that are strong predictors.

RNNs, in contrast, prefer the morpheme-based setting, and in the LSTM this is in a non-negligible margin. This is in line with our hypothesis that explicit modeling of morphology may capture interactions of elements in the sequence and allow for better generalization thereof. In contrast with MLPs and CNNs, RNNs consider the entire sequence for classification, and so it may be the case that previously unseen tokens in the sequence may undermine classification in token-based settings. This RNN representation

[3]This design outperformed CNNs with smaller (3,4,5) and larger (5,10,15) kernel sizes in our preliminary experiments.

can benefit from the more compact vocabulary of the morpheme-based models, and may also benefit from the decomposition of unknown tokens into potentially known morphemes to better generalize from seen sequences to unseen ones. That said, it is to note that token-based BiLSTM closes much of the gap with the morpheme-based setting, which suggests that the bidirectional representation of tokens perhaps already implicitly captures some aspects of tokens' internal decomposition and morphological signature.

The best result across all models is obtained with the token-based CNN. As in English, CNNs show excellent performance for this task, providing state of the art results for Hebrew sentiment analysis, above 89% accuracy. In particular, this is obtained in the token-based settings, where tokens are compatible with the "n-gram" views filtered through the convolution.

Table 3(b) presents the results of the same comparisons, for *char-based* vocabulary encoding. Interestingly, for the linear model the char-based representation outperforms the string-based counterpart, but these models still lag far behind the rest of the neural networks. With simple and convolutional feed-forward networks we continue to see that token-based representations outperform. Yet, in all cases, we see better scores for the parallel models in the *string-based* setting. RNNs are inferior to all CNNs and MLPs, still presenting an advantage to the morpheme-based representation, and almost no difference in the BiLSTM experiment. It seems that while characters may be able to capture some (morphological) regularities inside tokens, they are not very good at capturing the content of long character sequences.

For our best model overall, a token-based CNN in the string-based lexicon encoding, we compared the automatic prediction on a sample of 50 examples with the manually annotated sentiment. 26/50 (52%) of the mis-classifications are in cases when the human raters also happen to disagree, mostly due to mixed sentiment. This suggests that further improvement will require more sophisticated, aspect-based modeling, on finer-grained sentiment targets.

The main message coming out of this investigation is that the representation of linguistic items (tokens, words, morphemes) may influence the performance of NN-based architectures in the case of MRLs, and that the extent of the difference in performance depends on the type of architecture. It should be noted though that this outcome is — quite possibly — task-dependent. Sentiment analysis presents a three-way classification over complete sequences, for which long ngrams often seem to serve as good predictors. It may well be that for sequence transduction (as in SMT) or structure prediction (as in parsing) we will observe different trends in the influence of representation types on different languages and architectures.

Error Analysis: To gain further insight into the difference in task performance between different models and architectures, we performed a qualitative error analysis for 6 of our best models: the **MLP**, **CNN**, **LSTM** architecture, in *token-based* and *morpheme-based* settings, for the *string-based* lexicon.[4]

In general, all models are better at identifying positive sentiment than in identifying negative sentiment. The cases where both morpheme-based and token-based models correctly identified the *positive* sentiment constitute 91% of the *positive* comments in the MLP case, 93.8% in the CNN, and 94.1% in the LSTM. In contrast, morpheme-based and token-based models correctly identified *negative* sentiment only 27.8% of the *negative* comments in the MLP, 30.1% in CNN, and 48.1% in the LSTM. Cases where morpheme-based and token-based models wrongly identified a positive sentiment as negative one have different characteristics based on the architecture. In the MLP these are characterized by a double negation, and by addressing a third party. For CNN, comments that start with "Sorry, but.." are identified as negative, even when these happen to be positive. For the LSTMs, we see error in the prediction of positive sentiment in the case of extremely long comments, i.e., accuracy here is length sensitive.

We now turn to consider cases where the token-based models made the right prediction and morpheme-based models made the opposite prediction. The morpheme-based MLP wrongly identified a positive sentiment as a negative one in the case of short comments, many punctuations, and many negation elements. The CNN wrong classification of this type is characterized by a double negation, and also by addressing multiple parties in the same comment. For the RNN, this wrong prediction is again characterized by length — at moderate length of 12 morphemes the model often doesn't succeed in making the correct prediction. The opposite case, where morpheme-based models identify negative sentiment as a positive one has a different characterization in these architectures: in the MLP this is characterized

[4]For the complete report, confusion matrix, error characterization and translated examples, see our supplementary materials.

	MLP	CNN	LSTM
MB,TB-correct,pos	60.39 %	62.07%	62.4%
MB,TB-correct,neg	23.25%	24.14%	14.08%
MB,TB-should be pos	3.59% double negation	2.34% "sorry, but.."	2.12% very long comments
MB,TB-should be neg	4.2% honorary, criticism, !!!	3.125% sarcasm, irony	7.185% address+mixed modifiers
TB-pos MB-error,neg	1.09% short, honoraries+negations	0.39% double negation, third party	1.4% short-moderate length
TB-neg MB-error,pos	1.95% honorary title+negative words	2.3% mixed sentiments	0.93% honorary title+opposing words
MB-pos TB-error,neg	1.25% third party, neg words	1.52% direct address / "advise"	1.09 sequential short utterances
MB-neg TB-error,pos	1.36% third party ref	1.093% repetitions, moderate length	7.92% very short comments

Table 4: Qualitative Error Analysis and Confusion Quantitative Assessment on 2560 comments. TB denotes *token-based* and MB denote *morpheme-based* representation. We do not discuss neutral sentiment.

by kind and respectful words towards the president, alongside expressed criticism. For the CNN we see confused classification in comments that express mixed sentiments towards different topics. For the LSTM, we see the same pattern as the MLP; respectful attitude towards the president (Dear Mr. president, respected president) followed by an opinion which is opposed to the post's actual content.

Finally, cases where morpheme-based models made the right prediction and token-based models were erroneous, i.e., the cases that coincide with our linguistic hypothesis, can be characterized as follows. For the MLP, positive sentiments where identified as negative where the comments address a third party and also contain certain negation elements (i.e., "I choose Rivlin despite what Liberman says"). CNNs classified positive sentiments as negative where the comments directly address the president, proposing a sort of explicit "advice" ("Dear Mr. president we need to stop the violance"). The LSTM gets confused in cases of long comments consisting a sequence of very short utterances, typically also containing many negations and punctuation ("Me too. I'm opposed. Where do we go?" etc.) The opposite error also happens in the token-based models, where morpheme-based are correct. MLPs classify negative as positive where the comments do not address the president, but speak of a third party (the children, the soldiers, etc). Token-based CNNs mistakenly classify negative as positive in comments of moderate length (about 14 words) or those that have many repetitions ("sit down sit down sit down quietly and listen.."). The token-based LSTM is wrong in cases of *very* short comments, with only a handful of tokens.

All in all morpheme-based models do show advantages in identifying the correct sentiment of actual content (addressing of third parties, serving advise, etc.) beyond fixed expressions, and are better in classifying short texts or texts with repetitions. However it seems that the morpheme-based models do get more easily confused by the existence of negation elements, double negation, and many titles honoring the speaker (the president). These may steer the classification in a fairly rigid direction. Token-based models that consider larger n-grams, in particular in the context of CNN architectures, capture larger context windows, which currently present the best performance on Hebrew sentiment analysis.[5]

7 Discussion and Conclusion

Despite the great success of NN-based methods in NLP, the question of the interplay between *language type* and *architectural choices* has only been scarcely attended to, if at all. Yin et al. (2017), for instance, compare the strengths of different NN architectures for NLP tasks, but the discussion focuses solely on English. Dos Santos and Zadrozny (2014) propose char-based CNNs for NLP, but the benefits of the char-based modeling are not contrasted for different NN architectures and modeling. Finally, Al Sallab et al. (2015) contrast deep learning architectures for sentiment analysis in Arabic. They use a simple *bag of words* (BOW) approach without considering the MRL nature of the language, nor they are comparing different representation choices for their lexicon.

In this work we address neural sentiment analysis of Hebrew, a Semitic language known for its morphological complexities, and evaluate two choices of representing the input signal: (i) tokens vs. morphemes input, and (ii) string-based vs. char-based encoding. We experiment with different architectures: a linear model, feed-forward MLPs, CNNs, and (Bi)RNNs. We show that while linear models as

[5]Our data set, code, and models are publicly available at `https://github.com/omilab/Neural-Sentiment-Analyzer-for-Modern-Hebrew`.

well as MLPs and CNNs obtain higher accuracy at token-level granularity, RNNs prefer the morpheme-based settings. We further show that while char-based representation can in most cases dispense with morpheme-level information, it comes at a cost of an overall drop in accuracy. Our best model is a token-based CNN with above 89% on a new Hebrew benchmark based on social media content.

We believe that this investigation is only a first step in un-black-boxing the use of NN models for language processing tasks. Through this investigation, the two-level sentiment benchmark we deliver, and the strong baseline results we provide, we hope to encourage further investigation into the interplay between *task, language types* and *modeling choices* in general, and on NN models for MRLs in particular.

Acknowledgments

We thank Tzipy Lazar-Shoef for research assistance, and are thankful to three anonymous reviewers for their insightful comments. This research is funded by the Israel Science Foundation, ISF grant 1739/26, for which we are grateful.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Muhammad Abdul-Mageed, Mona Diab, and Sandra Kübler. 2014. Samar: Subjectivity and sentiment analysis for arabic social media. *Computer Speech & Language*, 28(1):20–37.

N. A. Abdulla, N. A. Ahmed, M. A. Shehab, and M. Al-Ayyoub. 2013. Arabic sentiment analysis: Lexicon-based and corpus-based. In *2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, pages 1–6, Dec.

Ahmad A Al Sallab, Ramy Baly, Gilbert Badaro, Hazem Hajj, Wassim El Hajj, and Khaled B Shaban. 2015. Deep learning models for sentiment analysis in arabic. In *ANLP Workshop 2015*, page 9.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. *CoRR*.

Andrea Ceron, Luigi Curini, Stefano M Iacus, and Giuseppe Porro. 2014. Every tweet counts? how sentiment analysis of social media can improve our knowledge of citizens' political preferences with an application to italy and france. *New Media & Society*, 16(2):340–358.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October. Association for Computational Linguistics.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL*, pages 167–176. The Association for Computer Linguistics.

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 49–54, Baltimore, Maryland, June. Association for Computational Linguistics.

Cícero Nogueira Dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78.

Cícero Nogueira Dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1818–II–1826. JMLR.org.

Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. *CoRR*, abs/1504.06580.

Greg Durrett and Dan Klein. 2015. Neural CRF parsing. *CoRR*, abs/1507.03641.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *CoRR*, abs/1505.08075.

Michael Etter, Elanor Colleoni, Laura Illia, Katia Meggiorin, and Antonino D'Eugenio. 2017. Measuring organizational legitimacy in social media: Assessing citizens' judgments with sentiment analysis. *Business & Society*, page 0007650316683926.

Yoav Goldberg. 2015. A primer on neural network models for natural language processing. *CoRR*, abs/1510.00726.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Deep learning. Book in preparation for MIT Press.

Alex Grave. 2008. *Supervised Sequence Labelling with Recurrent Neural Networks*. Ph.D. thesis.

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

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

Efthymios Kouloumpis, Theresa Wilson, and Johanna D Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! *Icwsm*, 11(538-541):164.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2267–2273. AAAI Press.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernández Astudillo, Silvio Amir, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. *CoRR*, abs/1508.02096.

Bing Liu. 2012. Sentiment analysis and opinion mining.

Amir More and Reut Tsarfaty. 2016. Data-driven morphological analysis and disambiguation for morphologically rich languages and universal dependencies. In *Proceedings of COLING 2016*, december.

Georgios Paltoglou and Mike Thelwall. 2012. Twitter, myspace, digg: Unsupervised sentiment analysis in social media. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(4):66.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, January.

Bernhard Rieder. 2013. Studying facebook via data extraction: the netvizz application. In *Proceedings of the 5th annual ACM web science conference*, pages 346–355. ACM.

Khalil Sima'an, Alon Itai, Yoad Winter, Alon Altman, and Noa Nativ. 2001. Building a tree-bank of modern hebrew text.

Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kübler, Yannick Versley, Marie Candito, Jennifer Foster, Ines Rehbein, and Lamia Tounsi. 2010. Statistical parsing of morphologically rich languages (spmrl) what, how and whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–12, Los Angeles, CA, USA, June. Association for Computational Linguistics.

Reut Tsarfaty. 2006. The interplay of syntax and morphology in building parsing models for modern hebrew. In *Proceedings of ESSLLI*.

Andranik Tumasjan, Timm Oliver Sprenger, Philipp G Sandner, and Isabell M Welpe. 2010. Predicting elections with twitter: What 140 characters reveal about political sentiment. *ICWSM*, 10(1):178–185.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of CNN and RNN for natural language processing. *CoRR*, abs/1702.01923.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *CoRR*, abs/1509.01626.

Scoring and Classifying Implicit Positive Interpretations: A Challenge of Class Imbalance

Chantal van Son, Roser Morante, Lora Aroyo, Piek Vossen
Vrije Universiteit Amsterdam
De Boelelaan 1105, 1081 HV Amsterdam
The Netherlands
{c.m.van.son, r.morantevallejo, lora.aroyo, piek.vossen}@vu.nl

Abstract

This paper reports on a reimplementation of a system on detecting implicit positive meaning from negated statements. In the original regression experiment, different positive interpretations per negation are scored according to their likelihood. We convert the scores to classes and report our results on both the regression and classification tasks. We show that a baseline taking the mean score or most frequent class is hard to beat because of class imbalance in the dataset. Our error analysis indicates that an approach that takes the information structure into account (i.e. which information is new or contrastive) may be promising, which requires looking beyond the syntactic and semantic characteristics of negated statements.

1 Introduction

Modeling inference is one of the most challenging tasks in computational linguistics, yet crucial for true natural language understanding. It requires looking beyond the literal meaning of statements and determining what exactly the author intends to convey. The position of *focus* (the new or contrastive information) in a sentence, for instance, is one pragmatic phenomenon that can affect its interpretation. The theory of alternative semantics (Rooth, 1992) assumes that the general function of focus is to evoke *alternatives*, i.e. a set of propositions potentially contrasting with the intended meaning of the sentence. A number of lexical items and constructions have been identified as focus-sensitive in English, such as *only*, *even*, *too*, counterfactual conditionals (*if*), frequency adverbs (*always*, *sometimes*, *often*) and negation. Their semantics depend on the information structure of the sentence; different choices of focus can result in different truth values. For example, Kawamura (2007) illustrates how focus contributes to the interpretation of negation in Sentence 1 below. In English, focus can be explicitly signaled prosodically, e.g. in the form of a strong pitch accent, or syntactically, e.g. by moving focused phrases to a special position in the sentence (Stevens, 2017). If no explicit signal is present, one has to rely on the context to determine which part of the sentence is in focus and, hence, what is its correct interpretation. This is a notoriously difficult task that so far has received little attention in the field.

1. (a) We do not cover PRAGMATICS in the lecture this semester.
   the lecture of this semester covers other fields of linguistics, but not pragmatics
   (b) We do not cover pragmatics IN THE LECTURE this semester.
   pragmatics is covered in the discussion sessions, homework, and exams, but not in the lecture
   (c) We do not cover pragmatics in the lecture THIS SEMESTER.
   the lecture usually covers pragmatics, but not this semester

This paper reports on a reproduction study for the purpose of understanding the performance of a system on detecting implicit positive meaning from negated statements. Such a system could support a range of Natural Language Processing (NLP) technologies that require deep understanding of language, such as Recognizing Textual Entailment (RTE) and Question Answering (QA). The research that we reproduce

Proceedings of the 27th International Conference on Computational Linguistics, pages 2253–2264
Santa Fe, New Mexico, USA, August 20-26, 2018.

is that of Blanco and Sarabi (2016), who propose an interesting methodology to automatically generate positive interpretations from negated statements and score them according to their likelihood. We reflect on the definition of the task and the contribution of the different features to the performance of their system. Our findings show that the features they propose are not able to improve upon our baseline because of class imbalance. Based on an error analysis that we perform on the output of our baseline, we discuss future lines of research that take levels of informativeness into account, which we expect to be also applicable to related problems that have to do with implicit meaning and pragmatic phenomena. The remainder of this paper is structured as follows. In Section 2, we report on related work. Section 3 describes the dataset and the task as created, defined and performed by Blanco and Sarabi (2016). In Section 4, we report on the replication of the original experiment on our test set and the ablation tests on feature combinations against the baseline. In Section 5, we redefine the task as a classification task and apply an error analysis to understand system performance against the baseline and the role of the different features in the classification task. Our error analysis leads to a discussion reported in Section 6 and plans for future work. Section 7 summarizes our conclusions.

2 Related Work

The task of scoring implicit positive meanings from negated statements was preceded by the task of detecting the *focus of negation*. This task was pioneered by Blanco and Moldovan (2011), who argued that the *scope* and *focus* of negation are crucial for a correct interpretation of negated statements. Following Huddleston and Pullum (2002), they defined scope as "the part of the meaning that is negated" and focus as "the part of the scope that is most prominently or explicitly negated." More specifically, according to Blanco and Moldovan (2011), the focus of negation gives rise to implicit positive meaning. To capture this information, they created the PB-FOC corpus, which contains annotations for the focus of negation in the 3,993 verbal negations in PropBank (Palmer et al., 2005). Candidates for focus annotation were the verb itself or any of its semantic roles and annotators were instructed to choose only one.

The PB-FOC corpus was used in the first edition of the *SEM Shared Task, which was dedicated to resolving the scope (Task 1) and focus (Task 2) of negation (Morante and Blanco, 2012). Only one team (Rosenberg and Bergler, 2012) participated in the Focus Detection task. Their system, called CLaCs NegFocus, is rule-based and consists of three components: the identification of explicit negation cues (*not*, *nor* and *never*), the detection of the syntactic scope of negation, and the detection of the focus of negation using a set of four syntactic heuristics (e.g. "if an adverb directly precedes the negated verb and is connected through an *advmod* dependency relation to the negated verb, this adverb is annotated as the focus"). They report an F-measure of 0.584. Blanco and Moldovan (2013) report an F-measure of 0.641 with their system FOC-DET, which is trained with bagging over standard C4.5 decision trees using a set of features derived from gold syntactic annotation and semantic role labels. Whereas both CLaCs NegFocus and FOC-DET use only information from within the sentence, Zou et al. (2015) argue that contextual discourse information plays a critical role in negation focus identification and propose a word-topic graph model that uses this contextual information from both lexical and topical perspectives. They report an accuracy of 69.39 on PB-FOC.

PB-FOC has given rise to various alternative approaches to the annotation of focus and different definitions of the corresponding task. Blanco and Moldovan (2012) introduce the concept of granularity of focus and add fine-grained foci on top of the course-grained foci in PB-FOC; whereas coarse-grained focus includes all words belonging to a semantic role, fine-grained focus comprises fewer words within the semantic role (e.g. *We didn't get [an offer for <u>more than</u> $40]*$_{\text{FOCUS}}$), allowing for more specific implicit positive interpretations ("we got something, but not an offer for more than $40" versus "we got an offer for $40 or less"). Anand and Martell (2012) evaluated the annotations of PB-FOC, arguing that positive interpretations resulting from scalar implicatures and neg-raising predicates should be separated from those (indirectly) resulting from focus, and reannotated the corpus by using an alternative annotation approach that relies on relevant questions under discussion (QUDs).

Another criticism on PB-FOC was raised by Blanco and Sarabi (2016), who point out that the guidelines required the annotators to choose one semantic role as the focus, prioritizing "the one that yields the

most meaningful implicit [positive] information" in case of multiple candidates, but that it is not specified what "most meaningful" means. Therefore, they designed a new annotation task where several positive interpretations per negation (automatically generated by manipulating semantic roles) are scored according to their likelihood (see Section 3 for more details). Similar to the distinction between fine-grained and coarse-grained foci, Sarabi and Blanco (2016) propose to extract and score more fine-grained positive interpretations by manipulating syntactic dependencies instead of semantic roles. Sanders and Blanco (2016) further extend this approach of generating and scoring implicit positive interpretations by applying it to modal constructions.

3 Dataset and Task

The dataset created by Blanco and Sarabi (2016) contains 1,888 positive interpretations generated from 600 negated verbs in OntoNotes 5.0 (Hovy et al., 2006). These positive interpretations are automatically generated by first converting the negated statement into its positive counterpart by (1) removing the negation mark, (2) removing auxiliaries, expanding contractions and fixing third-person singular and past tense, and (3) rewriting negatively-oriented polarity-sensitive items (e.g. *anyone* becomes *someone*). From this positive counterpart, positive interpretations are generated by rewriting each semantic role or the (originally negated) verb. For example, ARG0-ARG4 are rewritten as *someone / some people / something*, and ARGM-TMP is rewritten as *at some point of time*. To illustrate, this results in the three positive interpretations listed for Sentence 2.

2. [The proper climatic conditions]$_{\text{ARG1}}$ don't [exist]$_{\text{V}}$ [in many places in the world]$_{\text{ARGM-LOC}}$.
   (a) The proper climatic conditions {some verb / action / state} in many places in the world.
   but not 'exist'
   (b) {someone / some people / something} exist in many places in the world.
   but not 'The proper climatic conditions'
   (c) The proper climatic conditions exist {somewhere}.
   but not 'in many places in the world'

On average, 3.15 positive interpretations are generated per negation, each of which is manually scored with a value between 0 and 5. A positive interpretation with a score of 5 is deemed very plausible given the context (i.e. the previous and next sentence), whereas a score of 0 indicates that this interpretation is highly implausible. Inter-annotator agreement is calculated using Pearson's correlation and is reported to be 0.761. The task is defined as a regression task, where each positive interpretation becomes an instance for which the system should produce a score.

3.1 Data releases

The dataset of scored positive interpretations is not publicly available, but the authors provided us with all their annotations. Blanco and Sarabi (2016) have used the CoNLL-2011 Shared Task distribution of OntoNotes[1] (Pradhan et al., 2011). In order to use the annotations for learning, both OntoNotes 5.0 and the CoNLL-2011 Shared Task release are required, since the latter contains all annotations but not the underlying words. The organizers of this shared task provide scripts and instructions to merge the annotations with the words in OntoNotes. Originally, the data was meant to be merged with the data from OntoNotes 4.0,[2] but it can also be used with OntoNotes 5.0,[3] which is the final release of the OntoNotes project. The resulting `.conll` files contain a merged representation of all the OntoNotes layers in CoNLL-style tabular format with one line per token, and with multiple columns specifying the annotations for each token. This format is similar to the CoNLL-formatted release of OntoNotes 5.0.[4] However, there are two important differences between the two CoNLL-formatted releases. First, for

[1] `http://conll.cemantix.org/2011`
[2] `https://catalog.ldc.upenn.edu/ldc2011t03`
[3] `https://catalog.ldc.upenn.edu/ldc2013t19`
[4] `https://github.com/propbank/propbank-release`

Source	Feature description
verb	Word form and part-of-speech tag of verb
sem_role	Semantic role label of sem_role (sem_role_label)
	Number of tokens in sem_role
	Word form and part-of-speech tag of head of sem_role
	Syntactic node of sem_role, its parent and left and right siblings in the parse tree
verb-sem_role	Whether verb occurs before or after the sem_role in the negated statement
	Syntactic node of lowest common ancestor of verb and sem_role
	Syntactic paths from verb to sem_role
verbarg-struct	Flags indicating whether verb has each possible semantic role
	Semantic role labels of the first and last roles of verb
	Syntactic nodes and heads of each semantic role attaching to verb

Table 1: Features used by Blanco and Sarabi (2016, p. 1438)

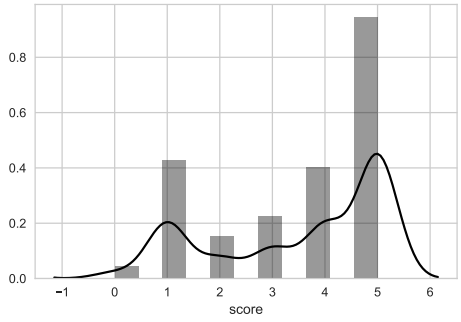


Figure 1: Overall distribution of scores in the complete dataset.

role_label	count	mean	std	min	max
ARG0	316	4.02	1.09	0.0	5.0
ARG1	416	4.53	1.01	0.0	5.0
ARG2	82	4.37	1.35	0.0	5.0
ARG3	1	5.00	NaN	5.0	5.0
ARG4	2	5.00	0.00	5.0	5.0
ARGM-ADV	80	2.40	1.65	0.0	5.0
ARGM-CAU	12	2.50	2.07	0.0	5.0
ARGM-DIR	7	2.43	2.44	0.0	5.0
ARGM-EXT	9	3.56	1.74	1.0	5.0
ARGM-LOC	16	3.62	1.75	0.0	5.0
ARGM-MNR	28	4.54	1.14	0.0	5.0
ARGM-PNC	3	5.00	0.00	5.0	5.0
ARGM-PRP	1	2.00	NaN	2.0	2.0
ARGM-TMP	65	4.03	1.48	0.0	5.0
V	472	2.20	1.40	0.0	5.0

Table 2: Statistics of scores per semantic role in the train set. The mean scores per role are used as a baseline.

CoNLL-2011 some of the documents were split into smaller parts to reduce the complexity of coreference annotation, which increases with the length of a document. Second, disfluencies in conversation data (e.g. restarts, hesitations), which are marked with a special EDITED phrase tag in the original OntoNotes parses, were removed. The splitting of documents and removal of disfluencies result in different sentence and token identifiers, since sentence numbering restarts for each document part and token count excludes disfluent tokens. Therefore, the CoNLL-2011 release is required for resolving the identifiers used in the positive interpretations dataset. However, the CoNLL-2011 release does not provide gold annotations for the complete dataset; for the test set used for this shared task, only automatic annotations are provided. The gold annotations required to replicate the experiment by Blanco and Sarabi (2016) were provided to us by the authors instead.

4 Regression Task: Scoring Positive Interpretations

The system for scoring the positive interpretations as reported in (Blanco and Sarabi, 2016) is not publicly available. However, since the authors provided us with the annotations and the required OntoNotes data, we were able to replicate their experiment for further analysis.

4.1 Experimental set-up

Simulating the approach reported by Blanco and Sarabi (2016), we trained a Support Vector Machine (SVM) for regression with RBF kernel using scikit-learn (Pedregosa et al., 2011) with the set of features

	Blanco & Sarabi	ANONYMIZED		
	Pearson's r	Pearson's r	Spearman's ρ	RMSE
BASELINE (MEAN)		**0.679**	**0.625**	**1.239**
{semrole_label}	**0.603**	**0.656**	0.639	**1.282**
{semrole}		0.632	**0.640**	1.383
{verb-semrole}		0.603	0.595	1.357
{verbarg-struct}		0.148*	0.140*	1.747
{verb}	-0.025	0.078*	0.061*	1.760
{semrole_label, verbarg-struct}		**0.655**	0.631	1.562
{semrole_label, verb-semrole}		0.640	0.619	**1.299**
{semrole, verb-semrole}		0.639	**0.653**	1.367
{verb, semrole_label}		0.633	0.637	1.383
{verb-semrole, verbarg-struct}		0.632	0.615	1.540
{verb, semrole}	**0.630**	0.622	0.634	1.391
{semrole, verbarg-struct}		0.605	0.623	1.431
{verb, verb-semrole}		0.585	0.605	1.436
{verb, verbarg-struct}		0.141*	0.136*	1.749
{verb, semrole_label, verbarg-struct}		**0.655**	0.628	1.591
{semrole_label, verb-semrole, verbarg-struct}		0.642	**0.640**	1.436
{verb, semrole, verb-semrole}	**0.627**	0.631	0.643	**1.387**
{semrole, verb-semrole, verbarg-struct}		0.625	0.633	1.404
{verb, semrole_label, verb-semrole}		0.618	0.615	1.397
{verb, semrole, verbarg-struct}		0.597	0.619	1.440
{verb, semrole_label, verb-semrole, verbarg-struct}		**0.646**	**0.642**	1.452
{verb, semrole, verb-semrole, verbarg-struct}	**0.642**	0.620	0.631	**1.417**

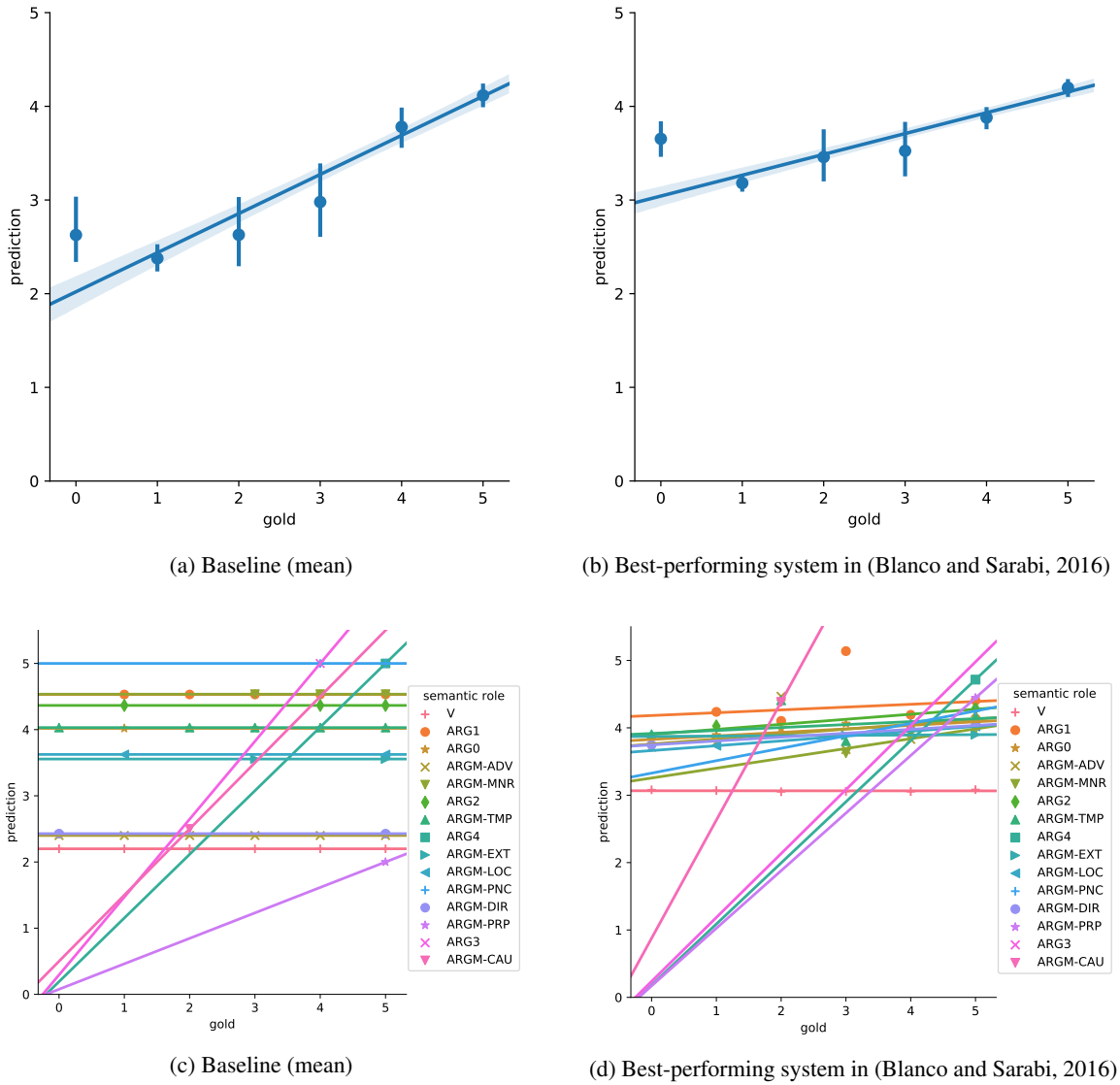
Table 3: Results of the Support Vector Machine (SVM) for regression with RBF kernel with different feature sets compared to those reported in (Blanco and Sarabi, 2016). All were statistically significant ($p < 0.001$), except those indicated with an asterisk.

presented in Table 1. Whereas Blanco and Sarabi (2016) report on results obtained with features extracted from gold-standard and predicted linguistic annotations, we only report on those obtained from the gold-standard annotations in the CoNLL-2011 dataset. We compare the results to a baseline system that simply takes the mean score of the semantic role from which the positive interpretation was generated as calculated over our train set (Table 2). Blanco and Sarabi (2016) evaluate their results using Pearson's correlation (r). However, we question the appropriateness of this evaluation measure, since Pearson's r assumes that the data is continuous and (at least approximately) normally distributed. Whereas the predicted scores are continuous indeed, the gold scores are rather ordinal of nature. Moreover, the data is not normally distributed, as can be seen in Figure 1. Therefore, we report our results using two additional evaluation measures: Spearman's correlation (ρ) with correction for ties, which can be used when the assumptions of Pearson's r are not met, and Root Mean Square Error (RMSE), which is the standard deviation of the residuals (prediction errors). RMSE is a measure of how spread out the residuals are; this way, it informs how concentrated the predicted scores are around the line of best fit. Lower values of RMSE indicate better fit, and since it is an absolute measure with the same units as the score being estimated, the values should be interpreted in the same range, in our case, between 0 and 5.

Unfortunately, due to time constraints, the authors were not (yet) able to provide us with the train/test splits as used in their paper. Therefore, we created our own test set using the same approach: we used a 80/20 split and made sure that all positive interpretations belonging to one negated verb were assigned to either the train or the test set.

4.2 Results

Table 3 presents the results of our experiments with the different combinations of feature sets on the regression task. We observe that, depending on the evaluation measure, different systems perform best. We also observe that our results are mostly comparable to those reported in (Blanco and Sarabi, 2016),



(a) Baseline (mean)

(b) Best-performing system in (Blanco and Sarabi, 2016)

(c) Baseline (mean)

(d) Best-performing system in (Blanco and Sarabi, 2016)

Figure 2: The upper figures show regression lines with a 95% confidence interval. The observations are collapsed in bins to plot an estimate of central tendency. The lower figures show regression lines per semantic role.

although just using the semrole_label as feature shows better results and is even the second-best system after our baseline, if we consider Pearson's *r* and the RMSE scores. This contradicts the finding of Blanco and Sarabi (2016), who found that considering all features yields the best performance. However, we emphasize that we tested on a different test set than Blanco and Sarabi (2016), so it is possible that this accounts for the differences in results. If we look at Spearman's ρ, the feature combination {semrole, verb-semrole} performs best, followed by {verb, semrole_label, verb-semrole, verbarg-struct}.

Figures 2a-2d visualize the regression lines for the baseline system and for the system that was reported as best-performing in (Blanco and Sarabi, 2016). We show both the overall tendency as the relation between the gold and the predicted scores per semantic role. As expected, Figure 2c shows flat lines for each semantic role, corresponding to the mean scores of each, except for those semantic roles that occur only once in the test set (ARG3, ARGM-CAU and ARGM-PRP) and ARG4, which is correctly scored as 5 in all cases. In contrast, Figure 2d shows more variation within most semantic roles (indicated by their slope), but in general, the predicted scores are too high, as can also be concluded from the overall overview in Figure 2b. In fact, the lowest predicted score according to this system is 2.96. The baseline seems to account a bit better for the lower and middle scores (the lowest predicted score is 2).

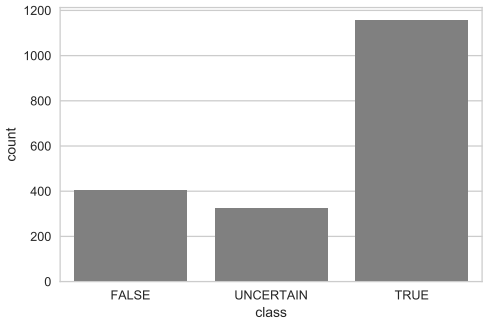


Figure 3: Overall distribution of the three classes in the complete dataset.

role_label	FALSE	UNCERTAIN	TRUE	total
ARG0	19	45	**252**	316
ARG1	14	26	**376**	416
ARG2	7	4	**71**	82
ARG3	-	-	**1**	1
ARG4	-	-	**2**	2
ARGM-ADV	**33**	24	23	80
ARGM-CAU	**6**	2	4	12
ARGM-DIR	**4**	-	3	7
ARGM-EXT	2	1	**6**	9
ARGM-LOC	3	3	**10**	16
ARGM-MNR	1	2	**25**	28
ARGM-PNC	-	-	**3**	3
ARGM-PRP	-	1	-	1
ARGM-TMP	8	9	**48**	65
V	**224**	163	85	472

Table 4: Statistics of the three classes per semantic role in the train set. The most frequent classes per role (in bold) are used as a baseline.

5 Classification Task: Classifying Positive Interpretations as TRUE or FALSE

In a second experiment, we redefine the task as a classification task. We argue that the scores assigned to each positive interpretation in the dataset reflect the confidence of the annotators rather than what we are interested in predicting. In other words, in the end a binary classification system that can predict whether some positive meaning is implied in a negated statement or not would suffice in most real-world applications. Instead of predicting a score between 0 and 5, one option is thus to redefine the task as a binary classification problem where each positive interpretation is to be classified as either TRUE or FALSE. The original annotated scores can easily be divided: all scores between 0-2 become FALSE, and all scores between 3-5 become TRUE. However, intuitively, it is clear that the scores 2 and 3 are somewhat problematic in the sense that they present the middle cases that less clearly belong to either class. Therefore, these two scores could potentially make up their own class of UNCERTAIN. As can be seen in Figure 3, the UNCERTAIN class comprises a considerably large portion of the complete dataset.

5.1 Experimental set-up

We use the same sets of features to train an SVM with RBF kernel, but now for classification. We apply it to both the binary classification task (TRUE/FALSE) and the tertiary classification task (TRUE/FALSE/UNCERTAIN). The results are evaluated using precision, recall and F1-measure. We use weighted averaging for all scores, which accounts for class imbalance by computing the average of binary metrics in which each class' score is weighted by its presence in the true data sample. Our baseline system takes the most frequent class of the semantic role from which the positive interpretation was generated (corresponding to the classes in bold in Table 4 in both tasks).

5.2 Results

We summarize our results on both the tertiary and the binary classification tasks in Table 5. None of the feature combinations were able to beat the most-frequent baseline, which achieves an F1-measure of 0.725 with three classes and 0.840 with two classes. The differences with the second-best system, which uses only semrole_label as a feature, are small (0.721 and 0.834 respectively), but adding more features only appears to hurt performance. This is mostly in line with what we observed in the regression task. We analyzed the precision, recall and F1-measure per class for each of the feature combinations and observe that for most of them, the FALSE class performs 0.0 on each in both tasks, except for the better performing ones (indicated with bold scores in Table 5). Moreover, the UNCERTAIN class always performs 0.0 in the tertiary task, including the baseline. In the binary task, the baseline achieves 0.65 / 0.89 / 0.75 on precision, recall and F1-measure on the FALSE class and 0.95 / 0.81 / 0.87 on the TRUE class. In the tertiary task, the respective scores are 0.54 / 0.91 / 0.68 (FALSE) and 0.91 / 0.85 / 0.88 (TRUE).

Features	Tertiary Classification Precision	Recall	F1	Binary Classification Precision	Recall	F1
BASELINE (MOST FREQUENT)	**0.711**	**0.762**	**0.725**	**0.864**	**0.833**	**0.840**
{semrole_label}	**0.702**	**0.759**	**0.721**	**0.853**	**0.828**	**0.834**
{verb-semrole}	0.681	0.759	0.716	0.832	0.825	0.828
{semrole}	0.619	0.714	0.663	0.514	0.717	0.599
{verb}	0.424	0.651	0.513	0.514	0.717	0.599
{verbarg-struct}	0.424	0.651	0.513	0.514	0.717	0.599
{semrole_label, verb-semrole}	**0.681**	**0.759**	**0.716**	**0.832**	**0.825**	**0.828**
{verb, verb-semrole}	**0.681**	**0.759**	**0.716**	**0.832**	**0.825**	**0.828**
{semrole, verb-semrole}	**0.681**	**0.759**	**0.716**	0.821	0.817	0.819
{verb, semrole_label}	**0.681**	**0.759**	**0.716**	0.514	0.717	0.599
{verb, semrole}	0.424	0.651	0.513	0.514	0.717	0.599
{verb, verbarg-struct}	0.424	0.651	0.513	0.514	0.717	0.599
{semrole, verbarg-struct}	0.424	0.651	0.513	0.514	0.717	0.599
{semrole_label, verbarg-struct}	0.424	0.651	0.513	0.514	0.717	0.599
{verb-semrole, verbarg-struct}	0.424	0.651	0.513	0.514	0.717	0.599
{verb, semrole_label, verb-semrole}	**0.681**	**0.759**	**0.716**	**0.832**	**0.825**	**0.828**
{verb, semrole, verb-semrole}	0.653	0.735	0.691	0.514	0.717	0.599
{verb, semrole, verbarg-struct}	0.424	0.651	0.513	0.514	0.717	0.599
{verb, semrole_label, verbarg-struct}	0.424	0.651	0.513	0.514	0.717	0.599
{verb, verb-semrole, verbarg-struct}	0.424	0.651	0.513	0.514	0.717	0.599
{semrole, verb-semrole, verbarg-struct}	0.424	0.651	0.513	0.514	0.717	0.599
{semrole_label, verb-semrole, verbarg-struct}	0.424	0.651	0.513	0.514	0.717	0.599
{verb, semrole, verb-semrole, verbarg-struct}	**0.424**	**0.651**	**0.513**	**0.514**	**0.717**	**0.599**
{verb, semrole_label, verb-semrole, verbarg-struct}	**0.424**	**0.651**	**0.513**	**0.514**	**0.717**	**0.599**

Table 5: Results of the Support Vector Machine (SVM) for classification with RBF kernel with different feature sets on the classification task with three classes and with two classes.

5.3 Error analysis

The results on the classification task can easily be explained if we have another look at Figure 3 and Table 4. ARG0, ARG1 and V are by far the most frequent cases in the dataset and their classes are very imbalanced. ARGM-ADV, ARGM-CAU and ARGM-DIR are the only roles for which FALSE is the most frequent class and the classes are somewhat balanced, but these roles are not as well represented in the dataset. As expected, our error analysis on the results of the baseline show that the positive interpretations generated from the verb itself comprise the largest portion of classification errors: 39 out of 63 (62%) were incorrectly predicted as FALSE. This is followed by ARGM-ADV (16%, incorrectly predicted as FALSE), ARG1 (8%, TRUE), ARGM-TMP (5%, TRUE) and ARG0 (3%, TRUE). ARG2, ARGM-DIR, ARGM-LOC and ARGM-PRP were only misclassified once. From these results, we learn that the challenge of this task is to make a system perform well for the minority classes of each role. In other words, we need to know when the verb or ARGM-ADV results to TRUE and when the other roles result to FALSE.

We discuss some examples of misclassification. According to the gold annotations, the positive interpretation generated from the verb *create* in Sentence 3 results to TRUE. We argue that this is because the sentence contains *contrastive focus*. That is, *create* contrasts with another verb mentioned in the previous context: *enrich*. Evidence is provided by the subject being shared between the verbs, i.e. *a diversity of cultures*. Generally speaking, implicit positive meanings involving contrasting actions seem to be more easily evoked when the verb has a strong antonym, even if this is not explicitly mentioned in the context. For example, verbs like *win* (*lose*) or *like* (*dislike*) are more likely to give rise to alternatives than verbs that do not have a clear opposite meaning, such as *write*.

3. [A diversity of cultures]$_{\text{ARG0}}$ will only enrich a person, not **create** [a crisis of identity]$_{\text{ARG1}}$.
   A diversity of cultures will only enrich a person, {*some verb / action / state*} *a crisis of identity, but not 'create'*

Another observation in the TRUE positive interpretations generated from the verb, is that many of them contain coreferential expressions. For example, all semantic roles belonging to the negated verb in Sentence 4 consist of pronouns, indicating that the agent and patient have already been introduced in the previous context. The acceptability of this positive interpretation seems to be a result of *new information focus*, since the only 'new' information is the action expressed by the verb. This makes it more likely that the negation of the action is what the author intended to convey.

4. But [he]$_{\text{ARG0}}$ didn't **win** [it]$_{\text{ARG1}}$.
   But he {some verb / action / state} it, but not 'won'

Sentence 5, on the other hand, illustrates how information that is already introduced in the previous context is less likely to be in the pragmatic scope of negation. The ARG1, *the advance*, is coreferential with *gained* in the preceding context; the new, relevant information is the lack of participation of *many issues* in this advance. Therefore, in this case, the positive interpretation generated from this semantic role is FALSE instead of the more frequent TRUE class associated with ARG1.

5. But while the Composite gained 1.19, to 462.89, [many issues]$_{\text{ARG0}}$ didn't **participate** [in the advance]$_{\text{ARG1}}$.
   But while the Composite gained 1.19, to 462.89, many issues participated {in someone / some people / something}, but not 'in the advance'

6 Discussion and future work

The relation between information structure and inference is an interesting research area that needs further exploration in computational linguistics. Blanco and Moldovan (2011) laid some important groundwork for deriving implicit positive meaning from negations by trying to detect the (single) focus of negation. However, as correctly noted by Anand and Martell (2012), implicit positive meaning only indirectly results from focus. This is partially because, as Kawamura (2007) explains, negation does not *require* a focused element (in contrast to focus-sensitive elements such as *only*, *even* and *either*) and at the same time induces ambiguity in focus constructions. That is, negation without focus can still be grammatical; its absence simply produces a different interpretation. If the sentence *does* contain focus, negation yields a focus-associated reading (6a) and a non-associated reading (6b), as Kawamura (2007) illustrates with the example below. This ambiguity has been regarded as a result of the scope interaction of focus and negation (Jackendoff, 1972; Krifka, 2006).

6. John didn't criticize Mary [at the conference]$_{\text{FOCUS}}$.
   (a) John criticized Mary. It was not at the conference.
   (b) John did something other than criticizing Mary. It happened at the conference.

Therefore, we think Blanco and Sarabi (2016) improved upon the task by scoring each and every potential positive interpretation generated from the negation. The question remains, however, whether we really need to score positive interpretations on a scale from 0 to 5. We believe that, from an application perspective, classifying them as TRUE/FALSE (and optionally UNCERTAIN) is sufficient.

We have shown that class imbalance in the dataset is the reason why our baseline performs so well on the task, and that –even though there is no one-to-one correspondence between focus and implicit positive meaning– knowing which part of the proposition is new or contrastive seems to play a crucial role in correctly predicting the infrequent classes. We therefore argue that an approach to detecting implicit meaning should aim to include this pragmatic layer of information. The features described in this paper only partially capture this layer since they rely on explicit signals within the sentence (i.e. by determining which information is present and in what position), but they do not take context into account. However, we need more annotated data and more experiments to confirm our hypothesis.

Special cases of negation that we would like to highlight here are those that contain negatively-oriented polarity-sensitive items, such as *anyone*, *anything* or *either*. In the current definition of the task, these

items are rewritten to positively-oriented polarity-sensitive items (e.g. *anyone* becomes *someone*, *anything* becomes *something*) before generating the positive interpretations. For example, Sentence 7 would result in the interpretations 7a and 7b with this procedure. While 7a may be classified as UNCERTAIN (or, less likely, as TRUE), 7b will always be FALSE. Moreover, neither interpretations cover the actual meaning of the sentence. Instead, it should be interpreted as *"Kim read nothing"*, which in turn brings us to the next question: how to deal with other markers of negation, such as *nothing* or *nobody*? We suggest treating sentences with these kind of markers of negation or with negatively-oriented polarity-sensitive items both as special cases that probably do not need any annotation, since their correct interpretation directly follows from the meaning of the words.

7. Kim didn't read anything.
   (a) {*someone / some people / something*} *read something (but not Kim)*
   (b) *Kim read* {*something*}

Finally, while the task explored in this paper restricts itself to negation, it could possibly be extended to include also inferences resulting from focus in sentences without negation or with other focus-sensitive elements. As Kawamura (2007), among others, describes, the semantic effect of focus depends on the type of focus-sensitive elements present in the sentence. As noted above, some focus-sensitive elements require a focused element for their interpretation, while others do not. In addition, in sentences without focus-sensitive elements, focus-shift does not affect the truth conditions (even though intuitively the interpretations are different), whereas the truth conditions of sentences with a focus-sensitive element such as *only* do change. That is, if John reads something other than books in the bathroom, Sentence 8a and 8b can both be TRUE, while the focus-shift causes Sentence 9a to be FALSE.

8. (a) John reads [books]$_{\text{FOCUS}}$ in the bathroom.
   (b) John reads books [in the bathroom]$_{\text{FOCUS}}$.

9. (a) John only reads [books]$_{\text{FOCUS}}$ in the bathroom.
   (b) John only reads books [in the bathroom]$_{\text{FOCUS}}$.

This leaves us with many opportunities for future work to further investigate the inferences and truth conditions of sentences in relation to the interaction between focus and focus-sensitive elements.

7 Conclusion

We reimplemented a system for scoring implicit positive interpretations from negated statements for the purpose of understanding its performance, the task itself and the role that different features play. We have argued that the original task could be redefined as a classification task, where each positive interpretation is to be classified as TRUE or FALSE (optionally, with a third class of UNCERTAIN representing the middle cases). We showed that a simple baseline that takes the mean over the scores or the most frequent class per semantic role is hard to beat in both the regression and the classification tasks. Our error analysis indicated that improvements can only be achieved with a system that is able to account for the infrequent classes, and we hypothesize that modeling informativeness (e.g. new information or contrastive focus) may be useful in this respect. This would require looking beyond the syntactic and semantic characteristics of the proposition and taking the broader context into account. Our code is publicly available at `https://github.com/cltl/positive-interpretations`.

Acknowledgements

This research is supported by the Amsterdam Data Alliance in the QuPiD project and the Netherlands Organization for Scientific Research (NWO) via the Spinoza-prize awarded to Piek Vossen in the project "Understanding Language by Machines" (SPI 30-673, 2014-2019). We would like to thank Zahra Sarabi and Eduardo Blanco for sharing their data and for their willingness to answer our many questions. We are also grateful to the anonymous reviewers for their valuable comments.

References

Pranav Anand and Craig Martell. 2012. Annotating the focus of negation in terms of Questions Under Discussion. In *Proceedings of the Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics*, pages 65–69.

Eduardo Blanco and Dan Moldovan. 2011. Semantic representation of negation using focus detection. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 581–589.

Eduardo Blanco and Dan Moldovan. 2012. Fine-grained focus for pinpointing positive implicit meaning from negated statements. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 456–465, Montréal, Canada, June. Association for Computational Linguistics.

Eduardo Blanco and Dan Moldovan. 2013. Retrieving implicit positive meaning from negated statements. *Natural Language Engineering*, 20(4):501–535.

Eduardo Blanco and Zahra Sarabi. 2016. Automatic generation and scoring of positive interpretations from negated statements. In *Proceedings of NAACL-HLT*, pages 1431–1441.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the Human Language Technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60.

Rodney Huddleston and Geoffrey K Pullum. 2002. *The Cambridge Grammar of English Language*. Cambridge: Cambridge University Press.

Ray S Jackendoff. 1972. *Semantic interpretation in generative grammar*. M.I.T. Press,, Cambridge, Mass.

Tomoko Kawamura. 2007. *Some interactions of focus and focus sensitive elements*. State University of New York at Stony Brook.

Manfred Krifka. 2006. Association with Focus. In Valerie Molnar and Susanne Winkler, editors, *The Architecture of Focus*, pages 105–136. Mouton de Gruyter.

Roser Morante and Eduardo Blanco. 2012. * SEM 2012 Shared Task: Resolving the scope and focus of negation. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 265–274, Montréal, Canada.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. CoNLL-2011 Shared Task: Modeling Unrestricted Coreference in OntoNotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–27, Portland, Oregon.

Mats Rooth. 1992. A theory of focus interpretation. *Natural language semantics*, 1(1):75–116.

Sabine Rosenberg and Sabine Bergler. 2012. UConcordia: CLaC negation focus detection at *SEM 2012. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 294–300.

Jordan Sanders and Eduardo Blanco. 2016. Automatic extraction of implicit interpretations from modal constructions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Austin, Texas, USA.

Zahra Sarabi and Eduardo Blanco. 2016. Understanding negation in positive terms using syntactic dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Austin, Texas, USA.

John Scott Stevens. 2017. The pragmatics of focus. In *Oxford Research Encyclopedia of Linguistics*. Oxford: Oxford University Press.

Bowei Zou, Qiaoming Zhu, and Zhou Guodong. 2015. Unsupervised negation focus identification with word-topic graph model. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1632–1636, Lisbon, Portugal.

Exploratory Neural Relation Classification for Domain Knowledge Acquisition

Yan Fan, Chengyu Wang, Xiaofeng He*
School of Computer Science and Software Engineering,
East China Normal University, Shanghai, China
{eileen940531,chywang2013}@gmail.com, xfhe@sei.ecnu.edu.cn

Abstract

The state-of-the-art methods for relation classification are primarily based on deep neural networks. This supervised learning method suffers from not only limited training data, but also the large number of low-frequency relations in specific domains. In this paper, we propose an exploratory relation classification method for domain knowledge harvesting. The goal is to learn a classifier on pre-defined relations while discovering new relations expressed in texts. A dynamically structured neural network is introduced to classify entity pairs to a continuously expanded relation set. We further propose the similarity sensitive Chinese restaurant process to discover new relations. Experiments conducted on a large corpus show that new relations are discovered with high precision and recall, illustrating the effectiveness of our method.

Title and Abstract in Chinese

基于领域知识获取的探索式神经网络关系分类

近几年来，关系分类任务主要利用神经网络模型来自动学习复杂的特征。然而这类监督式学习方法的局限性在于，首先它需要大量训练数据，其次特定领域存在长尾的低频关系无法被有效地预定义。在本论文中，我们提出了基于领域知识获取的探索式关系分类任务。它的目标在于学习预定义关系的分类器，同时从文本中发现新的语义关系。我们提出了一个动态结构的神经网络，它可以对持续扩充的关系集进行分类。我们进一步提出了相似度敏感的中餐馆过程算法，用于发现新关系。基于大语料库上的实验证明了该神经网络的分类效果，同时新发现的关系也有较高的准确率和召回率。

1 Introduction

Relation classification assigns semantic relation labels to entity pairs in texts. Besides traditional feature-based (Kambhatla, 2004) and kernel-based (Bunescu and Mooney, 2005) approaches, neural networks (NNs) are introduced to harvest relational facts in recent years. Classical architectures include convolutional neural networks (CNNs) (Zeng et al., 2014), recurrent neural networks (RNNs) (Xu et al., 2015), etc. However, above methods are still insufficient for domain knowledge acquisition due to two challenges: (i) most domain entities rarely occur in the corpus, hence pattern-based methods easily suffer from the feature sparsity problem; (ii) a domain knowledge graph tends to be incomplete w.r.t. relation labels and facts (Fan et al., 2017). As a result, unlabeled entity pairs are likely to be wrongly forced into existing relation labels by distant supervision approaches, instead of their true, unknown labels.

In this paper we propose a method, Exploratory Relation Classification (ERC), to populate domain knowledge graphs automatically. The goal of ERC is to not only classify entity pairs into a finite set of pre-defined relations, but also simultaneously discover previously unseen relation types and their respective instances from plain texts with high confidence. The resulting numbers of relation types and facts in the domain knowledge graph grow continuously.

*Corresponding author.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2265–2276
Santa Fe, New Mexico, USA, August 20-26, 2018.

To solve ERC problem, we propose a Dynamic Structured Neural Network (DSNN). In the network, the convolutional and recurrent network units are employed to model local information such as syntactic and lexical features from the sentence level. For the context sparsity problem (Challenge (i)) an embedding layer is introduced to encode corpus-level semantic features of domain entities. The output classes of DSNN can be automatically expanded during the iterative training process, in order to classify entity pairs to both known and newly discovered relations. To explore new relations and corresponding instances, we introduce the similarity sensitive Chinese restaurant process (ssCRP) to generate clusters of entity pairs from unlabeled data that can not be classified into any known relations (Challenge (ii)). We can see that the DSNN only takes the training data of known relations directly derived from a domain knowledge graph without requiring any training data for new relations. Therefore, our approach is a very weakly supervised one with minimal human intervention.

The rest of the paper is organized as follows. Section 2 summarizes the related work and discusses relations between existing methods and ours. The detailed approach is introduced in Section 3, Section 4 and Section 5, with experiments presented in Section 6. Finally, we conclude this paper in Section 7.

2 Related Work and Discussion

In NLP research, feature-based (Kambhatla, 2004) and kernel-based (Bunescu and Mooney, 2005) methods have been proposed to utilize lexical, syntactic and semantic features for relation classification. In recent years, neural network based approaches are introduced to improve the performance. Word embeddings of entities are frequently used as inputs, instead of one-hot representations (Mikolov et al., 2013; Pennington et al., 2014). As shown in (Wang and He, 2016; Wang et al., 2017), word embeddings can be used for relation prediction. For neural network architectures, CNNs are capable of learning consecutive contexts of entities as local lexical features (Zeng et al., 2014; Vu et al., 2016), while RNNs exploit syntactic features by modeling long-term dependencies of sentences via memory units. The model of Long Short Term Memories (LSTMs) (Hochreiter and Schmidhuber, 1997) is an effective variant of RNNs to encode the shortest dependency path (SDP) between entity pairs (Xu et al., 2015; Shwartz et al., 2016). More recent work focuses on the design of integrated architecture, combining both CNNs and RNNs. Cai et al. (2016) propose a bidirectional recurrent CNN to model the directional information along the SDP forwards and backwards. The integrated neural network proposed by Raj et al. (2017) also experiments with attention-based pooling strategy for biomedical texts. Our model provides a more intuitive representation by feature concatenation instead of network layering, enabling the adaption of dynamic changes as new relations are discovered in iterations.

For relation discovery, one way is to formulate the task as open relation extraction (OpenRE) (Banko et al., 2007). Typical OpenRE systems include TextRunner (Banko et al., 2007), WOE (Wu and Weld, 2007), ReVerb (Etzioni et al., 2011), etc. Unlike approaches confined to a fixed set of pre-defined relations, these systems explore relations in a more general way. Contrary to the fact that the contexts for domain knowledge are sparse, the underlying idea of OpenRE is to identify phrases from sentences that are able to indicate unknown relations based on data redundancy. Therefore, OpenRE methods are not suitable for domain-specific knowledge harvesting within a limited text corpus. Similarly, Riedel et al. (2013) learn universal schemas by matrix factorization without pre-defined relations. An alternative way is to use clustering algorithms. Compared to standard clustering algorithms (e.g., KMeans), non-parametric Bayesian models (Rasmussen, 1999) are more suitable in our scenario, because they can automatically learn the number of clusters. Studies afterwards exploit data features for spatial and temporal data, such as distance dependent CRP (ddCRP) (Blei and Frazier, 2010), etc.

Our task is also related to a typical case of multi-class semi-supervised learning when unlabeled data contains unknown classes. We suggest that existing paradigms for this problem are not much useful for domain knowledge acquisition. Generally, distant supervision (Mintz et al., 2009) is employed to generate training data by aligning knowledge bases with free texts. But its strong alignment assumption may lead to noisy annotation results (Zhang and Wang, 2017). Traditional approaches assign new class labels to part of the unlabeled data on condition that none of the existing classes can fit the data well (Nigam et al., 2000; Dalvi et al., 2013). Therefore, considering the sparsity of domain knowledge, these

methods applied on such corpus will introduce plenty of small classes, which are in fact noises and thus meaningless. In contrast, our ssCRP-based approach works more effectively since we only discover one large class with confident instances in each iteration. Hence such noises in the data will not be forced to generate small classes and are automatically discarded when the last iteration stops.

3 Dynamic Structured Neural Network for ERC

In this section, we present the definition of the task ERC and a high level introduction of the DSNN approach.

3.1 Task Definition

Before introducing the training process of DSNN, we first provide some preliminaries. Denote entity pairs as $X^l = \{(e_1, e_2)\}$ with corresponding labels Y^l, and unlabeled entity pairs as $X^u = \{(e_1, e_2)\}$. Exploratory Relation Classification (ERC) task is defined as follows:

Definition 1. *Given labeled data (X^l, Y^l) and unlabeled data X^u, the goal of ERC is to train a model to predict the relations for entity pairs in X^u with $K + n$ output labels, where K denotes the number of pre-defined relations in Y^l, and n is the number of newly discovered relations which is unknown.*

3.2 An Overview of DSNN

Algorithm 1 shows the iterative training process of DSNN. In each iteration, the algorithm tries to detect a new relation from unlabeled data based on ssCRP, and expands the neural network structure to perform relation classification over existing and new relations. Briefly, it consists of three modules: base neural network training[1], relation discovery and relation prediction.

Algorithm 1 DSNN Training Process

Input: Entity pairs X^l and their labels Y^l, unlabeled entity pairs X^u, pre-defined relation set $R_{old} = \{r_1, \ldots, r_K\}$
Output: New relation set $R_{new} = \{r_{K+1}, \ldots, r_{K+n}\}$, populated labeled entity pairs X^l and their labels Y^l

Initialize $R_{new} = R_{old}, t = 1$
while no new relations can be discovered **do**
  // Base neural network training
  Train base neural network NN_t with X^l and Y^l
  // Relation discovery
  Generate candidate clusters $\{C_1, \ldots, C_m\}$ = *ssCRP*(X^u, X^l)
  C^* = *PickTheBestCluster*$(C_1, \ldots, C_m)$
  r^* = *MapToRelation*(C^*)
  if $r^* \notin R_{old}$ **then**
    $R_{new} = R_{new} \cup \{r^*\}$
  end if
  Label all entity pairs in C^* with relation r^* to be Y^*
  $X^l = X^l \cup C^*, Y^l = Y^l \cup Y^*, X^u = X^u \backslash C^*$
  // Relation prediction
  for each (e_1, e_2) in X^u **do**
    $\Pr(r|e_1, e_2, NN_t)$ =*PredictRelDistribution*$(e_1, e_2, R_{old}, NN_t)$
    if *NotNearUniform*$(\Pr(r|e_1, e_2, NN_t))$ **then**
      Label (e_1, e_2) with relation $\text{argmax}_{r^*} \Pr(r|e_1, e_2, NN_t)$ to be Y^*
      $X^l = X^l \cup \{(e_1, e_2)\}, Y^l = Y^l \cup Y^*, X^u = X^u \backslash \{(e_1, e_2)\}$
    end if
  end for
  $R_{old} = R_{new}, t = t + 1$
end while
return R_{new}, X^l, Y^l

In each iteration t, the first step is to train base neural network NN_t with training data X^l and Y^l. Note that in the first iteration, model NN_t is trained fully supervised. When $t > 1$, NN_t is trained semi-supervisedly, utilizing both labeled training data and new relations discovered in previous $t - 1$ iterations.

[1]In this paper, we refer to the structure of DSNN without the output layer expansion step as "base neural network". In each iteration, the training process of base neural network is the same despite the number of output units.

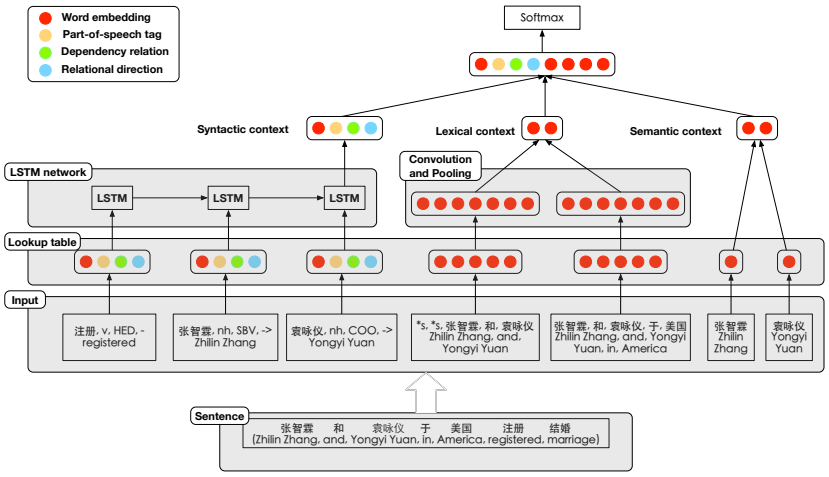


Figure 1: The architecture of the base neural network with k=2.

After NN_t is trained, ssCRP is employed to generate m clusters from X^u (i.e., $C_1, \ldots, C_m$), where each cluster represents the collection of entity pairs that are most likely to share the same underlying relation. We select the "best" cluster C^* from $C_1, \ldots, C_m$ based on the size and intra-cluster similarity, and map it to the relation r^*. r^* could either be one of the pre-defined relations, or a newly discovered relation. Either way, we update X^l and Y^l with C^* labeled as r^*, and remove them from X^u.

For unlabeled entity pairs $(e_1, e_2) \in X^u$, we use model NN_t to predict the probability distribution $\Pr(r|e_1, e_2, NN_t)$ over all possible relations. If the distribution is not "near uniform", the model is confident to predict the relation $r^* = \text{argmax}\Pr(r|e_1, e_2, NN_t)$. Hence, (e_1, e_2) will be labeled as r^* and added to labeled data (i.e., X^l and Y^l). Here, we only add unlabeled data with confident predictions to the training set, in order to avoid error propagation in a semi-supervised learning environment.

After the execution of the above three modules in one iteration, if a new relation is found with its seed relation instances (i.e., entity pairs), the structure of the neural network retrained in the next iteration will be adjusted dynamically with parameters updated.

4 The Architecture of the Base Neural Network

Base neural network takes texts with annotated entity pairs as input and classify them to a fixed relation set. It learns representation of texts by exploiting the following three contexts: (i) syntactic contexts; (ii) lexical contexts; and (iii) global semantic contexts. The overall architecture is illustrated in Fig. 1.

Syntactic contexts. Previous work exploits the shortest dependency path (SDP) (Bunescu and Mooney, 2005) to capture predicate-argument sequences (Cai et al., 2016; Xu et al., 2015; Shwartz et al., 2016). However, SDP can not handle the dependency relation properly between e_1 and e_2 due to the lack of the associated predicate. Consider the example in Fig. 2. Entities "张智霖 (Zhilin Zhang)" and "袁咏仪 (Yongyi Yuan)" are in the coordinate dependency relation via a direct arc. Thus, the SDP between two entities is "张智霖 (Zhilin Zhang) $\rightarrow$ 袁咏仪 (Yongyi Yuan)", which contains not much useful information. In this paper, we design a structure to capture the root verb along the dependency path, namely root augmented dependency path (RADP):

Definition 2. *An RADP is the combination of two subpaths derived from the dependency graph, one from e_1 to the root verb, and the other one from the root to e_2.*

Here, the RADP is "注册 (registered) $\rightarrow$ 张智霖 (Zhilin Zhang) $\rightarrow$ 袁咏仪 (Yongyi Yuan) ". Compared to SDP, it includes the extra root verb "注册 (registered)", a crucial complement indicating the relation "配偶 (spouse)" holds between two entities.

In the base neural network, each node along the RADP is considered as a syntactic unit (Shwartz et al., 2016) with four parts: 1) word embedding; 2) POS tag; 3) dependency relation, which is the label on the arc from the governing node; and 4) relational direction, one of the three labels "$\rightarrow$", "$\leftarrow$" and "—". Thus we present a syntactic unit as the concatenation of four parts: $\vec{v}_{syn} = [\vec{v}_{word}, \vec{v}_{pos}, \vec{v}_{rel}, \vec{v}_{dir}]$

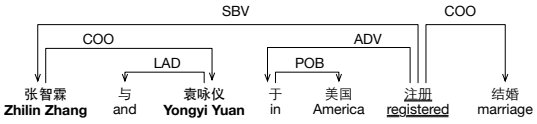


Figure 2: An example of the dependency graph.

where $\vec{v}_{word}$, $\vec{v}_{pos}$, $\vec{v}_{rel}$, $\vec{v}_{dir}$ are the embeddings of the word, POS tag, dependency relation and relational direction, respectively. $[\cdot]$ is the concatenation of embeddings. These embeddings are initialized as one-hot vectors except for $\vec{v}_{word}$, pre-trained on a large corpus.

We employ an LSTM network to handle the long-term dependencies. For an RADP with sequence $w_1, \ldots, w_j$, the corresponding embeddings $\vec{v}_{syn_1}, \ldots, \vec{v}_{syn_j}$ are fed into an LSTM network. The output vector $\vec{v}_{lstm}$ is the context vector modeling the syntactic features of the whole sequence.

Lexical contexts. Specific patterns around target entities can imply semantic relations between entity pairs (Hearst, 1992). Similar to previous research (Zeng et al., 2014; Vu et al., 2016), given a window size k, we define the lexical context of an annotated entity as k words ahead and k words following. Formally, the lexical context vector $\vec{v}_{con}$ is represented as follows: $\vec{v}_{con} = [\vec{v}_{w_{i-k}}, \ldots, \vec{v}_{w_{i-1}}, \vec{v}_{e_i}, \vec{v}_{w_{i+1}}, \ldots, \vec{v}_{w_{i+k}}]$ where $\vec{v}_{e_i}$ and $\vec{v}_{w_j}$ are embeddings of entity and context word at index i and j.

For entities e_1 and e_2 in the pair, $\vec{v}_{con_{e_1}}$ and $\vec{v}_{con_{e_2}}$ are fed into CNN units, respectively. After a convolutional layer and a max pooling layer, the resulting vectors are concatenated as $\vec{v}_{cnn}$.

Semantic contexts. The context-sparsity problem of domain knowledge motivates us to explore semantic contexts in the global corpus, instead of being limited to local features. Based on the distributional hypothesis, we train a Skip-gram model (Mikolov et al., 2013) to learn the distributional representations of words in a large corpus. For entity pair e_1 and e_2, the embeddings $\vec{v}_{e_1}$ and $\vec{v}_{e_2}$ are further concatenated to build the final representation of semantic contexts: $\vec{v}_{emb} = [\vec{v}_{e_1}, \vec{v}_{e_2}]$.

Prediction. The final context representation for prediction is the concatenation of three resulting vectors $\vec{v}_f = [\vec{v}_{lstm}, \vec{v}_{cnn}, \vec{v}_{emb}]$. Then we apply a *softmax* layer on $\vec{v}_f$ to predict the class distribution y: $\vec{y} = \text{softmax}(W \cdot \vec{v}_f + b)$ where W is the transformation matrix and b is the bias vector. The prediction of base neural network is the relation whose probability is maximal in $\vec{y}$.

5 Relation Discovery

To expand the output layer of base neural network, a clustering algorithm ssCRP is proposed to explore new relations from unlabeled data. In this part, we first introduce how the network expands, and briefly cover the basics of CRP, followed by the algorithm ssCRP in detail.

5.1 Network Expansion

After training base neural network, we take its final hidden layer as the representation for each entity pair, since it contains all high-level context features. The representations used as the input of ssCRP should be close to each other in the embedding space, if the same relation holds for these pairs.

In iteration t, ssCRP and the table selection process (to be discussed later) generate a cluster C^* from unlabeled data. Let r^* be the mapping relation that entity pairs in C^* hold, $\{r_1, \ldots, r_{K+l}\}$ be the K pre-defined relations and l new relations discovered in $t-1$ iterations. If $r^* \notin \{r_1, \ldots, r_{K+l}\}$, a new relation is found and base neural network trained in the next iteration $t+1$ expands its output layer with an extra class; otherwise, it remains the same with $K+l$ output classes. Fig 3 shows the base neural network and its expansion network.

5.2 Similarity Sensitive Chinese Restaurant Process

Preliminaries. The Chinese restaurant process (CRP) is a stochastic process, which groups customers into random tables where they sit (Aldous, 1985). Assume N_p denotes the number of customers sitting at table p, z_i denotes the index of the table where the i-th customer sits, and vector $\vec{z}_{-i}$ represents the

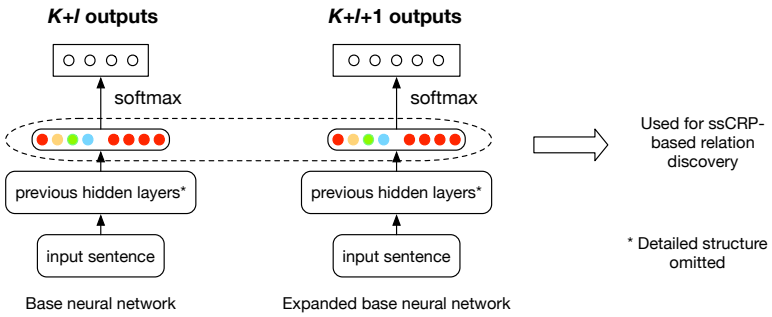


Figure 3: Base neural network and the expanded network.

table assignment for other customers. The distribution over table assignments is as follows:

$$\Pr(z_i = p \mid \vec{z}_{-i}, \alpha) \propto \begin{cases} N_p & \text{if } p \leq K \\ \alpha & \text{if } p = K+1 \end{cases} \tag{1}$$

where α is a scaling parameter and K is the number of occupied tables.

ssCRP. Although CRP can be applied for clustering in a non-parametric Bayesian fashion, it does not consider the similarity of relation instances. We observe that some extensions of CRP (e.g., ddCRP (Blei and Frazier, 2010)) leverage the distances between data points but they do not consider the iterative cluster generation process with pre-defined clusters. In this paper, we propose a similarity-based clustering algorithm especially designed for ERC, called similarity sensitive Chinese restaurant process (ssCRP).

In ssCRP, we turn the problem of table assignment into customer assignment. Unlike the CRP and ddCRP, we accommodate labeled data by initializing tables with K pre-defined classes. Customer i has the choice of sitting next to any customer j based on similarity, which leads to three cases: (i) i joins the table where j sits; (ii) i and an upcoming j generates a new table[2]; and (iii) i sits at an empty table alone. Case (i) is specifically introduced for task ERC due to the initialization of existing tables. Each table in case (i) can be represented as a virtual customer averaged from all its seated customers.

Denote $\eta = \{S, N, \alpha, \beta\}$ as the set of hyperparameters, where S is a similarity matrix between all customers, $N = (N_1, \cdots, N_K)$ is the size vector of existing K tables, α is the scaling parameter and β is a parameter balancing the weight of table size. The distribution of customer assignment c_i is:

$$\Pr(c_i = j \mid \eta) \propto \begin{cases} \alpha & \text{if } j \text{ is customer } i \text{ itself} \\ g(s_{ij}) & \text{if } j \text{ is an upcoming customer} \\ g(s_{ij})(1 + \beta \lg N_p) & \text{if } j \text{ is averaged from table } p \end{cases} \tag{2}$$

where g is a function modeling the cosine similarity s_{ij} between customer i and j. We design a magnifying function $g(s_{ij}) = -1/\ln s_{ij}$ where $s_{ij} > 0$, to increase the dissimilarity of inputs. The first two cases in Eq. (2) are derived from CRP. The third case gives existing tables additional weights to avoid generating too many small clusters. An alternative measuring system is to use the distance function $d_{ij} = 1 - s_{ij}$ and to apply different non-increasing functions to mediate the distances between customers. Given a parameter a, typical decay functions like window decay $f(d) = 1[d < a]$, exponential decay $f(d) = \exp(d/a)$ and logistic decay $f(d) = \exp(-d + a)/(1 + \exp(-d + a))$ are provided as alternatives (Blei and Frazier, 2010).

ssCRP-based relation discovery. The overall sampling and table selection process is illustrated in Fig. 4. In Step 1, we initialize fixed tables with classification results of labeled data. In Step 2, ssCRP draws customer assignments based on Gibbs sampling. Denote C_p as the collection of entity pairs w.r.t table p. Given the hyperparameter set η, the likelihood function of table p denoted by C_p is calculated as follows: $\Pr(C_p \mid \eta) = \prod_{i=1}^{|C_p|} \Pr(c_i^p \mid \eta)$ where c_i^p represents the customer assignment that customer i sits at table p.

[2]If the assignment of j is generated following i, then j is an upcoming customer w.r.t. i.

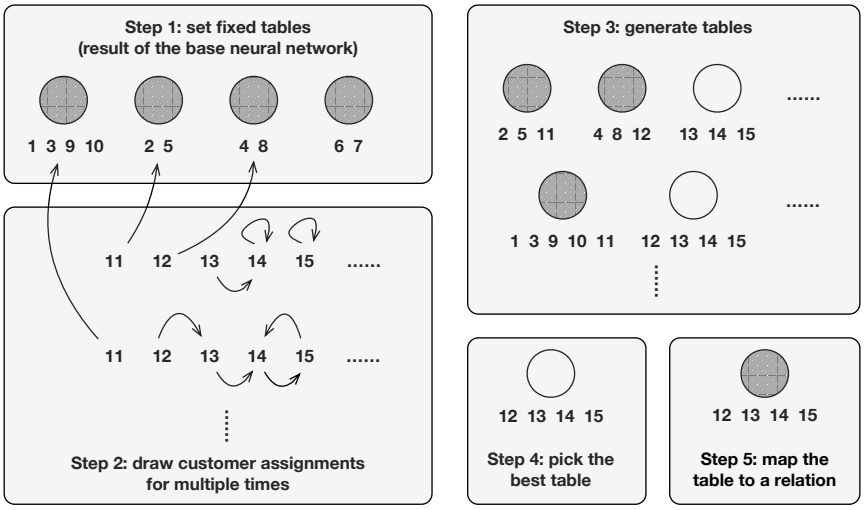


Figure 4: Relation discovery based on ssCRP. Circles with shadow are existing relations, while plain circles are new tables surrounded with customers representing entity pairs.

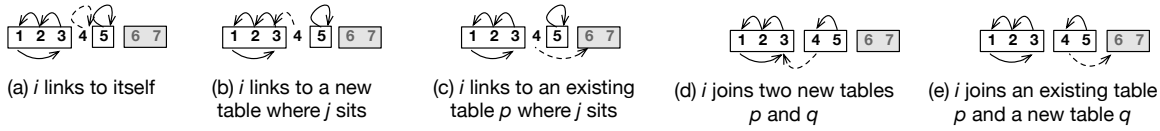


Figure 5: Five cases of customer assignment with $i = 4$ based on Gibbs sampling. Grey boxes are existing tables while white ones are generated new tables.

Given the rest of customer assignments $\mathbf{c}_{-i}$, customer assignment c_i can be divided into five cases (Blei and Frazier, 2010), generated by Gibbs sampling as Fig. 5 shows:

$$\Pr(c_i = j \mid \mathbf{c}_{-i}, \eta) \propto \begin{cases} \alpha & \text{(a)} \\ g(s_{ij}) & \text{(b)} \\ g(s_{ij})(1 + \beta \lg N_p) & \text{(c)} \\ g(s_{ij}) \frac{\Pr(C_p \cup C_q|\eta)}{\Pr(C_p|\eta)\Pr(C_q|\eta)} & \text{(d)} \\ g(s_{ij})(1 + \beta \lg N_p) \frac{\Pr(C_p \cup C_q|\eta)}{\Pr(C_p|\eta)\Pr(C_q|\eta)} & \text{(e)} \end{cases} \tag{3}$$

After this process, all customers are connected together with direct links from one to another or self-loops. New tables can be automatically generated by connected customers while existing tables are extended with new customers in Step 3. In our implementation, we run Gibbs sampling for multiple times to select the best configuration (i.e., largest likelihood), to enhance the fault tolerance.

Step 4 calculates a score for each table p considering pairwise similarities and table size, defined as:

$$score(p) = \frac{\lg N_p}{2\binom{N_p}{2}} \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} s_{ij}(i \neq j) \tag{4}$$

where $\frac{1}{2\binom{N_p}{2}} \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} s_{ij}(i \neq j)$ is the average pairwise similarity of instances in table p. $\lg N_p$ gives additional weight to large tables. It means the best table has similar relation instances and large size. The table with the highest $score(p)$ will be the selected cluster C^*. The last step (i.e., Step 5) maps the "best" cluster C^* to a semantic relation r^* with a proper relation predicate. The relation can be either a new one with confident seeds (i.e., entity pairs) or an existing relation extended with new entity pairs.

Predefined relations	执导 (directing)	演唱 (singing)	主演 (starring)	配偶 (spouse)
# Instances	633	648	1609	590

Table 1: The statistics of pre-defined relations extracted from (Fan et al., 2017).

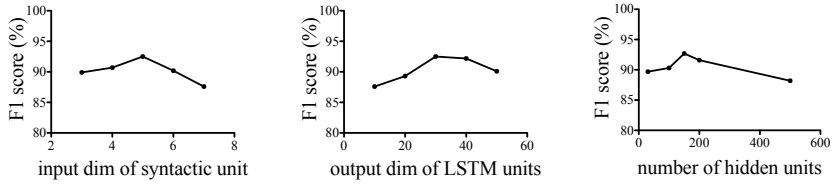


Figure 6: The effects of hyperparameters of the base neural network.

5.3 Relation Prediction

New relations generated via ssCRP only contain a small number of seed instances. To populate relations, part of unlabeled entity pairs will be assigned with labels based on model prediction, and added to the training set. To avoid error propagation, the algorithm requires that only if the prediction of DSNN is highly confident for an entity pair, can we add it to the training set.

Consider the distribution $[\Pr(r_1|e_1,e_2),\ldots,\Pr(r_{K+l}|e_1,e_2)]$ for entity pair (e_1,e_2). If it is "near uniform", none of the existing relation labels are confident enough to be the perfect fit. Goodness-of-fit tests are able to determine if the prediction follows uniform distribution (e.g. Kolmogorov-Smirnov test), but experiments show they do not work in this scenario (refer to Section 6.3). In this paper, we define a heuristic "Max-SecondMax" value to estimate the confidence score of a prediction:

$$conf(e_1,e_2) = \frac{\max([\Pr(r_1|e_1,e_2),\ldots,\Pr(r_{K+l}|e_1,e_2)])}{\text{secondMax}([\Pr(r_1|e_1,e_2),\ldots,\Pr(r_{K+l}|e_1,e_2)])} \tag{5}$$

where secondMax$(\cdot)$ is the second largest value of probabilities. If $conf(e_1,e_2)$ is no less than a given threshold τ, the label with the highest probability is indeed a convincing prediction. Such entity pairs will be added to labeled data together with their labels.

6 Experiments

In this section, we conduct extensive experiments for ERC task to evaluate our method and compare it with state-of-the-art approaches.

6.1 Datasets

We use distant supervision (Mintz et al., 2009) to create datasets. We choose four relations from an entertainment knowledge graph (Fan et al., 2017), and extract texts from Chinese Wikipedia where two entities from the same pair have joint occurrence. We manually check the correctness of relation labeling and remove annotation errors. In total, we have 3480 sentences with annotated entity pairs and relations, and spilt them into training data, testing data and validation data, with the proportion of 70%, 20% and 10%. The statistics of labeled data are summarized in Table 1. Similarly, two entities which do not appear in pairs of pre-defined relations are used for harvesting unlabeled data, which contains 3161 sentences.

6.2 Evaluation of Relation Classification

We implement base neural network with Keras[3] and use dependency parsing results generated by pyltp[4]. The word embeddings are initialized as 50 dimensions, trained on Chinese Wikipedia dump[5] via the Skip-gram model (Mikolov et al., 2013).

[3]https://github.com/fchollet/keras/tree/master/keras
[4]http://www.ltp-cloud.com/
[5]https://dumps.wikimedia.org/zhwiki/20170222

Classifier	Feature set	F1 (%)
logistic regression/ SVM	entity pairs (add)	77.3/ 77.4
	entity pairs (sub)	75.9/ 80.8
	entity pairs (concat)	89.0/ 87.5
	syntactic units, entity pairs (concat)	84.9/ 82.5
	context words, entity pairs (concat)	87.6/ 86.6
	syntactic units, context words	89.2/ 87.8
	syntactic units, context words, entity pairs (concat)	89.9/ 88.0
Shwartz et al. (Shwartz et al., 2016)	shortest dependency path, entity pairs	65.3
Zeng et al. (Zeng et al., 2014)	context words, entity pairs	81.5
RNN+E	syntactic units, entity pairs (concat)	66.8
CNN+E	context words, entity pairs (concat)	91.4
Full implementation	syntactic units, context words, entity pairs (concat)	**92.2**

Table 2: Classifiers with their feature sets and F1 score in relation classification.

Relation name	# Instances	Relation name	# Instances
团队成员 (group members)	1328	所属国家 (belong to the country)	956
家庭成员 (family members)	355	系列作品 (series works)	247
签约公司 (employed by)	144	制作公司 (produced by)	18

Table 3: Semantic relations discovered via ssCRP.

We first study the effects of hyperparameters, i.e., the input dimension of syntactic unit d_{syn}, the output dimension of LSTM units d_{lstm} and the number of convolutional hidden units h. We tune these hyperparameters on the validation set and illustrate the F1 scores with different settings in Fig. 6. As we can see, the best performance is achieved when $d_{syn} = 5$ and $d_{lstm} = 30$. The base neural network shows signs of overfitting with h larger than 150. We heuristically set the window size $k = 5$ and train the network with 0.5 weighted L_2 regularization.

The second part is to evaluate the effectiveness of the proposed features. We implement several state-of-the-art methods of representing the embeddings of entity pairs: concatenation $\vec{v}_{e_1} \oplus \vec{v}_{e_2}$, difference $\vec{v}_{e_1} - \vec{v}_{e_2}$ and sum $\vec{v}_{e_1} + \vec{v}_{e_2}$ model (Baroni et al., 2012; Roller et al., 2014; Mirza and Tonelli, 2016). We train logistic regression and SVM with the combination of the above features. As Table 2 shows, both classifiers achieve the highest F1 scores when trained with all three features. Similar to observations (Mirza and Tonelli, 2016), the concatenated embeddings of entity pairs are the most effective.

The third part is the comparison of our model and other approaches. We implement the CNN-based model (Zeng et al., 2014) and keep the way they use features. Another competitor is the RNN-based model for hypernymy detection (Shwartz et al., 2016), which is modified slightly to classify more generalized semantic relations. We also evaluate the variations of base neural network during iterations, e.g. removing LSTM units (CNN+E) or the convolution layer (RNN+E). The results shown in Table 2 prove that our model has the best performance. The CNN-based models such as CNN+E and model (Zeng et al., 2014) are both significantly effective than RNN-based models, e.g. RNN+E and model (Shwartz et al., 2016). It suggests that lexical features are more effective than syntactic features. The embedding layer improves the performance in either situation due to the use of semantic features.

6.3 Evaluation of Relation Discovery

We first introduce the semantic relations found during the relation discovery process. Table 3 summarizes six relations discovered via ssCRP, containing 3048 instances. The sizes of relations are unbalanced due to the random selection of entity pairs in order to construct unlabeled data automatically.

As studied in previous research (Qiu and Zhang, 2014), OpenRE systems have low performance for Chinese due to flexible language expressions and low data redundancy. Hence, OpenRE methods are not treated as strong baselines for ERC task. To measure the effectiveness of ssCRP, we propose two baseline models following Balvi et al. (Dalvi et al., 2013) for multi-class semi-supervised learning task. Seeded KMeans proposed by Basu et al. (2002) is a clustering method using labeled data to guide the clustering process. We implement an exploratory version where a new centroid is initialized as the most centered data in each iteration. Another intuitive approach is semi-supervised EM-based Naive Bayes with empty

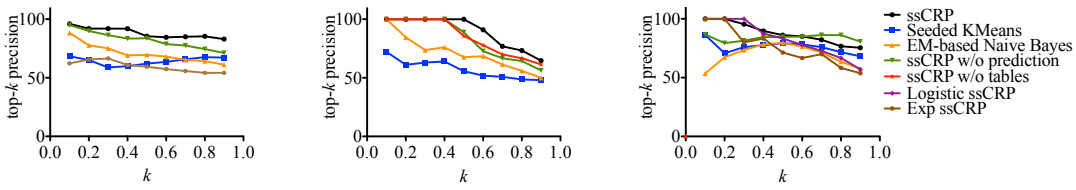


Figure 7: The top-k precisions (%) of three relations (series works, produced by, employed by) generated by various methods with different k.

Algorithm	# Instances	Precision (%)	Recall (%)	F1 (%)
Fit ssCRP	3161	31.0	35.7	33.2
Exploratory EM-based Naive Bayes	3161	70.7	40.2	52.8
Exploratory seeded KMeans	3161	80.5	53.0	63.9
ssCRP w/o tables	593	66.6	60.4	63.3
ssCRP w/o prediction	903	83.7	61.0	70.6
Exp ssCRP	3161	77.9	66.7	71.9
Logistic ssCRP	3161	81.4	66.9	73.0
Full implementation of ssCRP	3048	**83.1**	**68.4**	**75.0**

Table 4: Performance of different algorithms for relation discovery.

classes. The E-step starts with a random initialization of class assignment, and M-step retrains the model until convergence. We also try several variations of our model. For example, we replace the magnifying function with logistic decay (Logistic ssCRP) or exponential decay (Exp ssCRP) proposed in (Blei and Frazier, 2010). In relation prediction process, we make a comparison between the "Max-SecondMax" criterion and goodness-of-fit models such as Kolmogorov-Smirnov test with significance level of 0.05 (Fit ssCRP). Populating new clusters is essential for our model, so we conduct experiments of two related strategies, e.g. not allowing to join existing tables (ssCRP w/o tables) or removing the relation prediction process (ssCRP w/o prediction). We set $\tau = 2$ for all variations, fine tuned over the validation set.

The first experiment presents the top-k precision of newly found relations. Relation instances are sorted according to the cosine similarity between its embedding and averaged relation embedding. We ask human annotator to label whether the extracted top-k relations are correct or not, and evaluate the top-k precision with k ranging from 0.1 to 0.9, and show the results in Fig. 7. The illustrated relations achieve the best performance when they are generated by ssCRP, compared with other baseline models[6]. We heuristically choose $k = 0.4$ because the precision drops relatively faster when k is larger.

Next, we design a pairwise experiment to evaluate this non-standard clustering task. We manually construct a standard testing dataset by sampling pairs of instances from unlabeled data. For two entity pairs x_i and x_j with their respective sentences, we use the domain knowledge graph (Fan et al., 2017) as the ground truth to determine whether x_i and x_j have the same relation. We use Precision, Recall and F1 score as the evaluation metrics. We present the performance of ssCRP and other baseline models in Table 4. For two baseline models, exploratory seeded KMeans performs better than exploratory EM-based Naive Bayes. Experiments of ssCRP variations prove the effectiveness of our specially designed magnifying function and "Max-SecondMax" criterion. The reason that goodness-of-fit models fail is that they are sensitive to check whether the data follows uniform distribution or not, while our purpose is to select those with prominent peaks. Thus non-confident relation labels are assigned to unlabeled entity pairs roughly, increasing the number of false positive instances. For the strategies of table assignment and relation prediction process, the experimental results show that they not only populate new relations, but also improve the overall performance.

6.3.1 Error Analysis

To further obtain additional insights into our method, we study the errors in newly found relations and present three types of errors. A frequent type of confusion happens when entity pairs are closely related

[6]Some baselines do not generate certain relations, therefore these competitors are not included in corresponding figures.

under the same topic but not by the same relation, which accounts for 63.4% errors. For example, two entities involved with cooperation relation are be misclassified to the relation "团队成员 (group members)". Another 23.7% bad cases are due to the false positive results of NER where common nouns are mistaken as named entities, especially for specific names such as "黎明 (Ming Li, which also has the meaning of dawn)". The rest of errors result from the mixture of different grains of relation types. We observe that the relation "所属国家 (belong to the country)" contains a few instances where e_2 is not a country but a finer-grained city or a district.

7 Conclusion

In this paper, we propose the task of ERC to address the problem of domain-specific knowledge acquisition. We propose a DSNN model to address the task, consisting of three modules, an integrated base neural network for relation classification, a similarity-based clustering algorithm ssCRP to generate new relations and constrained relation prediction process with the purpose of populating new relations. Extensive experiments are conducted to evaluate the effectiveness of our approach.

Acknowledgements

This work is partially supported by the National Key Research and Development Program of China under Grant No. 2016YFB1000904.

References

David J Aldous. 1985. *Exchangeability and related topics*. Springer Berlin Heidelberg.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*, pages 2670–2676.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *EACL*, pages 23–32.

Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. 2002. Semi-supervised clustering by seeding. In *ICML*, pages 27–34.

David M. Blei and Peter I. Frazier. 2010. Distance dependent chinese restaurant processes. In *ICML*, pages 87–94.

Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *HLT/EMNLP*, pages 724–731.

Rui Cai, Xiaodong Zhang, and Houfeng Wang. 2016. Bidirectional recurrent convolutional neural network for relation classification. In *ACL*, pages 756–765.

Bhavana Bharat Dalvi, William W. Cohen, and Jamie Callan. 2013. Exploratory learning. In *ECML/PKDD*, pages 128–143.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. 2011. Open information extraction: The second generation. In *IJCAI*, pages 3–10.

Yan Fan, Chengyu Wang, Guomin Zhou, and Xiaofeng He. 2017. Dkgbuilder: An architecture for building a domain knowledge graph from scratch. In *DASFAA*, pages 663–667.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING*, pages 539–545.

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

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *ACL*, page 22.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL*, pages 1003–1011.

Paramita Mirza and Sara Tonelli. 2016. On the contribution of word embeddings to temporal relation classification. In *COLING*, pages 2818–2828.

Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom M. Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Likun Qiu and Yue Zhang. 2014. ZORE: A syntax-based system for chinese open relation extraction. In *EMNLP*, pages 1870–1880.

Desh Raj, Sunil Kumar Sahu, and Ashish Anand. 2017. Learning local and global contexts using a convolutional recurrent network model for relation classification in biomedical text. In *CoNLL 2017*, pages 311–321.

Carl Edward Rasmussen. 1999. The infinite gaussian mixture model. In *NIPS*, pages 554–560.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *HLT-NAACL*, pages 74–84.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *COLING*, pages 1025–1036.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *ACL*, pages 2389–2398.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *HLT-NAACL*, pages 534–539.

Chengyu Wang and Xiaofeng He. 2016. Chinese hypernym-hyponym extraction from user generated categories. In *COLING*, pages 1350–1361.

Chengyu Wang, Junchi Yan, Aoying Zhou, and Xiaofeng He. 2017. Transductive non-linear learning for chinese hypernym prediction. In *ACL*, pages 1394–1404.

Fei Wu and Daniel S. Weld. 2007. Autonomously semantifying wikipedia. In *CIKM*, pages 41–50.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *EMNLP*, pages 1785–1794.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344.

Qing Zhang and Houfeng Wang. 2017. Noise-clustered distant supervision for relation extraction: A nonparametric bayesian perspective. In *EMNLP*, pages 1809–1814.

Who is Killed by Police: Introducing Supervised Attention for Hierarchical LSTMs

Minh Nguyen[†] **and Thien Huu Nguyen**[#‡]
[†] Hanoi University of Science and Technology, Hanoi, Vietnam
[#] Montreal Institute for Learning Algorithms, University of Montreal, Canada
[‡] Department of Computer and Information Science, University of Oregon, USA
minh.nv142950@sis.hust.edu.vn,thien@cs.uoregon.edu

Abstract

Finding names of people killed by police has become increasingly important as police shootings get more and more public attention (police killing detection). Unfortunately, there has been not much work in the literature addressing this problem. The early work in this field (Keith et al., 2017) proposed a distant supervision framework based on Expectation Maximization (EM) to deal with the multiple appearances of the names in documents. However, such EM-based framework cannot take full advantages of deep learning models, necessitating the use of hand-designed features to improve the detection performance. In this work, we present a novel deep learning method to solve the problem of police killing recognition. The proposed method relies on hierarchical LSTMs to model the multiple sentences that contain the person names of interests, and introduce supervised attention mechanisms based on semantical word lists and dependency trees to upweight the important contextual words. Our experiments demonstrate the benefits of the proposed model and yield the state-of-the-art performance for police killing detection.

1 Introduction

We study the problem of police killing detection from text. The key challenge is to be able to take a person name in the pool of documents (corpus) and automatically decide whether the corresponding person is killed by police or not based on the textual evidences in the corpus. For instance, the following sentence describes the police-caused death of "*Micah Jester*": "*Old **Micah Jester** was fatally shot by APD officers.*". This problem has drawn much public attention recently; however, it has not been investigated adequately by the natural language processing (NLP) community. To our knowledge, the only NLP work for police killing recognition so far is by (Keith et al., 2017) who rely on machine learning models to perform the automatic detection.

It is challenging to apply the machine learning models in this case as identifying police killings from text is a relatively new problem in machine learning research with no available training datasets to supervise the models. The only sources of supervision on which we can rely for this problem are the current databases that record the names of the police-killed victims in the past. Among these databases, the Fatal Encounters[1] (FE) database has emerged as the most comprehensive database with a relatively large number of recorded victims (over 23,000 victims). In order to take advantage of this database, (Keith et al., 2017) employs distant supervision (Craven et al., 1999; Mintz et al., 2009) that extracts person names from a corpus and aligns them with the victim names in the database. The matched names are considered as corresponding to people killed by police (positive entities) while the non-matched names constitute the negative examples in a binary classification problem for names in police killing detection. As the name itself does not carry much information, each extracted name is associated with the set of sentences in which the name appears in the corpus. This set of sentences is called the sentence container for the corresponding name (person). The sentence containers along with the distant supervision labels

[1]This database has been produced by D. Brian Burghart and colleagues by manually reading millions of news headlines and ledes.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2277–2287
Santa Fe, New Mexico, USA, August 20-26, 2018.

(i.e, positive or negative) of the corresponding names would serve as the training data for the binary name classification problem.

An important characteristic of the sentence containers is that they might contain multiple sentences corresponding to the multiple appearances of the names in the corpus. Although all of these sentences mention the names of interests, some of them might not express police killing incidents. Consequently, it is crucial for the systems to be able to model the multiple sentences in the containers appropriately so that the correct sentences for police killing events can be captured to perform classification for the sentence containers of names. (Keith et al., 2017) solves this problem by introducing a latent variable for each sentence in the container of a name to predict whether the sentence describes the person as having been killed by police or not. Such latent variables are then modeled by sentence level classifiers (i.e, logistic regressions and convolutional neural networks) and aggregated for the final prediction for the name. (Keith et al., 2017) learns the parameters for the sentence classifiers via an Expectation Maximization (EM) based framework that alternates between estimating the latent variables and updating the model parameters. In (Keith et al., 2017), the authors demonstrate that the EM-based framework works well when the sentence level classifiers involve logistic regression with hand-crafted features. However, when deep learning models (i.e, convolutional neural networks) are employed for the sentence level classifiers, the performance of the EM-based framework drops significantly. We attribute this problem to the limitation of the EM-based framework to train the non-convex classifiers of deep learning, causing the inability to exploit the automatically learnt representations from deep learning and necessitating the use of complicated and laborious feature engineering.

In order to overcome this problem, we propose a novel deep learning framework for the problem of police killing recognition via hierarchical long short-term memory networks (LSTM) (Hochreiter and Schmidhuber, 1997; Yang et al., 2016). Our model does not make individual predictions on each sentence with the latent variables but involves a direct prediction to the name of interest based on its sentence container. Two layers of LSTMs are applied to model the sentence containers. The first LSTM layer learns the representations for the sentences in the containers by recurring over their words (the word level). The second LSTM layer, on the other hand, consumes the sentence representations (the sentence level) to produce container representations for police killing predictions. Attention mechanisms are then introduced into both LSTM layers to appropriately quantify the contribution of each word and sentence in the containers for the final predictions. This approach facilitates the modeling of multiple sentences in the containers, leading to the effective training of the deep learning models in a single framework and alleviating the reliance on manually designed features for this problem.

In the previous hierarchical LSTM models, attention scores are computed and normalized using the hidden representations of words and sentences generated by LSTMs (Yang et al., 2016). Unfortunately, for our problem of police killing recognition, this traditional attention computation tends to assign very high weight to the words in the names of interests and relatively low weights to the other important context words in the sentences. Such failure to adequately capture those context words would potentially lead to incorrect predictions for the containers. This problem is stem from the use of the position embeddings to specify the names of interests that might put too much emphasis on the current names. In order to solve this problem, we propose to integrate the supervised attention mechanisms into the hierarchical LSTM model that help to bias the attention scores toward the heuristically important words in the sentences (supervised attention) (Mi et al., 2016). In particular, we rely on linguistic intuitions to heuristically select the informative context words for the problem of police killing detection. These words are then used to guide the attention computation via penalizing the model parameters that generate low attention scores for such guidance words. We investigate several heuristics to choose the guidance words based on semantical word lists and dependency trees. The experiments show that the supervised attention mechanism with those heuristics helps to improve the performance of the hierarchical LSTM model and yield the state-of-the-art performance for the problem of detecting police killings. To the best of our knowledge, this is the first work that introduces supervised attention into the hierarchical LSTM models and employs semantical word lists and dependency trees to select guidance words.

2 Related Work

Although police killing recognition is a new task, it has some elements with the information extraction (IE) research of NLP that can be used to solve the task with some modifications. The most related IE task for police killing detection is event extraction that aims to detect events (i.e, marriage, attack, die etc.) in text (Li and Ji, 2014; Nguyen and Grishman, 2015b; Chen et al., 2015; Nguyen and Grishman, 2016b). Killings is one of the event types that the current event extraction systems can identify (Das et al., 2014; Li and Ji, 2014; Nguyen et al., 2016c), allowing the detection of police killing incidents when appropriate adaptations are introduced. Unfortunately, such adaptations result in poor performance for police killing recognition as shown in (Keith et al., 2017).

Distant supervision is another element of IE that is employed in this research to generate training data for police killing detection. In particular, distant supervision has been used to produce training data for relation extraction (Craven et al., 1999; Bunescu and Mooney, 2007; Mintz et al., 2009; Riedel et al., 2010; Surdeanu et al., 2012) and event extraction (Reschke et al., 2014).

Hierarchical deep learning techniques that model both the word and sentence levels have been employed for several NLP tasks, including relation extraction (Lin et al., 2016), question answering (Choi et al., 2017) and extractive summarization (Cheng and Lapata, 2016). Such work often uses convolutional neural networks to operate at the work level. This is different from our proposed model for police killing detection that employs LSTMs and supervised attentions to acquire sentence representations for police killing recognition. Perhaps the most related model to ours is (Yang et al., 2016) that utilizes hierarchical LSTMs for text categorization. Our model also relies on hierarchical LSTMs, but it is designed for police killing detection, characterizing position embeddings and supervised attentions to inject external knowledge (i.e, the heuristics for guidance words).

Finally, supervised attention mechanisms have been used recently for several natural language tasks. For machine translation, the attention guidance is based on word alignment (Mi et al., 2016; Liu et al., 2016) while entity mentions are chosen as the guidance words for event detection (Liu et al., 2017a). Our work in this paper is different as we consider supervised attention for police killing recognition using semantical word lists and dependency parsing trees (Schuster and Manning, 2016) to guide the attention components. Our model features hierarchical LSTMs to tackle distant supervision data that does not emerge in such prior work.

3 Model

We formalize the problem of finding people killed by police as follows.

Let D be a set of documents (corpus), $E = \{e_i\}_{i=1}^{N}$ be the set of entities (people) whose names appear in D (N is the number of the entities in E), and $C = \{c_i\}_{i=1}^{N}$ be the set of sentence containers for the entities in E (i.e, c_i is the set of sentences that contain the name of the entity e_i in D).

Each entity e_i in E has a label $y_{e_i} \in \{0, 1\}$, denoting whether e_i has been killed by police or not based on the distant supervision procedure (y_{e_i} is set to 1 if e_i is deemed to be killed by police via distant supervision and 0 otherwise). For convenience, let $Y = \{y_{e_i}\}_{i=1}^{N}$. Our goal is to use E, C and Y as training data to generate a model that can predict whether a new entity e is a victim of a police killing incident or not, given its sentence container c in some corpus. In machine learning, this essentially amounts to building models to estimate the probability $P(y_e = 1|c)$.

We will first introduce the hierarchical LSTM model for this problem, and then describe the supervised attention mechanisms with semantical word lists and dependency trees.

3.1 Hierarchical LSTMs

Each entity $e_i \in E$ along with its container $c_i \in C$ constitute an example for the model. Let $c_i = (s_{i,1}, s_{i,2}, \ldots, s_{i,L})$ be the list of sentences in c_i where L is the number of sentences and $s_{i,j}$ is the j-th sentence in the container c_i. Each sentence $s_{i,j}$ is in turn a word/token sequence: $s_{i,j} = (t_{i,j,1}, t_{i,j,2}, \ldots, t_{i,j,K})$ with K as the number of the tokens and $t_{i,j,k}$ as the k-th token in the sentence $s_{i,j}$. For each sentence $s_{i,j}$, let $k_{i,j}$ be the index of the name of the entity e_i (i.e, the token $t_{i,j,k_{i,j}}$). Note that the order of the sentences $s_{i,j}$ in c_i is obtained by sorting the sentences according to

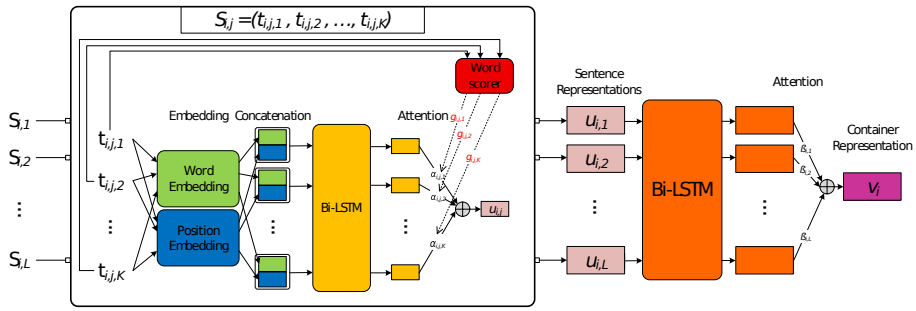


Figure 1: Hierarchical LSTMs with Supervised Attention for Police Killing Detection.

the download time of their corresponding documents in D^2. This helps to partially retain information about the order of being mentioned of the entities in the sentence containers in the corpus.

The hierarchical LSTM model in this work processes one entity e_i and its corresponding sentence container c_i at a time. For each entity e_i, the operation of the model can be divided into two main components: Embedding and Attention. Figure 1 shows an overview of the proposed model.

Embedding

In this component, each token $t_{i,j,k}$ of a sentence $s_{i,j}$ in c_i is transformed into an embedding vector $x_{i,j,k} = [q_{i,j,k}, p_{i,j,k}]$, which is the concatenation of the word embedding vector $q_{i,j,k}$ and the position embedding vector $p_{i,j,k}$ (Nguyen and Grishman, 2015a). These vectors are obtained as follows:

Word embedding vector: $q_{i,j,k}$ is obtained by taking the column vector corresponding to $t_{i,j,k}$ in the pre-trained word embedding matrix W_e (i.e, `word2vec` in our case): $q_{i,j,k} = W_e[t_{i,j,k}]$.

Position embedding vector: $p_{i,j,k}$ captures the relative distance $k - k_{i,j}$ from the token $t_{i,j,k}$ to the entity name token $t_{i,j,k_{i,j}}$ in the sentence: $p_{i,j,k} = W_d[k - k_{i,j}]$, the $(k - k_{i,j}) - th$ column in the position embedding matrix W_d (W_d is randomly initialized in this work) (Nguyen and Grishman, 2018).

Once each token $t_{i,j,k}$ has been transformed into a vector, the corresponding input sentence $s_{i,j}$ would become a sequence of vector $(x_{i,j,1}, x_{i,j,2}, \ldots, x_{i,j,K})$. This allows us to view the container c_i as an ordered list of vector sequences for its sentences $(s_{i,1}, s_{i,2}, \ldots, s_{i,L})$.

Attention

The attention component processes the list of vector sequences produced in the previous step for c_i at two levels (i.e, the word level and the sentence level) to produce a single representation vector for c_i.

The word level component consumes each vector sequence of c_i once at a time to compute the representation vector for the corresponding sentence. For a vector sequence $(x_{i,j,1}, x_{i,j,2}, \ldots, x_{i,j,K})$ of $s_{i,j}$, the model architecture consists of two LSTMs (Hochreiter and Schmidhuber, 1997) (i.e, the forward LSTM and the backward LSTM) that operate over two different directions of the vector sequence (bidirectional LSTMs) (Cho et al., 2014; Nguyen et al., 2016a). The resulting hidden vector sequences of the forward and backward LSTMs are concatenated at each position, generating the hidden vector sequence $(h_{i,j,1}, h_{i,j,2}, \ldots, h_{i,j,K})$ for the input vector sequence of $s_{i,j}$.

In order to combine the hidden vectors $h_{i,j,k}$, the attention mechanism computes the weighted sum of the hidden vectors to obtain a single representation vector for the input sentence $s_{i,j}$ (Bahdanau et al., 2015). Specifically, each hidden vector $h_{i,j,k}$ is given a weight $\alpha_{i,j,k}$ to estimate its contribution for the final representation of the container c_i for the problem of police killing recognition. In this work, the weight $\alpha_{i,j,k}$ is computed by:

$$\alpha_{i,j,k} = \frac{\exp(a_{i,j,k}^{\top} w_a)}{\sum_{k'} \exp(a_{i,j,k'}^{\top} w_a)} \quad (1)$$

where:

$$a_{i,j,k} = \tanh(W_{att} h_{i,j,k} + b_{att}) \quad (2)$$

[2]Such documents are retrieved via running pre-defined search queries once per hour throughout 2016 (Keith et al., 2017).

In such equations, W_{att}, b_{att} and w_a are the attention parameters at the word level that are learnt in the training process. Eventually, the representation vector $u_{i,j}$ of the sentence $s_{i,j}$ is:

$$u_{i,j} = \sum_k \alpha_{i,j,k} h_{i,j,k} \tag{3}$$

Once the word level component has been completed, every sentence $s_{i,j}$ in the container c_i would have a corresponding representation vector $u_{i,j}$. Such sentence representation vectors $u_{i,j}$ altogether form a new sequence of vector $(u_{i,1}, u_{i,2}, \ldots, u_{i,L})$ to represent the container c_i. At the sentence level, $(u_{i,1}, u_{i,2}, \ldots, u_{i,L})$ is processed in the same way we process the vector sequence $(x_{i,j,1}, x_{i,j,2}, \ldots, x_{i,j,K})$ at the word level to produce the sentence representation vector $u_{i,j}$. In particular, $(u_{i,1}, u_{i,2}, \ldots, u_{i,L})$ would be first passed to a bidirectional LSTM model to obtain the hidden vector sequence $(h_{i,1}, h_{i,2}, \ldots, h_{i,L})$. This is in turn fed into the attention component to obtain the attention weights $(\beta_{i,1}, \beta_{i,2}, \ldots, \beta_{i,L})$ (i.e, similar to Equations 1 and 2). Finally, we compute the vector representation v_i for the sentence container c_i via the weighted sum: $v_i = \sum_j \beta_{i,j} h_{i,j}$. As the sentences in c_i are sorted by their appearance time, the attentional bidirectional LSTMs at the sentence level attempt to estimate the contribution of each sentence $s_{i,j}$ in c_i for the representation vector v_i with respect to its past and future context information (i.e, the sentences before and after $s_{i,j}$ in c_i).

The container representation vector v_i for c_i allows us to compute the probability P_i of e_i being killed by police: $P_i = P(y_{e_i} = 1 \mid c_i) = \sigma(W_{out} v_i + b_{out})$ where W_{out} and b_{out} are the model parameters, and σ is the logistic function. In order to train the hierarchical LSTM model in this section, we use the cross-entropy between the predicted labels and the golden labels as the loss function:

$$L_c = -\sum_{e_i} y_{e_i} \log(P_i) + (1 - y_{e_i}) \log(1 - P_i) \tag{4}$$

3.2 Supervised Attention

The position embeddings $p_{i,j,k}$ in the initial representation vectors of the tokens is crucial to the hierarchical LSTM model as it helps to indicate the positions of the entity names of interests in the sentences. Technically, the position embeddings would tell the model to pay more attention to the words in the entity names by assigning higher attention weights to the representation vectors of the entity name tokens $h_{i,j,k_{i,j}}$ at the word level. Unfortunately, in the experiments, we find that this procedure might lead to extremely high weights for the tokens of the entity names, leaving essentially negligible weights for the other important context words in the sentences. The consequence is the incorrect predictions of the model for the entities in such situations. In order to solve this problem, in this work, we seek to use linguistic intuitions to obtain the rough estimations of the attention weights for the words in the sentences (intuitive attention weights), quantifying our belief about the importance of the words in the sentences for the problem of police killing recognition. The intuitive attention weights would then be used to guide the attention weights computed by the hierarchical LSTM model at the word level (i.e, Equation 1), penalizing the model parameters with significant difference between the two types of attention weights.

Formally, for an entity e_i with the sentence container c_i, suppose that we can obtain the intuitive attention weights $(g_{i,j,1}, g_{i,j,2}, \ldots, g_{i,j,K})$ for the tokens of every sentence $s_{i,j} = (t_{i,j,1}, t_{i,j,2}, \ldots, t_{i,j,K})$ in c_i ($\sum_k g_{i,j,k} = 1 \forall i, j$). The difference between the intuitive attention weights and the model attention weights in Equation 1 for e_i can be computed via the sum of the squared element difference: $L_i = \sum_{j,k} (g_{i,j,k} - \alpha_{i,j,k})^2$.

Our goal is to minimize this difference so that the model attention weights can encode our intuition about the importance of the words in the sentences. This essentially translates into an integrated loss function to train the hierarchical LSTM model, attempting to minimize the loss function in Equation 4 and the attention weight difference simultaneously: $L = L_c + \lambda \sum_i L_i$ where λ is a penalty coefficient to control the effect of the attention difference.

Generating Intuitive Attention Weights

The previous section has described the supervised attention mechanism for the hierarchical LSTM model. It remains to investigate the methods to obtain the intuitive attention weights. An important characteristics of the intuitive attention weights is that they should assign high weights to the linguistically important

words for police killing recognition. In this work, we first start by selecting the important words in the sentences based on our intuition. Afterward, non-negative scores are given to the selected words and their neighbors in the sentences, leaving zero scores for other words. These scores are then normalized to ensure a sum of 1. In order to generate the non-negative scores for the selected words and their neighbors, we employ a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$(Mi et al., 2016; Liu et al., 2017b) with the mean μ and the standard deviation σ so that the closer neighboring words would receive higher scores.

Selecting Important Words

This section presents two methods to select important words in the sentences of the containers for the problem of police killing detection. The first method relies on the semantical aspect while the second method concerns the syntactical heuristics. We will compare these methods in the experiments.

Semantical Aspect: The two concepts that are most related to our problem of police killing recognition would naturally be "*police*" and "*killing*". It is therefore intuitive to consider these two words ("*police*" and "*killing*") and their similar ones as the important words for our problem. Consequently, for every sentence in the containers, we search for such important words and use the matched word as the selected words. Following (Keith et al., 2017), we generate the lists of similar words for "*police*" and "*killing*" by looking up the nearest words in cosine distance via the word vectors pre-trained with `word2vec` and its corpus. Note that this method also includes the names of the entities e_i in the list of selected important words due to their roles in specifying the entities of interests in the sentences.

Syntactical Aspect: The semantical aspect with the lists of similar words for police killing detection might be helpful for the sentences that express a police killing instance (positive sentences) and contain the similar words. However, for the negative sentences that do not mention police killing incidents, the appearance of the similar words in the lists for "*killing*" and "*police*" might be harmful to the supervised attention mechanisms as the emphasis on such words might lead to an incorrect impression to consider the sentence as actually positive. For instance, consider the following negative sentence with the words in the similar word lists written in bold[3]:

*Marion **Police** Department have arrested **TARGET** , 20 , of Marion , in connection with the **fatal** hit.*

In this sentence, the extreme emphasis on "*Police*", "*TARGET*" and "*fatal*" might lead to the incorrect prediction that this sentence is expressing a fatal event caused by police. In order to overcome this issue, we observe that police killing recognition can be seen as a relation identification problem (Bunescu and Mooney, 2005), attempting to decide whether the entities of interests (i.e, the "*TARGET*") has a semantical relation of "*killed by*" with the similar words of "*police*" (if any) in the sentence or not. In such relation identification problem, it has been shown that the shortest dependency path connecting the two word of interests (i.e, the words "*TARGET*" and "*police*" in our case) in the dependency trees involve the most important context words for the problem (Bunescu and Mooney, 2005). Consequently, in this work, we propose to select the words along the shortest dependency paths between the entity name of interests (i.e, "*TARGET*") and the similar words of "*police*" in the sentence as the guidance words for the supervised attention mechanism for police killing recognition (dependency tress are obtained via Stanford CoreNLP (Schuster and Manning, 2016) in this work).

In the example sentence above, the shortest dependency path between "*TARGET*" and "*police*" is: *Police* $\rightarrow$ *Department* $\rightarrow$ *arrested* $\rightarrow$ *TARGET*. It is clear in this situation that the words along the path do not suggest a police-caused killing (i.e, not a "*killed by*" relation between "*TARGET*" and "*Police*"). Consequently, the attention of the models to such words would lead to a correct prediction in this case.

On the other hand, for the positive sentences, it might be the case that the words along the dependency paths help to include some words that are crucial to police killing prediction, but do not appear in the semantical word lists.

Baselines

For experimental purposes, we call the hierarchical LSTM model without supervised attention as "*H-LSTM*". The hierarchical LSTM model with the supervised attention mechanism would then be called

[3]Throughout this work, the names of the entities of interests would be replaced by "*TARGET*" while the other entity mentions in the sentences would be substituted by "*PERSON*" for generalization.

"*H-LSTM+SemAtt*" and "*H-LSTM+SynAtt*" depending on whether the semantical aspect (i.e, the lists of similar words) or the syntactical aspect (i.e, the dependency paths) is used to obtain the important words. In order to further demonstrate the benefits of supervised attention for police killing detection, we consider two more baseline models when the model attentions at the word level are excluded from "*H-LSTM+SemAtt*" and "*H-LSTM+SynAtt*", resulting in the models "*Mean+SemAtt*" and "*Mean+SynAtt*". In particular, in such models, the sentence representation vector $u_{i,j}$ for the sentence $s_{i,j}$ would not be obtained via the attention-based weighted sum in Equation 3. In contrast, $u_{i,j}$ would be computed using the mean vector of the vector set $\{h_{i,j,k_1}, h_{i,j,k_2}, \ldots, h_{i,j,k_I}\}$ from LSTMs where $k_1, k_2, \ldots, k_I$ are the indexes of the selected important words using the semantical or syntactical aspect for the sentence $s_{i,j} = (t_{i,j,1}, t_{i,j,2}, \ldots, t_{i,j,K})$: $u_{i,j} = 1/I \sum_{m=1}^{I} h_{i,j,k_m}$.

3.3 Training

We train all the models in this work using stochastic gradient descent with shuffled mini-batches, the Adam update rule, back-propagation and dropout. Non-embedding weights are also imposed to gradient clipping to rescale their $l2$-norms if they exceed a predefined threshold.

4 Experiments

4.1 Datasets and Parameters

We evaluate the models in this paper using the police killing dataset released by (Keith et al., 2017).

As there is no development data for this dataset, we divide the original training data into two parts, for which one part is used for training data while the other part functions as the development data. We use these newly-generated training data and development data to select the parameters for the models. For the comparison with the state-of-the-art models in (Keith et al., 2017), we utilize the original training data and test data with the chosen parameters from the development experiments to ensure a compatible comparison. We use the same procedure to split the original training data for development as those employed by (Keith et al., 2017) to generate the original dataset. In particular, we first sort all the positive entities in the original training data using the descending order of their death times. Afterward, we identify the death time of the entity at the bottom of the top 20% in the sorted list. The date we found is used as the split point. All the entities with the download time or death time after this date are utilized as the development data.

The parameters we found in the development experiments are as follow. The dimensionality parameters include: 8 dimensions for position embedding vectors, 300 dimensions for word embedding vectors, 256 hidden units for the LSTMs and 64 dimensions for the attention vectors. For supervised attention, the penalty coefficient λ is set to 1.0 while the neighbor window T for generating intuitive attention weights is 2. The threshold for gradient clipping is set to 5.0 while the dropout rate is 0.5. The mean and standard deviation of the Gaussian distribution have the values of $\mu = 0$ and $\sigma = 1.0$, respectively.

4.2 Evaluating the Models

We evaluate the models (i.e, *H-LSTM*, *H-LSTM+SemAtt*, *H-LSTM+SynAtt*, *Mean+SemAtt* and *Mean+SynAtt*) using the generated development data. Table 1 reports the performance.

Models	Precision	Recall	F1
H-LSTM+SynAtt	0.497	0.381	**0.431**
H-LSTM+SemAtt	0.366	0.504	0.424
H-LSTM	0.460	0.377	0.414
Mean+SynAtt	0.428	0.372	0.398
Mean+SemAtt	0.346	0.399	0.371

Table 1: Performance of the models on the development data. The comparisons in this table are significant with $p < 0.05$.

There are three major observations from the table. First, the performance of the baseline models *Mean+SynAtt* and *Mean+SemAtt* are much worse than the other models, demonstrating that the atten-

tions in Equation 1 are crucial to problem of police killing detection. Second, both *H-LSTM+SemAtt* and *H-LSTM+SynAtt* are better than *H-LSTM* (improvements of 1% and 1.7% on the absolute F1 scores for *H-LSTM+SemAtt* and *H-LSTM+SynAtt* respectively). This shows the benefits of the proposed supervised attention mechanisms with semantical and syntactical guidance in this work. Third, we see that *H-LSTM+SynAtt* outperforms *H-LSTM+SemAtt*, suggesting that the syntactical guidance with dependency trees are more effective than the semantical guidance with the word lists for supervised attention in our task. We also see that the recall of *H-LSTM+SemAtt* is much better than that of *H-LSTM+SynAtt*. We attribute this phenomenon to the fact that the dataset in (Keith et al., 2017) involves many sentences with the words in the similar words lists for "*police*" and "*killing*". This biases the supervised attention in *H-LSTM+SemAtt* to associate the appearance of such words with the positive entities and leads to the high recall for this method. Due to the poor performance of *Mean+SynAtt* and *Mean+SemAtt* in this development experiment, we only consider the other models (i.e, *H-LSTM*, *H-LSTM+SemAtt* and *H-LSTM+SynAtt*) in the following experiments.

4.3 Comparing to the State of the Art

This section compares our proposed models with the state-of-the-art models for police killing recognition. Such state-of-the-art models include *soft-RL* and *soft-CNN* that both apply the Expectation Maximization algorithm, but employ logistic regression and convolutional neural networks (respectively) for sentence classifiers (Keith et al., 2017). Table 2 shows the performance the models. Note that the performance in this section is obtained using the original training data and test data in (Keith et al., 2017).

As we can see from the table, the conclusions we have for the models *H-LSTM*, *H-LSTM+SemAtt* and *H-LSTM+SynAtt* in the previous section still hold in this case on the test data, thus further confirming those observations for police killing recognition. We also see that although *H-LSTM* does not use supervised attention, its performance is comparable with the best model *soft-LR* in (Keith et al., 2017). This is significant as *H-LSTM* does not employ any hand-crafted features while *soft-LR* needs to resort to complicated hand-designed features to perform well. The best performance is achieved with the *H-LSTM+SynAtt* model with an improvement of 3.3% in the absolute F1 measure over the best model *soft-LR* in (Keith et al., 2017). This testifies to the effectiveness of our proposed model in this work, featuring hierarchical LSTMs, supervised attention and syntactical guidance.

4.4 Analysis

In order to demonstrate the effectiveness of supervised attention and syntactical guidance for police killing detection, this section visualizes the attention weights $\alpha_{i,j,k}$ in Equation 1 for the words in several sentences in the test data.

The Effect of Supervised Attention

Figure 2 indicates the attention weights computed by the models *H-LSTM*, *H-LSTM+SemAtt* and *H-LSTM+SynAtt* for the words in an example sentence. This sentence corresponds to an entity in the test set that is correctly predicted (as being killed by police) by *H-LSTM+SynAtt*, but is incorrectly predicted by *H-LSTM* and *H-LSTM+SemAtt*. As we can see from the figure, *H-LSTM* fails in this case as it reserves a very high weight for the "*TARGET*" and essentially ignores the other words. This phenomenon is quite popular for *H-LSTM* and demonstrates the needs for supervised attention mechanisms as being motivated in the previous sections. In addition, *H-LSTM+SemAtt* cannot use this sentence as an evidence to make a

Models	Precision	Recall	F1	AUC
H-LSTM+SynAtt	0.442	0.288	**0.349**	**0.211**
H-LSTM+SemAtt	0.342	0.325	0.333	0.199
H-LSTM	0.419	0.259	0.320	0.194
soft-LR (EM)	0.447	0.243	0.316	0.193
soft-CNN (EM)	0.268	0.265	0.267	0.164

Table 2: Performance comparison on test data. AUC is the area under the precision/recall curve. The comparison in this table is significant with $p < 0.05$.

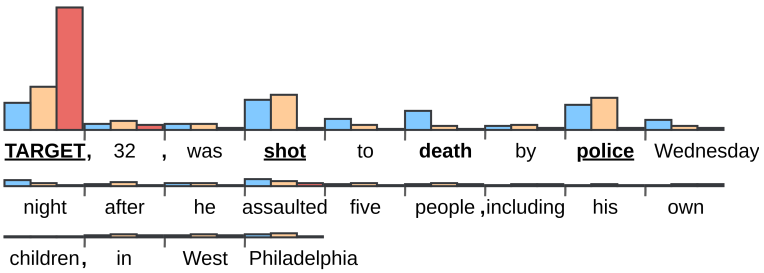


Figure 2: Attention weight visualization. The underlined and bold words are important words selected by *H-LSTM+SemAtt* and *H-LSTM+SynAtt* respectively. The blue, orange and red columns represents the word attention weights computed by *H-LSTM+SynAtt*, *H-LSTM+SemAtt*, and *H-LSTM* respectively. Weights of punctuations are not shown.

correct prediction in this case as it mainly attends to the words in the similar word lists (i.e,"*TARGET*", "*shot*" and "*police*") and misses the word "*death*". This is undesirable as "*death*" is the only clue showing that the victim of the shooting in this sentence is actually dead. *H-LSTM+SynAtt* is successful in this case as it is able to assign high weights to such important words along the dependency paths. This demonstrates our arguments in Section 3, showing the benefits of *H-LSTM+SynAtt* to suggest important words that cannot be captured by *H-LSTM+SemAtt* for police killing recognition.

Semantical vs. Syntactical Guidance

The previous part has shown the advantages of *H-LSTM+SynAtt* over *H-LSTM+SemAtt* for positive entities. This section focuses on the benefits of *H-LSTM+SynAtt* for the negative entities. Figure 3 illustrates the attention weights that *H-LSTM+SemAtt* and *H-LSTM+SynAtt* assign to the words of an example sentence in the test data.

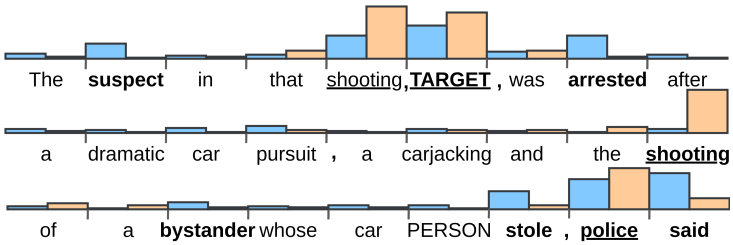


Figure 3: Attention weight visualization. The conventions in Figure 2 do apply here.

The entity of this sentence is negative that has been correctly recognized by *H-LSTM+SynAtt*, but has been incorrectly predicted by *H-LSTM+SemAtt*. As suggested in the figure, the failure of *H-LSTM+SemAtt* is due to its very high weights on "*shooting*", "*TARGET*" and "*police*", ignoring the effect of the words "*said*" and "*arrested*" that clearly negate the involvement of police in this shooting. *H-LSTM+SynAtt* can attend to such important words as they belong to the dependency paths between "*police*" and "*TARGET*" in this case.

5 Conclusions

We propose a novel deep learning model for the problem of police killing recognition. The proposed model involves hierarchical LSTMs to model the multiple sentences in the sentence containers of the entities. We introduce novel supervised attention mechanisms based on semantical and syntactical aspects for this problem. The experimental results demonstrate the effectiveness of the proposed models and lead to the state-of-the-art performance for police killing detection. In the future, we plan to apply the proposed method in a real system and extend it to other types of events (e.g, protests, epidemics).

References

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

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *EMNLP*.

Razvan Bunescu and Raymond Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *ACL*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL-IJCNLP*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *ACL*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In *ACL*.

Mark Craven, Johan Kumlien, et al. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, volume 1999, pages 77–86.

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. 2014. Frame-semantic parsing. *Computational linguistics*, 40(1):9–56.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. In *Neural Computation*.

Katherine A Keith, Abram Handler, Michael Pinkham, Cara Magliozzi, Joshua McDuffie, and Brendan O'Connor. 2017. Identifying civilians killed by police with distantly supervised entity-event extraction. *arXiv preprint arXiv:1707.07086*.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *ACL (1)*, pages 402–412.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *ACL*.

Lemao Liu, LemLiu, Masao Utiyama, Andrew Finch, ao Sumita, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. *arXiv preprint arXiv:1609.04186*.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017a. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1789–1798.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017b. Exploiting argument information to improve event detection via supervised attention mechanisms. In *ACL*.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. *arXiv preprint arXiv:1608.00112*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2015a. Relation extraction: Perspective from convolutional neural networks. In *The NAACL Workshop on Vector Space Modeling for NLP (VSM)*.

Thien Huu Nguyen and Ralph Grishman. 2015b. Event detection and domain adaptation with convolutional neural networks. In *ACL-IJCNLP*.

Thien Huu Nguyen and Ralph Grishman. 2016b. Modeling skip-grams for event detection with convolutional neural networks. In *EMNLP*.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *AAAI*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016a. Joint event extraction via recurrent neural networks. In *NAACL*.

Thien Huu Nguyen, Lisheng Fu, Kyunghyun Cho, and Ralph Grishman. 2016c. A two-stage approach for extending event detection to new types via neural networks. In *Proceedings of the 1st ACL Workshop on Representation Learning for NLP (RepL4NLP)*.

Kevin Reschke, Martin Jankowiak, Mihai Surdeanu, Christopher D. Manning, and Daniel Jurafsky. 2014. Event extraction using distant supervision. *Language*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. *Machine learning and knowledge discovery in databases*, pages 148–163.

Sebastian Schuster and Christopher D Manning. 2016. Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In *LREC*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *EMNLP*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. 2016. Hierarchical attention networks for document classification. In *HLT-NAACL*, pages 1480–1489.

Open Information Extraction from Conjunctive Sentences

Swarnadeep Saha*
IBM Research - India
swarnads@in.ibm.com

Mausam
Indian Institute of Technology, Delhi
mausam@cse.iitd.ac.in

Abstract

We develop CALM, a coordination analyzer that improves upon the conjuncts identified from dependency parses. It uses a language model based scoring and several linguistic constraints to search over hierarchical conjunct boundaries (for nested coordination). By splitting a conjunctive sentence around these conjuncts, CALM outputs several simple sentences. We demonstrate the value of our coordination analyzer in the end task of Open Information Extraction (Open IE).

State-of-the-art Open IE systems lose substantial yield due to ineffective processing of conjunctive sentences. Our Open IE system, CALMIE, performs extraction over the simple sentences identified by CALM to obtain up to 1.8x yield with a moderate increase in precision compared to extractions from original sentences.

1 Introduction

Open Information Extraction (Open IE) (Etzioni et al., 2008) extracts relational tuples from text in an unsupervised domain-independent manner, by identifying relational phrases and arguments from the sentences themselves. Recent work (Saha et al., 2017) has highlighted the lack of proper conjunction processing as the most significant source of missed yield in Open IE. We found Open IE 4.2 (Christensen et al., 2011; Pal and Mausam, 2016) and ClausIE (Corro and Gemulla, 2013) to frequently miss important extractions due to conjunctive relation phrases (see Table 1), and occasionally output conjunctive arguments, which are not ideal for readability or downstream applications (Angeli et al., 2015; Stanovsky et al., 2016a).

Most modern Open IE systems process dependency parses to obtain extractions. However, dependency parsers frequently make errors in resolving coordination ambiguity. Predicting the correct conjunct span is considered to be one of the biggest challenges for parsers (Ficler and Goldberg, 2016). The state of the art approach by Ficler and Goldberg (2016) trains an LSTM-based network for predicting the boundaries for the two conjuncts on either side of the coordinating conjunction, but does not handle cases where a conjunction coordinates more than two conjuncts.

Contributions: We propose a novel coordination analyzer called CALM (**C**oordination **A**nalyzer using **L**anguage **M**odel), which *corrects* the typical errors made by dependency parsers (specifically Clear parser, which is used in Open IE 4.2) using additional features and linguistic constraints. Under the intuition that one can split a conjunction to form multiple coherent simple sentences (see Table 2), CALM scores each simple sentence using a (modified) language model. It additionally employs several linguistic constraints to reduce errors further. An important linguistic constraint is that multiple coordination structures in a sentence must either be disjoint or completely nested. A key contribution of our work is operationalizing this constraint through a novel *hierarchical coordination tree*, which is helpful in sentences with multiple coordinations. Experiments on 577 conjunctive sentences in British News Corpus demonstrate that CALM improves upon the conjuncts from the parser, with significant benefits for sentences with multiple coordinations.

*Most work was done when Swarnadeep Saha was a graduate student at Indian Institute of Technology, Delhi.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2288–2299
Santa Fe, New Mexico, USA, August 20-26, 2018.

"Gates, an American investor and co-founder of Microsoft, stepped down as CEO of Microsoft in January 2000, but remained as chairman and created the position of chief software architect for himself and transferred his duties to Ray Ozzie and Craig Mundie."	
Extraction	**Systems**
1. (Gates; stepped down as; CEO of Microsoft)	[Cm, O4, C]
2. (Gates; stepped down as CEO of Microsoft; in January 2000)	[Cm, O4]
3. (Gates; is; an American investor)	[Cm]
4. (Gates; is an investor from; United States)	[Cm, O4]
5. (Gates; is co-founder of; Microsoft)	[Cm]
6. (Gates; is; an American investor and co-founder of Microsoft)	[C]
7. (Gates; remained as; chairman)	[Cm, O4, C]
8. (Gates; created; the position of chief software architect for himself)	[Cm, O4, C]
9. (Gates; transferred; his duties)	[Cm]
10. (Gates; transferred his duties to; Ray Ozzie)	[Cm]
11. (Gates; transferred his duties to; Craig Mundie)	[Cm]
12. (His; has; duties)	[C]
13. (Gates; transferred his duties to Ray Ozzie; the position of chief software architect for himself)	[C]
14. (Gates; transferred his duties to Craig Mundie; the position of chief software architect for himself)	[C]

Table 1: Comparison of extractions of different systems on a conjunctive sentence. [Cm] : CALMIE(O) , [O4] : Open IE4.2, [C] : ClausIE. Green = correct, Red = incorrect.

Gates, an American investor, stepped down as CEO of Microsoft in January 2000.
Gates, co-founder of Microsoft, stepped down as CEO of Microsoft in January 2000.
Gates remained as chairman.
Gates created the position of chief software architect for himself.
Gates transferred his duties to Ray Ozzie.
Gates transferred his duties to Craig Mundie.

Table 2: Simple sentences generated by our system for the conjunctive sentence in Table 1.

We use CALM's output for the goal of improving Open IE. Our system, CALMIE, splits a sentence into multiple simple sentences for each distributive coordination and pass those to two state-of-the-art Open IE systems, Open IE 4.2 and ClausIE. We call them CALMIE(O) and CALMIE(C) respectively. We evaluate on two different datasets and find that CALMIE(O) and CALMIE(C) always outperforms Open IE 4.2 and ClausIE respectively. CALMIE(O) particularly achieves 1.8x the yield with a 5 pt precision gain against Open IE 4.2 on a random sample of 100 conjunctive sentences from ClueWeb. Finally, we compare CALM with Ficler's system. On the subset of cases where a conjunction coordinates more than two conjuncts, our methods significantly outperform Ficler's, on an Open IE evaluation. We release our implementations of CALM, sentence splitter and CALMIE(O)[1] for further research.

2 Related Work

Open Information Extraction: Mausam (2016) surveys the progress in Open IE systems and its downstream applications (e.g., (Christensen et al., 2014; Stanovsky et al., 2015)). Existing Open IE systems are based on manually written patterns (Etzioni et al., 2011; de Sá Mesquita et al., 2013; Angeli et al., 2015), machine-learned patterns over bootstrapped training data (Mausam et al., 2012), sentence restructuring and decomposition (Bast and Haussmann, 2013; Schmidek and Barbosa, 2014), tree kernels (Xu et al., 2013) and simple inference (Bast and Haussmann, 2014). Some systems focus on specific kinds of extractions, e.g., noun-mediated (Pal and Mausam, 2016), numerical (Saha et al., 2017) and nested (Bhutani et al., 2016). We compare our methods to Open IE 4.2 (a combination of SRL-based IE (Christensen et al., 2011) and Relnoun (Pal and Mausam, 2016)) and ClausIE (Corro and Gemulla, 2013) – these outperformed others in a recent large-scale evaluation (Stanovsky and Dagan, 2016). ClausIE can also split some conjunctive clauses, thereby obtaining a higher yield.

There hasn't been any system with a specific focus on conjunctive sentences, which are recently found to be a reason for significant missed recall (Saha et al., 2017). While some of the missed recall is because of conjunctions appearing in arguments of extractions (extraction #6 in Table 1), which can be separated by reducing the argument spans (Stanovsky et al., 2016b), there are also important parts of the sentence which do not produce any extraction from either Open IE 4.2 or ClausIE (extraction #9, #10, #11 in

[1] CALMIE(O) is integrated into OpenIE 5.0, the latest software of OpenIE. It is publicly available at `https://github.com/dair-iitd/OpenIE-standalone`.

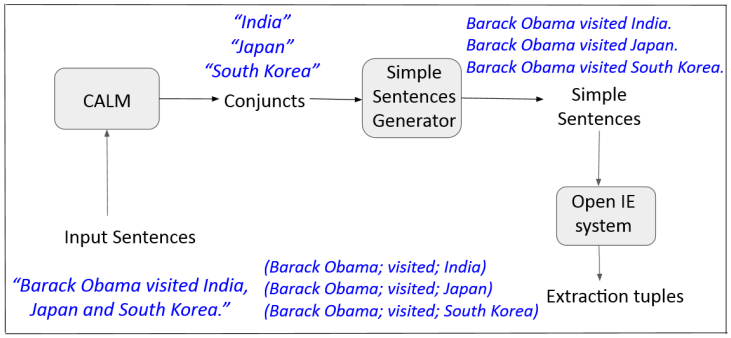


Figure 1: Flow diagram of CALMIE

Table 1), thereby motivating the need to split conjunctive sentences.

Conjunct Boundary Detection: Although co-ordination disambiguation has attracted attention of researchers over the years, it still remains one of the hardest problems to solve. Prior work has used two main principles for resolving ambiguities – (1) coordinated conjuncts have similar syntactic structures (Hogan, 2007; Shimbo and Hara, 2007; Hara et al., 2009; Hanamoto et al., 2012) and (2) replacing a full coordinated phrase with just one conjunct produces coherent (simple) sentences (Ficler and Goldberg, 2016).

The state-of-the-art coordination analyzer is by Ficler and Goldberg (2016), which uses LSTM-based components for operationalizing both these principles. It is a machine-learned model, which requires explicit annotation of coordination phrases for training (which isn't available in original Penn TreeBank). Importantly, it only outputs spans for the two conjuncts on either side of the conjunctive word and ignores any other conjuncts coordinated by the same word. Sentences often have a long list of comma separated conjuncts and not separating all of them would mean a substantial performance loss for end-tasks like Open IE.

In contrast, our approach eschews the first principle, as we find that it is not true often enough to be helpful. Our ranking of conjunct spans is based on the second principle (coherence of simple sentences). However, the search space is additionally restricted by various linguistic constraints for both single and nested coordination cases. Our system operates on top of Clear dependency parses (as opposed to constituency parses for Ficler's) and does not need any task-specific training data.

Sentence Simplification: CALM-based sentence splitting can be seen as a form of sentence simplification. Existing works on sentence simplification (Zhu et al., 2010; Vickrey and Koller, 2008; Vanderwende et al., 2007; Miwa et al., 2010) operate on top of syntactic parses, assuming them to be correct. CALM, on the other hand, corrects typical errors made by the parser to output corrected conjunction spans. These spans should naturally improve any sentence simplification task as well.

3 CALMIE

Figure 1 illustrates various steps of CALMIE. First, CALM identifies specific conjuncts to split the sentence into multiple simple sentences. Then an Open IE system acts on simple sentences to generate extractions. In this section we first focus on our key technical contribution, the coordination analyzer named CALM and conclude with a discussion on the generation of simple sentences.

3.1 CALM

CALM's goal is to output all *conjunct lists* from a sentence. For e.g., for sentence of Table 1, it should likely output five lists: (1) ⟨*'an American investor', 'co-founder of Microsoft'*⟩, (2) ⟨*'stepped down...2000', 'remained as...Craig Mundie'*⟩, (3) ⟨*'remained as chairman', 'created...Mundie'*⟩, (4) ⟨*'created...himself', 'transferred his...Mundie'*⟩, (5) ⟨*'Ray Ozzie', 'Craig Mundie'*⟩.[2]

[2]The conjunct lists #2, #3, #4 can change depending on the nesting level of the corresponding conjunctions.

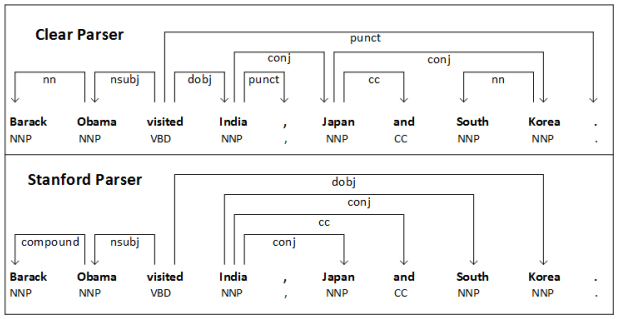


Figure 2: Clear and Stanford parses of a sample sentence

1. Generate the dependency parse of the sentence.
2. Find one of the conjunct heads through *'cc'* edge.
3. Find all other conjunct heads through *'conj'* edges.
4. Expand the conjunct heads to generate the conjuncts.

Table 3: Algorithm for Parser Baseline

We describe CALM's algorithm in two subsections. First, we describe CALM for a sentence that has only one coordinating conjunction (e.g., sentence of Figure 1). The case of multiple conjunctions, which may be nested (as in Table 1), is discussed in the latter subsection.

3.1.1 Single Coordinating Conjunction

We create a rule-based baseline to convert a dependency parse into a conjunct list (pseudo-code in Table 3). It finds various conjunct heads via *cc* and *conj* edges and expands the heads to generate conjunct spans. In the last step of the algorithm, while expanding on the conjunct heads, we don't expand on commas, the corresponding conjunction (*'and'*, *'but'*, etc) or any of the other conjunct heads.

We implement the baseline over Stanford[3] and Clear[4] parsers. Figure 2 shows that these dependency parsers have slightly different styles of notating conjuncts – Clear connects conjunct heads serially, whereas in Stanford parses one central conjunct heads connects to all others. Preliminary experiments reveal that Stanford parser makes many more mistakes than Clear. So we choose Clear parser for further algorithm design.

Analysis of Clear Parser-based Baseline Algorithm: Clear Parser identifies the conjunct heads correctly in most cases, but often makes mistakes in conjunct spans. We analyze a sample of 100 conjunctive sentences from BNC[5]. We find that 32 of them have correct parses with correctly identified conjuncts. Most correctly identified conjuncts are noun phrases (NP) while most incorrect ones involve verb phrases (VP). For e.g., in the sentence *"It also helps draw out toxins and excess oils."*, the NP conjuncts *'toxins'* and *'excess oils'* are identified correctly.

Our analysis also reveals that almost all wrong conjunct boundaries happen at the first and the last conjunct in the list, where the start of the first conjunct or the end of the last conjunct are identified incorrectly. Of the 68 incorrect parses, 57 of them have the first and last conjunct longer than necessary, suggesting that we can focus on shortening the first and last conjuncts. Further, out of these 57 sentences, 23 have a common NP subject distributed over multiple VPs and are wrongly represented in the parse. For e.g., in the sentence *"Angels danced in the air and settled reverently into their alcoves."*, the NP *'Angels'* incorrectly comes in the subtree of the VP *'danced in the air'*, leading to the generation of two conjuncts - *'Angels danced in the air'* and *'settled reverently into their alcoves'*. Most of the remaining 34 sentences also contain a phrase that is distributed over two VPs, but appears in the subtree of only one. For e.g., in the sentence *"He rejoices at the fact that they started off with smalltown views , and began thinking globally."*, the phrase *"that they"* appear in the subtree of *"started"*. Note that the conjuncts again are VPs. Finally, incorrect conjunct boundaries almost always result in the generation of ungrammatical simple sentences.

We design CALM on the basis of these observations – it shortens the first and last conjunct spans in a way that the resulting simple sentences are coherent, as evaluated by a language model.

Language-Model Based Algorithm: CALM needs to find the best start of the first conjunct and best end

[3] nlp.stanford.edu/software/stanford-dependencies.shtml
[4] https://github.com/clir/clearnlp
[5] http://nclt.computing.dcu.ie/~jfoster/resources/bnc1000.html

	Example 1	Example 2
Sentence	She gasped as he reached out and clasped her shoulders.	Still dazed, the man eventually got himself home and called police.
Parser Baseline (Clear Parser)	1. She gasped as he reached out. 2. She gasped clasped her shoulders.	1. Still dazed, the man eventually got himself home. 2. called police.
+ Language Model	1. She gasped as he reached out. 2. She gasped **as he** clasped her shoulders.	1. Still dazed, the man eventually got himself home. 2. called police.
+ Constraints	1. She gasped as he reached out. 2. She gasped **as he** clasped her shoulders.	1. Still dazed, the man eventually got himself home. 2. **Still dazed, the man eventually** called police.

Table 4: Comparison of sentences generated using different algorithms on two conjunctive sentences.

of the last conjunct. It first generates candidates by successively shortening conjuncts from the respective ends, until a conjunct is shortened to a single word. For e.g., for the first conjunct in *"Angels..reverently"* sentence, the candidates are *'Angels danced in the air', 'danced in the air', 'in the air', 'the air'* and *'air'*.

For each candidate, it constructs simple sentences by replacing the complete coordination structure with each of its conjuncts. It scores the coherence of each simple sentence via LMScore, a language model based score, described below. It picks the best span based on maximizing the total LMScore (over all simple sentences).

To score the coherence of a simple sentence, we could simply use its language model probability – product of probabilities of each word given the entire sentence so far. Most language models approximate this via an n-gram probability for a *fixed* context window of length $n-1$, instead of taking the *entire* sentence so far.

However, using the language model score is ineffective for two reasons. First, we are comparing simple sentences of varying lengths, and language model scores will usually score shorter sentences higher (product of fewer probabilities). A possible correction is to take the $|s|^{th}$ root of the language model score, with $|s|$ being the sentence length. But, this isn't enough for a second, subtle reason.

Consider the incorrect span *'the air'* for the first conjunct, which results in the simple sentence *"Angels danced in settled reverently into their alcoves"*. Notice that the n-gram probability scores will typically assign high probabilities for the parts: *'Angels danced in'* and *'settled reverently into their alcoves'*, since they are bona fide parts of the original sentence. Only the part around the boundary (*'...in settled...'*) would hurt the grammaticality. However, taking the product over the *whole* simple sentence has the tendency that the probabilities around the intersection could get overpowered by high scores from the initial and ending parts of the sentence, yielding an overall high score for an ungrammatical sentence.

To correct for this, CALM only multiplies probabilities starting from the intersection point and upto n–1 words (and takes n–1$^{\text{th}}$ root). All omitted probabilities are simply the product of original sentence fragments and thus are unhelpful in disambiguating the boundary. E.g., for $n = 4$ our method LMScore computes the following product:

LMScore(*"Angels danced in settled...alcoves."*)
= [P(*'settled'* | *'Angels danced in'*)
× P(*'reverently'* | *'dance in settled'*)
× P(*'into'* | *'in settled reverently'*)]$^{1/3}$

In case k ($k < n$–1) probabilities are multiplied (for e.g., if a conjunct is too short), then CALM takes k^{th} root of the product.

Use of Linguistic Constraints: Although LMScore corrects many of the wrongly identified conjuncts, we observe some limitations too. In sentences with proper nouns, the n-gram probabilities are low and significantly decrease the overall score of the sentence. Sentences with multi-word named entities also should not be split. For e.g., in the sentence *"Donald Trump gave a speech and flew back to U.S.."*, the simple sentence *"Donald flew back to U.S.."* is grammatical yet undesirable – named entity *'Donald Trump'* should not be split. We further see some obvious grammatical errors in the generated sentences, with two linguistically incoherent adjacent verbs or a preposition *'to'* preceding a past tense verb in the generated sentences.

To correct such errors, CALM picks the candidate that has the highest LMScore without violating any of the following linguistic constraints:

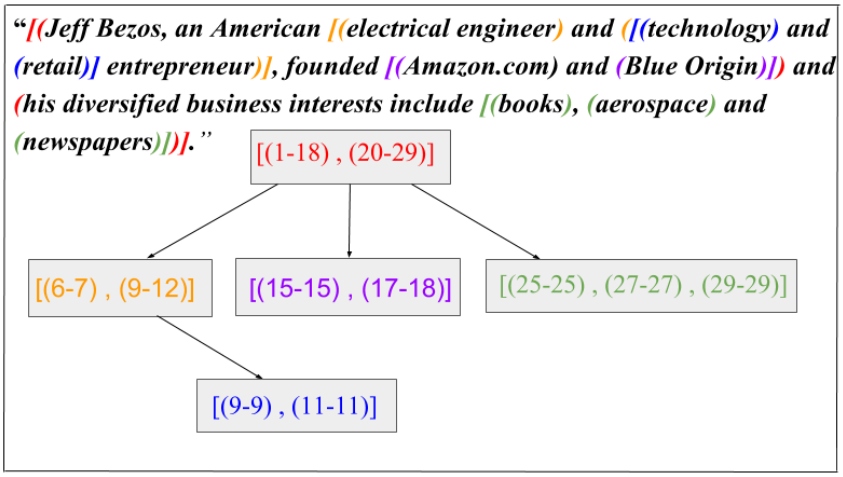


Figure 3: Hierarchical conjunct tree representation of a sentence with multiple conjunctions

1. Each simple sentence must have a subject. CALM looks for *nsubj* and *nsubjpass* dependency labels in the parse tree.
2. Named Entities (as identified by Stanford NER system[6]) should not be split.
3. If two verbs are adjacent, the first must be a light verb[7]. It handles ungrammatical sentences containing two adjacent incoherent verbs. For e.g., *"lives listening (to music)"* is nonsensical, but *"likes listening (to music)"* is not since *'like'* is a light verb.
4. A simple sentence should not have two consecutive occurrences of the same word.
5. Verb categories *VBD*, *VBZ* and *VBP* must precede a set of predefined POS tags[8]. English rarely allows a preposition, determiner or other such POS tags before the past/participle forms of a verb. However, language models do not always devalue these sentences. This constraint helps eliminate candidates that lead to such simple sentences.

3.1.2 Multiple Coordinating Conjunctions

When a sentence has multiple coordinating conjunctions, CALM identifies all coordination structures, i.e., the conjunct lists associated with each conjunction. Following English grammar, two coordination structures may have nothing in common (disjoint) or one conjunct list may be contained within the span of one of the conjuncts of another list (nested). For example, in sentence of Figure 3, the conjunct lists marked in yellow ({*'electrical engineer', 'technology and retail entrepreneur'*}) and green ({*'books', 'aerospace', 'newspapers'* }) are disjoint, while the one marked in blue ({*'technology', 'retail'*}) is nested within the second conjunct of yellow conjunct list.

Partial intersections, where one conjunct list is only partially contained in a conjunct of another list, are ungrammatical. CALM uses this knowledge as a search space constraint to jointly disambiguate all coordination structures in a sentence. We name this the *multiple conjunction constraint*. To operationalize this, we define a novel representation, *hierarchical coordination tree* (HCTree), for expressing the compositional containment between coordination structures.

An HCTree is a tree with each node representing a conjunct list for a single coordination structure, stored as a sequence of interval offsets. An edge from node A to B represents that B is fully contained within one of the conjuncts of A (nested case). Figure 3 illustrates an HCTree: *'electrical engineer'* is stored as interval 6-7, as it comprises sixth and seventh tokens in the sentence.

The number of legal HCTrees for a multiple conjunction sentence could be huge and a complete search over all tree structures and spans will be prohibitive. However, we reduce the search space by not performing a full search, as Clear parser usually returns a structurally accurate analysis, but, as before, makes errors in exact boundaries.

Use of Multiple Conjunction Constraint: CALM constructs an initial HCTree by converting the Clear parse into conjunct lists (using parser baseline), adding a node per conjunct list, and adding edges as per

[6] `https://nlp.stanford.edu/software/CRF-NER.shtml`
[7] We use the list of light verbs released by (Jain and Mausam, 2016) at `https://github.com/dair-iitd/kglr`.
[8] CC, CD, EX, NN, NNS, NNP, NNPS, PDT, PRP, RB, RBR, RBS, WDT, WP

containments in the parse. This provides us with a good structure for enforcing the multiple conjunction constraint. CALM keeps the HCTree structure constant, and re-evaluates the exact conjunct boundaries for each node in the tree.

In addition to providing a structure over all conjunct lists, an HCTree also imposes a natural order in which we could correct span errors. Typically, smaller conjuncts are easier to correct than longer ones. So, CALM makes a bottom up pass greedily correcting spans for the conjunct list in each node. As in previous subsection, a conjunct is only shortened. Moreover only start of first conjunct and end of last conjunct in a list are re-evaluated using the single-conjunction version of CALM.

Notice that shortening of a child conjunct doesn't hurt the consistency of an HCTree, since if a conjunct list was contained in a parent conjunct, then the shortened conjunct list is also contained in the same parent conjunct. However, the boundaries of child conjunct list impose an additional constraint on the candidate spans of the parent conjunct list – while shortening the parent conjunct, all child conjuncts must continue to be contained in it. Thus, when applying CALM at the parent node, this additional search constraint is imposed to maintain a valid HCTree.

3.2 Simple Sentences for Open IE

Once the spans of conjunct lists have been determined, the next task is to generate the simple sentences. Our Open IE system, CALMIE generates them in a top-down order of the HCTree. It starts with the original sentence. Now, at each level of the tree, it collects all the conjunct lists and generates all possible sentences out of the sentences generated from the previous level. Since at a particular level of the tree all conjunct lists are disjoint, generating simple sentences requires simply identifying parts of the sentence that do not belong to any conjunct list and concatenating them in order with a conjunct for each coordination structure. Note that it avoids generation of duplicate sentences by identifying which conjunct lists at a particular level are part of which simple sentences from the previous level and finally splitting only those. After processing the conjunct lists at the last level of the HCTree, we get all the simple sentences.

For Open IE, we wish to only produce those simple sentences, whose truth can be inferred from the original sentence. CALMIE doesn't split conjuncts coordinated by *'or'*, *'nor'*, and paired conjunctions like *'either-or'* and *'neither-nor'*. For e.g. splitting *"Adam's nationality is French or German."* will be inaccurate.

One common class of non-distributive coordination that cannot be split contains prepositions like *'between'* in connecting the conjuncts. For e.g. the sentence *"The world cup final was played between Germany and Argentina."* should not be split. Other examples aggregate information across conjuncts as in the sentence, *"The average of 3 and 5 is 4."* Thus, we create a list of triggers (*'between', 'among', 'total', 'sum', 'average','each other', ...*) using Thesaurus expansion of seed words which indicate non-distributive coordination . If the arguments of Open IE extractions on the original sentence contain any of the triggers, CALMIE does not split these conjuncts.

4 Experiments

We perform two sets of experiments. Section 4.1 evaluates CALM for coordination analysis task, and Section 4.2 demonstrates the performance increase, when using CALMIE for Open IE.

CALM's implementation uses the Berkeley Language Model,[9] which is based on the Google n-gram corpus. It uses Stupid Back-off smoothing (Brants et al., 2007) for infrequent n-grams. CALMIE runs Open IE 4.2 and ClausIE over the simple sentences generated in the previous section.

4.1 Experiments on Coordination Analysis

Previous work has given credit to a system only when both boundaries of a conjunct match exactly (Ficler and Goldberg, 2016). However, this is not ideal for downstream tasks like Open IE. Consider a sentence with polysyndetic coordination: *"Obama visited India and Japan and South Korea."*. Here, two analyses are equally good, depending upon whether we consider first conjunction as top-level or second, leading

[9] https://code.google.com/archive/p/berkeleylm/

	Parser Baseline (Stanford Parser)			Parser Baseline (Clear Parser)			+ Language Model			+ Constraints		
	SC	**MC**	**SC+MC**	**SC**	**MC**	**SC+MC**	**SC**	**MC**	**SC+MC**	**SC**	**MC**	**SC+MC**
Precision	83.93	69.20	79.30	**94.69**	86.78	92.14	94.33	87.85	**92.24**	94.22	**88.00**	92.21
Recall	80.91	80.78	80.86	90.22	78.34	86.39	91.36	82.75	88.58	**92.97**	**83.23**	**89.83**
F-score	82.39	74.75	80.07	92.40	82.34	89.17	92.82	85.22	90.37	**93.59**	**85.55**	**91.00**

Table 5: Results for simple sentence evaluation on British News Corpus. CALM obtains about 2 pt F-score improvement over Clear parser baseline. SC: Sentences with one conjunction, MC: Sentences with multiple conjunctions.

	(Ficler and Goldberg, 2016)	CALM
Precision	72.81	**75.12**
Recall	**72.61**	70.64
F1	72.7	**72.81**

Table 6: Results for conjunct boundary detection on Penn Treebank. CALM is competitive with state of the art.

to first conjunct being *'India'* or *'India and Japan'*, respectively. Such artifacts commonly happen when there are multiple conjunctions in a sentence, such as in Table 1.

In response, we evaluate a coordination analysis by generating the simple sentences and comparing them against the gold set of simple sentences. We split all coordinations (whether they are distributive or not) for this evaluation. For each conjunctive sentence, we compare its set of system-generated simple sentences with a gold set by first finding the best one-to-one mapping between the simple sentences in the two sets. Then for each mapping, precision is computed as the number of overlapping words upon the number of words in the predicted sentence. Recall is the number of overlapping words upon the number of words in the gold sentence. Notice that for the sentence above, all (correct) alternative analyses will yield the same simple sentences, obtaining perfect precision and recall scores.

We run our first set of experiments on all conjunctive sentences from British News Corpus test set.[10] It contains the gold parses for all sentences, which are used to generate the gold simple sentences. BNC testset has 577 conjunctive sentences – 391 with one conjunction and 186 with multiple conjunctions.

Table 5 compares the performances of all our algorithms on the whole BNC testset, and also individually on single and multiple conjunction sentences. We find that due to better conjunct heads identification, Clear parser does a much better job than Stanford parser in identifying conjunct boundaries. We get a point of F-score improvement by using language model over Clear parser baseline. Incorporating linguistic constraints yields another two-thirds of a point. More importantly, CALM obtains nearly 3.2 pt gain for sentences with multiple conjunctions; this highlights the value of our HCTree representation. Overall improvement of CALM over Clear baseline for the whole BNC is statistically significant using paired t-test with $p = 0.015$.

For sentences with a single conjunction, there is a slight drop in the final precision as compared to the parser baseline algorithm. Since we are only shortening the conjuncts, an already incorrect analysis can end up having even lesser number of common words with the gold sentence, reducing the precision. While the final precision is comparable, the recall improves significantly. This can be explained by examples in Table 4. E.g., in the second example, parser baseline has perfect precision, but low recall, due to missing words in the second simple sentence (*"called police"*). CALM corrects this by adding words in the sentence, increasing the recall to 1.0.

Comparison with Ficler's System: We also directly compare against Ficler's system on *their* dataset (Penn TreeBank). Recall that their evaluation metric is exact match of conjunct boundaries but only for the two conjuncts closest to the conjunction. We use their exact metric (Table 6) and find that CALM has slightly different characteristics – it has a higher precision and a lower recall; but, in aggregate it produces a similar F-score. Given that CALM is not trained directly (except for underlying parsers), we find it creditable that it can match performance of the state of the art system that was specifically trained on this data for this task. Unfortunately, their code isn't available for us to compare performance on BNC.

[10] http://nclt.computing.dcu.ie/~jfoster/resources/bnc1000.html

	ClueWeb12				News+Wikipedia			
	[C]	**[Cm[C]]**	**[O4]**	**[Cm[O]]**	**[C]**	**[Cm[C]]**	**[O4]**	**[Cm[O]]**
Precision	62.50	**64.50**	70.04	**74.80**	67.17	**68.12**	79.12	**81.2**
Yield	267	**381**	199	**349**	204	**325**	172	**315**

Table 7: Open IE comparison on two datasets. [C]: ClausIE, [O4]: Open IE 4.2, [Cm[O]]: CALMIE(O), [Cm[C]]: CALMIE(C)

	Two Conjuncts		More than Two Conjuncts	
	[FG]	**[Cm[O]]**	**[FG]**	**[Cm[O]]**
Precision	**72.71**	72.35	74.50	**74.78**
Yield	323	**330**	346	**445**

Table 8: Open IE comparison on Penn TreeBank. [FG]: Ficler+Open IE 4.2, [Cm[O]]: CALMIE(O).

Error Analysis: CALM sometimes misses conjunctions in sentences due to the inaccuracy of parsers (absence of a 'cc' edge). A more common problem is that of missing contexts in sentences while splitting. E.g., from the sentence *"Two years ago we were carrying huge inventories and that was the big culprit ."*, CALM generates two simple sentences: *"Two years ago we were carrying huge inventories."* and *"that was the big culprit."*. Although the sentences are grammatically correct, we miss the phrase *"Two years ago"* in the second sentence. While this is a missing prefix problem, CALM often tends to miss important phrases in the suffix as well. E.g., for the sentence *"We want to see Nelson Mandela and all our comrades out of the prison."*, it misses the suffix phrase *"out of the prison"*. We believe fixing such problems will require better understanding of the semantics of the sentence.

4.2 Experiments on Open IE

We now evaluate the improvement in performance on the final task of Open IE using conjunctive sentences from three different datasets. We randomly sample 100 conjunctive sentences each from ClueWeb12 (CW)[11] and Open IE benchmarking dataset (NW) of Newswire and Wikipedia sentences (Stanovsky and Dagan, 2016). We consider a sentence conjunctive if its parse has a *cc* edge. Note that we could not use the whole of NW dataset since its gold set of extractions does not split conjunctions in arguments.

We also test on Penn Treebank (PTB) testset used for coordination evaluation in (Ficler and Goldberg, 2016). We report two sets of numbers from the dataset – for a random sample of 100 sentences with two conjuncts per conjunction, and all of 95 conjunctive sentences with more than 2 conjuncts per conjunction.

For CW and NW, we compare CALMIE(C) (CALM generated simple sentences passed through ClausIE) and CALMIE(O) (CALM generated simple sentences passed through Open IE 4.2) against two state-of-the-art Open IE systems, ClausIE and Open IE 4.2. These Open IE systems outperform others in a recent large-scale evaluation. Moreover, ClausIE also splits some conjunctive clauses, making it an especially suitable system for this comparison. As there is no automated way to check the correctness of an extraction, two annotators with NLP experience annotate the extractions for correctness. Each annotator annotated about 6000 extractions in total. We obtain an inter-annotator agreement of 97.2% across all the datasets and report the results on the subset where both agree. Table 7 lists the precision and yield on these test sets. Note that yield is equal to the number of correct extractions and is proportional to recall. It is normally used in Open IE where recall denominator is hard to compute.

On CW, CALMIE(C) and CALMIE(O) obtain 1.4x and 1.8x yields compared to ClausIE and Open IE 4.2 respectively. The respective precision gains are 2 and 5 points. On NW, CALMIE(C) and CALMIE(O) achieve 1.6x and 1.8x better yields with marginal precision gains compared to ClausIE and Open IE 4.2 respectively. All the improvements are statistically significant using paired t-test at $p < 10^{-4}$. We find that both Open IE 4.2 and ClausIE are unable to separate out extractions if the conjunctions are in the arguments. This largely accounts for CALMIE's massive yield improvement. Overall, we see substantial improvement in extractions of both Open IE 4.2 and ClausIE after the application of CALM.

Finally, we split the sentences in PTB sample into simple sentences using CALM and Ficler. Both sets of simple sentences are then fed to Open IE 4.2 for generating extractions, i.e., the pipeline after

[11] http://www.lemurproject.org/clueweb12.php

coordination analysis is the same. Table 8 shows that on sentences with two conjuncts per conjunction, the two sets of extractions are comparable. However, on 95 sentences with more than two conjuncts per conjunction, CALMIE(O) achieves 1.3x better yield than Ficler. This is due to Ficler's inability to separate out all the conjuncts.

Error Analysis: The single major challenge in Open IE over conjunctive sentences is in figuring out when not to split. For example, in *"Japan's domestic sales of cars, trucks and buses in October rose by 18%."*, no obvious trigger is present that can indicate non-distributive coordination. Another common class is organization names with *'and'* as part of their names. For e.g., in sentence *"The Perch and Dolphin fields moved their headquarters."*, *'Perch and Dolphin'* should not be split. Use of LEX system could help with better NER tagging (Downey et al., 2007). Finally, non-context free constructions need to be handled differently while splitting as in, *"Germany and Argentina beat Brazil and Netherlands in the semis respectively."* We believe that this remains one of the key future directions for improving the precision of our system.

5 Conclusions and Discussion

Coordination disambiguity is regarded as one of the hardest problems in NLP (Ficler and Goldberg, 2016). We develop CALM, a coordination analyzer that corrects the errors of Clear parser by using three insights: (1) splitting into coherent simple sentences, as judged by a modified language model score, (2) linguistically inspired constraints to obtain better conjunct spans, and (3) a novel hierarchical coordination tree data structure that enforces consistency among conjunct lists for multiple coordinations, leading to performance gain.

We split distributive coordinations to obtain simple sentences for running downstream Open IE. Our Open IE system, CALMIE, obtains enormously higher yields (upto 1.8 times) and comparable or better precisions for conjunctive sentences against state of the art Open IE systems. Empirical evaluation across multiple datasets demonstrate than CALM should likely improve any Open IE system. We also believe that CALM has value outside of Open IE, such as for Closed IE and sentence simplification.

CALM's insights are general, but its implementation is specific to Clear parser, which is used in Open IE 4.2. Minor modifications may be needed when correcting coordination errors in other parsers. In the future, we plan to use CALM to improve Clear parser itself.

A strength (and also weakness) of our work is that it is not an ML system. While we test CALM on several domains, it is conceivable that for some domains or genres, it does not perform as well. Embedding our ideas in the context of a learning system may obtain even better in-domain performance. One approach is to use CALM output as (hard or soft) constraint at inference time, similar to how CCMs use constraints while learning parameters. We also note that our main idea of splitting into simple sentences, may not always work, e.g., in the case of ellipsis and gaping. Such sentences are infrequent in our datasets; extending to these is another direction for the future.

We release Open IE 5.0 which integrates CALMIE(O) into the latest software of OpenIE. We also release the annotated datasets used for evaluating CALMIE(O) and the implementations of CALM and the sentence splitter which can be used for splitting conjunctive sentences in various downstream NLP tasks. All of these are publicly available for further research at `https://github.com/dair-iitd/OpenIE-standalone`.

Acknowledgements

This work is supported by Google language understanding and knowledge discovery focused research grants, a Bloomberg award, an IBM SUR award and a Visvesvaraya faculty award by Govt. of India to the second author. We thank the anonymous reviewers for their insightful comments on improving earlier drafts of the paper. We acknowledge IBM Research, India for providing a travel grant to the first author.

References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 344–354.

Hannah Bast and Elmar Haussmann. 2013. Open information extraction via contextual sentence decomposition. In *2013 IEEE Seventh International Conference on Semantic Computing, Irvine, CA, USA, September 16-18, 2013*, pages 154–159.

Hannah Bast and Elmar Haussmann. 2014. More informative open information extraction via simple inference. In *Advances in Information Retrieval - 36th European Conference on IR Research, ECIR 2014, Amsterdam, The Netherlands, April 13-16, 2014. Proceedings*, pages 585–590.

Nikita Bhutani, H. V. Jagadish, and Dragomir R. Radev. 2016. Nested propositions in open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 55–64.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz Josef Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pages 858–867.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2011. An analysis of open information extraction based on semantic role labeling. In *Proceedings of the 6th International Conference on Knowledge Capture (K-CAP 2011), June 26-29, 2011, Banff, Alberta, Canada*, pages 113–120.

Janara Christensen, Stephen Soderland, Gagan Bansal, and Mausam. 2014. Hierarchical summarization: Scaling up multi-document summarization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 902–912.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: clause-based open information extraction. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 355–366.

Filipe de Sá Mesquita, Jordan Schmidek, and Denilson Barbosa. 2013. Effectiveness and efficiency of open relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 447–457.

Doug Downey, Matthew Broadhead, and Oren Etzioni. 2007. Locating complex named entities in web text. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2733–2739.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. 2008. Open information extraction from the web. *Commun. ACM*, 51(12):68–74.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. 2011. Open information extraction: The second generation. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 3–10.

Jessica Ficler and Yoav Goldberg. 2016. A neural network for coordination boundary prediction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 23–32.

Atsushi Hanamoto, Takuya Matsuzaki, and Jun'ichi Tsujii. 2012. Coordination structure analysis using dual decomposition. In *EACL 2012, 13th Conference of the European Chapter of the Association for Computational Linguistics, Avignon, France, April 23-27, 2012*, pages 430–438.

Kazuo Hara, Masashi Shimbo, Hideharu Okuma, and Yuji Matsumoto. 2009. Coordinate structure analysis with global structural constraints and alignment-based local features. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 967–975.

Deirdre Hogan. 2007. Coordinate noun phrase disambiguation in a generative parsing model. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, June.

Prachi Jain and Mausam. 2016. Knowledge-guided linguistic rewrites for inference rule verification. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 86–92.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 523–534.

Mausam. 2016. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 4074–4077.

Makoto Miwa, Rune Saetre, Yusuke Miyao, and Jun'ichi Tsujii. 2010. Entity-focused sentence simplification for relation extraction. In *Proceedings of the 23rd international conference on computational linguistics*, pages 788–796. Association for Computational Linguistics.

Harinder Pal and Mausam. 2016. Demonyms and compound relational nouns in nominal open IE. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction, AKBC@NAACL-HLT 2016, San Diego, CA, USA, June 17, 2016*, pages 35–39.

Swarnadeep Saha, Harinder Pal, and Mausam. 2017. Bootstrapping for numerical open IE. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 317–323.

Jordan Schmidek and Denilson Barbosa. 2014. Improving open relation extraction via sentence re-structuring. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014), Reykjavik, Iceland, May 26-31, 2014.*, pages 3720–3723.

Masashi Shimbo and Kazuo Hara. 2007. A discriminative learning model for coordinate conjunctions. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pages 610–619.

Gabriel Stanovsky and Ido Dagan. 2016. Creating a large benchmark for open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2300–2305.

Gabriel Stanovsky, Ido Dagan, and Mausam. 2015. Open IE as an intermediate structure for semantic tasks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 303–308.

Gabriel Stanovsky, Meni Adler, and Ido Dagan. 2016a. Specifying and annotating reduced argument span via qa-srl. In *Proceedings of the 54rd Annual Meeting of the Association for Computational Linguistics (ACL 2016)*.

Gabriel Stanovsky, Ido Dagan, and Meni Adler. 2016b. Specifying and annotating reduced argument span via QA-SRL. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*.

Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing & Management*, 43(6):1606–1618.

David Vickrey and Daphne Koller. 2008. Sentence simplification for semantic role labeling. *Proceedings of ACL-08: HLT*, pages 344–352.

Ying Xu, Mi-Young Kim, Kevin Quinn, Randy Goebel, and Denilson Barbosa. 2013. Open information extraction with tree kernels. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 868–877.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd international conference on computational linguistics*, pages 1353–1361. Association for Computational Linguistics.

Graphene: Semantically-Linked Propositions in Open Information Extraction

Matthias Cetto[1], Christina Niklaus[1], André Freitas[2], and **Siegfried Handschuh[1]**
[1] Faculty of Computer Science and Mathematics, University of Passau
{matthias.cetto, christina.niklaus, siegfried.handschuh}@uni-passau.de
[2] School of Computer Science, University of Manchester
andre.freitas@manchester.ac.uk

Abstract

We present an Open Information Extraction (IE) approach that uses a two-layered transformation stage consisting of a clausal disembedding layer and a phrasal disembedding layer, together with rhetorical relation identification. In that way, we convert sentences that present a complex linguistic structure into simplified, syntactically sound sentences, from which we can extract propositions that are represented in a two-layered hierarchy in the form of *core relational tuples and accompanying contextual information* which are *semantically linked via rhetorical relations*. In a comparative evaluation, we demonstrate that our reference implementation Graphene outperforms state-of-the-art Open IE systems in the construction of correct n-ary predicate-argument structures. Moreover, we show that existing Open IE approaches can benefit from the transformation process of our framework.

1 Introduction

Information extraction (IE) turns the unstructured information expressed in natural language (NL) text into a structured representation (Jurafsky and Martin, 2009) in the form of relational tuples that consist of a set of arguments and a phrase denoting a semantic relation between them, e.g. ⟨*Barack Obama*; *served as*; *the 44th President of the US*⟩. Traditional IE systems have concentrated on identifying and extracting relations of interest by taking as input the target relations, along with hand-crafted extraction patterns or patterns learned from hand-labeled training examples. As this manual labor scales linearly with the number of target relations, such a supervised approach does not scale to large, heterogeneous corpora which are likely to contain a variety of unanticipated relations. To tackle this issue, Banko et al. (2007) introduced Open IE as a new extraction paradigm that allows for a domain-independent discovery of relations in large amounts of text by not depending on any relation-specific human input. Instead, detecting the relations is part of the problem. State-of-the-art Open IE systems (see Section 2) identify relationships between entities in a sentence by matching patterns over either shallow syntactic features in terms of part-of-speech (POS) tags and noun phrase (NP) chunks or dependency tree structures. However, particularly long and syntactically complex sentences pose a challenge for current Open IE approaches. By analyzing the output of such systems (see Figure 1), we observed three common shortcomings.

First, relations often span over long nested structures or are presented in a non-canonical form that cannot be easily captured by a small set of extraction patterns. Therefore, such relations are commonly missed by state-of-the-art approaches. Consider for example the first sentence in Figure 1 which asserts that ⟨*Sonia Sotomayor*; *became*; *the first Supreme Court Justice of Hispanic descent*⟩. This information is encoded in a complex participial construction that is omitted by both reference Open IE systems, OLLIE (Mausam et al., 2012) and ClausIE (Del Corro and Gemulla, 2013).

Second, current Open IE systems tend to extract propositions with long argument phrases that can be further decomposed into meaningful propositions, with each of them representing a separate fact. Overly specific constituents that mix multiple - potentially semantically unrelated - propositions are difficult

Proceedings of the 27th International Conference on Computational Linguistics, pages 2300–2311
Santa Fe, New Mexico, USA, August 20-26, 2018.

```
He nominated Sonia Sotomayor on May 26, 2009 to replace David Souter; she was confirmed on August 6, 2009,
becoming the first Supreme Court Justice of Hispanic descent.

OLLIE:
(1) she                 was confirmed on                  August 6, 2009
(2) He                  nominated Sonia Sotomayor on      May 26
(3) He                  nominated Sonia Sotomayor         2009
(4) He                  nominated 2009 on                 May 26
(5) Sonia Sotomayor     be nominated 2009 on              May 26
(6) He                  nominated 2009                    Sonia Sotomayor
(7) 2009                be nominated Sonia Sotomayor on   May 26

ClausIE:
(8)  He     nominated       Sonia Sotomayor on May 26 2009 to replace David Souter
(9)  she    was confirmed   on August 6 2009 becoming the first Supreme Court Justice of Hispanic descent
(10) she    was confirmed   becoming the first Supreme Court Justice of Hispanic descent

Graphene:
(11) #1    0    he    nominated    Sonia Sotomayor
("a)     S:PURPOSE     to replace David Souter
("b)     S:TEMPORAL    on May 26, 2009
(12) #2    0    she    was confirmed
("a)     S:TEMPORAL    on August 6, 2009
(13) #3    0    she    was becoming    the first Supreme Court Justice of Hispanic descent


Although the Treasury will announce details of the November refunding on Monday, the funding will be delayed
if Congress and President Bush fail to increase the Treasury's borrowing capacity.

OLLIE:
(14) the Treasury                   will announce       details of the November refunding
(15) Congress and President Bush    fail to increase    the Treasury's borrowing capacity

ClausIE:
(16) the Treasury    will announce      details of the November refunding on Monday
(17) the Treasury    will announce      details of the November refunding
(18) the funding     will be delayed    if Congress and President Bush fail to increase the Treasury 's [...]
(19) the funding     will be delayed    if Congress and President Bush fail to increase the Treasury 's [...]
                                  Although the Treasury will announce details of the November [...]
(20) Congress and President Bush    fail    to increase the Treasury 's borrowing capacity
(21) the Treasury    has    borrowing capacity

Graphene:
(22) #1    0    the Treasury    will announce    details of the November refunding
("a)     S:TEMPORAL    on Monday
("b)     L:CONTRAST     #2
(23) #2    0    the funding    will be delayed
("a)     L:CONTRAST     #1
("b)     L:CONDITION    #3
("c)     L:CONDITION    #4
(24) #3    1    Congress    fail    to increase the Treasury 's borrowing capacity
(25) #4    1    president Bush    fail    to increase the Treasury 's borrowing capacity
```

Figure 1: Comparison of the output generated by different state-of-the-art Open IE systems. Contextual arguments of Graphene are differentiated between simple textual arguments (S:) or arguments that link to other propositions (L:). Both contextual types are semantically classified by rhetorical relations.

to handle for downstream applications, such as question answering (QA) or textual entailment tasks. Instead, such approaches benefit from extractions that are as compact as possible. This phenomenon can be witnessed particularly well in the extractions generated by ClausIE ((8-10) and (16-21)), whose argument phrases frequently combine several semantically independent statements. For instance, the argument in proposition (8) contains three unrelated facts, namely a direct object ⟨*Sonia Sotomayor*⟩, a temporal expression ⟨*on May 26 2009*⟩ and a phrasal description ⟨*to replace David Souter*⟩ specifying the purpose of the assertion on which it depends.

Third, they lack the expressiveness needed to properly represent complex assertions, resulting in incomplete, uninformative or incoherent propositions that have no meaningful interpretation or miss critical information asserted in the input sentence. Most state-of-the-art systems focus on extracting binary relationships without preserving the semantic connection between the individual propositions. This can be seen in the second example in Figure 1. Here, the proposition ⟨*Congress and President Bush*; *fail to increase*; *the Treasury's borrowing capacity*⟩ is not asserted by the input sentence, but rather represents the precondition for the predication that ⟨*the funding*; *will be delayed*; ∅⟩. However, both state-of-the-art Open IE systems fail to accurately reflect this contextual information in the extracted relational tuples. While OLLIE completely ignores above-mentioned inter-proposition relationship in its

extractions, ClausIE incorporates this information in its argument phrases ((18) and (19)), yet producing over-specified components that is likely to hurt the performance of downstream semantic applications.

To overcome these limitations, we developed an Open IE framework that transforms complex NL sentences into clean, compact structures that present a canonical form which facilitates the extraction of accurate, meaningful and complete propositions. The contributions of our work are two-fold. First, to remove the complexity of determining intricate predicate-argument structures with variable arity from syntactically complex input sentences, we propose a two-layered transformation process consisting of a clausal and phrasal disembedding layer. It removes clauses and phrases that convey no central information from the input and converts them into independent sentences, thereby reducing the source sentence to its main information. In that way, the input is transformed into a **novel hierarchical representation in the form of core facts and accompanying contexts**. Second, we **identify the rhetorical relations by which core sentences and their associated contexts are connected in order to preserve their semantic relationship** (see the output of our reference Open IE implementation Graphene in Figure 1). These two innovations enable us to enrich extracted relational tuples of the form $\langle arg_1, rel, arg_2 \rangle$ with contextual information that further specifies the tuple and to establish semantic links between them, allowing to fully reconstruct the informational content of the input. The resulting representation of the source text can then be used to facilitate a variety of artificial intelligence tasks, such as building QA systems or supporting semantic inferences. The idea of generating a syntactically sound representation from linguistically complex NL sentences can be easily transported to other tasks than Open IE, e.g. it might be helpful for problems such as sentiment analysis, coreference resolution or text summarization.

2 Related Work

Learning-based approaches. The line of work on Open IE begins with TEXTRUNNER (Banko et al., 2007), a self-supervised learning approach which uses a Naive Bayes classifier to train a model of relations over examples of extraction tuples that are heuristically generated from sentences in the Penn Treebank using unlexicalized POS and NP chunk features. The system then applies the learned extractor to label each word between a candidate pair of NP arguments as part of a relation phrase or not. WOE (Wu and Weld, 2010) also learns an open information extractor without direct supervision. It makes use of Wikipedia as a source of training data by bootstrapping from entries in Wikipedia infoboxes to learn extraction patterns on both POS tags (WOE^{pos}) and dependency parses (WOE^{parse}). By comparing their two approaches, Wu and Weld (2010) show that the use of dependency features results in an increase in precision and recall over shallow linguistic features (though, at the cost of extraction speed). OLLIE follows the idea of bootstrap learning of patterns based on dependency parse paths. However, while WOE relies on Wikipedia-based bootstrapping, OLLIE applies a set of high precision seed tuples from its predecessor system REVERB to bootstrap a large training set. Moreover, OLLIE is the first Open IE approach to identify not only verb-based relations, but also noun-mediated ones.

Rule-based approaches. The second category of Open IE systems make use of hand-crafted extraction rules. This includes REVERB (Fader et al., 2011), a shallow extractor that applies a set of lexical and syntactic constraints that are expressed in terms of POS-based regular expressions. In that way, the amount of incoherent, uninformative and overspecified relation phrases is reduced. While previously mentioned Open IE systems focus on the extraction of binary relations, KRAKEN (Akbik and Löser, 2012) is the first approach to be specifically built for capturing complete facts from sentences by gathering the full set of arguments for each relation phrase within a sentence, thus producing tuples of arbitrary arity. The identification of relation phrases and their corresponding arguments is based on hand-written extraction rules over typed dependency information. EXEMPLAR (Mesquita et al., 2013) applies a similar approach for extracting n-ary relations, as it uses hand-crafted patterns based on dependency parse trees to detect a relation trigger and the arguments connected to it.

Clause-based approaches. Aiming to improve the accuracy of Open IE approaches, more recent work is based on the idea of incorporating a sentence re-structuring stage whose goal is to transform the original sentence into a set of independent clauses that are easy to segment into Open IE tuples. An

example of such a paraphrase-based Open IE approach is ClausIE, which exploits linguistic knowledge about the grammar of the English language to map the dependency relations of an input sentence to clause constituents. In that way, a set of coherent clauses presenting a simple linguistic structure is derived from the input. Then, one or more predicate-argument extractions are generated for each clause. In the same vein, Schmidek and Barbosa (2014) propose a strategy to break down structurally complex sentences into simpler ones by decomposing the original sentence into its basic building blocks via chunking. The dependencies of each two chunks are then determined using dependency parsing or a Naive Bayes classifier. Depending on their relationships, chunks are combined into simplified sentences, upon which the task of relation extraction is carried out. Angeli et al. (2015) present Stanford Open IE, an approach in which a classifier is learned for splitting a sentence into a set of logically entailed shorter utterances which are then maximally shortened by running natural logic inference over them. In the end, a small set of hand-crafted patterns are used to extract a predicate-argument triple from each utterance.

On the basis of such re-arrangement strategies for decomposing a complex input sentence into a set of self-contained clauses that present a linguistic structure that is easier to process for Open IE systems, we have developed an Open IE pipeline which will be presented in the following section.

3 Proposed Open IE Approach

We propose a method for facilitating the task of Open IE on sentences that present a complex linguistic structure. It is based on the idea of disembedding clausal and phrasal constituents out of a source sentence. In doing so, our approach identifies and retains the semantic connections between the individual components, thus generating a novel lightweight semantic Open IE representation.

We build upon the concept presented in Niklaus et al. (2016), who distinguish between core and contextual information in the context of sentence simplification. This is done by disembedding and transforming supplementary material expressed in phrases (e.g. spatial or temporal information) into standalone context sentences, thus reducing the input sentences to their key information (core sentences). In our work, we now port this idea to a broader scope by targeting clausal disembedding techniques for complex, nested structures. Furthermore, by converting the whole simplification process into a recursive process, we are able to generate a hierarchical representation of syntactically simplified core and contextual sentences (see center image of Figure 2). Moreover, this allows us to detect both local and long-range rhetorical dependencies that hold between nested structures, similar to Rhetorical Structure Theory (RST) (Mann and Thompson, 1988). As a consequence, when carrying out the task of Open IE on the compressed core sentences, the complexity of determining intricate predicate-argument structures is removed and the extracted propositions can be semantically linked and enriched with the disembedded contextual information. Our proposed framework uses a two-layered transformation stage for recursive sentence simplification that is followed by a final relation extraction stage. It takes a text document as an input and returns a set of hierarchically ordered and semantically interconnected relational tuples (see the output of Graphene in Figure 1). The workflow of our approach is displayed in Figure 2.

3.1 Transformation Stage

The core component of our work is the two-layered transformation stage where sentences that present a complex linguistic form are recursively transformed into simpler, compact, syntactically sound sentences with accompanying contextual information.

3.1.1 Concept of the Discourse Tree Creation

During the transformation process, we aim to identify intra-sentential semantic relationships which can later be used to restore semantic relations between extracted propositions. To capture these semantic relations, we employ a concept of RST, in which a document is represented as a hierarchy of consecutive, non-overlapping text spans (so-called Elementary Discourse Units (EDUs)) that are connected by rhetorical relations such as *Condition*, *Enablement* or *Background*. In addition, Mann and Thompson (1988) introduced the notion of *nuclearity*, where text spans that are connected by rhetorical relations are classified as either *nucleus*, when embodying the central part of information, or *satellite*, whose role is to further specify the nucleus.

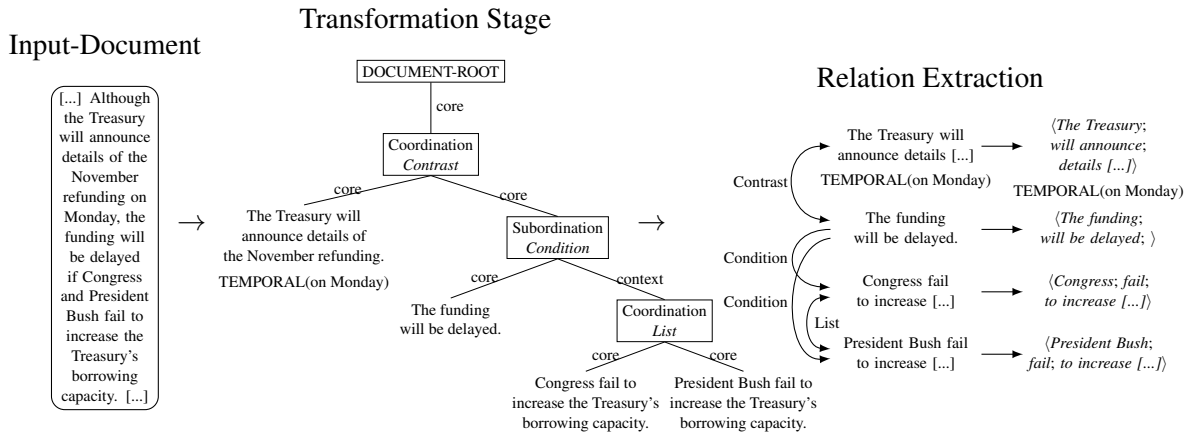


Figure 2: Extraction workflow for an example sentence.

We adapt the RST framework for our task of clausal sentence simplification, where the goal is to split up complex multi-clause sentences in a recursive fashion in order to obtain a hierarchical structure of the input similar to the diagrams used in RST. As opposed to RST, the resulting simplified sentences, which are represented as leaf nodes in the tree, cannot be specified prior to the transformation process, since they are not uniform and might require transformations (e.g. paraphrasing) for specific syntactic environments in which they occur. Therefore, the transformation process is carried out in a top-down fashion, starting with the input document and using a set of syntactic rule patterns that define how to split up, transform and recurse on complex syntactic sentence patterns (see Section 3.1.3). Each split will then create two or more simplified sentences that are connected with information about their constituency type (*coordinate* or *subordinate*) and identified rhetorical relation. The constituency type infers the concept of nuclearity, where coordinate sentences (which we will call *core* sentences) represent nucleus spans and subordinate sentences (*context* sentences) represent satellite spans. In this way, we obtain a hierarchical tree representation of the input document similar to RST. We will denote the resulting tree as a *discourse tree*.

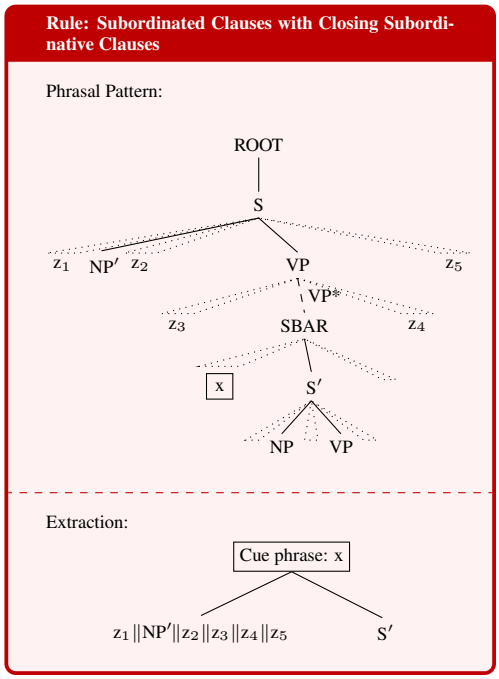


Figure 3: Rule for subordinated clauses with closing subordinative clauses ("||" denotes concatenation).

Note that our approach differs from RST in the following ways: 1) connected sentences do not have to be non-overlapping text spans; 2) our approach works in a top-down fashion where the final simplified sentences result from the transformation process, whereas in RST, the smallest elements of discourse (EDUs) are identified in advance.

3.1.2 Algorithm

The transformation algorithm (see Algorithm 1) takes as an input the document text as a string and outputs its discourse tree. In the initialization step (2), the discourse tree is set up in the form of a single, unclassified coordination node that represents the root of the document's discourse tree. It contains edges to every sentence that has been tokenized out of the document string, which are treated as coordinated leaf nodes. In this way, all sentences are considered as core sentences, thus being equally relevant.

Algorithm 1 Transformation Stage

Input: the document-string str
Output: the discourse tree $discourse\_tree$

1: **function** TRANSFORMATION(str)
2: $discourse\_tree \leftarrow$ initialize as a document root node with leaf-nodes for each sentence in str.
3: TRAVERSEDISCOURSETREE($discourse\_tree$)
4: $discourse\_tree \leftarrow$ further disembed contextual phrases (e.g. $temporal$, $spatial$) out of sentence leaves using the Sentence Simplification system proposed by Niklaus et al. (2016).
5: **return** $discourse\_tree$
6: **end function**

7: **procedure** TRAVERSEDISCOURSETREE($tree$)
8: ▷ Process leaves from left to right
9: **for** $leaf$ in $tree.leaves$ **do**
10: $match \leftarrow$ `False`
11: ▷ Check clausal transformation rules in fixed order
12: **for** $rule$ in $TRANSFORM\_RULES$ **do**
13: **if** $rule.pattern.matches(leaf.parse\_tree)$ **then**
14: $match \leftarrow$ `True`
15: **Break**
16: **end if**
17: **end for**
18: **if** $match ==$ `True` **then**
19: ▷ Disembedding / Simplification
20: $simplified\_sentences \leftarrow rule.disembed(leaf.parse\_tree)$
21: $new\_leaves \leftarrow$ convert $simplified\_sentences$ into leaf nodes. Nodes for subordinate sentences are labelled as `context`, else `core`
22: $new\_node \leftarrow$ create new parent node for new_leaves. If a context sentence is enclosed, the constituency is `subordinate`, else `coordinate`
23: ▷ Rhetorical relation identification
24: $cue\_phrase \leftarrow rule.extract\_cue(leaf.parse\_tree)$
25: $new\_node.relation \leftarrow$ match cue_phrase against a list of rhetorical cue words that are assigned to their most likely triggered rhetorical relation. Different cue word lists are used for different syntactic environments.
26: ▷ Recursion
27: $tree.replace(leaf, new\_node)$
28: TRAVERSEDISCOURSETREE(new_node)
29: **end if**
30: **end for**
31: **end procedure**

After initialization, the discourse tree is traversed and split up recursively in a top-down approach (3, 7). The tree traversal is done by processing sentence leaves in depth-first order, starting from the root node. For every leaf (9), we check if its phrasal parse structure matches one of the fixed ordered set of 16 hand-crafted syntactic rule patterns (see Section 3.1.3) that determine how a sentence is split up into two or more simplified sentence leaves (13). The first matching rule will be used, generating two or more simplified sentences (20) that will again be processed in this way. In addition to generating simplified sentences, each rule also determines the constituency type of the newly created node (22) and returns a cue phrase which is used as a lexical feature for classifying the type of rhetorical relation connecting the split sentences (24, 25). The algorithm terminates when no more rule matches the set of generated sentences. Finally, we use the sentence simplification system of Niklaus et al. (2016) to further disembed contextual phrases out of the simplified sentences from the discourse tree (4). The complete source code of our

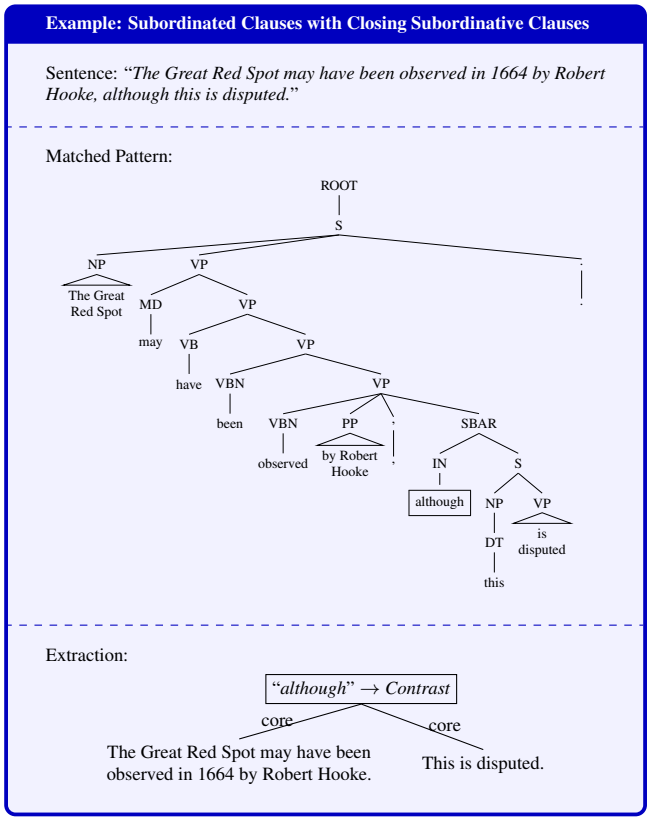


Figure 4: Example for subordinated clauses with closing subordinative clauses.

framework can be found online[1].

3.1.3 Transformation Rules

The set of hand-crafted transformation rules used in the simplification process are based on syntactic and lexical features that can be obtained from a sentence's phrase structure, which we generated with the help of Stanford's pre-trained lexicalized parser (Socher et al., 2013). These rules make use of regular expressions over the parse trees encoded in the form of Tregex patterns (Levy and Andrew, 2006). They were heuristically determined in a rule engineering process whose main goal was to provide a best-effort set of rules, targeting the challenge of being applied in a recursive fashion and to overcome biased or erroneous parse trees. During our experiments, we developed a fixed execution order of rules which achieved the highest F_1-score in the evaluation setting.

Each rule pattern accepts a sentence's phrasal parse tree as an input and encodes a certain parse tree pattern that, in case of a match, will extract textual parts out of the tree that are used to produce the following information:

Simplified sentences The simplified sentences are generated by combining extracted parts from the parse tree. Note that some of the extracted parts such as phrases also require to be arranged and complemented in a way so that they form grammatically correct sentences (*paraphrasing*), aiming for a canonical subject-predicate-object structure.

Constituency Our notion of constituency depicts the contextual hierarchy between the simplified sentences. If all sentences can be considered to be equally important, we will call it a *coordination*. In this case, all generated sentences are labeled as *core* sentences. Otherwise, i.e. if one sentence provides background information or further specifies another sentence, we use the term *subordinate* and label the sentence as *context* sentence. In most of the cases, our framework uses the syntactic information of coordinated and subordinated clauses to infer the same type of (semantic) constituency with respect to the rhetorical relations. An example where this does not apply are rules for identifying attributions. In these cases, we consider an actual statement (e.g. "*Economic activity continued to increase.*") to be more important than the fact that this was stated by some entity (e.g. "*This was what the Federal Reserve noted.*").

Classified rhetorical relation Both syntactic and lexical features are used during the transformation process to identify and classify rhetorical relations that hold between the simplified sentences. Syntactic features are manifested in the phrasal composition of a sentence's phrasal parse tree, whereas lexical features are extracted from the parse tree as so-called *cue phrases*. Cue phrases (often denoted as *discourse markers*) represent a sequence of words that are likely to include rhetorical cue words (e.g. "*because*", "*after that*" or "*in order to*") indicating a certain rhetorical relation. The way these cue phrases are extracted from a sentence's parse tree is specific to each rule pattern. For determining potential cue words and their positions in specific syntactic environments, we use the work of Knott and Dale (1994). The extracted cue phrases are then used to infer the kind of rhetorical relation. For this task, we use a predefined list of rhetorical cue words which are assigned to the rhetorical relation that they most likely trigger. The mapping of cue words to rhetorical relations was adapted from the work of Taboada and Das (2013). If one of these cue words is present in the cue phrase, the corresponding rhetorical relation is set between the newly constructed sentences.

An example for a transformation rule, targeting subordinate clausal constructions, is illustrated in Figure 3. Figure 4 shows its application to an example sentence. The full rule set can be found in the supplementary material online[2]. Here, a detailed example that demonstrates the process of turning a complex NL sentence into a discourse tree is provided, too.

3.2 Relation Extraction

[1] https://github.com/Lambda-3/Graphene

[2] https://github.com/Lambda-3/Graphene/tree/master/wiki/supplementary

The last step in our framework is realized by the actual relation extraction task. To do so, each of the previously generated leaf sentences is given as an input to any Open IE system that is able to extract one (or more) binary propositions from a given sentence. In order to map the identified rhetorical and contextual relationships from the discourse tree to the extracted propositions, we need to determine which of the extracted propositions embodies the main statement of the corresponding sentence. We use the following heuristic to decide whether an extracted proposition represents the main statement of the input sentence: 1) the head verb of the input sentence is contained in the relational phrase rel of the proposition $\langle arg_1; rel; arg_2 \rangle$ (e.g. $\langle$*Employees*; *are* nominated; *for the program*$\rangle$); or 2) the head verb of the input sentence equals the object argument arg_2 of the proposition (e.g. $\langle$*Employees*; *are*; nominated$\rangle$). We will denote any proposition that matches those criteria as a *representative* of the corresponding sentence. Once determined, the identified rhetorical and contextual relations that hold between two sentences from the discourse tree can be directly transferred as (semantically typed) additional arguments to the corresponding representatives, thus generating n-ary relations.

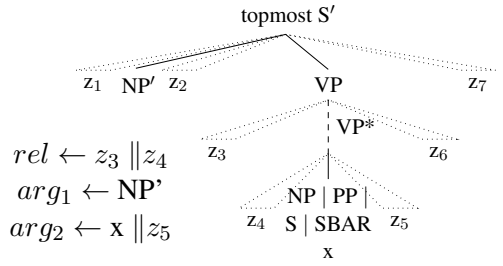


Figure 5: Graphene's default relation extraction pattern ("‖" denotes concatenation).

In this way, extractions from any Open IE system that uses our framework can be enriched with semantic information. An example of the semantically enriched output produced by the Open IE system OLLIE when using our framework as a preprocessing step is shown in Figure 6. In our baseline implementation Graphene, we use one syntactic rule pattern that operates on the sentence's phrase structure and identifies simple subject-predicate-object structures. The default pattern, which extracts one tuple for each sentence, is shown in Figure 5.

```
He nominated Sonia Sotomayor on May 26, 2009 to replace David Souter; she was confirmed on August 6, 2009,
becoming the first Supreme Court Justice of Hispanic descent.

OLLIE (alone):
(1) she                 was confirmed on                  August 6, 2009
(2) He                  nominated Sonia Sotomayor on      May 26
(3) He                  nominated Sonia Sotomayor         2009
(4) He                  nominated 2009 on                 May 26
(5) Sonia Sotomayor     be nominated 2009 on              May 26
(6) He                  nominated 2009                    Sonia Sotomayor
(7) 2009                be nominated Sonia Sotomayor on   May 26

Ollie (using framework):
(8)  #1    0    he    nominated    Sonia Sotomayor
("a)     S:PURPOSE     to replace David Souter .
("b)     S:TEMPORAL    on May 26 , 2009 .
(9)  #2    0    she    was becoming    the first Supreme Court Justice of Hispanic descent
```

Figure 6: Comparison of the output of OLLIE alone and with the framework.

4 Evaluation

In order to evaluate the performance of our Open IE reference implementation Graphene, we conduct an automatic evaluation using the Open IE benchmark framework proposed in Stanovsky and Dagan (2016), which is based on a QA-Semantic Role Labeling (SRL) corpus with more than 10,000 extractions over 3,200 sentences from Wikipedia and the Wall Street Journal[3]. This benchmark allows us to compare our system with a variety of current Open IE approaches in recall and precision. Moreover, we investigate whether our two-layered transformation process of clausal and phrasal disembedding improves the performance of state-of-the-art Open IE systems when applied as a preprocessing step.

[3]available under `https://github.com/gabrielStanovsky/oie-benchmark`

4.1 Experimental Setup

To conduct an automatic comparative evaluation, we integrate Graphene into Stanovsky and Dagan (2016)'s Open IE benchmark framework, which was created from a QA-SRL dataset where every verbal predicate was considered as constituting an own extraction. Since Graphene's structured output is not designed to yield extractions for every occurrence of a verbal predicate, we use an alternative relation extraction implementation which is able to produce more than one extraction from a simplified core sentence. To match extractions to reference propositions from the gold standard, we use the method described in Stanovsky and Dagan (2016), following He et al. (2015) by matching an extraction with a gold proposition if both agree on the grammatical head of both the relational phrase and its arguments. Since n-ary relational tuples with possibly more than two arguments have to be compared, we assign a positive match if the relational phrase and at least two of their arguments match. Since the gold standard is composed of n-ary relations, we add all contexts (S) of an extraction as additional arguments besides arg_1 and arg_2. We assess the performance of our system together with the state-of-the-art systems ClausIE, OLLIE, OpenIE-4 (Mausam, 2016), PropS (Stanovsky et al., 2016), REVERB and Stanford Open IE.

After evaluating Graphene as a reference implementation of the proposed approach, we investigate how clausal and phrasal disembedding applied as a preprocessing step affects the performance of other Open IE systems. Therefore, we compare the performance of ClausIE, OLLIE, OpenIE-4, REVERB and Stanford Open IE on the raw input data with their performance when operating inside of our framework where they act as different relation extraction implementations that take the simplified sentences of the transformation stage as an input.

4.2 Results and Discussion

Figure 7 shows the precision-recall curve for each system within the Open IE benchmark evaluation framework. In our experiments, with a score of 50.1% in average precision, our reference implementation Graphene achieves the best performance of all the systems in extracting accurate relational tuples, followed by OpenIE-4 (44.6%) and PropS (42.4%). Considering recall, Graphene (27.2%) is able to compete with other high-precision systems, such as PropS (26.7%). However, it does not reach the recall rate of ClausIE (33.0%) or OpenIE-4 (32.5%). This lack in recall was expected, since Graphene was not designed to create fine-grained relations for every possible verb, as opposed to the gold standard, but rather determines the main relations with attached (contextual) arguments that often contain some of these verbal expressions. In an in-depth analysis of the output, we proved that only 33.72% of unmatched gold extractions were caused by a wrong argument assignment of Graphene. In the majority of cases (66.28%), Graphene did not extract propositions with a matching relational phrase. We further calculated that in 56.80% of such cases, the relational phrase of a missed gold standard extraction was actually contained inside an argument, e.g.:

"*Japan may be a tough market for outsiders to penetrate, and the U.S. is hopelessly behind Japan in certain technologies.*"

- Unmatched Gold-Extraction: ⟨*outsiders*; *may penetrate*; *Japan*⟩
- Graphene-Extractions:
  1. ⟨*Japan*; *may be*; *a tough market for outsiders to penetrate*⟩
  2. ⟨*the U.S.*; *is hopelessly behind Japan*; *in certain technologies*⟩

This number even grows to 67.89% when considering the lemmatized form of the head:
"*The seeds of "Raphanus sativus" can be pressed to extract radish seed oil.*"

- Unmatched Gold-Extraction: ⟨*radish seed oil*; *can be extracted*; *from The seeds*⟩
- Graphene-Extractions:
  1. ⟨*the seeds of "Raphanus sativus"*; *can be pressed*; *to extract radish seed oil*⟩

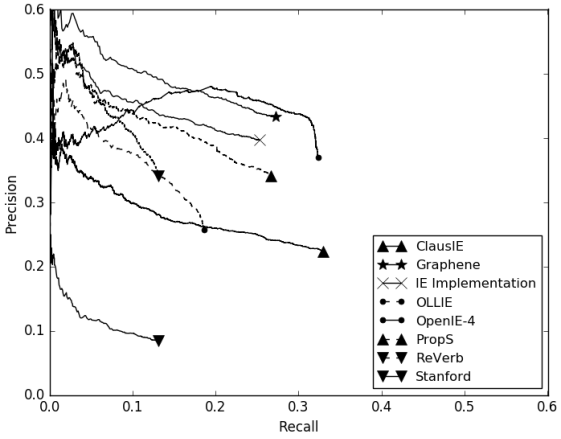


Figure 7: Performance of Graphene.

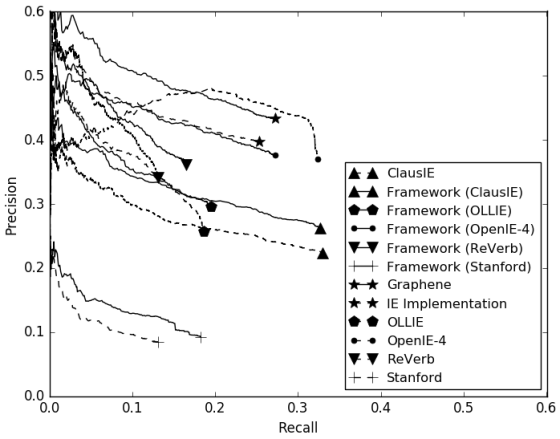


Figure 8: Improvements of state-of-the-art systems when operating as RE component of our framework.

Another 5.22% of unmatched relational phrases would have been recognized by Graphene if they were compared based on their lemmatized head.

Regarding the overall Area Under the Curve (AUC) score, OpenIE-4 is the best performing system with a score of 14.5%, closely followed by Graphene (13.6%), PropS (11.3%) and ClausIE (9.3%).

The precision-recall curve in Figure 8 shows that when using clausal and phrasal disembedding provided by our framework, all tested systems except for OpenIE-4 (-18%) gain in AUC. The highest improvement in AUC was achieved by Stanford Open IE, yielding a 63% AUC increase over the output of Stanford Open IE as a standalone system. AUC scores of OLLIE and REVERB improve by 34% and 22%. While OLLIE and REVERB primarily profit from a boost in recall (+4%, +26%), ClausIE mainly enhances precision (+16%). Furthermore, the performance of our reference relation extraction implementation increases both in precision (+10%) and recall (+8%) when applied as the relation extraction component inside of our system Graphene (+19% AUC).

5 Conclusion

In this paper, we introduce a novel Open IE approach that presents an innovative two-layered hierarchical representation of syntactically simplified sentences in the form of core facts and accompanying contexts that are semantically linked by rhetorical relations. In that way, the semantic connection of the individual components is preserved, thus allowing to fully reconstruct the informational content of the input.

The results of a comparative analysis conducted on a large-scale benchmark framework showed that with a score of 50.1%, our baseline system Graphene achieves the best average precision, while also providing a recall rate of 27.2% that is comparable to that of other high-precision systems. Moreover, we demonstrated that by using clausal and phrasal disembedding as a preprocessing step, the AUC score of state-of-the-art Open IE systems can be improved by up to 63%. In summary, we were able to show that by generating a two-layered representation of core and contextual information, the relations extracted by state-of-the-art Open IE systems can be semantically enriched without losing in precision or recall. Moreover, by using clausal and phrasal disembedding techniques, contextual information is detached from the core propositions and transformed into additional arguments, thus avoiding the problem of overspecified argument phrases and yielding a more compact structure. By enhancing the resulting syntactically sound sentence representation with rhetorical relations, semantically typed and interconnected relational tuples are created that may benefit downstream artificial intelligence tasks. In future work, we plan to evaluate the classification of the rhetorical relations, examine to what extent other NL Processing tasks such as QA systems may benefit from the results produced by our framework and investigate the creation of big knowledge graphs for QA by performing a large-scale Wikipedia extraction.

References

Alan Akbik and Alexander Löser, 2012. *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, chapter KrakeN: N-ary Facts in Open Information Extraction, pages 52–56. Association for Computational Linguistics.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China, July. Association for Computational Linguistics.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 2670–2676, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: Clause-based open information extraction. In *Proceedings of the 22Nd International Conference on World Wide Web*, pages 355–366, New York, NY, USA. ACM.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 643–653.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Alistair Knott and Robert Dale. 1994. Using linguistic phenomena to motivate a set of coherence relations. *Discourse processes*, 18(1):35–62.

Roger Levy and Galen Andrew. 2006. Tregex and tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of the fifth international conference on Language Resources and Evaluation*, pages 2231–2234.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534, Jeju Island, Korea, July. Association for Computational Linguistics.

Mausam. 2016. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 4074–4077.

Filipe Mesquita, Jordan Schmidek, and Denilson Barbosa. 2013. Effectiveness and efficiency of open relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 447–457. Association for Computational Linguistics.

Christina Niklaus, Bernhard Bermeitinger, Siegfried Handschuh, and André Freitas. 2016. A sentence simplification system for improving relation extraction. In *Prooceedings of COLING 2016: System Demonstrations, The 26th International Conference on Computational Linguistics, Osaka, Japan, December 11-16, 2016*, pages 170–174.

Jordan Schmidek and Denilson Barbosa. 2014. Improving open relation extraction via sentence re-structuring. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA).

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing With Compositional Vector Grammars. In *ACL*.

Gabriel Stanovsky and Ido Dagan. 2016. Creating a large benchmark for open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page (to appear), Austin, Texas, November. Association for Computational Linguistics.

Gabriel Stanovsky, Jessica Ficler, Ido Dagan, and Yoav Goldberg. 2016. Getting more out of syntax with props. *CoRR*, abs/1603.01648.

Maite Taboada and Debopam Das. 2013. Annotation upon annotation: Adding signalling information to a corpus of discourse relations. *D&D*, 4(2):249–281.

Fei Wu and S. Daniel Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127. Association for Computational Linguistics.

An Exploration of Three Lightly-supervised Representation Learning Approaches for Named Entity Classification

Ajay Nagesh and Mihai Surdeanu
University of Arizona, Tucson, AZ, USA
{ajaynagesh, msurdeanu}@email.arizona.edu

Abstract

Several semi-supervised representation learning methods have been proposed recently that mitigate the drawbacks of traditional bootstrapping: they reduce the amount of semantic drift introduced by iterative approaches through one-shot learning; others address the sparsity of data through the learning of custom, dense representation for the information modeled. In this work, we are the first to adapt three of these methods, most of which have been originally proposed for image processing, to an information extraction task, specifically, named entity classification. Further, we perform a rigorous comparative analysis on two distinct datasets. Our analysis yields several important observations. First, all representation learning methods outperform state-of-the-art semi-supervised methods that do not rely on representation learning. To the best of our knowledge, we report the latest state-of-the-art results on the semi-supervised named entity classification task. Second, one-shot learning methods clearly outperform iterative representation learning approaches. Lastly, one of the best performers relies on the mean teacher framework (Tarvainen and Valpola, 2017), a simple teacher/student approach that is independent of the underlying task-specific model.

1 Introduction

The paucity of labeled datasets presents tremendous challenges to training machine learning (ML) systems in the real world. Most natural language processing (NLP) tasks are data hungry. The recent success of deep learning for NLP tasks is at least partially due to the fact that such models are exposed to large quantities of labeled data. However, obtaining such rich labeled data is a costly proposition and seldom feasible in many realistic scenarios.

Semi-supervised learning (SSL) mitigates the supervision cost by combining a minimal amount of labeled data with a large amount of unlabeled data. In information extraction (IE), SSL often takes the form of bootstrapping, which starts with a few seed examples, e.g., "Barack Obama" as an example of a person's name, and continues with an iterative approach that alternates between learning extraction patterns such as word n-grams, e.g., the pattern "`@ENTITY , former president`" could be used to extract person names, and applying these patterns to extract the desired structures (entities, relations, etc.) (Riloff, 1996; Abney, 2007; Carlson et al., 2010b; Gupta and Manning, 2014; Gupta and Manning, 2015, inter alia). There are, however, two drawbacks to this direction. First, as learning in this iterative framework advances, the task often *drifts semantically* (Komachi et al., 2008) into a related but different space, e.g., from learning women names into learning flower names (McIntosh, 2010; Yangarber, 2003). Second, the statistics used to estimate the quality of the learned patterns are often *brittle* due to the sparsity of the extraction patterns.

Recently, several semi-supervised representation learning methods have been proposed to mitigate these issues. One direction learns custom representations for both patterns and the structures to be extracted to mitigate the sparsity mentioned above (see Section 4). A different direction, illustrated

Proceedings of the 27th International Conference on Computational Linguistics, pages 2312–2324
Santa Fe, New Mexico, USA, August 20-26, 2018.

by the ladder networks (LN) and mean teacher (MT) frameworks (Rasmus et al., 2015; Tarvainen and Valpola, 2017), replaces the iterative bootstrapping with one-shot learning driven by teacher/student architectures, trained to maximize consistency of predictions between the teacher and the student on the unlabeled data (Sections 5 and 6). Despite this progress, several important questions remain unanswered. First, most of these methods have been proposed for vision, and it is unclear how useful they are for lightly-supervised natural language processing tasks. Second, it is unknown how well these methods compare against "traditional" bootstrapping that does not use representation learning. Third, it is unclear how these representation learning methods compare against each other.

In this work, we answer the above questions by adapting all these methods to the same IE task, and performing a rigorous comparative analysis. Specifically, the contributions of this paper are the following:

- We are the first to adapt the mean teacher (MT) framework (Tarvainen and Valpola, 2017) to a semi-supervised IE task, in particular named entity classification (NEC).
- We implement and evaluate the three semi-supervised representation learning approaches summarized above for the same NEC task, using two distinct datasets (Pradhan et al., 2013; Tjong Kim Sang and De Meulder, 2003), the same inputs, and the same evaluation measures to understand how they compare both against each other and against more traditional semi-supervised NLP approaches.
- We perform a thorough comparative analysis of all these methods, which highlights several important observations. First, our analysis confirms that all these representation learning methods outperform state-of-the-art semi-supervised methods that do not rely on representation learning (Gupta and Manning, 2015; Gupta and Manning, 2014; Zhu and Ghahramani, 2002). Second, one-shot learning methods (the teacher/student approaches here) have superior performance compared to iterative representation learning approaches that are driven by objective functions tailored for the task at hand (e.g., which maximize the similarity of embeddings for entities in the same class). Lastly, the teacher/student approaches perform comparatively – each outperforms the other on one dataset – despite the fact that MT is a simpler framework than LN. That is, in the mean teacher framework both student and teacher are decoupled whereas in ladder networks there are connections between the two, which makes porting LNs to new tasks more difficult.

2 Related Work

There is a large body of work in semi-supervised learning for NLP, which encompasses many different types of techniques such as self-training or bootstrapping (Riloff, 1996; Abney, 2007; Carlson et al., 2010a; Carlson et al., 2010b; McIntosh, 2010; Gupta and Manning, 2015, inter alia), co-training (Blum and Mitchell, 1998), or graph-based methods such as label propagation (Delalleau et al., 2005). Among these, perhaps the most-widely adopted approach in the NLP field is bootstrapping, which has been used in many applications, including information extraction (Carlson et al., 2010a; Gupta and Manning, 2014; Gupta and Manning, 2015), lexicon acquisition (Neelakantan and Collins, 2015), named entity classification (Collins and Singer, 1999) and sentiment analysis (Rao and Ravichandran, 2009). However, as mentioned, most of these approaches suffer from brittle statistics due to the sparsity of the patterns learned, and from semantic drift, due to their iterative nature. The methods we investigate in this paper address these two limitations through representation learning and one-shot learning.

Auto-encoder frameworks have received considerable attention in the machine learning community recently. Such frameworks include recursive auto-encoders (Socher et al., 2011), denoising auto-encoders (Vincent et al., 2008), and others. They are primarily used as a pre-training mechanism before supervised training. Recently, such networks have also been used for semi-supervised learning as they are more amenable to combining supervised and unsupervised components of the objective functions (Zhai and Zhang, 2015).

Ladder networks (LN) are stacked denoising auto-encoders with skip-connections in the intermediate layers (Rasmus et al., 2015; Valpola, 2014). LNs have been shown to produce state-of-the-art perfor-

mance on both supervised and semi-supervised tasks on the MNIST dataset in image processing. Our work is among the first to apply LNs to NLP. While similar in spirit to Zhang et al. (2017), the only other work we found that applies a denoising auto-encoder to a semi-supervised spelling correction task, our work is much simpler, since it uses a multi-layer perceptron instead of convolution-deconvolution operations (see Section 5). Further, we demonstrate that LNs perform very well on a complex IE task, considerably outperforming several state-of-the-art approaches. In the same space, teacher-student networks have shown to provide state-of-the-art semi-supervised learning results on several image processing tasks. The most prominent example of such a network is the Mean Teacher framework (Tarvainen and Valpola, 2017).To our knowledge, we are the first to apply the MT framework to an IE task (Section 6).

Distributed representations of words serve as underlying representation for many NLP tasks. For example, Levy and Goldberg (2014a) generalized the `word2vec` algorithm (Mikolov et al., 2013) to use any arbitrary context instead of just individual words. Faruqui et al. (2015) demonstrated that embeddings learned without supervision can be retro-fitted to better conform to some semantic lexicon. Generating custom embeddings that encode more specific semantic properties has been shown to be useful for downstream tasks (Sharp et al., 2016). Riedel et al. (2013; Toutanova et al. (2015; Toutanova et al. (2016) used knowledge bases in conjunction with surface patterns to learn custom representations for relation extraction. Note, however, that most of these works that customize embeddings for a specific task rely on some form of supervision. In contrast, our approach that learns a custom representation for the NEC task is lightly supervised, with a only few seed examples per category (Section 4).

3 Task and Approaches

Named entity classification

We implement and evaluate the proposed semi-supervised learning approaches on the task of named entity classification (NEC), defined as identifying the correct label of an entity mention in a given context. In our setting, the context of a mention is defined as all the patterns that match that specific mention in its context. In this work, we consider patterns to be n-grams of size up to 4 tokens on either side of an entity. For instance, "`@ENTITY , former President`" is one of the patterns learned for the class `person`. Using all these patterns as input, the classifier must infer the mention's correct label. Importantly, the NEC task can be defined at mention level, i.e., as the classification of *individual mentions* of named entities as defined above, or at entity level, i.e., as the identification of all labels that apply to *all mentions* of a given entity jointly (e.g., "Washington" = {`person`, `location`}). Here we focus on mention classification, although in some of our evaluations we revert to entity classification, to be able to compare against other approaches.

Semi-supervised representation learning

We propose three representation learning approaches for the NEC task: (i) Embedding-based bootstrapping (Emboot), (ii) ladder networks (LN), and (iii) mean teacher (MT) architectures.

Emboot (Valenzuela-Escárcega et al., 2018) is an adaptation of iterative bootstrapping-based approaches, coupled with a component that learns custom representations (embeddings) for both the entities and the patterns learned. As mentioned, such iterative approaches suffer from semantic drift, i.e., as the learning advances, the learning often drifts into a related but different space. LN and MT are based on the emerging paradigm of teacher-student architectures: both approaches aim to reconcile differences between the teacher and the student models by minimizing a consistency cost when their predictions do not concur. Teacher-student architectures are more robust to semantic drift due to their single-shot semi-supervised learning algorithms.

Note that, of the above architectures, Emboot and MT additionally learn custom representations for the unsupervised data provided which could potentially be informative representations to be used in a downstream component aimed at interpreting the classifier decisions. Further, MT is the most flexible/customizable of all these architectures, as it is largely independent of the underlying task-specific model. We show in our experiments that this simplicity is coupled with good performance on two different datasets.

In the next three sections, we discuss each of these approaches in detail.

4 Embedding-based Bootstrapping (Emboot)

This algorithm (based on our earlier work (Valenzuela-Escárcega et al., 2018)) iteratively grows a pool of known entities (entPool$_c$) and patterns (patPool$_c$) for each category of interest c, and learns custom embeddings for both. These pools are initialized with a few seed examples (seeds$_c$) for each category. For example, in our experiments we initialize the pool for a `person names` category with 10 names such as *Bill Clinton* and *Mother Teresa*. Then the algorithm iteratively applies the following three steps for a specified number of epochs:

(1) *Learning custom embeddings*: The algorithm learns custom embeddings for all entities and patterns observed in the dataset, using the current entPool$_c$s. We train our embeddings for both entities and patterns by maximizing the objective function J:

$$J = \text{Skip-gram} + \text{Attract} + \text{Repel} \tag{1}$$

where the individual components of the objective function designed to model both the unsupervised, language model part of the task as well as the light supervision coming from the seed examples. The *Skip-gram* term is similar to Eq. 4 of Mikolov et al. (2013) (skip-gram with negative sampling), but, crucially, adapted to operate over *entities* (rather than words), and a context consisting of the bag of all *patterns* that match each entity (rather than context words). The *Attract* term, encourage entities or patterns in the same pool to be close to each other (*E.g.* two person entities should be attracted to each other), whereas the *Repel* term encourages that the pools be mutually exclusive, which is a soft version of the counter training approach of Yangarber (2003) or the weighted mutual-exclusive bootstrapping algorithm of McIntosh and Curran (2008) (e.g., person names should be far from organization names in the semantic embedding space). They are depicted by the following two equations:

$$\text{Attract} = \sum_{P} \sum_{x1,x2 \in P} \log(\sigma(V_{x1}^{\top} V_{x2})) \tag{2}$$

$$\text{Repel} = \sum_{\substack{P1,P2 \\ \text{if } P1 \neq P2}} \sum_{x1 \in P1} \sum_{x2 \in P2} \log(\sigma(-V_{x1}^{\top} V_{x2})) \tag{3}$$

where P is the entity/pattern pool for a category, $x1, x2$ are entities/patterns in said pool in Eq 2. and $P1, P2$ are different pools, and $x1$ and $x2$ are entities/patterns in $P1$, and $P2$, respectively.

(2) *Pattern promotion*: We generate the patterns that match the entities in each pool entPool$_c$, rank those patterns using point-wise mutual information (PMI) with the corresponding category, and select the top ranked patterns for promotion to the corresponding pattern pool patPool$_c$.

(3) *Entity promotion*: Entities are promoted to entPool$_c$ using a classifier that estimates the likelihood of an entity belonging to each class (Gupta and Manning, 2015). We differ from Gupta and Manning (2015) in that we use a multi-class classifier instead of many binary classifiers, and in the features used. Our feature set includes, for each category c: (a) edit distance between the candidate entity e and current e_cs $\in$ entPool$_c$, (b) the PMI (with c) of the patterns in patPool$_c$ that matched e in the training documents, and (c) similarity between e and e_cs in some semantic space. For the latter feature group, we use two sets of vector representations for entities. The first is the set of custom embedding vectors learned in step (1) for entities. The second includes pre-trained word embeddings (for multi-word entities, we simply average the embeddings of the component words). We use these vectors to compute the cosine similarity score of a given candidate entity e to the entities in entPool$_c$, and add the average and maximum similarities as features. The top 10 entities classified with the highest confidence for each class are promoted to the corresponding entPool$_c$ after each epoch.

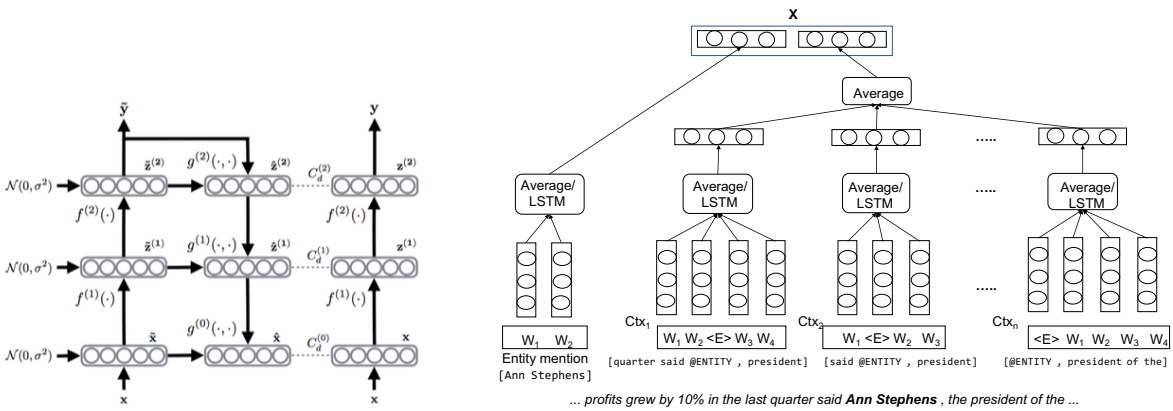


Figure 1: Architecture of the ladder network (Rasmus et al., 2015) (left), and network initialization of input vector X for the NEC task, where X captures the entity mention modeled and all the patterns matching it in the corresponding sentence (right).

5 Ladder Networks

Ladder networks (LN) (Rasmus et al., 2015) is a neural network architecture designed to use unsupervised learning as a scaffolding for the supervised task. It is a denoising autoencoder (DAE) with noise introduced in every layer. It consists of two sets of encoders, a clean one and another corrupted with noise, and a decoder. In addition, there are skip connections between the encoder and decoder (see left part of Figure 1). The ladder network is defined as follows:

$$\tilde{X}, \tilde{Z}^{(1)}, \dots \tilde{Z}^{(L)}, \tilde{y} = f_{corr}(X) \tag{4}$$

$$X, Z^{(1)}, \dots Z^{(L)}, y = f_{clean}(X) \tag{5}$$

$$\hat{X}, \hat{Z}^{(1)}, \dots \hat{Z}^{(L)} = g(\tilde{Z}^{(1)}, \dots \tilde{Z}^{(L)}) \tag{6}$$

where X, $\tilde{X}$ and $\hat{X}$ is an input datapoint, its corrupted version, and its reconstruction, respectively; $Z^{(l)}$ and $\tilde{Z}^{(l)}$ are clean and corrupted hidden representations in the l-th layer; and, lastly, y, $\tilde{y}$ are the clean and corrupted activations, converted to a probability distribution over the label set (using a softmax layer).

For our NEC task, X is the concatenation of an entity mention and its context embedding vectors, and y represents the mention's label (e.g., `person`). We initialize the words in the entities and patterns around them with pre-trained word embeddings. To obtain a single embedding for an entity mention and its context we: (a) average word embeddings to obtain a single embedding for the entity mention and each of its patterns; and (b) average the resulting pattern embeddings to produce the embedding of the corresponding context. We then concatenate the mention's embedding and context embedding to be given as input to the ladder network. This process is depicted schematically in the right part of Figure 1. We introduce noise by adding a random Gaussian to the pre-trained embeddings with a fixed mean and standard deviation.

The objective function is a combination of a supervised training cost and unsupervised reconstruction costs at each layer (including the hidden layers):

$$Cost = -\sum_{n=1}^{N} logP(\tilde{y_n} = y_n^* | X_n) + \sum_{n=N+1}^{M} \sum_{l=1}^{L} \lambda_l ReconstCost(Z_n^{(l)}, \hat{Z}_n^{(l)}) \tag{7}$$

where the first term is the supervised cross-entropy based on the N labeled datapoints $(X_1, y_1^*), (X_2, y_2^*), \dots (X_N, y_n^*)$, and the second term is the reconstruction loss on the M unlabeled datapoints $X_{N+1}, X_{N+2}, \dots X_{N+M}$, for each layer l. Typically $M \gg N$.

Pezeshki et al. (2016) analyze the different architectural aspects of LN and note that the lateral connections and corresponding reconstruction costs (second term in the above cost function) are critical for

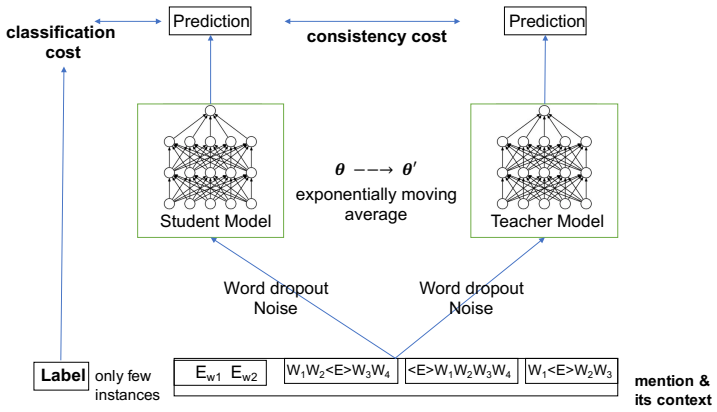


Figure 2: Mean Teacher architecture for the named entity classification task. E_{wi} are words contained in the current entity mention; W_i are words that occur in a pattern that matches the entity mention; and <E> is a placeholder token to indicate where the entity mention occurs inside a pattern.

semi-supervised learning. In other words, it is important for unlabeled data to be used for regularization to be able to learn good abstractions in the different layers. We have similar observations for the NEC task (see Experiments).[1]

6 Mean Teacher Framework

The mean teacher framework (Tarvainen and Valpola, 2017) belongs to a general class of teacher-student algorithms that learn with unlimited unlabeled data and limited supervision. The broad motivation behind MT is provided by the shortcomings of auto-encoders, which suffer from confirmation bias in semi-supervised settings, i.e., when they use their own predictions instead of gold labels (Tarvainen and Valpola, 2017; Laine and Aila, 2016). To mitigate confirmation bias, the teacher weights in MT are averaged over the training epochs, akin to the averaged perceptron. This provides a more robust scaffolding that the student model can rely on during training when gold labels are not available (Tarvainen and Valpola, 2017).

The MT architecture is summarized in Figure 2. It consists of two models with identical architectures where one is termed the teacher, and the other the student. The weights in the teacher network are set through an exponential moving average of the student weights. The input to both of these models is the same datapoint, which is augmented by some noise that is specific to each model. In the case of our NEC task, we introduce noise through random word dropout in the patterns that apply to an entity mention.

The MT algorithm is designed for semi-supervised tasks, where only a few of the data points are labeled. The cost function is a combination of two different type of costs: classification and consistency. The classification cost applies to supervised data points, and can be instantiated with any supervised cost function (e.g., we used categorical cross-entropy). The consistency costs is used for unlabeled data points, and aims to minimize the differences in predictions between the teacher and the student models. The consistency cost, J is:

$$J(\theta) = \mathrm{E}_{x,\eta',\eta}\left[\|f(x,\theta',\eta') - f(x,\theta,\eta)\|^2\right] \tag{8}$$

where this function is defined as the expected distance between the prediction of the teacher model (with weights θ' and noise η') and the prediction of the student model (with weights θ and noise η).

The student weights are updated via back propagation. The teacher weights are deterministically updated after each mini batch of gradient descent on the student, through an exponentially weighted moving average (EMA) from the previous iterations of the student, given by the following equation:

[1]LNs for semi-supervised named entity classification was presented by our earlier work (Nagesh and Surdeanu, 2018). In this work, we compare LNs with MT and provide a more in-depth analysis of these frameworks under the umbrella of teacher-student architectures.

$$\theta_t^{'} = \alpha\theta_{t-1}^{'} + (1-\alpha)\theta_t \quad (9)$$

where $\theta_t^{'}$ is defined at training step t as the EMA of successive weights θ (Tarvainen and Valpola, 2017).

We explored two different architectures for the student model:

Simple model (MT_simple): In this model, we used bag-of-word averaging to obtain embeddings for entity mentions and the patterns matching them. In particular, we: (a) average word embeddings to obtain a single embedding for the entity mention and each of its patterns; and (b) average the resulting pattern embeddings to produce the embedding of the corresponding context. We then concatenate the mention's embedding and context embedding to be given as input to the mean teacher framework. This process is depicted schematically in the right part of Figure 1. This replicates the input processing in LN.

Custom Embedding model (MT_custom): We use a bidirectional LSTM layer on top of the word embedding layer. We train a biLSTM on both the entity tokens and a separate LSTM for each of the pattern tokens. We average each of pattern LSTM representations and concatenate the entity and pattern representations as depicted in the right part of Figure 1. The addition of the LSTM layer, enables this model learn custom embeddings. These embeddings are further analyzed in Section 7.

For both student and teacher models, we initialized the words in the entity mentions and corresponding patterns with pre-trained word embeddings. We have a *linear-ReLU-linear* fully connected layers from the embedding layer before computing the loss.

7 Experiments

7.1 Experimental Setup[2]

Datasets: We used two datasets, the CoNLL-2003 shared task dataset (Tjong Kim Sang and De Meulder, 2003), which contains 4 entity types, and the OntoNotes dataset (Pradhan et al., 2013), which contains 11[3], both of which are benchmark datasets for supervised named entity recognition (NER). These datasets contain marked entity boundaries with labels for each marked entity. Here we only use the entity boundaries but *not* the labels of these entities during the training of our bootstrapping systems. To simulate learning from large texts, we ran the actual experiments on the *train* partitions.[4]

Baselines: We compared against two baselines:

Explicit Pattern-based Bootstrapping (EPB): This is our implementation of the state-of-the-art bootstrapping system of Gupta and Manning (2015), adapted to NEC. The algorithm grows a pool of known entities and patterns for each category of interest, from a few seed examples per category, by iterating between pattern promotion and entity promotion. The former is implemented using a ranking formula driven by the point-wise mutual information (PMI) between each pattern with the corresponding category; the top ranked patterns are promoted to the pattern pool in each iteration. The latter component promotes entities using a classifier that estimates the likelihood of an entity belonging to each class. Our feature set includes, for each category c: (a) edit distance between the candidate entity e and known entities for c; (b) the PMI (with c) of the patterns in the pool of c that matched e in the training documents; and (c) similarity between e and entities in c's pool in some semantic space.[5] Entities classified with the highest confidence for each class are promoted to the corresponding pool after each epoch.

[2]For ease of reproducibility, we release the code for the various semi-supervised learning algorithms at `https://github.com/clulab/releases/tree/master/coling2018-ssl-nec`

[3]We excluded numerical categories such as `DATE`.

[4]Although we run the evaluation on the train dataset, we do not use the labels present in it, as it is a semi-supervised task. In general, in any machine learning dataset, usually the test dataset is quite small compared to train. Since we wanted to analyze the performance in a larger context, we used the train dataset (without the labels).

[5]We used pre-trained word representations, averaged for multi-word entities, to compute cosine similarities between pairs of entities.

	CoNLL	*Ontonotes*
EPB	40.75	21.07
LP	26.07	19.40
Emboot	67.97	45.20
Ladder	68.92 (±2.9)	65.20 (±3.2)
MT_simple	69.50 (±2.7)	51.10 (±4.8)
MT_custom	66.81 (±3.5)	36.17 (±2.5)

Table 1: Overall Accuracies (entity-based)

	CoNLL	*Ontonotes*
Ladder	71.58 (± 1.6)	72.04 (± 0.7)
MT_simple	73.91 (± 3.4)	66.27 (± 1.9)
MT_custom	68.14 (± 4.3)	64.04 (± 3.8)

Table 2: Overall Accuracies (mention–based)

Label Propagation (LP): We used the implementation available in the `scikit-learn` package of the LP algorithm (Zhu and Ghahramani, 2002).[6] In each bootstrapping epoch, we run LP, select the entities with the lowest entropy, and add them to their top category. Each entity is represented by a feature vector that contains the co-occurrence counts of the entity and each of the patterns that matches it in text.[7]

Settings: For each entity mention, we consider a n-gram window of size 4 on either side as a pattern. We initialized the mention and contexts embeddings input to either the ladder network or the mean teacher as well as the baseline systems with pre-trained embeddings from Levy and Goldberg (2014b) (size 300d) as this gave us improved results on the baseline compared to vanilla `word2vec` initialization. We used a 600d dimensional embedding for each datapoint (300 each from entity and context concatenated). We used a 3-layer ladder network with dimensions 600-500-K where K is the number of labels present in the dataset. Further, we used a noise of 0.3 for the corrupted encoder and reconstruction cost for the 3-layers were 1000-10-0.1. In the custom embeddings architecture of the mean teacher framework we used an bi-directional LSTM of size 200d on the entities and the each of the patterns in the context. The hidden size of fully connected layer was set to 300. We used a random word dropout of size 1 on the patterns to be input to the student and the teacher. Following Tarvainen and Valpola (2017), in the MT runs, we set the consistency cost weight to 1 and consistency-rampup to 5. We set the supervised examples (mentions along their corresponding contexts and labels) randomly. For CoNLL we used 40 and Ontonotes 440 examples, with equal representation from their labels' set.

Note that both MT and LN classify individual mentions, whereas Emboot and the two baselines classify entities. To facilitate the comparison between MT and LN and the baselines, for MT and LN we generate entity-level predictions by averaging the scores of the top predictions of all mentions corresponding to the same entity, and then select the label with the highest score for each entity. Further, to create the precision/throughput curves we we sorted the predictions returned by the LN and MT in decreasing order of their activation scores, and introduce multiple cutout thresholds to generate the precision/throughput points for the curves. We ran the baselines until they predicted labels for all the entities. For the baselines, in each iteration we promoted 100 entities per category.[8]

To compare with the baselines, which classify entities rather than mentions, we sorted the predictions returned by the LN and MT in decreasing order of their activation scores and chose the most confident entity label (when all its mention scores were averaged). For each of the LN and MT results, we report averaged results of 5 random runs (along with the standard deviation).

7.2 Results and Observations

Tables 1 and 2 shows the overall accuracies of the various systems on entity-based and mention-based evaluations, respectively. The key observation is that there is a consistent improvement in performance in approaches that are based on representation learning (namely, Emboot [9], LN and MT) on both datasets compared to traditional bootstrapping techniques (EPB and LP). Furthermore, we observe that one-shot learning systems such as Ladder and MT have a clear edge over the embedding based iterative

[6] `http://scikit-learn.org/stable/modules/generated/sklearn.semi_supervised.LabelPropagation.html`

[7] We experimented with other feature values, e.g., pattern PMI scores, but all performed worse than raw counts.

[8] We also ran a cautious approach of promoting 10 entities per category per iteration and noticed that the former had slightly better performance.

[9] Note: The Emboot results are for 20 epochs only. As running more number of epochs to cover the entire dataset does not complete even after a long time and sometimes results in memory overflow in the current implementation.

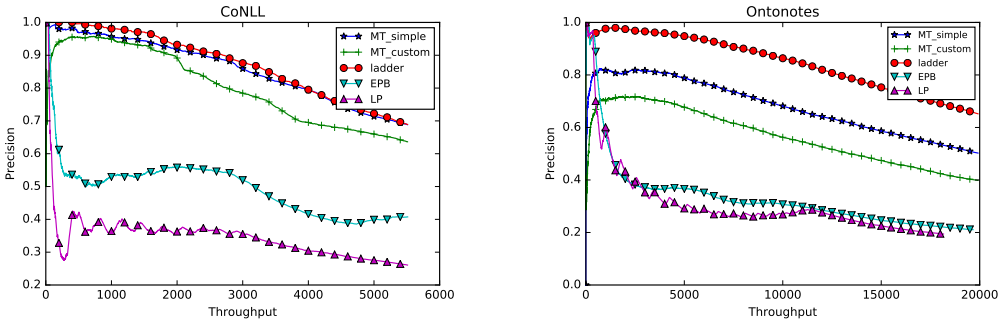


Figure 3: Overall results on the CoNLL (left) and Ontonotes (right) datasets. Throughput is the number of entities classified, and precision is the proportion of entities that were classified correctly.

bootstrapping (Emboot). To our knowledge, the results in Table 2 are the new state of the art for this semi-supervised NEC task.

While MT and LN outperform Emboot on both datasets, it is yet not fully clear why. As mentioned, MT and LN use one-shot learning whereas Emboot is iterative and, thus, more prone to semantic drift. However, a second potential cause for this difference is that MT and LN assemble their input representation from word-level embeddings, whereas Emboot learns from scratch embeddings for atomic entities and patterns, which forces it to work with sparser information. We leave this analysis, as well as efforts to align these three frameworks more closely, as future work.

Figure 3 shows the precision vs. throughput curves for the baselines, LN and MT approaches. We see that on both the datasets the one-shot learning methods outperform the iterative baselines by a large margin. Further we notice that both LN and MT variants are reasonably stable for most of the curve whereas EPB degrades quickly. Bootstrapping systems inherently suffer from semantic drift: as the iterations progress the system begins to drift into a different semantic space due to incomplete statistics and ambiguity (Komachi et al., 2008).

MT outperforms slightly LN on the CoNLL dataset, whereas LN performs better than MT on OntoNotes. We speculate that the latter result is caused by the fact that the space of hyper parameters were not sufficiently explored in the MT framework. For example, we ran LN for approximately 200 iterations, and we ran the MT experiments for 20 epochs. We will investigate a larger hyper parameter space in future work. However, the comparable performance of the two approaches is an encouraging result for MT, which is the simpler architecture of the two. With MT, we can "plug in" any task simply by generating a different input vector X (and, potentially, a different scheme to insert noise), whereas LN requires an adaptation of the entire architecture due to the tight connections between teacher and student.

Qualitative analysis: In Figure 4, we plot the t-SNE (van der Maaten and Hinton, 2008) projections of the learned custom embeddings from the MT framework. We contrast these projections with the t-SNE projections of the entities constructed by averaging the vectors present in the entity tokens, as provided by the pre-trained embeddings we used to initialize the various models. It is interesting to note that we can see clear clustering of the entities in both datasets for MT. The number of clusters is roughly equal to the number of classes present in the datasets. This is especially accentuated on the OntoNotes dataset.Upon closer observation on a small number of data points, we could ascertain that the clusters were semantically meaningful. On the other hand, we do not see such a clustering of the pre-trained embeddings (word2vec). This opens the gates for further investigation of these custom embeddings to construct interpretable deep learning models.

Amount of training data vs. accuracy: Table 3 lists the accuracies of LN and MT_simple approaches on all the data points, as we varied the amount of supervision. As expected, as we increase the amount of supervision, we observe improvements in accuracy. More importantly, the table shows both LN and MT outperform the overall accuracy of EPB (right-most points in Figure 3) with much fewer annotations (e.g., with 55 annotations in OntoNotes, LN and MT outperform the performance of EPB with 550

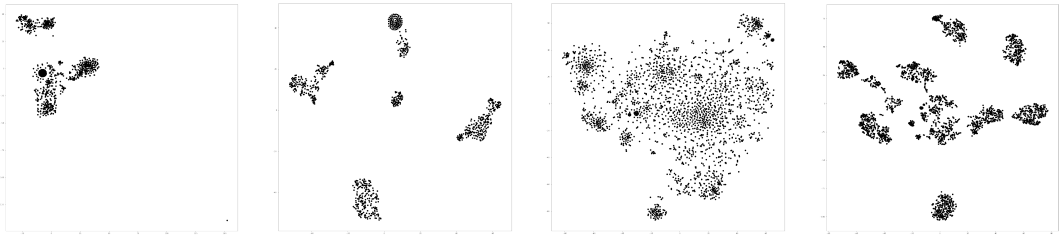


Figure 4: t-SNE projections of the learned embeddings of four models: (i) pre-trained word2vec on the CoNLL dataset; (ii) the custom MT model on CoNLL; (iii) pre-trained word2vec on OntoNotes; (iv) custom MT on OntoNotes.

CoNLL				*OntoNotes*			
Num. labels	% Train	*Ladder*	*MT*	*Num. labels*	% Train	*Ladder*	*MT*
20	0.152	46.46	72.74	55	0.082	26.04	57.05
40	0.303	66.46	80.53	110	0.164	48.53	58.40
80	0.606	75.37	80.96	220	0.328	59.66	58.87
160	1.212	81.11	82.06	440	0.656	73.10	61.72
320	2.424	80.94	82.58	880	1.313	73.58	61.96
640	4.848	82.51	82.60	1760	2.626	73.23	67.38
1280	9.696	81.22	83.31	3520	5.253	73.77	70.52
2560	19.393	81.34	83.47	7040	10.507	73.31	72.65
5120	38.787	81.26	84.88	14080	21.014	82.47	73.50
10240	77.575	81.91	86.30	28160	42.029	83.32	75.91

Table 3: Number of labeled examples used for training and corresponding accuracy on the two datasets.

annotated examples).

8 Conclusion

To our knowledge, this is the first work that rigorously analyzes several recent semi-supervised representation-learning approaches in the context of an information extraction task – named entity classification, in particular. To achieve this, we adapted two teacher-student methods that were initially proposed for image processing to the named entity classification task, and compared them against an iterative representation learning that learns custom embeddings for the underlying data, as well as other semi-supervised learning approaches that do not rely on representation learning. Our analysis, which was performed on two NEC datasets, highlighted several important observations, which, we believe, should inform the design of future semi-supervised IE systems. First, all representation learning methods investigated outperform state-of-the-art semi-supervised methods that do not rely on representation learning. Second, one-shot learning methods outperform iterative approaches, which suggests that one-shot learning is a practical solution to control for semantic drift. Third, the mean teacher framework performs comparatively similar to ladder networks, despite its simplicity and independence of the underlying task-specific model. These results advocate for a modular take on semi-supervised IE, where the learning is performed by a generic MT framework, which is then extended for specific tasks through independent components that model the corresponding data and task.

As part of future work, we would like to explore other tasks such as fine-grained entity typing, where the number of labels are an order of magnitude larger. Further, another interesting avenue for further research is to perform semi-supervised relation extraction, which is a significantly more challenging task and would be greatly benefited by such techniques.

Acknowledgements

Dr. Surdeanu declares a financial interest in lum.ai. This interest has been properly disclosed to the University of Arizona Institutional Review Committee and is managed in accordance with its conflict of interest policies. This work was funded by the DARPA Big Mechanism program under ARO contract W911NF-14-1-0395 and by the Bill and Melinda Gates Foundation HBGDki Initiative.

References

Steven Abney. 2007. *Semisupervised Learning for Computational Linguistics*. Chapman & Hall/CRC, 1st edition.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 92–100, New York, NY, USA. ACM.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010a. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*.

Andrew Carlson, Justin Betteridge, Richard C Wang, Estevam R Hruschka Jr, and Tom M Mitchell. 2010b. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 101–110. ACM.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Olivier Delalleau, Yoshua Bengio, and Nicolas Le Roux. 2005. Efficient non-parametric function induction in semi-supervised learning. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS'05)*, pages 96–103. Society for Artificial Intelligence and Statistics, January.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL*.

Sonal Gupta and Christopher D Manning. 2014. Improved pattern learning for bootstrapped entity extraction. In *CoNLL*, pages 98–108.

Sonal Gupta and Christopher D. Manning. 2015. Distributed representations of words to guide bootstrapped entity classifiers. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Mamoru Komachi, Taku Kudo, Masashi Shimbo, and Yuji Matsumoto. 2008. Graph-based analysis of semantic drift in espresso-like bootstrapping algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 1011–1020, Stroudsburg, PA, USA. Association for Computational Linguistics.

Samuli Laine and Timo Aila. 2016. Temporal ensembling for semi-supervised learning. *CoRR*, abs/1610.02242.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *ACL (2)*, pages 302–308.

Omer Levy and Yoav Goldberg. 2014b. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland, June. Association for Computational Linguistics.

Tara McIntosh and James R Curran. 2008. Weighted mutual exclusion bootstrapping for domain independent lexicon and template acquisition. In *Proceedings of the Australasian Language Technology Association Workshop*, volume 2008.

Tara McIntosh. 2010. Unsupervised discovery of negative categories in lexicon bootstrapping. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 356–365. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Ajay Nagesh and Mihai Surdeanu. 2018. Keep your bearings: Lightly-supervised information extraction with ladder networks that avoids semantic drift. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 352–358. Association for Computational Linguistics.

Arvind Neelakantan and Michael Collins. 2015. Learning dictionaries for named entity recognition using minimal supervision. *CoRR*, abs/1504.06650.

Mohammad Pezeshki, Linxi Fan, Philemon Brakel, Aaron Courville, and Yoshua Bengio. 2016. Deconstructing the ladder network architecture. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2368–2376, New York, New York, USA, 20–22 Jun. PMLR.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Bjrkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria, August. Association for Computational Linguistics.

Delip Rao and Deepak Ravichandran. 2009. Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 675–682, Stroudsburg, PA, USA. Association for Computational Linguistics.

Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. 2015. Semi-supervised learning with ladder networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3546–3554. Curran Associates, Inc.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of NAACL-HLT*.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the national conference on artificial intelligence*, pages 1044–1049.

Rebecca Sharp, Mihai Surdeanu, Peter Jansen, Peter Clark, and Michael Hammond. 2016. Creating causal embeddings for question answering with minimal supervision. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1195–1204. Curran Associates, Inc.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, volume 15, pages 1499–1509. Citeseer.

Kristina Toutanova, Xi Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge bases and text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1434–1444.

Marco A. Valenzuela-Escárcega, Ajay Nagesh, and Mihai Surdeanu. 2018. Lightly-supervised representation learning with global interpretability. *CoRR*, abs/1805.11545.

Harri Valpola. 2014. From neural PCA to deep unsupervised learning. *CoRR*, abs/1411.7783.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1096–1103, New York, NY, USA. ACM.

Roman Yangarber. 2003. Counter-training in discovery of semantic patterns. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Shuangfei Zhai and Zhongfei Zhang. 2015. Semisupervised autoencoder for sentiment analysis. *CoRR*, abs/1512.04466.

Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. 2017. Deconvolutional paragraph representation learning. *CoRR*, abs/1708.04729.

X. Zhu and Z. Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University.

Multimodal Grounding for Language Processing

Lisa Beinborn[°*]◇ **Teresa Botschen**[*]△ **Iryna Gurevych** △

◇ Language Technology Lab, University of Duisburg-Essen

△ Ubiquitous Knowledge Processing Lab (UKP) and Research Training Group AIPHES
Department of Computer Science, Technische Universität Darmstadt
www.ukp.tu-darmstadt.de

Abstract

This survey discusses how recent developments in multimodal processing facilitate conceptual grounding of language. We categorize the information flow in multimodal processing with respect to cognitive models of human information processing and analyze different methods for combining multimodal representations. Based on this methodological inventory, we discuss the benefit of multimodal grounding for a variety of language processing tasks and the challenges that arise. We particularly focus on multimodal grounding of verbs which play a crucial role for the compositional power of language.

Title and Abstract in German

Multimodale konzeptuelle Verankerung für die automatische Sprachverarbeitung

Dieser Überblick erörtert, wie aktuelle Entwicklungen in der automatischen Verarbeitung multimodaler Inhalte die konzeptuelle Verankerung sprachlicher Inhalte erleichtern können. Die automatischen Methoden zur Verarbeitung multimodaler Inhalte werden zunächst hinsichtlich der zugrundeliegenden kognitiven Modelle menschlicher Informationsverarbeitung kategorisiert. Daraus ergeben sich verschiedene Methoden um Repräsentationen unterschiedlicher Modalitäten miteinander zu kombinieren. Ausgehend von diesen methodischen Grundlagen wird diskutiert, wie verschiedene Forschungsprobleme in der automatischen Sprachverarbeitung von multimodaler Verankerung profitieren können und welche Herausforderungen sich dabei ergeben. Ein besonderer Schwerpunkt wird dabei auf die multimodale konzeptuelle Verankerung von Verben gelegt, da diese eine wichtige kompositorische Funktion erfüllen.

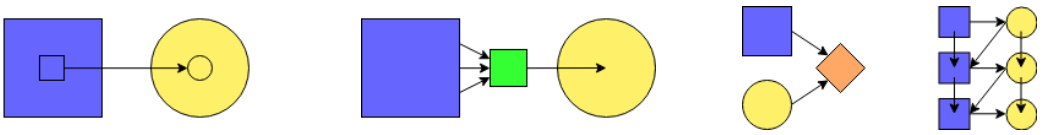
1 Introduction

Natural languages are continually developing constructs that include numerous variations and irregularities. Modeling the subtleties of language in a formal, processable way has driven computational linguistics for decades. In recent years, distributional approaches have become the most widely accepted solution to model the associative character of word meaning (Harris, 1954; Collobert et al., 2011; Mikolov et al., 2013; Pennington et al., 2014). These approaches learn word representations in a high-dimensional vector space based on context patterns in large text collections. Machine learning researchers aim at reducing external knowledge to an absolute minimum and simply interpret language as a continuous stream of characters. From an engineering perspective, these data-driven approaches are highly attractive because they reduce the need of domain experts.

From a cognitive perspective, processing language in isolation without information on situational context seems to be an overly artificial setup. Human acquisition of semantic representations does not occur based on pure language input. The term conceptual grounding refers to the idea that language is

[°] The work by Lisa Beinborn has been carried out during her affiliation with Ubiquitous Knowledge Processing Lab (UKP) and Research Training Group AIPHES, Technische Universität Darmstadt.
[*] The first and the second authors contributed equally to this work.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2325–2339
Santa Fe, New Mexico, USA, August 20-26, 2018.



(a) Cross-modal transfer. Information from modality A is aligned to comparable information in B.

(b) Cross-modal interpretation. Relevant information in modality A is summarized and interpreted in modality B.

(c) Joint multimodal processing. Left: Modality A and B both contribute to a joint prediction. Right: Interactive exchange of information between modalities.

Figure 1: Information flow in multimodal tasks. Blue and yellow shapes refer to modality A and B.

grounded in perceptual experience and sensorimotor interactions with the environment (Barsalou, 2008). In its strictest interpretation, this embodied perspective implies that language production and language comprehension involve perceptual and motor simulations of the described situation (Goldman, 2006). An impressive number of recent neuroimaging studies indicate that processing a word activates areas in the brain that correspond to the associated sensory modality of its semantic category: action-related words like *kick* trigger activity in the motor cortex and object-related words like *cup* activate visual areas (Pulvermüller et al., 2005; Garagnani and Pulvermüller, 2016). While it remains a controversial question to what extent conceptual representations are actually shared across modalities (Louwerse, 2011; Leshinskaya and Caramazza, 2016), it has been widely accepted that conceptual and sensorimotor representations are tightly coupled and interact with each other. Cognitively plausible language processing should thus interpret language as one modality within a multimodal environment.

This survey discusses how recent developments in multimodal processing facilitate conceptual grounding of language. It intents to provide a bridge between the field of multimodal machine learning (Baltrušaitis et al., 2017) and the cognitive theories for grounding distributional semantics (Baroni, 2016). As this is a wide interdisciplinary topic which influences many subfields, we focus on multimodal grounding for computational linguistics. For a better understanding of the interaction between modalities, we categorize multimodal tasks according to the information flow between the modalities. In a second step, we analyze different methods for combining multimodal information. Based on this methodological inventory, we discuss the benefit of multimodal grounding for a variety of language processing tasks. In multimodal processing, grounding is usually limited to concrete concepts leading to a reduction of referential ambiguity. We provide a detailed analysis of the challenges that arise when multimodal grounding is extended to open-domain language settings. We particularly focus on multimodal grounding of verbs which is essential for the interpretation of sequences and the identification of relations between concepts.

2 Multimodal processing models

The term "multimodal" has been used in a broad range of different interpretations even in the computational linguistics literature alone. In the common interpretation, modalities refer to sensory input such as audio, vision, touch, smell, and taste. Other definitions stretch over different communicative channels such as language and gesture, or simply different "modes" of the same modality (e.g., day and night pictures). In this section, we analyze the flow of multimodal information in different multimodal tasks exemplified by three modalities: natural language encoded as texts, visual signals encoded as images or videos, and audio signals encoded as sound files. For an overview of the challenges and machine learning methods associated with each task, the interested reader is referred to Baltrušaitis et al. (2017). We propose a classification of multimodal tasks with respect to the information flow between modalities into cross-modal transfer, cross-modal interpretation, and joint multimodal processing. From a historical perspective, progress in multimodal processing can be aligned with cognitive theories of multimodal organization in the human brain.

2.1 Cross-modal transfer

In the 1980s and 90s, cognitive processing theories were heavily influenced by the theory of the modularity of mind (Fodor, 1985). It assumes that processing occurs in domain-specific encapsulated modules

that do not interact with each other. Earlier approaches to multimodal engineering have taken a similar modular perspective. They model the information flow in each modality separately and the final outcome is then transferred or aligned to another modality. We group tasks in which one modality serves as the interface to query or represent the content from another modality under the category of cross-modal transfer, see Figure 1a.

A classical example for cross-modal transfer are search and retrieval tasks. The human user provides a natural language description to query an artifact (i.e., an image, video, or audio file) from a database (Atrey et al., 2010). The cross-modal alignment between the query and the artifact requires query expansion and disambiguation for referential indexing. In speech-related transfer tasks, textual content needs to be mapped to audio samples. Speech synthesis transforms text into artificially generated phonemes for users who cannot read (Zen et al., 2009). The reverse task of transcribing audio and video content is addressed by approaches for speech recognition (Juang and Rabiner, 2005) and subtitle generation (Daelemans et al., 2004). For lipreading tasks, mute video input of people speaking is transformed into text representing their utterances (Ngiam et al., 2011).

In these cross-modal transfer tasks, synchronous processing of the input in one modality is not directly influenced by information from the output modality. The main challenge lies in finding appropriate translations or alignments from one modality to the other. Information from the output modality is mainly used for reranking of input hypotheses. This view corresponds to mental models of a language hub in the brain that does not directly incorporate perceptual information (Chomsky, 1986).

2.2 Cross-modal interpretation

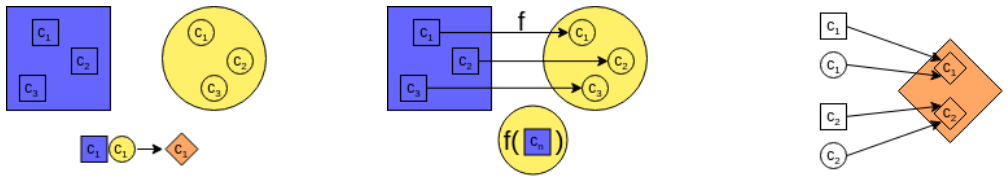
In order to explain how humans can select relevant information from perceptual input, the concept of attention has become very popular. Bridewell and Bello (2016) argue that attention serves as "a bottleneck for information flow in a cognitive system" that redirects mental resources. In multimodal processing, the concept of attention as a mediator between modalities is relevant for cross-modal interpretation. For these tasks, the goal is to obtain a compressed and structured intermediate representation of the input to generate a useful interpretation in the target modality. Attention mechanisms (Bahdanau et al., 2014) are used for the identification of relevant information, see Figure 1b.

A textual interpretation of a visually presented scene is generated in image captioning (Xu et al., 2015) and sketch recognition (Li et al., 2015). The goal is to identify relevant elements, group individual elements to semantic concepts, identify relations between concepts, and express these relations in natural language. The output sequence is generated while paying attention to different salient areas in the image. To our knowledge, a bidirectional information flow that includes cues from the language generation module in the image recognition process has not yet been implemented. However, semantic information could help to better direct the attention for image recognition, e.g., the generation of a verb like *eat* could constrain the visual recognition to edible objects as filler roles. Yatskar et al. (2016) propose the task of situation recognition to approximate this problem.

Complementary approaches attempt to generate visual representations to summarize documents and present the most relevant information in an intuitive way (Kucher and Kerren, 2015). The most popular approach are so-called word clouds which are frequency-based visualizations for topic modeling (Bateman et al., 2008). More recent approaches include semantic relations between words for a more conceptual-driven interpretation (Xu et al., 2016). Concept maps highlight structural relations between concepts in a graph-based visualization (Zubrinic et al., 2012). One key challenge for cross-modal interpretation tasks lies in the evaluation of the output because interpretations are by definition subjective and divergent solutions can be equally valid. Accumulations over various human ratings are currently considered to be better quality approximations than any automatic metrics (Vedantam et al., 2015).

2.3 Joint multimodal processing

Due to a wave of experimental findings that support the cognitive theory of embodied processing, the separating aspects between different modalities have become blurred (Pulvermüller et al., 2005). A similar development can be observed in multimodal machine learning. Tasks which explicitly require the combination of knowledge from different modalities gave rise to joint multimodal processing (Figure 1c). For



(a) Multimodal fusion. Concatenate known representations from modality A and B and apply dimensionality reduction.

(b) Mapping. Learn a mapping function f from modality A to B that can be applied on unknown concepts c_n.

(c) Joint learning. Optimize two objectives simultaneously: quality of unimodal representations and cross-modal alignment.

Figure 2: Methods for learning multimodal representations. Blue and yellow shapes indicate the representation space of modality A and B.

emotion recognition (Morency et al., 2011) or persuasiveness prediction (Santos et al., 2016), the actual content of an utterance and paraverbal cues (e.g., pitch, facial expression) need to be jointly evaluated. An ironic tone of voice might reverse the conceptual interpretation of the language content.

Recent work from the vision community goes one step further and tackles tasks that imperatively require an interactive flow of information. In visual question answering, a human user can ask questions about an image that the system should answer (Malinowski et al., 2015). This requires several steps: understanding the question, determining the salient elements in the image, interpreting the image with respect to the question, and generating a coherent natural language answer that matches the question. For this task, exchange of information between the modalities is crucial. In an overview, Wu et al. (2017) compare 29 approaches to visual question answering. 23 of these approaches use a joint representation of textual and visual information. The remaining 6 approaches organize the exchange either through a coordinated network architecture or through shared external knowledge bases. Novel interactive approaches make it possible to directly modulate the information flow in one modality by input from another modality (Vries et al., 2017) or by human feedback (Ling and Fidler, 2017).

The main challenge for joint processing lies in efficiently combining information from the modalities, so that redundant information is integrated without losing complementary cues. In human language understanding, this process seems to be performed in an effortless and highly accurate manner (Crocker et al., 2010). However, the underlying mechanisms of multimodal representations remain poorly understood. In Section 3, we discuss different methods for obtaining joint representations computationally.

3 Multimodal representation learning

Multimodal representations combine information from separate modalities. We discuss methods for representing known concepts, projecting information to represent unknown concepts, and for combining concept representations into compositional representations for sequences.

3.1 Concept representations

Even in unimodal tasks, researchers experiment with many different variations of representing concepts and their relations. Earlier work on multimodal representations used human-elicited visual features (Silberer and Lapata, 2012; Roller and Schulte Im Walde, 2013). Conveniently, integrating knowledge from different modalities has been facilitated due to the now common low-level representations of the input (also known as embeddings). Images are represented as groups of pixels, videos as series of image frames, audio data as windows of waveform samples, and language as sequences of distributional word representations. These values are then fed into a neural network that learns to compress and normalize the representation such that it better generalizes across input samples (Kiela and Bottou, 2014). A concept representation is usually obtained by averaging over many different samples for the concept (e.g., the concept *bird* is represented by averaging over the representation for n images showing a bird). The representations are expressed as high-dimensional matrices which can be projected into a common space. This approach facilitates a joint information flow between different modalities and has contributed to the growing success of multimodal processing.

Fusion The most intuitive approach is multimodal fusion (Figure 2a). Assuming that a unimodal vector representation v for the concept c and the modalities m_1 and m_2 exists, the multimodal representation v_{mm} consists of the concatenation $\frown$ of the two vectors weighted by a tunable parameter α:

$$v_{mm}(c) = \alpha \cdot v_{m_1}(c) \frown (1 - \alpha) \cdot v_{m_2}(c).$$

The concatenation occurs directly on the concept level and is thus called feature-level fusion or early fusion (Leong and Mihalcea, 2011; Bruni et al., 2011). In the case of pure concatenation, the unimodal representations reside in separate conceptual spaces. The concatenated representation for *cat* could give us the information that *cat* is visually similar to *panther* and textually similar to *dog*, but it is not possible to determine cross-modal similarity. In order to smooth the concatenated representations while maintaining multimodal correlations, dimensionality reduction techniques such as singular value decomposition (Bruni et al., 2014) or canonical correlation analysis (Silberer and Lapata, 2012) have been applied.

3.2 Projection

In practice, concepts that have a representation in one modality are not necessarily covered by representations in another modality. The projection of unseen concepts is known as zero shot learning. It can either be performed on a mapped or a joint representational space.

Mapping To overcome the lack of representations for one modality, several researchers proposed to map representations from one modality to the other (Figure 2b). The idea is to learn a mapping function f from m_1 to m_2 that maximizes the similarity between a known representation of c in m_2 and its projection from the representation in m_1: $c_{m_2} \sim f(c_{m_1})$.

The choice of the similarity and the loss measures for learning the mapping function vary. A max-margin optimization which maximizes the similarity between true pairs of concept representations (c_{m_1}, c_{m_2}) and minimizes the similarity for pairs with random target representations $(c_{m_1}, random_{m2})$ has been shown to be a good choice for image labeling (Frome et al., 2013). In this task, the mapping approach is applied in the image-to-text direction to classify unknown objects in images based on their semantic similarity to known objects (Socher et al., 2013) . Lazaridou et al. (2014) and Collell et al. (2017) proceed in the reverse text-to-image direction to ground words in the visual world. Similar propagation approaches had already been examined by Johns and Jones (2012) and Hill and Korhonen (2014), but they used human-elicited perceptual features from the McRae dataset (McRae et al., 2005) instead of automatically derived image representations.

Joint learning The mapping approaches assume a directed transformation from one modality to the other. Joint estimation approaches aim to learn shared representations instead (Figure 2b). An approach inspired by topic modeling interprets aligned data as a multimodal document and uses Latent Dirichlet Allocation to derive multimodal topics (Andrews et al., 2009; Feng and Lapata, 2010; Silberer and Lapata, 2012; Roller and Schulte Im Walde, 2013). Unfortunately, this approach cannot be easily used for zero shot learning. Lazaridou et al. (2015) enrich the skip-gram model by Mikolov et al. (2013) with visual features. Their model optimizes two constraints: the representation of concepts c with respect to their textual contexts (unsupervised skip-gram objective in m_1) and the similarity between word representations and their visual counterparts (supervised max-margin objective for (c_{m_1}, c_{m_2})). In their approach, the visual representations remain fixed, but the textual representations are learned from scratch. Silberer and Lapata (2014) go one step further and use stacked multimodal autoencoders to simultaneously learn good representations for each modality (unsupervised reconstruction objective for m_1 and m_2) and their optimal multimodal combination (supervised classification objective for (c_{m_1}, c_{m_2})). Both approaches implicitly also learn a mapping between the two modalities and can be adjusted to induce a directional projection for zero shot learning. Joint learning of multimodal representations is very popular in the vision community (Karpathy et al., 2014; Srivastava and Salakhutdinov, 2012; Ngiam et al., 2011).

3.3 Compositional representations

For tasks that require representing longer sequences, a naïve approach is sequence-level fusion. In this setting, the unimodal sequence representation is obtained by performing an arithmetic operation (e.g.,

average, max) over the concept representations for each word in the sequence. Multimodal fusion is then performed on this averaged representation (Glavaš et al., 2017; Bruni et al., 2014). Shutova et al. (2016) work with short phrases consisting of two words and directly learn phrase representations. Missing concept representations in one modality can be obtained by mapping functions (Botschen et al., 2018).

For image captioning approaches, representations for a pair of an image and the corresponding caption are learned jointly (Kiros et al., 2014; Socher et al., 2014). Pre-trained unimodal representations are fed into a neural network which is trained with the max-margin objective to distinguish between true and false captions for an image. The multimodal sequence representation can be obtained from the last hidden layer of the network. The introduction of attention variables can function as a mediator between the visual and the textual modality (see Section 2.2). For a more detailed overview of multimodal sequence representations in the vision community, the interested reader is referred to Wu et al. (2017). The approaches for compositional representations have focused on enriching noun and adjective meaning multimodally. The multimodal interpretation of verbs as an integral part of compositional sequences has not yet been thoroughly examined.

4 Multimodal grounding for language processing

The progress in joint multimodal processing and the increasing availability of multimodal datasets and representations open up new possibilities for grounded approaches to language processing. We review recent works for grounding concepts, grounding phrases, and grounding interaction. The challenges that arise from these efforts are discussed in Section 5.

4.1 Grounding concepts

Multimodal concept representations are motivated by the idea that semantic relations between words are grounded in perception. Being able to assess semantic relations between concepts is an important prerequisite for modeling generalization capabilities in language processing. The combination of the textual and the visual modality has received most attention for conceptual grounding, but perceptual information from the auditory and the olfactory channel have also been used for dedicated tasks (Kiela et al., 2015; Kiela and Clark, 2017). In order to provide a more concrete discussion, we focus on the combination of textual and visual cues for the remainder of the survey.

The quality of concept representations is commonly evaluated by their ability to model semantic properties. Different approaches to learning conceptual models are compared by their performance on similarity datasets, e.g., *WordSim353* (Finkelstein et al., 2002), *SimLex-999* (Hill et al., 2015), *MEN* (Bruni et al., 2012), *SemSim*, and *VisSim* (Silberer and Lapata, 2014)). These datasets contain pairs of words that have been annotated with similarity scores for the two concepts. Several evaluations of semantic models have shown that multimodal concept representations outperform unimodal ones (Feng and Lapata, 2010; Silberer and Lapata, 2012; Bruni et al., 2014; Kiela et al., 2014). Kiela et al. (2016) perform a comparison of different image sources and architectures and their ability to model semantic similarity. Despite the advantages of multimodal models in capturing semantic relations, it remains an open question whether they contribute to a cognitively more plausible approximation of human conceptual grounding. Bulat et al. (2017b) and Anderson et al. (2017) conduct experiments to label brain activity scans by human subjects with the corresponding concepts that elicited the brain activity. They compare different distributional semantic models and obtain mixed results. Bulat et al. (2017b) find that visual information is beneficial for modeling concrete words, whereas Anderson et al. (2017) conclude that textual models sufficiently integrate visual properties. Further interdisciplinary research involving computer science, neuroscience, and psycholinguistics is required to obtain a deeper understanding of cognitively plausible language processing (Embick and Poeppel, 2015).

4.2 Grounding phrases

Most experiments for conceptual grounding indicate that providing a multimodal representation for abstract concepts is significantly more challenging due to the lack of perceptual patterns associated with abstract words (Hill et al., 2014). For grounding phrases, the meaning for concrete and abstract concepts

need to be combined (see Section 3.3). Bruni et al. (2012) examine the compositional meaning of color adjectives and find that multimodal representations are superior in modeling color. However, they fail to distinguish between literal and non-literal usage of color adjectives (e.g., *green cup* vs *green future*).

Vivid imagery and synaesthetic associations play an important role in the interpretation of figurative language. In their influential theory of metaphor, Lakoff and Johnson (1980) argue that abstract concepts can be grounded metaphorically in embodied and situated knowledge. They assume that metaphors can be understood as a mapping from a concrete source domain to a more abstract target domain. For example, *time* is often viewed as a *stream* that *flows* in a direction. Turney et al. (2011) operationalize this theoretical account by identifying metaphoric phrases based on the discrepancy in concreteness of source and target term. Shutova et al. (2016) and Bulat et al. (2017a) build on their approach and use multimodal models for identifying metaphoric word usage in adjective-noun combinations. They show that words used in a metaphorical combination (*dry wit*) exhibit less similarity than words in non-metaphorical phrases (*dry skin*). We strongly believe that progress in multimodal compositional grounding will pave the way for a more holistic understanding of figurative language processing. As a prerequisite, multimodal grounding needs to be examined beyond the representation of concrete objects (see Section 5.2). Representing verbs, compositional phrases, and even full sentences by means of multimodal information has not yet been sufficiently examined.

4.3 Grounding interaction

The origins of grounding theories were initiated to account for situational language use and interaction. We distinguish two main scenarios for interactive language use: language learning and situational grounding of action descriptions.

Grounded language learning Language learning is deeply rooted in social interaction and initially emerges with respect to a concrete referential context (Tomasello, 2010). Children acquire language in interaction with their parents and foreign language learning proceeds much faster in an environment that forces the learner to interact in the foreign language (Nation, 1990). Usage-based approaches to language learning that account for the frequency and the quality of the language stimulus have a long tradition (Dale and Chall, 1948). Brysbaert and New (2009) have shown that frequency information grounded in auditory and visual communicative cues can better model human processing effects than frequency information extracted from purely textual corpora. Lazaridou et al. (2016) show that a multimodal distributional approach better approximates word learning from interactive child-directed input than unimodal approaches. The same model can also convincingly simulate word meaning induction by adults (Lazaridou et al., 2017a). Psycholinguistic research indicates that conceptual mapping modulated by visual properties is not only relevant for first language acquisition, but is also used as a means to establish cross-lingual links in foreign language learning (Beinborn et al., 2014). Bergsma and Van Durme (2011) and Vulić et al. (2016) take advantage of this observation and use multimodal representations to induce multilingual representations.

Grounding sequences in actions Situational grounding of action descriptions requires the representation of sequences and their compositional interpretation. Regneri et al. (2013) build a corpus that grounds descriptions of actions in videos showing these actions. For the interpretation of sequences, evaluating verbs and their arguments plays a fundamental role. Yatskar et al. (2016) developed the *imSitu* dataset which consists of images depicting verbs and annotations which link the verb arguments to visual referents. This dataset can be used for the multimodal task of situation recognition (Mallya and Lazebnik, 2017; Zellers and Choi, 2017), and it serves as a multimodal resource for verb processing. Grounding verbs is particularly challenging because of the variety of their possible visual instantiations. For example, an image of an adult drinking beer has very little in common with a zebra drinking water.

Multimodal interpretation of sequences is highly relevant for robotics research (Chaplot et al., 2017). Mordatch and Abbeel (2017) examine the emergence of compositionality in grounded multi-agent communication. The language learned by artificial agents is not necessarily interpretable by humans. Lazaridou et al. (2017b) show that agents which develop their own language for representing concepts that are grounded in images infer similar taxonomic relations as humans. Their work suggests that the learned

concepts can even be mapped back into natural language. Agent-agent communication has already been examined in the talking head experiments, in which two agents learn to discriminate between objects and develop their own language of referring expressions (Steels and Vogt, 1997; Steels, 2002). Hermann et al. (2017) and Heinrich and Wermter (2018) explicitly focus on human-robot interaction and train their agent to associate natural language descriptions of actions with perceptual input from its sensors.

For experiments on grounded language understanding, the situational environment is usually artificially restricted to a very small domain. This confined setting facilitates the analysis of compositional expressions and their referential interpretation as complex object descriptions or action sequences. In open-domain language understanding, semantic disambiguation is even more challenging. Approaches using multimodal information for the disambiguation of concepts (Xie et al., 2017), named entities (Moon et al., 2018), and sentences (Botschen et al., 2018; Shutova et al., 2017) show promising tendencies, but the underlying compositional principles are not yet understood.

5 Challenges for grounded language processing

Multimodal grounding of language has been a longstanding goal of language researchers. The discussion has gained new momentum due to the recent developments in learning distributed multimodal representations. Most evaluations indicate that multimodal representations are beneficial for a variety of tasks, but explanatory analyses of this effect are still in a developing phase. In this section, we discuss open challenges that arise from existing work. For future work, we propose to examine multimodal grounding beyond concrete nouns and adjectives. In order to do this, larger multimodal datasets encompassing a wider range of word classes need to be build. These datasets would enable us to analyze compositional representations in more detail and to develop more elaborate models of selective multimodal grounding.

5.1 Combining complementary information

Different modalities contribute qualitatively different conceptual information. Bruni et al. (2014) argue that highly relevant visual properties are often not represented by linguistic models because they are too obvious to be explicitly mentioned in text (e.g., birds have wings, violins are brown). Textual models, on the other hand, provide a better intuition of taxonomic and functional relations between concepts which cannot easily be derived from images (Collell and Moens, 2016). Ideally, multimodal representations should integrate the complementary perspectives for a more coherent grounded interpretation of language. From a more skeptical perspective, Louwerse (2011) states that perceptual information is already sufficiently encoded in textual cues. In this case, the superior performance of multimodal representations that has been established by several researchers would mainly be due to a more robust representation of highly redundant information. The results by Silberer and Lapata (2014) and Hill et al. (2014) support the intuitive assumption that textual representations better model textual similarity and visual representations better model visual similarity. As the multimodal models improve on both similarity tasks, the integration of complementary information seems to be successful. Interestingly, both evaluations show that simply concatenating the two modalities already yields a quite competitive model. The reported findings have been evaluated on models working with human-annotated perceptual features. These features inherently represent taxonomic knowledge that cannot be directly inferred from visual input. It remains an open question to which extent automatically derived image representations can contribute complementary information when combined with textual representations. Most multimodal research to date focuses on the representation of individual concepts (nouns) and their properties (adjectives). The benefit of multimodal representations for language tasks going beyond concept similarity needs to be examined in more detail from both, engineering and theoretical perspectives.

Multimodal grounding of verbs Verbs play a fundamental role for expressing relations between concepts and their situational functionality (Hartshorne et al., 2014). The dynamic nature of verbs poses a challenge for multimodal grounding. To our best knowledge, only Hill et al. (2014) and Collell et al. (2017) consider verbs in their evaluation. They report that results for verbs are significantly worse, but

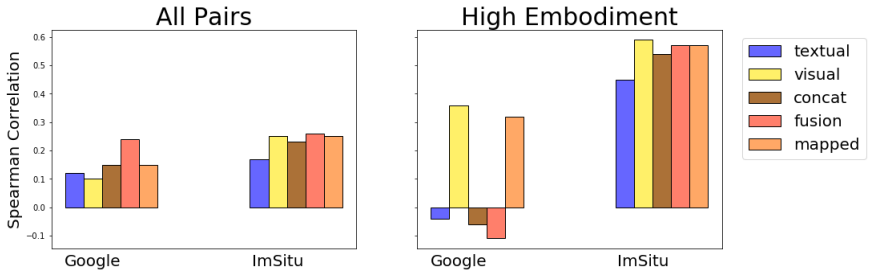


Figure 3: Illustration for the quality of verb representations indicated as Spearman correlation between the cosine similarity of verbs and their corresponding similarity rating in the *SimVerb* dataset.

do not elaborate on this finding. We present first steps towards an investigation of verb grounding.[1] Figure 3 illustrate the quality of verb representations in the most common publicly available approaches for multimodal representations. In line with previous work, the quality of the representations is evaluated as the Spearman correlation between the cosine similarity of two verbs and their corresponding similarity rating in the *SimVerb* dataset (Gerz et al., 2016). We compare the quality of 3498 verb pairs[2] in textual Glove representations (Pennington et al., 2014) and two visual datasets: the Google dataset that performed best in Kiela et al. (2016) and has the highest coverage for the verb pairs (493 pairs, 14%)[3] and the *imSitu* dataset which has been intentionally designed for verb identification (354 pairs, 10%). The results show that models which include visual information outperform purely textual representations for known concepts. However, the general quality of the verb representations is much lower than the quality reported for nouns. As a consequence, the mapping to unseen verb pairs yields unsatisfactory results for the full *SimVerb* dataset. Our encouraging results for the *imSitu* dataset indicate that it is recommended to directly obtain visual representations for verbs instead of projecting the meaning. Building larger multimodal datasets with a focus on verbs seems to be a promising strand of research for future work.

5.2 Imageability of abstract words

Conceptual grounding of language can be intuitively performed for concrete words that have direct referents in sensory experience. Bruni et al. (2014) and Hill and Korhonen (2014) show that multimodal representations are beneficial for evaluating concrete words, but have little to no impact on the evaluation of abstract words. Projecting unseen concepts into the representation space based on their relations to seen concepts in another modality provides an elegant method for zero shot learning, but it is questionable whether multimodal relations between concrete concepts are sufficient to infer relations between abstract concepts. Lazaridou et al. (2015) analyze projected abstract words by extracting the nearest visual neighbor from their multimodal representation. The neighbors were paired with random images and human raters judged how well each image represents the word. The hypothesis that concrete objects are more likely to be captured adequately by multimodal representations was confirmed. However, they also provide illustrating examples which represent abstract words like *together* or *theory* surprisingly well.

Embodiment of verbs From a multimodal perspective, verbs can be categorized according to their degree of embodiment. This measure indicates to which extent verb meanings involve bodily experience (Sidhu et al., 2014). We obtain embodiment ratings for 1163 pairs.[4] The class *high embodiment* contains pairs like *fall-dive* in which the embodiment of both verbs can be found in the highest quartile (135 pairs), *low embodiment* contains pairs with embodiment ratings in the lowest quartile (81 pairs) like *know-decide*.[5] Coherent with previous work on concrete and abstract nouns (Hill et al., 2014), it can be

[1]The pre-trained embeddings and the script to reproduce our results are available for research purposes: https://github.com/UKPLab/coling18-multimodalSurvey.

[2]Two pairs had to be excluded because *misspend* was not covered in the textual representations.

[3]The coverage in *WN9-IMG* (Xie et al., 2017) and the dataset used by Collell et al. (2017) is lower.

[4]`https://psychology.ucalgary.ca/languageprocessing/node/22`. We only include a pair, if an embodiment rating is available for both verbs.

[5]It should be noted that not all instances of the two classes are covered by the visual representations. The small number of instances might have an impact on the correlation values.

seen that visual representations better capture the similarity of verbs with a high level of embodiment. The mapped representations maintain this sense of embodiment, whereas the concatenated and fused representations better capture the similarity for verbs referring to more conceptual actions. This finding indicates that multimodal information is not equally beneficial for all words.

5.3 Selective multimodal grounding

The expressive power of language is essentially due to its combinatorial capabilities. Understanding how to combine concept representations to represent multi-word expressions or even full sentences has been a question of ongoing research in computational linguistics for decades. The inclusion of additional modalities further complicates this debate. Glavaš et al. (2017) and Botschen et al. (2018) obtain multimodal sentence representations by averaging over the multimodal representations for each word. They report improved results for the tasks of sentence similarity and frame identification. Our comparison above indicates that this superior performance is mainly due to a better representation of concepts. This raises the assumption that multimodal grounding should only be performed on selected words. Glavaš et al. (2017) propose to condition the inclusion of visual information on the prototypicality of a concept as measured by the image dispersion score (Kiela et al., 2014). This measure calculates the average pairwise cosine distance in a set of images to model the assumption that an image collection for an abstract concept like *happiness* is more diverse than for a concrete concept like *ladder*. Lazaridou et al. (2015) and Hessel et al. (2018) propose alternative concreteness measures based on the same idea. Unfortunately, these measures are highly dependent on the image retrieval algorithm which might be optimized towards obtaining a diverse range of images. Nevertheless, we assume that selective multimodal grounding constitutes a more plausible approach to sentence processing. Some functional words (e.g., locative expressions) might benefit from multimodal information, but it currently remains unclear how words with syntactic functions (e.g., coordinating expressions) should be represented visually.

6 Conclusion

We analyzed how multimodal processing has developed from transfer between encapsulated modalities to interactive processing over joint multimodal representations. These developments contribute to new avenues of research for grounded language processing. We strongly believe that the integration of multimodal information will improve our understanding of conceptual semantic models, figurative language processing, language learning, and situated interaction. Image datasets are often optimized towards providing a variety of visual instantiations. Developing algorithms for determining more prototypical visual representations could contribute to better grounding of verbs and might also serve as a criterion for selective multimodal grounding.

Acknowledgements

This work has been supported by the DFG-funded research training group "Adaptive Preparation of Information form Heterogeneous Sources" (AIPHES, GRK 1994/1) at Technische Universität Darmstadt. We thank Faraz Saeedan for his assistance with the computation of the visual embeddings for the imSitu images. We thank the anonymous reviewers for their insightful comments.

References

Andrew J Anderson, Douwe Kiela, Stephen Clark, and Massimo Poesio. 2017. Visually grounded and textual semantic models differentially decode brain activity associated with concrete and abstract nouns. *Transactions of the Association for Computational Linguistics (TACL)*, 5(1):17–30.

Mark Andrews, Gabriella Vigliocco, and David Vinson. 2009. Integrating experiential and distributional data to learn semantic representations. *Psychological review*, 116(3):463–498.

Pradeep K Atrey, M Anwar Hossain, Abdulmotaleb El Saddik, and Mohan S Kankanhalli. 2010. Multimodal fusion for multimedia analysis: a survey. *Multimedia systems*, 16(6):345–379.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. 2017. Multimodal machine learning: A survey and taxonomy. *arXiv preprint arXiv:1705.09406*.

Marco Baroni. 2016. Grounding distributional semantics in the visual world. *Language and Linguistics Compass*, 10(1):3–13.

Lawrence W Barsalou. 2008. Grounded cognition. *Annual Review of Psychology*, 59:617–645.

Scott Bateman, Carl Gutwin, and Miguel Nacenta. 2008. Seeing things in the clouds: the effect of visual features on tag cloud selections. In *Proceedings of the nineteenth ACM conference on Hypertext and hypermedia*, pages 193–202. ACM.

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2014. Readability for foreign language learning: The importance of cognates. *International Journal of Applied Linguistics*, 165(2):136–162.

Shane Bergsma and Benjamin Van Durme. 2011. Learning bilingual lexicons using the visual similarity of labeled web images. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, volume 22, pages 1764–1769.

Teresa Botschen, Iryna Gurevych, Jan-Christoph Klie, Hatem Mousselly Sergieh, and Stefan Roth. 2018. Multimodal frame identification with multilingual evaluation. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 1481–1491. Association for Computational Linguistics.

Will Bridewell and Paul F Bello. 2016. A theory of attention for cognitive systems. *Advances in Cognitive Systems*, 4:1–16.

Elia Bruni, Giang Binh Tran, and Marco Baroni. 2011. Distributional semantics from text and images. In *Proceedings of the GEMS workshop on geometrical models of natural language semantics*, pages 22–32. Association for Computational Linguistics.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Conference of the Association for Computational Linguistics (ACL)*, pages 136–145. Association for Computational Linguistics.

Elia Bruni, Nam-Khanh Tram, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

Marc Brysbaert and Boris New. 2009. Moving beyond Kučera and Francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for American English. *Behavior research methods*, 41(4):977–990.

Luana Bulat, Stephen Clark, and Ekaterina Shutova. 2017a. Modelling metaphor with attribute-based semantics. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 523–528. Association for Computational Linguistics.

Luana Bulat, Stephen Clark, and Ekaterina Shutova. 2017b. Speaking, seeing, understanding: Correlating semantic models with conceptual representation in the brain. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1081–1091.

Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. 2017. Gated-attention architectures for task-oriented language grounding. *arXiv preprint arXiv:1706.07230*.

Noam Chomsky. 1986. *Knowledge of language: Its nature, origin, and use*. Greenwood Publishing Group.

Guillem Collell and Marie-Francine Moens. 2016. Is an image worth more than a thousand words? On the fine-grain semantic differences between visual and linguistic representations. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 2807–2817.

Guillem Collell, Ted Zhang, and Marie-Francine Moens. 2017. Imagined visual representations as multimodal embeddings. In *Proceedings of the Thirty-First Conference on Artificial Intelligence (AAAI)*, pages 4378–4384.

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

Matthew W Crocker, Pia Knoeferle, and Marshall R Mayberry. 2010. Situated sentence processing: The coordinated interplay account and a neurobehavioral model. *Brain and language*, 112(3):189–201.

Walter Daelemans, Anja Höthker, and Erik F Tjong Kim Sang. 2004. Automatic sentence simplification for subtitling in dutch and english. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)*, pages 1045–1048.

Edgar Dale and Jeanne S Chall. 1948. A formula for predicting readability: Instructions. *Educational Research Bulletin*, 27(2):37–54.

David Embick and David Poeppel. 2015. Towards a computational(ist) neurobiology of language: correlational, integrated and explanatory neurolinguistics. *Language, Cognition and Neuroscience*, 30(4):357–366.

Yansong Feng and Mirella Lapata. 2010. Visual information in semantic representation. In *Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 91–99, Stroudsburg, USA. Association for Computational Linguistics.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Jerry A Fodor. 1985. Precis of the modularity of mind. *Behavioral and brain sciences*, 8(1):1–5.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. 2013. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2121–2129.

Max Garagnani and Friedemann Pulvermüller. 2016. Conceptual grounding of language in action and perception: a neurocomputational model of the emergence of category specificity and semantic hubs. *European Journal of Neuroscience*, 43(6):721–737.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. SimVerb-3500: A Large-Scale Evaluation Set of Verb Similarity. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2173–2182.

Goran Glavaš, Ivan Vulić, and Simone Paolo Ponzetto. 2017. If sentences could see: Investigating visual information for semantic textual similarity. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS)*, Montpeiller, France.

Alvin I Goldman. 2006. *Simulating minds: The philosophy, psychology, and neuroscience of mindreading*. Oxford University Press.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Joshua K Hartshorne, Claire Bonial, and Martha Palmer. 2014. The verbcorner project: Findings from phase 1 of crowd-sourcing a semantic decomposition of verbs. In *Proceedings of the 52nd Annual Conference of the Association for Computational Linguistics (ACL)*, pages 397–402. Association for Computational Linguistics.

Stefan Heinrich and Stefan Wermter. 2018. Interactive natural language acquisition in a multi-modal recurrent neural architecture. *Connection Science*, 30(1):99–133.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojtek Czarnecki, Max Jaderberg, Denis Teplyashin, et al. 2017. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*.

Jack Hessel, David Mimno, and Lillian Lee. 2018. Quantifying the visual concreteness of words and topics in multimodal datasets. *arXiv preprint arXiv:1804.06786*.

Felix Hill and Anna Korhonen. 2014. Learning abstract concept embeddings from multi-modal data: Since you probably can't see what i mean. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 255–265.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Multi-modal models for concrete and abstract concept meaning. *Transactions of the Association for Computational Linguistics (TACL)*, 2:285–296.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Brendan T Johns and Michael N Jones. 2012. Perceptual inference through global lexical similarity. *Topics in Cognitive Science*, 4(1):103–120.

Biing-Hwang Juang and Lawrence R Rabiner. 2005. Automatic speech recognition – a brief history of the technology development. *Georgia Institute of Technology. Atlanta Rutgers University and the University of California. Santa Barbara*, 1:67.

Andrej Karpathy, Armand Joulin, and Li F Fei-Fei. 2014. Deep fragment embeddings for bidirectional image sentence mapping. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1889–1897.

Douwe Kiela and Léon Bottou. 2014. Learning Image Embeddings using Convolutional Neural Networks for Improved Multi-Modal Semantics. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Douwe Kiela and Stephen Clark. 2017. Learning neural audio embeddings for grounding semantics in auditory perception. *Journal of Artificial Intelligence Research*, 60:1003–1030.

Douwe Kiela, Felix Hill, Anna Korhonen, and Stephen Clark. 2014. Improving multi-modal representations using image dispersion: Why less is sometimes more. In *Proceedings of the 52nd Annual Conference of the Association for Computational Linguistics (ACL)*, volume 2, pages 835–841. Association for Computational Linguistics.

Douwe Kiela, Luana Bulat, and Stephen Clark. 2015. Grounding semantics in olfactory perception. In *Proceedings of the 53nd Annual Conference of the Association for Computational Linguistics (ACL)*, pages 231–236, Beijing, China, July. Association for Computational Linguistics.

Douwe Kiela, Anita Verő, and Stephen Christopher Clark. 2016. Comparing Data Sources and Architectures for Deep Visual Representation Learning in Semantics. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 447–456.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*.

Kostiantyn Kucher and Andreas Kerren. 2015. Text visualization techniques: Taxonomy, visual survey, and community insights. In *Pacific Visualization Symposium (PacificVis)*, pages 117–121. IEEE.

George Lakoff and Mark Johnson. 1980. *Metaphors We Live By*. University of Chicago press.

Angeliki Lazaridou, Elia Bruni, and Marco Baroni. 2014. Is this a wampimuk? cross-modal mapping between distributional semantics and the visual world. In *Proceedings of the 52nd Annual Conference of the Association for Computational Linguistics (ACL)*, volume 1, pages 1403–1414. Association for Computational Linguistics.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015. Combining language and vision with a multimodal skip-gram model. In *Proceedings of the 14th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 153–163. Association for Computational Linguistics.

Angeliki Lazaridou, Grzegorz Chrupała, Raquel Fernández, and Marco Baroni. 2016. Multimodal semantic learning from child-directed input. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 387–392. Association for Computational Linguistics.

Angeliki Lazaridou, Marco Marelli, and Marco Baroni. 2017a. Multimodal word meaning induction from minimal exposure to natural text. *Cognitive science*, 41(S4):677–705.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. 2017b. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*.

Chee Wee Leong and Rada Mihalcea. 2011. Going beyond text: A hybrid image-text approach for measuring word relatedness. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 1403–1407.

Anna Leshinskaya and Alfonso Caramazza. 2016. For a cognitive neuroscience of concepts: Moving beyond the grounding issue. *Psychonomic bulletin & review*, 23(4):991–1001.

Yi Li, Timothy M. Hospedales, Yi-Zhe Song, and Shaogang Gong. 2015. Free-hand sketch recognition by multikernel feature learning. *Computer Vision and Image Understanding*, 137:1–11.

Huan Ling and Sanja Fidler. 2017. Teaching machines to describe images via natural language feedback. In *Advances in Neural Information Processing Systems (NIPS)*.

Max M. Louwerse. 2011. Symbol interdependency in symbolic and embodied cognition. *Topics in Cognitive Science*, 3(2):273–302.

Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. 2015. Ask your neurons: A neural-based approach to answering questions about images. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December.

Arun Mallya and Svetlana Lazebnik. 2017. Recurrent models for situation recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 455–463.

Ken McRae, George S Cree, Mark S Seidenberg, and Chris McNorgan. 2005. Semantic feature production norms for a large set of living and nonliving things. *Behavior research methods*, 37(4):547–559.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *International Conference on Learning Representations Workshop (ICLR)*.

Seungwhan Moon, Leonardo Neves, and Vitor Carvalho. 2018. Multimodal named entity recognition for short social media posts. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 852–860. Association for Computational Linguistics.

Igor Mordatch and Pieter Abbeel. 2017. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*.

Louis-Philippe Morency, Rada Mihalcea, and Payal Doshi. 2011. Towards multimodal sentiment analysis: Harvesting opinions from the web. In *Proceedings of the 13th international conference on multimodal interfaces*, pages 169–176. ACM.

Ian Stephen Paul Nation. 1990. *Teaching and Learning Vocabulary*. Teaching Methods. Heinle & Heinle.

Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. 2011. Multimodal deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 689–696.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Friedemann Pulvermüller, Olaf Hauk, Vadim V. Nikulin, and Risto J. Ilmoniemi. 2005. Functional links between motor and language systems. *European Journal of Neuroscience*, 21(3):793–797.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *Transactions of the Association of Computational Linguistics (TACL)*, 1:25–36.

Stephen Roller and Sabine Schulte Im Walde. 2013. A multimodal lda model integrating textual, cognitive and visual modalities. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1146–1157.

Pedro Bispo Santos, Lisa Beinborn, and Iryna Gurevych. 2016. A domain-agnostic approach for opinion prediction on speech. In Malvina Nissim, Viviana Patti, and Barbara Plank, editors, *Proceedings of the Workshop on Computational Modeling of Peoples Opinions, Personality, and Emotions in Social Media (PEOPLES)*, pages 163–172.

Ekaterina Shutova, Douwe Kiela, and Jean Maillard. 2016. Black holes and white rabbits: Metaphor identification with visual features. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 160–170. Association for Computational Linguistics.

Ekaterina Shutova, Andreas Wundsam, and Helen Yannakoudakis. 2017. Semantic frames and visual scenes: Learning semantic role inventories from image and video descriptions. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 149–154.

David M Sidhu, Rachel Kwan, Penny M Pexman, and Paul D Siakaluk. 2014. Effects of relative embodiment in lexical and semantic processing of verbs. *Acta psychologica*, 149:32–39.

Carina Silberer and Mirella Lapata. 2012. Grounded models of semantic representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1423–1433. Association for Computational Linguistics.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *Proceedings of the 52nd Annual Conference of the Association for Computational Linguistics (ACL)*, volume 1, pages 721–732. Association for Computational Linguistics.

Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. 2013. Zero-shot learning through cross-modal transfer. In *Advances in Neural Information Processing Systems (NIPS)*, pages 935–943.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics (TACL)*, 2(1):207–218.

Nitish Srivastava and Ruslan Salakhutdinov. 2012. Learning representations for multimodal data with deep belief nets. In *International conference on machine learning workshop*, volume 79.

Luc Steels and Paul Vogt. 1997. Grounding adaptive language games in robotic agents. In *Proceedings of the 4th european conference on artificial life*, volume 97.

Luc Steels. 2002. Grounding symbols through evolutionary language games. In *Simulating the evolution of language*, pages 211–226. Springer.

Michael Tomasello. 2010. *Origins of human communication*. MIT press.

Peter D Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 680–690. Association for Computational Linguistics.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4566–4575.

Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C. Courville. 2017. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6597–6607.

Ivan Vulić, Douwe Kiela, Stephen Clark, and Marie-Francine Moens. 2016. Multi-modal representations for improved bilingual lexicon learning. In *Proceedings of the 54th Annual Conference of the Association for Computational Linguistics (ACL)*, pages 188–194. Association for Computational Linguistics.

Qi Wu, Damien Teney, Peng Wang, Chunhua Shen, Anthony Dick, and Anton van den Hengel. 2017. Visual question answering: A survey of methods and datasets. *Computer Vision and Image Understanding*, 163:21–40.

Ruobing Xie, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2017. Image-embodied knowledge representation learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3140–3146.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Jin Xu, Yubo Tao, and Hai Lin. 2016. Semantic word cloud generation based on word embeddings. In *Pacific Visualization Symposium (PacificVis)*, pages 239–243. IEEE.

Mark Yatskar, Luke Zettlemoyer, and Ali Farhadi. 2016. Situation recognition: Visual semantic role labeling for image understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5534–5542.

Rowan Zellers and Yejin Choi. 2017. Zero-shot activity recognition with verb attribute induction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 946–958.

Heiga Zen, Keiichi Tokuda, and Alan W Black. 2009. Statistical parametric speech synthesis. *Speech Communication*, 51(11):1039–1064.

Krunoslav Zubrinic, Damir Kalpic, and Mario Milicevic. 2012. The automatic creation of concept maps from documents written using morphologically rich languages. *Expert systems with applications*, 39(16):12709–12718.

Stress Test Evaluation for Natural Language Inference

Aakanksha Naik[1]*, Abhilasha Ravichander[1]*,
Norman Sadeh[2], Carolyn Rose[1], Graham Neubig[1]
[1]Language Technologies Institute, Carnegie Mellon University
[2]Institute of Software Research, Carnegie Mellon University
{anaik, aravicha, sadeh, cprose, gneubig}@cs.cmu.edu

Abstract

Natural language inference (NLI) is the task of determining if a natural language hypothesis can be inferred from a given premise in a justifiable manner. NLI was proposed as a benchmark task for natural language understanding. Existing models perform well at standard datasets for NLI, achieving impressive results across different genres of text. However, the extent to which these models understand the semantic content of sentences is unclear. In this work, we propose an evaluation methodology consisting of automatically constructed "stress tests" that allow us to examine whether systems have the ability to make real inferential decisions. Our evaluation of six sentence-encoder models on these stress tests reveals strengths and weaknesses of these models with respect to challenging linguistic phenomena, and suggests important directions for future work in this area.

1 Introduction

Natural language inference (NLI), also known as recognizing textual entailment (RTE), is concerned with determining the relationship between a premise sentence and an associated hypothesis. This requires a model to make the 3-way decision of whether a hypothesis is true given the premise (*entailment*), false given the premise (*contradiction*), or whether the truth value cannot be determined (*neutral*). NLI has been proposed as a benchmark task for natural language understanding research (Cooper et al., 1996; Dagan et al., 2006; Giampiccolo et al., 2007; Dagan et al., 2013; Bowman et al., 2015; Nangia et al., 2017), due to the requirement for models to reason about several difficult linguistic phenomena[1] to perform well at this task (Bowman et al., 2015; Williams et al., 2017).

Due to its importance, significant prior work (Dagan et al., 2006; Dagan et al., 2009; Dagan et al., 2013; Marelli et al., 2014a) has focused on developing datasets and models for this benchmark task. Most recently, this task has been concretely implemented in the Stanford NLI (SNLI; Bowman et al. (2015)), and Multi-genre NLI (MultiNLI; Williams et al. (2017)) datasets, where crowdworkers are given a premise sentence and asked to generate novel sentences representing the three categories of entailment relations. Within the scope of these benchmark datasets, state-of-the-art deep learning-based sentence encoder models[2] (Nie and Bansal (2017), Chen et al. (2017), Conneau et al. (2017), Balazs et al. (2017), *inter alia*) have been shown to consistently achieve high accuracies, which may lead us to believe that these models excel at NLI across genres of text. However, machine learning models are known to exploit idiosyncrasies of how the data was produced, allowing them to imitate desired behavior using tricks such as pattern matching (Levesque, 2014; Rimell et al., 2009; Papernot et al., 2017). In NLI this arises when the test set contains many easy examples, and the extent to which difficult components of language understanding are required is masked within traditional evaluation. Therefore we ask a natural question:

*The first two authors contributed equally to this work.

[1]such as scope, coreference, quantification, lexical ambiguity, modality and belief.

[2]Sentence-encoder models are considered to be especially important to natural language understanding due to the requirement to represent sentences as fixed length vectors and perform reasoning based on these representations (Nangia et al., 2017). Thus, they will be our primary focus.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2340–2353
Santa Fe, New Mexico, USA, August 20-26, 2018.

is good model performance on NLI benchmarks a result of sophisticated pattern matching, or does it reflect true competence at natural language understanding?

In this work, we propose an evaluation based on "stress" tests for NLI, which tests robustness of NLI models to specific linguistic phenomena. This methodology is inspired by the work of Jia and Liang (2017) (and other related work: §5), which proposes the use of adversarial evaluation for reading comprehension by adding a distracting sentence at the end of a paragraph (known as concatenative adversaries), and evaluating models on this test set. However, this evaluation scheme cannot easily be applied to NLI as i) the adversarial perturbations suggested are not *label preserving*, that is the semantic relation between premise and hypothesis will not be maintained by such edits, ii) the perturbations may decrease system performance but are not interpretable and iii) premise-hypothesis pairs in inference usually consist of a single sentence, but concatenative adversaries break this assumption. In addition, besides evaluating model robustness to *distractions* in the form of adversarial examples, we are also interested in evaluating model *competence* on types of reasoning necessary to perform well at the task.

The proposed method offers a fine-grained evaluation for NLI (with label-preserving adversarial perturbations when required), in the form of "stress tests". In the stress testing methodology, systems are tested beyond normal operational capacity in order to identify weaknesses and to confirm that intended specifications are being met (Hartman and Owens, 1967; Tretmans, 1999; Beizer, 2003; Pressman, 2005; Nelson, 2009). To construct these stress tests for NLI, we first examine the predictions of the best-performing sentence encoder model on MultiNLI (Nie and Bansal, 2017), and create a typology of phenomena that it finds challenging (§2). Based on this typology, we present a methodology to automatically construct stress tests, which may cause models that suffer from similar weaknesses to fail (§3). The resulting tests make it possible to perform evaluation on a phenomenon-by-phenomenon basis, which is not the case for Jia and Liang (2017).[3] We benchmark the performance of four state-of-the-art models on the MultiNLI dataset on our constructed stress tests (§4), and observe performance drops across stress tests. We view these results as a first step towards robust, fine-grained evaluation of NLI systems. To encourage development of models that perform true natural language understanding for NLI, we release our code and all stress tests for future evaluation. [4]

2 Weaknesses of State-of-the-art NLI

Before creating adversarial examples that challenge state-of-the-art systems, it is helpful to study what phenomena systems find difficult. To elucidate this, we conduct a comprehensive error analysis of the best-performing sentence encoder model for MutliNLI (Nie and Bansal, 2017). For this analysis, we sample 100 misclassified examples from both genre-matched and mismatched sets, analyze their potential sources of errors, and group them into a typology of common reasons for error.In the end, the reasons for errors can broadly be divided into the following categories,[5] with examples shown in Table 1:

1. **Word Overlap (29%)**: Large word-overlap between premise and hypothesis sentences causes wrong entailment prediction, even if they are unrelated. Very little word overlap causes a prediction of neutral instead of entailment.

2. **Negation (13%)**: Strong negation words ("no", "not") cause the model to predict contradiction for neutral or entailed statements.

3. **Antonymy (5%)**: Premise-hypothesis pairs containing antonyms (instead of explicit negation) are not detected as contradiction by the model.

4. **Numerical Reasoning (4%)**: For some premise-hypothesis pairs, the model is unable to perform reasoning involving numbers or quantifiers for correct relation prediction.

[3] Isabelle et al. (2017) have proposed a similar fine-grained evaluation approach for machine translation, but it requires the manual construction of examples, unlike our automatic approach.

[4] All stress tests and resources available at `https://abhilasharavichander.github.io/NLI_StressTest/`

[5] % indicates what proportion of examples from our error analysis on the matched development set, fall into each category. Results on the mismatched development set follow similar trends and are available in appendix A

Error	Premise	Hypothesis
Word Overlap (N→E)	And, could it not result in a decline in Postal Service volumes across–the–board?	There may not be a decline in Postal Service volumes across–the–board.
Negation (E→C)	Enthusiasm for Disney's Broadway production of The Lion King dwindles.	The broadway production of The Lion King is no longer enthusiastically attended.
Numerical Reasoning (C→E)	Deborah Pryce said Ohio Legal Services in Columbus will receive a $200,000 federal grant toward an online legal self-help center.	A $900,000 federal grant will be received by Missouri Legal Services, said Deborah Pryce.
Antonymy (C→E)	"Have her show it," said Thorn.	Thorn told her to hide it.
Length Mismatch (C→N)	So you know well a lot of the stuff you hear coming from South Africa now and from West Africa that's considered world music because it's not particularly using certain types of folk styles.	They rely too heavily on the types of folk styles.
Grammaticality (N→E)	So if there are something interesting or something worried, please give me a call at any time.	The person is open to take a call anytime.
Real World Knowledge (E→N)	It was still night.	The sun hadn't risen yet, for the moon was shining daringly in the sky.
Ambiguity (E→N)	Outside the cathedral you will find a statue of John Knox with Bible in hand.	John Knox was someone who read the Bible.
Unknown (E→C)	We're going to try something different this morning, said Jon.	Jon decided to try a new approach.

Table 1: Examples of misclassified samples for each error category (Gold Label→Predicted Label)

5. **Length Mismatch (3%)**: The premise is much longer than the hypothesis and this extra information could act as a distraction for the model.

6. **Grammaticality (3%)**: The premise or the hypothesis is ill-formed because of spelling errors or incorrect subject-verb agreement.

7. **Real-World Knowledge (12%)**: These examples are hard to classify without some real-world knowledge.

8. **Ambiguity (6%)**: For some instances, the correct answer is unclear to humans. These are the most difficult cases.

9. **Unknown (26%)**: No obvious source of error is discernible in these samples.

Some of our error categories such as real world knowledge are well-known "hard" problems. However, error categories such as negation scope and antonymy, are crucial for natural language understanding and been of significant interest to study in formal semantics (Kroch, 1974; Muehleisen, 1997; Murphy, 2003; Moscati, 2006; Brandtler, 2006). Some of these phenomena have long been suspected to be challenging for entailment models (Jijkoun and De Rijke, 2006; LoBue and Yates, 2011; Roy, 2017). In fact, at roughly the same time as this submission, Gururangan et al. (2018) corroborate our findings by identifying lexical choice such as negation words, as well as sentence length as biasing factors in NLI datasets.

3 Stress Test Set Construction

While the error analysis is informative for Nie and Bansal (2017), performing a manual analysis for every system is not scalable. To create an automatically calculable proxy, we focus on automatically constructing large-scale datasets (*stress tests*), which test NLI models on phenomena that account for most errors in our analysis. In particular, we generate adversarial examples which test "word overlap", "negation", "length mismatch", "antonyms", "spelling error" and "numerical reasoning".[6]

[6]Notably, we focus only on "spelling error" for "grammaticality". We omit "real world knowledge" as it is not trivial to create a large dataset without human input, the "ambiguity" category because it is unreasonable that models can handle such cases, and the "unknown" category because it does not correspond to a particular phenomenon.

Error Cat.	Premise	Hypothesis
Antonyms	I love the Cinderella story.	I hate the Cinderella story.
Numerical Reasoning	Tim has 350 pounds of cement in 100, 50, and 25 pound bags	Tim has less than 750 pounds of cement in 100, 50, and 25 pound bags
Word Overlap	Possibly no other country has had such a turbulent history.	The country's history has been turbulent and true is true
Negation	Possibly no other country has had such a turbulent history.	The country's history has been turbulent and false is not true
Length Mismatch	Possibly no other country has had such a turbulent history and true is true and true is true and true is true and true is true and true is true	The country's history has been turbulent.
Spelling Errors	As he emerged, Boris remarked, glancing up at teh clock: "You are early	Boris had just arrived at the rendezvous when he appeared

Table 2: Example constructions from stress tests

We organize our stress tests into three classes, based on their perceived difficulty for the model. The first class (*competence tests*) evaluates the model's ability to reason about quantities and understand antonymy relations. The second class (*distraction tests*), estimates model robustness to shallow distractions such as lexical similarity or presence of negation words. This category contains "word overlap", "negation" and "length mismatch" tests. The final class (*noise tests*) checks model robustness to noisy data and consists of our "spelling error" test. Our adversarial construction uses three techniques: heuristic rules with external knowledge sources (for competence tests), a propositional logic framework (for distraction tests) and randomized perturbation (for noise tests). The following subsections describe our stress test construction, with examples shown in Table 2.

3.1 Competence Test Construction

Antonymy: For this construction, we consider every sentence from premise-hypothesis pairs in the development set independently. We perform word-sense disambiguation for each adjective and noun in the sentence using the Lesk algorithm (Lesk, 1986). We then sample an antonym for the word from WordNet (Miller, 1995). The sentence with the word substituted by its antonym and the original sentence become a new premise-hypothesis pair in our set. This results in 1561 and 1734 premise-hypothesis pairs for matched and mismatched sets respectively.

Substituting a word with its antonym may not always result in a contradiction [7] . Hence, three annotators were provided 100 random samples from the stress test set to evaluate for correctness. At least two annotators agreed on 86% of the labels being contradiction. We also evaluated grammaticality of our constructions, with at least two annotators agreeing on 87% being grammatical.

Numerical Reasoning: Creating a stress test for numerical reasoning is non-trivial as most of the MultiNLI development set does not involve quantities. Hence, we extract premise sentences from AQuA-RAT (Ling et al., 2017a), a dataset specifically focused on algebraic word problems along with rationales for their solutions. However, word problems from AQuA-RAT are quite complicated[8] and general-purpose NLI models cannot reasonably be expected to solve them.

To generate a reasonable set of premise sentences, we first discard problems which do not have numerical answers or have long rationales (>3 sentences) as such problems are inherently complex. We then split all problems into individual sentences and discard sentences without numbers, resulting in a set of 40,000 sentences. From this set, we discard sentences which do not contain at least one named entity (we consider "PERSON", "LOCATION" and "ORGANIZATION"), since such sentences mostly deal with abstract concepts.[9] This results in a set of 2500 premise sentences. For each premise, we generate entailed, contradictory and neutral hypotheses using heuristic rules:

[7] Consider examples of sentences with modalities, belief, conjunction or even conversational text such as *"They can change the tone of people's voice yes."*, *"They can change the tone of people's voice no."*, coreference, word substitution in metaphors or failure of word sense disambiguation.

[8] Including concepts such as probability, geometry and theoretical proofs.

[9] For example, "Find the smallest number of five digits exactly divisible by 22, 33, 66 and 44".

1. **Entailment:** Randomly choose and change one numerical quantity from the premise, prefixing it with the phrase "less than" or "more than" based on whether the new number is higher or lower.
2. **Contradiction:** Perform one of two actions with equal probability: randomly choose a numerical quantity from the premise and change it, or randomly choose a numerical quantity from the premise and prefix it with "less than/ more than" without changing it.
3. **Neutral:** Flip the corresponding entailed premise-hypothesis pair.
Using these rules, we generate a set of 7,596 premise-hypothesis pairs testing models on their ability to perform numerical reasoning. We further instruct three human annotators to evaluate 100 randomly sampled examples for difficulty, grammaticality and label correctness (since the labels are automatically generated). At least two annotators agreed with our generated label for 91% of the samples. Additionally, at least two annotators agreed on 92% of the examples being grammatical, and 98% being trivial numerical reasoning for humans.

3.2 Distraction Test Construction

This class includes stress tests for "word overlap", "negation" and "length mismatch", which test model ability to avoid getting distracted by simple cues such as lexical similarity or strong negation words. Models usually exploit such cues to achieve high performance since they have strong but spurious correlations with gold labels, but reliance on shallow reasoning can be used to distract models easily, as we demonstrate. We use a framework inspired by propositional logic to construct adversarial examples.

Propositional Logic Framework: Assume a premise p and a hypothesis h. For entailment, $(p \Rightarrow h) \implies (p \wedge True \Rightarrow h)$ since $(p \wedge True = p)$. Similarly for contradiction, $(p \Rightarrow\Leftarrow h) \implies (p \wedge True \Rightarrow \Leftarrow h)$ and if p and h are neutral, they remain neutral. In other words, if the premise or hypothesis is in conjunction with a statement that is independently true in all worlds, the entailment relation is preserved.

The next step is to construct such statements whose values are true in all worlds (*tautologies*). We then define stress test accuracy for NLI as :

$$Adv(a) = \frac{1}{|D_{test}|} \sum_{i \in D_{test}} \mathbb{1}\{f(p, h, a) = c_i\}$$

where a is an adversarial tautology which can be attached as a conjunction to either the premise or hypothesis without changing the relation. We use this framework to construct distraction tests for "word overlap", "negation" and "length mismatch". For all sets, we use simple tautologies, which do not contain words that share any topical significance with the premise or hypothesis.

A natural concern is that statements obtained from such constructions are unnatural (Grice, 1975), making the NLI task more difficult for humans. To study this, we run a human evaluation where three annotators are shown premise-hypothesis pairs from these sets and instructed to label the relation. On word overlap, we find the provided label has 91% agreement with the gold label. For length mismatch, the provided label has 85% agreement with gold. This is similar to the agreement reported in Williams et al. (2017), leading us to believe the constructed examples are not too unnatural or difficult. The constructions also remain grammatical; after annotating 100 samples from our adversarially generated set, only two were deemed ungrammatical, and both were because of reasons unrelated to our perturbations. Specific details for our sets are as follows:
Word Overlap: For this set, we append the tautology *"and true is true"* to the end of the hypothesis sentence for every example in the MultiNLI development set.
Negation: For this set, we append the tautology *"and false is not true"*, which contains a strong negation word (*"not"*), to the end of the hypothesis sentence for every example in the MultiNLI development set.
Length Mismatch: For this adversarial set, we append the tautology *"and true is true"* five times to the end of the premise sentence for every example in the MultiNLI development set. We modify the premise sentence in this case as we hypothesize that errors in this category mainly arise due to the premise sentence being unwieldy.

System	Original MultiNLI Dev		Competence Test			Distraction Test						Noise Test	
			Antonymy		Numerical	Word Overlap		Negation		Length Mismatch		Spelling Error	
	Mat	Mis	Mat	Mis	Reasoning	Mat	Mis	Mat	Mis	Mat	Mis	Mat	Mis
NB	74.2	74.8	15.1	19.3	21.2	47.2	47.1	39.5	40.0	48.2	47.3	51.1	49.8
CH	73.7	72.8	11.6	9.3	30.3	58.3	58.4	52.4	52.2	63.7	65.0	68.3	69.1
RC	71.3	71.6	36.4	32.8	30.2	53.7	54.4	49.5	50.4	48.6	49.6	66.6	67.0
IS	70.3	70.6	14.4	10.2	28.8	50.0	50.2	46.8	46.6	58.7	59.4	58.3	59.4
BiLSTM	70.2	70.8	13.2	9.8	31.3	57.0	58.5	51.4	51.9	49.7	51.2	65.0	65.1
CBOW	63.5	64.2	6.3	3.6	30.3	53.6	55.6	43.7	44.2	48.0	49.3	60.3	60.6

Table 3: Classification accuracy (%) of state-of-the-art models on our constructed stress tests. Accuracies shown on both genre-matched and mismatched categories for each stress set. For reference, random baseline accuracy is 33%.

3.3 Noise Test Construction

This class consists of an adversarial example set which tests model robustness to spelling errors. Spelling errors occur often in MultiNLI data, due to involvement of Turkers and noisy source text (Ghaeini et al., 2018), which is problematic as some NLI systems rely heavily on word embeddings. Inspired by Belinkov and Bisk (2017), we construct a stress test for "spelling errors" by performing two types of perturbations on a word sampled randomly from the hypothesis: random swap of adjacent characters within the word (for example, *"I saw Tipper with him at teh movie."*), and random substitution of a single alphabetical character with the character next to it on the English keyboard. For example, *"Agencies have been further restricted and given less choice in selecting contractimg methods"*.

4 Experiments

4.1 Experimental Setup

We focus on the following sentence-encoder models, which achieve strong performance on MultiNLI:
Nie and Bansal (2017) (NB): This model uses a sentence encoder consisting of stacked BiLSTM-RNNs with shortcut connections and fine-tuning of embeddings. It achieves the top non-ensemble result in the RepEval-2017 shared task (Nangia et al., 2017).
Chen et al. (2017) (CH): This model also uses a sentence encoder consisting of stacked BiLSTM-RNNs with shortcut connections. Additionally, it makes use of character-composition word embeddings learned via CNNs, intra-sentence gated attention and ensembling to achieve the best overall result in the RepEval-2017 shared task.
Balazs et al. (2017) (RiverCorners - RC): This model uses a single-layer BiLSTM with mean pooling and intra-sentence attention.
Conneau et al. (2017) (InferSent - IS): This model uses a single-layer BiLSTM-RNN with max-pooling. It is shown to learn robust universal sentence representations which transfer well across several inference tasks.
We also set up two simple baseline models:
BiLSTM: The simple BiLSTM baseline model described by Nangia et al. (2017).
CBOW: A bag-of-words sentence representation from word embeddings.

4.2 Model Performance on Stress Tests

Table 3 shows the classification accuracy of all six models on our stress tests and the original MultiNLI development set. We see that performance of all models drops across all stress tests. On competence stress tests, no model is a clear winner, with **RC** and **CH** performing best on antonymy and numerical reasoning respectively. On distraction tests, **CH** is the best-performing model, suggesting that their gated-attention mechanism handles shallow word-level distractions to some extent. Interestingly, our

BiLSTM baseline is the second-best model on two out of three distraction tests. On the noise test, **CH**, **RC** and both baselines **[BiLSTM;CBOW]** do not show much performance degradation, most likely due to the benefit of subword modeling via character-CNNs and the use of mean pooling. We further analyze model performance on each class of tests.

4.3 Model Competence

Model Performance on Antonymy: Table 3 shows that all models perform poorly on antonymy. **RC** achieves the best performance, with 36.4% and 32.8% on matched and mismatched sets respectively which is just higher than random performance. Our analysis shows that models tend to overpredict entailment (due to a high amount of word overlap in this test). This accounts for, on average, 86.4% and 87.6% of total errors on matched and mismatched sets.[10]

We study which antonym pairs are easy and difficult for models by examining the errors of the best and worst performing models on this test **[RC;CH]**. On 982 samples where both models fail, we find 617 unique antonym pairs, and on 171 samples where both models succeed, we find 84 unique antonym pairs. 89.8% of the "easy" and 57.2% of the "hard" antonym pairs appear in a contradiction relation within the training data, suggesting that models succeed on easy antonym-pairs seen in the training data but struggle to generalize.

We were also curious about error variation by antonym type. We randomly sample 100 examples where both models fail and 100 samples where both succeed, and manually annotate whether the antonym present was gradable, relational or complementary. Among successful examples, 99% are complementary antonyms with only one relational antonym. Amongst the failure cases, 20% are relational antonym pairs, 73% are complementary and 7% are gradable, suggesting that models find relational and gradable antonyms hard, but get complementary antonyms both right and wrong. Finally, we examine differences between models by analyzing examples classified correctly by the best model which are not handled by the worst. We find that antonym pairs recognized by the weaker model occur, on average, nearly twice as often in the training data as antonym pairs recognized by the stronger model, suggesting that **RC** is able to learn antonymy from fewer examples (though these examples must be present in training data).

Model Performance on Numerical Reasoning: Table 3 shows that all models exhibit a significant performance drop on numerical reasoning, with none achieving an accuracy better than random (33%). We analyze the predictions of the best and worst performing models on this test **[BiLSTM;NB]**. The biggest source of common errors for both models (1703 out of 4337 errors) is misclassifying neutral pairs as entailment, which arises because our construction technique flips entailed premise-hypothesis pairs to create neutral pairs, leading to high word overlap for neutral pairs. Our constructions also lead to high word overlap for contradiction pairs, leading to a large number of C-E errors for both models (1695 out of 4337 errors). Thus, 78.3% of all errors are caused due to the models falsely predicting entailment. Most of the remaining errors are caused by entailment examples containing the phrases "more than" or "less than" being incorrectly classified as contradiction. This behavior could arise as these phrases are often used by crowdworkers to create contradictory examples in the original MultiNLI data, fooling models into marking examples with this phrase as "contradiction" without reasoning about involved quantities. Our observations suggest that models do not perform quantitative reasoning, but rely on word overlap and other shallow lexical cues for prediction.

4.4 Model Distraction

Our distraction tests are designed to check model robustness to: 1) decreasing lexical similarity between premise-hypothesis pairs, 2) strong negation words in sentence pairs.

Effect of Decreasing Lexical Similarity: Due to our construction methodology (appending tautologies), accuracy on *word overlap* and *length mismatch* demonstrates the effect of decreasing lexical similarity

[10] Detailed results from this analysis are provided in appendix B

System	MultiNLI Dev		Word Overlap		Length Mismatch	
	Mat	Mis	Mat	Mis	Mat	Mis
NB	33.2	33.1	43.2	38.3	46.0	46.9
CH	32.9	31.7	84.7	85.3	65.8	65.8
RC	37.1	39.1	74.3	83.3	74.2	79.5
IS	34.7	31.4	86.3	87.0	43.5	44.2
BiLSTM	38.5	37.9	83.2	81.9	75.9	79.1
CBOW	33.9	30.2	74.5	72.3	54.7	59.9

Table 4: % of FALSE NEUTRAL in total errors on MultiNLI development set, word overlap test and length mismatch test.

System	BiLSTM		NB	
	Mat	Mis	Mat	Mis
MultiNLI Dev	70.2	70.4	66.6	66.6
NEGATION	68.9	70.4	49.3	48.7
DIFF TAUT	49.0	49.3	49.9	49.7

Table 5: Effect of training on distraction data on original DEV set, original distraction set and new distraction set

on model performance. Table 3, shows accuracy decreases for all models on both tests. This drop is lower for **CH**, suggesting that their gated attention mechanism might help in focusing on relevant parts of the sentence.

The significant decrease in accuracy indicates that lexical similarity is a strong signal for entailment prediction, failing which models default to predicting neutral. To provide further justification, we compare the proportion of false neutral errors for all models on word overlap and length mismatch stress sets vs. the original MultiNLI development set. As shown in Table 4, we find it increases for all models on both sets.

Effect of Introducing Strong Negation Words: Table 3 shows results on negation, and we see that all state-of-the-art models perform poorly, with accuracies decreasing by 23.4% and 23.38%, on average, on matched and mismatched sets respectively. However, comparing the number of E-C (entailment predicted as contradiction) and N-C (neutral predicted as contradiction) errors for these models on the negation test vs. the original MultiNLI development set, we do not find an increase in these error types on negation. Instead, we observe an increase in false neutral errors for all models. This could occur due to the introduction of extra words ("false", "is" and "true") apart from "not", indicating that decreasing lexical similarity has a stronger effect on models than introducing negation.

4.5 Effect of Noise

Our noise test results in Table 3 show that **NB** and **IS** exhibit a huge decrease in accuracy, since both models rely on word embeddings. Other models show little performance degradation on this test. **CH** performs subword modeling via character-level CNNs, which provides robustness towards perturbation attacks. **RC** and **BiLSTM** perform well despite relying on word embeddings since both use mean pooling, which might reduce the effect of single-word edits on the final representation. **CBOW** is also very robust to this test, which can arise from the fact that it sums word embeddings to create the final sentence embedding, diluting the effect of changing a single word on final model performance[11].

4.6 Training with Distraction

Finally, we study the effect of training with distractions generated via our adversarial construction. We generate an equivalent sample with the negation distraction for every sample in the training data, and retrain **NB** and **BiLSTM** on the union of these examples and original training data.

We observe the performance of the trained models on three tests: the original MultiNLI development set, the negation stress test and a new distraction test creating using a different tautology *"green is not red"* (DIFF TAUT). We observe that **NB** shows performance degradation across all tests, but training

[11] We analyze difference in model performance across perturbation techniques such as adjacent character swapping, keyboard character swapping, function word and content word perturbations, but do not observe significant differences. Results from these experiments are included in appendix C.

BiLSTM on distraction data helps it become robust to the tautology it was trained on. However, it collapses when evaluated on a different tautology. Ignoring such distractions is something humans do naturally. Models should not have to train on the specific distraction to succeed on this evaluation.

5 Related Work and Discussion

Adversarial evaluation schemes have been proposed to evaluate model robustness on various NLP tasks. Smith (2012) discuss dangers of community-wide "overfitting" to benchmark datasets and emphasize the need to correlate model errors to well-defined linguistic phenomena to understand specific model strengths and weaknesses. Prior work (Rimell et al., 2009; Schneider et al., 2017) performed analyses of model errors in dependency parsing and information extraction. Motivated by this desideratum, we analyze errors in Multi-Genre NLI and automatically construct large stress sets to evaluate NLI systems on identified difficulties. This is analogous to recent work (Jia and Liang, 2017; Burlot and Yvon, 2017) on developing automated adversarial evaluation schemes for reading comprehension and machine translation. However, unlike these efforts, our stress tests allow us to study model performance on a range of linguistic phenomena. Unlike work on manual construction of small adversarial evaluation sets for various NLP tasks (Levesque, 2014; Mahler et al., 2017; Staliūnaite and Bonfil, 2017; Isabelle et al., 2017; Belinkov and Bisk, 2017; Bawden et al., 2017), our work focuses on a more exhaustive large-scale evaluation for NLI. It is interesting to note that Bayer et al. (2006) discuss the daunting cost of finding entailment pairs for NLI evaluation, but our techniques can be used to construct such pairs with low cost.

The NLI task attracted significant interest before datasets became large enough for the application of neural methods (Glickman et al., 2005; Harabagiu and Hickl, 2006; Romano et al., 2006; Dagan et al., 2006; Giampiccolo et al., 2007; Dagan et al., 2010; MacCartney, 2009; Zanzotto et al., 2006; Malakasiotis and Androutsopoulos, 2007; Haghighi et al., 2005; Angeli and Manning, 2014). de Marneffe et al. (2009) analyze the effect of multi-word expressions and find that they do not significantly affect the performance of NLI systems. Perhaps the closest to our contribution are the works of Cooper et al. (1996), which manually constructs sentences containing phenomena that NLI systems are expected to handle (FraCaS), and Marelli et al. (2014b), which constructs sentences that require compositional knowledge (SICK). Our constructions differ from SICK and FraCaS in several aspects. Since we use large datasets (Williams et al., 2017; Ling et al., 2017b) as base data, our sets are larger and more lexically diverse than both SICK (which used a seed set of 1500 sentences) and FraCaS (which was manually constructed). Secondly, while SICK uses handcrafted rules and incorporates linguistic phenomena, sentence-pairs are not constrained to exhibit only one phenomenon, which may introduce confounding factors during analysis. Though FraCaS follows the constraint of restricting sentence-pairs to exhibit only one phenomenon, it contains very few examples of each phenomenon. Conversely, our techniques generate large evaluation sets, with each set focusing on a single phenomenon, providing a testbed for fine-grained evaluation and analysis. Lastly, our stress tests are grounded in failings of current state-of-the-art models, rather than on phenomena hypothesized to be challenging for NLI models. Our evaluation sets also differ from the small portion of the MultiNLI development set annotated for challenging linguistic phenomena, as similar to SICK, each sentence pair is not constrained to exhibit a single phenomenon. In addition, presence of biases in the MultiNLI development and test data (Gururangan et al., 2018; Poliak et al., 2018) could also lead to models exploiting them as shallow cues for prediction (for example, the performance of baseline models on the subset of the MultiNLI dataset annotated for antonymy averages 67% , while the same baselines perform much worse on our antonymy stress-test).

We hope that insights derived from our stress tests will stimulate future research in NLI. One promising direction would be the development and investigation of more linguistically-motivated neural models on NLI (such as models which incorporate explicit negation scope information or semantic roles for example), as our stress tests now provide a framework for in-depth analysis of model performance and demonstrate significant room for improvement in these areas. While we benchmark the performance of state-of-the-art models on our stress tests, in the future, it would be interesting to investigate which architectural choices contribute to model successes and *why*. Another interesting research direction is the identification of "core competencies", such as quantitative reasoning or antonymy, which can

enhance model performance across multiple NLP tasks such as sentiment analysis, question answering and relation extraction and studying transfer of representations from "competent" models.

6 Conclusion

In this work, we present a suite of large-scale stress tests to perform targeted evaluation of NLI models, along with a set of techniques for their automatic construction. Our stress tests evaluate a model's ability to reason about quantities and antonymy (*competence tests*), its susceptibility to shallow lexical cues (*distraction tests*) and its robustness to random perturbations (*noise tests*). We benchmark the performance of four state-of-the-art sentence encoding models on our tests and find that they struggle on many phenomena, despite reporting high accuracy on NLI.

Overall, we consider the MultiNLI dataset to be a valuable resource for the NLP community, with entailment pairs drawn from several different genres of text. However, we argue that the community would benefit by having NLI models pass sanity checks, in the form of "stress tests", to ensure models evolve against exploiting simple idiosyncrasies of training data. Similar to Isabelle et al. (2017), we intend our stress tests to supplement existing NLI evaluation rather than replace it. In the future, we hope benchmarking model performance on stress tests in addition to standard evaluation criteria will provide deeper insight into model strengths and weaknesses, and guide more informed model choices. We would also like to note that the "stress test" evaluation paradigm that we propose for NLI can be further updated in the future when new forms of models are devised, increasing the coverage of the tests to cover problems of future models as well. We release all our stress tests and associated resources to the community to promote work on models that get us closer to true natural language understanding.

Acknowledgements

This work has partially been supported by the National Science Foundation under Grant No. CNS 13-30596. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the NSF, or the US Government. This work has also been supported through the CMLH Fellowship in Digital Health for the author Naik. The authors would also like to thank Shruti Rijhwani, Siddharth Dalmia, Shruti Palaskar, Khyathi Chandu, Aditya Chandrasekar, Paul Michel, Varshini Ramaseshan and Rajat Kulshreshtha for helpful discussion and feedback with various aspects of this work.

References

[Angeli and Manning2014] Gabor Angeli and Christopher D Manning. 2014. Naturalli: Natural logic inference for common sense reasoning. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 534–545.

[Balazs et al.2017] Jorge Balazs, Edison Marrese-Taylor, Pablo Loyola, and Yutaka Matsuo. 2017. Refining raw sentence representations for textual entailment recognition via attention. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*, pages 51–55, Copenhagen, Denmark, September. Association for Computational Linguistics.

[Bawden et al.2017] Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. 2017. Evaluating discourse phenomena in neural machine translation. *arXiv preprint arXiv:1711.00513*.

[Bayer et al.2006] Sam Bayer, John Burger, Lisa Ferro, John Henderson, Lynette Hirschman, and Alex Yeh. 2006. Evaluating semantic evaluations: How rte measures up. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 309–331. Springer.

[Beizer2003] Boris Beizer. 2003. *Software Testing Techniques*. Dreamtech Press.

[Belinkov and Bisk2017] Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

[Bowman et al.2015] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on*

Empirical Methods in Natural Language Processing, pages 632–642, Lisbon, Portugal, September. Association for Computational Linguistics.

[Brandtler2006] Johan Brandtler. 2006. On aristotle and baldness: Topic, reference, presupposition of existence, and negation. *Working papers in Scandinavian syntax*, 77:177–204.

[Burlot and Yvon2017] Franck Burlot and François Yvon. 2017. Evaluating the morphological competence of machine translation systems. In *Proceedings of the Second Conference on Machine Translation*, pages 43–55.

[Chen et al.2017] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Recurrent neural network-based sentence encoder with gated attention for natural language inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*, pages 36–40, Copenhagen, Denmark, September. Association for Computational Linguistics.

[Conneau et al.2017] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark, September. Association for Computational Linguistics.

[Cooper et al.1996] Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, et al. 1996. Using the framework. Technical report.

[Dagan et al.2006] Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

[Dagan et al.2009] Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, 15(4):i–xvii.

[Dagan et al.2010] Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2010. The fourth pascal recognizing textual entailment challenge. *Journal of Natural Language Engineering*.

[Dagan et al.2013] Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220.

[de Marneffe et al.2009] Marie-Catherine de Marneffe, Sebastian Padó, and Christopher D Manning. 2009. Multi-word expressions in textual inference: Much ado about nothing? In *Proceedings of the 2009 Workshop on Applied Textual Inference*, pages 1–9. Association for Computational Linguistics.

[Ghaeini et al.2018] Reza Ghaeini, Sadid A Hasan, Vivek Datla, Joey Liu, Kathy Lee, Ashequl Qadir, Yuan Ling, Aaditya Prakash, Xiaoli Z Fern, and Oladimeji Farri. 2018. Dr-bilstm: Dependent reading bidirectional lstm for natural language inference. *arXiv preprint arXiv:1802.05577*.

[Giampiccolo et al.2007] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics.

[Glickman et al.2005] Oren Glickman, Ido Dagan, and Moshe Koppel. 2005. Web based probabilistic textual entailment.

[Grice1975] H Paul Grice. 1975. Logic and conversation. *1975*, pages 41–58.

[Gururangan et al.2018] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. *arXiv preprint arXiv:1803.02324*.

[Haghighi et al.2005] Aria Haghighi, Andrew Ng, and Christopher Manning. 2005. Robust textual inference via graph matching. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

[Harabagiu and Hickl2006] Sanda Harabagiu and Andrew Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 905–912. Association for Computational Linguistics.

[Hartman and Owens1967] Philip H. Hartman and David H. Owens. 1967. How to write software specifications. In *Proceedings of the November 14-16, 1967, Fall Joint Computer Conference*, AFIPS '67 (Fall), pages 779–790, New York, NY, USA. ACM.

[Isabelle et al.2017] Pierre Isabelle, Colin Cherry, and George Foster. 2017. A challenge set approach to evaluating machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2486–2496, Copenhagen, Denmark, September. Association for Computational Linguistics.

[Jia and Liang2017] Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark, September. Association for Computational Linguistics.

[Jijkoun and De Rijke2006] Valentin Jijkoun and Maarten De Rijke. 2006. Recognizing textual entailment: Is word similarity enough? In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 449–460. Springer.

[Kroch1974] Anthony S Kroch. 1974. *The semantics of scope in English.* Ph.D. thesis, Massachusetts Institute of Technology.

[Lesk1986] Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, SIGDOC '86, pages 24–26, New York, NY, USA. ACM.

[Levesque2014] Hector J Levesque. 2014. On our best behaviour. *Artificial Intelligence*, 212:27–35.

[Ling et al.2017a] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017a. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada, July. Association for Computational Linguistics.

[Ling et al.2017b] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017b. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

[LoBue and Yates2011] Peter LoBue and Alexander Yates. 2011. Types of common-sense knowledge needed for recognizing textual entailment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 329–334. Association for Computational Linguistics.

[MacCartney2009] Bill MacCartney. 2009. *Natural language inference*. Stanford University.

[Mahler et al.2017] Taylor Mahler, Willy Cheung, Micha Elsner, David King, Marie-Catherine de Marneffe, Cory Shain, Symon Stevens-Guille, and Michael White. 2017. Breaking nlp: Using morphosyntax, semantics, pragmatics and world knowledge to fool sentiment analysis systems. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 33–39, Copenhagen, Denmark, September. Association for Computational Linguistics.

[Malakasiotis and Androutsopoulos2007] Prodromos Malakasiotis and Ion Androutsopoulos. 2007. Learning textual entailment using svms and string similarity measures. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 42–47. Association for Computational Linguistics.

[Marelli et al.2014a] Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014a. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 1–8, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

[Marelli et al.2014b] Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, Roberto Zamparelli, et al. 2014b. A sick cure for the evaluation of compositional distributional semantic models.

[Miller1995] George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

[Moscati2006] Vincenzo Moscati. 2006. *The scope of negation*. Ph.D. thesis, Università degli Studi di Siena.

[Muehleisen1997] Victoria Lynn Muehleisen. 1997. *Antonymy and semantic range in english*. na.

[Murphy2003] M Lynne Murphy. 2003. *Semantic relations and the lexicon: Antonymy, synonymy and other paradigms*. Cambridge University Press.

[Nangia et al.2017] Nikita Nangia, Adina Williams, Angeliki Lazaridou, and Samuel Bowman. 2017. The repeval 2017 shared task: Multi-genre natural language inference with sentence representations. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*, pages 1–10, Copenhagen, Denmark, September. Association for Computational Linguistics.

[Nelson2009] Wayne B Nelson. 2009. *Accelerated testing: statistical models, test plans, and data analysis*, volume 344. John Wiley & Sons.

[Nie and Bansal2017] Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*, pages 41–45, Copenhagen, Denmark, September. Association for Computational Linguistics.

[Papernot et al.2017] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM.

[Poliak et al.2018] Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. *arXiv preprint arXiv:1805.01042*.

[Pressman2005] Roger S Pressman. 2005. *Software engineering: a practitioner's approach*. Palgrave Macmillan.

[Rimell et al.2009] Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 813–821. Association for Computational Linguistics.

[Romano et al.2006] Lorenza Romano, Milen Kouylekov, Idan Szpektor, Ido Dagan, and Alberto Lavelli. 2006. Investigating a generic paraphrase-based approach for relation extraction.

[Roy2017] Subhro Roy. 2017. *Reasoning about quantities in natural language*. Ph.D. thesis, University of Illinois at Urbana-Champaign.

[Schneider et al.2017] Rudolf Schneider, Tom Oberhauser, Tobias Klatt, Felix A. Gers, and Alexander Löser. 2017. Analysing errors of open information extraction systems. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 11–18, Copenhagen, Denmark, September. Association for Computational Linguistics.

[Smith2012] Noah A Smith. 2012. Adversarial evaluation for models of natural language. *arXiv preprint arXiv:1207.0245*.

[Staliūnaite and Bonfil2017] Ieva Staliūnaite and Ben Bonfil. 2017. Breaking sentiment analysis of movie reviews. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 61–64.

[Tretmans1999] Jan Tretmans. 1999. Testing concurrent systems: A formal approach. In *International Conference on Concurrency Theory*, pages 46–65. Springer.

[Williams et al.2017] Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

[Zanzotto et al.2006] F Zanzotto, Alessandro Moschitti, Marco Pennacchiotti, and M Pazienza. 2006. Learning textual entailment from examples. In *Second PASCAL recognizing textual entailment challenge*, page 50. PASCAL.

Appendix A. Error Analysis on Mismatched Set

Table 6 describes what proportion of misclassified examples from the mismatched set fall into each error category, as defined in the typology presented in §2.

Appendix B. Error Types on Antonymy

The high amount of word overlap in this test causes models to overpredict entailment, accounting for, on average, 86.4% and 87.6% of total errors on matched and mismatched sets respectively. We present the exact proportion of False Entailment and False Neutral errors in Table 7. Keep in mind there is only one gold class in this category: contradiction.

As expected, all four models make a high amount of false entailment errors because they notice high amounts of lexical similarity between the premise and the hypothesis.

Error Category	% of Misclassified Examples
Word Overlap	28
Negation	12
Antonymy	4
Numerical Reasoning	4
Length Mismatch	6
Grammaticality	4
Real-World Knowledge	14
Ambiguity	8
Unknown	20

Table 6: Distribution of misclassified examples from the MultiNLI mismatched development set.

System	C-E Errors		C-N Errors	
	Mat	**Mis**	**Mat**	**Mis**
NB	79.83	82.40	20.17	17.60
CH	99.78	99.75	0.22	0.25
RC	66.67	68.50	33.33	31.50
IS	99.40	99.81	0.60	0.19

Table 7: Percentage of C-E and C-N errors on antonymy test.

Appendix C. Additional Experiments with Spelling Error Stress Tests

This stress test consists of an adversarial example set which tests model robustness to spelling errors. Spelling errors occur often in MultiNLI data, due to involvement of Turkers and noisy source text, which is problematic as some NLI systems rely heavily on word embeddings. We construct a stress test for "spelling errors" by performing two types of perturbations:

AdjSWAP: Swap adjacent characters in a single word sampled randomly from the hypothesis. For example, *"I saw Tipper with him at teh movie."*.

KBSWAP: Substitute a single alphabetical character randomly sampled from the hypothesis with the character next to it on the English keyboard. For example, *"Agencies have been further restricted and given less choice in selecting contractimg methods."* We additionally perform perturbations on only function words (conjunctions, pronouns and articles), and on only content words (nouns and adjectives) in the hypothesis to study the effects. We do not address perturbations in verbs and adverbs in the content word vs. function word analysis. The results are presented in Table 8.

Sys	ADJ SWAP		KB SWAP		CN SWAP		FN SWAP	
tem	**Mat**	**Mis**	**Mat**	**Mis**	**Mat**	**Mis**	**Mat**	**Mis**
NB	43.0	42.9	47.7	47.9	51.1	49.8	49.7	49.6
CH	68.24	68.1	68.5	68.3	68.3	69.1	69.9	70.3
RC	66.6	66.4	67.0	66.8	66.6	67.0	68.4	68.4
IS	57.8	58.6	57.7	58.7	58.3	59.4	57.5	57.6

Table 8: Model Performance on grammaticality test

We observe that there is no significant effect of perturbing a function word or a content word. One hypothesis is that content words can often be named entities for which the models do not find word embeddings. We also do not find a considerable difference in performance between the different kinds of perturbations but this is expected behaviour as most models use word embeddings, and these will just be categorized as unknown words.

Grounded Textual Entailment

Hoa Trong Vu[*+], **Claudio Greco**[†], **Aliia Erofeeva**[†+], **Somayeh Jafaritazehjan**[*+]
Guido Linders[†+], **Marc Tanti**[*], **Alberto Testoni**[†], **Raffaella Bernardi**[†], **Albert Gatt**[*]
[+]Erasmus Mundus European Program in Language and Communication Technology
[*]University of Malta, [†]University of Trento
[*]name.surname@um.edu.mt, [†]name.surname@unitn.it

Abstract

Capturing semantic relations between sentences, such as entailment, is a long-standing challenge for computational semantics. Logic-based models analyse entailment in terms of possible worlds (interpretations, or situations) where a premise P entails a hypothesis H iff in all worlds where P is true, H is also true. Statistical models view this relationship probabilistically, addressing it in terms of whether a human would likely infer H from P. In this paper, we wish to bridge these two perspectives, by arguing for a visually-grounded version of the Textual Entailment task. Specifically, we ask whether models can perform better if, in addition to P and H, there is also an image (corresponding to the relevant "world" or "situation"). We use a multimodal version of the SNLI dataset (Bowman et al., 2015) and we compare "blind" and visually-augmented models of textual entailment. We show that visual information is beneficial, but we also conduct an in-depth error analysis that reveals that current multimodal models are not performing "grounding" in an optimal fashion.

1 Introduction

Evaluating the ability to infer information from a text is a crucial test of the capability of models to grasp meaning. As a result, the computational linguistics community has invested huge efforts into developing textual entailment (TE) datasets.

After formal semanticists developed FraCas in the mid '90 (Cooper et al., 1996), an increase in statistical approaches to computational semantics gave rise to the need for suitable evaluation datasets. Hence, Recognizing Textual Entailment (RTE) shared tasks have been organized regularly (Sammons et al., 2012). Recent work on compositional distributional models has motivated the development of the SICK dataset of sentence pairs in entailment relations for evaluating such models (Marelli et al., 2014). Further advances with Neural Networks (NNs) have once more motivated efforts to develop a large natural language inference dataset, SNLI (Bowman et al., 2015), since NNs need to be trained on big data.

However, meaning is not something we obtain just from text and the ability to reason is not unimodal either. The importance of enriching meaning representations with other modalities has been advocated by cognitive scientists, (e.g., (Andrews et al., 2009; Barsalou, 2010)) and computational linguists (e.g., (Glavaš et al., 2017)). While efforts have been put into developing multimodal datasets for the task of checking Semantic Text Similarity Text (Agirre et al., 2017), we are not aware of any available datasets to tackle the problem of Grounded Textual Entailment (GTE). Our paper is a first effort in this direction.

Textual Entailment is defined in terms of the likelihood of two sentences (a premise P and an hypothesis H) to be in a certain relation: P entails, contradicts or is unrelated to H. For instance, the premise "People trying to get warm in front of a chimney" and the hypothesis "People trying to get warm at home" are highly likely to be in an entailment relation. Our question is whether having an image that illustrates the event (e.g., Figure 1a) can help a model to capture the relation. In order to answer this

Correspondence should be addressed to Raffaella Bernardi (raffaella.bernardi@unitn.it) and Albert Gatt (albert.gatt@um.edu.mt).

 The dataset, annotation and code is available from https://github.com/claudiogreco/coling18-gte.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2354–2368
Santa Fe, New Mexico, USA, August 20-26, 2018.

(a) P: *A family in front of a chimney* and H: *A family trying to get warm*

(b) P: *People trying to get warm* and H: *People are outside on a chilly day*. TE: unrelated vs. GTE: related

Figure 1: Premise, Hypothesis and Image examples

question, we augment the largest available TE dataset with images, we enhance a state of the art model of textual entailment to take images into account and we evaluate it against the GTE task.

The inclusion of images can also alter relations which, based on text alone, would seem likely. For example, to a "blind" model the sentences of the sentence pair in Figure 1b would seem to be unrelated, but when the two sentences are viewed in the context of the image, they do become related.

A suitable GTE model therefore has to perform two sub-tasks: (a) it needs to ground its linguistic representations of P, H or both in non-linguistic (visual) data; (b) it needs to reason about the possible relationship between P and H (modulo the visual information).

2 Related Work

Grounding language through vision has recently become the focus of several tasks, including Image Captioning (IC, e.g. (Hodosh et al., 2013; Xu et al., 2015)) and Visual Question Answering (VQA, eg. (Malinowski and Fritz, 2014; Antol et al., 2015)), and even more recently, Visual Reasoning (Johnson et al., 2017; Suhr et al., 2017) and Visual Dialog (Das et al., 2017). Our focus is on Grounded Textual Entailment (GTE). While the literature on TE is rather vast, GTE is still rather unexplored territory.

Textual Entailment Throughout the history of Computational Linguistics various datasets have been built to evaluate Computational Semantics models on the TE task. Usually they contain data divided into entailment, contradiction or unknown classes. The "unknown" label has sometimes been replaced with the "unrelated" or "neutral" label, capturing slightly different types of phenomena. Interestingly, the "entailment" and "contradiction" classes also differ across datasets. In the mid-'90s a group of formal semanticists developed FraCaS (Framework for Computational Semantics). (Cooper et al., 1996)[1] The dataset contains logical entailment problems in which a conclusion has to be derived from one or more premises (but not necessarily all premises are needed to verify the entailment). The entailments are driven by logical properties of linguistic expressions, like the monotonicity of quantifiers, or their conservativity property etc. Hence, the set of premises entails the conclusion iff in all the interpretations (worlds) in which the premises are true the conclusion is also true; otherwise the conclusion contradicts the premises. In 2005, the PASCAL RTE (Recognizing Textual Entailment) challenge was launched, to become a task organized annually. In 2008, the RTE-4 committee made the task more fine-grained by requiring the classification of the pairs as "entailment", "contradiction" and "unknown" (Giampiccolo et al., 2008). The RTE datasets, unlike FraCaS, contain real-life natural language sentences and the sort of entailment problems which occur in corpora collected from the web. Importantly, the sentence pair relations are annotated as entailment, contradiction or neutral based on a likelihood condition: if a human reading the premise would typically infer that the conclusion (called the hypothesis) is most likely true (entailment), its negation is most likely true (contradiction) or the conclusion can be either true or false (neutral).

At SemEval 2014, in order to evaluate Compositional Distributional Semantics Models focusing on the compositionality ability of those models, the SICK dataset (Sentences Involving Compositional Knowledge) was used in a shared entailment task (Marelli et al., 2014). Sentence pairs were obtained through

[1] A cleanly processable version of it has been made available only recently: `http://www-nlp.stanford.edu/~wcmac/downloads/fracas.xml`

re-writing rules and annotated with the three RTE labels via a crowdsourcing platform. Both in RTE and SICK the label assigned to the sentence pairs captures the relation holding between the two sentences.

A different approach has been used to build the much larger SNLI (Stanford Natural Language Inference) dataset (Bowman et al., 2015): Premises are taken from a dataset of images annotated with descriptive captions; the corresponding hypotheses were produced through crowdsourcing, where for a given premise, annotators provided a sentence which is true or not true with respect to a possible image which the premise could describe. A consequence of this choice is that the contradiction relation can be assigned to pairs which are rather unrelated ("A person in a black wetsuit is surfing a small wave" and "A woman is trying to sleep on her bed"), differently from what happens in RTE and SICK.

Since the inception of RTE shared tasks, there has been an increasing emphasis on data-driven approaches which, given the hypothesis H and premise P, seek to classify the semantic relation (see (Sammons et al., 2012) for a review). More recently, neural approaches have come to dominate the scene, as shown by the recent RepEval 2017 task (Nangia et al., 2017), where all submissions relied on bidirectional LSTM models, with or without pretrained embeddings. RTE also intersects with a number of related inference problems, including semantic text similarity and Question Answering, and some models have been proposed to address several such problems. In one popular approach, both P and H are encoded within the same embedding space, using a single RNN, with a decision made based on the encodings of the two sentences. This is the approach we adopt for our baseline LSTM in Section 4, based on the model proposed by Bowman et al. (2015), albeit with some modifications (see also (Tan et al., 2016)). A second promising approach, based on which we adapt our state of the art model, relies on matching and aggregation (Wang et al., 2017). Here, the decision concerning the relationship between P and H is based on an aggregate representation achieved after the two sentences are matched. Yet another area where neural approaches are being applied to sentence pairs in an entailment relationship is generation, where an RNN generates an entailed hypothesis (or a chain of such hypotheses) given an encoding of the premise (Kolesnyk et al., 2016; Starc and Mladenić, 2017).

Vision and Textual Entailment In recent years, several models have been proposed to integrate the language and vision modalities; usually the integration is operationalized by element-wise multiplication between linguistic and visual vectors. Though the interest in these modalities has spread in an astonishing way thanks to various multimodal tasks proposed, including the IC, VQA, Visual Reasoning and Visual Dialogue tasks mentioned above, very little work has been done on grounding entailment. Interestingly, Young et al. (2014) has proposed the idea of considering images as the "possible worlds" on which sentences find their denotation. Hence, they released a "visual denotation graph" which associates sentences with their denotation (sets of images). The idea has been further exploited by Lai and Hockenmaier (2017) and Han et al. (2017). Vendrov et al. (2016) look at hypernymy, textual entailment and image captioning as special cases of a single visual-semantic hierarchy over words, sentences and images, and they claim that modelling the partial order structure of this hierarchy in visual and linguistic semantic spaces improves model performance on those three tasks.

We share with this work the idea that the image can be taken as a possible world. However, we don't use sets of images to obtain the visual denotation of text in order to check whether entailment is logically valid/highly likely. Rather, we take the image to be the world/situation in which the text finds its interpretation. The only work that is close to ours is an unpublished student report (Sitzmann et al., 2016), which however lacks the in-depth analysis presented here.

3 Annotated dataset of images and sentence pairs

We took as our starting point the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015), the largest natural language inference dataset available with sentence pairs labelled with entailment, contradiction and neutral relations. We augmented this dataset with images. It has been shown very recently that SNLI contains language bias, such that a simple classifier can achieve high accuracy in predicting the three classes just by having as input the hypothesis sentence. A subset of the SNLI test set with 'hard' cases, where such a simplistic classifier fails (hereafter SNLI$_{hard}$) has been released (Gururangan et al., 2018). Hence, in this paper we will report our results on both the full dataset and the hard

test set, but then zoom in on SNLI$_{hard}$ to understand the models' behaviour. We briefly introduce SNLI and the new test set and compare them through our annotation of linguistic phenomena.

3.1 Dataset construction

SNLI and SNLI$_{hard}$ test set The SNLI dataset (Bowman et al., 2015) was built through Amazon Mechanical Turk. Workers were shown captions of photographs without the photo and were asked to write a new caption that (a) is definitely a true description of the photo (entailment); (b) might be a true description of the photo (neutral); (c) is definitely a false description of the photo (contradiction). Examples were provided for each of the three cases. The premises are captions which come mostly from Flickr30K (Young et al., 2014); only 4K captions are from VisualGenome (Krishna et al., 2017). In total, the dataset contains 570,152 sentence pairs, balanced with respect to the three labels. Around 10% of these data have been validated (4 annotators for each example plus the label assigned through the previous data collection phase). The development and test datasets contain 10K examples each. Moreover, each Image/Flickr caption occurs in only one of the three sets, and all the examples in the development and test sets have been validated.

V-SNLI and V-SNLI$_{hard}$ test set Our grounded version of SNLI, V-SNLI, has been built by matching each sentence pair in SNLI with the corresponding image coming from the Flickr30K dataset; thus the V-SNLI dataset is slightly smaller than the original, which also contains captions from VisualGenome. V-SNLI consists of 565,286 pairs (187,969 neutral, 188,453 contradiction, and 188,864 entailment). Training, test, and development splits have been built according to the splits in SNLI. The main statistics of the splits of the dataset are reported in Table 1 together with statistics for the visual counterpart of Hard SNLI, namely V-SNLI$_{hard}$. By construction, V-SNLI contains datapoints such that the premise is always true with respect to the image, whereas the hypothesis can be either true (entailment or neutral cases) or false (contradiction or neutral cases.)

Split	# entailment	# contradiction	# neutral	# total
V-SNLI train	182,167	181,938	181,515	545,620
V-SNLI dev	3,329	3,278	3,235	9,842
V-SNLI test	3,368	3,237	3,219	9,824
V-SNLI$_{hard}$ test	1,058	1,135	1,068	3,261

Table 1: Statistics of the V-SNLI dataset.

3.2 Dataset annotation

For deeper analysis and comparison of the contents of SNLI and SNLI$_{hard}$, we have annotated the SNLI dataset by both automatically detecting some surface linguistic cues and manually labelling less trivial phenomena. Using an in-house annotation interface, we collected human judgments aiming to (a) filter out those cases for which the gold-standard annotation was considered to be wrong[2]; (b) connect the three ungrounded relations to various linguistic phenomena. To achieve this, we annotated a random sample of the SNLI test set containing 527 sentence pairs (185 entailment, 171 contradiction, 171 neutral), out of which 176 were from the hard test set (56 entailment, 62 contradiction, 58 neutral).

All the pairs were annotated by at least two annotators, as follows: (a) We filtered out all the pairs which had a wrong gold label (see Table 2 for details). When our annotators did not agree whether a given relation holds for a specific pair, we appealed to the corresponding five judgments coming from the validation stage of the SNLI dataset to reach a consensus based on the majority of labels. (b) We considered as valid any linguistic tag assigned by at least one annotator. Since the annotation for (a) is binary whereas for (b) it is multi-class, we used Cohen's κ for the former and also Scott's π and Krippendorff's α for the latter as suggested by Passonneau (2006). The inter-annotator agreement for the relation type (a) was $\kappa = 0.93$; for (b) linguistic tags it was $\pi = 0.63$, $\alpha = 0.61$, and $\kappa = 0.64$[3].

[2] An example of a wrong annotation is the pair *A white greyhound dog wearing a muzzle runs around a track* and *The dog is racing other dogs*, labelled as entailment in the SNLI test set.

[3] Inter-rater agreement was calculated using the NLTK implementation, `http://www.nltk.org`

	Ungrounded			Grounded		
	entailment	contradiction	neutral	entailment	contradiction	neutral
SNLI	7%	16%	2%	1%	1%	31%
SNLI$_{hard}$	6%	10%	1%	<1%	1%	20%

Table 2: Wrong gold-standard labels: Data for which the gold standard label was considered to be wrong (a) in the *ungrounded* setting or (b) correct in the ungrounded setting but not in the *grounded* one. We filter out the data in (a) and keep those in (b).

Tag	Description	Example
Paraphrase	Two-way entailment, i.e., H entails P and vice versa.	P: *A middle eastern marketplace*, H: *A middle eastern store*
Generalisation	One-way entailment, i.e., H entails P but not necessarily vice versa.	P: *A group of people on the beach with cameras*, H: *People are on a beach*.
Entity	P and H describe different entities (e.g., subject, object, location) or incompatible properties of entities (e.g., color).	P: *A dog runs along the ocean surf*, H: *A cat is running in the waves*.
Verb	The sentences describe different, incompatible actions.	P: *Military personnel are shopping*, H: *People in the military are training*.
Insertion	H contains details and facts not present in P (e.g., subjective judgments and emotions.)	P: *Woman reading a book in a laundry room*, H: *The book is old*.
Unrelated	The sentences are completely unrelated.	P: *A woman is applying lip makeup to another woman*, H: *The man is ready to fight*.
Quantifier	The sentences contain numbers or quantifiers (e.g., *all, no, some, both, group*).	P: *A group of people are taking a fun train ride*, H: *People ride the train*.
World knowledge	Commonsense assumptions are needed to understand the relation between sentences (e.g., if there are named entities).	P: *A crowd gathered on either side of a Soap Box Derby*, H: *The people are at the race*.
Voice	The premise is an active/passive transformation of the hypothesis.	P: *Kids being walked by an adult*, H: *An adult is escorting some children*.
Swap	The sentences' subject and object are swapped from P to H.	P: *A woman walks in front of a giant clock*, H: *The clock walks in front of the woman*.

Table 3: Tags used in manual annotation of a subset of the SNLI test set.

Linguistic phenomena Following the error analysis approach described in recent work (Nangia et al., 2017; Williams et al., 2018), we compiled a new list of linguistic features that can be of interest when contrasting SNLI and SNLI$_{hard}$, as well as for evaluating RTE models. Some of these were detected automatically, while others were assigned manually. Automatic tags included SYNONYM and ANTONYM, which were detected using WordNet (Miller, 1995). QUANTIFIER, PRONOUN, DIFF TENSE, SUPERLATIVE and BARE NP were identified using Penn treebank labels (Marcus et al., 1993), while labels such as NEGATION were found with a straightforward keyword search. The tag LONG has been assigned to sentence pairs with a premise containing more than 30 tokens, or a hypothesis with more than 16 tokens. Details about the tags used in the manual annotation are presented in Table 3.

We examined the differences in the tags distributions between the SNLI and SNLI$_{hard}$ test sets (Table 4). Interestingly, the hard sentence pairs from our random sample include proportionately more antonyms but fewer pronouns, as well as examples with different verb tenses in the premise and hypothesis, compared to the full test set. Furthermore, SNLI$_{hard}$ contains a significantly larger proportion of gold-standard labels which become wrong when the image is factored in (χ^2-test with $\alpha = 0.05$).

4 Models

In this section, we describe a variety of models that were compared on both V-SNLI and V-SNLI$_{hard}$, ranging from baseline models based on Bowman et al. (2015) to a state of the art model by Wang et al. (2017). We compare the original 'blind' version of a model with a visually-augmented counterpart. In what follows, we use *P* and *H* to refer to a premise and hypothesis, respectively.

LSTM baseline (Blind) This model exploits a Recurrent Neural Network with Long Short-Term Memory units (Hochreiter and Schmidhuber, 1997) to encode both P and H in 512D vectors. The two vectors

	SNLI		SNLI$_{hard}$			SNLI		SNLI$_{hard}$	
Manual tags	Freq	%	Freq	%	Automatic tags	Freq	%	Freq	%
Insertion	167	32	57	32	DIFF TENSE	7431	76	2384	↓74
Generalisation	163	31	46	26	QUANTIFIER	3779	39	1244	38
Entity	107	20	37	21	PRONOUN	3203	33	979	↓30
Verb	101	19	31	18	SYNONYM	1798	18	605	19
World knowledge	93	18	34	19	ANTONYM	882	9	327	↑10
Quantifier	91	17	23	13	SUPERLATIVE	304	3	106	3
Paraphrase	7	1	4	2	LONG	303	3	109	3
Unrelated	6	1	2	1	BARE NP	281	3	107	3
Voice	3	1	0	0	NEGATION	185	2	55	2
Swap	1	<1	1	1					

Table 4: Distribution of the automatic and manually assigned tags in the SNLI and SNLI$_{hard}$ test sets. Automatic tags are detected in the whole test set, manual ones are assigned to its random subset. Arrows ↑↓ signify a statistically significant difference in tag proportions between the datasets (Pearson's χ^2-test).

are then concatenated in a stack of three 512D layers having a ReLU activation function (Nair and Hinton, 2010), with a final softmax layer to classify the relation between the two sentences as entailment, contradiction or neutral. The model is inspired by the LSTM baseline proposed by Bowman et al. (2015)[4]. The model exploits the 300,000 most frequent pretrained GloVe embeddings (Pennington et al., 2014) and improves them during the training process. To regularize the model, Dropout (Srivastava et al., 2014) is applied to the inputs and outputs of the recurrent layers and to the ReLU fully connected layers with a keeping probability of 0.5. The model is trained using the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.001 until its accuracy on the development set drops for three successive iterations.

V-LSTM baseline The LSTM model described above is augmented with a visual component following a standard Visual Question Answering baseline model (Antol et al., 2015). Following initial representation of P and H in 512D vectors through an LSTM, a fully-connected layer projects the L2-normalized 4096D image vector coming from the penultimate layer of a VGGnet16 Convolutional Neural Network (Simonyan and Zisserman, 2014) to a reduced 512D vector. A fully-connected layer with a ReLU activation function is also applied to P and H to obtain two 512D vectors. The multimodal fusion between the text and the image is obtained by performing an element-wise multiplication between the vector of the text representation and the reduced vector of the image. The multimodal fusion is performed between the image and both the premise and the hypothesis, resulting in two multimodal representations. The relation between them is captured as in the model described above. This model uses GloVe embeddings and the same optimization and procedure described above.

We have also adapted a state-of-the-art attention-based model for IC and VQA (Anderson et al., 2017; Teney et al., 2017) to the GTE task. It obtains results comparable to the V-LSTM. This lack of improvement might be due to the need of further parameter tuning. We report the details of our implementation and its results in the Supplementary Material.

BiMPM The Bilateral Multi-Perspective Matching (BiMPM) model (Wang et al., 2017) obtains state-of-the-art performance on the SNLI dataset, achieving a maximum accuracy of 86.9%, and going up to 88.8% in an ensemble setup. An initial embedding layer vectorises words in P and H using pretrained GLoVe embeddings (Pennington et al., 2014), and passing them to a context representation layer, which uses bidirectional LSTMs (BiLSTMs) to encode context vectors for each time-step. The core part of the model is the subsequent matching layer, where each contextual embedding or time-step of P is matched against all the embeddings of H, and vice versa. The output of this layer is composed of two sequences of matching vectors, which constitute the input to another BiLSTM at the aggregation layer. The vectors from the last time-step of the BiLSTM are concatenated into a fixed-length vector, which is passed to the final prediction tier, a two-layer feed-forward network which classifies the relation between P and H

[4]There are some differences between our baseline and the LSTM baseline used in (Bowman et al., 2015). In particular, we used the Adam optimizer instead of the AdaDelta optimizer, the ReLU activation function instead of the tanh activation function, 512D instead of 100D for the output dimension of LSTM units, and dropout in all fully-connected layers instead of L2 regularization. Our settings outperformed the original ones in our experiments.

	LSTM [H]	LSTM	V-LSTM	BiMPM	V-BiMPM
Entailment	72.65	87.71	87.14	90.03	90.38
Contradiction	66.29	79.7	71.39	86.25	87.53
Neutral	66.36	76.79	68.06	82.79	82.91
Overall	68.49	81.49	75.70	86.41	**86.99**

Table 5: Accuracies (%) for V-SNLI. [H] indicates a baseline model encoding only the hypothesis.

	LSTM [H]	LSTM	V-LSTM	BiMPM	V-BiMPM
Entailment	31.28	72.12	69.09	80.43	81.38
Contradiction	25.29	60.79	46.34	77.62	76.12
Neutral	20.22	50.19	32.02	59.36	63.67
Overall	25.57	60.99	49.03	72.55	**73.75**

Table 6: Accuracies (%) for V-SNLI$_{hard}$. [H] indicates a baseline model encoding only the hypothesis.

via softmax. Matching is performed via a cosine operation, which yields an l-dimensional vector, where l is the number of perspectives. Wang et al. (2017) experiment with four different matching strategies. In their results, the best-performing version of the BiMPM model used all four matching strategies. We adopt this version of the model in what follows.

V-BiMPM model We enhanced BiMPM to account for the image, too. Our version of this model is referred to as the *V*-BiMPM. Here, the feature vector for an image is obtained from the layer before the fully-connected layer of a VGGnet-16. This results in a $7 \times 7 \times 512$ tensor, which we consider as 49 512-dimensional vectors. The same matching operations are performed, except that matching occurs between P, H, and the image. Since the textual and visual vectors have different dimensionality and belong to different spaces, we first map them to a mutual space using an affine transformation. We match textual and image vectors using a cosine operation, as before. Full details of the model are reported in the Supplementary Materials for this paper.

5 Experiments and Results

The models described in the previous sections were evaluated on both (V-)SNLI and (V-)SNLI$_{hard}$. For the visually-augmented models, we experimented with configurations where image vectors were combined with both P and H (namely P+I and H+I), or only with H (P and H+I). The best setting was invariably the one where only H was grounded; hence, we focus on these results in what follows, comparing them to "blind" models. In view of recent results suggesting that biases in SNLI afford a high accuracy in the prediction task with only the hypothesis sentence as input (Gururangan et al., 2018), we also include results for the blind models without the premise (denoted with [H] in what follows).

Table 5 shows the results of the various models on the full V-SNLI dataset. The same models are compared in Table 6 on V-SNLI$_{hard}$. First, note that the LSTM [H] model evinces a drop in performance compared to LSTM (from 81.49% to 68.49%), though the drop is much greater on the unbiased SNLI$_{hard}$ subset (from 60.99 to 25.57%). This confirms the results reported by Gururangan et al. (2018) and justifies our additional focus on this subset of the data.

The effect of grounding in these models is less clear. The LSTM baseline performs worse when it is visually augmented; this is the case of V-SNLI and, even more drastically, V-SNLI$_{hard}$. It is also true irrespective of the relationship type. On the other hand, the V-BiMPM model improves marginally across the board, compared to BiMPM, on the V-SNLI data. On the hard subset, the images appear to hurt performance somewhat in the case of contradiction (from 77.62% to 76.12%), but improve it by a substantial margin on neutral cases (from 59.36% to 63.67%). The neutral case is the hardest for all models, with the possible exception of LSTM [H] on the full dataset.

Overall, the results suggest that factoring in images either hinders performance (as in the case of the V-LSTM baseline), or helps only marginally (as in the case of V-BiMPM). In the latter case, we also observe instances where factoring in images hurts performance. In an effort to understand the results, we

turn to a more detailed error analysis of the V-BiMPM model, first in relation to the dataset annotations, and then by zooming in somewhat closer on V-SNLI$_{hard}$.

5.1 Error analysis by linguistic annotation label

Manual tags	BiMPM	V-BiMPM	Automatic tags	BiMPM	V-BiMPM
Insertion	58	63	ANTONYM	84	84
Generalisation	93	89	BARE NP	79	75
Entity	95	↓78	QUANTIFIER	73	73
Verb	77	68	DIFF TENSE	72	73
World knowledge	79	71	PRONOUN	69	70
Quantifier	78	70	SYNONYM	69	71
Paraphrase	75	75	LONG	67	73
Unrelated	50	50	SUPERLATIVE	64	63
Swap	0	0	NEGATION	51	56

Table 7: Accuracies obtained by BiMPM and V-BiMPM models on SNLI$_{hard}$, by annotation tags. Arrows ↑↓ signify a statistically significant difference in tag proportions between the datasets (Pearson's χ^2-test).

In Table 7, accuracies for the blind and grounded version of BiMPM are broken down by the labels given to the sentence pairs in the annotated subset of SNLI described in Section 3. We only observe a significant difference in the *Entity* case, that is, where the referents in P and H are inconsistent. Here, the blind model outperforms the grounded one, an unexpected result, since one would assume a grounded model to be better equipped to identify mismatched referents. Hence, in the following we aim to understand whether the models properly deal with the grounding sub-task.

5.2 Error analysis on grounding in the SNLI$_{hard}$

We next turn to the "hard" subset of the data, where V-BiMPM showed some improvement over the blind case, but suffered on contradiction cases (Table 6). We analysed the 207 cases in SNLI$_{hard}$ where the V-BiMPM made incorrect predictions compared to the blind model, that is, where the image hurt performance. These were annotated independently by two of the authors (raw inter-annotator agreement: 96%) who (a) read the two sentences, P and H; (b) checked whether the relation annotated in the dataset actually held or whether it was an annotation error; (c) in those cases where it held, checked whether including the image actually resulted in a change in the relation.

Table 8 displays the proportions of image mismatch and incorrect annotations. As the table suggests, in the cases where images hinder performance in the V-BiMPM, it is usually because the image changes the relation (thus, these are cases of image mismatch; see Section 1 for an example); this occurs in a large proportion of cases labelled as neutral in the dataset.

Inspired by the work in (Mironenco et al., 2017), we further explored the impact of visual grounding in both the V-LSTM and V-BiMPM by comparing their performance on SNLI$_{hard}$, with the same subset incorporating image "foils". Vectors for the images in the V-SNLI test set were compared pairwise using cosine, and for each test case in V-SNLI$_{hard}$, the actual image was replaced with the most dissimilar image in the full test set. The rationale is that, if visual grounding is really helpful in recognising the semantic relationship between P and H, we should observe a drop in performance when the images are

Relation	Image mismatch	Incorrect annotation
Entailment	6.82	15.91
Neutral	44.58	1.20
Contradiction	3.80	22.78
Overall	24.76	12.62

Table 8: Cases where images hurt the V-BiMPM's performance: % of cases in which including the image modifies the original SNLI relation (Image mismatch), and % of cases in which the original SNLI relation is incorrectly annotated (Incorrect annotation).

	V-LSTM		V-BiMPM	
	Original	Foil	Original	Foil
Entailment	69.09	65.03	81.38	80.81
Contradiction	46.34	30.92	76.12	74.98
Neutral	32.02	31.46	63.67	63.39
Overal	49.03	46.92 (-2.11)	73.75	73.08 (-0.67)

Table 9: Accuracies of the visually-augmented models on V-SNLI$_{hard}$ with original or foil image.

	V-LSTM	V-BiMPM
Entailment	40.74	51.89
Contradiction	30.22	40.7
Neutral	22.47	32.02
Overall	31.09	41.49

GT class	Prediction	V-LSTM	V-BiMPM
Contradiction	Contradiction	343	462
Contradiction	*Entailment	442	327
Contradiction	Neutral	350	346
Entailment	Entailment	431	549
Entailment	*Contradiction	254	166
Entailment	Neutral	373	343
Neutral	Neutral	240	342
Neutral	Contradiction	377	263
Neutral	Entailment	451	463

Table 10: Confusion matrices for [H+I]. (*) marks implausible errors.

unrelated to the scenario described by the sentences. The results are displayed in Table 9, which also reproduces the original results on V-SNLI$_{hard}$ from Table 6 for ease of reference.

As the results show, models are not hurt by the foil image, contrary to our expectations. V-BiMPM overall drops just by 0.67% whereas V-LSTM drop is somewhat higher (-2.11%) showing it might be doing a better job on the grounding sub-task.

As a final check, we sought to isolate the grounding from the reasoning sub-task, focusing only on the former. We compared the models when grounding only the hypothesis [H+I], while leaving out the premise. Note that this test is different from the evaluation of the model using only the hypothesis [H]: Whereas in that case the input is not expected to provide any useful information to perform the task, here it is. As we noted in Section 3, by construction the premise is always true with respect to the image while the hypothesis can be either true (entailment or neutral cases) or false (contradiction or neutral cases). A model that is grounding the text adequately would be expected to confuse both entailment and contradiction cases with neutral ones; on the other hand, neutral cases should be confused with entailments or contradictions. Confusing contradictions with entailments would be a sign that a model is grounding inadequately, since it is not recognising that H is false with respect to the image.

As the left panel of Table 10 shows, V-BiMPM outperforms V-LSTM by a substantial margin, though the performance of both models drops substantially with this setup. The right panel in the table shows that neither model is free of implausible errors (confusing entailments and contradictions), though V-BiMPM makes substantially fewer of these.

6 Conclusion

This paper has investigated the potential of grounding the textual entailment task in visual data. We argued that a Grounded Textual Entailment model needs to perform two tasks: (a) the grounding itself, and (b) reasoning about the relation between the sentences, against the visual information. Our results suggest that a model based on matching and aggregation like the BiMPM model (Wang et al., 2017) can perform very well at the reasoning task, classifying entailment relations correctly much more frequently than a baseline V-LSTM. On the other hand, it is not clear that grounding is being performed adequately in this model. It is primarily in the case of contradictions that the image seems to play a direct role in biasing the classification towards the right or wrong class, depending on whether the image is correct.

In summary, two conclusions can be drawn from these results. First, in those cases where the inclusion of visual information results in a loss of accuracy, this is often due to the image resulting in a change in the original relation annotated in the dataset. A related observation is that using foil images results in a greater drop in performance on contradiction cases, possibly because in such cases, grounding serves to

identify a mismatch between the hypothesis and the scene described by the premise, a situation which is rendered opaque by the introduction of foils. Second, in those cases where improvements are observed in the state of the art V-BiMPM, the precise role played by the image is not straightforward. Indeed, we find that this model still marginally outperforms the 'blind', text-only model overall, when the images involved are foils rather than actual images.

We believe that further research on grounded TE is worthy of the NLP community's attention. While linking language with perception is currently a topical issue, there has been relatively little work on linking grounding directly with inference. By drawing closer to a joint solution to the grounding and inference tasks, models will also be better able to address language understanding in the real world.

The present paper presented a first step in this direction using a version of an existing TE dataset which was augmented with images that could be paired directly with the premises, since these were originally captions for those images. However, it is important to note that in this dataset premise-hypotheses pairs were not generated directly with reference to the images themselves. An important issue to consider in future work on GTE, besides the development of better models, is the development of datasets in which the role of perceptual information is controlled, ensuring that the data on which models are trained represents truly grounded inferences.

Acknowledgements

We kindly acknowledge the European Network on Integrating Vision and Language (iV&L Net) ICT COST Action IC1307. Moreover, we thank the Erasmus Mundus European Program in Language and Communication Technology. Marc Tanti's work is partially funded by the Endeavour Scholarship Scheme (Malta), part-financed by the European Union's European Social Fund (ESF). Finally, we gratefully acknowledge the support of NVIDIA Corporation with the donations to the University of Trento of the GPUs used in our research.

Appendix A: Bottom-up top-down attention (VQA)

We adapted the Visual Question Answering model proposed in (Anderson et al., 2017; Teney et al., 2017) to the Grounded Textual Entailment task. The model presents a more fine-grained attention mechanism which allows to identify the most important regions discovered in the image and to perform attention over each of them.

The model uses a a Recurrent Neural Network with Long Short-Term Memory units to encode the premise P and hypothesis H in 512D vectors. A bottom-up attention mechanism exploits a Fast R-CNN (Girshick, 2015) based on a ResNet-101 convolutional neural network (He et al., 2016) to obtain region proposals corresponding to the 36 most informative regions of the image. A top-down attention mechanism is used between the premise (resp. hypothesis) and each of the L2-normalized 2048D image vectors corresponding to the region proposals to obtain an attention score for each of them. Then, a 2048D image vector encoding the most interesting visual features for the premise (hypothesis) is obtained as a sum of the 36 image vectors weighted by the corresponding attention scores for the premise (hypothesis). A fully-connected layer with a gated tanh activation function is applied to the image vector of the most interesting visual features for the premise and for the hypothesis to obtain a reduced 512D vector for each of them. A fully-connected layer with a gated tanh activation function is also applied to the premise and to the hypothesis in order to obtain a reduced 512D vector for each of them.

The multimodal fusion between the premise (hypothesis) and the image vector of the most interesting visual features for the premise (hypothesis) is obtained by performing an element-wise multiplication between the reduced vector of the premise (hypothesis) and the reduced vector of the most interesting visual features for the premise (hypothesis). After that, the model feeds the concatenation of the two resulting multimodal representations to a stack of three 512D layers having a gated tanh activation function, with a final softmax layer to classify the relation between the two sentences as entailment, contradiction or neutral. This model uses GloVe embeddings and the same optimization tricks and procedure of the LSTM and V-LSTM models.

We report the accuracies of the VQA models against the various tests reported in the paper. For ease of comparison we reproduce the full table from the main paper, with the addition of the VQA results.

	LSTM [H]	LSTM	V-LSTM	VQA	BiMPM	V-BiMPM
Entailment	72.65	87.71	87.14	86.1	90.03	90.38
Contradiction	66.29	79.7	71.39	78.99	86.25	87.53
Neutral	66.36	76.79	68.06	73.56	82.79	82.91
Overall	68.49	81.49	75.70	79.65	86.41	**86.99**

Table 11: Accuracies (%) for V-SNLI. [H] indicates a baseline model encoding only the hypothesis (Table 5 in the paper).

	LSTM [H]	LSTM	V-LSTM	VQA	BiMPM	V-BiMPM
Entailment	31.28	72.12	69.09	67.39	80.43	81.38
Contradiction	25.29	60.79	46.34	59.03	77.62	76.12
Neutral	20.22	50.19	32.02	42.13	**59.36**	**63.67**
Overall	25.57	60.99	49.03	56.21	72.55	73.75

Table 12: Accuracies (%) for V-SNLI$_{hard}$. [H] indicates a baseline model encoding only the hypothesis (Table 6 in the paper).

	V-LSTM		V-BiMPM		VQA	
	Original	Foil	Original	Foil	Original	Foil
Entailment	69.09	65.03	81.38	80.81	67.39	60.4
Contradiction	46.34	30.92	76.12	74.98	59.03	60.97
Neutral	32.02	31.46	63.67	63.39	42.13	42.79
Overal	49.03	46.92 (-2.11)	73.75	73.08 (-0.03)	56.21	54.83 (-1.38)

Table 13: Accuracies of the visually augmented models on V-SNLI$_{hard}$ containing the original or foil image (Table 9 in the paper).

	V-LSTM	V-BiMPM	VQA
Entailment	40.74	51.89	48.11
Contradiction	30.22	40.7	47.05
Neutral	22.47	32.02	31.37
Overall	31.09	41.49	42.26

GT class	Prediction	V-LSTM	V-BiMPM	VQA
Contradiction	Contradiction	343	462	534
Contradiction	*Entailment	442	327	286
Contradiction	Neutral	350	346	315
Entailment	Entailment	431	549	509
Entailment	*Contradiction	254	166	188
Entailment	Neutral	373	343	361
Neutral	Neutral	240	342	335
Neutral	Contradiction	377	263	272
Neutral	Entailment	451	463	461

Table 14: Confusion matrices for [H+I]. (*) marks implausible errors (Table 10 in the paper).

Appendix B: V-biMPM Model details

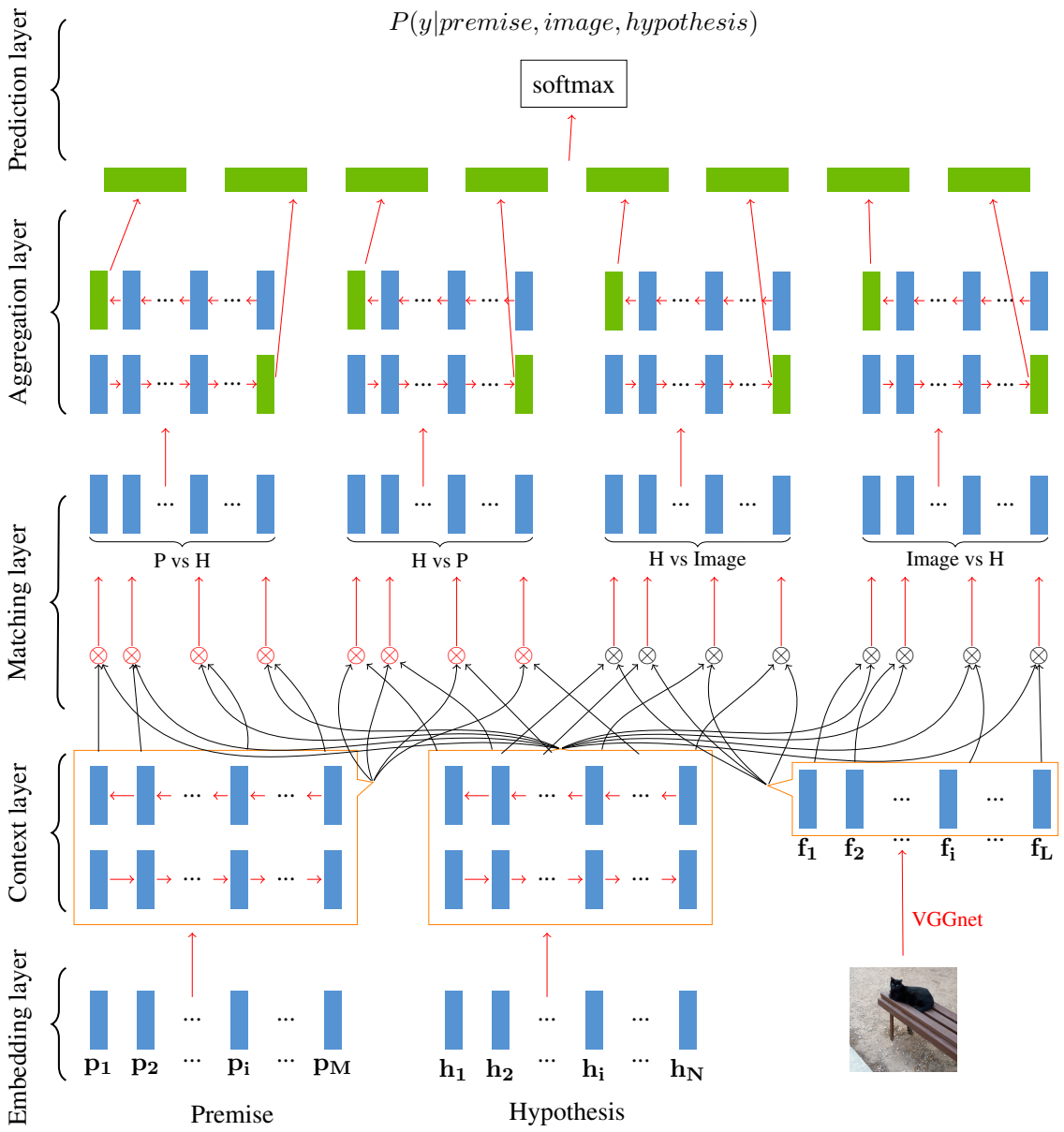


Figure 2: Vision-enhanced Bilateral Multi-Perspective Matching model (v-BiMPM).

Here, we report some further details of our implementation of the V-BiMPM model described in Section 4 of the main paper, based on the work of Wang et al. (2017). Our model is displayed in Figure 2.

The core part of the original BiMPM is the matching layer. Given two d-dimensional vectors v_P and v_H, each replicated l times (l is the number of 'perspectives') and a trainable $l \times d$ weight matrix W, matching involves a cosine similarity computation that yields an l-dimensional matching vector m, whose elements are defined as follows:

$$m_k = \textit{cosine}\left(W_k \circ v_P, W_k \circ v_H\right) \tag{1}$$

The matching operations included are the following:

1. *full-matching*, where each forward or backward contextual embedding of the premise P (resp. the hypothesis H) is matched to the last time-step of H (resp. P);

2. *max-pooling*, where each forward/backward contextual embedding of one sentence is compared to the embeddings of the other, retaining the maximum value for each dimension;

3. *attentive matching*, where first, the pairwise cosine similarity between forward/backward embeddings of P and H is estimated, before calculating an attentive vector over the weighted sum of contextual embeddings for H and matching each forward/backward embedding of P against the attentive vector;

4. *max-attentive matching*, a version of attentive matching where the contextual embedding with the highest cosine is used as the attentive vector, instead of the weighted sum.

The visually-augmented version of the original model, V-BiMPM, is displayed in Figure 2. To perform multimodal matching, the visual and textual vectors are mapped to a mutual space using the following affine transformation:

$$v_i = \mathbf{W}_t f_i + b_t; f_i \in \mathbb{R}^e; \mathbf{W}_t \in \mathbb{R}^{e \times d}; b_t, v_i \in \mathbb{R}^d \tag{2}$$

where $\mathbf{W}_t$, b_t, f_i, and v_i are the weight matrix, the bias, the input features and output features, respectively, and t is any text (P or H). Given weight matrices $\mathbf{W} \in \mathbb{R}^{l \times d}$ for text and $\mathbf{U}^{l \times d}$ for images, we compute the matching vector m between a textual vector v_t and image vector v_i as:

$$m_k = \textit{cosine}\left(W_k \circ v_t, U_k \circ v_i\right) \tag{3}$$

References

Eneko Agirre, Oier Lopez de Lacalle, and Aitor Soroa. 2017. Evaluating Multimodal Representations on Sentence Similarity : vSTS , Visual Semantic Textual Similarity Dataset. In *Proceedings of the Second Workshop on Closing the Loop Between Vision and Language (ICCV'17)*.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2017. Bottom-up and top-down attention for image captioning and visual question answering. *arXiv preprint arXiv:1707.07998*.

Mark Andrews, Gabriella Vigliocco, and David Vinson. 2009. Integrating experiential and distributional data to learn semantic representations. *Psychological Review*, 116(3):463–498.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual question answering. In *International Conference on Computer Vision (ICCV)*.

Lawrence W. Barsalou. 2010. Grounded Cognition: Past, Present, and Future. *Topics in Cognitive Science*, 2(4):716–724.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, and Steve Pulman. 1996. Using the framework. Technical Report Technical Report LRE 62-051 D-16, The FraCaS Consortium.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M.F. Moura, Devi Parikh, and Dhruv Batra. 2017. Visual Dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

D. Giampiccolo, H. T. Dang, M. Bernardo, I. Dagan, and E. Cabrio. 2008. The fourth pascal recognising textual entailment challenge. In *Proceedings of the TAC 2008 Workshop on Textual Entailment*.

Ross Girshick. 2015. Fast r-cnn. *arXiv preprint arXiv:1504.08083*.

Goran Glavaš, Ivan Vuli, and Simone Paolo Ponzetto. 2017. If Sentences Could See: Investigating Visual Information for Semantic Textual Similarity. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS'17)*, pages 1–15.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation Artifacts in Natural Language Inference Data. *arXiv preprint arXiv:1803.02324*.

Dan Han, Pascual Martinez Gomez, and Koji Mineshima. 2017. Visual denotations for recognizing textual entailment. In *EMNLP*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of CVPR 2017*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Vladyslav Kolesnyk, Tim Rocktäschel, and Sebastian Riedel. 2016. Generating Natural Language Inference Chains. *arXiv preprint arXiv:1606.01404*.

Ranjay Krishna, Yuke Zhu, Oliver GrothJustin, Johnson Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73.

Alice Lai and Julia Hockenmaier. 2017. Learning to predict denotational probabilities for modeling entailment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 721–730, Valencia, Spain, April. Association for Computational Linguistics.

M. Malinowski and M. Fritz. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in Neural Information Processing Systems*.

Mitchell P Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC 2014*, pages 216–223. ELRA.

George A. Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

M. Mironenco, D. Kianfar, K. Tran, E. Kanoulas, and E. Gavves. 2017. Examining cooperation in visual dialog models. In *Neural Informations Processing Systems, Workshop on Visually-Grounded Interaction and Language*.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Nikita Nangia, Adina Williams, Angeliki Lazaridou, and Samuel R Bowman. 2017. The RepEval 2017 Shared Task: Multi-Genre Natural Language Inference with Sentence Representations. *arXiv preprint arXiv:1707.08172*.

R. Passonneau, N. Habash, and O. Rambow. 2006. Inter-annotator agreement on a multilingual semantic annotation task. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 1951–1956.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Mark Sammons, Vinod Vydiswaran, and Dan Roth. 2012. Recognising Textual Entailment. In Daniel M. Bikel Zitouni and Imed, editors, *Multilingual Natural Language Processing*, pages 209–281. IBM Press, Westford, MA.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Vincent Sitzmann, Martina Marek, and Leonid Keselman. 2016. Multimodal natural language inference. Technical report, Standford.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Janez Starc and Dunja Mladenić. 2017. Constructing a Natural Language Inference dataset using generative neural networks. *Computer Speech and Language*, 46:94–112.

Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. 2017. A corpus of natural language for visual reasoning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 217–223, Vancouver, Canada, July. Association for Computational Linguistics.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. LSTM-based Deep Learning Models for Non-factoid Answer Selection. *arXiv preprint arXiv:1511.04108*, pages 1–11.

Damien Teney, Peter Anderson, Xiaodong He, and Anton van den Hengel. 2017. Tips and tricks for visual question answering: Learnings from the 2017 challenge. *arXiv preprint arXiv:1708.02711*.

Ivan Vendrov, Ryan Kiors, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *Proceedings of the International Conference of Learning Representations (ICLR)*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral Multi-Perspective Matching for Natural Language Sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI'17)*, pages 4144–4150, Melbourne.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of NAACL*.

K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Recurrent One-Hop Predictions for Reasoning over Knowledge Graphs

Wenpeng Yin[1], **Yadollah Yaghoobzadeh**[2], **Hinrich Schütze**[3]
[1]Dept. of Computer Science, University of Pennsylvania, USA
[2] Microsoft Research, Montreal, Canada
[3]CIS, LMU Munich, Germany
wenpeng@seas.upenn.edu

Abstract

Large scale knowledge graphs (KGs) such as Freebase are generally incomplete. Reasoning over multi-hop (mh) KG paths is thus an important capability that is needed for question answering or other NLP tasks that require knowledge about the world. mh-KG reasoning includes diverse scenarios, e.g., given a head entity and a relation path, predict the tail entity; or given two entities connected by some relation paths, predict the unknown relation between them. We present *ROPs*, recurrent one-hop predictors, that predict entities at each step of mh-KB paths by using recurrent neural networks and vector representations of entities and relations, with two benefits: (i) modeling mh-paths of arbitrary lengths while updating the entity and relation representations by the training signal at each step; (ii) handling different types of mh-KG reasoning in a unified framework. Our models show state-of-the-art for two important multi-hop KG reasoning tasks: Knowledge Base Completion and Path Query Answering.[1]

1 Introduction

Natural language understanding (NLU) is impossible without knowledge about the world. Large scale knowledge graphs (KGs) such as Freebase (Bollacker et al., 2008) are structures that store world knowledge. Unfortunately, KGs suffer from incomplete coverage (Min et al., 2013); e.g., Freebase contains Brandon Lee, but not his ethnicity.

The knowledge in KGs needs to be expanded to cover more facts; reasoning is one way to do so. For example, we could infer that *(Microsoft, ?, United States)* instantiates "CountryOfHQ" given the facts *(Microsoft, IsBasedIn, Seattle)* and *(Seattle, LocatedIn, United States)*; or we could infer Brandon Lee's ethnicity from his parents' ethnicity, i.e., answering the query *(Brandon Lee, Ethnicity, ?)* by facts *(Brandon Lee, Father, Bruce Lee)* and *(Bruce Lee, Ethnicity, Chinese)*. We refer to the two reasoning examples as "knowledge base completion (KBC)" and "path query answering (PQA)", respectively.

The most successful approach for modeling KGs is the *embedding approach*. It embeds KG elements (entities and relations) into low-dimensional dense vectors; controlling the dimensionality of the vector space forces generalization to new facts (Nickel et al., 2011).

In this work, we are mainly interested in three issues. (i) Compared to modeling one-hop KG paths, a bigger challenge is how to model multi-hop paths, e.g., the path query *(U.S.A, president→spouse→born_in, ?)* for the question "Where was the first lady of the United States born?" (ii) How can we address different KG reasoning problems driven by multi-hop paths in a universal paradigm rather than via different systems? (iii) How can we combine specific multi-hop KG reasoning tasks with generic KG representation learning, so that KG representation learning can either stand alone or be incorporated into diverse multi-hop KG-related NLU problems. We get inspiration from following two types of work.

First, *prior work in KG reasoning*. Guu et al. (2015) extend one-hop reasoning regimes such as TransE (Bordes et al., 2013) to multi-hop PQA. However, these basic one-hop models do not encode the relation

[1]https://github.com/yinwenpeng/KBPath

Proceedings of the 27th International Conference on Computational Linguistics, pages 2369–2378
Santa Fe, New Mexico, USA, August 20-26, 2018.

order when used in compositional training schemes. For example, path query q_1 = (h, $r_1 \rightarrow r_2 \rightarrow \cdots \rightarrow r_k$, ?) will be encoded into the same embedding as path query q_2 = (h, $r_2 \rightarrow r_1 \rightarrow \cdots \rightarrow r_k$, ?), resulting in (often incorrect) prediction of the same tail entity. Instead, the relation order should influence the prediction. This limitation in modeling multi-hop relation paths motivates the *RNN approach*: using recurrent neural networks (RNN (Elman, 1990)) to model relation paths (Neelakantan et al., 2015). Das et al. (2017) further extend this approach by incorporating entity information and apply it to multi-hop KBC. Intermediate entities should influence the reasoning decision. For example, given two paths with the same relation sequence: (Donald Trump, *child*, Ivanka Trump, *mother*, Ivana Trump) and (Donald Trump, *child*, Barron Trump, *mother*, Melania Trump), even though both paths have the relation sequence [$child$, $mother$], the relation between (Donald Trump, Melania Trump) is "spouse" while it does not hold between (Donald Trump, Ivana Trump) due to the intermediate entities: "Ivanka Trump" vs. "Barron Trump". Similarly, paths (JFK, $located\_in$, NYC, $located\_in$, NY) and (Yankee Stadium, $located\_in$, NYC, $located\_in$, NY) would predict the same score for target relation "airport_serves_place" if we do not consider that Yankee Stadium is not an airport (Das et al., 2017).

Second, *sequence labeling tasks* such as POS tagging, chunking and NER have been successfully addressed by RNNs (Huang et al., 2015; Lample et al., 2016). These approaches model the mechanism in a structure of form "input$_1$, tag$_1$, input$_2$, tag$_2$, input$_3$, tag$_3$, $\cdots$, input$_t$, tag$_t$", which resembles the structure of multi-hop KG paths.

Inspired by this prior work, we propose *ROP*, Recurrent One-hop Predictor. Given a head entity, ROP encodes a multi-hop sequence of relations and predicts a sequence of entities using an RNN. More formally, given relation sequence "$r_1, r_2, \cdots, r_t$" and the head entity e_h, ROP predicts the sequence $e_1, e_2, \cdots, e_t$, thus generating a complete KG path $e_h, r_1, e_1, r_2, e_2, \cdots, r_t, e_t$. Intuitively, our model memorizes history of path context; given a new relation, it predicts the next entity, then the memory is updated, and the process – given new relation, predicting new entity, updating memory – keeps going. Grouping step-wise updates in a chain gives our model two advantages. (i) A better vector space representation of KG entities and relations, with training signals either from in-path entities or from labels of reasoning tasks or from both. (ii) A unified approach for two different reasoning tasks (mh-KBC and mh-PQA) and state-of-the-art in each.

In summary, our contributions are: (i) ROP, a novel RNN sequence modeling of multi-hop KG paths that updates entity and relation embeddings by training signal at each step and leads to better KG embeddings; (ii) unified framework for solving different multi-hop reasoning tasks over KGs; (iii) showing the importance of modeling within-path entities in mh-PQA; (iv) state-of-the-art results for both mh-KBC and mh-PQA; (v) releasing an enhanced version of a mh-PQA dataset by adding within-path entities.

§2 introduces the mh-KBC and mh-PQA tasks. §3 discusses related work. §4 presents three ROP architectures and §5 evaluates them. §6 concludes.

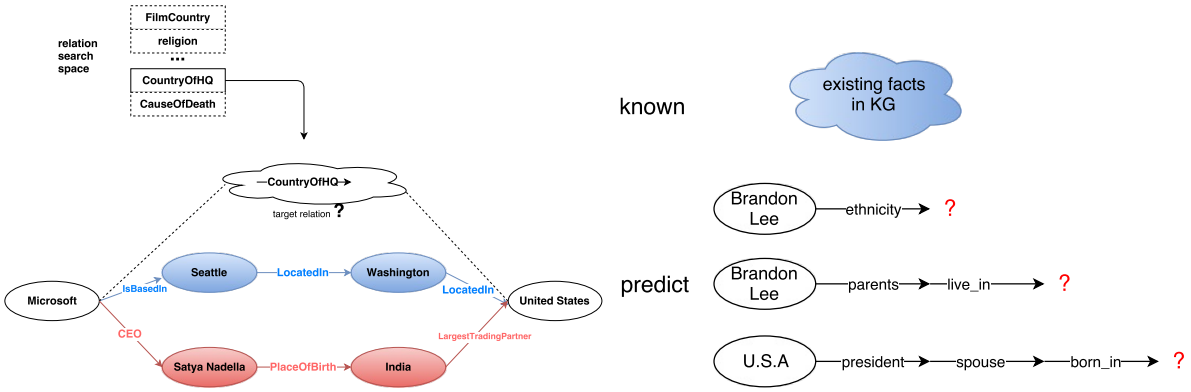
2 Multi-Hop Path Reasoning Tasks

We first give background on the two KG reasoning tasks we address in this work.

Knowledge Base Completion (mh-KBC). In mh-KBC, the goal is to predict new relations between entities using existing path connections. For example, *(A, LivesIn, B)* is implied, with some probability, from *(A, CEO, X)* and *(X, HQIn, B)*. This kind of reasoning lets us infer new or missing facts from KGs.

Figure 1(a) shows two paths between *Microsoft* and *United States*: *(Microsoft, IsBasedIn, Seattle, IsLocatedIn, Washington, IsLocatedIn, United States)* (blue) and *(Microsoft, CEO, Satya Nadella, PlaceOfBirth, India, LargestTradingPartner, United States)* (red). The task is then to predict the direct relation that connects *Microsoft* and *United States*; i.e., *CountryOfHQ* in this case. There can exist multiple long paths between two entities; the example shows that the target relation may only be inferrable from one path. The difficulty of finding the most informative path makes this task challenging.

Path Query Answering (mh-PQA). In mh-PQA, the goal is to *predict missing properties* of an entity, such as the earlier mentioned ethnicity of Brandon Lee. More generally, given an entity and relations of interest, predict what the target entity is, as Figure 1(b) shows. This task corresponds to answering



(a) Multi-hop knowledge base completion (b) Multi-hop path query answering

Figure 1: Multi-Hop Path Reasoning Tasks

compositional natural questions. For example, the question "Where do Brandon Lee's parents live?" can be formulated by the path query brandon_lee/parents/live_in. mh-PQA tries to find answers to the path queries and hence compositional questions. Unfortunately, KGs often have missing facts (edges), which makes mh-PQA a non-trivial problem.

A path query q_t consists of an initial anchor entity, e_h, followed by a sequence of t relations to be traversed, $p = (r_1, \cdots, r_t)$. Following (Guu et al., 2015), the answer or denotation of the query is the set of all entities that can be reached from e_h by traversing p.

3 Related Work

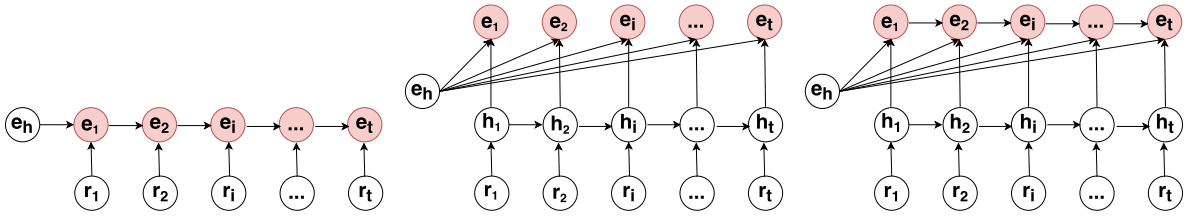
Here we focus on the multi-hop path reasoning literature. Some work (Neelakantan et al., 2015; Guu et al., 2015; Lin et al., 2015; Lin et al., 2016; Shen et al., 2016) does some composition over relation paths. Given relation path $p = (r_1, \cdots, r_t)$, the composition operation is *add* ($\mathbf{p} = \mathbf{r}_1 + \cdots + \mathbf{r}_t$), *multiplication* ($\mathbf{p} = \mathbf{r}_1 \cdots \mathbf{r}_t$) or an *RNN step*: $\mathbf{p}_i = \text{RNN}(\mathbf{p}_{i-1}, \mathbf{r}_i)$, where $\mathbf{p}_i$ is the accumulated relation information up to step i. Some work explores compositional encoding of long paths (Lin et al., 2016; Shen et al., 2016), but still performs reasoning in one-hop scenario. Neelakantan et al. (2015) use RNNs to model multi-hop paths.

Das et al. (2017) extend the RNN approaches by leveraging within-path entities into the encoding of inputs along with relations. We also include within-path entities, but we do not give them as inputs; instead, *we force our RNN to predict them as outputs and do updates at each step in the path*. This supports representation learning for KG entities and relations even without task-specific annotations. Toutanova et al. (2016) propose a dynamic programming algorithm to model both relation types and intermediate entities in the compositional path representations and test on WordNet and a biomedical KG. These two works address mh-KBC; for mh-PQA, there is no prior work on using within-path entities[2], including Das et al. (2017), in which the system uses within-path entities as input, while those entities are not available for testing. So Das et al. (2017) use RNN for mh-PQA to encode the relation sequence, but it does not incorporate the intermediate entities involved.

4 Recurrent One-Hop Prediction

We propose three ROPs, recurrent one-hop predictors, to model paths such as $p = (e_h, r_1, e_1, \cdots, r_i, e_i, \cdots, r_t, e_t)$. Entities and relations appear alternately in p; e_h and e_t are head and tail entities; t steps (hops) connect e_h and e_t; each step is a single fact triple (e_{i-1}, r_i, e_i). We stipulate $\mathbf{e}_i \in \mathbb{R}^{d_e}$ and $\mathbf{r}_i \in \mathbb{R}^{d_r}$. We allow $d_e \neq d_r$.

[2]Guu et al. (2015) attempt to measure how severe the cascading errors along the path are by reconstructing the intermediate entities along a path. Their operation only generates an evaluation score for the path representation. We instead turn this into a training objective to fine-tune the path representations.



(a) ROP_ARC1 (b) ROP_ARC2 (c) ROP_ARC3

Figure 2: Three recurrent one-hop predictors

Paths are encoded by GRUs (Cho et al., 2014):

$$\mathbf{g}_z = \sigma(\mathbf{x}_i\mathbf{U}^z + \mathbf{h}_{i-1}\mathbf{W}^z) \quad (1)$$
$$\mathbf{g}_r = \sigma(\mathbf{x}_i\mathbf{U}^r + \mathbf{h}_{i-1}\mathbf{W}^r) \quad (2)$$
$$\hat{\mathbf{h}}_i = \tanh(\mathbf{x}_i\mathbf{U}^q + (\mathbf{h}_{i-1} \circ \mathbf{g}_r)\mathbf{W}^q) \quad (3)$$
$$\mathbf{h}_i = (1 - \mathbf{g}_z) \circ \hat{\mathbf{h}}_i + \mathbf{g}_z \circ \mathbf{h}_{i-1} \quad (4)$$

where x is the input sequence with x_i at position i, h is the output sequence with h_i at position i. g_z and g_r are gates. All $\mathbf{U}$s and $\mathbf{W}$s are parameters. In following, *we define the whole Eqs. 1–4 as a single GRU step as*:

$$h_i = \text{GRU}(h_{i-1}, x_i) \quad (5)$$

Thus, we interpret each GRU step as a composition function of two objects: h_{i-1} and x_i.

We now introduce the three architectures ROP_ARC1, ROP_ARC2 and ROP_ARC3 that encode the context "$e_h, r_1, e_1, \cdots, r_i$" in different ways to predict entity e_i.

ROP_ARC1 (Figure 2(a)) models KG paths as:

$$\hat{\mathbf{e}}_0 = \mathbf{e}_h \quad (6)$$
$$\hat{\mathbf{e}}_i = \text{GRU}(\hat{\mathbf{e}}_{i-1}, \mathbf{r}_i) \quad (7)$$

$\hat{\mathbf{e}}_0$ is initialized to the embedding of the *true* head entity $\mathbf{e}_h$. At position $i, i > 0$, $\hat{\mathbf{e}}_i$ is the *predicted* entity embedding.

ROP_ARC1 is essentially a recurrent process with a pre-set starting state; the key is to use the head entity embedding $\mathbf{e}_h$ as the initialization of the hidden state of GRU. We hope this start point guides where the path goes and what state to reach at each position. As a result, *relations lie in the input space and entities lie in the hidden space*. All *predicted* entities $\hat{\mathbf{e}}_i$ will be compared with the gold intermediate entities $\mathbf{e}_i$, then the loss (red in Figure 2) is used to train the system.

ROP_ARC1 only encodes head entity e_h at the starting hidden state, possibly far from the tail entity e_t for long paths. Thus, e_h cannot provide effective guidance for prediction of e_t. This motivates ROP_ARC2, a modification of ROP_ARC1.

In **ROP_ARC2**, we want the head entity e_h to participate in the entity prediction at each step more directly and effectively. To this end, ROP_ARC2 first encodes the relation sequence as standard GRU:

$$\begin{aligned} \mathbf{h}_0 &= \mathbf{0} \\ \mathbf{h}_i &= \text{GRU}(\mathbf{h}_{i-1}, \mathbf{r}_i) \end{aligned} \quad (8)$$

$\mathbf{h}_0$ is initialized to $\mathbf{0}$. $\mathbf{h}_i$, $i > 0$, contains the information of the relation sequence $r_1, r_2, \cdots, r_i$ independent of the head entity e_h. To make sure the path reaches the correct state, ROP_arc2 then composes each h_i with e_h to predict e_i as:

$$\hat{\mathbf{e}}_i = \text{COMP}_1(\mathbf{e}_h, \mathbf{h}_i) \quad (9)$$

As composition function COMP_1 we use ADD (addition, as in TransE) or GRU (Eq 5).

In ROP_ARC2, head entity e_h directly participates in entity prediction at each step. Unfortunately, there is often another issue – head entity e_h may not match the hidden state h_i if they are far away from each other – h_i mainly encodes some latest relation inputs that are less related to the head entity. Besides, there is a second information source, in addition to e_h, that is clearly relevant for accurate prediction: the preceding predicted entity $\hat{e}_{i-1}$. But ROP_ARC2 does not make it available. Our solution is ROP_ARC3. This architecture predicts the next entity e_i using both e_h and $\hat{e}_{i-1}$, based on the intuition that e_i is directly related to its predecessor e_{i-1}.

ROP_ARC3 combines the benefits of ROP_ARC1 and ROP_ARC2. It encodes the relation sequence as in Eq 8 in ROP_ARC2, but composes head entity e_h as well as the *predicted* entity $\hat{e}_{i-1}$ with h_i to predict the entity e_i as:

$$\hat{\mathbf{e}}_i = \text{COMP}_2(\mathbf{e}_h, \hat{\mathbf{e}}_{i-1}, \mathbf{h}_i) \tag{10}$$

As composition function COMP_2, we use ADD (addition) or an extended GRU step, defined as:

$$\begin{aligned}
\mathbf{g}_{zh} &= \sigma(\mathbf{h}_i\mathbf{U}^{zh} + \mathbf{e}_h\mathbf{W}^{zh}) && (11)\\
\mathbf{g}_{rh} &= \sigma(\mathbf{h}_i\mathbf{U}^{rh} + \mathbf{e}_h\mathbf{W}^{rh}) && (12)\\
\mathbf{g}_{zp} &= \sigma(\mathbf{h}_i\mathbf{U}^{zp} + \hat{\mathbf{e}}_{i-1}\mathbf{W}^{zp}) && (13)\\
\mathbf{g}_{rp} &= \sigma(\mathbf{h}_i\mathbf{U}^{rp} + \hat{\mathbf{e}}_{i-1}\mathbf{W}^{rp}) && (14)\\
\hat{\mathbf{h}}_i &= \tanh(\mathbf{h}_i\mathbf{U}^{q} + (\mathbf{e}_h \circ \mathbf{g}_{rh})\mathbf{W}^{q_h} + (\hat{\mathbf{e}}_{i-1} \circ \mathbf{g}_{rp})\mathbf{W}^{q_p}) && (15)\\
\hat{\mathbf{e}}_i &= (1 - \mathbf{g}_{zh} - \mathbf{g}_{zp}) \circ \hat{\mathbf{h}}_i + \mathbf{g}_{zh} \circ \mathbf{e}_h + \mathbf{g}_{zp} \circ \hat{\mathbf{e}}_{i-1} && (16)
\end{aligned}$$

where super/subscript h refers to *head* entity and p refers to *prior* predicted entity $\hat{e}_{i-1}$.

In following work, we define Eqs. 11–16 as eGRU (extended GRU) step as:

$$\hat{\mathbf{e}}_i = \text{eGRU}(\mathbf{e}_h, \hat{\mathbf{e}}_{i-1}, \mathbf{h}_i) \tag{17}$$

We extend GRU into eGRU, so that one architecture can compose three objects: a hidden state h_i and two entity states e_h and $\hat{e}_{i-1}$.

Training. For the gold entity sequence $e_1, e_2, \cdots, e_t$, we define the **loss function** as the margin-based ranking criterion used in TransE (Bordes et al., 2013):

$$l_{\text{seq}} = \textstyle\sum_i \max(0, \alpha + \text{s}(\mathbf{e}_i^-, \hat{\mathbf{e}}_i) - \text{s}(\mathbf{e}_i, \hat{\mathbf{e}}_i)) \tag{18}$$

where α is the margin, e_i^- a negative sample for entity e_i and $\text{s}()$ a similarity function.

Discussion. The three ROP models *differ* in three aspects.

(i) ROP_ARC1 has only one composition process, i.e., GRU, to encode from head entity e_h to r_i; ROP_ARC2 and ROP_ARC3 each have two composition processes, one composes relation sequence $r_1, \cdots, r_i$ into hidden state $\mathbf{h}_i$, the other composes entities (e_h in ROP_ARC2, e_h and $\hat{e}_{i-1}$ in ROP_ARC3) with $\mathbf{h}_i$ to predict e_i.

Figure 2 shows that ROP_ARC1 uses the GRU hidden states as outputs to compare with gold entities. ROP_ARC2 and ROP_ARC3 instead compose their hidden states with head entity or preceding predicted entity to generate a new output space, then compare with gold entities.

(ii) ROP_ARC2 only uses the head entity e_h along with the current hidden state $\mathbf{h}_i$ to predict the next entity e_i whereas ROP_ARC1 and ROP_ARC3 in addition use the preceding predicted entity $\hat{e}_{i-1}$. Hence, ROP_ARC3 roughly uses the combined information of ROP_ARC1 and ROP_ARC2 to do the prediction.

(iii) GRUs like ROP_ARC2&ARC3 often zero-initialize first hidden state $\mathbf{h}_0$. Setting $\mathbf{h}_0$ to the head entity in ROP_ARC1 supports prediction of subsequent entities.

The ROP models are *similar* in three aspects.

(i) They model entities and relations in different spaces: relations are in the input space, entities are in hidden or output spaces. Our motivation is similar to SE, TransH and TransR (cf. §3).

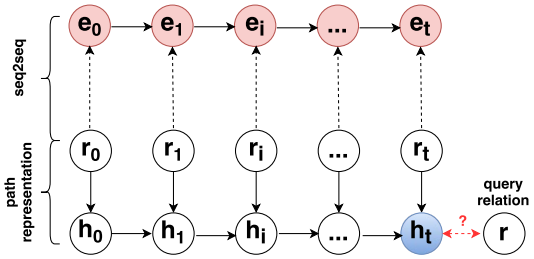


Figure 3: Architecture for mh-KBC task

(ii) The gating mechanism enables flexible compositions between entities and relations based on path context. In contrast, *one-hop KG embedding approaches model compositions statically as shared parameters and consider no context.*

(iii) Unlike Neelakantan et al. (2015), Guu et al. (2015) and Lin et al. (2015) (who also do some composition over relation paths), we finetune entity/relation embeddings in each step of the path. Our intuition is that many more training signals coming from each step of the sequence enable better learning of entity/relation embeddings. Das et al. (2017), as prior work that incorporates entities in the paths, only update the entity embeddings in the path once, when reaching the end of the path. We test this effect in our experiments.

5 Experiments

5.1 Knowledge Base Completion (mh-KBC)

Dataset. We use Das et al. (2017)'s dataset, in which there are 46 query relations, each has *train, dev and test* files containing positive and negative entity pairs (head entity, tail entity). Each entity pair is also provided with multiple multi-hop paths connecting them.

This is a binary classification task for each query relation: does the query relation hold between a pair of entities? The classifier builds a ranked list of entity pairs for corresponding query relation. Evaluation: mean average precision (MAP) across all 46 relations.

Task setup. We use ROP for mh-KBC. Figure 3 shows that we extend the basic ROP architecture (§4) by a GRU layer (bottom layer in Figure 3), denoted as GRU_r, that learns the representation of the relation sequence. Finally, we use $\mathbf{h}_t$ (the last hidden state of GRU_r, shown in blue in Figure 3) to match the query relation $\mathbf{r}$.[3]

Thus, the training loss of this task has two parts: one comes from the loss of ROP training (l_{seq}, Eq 18); the other is the prediction loss of query relation, denoted as l_{pred}. Our preliminary experiments always showed worse performance of ADD, so the first loss l_{seq} here only considers (e)GRU. The second loss is as in (Neelakantan et al., 2015); we show that our ROP part boosts the system as it enables relation paths to encode entity information with multiple updating – as opposed to a single update per path as in (Das et al., 2017). Similar to l_{seq}, we define:

$$l_{\text{pred}} = \sum_{j,i} \max(0, \beta + \text{s}(\mathbf{h}_j^-, \mathbf{r}) - \text{s}(\mathbf{h}_i, \mathbf{r})) \tag{19}$$

where $\mathbf{r}$ is the embedding of a query relation in Figure 3, $\mathbf{h}_i$ (resp. $\mathbf{h}_j^-$) is the representation of positive (resp. negative) path example i (resp. j), shown as $\mathbf{h}_t$ in Figure 3. In testing, all paths are ranked in terms of the given query relation.

The target relation is often not inferrable from all paths between head and tail (see Figure 1(a)); it can be entailed by a single path or a subset of paths. Hence, given the representation of a group of paths, how to match them with the representation of the target relation is a key problem. We use max (over all paths) because we found it works well in selecting the best path for the target relation. See last paragraph of §5.1 for discussion.

[3]Using the last hidden state in ROP (which participates in predicting the tail entity) to predict the query relation performed much worse. We suspect the finetuning by tail entity makes it not indicative for query relations.

Model	MAP
PRA (Lao et al., 2011)	64.43
PRA+ Bigram (Neelakantan et al., 2015)	64.93
RNN-path (Neelakantan et al., 2015)	68.43
RNN-path-entity (Das et al., 2017)	71.74
RNN-path-types (Das et al., 2017)	73.26
ROP_ARC1	74.23
ROP_ARC2	74.46
ROP_ARC3	**76.16**

Table 1: Results of mh-KBC task

We use AdaGrad (Duchi et al., 2011) with learning rate 0.1. Relation embedding dimension is 200, margins α and β are 0.5 (Eqs. 18–19), entity embedding dimension 200, batch size 20, negative sampling size 4. Longer paths are truncated to 8, the first 30 paths are kept for each entity pair.

Results. Table 1 compares our ROP systems to five baselines. RNN-path (Neelakantan et al., 2015) composes the relations occurring in a path using a vanilla RNN. It ignores all information about within-path entities and trains separate models per relation. RNN-path-entity (Das et al., 2017) models the path entities and improves the results. RNN-path-type (Das et al., 2017) further improves the result by representing entities with the sum of their type embeddings. Thus, RNN-path-type uses extra information, the entity types. Apart from these baselines, it is also feasible to compare with the baselines based on composition of one-hop triple-based embedding models. However, the performance of these baselines is very poor for this task (Neelakantan et al., 2015) and therefore, we do not include them in our comparison.

The performance order of our architectures for mh-KBC is ROP_ARC3 $>$ ROP_ARC2 $>$ ROP_ARC1. All our ROP systems are superior to the state-of-the-art. In particular, they are superior to RNN-path-type (Das et al., 2017), the state-of-the-art, even though we do not need and do not use information about the types of entities. Comparing ROP performance to RNN-path-entity is more fair, and this makes it even clearer that our modeling is effective. Das et al. (2017) encode entities along with relations into the path representation explicitly. Their motivation is that entity incorporation prevents prediction of the same target relation when relation sequences are the same, but path entities are different. Our ROP models achieve this implicitly, as the relation sequence can predict the entity sequence; this makes sure the representation of the relation sequence is specific to the entity sequence. As a result, the same goal can be achieved as (Das et al., 2017).

In (Das et al., 2017), an entity and a relation *are concatenated as a new unit* in the path; both entity and relation representations are trained based on the given gold relation as label. ROP predicts intermediate entities and updates the entity/relation embeddings and other parameters in each step; the training signals can come from the in-path target entities and from the reasoning task specific annotations. Thus, ROP makes use of the in-path structures to learn good quality entity/relation embeddings, which are further employed and finetuned to solve the reasoning problem.

This can also explain why max (over all paths) works well for us while Das et al. (2017) found Log-SumExp (over all paths) works better. As their system solely relies on target relations as training signals, max selection prevents all other paths from generating gradients to update, so max tends to select randomly in the initial stage. However, in our system, the representations of entities and relations get rich training signals at each path hop, so that the path representations are more reliable. Hence, max can select the most informative path and avoid misleading paths to support the claim of target relation. In a similar vein, max pooling is widely found more effective than mean/sum pooling for classification as this task mostly relies on the dominant features.

5.2 Path Query Answering (mh-PQA)

Dataset. We use the mh-PQA dataset released by Guu et al. (2015) and refer to it as *BaseKGP*[4]. BaseKGP contains paths like $e_h, r_1, r_2, \cdots, r_t, e_t$, where e_h and e_t are head and tail entities, connected

[4]Das et al. (2017) use a dataset based on the WordNet, however they conclude that the dataset is not an ideal test bed for mh-PQA due to some limitations: it is fairly small, with very short paths, few unseen paths during test time, and only one path between an entity pair. Therefore, we experiment on BaseKGP.

	#paths	#entities	#relations
train	6,266,058	75,043	26
dev	27,163	41,010	26
test	109,557	96,858	26

Table 2: Statistics of EnhancedKGP for mh-PQA

	methods	MQ	H@10
BaseKGP (baselines)	Comp-Bilinear	83.5	42.1
	Comp-Bilinear-Diag	84.8	38.6
	Comp-TransE	88.0	50.5
BaseKGP (our model)	ROP_ARC1	88.3	52.7
	ROP_ARC2 (ADD)	89.3	54.3
	ROP_ARC2 (GRU)	89.6	54.9
EnhancedKGP (our model)	ROP_ARC1	89.4	54.2
	ROP_ARC2 (ADD)	90.3	55.5
	ROP_ARC2 (GRU)	90.5	56.3
	ROP_ARC3 (ADD)	90.3	55.8
	ROP_ARC3 (eGRU)	**90.7**	**56.7**

Table 3: Results of mh-PQA

by relation sequence $r_1, \cdots, r_t$. There are 6,266,058/27,163/109,557 paths in train/dev/test.

BaseKGP is based on a Freebase subset released by Socher et al. (2013). The original subset contains a collection of Freebase triplets in form of (head, relation, tail). Guu et al. (2015) generated paths by traversing the triplet space. Paths in BaseKGP of form $e_h, r_1, r_2, \cdots, r_t, e_t$ do not contain intermediate entities. We create *EnhancedKGP* by enhancing each BaseKGP path *in train* as follows. We search entities at each step of a path by traversing the subset (Socher et al., 2013) until reaching the tail entity e_t. When there are multiple entity choices at a step, we randomly choose one. EnhancedKGP *train* has the same size as BaseKGP *train*, except that paths are filled by intermediate entities. Table 2 gives statistics. We will release EnhancedKGP, the first dataset for mh-PQA that includes within-path entities.

Task setup. We tune parameters on dev. We sample 10 negative entities for each ground truth entity in the path and use ranking loss (Eq 18) (with α=0.3, s() = cosine similarity). For testing, we ignore intermediate predicted entities and only output the tail entity. We update parameters – relation embeddings, entity embeddings (both dimension 300) and GRU parameters – using AdaGrad with learning rate 0.01 and mini-batch size 50.

We compare ROP with the three compositional training schemes *Bilinear*, *Bilinear-Diag* and *TransE* in (Guu et al., 2015). The compositional training of TransE, denoted as *Comp-TransE* in this work, is the state-of-the-art in mh-PQA. For ROP_ARC2, we report the performance of using ADD and GRU for COMP_1 in Eq 9. Similarly, for ROP_ARC3, we report the performance of using ADD and eGRU for COMP_2 in Eq 10.

In addition, to better investigate ROP models, we run them on both BaseKGP and EnhancedKGP. Note that since there are no intermediate entities in BaseKGP, ROP_ARC3 is reduced to ROP_ARC2. Hence, we report results on BaseKGP only for ROP_ARC1 and ROP_ARC2.

Two benchmark metrics are reported for this task (Guu et al., 2015): *hits at 10* (H@10), percentage of ground truth tail entities ranked in the top 10 of all retrieved; and *mean quantile* (MQ), normalized version of mean rank.

Results. Table 3 gives results for mh-PQA: top baselines on BaseKGP (block 1), ROP results on BaseKGP (block 2) and ROP results on EnhancedKGP (block 3). Overall, our ROP models all get new state-of-the-art by large margins (2–4% H@10 on BaseKGP, 4–6% H@10 on EnhancedKGP). The improvements of the second block over the first are evidence that recurrent neural networks are an appropriate paradigm in this task. The third block shows that encoding intermediate entities in an appropriate

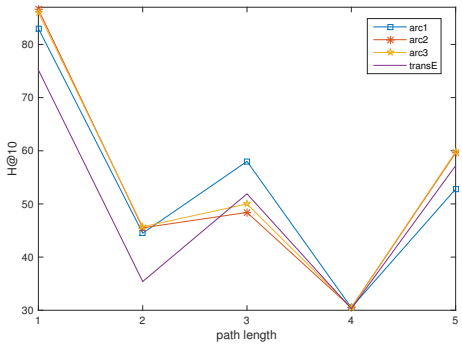


Figure 4: H@10 vs. path lengths on test set

way, like ROP does, can give a further boost.

Comparing ROP_ARC2 and ROP_ARC1, we realize that forwarding head entity e_h *directly* to predict each intermediate entity e_i is better than putting e_h as the start state of the GRU. We suspect this is due to the fact that the latest state of GRU is gradually less influenced by e_h when the following relation path is getting longer and longer. In ROP_ARC2, no matter how long the relation sequence $r_1, r_2, \cdots, r_i$ is, we always compose its whole representation $\mathbf{h}_i$ with the representation of e_h so that e_h can participate in the prediction of e_i more directly.

ROP_ARC3 further improves results by composing not only head entity e_h, but also the preceding predicted entity $\hat{e}_{i-1}$ with h_i to predict entity e_i. The result suggests that $\hat{e}_{i-1}$ provides critical information in this prediction process. This may be due to the property of RNN that latest inputs tend to be remembered better than old inputs, so the latest hidden state $\mathbf{h}_i$ is heavily influenced by the adjacent relations of position i; employing the preceding predicted entity $\hat{e}_{i-1}$ hence can help the prediction of e_i.

For composition, GRU/eGRU always surpass ADD – presumably due to the higher expressibility of (e)GRU's gating mechanism.

Performance vs. path lengths. Figure 4 graphs H@10 on different path lengths for our three ROP systems and the Comp-TransE baseline. We expected that H@10 should decrease gradually. However, Figure 4 shows that even-length paths are harder than odd-length ones. This surprising phenomenon is *due to a large number of inverse relations, as most inverse relations in BaseKGP are 1-to-N connections*.

The percentages of inverse relations in paths of length 1, 2, 3, 4, 5 are 0.0, 49.7, 38.3, 49.5 and 42.9, respectively. Spearman correlation coefficients between these percentages and system performance are always higher than 0.9. Thus, the more inverse relations, the worse performance. Unfortunately, to date there is no good way to model pairs of a relation $\mathbf{r}$ and its inverse $*\mathbf{r}$, e.g, "gender" and "*gender", as separate, yet at the same time as systematically related. Guu et al. (2015)'s TransE implementation enforces $*\mathbf{r}+\mathbf{r} = \mathbf{0}$. Figure 4 shows this is not as effective as expected, perhaps because it fails for 1-to-N projections. ROP treats inverse relations as independent, but we did not observe any worse performance. Better modeling of inverse relations is an interesting challenge for future work.

6 Conclusion

This work presented three ROP architectures for multi-hop KG reasoning. ROP models a KG path of arbitrary length as a pair of a relation sequence and an entity sequence, using the former to predict the latter by encoding and decoding step by step. Our neural sequential modeling of KG paths enables better learning of entity/relation embeddings because there are more training signals at each multi-hop path. ROPs showed state-of-the-art in two representative KG reasoning tasks, multi-hop KBC and multi-hop PQA. Incorporating knowledge from textual sources by initializing the entity embeddings with distributional representation of entities (Yaghoobzadeh and Schütze, 2015) could improve our results further, which we will explore in the future.

Acknowledgement

We gratefully acknowledge the support of the European Research Council for this research (ERC Advanced Grant NonSequeToR, # 740516).

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of NIPS*, pages 2787–2795.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of EACL*, pages 132–141.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

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

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of EMNLP*, pages 318–327.

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

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL*, pages 260–270.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of EMNLP*, pages 529–539.

Yankai Lin, Zhiyuan Liu, Huan-Bo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of EMNLP*, pages 705–714.

Xixun Lin, Yanchun Liang, and Renchu Guan. 2016. Compositional learning of relation paths embedding for knowledge base completion. *CoRR*, abs/1611.07232.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *Proceedings of HLT-NAACL*, pages 777–782.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *Proceedings of ACL*, pages 156–166.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of ICML*, pages 809–816.

Yelong Shen, Po-Sen Huang, Ming-Wei Chang, and Jianfeng Gao. 2016. Implicit reasonet: Modeling large-scale structured relationships with shared memory. *CoRR*, abs/1611.04642.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of NIPS*, pages 926–934.

Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *Proceedings of ACL*.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of EMNLP*, pages 715–725.

Hybrid Attention based Multimodal Network for Spoken Language Classification

Yue Gu, Kangning Yang*, Shiyu Fu*, Shuhong Chen, Xinyu Li and Ivan Marsic
Multimedia Image Processing Lab
Electrical and Computer Engineering Department
Rutgers University, Piscataway, NJ, USA
{yue.guapp, ky189, sf568, sc1624, Xinyu.li1118, marsic}@rutgers.edu

Abstract

We examine the utility of linguistic content and vocal characteristics for multimodal deep learning in human spoken language understanding. We present a deep multimodal network with both feature attention and modality attention to classify utterance-level speech data. The proposed hybrid attention architecture helps the system focus on learning informative representations for both modality-specific feature extraction and model fusion. The experimental results show that our system achieves state-of-the-art or competitive results on three published multimodal datasets. We also demonstrated the effectiveness and generalization of our system on a medical speech dataset from an actual trauma scenario. Furthermore, we provided a detailed comparison and analysis of traditional approaches and deep learning methods on both feature extraction and fusion.

1 Introduction

Understanding human conversation is fundamental for human-computer interaction (HCI) and artificial intelligence (AI). However, it is hard for a computer to precisely interpret human meaning because: 1. Giving computers the ability to interpret speech requires a complete understanding of how it works for a human. Unfortunately, we still cannot identify how humans understand the information during conversation. 2. It is hard to extract associated features; there is a gap between the extracted modality-specific features and the actual human state of mind. 3. During conversation, people often accept messages from multiple sources such as facial expression, gesture, linguistic content, and vocal signals. But how to integrate the heterogeneous inputs into a computer is still an open-ended question. In this paper, we focus on exploring and addressing the issues above on the spoken language classification task, which is the most commonly used method for human sentiment analysis, emotion recognition, and speech topic categorization.

Comprehending spoken language can be rephrased as analyzing vocal signals. A variety of research focuses on extracting the human meaning from audio data (Koolagudi et al., 2012; Busso et al., 2013; Mirsamadi et al., 2017). However, instead of directly processing audio information at the frame-level as a computer does, humans comprehend the meaning of utterances on the word-level, which can be seen as extracting meaning from linguistic content. It is natural for beginners in English to translate speech information into their native language word-by-word to understand the conversation, effectively using textual information for spoken language understanding. Based on this assumption, we propose a multimodal structure that considers both the audio signal and text as inputs to classify utterance-level speech data, where text data can be either transcribed speech or automatic speech-to-text output.

Another challenge for classifying spoken language is extracting the associated features. As mentioned in previous research (Poria et al., 2017a), traditional approaches used handcrafted features for both the text and audio branches, which cannot fully represent the high-level characteristics. Recent studies tried to overcome such issues with deep learning structures, including convolutional neural networks (CNNs)

* Equally Contribution

Proceedings of the 27th International Conference on Computational Linguistics, pages 2379–2390
Santa Fe, New Mexico, USA, August 20-26, 2018.

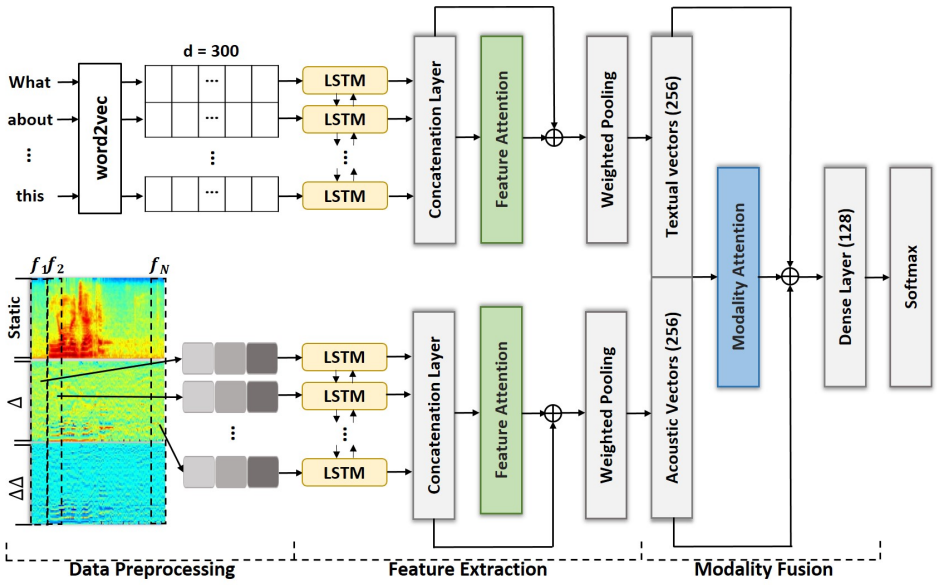


Figure 1: Overall system structure.

(Poria et al., 2015) and long short-term memory networks (LSTMs) (Zadeh et al., 2017). However, the words and audio frames should have different importance. For example, the word hate carries all the anger information in "I hate you". Even the same word or acoustic frame may have different contributions for different classes. To select the informative words and frames, we introduced an LSTM with an attention mechanism as the feature extractor on both the text and audio branches. A weighted pooling strategy was applied over the feature extractor to form a modality-specific feature representation. Compared to the previous research using independent software or networks to extract the modality-specific features, our system simultaneously tunes both the feature extraction and fusion modules, encouraging optimal feature learning.

Decision-level fusion is a commonly used strategy for fusing heterogeneous inputs, combining the independent modality outputs by using several specific rules. However, the lack of mutual association learning across modalities is a major limitation of applying decision-level fusion (Zhang et al., 2017). Feature-level fusion aims to fuse the extracted modality-specific features as a general feature vector, and so has the ability to learn correlations across modalities. Unfortunately, it is difficult to ensure data synchronization, as different features may consist of diverse time scales and formats (Poria et al., 2017a). To address the above issues, model-level fusion has been introduced, which generates a joint or shared feature representation consisting of both the feature-level fusion and decision-level fusion characteristics. Instead of using shallow structures in fusion as previous approaches did (Kim et al., 2013; Pang et al., 2015), we designed a modality attention fusion that allows system fusion at the feature-level and applies weighted modality scores over the extracted features to indicate the importance of different modalities. This keeps advantages of both feature-level and decision-level fusion.

We evaluated our system on three published multimodal datasets for spoken language understanding tasks, including speech sentiment analysis (CMU-MOSI and MOUD) and speech emotion recognition (IEMOCAP). Our system achieves state-of-the-art on CMU-MOSI and IEMOCAP, and competitive results on MOUD. We also generalize the system to classify speech content for utterance-level data on an an actual trauma resuscitation speech dataset (TRS). The results show that the proposed network improves performance on noisy data. Specifically, this paper addresses the following issues:

1. Does multi-source input data improve performance, and is it necessary to apply text information to a spoken language classification task?

2. Is the proposed attention-based LSTM structure helpful? Does extracting high-level representations perform better than low-level handcrafted features?

3. Is modality attention needed?

4. Is the proposed system enough to perform spoken language classification in a real scenario?

The paper is organized as follows: we present the related work in section 2. Section 3 describes the proposed network. We provide the experiments in section 4 and discuss the results in detail in section 5. We conclude in section 6.

2 Related Work

Previous works in spoken language classification focused on extracting acoustic features from different aspects of speech data (Koolagudi et al., 2012; Busso et al., 2013; Mirsamadi et al., 2017). While there is much previous research using audio-visual data on emotion recognition and sentiment analysis, only a few of them consider text as input. These approaches are difficult to generalize to spoken language understanding (due to the lack of visual data in many scenarios) and ignore the contribution of text. Recent studies on speech-based analysis have proposed to use both audio and text data for classification (Jin et al., 2015; Gu et al., 2018). However, they only applied the experiments to emotion recognition.

A variety of feature extraction strategies were proposed in the last decade. Early research used low-level acoustic descriptors and derivations (LLDs) with functional statistics as acoustic features (Rosas et al., 2013). For textual features, they used SVMs with bag of words (BoW) and part of speech (PoS) features in addition to low-level acoustic features (Rozgic et al., 2012; Rosas et al., 2013). Since low-level features represent limited high-level associations (Poria et al., 2015), various deep learning approaches have been proposed, like CNNs (Poria et al., 2016) and LSTMs (Gu et al., 2017b; Zadeh et al., 2017), to learn high-level representations. To further improve system performance, an attention mechanism was introduced in machine translation and text classification (Bahdanau et al., 2014; Yang et al., 2016)

There exist two commonly used fusion strategies in previous research: decision-level fusion and feature-level fusion. Specifically, Poria et al. (Poria et al., 2015, 2016) used a multiple kernel learning strategy to fuse the modality data on the feature-level. A decision-level fusion was applied by Wöllmer et al. (Wöllmer et al., 2013) that combines the results of the text and audio-visual modalities by a threshold score vector. Deep neural network fusion was proposed in a recent study to fuse the extracted modality-specific features (Zhang et al., 2017; Gu et al., 2018). More recent approaches introduced LSTM structures to fuse the features at each time step (Poria et al., 2017b; Chen et al., 2017)

3 Method

We introduce the design of the proposed architecture in this section (shown in Figure 1). There are three major parts of the system: the data preprocessing, feature extraction, and modality fusion.

3.1 Data Preprocessing

The system accepts raw audio signal and text as inputs. The data preprocessing module formats the heterogeneous inputs into specific representations, which can be effectively used in the feature extraction network. We embedded the words and extracted Mel-frequency spectral coefficients (MFSCs) from the text and audio inputs for the feature extraction module.

We first embedded each word into a 300-dimensional word vector by *word2vec*, which is a pre-trained word embedding dictionary trained on 100 million words from Google news (Mikolov et al., 2013). Compared to *GloVe* and *LexVec*, *word2vec* provides us the best performance. For all embedded vectors, we allow fine-tuning of the embedding layer via backpropagation during the training stage. We removed all punctuation, as spoken language does not provide tokens. Unknown words were randomly initialized and each sentence was represented as a $N \times 300$ matrix, where N is the number of the words for the given sentence.

Unlike most previous research extracting LLDs or using Mel-frequency cepstral coefficients (MFCCs) as the acoustic features (Poria et al., 2016; Mirsamadi et al., 2017), we represented the raw audio signal using MFSCs because: 1. MFSCs maintain the locality of the data by preventing new bases of spectral energies resulting from discrete cosine transform in MFCCs extraction (Abdel-Hamid et al., 2014). 2. Compared to the MFCCs that only have 39 dimensions for each audio frame, MFSCs allow more dimensions in the frequency domain that aid learning in deep models. 3. Instead of using MFCCs, voice

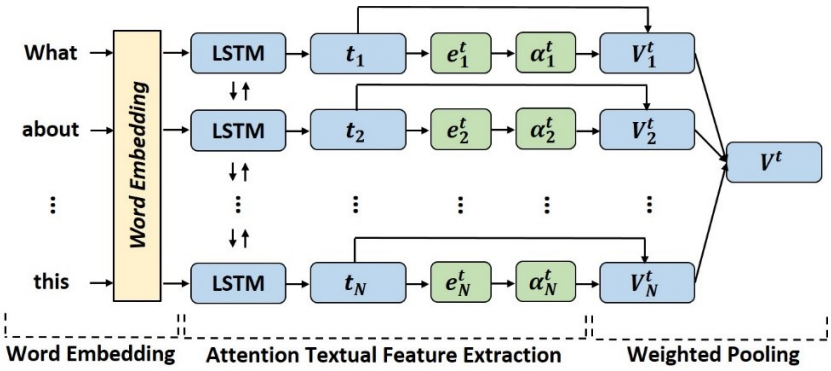


Figure 2: Textual feature extraction with attention.

intensity, pitch, etc. as in (Poria et al., 2017b) that need voice normalization and statistic computations, MFSC extraction does not require additional operations. As suggested in (Gu et al., 2017), we used 64 filter banks to extract static, delta (Δ), and double delta ($\Delta\Delta$) of the MFSCs as the MFSCs map. The final representation is a 3-D array with $64 \times F \times 3$ dimensions, where F is number of extracted MFSCs frames.

3.2 Textual Feature Extraction with Attention

We applied the LSTM structure with an attention mechanism to extract temporal associations and select informative words.

The textual feature extraction module consists of two parts. Firstly, it has a regular bidirectional LSTM structure used to generate the contextual hidden states for each word vector. Secondly, it has an attention layer connected to the bidirectional LSTM to provide a weight vector over the contextual hidden states to amplify the representative vectors. As shown in Figure 2, we fed the words into the bidirectional LSTM in sequence. Specifically,

$$t_i^{\rightarrow}, t_i^{\leftarrow} = bi\_LSTM(E_i), i \in [1, N]$$

where E_i is the embedded word vector of the ith word, bi_LSTM is the bidirectional LSTM, and $t_i^{\rightarrow}$ and $t_i^{\leftarrow}$ denote respectively the forward and backward contextual states of the given input word vector. Each contextual state is a word-level feature representation with forward and backward temporal associations. As not all words equally contribute to the final prediction, we added a learnable attention layer over the contextual states to denote the importance of the representations. As defined by (Bahdanau et al., 2014), we first computed the text attention energies (e_i^t) by:

$$e_i^t = tanh(W_t[t_i^{\rightarrow}, t_i^{\rightarrow}] + b_t), i \in [1, N]$$

Then, we calculated the text attention distribution (α_i^t) for word representations via a softmax function:

$$\alpha_i^t = \frac{exp(e_i^{t^\top} v_t)}{\sum_{k=1}^{N} exp(e_k^{t^\top} v_t)}$$

where W_t, b_t, and v_t are the learnable parameters. To form the final textual feature representation (V^t), we applied a weighted-pooling by computing a weighted sum of the text contextual states and the attention distribution:

$$V^t = \sum_{i=1}^{N} [t_i^{\rightarrow}, t_i^{\leftarrow}] \alpha_i^t$$

Unlike the systems that apply convolutional neural networks to extract the sentimental and emotional textual features using a fixed window size (Poria et al., 2015; Poria et al., 2017b), we used LSTM structures that can fully capture the sequential information with varying length and learn the temporal associations between words. We notice that Zadeh also applied LSTMs as the textual feature extractor

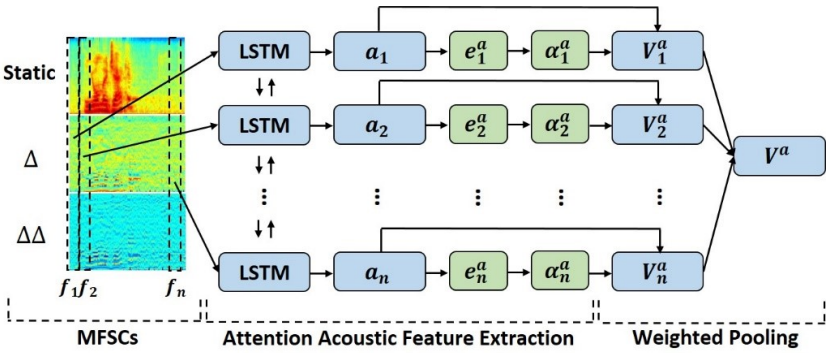


Figure 3: Acoustic feature extraction with attention.

(Zadeh et al., 2017). However, they used a mean-pooling strategy to form the final utterance-level feature representation by passing all the contextual states into the dense layer. This assumes all the outputs can correctly contribute to the final prediction. Unfortunately, as we know, even the same word may carry diverse information that may make a different contributions to the final prediction. The proposed attention layer allows the system to focus on the most informative words to further improve the representations.

3.3 Acoustic Feature Extraction with Attention

Similar to textual feature extraction, we also introduced a bidirectional LSTM with attention to focus on extracting informative contextual states on frame-level MFSCs.

Unlike the textual feature extraction that only has one channel (2D-array), the input MFSCs map is a 3D-array. We first concatenated the synchronized frames from static, delta, and double delta feature maps to form the input acoustic feature vector (A_j):

$$A_j = [s_j, \Delta_j, \Delta\Delta_j], j \in [1, F]$$

Again, we used the same approach as in textual feature extraction to compute the bidirectional acoustic contextual states $([a_j^{\rightarrow}, a_j^{\leftarrow}])$, acoustic attention energies (e_j^a), and acoustic attention distribution (α_j^a). The α_j^a can be understood as the importance score for the jth frame. We computed the weighted sum of the bidirectional acoustic contextual states and acoustic attention distribution as the final acoustic representation (V^a).

Unlike previous research that directly uses the acoustic LLDs as the extracted features (Degottex et al., 2014; Poria et al., 2016), the proposed architecture learns high-level acoustic associations. We didn't use convolutional neural networks to extract the acoustic features as in (Gu et al., 2017) because CNNs only capture spatial associations whereas acoustic data contains many temporal associations. The fixed window size of CNNs limits the temporal interaction extraction. As the number of audio frames is large (hundreds per sentence), the LSTM structure ensures the system captures long-term dependencies among the MFSCs frames. Even if a deep neural network was used for extracting the high-level associations on LLDs (Zadeh et al., 2017; Gu et al., 2018), the generation of attention over the extracted features is still desirable, as it can help indicate the importance at the frame-level. The weighted pooling based on the attention distribution makes sure the final acoustic feature representations contain the most informative features.

3.4 Modality Fusion

Simply concatenating the features cannot reveal the actual importance of different modalities; the same modality may have different contributions in different spoken language understanding tasks. For example, people rely more on the vocal delivery and acoustic characteristics to express their emotions, but linguistic content and text are more important to speech content classification. Even for the same task, the modality may have distinct influences on different categories. Acoustic information might provide useful information for the *Anger* class, but it is hard to distinguish *neutral* and *happy* without considering text. To make the system learn this difference, we proposed a modality attention fusion that puts an attention

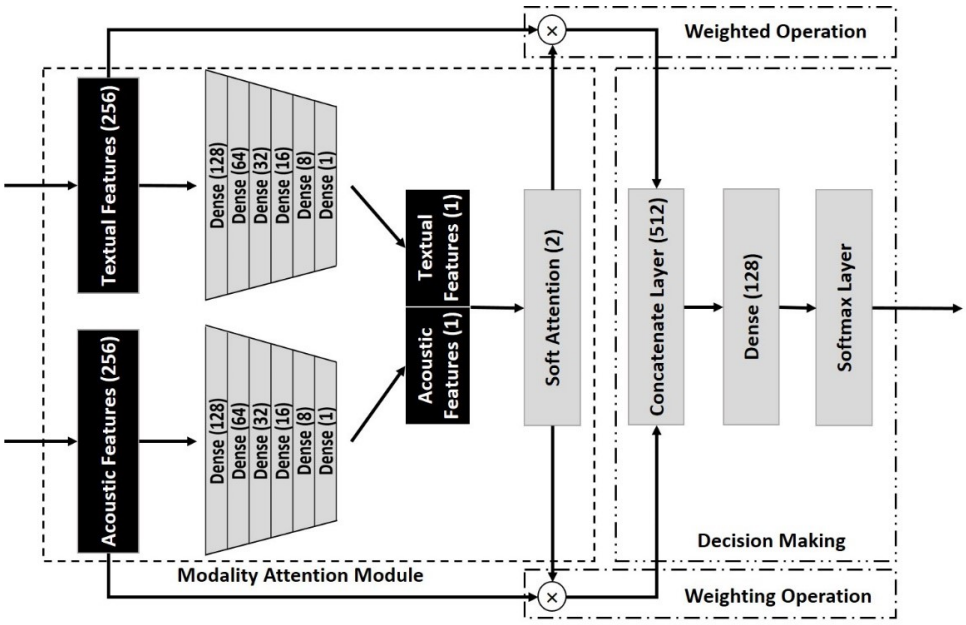


Figure 4: Modality Fusion

layer over the extracted modality-specific features, helping the system focus on the informative modality. It can be intuitively understood as giving a weighted score vector at the modality-level to indicate the importance of individual branches.

The proposed modality fusion consists of three major parts: a modality attention module, a weighted operation, and a decision making module. We first set up five dense layers after the attention layer to fuse the modality-specific features (as shown in figure 4). Then, we used softmax regression to generate the weighted score (s) for the given modality:

$$s = softmax(tanh(W_f[V_t^*, V_a^*] + b_f))$$

where W_f and b_f are the trainable fusion attention parameters, s is a n-dimension vector, and n=2 in this study (representing the text and audio modalities respectively). We computed a soft-attention over the original modality features and concatenated them. A dense layer was used to learn the associations across weighted modality-specific features by:

$$r = tanh(W_r[(1 + s_t)V^t, (1 + s_a)V^a] + b_r)$$

where r is the final representation, and W_r and b_r are the additional parameters for the last dense layer. We used $(1 + s)$ as the attention score to keep the original modality characteristics. We made the final decision by a softmax classifier using r as input.

4 Experiment

We evaluated the proposed system on three published multimodal datasets and an actual trauma resuscitation speech dataset. We compared our structure with the baselines from three major aspects: 1. proposed system vs previous methods; 2. low-level handcrafted features vs high-level features; 3. shallow fusion vs deep fusion. We also conducted an experiment on a trauma resuscitation speech dataset that uses speech-to-text results as text input to test the generalizability of the system.

4.1 Dataset

We selected three multimodal datasets that contain spoken language information. We used audio and text data as inputs in this study. Table 1 shows dataset details.

CMU-MOSI: This dataset is a multimodal sentiment intensity and subjectivity dataset consisting of 93 review videos in English with 2199 utterance segments (Zadeh et al., 2016). Each segment is labelled

by five individual annotators between -3 (strong negative) to +3 (strong positive). The aim of using this dataset is to extract the sentiments from spoken language information by applying the audio segments and the corresponding transcripts. We used binary labels (positive and negative) based on the sign of the annotations average. We used an 80-20 training-testing split that considers speaker independency. Specifically, there are 1755 utterances for training and 444 utterances for testing.

Dataset	Class	Speaker Independent	Training Set	Testing Set
CMU-MOSI	2	93 (74|19)	1755	444
IEMOCAP	4	151(121|30)	4295	1103
MOUD	2	79 (59|20)	322	115
TRS	7	50 (40|10)	7261	1843

Table 1: Dataset details.

IEMOCAP: The interactive emotional dyadic motion capture database is a multimodal emotion dataset including visual, audio, and text data (Busso et al., 2008). For this study, we only used the audio and text data and classified emotion at the utterance-level. We used the label agreed on by the majority and combined the *happy* and *excited* classes following previous research (Poria et al., 2016). The final dataset consists of four categories including 1591 *hap (happy+excited)*, 1054 *sad*, 1076 *anger*, 1677 *neutral*. We still used an 80-20 speaker independent data split. Table 1 shows the detailed separation.

MOUD: The MOUD dataset is a Spanish multimodal utterance-level dataset. Following previous research (Poria et al., 2016), we only consider the positive and negative labels during training and testing. Instead of translating the sentences into English as previous research did, we initialize the word embedding layer randomly.

In addition, we tested the generalizability of the proposed system on a trauma resuscitation speech dataset (TRS).

TRS: This dataset was collected from 50 actual trauma cases with 9104 utterance-level audio segments. For each segment, it contains one utterance with at least 2 seconds. The dataset contains the following utterance-level medical category labels: *airway*, *breathing*, *circulation*, *disability*, *exposure*, *secondary-survey*, and *others*. Each utterance was assigned one category by trauma experts. The audio data was collected by two shotgun microphones placed in the resuscitation room. We used two different transcripts as the text input: human transcribed text and speech-to-text transcript. These experiments can then evaluate the influence of noise in the text branch. We reserved 40 cases as the training set and the 10 others as the testing set.

4.2 Baselines

We first compared our system with several state-of-the-art methods.

SVM Trees: an ensemble of SVM trees was used for classifying concatenated bag-of-words and LLDs (Rozgic et al., 2012).

BL-SVM: extracted bag-of-words and low-level descriptors as textual and acoustic features, respectively. The model used an SVM classifier (Rosas et al., 2013).

GSV-eVector: this model used Gaussian Supervectors to select LLDs as acoustic features and extracted a set of weighted handcrafted vectors (eVector) as textual features. A linear kernel SVM was used as the final classifier (Jin et al., 2015).

C-MKL: the system used a multiple kernel learning structure as the final classifier (Poria et al., 2016). The model extracted textual and acoustic features by using a convolution neural network and OpenSMILE software, respectively.

TFN: a tensor fusion network was used to fuse the extracted features from different modalities (Zadeh et al., 2017).

WF-LSTM: a word-level LSTM with temporal attention structure to predict sentiments on the CMU-MOSI dataset (Chen et al., 2017).

BC-LSTM: a bidirectional LSTM structure to learn contextual information among utterances (Poria et al., 2017b).

Approach	CMU-MOSI		IEMOCAP		MOUD		TRS	
	Acc	W-F1	Acc	W-F1	Acc	W-F1	Acc	W-F1
SVM Tree	67.3	66.1	66.4	66.7	60.4	50.4	58.4	45.7
BL-SVM	68.4	67.8	65.2	65.0	60.3	52.8	59.2	50.1
GSV-eVector	65.7	65.5	64.2	64.3	61.1	52.3	58.4	48.4
C-MKL	71.3	71.0	67.0	67.2	72.0	72.2	62.1	58.1
TFN	73.6	73.5	70.4	70.2	62.1	61.2	64.4	61.5
WF-LSTM	73.9	73.3	69.5	69.4	72.7	72.8	65.6	61.5
BC-LSTM	72.4	72.6	70.8	70.8	72.4	72.4	67.9	64.4
H-DMS	70.4	70.2	70.2	69.8	68.4	67.6	66.7	64.3
Our Method	**76.2**	**74.8**	**72.1**	**72.2**	**72.8**	**73.0**	**69.4**	**66.0**

Table 2: Proposed system vs previous methods. Acc = accuracy (%). W-F1 = weighted F1 score.

H-DMS: a hybrid deep multimodal structure to extract and fuse the textual and acoustic features on the IEMOCAP dataset (Gu et al., 2018).

We further tested the performance of models using different feature extraction methods.

BoW: using bag-of-words as the textual features to make the final prediction (Wöllmer et al., 2013).

WEV: directly using word embedding vectors as the textual features (Zadeh et al., 2018).

CNNs-t: Convolutional neural networks were used for extracting the textual features based on embedding word vectors (Poria et al., 2015).

LSTM-t: using an LSTM structure to learn contextual word-level textual features (Gu et al., 2017b).

OpenSimle: extracts 6373 low-level acoustic features from an entire audio clip (Poria et al., 2017b).

COVAREP: extracts low-level acoustic features including MFCCs, pitch tracking, glottal source parameters, peak slope, and maxima dispersion quotients (Chen et al., 2017).

CNNs-a: using convolutional neural networks on extracted MFSCs (Gu et al., 2017).

LSTM-a: using an LSTM structure to learn the temporal associations based on LLDs extracted by OpenSmile (Gu et al., 2018).

To make the comparison more reasonable, we introduced a shallow fusion and a deep fusion that combines with the previous feature extraction strategies to make the final predictions.

SVM: an SVM was trained on modality-specific features or concatenated features for classification.

DF: a deep neural network with three hidden layers was trained as the fusion module and a softmax classifier was used for decision-making.

4.3 Implementation

We implemented the system in Keras using the Tensorflow backend. Instead of directly training the entire network, we first pre-trained the feature extraction networks by using two individual softmax classifiers. Then, we tuned the entire network by combining the feature extraction module and modality fusion module. The system was trained on a GTX 1080 GPU with 32GB RAM. We set 256 as the dimension for the bidirectional LSTM. We selected the ReLU activation function except for the attention layers. To overcome overfitting and internal covariate shift (Ioffe and Szegedy, 2015), we applied dropout and batch normalization after the bidirectional LSTM layer and attention layers. We initialized 0.01 as the learning rate, used the Adam optimizer, and binary/categorical cross-entropy loss. We further split 20 percent of the data from the training set as validation and used mini-batch size 8. To make a fair comparison between the proposed system and baselines, we re-trained all models on the same training-testing set split (shown in Table 1). We directly built the models for the baselines that provided the source code. For the rest, we re-implemented the models based on the methods described in their papers.

5 Experiment Result

We first compared the performance of the proposed system with the previous methods. The result shows that our system achieves state-of-the-art on all three published datasets. Specifically, we achieved 76.2% accuracy and 74.8 weighted F1 score on CMU-MOSI, outperforming the previous methods by a margin

(a) Comparison of modalities			(b) Comparison of Features					
Approach	CM	IE	Approach	CM	IE	Approach	CM	IE
BoW+SVM	65.3	53.2	BoW+SVM	65.3	53.2	OS*+SVM	52.9	56.4
OS*+SVM	52.9	56.4	WEV+SVM	65.4	54.7	COV*+SVM	51.5	52.7
BoW+OS*+SVM	65.9	61.7	CNNt+SVM	67.3	55.2	CNNa+SVM	54.1	55.4
CNNt+DF	69.2	57.8	LSTMt+SVM	68.2	55.7	LSTMa+SVM	56.9	56.1
CNNa+DF	57.3	59.9	ATFE+SVM	72.2	61.0	AAFE+SVM	57.1	59.1
CNNt+CNNa+DF	71.6	64.2	CNNt+DF	69.2	57.8	OS*+DF	56.1	58.7
ATFE+DF	74.5	61.8	LSTMt+DF	71.2	58.2	COV*+DF	55.1	56.3
AAFE+DF	60.4	62.5	LSTMa+DF	58.5	60.5	CNNa+DF	57.3	59.9
ATFE+AAFE+MAF	**76.2**	**72.1**	ATFE+DF	**74.5**	**61.4**	AAFE+DF	**60.4**	**62.5**

(d) Generalization	
Approach	TRS
AAFE+DF	56.5
ATFE(trans)+DF	66.8
ATFE(asr)+DF	47.7
ATFE(trans)+AAFE+DF	69.4
ATFE(asr)+AAFE+DF	58.9

(c) Comparison of Fusion					
Approach	CM	IE	Approach	CM	IE
BoW+OS*+SVM	65.9	61.7	CNNt+CNNa+SVM	65.7	63.4
BoW+OS*+DF	67.2	63.2	CNNt+CNNa+DF	71.6	64.2
BoW+OS*+MAF	68.7	64.7	CNNt+CNNa+MAF	72.9	66.1
WEA+COV*+SVM	65.8	62.7	ATFE+AAFE+SVM	71.1	65.1
WEA+COV*+DF	67.7	64.1	ATFE+AAFE+DF	74.8	70.5
WEA+COV*+MAF	68.5	64.8	ATFE+AAFE+MAF	**76.2**	**72.1**

Table 3: Detailed comparison on CMU-MOSI (CM) dataset and IEMOCAP (IE) dataset (accuracy percentage). OS* = OpenSmile. COV* = COVAREP. ATFE = proposed attention based textual feature extraction. AAFE = proposed attention based acoustic feature extraction. MAF = modality attention fusion.

of 2.3% to 7.8%, which demonstrates the effectiveness of the proposed architecture. Compared to the traditional approaches using low-level handcrafted features and shallow fusion strategies (GSV-eVector and SVM Trees), the proposed method shows a significant performance improvement on IEMOCAP (9.3% and 8.7% accuracy gain, respectively). Experiments also indicate that our system performs better than the deep approaches (including C-MKL, TFN, H-DMS), showing the necessity of learning attentive information on feature extraction and fusion levels. Our approach achieves a competitive result (72.8% accuracy) on the MOUD dataset. We further re-implemented all previous methods on the TRS dataset, and our system reports the best performance in terms of both accuracy (69.4%) and weighted F1 score (66.0).

We further compare low-level vs high-level features and shallow vs deep fusion. We re-trained all the individual feature extraction baselines and fusion structures on both IEMOCAP and CMU-MOSI with the same training-testing split. As shown in Table 3 (a), (b), and (c), we made several different combinations of the feature extraction baselines with fusion baselines. We first evaluated the performance of unimodal and multi-modal systems. From Table 3 (a), in all of combinations, multi-modal systems performed better than unimodal ones. In general, the performance of text is similar to that of audio on the IEMOCAP dataset, but text dominates the system performance on MOSI. This might because humans rely more on vocal delivery to express emotions, but less on sentiments. Combining textual and acoustic modalities using an ATFE+AAFE structure leads to 9.6% performance boost on IEMOCAP, which proves the necessity of using multimodal inputs in spoken language understanding. However, there is only 1.7% accuracy improvement on CMU-MOSI by using a multimodal structure. This might because humans express their attitudes without using many vocal characteristics.

Table 3 (b) compares the different feature extraction methods. Compared to traditional textual feature extraction (BoW), the deep models achieve better performance by extracting high-level associations on both datasets. It worth mentioning that directly using the word vectors extracted by *word2vec* model as textual features (WEA+SVM) cannot outperform CNN and LSTM word vector feature extractors (CNNs-t+SVM and LSTM-SVM). This observation demonstrates the necessity of extracting high-level features. On IEMOCAP, the high-level acoustic features extracted by CNNs-a and LSTM-a achieves 59.9% and 60.5% accuracy, outperforming the low-level handcrafted acoustic features (OpenSimle+SVM and COVAREP+SVM) between 1.7% to 7.8% in accuracy. We notice that applying the LSTM architecture over the LLDs gives a 2.4% accuracy increase compared to directly using the LLDs

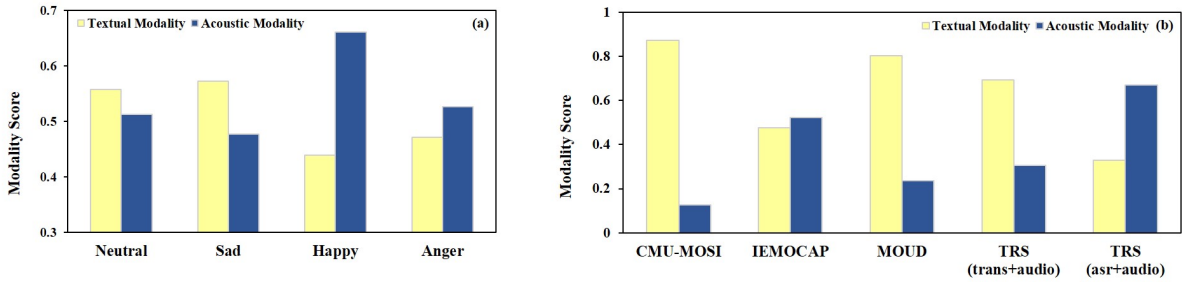


Figure 5: The weighted scores of modality attention. (a) Modality attention scores of different categories on IEMOCAP. (b) Modality attention scores of different datasets.

on CMU-MOSI, which shows that modeling the temporal associations improve system performance. As expected, the proposed attention-based textual and acoustic feature extraction performs the best on each individual branch. Based on the above observations, we conclude that learning the high-level features from textual and acoustic data improves the system performance, and that the proposed attention-based LSTM structure indeed helps extract associated features.

Compared to the performance of shallow fusion (SVM) in Table 3 (c), deep fusion (DF) gives a significant performance improvement on combinations that use deep feature extractors (CNNs, LSTM, and proposed attention structure), demonstrating that extracting associations across modality-specific features indeed helps the final decision-making. The modality fusion outperforms both shallow fusion (directly using SVM classifier) and deep fusion (DF) on diverse feature extraction combinations. Using an MAF structure instead of SVM and DF brings 5.1% and 1.4% accuracy gain on CMU-MOSI, respectively. To further compare, we visualized the weighed scores from the modality attention on different datasets and categories (shown in Figure 5). We computed the average scores of one hundred random testing samples from each category and dataset. The results indicate the proposed modality attention can learn the distinct scores on different categories and datasets.

We further tested the generalization of the proposed system by applying it to the TRS dataset. Instead of just using the transcribed speech text, we fed the raw audio data into the IBM Watson speech to text API to automatically recognize speech (ASR). From Table 3 (d), using the ASR text leads to a 19.1% accuracy decrease compared to the transcribed text on unimodal systems. However, the multimodal structure only has a 10.5% accuracy drop. These observations indicate that the multimodal system is tolerant to noisy data, demonstrating the generalizability of the proposed multimodal architecture with modality attention.

6 Conclusion

In this paper, we introduced a hybrid attention based multimodal architecture for different spoken language understanding tasks. Our system used feature attention and modality attention to select the representative information at both the feature-level and modality-level. The proposed modality attention fusion overcomes the limitations from feature-level and decision-level fusion by performing feature-level fusion with modality scores over the features. We evaluated our system on three published datasets and a trauma resuscitation speech dataset. The results show that the proposed architecture achieves state-of-the-art performance. We also demonstrated the necessity of applying a multimodal structure, extracting high-level feature representations, and using modality attention fusion. The generalization testing established that our system has the ability to handle actual speech data.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and feedback. This research was funded by the National Institutes of Health under Award Number R01LM011834.

References

Abdel-Hamid, O., Mohamed, A. R., Jiang, H., Deng, L., Penn, G., & Yu, D. 2014. *Convolutional neural networks for speech recognition.* IEEE/ACM Transactions on audio, speech, and language processing, 22(10), 1533-1545.

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

Busso, C., Bulut, M., Lee, C. C., Kazemzadeh, A., Mower, E., Kim, S., ... & Narayanan, S. S. 2008. *IEMOCAP: Interactive emotional dyadic motion capture database.* Language resources and evaluation, 42(4), 335.

Busso, C., Bulut, M., Narayanan, S., Gratch, J., & Marsella, S. 2013. *Toward effective automatic recognition systems of emotion in speech. Social emotions in nature and artifact: emotions in human and human-computer interaction*, J. Gratch and S. Marsella, Eds, 110-127.

Chen, M.,Wang, S., Liang, P. P., Baltruaitis, T., Zadeh, A., & Morency, L. P. 2017, November. *Multimodal sentiment analysis with word-level fusion and reinforcement learning.* In Proceedings of the 19th ACM International Conference on Multimodal Interaction (pp. 163-171). ACM.

Degottex, G., Kane, J., Drugman, T., Raitio, T., & Scherer, S. 2014, May. *COVAREPA collaborative voice analysis repository for speech technologies.* In Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on (pp. 960-964). IEEE.

Gu, Y., Li, X., Chen, S., Zhang, J., & Marsic, I. 2017, May. *Speech Intention Classification with Multimodal Deep Learning.* In Canadian Conference on Artificial Intelligence (pp. 260-271). Springer, Cham.

Gu, Y., Li, X., Chen, S., Li, H., Farneth, R. A., Marsic, I., & Burd, R. S. 2017, August. *Language-Based Process Phase Detection in the Trauma Resuscitation.* In Healthcare Informatics (ICHI), 2017 IEEE International Conference on (pp. 239-247). IEEE.

Gu, Y., Chen, S., & Marsic, I. 2018. *Deep Multimodal Learning for Emotion Recognition in Spoken Language.* arXiv preprint arXiv:1802.08332.

Ioffe, S., & Szegedy, C. 2015, June. *Batch normalization: Accelerating deep network training by reducing internal covariate shift.* In International conference on machine learning (pp. 448-456).

Jin, Q., Li, C., Chen, S., &Wu, H. 2015, April. *Speech emotion recognition with acoustic and lexical features.* In Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on (pp. 4749-4753). IEEE.

Kim, Y., Lee, H., & Provost, E. M. 2013, May. *Deep learning for robust feature generation in audiovisual emotion recognition.* In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on (pp. 3687-3691). IEEE.

Koolagudi, S. G., & Rao, K. S. 2012. *Emotion recognition from speech: a review.* International journal of speech technology, 15(2), 99-117.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. 2013. *Distributed representations of words and phrases and their compositionality.* In Advances in neural information processing systems (pp. 3111-3119).

Mirsamadi, S., Barsoum, E., & Zhang, C. 2017, March. *Automatic speech emotion recognition using recurrent neural networks with local attention.* In Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on (pp. 2227-2231). IEEE.

Pang, L., Zhu, S., & Ngo, C. W. 2015. *Deep multimodal learning for affective analysis and retrieval.* IEEE Transactions on Multimedia, 17(11), 2008-2020.

Poria, S., Cambria, E., & Gelbukh, A. 2015. *Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis.* In Proceedings of the 2015 conference on empirical methods in natural language processing (pp. 2539-2544).

Poria, S., Chaturvedi, I., Cambria, E., & Hussain, A. 2016, December. *Convolutional MKL based multimodal emotion recognition and sentiment analysis.* In Data Mining (ICDM), 2016 IEEE 16th International Conference on (pp. 439-448). IEEE.

Poria, S., Cambria, E., Bajpai, R., & Hussain, A. 2017. *A review of affective computing: From unimodal analysis to multimodal fusion.* Information Fusion, 37, 98-125.

Poria, S., Cambria, E., Hazarika, D., Majumder, N., Zadeh, A., & Morency, L. P. 2017. *Context-dependent sentiment analysis in user-generated videos.* In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (Vol. 1, pp. 873-883).

Rosas, V. P., Mihalcea, R., & Morency, L. P. 2013. *Multimodal sentiment analysis of Spanish online videos.* IEEE Intelligent Systems, 28(3), 38-45.

Rozgic, V., Ananthakrishnan, S., Saleem, S., Kumar, R., & Prasad, R. 2012, December. *Ensemble of svm trees for multimodal emotion recognition.* In Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific (pp. 1-4). IEEE.

Wöllmer, M., Weninger, F., Knaup, T., Schuller, B., Sun, C., Sagae, K., & Morency, L. P. 2013. *Youtube movie reviews: Sentiment analysis in an audio-visual context.* IEEE Intelligent Systems, 28(3), 46-53.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. 2016. *Hierarchical attention networks for document classification.* In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computa-tional Linguistics: Human Language Technologies (pp. 1480-1489).

Zadeh, A., Zellers, R., Pincus, E., & Morency, L. P. 2016. *MOSI: multimodal corpus of sentiment intensity and subjectivity analysis in online opinion videos.* arXiv preprint arXiv:1606.06259.

Zadeh, A., Chen, M., Poria, S., Cambria, E., & Morency, L. P. 2017. *Tensor fusion network for multimodal sentiment analysis.* arXiv preprint arXiv:1707.07250.

Zadeh, A., Liang, P. P., Poria, S., Vij, P., Cambria, E., & Morency, L. P. 2018. *Multi-attention recurrent network for human communication comprehension.* arXiv preprint arXiv:1802.00923.

Zhang, S., Zhang, S., Huang, T., Gao, W., & Tian, Q. 2017. *Learning Affective Features with a Hybrid Deep Model for Audio-Visual Emotion Recognition.* IEEE Transactions on Circuits and Systems for Video Technology.

Exploring the Influence of Spelling Errors on Lexical Variation Measures

Ryo Nagata[†‡], **Taisei Sato**[†], **and Hiroya Takamura**[♭♠]
[†]Konan University
[‡]RIKEN AIP
[♭]National Institute of Advanced Industrial Science and Technology
[♠]Tokyo Institute of Technology
nagata-coling2018-k @ hyogo-u.ac.jp.

Abstract

This paper explores the influence of spelling errors on lexical variation measures. Lexical variation measures such as Type-Token Ration (TTR) and Yule's K are often used for learner English analysis and assessment. When applied to learner English, however, they can be unreliable because of the spelling errors appearing in it. Namely, they are, directly or indirectly, based on the counts of distinct word types, and spelling errors undesirably increase the number of distinct words. This paper examines the hypothesis that lexical variation measures become unstable in learner English because of spelling errors. Specifically, it tests the hypothesis on English learner corpora of three groups (middle school, high school, and college students). To be precise, it estimates the difference in TTR and Yule's K caused by spelling errors, by calculating their values before and after spelling errors are manually corrected. Furthermore, it examines the results theoretically and empirically to deepen the understanding of the influence of spelling errors on them.

Title and Abstract in French

Une exploration de l'influence des erreurs orthographiques sur les mesures de variation lexicale

Cet article explore l'influence des erreurs orthographiques sur les mesures de variation lexicale. Les mesures de variation lexicale telles que le rapport type-occurrence (*Type-Token Ratio*; TTR) et le K de Yule sont souvent employées pour analyser et évaluer l'anglais d'apprenants langue seconde. Lorsqu'on les applique a l'anglais d'apprenants, elles peuvent cependant s'avérer peu fiables du fait des erreurs orthographiques que l'on y rencontre. De fait, elles se fondent directement ou indirectement sur le décompte des types de mots diffŕents, et les erreurs orthographiques augmentent artificiellement le nombre de mots différents. Cet article examine l'hypothèse selon laquelle les mesures de variation lexicale deviennent instables sur l'anglais d'apprenants à cause des erreurs orthographiques. Il teste cette hypothèse sur des corpus d'anglais d'apprenants de trois groupes (élèves de collège, de lycée, et étudiants à l'université). Plus précisément, il estime la différence causée par les erreurs orthographiques sur le TTR et le K de Yule en calculant leurs valeurs avant et après correction manuelle des erreurs orthograpiques. Il examine de plus ces résultats théoriquement et empiriquement pour approfondir notre compréhension de l'influence des erreurs orthographiques sur ces mesures.

1 Introduction

Vocabulary richness measures are often used for learner language analysis and assessment as in the work by Arnaud (1984), Attali and Burstein (2006), Ishikawa (2015), Gregori-Signes and Clavel-Arroitia (2015), and Šišková (2012). Among a wide variety, Type-Token Ratio (TTR), which is defined as the

Proceedings of the 27th International Conference on Computational Linguistics, pages 2391–2398
Santa Fe, New Mexico, USA, August 20-26, 2018.

number of distinct words divided by the total number of words, is probably the most popular one because of its readiness; it is often used in the related research areas including the work as just mentioned above. Many modifications have been made to it, resulting in, for example, Guiraud's R (Guiraud, 1959), Herdan's C (Herdan, 1960), and Malvern and Richard's D (Malvern et al., 2004), to name a few (see Malvern et al. (2004) for the information about other measures). Another well known measure is Yule's K (Yule, 1944; Tanaka-Ishii and Aihara, 2015), which has the nice property that it is stable with respect to text length.

When applied to learner English[1], however, they might be unreliable because of spelling errors appearing in it. Typically, directly or indirectly, vocabulary richness measures including TTR and K are based on the number of distinct words in the target text. Here, it should be emphasized that the more spelling errors occur, the more (superficially) distinct words tend to appear (and thus, the greater number of distinct words). In the view of vocabulary richness, however, spelling errors derived from a canonical spelling should be treated as one word type. For example, the word *because* is mistakenly spelt as *becose*, *becouse*, and *becous* in learner English and they would unnecessarily increase the value of TTR if they were individually counted as unique word types. With the accumulation of the influence, vocabulary richness measures will likely deviate from their true values. A similar argument can be made about K that also uses the statistics on distinct words in the target text (in a different way, though).

Granger and Wynne (1999) have already pointed out this problem. They actually reported on the difference in TTR between an original text and its spelling-error-corrected version. At the same time, their report is not complete in that they only targeted spelling errors whose edit distance is within one from their correct spelling. Considering that spelling errors exist whose edit distance is more than one, one should adequately take care of all spelling errors in the target text to estimate accurately their influence on vocabulary richness measures. As Arnaud (1984) show (and also we will in Sect. 2), it is not straightforward to determine which type of spelling error to correct and how.

Apart from TTR, as far as we know, no one has yet revealed their influence on other lexical variation measures such as K. It would be especially interesting to see how they affect K which is stable with respect to text length.

In view of this background, in this paper, we explore the influence of spelling errors on TTR and K. Specifically, we explore the following hypothesis:

> **Hypothesis**: Lexical variation measures become unstable in learner English because of spelling errors.

to augment the findings Granger and Wynne (1999) showed. We test this hypothesis on English learner corpora of three groups (middle school, high school, and college students). To be precise, we estimate the differences in TTR and K caused by spelling errors by calculating their values before and after spelling errors are manually corrected. In addition, we examine the results theoretically and empirically to deepen our understanding of the influence of spelling errors on TTR and K.

The rest of this paper is structured as follows. Section 2 describes the corpus data we used in this work. It also explores the spelling errors we targeted. Section 3 describes the method we used to investigate the influence of spelling errors on TTR and K. Section 4 shows the results. Section 5 investigates them to discuss the influence of spelling errors on lexical richness theoretically and empirically.

2 Data and Spelling Errors

We use for our investigation English learner corpora of three groups ranging over middle school, high school, and college students whose mother tongue is all Japanese[2]. They consist of essays written on different topics; their statistics are shown in Table 1. Note that the essays in each group are written on different topics and thus it would be difficult to make intra-group comparisons; our emphasis here is on the inter-group comparisons before and after spelling error correction.

[1] In this paper, *learner English* refers to English as a Foreign Language.

[2] Because of the copyright, not all the corpus data can be open to the public. However, we have released a part of it with the information about spelling errors and their corrections to the public. It corresponds to the Konan-JIEM (KJ) learner corpus, which is available at `http://http://www.gsk.or.jp/en/catalog/gsk2016-b/`.

Group	# essays	# words	# errors	# topics	Av. edit distance[3]
Middle school	384	21,324	583	3	1.33
High school	251	23,561	680	3	1.37
College	438	37,774	1,271	14	1.42
TOTAL	1,073	82,659	2,534	20	1.38

Table 1: Basic Statistics on the Corpus Data.

To measure the difference, one has to determine which types of spelling errors should be corrected; we identified 13 error types in the corpora. Table 2 shows them with their acronyms and short explanations (e.g., RE for *Real word spelling error*).

Based on the taxonomy, we test two ways of correcting spelling errors to measure the difference. The first is to correct all 13 error types. The second is to classify them into three groups: those corrected, not corrected, and not counted. The first group (those corrected) consists of SP, PC, OC, GC, and NM. Spelling errors in these types would unnecessarily increase the number of distinct words if they were not corrected and left as they are. The second only consists of RE as in *Their is a house.* $\rightarrow$ *There is a house.* For this type of error, the writer might not know the correct word, and thus they should be left as is. Considering this, this type of error is not corrected (and thus it is counted as a unique word type). The rest (the third group) are those that do not exist in the English language. Therefore, they are not considered in calculating TTR and K. Namely, they are not included in the number of distinct words nor in the total number.

3 Method

We first applied the following preprocessing steps to the target corpora. We respectively used the sentence splitter and the tokenizer in Stanford Parser 3.5.0 (Chen and Manning, 2014) to split the essays into sentences and then into word tokens. We converted all word tokens into lowercase. After this, we removed those tokens containing no English alphabet letter from the corpora. In addition, we removed spelling errors whose correct spellings were not identified.

After this, we calculated TTR and K for the three corpora before and after spelling error correction. We tested two ways of correcting spelling errors as described in Sect. 2. Namely, we corrected all 13 types of errors and also only a part of them. We excluded mistakenly concatenated and split words from spelling errors. The former are word form errors that should be spelled with more than one word token as in *highschool* $\rightarrow$ *high school*. The latter are those that should spelled as one word token as in *grand father* $\rightarrow$ *grandfather*. We then compared the resulting TTR and K with those for the original corpora.

We used the following definitions for TTR and K. Let N be the total number of words in the target corpus, V be the number of distinct words in it. Then, TTR is defined as

$$\text{TTR} = \frac{V}{N}. \tag{1}$$

Also, let $V(m, N)$ $(1 \leq m \leq N)$[4] be the number of words appearing m times in a corpus whose total number of words is N. Then, K is defined as

$$K = c[-\frac{1}{N} + \sum_{m=1}^{N} V(m, N)(\frac{m}{N})^2] \tag{2}$$

where c is a constant enlarging of the value of K. Note that the larger the value of TTR is, the richer the vocabulary is and that the opposite holds for K. Also note that the symbols introduced here will be used again to discuss the results in Sect. 5.

[3]The average was calculated for spelling errors falling into SP error types, excluding those whose correct forms were unknown.

[4]Theoretically, m ranges between one and N. In practice, however, $V(m, N)$ tends to be zero for large values of m.

Error code	Explanation	Treatment
SP	Spelling that does not exist in English. e.g., I am a *sistem* engineer.	✓
PC	Inappropriate plural form conjugation. e.g., I didn't do *anythings*.	✓
OC	Over-regularized morphology. e.g., I *gived* her her hat.	✓
GC	Conjugation error other than the above two. e.g., I am *driveing*.	✓
NM	Spelling error in names. e.g., I went to *Desneyland*.	✓
RE	Real word spelling error (i.e., context sensitive error). e.g., *Their* is a house.	×
RO	Romanized Japanese. e.g., I ate an *omusubi*.	-
SR	Romanized Japanese that has no equivalent English expression. e.g., I went to *Hukuoka*. or that becomes proper English if transliterated. e.g., I ate *susi*.(susi → sushi).	-
CW	Coined word that is not used in English. e.g., I want to be a *nailist*.	-
FW	Foreign words other than Japanese. e.g., I have an *Arbeit*.	-
AL	Non-American (e.g. British English) spelling. e.g., It's my *favourite*.	-
AB	Improper abbreviation that is not used in English. e.g., I went to *USJ*.	-
O	Other than the above.	-

Table 2: Spelling Error Taxonomy and its Treatment: Each symbol denotes as follows: ✓: corrected; ×: not corrected; -: not counted.

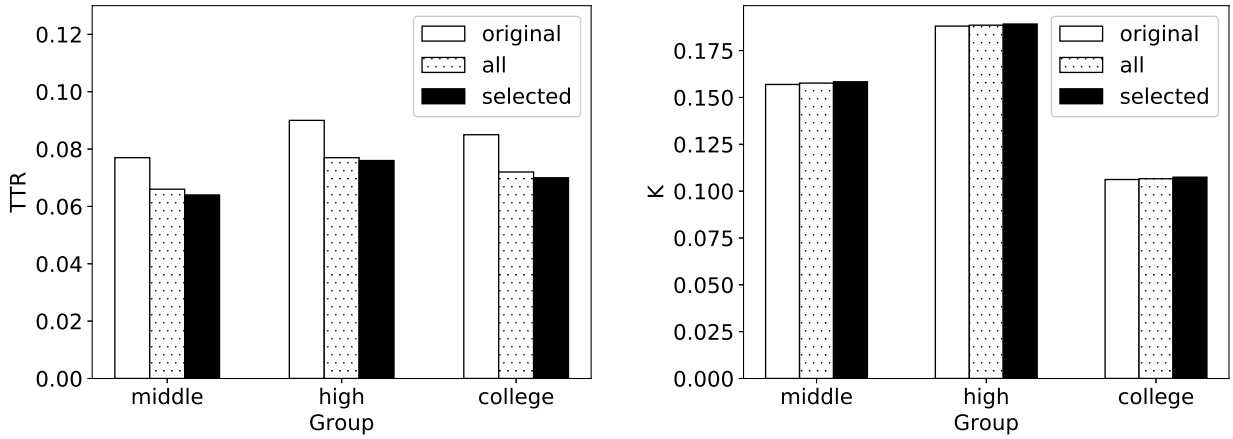
4 Results

Figure 1 shows the results. Figures 1–(a) and 1–(b) show the differences in TTR and K, respectively. The horizontal and vertical axes correspond to the three groups of the writers and the values of TTR and K, respectively. The labels *original*, *all*, and *selected* refer to the original corpus, the one where all 13 types of spelling errors are corrected, and the one where spelling errors are partly corrected, respectively.

Figure 1–(a) reveals that across the three groups, the difference in TTR is relatively large. For example, even the smallest difference, which is between *original* and *all* in *college*, approximately amounts to 16%. In *selected*, because the total number of words are slightly different (because of the not-counted spelling errors), one should be careful about its comparison with the other two. Having said that, it would be safe to say that there are at least some differences in TTR between them. These results imply that one would get varying values of TTR regardless of how he or she corrects spelling errors.

Here, note that although one might expect the values of TTR to decrease in order of *original*, *selected*, and *all*, it is not the case in the results. This is because there exist not-counted spelling errors in *selected*, which decreases both the number of distinct words and the total number of words.

Contrary to our expectation, Fig. 1–(b) reveals that almost no difference is observable in K. This applies to all groups with both ways of spelling error treatment; the difference is not more than 1% in all cases. This empirically shows that K is highly stable with respect to spelling errors.



(a): Difference in TTR.

(b): Difference in K.

Figure 1: Differences in TTR and K caused by Spelling Errors.

To sum up, the results give double answers to the hypothesis introduced in Sect. 1. TTR becomes unstable in learner English because of spelling errors. In contrast, this is not the case with K. In the next section, we explore the results theoretically and empirically to deepen our understanding of these phenomena.

5 Discussion

In this section, we explore why the influence of spelling errors is large in TTR and is very small in K. We begin with TTR, which is relatively easy to analyze, and then move on to K.

To begin with, let us assume the following situation. Suppose we have a spelling error-free corpus. Further suppose it undergoes a certain filter (or noise), which replaces a certain amount of words with misspellings (just as in the noisy channel model). The filter could be the learners' language device in their brain. Hereafter, we will refer to the error-free corpus and the erroneous corpus as *original corpus* and *error corpus*, respectively.

Now let us introduce some more symbols in addition to those defined in Sect. 3 to discuss TTR and K in more detail. Let $\mathbb{W}$ be the set of words appearing in the original corpus, $f(w)$ be the frequency of the word $w \in \mathbb{W}$, and n be the number of new spellings (words) occurring from the spelling errors in the error corpus. Here, it should be emphasized that the total number of words N is the same for the original and error corpora[5].

With these symbols, we can denote the number of distinct words (or spellings) in the error corpus as $V + n$. Strictly, words that occur only once in the corpus in question (i.e., hapax legomena) do not increase the number of distinct words or in other words do not contribute to $V + n$. Excluding spelling errors in these hapax legomena, it follows that the number of distinct spellings newly occurring from spelling errors directly affects TTR. For example, if the number of distinct spelling doubles, the increase in the number of distinct words also doubles. Also, $V + n$ shows that the frequencies of each spelling error do not affect TTR at all.

The discussion so far can be extended to other lexical richness measures that are based on the number

[5]This is because we assume that correctly spelt words in the original corpus are replaced with spelling errors.

of distinct words (i.e., V). For instance, one can apply the exact same discussion as above to Guiraud's R, which is defined as $\frac{V}{\sqrt{N}}$. The difference becomes relatively small in the measures that take the logarithm of V, but it is still relatively large compared to K (as we will show in the next paragraphs). For instance, Herdan's C, which is defined as $\frac{\log V}{\log N}$, suffers from the influence of n in the logarithm.

Now let us move on to K. The definition of K given by Eq. (2) can be rewritten as

$$K = c[-\frac{1}{N} + \sum_{w \in \mathbb{W}} (\frac{f(w)}{N})^2]. \tag{3}$$

Here, the summation is taken over the word set $\mathbb{W}$ in Eq. (3) whereas it is over the frequency of m in Eq. (2).

From Eq. (3), we can see that K is based on $f(w)^2$, i.e., the second power of each word frequency. This implies that even if n doubles in the original corpus because of spelling errors, that does not necessarily mean that their influence will double. Also, the influence from highly frequent words is dominant in the difference in K because of the second power.

We can further discuss the influence of each word. Suppose that a spelling error was created from $w \in \mathbb{W}$ and that $100r\%$ of all occurrences of w underwent the noise and were replaced with it. Then, the contribution of w to the entire value of K decreases by:

$$\frac{1}{N^2}\{(1-r)f(w)\}^2. \tag{4}$$

On the other hand, the newly created spelling increases the value of K by:

$$\frac{1}{N^2}\{rf(w)\}^2. \tag{5}$$

In total, the difference caused by a spelling error can generally be written as:

$$\frac{1}{N^2}[\{(1-r)f(w)\}^2 + \{rf(w)\}^2] = \frac{1}{N^2}\{(1-r)^2 + r^2\}f(w)^2. \tag{6}$$

Accordingly, it follows that the influence is only dependent on r. The difference becomes a maximum when $r = \frac{1}{2}$. At the same time, the values of r are expected to be small for most words. In other words, it is expected that a word is spelt correctly most of the time and only a few are misspelled[6]. If this is true, the following approximation holds:

$$\frac{1}{N^2}\{(1-r)^2 + r^2\}f(w)^2 = \frac{1}{N^2}(1 - 2r + 2r^2)f(w)^2 \tag{7}$$

$$\approx \frac{1}{N^2}(1-2r)f(w)^2. \tag{8}$$

This expression immediately shows that the difference caused by one type of spelling error is negligible when r is small. Even if r is relatively large, its influence becomes relatively small with respect to the whole value of K considering the term has the coefficient $\frac{1}{N^2}$. The discussion can be extended to the general situation where t spellings are created from $w \in \mathbb{W}$. The influence becomes maximum when $r = \frac{1}{2}$ (or generally, $r = \frac{1}{t+1}$). However, one can safely expect that it will rarely occur in frequent words in learner corpora (or corpora in general). For example, it would be rare to encounter a situation where half of the occurrences of a frequent word (e.g., *the*) are spelt correctly and the rest mistakenly (e.g., *hte*). It may happen to infrequent words, but again they scarcely affect the whole value of K because the influence of highly frequent words is dominant in K as we have just discussed at the beginning of this section.

We actually estimated r from the three corpora. It turned out that its mean and standard deviation were 0.06 and 0.10 for words where $10 \leq f(w) < 100$; similarly, 0.01 and 0.024 for words where

[6]Strictly, it can be restated that every word has, correctly or incorrectly, its representative spelling. Namely, it is assumed that $1 - r \ll 1$ or $r \ll 1$ holds.

$f(w) \geq 100$. These observations agree with our expectation and empirically explain why K is stable with respect to spelling errors.

The discussion above reconfirms the empirical findings that the difference caused by spelling errors is large in TTR and is negligible in K. Whether or not the difference in TTR can be problematic depends on the purpose and/or the way it is used. However, one should always keep in mind the fact that its difference is (at least superficially) large when one uses it. For example, the difference might cause a problem to a machine learning-based classifier when it is used as a feature; actually, some researchers (Pilán et al., 2016; Tack et al., 2017) are aware of this and thus they correct spelling errors as a preprocessing in their applications. In contrast, there is much less concern about the use of K. Besides, it has a nice property that it is also stable with respect to text length as Kimura and Tanaka-Ishii (2011) show. Consequently, it follows that it would be more accurate to use K instead of TTR when one analyzes learner English.

6 Conclusions

In this paper, we have addressed the influence of spelling errors on lexical variation measures. Specifically we have explored the following hypothesis:

> **Hypothesis**: Lexical variation measures become unstable in learner English because of spelling errors.

to augment the findings Granger and Wynne (1999) showed. We have tested it for TTR and Yule's K using three groups of learner English.

As a result, we have found that the hypothesis holds for TTR. To be precise, we have revealed that the difference in its value caused by spelling errors is relatively high throughout the tree groups, observing not less than 16% difference.

In contrast, this is not the case with K, which shows no more than 1% difference throughout the three groups. In other words, it is highly stable with respect to spelling errors.

We have investigated the results in detail to reveal why such a counterintuitive phenomenon occurs. We have shown theoretical and empirical reasons why new spellings derived from correct word forms by spelling errors directly affect the value of TTR, but not the value of K. Based on the results and the discussion, we have concluded that it would be more accurate to use K instead of TTR when one measures the lexical variation of learner English.

It will be interesting to test the present hypothesis on English texts written by learners other than Japanese. Also, it will be interesting to do the same for other languages than English. While our findings suggest that it will likely hold for other learner Englishes and also for other languages, mother tongue interference or the nature of other languages might affect the results. In particular, in certain languages (e.g., Japanese), it becomes much harder to identify spelling errors. Accordingly, the future work should include how to identify spelling errors in such languages. We will investigate these research questions in our future work.

Acknowledgments

We would like to thank Pierre Zweigenbaum for proofreading the abstract in French. We would also like to thank the three anonymous reviewers for their valuable comments on our paper.

References

Pierrre J.L. Arnaud. 1984. The lexical richness of L2 written productions and the validity of vocabulary tests. In *Proc. of International Symposium on Language Testing*, pages 14–28.

Yigal Attali and Jill Burstein. 2006. Automated essay scoring with E-rater v.2.0. *The Journal of Technology, Learning, and Assessment*, 4(3):3–30.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of 2014 Conference on Empirical Methods in Natural Language Processing*, pages 740–750.

Sylviane Granger and Martin Wynne, 1999. *Optimising measures of lexical variation in EFL learner corpora*, pages 249–257. Corpora Galore. Rodopi.

Carmen Gregori-Signes and Begoñ Clavel-Arroitia. 2015. Analysing lexical density and lexical diversity in university. In *Proc. of 7th International Conference on Corpus Linguistics: Current Work in Corpus Linguistics: Working with Traditionally-conceived Corpora and Beyond*, pages 546–556.

Pierre Guiraud. 1959. *Problèmes Et Méthodes De La Statistique Linguistique*. D. Reidel Publishing Company, Dordrecht.

Gustav Herdan. 1960. *Type-Token Mathematics: A Textbook of Mathematical Linguistics*. The Hague: Mouton, Amsterdam.

Shin'ichiro Ishikawa. 2015. Lexical development in L2 English learners' speeches and writings. In *Proc. 7th International Conference on Corpus Linguistics: Current Work in Corpus Linguistics: Working with Traditionally-conceived Corpora and Beyond*, pages 202–210.

Daisuke Kimura and Kumiko Tanaka-Ishii. 2011. A study on constants of natural language texts. *Journal of Natural Language Processing*, 18(2):119–137.

David D. Malvern, Brian J. Richards, Ngoni Chipere, and Pilar Duràn. 2004. *Lexical Diversity and Language Development*. Palgrave Macmillan, London.

Ildikó Pilán, Elena Volodina, and Torsten Zesch. 2016. Predicting proficiency levels in learner writings by transferring a linguistic complexity model from expert-written coursebooks. In *Proc. of 26th International Conference on Computational Linguistics*, pages 2101–2111.

Anaïs Tack, Thomas François, Sophie Roekhaut, and Cédrick Fairon. 2017. Human and automated CEFR-based grading of short answers. In *Proc. of 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 169–179.

Kumiko Tanaka-Ishii and Shunsuke Aihara. 2015. Computational constancy measures of texts–Yule's K and Rèyi's entropy. *Computational Linguistics*, 1(3).

Zdislava Šišková. 2012. Lexical richness in EFL students' narratives. *Language Studies Working Papers*, 4:26–36.

G. U. Yule. 1944. *The Statistical Study of Literary Vocabulary*. Cambridge.

Stance Detection with Hierarchical Attention Network

Qingying Sun[1,3], **Zhongqing Wang**[1], **Qiaoming Zhu**[1,2] and **Guodong Zhou**[1]
[1]School of Computer Science and Technology, Soochow University, Suzhou, China
[2]Institute of Artificial Intelligence, Soochow University, Suzhou, China
[3]Huaiyin Normal University, Huaian, China
qingying.sun@foxmail.com, wangzq.antony@gmail.com,
{qmzhu, gdzhou}@suda.edu.cn

Abstract

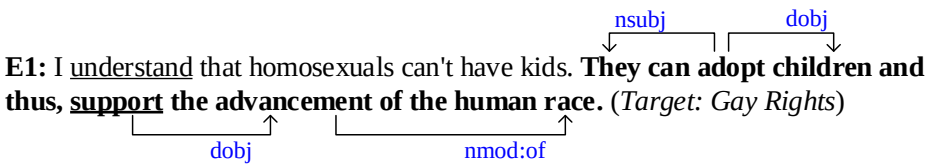
Stance detection aims to assign a stance label (*for* or *against*) to a post toward a specific target. Recently, there is a growing interest in using neural models to detect stance of documents. Most of these works model the sequence of words to learn document representation. However, much linguistic information, such as polarity and arguments of the document, is correlated with the stance of the document, and can inspire us to explore the stance. Hence, we present a neural model to fully employ various linguistic information to construct the document representation. In addition, since the influences of different linguistic information are different, we propose a hierarchical attention network to weigh the importance of various linguistic information, and learn the mutual attention between the document and the linguistic information. The experimental results on two datasets demonstrate the effectiveness of the proposed hierarchical attention neural model.

1 Introduction

While a lot of works on document-level opinion mining have involved determining the polarity expressed in a customer review (Pang et al., 2008; Liu, 2012), researchers have begun exploring new opinion mining tasks in recent years. One such task is stance detection: given a post written for a two-sided topic discussed in an online debate forum, determine which of the two sides (i.e., *for* and *against*) its author is taking. Stance detection system can potentially have a positive social impact and are of practical interest to non-profits, governmental organizations, and companies.

Previously, some of the related researches used feature engineering to extract either linguistic (Somasundaran and Wiebe, 2010) or structure (Sridhar et al., 2015) features manually. Recently, neural models have achieved high success and obtained the best results on stance detection (Zarrella and Marsh, 2016; Du et al., 2017).

A key issue of the neural model is document representation, and most of previous works learn document representation from word sequence using Convolutional Neural Networks (CNNs) (Chen and Ku, 2016; Vijayaraghavan et al., 2016) or Recurrent Neural Networks (RNNs) (Augenstein et al., 2016; Du et al., 2017).



However, besides the word sequence, stance is correlated with some explicit and implicit linguistic factors. Take E1 for example, both sentimental words (i.e., understand, support) and argument sentence (i.e., They can adopt ...) support the *favor* stance toward the target "Gay Rights". In addition, the dependency pair, $(adopt, children)$ and $(advancement, race)$, also express a positive stance. Hence,

Proceedings of the 27th International Conference on Computational Linguistics, pages 2399–2409
Santa Fe, New Mexico, USA, August 20-26, 2018.

we should learn the document representation by fully employing these linguistic factors (i.e., sentiment, argument, dependency) into neural model.

Moreover, the influence of different linguistic factors of each document is different. For example, the argument sentence in E1 is more important than sentiment and dependency information, and indicates the stance toward the target directly. Hence, we should measure the importance of different linguistic factors for each document through the neural model.

To address the above challenges, we propose a hierarchical attention model, which stands for the mutual attention between the document and the linguistic factors. The model contains two parts: linguistic attention part and hyper attention part. The former helps learn flexible and adequate document representation with different linguistic feature set, and the latter helps adjust the weight of different feature sets. In summary, the contributions are as follows.

- We present a novel hierarchical attention based neural model tailored to stance detection task, which considers the mutual influence between the representation of documents and the corresponding feature sets.
- We make a systematic comparison of LSTM with different linguistic features using the same benchmarks, and we explore influence of different linguistic features on neural models for stance detection.
- Experimental results on two open datasets demonstrate the effectiveness of the proposed approach.

2 Related Works

Previous works on stance detection have focused on congressional debates (Thomas et al., 2006; Burfoot et al., 2011), company-internal discussions (Agrawal et al., 2003), and debates in online forums (Somasundaran and Wiebe, 2010; Anand et al., 2011). Recently, there is a growing interest in performing stance detection on microblogs (Mohammad et al., 2016; Du et al., 2017). Most of existing works on stance detection can be separated into three categories: using linguistic features, using structure features, and using neural models. In the following subsections, we discuss the works on these three categories one by one.

2.1 Linguistic Features

Mosts of previous works employed various kinds of linguistic features for stance detection. For example, Somasundaran and Wiebe (2010) developed a baseline for stance detection by modeling verbs and sentiments. Anand et al. (2011) augmented the n-gram features with lexicon-based and dependency-based features. Except the above transitional linguistic factors, Hasan and Ng (2014) considered the importance of argument for stance detection, and explored the relations between stance and argument, and proposed a pipeline system to predict the stance and extract argument sentence from documents jointly.

2.2 Structure Features

Although linguistic features show effectiveness for stance detection in many works, stance detection has been newly posed as collective classification task with extra structure information. For example, citation structure (Burfoot et al., 2011) or rebuttal links (Walker et al., 2012), was used as extra information to model agreements or disagreements in debate posts and to infer their labels. Moreover, since similar users should express a similar opinion toward the same topic, user information is always used in stance detection. For example, Murakami and Raymond (2010) proposed a maximum cut method to aggregate stances in multiple posts to infer a user's stance on the target. Rajadesingan and Liu (2014) used a retweet-based label propagation method, which started from a set of opinionated users and labeled tweets by the people who are in the retweet network. Sridhar et al. (2015) utilized Probabilistic Soft Logic (PSL) to collectively classify the stance of users and stance in posts.

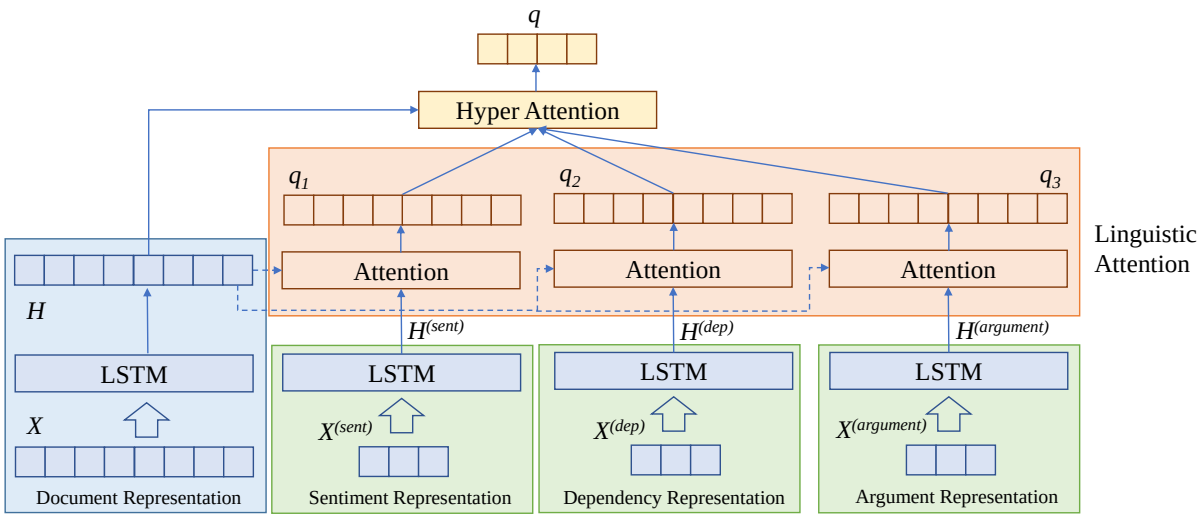


Figure 1: Overview of proposed model.

2.3 Neural Network Models

Since the deep neural network approaches can automatically extract features, there are many works using neural models to detect the stance recently, instead of using various kinds of linguistic features. For example, Augenstein et al. (2016) used two bidirectional RNN to model both target and text for stance detection, and Vijayaraghavan et al. (2016) proposed a combination of classifiers using word-level or character-level CNN models for stance detection. In addition, extra-information, especially user information, is incorporated into neural models for stance detection. For example, Chen and Ku (2016) proposed a user-based neural model to classify stance by using user tastes, topic tastes, and user comments on posts. Du et al. (2017) proposed a neural network-based model, which incorporated target-specific information into stance detection by following a novel attention mechanism.

Different from previous works, which either consider linguistic features individually, or learn the document representation on the word sequence, we propose a neural model to learn mutual attention between the document and the linguistic factors, and achieve best results.

3 Hierarchical Attention Neural Model

Formally, given a natural language document X, the stance detection system returns a binary label Y to denote the stance of the document, where $Y = 1$ denotes *favor* and $Y = 0$ denote *against* toward a special target. The architecture of our proposed hierarchical attention based stance detection system is shown in Figure 1, which illustrates the basic flow of our approach. First, we learn the representation of each document and linguistic features. Then, a hierarchical attention neural network is employed to represent the document under the influence of the different linguistic features: linguistic attention is used to learn the correlations between document representation and different linguistic feature set, and the hyper attention is employed to adjust the weight of different feature sets.

In the following subsections, we firstly show the representation of a document, and then show the representation of different linguistic features. Finally, we illustrate hierarchical attention model, and training process.

3.1 Document Representation

We use a standard Long Short-Term Memory (LSTM) model (Hochreiter and Schmidhuber, 1997) to learn the document representation. LSTM has been proven to be effective in many natural language processing (NLP) tasks such as machine translation (Sutskever et al., 2014) and dependency parsing (Dyer et al., 2015), and it is adept in harnessing long sentences. Let $X = (w_1, w_2, ..., w_n)$ be a document, where n is the document length and w_i is the i-th token. We transform each token w_i into a real-valued vector x_i using the word embedding vector of w_i, obtained by looking up a pre-trained word embedding

table D. We use the skip-gram algorithm to train embeddings (Mikolov et al., 2013).

The LSTM is used over X to generate a hidden vector sequence $(h_1, h_2, ..., h_n)$. At each step t, the hidden vector h_t of LSTM model is computed based on the current vector x_t and the previous vector h_{t-1}, and $h_t = LSTM(x_t, h_{t-1})$. The initial state and all stand LSTM parameters are randomly initialized and tuned during training. We use $H = h_n$ as the representation for X.

3.2 Representation of Linguistic Information

After obtaining the document representation, we learn the representation of different linguistic information.

Sentiment Representation

In principle, sentiment information of a post highly influences the stance. For example, the author is in favor of the target in E2, since he expresses the positive opinion toward the target. In addition, the stance toward the target and the sentiment of the post may be opposite in some posts. As in E3, the author is in favor of the given target although the post expresses the negative opinion.

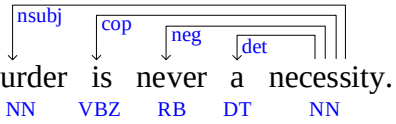
> **E2:** Hillary is our best choice if we truly want to continue being a progressive nation. (*Target: Hillary Clinton*)
>
> **E3:** When the last tree is cut down, the last fish eaten & the last stream poisoned, you will realize that you cannot eat money. (*Target: Climate Change is a Real Concern*)

Hence, it is a challenge task to employ sentiment information for stance detection, since the polarity of a post could be either the same or opposite toward the stance. In this study, we do not integrate sentiment information into document representation directly, but use attention mechanism to measure the correlation between document representation and sentiment information.

Particularly, we simply extract the sentimental word sequence $X^{(sent)} = \{x_1^s, x_2^s, ..., x_m^s\}$ of each document from sentiment lexicon, where x_i^s means sentiment word. We then learn the representation of sentiment information through this sentimental word sequence using LSTM model, and use $H^{(sent)} = LSTM(x_m^s, h_{m-1})$ as sentiment representation.

Dependency Representation

The dependency-based features can be utilized to capture the inter-word relationships (Anand et al., 2011; Persing and Ng, 2016). The dependency structure involves some stance-taking text related to the given target. As in E4, we notice that the important words that express a stance in the sentence are "murder", "never" and "necessity". By analyzing the dependency structure in this and other prompt parts, we can extract the stance-taking dependency information we identify as "murder-never-necessity".



Hence, we extract the pair of arguments sequence $X^{(dep)} = \{x_1^d, x_2^d, ..., x_m^d\} = \{x_1 \oplus x_3, ..., x_i \oplus x_j\}$ involved in each dependency relation from a dependency parser as a feature, where $x_i^d = x_j \oplus x_k$ is arguments pair from dependency relation.

We then learn the representation of dependency information through the dependency sequence $X^{(dep)}$ using LSTM model, and we use $H^{(dep)} = LSTM(x_m^d, h_{m-1})$ as dependency representation.

Argument Representation

In general, there are not only the stance of the author toward the target in a post, but also personal beliefs about what is true or what action should be taken. This personal belief is called argument (Wilson and Wiebe, 2005). There exist some arguments behind the stance people express on a specified target. If we can extract the argument sentence from the post, then it will be beneficial to detect the author's stance.

As in E5, with the help of argument sentence with bold, it will be easy to detect that the author support the legalization of marijuana.

> **E5: Most issues like this, such as sex between minors and alcohol, come down to one thing: it's your choice.** If you want to ruin your life, be my guest. It isn't the government's job to control that. (*Target: Legalization of marijuana*)

In order to utilize argument information for stance detection, we extract the argument sentence from each document, and treat argument sentence detection as a binary classification task (Hasan and Ng, 2014). After we get the argument sequence $X^{(argument)} = \{x_1^a, x_2^a, ..., x_m^a\}$ from argument sentences, where x_i^a is the i-th word in the argument sentences. We then learn the representation of argument information through the argument sequence $X^{(argument)}$ using LSTM model, and we use $H^{(argument)} = LSTM(x_m^a, h_{m-1})$ as argument representation.

3.3 Linguistic Attention

Based on our assumption, each linguistic resource should focus on different words of the same document. The extent of attention can be measured by the relatedness between each word representation $h_j \in H$ and a linguistic representation $l_i \in \{H^{(sent)}, H^{(dep)}, H^{(argument)}\}$. We propose the following formulas to calculate the weights.

$$\alpha_{ij} = \frac{\exp(w_{ij})}{\sum_{j=1}^{n} \exp(w_{ij})} \tag{1}$$

$$w_{ij} = \tanh(W^T[h_j : l_i] + b) \tag{2}$$

Here, α_{ij} denotes the weight of attention from a feature set l_i to the jth word in the document. $\tanh$ is a non-linear activation function. W is an intermediate matrix, and b is the offset. Both of them are randomly initialized and updated during training. Subsequently, according to the specific linguistic feature l_i, the attention weights are employed to calculate a weighted sum of the hidden representations, resulting in a semantic vector q_i that represents the document with the corresponding feature set.

$$q_i = \sum_{j=1}^{n} \alpha_{ij} h_j \tag{3}$$

3.4 Hyper Attention

Intuitively, different documents should value the three linguistic feature set differently. We thus define the final representation by weighting document representation and different linguistic representations $\{H, q_1, q_2, q_3\}$, as follows.

$$q = \sum_{j=1}^{4} \beta_j k_j, \quad k_j \in \{H, q_1, q_2, q_3\} \tag{4}$$

$$\beta_j = \frac{\exp(w_j)}{\sum_{j=1}^{4} \exp(w_j)} \tag{5}$$

$$w_j = \tanh(W^T V_j + b) \tag{6}$$

Here β_j denotes the attention of different document representation, indicating which document representation should be more focused. W is also an intermediate matrix, V is the weight matrix of different document representation and b is an offset value. Both of them are randomly initialized and updated during training. We use q as the final representation to learn and predict the model.

Stance	Abortion	GayRights	Obama	Marijuana
Favor (%)	54.9	63.4	53.9	69.5
Against (%)	45.1	36.6	46.1	30.5
Total	1741	1376	985	626

Table 1: Distribution of H&N14 dataset.

Stance	Atheism	Climate	Feminism	Hillary	Abortion
Favor (%)	16.9	59.4	28.2	16.6	17.9
Against (%)	63.3	4.6	53.9	57.4	58.3
None (%)	19.8	36.0	17.9	26.0	23.8
Total	733	564	949	984	933

Table 2: Distribution of SemEval16 dataset.

3.5 Training

We employ cross-entropy loss function to train the model. Specially, the loss function is defined as follows:

$$L(\Theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}\log p_{ij} + \frac{\lambda}{2}||\Theta||^2 \quad (7)$$

where n is the total number of items of training data; m is the number of category; y_{ij} indicates whether the i-th sample belongs to the j-th category, which is the ground-truth label; and p_{ij} represents the predicted probability. Θ is the set of model parameters and λ is a parameter for L2 regularization.

Standard back propagation is performed to optimize parameters. We employ Adam (Kingma and Ba, 2014) for parameter optimization and Word embeddings are trained using the *Skip-gram* algorithm (Mikolov et al., 2013)[1].

4 Experiments

4.1 Datasets and Preprocessing

We use two datasets to evaluate the performance of the proposed system. **H&N14** is collected by Hasan and Ng (2014), and **SemEval16** is from SemEval-2016 Share Task 6.A (Mohammad et al., 2016).

H&N14 is collected from an English online debate forum with four targets: "Abortion" , "Gay Rights", "Obama", and "Marijuana". The distribution of the dataset is shown in Table 1. In Hasan and Ng (2014)'s setting, the dataset is split into five folds, and they conduct five-fold cross-validation on the division. For fair comparison, we follow their setting in all experiments.

SemEval16 is the dataset for stance detection from English tweets, and each tweet corresponds to a special target: "Atheism", "Climate Change is a Real Concern"("Climate"), "Feminist Movement"("Feminist"), "Hillary Clinton"("Hillary"), and "Legalization of Abortion"("Abortion"). The distribution of the dataset is shown in Table 2. The dataset has already been separated into training and testing set (Mohammad et al., 2016).

The average length of documents in H&N14 and SemEval16 is 114 and 18, respectively. The length of document in SemEval16 is much shorter than H&N14, since SemEval16 is collected from tweets. Based on such difference of source and document length, we can evaluate the performance of the proposed model on a different setting.

Some preprocessing approaches are employed to extract sentiment, dependency, and argument information from the two datasets:

- The sentiment word sequence of each document is extracted from MPQA subjective lexicon[2].

[1] https://code.google.com/p/word2vec/

[2] http://mpqa.cs.pitt.edu/lexicons/subj_lexicon/

- The dependency sequence is extracted from each sentence by using Stanford Parser[3]. We select the relations which include noun, verb, adjective, adverb or negation words such as "nsubj", "acl", "dobj" etc. and get the argument pairs in these relations.

- The argument sentence is extracted from each document. We firstly separate the document into a sentence set based on punctuation and connective, and then we use a SVM classifier to predict the argument label of each sentence. We employ the annotated training set in (Hasan and Ng, 2014) and simplify the argument sentence detection as a binary classification task which means that sentence is just identified whether or not an argument of the related stance specifically.

Hyper-parameters

We train 300 dimensional word embedding using Word2Vec (Mikolov et al., 2013) on all the training data, and fine-turning during the training process. The dimension of LSTM layers and the output of attention mechanism is set to 300. The dropout rate is set to 0.25. The other hyper-parameters and learning rate are tuned on the development data. The development data are extracted from 10% of training data.

4.2 Evaluation Metrics

Both micro and macro average of F1-score across topics are adopted as the metrics (Mohammad et al., 2016). F1-score for Favor and Against categories of all instances is calculated as:

$$F_{favor} = \frac{2 \times P_{favor} \times R_{favor}}{P_{favor} + R_{favor}} \tag{8}$$

$$F_{against} = \frac{2 \times P_{against} \times R_{against}}{P_{against} + R_{against}} \tag{9}$$

where P and R is precision and recall. Then the average of F_{Favor} and $F_{Against}$ is calculated as the final metrics:

$$F_{avg} = \frac{F_{favor} + F_{against}}{2} \tag{10}$$

We average the F_{avg} on each topic to calculate macro-average F score($MacF_{avg}$). Besides, we calculate F_{avg} score across all targets to get micro-average F score ($MicF_{avg}$).

Note that, different from H&N14, which only contains *favor* and *against* posts, SemEval16 dataset contains *none* posts, which do not express any stance. However, the final metrics disregard the None class. Following Mohammad et al. (2016) and Du et al. (2017), we take the average F-score for Favor and Against classes, and treat None as a class that is not of interest.

4.3 Experiment Results

Comparison with Baselines

We compare our proposed model with the following state-of-the-art baseline systems:

- *SVM* is a baseline model in many previous works (Hasan and Ng, 2014; Mohammad et al., 2016). We use libSVM[4] to train the SVM model with bag-of-words features.

- *LSTM* employs word embedding, and learn the document representation through LSTM model.

- *TAN* is proposed by Du et al. (2017), it uses an attention mechanism to learn the correlation between topic and document representation. It reports the best results on SemEval16 dataset.

- *HAN* is our proposed model, which uses Hierarchical Attention Neural (HAN) model to learn the mutual attention between document and linguistic information.

Model	Abortion	GayRights	Obama	Marijuana	$MacF_{avg}$	$MicF_{avg}$
SVM	59.48	**59.52**	63.02	55.02	59.26	60.52
LSTM	60.72	56.07	60.14	55.58	58.13	59.45
TAN	**63.96**	58.13	63.00	56.88	60.49	62.35
HAN	63.66	57.36	**65.67**	**62.03**	**62.18**	**63.25**

Table 3: Comparison with baselines on H&N14 dataset.

Model	Atheism	Climate	Feminism	Hillary	Abortion	$MacF_{avg}$	$MicF_{avg}$
SVM	62.16	42.91	56.43	55.68	60.38	55.51	67.01
LSTM	58.18	40.05	49.06	61.84	51.03	52.03	63.21
TAN	59.33	**53.59**	55.77	**65.38**	63.72	59.56	68.79
HAN	**70.53**	49.56	**57.50**	61.23	**66.16**	**61.00**	**69.79**

Table 4: Comparison with baselines on SemEval16 dataset.

Model	Abortion	GayRights	Obama	Marijuana	$MacF_{avg}$	$MicF_{avg}$
SVM	59.48	59.52	63.02	55.02	59.26	60.52
SVM+Ling	62.17	58.69	65.14	56.56	60.64	62.25
LSTM	60.72	56.07	60.14	55.58	58.13	59.45
LSTM+Att+Sentiment	63.88	58.69	63.76	57.08	60.85	62.45
LSTM+Att+Dependency	63.15	59.48	64.69	59.21	61.63	62.67
LSTM+Att+Argument	62.82	58.31	64.50	61.91	61.89	63.14
HAN	63.66	57.36	65.67	62.03	62.18	63.25

Table 5: Influence of linguistic feature.

We compare the proposed HAN model with baseline systems on H&N14 and SemEval16 datasets, and the results are given in Table 3 and Table 4. From the results, we can find that SVM outperforms LSTM in these two datasets. It indicates that, simply document representation on word sequence cannot capture useful information for stance detection. TAN outperforms both SVM and LSTM, and it indicates that target information is helpful for stance detection. Moreover, our proposed model outperforms all the discrete and neural models. It shows the effectiveness of linguistic features and proposed hierarchical attention model.

4.4 Influence of Linguistic Features

We then analyze the influence of proposed different linguistic features on H&N14 dataset. The results are shown in Table 5, where *SVM+Ling* is SVM classification with all the linguistic features employed in our paper, and *LSTM+Att+** employs attention mechanism to consider the correlations between document representation and each linguistic representation individually. For example, *LSTM+Att+Sentiment* means the model just employs linguistic attention of sentiment information in Section 3.3.

From the results, we can find that SVM+Ling outperforms SVM, it proves the effectiveness of linguistic information directly. Moreover, we find that every linguistic information with attention mechanism is effective for stance detection. Especially, the argument information outperforms other features, it may be due to that argument information is highly related with stance than other linguistic information. In addition, the proposed model with all kinds of linguistic information outperforms all other models which consider each linguistic information individually.

[3]https://nlp.stanford.edu/software/lex-parser.shtml
[4]https://www.csie.ntu.edu.tw/ cjlin/libsvm/

Model	Abortion	GayRights	Obama	Marijuana	$MacF_{avg}$	$MicF_{avg}$
HAN	63.66	57.36	65.67	62.03	62.18	63.25
-Hyper	62.84	57.63	64.71	58.61	60.95	62.45
-Hyper,-Ling	62.33	56.96	63.26	57.60	60.04	61.50
LSTM	60.72	56.07	60.14	55.58	58.13	59.45

Table 6: Influence of network configuration.

4.5 Influence of Network Configurations

We study the influence of various network configurations by performing ablation experiments on H&N14 dataset, as shown in Table 6. *-Hyper* denotes ablation of the hyper attention layer, and concatenate $\{H, q_1, q_2, q_3\}$ directly and put them into a fully-connected layer. *-Hyper,-Ling* denotes ablation both of the hyper attention and linguistic attention layer, and concatenates $\{H, H^{(sent)}, H^{(dep)}, H^{(argument)}\}$ together into a fully-connected layer.

By comparing between HAN and "-Hyper", we can find that hyper attention over all the linguistic representation does have significant influence on the results. Comparison between "-Hyper" and "-Hyper, -Ling", we can find that linguistic attentions with each linguistic features does have significant improvement. On the other hand, using LSTM to detect stance (LSTM) does not obtain a better performance compared to concatenating linguistic information directly ("-Hyper, -Ling"). This confirms our intuition that stance of document is influenced by various linguistic information.

5 Conclusion

In this paper, we focus on stance detection task. We consider the impacts of different linguistic features when representing the document, and propose a novel hierarchical attention model for stance detection. Specifically, we employ a neural model to represent the documents and their linguistic features with attention mechanism. In addition, on the top of different document representation, we use an attention model to measure the importance of different linguistic features, and learn the mutual attention between the document and the linguistic information. The extensive experiments demonstrate that the proposed model achieves better performance compared with the state-of-the-art models on two datasets.

Acknowledgements

The corresponding author is Zhongqing Wang. We are grateful for the help of Jingjing Wang and Dr. Zhenghua Li for their initial discussion. We thank our anonymous reviewers for their constructive comments, which helped to improve the paper. This work is supported by the National Natural Science Foundation of China (61331011, 61751206, 61773276) and Jiangsu Provincial Science and Technology Plan (No. BK20151222).

References

Rakesh Agrawal, Sridhar Rajagopalan, Ramakrishnan Srikant, and Yirong Xu. 2003. Mining newsgroups using networks arising from social behavior. In *Proceedings of the 12th International World Wide Web Conference*, pages 529–535.

Pranav Anand, Marilyn A. Walker, Rob Abbott, Jean E. Fox Tree, Robeson Bowmani, and Michael Minor. 2011. Cats rule and dogs drool!: Classifying stance in online debate. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, pages 1–9.

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 876–885.

Clinton Burfoot, Steven Bird, and Timothy Baldwin. 2011. Collective classification of congressional floor-debate transcripts. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference*, pages 1506–1515.

Wei-Fan Chen and Lun-Wei Ku. 2016. UTCNN: a deep learning model of stance classification on social media text. In *26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers*, pages 1635–1645.

Jiachen Du, Ruifeng Xu, Yulan He, and Lin Gui. 2017. Stance classification with target-specific neural attention. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3988–3994.

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

Kazi Saidul Hasan and Vincent Ng. 2014. Why are you taking this stance? identifying and classifying reasons in ideological debates. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 751–762.

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

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

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *27th Annual Conference on Neural Information Processing Systems*, pages 3111–3119.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiao-Dan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, pages 31–41.

Akiko Murakami and Rudy Raymond. 2010. Support or oppose? classifying positions in online debates from reply activities and opinion expressions. In *23rd International Conference on Computational Linguistics*, pages 869–875.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Isaac Persing and Vincent Ng. 2016. Modeling stance in student essays. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2174–2184.

Ashwin Rajadesingan and Huan Liu. 2014. Identifying users with opposing opinions in twitter debates. In *Social Computing, Behavioral-Cultural Modeling and Prediction - 7th International Conference*, pages 153–160.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 116–124.

Dhanya Sridhar, James R. Foulds, Bert Huang, Lise Getoor, and Marilyn A. Walker. 2015. Joint models of disagreement and stance in online debate. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 116–125.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 327–335.

Prashanth Vijayaraghavan, Soroush Vosoughi, and Deb Roy. 2016. Automatic detection and categorization of election-related tweets. In *Proceedings of the 10th International Conference on Web and Social Media*, pages 703–706.

Marilyn A. Walker, Pranav Anand, Rob Abbott, and Ricky Grant. 2012. Stance classification using dialogic properties of persuasion. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, pages 592–596.

Theresa Wilson and Janyce Wiebe. 2005. Annotating attributions and private states. In *Proceedings of the Workshop on Frontiers in Corpus Annotations II: Pie in the Sky*, pages 53–60.

Guido Zarrella and Amy Marsh. 2016. MITRE at semeval-2016 task 6: Transfer learning for stance detection. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 458–463.

Correcting Chinese Word Usage Errors for Learning Chinese as a Second Language

Yow-Ting Shiue[1], Hen-Hsen Huang[1], and Hsin-Hsi Chen[1,2]

[1]Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan
[2]MOST Joint Research Center for AI Technology and All Vista Healthcare, Taipei, Taiwan
orina1123@gmail.com, hhhuang@nlg.csie.ntu.edu.tw,
hhchen@ntu.edu.tw

Abstract

With more and more people around the world learning Chinese as a second language, the need of Chinese error correction tools is increasing. In the HSK dynamic composition corpus, word usage error (WUE) is the most common error type. In this paper, we build a neural network model that considers both target erroneous token and context to generate a correction vector and compare it against a candidate vocabulary to propose suitable corrections. To deal with potential alternative corrections, the top five proposed candidates are judged by native Chinese speakers. For more than 91% of the cases, our system can propose at least one acceptable correction within a list of five candidates. To the best of our knowledge, this is the first research addressing general-type Chinese WUE correction. Our system can help non-native Chinese learners revise their sentences by themselves.

Title and Abstract in Chinese

非中文母語學習者中文寫作用詞錯誤之更正

以中文爲第二語言的學習者與日俱增，遍及全球，對於中文更正工具的需求也相應而生。在HSK動態作文語料庫中，用詞錯誤（WUE）是中文學習者最常犯的錯誤類型。在這篇論文中，針對中文用詞的更正，我們建立了專屬的神經網路模型，同時參酌目標錯誤詞彙及其前後文資訊，以產生推薦的更正方式。經過中文母語人士的評估，在91%的測試案例中，系統所推薦的前五名候選詞彙，至少有一個是適當的更正方式。據我們所知，這是目前第一個全面性的中文用詞錯誤更正之研究。我們的系統可以幫助非中文母語人士自主修正其所寫的中文句子，促進華語文教學之成效。

1 Introduction

Grammatical error correction (GEC) tools can help language learners revise their writing. Chinese GEC tools are in high demand since Chinese has become an increasingly popular second language worldwide. Despite the increasing need, most of the existing studies on GEC are based on English learner data. The method of correcting sentences in Chinese, a language which differs substantially from English in major aspects such as the morphological structure and the distribution of learner errors, has not yet been fully developed.

This paper focuses on the correction of Chinese word usage errors (WUEs). According to the definition of Shiue and Chen (2016), a WUE refers to an incorrect token that involves morphological, syntactical, or semantical problems. The token is either an incorrect word form, or a correct existent word that is improper for its context.

Given a token in a sentence segment that is known to be erroneous, we aim to generate a suitable correction for it. The criteria for a suitable correction are:

- **Correctness**: After substituting the erroneous token with the correction token, the result is a syntactically and semantically correct Chinese sentence segment.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2410–2422
Santa Fe, New Mexico, USA, August 20-26, 2018.

• **Similarity**: The meaning of the correction is close to the writer's intended meaning.

We discuss the criteria with the following example sentence segments.

(E1-1) *生活方式已經**猛烈**地改變了 (The way of living has been **fiercely** changed.)

(E1-2) *生活方式已經**暴烈**地改變了 (... has been **overpoweringly** changed.)

(E1-3) 生活方式已經**緩慢**地改變了 (... has been **slowly** changed.)

(E1-4) 生活方式已經**劇烈**地改變了 (... has been **dramatically** changed.)

For wrong segment (E1-1), (E1-2) is not a correction since it is incorrect itself. The adverb "暴烈地" (overpoweringly) does not collocate with the verb "改變" (change). (E1-3) is grammatical, but its meaning differs from the original meaning of (E1-1), so neither is it suitable. (E1-4) is a good correction that meets both criteria.

Nevertheless, there are some cases in which the similarity criterion is hard to meet. For example, the intended meaning of (E2-1) is very difficult to recognize. (E2-2) is the ground-truth correction, but the association between the original erroneous token "情緒" (emotion) and the correction "因素" (factor) is unclear.

(E2-1) *發生這種情況的**情緒**很多 (Many **emotions** happen this situation.)

(E2-2) 發生這種情況的**因素**很多 (Many **factors** can lead to this situation.)

When two criteria cannot be met at the same time, the correctness criterion should have higher priority, since an incorrect sentence can confuse the language learner.

In this paper, *target* refers to the original erroneous token written by the language learner, and *context* refers to other words in the sentence segment. A pair consisting of the target and its corresponding correction is called a *correction pair*. In example (E1) and (E2), "猛烈" and "情緒" are targets, and (猛烈, 劇烈) and (情緒, 因素) are correction pairs. According to the previous discussions, the major challenge of the WUE correction task lies in the derivation of the intended semantics and the generation of valid sentence segments.

We treat WUE correction as a candidate selection problem and propose a neural network-based model considering both target and context to rank correction candidates. Our main contributions are: (1) Though the detection of Chinese WUE and correction of certain types have been studied, this is the first research dealing with the correction of all types of WUEs. (2) To consider alternative corrections, we perform human evaluation and show that our system can propose at least one acceptable correction within the top five candidates for more than 91% of the cases. (3) We release the HSK WUE dataset with additional human annotations.

2 Related Work

English GEC is a rather mature field of study in NLP. Several shared tasks have been conducted for English GEC (Dale and Kilgarriff, 2011; Dale et al., 2012; Ng et al., 2013; Ng et al., 2014). Language models, machine learning classifiers, rule-based classifiers, and machine translation models are used. The machine translation approach has the advantage that there is no need to explicitly formulate the types of the errors. A series of English GEC studies are based on the phrase-based statistical machine translation (SMT) framework (Dahlmeier and Ng, 2011; Chollampatt et al., 2016b; Chollampatt et al., 2016a; Chollampatt and Ng, 2017).

Nevertheless, the satisfactory performance of the SMT approach cannot be reached without sufficient training data. In fact, Chollampatt et al. (2016a) have shown that the model trained with smaller training data from writers with the same first language (L1) as writers of the test data performs even worse than the model trained with larger training data from writers whose L1 differs from that of the test data. The amount of available Chinese learner data are even less sufficient than that of English ones. Therefore, as a preliminary research in Chinese WUE correction, we impose restrictions on our setting that there is exactly one error in a sentence segment, the error position is known, and the error can be corrected by replacing the erroneous token with an appropriate word.

The distribution of errors of non-native Chinese differ a lot from that of non-native English. In the CoNLL 2013 shared task (Ng et al., 2013), the most frequent error types are article, preposition, noun number, verb form, and subject-verb agreement. These error types are mostly in violation of English

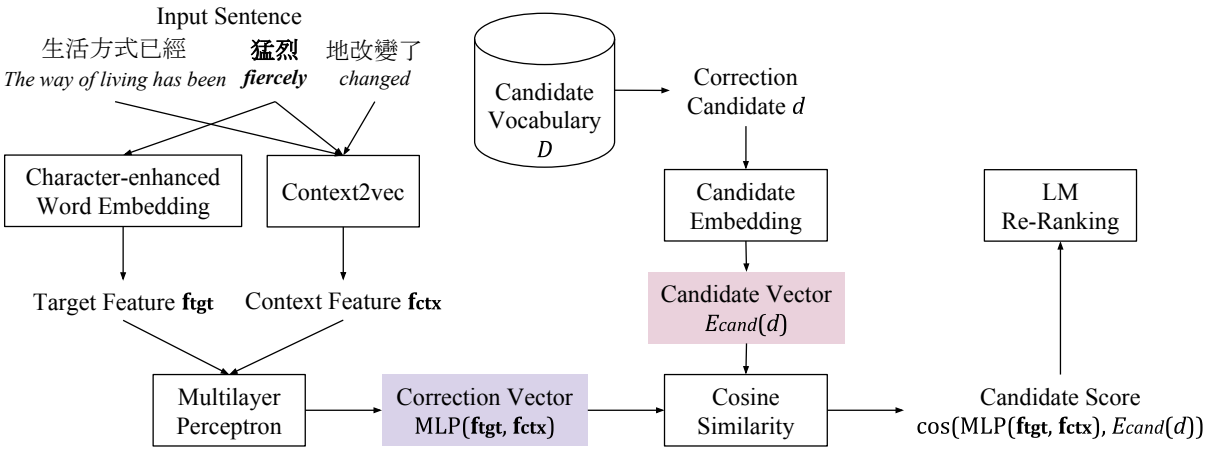


Figure 1: Overview of our correction generation model. The score of a correction candidate d is predicted for replacing the target erroneous word "猛烈" (fiercely) with the target and the context information.

grammar rules, so it is relatively easy to grasp some patterns of correction. In contrast, in the HSK dynamic composition corpus built by Beijing Language and Culture University, which is the largest available Chinese learner corpus at the time of this study, WUE is the most frequent lexical-level error. In most cases, a wrong usage of a word does not lead to violation of syntactic rules. Instead, its incorrectness cannot be determined without understanding the meaning of the whole sentence. As a result, besides directly adopting the techniques used for English GEC, many aspects of Chinese GEC are worth studying.

The Chinese spelling check task (Wu et al., 2013; Yu et al., 2014b; Tseng et al., 2015; Fung et al., 2017) evaluates the detection and correction of character errors. The Shared Task for Chinese Grammatical Error Diagnosis (Yu et al., 2014a; Lee et al., 2015; Lee et al., 2016; Rao et al., 2017) extends the above task to word errors, including redundant word, missing word, word disorder and word selection. Nevertheless, these tasks only deal with detection but not correction.

Some researchers focus on certain types of Chinese writing errors. For example, Yu and Chen (2012) identify word ordering errors (WOEs), and Cheng et al. (2014) further recommend word ordering correction candidates with the use of ranking support vector machine (RankSVM).

Previous researches on Chinese WUE include segment-level (Shiue and Chen, 2016) and token-level detection (Shiue et al., 2017). Huang et al. (2016) study the Chinese preposition selection problem, which is subsumed by WUE correction. Gated recurrent unit (GRU)-based models are trained to select the most suitable one from a closed set of 43 Chinese prepositions in a context.

Nevertheless, it is still worth investigating how to treat WUEs involving other types of words such as verbs and nouns. Correcting errors of such open-set types could be much more difficult since the set of candidates can be huge. To the best of our knowledge, this is the first research dealing with general-type Chinese WUE correction.

3 Neural Network-based Correction Generation Model

As shown in Figure 1, our correction generation model is a multilayer perceptron (MLP) neural network that takes target feature vector $\mathbf{f_{tgt}}$ and context feature vector $\mathbf{f_{ctx}}$ as input and outputs a *correction vector* MLP($\mathbf{f_{tgt}}, \mathbf{f_{ctx}}$). $\mathbf{f_{tgt}}$ and $\mathbf{f_{ctx}}$ are derived either from the segment itself or with the help of some external resources. Section 5 will give the details for several kinds of features we used.

Every candidate word d in a set D of all possible corrections is mapped to an embedding $E_{cand}(d)$. The details of candidate embedding model E_{cand} will be elaborated in Section 4. The training objective of our model is to make MLP($\mathbf{f_{tgt}}, \mathbf{f_{ctx}}$) as close to $E_{cand}(\hat{d})$ as possible, where $\hat{d}$ is the ground-truth correction. While MLP contains a bunch of trainable parameters, E_{cand} is fixed to preserve the ability to generalize the corrections that are unseen in the training data.

When testing, given the original erroneous sentence segment, our model selects the most probable correction $d*$:

$$d* = \underset{d \in D}{\operatorname{argmax}} \cos(\text{MLP}(\mathbf{f_{tgt}}, \mathbf{f_{ctx}}), E_{cand}(d)) \quad (1)$$

In fact, this model can propose several candidates, ranked by their cosine similarities to the correction vector. The goal is to rank the ground-truth correction toward the top of the list.

4 Candidate Embedding

For the candidate embedding model E_{cand}, we use the character-enhanced word embedding model (CWE), in which word vectors (hereafter WE) and character vectors (hereafter CE) are learned jointly (Chen et al., 2015). This model is developed based on one idiosyncratic characteristic of the Chinese writing system, that meanings of individual characters in a word usually contribute to the meaning of the word. Therefore, character-level information can help capture the meaning of Chinese words much better.

On the other hand, one common case of WUE is that the characters within a word is wrongly chosen or permuted. For instance, "決解" (jué jiě) is a misused form of "解決" (solve). Though the misused form is a non-existent word, its character components serve as an important clue for discovering what the writer originally means and help provide more suitable correction.

We adopt the position-based CWE (CWE+P), which keeps three embeddings for each character according to the character's position. This variant is designed to capture different morphological functions of a Chinese character when it is at different positions in a word. A character vector is denoted as $\text{CE}(c, p)$, where c is the character and $p = s, m, e$ depending on the position (start, middle or end, respectively) of c within a certain word. The word representation is the word vector plus the average of the component character vectors. For example, the representation of "農產品" (agricultural product) is $\text{CWE}_w(\text{農產品}) = \text{WE}(\text{農產品}) + \frac{1}{3}[\text{CE}(\text{農}, s) + \text{CE}(\text{產}, m) + \text{CE}(\text{品}, e)]$. If we ignore the internal structure of the word, a misused form "農作品" ('nóng zuò pǐn') is simply an out-of-vocabulary (OOV) word and has no association with the correction "農產品". With the use of CWE vectors, it is more possible for the model to learn a transformation from the incorrect token to its correction.

5 Input Features

To meet the two criteria for suitable WUE correction, we adopt several target and context features as input to the correction generation model. The division of $\mathbf{f_{tgt}}$ and $\mathbf{f_{ctx}}$ is just for indicating the source of information. These two feature vectors are concatenated and fed to the MLP model.

5.1 Target CWE+P Word Embedding

This set of features is derived from the same embedding model as we used for the candidate embedding. The way of composing a representation of an existent word is also the same. For OOV word, the representation is the average of the vectors of all characters. By doing so, among the three character vector terms in the representation of "農作品", $\text{CE}(\text{農}, s)$ and $\text{CE}(\text{品}, e)$ are shared with the correction vector $\text{CWE}_w(\text{農產品})$. The word and character vectors used to calculate this set of features are in the same space with the candidate embedding, enabling the model to directly learn a transformation between a correction pair. We use CWE_w to denote this set of features.

5.2 Target CWE Position-Insensitive Character Embedding

Although a character's position in a word could reflect its morphological function, non-native Chinese learners might not be so familiar with Chinese morphology. One common type of morphological WUE is incorrect ordering of characters within a word. For example, (*決解, 解決) is the tenth frequent correction pair in our dataset. With the use of CWE+P embeddings, the similarity between these two tokens might be underestimated since all the character vector terms are different.

To cope with this problem, we experimented with the CWE variant that only keeps one vector for each Chinese character regardless of its position, but the performance is not as good as the model with

CWE+P features. Alternatively, we design a separate set of character embedding features CWE_c which is the sum of the character embedding of all positions divided by the number of characters in the word. For instance, $\text{CWE}_c(\text{決解}) = \frac{1}{2}\sum_{p=s,m,e}[\text{CE}(\text{決}, p) + \text{CE}(\text{解}, p)]$. As can be seen, $\text{CWE}_c(\text{決解})$ will contain $\text{CE}(\text{解}, s)$ and $\text{CE}(\text{決}, e)$, which are the terms of $\text{CWE}_w(\text{解決})$.

5.3 Context2vec Features

Context2vec (Melamud et al., 2016) is a bidirectional LSTM-based model that can encode a "context" into a real-valued vector. A context is a sequence of words with a certain position blanked out. For instance, (E3-1) is a context:

(E3-1) 可是 每 個 人 的 [ ] 都 千差萬別 (but everyone's [ ] is different)

The representation of a context is a combination of the sequence of words before and after the blank:

$$\text{C2V}_{ctx}(w_1...w_{p-1}[\ ]w_{p+1}...w_L) = \text{LSTM}(w_1...w_{p-1}) \oplus \text{LSTM}(w_{p+1}...w_L) \quad (2)$$

where each w_i is a token, L is the number of tokens, p is the index of the blank, and $\oplus$ is the vector concatenation operation.

Context2vec also keeps the embeddings of individual words, which are called target embeddings[1] by Melamud et al. (2016). We use C2V_{tgt} to denote the vector of target word. Both target embeddings and the parameters in the LSTM layers are updated during training. The objective of the model is to predict the target word that actually occurs in the training sentence, given the encoded context vector.

The formulation of context makes Context2vec suitable for the sentence completion task. A candidate to fill the blank can be selected according to how similar its vector is to the context vector. That is, given a context where the p-th position is the blank, the best candidate $d*$ would be:

$$d* = \underset{d \in D}{\text{argmax}} \cos(\text{C2V}_{tgt}(d), \text{MLP}(\text{C2V}_{ctx}(w_1...w_{p-1}[\ ]w_{p+1}...w_L))) \quad (3)$$

where MLP is a non-linear projection that maps the context representation to a vector with dimensionality same as that of the target embeddings. Melamud et al. (2016) have shown the promising results of Context2vec in several sentence completion benchmarks. For the example context (E3-1), the best candidate selected in this way using our trained model is "境況", which can be put into the blank and the result is a correct sentence segment.

(E3-2) 可是 每 個 人 的 [境況] 都 千差萬別 (but everyone's [situation] is different)

In fact, (E3-1) is extracted from a wrong segment in our dataset. The original erroneous segment and the corresponding correction are shown in (E3-3).

(E3-3) 可是每個人的(*對應, 反應)都千差萬別

(but everyone's (correspondence, reaction) is different)

Given the original segment, one can conclude that the candidate "境況" (situation) selected by Context2vec is less suitable compared to the ground-truth "反應" (reaction), according to the similarity criterion. Therefore, WUE correction is different from sentence completion in that if the model ignores the word originally written by the language learner, it is likely to generate a correction that changes the meaning of the original sentence segment.

To take both context and target information into account, we include two feature vectors C2V_{tgt} and C2V_{ctx} , which belong to target and context features, respectively. C2V_{tgt} refers to the embedding of the original erroneous token, which can reveal important information about the writer's intended meaning as we discussed above.

5.4 Target POS Features

We analyze the part-of-speech (POS) tags before and after correction, and show the most frequent POS changes in the validation set in Table 1. The POS tagging is performed by using Stanford CoreNLP

[1]Note that the definition of "target" in the Context2vec paper is slightly different from ours. In our definition, target only refers to the original erroneous token, while for Context2vec, target can refer to any word to be put into the blank, regardless of whether the result is a correct sentence.

Original POS	Correction POS	# instances	Frequency
Unchanged		722	68.70%
VV	NN	27	2.57%
NN	VV	21	2.00%
P	VV	17	1.62%
DEC	DEV	15	1.43%
VV	P	13	1.24%
AD	VV	10	0.95%
VV	VA	10	0.95%

Table 1: Part-of-speech changes occurring at least 10 times.

(Manning et al., 2014). As can be seen, the POS tag does not change after the correction in nearly 70% of the cases. Besides, there are some systematic changes. For example, non-native Chinese learners seem to confuse some nouns with some verbs, so we can observe the interchanging phenomenon of the VV and NN tags. The case of VV and P is similar. These systematic changes indicate that it is possible to reduce the candidate vocabulary. We tried to limit the candidates to the POS transitions observed in the training and validation set. However, this results in slightly lower accuracy and MRR. Therefore, instead of modifying the candidate set directly, we encode the POS of the erroneous token in a one-hot vector and feed this feature to the correction generation model. This allows the model to learn different transformation function for different source POS, that is, the POS of the erroneous token.

6 Language Model Re-ranking

One drawback of our MLP correction generation model is that the correctness criterion does not explicitly take priority over the similarity criterion. In our experiments, we found that our model sometimes generates segments that seriously violate the correctness criterion, since it can bias toward the similarity criterion. (E4) is an example.

Wrong segment:

(E4-1) *到山頂**之間**路走得不容易 (The road **between** the hilltop was not easy to walk.)

Model prediction:

(E4-2) *到山頂**期間**路走得不容易 (The road **period** the hilltop ...)

Ground-truth correction:

(E4-3) 到山頂**的**路走得不容易 (The road **to** the hilltop ...)

The candidate "期間" (period) is selected since it is similar to the target "之間" (between), but the result sentence segment is incorrect. It is necessary to deal with this problem since the correctness criterion is more important, as we previously discussed in the introduction.

It is expected that this kind of unsuitable candidates can be eliminated by a language model (LM), if we assume that LM probability reflects the level of correctness of a sentence segment. Therefore, we emphasize the correctness criterion by incorporating LM scores of traditional n-gram LM or Recurrent Neural Network Language Model (RNNLM) (Mikolov et al., 2011) into the candidate selection process. One possible approach is to apply a probability cut-off and discard candidates that result in segments with low probability. Nevertheless, the "acceptable" LM probability varies from sentence to sentence since it can be affected by, for instance, length and lexical complexity of the sentence. Though we can let the cutoff be a function of various factors, it is difficult to design such a function explicitly.

Therefore, instead of performing combination of scores, we combine the rank proposed by the LM with the rank based on our correction generation model. One advantage of this approach is that the range of ranks is the same for all instances given fixed candidate vocabulary size, so the ranks can be evaluated with the same standard across different instances. For a candidate correction, let r_{LM} be its rank based on the LM probability, and r_{MLP} be the rank based on its cosine similarity to the correction vector generated by our MLP model. We adopt the following weighted harmonic mean to obtain a new "rank" for the candidate.

$$r_{com} = \frac{1}{\frac{\alpha}{r_{LM}} + \frac{1-\alpha}{r_{MLP}}} \tag{4}$$

where α is a parameter that can be tuned with the validation set (actual values will be given in Section 8.2). Preliminary experiments show that harmonic mean performs better than arithmetic and geometric mean. Though r_{com} may not be an integer, it can be interpreted as a rank. The correction with smaller r_{com} is considered better.

7 Experimental Settings

We adopt the HSK WUE dataset released by Shiue et al. (2017)[2] and follow their train/validation/test split. We use Stanford CoreNLP (Manning et al., 2014) for Chinese word segmentation and POS tagging. For each split, we filter the instances where the correction is not within the top 50,000 frequent words in the Chinese part of the ClueWeb corpus[3] (Yu et al., 2012). This decision is made based on the fact that the vocabulary used by non-native language learners is limited. After filtering, the number of instances in the train, validation, and test sets are 8,205, 1,026, and 1,025, respectively. With punctuation marks and English words eliminated, the candidate vocabulary size $|D|$ is 48,394.

Our MLP model has two hidden layers of size 1,024. The activation function is Rectified Linear Unit (ReLU) and the dropout rate is set to 0.2. The parameters are optimized with Adagrad (Duchi et al., 2011) under a cosine proximity objective function. CWE (embedding size 400), Context2vec (300 units), 5-gram LM and RNNLM (size-128 GRU) are all trained on Chinese ClueWeb. Please refer to Appendix A for detailed parameter settings.

8 Automatic Evaluation

8.1 Result before LM Re-ranking

We first evaluate our correction generation model with the single ground-truth correction per segment. The results are shown in Table 2. The baselines include the two LMs and the Context2vec sentence completion method, which selects a candidate most similar to the context but ignores the original erroneous token. The 5-gram LM is the strongest baseline for the WUE correction task.

The second part of Table 2 shows the result of a set of experiments with Context2vec features. The MLP model with only C2V$_{ctx}$ features differs from the Context2vec baseline in that it is trained with the WUE dataset. Learning a transformation from the erroneous token to the correction seems to be easier than guessing a correction only from context, probably because some common correction pairs can be learned. The model using only C2V$_{tgt}$ achieves performance substantially better than that using only C2V$_{ctx}$. Note that the Context2vec vectors does not lay in the same space with the CWE+P vectors we used for E_{cand}. This indicates that our model is indeed capable of learning a transformation, not just copying the vector terms from the input features. Combining C2V$_{tgt}$ and C2V$_{ctx}$ can further enhance the performance.

The third part of Table 2 shows another set of experiments, in which different kinds of features are included incrementally starting from the CWE+P target features. The last row indicates the performance when all features are used. The model with CWE$_w$ features performs better than that with C2V$_{tgt}$, since CWE+P composes a vector representation for OOV targets such as "農作品", giving the model more clues for generating the correction. The position-insensitive character feature CWE$_c$ slightly improves the performance over CWE$_w$. After including Context2vec context and target features, the model can consider the context and reaches accuracy 0.3512. Finally, incorporating POS information further enhances the accuracy to 0.3717 ($p < 0.05$ significance) and MRR to 0.4378.

8.2 Effect of LM Re-ranking

We apply LM re-ranking to the candidate ranks generated by the best correction generation model. The results of two alternative LMs are shown in Table 3. Although there is only slight improvement on the accuracy, the MRR and hit rates increase substantially after LM re-ranking is applied. The 5-gram LM

[2]http://anthology.aclweb.org/attachments/P/P17/P17-2064.Datasets.zip ; corresponding ground-truth corrections retrieved from: http://202.112.195.192:8060/hsk/login.asp

[3]http://www.lemurproject.org/clueweb09/

	Target	Context	Accuracy	MRR	Hit@5	Hit@10	Hit@50
Baselines (No training on the WUE dataset)	-	5-gram LM	0.1659	0.2438	0.3268	0.4029	0.5951
	-	RNNLM	0.1468	0.2208	0.2847	0.3611	0.5793
	-	$C2V_{ctx}$	0.0714	0.1170	0.1575	0.2114	0.3611
Correction Generation Model with Context2vec Features	-	$C2V_{ctx}$	0.1249	0.1746	0.2273	0.2741	0.4010
	$C2V_{tgt}$	-	0.2507	0.3030	0.3561	0.3932	0.5024
	$C2V_{tgt}$	$C2V_{ctx}$	0.3249	0.3891	0.4566	0.4976	0.6185
Correction Generation Model with CWE + Other Features	CWE_w		0.2898	0.3545	0.4195	0.4693	0.5971
	+ CWE_c		0.2946	0.3570	0.4234	0.4722	0.6078
	+ $C2V_{tgt}$	+ $C2V_{ctx}$	0.3512	0.4250	0.5024	0.5571	0.6800
	+ POS		0.3717	0.4378	0.5063	0.5688	0.6956

Table 2: Performance of the correction generation model with various target and context features.

Model	Accuracy	MRR	Hit@5	Hit@10	Hit@50	Hit@100
Best MLP	0.3717	0.4378	0.5063	0.5688	0.6956	0.7415
+ 5-gram LM	**0.3727**	**0.4605**	**0.5561**	**0.6439**	**0.8039**	0.8488
+ RNNLM	**0.3727**	0.4527	0.5278	0.6205	0.7808	0.8302

Table 3: Performance with LM re-ranking.

gives slightly better MRR than RNNLM. The optimal α for 5-gram LM and RNNLM are 0.355 and 0.255, respectively.

(E5) is an example in which LM re-ranking helps promote the rank of the answer. Though sharing a common Chinese character, the meaning of "一起" (together) and "一直" (always, all the time) are not quite similar. Thus, the MLP rank is very low. In contrast, the LM rank is high, since "就...都不..." (have always not...) is a suitable context for "一直". The higher combined rank leads to enhanced MRR.

(E5) 我從上小學起成績就(*一起, 一直)都不理想

(Since I began elementary school, my grade has (together, always) been unsatisfactory.)

$r_{LM} = 7, r_{MLP} = 1284 \rightarrow$ Combined rank: 19

9 Human Evaluation

In the automatic evaluation, there is only one answer for each test instance. However, correction can be highly subjective and alternatives may exist. Moreover, since the HSK WUE dataset is composed of sentence segments, the model has no access to the context outside of the segment. This results in difficulties in making the choice among several candidate corrections that are different in meaning but all seem to be acceptable. Table 4 shows an example. The erroneous token "定心" ('dìng xīn') is not a valid noun in Chinese. The meaning of character "定" is related to "stable, fixed", and the meaning of "心" is related to "mind". The top five candidates proposed by our system are all differ from the ground-truth correction. However, except for the rank 3 candidate, all other four candidates are acceptable corrections. This example shows that the single-answer automatic evaluation can underestimate the performance of our system. Therefore, we perform human evaluation, in which the top candidates proposed by our model are judged by annotators.

Wrong segment	不過我們要以堅定的**定心**與病對抗 (but we should fight against the disease with strong **'dìng xīn'**.)
System 1st	不過我們要以堅定的**自信**與病對抗 (... with strong **self-confidence**.)
System 2nd	不過我們要以堅定的**信念**與病對抗 (... with strong **faith**.)
System 3rd	不過我們要以堅定的**理智**與病對抗 (... with strong **rationality**.)
System 4th	不過我們要以堅定的**自信心**與病對抗 (... with strong **sense of self-confidence**.)
System 5th	不過我們要以堅定的**毅力**與病對抗 (... with strong **persistence**.)
Ground-truth	不過我們要以堅定的**決心**與病對抗 (... with strong **determination**.)

Table 4: An example of alternative corrections.

9.1 Annotation Guideline

Each instance of annotation consists of two sentence segments:

(S0): the original wrong segment

(S1): a correction segment, which is either the ground-truth or one of the top k proposed candidates

We set $k = 5$; however, only the candidates ranked before the ground-truth need annotation. For example, if our system gives rank 3 to the ground-truth correction, only the ground-truth, the first candidate and the second candidate need to be judged by human. For the 1,025 test segments, a total of 3,692 annotation instances are generated.

An annotator is asked to answer (at most) two questions for each annotation instance. The questions and the instructions given to the annotators are shown in Table 5. The answers to both questions are either Yes (1) or No (0). If the answer to Q1 is No, the answer to Q2 must be No, since (S1) violates the correctness criterion; we will skip Q2 in such cases. As long as the meaning of (S1) is similar to that of (S0), the annotator should answer Yes, regardless of whether (S0) is incorrect. In case the meaning of (S0) is not understandable, as illustrated by (E2-1), the annotator should answer Yes. This corresponds to our previous claim that the correctness criterion is more important.

Q1 (*is_g*): Is (S1) a correct sentence segment?
1. If (S1) is ungrammatical, please answer No. 2. If (S1) is grammatical but its semantics is not logical, or violates some common-sense knowledge, please also answer No. 3. Since we split sentences into segments by punctuation marks, if you encounter an "incomplete" sentence, please answer Yes to it, if it is itself correct and can be completed in some reasonable way.
Q2 (*is_c*): Is (S1) a correction of (S0)?
1. This question will be presented only if you answered Yes to Q1, which means the grammaticality of candidate correction (S1) has been confirmed. 2. If the meaning of (S1) is the intended meaning of (S0), please answer Yes; otherwise please answer No.

Table 5: WUE correction annotation instructions.

There are 10 annotators who are native speakers of Chinese. They are properly trained by giving examples selected from the validation set for every case in the instructions. Each annotation instance is assigned to two annotators randomly, so we can assume that the difficulty of the data assigned to each annotator is the same. Therefore, we calculate the average inter-annotator agreement of all pairs of annotators. The average Cohen's Kappa for Q1 and Q2 are 0.4205 and 0.4070 respectively, showing moderate level of agreement. If the two annotators disagree on either question, a third annotator is introduced to break the tie. We use majority voting to determine the final answer.

9.2 Evaluation with Human Annotation

Total 95.60% of the ground-truth corrections and 82.73% of the system top candidates are judged as correct by the annotators. This indicates that the grammaticality of the system output is acceptable. We use the updated rank of the test instances, which is the rank of the highest ranked candidate that meets both correctness and similarity criteria according to annotation results, to re-evaluate our best model. The performance before and after applying annotation results are shown in Table 6.

As can be seen, there is large performance increase in all metrics, verifying the existence of alternative corrections. Both accuracy and MRR increase by more than 30%. Moreover, the hit@5 rate is above 91%, which means that for most of the test data, at least one of the top five candidates is an acceptable correction. A language learner can choose the one that is close to his or her intended meaning from the list of candidates. In fact, only the writer knows the exact "intended meaning", so it is nearly impossible for a system to guess the right meaning all the time. We argue that a fairly short list of candidates can be helpful for learning to write in foreign languages.

Evaluation	Accuracy	MRR	Hit@5	Hit@10	Hit@50	Hit@100
Ground-truth	0.3727	0.4605	0.5561	0.6439	0.8039	0.8488
+ Annotation	0.6829	0.7784	0.9122	0.9171	0.9502	0.9600

Table 6: Human evaluation results.

10 Error Analysis

We analyze the human evaluation result according to different POS tags of the erroneous token. The results of POS tags that occur more than 20 times are shown in Table 7. The most frequent POS tags, VV, NN and AD, which are open-set word types, contribute the most difficult cases. The accuracy is less than 70% and MRR is less than 80%. On the other hand, for closed-set word types such as prepositions (P), our system performs very well, reaching accuracy 0.81 and MRR 0.88. DEV is the POS tag of Chinese adverb marker "地". The marker inherits very regular usage, and regular pattern of learner errors, so the system can achieve perfect performance. This also indicates that our system is capable of handling various kind of errors, and there is no need to include a separate rule-based module for a specific error type.

POS	# Tests	Accuracy	MRR	Hit@5	Hit@10	Mean rank	Std.
VV	316	0.67	0.77	0.91	0.92	26.12	6.75
NN	277	0.64	0.73	0.88	0.88	73.97	11.50
AD	130	0.65	0.75	0.88	0.89	96.16	13.41
P	62	0.81	0.88	0.95	0.95	3.10	1.92
VA	45	0.60	0.76	0.98	0.98	1.98	1.08
DEV	23	1.00	1.00	1.00	1.00	1.00	0.00
PN	21	0.71	0.80	0.95	0.95	2.33	1.40

Table 7: Performance on most frequent POS tags.

11 Conclusion

In this paper, given a Chinese sentence segment with a known error position, we aim to generate correction candidates that not only result in a correct segment, but also preserve the original meaning of the writer. Our MLP correction generation model takes target and context features as input and outputs a correction vector, which can be compared against the vectors of the candidate vocabulary. We apply LM re-ranking to put emphasis on correctness, avoiding misleading corrections.

In the single-ground-truth automatic evaluation, we achieve accuracy 0.3727 and MRR 0.4605. With human evaluation, in which the top five proposed candidates are judged by native Chinese speakers to include some alternative acceptable corrections, the accuracy increases to 0.6829 and MRR to 0.7784. Moreover, the hit@5 rate reaches 0.9122. Since a list of five candidates is rather short, a language learner can choose a suitable one from the candidate list and revise his or her sentences even without the help of a language teacher.

To enable future investigations, the dataset we used is attached. We have dealt with semantic similarity and similarity in overlapping Chinese characters in this paper. Nevertheless, we have not handled phonetical similarity. For example, in the correction pair (影響, 印象), the source of confusion is that the pronunciation of "影響" (influence 'yǐng xiǎng') and "印象" (impression 'yìn xiàng') are very similar. Pronunciation information can serve as clues for selecting the suitable correction.

Acknowledgements

This research was partially supported by Ministry of Science and Technology, Taiwan, under grants MOST-105-2221- E-002-154-MY3, MOST-106-2923-E-002-012-MY3 and MOST-107-2634-F-002-011-.

A Detailed Model Settings

A.1 MLP Model Parameters

We implement our MLP correction generation model with Keras (Chollet, 2015). Table 8 shows the parameters. Models with this setting generally perform the best on the validation set across different combinations of features. The activation function is not applied at the output layer, so that the model output can fit better to the candidate embedding of the ground-truth correction.

When the validation accuracy does not increase for two consecutive epochs, the training process is terminated. In most cases, our model converges in 5 to 9 epochs. We choose the model with the highest validation accuracy for each feature combination to evaluate on the test set.

Parameter	Value
Hidden layer size	4096
Number of hidden layers	2
Activation function	ReLU
Dropout rate	0.2
Cost function	cosine proximity
Optimizer	Adagrad
Initial learning rate	0.01
Batch size	32

Table 8: Parameter settings of our MLP model.

A.2 CWE Parameters

We use the publicly released implementation of CWE[4] to train the CWE+P model on the Chinese ClueWeb corpus. We set the embedding size to 400 and train for 20 iterations. All other hyperparameters are left default.

A.3 Context2vec Parameters

We use the publicly available toolkit[5] to obtain the Context2vec context and target representation model. The training is also performed on the Chinese ClueWeb corpus. We set the number of units to 300 and train for 5 epochs.

A.4 N-gram LM Settings

We fit a 5-gram language model with KenLM[6] on Chinese ClueWeb. The modified Kneser-Ney smoothing (Heafield et al., 2013) is applied to handle unseen n-grams.

A.5 RNNLM Parameters

We use the Faster RNNLM toolkit[7], which speeds up the training process of the original RNNLM by using the Hierarchical Softmax (HS) or Noise Contrastive Estimation (NCE). We choose NCE because it gives better performance on our WUE validation set.

We process the ClueWeb corpus before training. The words whose frequency is less than 10 are replaced with a "<unk>" token. When testing, an OOV word is treated as "<unk>". We split 10% of the corpus for validation. The toolkit automatically adjusts the learning rate and early-stops the training process based on validation entropy. The hyperparameter settings are shown in Table 9.

Hyperparameter	Value
Layer type	GRU
Layer size	128
Number of negative samples	20

Table 9: Hyperparameter settings of our RNNLM.

References

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huan-Bo Luan. 2015. Joint learning of character and word embeddings. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1236–1242.

[4]https://github.com/Leonard-Xu/CWE
[5]https://github.com/orenmel/context2vec
[6]https://github.com/kpu/kenlm
[7]https://github.com/yandex/faster-rnnlm

Shuk-Man Cheng, Chi-Hsin Yu, and Hsin-Hsi Chen. 2014. Chinese word ordering errors detection and correction for non-native chinese language learners. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 279–289. Dublin City University and Association for Computational Linguistics.

Shamil Chollampatt and Hwee Tou Ng. 2017. Connecting the dots: Towards human-level grammatical error correction. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 327–333.

Shamil Chollampatt, Duc Tam Hoang, and Hwee Tou Ng. 2016a. Adapting grammatical error correction based on the native language of writers with neural network joint models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1901–1911. Association for Computational Linguistics.

Shamil Chollampatt, Kaveh Taghipour, and Hwee Tou Ng. 2016b. Neural network translation models for grammatical error correction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2768–2774.

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Daniel Dahlmeier and Hwee Tou Ng. 2011. Correcting semantic collocation errors with l1-induced paraphrases. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 107–117. Association for Computational Linguistics.

Robert Dale and Adam Kilgarriff. 2011. Helping our own: The hoo 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 242–249. Association for Computational Linguistics.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. Hoo 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 54–62. Association for Computational Linguistics.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Gabriel Fung, Maxime Debosschere, Dingmin Wang, Bo Li, Jia Zhu, and Kam-Fai Wong. 2017. Nlptea 2017 shared task – chinese spelling check. In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)*, pages 29–34, Taipei, Taiwan, December. Asian Federation of Natural Language Processing.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified kneser-ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 690–696. Association for Computational Linguistics.

Hen-Hsen Huang, Yen-Chi Shao, and Hsin-Hsi Chen. 2016. Chinese preposition selection for grammatical error diagnosis. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 888–899. The COLING 2016 Organizing Committee.

Lung-Hao Lee, Liang-Chih Yu, and Li-Ping Chang. 2015. Overview of the nlp-tea 2015 shared task for chinese grammatical error diagnosis. In *Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2015)*, pages 1–6. Association for Computational Linguistics.

Lung-Hao Lee, Gaoqi Rao, Liang-Chih Yu, Endong Xun, Baolin Zhang, and Li-Ping Chang. 2016. Overview of nlp-tea 2016 shared task for chinese grammatical error diagnosis. In *Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2016)*, pages 40–48. The COLING 2016 Organizing Committee, December.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *CoNLL*, pages 51–61.

Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and Jan Cernocky. 2011. Rnnlm-recurrent neural network language modeling toolkit. In *Proceedings of the 2011 ASRU Workshop*, pages 196–201.

Tou Hwee Ng, Mei Siew Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The conll-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12. Association for Computational Linguistics.

Tou Hwee Ng, Mei Siew Wu, Ted Briscoe, Christian Hadiwinoto, Hendy Raymond Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14. Association for Computational Linguistics.

Gaoqi Rao, Baolin Zhang, Endong Xun, and Lung-Hao Lee. 2017. Ijcnlp-2017 task 1: Chinese grammatical error diagnosis. In *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 1–8, Taipei, Taiwan, December. Asian Federation of Natural Language Processing.

Yow-Ting Shiue and Hsin-Hsi Chen. 2016. Detecting word usage errors in chinese sentences for learning chinese as a foreign language. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 220–224. European Language Resources Association (ELRA), may.

Yow-Ting Shiue, Hen-Hsen Huang, and Hsin-Hsi Chen. 2017. Detection of chinese word usage errors for non-native chinese learners with bidirectional lstm. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 404–410.

Yuen-Hsien Tseng, Lung-Hao Lee, Li-Ping Chang, and Hsin-Hsi Chen. 2015. Introduction to sighan 2015 bake-off for chinese spelling check. *ACL-IJCNLP 2015*, page 32.

Shih-Hung Wu, Chao-Lin Liu, and Lung-Hao Lee. 2013. Chinese spelling check evaluation at sighan bake-off 2013. In *Proceedings of the 7th SIGHAN Workshop on Chinese Language Processing*, pages 35–42.

Chi-Hsin Yu and Hsin-Hsi Chen. 2012. Detecting word ordering errors in chinese sentences for learning chinese as a foreign language. In *Proceedings of COLING 2012*, pages 3003–3018. The COLING 2012 Organizing Committee.

Chi-Hsin Yu, Yi jie Tang, and Hsin-Hsi Chen. 2012. Development of a web-scale chinese word n-gram corpus with parts of speech information. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC 2012)*, pages 320–324. European Language Resources Association (ELRA), may.

Liang-Chih Yu, Lung-Hao Lee, and Li-Ping Chang. 2014a. Overview of grammatical error diagnosis for learning chinese as a foreign language. In *Proceedings of the 1st Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2014)*, pages 42–47.

Liang-Chih Yu, Lung-Hao Lee, Yuen-Hsien Tseng, Hsin-Hsi Chen, et al. 2014b. Overview of sighan 2014 bake-off for chinese spelling check. In *Proceedings of the 3rd CIPSSIGHAN Joint Conference on Chinese Language Processing (CLP'14)*, pages 126–132.

Retrofitting Distributional Embeddings to Knowledge Graphs with Functional Relations

Benjamin J. Lengerich
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
blengeri@cs.cmu.edu

Andrew L. Maas and **Christopher Potts**
Stanford University, Roam Analytics
195 East 4th Avenue
San Mateo, CA 94401
{amaas, cgpotts}@roaminsight.com

Abstract

Knowledge graphs are a versatile framework to encode richly structured data relationships, but it can be challenging to combine these graphs with unstructured data. Methods for retrofitting pre-trained entity representations to the structure of a knowledge graph typically assume that entities are embedded in a connected space and that relations imply similarity. However, useful knowledge graphs often contain diverse entities and relations (with potentially disjoint underlying corpora) which do not accord with these assumptions. To overcome these limitations, we present *Functional Retrofitting*, a framework that generalizes current retrofitting methods by explicitly modeling pairwise relations. Our framework can directly incorporate a variety of pairwise penalty functions previously developed for knowledge graph completion. Further, it allows users to encode, learn, and extract information about relation semantics. We present both linear and neural instantiations of the framework. Functional Retrofitting significantly outperforms existing retrofitting methods on complex knowledge graphs and loses no accuracy on simpler graphs (in which relations do imply similarity). Finally, we demonstrate the utility of the framework by predicting new drug–disease treatment pairs in a large, complex health knowledge graph.

1 Introduction

Distributional representations of concepts are often easy to obtain from unstructured data sets, but they tend to provide only a blurry picture of the relationships that exist between concepts. In contrast, knowledge graphs directly encode this relational information, but it can be difficult to summarize the graph structure in a single representation for each entity.

To combine the advantages of distributional and relational data, Faruqui et al. (2015) propose to *retrofit* embeddings learned from distributional data to the structure of a knowledge graph. Their method first learns entity representations based solely on distributional data and then applies a retrofitting step to update the representations based on the structure of a knowledge graph. This modular approach conveniently separates the distributional data and entity representation learning from the knowledge graph and retrofitting model, allowing one to flexibly combine, reuse, and adapt existing representations to new tasks.

However, a core assumption of Faruqui et al. (2015)'s retrofitting model is that connected entities should have similar embeddings. This assumption often fails to hold in large, complex knowledge graphs, for a variety of reasons. First, subgraphs of a knowledge graph often contain distinct classes of entities that are most naturally embedded in disconnected vector spaces. In the extreme case, the representations for these entities might derive from very different underlying data sets. For example, in a health knowledge graph, the subgraphs containing diseases and drugs should be allowed to form disjoint vector spaces, and we might want to derive the initial representations from radically different data sets. Second, many knowledge graphs contain diverse relationships whose semantics are different from – perhaps even in conflict with – similarity. For instance, in the knowledge graph in Figure 1, the model

Proceedings of the 27th International Conference on Computational Linguistics, pages 2423–2436
Santa Fe, New Mexico, USA, August 20-26, 2018.

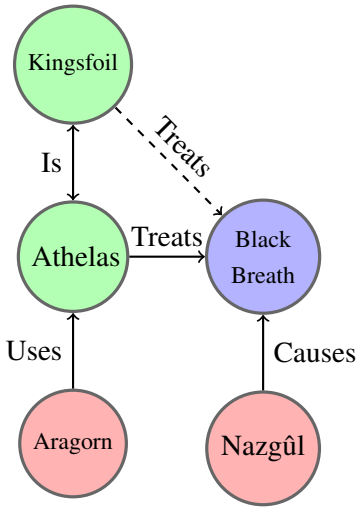


Figure 1: Toy knowledge graph with diverse relation types that connect treatments (green), diseases (blue), and persons (red) by known (solid) and unknown (dashed) relations. Traditional methods, which assume that all relations imply similarity, would retrofit Aragorn and Nazgûl toward similar embeddings.

of Faruqui et al. (2015) would model embeddings $\boldsymbol{q_{Aragorn}} \approx \boldsymbol{q_{Athelas}} \approx \boldsymbol{q_{BlackBreath}} \approx \boldsymbol{q_{Nazgûl}}$, which is problematic as Aragorn is not semantically similar to a Nazgûl (they are enemies).

To address these limitations, we present *Functional Retrofitting*, a retrofitting framework that explicitly models pairwise relations as functions. The framework supports a wide range of different instantiations, from simple linear relational functions to complex multilayer neural ones. Here, we evaluate both linear and neural instantiations of Functional Retrofitting on a variety of diverse knowledge graphs. For benchmarking against existing approaches, we use FrameNet and WordNet. We then move into the medical domain, where knowledge graphs play an important role in knowledge accumulation and discovery. These experiments show that even simple instantiations of Functional Retrofitting significantly outperform baselines on knowledge graphs with semantically complex relations and sacrifice no accuracy on graphs where Faruqui et al. (2015)'s assumptions about similarity do hold. Finally, we use the model to identify promising new disease targets for existing drugs.

Code which implements Functional Retrofitting is available at `https://github.com/roaminsight/roamresearch`.

2 Notation

A knowledge graph $\mathcal{G}$ is composed of a set of vertices $\mathcal{V}$, a set of relation types $\mathcal{R}$, and a set of edges $\mathcal{E}$ where each edge $e \in \mathcal{E}$ is a tuple (i, j, r) in which the relationship $r \in \mathcal{R}$ holds between vertices $i \in \mathcal{V}$ and $j \in \mathcal{V}$. Our goal is to learn a set of representations $\mathcal{Q} = \{\boldsymbol{q_i} : i \in \mathcal{V}\}$ which contain the information encoded in both the distributional data and the knowledge graph structure, and can be used for downstream analysis. Throughout this paper, we use a to refer to a scalar, $\boldsymbol{a}$ to refer to a vector, and $\boldsymbol{A}$ to refer to a matrix or tensor.

3 Related Work

Here we are interested in *post-hoc* retrofitting methods, which adjust entity embeddings to fit the structure of a previously unseen knowledge graph.

3.1 Retrofitting Models

The primary introduction of retrofitting was Faruqui et al. (2015), in which the authors showed the value of retrofitting semantic embeddings according to minimization of the weighted least squares problem

$$\Psi_{\mathcal{G}}(\mathcal{Q}) = \sum_{i \in \mathcal{V}} \alpha_i \left\| \boldsymbol{q_i} - \hat{\boldsymbol{q}}_{\boldsymbol{i}} \right\|^2 + \sum_{(i,j,r) \in \mathcal{E}} \beta_{ij} \left\| \boldsymbol{q_i} - \boldsymbol{q_j} \right\|^2 \tag{1}$$

where $\hat{\mathcal{Q}} = \{\hat{\boldsymbol{q}}_{\boldsymbol{i}} : i \in \mathcal{V}\}$ is the embedding learned from the distributional data and α_i, β_{ij} set the relative weighting of each type of data. When $\alpha_i = 1$ and $\beta_{ij} = \frac{1}{degree(i)}$, this model assigns equal weight to the distributional data and the structure of the knowledge graph.

More recently, Hamilton et al. (2017) presented GraphSAGE, a two-step method which learns both an aggregation function $f : \mathbb{R}^{d \times n} \rightarrow \mathbb{R}^k$, to condense the representations of neighbors into a single point, and an update function $g : \mathbb{R}^{k+d} \rightarrow \mathbb{R}^d$, to combine the aggregation with a central vertex. Here, d is the embedding dimensionality, k is the aggregation dimensionality, and $n > 0$ is the number of neighbors for each vertex. Note that $k > d$ is permitted, allowing for aggregation by concatenation. While this method is extremely effective for learning representations on simple knowledge graphs, it is not formulated for knowledge graphs with multiple types of relations. Furthermore, when the representation of a relation is known *a priori*, it can be useful to explicitly set the penalty function (e.g., Mrkšic et al. (2016) use hand-crafted functions to effectively model antonymy and synonymy). By aggregating neighbors into a point estimate before calculating relationship likelihoods, GraphSAGE makes it difficult to encode, learn, or extract the representation of a pairwise relation.

In a similar vein, Faruqui et al. (2016) developed a graph-based semi-supervised learning method to expand morpho-syntactic lexicons from seed sets. Though the task is different from the retrofitting task we consider here, the performance and scalability of their method demonstrate the utility of directly encoding pairwise relations as message-passing functions.

3.2 Relational Penalty Functions

Our new Functional Retrofitting framework models each relation via a penalty function $f_r : \mathbb{R}^{d_i+d_j} \rightarrow \mathbb{R}_{\geq 0}$ acting on a pair of entities (i, j) with embedding dimensionalities d_i and d_j, respectively. By explicitly modeling relations between pairs of entities, Functional Retrofitting supports the use of a wide array of scoring functions that have previously been developed for knowledge graph completion. Here, we present a brief review of such scoring functions; for an extensive review, see (Nickel et al., 2016).

TransE (Bordes et al., 2013) uses additive relations in which the penalty function $f_r(\boldsymbol{q_i}, \boldsymbol{q_j}) = \|\boldsymbol{q_i} + \boldsymbol{a_r} - \boldsymbol{q_j}\|_2^2$ is low iff $(i, j, r) \in \mathcal{E}$. The simple Unstructured Model (Bordes et al., 2012) was proposed as a naïve version of TransE that assigns all $\boldsymbol{a_r} = \mathbf{0}$, leading to the penalty function $f_r(\boldsymbol{q_i}, \boldsymbol{q_j}) = \|\boldsymbol{q_i} - \boldsymbol{q_j}\|_2^2$. This is the underlying penalty function of (Faruqui et al., 2015). It cannot consider multiple types of relations. In addition, while it models 1-to-1 relations well, it struggles to model multivalued relations.

TransH (Wang et al., 2014) was proposed to address this limitation by using multiple representations for a single entity via relation hyperplanes. For a relation r, TransH models the relation as a vector $\boldsymbol{a_r}$ on a hyperplane defined by normal vector $\boldsymbol{w_r}$. For a triple $(i, j, r) \in \mathcal{E}$, the entity embeddings $\boldsymbol{q_i}$ and $\boldsymbol{q_j}$ are first projected to the hyperplane of $\boldsymbol{w_r}$. By constraining $\|\boldsymbol{w_r}\|_2 = 1$, we have the penalty function $f_r(\boldsymbol{q_i}, \boldsymbol{q_j}) = \|g_r(\boldsymbol{q_i}) + \boldsymbol{a_r} - g_r(\boldsymbol{q_j})\|_2^2$ where $g_r(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{w_r}^T \boldsymbol{x} \boldsymbol{w_r}$.

TransR (Lin et al., 2015) embeds relations in a separate space from entities by a relation-specific matrix $\boldsymbol{M_r} \in \mathbb{R}^{d \times k}$ that projects from entity space to relation space and a shared relation vector $\boldsymbol{a} \in \mathbb{R}^k$ that translates in relation space by $f_r(\boldsymbol{q_i}, \boldsymbol{q_j}) = \|\boldsymbol{q_i}\boldsymbol{M_r} + \boldsymbol{a} - \boldsymbol{q_j}\boldsymbol{M_r}\|_2^2$. We use this model as the inspiration for our linear penalty function.

The Neural Tensor Network (NTN; Socher et al. (2013)) defines a score function $f_r(\boldsymbol{q_i}, \boldsymbol{q_j}) = \boldsymbol{u_r}^T g(\boldsymbol{q_i}^T \boldsymbol{M_r} \boldsymbol{q_j} + \boldsymbol{M_{r,1}} \boldsymbol{q_i} + \boldsymbol{M_{r,2}} \boldsymbol{q_j} + \boldsymbol{b_r})$ where $\boldsymbol{u_r}$ is a relation-specific linear layer, $g : \mathbb{R}^k \rightarrow \mathbb{R}^k$ is the tanh operation applied element-wise, $\boldsymbol{M_r} \in \mathbb{R}^{d \times d \times k}$ is a 3-way tensor, and $\boldsymbol{M_{r,1}}, \boldsymbol{M_{r,2}} \in \mathbb{R}^{k \times d}$ are weight matrices. All of these models can be directly incorporated in our Functional Retrofitting framework.

4 Functional Retrofitting

We propose the framework of Functional Retrofitting (FR) to incorporate a set $\mathcal{F}$ of relation-specific penalty functions $f_r : \mathbb{R}^{d_i+d_j} \rightarrow \mathbb{R}_{\geq 0}$ which penalizes embeddings of entities i, j with dimensionalities

d_i, d_j, respectively. This gives the complete minimization:

$$\Psi_{\mathcal{G}}(\mathcal{Q};\mathcal{F}) = \sum_{i\in\mathcal{Q}} \alpha_i ||\boldsymbol{q_i} - \hat{\boldsymbol{q}}_{\boldsymbol{i}}||^2 + \sum_{(i,j,r)\in\mathcal{E}} \beta_{i,j,r} f_r(\boldsymbol{q_i}, \boldsymbol{q_j}) - \sum_{(i,j,r)\in\mathcal{E}^-} \beta_{i,j,r} f_r(\boldsymbol{q_i}, \boldsymbol{q_j}) + \sum_{r\in\mathcal{R}} \rho_\lambda(f_r) \tag{2}$$

where $\hat{\boldsymbol{q}}_{\boldsymbol{i}}$ is observed from distributional data, α_i and $\beta_{i,j,r}$ set the relative strengths of the distributional data and the knowledge graph structure, and ρ regularizes f_r with strength set by λ. $\mathcal{E}^-$ is the *negative space* of the knowledge graph, a set of edges that are not annotated in the knowledge graph. FR uses $\mathcal{E}^-$ to penalize relations that are implied by the representations but not annotated in the graph. To populate $\mathcal{E}^-$, we sample a single negative edge (i, j', r) with the same outgoing vertex for each true edge $(i, j, r) \in \mathcal{E}$. The user can calibrate trust in the completeness of the knowledge graph via the β hyperparameter.

In contrast to prior retrofitting work, FR explicitly encodes directed relations. This allows the model to fit graphs which contain diverse relation types and entities embedded in disconnected vector spaces. Here, we compare the performance of two instantiations of FR – one with all linear relations and one with all neural relations – and show that even these simple models provide significant performance improvements. In practice, we recommend that users select relation-specific functions in accordance with the semantics of their graph's relations.

4.1 Linear Relations

We implement a linear relational penalty function $f_r(\boldsymbol{q_i}, \boldsymbol{q_j}) = \|\boldsymbol{A_r q_j} + \boldsymbol{b_r} - \boldsymbol{q_i}\|^2$ with ℓ_2 regularization for minimization of:

$$\begin{aligned}\Psi_{\mathcal{G}}(\mathcal{Q};\mathcal{F}) = &\sum_{i=1}^{n} \alpha_i \left\|\boldsymbol{q_i} - \hat{\boldsymbol{q}}_{\boldsymbol{i}}\right\|^2 + \sum_{(i,j,r)\in\mathcal{E}} \beta_{i,j,r} \left\|\boldsymbol{A_r q_j} + \boldsymbol{b_r} - \boldsymbol{q_i}\right\|^2 \\ &- \sum_{(i,j,r)\in\mathcal{E}^-} \beta_{i,j,r} \left\|\boldsymbol{A_r q_j} + \boldsymbol{b_r} - \boldsymbol{q_i}\right\|^2 + \lambda \sum_{r\in\mathcal{R}} ||\boldsymbol{A_r}||^2 \end{aligned} \tag{3}$$

Identity Relations

Faruqui et al. (2015)'s model is a special case of this formulation in which

$$\boldsymbol{A_r} = \boldsymbol{I}, \quad \boldsymbol{b_r} = \boldsymbol{0} \quad \forall r, \qquad \beta_{i,j,r} = \begin{cases} \frac{1}{degree(i)} & (i,j,r) \in \mathcal{E} \\ 0 & (i,j,r) \in \mathcal{E}^- \end{cases}$$

Throughout the remainder of this paper, we refer to this baseline model as the "FR-Identity" retrofitting method.

Initialization

We initialize embeddings as those learned from distributional data and relations to imply similarity:

$$\boldsymbol{A_r} = \boldsymbol{I}, \quad \boldsymbol{b_r} = \boldsymbol{0} \quad , \quad \alpha_i = \begin{cases} 0 & \hat{\boldsymbol{q}}_{\boldsymbol{i}} = \boldsymbol{0} \\ \alpha & \hat{\boldsymbol{q}}_{\boldsymbol{i}} \neq \boldsymbol{0} \end{cases}, \quad \beta_{i,j,r} = \begin{cases} \frac{\beta^+}{d_r(i)} & (i,j,r) \in \mathcal{E} \\ \frac{\beta^-}{d_r(i)} & (i,j,r) \in \mathcal{E}^- \end{cases}$$

where $d_r(i)$ is the out-degree of vertex i for relation type r, α is a hyperparameter to trade off distributional data against structural data, and β sets the trust in completeness of the knowledge graph structure. In our experiments, we use $\beta^+ = 1,\ \beta^- = 0$ for straightforward comparison with the method of Faruqui et al. (2015) and optimize α by cross-validation. Given prior knowledge about the semantic meaning of relations, we could initialize relations to respect these meanings (e.g., antonymy could be represented by $\boldsymbol{A_r} = -\boldsymbol{I}$).

Learning Procedure

We optimize this model by block optimization. Conveniently, we have closed-form solutions where the partial derivatives of Eq. 3 equal 0:

$$\boldsymbol{b_r} = \frac{\sum_{(i,j)} (-1)^{I_{\{(i,j,r)\notin\mathcal{E}\}}} \beta_{i,j,r}(\boldsymbol{A_r q_j} - \boldsymbol{q_i})}{\sum_{(i,j)} (-1)^{I_{\{(i,j,r)\notin\mathcal{E}\}}} \beta_{i,j,r}} \tag{4}$$

$$\tilde{\boldsymbol{A}}_{\boldsymbol{r}} = \boldsymbol{U}\boldsymbol{V}^{-1} \tag{5}$$

$$\boldsymbol{U} = \sum_{(i,j):(i,j,r)\in\mathcal{E}} \beta_{i,j,r}(\boldsymbol{q_i} - \boldsymbol{b_r})\boldsymbol{q_j}^T - \sum_{(i,j):(i,j,r)\in\mathcal{E}^-} \beta_{i,j,r}(\boldsymbol{q_i} - \boldsymbol{b_r})\boldsymbol{q_j}^T \tag{6}$$

$$\boldsymbol{V} = \sum_{(i,j):(i,j,r)\in\mathcal{E}} \beta_{i,j,r}\boldsymbol{q_j}\boldsymbol{q_j}^T - \sum_{(i,j):(i,j,r)\in\mathcal{E}^-} \beta_{i,j,r}\boldsymbol{q_j}\boldsymbol{q_j}^T + \lambda\boldsymbol{I} \tag{7}$$

Constraining $\boldsymbol{A_r}$ to be orthogonal by $\boldsymbol{A_r} = \tilde{\boldsymbol{A}}_{\boldsymbol{r}}(\tilde{\boldsymbol{A}}_{\boldsymbol{r}}^T\tilde{\boldsymbol{A}}_{\boldsymbol{r}})^{-1/2}$, we have $\boldsymbol{q_i} = \frac{\boldsymbol{a_i}}{b_i}$ where

$$\begin{aligned}\boldsymbol{a_i} = \alpha_i\hat{\boldsymbol{q}}_{\boldsymbol{i}} &+ \sum_{(j,r):(i,j,r)\in\mathcal{E}} \beta_{i,j,r}(\boldsymbol{A_r q_j} + \boldsymbol{b_r}) + \sum_{(j,r):(j,i,r)\in\mathcal{E}} \beta_{j,i,r}\boldsymbol{A_r}^T(\boldsymbol{q_j} - \boldsymbol{b_r}) \\ &- \sum_{(j,r):(i,j,r)\in\mathcal{E}^-} \beta_{i,j,r}(\boldsymbol{A_r q_j} + \boldsymbol{b_r}) - \sum_{(j,r):(j,i,r)\in\mathcal{E}^-} \beta_{j,i,r}\boldsymbol{A_r}^T(\boldsymbol{q_j} - \boldsymbol{b_r})\end{aligned} \tag{8}$$

$$b_i = \alpha_i + \sum_{(j,r):(i,j,r)\in\mathcal{E}} \beta_{i,j,r} + \sum_{(j,r):(j,i,r)\in\mathcal{E}} \beta_{j,i,r} - \sum_{(j,r):(i,j,r)\in\mathcal{E}^-} \beta_{i,j,r} - \sum_{(j,r):(j,i,r)\in\mathcal{E}^-} \beta_{j,i,r} \tag{9}$$

4.2 Neural Relations

We also instantiate FR with a neural penalty function $f_r(\boldsymbol{q_i}, \boldsymbol{q_j}) = \sigma(\boldsymbol{q_i}^T\boldsymbol{A_r q_j})$ where σ is the element-wise tanh operation, $\boldsymbol{A_r} \in \mathbb{R}^{d_i \times d_j}$, again with ℓ_2 regularization.We initialize weights in a similar manner as for the linear relations and update via stochastic gradient descent. In our experiments, we use $\beta^+ = \beta^- = 1$, and sample the same number of non-neighbors as true neighbors.

5 Experiments

We test FR on four knowledge graphs. The first two are standard lexical knowledge graphs (FrameNet, WordNet) in which FR significantly improves retrofitting quality on complex graphs and loses no accuracy on simple graphs. The final two graphs are large healthcare ontologies (SNOMED-CT, Roam Health Knowledge Graph), which demonstrate the scalability of the framework and the utility of the new embeddings.

For each graph, we successively evaluate link prediction accuracy after retrofitting to links of other relation types. Specifically, for each relation type $r \in \mathcal{R}$, we retrofit to $\mathcal{G}_{\setminus r} = (\mathcal{V}, \mathcal{E}_{\setminus r})$ where $\mathcal{E}_{\setminus r} = \{(i,j,r') : (i,j,r') \in \mathcal{E}, r' \neq r\}$ is the set of edges with all relations of type r removed. After retrofitting, we train a Random Forest classifier to predict the presence of relation r between entities i and j (with 70% of vertices selected as training examples and the remainder reserved for testing). To have balanced class labels, we sample an equivalent number of non-edges, $\mathcal{E}_r^- = \{(i,j,r) : (i,j,r) \notin \mathcal{E}\}$ with $|\mathcal{E}_r^-| = |\mathcal{E}|$ and $|\{j : (i,j,r) \in \mathcal{E}_r^-\}| = |\{j : (i,j,r) \in \mathcal{E}\}| \; \forall \; i$. Thus, the random baseline classification rate is set to 50%. Other baselines are the embeddings built from distributional data and the retrofitting method of Faruqui et al. (2015), denoted as "None" and "FR-Identity", respectively.

5.1 FrameNet

FrameNet (Baker et al., 1998; Fillmore et al., 2003) is a linguistic knowledge graph containing information about lexical and predicate argument semantics of the English language. FrameNet contains two distinct entity classes: *frames* and *lexical units*, where a *frame* is a meaning and a *lexical unit* is a single meaning for a word. To create a graph from FrameNet, we connect lexical unit i to frame j if i occurs

in j. We denote this relation as "Frame", and its inverse "Lexical unit". Finally, we connect frames by the structure of FrameNet (Table 6). Distributional embeddings are from the Google News pre-trained Word2Vec model (Mikolov et al., 2013a); the counts of each entity type that were also found in the distributional corpus are shown in Table 6.

Results

As seen in Table 1, the representations learned by FR-Linear and FR-Neural are significantly more useful for link prediction than those of the baseline methods.

Retrofitting Model	**'Inheritance'** (2132/992)	**'Using'** (1552/668)	**'Reframing Mapping'** (544/312)	**'Subframe'** (356/168)	**'Perspective On'** (336/148)
None	87.58 ± 1.04	88.59 ± 1.93	85.60 ± 1.80	91.24 ± 0.86	89.59 ± 3.25
FR-Identity	90.79 ± 0.69	87.87 ± 1.48	87.02 ± 0.63	94.50 ± 1.70	94.24 ± 1.02
FR-Linear	$\mathbf{92.92 \pm 0.16}$	92.04 ± 1.45	89.37 ± 2.45	94.65 ± 1.05	$\mathbf{94.73 \pm 1.12}$
FR-Neural	92.46 ± 0.67	$\mathbf{92.54 \pm 1.45}$	$\mathbf{89.57 \pm 0.70}$	$\mathbf{95.65 \pm 2.21}$	94.04 ± 0.58

Retrofitting Model	**'Precedes'** (220/136)	**'See Also'** (268/76)	**'Causative Of'** (204/36)	**'Inchoative Of'** (60/16)
None	87.30 ± 4.33	85.11 ± 3.20	86.11 ± 6.00	82.50 ± 14.29
FR-Identity	85.26 ± 4.46	83.81 ± 2.14	84.49 ± 8.72	78.33 ± 20.14
FR-Linear	87.00 ± 2.18	91.93 ± 1.06	92.09 ± 6.34	82.50 ± 14.29
FR-Neural	$\mathbf{89.16 \pm 5.60}$	$\mathbf{93.25 \pm 1.79}$	$\mathbf{94.33 \pm 4.68}$	$\mathbf{85.00 \pm 7.07}$

Table 1: Retrofitting to FrameNet. Reported values are mean and standard deviation of the link prediction accuracies over three experiments. The number of edges used for (training/testing) is shown below each edge type.

5.2 WordNet

WordNet (Miller, 1995; Fellbaum, 2005) is a lexical database consisting of words (lemmas) which are grouped into unordered sets of synonyms (synsets). To examine the performance of FR on knowledge graphs which predominately satisfy the assumptions of Faruqui et al. (2015), we extract a simple knowledge graph of lemmas and the connections between these lemmas that are annotated in WordNet. These connections are dominated by hypernymy and hyponymy (Table 7), which correlate with similarity, so we expect the baseline retrofitting method to perform well.

Results

As seen in Table 2, the increased flexibility of the FR framework does not degrade embedding quality even when this extra flexibility is not intuitively necessary. Here, we evaluate standard lexical metrics for word embeddings: word similarity and syntatic relations. For word similiarity tasks, the evaluation metric is the Spearman correlation between predicted and annotated similarities; for syntatic relation, the evaluation metric is the mean cosine similarity between the learned representation of the correct answer and the prediction by the vector offset method (Mikolov et al., 2013b). In contrast to our other experiments, here the only stochastic behavior is due to stochastic gradient descent training, not sampling of evaluation samples. Even though the WordNet knowledge graph largely satisfies the assumptions of the naïve retrofitting model, the flexible FR framework achieves sustained improvements for both word similarity datasets (WordSim-353; Finkelstein et al. (2001), Mturk-771[1], and MTurk-287and syntatic relations (Google Analogy Test Set[2]).

[1] `http://www2.mta.ac.il/~gideon/mturk771.html`

[2] `http://download.tensorflow.org/data/questions-words.txt`

Retrofitting Model	Word Similarity			Syntactic Relation
	WordSim-353	MTurk-771	MTurk-287	Google Analogy
None	0.512	0.538	0.671	0.772
FR-Identity	0.512	0.532	0.664	0.774
FR-Linear	**0.542**	**0.562**	**0.679**	**0.793**
FR-Neural	$\underline{0.516 \pm 0.001}$	$\underline{0.543 \pm 0.001}$	$\underline{0.676 \pm 0.001}$	$\underline{0.784 \pm 0.000}$

Table 2: Retrofitting to WordNet. Reported values are Spearman correlations for the word similarity tasks and mean cosine similarity for the syntatic relation task. These are deterministic evaluations, so the only source of stochasticity is the optimization of the FR-Neural model.

5.3 SNOMED-CT

SNOMED-CT is an ontology of clinical healthcare terms and concepts including diseases, treatments, anatomical terms, and many other types of entities. From the publicly available SNOMED-CT knowledge graph,[3] we extracted 327,001 entities and 3,809,639 edges of 169 different types (Table 8). To create distributional embeddings, we first link each SNOMED-CT concept to a set of Wikipedia articles by indexing the associated search terms in WikiData.[4] We aggregate each article set by the method of Arora et al. (2016), which performs TF-IDF weighted aggregation of pre-trained term embeddings to create sophisticated distributional embeddings of SNOMED-CT concepts. This creates a single 300-dimensional vector for each entity.

Results

As the SNOMED-CT ontology is dominated by synonymy-like relations, we expect the simple retrofitting methods to perform well. Nevertheless, we see minimal loss in link prediction performance by using the more flexible FR framework (Table 3). Our implementation supports the use of different function classes to represent different relation types; in practice, we recommend that users select function classes in accordance with relation semantics.

Retrofitting Model	'Has Finding Site' (113748/49070)	'Has Pathological Process' (19318/8124)	'Due to' (5042/2042)	'Cause of' (1166/376)
None	95.26 ± 0.01	98.79 ± 0.07	91.47 ± 0.88	79.61 ± 1.27
FR-Identity	$\underline{95.25 \pm 0.11}$	$\underline{99.09 \pm 0.11}$	$\mathbf{94.69 \pm 0.61}$	$\mathbf{86.67 \pm 1.27}$
FR-Linear	$\mathbf{95.35 \pm 0.01}$	$\mathbf{99.35 \pm 0.01}$	$\underline{93.50 \pm 0.46}$	$\underline{80.82 \pm 0.49}$
FR-Neural	95.22 ± 0.00	98.97 ± 0.22	91.70 ± 0.15	80.29 ± 0.80

Table 3: Retrofitting to SNOMED-CT. Reported values are mean and standard deviation of the link prediction accuracies over three experiments. The number of edges used for (training/testing) is shown below each edge type.

5.4 Roam Health Knowledge Graph

Finally, we investigate the utility of FR in the Roam Health Knowledge Graph (RHKG). The RHKG is a rich picture of the world of healthcare, with connections into numerous data sources: diverse medical ontologies, provider profiles and networks, product approvals and recalls, population health statistics, academic publications, financial data, clinical trial summaries and statistics, and many others. As of June 2, 2017 the RHKG contains 209,053,294 vertices, 1,021,163,726 edges, and 6,231,287,999 attributes. Here, we build an instance of the RHKG using only public data sources involving drugs and diseases.

[3] https://www.nlm.nih.gov/healthit/snomedct/index.html

[4] https://dumps.wikimedia.org/wikidatawiki/entities/

The structure of this knowledge graph is summarized in Table 9. In total, we select 48,649 disease–disease relations, 227,051 drug–drug relations, and 13,667 drug–disease relations used for retrofitting. A disjoint set of 11,306 drug–disease relations is reserved for evaluation.

In the RHKG, as in many industrial knowledge graphs, different distributional corpora are available for each type of entity. First, we mine 2.9M clinical texts for co-occurrence counts in physician notes. After counting co-occurrences, we perform a pointwise mutual information transform and ℓ_2 row normalization to generate embeddings for each entity. For drug embeddings, we supplement these embeddings with physician prescription habits. We extract prescription counts for each of 808,020 providers in the 2013 Centers for Medicare & Medicaid (CMS) dataset[5] and 837,629 providers in the 2014 CMS dataset. By aggregating prescriptions counts across provider specialty, we produce 201-dimensional distributional embeddings for each drug. Finally, we retrofit these distributional embeddings to the structure of the knowledge graph (excluding 'Treats' edges reserved for evaluation).

Results

As shown in Table 5, the FR framework significantly improves prediction of 'Treats' relations. We hypothesize that this is due to the separable nature of the graph; as seen in Figure 2, the FR retrofitting framework can learn Disease and Drug subgraphs that are nearly separable. In contrast, Identity retrofitting generates a single connected space and distorts the embeddings.

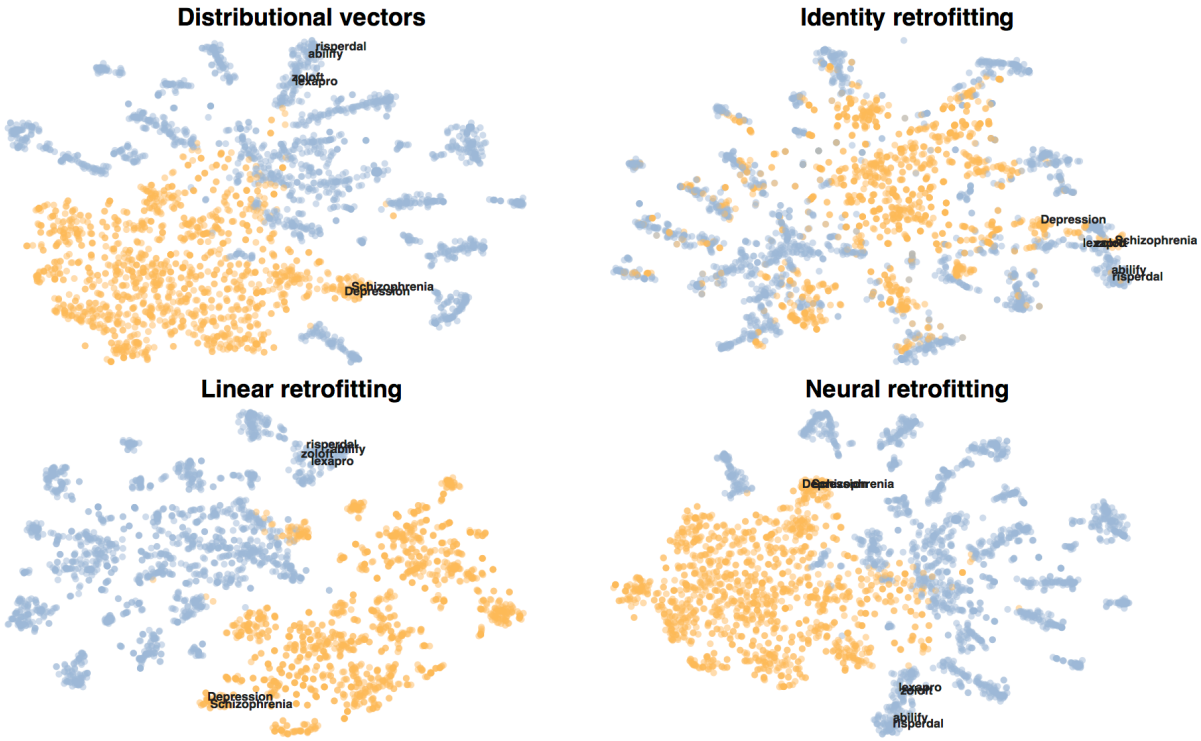


Figure 2: t-SNE Projections of the retrofitted embeddings of the drugs (blue) and diseases (orange) in the Roam Health Knowledge Graph, with selected annotations reflecting the 'Treats' relation. The distributional space strongly separates the two kinds of entity because their representations were learned in different ways. Identity retrofitting blurs this basic semantic distinction in order to make diseases and drugs in 'Treats' relations more similar. As Table 5 shows, the FR models achieve this same unification, but they need not distort the basic drug/disease distinction to do it.

We also investigate the predictions induced by the retrofitted representations. An interesting use of healthcare knowledge graphs is to predict drug *retargets*, that is, diseases for which there is no annotated treatment relationship with the drug but such a relationship may exist medically. As shown in Table 4,

[5] https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Part-D-Prescriber.html

Retrofitting Model	Drug	Disease Target	Model Score	Plausible
None	Naproxen	Ankylosing Spondylitis	0.98	Y
	Latanoprost	Superficial injury of ankle, foot and toes	0.96	N
	Pulmicort	Psoriasis, unspecified	0.96	Y
	Furosemide	Aneurysm of unspecified site	0.92	Y
	Desonide	Chlamydial lymphogranuloma (venereum)	0.92	N
FR-Identity	Latanoprost	Superficial injury of ankle, foot and toes	0.98	N
	Elixophyllin	Pneumonia in diseases classified elsewhere	0.94	Y
	Furosemide	Aneurysm of unspecified site	0.92	Y
	Oxistat	Mycosis fungoides	0.90	Y
	Trifluridine	Congenital Pneumonia	0.90	N
FR-Linear	Kenalog	Unspecified contact dermatitis	0.96	Y
	Kenalog	Pemphigus	0.96	Y
	Methyprednisolone Acetate	Nephrotic Syndrome	0.96	Y
	Furosemide	Aneurysm of unspecified site	0.94	Y
	Dexamethasone	Pemphigus	0.90	Y
FR-Neural	Onglyza	Type 2 diabetes mellitus	0.98	Y
	Pradaxa	Essential (primary) hypertension	0.96	Y
	Oxytocin	Pauciarticular juvenile rheumatoid arthritis	0.94	Y
	Terbutaline sulfate	HIV 2 as the cause of diseases classified elsewhere	0.94	N
	Lipitor	Cerebral infarction	0.92	Y

Table 4: Highest confidence drug targets that were not annotated in the Roam Health Knowledge Graph.

Retrofitting Model	'Treats' (9152/2490)
None	72.02 ± 0.50
FR-Identity	72.93 ± 0.82
FR-Linear	$\mathbf{84.22 \pm 0.82}$
FR-Neural	$\underline{73.52 \pm 0.89}$

Table 5: Drug-Disease Link Prediction Accuracies.

the top retargets predicted by the linear retrofitting model are all medically plausible. In particular, the model confidently predicts that Kenalog would treat contact dermatitis, an effect also found in a clinical trial (Usatine and Riojas, 2010).The second most confident prediction of drug retargets was that Kenalog can treat pemphigus, which is indicated on Kenalog's drug label,[6] but was not previously included in the knowledge graph. The third prediction was that methyprednisolone acetate would treat nephrotic syndrome, which is reasonable as the drug is now labelled to treat idiopathic nephrotic syndrome.[7] Interestingly, several models predict that furosemide treats "aneurysm of unspecified site", a relationship not indicated on the drug label[8], though furosemide has been observed to reduce intracranial pressure (Samson and Beyer Jr, 1982), a key factor in brain aneurysms. Finally, both the distributional data and the embeddings produced by the baseline identity retrofitting model make the nonsensical prediction that Latanoprost, a medication used to treat intraocular pressure, would also treat superficial ankle and foot injuries.

The accuracy of the predictions from the more complex models underscores the utility of the new framework for retrofitting distributional embeddings to knowledge graphs with relations that do not imply similarity.

[6] https://www.accessdata.fda.gov/drugsatfda_docs/label/2014/014901s042lbledt.pdf

[7] https://dailymed.nlm.nih.gov/dailymed/fda/fdaDrugXsl.cfm?setid=978b8416-2e88-4816-8a37-bb20b9af4b1d

[8] https://dailymed.nlm.nih.gov/dailymed/drugInfo.cfm?setid=eadfe464-720b-4dcd-a0d8-45dba706bd33

6 Conclusions and Future Work

We have presented *Functional Retrofitting*, a new framework for *post-hoc* retrofitting of entity embeddings to the structure of a knowledge graph. By explicitly modeling pairwise relations, this framework allows users to encode, learn, and extract information about relation semantics while simultaneously updating entity representations. This framework extends the popular concept of retrofitting to knowledge graphs with diverse entity and relation types. Functional Retrofitting is especially beneficial for graphs in which distinct distributional corpora are available for different entity classes, but it loses no accuracy when applied to simpler knowledge graphs. Finally, we are interested in the possibility of improvements to the optimization procedure outlined in this paper, including dynamic updates of the β and α parameters to increase trust in the graph structure while the relation functions are learned.

Acknowledgements

We would like to thank Adam Foster, Ben Peloquin, JJ Plecs, and Will Hamilton for insightful comments, and anonymous reviewers for constructive criticism.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embedtdings. *International Conference on Learning Representations*.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *JMLR W&CP: Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2012)*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615. Association for Computational Linguistics.

Manaal Faruqui, Ryan McDonald, and Radu Soricut. 2016. Morpho-syntactic lexicon generation using graph-based semi-supervised learning. *Transactions of the Association of Computational Linguistics*, 4(1):1–16.

Christiane Fellbaum. 2005. Wordnet and wordnets. In Keith Brown et al., editor, *Encyclopedia of Language and Linguistics*, page 665670. Oxford: Elsevier, second edition.

Charles J Fillmore, Christopher R Johnson, and Miriam RL Petruck. 2003. Background to framenet. *International journal of lexicography*, 16(3):235–250.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 13, pages 746–751.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Nikola Mrkšic, Diarmuid OSéaghdha, Blaise Thomson, Milica Gašic, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–148.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Duke Samson and Chester W Beyer Jr. 1982. Furosemide in the intraoperative reduction of intracranial pressure in the patient with subarachnoid hemorrhage. *Neurosurgery*, 10(2):167–169.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934.

Richard P Usatine and Marcela Riojas. 2010. Diagnosis and management of contact dermatitis. *American family physician*, 82(3):249–255.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

A Structure of Knowledge Graphs

A.1 FrameNet

The structure of the FrameNet (Baker et al., 1998; Fillmore et al., 2003) knowledge graph is shown in Table 6.

Entity Type	**Count**	**W2V Count**
Token	13572	12167
Frame	1221	464

Edge Type	**Connects**	**Count**
Frame	Token → Frame	13572
Lexical_Unit	Frame → Token	13572
Inheritance	Frame → Frame	1562
Using	Frame → Frame	1110
ReFraming_Mapping	Frame → Frame	428
Persepctive_on	Frame → Frame	242
Precedes	Frame → Frame	178
See_also	Frame → Frame	172
Causative_of	Frame → Frame	120
Inchoative_of	Frame → Frame	38
Metaphor	Frame → Frame	8

Table 6: Structure of the FrameNet knowledge graph.

A.2 WordNet

The structure of the WordNet knowledge graph (Miller, 1995) is shown in Table 7.

Entity Type	Count	W2V Count
Lemma	206978	115635

Edge Type	Count
Hypernym	136,235
Hyponym	136,235
Derivationally Related Form	60,250
Antonym	5,922
Pertainym	5,573
Usage Domain	69

Table 7: Structure of the WordNet knowledge graph.

A.3 SNOMED-CT

The structure of the knowledge graph extracted from the SNOMED-CT ontology is shown in Table 8.

A.4 Roam Health Knowledge Graph

The structure of the extracted subgraph of the RHKG is summarized in Table 9. A disjoint set of 11,306 drug–disease relations is reserved for evaluation.

Edge Type	Count	Edge Type	Count
associated_clinical_finding	493258	child	242130
has_finding_site	205263	has_method	200507
has_associated_morphology	169778	has_procedure_site	79171
has_causative_agent	69284	interprets	67900
has_active_ingredient	58976	part_of	47776
has_direct_procedure_site	45693	mapped_to	37287
same_as	30670	has_pathological_process	23641
has_dose_form	23259	has_intent	22845
causative_agent_of	19833	finding_site_of	19525
has_direct_morphology	18380	has_direct_substance	16913
has_component	15597	has_indirect_procedure_site	15596
occurs_in	14003	possibly_equivalent_to	13459
has_finding_method	12754	active_ingredient_of	12423
has_definitional_manifestation	11788	has_direct_device	11223
is_interpreted_by	10908	has_interpretation	10077
procedure_site_of	9559	occurs_after	7825
has_temporal_context	7786	associated_morphology_of	7524
has_subject_relationship_context	7465	has_part	6851
uses_device	6407	associated_with	6399
has_measured_component	6353	uses	6221
has_associated_finding	6205	has_focus	6122
uses_substance	5474	component_of	5256
temporally_follows	5029	due_to	4884
has_finding_context	4883	direct_procedure_site_of	4252
has_specimen	3767	replaces	3726
has_laterality	3641	associated_finding_of	3432
has_associated_procedure	3397	has_clinical_course	3309
has_course	3241	has_procedure_context	2945
has_approach	2808	measured_component_of	2741
has_access	2660	has_specimen_source_topography	2457
has_finding_informer	2229	has_onset	2168
has_priority	1854	mth_xml_form_of	1794
mth_plain_text_form_of	1794	mth_has_xml_form	1794
mth_has_plain_text_form	1794	direct_substance_of	1783
focus_of	1680	indirect_procedure_site_of	1662
has_revision_status	1599	uses_access_device	1587
has_access_instrument	1518	direct_device_of	1434
has_indirect_morphology	1426	associated_procedure_of	1320
has_specimen_procedure	1309	has_communication_with_wound	1155
cause_of	1121	has_extent	1082
has_specimen_substance	1030	method_of	921
has_procedure_device	770	uses_energy	753
has_procedure_morphology	752	has_surgical_approach	697
dose_form_of	676	direct_morphology_of	673
referred_to_by	667	has_associated_etiologic_finding	656
used_by	608	priority_of	586
specimen_source_topography_of	584	occurs_before	574
specimen_procedure_of	555	has_severity	525
device_used_by	525	substance_used_by	507
definitional_manifestation_of	436	temporally_followed_by	406
has_specimen_source_identity	327	has_property	282
has_instrumentation	274	has_subject_of_information	272
has_specimen_source_morphology	251	access_instrument_of	226
has_scale_type	206	specimen_substance_of	171
has_episodicity	168	has_route_of_administration	143
has_recipient_category	143	associated_etiologic_finding_of	143
specimen_of	134	approach_of	125
subject_relationship_context_of	115	has_indirect_device	114
interpretation_of	109	procedure_device_of	107
course_of	106	indirect_morphology_of	10

Table 8: Structure of the SNOMED-CT knowledge graph.

Entity Type	Count
Drug	223,019
Disease	95,559

Edge Type	Connects	Count
Ingredient Of	Drug → Drug	49,218
Has Ingredient	Drug → Drug	49,208
Is A	Drug → Drug	28,297
Has Descendent	Disease → Disease	22,344
Treats	Drug → Disease	19,374
Has Active Ingredient	Drug → Drug	18,422
Has Child	Disease → Disease	18,066
Active Ingredient Of	Drug → Drug	17,175
Has TradeName	Drug → Drug	11,783
TradeName Of	Drug → Drug	11,783
Inverse Is A	Drug → Drug	10,369
Has Symptom	Disease → Disease	7,892
Part Of	Drug → Drug	6,882
Has Part	Drug → Drug	6,624
Same As	Drug → Drug	5,882
Precise Ingredient Of	Drug → Drug	3,562
Has Precise Ingredient	Drug → Drug	3,562
Possibly Equivalent To	Drug → Drug	1,233
Causative Agent of	Drug → Drug	1,070
Has Form	Drug → Drug	602
Form of	Drug → Drug	602
Component of	Drug → Drug	436
Includes	Disease → Disease	347
Has Dose Form	Drug → Drug	138

Table 9: Structure of the subgraph of the Roam Health Knowledge Graph.

Context-Sensitive Generation of Open-Domain Conversational Responses

Wei-Nan Zhang*, Yiming Cui†, Yifa Wang*, Qingfu Zhu*, Lingzhi Li*, Lianqiang Zhou‡, Ting Liu*
*Research Center for Social Computing and Information Retrieval,
Harbin Institute of Technology, Harbin, China.
†Joint Laboratory of HIT and iFLYTEK (HFL), iFLYTEK Research, Beijing, China
‡Joint Laboratory of HIT and Tencent Corporation, Shenzhen, China
*{wnzhang, yfwang, qfzhu, lzli, tliu}@ir.hit.edu.cn
†ymcui@iflytek.com
‡tomcatzhou@tencent.com

Abstract

Despite the success of existing works on single-turn conversation generation, taking the coherence in consideration, human conversing is actually a context-sensitive process. Inspired by the existing studies, this paper proposed the static and dynamic attention based approaches for context-sensitive generation of open-domain conversational responses. Experimental results on two public datasets show that the proposed static attention based approach outperforms all the baselines on automatic and human evaluation.

1 Introduction

Until recently, training open-domain conversational systems that can imitate the way of human conversing is still not a well-solved problem and non-trivial task. Previous efforts focus on generating open-domain conversational responses as an unsupervised clustering process (Ritter et al., 2010), a phrase-based statistical machine translation task (Ritter et al., 2011) and a search problem based on the vector space model (Banchs and Li, 2012), etc. With the booming of deep learning, particularly the neural network based sequence-to-sequence models, generating open-domain conversational responses gradually turns into an end-to-end encoding and decoding process (Sutskever et al., 2014; Vinyals and Le, 2015; Shang et al., 2015; Serban et al., 2016b; Li et al., 2016a; Li et al., 2016b; Shao et al., 2017; Yao et al., 2017). Despite the success of the above research on single-turn conversational response generation, human conversations are usually coherent (Li et al., 2016c) and **context-sensitive** (Tian et al., 2017; Xing et al., 2017). Table 1 illustrates how contextual information in conversations impact on the response generation. For instance, given a message[1] "*How should I tell my mom?*", as input, to a single-turn

Conversation 1	Conversation 2
A: I got a high score on my exam.	A: I failed to pass the exam.
B: Oh! Great!	B: That's too bad.
A: *How should I tell my mom?*	A: *How should I tell my mom?*
B: ***Go and give her a big surprise!***	B: ***Just tell her the truth and do well next time.***

Table 1: An example of the impact of contextual information on human conversations. "A" and "B" denote two speakers in the conversations.

conversational response generation model, it should output a fixed response regardless of the content in previous utterances. However, as shown in Table 1, in the conversations[2], the responses to be generated (the last utterance in Table 1) should not only dependent on the last one message ("*How should I tell my mom?*"), but also need to consider the longer **historical utterances** in the conversations.

[1]Here, a "message" indicates an input of a response in single-turn conversational response generation.

[2]In this paper, a "conversation" equals to an "open-domain conversation"and a "conversational response" or "response" equals to an "open-domain conversational response".

Proceedings of the 27th International Conference on Computational Linguistics, pages 2437–2447
Santa Fe, New Mexico, USA, August 20-26, 2018.

Recent studies on generating open-domain conversational responses begin to explore the context information to generate more informative and coherent responses. Serban et al. (2016a) presented a hierarchical recurrent encoder-decoder (HRED) to recurrently model the dialogue context. Serban et al. (2017b) further introduced a stochastic latent variable at each dialogue turn to improve the diversity of the HRED model. Zhao et al. (2017) proposed a conditional variational autoencoder based approach to learning contextual diversity for neural response generation. Xing et al. (2017) proposed a hierarchical recurrent attention network (HRAN) to jointly model the importance of tokens and utterances. Tian et al. (2017) treated the hierarchical modeling of contextual information as a recurrent process in encoding. We could make two conclusions from these works.

- First, existing studies of utterance modeling mainly focus on representing utterances by using bidirectional GRU (Xing et al., 2017) or unidirectional GRU (Tian et al., 2017).
- Second, there are two types of approaches on context (inter-utterance) modeling. One is the attention-based approach (Xing et al., 2017), the other is the sequential integration approach (Tian et al., 2017).

Drawing the advantages of the existing approaches, in this paper, we proposed a novel context-sensitive generation approach, which obtains the context representation of a conversation by weighing the importance of each utterance using two attention mechanisms, namely dynamic and static attention, to generate open-domain conversational responses.

2 The Proposed Context-Sensitive Generation Approach

2.1 Preliminary

A typical neural network based sequence-to-sequence model for generating open-domain conversational responses usually includes an encoder and a decoder. The encoder expresses an input message as a dense vector which represents the semantics of the input message. The decoder then generates a conversational response according to the semantic representation of the input message. In context-sensitive generation of open-domain conversational responses, the input message to the encoder usually includes several historical utterances in a conversation. Therefore, one of the key problems in context-sensitive generation is how to encode historical utterances in a conversation. Figure 1 presents two state-of-the-art approaches to encoding contextual information for context-sensitive response generation. Here, u_i, u_{i+1} and u_j

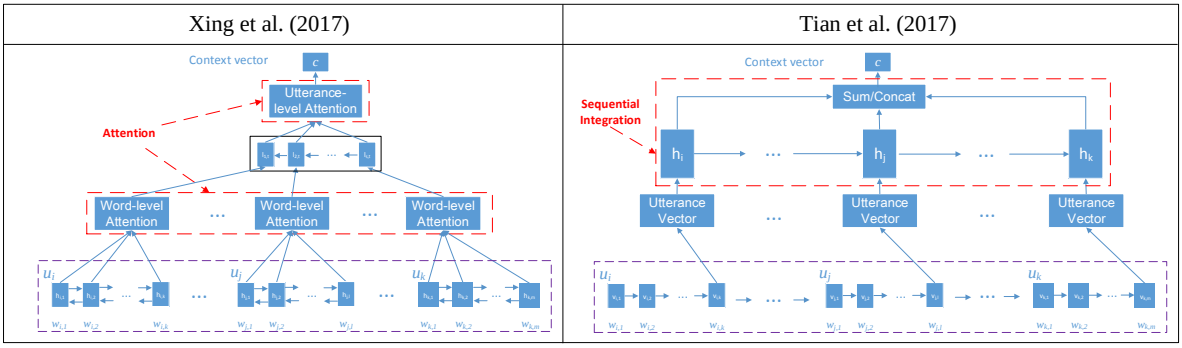


Figure 1: The encoders of two state-of-the-art approaches of open-domain conversational response generation using contextual information.

denote the i-th, i+1-th and j-th utterance, respectively, in a conversation. As the inputs of the two models, they are then represented to utterance-level vectors as shown in the second layer of the two models in Figure 1. The context vectors of the two models are obtained by hierarchically representing the utterances to a dense vector c for decoding. It is easy to recognize that the frameworks used to illustrate the encoders of two existing context-sensitive generation models look similar to each other. There are two different parts between the two frameworks:

- **Utterance Representations: Bidirectional GRU vs. Unidirectional GRU**

Xing et al. (2017) utilized a bidirectional GRU and a word-level attention mechanism to transfer word representations to utterance representations. Tian et al. (2017) represented the utterance in a simpler way, which is a unidirectional GRU.

- **Inter-utterance Representations: Attention vs. Sequential Integration**

Xing et al. (2017) proposed a hierarchical attention mechanism to feed the utterance representations to a backward RNN to obtain contextual representation. Tian et al. (2017) proposed a weighted sequential integration (WSI) approach to use an RNN model and a heuristic weighting mechanism to obtain inter-utterance representation.

2.2 The Proposed Model

The proposed context-sensitive generation model is under the framework of encoder-decoder. To obtain the contextual representations, the proposed model consists of a hierarchical representation mechanism for encoding. For utterance representation, we consider the advantages of the two state-of-the-art approaches to encoding contextual information for context-sensitive response generation (Xing et al., 2017; Tian et al., 2017). We utilize a GRU model to obtain utterance representation. For inter-utterance representation, inspired by the above approaches of modeling inter-utterance representations, we proposed two attention mechanisms, namely dynamic and static attention, to weigh the importance of each utterance in a conversation and obtain the contextual representation. Figure 2 shows the framework of the proposed context-sensitive generation model. Drawing the advantages of attention mechanism on

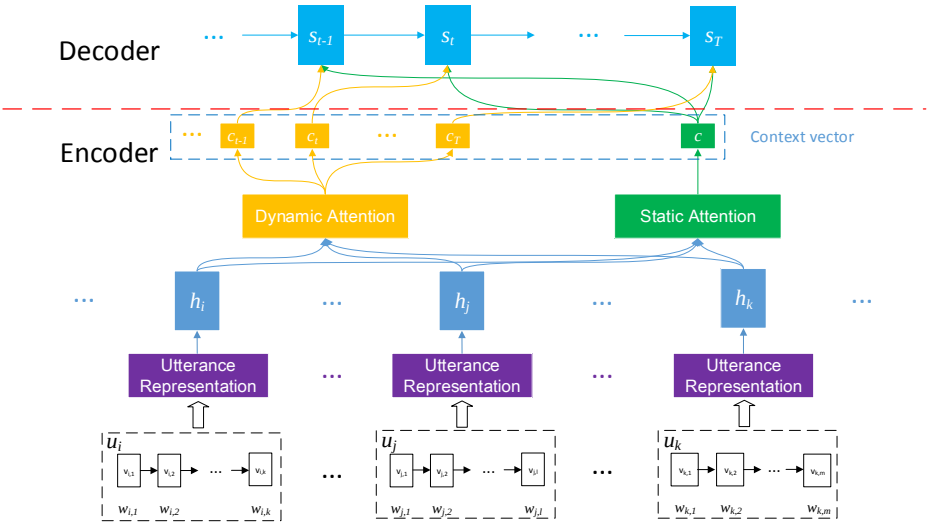


Figure 2: The proposed context-sensitive generation model for open-domain conversational response. Here, u_* denotes the $*$-th utterance in a conversation.

weighing the importance of utterances for generating open-domain conversational responses (Xing et al., 2017), we thus model the inter-utterance representation to obtain the context vector in two measures, namely static and dynamic attention, as shown in Figure 2. We then formally describe the static and dynamic attention for decoding process.

- **Static Attention based Decoding**

As shown in Figure 2, the static attention mechanism calculates the importance of each utterance as e_i or α_i ($i \in \{1, ..., s\}$).

$$e_i = V^T \tanh(Wh_i + Uh_s) \tag{1}$$

$$\alpha_i = \frac{\exp(e_i)}{\sum_i \exp(e_i)} \tag{2}$$

$$c = \sum_i \alpha_i h_i \tag{3}$$

Here, h_i and h_s denote the representations of hidden state of the i-th and the last utterance in a conversation, respectively. V, W and U are parameters. We can see that once the weights of each utterance α_i ($i \in \{1, ..., s\}$) are produced, they will be unchanged in the decoding process. In decoding, the t-th hidden state s_t can be calculated as follows:

$$s_t = f(y_{t-1}, s_{t-1}, c) \tag{4}$$

Here, y_{t-1} is the $t-1$-th output of the decoder and s_{t-1} is the hidden state of t-1-th time step in decoding. Notice that y_0 is set to be a special character and s_0 is initialized by h_s. The generated response is thus represented as a sequence of $y_1, y_2, ..., y_T$, where T denotes the last time step.

• **Dynamic Attention based Decoding**

Rather than the static attention mechanism fixes the weights of each utterance before decoding process, the dynamic attention mechanism maintains a weighting matrix and updates the weights of each utterance during decoding process as shown in Figure 2. The formal illustration of the dynamic attention mechanism for decoding is as follows:

$$e_{i,t} = V^T \tanh(W h_i + U s_{t-1}) \tag{5}$$

$$\alpha_{i,t} = \frac{\exp(e_{i,t})}{\sum_i \exp(e_i)} \tag{6}$$

$$c_t = \sum_i \alpha_{i,t} h_i \tag{7}$$

Here, V, W and U are also parameters that are independent to those in the static attention. T denotes the transposition operation of V. The $e_{i,t}$ and $\alpha_{i,t}$ are calculated in each time step t of decoding. The t-th hidden state s_t in dynamic attention-based decoder can be calculated as follows:

$$s_t = f(y_{t-1}, s_{t-1}, c_t) \tag{8}$$

The main difference between our proposed conversational response generation model and the above two state-of-the-art models is the two attention mechanisms for obtaining the contextual representation of a conversation. Rather than use a hierarchical attention neural network (Xing et al., 2017) to obtain the contextual representation of a conversation, we propose two utterance-level attentions for weighting the importance of each utterance in the context, which is more simple in structure and has less number of parameters than the hierarchical attention approach. Meanwhile, rather than use a heuristic approach to weigh the importance of each utterance in the context (Tian et al., 2017), in our proposed approach, the weights of utterance in the context are learned by two attention mechanisms from the data, which is more reasonable and flexible than the heuristic based approach.

3 Experimental Results

3.1 Experiment Settings

Dataset: Two datasets are selected for the experiment of generation of open-domain conversational responses. First is the Ubuntu dataset which is developed by Lowe et al. (2015). The dataset is extracted from the Ubuntu Internet Relayed Chat (IRC) channel and recently used for the generation of conversational responses in (Serban et al., 2016a; Serban et al., 2017b; Serban et al., 2017a). We follow the train-test split proposed by Serban et al. (2017a). It is worthy to note that there is no development set in Serban et al. (2017a). In this paper, we randomly select the same number of sessions to that in the test set from the training set. Second is the OpenSubtitles dataset which is proposed by Tiedemann (2009) and also used by Li et al. (2016a; Li et al. (2016c). The detailed statistics of the two datasets are shown in Table 2. It is worthy to note that the original released data of OpenSubtitles consists of about 40,000,000

utterances without partitions of conversational session, which is called "session" for short in the following of this paper. Therefore, we split each of 10 continuous utterances as a session. We then randomly sample 800,000 sessions for training (including 8,000 sessions for developing) and remove them from the complete dataset. In the rest of the complete dataset, we again randomly sample 8,000 sessions for testset. The vocabulary size equals to the number of unique tokens in both two datasets, respectively.

	Ubuntu	OpenSubtitles
Train size	429,915	792,000
Dev size	18,920	8,000
Test size	18,920	8,000
Vocabulary size	155,490	91,405
Avg # of u per session	7.5	10
Avg # of w per u	12.3	7.5

Table 2: The statistics of two experimental datasets. Avg is short for average. # represents number. u and w denote utterance and word respectively. The unit of training and test is a conversational session.

Hyper-Parameters: For the static attention model, the dimension of hidden layer in encoder and decoder is 512. The padding length is set to 15. The dimension of word embedding equals to 200. The word embedding is pre-trained using the skip-gram model in word2vec (Mikolov et al., 2013) and fine-tuned during the learning process. For the dynamic attention model, the dimension of hidden layer in encoder and decoder is 1024. The padding length and dimension of word embedding are same to the static attention model. Adam is used for optimization. The initial learning rate is 0.001 and the weight_decay is set to 10^{-5}. The dropout parameter equals to 0.5. Mini-batch is used and the batch size equals to 80. The number of iterations in training is 10.
Baselines: For the experimental comparisons, six baselines are chosen. Four out of them are state-of-the-art approaches. They are VHRED, CVAE, WSI, and HRAN.

- **LSTM:** Under the sequence-to-sequence framework for generation of conversational responses, the most simple but natural idea is to directly use the LSTM to encode all the utterances in a session word by word and then decode to generate a response.

- **HRED:** The first hierarchical recurrent encoder-decoder model, which is proposed by Serban et al. (2016a), for conversational response generation.

- **VHRED:** The augmented HRED model, which incorporates a stochastic latent variable at utterance level for encoding and decoding, is proposed by Serban et al. (2017b).

- **CVAE:** The conditional variational autoencoder based approach, which is proposed by Zhao et al. (2017), to learn context diversity for conversational responses generation.

- **WSI** and **HRAN** are proposed by Tian et al. (2017) and Xing et al. (2017) respectively. We detailed describe and compare the two models in Section 2.1 and 2.2 and their frameworks are shown in Figure 1.

3.2 Evaluation and Results

3.2.1 Automatic Evaluation

Until now, automatically evaluating the quality of open-domain conversational response is still an open problem. The BLEU score (Papineni et al., 2002), which is a widely used evaluation metric for machine translation, is not a suitable metric for conversation generation, as the appropriate responses to the same message may share less common words. Moreover, it is also impossible to construct a reference set, which includes all appropriate responses, of each message. The perplexity that is used to evaluate language model, is also not suitable to evaluate the relevance between messages and responses (Shang

Models	Ubuntu			OpenSubtitles		
	Average	**Greedy**	**Extrema**	**Average**	**Greedy**	**Extrema**
LSTM	0.2300	0.1689	0.1574	0.5549	0.5029	0.3897
HRED	0.5770	0.4169	0.3914	0.5571	0.5033	0.3932
VHRED	0.5419	0.3839	0.3627	0.5248	0.4821	0.3556
CVAE	0.5672	0.3982	0.3689	0.4708	0.3390	0.3173
WSI	0.5775	0.4196	0.3893	0.5598	0.4964	0.3903
HRAN	0.5964	0.4139	0.3898	0.5617	0.5195	0.3898
Dynamic$_{\rightleftarrows}$	0.5750	0.4043	0.3802	0.5487	0.5054	0.3812
Dynamic$_{\rightarrow}$	0.5968	0.4132	0.3877	0.5629	0.5193	0.3905
Static$_{\rightleftarrows}$	0.5998	0.4124	0.3886	0.5475	0.5147	0.3862
Static$_{\rightarrow}$	**0.6121**†	**0.4293**†	**0.3975**†	**0.5656**†	**0.5232**†	**0.3937**†

Table 3: The results of automatic evaluation on Ubuntu and OpenSubtitles datasets. Dynamic and Static are our proposed approaches whose framework is shown in Figure 2. The other models are baselines. $\rightarrow$ and $\rightleftarrows$ denote the use of unidirectional and bidirectional GRU in the proposed model to obtain utterance representations, respectively. † denotes the results pass the statistical significance test with $p < 0.05$.

et al., 2015; Li et al., 2016c). In this paper, we employ an evaluation metric that is proposed by Serban et al. (2016a) and also used in (Serban et al., 2017b). Rather than calculating the token-level or n-gram similarity as the perplexity and BLEU (Papineni et al., 2002), the metric measure the semantic similarity between a generated responses $\hat{r}$ and the ground-truth responses r by matching their semantic representations. The metric also has three aspects, namely $\mathbf{Average}$, $\mathbf{Greedy}$ and $\mathbf{Extrema}$. For the $\mathbf{Average}$, it first calculates the element-wise arithmetic average of embeddings of all words in $\hat{r}_a$ and r_a, respectively and produces two response representations $v_{\hat{r}_a}$ and v_{r_a}. The value of $\mathbf{Average}$ is then equals to the cosine similarity of $v_{\hat{r}_a}$ and v_{r_a}. For the $\mathbf{Greedy}$, every word in $\hat{r}$ will find a most similar word in r by calculating the cosine similarity of their word embeddings. After that, the element-wise arithmetic average of embeddings of all words in $\hat{r}_a$ and the corresponding words in r are calculated and two response representations $v_{\hat{r}_g}$ and v_{r_g} are produced. The value of $\mathbf{Greedy}$ is then equals to the cosine similarity of $v_{\hat{r}_g}$ and v_{r_g}. For the $\mathbf{Extrema}$, two embedding matrices $m_{\hat{r}}$ and m_r can be obtained by arranging the embeddings of all words in $\hat{r}_a$ and r_a, respectively. The i-th column of $m_{\hat{r}}$ is the embedding of the i-th word in $\hat{r}$ as well as that in m_r. Getting the maximum value of each row in $m_{\hat{r}}$ and m_r, respectively, we then obtain two response representations $v_{\hat{r}_e}$ and v_{r_e}. The value of $\mathbf{Extrema}$ is then equals to the cosine similarity of $v_{\hat{r}_e}$ and v_{r_e}.

Table 3 shows the experimental results on two datasets.

We can see that our proposed context-sensitive generation model with static attention outperforms all the baselines in the two datasets. It verifies the effectiveness of the proposed utterance-level attention mechanism on modeling context representations for generating conversational responses. To compare the dynamic and static attentions, we find that for the generation of conversational response, dynamically estimate the importance of each utterance in context performs worse than the static attention approach. The reason may be that the context vector in dynamic attention model is changed in every time step of decoding. The change of context vector may lead to decoding incoherent responses. Meanwhile, the unidirectional GRU based models outperform the bidirectional GRU based models. It doesn't illustrate the unidirectional GRU is better than the bidirectional GRU in utterance representation. It only indicates that in the current experimental settings, the unidirectional GRU based model outperforms the bidirectional one.

3.2.2 Human Evaluation

For human evaluation, we proposed 2 metrics, namely **Coherence** and **Naturalness**. As the example shown in Table 1, in context-sensitive generation of conversational responses, a generated response should not only dependent on the last one message but also need to consider the longer context in the

Models	Ubuntu			OpenSubtitles		
	Coherence	**Naturalness**	**Diversity**	**Coherence**	**Naturalness**	**Diversity**
LSTM	0.930	0.477	0.069	0.963	0.443	0.099
HRED	0.967	0.490	0.141	0.963	0.443	0.098
VHRED	1.010	0.507	0.140	0.986	0.473	0.093
CVAE	0.987	**0.513**	0.140	1.000	0.477	**0.114**
WSI	1.010	0.507	0.141	1.013	0.490	0.110
HRAN	1.027	0.510	0.147	**1.033**	0.477	0.109
Dynamic	0.987	0.507	**0.158**	1.013	0.477	0.109
Static	**1.070**	**0.513**	0.150	1.027	**0.497**	0.110

Table 4: The results of human evaluation on Ubuntu and OpenSubtitles datasets.

conversation. **Coherence** is thus used to evaluate the semantic consistency between a generated response and its context. The Coherence score is in the range of 0 to 2, where 0, 1, 2 denote *incoherent*, *neutral* and *coherent*, respectively. In some cases, a coherent response may not be a natural one. Given an example message, "*Can you tell me the way to the nearest bazaar?*", the response "*Yes, I can tell you the way to the nearest bazaar.*" is definitely a coherent but not a natural response. A more extreme example of a message-response pair is "*I don't know what you are talking about!*" and "*I don't know what you are talking about!*". Therefore, besides the Coherence, we proposed another metric, Naturalness, to evaluate the quality of generated responses. For human evaluation, given a context and a conversational response generated by a model, **Naturalness** denotes whether the response can be an alternative to a human response. The Naturalness value equals to 1 or 0, which represents the generated response can be an alternative to a human response or not, respectively. Besides the Coherence and Naturalness, we also want to compare the **Diversity** of the responses generated by all baselines and our proposed approach. Here, diversity score of a generated response equals to the number of distinct tokens in the response divided by the total number of distinct tokens in its context (including the number of distinct tokens in the response). The final Diversity score is the average diversity of all the generated responses in test set.

In the human evaluation, for each model, we randomly sample 500 test results from Ubuntu and OpenSubtitles datasets, respectively. Each of the three annotators, who are undergraduate students and not involved in other parts of the experiment, is asked to provide the evaluation scores for all the 8,000 test results. The final score of each model equals to the average score of the three annotators. Table 4 shows the human evaluation results on the two datasets. Generally speaking, we can see that the proposed static attention-based model outperforms the baselines in Coherence and Naturalness on Ubuntu dataset and obtains comparable performance with the HRAN model in Coherence on OpenSubtitles dataset. For the Diversity, we can see that the proposed dynamic attention-based model is better at generating diverse responses than other models on Ubuntu dataset. We also notice that the CVAE model obtains the best diversity performance on OpenSubtitles dataset and the best Naturalness performance on Ubuntu dataset.

3.2.3 Analysis of Context Length

To verify the impact of context length on the performance of the proposed model for the generation of conversational responses, we use different length of context to re-train the proposed models, which are called context varied models, on two datasets. Here, context length indicates the number of historical utterances that are used for encoding in a context. Figure 3 shows that the performance of the proposed static and dynamic attention models are varying with the change of context length. The values denote the difference between the results of Static$_{\rightarrow}$ and Dynamic$_{\rightarrow}$ in Table 3 and the results of the context varied models. It also verifies that the generation of conversational responses is a context-sensitive process, which relates to the numbers of utterance in context for encoding. Table 5 shows the conversational responses, which are sampled from the test result generated by the proposed static attention model. We can see that the attention values predicted by the static attention model can appropriately reveal the importance of the utterance in a context for generating conversational responses.

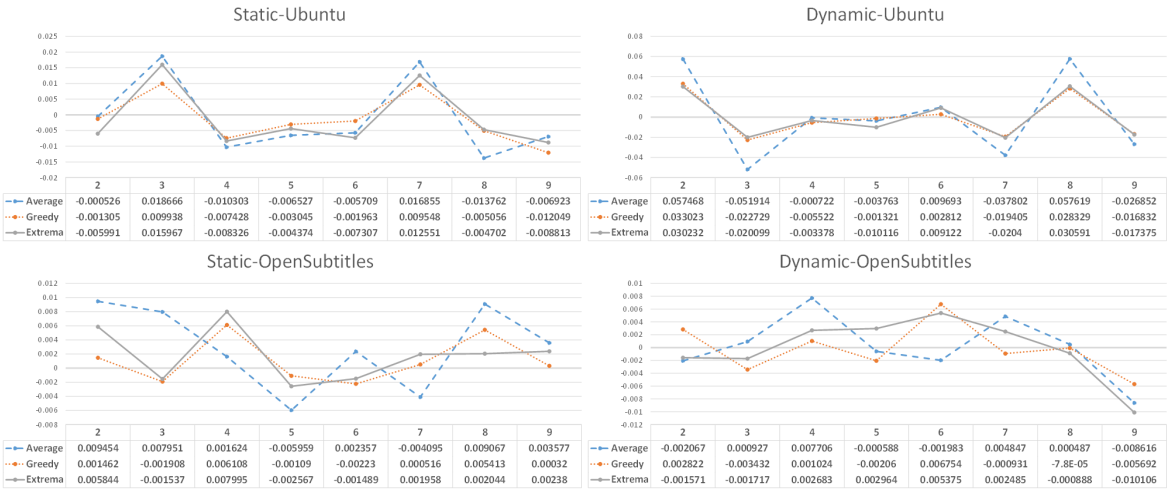


Figure 3: The impact of different number of utterances, used for encoding, in a context (context length) on the performance of the proposed static and dynamic attention based models on two experimental datasets. The range of context length is from 2 to 9.

Sample 1		α
Context	Bonhoeffer, what drives do you want to mount?	0.4872
	What?	0.1535
	I have an ext3 usb drive.	0.3593
Response	I have a usb drive, I want to use it for testing.	

Sample 2		α
Context	Are you using a 32bit version of firefox? why?	0.2284
	Need to use webex, doesn't support 64bit java.	0.4660
	Using 64bit firefox.	0.3056
Response	I have a 64-bit machine, and I want to use 64 bit.	

Table 5: Samples of the generated responses from the Ubuntu test result. α indicates the attention value of each utterance in context calculated by the proposed static attention mechanism.

4 Related Work

Ritter et al. (2010) proposed an unsupervised approach to model dialogue response by clustering the raw utterances. They then presented an end-to-end dialogue response generator by using a phrase-based statistical machine translation model (Ritter et al., 2011). Banchs and Li (2012) introduced a search-based system, named IRIS, to generate dialogues using vector space model and then released the experimental corpus for research and development (Banchs, 2012). Recently, benefit from the advantages of the **sequence-to-sequence learning** framework with neural networks, Sutskever et al. (2014) and Shang et al. (2015) had drawn inspiration from the neural machine translation (Bahdanau et al., 2014) and proposed an RNN encoder-decoder based approach to generate dialogue by considering the last one sentence and a larger range of context respectively. Serban et al. (2016b) proposed a parallel stochastic generation framework which first generates a coarse sequence and then generates an utterance conditioned on the coarse sequence. Shao et al. (2017) introduced the "glimpse-model" which adds self-attention to the decoder to maintain coherence for generating long, informative, coherent and diverse responses in single turn setting. Yao et al. (2017) first predicted cue words using point-wise mutual information (PMI) for short text conversation generation and then added them into the encoder-decoder framework. To consider the **context information** for improving the diversity of generated conversations, Serban et

al. (2016a) presented a hierarchical recurrent encoder-decoder (HRED) approach to encode each utterance and recurrently model the dialogue context to generate context-dependent responses. Serban et al. (2017b) further introduced a stochastic latent variable at each dialogue turn to improve the ambiguity and uncertainty of the HRED model for dialogue generation. Xing et al. (2017) proposed a hierarchical recurrent attention network (HRAN) to jointly model the importance of tokens in utterances and the utterances in context for context-aware response generation. Tian et al. (2017) presented a context-aware hierarchical model to generate conversations by jointly modeling the utterance and inter-utterance information for encoding process. As the advantages of **generative adversarial net (GAN)** and **variational autoencoder (VAE)**, Yu et al. (2017) proposed a sequence generative adversarial net model to assess a partially generated sequence with policy gradient and obtain the intermediate rewards by using Monte Carlo search. Zhao et al. (2017) modified the VAE model by conditioning the response into the VAE model in training step to optimize the similarity of prior network and recognition network for dialogue generation. Similarly, Shen et al. (2017) presented a conditional variational framework to generate specific responses based on the dialog context. Due to the recent advantages of **reinforcement learning** on modeling human-computer interactions, such as the AlphaGo (Silver et al., 2016), researchers begin to focus on modeling the success of a conversation by not only considering the quality of single turn response generation, but also considering long-term goal of the conversation. To address the problems of generating generic and repetitive response of the RNN encoder-decoder framework, Li et al. (2016c) proposed a deep reinforcement learning approach to either generate meaningful and diverse response or increase the length of the generated dialogues. Dhingra et al. (2017) presented an end-to-end dialogue system for information accquisition, which is called KB-InfoBot from knowledge base (KB) by using reinforcement learning. Asghar et al. (2017) proposed an active learning approach to learn user explicit feedback online and combine the offline supervised learning for response generation of conversational agents.

5 Conclusion and Future Work

This paper proposed a novel context-sensitive generation approach for open-domain conversational responses. The proposed model gained from the proposed static and dynamic attention for context or inter-utterance representation. Experimental results show that the proposed model generally outperforms all the baselines in automatic and human evaluations. It is also verified the impact of context length on the performance of the proposed generation models for conversational responses. In future work, the way to uniformly integrate the static and dynamic attention for decoding will be explored.

Acknowledgements

The authors would like to thank all the anonymous reviewers for their insightful comments. The paper is supported by the NSFC (No. 61502120, 61472105, 61772153).

References

Nabiha Asghar, Pascal Poupart, Xin Jiang, and Hang Li. 2017. Deep active learning for dialogue generation. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 78–83, Vancouver, Canada, August. Association for Computational Linguistics.

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

Rafael E. Banchs and Haizhou Li. 2012. Iris: a chat-oriented dialogue system based on the vector space model. In *ACL*, pages 37–42.

Rafael E. Banchs. 2012. Movie-dic: a movie dialogue corpus for research and development. In *ACL*, pages 203–207.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the*

55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 484–495. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 994–1003.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016c. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202. Association for Computational Linguistics.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294, Prague, Czech Republic, September. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *NIPS*, 26:3111–3119.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *NAACL*, pages 172–180.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *EMNLP*, pages 583–593.

Iulian Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016a. Building end-to-end dialogue systems using generative hierarchical neural network models.

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron Courville. 2016b. Multiresolution recurrent neural networks: An application to dialogue response generation. *arXiv preprint arXiv:1606.00776*.

Iulian Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron Courville. 2017a. Multiresolution recurrent neural networks: An application to dialogue response generation.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017b. A hierarchical latent variable encoder-decoder model for generating dialogues. pages 3295–3301.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1577–1586.

Yuanlong Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating high-quality and informative conversation responses with sequence-to-sequence models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2200–2209. Association for Computational Linguistics.

Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. 2017. A conditional variational framework for dialog generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 504–509. Association for Computational Linguistics.

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

Ilya Sutskever, Oriol Vinyals, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *NIPS*, 4:3104–3112.

Zhiliang Tian, Rui Yan, Lili Mou, Yiping Song, Yansong Feng, and Dongyan Zhao. 2017. How to make context more useful? an empirical study on context-aware neural conversational models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–236. Association for Computational Linguistics.

J Tiedemann. 2009. *News from OPUS - A Collection of Multilingual Parallel Corpora with Tools and Interfaces*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Chen Xing, Wei Wu, Yu Wu, Ming Zhou, Yalou Huang, and Wei-Ying Ma. 2017. Hierarchical recurrent attention network for response generation. *arXiv preprint arXiv:1701.07149*.

Lili Yao, Yaoyuan Zhang, Yansong Feng, Dongyan Zhao, and Rui Yan. 2017. Towards implicit content-introducing for generative short-text conversation systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2180–2189. Association for Computational Linguistics.

L Yu, W Zhang, J Wang, and Y Yu. 2017. Seqgan: sequence generative adversarial nets with policy gradient. In *AAAI Conference on Artificial Intelligence, 4-9 February 2017, San Francisco, California, Usa*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–664. Association for Computational Linguistics.

A LSTM Approach with Sub-word Embeddings for Mongolian Phrase Break Prediction

Rui Liu, Feilong Bao ✉, Guanglai Gao, Hui Zhang, Yonghe Wang
College of Computer Science, Inner Mongolia University,
Inner Mongolia Key Laboratory of Mongolian Information Processing Technology,
Hohhot 010021, China
liurui_imu@163.com;csfeilong@imu.edu.cn

Abstract

In this paper, we first utilize the word embedding that focuses on sub-word units to the Mongolian Phrase Break (PB) prediction task by using Long Short-Term Memory (LSTM) model. Mongolian is an agglutinative language. Each root can be followed by several suffixes to form probably millions of words, but the existing Mongolian corpus is not enough to build a robust entire word embedding, thus it suffers a serious data sparse problem and brings a great difficulty for Mongolian PB prediction. To solve this problem, we look at sub-word units in Mongolian word, and encode their information to a meaningful representation, then fed it to LSTM to decode the best corresponding PB label. Experimental results show that the proposed model significantly outperforms traditional CRF model using manually features and obtains 7.49% F-Measure gain.

1 Introduction

A Text-to-Speech (TTS) system converts the input text into synthetic speech with high naturalness and intelligibility. Naturalness is mainly influenced by the prosody modeling, especially by the Phrase Break (PB) prediction. Because the PB prediction is the first step of TTS, any error in this step will propagate to downstream steps such as intonation prediction and duration modeling. Those errors will result in the synthetic speech which is unnatural and difficult to understand. So that many researchers devote themselves to improving the performance of the PB prediction.

Typically PB prediction methods usually use machine learning models like Hidden Markov Models (HMMs) or Conditional Random Fields (CRFs) which trained with large sets of labeled training data. In these PB prediction models, the Part-of-Speech (POS) tag have been shown to be an effective feature and usually been included in the input feature set. The POS estimation itself is also a challenging task, and relies on large labeled training corpus, too. Its accuracy is always lower than our expectation, especially for those low-resource languages like Mongolian where the required linguistic resources are not readily available, and manual annotation is expensive and time-consuming.

In recent years, there are many works applying the word embedding techniques to Natural Language Processing (NLP) tasks, such as question answering, machine translation and so on (Bordes, 2014; Xiong, 2017; Devlin, 2014). Previous work has shown that the POS prediction task can be solved with high accuracy only using the word embedding feature as the input (Wang, 2015). POS information is most likely to be included in the word embedding representations. Therefore, some PB prediction systems which don't rely on the POS feature are developed (Watts, 2011; Vadapalli, 2014; Vadapalli, 2016). In (Watts, 2011), the authors obtain continuous-valued word embedding features that summarize the distributional characteristics of word types as surrogates of POS features. In (Vadapalli, 2014), researchers propose a neural network dictionary learning architecture to induce task-specified word embedding representations and show that these features perform better at PB prediction task. (Vadapalli, 2016) presents their investigations of recurrent neural networks (RNNs) for the phrase break prediction task by using word embedding. The above efforts have also been directed toward unsupervised methods of inducing word representations, which can be used as surrogates for POS tags, in the PB prediction task.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2448–2455
Santa Fe, New Mexico, USA, August 20-26, 2018.

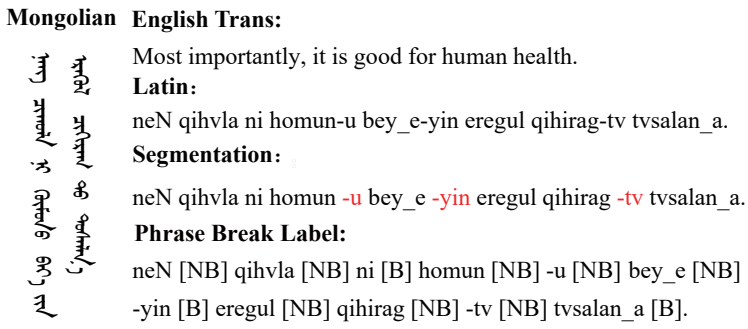


Figure 1: NNBS suffixes within a Mongolian sentences, the red part is the segmented NNBS suffixes from the word. There are three pauses in the sentence, one of which is located at the NNBS suffix: "-yin".

Although the word embedding training operates in an unsupervised way, this approach face an issue when applied to Mongolian languages, which are agglutinative in nature and the available Mongolian corpus is not large enough for the huge Mongolian vocabulary. Fortunately, Mongolian is a morphologically rich language. Its suffixes often act as a positive signal which implies the POS information of the word. It's like that the word implied by the suffix '-ly' is an adverb in English. Morphologically, unlike many other languages, a Mongolian word is not just a concatenation of characters. It is constructed by the special agglutinative property. Mongolian words can be decomposed into a set of morphemes: one root and several suffixes.

In this paper, we investigate Mongolian PB prediction models that operate on the level of sub-word units: stem and suffixes (the part without suffix). We hypothesize that stem and suffix serve to discriminate words based on syntactic meaning, and that these sub-word units can be used to model PB. We automatically segment every Mongolian word to a sequence of sub-word units, then map all sub-word units into a continuous vector representations by lookup table, which are then fed into a neural network. Instead of a feed-forward network, we use the Long Short-Term Memory (LSTM) network to predict the right PB label. The segmentation process reduces the vocabulary and alleviates the data sparse problem. Therefore, the learned word embedding for sub-word is more robust, then the performance of the PB prediction system can be improved.

Our experiments show the proposed model can achieve significant performance than the conventional CRF-based models, and the sub-word embedding based method outperforms the entire word embedding based method.

2 Mongolian characteristics

As an agglutinative language, like Turkish, Japanese and Korean, Mongolian has complex morphological structure. Most Mongolian words can be decomposed into root, derivational suffixes, and inflectional suffixes (Bao, 2013). The first two parts together are called a word-stem, it holds the major information in a word, and inflectional suffixes server to discriminate words based on lexical meaning. As for nouns, the inflectional suffixes contain case suffixes, reflexive suffixes, and plural suffixes. All above three suffixes are attached to stem through a Narrow Non-Break Space (NNBS) (U+202F, Latin: "-"). We call them NNBS suffixes. The NNBS suffixes used are very pervasive, in Fig.1, there are 3 NNBS suffixes in a sentence with only 8 words.

New words can be formed by connecting different suffixes to the end of a stem. A lot of new words can be induced from a single stem. For example:  ,  ,  ,  ,  . These words share the same word-stem "" (Latin: "sandali", means: "chair"). It makes Mongolian has a huge vocabulary, about one million, with only about 30 thousands stems (Bao, 2013). The large vocabulary leads to data sparse, and a serious dependence on a large corpus. However, the available Mongolian corpus is not enough for the word embedding training. The bad word embedding further reduces the accuracy of the PB prediction system. To get through the problem, we segment the NNBS suffixes from the Mongolian

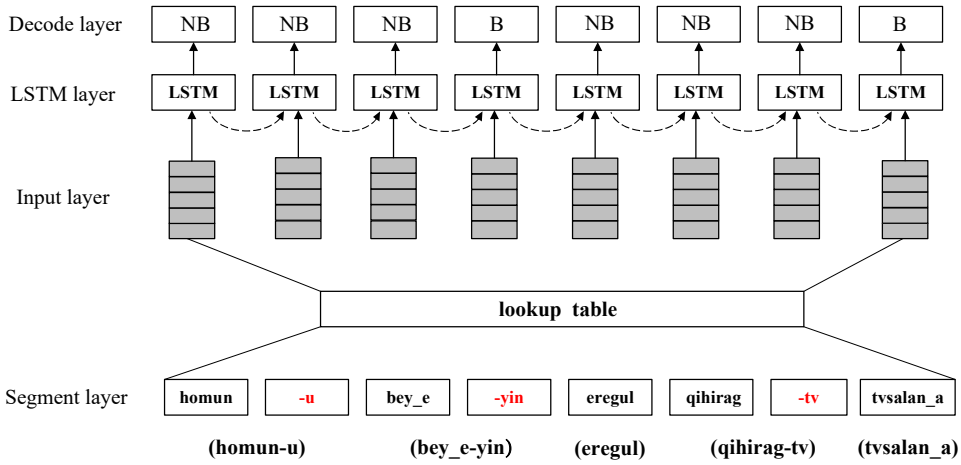


Figure 2: A framework for the proposed Mongolian PB prediction system based on sub-word embedding. The segment layer convert the Latin-cased Mongolian word to its sub-word form: stem (black part) and suffixes (red part); the sub-word embeddings are generated by a lookup table; the information is passed through a LSTM layer and the decode layer.

word and learn embedding representation on these individual suffixes and stems. After segmentating, the sentence will include more tokens, for example, in Fig.1, 3 words with NNBS will be turned into new units: "*homun-u*" turned into "*homun*" and "*-u*", "*bey_e-yin*" turned into "*bey_e*" and "*-yin*", "*qihirag-tv*" turned into "*qihirag*" and "*-tv*".

3 Proposed model

Our system framework is shown in Fig.2. This architecture consists of a segment layer, an input layer, a LSTM layer and a decode layer.

The system input is raw Mongolian sentence consisting of entire word. First, the segment layer converts every Mongolian word into stem and NNBS suffixes according to the location of NNBS inside of the word. Second, the input layer maps these processed Mongolian sub-word units into sub-word embeddings. The remaining layers are a LSTM network (Greff, 2017) used as a discriminative classifier and a decode layer to obtain the final PB label: "B" or "NB". "B" and "NB" are PB labels means *break after a word* and *non-break* respectively.

3.1 Sub-word embedding

In current work, we hypothesize that stem and suffix serve to discriminate words based on semantic meaning in Mongolian, and that these sub-word units can be used to model PB. We learn the embedding representation for sub-word units inspired the word embedding technique.

Word embedding represents words as continuous vectors in a low-dimensional space based on the distributional hypothesis that words in similar contexts have an analogous meaning. Based on this hypothesis, various word embedding models have been developed, including continuous bag-of-words model (CBOW), Skip-Gram model (Mikolov, 2013), and Global C&W(Glove) (Pennington, 2014). We use Skip-Gram model to train the sub-word embedding representation. Given a sequence of training unit $u_1, ..., u_T$, the Skip-Gram model try to maximize the average of log probability:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c}\log P\left(u_{t+c}|u_t\right) \tag{1}$$

where c is the training context around the center unit u_t. The prediction probability can be defined as:

$$P(o|i) = \frac{\exp(V_o^{\mathrm{T}}W_i)}{\sum_{u=1}^{U}\exp(V_u^{\mathrm{T}}W_i)} \tag{2}$$

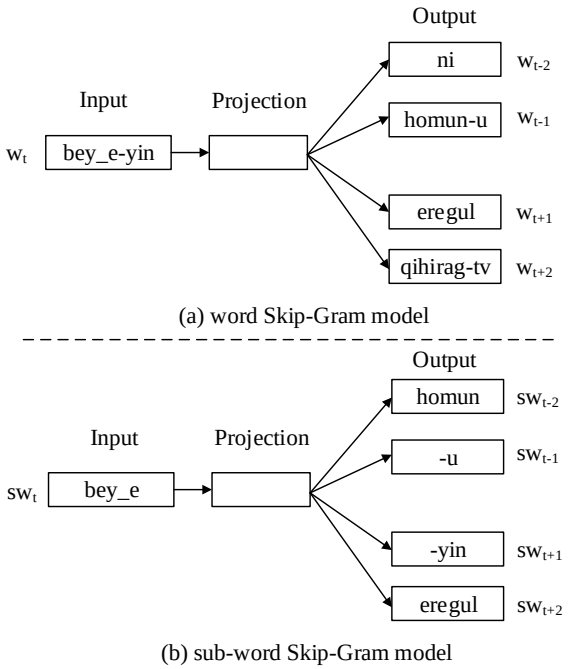


Figure 3: Comparision of the Mongolian Skip-Gram model representation of word unit (a) and sub-word unit (b). The sub-word Skip-Gram model regard stem and suffix as a basic unit, it allows the model to learn more information from suffixes with same context windows.

where W and V are the "input" and "output" unit vector representations of u, and U is the set of sub-word units. X^{T} is the transpose of matrix X.

The difference between word and sub-word Skip-Gram model is shown in in Fig.3 for the sentence: " *homun-u bey_e-yin eregul qihirag-tv tvsalan_a*". In the word Skip-Gram model, if the center word is "*bey_e-yin*", the nearby words are "*ni*", "*homun-u*", "*eregul*" and "*qihirag-tv*". While in sub-word Skip-Gram model, when the basic learning unit is changed to sub-word, the center word is turned to "*bey_e*", the nearby words to "*homun*", "*-u*", "*-yin*" and "*eregul*". The sub-word Skip-Gram model lies in dealing with sub-word units (stem and suffixes). It captures more information from the nearby suffixes under same context window size.

3.2 LSTM layer & Decode layer

PB prediction can be treated as a sequential labeling task that assigns boundary labels to words of an input sentence. Recurrent neural networks (RNNs) have recently produced outstanding performances on many tasks including sequential labelling (Vadapalli, 2016). In theory, RNN can learn from the entire historical inputs. But in practice, it can access only a limited range of context because of the vanishing gradient problem. LSTM uses purpose-built memory cells to store information, which is designed to overcome this problem. LSTM is composed of a set of recurrently connected memory blocks and each block consists of one or more self-connected memory cells and three multiplicative gates, i.e., input gate, forget gate and output gate. The three gates are designed to capture long-range contextual information by using nonlinear summation units.

Specifically, in this study, we follow the LSTM with forget gates and peephole connections to predict Mongolian phrase break. We use the following implementation:

$$i_t = \sigma(W_{xi}x_t) + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{3}$$

$$c_t = (1 - i_t) \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{4}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

where $\odot$ indicates element-wise product and σ indicates element-wise sigmoid function. More detail description can be found in (Greff, 2017).

The LSTM is trained using standard backpropagation through time to maximize the data conditional likelihood:

$$\prod_t P(y_t|x_1 \cdots x_t) \tag{7}$$

where x_t, y_t are the input and output respectively at time t. As mentioned in Section 3.1, the input x_t to the LSTM at time t, is the sub-word embedding corresponding to the token at time t.

The probability distribution is strictly a function of the hidden layer activations, which in turn depend only on the inputs (and their own past values). Thus, the most likely sequence of phrase break labels can be computed as:

$$y_t^* = \arg\max P(y_t|x_1 \cdots x_t) \tag{8}$$

4 EXPERIMENTS AND RESULTS

4.1 Dataset

4.1.1 Embedding corpus

The embedding train data were crawled from mainstream websites in Mongolian. After cleaning web page tags and filtering longer sentences, its token size and vocabulary are about 200 million and 3 million respectively. After we split the suffixes into a new token, we find a dramatic decrease on vocabulary even with the token number growth. The segmented corpus' token size and vocabulary are about 300 million and 2.5 million respectively.

4.1.2 PB prediction corpus

To evaluate the proposed method, a Mongolian speech synthesis corpus is involved. This corpus is recorded by a native Mongolian female speaker, who is a television news announcer. This corpus contains 58,695 utterances, where there are about 449,000 Mongolian words and 22,050 vocabularies. This corpus is labeled with prosodic phrase boundary by hand according to its speech. The labels correspond to the actual stops in the utterance. Specifically, each word is labeled as "B" (means *break*) if there is a short break after the word. Otherwise, the word is labeled as "NB" (means *non-break*). We divide the corpus into suitable subsets for training, validation and test as 8:1:1.

4.2 Evaluation

We conduct three sessions of experimentation. The first session is designed to verify the effectiveness of sub-word method under the CRF model, which aims to investigate the performance of regard the stem and suffixes as individual tokens. The second session about CRF-based systems is built to evaluate the idea of replacing the POS with the embedding representation for word and sub-word. The third session aims to investigate whether utilizing the LSTM model with sub-word embedding can improve the performance of Mongolian PB prediction. All Mongolian words are Latin-cased before passing through the lookup table to convert to their corresponding embeddings.

We evaluate the systems with F-Measure of the PB prediction, which this the harmonic mean of the *Precision* and the *Recall*. F-Measure values range from 0 to 1. The higher F-Measure means better PB prediction performance.

4.2.1 CRF with subword

We use CRF++ toolkit[1] to build a CRF-based Mongolian phrase break prediction system as a baseline named "CPw". "CPw" system consider the nearby words and its POS feature within the fixed context window size. Another two CRF-based systems are built to analyze the effects of the sub-word named "CPs" and "CPB", which have the same configuration as the CRF baseline, except the word feature. The 'CPs' system removes the segmented suffixes from the input token. The 'CPB' system segments the

[1] *https://taku910.github.io/crfpp/*

word into stem and suffixes, and uses them both as individual input tokens. All the CRF models used in this paper is a linear-chain CRF, we carry on all experiments under two context windows type: Unigram (U: previous one word, current word and future one word) and Bigram (B: previous two words, current words and future two words).

As illustrated in Table 1. Compared with the 'CPw' baseline under all context windows type, we can see the performance of the 'CPs' drop down. It is because that the word stems have less discriminative information than the entire word, since its suffixes are removed. 'CPB' outperforms the baseline and achieves the peak performance (82.96%) under Bigram context window. It can learn all semantic information from the individual suffix and stem but also alleviate the data sparse problem in a limited corpus.

Model	**F-Measure (U)**	**F-Measure(B)**
CPw	82.23	82.40
CPs	82.12	82.34
CPB	82.52	**82.96**

Table 1: Performance of F-Measure for CRF-based model with different context window types. (U: Unigram, B: Bigram)

	F-Measure (U)					**F-Measure (B)**				
Dim / **Model**	50	100	150	200	300	50	100	150	200	300
CEw	82.67	82.89	82.85	82.81	82.73	82.86	82.92	82.98	82.87	82.78
CEs	82.67	82.70	82.73	82.68	82.59	82.65	82.73	82.83	82.80	82.79
CEB	83.45	83.59	83.68	83.44	83.53	83.66	83.72	**83.79**	83.68	83.55

Table 2: Performance of F-Measure for CRF-based model using embeddings for word and sub-word with different dimensions and different context window types. (U: Unigram, B: Bigram, Dim: embedding dimension)

	F-Measure				
Dim / **Model**	50	100	150	200	300
LEw (Vadapalli, 2016)	85.63	85.74	85.89	85.78	85.64
LEs	84.78	84.83	85.01	84.88	84.78
LEB **(proposed)**	89.51	89.77	**89.89**	89.79	88.93

Table 3: Performance of F-Measure for LSTM-based model using embeddings for word and sub-word with different dimensions. (Dim: embedding dimension)

4.2.2 CRF with sub-word embedding

In these systems, the POS feature is replaced by the embedding, i.e. the systems input is the word or sub-word unit and its corresponding embedding. And follow the experimental setting of the Section 4.2.1, we index the three system as 'CEw', 'CEs' and 'CEB' respectively, which denotes CRF model using embedding feature on the entire word, word stem or both the suffix and stem. We test the performance of all systems with five embeddings dimension: 50, 100, 150, 200, 300. The evaluation results are listed in Table 2.

Comparing these three systems, we get the same conclusion as the previous experiment, sub-word based methods performance is better than the entire word setting. For Bigram context, 'CEB' achieves the highest F-Measure (83.79%) in all embedding dimensions. The performance of all systems is first

raised and then decreased with the embedding dimension range from 50 to 300, and reaches the best in 150. While a too long dimension will include other boring information that decoder cannot utilize, a too small dimension can not learn enough information from context. More informative, the performance of the Unigram context is worse than that of the Bigram context but shows the same trend.

The systems using sub-word embedding feature performs better than the systems utilizing POS feature (Section 4.2.1). As can be seen, the performances of 'CEw', 'CEs', 'CEB' are obviously higher than that of 'CPw', 'CPs', 'CPB'. This is mainly caused by the representation power of the word embedding technique. By using the sub-word embedding, we make a better use of the very limited training data in Mongolian. And again the proposed sub-word method alleviate the data sparse problem for both the word embedding training and the PB prediction models training.

4.2.3 LSTM with sub-word embedding

In this experiment, we replace the CRF model with the powerful LSTM model. All of the LSTM models used a single hidden layer of 512 units. All models are trained with a momentum of 0.3, an initial learning rate of 0.01. We select *tanh()* as our activation function, the minibatch size and forget gate bias is 1, weight and inner cells initialization are glorot uniform (Glorot, 2010) and orthogonal (Saxe, 2013) respectively. We train the LSTM model 50 epochs according to the development set. The evaluation results are listed in Table 3 under the name of 'LEw', 'LEs' and 'LEB', which means LSTM model using word embedding feature on the entire word (Vadapalli, 2016), sub-word embedding feature on word stem or both the suffix and stem.

Compared with CRF-based systems (Section 4.2.2) the LSTM-based systems show a clear advantage. 'LEw', 'LEs', 'LEB' systems respectively increased the performance by 2.91%, 2.18%, and 6.1% compared with 'CEw', 'CEs', 'CEB' under the optimum embedding dimension – 150. Our proposed method ('LEB') obtains 7.49% F-Measure gain compared with the baseline system - 'CPw' (Section 4.2.1) under 150 embedding dimensions. It is another evidence of the power of the LSTM model. LSTM model is more suitable than the CRF model in the PB prediction task. It shows that the LSTM model can fully absorb the nutrients of the embedding representation and get more benefits from the sub-word embedding.

5 CONCLUSIONS

In this paper, we look at sub-word units and explore the use of sub-word embedding on stem and suffixes to model Mongolian phrase break by using LSTM network. Embedding representation for the sub-word unit is learned in an unsupervised manner from an untagged Mongolian text corpus. These sub-word units can be directly identified from the text with a simple and effective approach. It provides more information for model Mongolian PB and eliminates the need for additional manually features like part-of-speech (POS) taggers. Experimental results demonstrate by means of the objective measure that LSTM model built using these sub-word embeddings perform significantly improvement than conventional CRF model built using POS sequence information. This work can also inspire other agglutinative language research.

Acknowledgements

This research was supports by the China national natural science foundation (No.61563040, No.61773224), Inner Mongolian nature science foundation (No. 2016ZD06) and the Enhancing Comprehensive Strength Foundation of Inner Mongolia University (No. 10000-16010109-23).

References

Lafferty, John and McCallum, Andrew and Pereira, Fernando CN. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Qian, Yao and Wu, Zhizheng and Ma, Xuezhe and Soong, Frank. 2010. Automatic prosody prediction and detection with Conditional Random Field (CRF) models. *Chinese Spoken Language Processing (ISCSLP), 2010 7th International Symposium on*, 135–138.

Bordes, Antoine and Chopra, Sumit and Weston, Jason. 2014. Assigning phrase breaks from part-of-speech sequences. *arXiv preprint arXiv:1406.3676*.

Xiong, Caiming and Zhong, Victor and Socher, Richard. 2017. Dynamic coattention networks for question answering. *ICRL*.

Devlin, Jacob and Zbib, Rabih and Huang, Zhongqiang and Lamar, Thomas and Schwartz, Richard M and Makhoul, John. 2014. Fast and Robust Neural Network Joint Models for Statistical Machine Translation. *ACL*, 1370–1380.

Wang, Peilu and Qian, Yao and Soong, Frank K and He, Lei and Zhao, Hai. 2015. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *arXiv preprint arXiv:1510.06168*.

Watts, Oliver and Yamagishi, Junichi and King, Simon. 2011. Unsupervised continuous-valued word features for phrase-break prediction without a part-of-speech tagger. *Twelfth Annual Conference of the International Speech Communication Association*.

Vadapalli, Anandaswarup and Prahallad, Kishore. 2014. Learning continuous-valued word representations for phrase break prediction. *Fifteenth Annual Conference of the International Speech Communication Association*.

Liu, Rui and Bao, Feilong and Gao , Guanglai and Zhang,Hongwei. 2015. Approach to Predicition Mongolian Prosody Phrase Based on CRF Model. *Proceedings of the National Conference on Man-Machine Speech Communication*.

Bao, Feilong and Gao, Guanglai and Yan, Xueliang and Wang, Weihua. 2013. Segmentation-based Mongolian LVCSR approach. *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8136–8139.

Mikolov, Tomas and Sutskever, Ilya and Chen, Kai and Corrado, Greg S and Dean, Jeff. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 3111–3119.

Saxe, Andrew M and McClelland, James L and Ganguli, Surya. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Vadapalli, Anandaswarup and Gangashetty, Suryakanth V. 2016. An Investigation of Recurrent Neural Network Architectures Using Word Embeddings for Phrase Break Prediction.. *Interspeech*, 2308–2312.

Pennington, Jeffrey and Socher, Richard and Manning, Christopher. 2014. Glove: Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Mikolov, Tomas and Chen, Kai and Corrado, Greg and Dean, Jeffrey. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

S.Loglo and HuaShabao and Sarula. 2010. Research on Mongolian Lexical Parser Algorithm Based on FNA. *Proceedings of the The Second National Symposium on Multi-lingual Knowledge Base Construction*.

Glorot, Xavier and Bengio, Yoshua. 2010. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256.

Greff, Klaus and Srivastava, Rupesh K and Koutník, Jan and Steunebrink, Bas R and Schmidhuber, Jürgen. 2017. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*.

Synonymy in Bilingual Context: The CzEngClass Lexicon

Zdeňka Urešová Eva Fučíková Eva Hajičová Jan Hajič
Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské nám. 25, CZ-11800 Prague, Czech Republic
{uresova,fucikova,hajicova,hajic}@ufal.mff.cuni.cz

Abstract

This paper describes CzEngClass, a bilingual lexical resource being built to investigate verbal synonymy in bilingual context and to relate semantic roles common to one synonym class to verb arguments (verb valency). In addition, the resource is linked to existing resources with the same or a similar aim: English and Czech WordNet, FrameNet, PropBank, VerbNet (SemLink), and valency lexicons for Czech and English (PDT-Vallex, Vallex and EngVallex). There are several goals of this work and resource: (a) to provide gold standard data for automatic experiments in the future (such as automatic discovery of synonym classes, word sense disambiguation, assignment of classes to occurrences of verbs in text, coreferential linking of verb and event arguments in text, etc.), (b) to build a core (bilingual) lexicon linked to existing resources, serving for comparative studies and possibly for training automatic tools, and (c) to enrich the annotation of a parallel treebank, the Prague Czech English Dependency Treebank, which so far contained valency annotation but has not linked synonymous senses of verbs together. The method used for extracting the synonym classes is a semi-automatic process with a substantial amount of manual work during filtering, role assignment to classes and individual class members' arguments, and linking to the external lexical resources. We present the first version with 200 classes (about 1800 verbs) and evaluate interannotator agreement using several metrics.

1 Introduction

Lexical resources, despite the fast progress in building end-to-end systems based on deep learning and artificial neural networks, are an important piece of the puzzle in Computational Linguistics and Natural Language Processing (NLP). They provide information that humans need to understand relations between words as well as the usage of these words in text. In addition, they can help various NLP tasks. This is why lexicons like WordNet (Miller, 1995; Fellbaum, 1998; Pala et al., 2011), FrameNet (Baker et al., 1998; Fillmore et al., 2003), VerbNet (Schuler, 2006), PropBank (Palmer et al., 2005) or EngVallex (Cinková, 2006; Cinková et al., 2014) have been created. They are for English, but there are also similar resources for other languages, often in a multilingual setting: WordNet is available in many languages (Vossen, 2004; Fellbaum and Vossen, 2012), Predicate Matrix (Lopez de Lacalle et al., 2016) extend coverage of several verbal resources and adds more romance languages, FrameNet has been extended to multiple languages (Boas, 2009), or there is the bilingual valency lexicon CzEngVallex for the case of Czech and English (Urešová et al., 2016).

One might thus question why it is necessary to develop another resource, and not just to connect some additional information to one or more of existing resources. After a thorough review (cf. also Sect. 2), it appears that for verbal synonymy, none of those resources fully corresponds to our goal of providing a lexicon of verbal synonyms based on at least semi-defined/semi-formal criteria, supported by real (parallel) texts usage. In addition, most of those resources have been first created for English and as some of the above publications acknowledge, there are then challenges to extend these resources to other languages (Fellbaum and Vossen, 2012).

Proceedings of the 27th International Conference on Computational Linguistics, pages 2456–2469
Santa Fe, New Mexico, USA, August 20-26, 2018.

In this paper, we describe CzEngClass, a lexical resource containing verbs in synonym groups ("classes") that is based on certain criteria that help to determine membership in such classes. These criteria use information from valency lexicons and semantic frames, and rely on actual examples of use in a bilingual context. The lexicon is thus built "bottom up", with emphasis on corpus evidence. In the resulting lexicon, links to existing lexical resources are added.Through these links, all entries also refer to examples from real bilingual texts with rich annotation, providing additional syntactic and morphological information for all CzEngClass entries.

The paper is structured as follows. In Sect. 2, related literature is discussed in more detail, while the resources actually used or referred to are described in Sect. 4. The criteria used for determining synonym class membership, specifically in bilingual context, are specified in Sect. 3. The resulting lexicon structure is presented in Sect. 5. Sect. 6 contains a description of the annotation process, while the interannotator agreement reached so far is analyzed in Sect. 7. We summarize some open questions and outline future work in Sect. 9.

2 Related work

Due to the needs of various NLP tasks, attention is paid to building interlinked resource(s) which integrates semantic information by mapping semantic knowledge from the individual lexical resources. One of the goals is to establish semantic interoperability between various semantic databases. As examples, we can name the SemLink (Palmer, 2009; Bonial et al., 2013) which provides manual mappings of four lexical resources (PropBank, VerbNet, FrameNet and WordNet), while the Predicate Matrix (Lopez de Lacalle et al., 2016) integrates and extends semantic information included in SemLink via automatic methods. BabelNet (Navigli and Ponzetto, 2012) integrates the largest multilingual Web encyclopedia(s), but mostly concetrates on "facts". Besides a whole range of monolingual synonym lexicons such as Roget's thesaurus (PSI and Associates, 1988) or WordNet (Miller, 1995; Fellbaum, 1998) for English, there are also cross-lingual synonym resources such as EuroWordNet, linking WordNet synsets across languages (Fellbaum and Vossen, 2012). WordNet contains a rich network or relations between its synsets (hyperonymy, hyponymy, ...), however, it does not contain information about syntactic and compositional semantic behavior, which is a drawback especially in case of verbs.

Our approach to verbal synonymy builds on previous research which also considered multilingual data (translations) in parallel corpora as an important resource. Translational context is regarded as a rich source of semantic information (de Jong and Appelo, 1987; Dyvik, 1998; Adamska-Sałaciak, 2013; Andrade et al., 2013). Parallel corpora have also been used (Resnik, 1997; Ide, 1999; Ide et al., 2002) for automatic methods for sense induction and disambiguation, for cross-lingual similarity detection and synonym extraction (Wu et al., 2010; Wu and Palmer, 2011). (Wang and Wu, 2012) studies various assumptions about synonyms in translation, for the purpose of trend detection from titles. While these works aim at automatic methods and applications, they share the idea that if two words are semantically similar in a language, their translations in another language would be also similar. Translations of a word from another language are often synonyms of one another (Lin et al., 2003; Wu and Zhou, 2003). A similar idea, i.e., that words sharing translational context are semantically related, can be found in (Plas and Tiedemann, 2006).

Interannotator agreement evaluation is regularly used in corpus annotation. However, it is much more scarcely used in lexicon entry creation and annotation, especially in a multilingual setting. For assignment of topics to words in Hindi and English, see (Kanojia et al., 2016). More detailed account on the influence of semantic lexical granularity within the Context Pattern Analysis paradigm on interannotator agreement can be found in (Cinková et al., 2012).

3 Synonymy in bilingual context

Synonymy in bilingual context is closely related to translational equivalence, sameness, similarity, meaning, word sense, etc. These terms - and the term synonymy itself - are not always used in an unambiguous way. We thus discuss the terminology first, and then specify how we define verbal synonymy in bilingual context in the work on building the CzEngClass lexicon.

3.1 Terminology

Although Cruse (1986) notes that "there is unfortunately no neat way of characterising synonyms", the notion of synonymy is mostly seen as "sameness or identity of meaning" (Palmer, 1976; Sparck Jones, 1986). Leech (2012) restricts synonymy to equivalence of conceptual meaning. *Synonym* is mostly defined as "a same-language equivalent" (Adamska-Sałaciak, 2010; Adamska-Sałaciak, 2013) and "does not exceed the limits of a single language" (Gouws, 2013), while for bilingual contexts the term *translational equivalent* is used. On the other hand, (Martin, 1960; Klégr, 2004; Hahn et al., 2005; Hayashi, 2012; Haiyan, 2015; Dinu et al., 2015) recognize interlingual synonymy and use either the term *foreign-language equivalent, cross-lingual synonym, synonymous translation equivalent* or *bilingual synonym*.

In building CzEngClass, we consider the relationship between the lexical unit of the source language (SL) and of the target language (TL) unit as a specific type of synonymy, an *interlingual* synonymy. For words from different languages which are interlingual synonyms, we prefer to use the term *bilingual synonyms*. We understand that the meaning correspondence does not mean absolute equivalence and automatic interchangeability. We agree with (Louw, 2012) that the translation equivalent might be a TL item that can only be a substitute for the SL item in one or in some of its uses and each equivalent has to be supported with additional contextual and co-textual restrictions that will allow the user to make an appropriate choice of an "equivalent" for a given usage situation. Accordingly, we believe that interlingual synonymy considerations are essential in the translation process because it is up to the translator to choose the most suitable expression among (intralingual) synonyms, based on context and the meaning of the SL text (Catford, 1965; Newmark, 1988).

Intralingual synonyms are often not interchangeable, i.e., not quite equivalent. Synonymy, as viewed by (Lyons, 1968; Cruse, 1986), has to be seen as a scale of similarity (absolute, near and partial synonymy) and it is generally acknowledged that absolute synonymy is rare in natural languages. Therefore, it is believed that context must be taken into account (Palmer, 1981). Such synonyms are then called contextual synonyms (Zeng, 2007) or contextual correlates of synonymy (Rubenstein and Goodenough, 1965) and described as words that are "synonymously" used in certain specific texts.

3.2 Definition of contextual synonymy in CzEngClass

Both types of synonymy (interlingual and intralingual) are captured in the CzEngClass lexicon, which aims to group verbs into synonym classes both monolingually and cross-lingually. Along with (Palmer, 1981), we believe that synonymy can only be considered in context. We define two (or more) verb senses to be *bilingual synonyms* if they both (or all) convey the same meaning in a given particular context. Similarly, for the intralignual case, we work with the "loose" interpretation of synonymy (Lyons, 1968; Palmer, 1981), and consider context as a key factor that helps to overcome the vagueness of such "looseness".

In CzEngClass, *context* is defined as the set of semantic roles (SRs) that the given verb, as a member of a bilingual synonym class, expresses by its arguments and/or adjuncts, or which are implicitly present, possibly with additional structural or semantic restrictions. Each class has an associated, single (common) set of SRs while such a set is shared by all its members, even if each SR can be expressed (*mapped* to) by a different argument (or by an adjunct, or implicitly or explicitly in the verb's dependent substructure) for different verbs as members of that class. Conversely, such mapping must exist at least for all obligatory valency slots as defined in the two corresponding valency lexicons. To keep each class focused, only a relatively small set of SRs is usually assigned to it (corresponding roughly to "core" Frame Elements in FrameNet, even though the labels might not match).[1]

Such focus then helps to answer the question of candidate verb membership in a particular synonym class. If a mapping of all arguments to the SRs associated with that class is found, as well as a mapping

[1] In fact, there is one substantial difference between the intended properties of CzEngClass's SRs and those found in FrameNet: while FrameNet explicitly states that SRs (FEs) from one Frame, even if under the same label, should not be construed as being the same as equally labeled SR (FE) in a different Frame, we would like to use such a set of SRs that *is* labeled consistently across CzEngClass classes. For the moment, however, we resort to VerbNet thematic roles for the most common SR "slots", such as Agent and to a certain extent also Theme, renaming them in the future consistently once we have a larger set of classes ready. For more details on the process, see Sect. 6.

of all SRs of that class to the candidate verb arguments, adjuncts (or implicit, or even more deeply embedded dependents), and it does not violate any of the associated restrictions, then the candidate verb is considered to be a valid member of such class. Since CzEngClass is built mainly manually,[2] it is important to establish such criteria in order to achieve higher interannotator agreement (IAA) as well as for guiding the adjudication process. Table 1 illustrates one possible class and the context (SR mappings) for each member.

	Roles		
	Speaker_or_Event	Addressee	Content
povzbudit	ACT	ADDR	PAT
encourage	ACT	ADDR	PAT
galvanize	ACT	ADDR	PAT
inspire	ACT	ADDR	PAT
prod	ACT	ADDR	PAT
inspirovat	ACT	PAT	AIM
nabádat	ACT	ADDR	PAT
pobídnout	ACT	ADDR	PAT
podněcovat	ACT	ADDR	PAT
podpořit	ACT	ADDR	PAT
povzbuzovat	ACT	ADDR	PAT
vést	ACT	PAT	AIM

Table 1: Mappings for ENCOURAGE class

The following (parallel) example from the PCEDT illustrates the SR/argument use and alignment:

En: *Beth Marchand says Mrs. Yeargin*.ACT/Speaker_or_Event *inspired her*.ADDR/Addressee *to go*.PAT/Content *into education*.

Cz: *Beth Marchandová říká, že ji*.PAT/Addressee *učitelka Yearginová*.ACT/Speaker_or_Event *inspirovala, aby se dala*.AIM/Content *na studium vzdělávání.*

4 Resources used

The resources used to create the CzEngClass lexicon are divided into two groups - primary and secondary.

Primary resources are those used for word sense disambiguation and valency information when creating the class member candidates. We use the parallel Prague Czech-English Dependency Treebank 2.0 (PCEDT 2.0) (Hajič et al., 2012) with its associated lexicons. This treebank contains over 1.2 million words in almost 50,000 sentences for each language. About 90,000 tokens are verbs on each side. The English part contains the entire Penn Treebank - Wall Street Journal Section (Marcus et al., 1993). The Czech part is a manual translation of all the Penn Treebank-WSJ texts to Czech. PCEDT is annotated using The Prague Dependency Treebank style (Hajič et al., 2006; Hajič et al., 2018) manual linguistic annotation based on the Functional Generative Description framework (Sgall et al., 1986). Our research benefits primarily from the deep syntactico-semantic (tectogrammatical) dependency trees, interlinked across the two languages on sentence and node (content word) levels. Each deep syntax verb occurrence in the PCEDT is linked to the corresponding valency frame (predicate-argument structure frame) in the associated valency lexicons, PDT-Vallex (Urešová et al., 2014; Urešová, 2011) and EngVallex (Cinková, 2006; Cinková et al., 2014), effectively providing also word sense labeling. The parallel bilingual valency lexicon CzEngVallex (Urešová et al., 2016; Urešová et al., 2015), built over the PCEDT, is also heavily used, both in the annotation process itself but also for automatic preannotation.

Secondary resources are the well-known English lexical resources: FrameNet (Baker et al., 1998; Ruppenhofer et al., 2006), FrameNet+ (Pavlick et al., 2015), VerbNet and OntoNotes (Schuler, 2006; Pradhan et al., 2007), SemLink (Palmer, 2009; Bonial et al., 2012), PropBank (Palmer et al., 2005), and English WordNet (Miller, 1995; Fellbaum, 1998). For Czech, we use Czech WordNet (Pala and Smrž, 2004) and VALLEX (Lopatková et al., 2016). These resources are used for the extraction of an initial set of SRs (taken from FrameNet and VerbNet), and most importantly, their entries (if possible to the level of

[2]With automatic preselection of candidate class members based on parallel corpora and the valency lexicons available for Czech and English, see Sect. 6.

exact lexical units / frames / synsets) are referred to (explicitly linked) from all the corresponding entries in the CzEngClass lexicon.

5 Lexicon Structure

The structure of CzEngClass is described in detail (Urešová et al., 2018a; Urešová et al., 2018d; Urešová et al., 2017a; Urešová et al., 2017b). Here we summarize only its main characteristics.

The CzEngClass lexicon is in principle a set of (bilingual synonym) classes. Each class contains both Czech and English verbs (class members), identified by their valency frame identifier (i.e., a link to the Czech and English valency lexicons that served as the initial sense inventory for each verb). For each class, the lexicon records also the set of semantic roles, which is common for the class. For each class member, its arguments (valency slots) are mapped to this set (not necessarily 1 : 1 - arbitrary $m : n$ mappings are allowed). For each member, additional restrictions (for the time being, in the form of a plain text description) are recorded as well. In addition, the lexicon also records, within each class, the original verb pairs as found aligned across the two languages in the PCEDT, and their argument alignment as coming from the CzEngVallex bilingual valency lexicon. All external links (to FrameNet frame(s), Ontonotes Sense Groupings, VerbNet type(s), WordNet synset(s)) are also recorded (see Sect. 4).

The lexicon, technically a single XML file with external references, also contains a header with all the SRs used and their description, annotator IDs, bookkeeping information about all the external resources in order to create a URL for each external link, and all entries also contain the usual annotation information (log with timestamps, etc.).[3]

6 The lexicon annotation process

First, we automatically aligned Czech verbs with their English verbal translations[4] as found in the PCEDT. There are two phases, each further broken down to several steps.

In Step 1 of the first phase, we have automatically extracted 200 Czech verbs[5] and their English translations. These 200 verbs have been selected to represent both high-, medium- and low-frequency verbs in the PCEDT. They have been found in 23,769 sentences, covering about half of the corpus.[6] Each Czech verb serves as a "seed" in the future bilingual synonym class. The English translations are the (first-phase, English-language) class member candidates. Using the principles of both intra-lingual and inter-lingual synonymy (Sect. 3), and with the help of comparing the valency frames of the Czech verb and the English verb and the English verbs among themselves, we then manually pruned these candidates, obtaining a list of (English) synonym class members.

In Step 2 of the first phase, we have used these English verbs and added, similarly to Step 1 but in an opposite direction and using the same corpus, additional Czech synonym class member candidates. These have been pruned manually again and a result of this pruning was a (first-phase) bilingual synonym class, built around the original Czech verb.

In the second phase, two tasks had to be carried out for all classes. First, to every class member the appropriate linking to English and Czech resources had to be provided. Using a dedicated class editor, we started with mapping every English verb sense in the class to Ontonotes Sense Grouping. Then, links to FrameNet, PropBank, WordNet and Czech VALLEX have been added.[7] Second, core SRs inventory (for each class) had to be created. We were mostly inspired by FrameNet's *frame elements* (FEs) resorting to more general VerbNet's *thematic roles* if FEs were deemed to be specific. For each class member, its valency arguments (taken from PDT-Vallex for Czech verbs and EngVallex for English ones) have been mapped to the appropriate SR from the set of SRs assigned to the whole class. So far, this second step has been performed for the first 60 classes only.

[3] Current version of CzEngClass: `http://hdl.handle.net/11234/1-2824`

[4] As already mentioned, we pay attention only to verbs.

[5] I.e., 200 verb senses represented by their valency frames as annotated in the data.

[6] Average number of sentences per Czech verb is 119, median is 26. These sentences may contain also other verbs.

[7] Links to Czech WordNet will be added later.

The process described above was first tested and fine-tuned by two annotators, and then a detailed testing (so far, of Step 1 of the first phase), as reported in Sect. 7 below, has been performed by eight annotators.

7 Evaluation: interannotator agreement

Using the first part of the annotation process (filtering of the precomputed list of English verbs for a given Czech verb, in a particular sense), an IAA evaluation scheme has been set up (Table 2).

	No. of classes (Czech verbs)	No. of English verbs (sum over all clases)
Part 1	15	131
Part 2	15	134
Part 3	17	127
Part 4	13	152
Total	60	544

Table 2: Interannotator agreement experiment setup

As described in Sect. 6, the 60 Czech verbs (senses) have been selected in the PCEDT such that they represented high-, medium- and low-frequency verbs in roughly the same proportion.[8] For those 60 verbs, all their English translational verbal counterparts[9] aligned with them have been extracted automatically from the corpus, with the help of the CzEngVallex lexicon which refers to particular verb senses.This alignment is however not perfect for our purposes. The task of the annotators was to prune, for each Czech verb sense (i.e., the potential/future synonym class), the automatically preselected list of its English verbal translational equivalents so that only synonym candidates corresponding to the meaning of the Czech verbs as it appears in the bilingual corpus are preserved. For this experiment, the annotators have not been given the full definition of the synonym as presented here in Sect. 3. They have been instructed, for each pair of the Czech verb and its English synonym candidate, to check the corpus examples through the CzEngVallex valency argument alignments, i.e., the usage in context as defined in Sect. 3, but not to validate (yet) against the set of SRs for the given class.[10]

There have been 8 annotators, each of them annotating 2 different parts. 1 of the annotators is a professional translator, 2 of them are researchers in the field of Computational linguistics with a Ph.D. degree, and the remaining 5 are undergraduate students of various language and linguistic programs, all with high-level fluency in English. All annotations have been treated as equal.

Each English synonym class member candidate verb was annotated by 4 annotators (Table 2). The annotators have been asked to assign, to each candidate English verb, one of the five labels summarized in Table 3, corresponding to a scaled decision of how strong the annotators feel about including (or not including) the English word in the resulting synonym group.

Only the labels have been presented to the annotators, which have been then converted to the scale used for some of the evaluation metrics, as described below. The data for the interannotator agreement as described in this paper are available with the previous version of CzEngClass (Urešová et al., 2018b).[11]

7.1 Interannotator agreement: Cohen's kappa

Cohen's kappa, used for comparing annotations pairwise, is defined as follows:

$$\kappa = (p_0 - p_e)/(1 - p_e) \quad (1)$$

[8]This selection is to gain insight to the issues with verbs of all frequencies; overall, of course, the verbs with low frequency will prevail, making the statistics more favorable since classes for these verbs tend to have less members and thus higher IAA.

[9]We are aware of the fact that many verbs are translated by nouns or other kinds of phrases. These translational counterparts are not yet part of the CzEngClass.

[10]We understand this as a first step. The class membership might be–and surely will be–changed in the second step, namely, when assigning the SR mappings to valency slots. This is the most important "context" criterion defining the class membership based on our principles of synonymy as defined in Sect. 3, and specifically in Sect. 3.2.

[11]http://hdl.handle.net/11234/1-2823

Label	Explanation	Inclusion weight
Y	Yes	4
R_Y	Rather Yes	3
R_N	Rather No	2
N	No	1
D	Delete (Alignment or other system error)	0

Table 3: Labels assigned to each English verb by the annotators

where p_0 is the observed agreement, and p_e the expected agreement based on the two annotations. Results are summarized in Table 4. For determining agreement, we have mapped the labels assigned by the annotators to two values only: Y (if they used labels Y and R_Y) and N (if they used labels N, R_N and D), since Cohen's kappa would be hard to interpret if scaled values are behind the labels used.

Annotator	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
A_1	-	0.46	0.19	0.42		0.43	0.43	0.33
A_2		-	0.57	0.55	0.46	0.68		0.31
A_3			-	0.42	0.37		0.26	
A_4				-	0.35		0.45	
A_5					-	0.48	0.56	0.41
A_6						-	0.48	0.51
A_7							-	0.48
A_8								-

Table 4: Interannotator agreement: Cohen's kappa, pairwise

Due to the distribution of Parts 1-4 to annotators, not all pairs of annotators could be compared (empty cells in Table 4). The parts in any row or column are different for different annotators.[12]

While averaging over the annotators is not well defined within such pairwise distribution of data, we can see quite clearly that A_3 has overall the lowest agreement with the others, but the differences are not that high. The values are low, but this is due to the fact that the label assignment has been biased to assigning Y (i.e., to keep the verb in) in about an 5:1 ratio.

7.2 Interannotator agreement: Fleiss' kappa

As opposed to Cohen's kappa, Fleiss' kappa can be used in a multiannotator setup with n annotators and N datapoints (total of English verbs in the 60 classes annotated, i.e., 544 in our case):

$$\kappa = (\bar{P} - \bar{P}_e)/(1 - \bar{P}_e) \tag{2}$$

where

$$\bar{P} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{n(n-1)}(n_{iYes}(n_{iYes}-1) + n_{iNo}(n_{iNo}-1)), \tag{3}$$

$\bar{P}_e = {p_Y}^2 + {p_N}^2$, $p_Y = \frac{1}{Nn}\sum_{i=1}^{N} n_{iYes}$ and $p_N = \frac{1}{Nn}\sum_{i=1}^{N} n_{iNo}$. The n_{iYes} (n_{iNo}) counts are defined as the number of times the i-th datapoint has been annotated Y and N, respectively (i.e., sum of n_{iYes} and n_{iNo} is n). The Ys and Ns have been obtained by the same mapping as in the Cohen's kappa case: Y and R_Y mapped to Y and N, R_N and D mapped to N. Table 5 shows the intermediate and final values of Fleiss' kappa. The 0.45 value characterizes the "global" difficulty of the task; again, we have to stress that the bias is high (about 5:1), so that any disagreement lowers the kappa value substantially.

[12] Given the amount of effort available for the annotation, the four parts have been distributed in such a way that at least those pairs listed in the table could be covered.

p_Y	p_N	$\bar{P}$	$\bar{P}_e$	κ
0.84	0.16	0.85	0.73	**0.45**

Table 5: Interannotator agreement: Fleiss' kappa, global

Given the low kappa values, we have been interested in numbers that could perhaps be easier to interpret from the point of view of adjudication, which will inevitably follow to ensure high quality of the resulting lexicon. In addition, we are not convinced that these widely used interannotator metrics are appropriate for highly biased lexical task (as opposed for running text annotation in a corpus). We have thus measured two more things: consensus and deviation from the assumed correct value, both described below.

7.3 Annotator consensus

As a first step, we have measured consensus. Of the 544 datapoints the annotators assigned the label (again mapped to Y and N only), in 488 cases a majority consensus could be reached (89.7%); i.e., only in slightly over 10% of the cases the result was 2:2 (with 4 annotators working on each datapoint). This number naturally varies with the number of annotators (it would be higher for 2 annotators, and always 100% for an odd number of annotators unless a higher threshold is set for determining consensus), but when combined with full agreement among the 4 annotators (358 cases out of the 544, or 65.8%), it gives a good idea of the adjudication effort needed (for 4 annotators, in our case).[13]

7.4 Comparison based on the annotated scale

Since the annotation of class membership for the candidate English synonym verbs has been in fact on a scale 0-4 (see Table 3), we tried to compute a global (averaged) score of deviation from a centerpoint, namely the assumed correct value, computed either as a majority or average value as assigned to every English verb by the four different annotators that had labelled it. We have used two methods.

In the **first method**, we have determined the "correct" value by taking the majority annotation, restricted again to two values: Y(es) and N(o) (with mapping from the five different (scaled) annotated labels as in the kappa computation, Sect. 7.1). For computing the deviation on a particular datapoint, we use the same mapping. In other words, the deviation can only be 0 (agreement with Y or N) or 3 (disagreement). The averaged (absolute) deviation $DevA2_j$ for an annotator j is then defined as

$$DevA2_j = \frac{\sum_{i=1}^{N} |B_i - S_{ij}|}{N} \tag{4}$$

where B_i is the correct value (4 for Y, 1 for N) for verb i (out of N verbs, where $N = 544$ in our dataset), and S_{ij} is the mapped value on that i-th verb for annotator j.

The deviation computed on this mapped (Y/N) values is summarized for all annotators in Table 6; the average over all annotators is 0.27 (within a span of maximum possible difference of 3 (4-1)). Since the bias towards the Y label in the consensus labeling is about 5:1, giving everything Y would result in approx. average deviation $1/6 \times 3 = 0.5$, so the annotation cut this deviation to half.

Annotator	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	**Data avg.**
Avg. deviation	0.21	0.20	0.23	0.26	0.41	0.21	0.31	0.31	**0.27**

Table 6: Deviation from consensus value, mapping to 2 values only (Y/N)

The **second method**, which is meant for evaluation purposes only, does not select one consensus value, but simply averages the scaled labels over all (four) labels assigned by the annotators to every datapoint (verb to annotate) i:

[13] The total agreement has been measured using all labels; i.e., if 3 annotators assigned N and 1 assigned R_N, it did not count as full agreement.

$$P_i = (\sum_{k=1}^{n} v_{ik})/n$$

where n is a number of annotations for a given datapoint (i.e., always four in our case, since each verb has been annotated by 4 annotators), k runs over all labels assigned to the i-th datapoint, and v_{ik} is the k-th value (0-4, mapped from (D, N, R_N, R_Y, Y; see Table 3) assigned to i-th datapoint.

The deviation for each annotator is then computed as

$$DevA5_j = \frac{\sum_{i=1}^{N} |P_i - A_{ij}|}{N} \quad (5)$$

where P_i is the averaged value as defined above for verb i (out of N verbs, where $N = 544$ in our dataset), and A_{ij} is value assigned to the i-th datapoint by annotator j.

The averaged values are shown in Table 7.

Annotator	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	**Data avg.**
Avg. deviation	0.32	0.32	0.32	0.32	0.45	0.36	0.37	0.41	**0.36**

Table 7: Deviation from average assigned value, all 5 values used (0-4)

These deviations are (somewhat surprisingly) larger than the deviations computed after mapping the labels to the binary (Y/N) values only. In other words, the deviation is *smaller* when the "unsure" labels (R_Y, R_N) are mapped to Y and N, and the D value also to N, disregarding the difference between these two.

Since the final lexicon will not contain any manually assigned weights, i.e., a given verb will be considered either in a class or not in a class. we believe that the relevant numbers are those in Table 6.

While the two methods of computing interannotator "deviation" compute absolute difference between the annotator-assigned value and the "truth", we have been interested yet in another value, namely whether a given annotator tends to keep the preselected verbs in the class or not. This number thus does not describe annotator's distance from the consensus value or the average, but rather his average positive attitude (prefers to keep many verbs in) or a negative one (prefers to mark verbs as not belonging to the synonym class in question).

We compute the "truth" as a simple average of the graded labels for every verb:

$$DevS5_j = \frac{\sum_{i=1}^{N} (P_i - A_{ij})}{N} \quad (6)$$

The resulting value can now be also negative, meaning that the annotator assigned lower values on the 0-4 scale than average, and by how much.

Annotator	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
Avg. (signed) deviation	0.09	0.05	0.13	0.08	-0.19	0.07	-0.02	0.11

Table 8: Signed deviation from average assigned value, all 5 values used (0-4)

Table 8 shows the results. The deviation for 2 annotators, A_5 and A_7 is in the negative territory. These annotators, and specifically A_5, are considered "conservative", since their annotation keeps the synonym classes small - they prefer only clear synonyms to become included in the class. Annotator A_3, on the other hand, is the most "liberal" of all and leans towards including more words in the class.

8 Extending CzEngClass to Other Languages

An interesting question is what is necessary in order to create a similar lexical resource–using the approach described here–for other languages or to extend our resource by another language. Based on our

experience so far, we believe this is the minimal set of necessities to start with: a valency or predicate-argument lexicon for that language (a corresponding parallel resource - with English, such as our CzEng-Vallex lexicon - might be helpful but it is not essential) and a word-aligned and (at least automatically) syntactically analyzed parallel corpus (pairing the language in question to English, or to any language that is already included in the Class lexicon being extended). In addition, some word-sense-distinguishing lexicon (e.g., WordNet in that language) is also nice to have to link the class members to. Moreover, having additional lexical resources in the given language (such as FrameNet) would be an advantage that will help the consistency of annotation and the usefulness of the final, externally linked synonym lexicon, but this is not strictly necessary.

9 Conclusions and future work

We have described the first version of a semi-manually built bilingual synonym lexicon of verbs, CzEng-Class, based on a parallel corpus and existing lexical resources. The process of building the lexicon is relatively complex, and we have presented results of the first step of "soft" (scaled) lexical unit annotation by several annotators. We have evaluated their agreement by several metrics, giving us some indication of how much future effort will be needed for adjudication in order to obtain a high-quality resource.

These first findings indicate that the traditional measures for IAA show low agreement, primarily due to high bias in the automatic preselection which favors the preselected verbs to be kept (in 5:1 ratio).[14] Using a different measure based on deviation from consensus, we found that a full consensus has been achieved among 4 annotators in almost 2/3rd of test cases, and if we lower the criterion of agreement to 3 of 4 annotators, in 9 of 10 cases (Sect. 7.3). This is only when considering the verb sense and valency; adding the semantic roles, which is the next step, would likely lower that agreement. For example, we will explore more the assumption (and try to explicitly define where and when it can be used) that the valency slots to semantic roles mapping might need to be *m:n*, which of course greatly influences the interannotator agreement as opposed to the case where only *1:1* mapping is allowed.

The first version (200 classes, i.e., approx. 1800 class members, Phase 1 annotation) has been released publicly (Urešová et al., 2018c).[15]

Phase 2 annotation is ongoing, with semantic role assignment to classes and the detailed mapping of valency to these roles. Eventually, we will extend CzEngClass to all verbs found in the PCEDT. This is expected to go smoothly, perhaps with the usual exception of the verbs *to be*, *to have*, *to become* and *to do*. Otherwise, since we have already (intentionally) dealt with verbs of various frequencies and degrees of homonymy, representing a wide range of possible valency-to-semantic-roles mappings, we believe no substantial issues in the specification nor the annotation process should arise.

For the more distant future, we plan to use the core PCEDT-derived CzEngClass lexicon to do experiments on automatic extraction of synonym classes from other (larger) data, while using CzEngClass as a gold standard. If successful, we plan to extend CzEngClass to other verbs similarly to what the Predicate Matrix project did, but with the arguably best system based on the above experiments; in all cases, we plan for manual pass to obtain consistent classes with those done fully manually.

Acknowledgements

This work has been supported by the grant No. GA17-07313S of the Grant Agency of the Czech Republic. Part of this work was conducted during a fellowship (of the 4th author) at the Oslo Center for Advanced Study at the Norwegian Academy of Science and Letters. The data used in this work have been created and are maintained in the LINDAT/CLARIN digital repository (http://lindat.cz), supported by the Ministry of Education, Youth and Sports of the Czech Republic as projects No. LM2015071 and CZ.02.1.01/0.0/0.0/16_013/0001781. The creation of part of the data used has been also supported by the European Commission under the T4ME (META-NET) project No. FP7-ICT-2009-4-249119.

[14] Which confirms, as a collateral result, that the data from which this preselection has been done are of quite high quality in terms of verb and argument alignment.

[15] `http://hdl.handle.net/11234/1-2824` available under CC BY-NC-SA 4.0.

References

Arleta Adamska-Sałaciak. 2010. Examining equivalence. *International Journal of Lexicography*, 23(4):387–409.

Arleta Adamska-Sałaciak. 2013. Equivalence, Synonymy, and Sameness of Meaning in a Bilingual Dictionary. *International Journal of Lexicography*, 26:329.

Daniel Andrade, Masaaki Tsuchida, Takashi Onishi, and Kai Ishikawa. 2013. Synonym acquisition using bilingual comparable corpora. In *Sixth International Joint Conference on Natural Language Processing, IJCNLP 2013, Nagoya, Japan, October 14-18, 2013*, pages 1077–1081.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, pages 86–90, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hans C. Boas, editor. 2009. *Multilingual FrameNets in Computational Lexicography. Methods and Applications*. Trends in Linguistics. Studies and Monographs [TiLSM] 200. DE GRUYTER MOUTON.

Claire Bonial, Weston Feely, Jena D Hwang, and Martha Palmer. 2012. Empirically Validating VerbNet Using SemLink. In *Seventh Joint ACL-ISO Workshop on Interoperable Semantic Annotation*, Istanbul, Turkey, May.

Claire Bonial, Kevin Stowe, and Martha Palmer, 2013. *Proceedings of the 2nd Workshop on Linked Data in Linguistics (LDL-2013): Representing and linking lexicons, terminologies and other language data*, chapter Renewing and Revising SemLink, pages 9 – 17. Association for Computational Linguistics.

John Cunnison Catford. 1965. *A Linguistic Theory of Translation: an Essay on Applied Linguistics*. Oxford Univ. Press, London.

Silvie Cinková, Martin Holub, and Vincent Kríž. 2012. Optimizing semantic granularity for NLP - report on a lexicographic experiment. In Ruth Fjeld and Julie Torjusen, editors, *Proceedings of the 15th EURALEX International Congress*, number hardcopy ISBN: 978-82-303-2095-2, pages 523–531, Oslo, Norway. Department of Linguistics and Scandinavian Studies, University of Oslo, Department of Linguistics and Scandinavian Studies, University of Oslo.

Silvie Cinková, Eva Fučíková, Jana Šindlerová, and Jan Hajič. 2014. *EngVallex - English Valency Lexicon*. LINDAT/CLARIN digital library. `http://hdl.handle.net/11858/00-097C-0000-0023-4337-2`.

Silvie Cinková. 2006. From PropBank to EngVallex: Adapting the PropBank-Lexicon to the Valency Theory of the Functional Generative Description. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, pages 2170–2175, Genova, Italy. ELRA.

D.Alan Cruse. 1986. *Lexical Semantics*. Cambridge University Press, UK.

Franciska M.G. de Jong and Lisette Appelo. 1987. Synonymy and translation, 12.

Anca Dinu, Liviu P. Dinu, and Ana Sabina Uban. 2015. Cross-lingual synonymy overlap. In *Recent Advances in Natural Language Processing, RANLP 2015, 7-9 September, 2015, Hissar, Bulgaria*, pages 147–152.

Helge Dyvik. 1998. Translations as Semantic Mirrors: From Parallel Corpus to Wordnet. In *Proceedings of Workshop Multilinguality in the Lexicon II, ECAI 98*.

Christiane Fellbaum and Piek Vossen. 2012. Challenges for a multilingual wordnet. *Lang. Resour. Eval.*, 46(2):313–326, June.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. MIT Press, Cambridge, MA. 423 pp.

Charles J. Fillmore, Ch. R. Johnson, and M. R. L.Petruck. 2003. Background to FrameNet: FrameNet and Frame Semantics. *International Journal of Lexicography*, 16(3):235–250.

Rufus H. Gouws. 2013. Contextual and co-textual guidance regarding synonyms in general bilingual dictionaries. *International Journal of Lexicography*, 26(3):346–361.

Udo Hahn, Philipp Daumke, Stefan Schulz, and Kornél G. Markó. 2005. Cross-language mining for acronyms and their completions from the web. In *Discovery Science, 8th International Conference, DS 2005, Singapore, October 8-11, 2005, Proceedings*, pages 113–123.

Men Haiyan. 2015. *Vocabulary increase and collocation learning: a corpus-based cross-sectional study of Chinese EFL learners*. Ph.D. thesis, Birmingham City University.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. *Prague Czech-English Dependency Treebank 2.0*. https://catalog.ldc.upenn.edu/LDC2004T25.

Jan Hajič, Eduard Bejček, Alevtina Bémová, Eva Buráňová, Eva Hajičová, Jiří Havelka, Petr Homola, Jiří Kárník, Václava Kettnerová, Natalia Klyueva, Veronika Kolářová, Lucie Kučová, Markéta Lopatková, Marie Mikulová, Jiří Mírovský, Anna Nedoluzhko, Petr Pajas, Jarmila Panevová, Lucie Poláková, Magdaléna Rysová, Petr Sgall, Johanka Spoustová, Pavel Straňák, Pavlína Synková, Magda Ševčíková, Jan Štěpánek, Zdeňka Urešová, Barbora Vidová Hladká, Daniel Zeman, Šárka Zikánová, and Zdeněk Žabokrtský. 2018. *Prague Dependency Treebank 3.5*. LINDAT/CLARIN digital library. http://hdl.handle.net/11234/1-2621.

Jan Hajič, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, Zdeněk Žabokrtský, Magda Ševčíková Razímová, and Zdeňka Urešová. 2006. *Prague Dependency Treebank 2.0*. Number LDC2006T01. LDC, Philadelphia, PA, USA. https://catalog.ldc.upenn.edu/ldc2006t01.

Yoshihiko Hayashi. 2012. Computing Cross-lingual Synonym Set Similarity by Using Princeton Annotated Gloss Corpus. In *GWC 2012: 6th International Global Wordnet Conference, Proceedings*, pages 134–141. Tribun EU s. r. o.

Nancy Ide, Tomaz Erjavec, and Dan Tufis. 2002. Sense discrimination with parallel corpora. In *Proceedings of the ACL-02 Workshop on Word Sense Disambiguation: Recent Successes and Future Directions*.

Nancy Ide. 1999. Parallel translations as sense discriminators. In *SIGLEX99: Standardizing Lexical Resources*.

Diptesh Kanojia, Aditya Joshi, Pushpak Bhattacharyya, and Mark James Carman. 2016. That'll do fine!: A coarse lexical resource for english-hindi mt, using polylingual topic models. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Aleš Klégr. 2004. Vnitrojazyková a mezijazyková synonymie (Intralingual and Interlingual Synonymy), philologica.net, ISSN 1214-5505, available online. (2004-01-13).

Geoffrey N. Leech, Marianne Hundt, Christian Mair, and Nicholas Smith. 2012. *Change in Contemporary English*. Cambridge University Press, New York.

Dekang Lin, Shaojun Zhao, Lijuan Qin, and Ming Zhou. 2003. Identifying synonyms among distributionally similar words. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 1492–1493, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Markéta Lopatková, Václava Kettnerová, Eduard Bejček, Anna Vernerová, and Zdeněk Žabokrtský. 2016. *Valenční slovník českých sloves VALLEX*. Nakladatelství Karolinum, Praha, Czechia.

Maddalen Lopez de Lacalle, Egoitz Laparra, Itziar Aldabe, and German Rigau. 2016. Predicate matrix: automatically extending the semantic interoperability between predicate resources. *Language Resources and Evaluation*, 50(2):263–289, Jun.

Phillip Adriaan Louw. 2012. Synonymy in the translation equivalent paradigms of a standard translation dictionary. *Lexikos*, 8(1).

John Lyons. 1968. *Introduction to Theoretical Linguistics*. Cambridge University Press.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *COMPUTATIONAL LINGUISTICS*, 19(2):313–330.

Samuel E. Martin. 1960. *Selection and presentation of ready equivalents in a translation dictionary, work paper for conference on lexicography, Indiana university, November 11-12, 1960 [microform] / Samuel E. Martin*. Distributed by ERIC Clearinghouse [Washington, D.C.].

George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM*, 38(11):39–41, November.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Peter Newmark. 1988. *A textbook of translation*. Prentice-Hall International, New York.

Karel Pala and Pavel Smrž. 2004. Building Czech Wordnet. *Romanian Journal of Information Science and Technology*, 7(1-2):79–88.

Karel Pala, Tomáš Čapek, Barbora Zajíčková, Dita Bartůšková, Kateřina Kulková, Petra Hoffmannová, Eduard Bejček, Pavel Straňák, and Jan Hajič. 2011. *Czech WordNet 1.9 PDT*. LINDAT/CLARIN digital library. http://hdl.handle.net/11858/00-097C-0000-0001-4880-3.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106, March.

Frank Robert Palmer. 1976. *Semantics: A New Outline*. Cambridge University Press.

Frank Robert Palmer. 1981. *Semantics. 2nd edn.* Cambridge University Press, UK.

Martha Palmer. 2009. SemLink: Linking PropBank, VerbNet and FrameNet. In *Proceedings of the Generative Lexicon Conference*, page 9–15.

Ellie Pavlick, Travis Wolfe, Pushpendre Rastogi, Chris Callison-Burch, Mark Dredze, and Benjamin Van Durme. 2015. Framenet+: Fast paraphrastic tripling of framenet. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 408–413, Beijing, China, July. Association for Computational Linguistics.

Lonneke Plas and Jörg Tiedemann. 2006. Finding synonyms using automatic word alignment and measures of distributional similarity. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 866–873. Association for Computational Linguistics.

Sameer S. Pradhan, Eduard Hovy, Mitch Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2007. Ontonotes: A unified relational semantic representation. *International Journal of Semantic Computing*, 01(04):405–419.

PSI and Associates. 1988. *Roget's thesaurus of synonyms and antonyms*. P.S.I. & Associates.

Philip Resnik. 1997. A perspective on word sense disambiguation methods and their evaluation. In *Tagging Text with Lexical Semantics: Why, What, and How?*

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633, October.

J. Ruppenhofer, M. Ellsworth, M. R. L. Petruck, Ch. R. Johnson, and J. Scheffczyk. 2006. FrameNet II: Extended theory and practice. *Unpublished Manuscript*.

Karin Kipper Schuler. 2006. *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania.

Petr Sgall, Eva Hajičová, and Jarmila Panevová. 1986. *The meaning of the sentence in its semantic and pragmatic aspects*. D. Reidel, Dordrecht.

Karen Sparck Jones. 1986. *Synonymy and Semantic Classification*. Edinburgh University Press, Edinburgh, Scotland, Scotland.

Zdeňka Urešová, Jan Štěpánek, Jan Hajič, Jarmila Panevová, and Marie Mikulová. 2014. *PDT-Vallex*. LINDAT/CLARIN digital library. http://hdl.handle.net/11858/00-097C-0000-0023-4338-F.

Zdeňka Urešová, Eva Fučíková, Jan Hajič, and Jana Šindlerová. 2015. *CzEngVallex - Czech-English Valency Lexicon*. LINDAT/CLARIN digital library. http://hdl.handle.net/11234/1-1512.

Zdeňka Urešová, Eva Fučíková, and Jana Šindlerová. 2016. CzEngVallex: a bilingual Czech-English valency lexicon. *The Prague Bulletin of Mathematical Linguistics*, 105:17–50.

Zdeňka Urešová, Eva Fučíková, and Eva Hajičová. 2017a. Czengclass – towards a lexicon of verb synonyms with valency linked to semantic roles.

Zdeňka Urešová, Eva Fučíková, Eva Hajičová, and Jan Hajič. 2017b. Syntactic-semantic classes of context-sensitive synonyms based on a bilingual corpus. In Zygmunt Vetulani and Joseph Mariani, editors, *Proceedings of 8th Language and Technology Conference*, pages 201–205, Poznań, Poland. Fundacja Uniwersytetu im. Adama Mickiewicza, Fundacja Uniwersytetu im. Adama Mickiewicza w Poznaniu.

Zdeňka Urešová, Eva Fučíková, Eva Hajičová, and Jan Hajič. 2018a. Creating a Verb Synonym Lexicon Based on a Parallel Corpus. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC'18)*, Miyazaki, Japan, May. European Language Resources Association (ELRA).

Zdeňka Urešová, Eva Fučíková, Eva Hajičová, and Jan Hajič. 2018b. CzEngClass 0.1. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University, `http://hdl.handle.net/11234/1-2823`, Creative Commons - Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0).

Zdeňka Urešová, Eva Fučíková, Eva Hajičová, and Jan Hajič. 2018c. CzEngClass 0.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University, `http://hdl.handle.net/11234/1-2824`, Creative Commons - Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0).

Zdeňka Urešová, Eva Fučíková, Eva Hajičová, and Jan Hajič. 2018d. Tools for Building an Interlinked Synonym Lexicon Network. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC'18)*, Miyazaki, Japan, May. European Language Resources Association (ELRA).

Zdeňka Urešová. 2011. *Valenční slovník Pražského závislostního korpusu (PDT-Vallex)*. Studies in Computational and Theoretical Linguistics. Ústav formální a aplikované lingvistiky, Praha, Czechia. 375 pp.

Piek Vossen. 2004. Eurowordnet: A multilingual database of autonomous and language-specific wordnets connected via an inter-lingualindex. *International Journal of Lexicography*, 17(2):161–173.

Fei Wang and Yunfang Wu. 2012. Mining market trend from blog titles based on lexical semantic similarity. In *Computational Linguistics and Intelligent Text Processing - 13th International Conference, CICLing 2012, New Delhi, India, March 11-17, 2012, Proceedings, Part II*, pages 261–273.

Shumin Wu and Martha Palmer. 2011. Semantic mapping using automatic word alignment and semantic role labeling. In *Proceedings of Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 21–30. Association for Computational Linguistics.

Hua Wu and Ming Zhou. 2003. Optimizing synonym extraction using monolingual and bilingual resources. In *Proceedings of the Second International Workshop on Paraphrasing - Volume 16*, PARAPHRASE '03, pages 72–79, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shumin Wu, Jinho D. Choi, and Martha Palmer. 2010. Detecting Cross-lingual Semantic Similarity Using Parallel PropBanks. In *Proceedings of the 9th Conference of the Association for Machine Translation in the Americas*, AMTA'10, Denver, CO.

Xian-mo Zeng. 2007. Semantic relationships between contextual synonyms. *US-China Education Review*, 4(9):33–37.

Convolutional Neural Network for Universal Sentence Embeddings

Xiaoqi Jiao[1] **Fang Wang**[1,2] **Dan Feng**[1]
[1]*Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System, (School of Computer Science and Technology, Huazhong University of Science and Technology), Ministry of Education of China*
[2]*Shenzhen Huazhong University of Science and Technology Research Institute*
Corresponding author: wangfang@hust.edu.cn

Abstract

This paper proposes a simple CNN model for creating general-purpose sentence embeddings that can transfer easily across domains and can also act as effective initialization for downstream tasks. Recently, averaging the embeddings of words in a sentence has proven to be a surprisingly successful and efficient way of obtaining sentence embeddings. However, these models represent a sentence, only in terms of features of words or uni-grams in it. In contrast, our model (CSE) utilizes both features of words and n-grams to encode sentences, which is actually a generalization of these bag-of-words models. The extensive experiments demonstrate that CSE performs better than average models in transfer learning setting and exceeds the state of the art in supervised learning setting by initializing the parameters with the pre-trained sentence embeddings.

1 Introduction

Representing word sequences such as phrases and sentences plays an important role in natural language understanding systems. In recent years, many composition functions have been applied to word embeddings to obtain vectors for longer phrases or sentences, ranging from simple operations like addition (Mitchell and Lapata, 2008) to richly-structured functions like recursive neural networks (Socher et al., 2011), convolutional neural networks (Kalchbrenner et al., 2014), and recurrent neural networks (Tai et al., 2015).

In this paper, based on convolutional neural networks (CNN), we introduce an architecture that can use both features of words and features of n-grams to encode sentences into real-valued vectors with the property that sentences with similar meaning have high cosine similarity in the embedding space. Our goal is to learn general-purpose sentence representations that can be transferred for measuring semantic textual similarity (STS) (Agirre et al., 2012) and can also act as effective initialization for downstream tasks.

Originally, the CNN model is invented to extract local features in computer vision (Lecun et al., 1998). But it has been shown to be effective for NLP and has achieved excellent results in sentence modeling (Kalchbrenner et al., 2014), and other traditional NLP tasks (Collobert et al., 2011). To encode the meaning of a sentence of varying length into a fix-length vector, our simple CNN model first utilizes multiple filters with different size to extract possible features of n-grams in it. Then we obtain one feature corresponding to one filter through pooling operation. The intuition is to capture one aspect of semantic features of n-grams for each filter. Inspired by the strong performance of simply averaging word embeddings (Wieting et al., 2016), we also set filter size to 1 to extract word semantic information.

To evaluate our model, we consider two types of learning settings: transfer learning evaluation and supervised learning evaluation. For the transfer leaning setting, we first train our model on noisy phrase pairs from the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013), then evaluate it on every SemEval STS task from 2012 through 2015 and the 2014 SemEval SICK dataset (Marelli et al., 2014). Particularly, we follow the experiment setting of Wieting et al. (2016), in which they compared various types of neural network architectures except CNN, spanning the range of complexity from word averaging to LSTMs

Proceedings of the 27th International Conference on Computational Linguistics, pages 2470–2481
Santa Fe, New Mexico, USA, August 20-26, 2018.

and showed the impressive performance of Paragram-Phrase (PP), a simple word average model. So our work can also serve as complementary to theirs. As results, we found our model performs better than their PP model in most evaluation datasets. Moreover, researchers have recently found that since models are ultimately tested on sentences, sentence representations trained on noisy sentence pairs, obtained automatically by aligning Simple English to standard English Wikipedia (Coster and Kauchak, 2011), perform much better than those trained on noisy phrase pairs from PPDB dataset (Wieting and Gimpel, 2017). So we conduct the same transfer learning experiment on noisy sentence pairs with our model which is initialized with the parameters obtained from PPDB, and again find our model outperforms two state-of-the-art models: ATT-CCG (Shaonan Wang, 2017) and GRAN (Wieting and Gimpel, 2017). For the supervised learning setting, we use the model from transfer setting as a prior and fine-tune this model on the SICK dataset in a supervised style. With useful initialization, we achieve new state-of-the-art results on this task.

2 Related Work

Obtaining the most useful distributed representations of phrases or sentences could ultimately have a significant impact on language understanding systems since it is phrases and sentences, rather than individual words, that encode the human-like general world knowledge (or common sense) (Norman, 1972). According to their purposes, sentence embeddings generally fall into two categories: task-specific sentence embeddings and general-purpose sentence embeddings.

The first consists of sentence embeddings trained specifically for a certain task. They are usually combined with downstream applications and trained by supervised learning. Researchers have proposed many models along this line, and they typically use recursive neural networks (Socher et al., 2012; Socher et al., 2013), convolutional neural networks (Kalchbrenner et al., 2014; dos Santos and Gatti, 2014; Kim, 2014) or recurrent neural networks with long short-term memory (LSTM) (Hochreiter et al., 1997; Chung et al., 2014) as an intermediate step in creating sentence embeddings to solve a variety of NLP tasks including paraphrase identification and sentiment classification (Yin and Schütze, 2015; Tan et al., 2016; Lin et al., 2017).

The other category consists of universal sentence embeddings, which are usually trained by unsupervised or semi-supervised learning and can be served as features for many other NLP tasks such as text classification and semantic textual similarity. This include recursive auto-encoders (Socher et al., 2011), ParagraphVector (Le and Mikolov, 2014), SkipThought vectors (Kiros et al., 2015), Sequential Denoising Autoencoders (SDAE), FastSent (Hill et al., 2016), PP (Wieting et al., 2016), Sent2Vec (Pagliardini et al., 2017), GRAN (Wieting and Gimpel, 2017), etc.

In this paper, we also aim to learn general-purpose, paraphrastic sentence embeddings, the same purpose as Wieting et al. (2016). In their work, they demonstrate the strong performance of the PP model across a broad range of tasks and domains, but also show some limitations of this bag-of-words model. For future work, they leave a challenge: can we find a model that takes context into account while still generalizing as well as the PP model? Then in their following work (Wieting and Gimpel, 2017), they proposed a novel architecture, called GRAN that utilizes the context information to create gate for each word in a sentence. After obtaining gates which can be seen as a kind of attention mechanisms, they average the word embeddings, multiplied with their respective gates, to get the embedding of the sentence. Besides, there is another model, called ATT-CCG (Shaonan Wang, 2017), which also incorporates attention mechanism into PP model by using word attributes information (CCG supertag of words). Unlike GRAN and ATT-CCG, we propose a convolutional architecture, called Convolutional Sentence Encoder (CSE), which directly consider the semantics of n-grams by using convolving filters with different size. Further description of this model is included in Section 3.1.

3 Model and Training

3.1 Model

Figure 1 shows the architecture of the proposed model CSE. Our model encodes the meaning of a sentence of varying length into a fixed-length real-valued vector such that the semantic similarity between

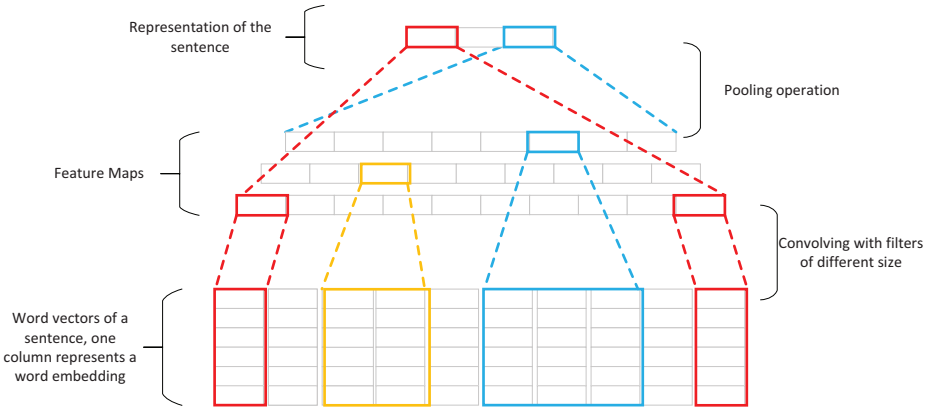


Figure 1: Model architecture

sentences can be measured by the cosine similarity of their corresponding vectors. We denote $\mathbf{x}_i \in \mathbb{R}^d$ as the d-dimensional word vector of the i-th word in a sentence. Then a sentence of length n (padded when necessary) is represented as

$$\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \cdots \oplus \mathbf{x}_n, \tag{1}$$

Where $\oplus$ is the concatenation operator. In general, let $\mathbf{x}_{i:j}$ refers to the concatenation of words $\mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_j$. A convolution operation involves a filter $\mathbf{w} \in \mathbb{R}^{h*d}$, which is applied to a window of h words to produce a new feature. For example, a feature c_i is generated from a window of words $\mathbf{x}_{i:i+h-1}$ by

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b). \tag{2}$$

Here $b \in \mathbb{R}$ is a bias term and f is a activation function (either tanh or linear unit; linear unit is slightly better in our experiments, so we only report the linear unit results). This filter is applied to each possible window of words in the sentence $\{\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{n-h+1:n}\}$ to produce a feature map

$$\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \tag{3}$$

with $\mathbf{c} \in \mathbb{R}^{n-h+1}$. Instead of using mean pooling, we then apply a sum pooling operation over the feature map for forcing our model to capture semantic similarity between words and phrases.[1] And we take the sum value $\hat{c} = sum(\mathbf{c})$ as the feature corresponding to this particular filter. The idea is to capture one aspect of n-gram semantic features for each feature map. This pooling scheme also naturally deals with variable sentence lengths.

We have described the process by which one feature is extracted from one filter. The model uses k filters (with varying window sizes) to obtain multiple features. These features form the representation of a sentence:

$$g(\mathbf{x}_{1:n}) = [\hat{c_1}, \hat{c_2}, \ldots, \hat{c_k}]. \tag{4}$$

For the learnable parameters in our model, we denote the word embeddings by W_w and all the other compositional parameters by W_c. In addition, We initialize W_w by some pretrained embeddings in all experiments.

3.2 Training

We follow the training procedure of Wieting et al. (2016), described below. The training data consists of a set S of phrase or sentence pairs $\langle s_1, s_2 \rangle$ from either PPDB (Ganitkevitch et al., 2013) or the aligned

[1] Suppose we have two sentences that are very different in length but have very close meanings. In order to get similar embeddings for them, our model must find a way to capture semantic similarity between words in the short sentence and n-grams in the long one.

Wikipedia sentences (Coster and Kauchak, 2011) where s_1 and s_2 are assumed to be paraphrases. We optimize a margin-based loss:

$$\min_{W_w, W_c} \frac{1}{|S|} \Bigg(\sum_{\langle s_1, s_2 \rangle \in S} \max(0, \delta - \cos(g(s_1), g(s_2)) + \cos(g(s_1), g(t_1))) + \max(0, \delta - \cos(g(s_1), g(s_2)) + \cos(g(s_2), g(t_2))) \Bigg) + \lambda_w \|W_{w_{initial}} - W_w\|^2 + \lambda_c \|W_c\|^2 \tag{5}$$

Where g is our sentence representation model, δ is the margin, λ_w and λ_c are the regularization parameters, $W_{w_{initial}}$ is the initial word embedding matrix, and t_1 and t_2 are carefully-selected negative examples taken from a nimi-batch during optimization. The intuition is that we want the two sentences to be more similar to each other $\cos(g(s_1), g(s_2))$ than either is to their respective negative examples t_1 and t_2, by a margin of at least δ.

Selecting Negative Examples: To select t_1 and t_2 in Eq. 5, we simply choose the most similar example in some set of phrases or sentences (other than those in the given pair). For simplicity, we use the mini-batch for this set, but it could be a different set. That is, we choose t_1 for a given $\langle s_1, s_2 \rangle$ as follows:

$$t_1 = \underset{t:\langle t, \cdot \rangle \in S_b \setminus \{\langle s_1, s_2 \rangle\}}{\arg\max} \cos(g(s_1), g(t))$$

where $S_b \in S$ is the current mini-batch. That is, we want to choose a negative example t_i that is similar to s_i according to the current model.

4 Experiments

In this part, we present the results of our experiments. We first use the semantic textual similarity (STS) tasks from 2012 to 2016 (Agirre et al., 2012; Agirre et al., 2013; Agirre et al., 2014; Agirre et al., 2015; Agirre et al., 2016) and the SemEval 2014 SICK (Marelli et al., 2014) test set to evaluate the performance of our model under the transfer learning setting. Then for supervised setting, we choose the SICK 2014 dataset with its standard train/dev/test split.

Given two sentences, the aim of the STS tasks is to predict their similarity on a 0-5 scale, where 0 indicates the sentences are on different topics and 5 indicates that they are completely equivalent. These STS datasets cover a broad range of domains, including news, image and video descriptions, glosses, web forum, twitter and so on. As for the supervised SICK task, it requires us to give a relatedness score for sentence pair on a 1-5 continuous scale similar as STS, but is an easier learning problem since the training and test examples are all drawn from the same distribution.

4.1 Transfer Learning

DataSets and Experimental Settings: As training data, we use noisy phrase pairs from Paraphrase Database (PPDB) (Ganitkevitch et al., 2013) which is automatically derived from naturally-occurring bilingual text. PPDB comes in different sizes (S, M, L, XL, XXL, and XXXL), where each larger size subsumes all smaller ones but contains noisier paraphrases. Considering precision and data size, we choose the XL section of PPDB which contains 3,033,753 unique phrase pairs. We use PARAGRAM-SL999 embeddings (Wieting et al., 2015) to initialize the word embedding matrix (W_w) for the model, and fix the learning procedure, using Adam (Kingma and Ba, 2015) optimizer with a learning rate of 0.001. Moreover, we use filter windows (h) of 1, 2, 3 with 100 feature maps each[2], since we want the dimension of sentence vectors of CSE to be same as the baseline models. The reason we use filter window of 1 is inspired by the strong performance of the bag-of-words PP model. For hyperparameter tuning, we search $\delta \in \{0.4, 0.6, 0.8\}$, $\lambda_w \in \{10^{-4}, 10^{-5}, 10^{-6}\}$, $\lambda_c \in \{10^{-5}, 10^{-6}, 10^{-7}, 0\}$, and batch

[2]We put filter window size selecting part in Section 5.2 for further analyzing the difference between CSE and PP.

size over {32, 64, 128, 256}. We train the model on PPDB for 10 epochs, using STS 2016 datasets for validation, then evaluate on STS 2012-2015 tasks and the SICK test set.

Wieting and Gimpel (2017) found the sentences in the STS test sets are quite different from the short training fragments in PPDB, which may affect performance of models that are more sensitive to overall characteristics of the word sequences. Therefore, after training on PPDB, we train the model for 10 more epochs on another source of data, called SimpWiki, which is a set of sentence pairs, automatically extracted from Simple English Wikipedia and English Wikipedia articles by Coster and Kauchak (2011). This dataset, consists of 167,689 sentence pairs, has been proved useful for semantic textual similarity task by Wieting and Gimpel (2017), although it was extracted for developing text simplification systems. When continuing to train on this dataset, we initialize the model with the parameters learned from PPDB, use the hyperparameters that maximize Pearson's r on the 2016 STS tasks and try dropout (Srivastava et al., 2014) on the word embeddings. Then, we again test CSE on all STS tasks and the SICK test set.

Dataset	**PP**	**CSE**
MSRpar	42.6	**51.6**
MSRvid	74.5	**79.8**
SMT-eur	47.3	**50.7**
OnWN	**70.6**	69.9
SMT-news	58.4	**64.4**
STS 2012 Average	58.7	**63.3**
headline	72.4	**74.0**
OnWN	67.7	**72.1**
FNWN	**43.9**	29.8
STS 2013 Average	**61.3**	58.6
deft forum	48.7	**56.1**
deft news	**73.1**	70.0
headline	69.7	**70.9**
images	78.5	**81.6**
OnWN	78.8	**79.2**
tweet news	**76.4**	74.6
STS 2014 Average	70.9	**72.1**
answers-forums	**68.3**	66.2
answers-students	78.2	**78.4**
belief	**76.2**	68.5
headline	74.8	**76.5**
images	81.4	**82.9**
STS 2015 Average	**75.8**	74.5
2014 SICK	71.6	**71.9**
Total Average	67.7	**68.5**

Table 1: Results on SemEval textual similarity datasets (Pearson's $r \times 100$) when models trained on PPDB XL only. The highest score in each row is in boldface.

Results: In Table 1, we show the results of our model when trained on PPDB XL only. Particularly, this set of results is the median of five runs with random initialisations of the learnable parameters. As baseline, we compare to PP model (Wieting et al., 2016) since it is simple and proved to performs strongly in the STS and SICK tasks, and more importantly, it uses the same training dataset as CSE, which ensures a fair comparison. As we can see, CSE outperforms the baseline in the majority of datasets (14 out of 20) and in total average. We attribute the strong performance of CSE to the functional architecture which represents a sentence in terms of features that depend on words and n-grams. In contrast, PP model only extracts the features of words. But on very few datasets such as 2013 FNWN, our model performs poorly. Upon examination, two characteristics of this dataset may explain the poor performance: a).it

contains many sentence pairs of low similarity with low word overlap. But it is more likely for CSE to overestimate the similarity of sentences for capturing extra grammatical features of n-grams. b).most sentence pairs are very different in length. Since addition pooling differentiates more the length of sentences than mean pooling(or average), it is more difficult for CSE to determine the similarity of this kind of sentence pairs.

Dataset	**GRAN**	**ATT-CCG**	**CSE**
MSRpar	47.7	**49.9**	48.6
MSRvid	85.2	84.2	**86.0**
SMT-eur	49.3	49.3	**49.8**
OnWN	71.5	**72.7**	72.6
SMT-news	58.7	**66.6**	57.8
STS 2012 Average	62.5	**64.5**	63.0
headline	76.1	73.6	**76.3**
OnWN	**81.4**	79.3	78.5
FNWN	**55.6**	50.7	51.5
STS 2013 Average	**71.0**	67.9	68.8
deft forum	55.7	55.2	**58.1**
deft news	**77.1**	75.5	**77.1**
headline	72.8	72.2	**74.6**
images	85.8	83.1	**86.2**
OnWN	**85.1**	84.1	84.2
tweet news	**78.7**	77.7	77.7
STS 2014 Average	75.9	74.6	**76.3**
answers-forums	73.1	69.2	**73.5**
answers-students	72.9	**78.3**	76.7
belief	78.0	**78.4**	76.8
headline	78.6	77.4	**79.8**
images	85.8	85.3	**86.0**
STS 2015 Average	77.7	77.7	**78.6**
answer	-	**64.5**	59.3
headlines	-	70.1	**75.5**
plagiarism	-	82.5	**82.6**
postediting	-	**83.0**	81.4
question	-	58.5	**73.2**
STS 2016 Average	-	71.8	**74.4**
2014 SICK	72.9	-	**73.0**

Table 2: Results on SemEval textual similarity datasets (Pearson's $r \times 100$) after CSE trained on SimpWiki. The highest score in each row is in boldface.

After CSE continues to be trained on SimpWiki, we compare it with two state-of-the-art models: ATT-CCG and GRAN. From Table 2, in the 19 datesets, tested by all three models, CSE outperforms the two other models on 10 datasets. In addition, on SICK 2014 test set, our model performs slightly better than GRAN, and on STS 2016 tasks, our average Pearson's r is 2.6 points higher than ATT-CCG. Our hypothesis explaining the strong performance is that when training our model on SimpWiki, we initialize it with the parameters learned from PPDB, which makes learning process easier.

4.2 Supervised Learning

To investigate whether CSE can also get strong performance in supervised setting, we evaluate it on the SICK 2014 dataset which contains 4,500 sentence pairs in the training set, 500 in the development

set, and 4,927 in the test set. We minimize the objective function[3] from Tai et al. (2015). Given a sentence pair $\langle s_1, s_2 \rangle$ with representations $\langle h_1, h_2 \rangle$ and a relatedness score y in the range $[1, K]$, they first compute:

$$\begin{aligned} h_* &= h_1 \oplus h_2, \\ h_+ &= |h_1 - h_2|, \\ h_s &= \sigma\left(W^{(*)} h_* + W^{(+)} h_+ + b^{(h)}\right), \\ \hat{p_\theta} &= softmax\left(W^{(p)} h_s + b^{(p)}\right), \\ \hat{y} &= r^T \hat{p_\theta}, \end{aligned}$$

Where $r^T = [1\ 2 \ldots K]$, θ represents model parameters. They then define a sparse target distribution p that satisfies $y = r^T p$:

$$p_i = \begin{cases} y - \lfloor y \rfloor, & \text{if } i = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor - y + 1, & \text{if } i = \lfloor y \rfloor \\ 0, & \text{otherwise} \end{cases}$$

for $1 \leq i \leq K$. Finally, they use the regularized KL-divergence between p and $\hat{p_\theta}$ as cost function:

$$J(\theta) = \frac{1}{m} \sum_{k=1}^{m} KL\left(p^{(k)} \| \hat{p_\theta}^{(k)}\right) + \frac{\lambda}{2} \|\theta\|^2, \tag{6}$$

where m is the number of training pairs and the superscript k indicates the k-th sentence pair. For training, we initialize the supervised model with the parameters (including W_w and W_c) from Table 2, and regularize back to their initial values. Besides, we tune λ_w, λ_c and batch size same as in Section 4.1 and the extra hyperparameter $\lambda \in \{10^{-5}, 10^{-6}, 10^{-7}, 0\}$ in Eq. 6, referring to this setup as "universal". Again, we train the model for 10 epochs, using Adam optimizer with a learning rate of 0.001 and use Pearson's r index for validation. As a comparison, We also experiment with CSE with random initialization except initializing word embeddings W_w with PARAGRAM-SL999.

Model	r	ρ	*MSE*
Illinois-LH (Lai and Hockenmaier, 2014)	0.7993	0.7538	0.3692
UNAL-NLP (Jimenez et al., 2014)	0.8070	0.7489	0.3550
Meaning Factory (Bjerva et al., 2014)	0.8268	0.7721	0.3224
ECNU (Zhao et al., 2014)	0.8414	-	-
Constituency Tree-LSTM (Tai et al., 2015)	0.8491	0.7873	0.2852
Dependency Tree-LSTM (Tai et al., 2015)	0.8676	0.8083	0.2532
CNN (He et al., 2015)	0.8686	0.8047	0.2606
PP$_{universal}$ (Wieting et al., 2016)	0.8684	-	-
GRAN$_{universal}$ (Wieting and Gimpel, 2017)	0.8600	-	-
CSE$_{random}$(This work)	0.8558	0.8059	0.2730
CSE$_{universal}$(This work)	**0.8696**	**0.8087**	**0.2480**

Table 3: Test results on the SICK dataset. Evaluation metrics are Pearson's r, Spearman's ρ, and mean squared error (*MSE*). The results of the first group are from SemEval 2014 systems. The best performance in each metric is in boldface.

The results are shown in Table 3. First, with random initialization, our model is not good enough, slightly better than models from SemEval 2014 systems and Constituency Tree-LSTM. But, initializing with universal parameters and regularizing back to them significantly improves the performance of our

[3]This objective function has been shown to perform very strongly on text similarity tasks, significantly better than squared or absolute error.

model, exceeding the state of the art on all of the three evaluation metrics. The results demonstrate that our universal sentence embeddings carry rich semantic information and therefore can be used as effective features for downstream tasks.

5 Analysis

5.1 Training Data Analysis

The quality of sentence embedding depends heavily on training data source, particularly when experimenting in transfer learning setting, since test examples are not drawn from the same distribution. Wieting and Gimpel (2017) found changing training data from PPDB to SimpWiki is an effective method for improving the performance of sentence model. But, in this paper, we also want to know if our model can get extra boost by cumulative learning, that is, learning on phrase pairs first, then on sentence pairs.

Dateset	CSE$_{phrase}$	CSE$_{sentence}$	CSE$_{cumulative}$
STS 2012 Average	**63.3**	62.0	63.0
STS 2013 Average	58.6	67.3	**68.8**
STS 2014 Average	72.1	75.0	**76.3**
STS 2015 Average	74.5	77.4	**78.6**

Table 4: Results on SemEval textual similarity datasets (Pearson's $r \times 100$). The highest score in each row is in boldface

To answer this question, we experiment with CSE, using phrase pairs dataset PPDB and sentence pairs dataset SimpWiki as training set. In Table 4, we show the results when CSE is trained on phrase or sentence pairs individually and a cumulative style of them two. First, we can see that it is better to use sentence pairs than phrase pairs for training since test sets are all sentences, which suggests CSE is sensitive to the overall characteristics of the word sequences, and the difference between train and test matters much. Otherwise, we find that CSE gains a large margin of performance improvement through the cumulative learning from phrase pairs to sentence pairs, justifying our claim that incremental learning is an important idea to gain extra improvements.

5.2 Filter Size Selecting

Dateset	1-gram	2-gram	3-gram	4-gram	combined
STS 2012 Average	*61.82*	61.53	61.44	60.73	**62.87**
STS 2013 Average	*63.06*	62.14	62.56	62.37	**64.26**
STS 2014 Average	***75.32***	73.61	73.52	73.39	74.26
STS 2015 Average	*76.45*	76.14	76.06	76.11	**76.99**
Total Average	*70.13*	69.29	69.28	69.04	**70.40**

Table 5: Results on SemEval textual similarity datasets (Pearson's $r \times 100$) where n-gram columns show the performance of CSE with filter windows of only size n. The highest score in each row is in boldface, and the highest score for the group of n-gram columns in each dateset is in italic

In this section, we explain how we choose the filter size of our model and analyze how big is the "n" on n-grams that it could be capturing. For comparison, we fix all the other hyperparameters except filter windows size.[4] Then we train the model with default Adam optimizer for 10 epochs, using SimpWiki as training data, STS 2016 for validation and STS 2012 to 2015 as test dataset. Results are shown in Table 5. From the first colunm group, we can see CSE with filter windows of size 1 performs much better than it with bigger window size, which indicates the meaning of words in a sentence plays a very important

[4]For qualitative explanation, we do not fine-tune the hyperparameters and simply fixed δ to 0.6, λ_w to 10^{-6}, λ_c to 0, and batch size to 100. Besides, we initialize the word embedding matrix (W_w) with PARAGRAM-SL999 embeddings and always keep the number of filters equal to 300.

role in composing the semantics of the sentence, which also proves why bag-of-words models like PP can have strong performance in STS tasks. In addition, with the window size increases, the performance of our model is getting poorer, and there is a significant drop in performance when n-gram comes to 4. Inspired by Kim (2014), we then try to combine filters with different window size for additional performance improvements. Simply choosing the three smallest filter windows size of 1, 2, 3 with 100 feature maps each leads us to better results, as shown in the *combined* column.

5.3 Error Analysis

#	Sentence A	Sentence B	CSE	PP	Gold
1	There are a lot of push up variations you can do and they all stress different muscle systems.	There are several different pushup variations out there and most of them provide a unique advantage.	**3.32**	3.14	4.2
2	the methodology takes much less time rather than naive methods.	this is a much quicker method than other more naive methods.	**4.88**	4.79	5.0
3	a boy plays with a noodle by the pool.	a boy plays with a foam noodle toy by a pool.	4.51	**4.61**	4.6
4	two hockey players fighting on the ice.	two hockey players in a struggle on the ice.	**4.78**	4.74	4.8
5	A group of sheep in a field.	A group of horses grazing in a field.	**3.86**	3.89	1.6
6	The lamb is looking at the camera.	A cat looking at the camera.	**3.89**	4.13	0.8
7	A person is riding their bike on a trail next to the woods.	A small child in a yellow shirt is holding their arms out to the sides.	2.63	**2.61**	0.4
8	A boy in a red sled is riding down the hill.	A multicolour dog in a red collar crouching on the grass.	3.29	**3.27**	0.0

Table 6: Illustrative sentence pairs from the STS datasets showing errors made by CSE and PP. The last three columns show the similarity score of CSE, the similarity score of PP, and the gold similarity score. Boldface indicates smaller error compared to gold standard scores.

In this section, we analyze the predictions of our model CSE and the average model PP on the STS datasets. After scaling the predicted cosine similarities into the range of [0, 5], We compare them to the gold standard scores. Examples are illustrated in Table 6. In the first group (examples 1 and 2), we found that when two sentences have a small amount of word overlap but similar meaning, CSE tends to make smaller error for taking the n-gram semantics into account. Then, just as we expected, when two sentences have a lot of word overlap, and have little differences in key semantic roles, PP performs better, as shown in example 3. But in example 4, we got the opposite result, our hypothesis explaining this result is that CNN architectures happen to be good at capturing semantic similarity between words and phrases (*fighting*, *in a struggle*). For the third group (examples 5 and 6), we inspect sentence pairs which have high word overlap rate but different meanings. Under this circumstance, the scores, predicted by CSE, are closer to the golds which suggests PP model is more easily fooled by the high amount of word overlap in such pairs and our model, CSE, is better able to recognize the semantic differences. But in last group (examples 7 and 8) where sentence pairs differ in meaning and have few words in common, CSE performs slightly worse than PP. We think this may due to its capability of capturing extra grammatical features, which leads it more likely to overestimate the similarity of sentences than bag-of-words models (or PP). In addition, from last two groups, both models tend to overestimate sentence pairs of low similarity which may be due to connections between words, whether grammatical or semantic.

6 Conclusion

We introduced a simple CNN architecture to create universal, paraphrastic sentence embeddings. Our model improves upon the state-of-the-art sentence representation models in transfer learning setting and exceeds the state of the art through initialization in the supervised setting. Furthermore, we analyzed the

advantages and disadvantages of our model, and found that it is better at capturing semantic similarity of two sentences than averaging models, especially when they have little word overlap but similar meanings. However, it tends to overestimate the low semantic similarity of a sentence pair. In addition, we released our trained model and codes to facilitate downstream tasks[5]. Future work could extend this model to related tasks including sentiment analysis, text classification and information retrieval.

Acknowledgements

We would like to thank the Writing Mentoring Program of COLING 2018, the anonymous mentor who carefully read our paper and gave very detailed feedback, and the reviewers for their insightful comments. This work was supported in part by National Key R&D Program of China NO.2018YFB10033005, NSFC No.61772216, National Defense Preliminary Research Project (31511010202), Hubei Province Technical Innovation Special Project (2017AAA129), Wuhan Application Basic Research Project (2017010201010103), Project of Shenzhen Technology Scheme JCYJ20170307172248636, Fundamental Research Funds for the Central Universities. This work was also supported by CERNET Innovation Project NGII20170120.

References

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 385–393, Stroudsburg, PA, USA. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic Textual Similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability . In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 Task 1: Semantic Textual Similarity, Monolingual and Cross-Lingual Evaluation . In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511. Association for Computational Linguistics.

Johannes Bjerva, Johan Bos, Rob van der Goot, and Malvina Nissim. 2014. The Meaning Factory: Formal Semantics for Recognizing Textual Entailment and Determining Semantic Similarity . In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 642–646. Association for Computational Linguistics.

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

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *J. Mach. Learn. Res.*, 12:2493–2537, nov.

William Coster and David Kauchak. 2011. Simple English Wikipedia: A New Text Simplification Task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 665–669. Association for Computational Linguistics.

[5]Trained model and code for training and evaluation are available at `https://github.com/XiaoqiJiao/COLING2018`

Cicero dos Santos and Maira Gatti. 2014. Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78. Dublin City University and Association for Computational Linguistics.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764. Association for Computational Linguistics.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks . In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586. Association for Computational Linguistics.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning Distributed Representations of Sentences from Unlabelled Data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377. Association for Computational Linguistics.

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

Sergio Jimenez, George Dueñas, Julia Baquero, and Alexander Gelbukh. 2014. UNAL-NLP: Combining Soft Cardinality Features for Semantic Textual Similarity, Relatedness and Entailment . In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 732–742. Association for Computational Linguistics.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Alice Lai and Julia Hockenmaier. 2014. Illinois-LH: A Denotational and Distributional Approach to Semantics. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 329–334, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Yann Lecun, Leon Bottou, Y Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. 86:2278 – 2324, 12.

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

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. SemEval-2014 Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment . In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 1–8. Association for Computational Linguistics.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based Models of Semantic Composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio, June. Association for Computational Linguistics.

Donald A Norman. 1972. Memory, knowledge, and the answering of questions.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2017. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. *arXiv preprint arXiv:1703.02507*.

Chengqing Zong Shaonan Wang, Jiajun Zhang. 2017. Learning Sentence Representation with Guidance of Human Attention. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, (IJCAI-17)*, pages 4137–4143.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, pages 801–809, USA. Curran Associates Inc.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality Through Recursive Matrix-vector Spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1201–1211, Stroudsburg, PA, USA. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566. Association for Computational Linguistics.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved Representation Learning for Question Answer Matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 464–473. Association for Computational Linguistics.

John Wieting and Kevin Gimpel. 2017. Revisiting Recurrent Networks for Paraphrastic Sentence Embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2078–2088. Association for Computational Linguistics.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From Paraphrase Database to Compositional Paraphrase Model and Back. *Transactions of the Association of Computational Linguistics*, 3:345–358.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards Universal Paraphrastic Sentence Embeddings. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911.

Jiang Zhao, Tiantian Zhu, and Man Lan. 2014. ECNU: One Stone Two Birds: Ensemble of Heterogenous Measures for Semantic Relatedness and Textual Entailment . In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 271–277. Association for Computational Linguistics.

Rich Character-Level Information for Korean Morphological Analysis and Part-of-Speech Tagging

Andrew Matteson
Korea University
amatteson@korea.ac.kr

Chanhee Lee
Korea University
chanhee0222@korea.ac.kr

Heuiseok Lim*
Korea University
limhseok@korea.ac.kr

Young-Bum Kim
Amazon Alexa
youngbum@amazon.com

Abstract

Due to the fact that Korean is a highly agglutinative, character-rich language, previous work on Korean morphological analysis typically employs the use of sub-character features known as graphemes or otherwise utilizes comprehensive prior linguistic knowledge (i.e., a dictionary of known morphological transformation forms, or actions). These models have been created with the assumption that character-level, dictionary-less morphological analysis was intractable due to the number of actions required. We present, in this study, a multi-stage action-based model that can perform morphological transformation and part-of-speech tagging using arbitrary units of input and apply it to the case of character-level Korean morphological analysis. Among models that do not employ prior linguistic knowledge, we achieve state-of-the-art word and sentence-level tagging accuracy with the Sejong Korean corpus using our proposed data-driven Bi-LSTM model.

Title and Abstract in Korean

한국어 형태소 분석 및 품사 부착에 효과적인 문자 단위 정보

한국어는 대표적인 교착어 중 하나이며, 문자 단위 정보의 중요도가 높다. 이에 따라 기존의 한국어 형태소 분석 연구는 자소 정보와 같은 문자 단위 자질을 사용하거나, 형태소 사전 및 변형 규칙과 같은 다량의 언어학적 지식을 활용하였다. 이러한 접근 방법은 사전을 사용하지 않은 문자 단위 형태소 분석은 변형 규칙의 복잡성으로 인해 불가능에 가깝다는 것을 전제로 하고 있다. 이러한 한계를 극복하고자, 본 연구에서는 어떠한 언어 단위도 입력으로 사용할 수 있으며 다단계 변형을 기반으로 형태소 분석 및 품사 부착을 수행하는 방법을 제안한다. 제안된 방법을 적용하여 구현된 데이터 기반 양방향 LSTM 모델의 성능을 세종 말뭉치를 이용하여 정량적으로 평가한 결과, 언어학적 지식을 활용하지 않은 접근 방법들 중 가장 높은 단어 및 문장 단위 부착 정확도를 보임을 확인하였다.

1 Introduction

Korean has traditionally posed a challenge for word segmentation and morphological analysis. In addition to virtually unbounded vocabulary sizes, out-of-vocabulary (OOV) rates for models can be high. Korean is an agglutinative, phonetic language with a SOV (Subject-Object-Verb) syntax and a flexible word order, although certain word orders are considered to be "canonical". Honorifics, conditionals, imperatives, and other forms are all signified using agglutinative endings which sometimes involve transformation of the stem to which they attach. Some endings can be further combined or fused to other endings in a defined order, and furthermore, morphological transformation rules also apply during this process. Transformation rules are mostly consistent at the grapheme level and can be represented by a handful of spelling rules, but many irregular forms do exist.

* Corresponding author

Proceedings of the 27th International Conference on Computational Linguistics, pages 2482–2492
Santa Fe, New Mexico, USA, August 20-26, 2018.

#	Morphemes
1	나(na)/VV + 는(neun)/ETM
2	날(nal)/VV + 는(neun)/ETM
3	나(na)/NP + 는(neun)/JX
4	나(na)/NNP + 는(neun)/JX
5	나(na)/VX + 는(neun)/ETM

Table 1: Ambiguous parses of Eojeol "na-neun"

#	Eojeol	Morphemes
(3)	나는 na-neun	나(na)/NP + 는(neun)/JX
	하늘에 ha-neul-e	하늘(ha-neul)/NNG + 에(e)/JKB
(2)	나는 na-neun	날(nal)/VV + 는(neun)/ETM
	새를 sae-leul	새(sae)/NNG + 를(leul)/JKO
	보았다 bo-ass-da.	보(bo)/VV + 았(ass)/EP + 다(da)/EF + ./SF

Table 2: Correct transformation and tag sequence for sample sentence containing ambiguous Eojeol "na-neun". # corresponds to correct parse sequence in Table 1, only labeled here for "na-neun".

In Unicode, Hangul (Korean alphabet) characters are allocated 11,140 codepoints. Each character contains an initial consonant, vowel, and final consonant represented in C+V+C, C+V, or V+C form. In Unicode, the form is always assumed to be C+V+C and the initial or final consonants are set to null according to the desired target form. Consonants and vowels are considered to be sub-character units called graphemes. Each character is represented using a combination of 19 initial consonants (including null), 21 vowels, and 27 final consonants (including null), and there is a mathematical formula that can be used to combine graphemes to generate the codepoint of a Hangul character. The character "김" (gim) can be represented in C+V+C form as follows.

Initial Consonant	Vowel	Final Consonant
ㄱ (g)	ㅣ (i)	ㅁ (m)

The Korean language has "fusion" spelling rules that apply across character boundaries (within an agglutinative unit), which implies that morphological transformation may occur among adjacent graphemes. When the final consonant of one character meets the initial consonant of the next character during verb inflection, there may be a change in the resulting combined character. This presents character-level embeddings with a unique challenge that is not present in most other languages.

In order to avoid confusion of terminology, we must define the precise meaning of morphological analysis in the context of Korean. For most languages, morphological analysis refers to a word-level tag that describes the aspect, tense, plurality, and other features of the word, whereas part-of-speech (POS) tagging serves to classify the word as a noun, verb, etc. The POS tag is sometimes concatenated to the morphological tag string as in the POSMORPH annotation employed by Heigold, et al (2016a).

In Korean, morphological analysis refers to the segmentation and restoration of morphemes within a "word" unit called an Eojeol and the POS tagging of each constituent morpheme. An Eojeol encodes not only lexical information but also grammatical information due to the agglutinative nature of the Korean language. The recovered morpheme segments often include a stem and other morphemes which indicate tense or other linguistic features. Traditional Korean morphological analysis algorithms operate at the Eojeol level and yield all ambiguous parses (Table 1) that lead to that particular Eojeol, including the morpheme transformations and tags. However, the model[1] proposed in this paper receives input at the sentence level and attempts to

[1]Model source code is made available at https://github.com/xtknight/rich-morphological-tagger

produce the one correct sequence of transformations and tags for all Eojeol within the sentence according to the context (Table 2).

2 Related Work

Morphological analysis of the Korean language has traditionally been performed in several ways, including separation of Korean characters into graphemes by using linguistic knowledge, lattice tree lookup (Park et al., 2010), application of regular and irregular inflection rules (Kang and Kim, 1992), morphosyntactic rule sets, and by using a pre-computed dictionary (Shim and Yang, 2004). However, we investigate whether morphological analysis of Korean is feasible without the use of any of these techniques and without a dictionary by making the assumption that common transformations and their underlying grapheme modifications can be easily recognized and learned with a Bi-LSTM model.

Bi-LSTM-CRFs have been used for sequential tagging with BIO annotation (Sang and Veenstra, 1999). Huang, et al (2015) show their effectiveness for POS tagging, chunking, named entity recognition (NER). These models show state-of-the-art accuracy at several tasks.

Similar models have also been proposed in universal morphological analysis. Heigold, et al (2016b) show how a nested LSTM architecture can be applied to word-level morphological tagging for a wide variety of languages. At the lower level, an LSTM network is used for character-level embedding to reduce OOV errors. However, this work does not investigate how such a model would operate for the most widely used Korean Sejong Corpus.

Sub-character tagging has also been attempted. Dong, et al (2016) demonstrate how radical-level features incorporated at the character-level for named entity recognition achieve state-of-the-art accuracy for Chinese. The most convincing attempt to tag Korean at the morpheme level is by Choi, et al (2016) who achieve state-of-the-art (dictionary-less) performance by using a multi-stage Bi-LSTM-CRF model that involves the splitting of Korean character input into constituent graphemes. However, the implicit assumption that Korean characters must first be split into graphemes to achieve optimal performance for morphological analysis is not well supported, and we should consider the splitting of characters into graphemes to be employing linguistic knowledge specific to Korean.

In our paper, we seek to answer the question of whether Korean morphemes can be tagged without grapheme-level splitting, rules specific to the language, or a dictionary. Although we initially considered Bi-LSTM-CRF for our model architecture, we show that the performance benefit by adding CRF is minimal and practically unnecessary compared to a standard Bi-LSTM model. Furthermore, CRF adds training and inference computational complexity due to the Viterbi algorithm.

3 Lemma and Form Alignment

고(go)	통(tong)	스(seu)	런(reon)
B-KEEP	I-KEEP	B-KEEP	I-MOD-렵, B-MOD-ㄴ
고통(go-tong)		스렵(seu-reob)	ㄴ(n)

Figure 1: Gold morphological transformation actions given by alignment oracle, including the resulting morphemes after running the BIO actions (Sejong corpus)

고통(go-tong)	스렵(seu-reob)	ㄴ(n)
NNG	XSA	ETM

Figure 2: Gold tagging actions (Sejong corpus)

In the Sejong corpus, Eojeol are annotated with their corresponding POS-tagged morpheme constituents, exactly as shown in Table 2. As mentioned earlier, morpheme spelling transformations may occur, and therefore the morpheme constituents may have slightly different graphemes than what is present in the original Eojeol form. To generate our training data, we must align the Eojeol form and its constituent morphemes at the character level, as we forbid using linguistic knowledge such as sub-character elements (graphemes) in our model. Like most agglutinative

languages, the Eojeol form and lemmas (morphemic elements in the Sejong corpus) often share overlapping characters at the beginning or end, and we utilize this assumption in our algorithm.

Gold Action	Count
B-KEEP	23,725,534
I-KEEP	8,650,166
B-MOD-하(ha), B-MOD-ㄴ(n)	153,130
NOOP	131,016
B-MOD:하(ha), B-MOD-았(ass)	61,592
B-MOD:이(i), B-MOD-ㄴ(n)	58,093
B-MOD:하(ha), B-MOD-아(a)	57,515
I-MOD:하(ha), B-MOD-아(a)	48,987
B-MOD:되(doe), B-MOD-ㄴ(n)	41,335
B-MOD:하(ha), B-MOD-ㄹ(l)	36,419

Table 3: Top 10 morphing actions

We present an action-based algorithm (called an "alignment oracle") to align two arbitrary strings. Our oracle attempts to generate a 1:1 character-level mapping between the morphological form and the lemmas of an Eojeol by searching for a prefix, suffix, and modified inner string portion. Three primary actions are defined: KEEP (no modification to character), NOOP (drop character), and MOD (modify character). In the case of Korean, morphological transformations happen at the end of a form, so there is rarely a common suffix unless no transformation occurs at all. These primary actions are then augmented with B- and I- actions to facilitate morpheme segmentation. It is important to note that our algorithm is not specific in any way to the Sejong corpus or Korean itself.

The full process is demonstrated in Figure 1 starting from the source form. The gold untagged segmented lemma form is shown in the bottom row, and the actions generated by the oracle to generate the lemma are given in the middle row. The first three characters (go-tong-seu) are preserved with KEEP actions and the last character is considered the "modified inner string". In this case, the number of actions (5) exceeds the number of full input characters (4), and therefore two actions are assigned to the last character which split the "reon" syllable into "reob" and "n". The output after morpheme segmentation can be seen in the bottom row. In Figure 2, these output morphemes are then placed through a standard sequential tagger to assign part-of-speech tags.

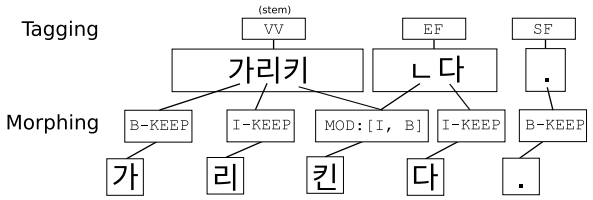


Figure 3: Actions for transformation with output segmentation

For Korean, the task is considerably more complicated. Rather than merely character-level transformation, new morpheme boundaries based on the results of those transformations are also required. A segmentation module adds B-/I- (beginning and inside) annotations to the KEEP and MOD actions. These actions allow morpheme segmentation to take place even amidst the modified character output sequence. This is detailed in Figure 1, where the final consonant sub-character unit ("n") of the last character of input ("reon") is transformed to "b" and the resulting fused full character is appended to the previous output morpheme, whereas the "n"

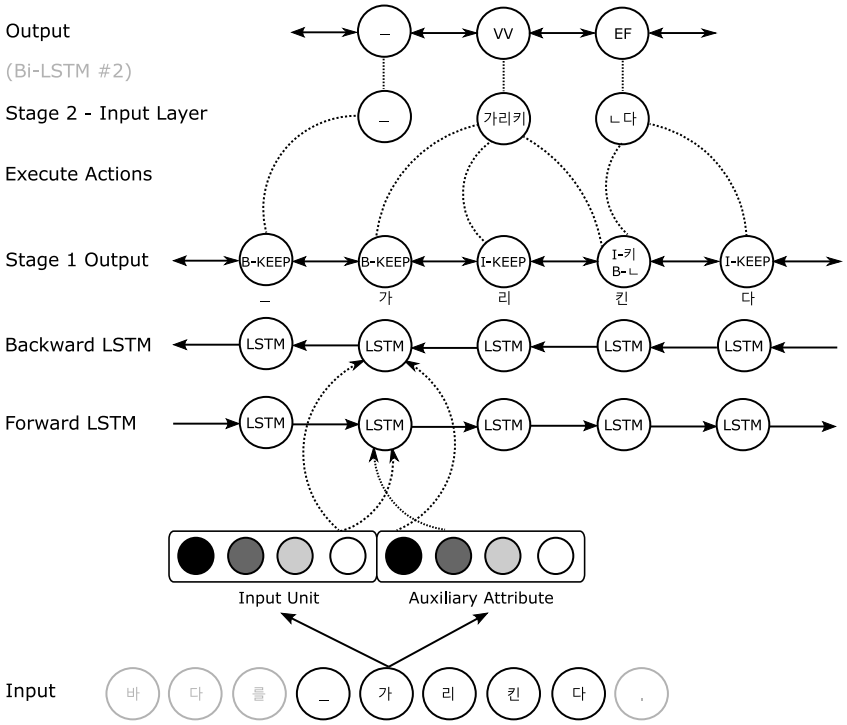


Figure 4: Partial example of two-stage tagging process for Korean phrase "to point out the sea"

sub-character unit becomes separated and represented as an entirely new morpheme itself. The top 10 resulting actions for the Sejong corpus on the form and lemma alignment stage are shown in Table 3.

4 The Model

Our model makes the assumption that in order to support morphological analysis for languages like Korean, two stages are required, which we call morphing and tagging. For tasks such as morphological transformation, word-level morphological analysis, or morpheme segmentation, only one stage is strictly necessary. To obtain tags for morphemes following morphological transformation, as in Korean, both stages are necessary, with tagging following morphing. The stages have no fundamental difference from each other: the second stage simply acts on the result of applying actions output by the first stage. Each stage outputs a single action for a single input unit. A single action could be as simple as a tag or as complicated as information resulting in advanced multi-character transformation along with the specification of the morphemic segments of those resulting characters. Model parameters are trained independently for each stage unless otherwise specified.

The model presented in this paper is inspired by word-level morphological analysis work by Heigold, et al (2016a) with the goal of allowing analysis at arbitrary units of input at each stage. Because we do not specify whether the input unit should be a word, morpheme, character, or even a unit at the sub-character level (such as graphemes in Korean or radicals in Chinese), we theoretically have the flexibility to tag a variety of languages at any level.

We employ a bidirectional Long Short-Term Memory (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005) network in our model and also experiment with optimization to a conditional random field (CRF) objective (Lafferty et al., 2001). In theory, CRF allows us to consider the likelihood of neighboring outputs and therefore jointly decode the highest probable chain of output labels for a given set of inputs. Although we posit that CRF is not strictly necessary, the overall architecture of our model is otherwise identical to a standard Bi-LSTM-CRF sequence tagging model (Huang et al., 2015) used for POS tagging and NER.

An overview of the architecture is shown in Figure 4. The input unit is embedded as a multi-dimensional vector. At the input level, an auxiliary attribute may be concatenated with the input unit to include auxiliary information for the unit, such as word break-level information, although empirical findings indicate our model performs best when using only the input unit. Nested embeddings as in (2016a) may also be appended at the input level.

All embeddings are concatenated to form a combined embedding which is then passed to the primary Bi-LSTM-CRF network and trained against a set of output actions. Whitespace delimiting Eojeols is represented as a reserved spacing token in the input unit.

Although each stage is independent and can accept an arbitrary unit of input suitable for any language, the following sections describe how this model pertains to our primary task of morpheme-level morphological analysis for the Korean language.

4.1 Morphing

The first stage of the model operates at the character level and is responsible for morphological transformation of the input form into the desired output lemma(s). During training, each input character is assigned one of three primary types by the alignment oracle as described in Section 3. Morpheme segmentation actions are also generated and augmented to the transformation actions at this stage for proper morpheme boundary identification. During inference, instead of using the alignment oracle, one action (including transformation and B-/I- tags) is predicted for each character based on trained parameters. At this point, tags are not yet assigned to each output morpheme.

4.2 Tagging

After the necessary morphological transformation and segmentation, tagging occurs at the morpheme level and acts on output produced by actions in the first stage of the model. An example of this is shown in Figure 2. In this stage, the action is simply to assign a POS tag to the morpheme, which is the input unit.

5 Experiments

5.1 Datasets

We conduct experiments using the full Sejong Korean Balanced Corpus dataset. The experiments are coded in Python using the TensorFlow library. The Sejong Corpus has been preprocessed to resolve punctuation inconsistencies and other surface-level errors. All datasets are converted at the sentence-level to a simple two-column format with each line containing an input unit and target action.

For all experiments, we follow an 85/10/5 cross-validation split for training, testing, and validation sets respectively. All data is randomly shuffled prior to splitting. Actions are inferred from the dataset by using lemma and form alignment. For evaluation, output from predicted actions in the first stage is used as input to the subsequent stage.

UniTagger represents the model proposed in this paper. The following number (for example, 500) represents the maximum action count for the morphing stage. The tagging stage only has as many actions as possible POS tags (45 in the Sejong corpus, including the reserved space token). Action pruning is performed at the training level, which removes from the training set the least common morphological transformation actions generated by the alignment oracle. For fair evaluation, actions are not removed from validation or test sets. All accuracy figures in this paper are reported based on a held-out test set.

5.2 Training

Optimization is performed using Adam (Kingma and Ba, 2014) with a learning rate of 0.001 and decay of 0.9. In the case of multi-stage models, model parameters are optimized independently for each stage. We use identical hyperparameters for all morphing and tagging models. Input

unit embedding size was set to 300 (for character and morpheme input). The final Bi-LSTM concatenating all embeddings before an optional CRF layer was used with an LSTM unit size of 300. Batch size was set to 64 for all experiments, except for the CRF experiment where it was set to 16. The maximum LSTM input length was set at a per-batch level which yielded optimal performance, and the maximum number of input units (whether characters or morphemes) was limited to 400 in both stages. A dropout of 10% was used for the reported model with best performance. Dropout is only applied at the unit embedding layer. Epoch count was set to 100 with early-stopping after 3 epochs with no improvement in validation set performance. Experiments were performed on GTX 1080 Ti 11GB GPUs. Average total training duration was around 5 hours for the entire Sejong dataset on a GTX 1080 Ti. In TensorFlow, the NVIDIA CuDNN-optimized LSTM was used (Appleyard et al., 2016).

5.3 Results

In Table 4, we show Eojeol-level morphological analysis accuracy for Korean. Note here that an Eojeol is considered correctly tagged only if all its constituent morphemes have been transformed, segmented, and tagged properly. Table 5 measures sentence-level tagging performance, which is the accuracy of all morphemes being transformed and tagged properly.

Model	**Accuracy**
Lee, et al. (2005)	92.96
Ahn, et al. (2007)	93.12
Lee, et al. (2009)	92.95
Choi, et al. (2016)	94.89
UniTagger-500	**96.20**

Table 4: End-to-end Eojeol-level accuracy for morphological analysis of Korean (Sejong Corpus)

Model	**Model Type**	**Acc**
Choi, et al. (2016)	Bi-LSTM-CRF	61.00
UniTagger-500	Bi-LSTM	**70.83**

Table 5: End-to-end sentence-level accuracy for morphological analysis of Korean (Sejong Corpus)

Form	**Gold Action**
났(nass)	B-나 + B-았
샀(sass)	B-사 + B-았
잤(jass)	B-자 + B-았
팠(pass)	B-파 + B-았
됐(daess)	B-되 + B-었
했(haess)	B-하 + B-았
녔(nyeoss)	B-니 + B-었
렸(ryeoss)	B-리 + B-었
셨(syeoss)	B-시 + B-었
졌(jieoss)	B-지 + B-었
겼(gyeoss)	B-기 + B-었
왔(oass)	B-오 + B-았
놨(noass)	B-놓 + B-았
췄(chuweoss)	B-추 + B-었

Table 6: Past tense morphing actions shown in embedding and gold actions from Sejong corpus

Model	**Model Type**	**Acc**
Choi, et al. (2016)	Bi-LSTM-CRF	67.25
UniTagger-500	Bi-LSTM	**79.49**

Table 7: Eojeol-level OOV accuracy

5.4 Analysis and Discussion

Our results show that our model can outperform previous state-of-the-art performance for Eojeol and sentence-level morphological analysis of Korean without linguistic knowledge.

When the dropout factor is adjusted, all metrics follow a similar trend as seen in Figure 5. Sentence accuracy is most sensitive to dropout factor adjustment. Best performance is achieved

with a dropout rate of 10%, and increasing the dropout rate further does not increase Eojeol-level OOV accuracy. This is a positive finding, as it indicates the model is not considerably overfitting to the training data beyond approximately the 10% level.

In Figure 6, a 300-dimensional unit embedding layer of the morphing stage is visualized using 2-component t-SNE. The corresponding gold actions are shown in Table 6, where all past tense morphemes end with final consonant "ㅆ" (ss). The model is able to infer that most of the forms shown in the graph represent the past tense and that they share a similar transformation pattern at the final consonant grapheme level. This shows that our model is able to correlate similar sub-character level morphological transformations even when operating at the character level. Furthermore, it is worth noting that Chinese characters still occur rarely in the Korean language in certain contexts. We can see Chinese characters grouped in a cluster, which shows that the model is able to distinguish one character-rich language (Korean) from another (Chinese). Other characters, such as punctuation, are also grouped by type in largely distinct clusters with occasional overlap.

Joint training of both stages was also attempted, though an initial investigation suggests that performance is not significantly different from training each stage's parameters independently.

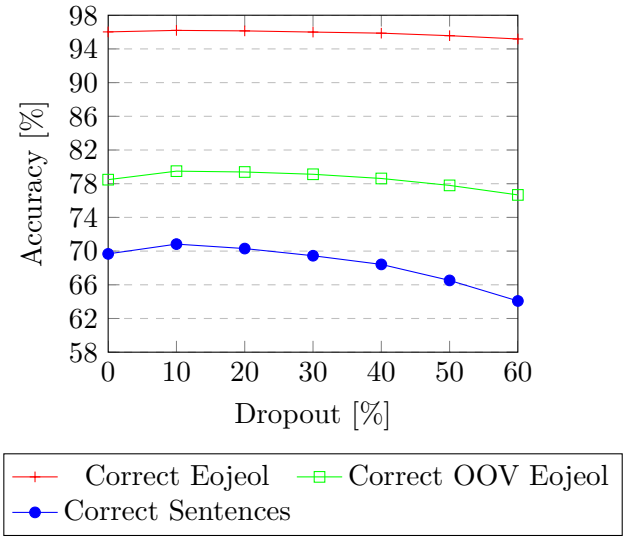


Figure 5: Impact of dropout on end-to-end tagging performance

The use of using an auxiliary binary break level attribute to represent whitespace was also investigated, but significantly higher accuracy was achieved by using a reserved spacing token instead. Despite the auxiliary break level attribute embedding, both stages of the model have a tendency to learn ambiguous morpheme transformations for adjacent Eojeol. In other words, even though the morpheme transformation and tags are correct, the Eojeol boundaries were incorrectly identified. With the reserved spacing token, this issue was extremely rare.

6 Conclusion and Future Work

In this work, we address the commonly held notion that Korean can not be tagged with competitive performance at the character level without prior linguistic knowledge. Our model architecture is not novel compared to previous work. The novelty of our morphological analyzer is its striking simplicity compared to previous approaches for character-rich languages such as Korean. The alignment oracle does not require any cost value for alignment operations such as in the Needleman–Wunsch algorithm (Needleman and Wunsch, 1970). The Bi-LSTM model is able to learn and utilize alignments that are purely arbitrary and apply them to unseen test data. Even when significantly limiting the number of actions in the training data, we show that by using the most common morphological transformation actions in an agglutinative language, we can

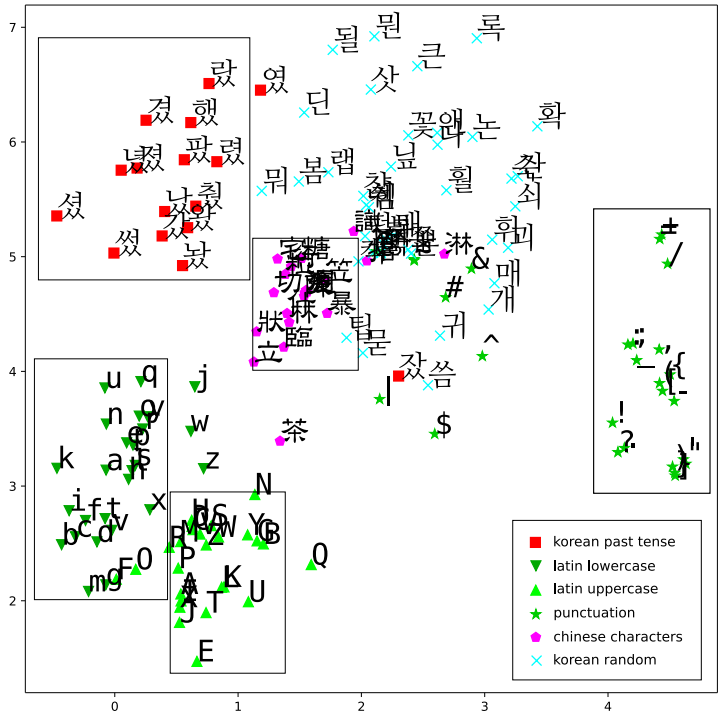


Figure 6: Deep embeddings of characters at the morphing stage (t-SNE)

exceed the performance of a model that uses linguistic knowledge such as sub-character features. We also show that the widely used CRF layer may in fact be unnecessary for high performance and add unnecessary computational complexity. This exceeded our own expectations and raises the possibility that a single architecture can handle tagging universally with only two simple Bi-LSTM stages. We contribute the necessary source code to replicate the experiments and to attempt alignment and training for any other language, assuming a corpus exists. Nevertheless, there are several points that future work should address.

Out-of-vocabulary morphemes generated by the morphing stage can also result in errors at the tagging stage, as the tagging stage was trained on the assumption of gold morphemes. We would like to experiment with including possible morpheme transformation errors at the tagging stage to determine if tagging performance can be improved.

We attempted joint training but found that end-to-end accuracy was marginally lower. We suspect this is because optimization of each individual stage is hindered by attempting to find optimal parameters for both stages. Future work should attempt joint training of an end-to-end model with the preinitialized parameters from optimizing each stage independently, which has been shown to be ideal in sequential models (Tang et al., 2016).

Lastly, although we are unaware of a language more character-rich and more morphologically complex than Korean, we would like to see our model applied to other morphologically complex languages to prove its universality. At the time of writing, we lacked sufficient baseline figures and methodology for generating training data for analyzing other languages at the morpheme level using the Universal Dependencies corpus, and the morphological tags were often conflated with part-of-speech tags. The baselines we found did not specify whether or not morpheme segmentation was taken into account. Without this information, it would be difficult to prove the performance of our model for other languages and we decided to leave training other languages as future work. That being said, our model does not employ any linguistic knowledge specific to Korean, and we therefore have no reason to believe it cannot be trained on any other arbitrary corpus with minor modifications at the preprocessing level.

Acknowledgements

This research was supported by the MSIT (Ministry of Science and ICT), South Korea, under the ITRC (Information Technology Research Center) support program ("Research and Development of Human-Inspired Multiple Intelligence") supervised by the IITP (Institute for Information & Communications Technology Promotion). Additionally, this work was supported by the National Research Foundation of Korea (NRF) grant funded by the South Korean government (MSIP) (No. NRF-2016R1A2B2015912).

References

Young-Min Ahn and Young-Hoon Seo. 2007. Korean part-of-speech tagging using disambiguation rules for ambiguous word and statistical information. *IEEE International Conference on Convergence Information Technology*, pages 1598–1601.

Jeremy Appleyard, Tomáš Kociský, and Phil Blunsom. 2016. Optimizing performance of recurrent neural networks on gpus. *arXiv*, (1604.01946).

Jihun Choi, Jonghem Youn, and Sang goo Lee. 2016. A grapheme-level approach for constructing a korean morphological analyzer without linguistic knowledge. *IEEE International Conference on Big Data*, pages 3872–3879.

Chuanhai Dong, Jiajun Zhang, Chengqing Zong, Masanori Hattori, and Hui Di. 2016. Character-based lstm-crf with radical-level features for chinese named entity recognition. *International Conference on Computer Processing of Oriental Languages*, pages 239–250.

A. Graves and J. Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5–6):602–610.

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2016a. Neural morphological tagging from characters for morphologically rich languages. *arXiv*, (1606.06640).

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2016b. Scaling character-based morphological tagging to fourteen languages. *IEEE International Conference on Big Data*.

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

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv*, (1508.01991).

Seungshik Kang and Yungtaek Kim. 1992. A computational analysis model of irregular verbs in korean morphological analyzer. *Journal of Korea Information Science Society*, 19(2):151–164.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv*, (1412.6980).

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of ICML*.

Do-Gil Lee and Hae-Chang Rim. 2005. Probabilistic models for korean morphological analysis. *Companion to the Proceedings of the International Joint Conference on Natural Language Processing*, pages 197–202.

Do-Gil Lee and Hae-Chang Rim. 2009. Probabilistic modeling of korean morphology. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(5):945–955.

Saul B. Needleman and Christian D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*.

Sangwon Park, D Choi, E Kim, and K.-S Choi. 2010. A plug-in component-based korean morphological analyzer.

Tjong Kim Sang and Jorn Veenstra. 1999. Representing text chunks. *EACL '99 Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 173–179.

Gwang-Seob Shim and Jae-Hyung Yang. 2004. High speed korean morphological analysis based on adjacency condition check. *KIISE: Software and Applications*, 31(1):89–99.

Hao Tang, Weiran Wang, Kevin Gimpel, and Karen Livescu. 2016. End-to-end training approaches for discriminative segmental models. *IEEE Spoken Language Technology Workshop*.

Why does PairDiff *work*? – A Mathematical Analysis of Bilinear Relational Compositional Operators for Analogy Detection

Huda Hakami
The University of Liverpool
Liverpool, UK
h.a.hakami@liv.ac.uk

Kohei Hayashi
Preferred Networks
Tokyo, Japan
hayashi.kohei@gmail.com

Danushka Bollegala
The University of Liverpool
Liverpool, UK
danushka.bollegala@liv.ac.uk

Abstract

Representing the semantic relations that exist between two given words (or entities) is an important first step in a wide-range of NLP applications such as analogical reasoning, knowledge base completion and relational information retrieval. A simple, yet surprisingly accurate method for representing a relation between two words is to compute the vector offset (PairDiff) between their corresponding word embeddings. Despite the empirical success, it remains unclear as to whether PairDiff is the best operator for obtaining a relational representation from word embeddings. We conduct a theoretical analysis of generalised bilinear operators that can be used to measure the ℓ_2 relational distance between two word-pairs. We show that, if the word embeddings are standardised and uncorrelated, such an operator will be independent of bilinear terms, and can be simplified to a linear form, where PairDiff is a special case. For numerous word embedding types, we empirically verify the uncorrelation assumption, demonstrating the general applicability of our theoretical result. Moreover, we experimentally discover PairDiff from the bilinear relational compositional operator on several benchmark analogy datasets.

1 Introduction

Different types of semantic relations exist between words such as HYPERNYMY between *ostrich* and *bird*, or ANTONYMY between *hot* and *cold*. If we consider entities[1], we can observe even a richer diversity of relations such as FOUNDER-OF between *Bill Gates* and *Microsoft*, or CAPITAL-OF between *Tokyo* and *Japan*. Identifying the relations between words and entities is important for various Natural Language Processing (NLP) tasks such as automatic knowledge base completion (Socher et al., 2013), analogical reasoning (Turney and Littman, 2005; Bollegala et al., 2009) and relational information retrieval (Duc et al., 2010). For example, to solve a word analogy problem of the form "*a* is to *b* as *c* is to *?*", the relationship between the two words in the pair (a, b) must be correctly identified in order to find candidates d that have similar relations with c. For example, given the query "*Bill Gates* is to *Microsoft* as *Steve Jobs* is to *?*", a relational search engine must retrieve *Apple Inc.* because the FOUNDER-OF relation exists between the first and the second entity pairs.

Two main approaches for creating relation embeddings can be identified in the literature. In the first approach, from given corpora or knowledge bases, word and relation embeddings are *jointly* learnt such that some objective is optimised (Guo et al., 2016; Yang et al., 2015; Nickel et al., 2016; Bordes et al., 2013; Rocktäschel et al., 2016; Minervini et al., 2017; Trouillon et al., 2016). In this approach, word and relation embeddings are considered to be *independent* parameters that must be learnt by the embedding method. For example, TransE (Bordes et al., 2013) learns the word and relation embeddings such that we can accurately predict relations (links) in a given knowledge base using the learnt word and relation embeddings. Because relations are learnt independently from the words, we refer to methods that are based on this approach as *independent* relational embedding methods.

[1] We interchangeably use the terms *word* and *entity* to represent both unigrams as well as a multi-word expressions including named entities.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2493–2504
Santa Fe, New Mexico, USA, August 20-26, 2018.

A second approach for creating relational embeddings is to apply some operator on two word embeddings to *compose* the embedding for the relation that exits between those two words, if any. In contrast to the first approach, we do not have to learn relational embeddings and hence this can be considered as an unsupervised setting, where the compositional operator is predefined. A popular operator for composing a relational embedding from two word embeddings is PairDiff, which is the vector difference (offset) of the word embeddings (Mikolov et al., 2013b; Levy and Goldberg, 2014; Vylomova et al., 2016; Bollegala et al., 2015b; Blacoe and Lapata, 2012). Specifically, given two words a and b represented by their word embeddings respectively $\boldsymbol{a}$ and $\boldsymbol{b}$, the relation between a and b is given by $\boldsymbol{a} - \boldsymbol{b}$ under the PairDiff operator. Mikolov et al. (2013b) showed that PairDiff can accurately solve analogy equations such as $\overrightarrow{\mathbf{king}} - \overrightarrow{\mathbf{man}} + \overrightarrow{\mathbf{woman}} = \overrightarrow{\mathbf{queen}}$, where we have used the top arrows to denote the embeddings of the corresponding words. Bollegala et al. (2015a) showed that PairDiff can be used as a proxy for learning better word embeddings and Vylomova et al. (2016) conducted an extensive empirical comparison of PairDiff using a dataset containing 16 different relation types. Besides PairDiff, concatenation (Hakami and Bollegala, 2017; Yin and Schütze, 2016), circular correlation and convolution (Nickel et al., 2016) have been used in prior work for representing the relations between words. Because the relation embedding is composed using word embeddings instead of learning as a separate parameter, we refer to methods that are based on this approach as *compositional* relational embedding methods. Note that in this approach it is implicitly assumed that there exist only a single relation between two words.

In this paper, we focus on the operators that are used in compositional relational embedding methods. If we assume that the words and relations are represented by vectors embedded in some common space, then the operator we are seeking must be able to produce a vector representing the relation between two words, given their word embeddings as the only input. Although there have been different proposals for computing relational embeddings from word embeddings, it remains unclear as to what is the best operator for this task. The space of operators that can be used to compose relational embeddings is open and vast. A space of particular interest from a computational point-of-view is the bilinear operators that can be parametrised using tensors and matrices. Specifically, we consider operators that consider pairwise interactions between two word embeddings (second-order terms) and contributions from individual word embeddings towards their relational embedding (first-order terms). The optimality of a relational compositional operator can be evaluated, for example, using the expected relational distance/similarity such as ℓ_2 between analogous (positive) vs. nonanalogous (negative) word-pairs.

If we assume that word embeddings are standardised, uncorrelated and word-pairs are i.i.d, then we prove in §3 that bilinear relational compositional operators are independent of bilinear pairwise interactions between the two input word embeddings. Moreover, under regularised settings (§3.1), the bilinear operator further simplifies to a linear combination of the input embeddings, and the expected loss over positive and negative instances becomes zero. In §4.1, we empirically validate the uncorrelation assumption for different pre-trained word embeddings such as the Continuous Bag-of-Words Model (CBOW) (Mikolov et al., 2013a), Skip-Gram with negative sampling (SG) (Mikolov et al., 2013a), Global Vectors (GloVe) (Pennington et al., 2014), word embeddings created using Latent Semantic Analysis (LSA) (Deerwester et al., 1990), Sparse Coding (HSC) (Faruqui et al., 2015; Yogatama et al., 2015), and Latent Dirichlet Allocation (LDA) (Blei et al., 2003). This empirical evidence implies that our theoretical analysis is applicable to relational representations composed from a wide-range of word embedding learning methods. Moreover, our experimental results show that a bilinear operator reaches its optimal performance in two different word-analogy benchmark datasets, when it satisfies the requirements of the PairDiff operator. We hope that our theoretical analysis will expand the understanding of relational embedding methods, and inspire future research on accurate relational embedding methods using word embeddings as the input.

2 Related Work

As already mentioned in §1, methods for representing a relation between two words can be broadly categorised into two groups depending on whether the relational embeddings are learnt *independently* of the word embeddings, or they are *composed* from the word embeddings, in which case the relational

embeddings fully depend on the input word embeddings. Next, we briefly overview the different methods that fall under each category. For a detailed survey of relation embedding methods see Nickel et al. (2015).

Given a knowledge base where an entity h is linked to an entity t by a relation r, the TransE model (Bordes et al., 2013) scores the tuple (h, t, r) by the ℓ_1 or ℓ_2 norm of the vector $(\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t})$. Nickel et al. (2011) proposed RESCAL, which uses $\boldsymbol{h}^\top \mathbf{M}_r \boldsymbol{t}$ as the scoring function, where $\mathbf{M}_r$ is a matrix embedding of the relation r. Similar to RESCAL, Neural Tensor Network (Socher et al., 2013) also models a relation by a matrix. However, compared to vector embeddings of relations, matrix embeddings increase the number of parameters to be estimated, resulting in an increase in computational time/space and likely to overfit. To overcome these limitations, DistMult (Yang et al., 2015) models relations by vectors and use elementwise multilinear dot product $\boldsymbol{r} \odot \boldsymbol{h} \odot \boldsymbol{t}$. Unfortunately, DistMult cannot capture directionality of a relation. Complex Embeddings (Trouillon et al., 2016) overcome this limitation of DistMult by using complex embeddings and defining the score to be the real part of $\boldsymbol{r} \odot \boldsymbol{h} \odot \bar{\boldsymbol{t}}$, where $\bar{\boldsymbol{t}}$ denotes the complex conjugate of $\boldsymbol{t}$.

The observation made by Mikolov et al. (2013b) that the relation between two words can be represented by the difference between their word embeddings sparked a renewed interest in methods that compose relational embeddings using word embeddings. Word analogy datasets such as Google dataset (Mikolov et al., 2013b), SemEval 2012 Task2 dataset (Jurgens et al., 2012), BATS (Drozd et al., 2016) etc. have established as benchmarks for evaluating word embedding learning methods.

Different methods have been proposed to measure the similarity between the relations that exist between two given word pairs such as CosMult, CosAdd and PairDiff (Levy and Goldberg, 2014; Bollegala et al., 2015a). Vylomova et al. (2016) studied as to what extent the vectors generated using simple PairDiff encode different relation types. Under supervised classification settings, they conclude that PairDiff can cover a wide range of semantic relation types. Holographic embeddings proposed by Nickel et al. (2016) use circular convolution to mix the embeddings of two words to create an embedding for the relation that exist between those words. It can be showed that circular correlation is indeed an elementwise product in the Fourier space and is mathematically equivalent to complex embeddings (Hayashi and Shinbo, 2017).

Although PairDiff operator has been widely used in prior work for computing relation embeddings from word embeddings, to the best of our knowledge, no theoretical analysis has been conducted so far explaining why and under what conditions PairDiff is optimal, which is the focus of this paper.

3 Bilinear Relation Representations

Let us consider the problem of representing the semantic relation $\boldsymbol{r}(\boldsymbol{h}, \boldsymbol{t})$ between two given words h and t. We assume that h and t are already represented in some d-dimensional space respectively by their word embeddings $\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^d$. The relation between two words can be represented using different linear algebraic structures. Two popular alternatives are vectors (Nickel et al., 2016; Bordes et al., 2013; Minervini et al., 2017; Trouillon et al., 2016) and matrices (Socher et al., 2013; Bollegala et al., 2015b). Vector representations are preferred over matrix representations because of the smaller number of parameters to be learnt (Nickel et al., 2015).

Let us assume that the relation r is represented by a vector $\boldsymbol{r} \in \mathbb{R}^\delta$ in some δ-dimensional space. Therefore, we can write $\boldsymbol{r}(\boldsymbol{h}, \boldsymbol{t})$ as a function that takes two vectors (corresponding to the embeddings of the two words) as the input and returns a single vector (representing the relation between the two words) as given in (1).

$$\boldsymbol{r}: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^\delta \tag{1}$$

Having both words and relations represented in the same $\delta = d$ dimensional space is useful for performing linear algebraic operations using those representations in that space. For example, in TransE (Bordes et al., 2013), the strength of a relation r that exists between two words h and t is computed as the $\ell_{1,2}$ norm of the vector $(\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t})$ using the word and relation embeddings. Such direct comparisons between word and relation embeddings would not be possible if words and relations were not embedded in the

same vector space. If $\delta < d$, we can first project word embeddings to a lower δ-dimensional space using some dimensionality reduction method such as SVD, whereas if $\delta > d$ we can learn higher δ-dimensional overcomplete word representations (Faruqui et al., 2015) from the original d-dimensional word embeddings. Therefore, we will limit our theoretical analysis to the $\delta = d$ case for ease of description.

Different functions can be used as $\boldsymbol{r}(\boldsymbol{h}, \boldsymbol{t})$ that satisfy the domain and range requirements specified by (1). If we limit ourselves to bilinear functions, the most general functional form is given by (2).

$$\boldsymbol{r}(\boldsymbol{h}, \boldsymbol{t}) = \boldsymbol{h}^\top \underline{\mathbf{A}} \boldsymbol{t} + \mathbf{P}\boldsymbol{h} + \mathbf{Q}\boldsymbol{t} \tag{2}$$

Here, $\underline{\mathbf{A}} \in \mathbb{R}^{d \times d \times d}$ is a 3-way tensor in which each slice is a $d \times d$ real matrix. Let us denote the k-th slice of $\underline{\mathbf{A}}$ by $\mathbf{A}^{(k)}$ and its (i, j) element by $A_{ij}^{(k)}$. The first term in (2) corresponds to the pairwise interactions between $\boldsymbol{h}$ and $\boldsymbol{t}$. $\mathbf{P}, \mathbf{Q} \in \mathbb{R}^{d \times d}$ are the nonsingular[2] projection matrices involving first-order contributions respectively of $\boldsymbol{h}$ and $\boldsymbol{t}$ towards $\boldsymbol{r}$.

Let us consider the problem of learning the simplest bilinear functional form according to (2) from a given dataset of analogous word-pairs $\mathcal{D}_+ = \{((h, t), (h', t'))\}$. Specifically, we would like to learn the parameters $\underline{\mathbf{A}}$, $\mathbf{P}$ and $\mathbf{Q}$ such that some distance (loss) between analogous word-pairs is minimised. As a concrete example of a distance function, let us consider the popularly used Euclidean distance[3] (ℓ_2 loss) for two word pairs given by (3).

$$J((h, t), (h', t')) = \left|\left|\boldsymbol{r}(\boldsymbol{h}, \boldsymbol{t}) - \boldsymbol{r}(\boldsymbol{h'}, \boldsymbol{t'})\right|\right|_2^2 \tag{3}$$

If we were provided only analogous word-pairs (i.e. positive examples), then this task could be trivially achieved by setting all parameters to zero. However, such a trivial solution would not generalise to unseen test data. Therefore, in addition to $\mathcal{D}_+$ we would require a set of non-analogous word-pairs $\mathcal{D}_-$ as negative examples. Such negative examples are often generated in prior work by randomly corrupting positive relational tuples (Nickel et al., 2016; Bordes et al., 2013; Trouillon et al., 2016) or by training an adversarial generator (Minervini et al., 2017).

The total loss J over both positive and negative training data can be written as follows:

$$J = \sum_{((h,t),(h',t')) \in \mathcal{D}_+} \left|\left|\boldsymbol{r}(\boldsymbol{h}, \boldsymbol{t}) - \boldsymbol{r}(\boldsymbol{h'}, \boldsymbol{t'})\right|\right|_2^2 - \sum_{((h,t),(h',t')) \in \mathcal{D}_-} \left|\left|\boldsymbol{r}(\boldsymbol{h}, \boldsymbol{t}) - \boldsymbol{r}(\boldsymbol{h'}, \boldsymbol{t'})\right|\right|_2^2 \tag{4}$$

Assuming that the training word-pairs are randomly sampled from $\mathcal{D}_+$ and $\mathcal{D}_-$ according to two distributions respectively p_+ and p_-, we can compute the total expected loss, $\mathbb{E}_p[J]$, as follows:

$$\mathbb{E}_p[J] = \mathbb{E}_{p_+}\left[\left|\left|\boldsymbol{r}(\boldsymbol{h}, \boldsymbol{t}) - \boldsymbol{r}(\boldsymbol{h'}, \boldsymbol{t'})\right|\right|_2^2\right] - \mathbb{E}_{p_-}\left[\left|\left|\boldsymbol{r}(\boldsymbol{h}, \boldsymbol{t}) - \boldsymbol{r}(\boldsymbol{h'}, \boldsymbol{t'})\right|\right|_2^2\right] \tag{5}$$

We make the following assumptions to further analyse the properties of relational embeddings.

Uncorrelation: The correlation between any two distinct dimensions of a word embedding is zero. One might think that the uncorrelation of word embedding dimensions to be a strong assumption, but we later show its validity empirically in §4.1 for a wide range of word embeddings.

Standardisation: Word embeddings are standardised to zero mean and unit variance. This is a linear transformation in the word embedding space and does not affect the topology of the embedding space. In particular, translating word embeddings such that they have a zero mean has shown to improve performance in similarity tasks Mu et al. (2018).

Relational Independence Word pairs in the training data are assumed to be i.i.d. For example, whether a particular semantic relation r exists between h and t, is assumed to be independent of any other relation r' that exists between h' and t' in a different pair. For example, (*ostrich*, is-a-large, *bird*) is independent from (*Trump*, president-of, *USA*).

[2] If the projection matrix is nonsingular, then the inverse projection exists, which preserves the dimensionality of the embedding space.

[3] For ℓ_2 normalised vectors, their Euclidean distance is a monotonously decreasing function of their cosine similarity.

For relation representations given by (2), 1 holds:

Theorem 1. *Consider the bilinear relational embedding defined by 2 computed using uncorrelated word embeddings. If the word embeddings are standardised, then the expected loss given by 5 over a relationally independent set of word pairs is independent of* $\underline{\mathbf{A}}$.

Proof. Let us consider the bilinear term in (2), because i and $j(\neq i)$ dimensions of word embeddings are uncorrelated by the assumption (i.e. $\text{corr}(u_i, u_j) = 0$), from the definition of correlation we have,

$$\text{corr}(u_i, u_j) = \mathbb{E}[u_i u_j] - \mathbb{E}[u_i]\mathbb{E}[u_j] = 0 \tag{6}$$

$$\mathbb{E}[u_i u_j] = \mathbb{E}[u_i]\mathbb{E}[u_j]. \tag{7}$$

Moreover, from the standardisation assumption we have, $\mathbb{E}[u_i] = 0, \ \ \forall_{i=1\ldots n}$. From (7) it follows that:

$$\mathbb{E}[u_i u_j] = 0 \tag{8}$$

for $i \neq j$ dimensions.

We will next show that (5) is independent of $\underline{\mathbf{A}}$. For this purpose, let us consider the $\mathbb{E}_{p_+}$ term first and write the k-th dimension of $\boldsymbol{r}(\boldsymbol{h}, \boldsymbol{t})$ using $\mathbf{A}^{(k)}$, $\mathbf{P}$ and $\mathbf{Q}$ as follows:

$$\sum_{i,j} \left(A^{(k)}_{ij} h_i t_j \right) + \sum_n P_{kn} h_n + \sum_n Q_{kn} t_n \tag{9}$$

Plugging (9) in (5) and computing the loss over all positive training instances we get,

$$\mathbb{E}_{p_+}[\sum_k (\sum_{i,j} \left(A^{(k)}_{ij} (h_i t_j - h'_i t'_j) \right) + \sum_n P_{kn}(h_n - h'_n) + \sum_n Q_{kn}(t_n - t'_n))^2] \tag{10}$$

Terms that involve only elements in $\mathbf{A}^{(k)}$ take the form:

$$\begin{aligned} &\sum_{i,j} \sum_{l,m} \mathbb{E}_{p_+} \left[A^{(k)}_{ij} A^{(k)}_{lm} (h_i t_j - h'_i t'_j)(h_l t_m - h'_l t'_m) \right] \\ &= \sum_{i,j} \sum_{l,m} A^{(k)}_{ij} A^{(k)}_{lm} (\mathbb{E}_{p_+}[h_i t_j h_l t_m] - \mathbb{E}_{p_+}[h_i t_j h'_l t'_m] - \mathbb{E}_{p_+}[h'_i t'_j h_l t_m] + \mathbb{E}_{p_+}[h'_i t'_j h'_l t'_m]) \end{aligned} \tag{11}$$

Lets first analyse the cases where $i \neq j$ and $l \neq m$. Because of the relational independence assumption, the second and the third expectations in (11) can be written as follows: $\mathbb{E}_{p_+}[h_i t_j]\mathbb{E}_{p_+}[h'_l t'_m]$ and $\mathbb{E}_{p_+}[h'_i t'_j]\mathbb{E}_{p_+}[h_l t_m]$, respectively. Each of the these expectations contains the product of different dimensionalities in two different words. Expected correlation of different dimensions in the same word is zero from (8). Therefore, such cross-correlations are likely to be small for different word pairs, which are nonsynomyous. On the other hand, first and fourth expectations in (11) involve the same pair of words. For example, we could write the first expectation as follows:

$$\mathbb{E}_{p_+}[h_i t_j h_l t_m] = \mathbb{E}_{p_+}[(h_i h_l)(t_j t_m) = \mathbb{E}_{p_+}[H_{il} T_{jm}]$$

where $H = h_i h_l$ and $T_{jm} = t_j t_m$. If we think of $\boldsymbol{H}$ and $\boldsymbol{T}$ as d^2-dimensional word embeddings, $\mathbb{E}_{p_+}[H_{il} T_{jm}]$ represents the expectation over two distinct dimensions of $\boldsymbol{H}$ and $\boldsymbol{T}$ for $il \neq jm$. Therefore, from the same logic as above, this expectation is approximately zero.[4]

For $i = j = l = m$ case we have,

$$A^{(k)2}_{ij} (\mathbb{E}_{p_+}[h_i^2 t_i^2] - 2\mathbb{E}_{p_+}[h_i t_i h'_i t'_i] + \mathbb{E}_{p_+}[h'^2_i t'^2_i]) \tag{12}$$

[4]Note that il could be equal to jm even when $i \neq j$ and $l \neq m$. However, such cases will be a rare minority. Nevertheless, it is an approximation and not an exact zero.

Because we are considering word-pairs (h, t) for which a relation r is known to hold, from the definition of the correlation between the same dimension in different words we have:

$$\text{corr}(h_i^2, t_i^2) = \mathbb{E}[h_i^2 t_i^2] - \mathbb{E}[h_i^2]\mathbb{E}[t_i^2] = 1$$
$$\mathbb{E}[h_i^2 t_i^2] = \mathbb{E}[h_i^2]\mathbb{E}[t_i^2] + 1$$

Because $\mathbb{E}[h_i^2] = \mathbb{E}[t_i^2] = 1$ from the standardisation, we get:

$$\mathbb{E}[h_i^2 t_i^2] = 2 \tag{13}$$

Lets analyse the second term in (12). From the relational independence and because the word embeddings are assumed to be standardised to unit variance, we obtain the follows:

$$2\mathbb{E}_{p_+}[h_i t_i h_i' t_i'] = 2\mathbb{E}_{p_+}[h_i t_i]\mathbb{E}_{p_+}[h_i' t_i'] = 2. \tag{14}$$

According to (13) and (14), (12) evaluates to $2{A_{ij}^{(k)}}^2$. We will then get the same term from the negative expectations and they would cancel out as $2{A_{ij}^{(k)}}^2$ is independent of the training dataset.

Next, lets consider the $A_{ij}^{(k)} P_{kn}$ terms in the expansion of (10) given by,

$$2\sum_{i,j}\sum_{n} A_{ij}^{(k)} P_{kn}(h_i t_j - h_i' t_j')(h_n - h_n'). \tag{15}$$

Taking the expectation of (15) w.r.t. p_+ we get,

$$2\sum_{i,j}\sum_{n} A_{ij}^{(k)} P_{kn}(\mathbb{E}_{p_+}[h_i t_j h_n] - \mathbb{E}_{p_+}[h_i t_j h_n'] - \mathbb{E}_{p_+}[h_i' t_j' h_n] + \mathbb{E}_{p_+}[h_i' t_j' h_n']). \tag{16}$$

Likewise, from the uncorrelation assumption and following the same logic as above it follows that all the expectations in (16) are approximately zero. A similar argument can be used to show that terms that involve $A_{ij}^{(k)} Q_{kn}$ disappear from (10). Therefore, $\underline{\mathbf{A}}$ does not play any part in the expected loss over positive examples. Similarly, we can show that $\underline{\mathbf{A}}$ is independent of the expected loss over negative examples. Therefore, from (5) we see that the expected loss over the entire training dataset is independent of $\underline{\mathbf{A}}$. □

3.1 Regularised ℓ_2 loss

As a special case, if we attempt to minimise the expected loss under some regularisation on $\underline{\mathbf{A}}$ such as the Frobenius norm regularisation, then this can be achieved by sending $\underline{\mathbf{A}}$ to zero tensor because according to 1 2 is independent from $\underline{\mathbf{A}}$.

With $\underline{\mathbf{A}} = \underline{\mathbf{0}}$, the relation between h and t can be simplified to:

$$\boldsymbol{r}(\boldsymbol{h}, \boldsymbol{t}) = \mathbf{P}\boldsymbol{h} + \mathbf{Q}\boldsymbol{t} \tag{17}$$

Then the expected loss over the positive instances is given by (18).

$$\begin{aligned}
&\mathbb{E}_{p_+}[\left|\left|\mathbf{P}(\boldsymbol{h} - \boldsymbol{h'}) + \mathbf{Q}(\boldsymbol{t} - \boldsymbol{t'})\right|\right|_2^2] \\
&= \mathbb{E}_{p_+}[(\boldsymbol{h} - \boldsymbol{h'})^\top \mathbf{P}^\top \mathbf{P}(\boldsymbol{h} - \boldsymbol{h'})] + \mathbb{E}_{p_+}[(\boldsymbol{h} - \boldsymbol{h'})^\top \mathbf{P}^\top \mathbf{Q}(\boldsymbol{t} - \boldsymbol{t'})] + \\
&\mathbb{E}_{p_+}[(\boldsymbol{t} - \boldsymbol{t'})^\top \mathbf{Q}^\top \mathbf{P}(\boldsymbol{h} - \boldsymbol{h'})] + \mathbb{E}_{p_+}[(\boldsymbol{t} - \boldsymbol{t'})^\top \mathbf{Q}^\top \mathbf{Q}(\boldsymbol{t} - \boldsymbol{t'})]
\end{aligned} \tag{18}$$

The second expectation term in the right hand side of (18) can be computed as follows:

$$\begin{aligned}
&\mathbb{E}_{p_+}[(\boldsymbol{h} - \boldsymbol{h'})^\top \mathbf{P}^\top \mathbf{Q}(\boldsymbol{t} - \boldsymbol{t'})] \\
&= \sum_{i,j} (\mathbf{P}^\top \mathbf{Q})_{ij} \mathbb{E}_{p_+}[(h_i - h_i')(t_j - t_j')] \\
&= \sum_{i,j} (\mathbf{P}^\top \mathbf{Q})_{ij} \left(\mathbb{E}_{p_+}[h_i t_j] - \mathbb{E}_{p_+}[h_i t_j'] - \mathbb{E}_{p_+}[h_i' t_j] + \mathbb{E}_{p_+}[h_i' t_j']\right)
\end{aligned} \tag{19}$$

When $i \neq j$, each of the four expectations in the RHS of (21) are zero from the uncorrelation assumption. When $i = j$, each term will be equal to one from the standardisation assumption (unit variance) and cancel each other out. A similar argument can be used to show that the third expectation term in the RHS of (18) vanishes.

Now lets consider the first expectation term in the RHS of (18), which can be computed as follows:

$$\begin{aligned}
&\mathbb{E}_{p_+}[(\boldsymbol{h}-\boldsymbol{h'})^\top \mathbf{P}^\top \mathbf{P}(\boldsymbol{h}-\boldsymbol{h'})] \\
&= \sum_{i,j} (\mathbf{P}^\top \mathbf{P})_{ij} \mathbb{E}_{p_+}[(h_i - h'_i)(h_j - h'_j)] \\
&= \sum_{i,j} (\mathbf{P}^\top \mathbf{P})_{ij} (\mathbb{E}_{p_+}[h_i h_j] - \mathbb{E}_{p_+}[h_i h'_j] - \mathbb{E}_{p_+}[h'_i h_j] + \mathbb{E}_{p_+}[h'_i h'_j])
\end{aligned} \tag{20}$$

When $i \neq j$, it follows from the uncorrelation assumption that each of the four expectation terms in the RHS of (20) will be zero. For $i = j$ case we have,

$$\begin{aligned}
&\sum_{i,j} (\mathbf{P}^\top \mathbf{P})_{ii} \left(\mathbb{E}_{p_+}[h_i^2] - 2\mathbb{E}_{p_+}[h_i h'_i] + \mathbb{E}_{p_+}[h_i'^2] \right) \\
&= 2 \sum_{i,j} (\mathbf{P}^\top \mathbf{P})_{ii}
\end{aligned} \tag{21}$$

Note that from the relational independence between h and h' we have $\mathbb{E}_{p_+}[h_i h'_i] = \mathbb{E}_{p_+}[h_i]\mathbb{E}_{p_+}[h'_i]$. From the standardisation (zero mean) assumption this term is zero. On the other hand $\mathbb{E}_{p_+}[h_i^2] = \mathbb{E}_{p_+}[h_i'^2] = 1$ from the standardisation (unit variance) assumption, which gives the result in (21).

Similarly, the fourth expectation term in the RHS of (18) evaluates to $2\sum_{i,j} (\mathbf{Q}^\top \mathbf{Q})_{ii}$, which shows that (18) evaluates to $2\sum_{i,j} \left((\mathbf{P}^\top \mathbf{P})_{ii} + (\mathbf{Q}^\top \mathbf{Q})_{ii} \right)$. Note that this is independent of the positive instances and will be equal to the expected loss over negative instances, which gives $\mathbb{E}_p[J] = 0$ for the relational embedding given by (17).

It is interesting to note that PairDiff is a special case of (17), where $\mathbf{P} = \mathbf{I}$ and $\mathbf{Q} = -\mathbf{I}$. In the general case where word embeddings are nonstandardised to unit variance, we can set $\mathbf{P}$ to be the diagonal matrix where $\mathbf{P}_{ii} = 1/\sigma_i$, where σ_i is the variance of the i-th dimension of the word embedding space, to enforce standardisation. Considering that $\mathbf{P}, \mathbf{Q}$ are parameters of the relational embedding, this is analogous to *batch normalisation* (Ioffe and Szegedy, 2015), where the appropriate parameters for the normalisation are learnt during training.

4 Experimental Results

4.1 Cross-dimensional Correlations

A key assumption in our theoretical analysis is the uncorrelations between different dimensions in word embeddings. Here, we empirically verify the uncorrelation assumption for different input word embeddings. For this purpose, we create SG, CBOW and GloVe embeddings from the ukWaC corpus[5]. We use a context window of 5 tokens and select words that occur at least 6 times in the corpus. We use the publicly available implementations for those methods by the original authors and set the parameters to the recommended values in (Levy et al., 2015) to create 50-dimensional word embeddings. As a representative of counting-based word embeddings, we create a word co-occurrence matrix weighted by the positive pointwise mutual information (PPMI) and apply singular value decomposition (SVD) to obtain 50-dimensional embeddings, which we refer to as the Latent Semantic Analysis (LSA) embeddings.

We use Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to create a topic model, and represent each word by its distribution over the set of topics. Ideally, each topic will capture some semantic category and the topic distribution provides a semantic representation for a word. We use gensim[6] to extract 50 topics from a 2017 January dump of English Wikipedia. In contrast to the above-mentioned word embeddings,

[5] http://wacky.sslmit.unibo.it/doku.php?id=corpora

[6] https://radimrehurek.com/gensim/wiki.html

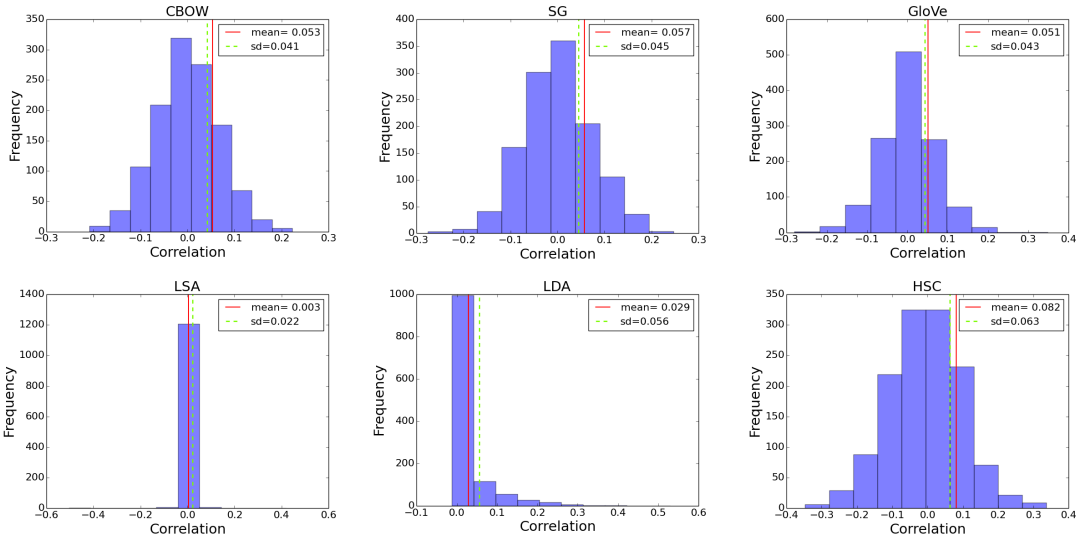


Figure 1: Cross-dimensional correlations for six word embeddings

which are dense and flat structured, we used Hierarchical Sparse Coding[7] (HSC) (Yogatama et al., 2015) to produce sparse and hierarchical word embeddings.

Given a word embedding matrix $\mathbf{W} \in \mathbb{R}^{m \times d}$, where each row correspond to the d-dimensional embedding of a word in a vocabulary containing m words, we compute a correlation matrix $\mathbf{C} \in \mathbb{R}^{d \times d}$, where the (i, j) element, C_{ij}, denotes the Pearson correlation coefficient between the i-th and j-th dimensions in the word embeddings over the m words. By construction $C_{ii} = 1$ and the histograms of the cross-dimensional correlations ($i \neq j$) are shown in Figure 1 for 50 dimensional word embeddings obtained from the six methods described above. The mean of the absolute pairwise correlations for each embedding type and the standard deviation (sd) are indicated in the figure.

From Figure 1, irrespective of the word embedding learning method used, we see that cross-dimensional correlations are distributed in a narrow range with an almost zero mean. This result empirically validates the uncorrelation assumption we used in our theoretical analysis. Moreover, this result indicates that Theorem 1 can be applied to a wide-range of existing word embeddings.

4.2 Learning Relation Representations

Our theoretical analysis in §3 claims that the performance of the bilinear relational embedding is independent of the tensor operator $\underline{\mathbf{A}}$. To empirically verify this claim, we conduct the following experiment. For this purpose, we use the BATS dataset (Gladkova et al., 2016) that contains of 40 semantic and syntactic relation types[8], and generate positive examples by pairing word-pairs that have the same relation types. Approximately each relation type has 1,225 word-pairs, which enables us to generate a total of 48k positive training instances (analogous word-pairs) of the form $((h, t), (h', t'))$. For each pair (h, t) related by a relation r, we randomly select pairs (h', t') with a different relation type r', according to the ℓ_2 distance between the two pairs to create negative (nonanalogous) instances.[9] We collectively refer both positive and negative training instances as the *training* dataset.

Using the $d = 50$ dimensional word embeddings from CBOW, SG, GloVe, LSA, LDA, and HSC methods created in §4.1, we learn relational embeddings according to (2) by minimising the ℓ_2 loss, (4). To avoid overfitting, we perform ℓ_2 regularisation on $\underline{\mathbf{A}}$, $\mathbf{P}$ and $\mathbf{Q}$ are regularised to diagonal matrices $p\mathbf{I}$ and $q\mathbf{I}$, for $p, q \in \mathbb{R}$. We initialise all parameters by uniformly sampling from $[-1, +1]$ and use AdaGrad (Duchi et al., 2011) with initial learning rate set to 0.01.

Figure 2 shows the Frobenius norm of the tensor $\underline{\mathbf{A}}$ (on the left vertical axis) and the values of p

[7] http://www.cs.cmu.edu/~ark/dyogatam/wordvecs/

[8] http://vsm.blackbird.pw/bats

[9] 10 negative instances are generated from each word-pair in our experiments.

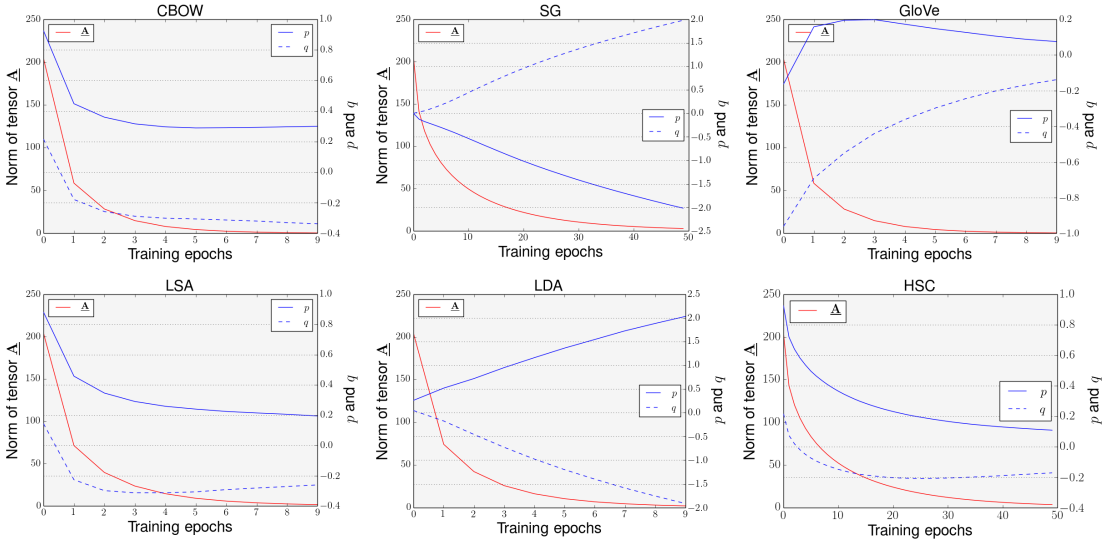


Figure 2: The learnt model parameters for different word embeddings of 50 dimensions.

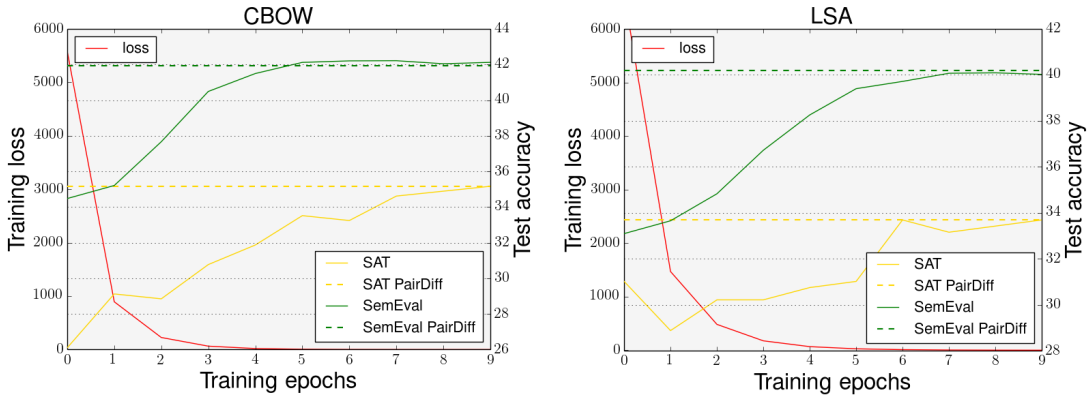


Figure 3: The training loss and test performance on SAT and SemEval benchmarks for relational embeddings.

and q (on the right vertical axis) for the six word embeddings. In all cases, we see that as the training progresses, $\underline{\mathbf{A}}$ goes to zero as predicted by Theorem 1 under regularisation. Moreover, we see that approximately $p \approx -q = c$ is reached for some $c \in \mathbb{R}$ in all cases, which implies that $\mathbf{P} \approx -\mathbf{Q} = c\mathbf{I}$, which is the PairDiff operator. Among the six input word embeddings compared in Figure 1, HSC has the highest mean correlation (0.082), which implies that its dimensions are correlated more than in the other word embeddings. This is to be expected by design because a hierarchical structure is imposed on the dimensions of the word embedding during training. However, HSC embeddings also satisfy the $\underline{\mathbf{A}} \approx \underline{\mathbf{0}}$ and $p \approx -q = c$ requirements, as expected by the PairDiff. This result shows that the claim of Theorem 1 is empirically true even when the uncorrelation assumption is mildly violated.

4.3 Generalisation Performance on Analogy Detection

So far we have seen that the bilinear relational representation given by (2) does indeed converge to the form predicted by our theoretical analysis for different types of word embeddings. However, it remains unclear whether the parameters learnt from the training instances generated from the BATS dataset accurately generalise to other benchmark datasets for analogy detection. To emphasize, our focus here is not to outperform relational representation methods proposed in previous works, but rather to empirically show that the learnt operator converges to the popular PairDiff for the analogy detection task.

To measure the generalisation capability of the learnt relational embeddings from BATS, we measure their performance on two other benchmark datasets: SAT Turney and Bigham (2003) and SemEval 2012-Task2[10]. Note that we *do not* retrain $\underline{\mathbf{A}}$, $\mathbf{P}$ and $\mathbf{Q}$ in (2) on SAT nor SemEval, but simply to use their values learnt from BATS because the purpose here to evaluate the generalisation of the learnt operator.

In SAT analogical questions, given a stem word-pair (a, b) with five candidate word-pairs (c, d), the task is to select the word-pair that is relationally similar to the the stem word-pair. The relational similarity between two word-pairs (a, b) and (c, d) is computed by the cosine similarity between the corresponding relational embeddings $\boldsymbol{r}(\boldsymbol{a}, \boldsymbol{b})$ and $\boldsymbol{r}(\boldsymbol{c}, \boldsymbol{d})$. The candidate word-pair that has the highest relational similarity with the stem word-pair is selected as the correct answer to a word analogy question. The reported accuracy is the ratio of the correctly answered questions to the total number of questions. On the other hand, SemEval dataset has 79 semantic relations, with each relation having ca. 41 word-pairs and four prototypical examples. The task is to assign a score for each word pair which is the average of the relational similarity between the given word-pair and prototypical word-pairs in a relation. Maximum difference scaling (MaxDiff) is used as the evaluation measure in this task.

Figure 3 shows the performance of the relational embeddings composed from 50-dimensional CBOW and LSA embeddings[11]. For CBOW, the level of performance reported by PairDiff on SAT and SemEval datasets are respectively 35.16% and 41.94%, and are shown by horizontal dashed lines. From Figure 3, we see that the training loss gradually decreases with the number of training epochs and the performance of the relational embeddings on SAT and SemEval datasets reach that of the PairDiff operator. This result indicates that the relational embeddings learnt not only converge to PairDiff operator on training data but also generalise to unseen relation types in SAT and SemEval test datasets.

5 Conclusion

This paper theoretically analyses the bilinear operator for representing relations between words using their embeddings. We showed that, if the word embeddings are standardised and uncorrelated, then the expected ℓ_2 distance between analogous and non-analogous word-pairs is independent of bilinear terms, and the relation embedding further simplifies to the popular PairDiff operator under regularised settings. Among diverse methods for calculating word embeddings, we empirically show the uncorrelation in word embedding dimensions, which is one of the prerequisites for simplifying the bilinear operator to a linear one. Empirically, we supports the theoretical analysis by showing that when optimising a general bilinear formulation on a labeled word pair relational dataset, the solution converges to the simple linear form, and more specifically to the simple PairDiff formulation.

In this work, we model relations as vectors and we measure relational strength using Euclidean distance. We are aware that there are many other relation representation methods and relational strength measurement methods besides what we have considered in the paper. Similar analysis can be conducted in follow-up work for different types of relation representations and strength measures. For instance, an interesting future research direction of this work is to extend the theoretical analysis to nonlinear relation composition operators, such as for nonlinear neural networks.

Acknowledgement

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

References

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proc. of EMNLP*. pages 546–556.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

[10] https://sites.google.com/site/semeval2012task2/

[11] Similar trends were observed for all six word embedding types.

Danushka Bollegala, Takanori Maehara, and Ken ichi Kawarabayashi. 2015a. Embedding semantic relationas into word representations. In *Proc. of IJCAI*. pages 1222 – 1228.

Danushka Bollegala, Takanori Maehara, Yuichi Yoshida, and Ken ichi Kawarabayashi. 2015b. Learning word representations from relational graphs. In *Proc. of AAAI*. pages 2146 – 2152.

Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. 2009. A relational model of semantic similarity between words using automatically extracted lexical pattern clusters from the web. In *Proc. of EMNLP*. pages 803–812.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhenko. 2013. Translating embeddings for modeling multi-relational data. In *Proc. of NIPS*. pages 2787–2795.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *American Society for Information Science* 41(6):391–407.

Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. 2016. Word embeddings, analogies, and machine learning: Beyond king - man + woman = queen. In *Proc. of COLING*. pages 3519–3530.

Nguyen Tuan Duc, Danushka Bollegala, and Mitsuru Ishizuka. 2010. Using relational similarity between word pairs for latent relational search on the web. In *Proc. of WI*. pages 196–199.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12:2121–2159.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015. Sparse overcomplete word vector representations. In *Proc. of ACL*. pages 1491–1500.

Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. 2016. Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *Proc. of SRW@ HLT-NAACL*. pages 8–15.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2016. Jointly embedding knowledge graphs and logical rules. In *Proc. of EMNLP*. pages 192–202.

Huda Hakami and Danushka Bollegala. 2017. Compositional approaches for representing relations between words: A comparative study. *Knowledge-Based Systems* 136:172–182.

Katsuhiko Hayashi and Masashi Shinbo. 2017. On the equivalence of holograpic and complex embeddings for link prediction. In *Proc. of ACL*. pages 554–559.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. of Machine Learning Research*. pages 448–456.

David A. Jurgens, Saif Mohammad, Peter D. Turney, and Keith J. Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proc. of *SEM*. pages 356 – 364.

Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *Proc. of CoNLL*. pages 171–180.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of Association for Computational Linguistics* 3:211–225.

Tomas Mikolov, Kai Chen, and Jeffrey Dean. 2013a. Efficient estimation of word representation in vector space. In *Proc. of ICLR*. pages 1–12.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proc. of HLT-NAACL*. pages 746–751.

Pasquale Minervini, Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. 2017. Adversarial sets for regularising neural link predictors. In *Proc. of UAI*.

J. Mu, S. Bhat, and P. Viswanath. 2018. All-but-the-Top: Simple and Effective Postprocessing for Word Representations. In *Proc. of ICLR*.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A review of relational machine learning for knowledge graphs. In *Proc. of the IEEE*. 1, pages 11–33.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proc. of AAAI*. pages 1955–1961.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proc. of ICML*. pages 809–816.

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

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proc. of ICLR*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proc. of NIPS*. pages 926–934.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proc. of ICML*. pages 2071–2080.

Peter D Turney and Jeffrey Bigham. 2003. Combining independent modules to solve multiple-choice synonym and analogy. In *Proc. of RANLP*. pages 482–489.

Peter D Turney and Michael L Littman. 2005. Corpus-based learning of analogies and semantic relations. *Machine Learning* 60(1):251–278.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relational learning. In *Proc. of ACL*. pages 1671–1682.

Bishan Yang, Wen tau Yih, Xiadong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proc. of ICLR*.

Wenpeng Yin and Hinrich Schütze. 2016. Learning meta-embeddings by using ensembles of embedding sets. In *Proc. of ACL*. pages 1351–1360.

Dani Yogatama, Manaal Faruqui, Chris Dyer, and Noah A. Smith. 2015. Learning word representations with hierarchical sparse coding. In *Proc. of ICML*. pages 87 – 96.

Real-time Change Point Detection using On-line Topic Models

Yunli Wang
Scientific Data Mining
National Research Council Canada
1200 Montreal Rd., Ottawa, ON
Yunli.Wang@nrc.ca

Cyril Goutte
Multilingual Text Processing
National Research Council Canada
1200 Montreal Rd., Ottawa, ON
Cyril.Goutte@gmail.com

Abstract

Detecting changes within an unfolding event in real time from news articles or social media enables to react promptly to serious issues in public safety, public health or natural disasters. In this study, we use on-line Latent Dirichlet Allocation (LDA) to model shifts in topics, and apply on-line change point detection (CPD) algorithms to detect when significant changes happen. We describe an on-line Bayesian change point detection algorithm that we use to detect topic changes from on-line LDA output. Extensive experiments on social media data and news articles show the benefits of on-line LDA versus standard LDA, and of on-line change point detection compared to off-line algorithms. This yields F-scores up to 56% on the detection of significant real-life changes from these document streams.

1 Introduction

In recent years, the detection of emerging events from publicly available data streams such as twitter messages has received a lot of attention. The approaches used range from topic modeling, incremental clustering, the concept of interestingness, and others (Hasan et al., 2017). For example, methods relying on topic models detect latent topics from tweets and use that semantic structure to guide the event detection task. In particular, Latent Dirichlet Allocation, *aka* LDA (Blei et al., 2003), and extensions have been widely used to model topics from large corpora. It has been used for event detection (Pozdnoukhov and Kaiser, 2011; Ertl et al., 2012; Vavliakis et al., 2012; Lau et al., 2012; Zhou and Chen, 2014), summarization, or finding influential users in social media. However, only few studies used LDA to detect topic changes over time (Lau et al., 2012; Zhou and Chen, 2014).

Once a main event is identified, detecting key sub-events is an essential task. For example, during a public health epidemic crisis, detecting turning points in the spread of the disease is extremely important. Relatively little attention has been paid on detecting these change points during an event. For example, Bruggermann et al. (2016) used the dynamic topic model, *aka* DTM (Blei and Lafferty, 2006), to detect and track stories in news articles. In addition, although many studies use topic models for event detection, very few focus on real-time or sub-event detection.

In statistics, change point detection (CPD) is the task of finding locations where the underlying stochastic process governing time series changes. Change-point detection algorithms can be split into two categories: Real-time (or on-line) detection and retrospective (or off-line) detection, depending on how data is used. Most CPD algorithms are retrospective (Barry and Hartigan, 1993; James and Matteson, 2015): the main drawback is that they can't be run before all the data has been acquired, which is a significant operational constraint when monitoring unfolding safety or health crises. Few algorithms address real-time detection (Adams and MacKay, 2007).

Our study focuses on real-time change point detection in document streams by combining on-line LDA with on-line CPD. Most previous studies on event detection used off-line topic models (Ertl et al., 2012; Vavliakis et al., 2012), which is not appropriate for document streams. In addition, many studies using LDA or on-line LDA for event detection only tested on several case studies (Pozdnoukhov and

Proceedings of the 27th International Conference on Computational Linguistics, pages 2505–2515
Santa Fe, New Mexico, USA, August 20-26, 2018.

Kaiser, 2011; Ertl et al., 2012): there were no reference events to test against. By contrast, we collected document streams from social media and news articles, and gathered reference change points for each dataset, in order to evaluate the performance on these datasets.

In the following section, we review some related work in order to position ours. In Section 3 we describe the methods: topic modeling, CPD and their combination. Sections 4 and 5 report on our experiments and results, before we discuss our findings (Section 6).

2 Related Work

LDA is a three-level hierarchical Bayesian model (Blei et al., 2003) where each document is a multinomial distribution over topics, and each topic is a multinomial distribution over the vocabulary. The Dynamic Topic Model (Blei and Lafferty, 2006) is an extension of LDA that captures the evolution of topics in a sequentially organized corpus of documents. In order to process document streams, several on-line LDA (oLDA) methods have also been proposed (Hoffman et al., 2010; Lau et al., 2012; Zhai and Boyd-Graber, 2013).

These models have been used for event detection from social media data. Pozdnoukhov and Kaiser (2011) used LDA to identify topics and classified tweets according to the most probable topic. Ertl et al. (2012) combined LDA with Seasonal Trend Decomposition in order to detect and rank topics, remove daily chatter and detect abnormal topics. Vavliakis et al. (2012) used LDA for topic identification and event detection in the MediaEval-2012 Social Event Detection task. Zhou and Chen (2014) proposed a location-time constrained topic model to represent social data information over content, time, and location. By considering the time and location of messages as additional variables, it outperformed oLDA on the tested datasets. In addition to using spatial information on top of LDA, none of the previous studies consider real-time event detection.

Other recent work attempt to build storylines from either news articles or social media data. Bruggermann et al. (2016) use the word-topic distribution from the DTM model to represent the changes during events, but did not evaluate the performance of the DTM model on topic identification, or whether the turning points from the word-topic distribution discover actual sub-events. Wang and Goutte (2017) evaluated the performance of change point detection algorithms using the temporal profiles of hashtags and frequency of tweets on two twitter data sets. Their work is an early attempt to use CPD algorithms on the content of document streams. Previous studies (Guralnik and Srivastava, 1999; James et al., 2016) detected significant changes from sensor signals but do not use the textual content of message streams. Recently, Goutte et al. (2018) used a similar on-line CPD approach, but applied it on time series recording the sentiment polarity in streams of tweets.

3 Methods

In this section, we describe on-line topic modeling, several competing ways to produce time series from the topic models, and several change point detection algorithms.

3.1 On-line Topic Model

We use both standard LDA and an on-line version of LDA with infinite vocabulary, oLDA$^\infty$. The basic LDA generates documents from a distribution over topics, each topic having a distribution over words. For D documents over K topics, the generative process is as follows (Blei et al., 2003):

1. For each doc $d=1\ldots D$, pick $\theta_d \sim \mathsf{Dir}(\alpha)$,
2. For each topic $k=1\ldots K$, pick $\phi_k \sim \mathsf{Dir}(\beta)$,
3. For each word w_n, $n=1\ldots N_d$, in document d (of size N_d),
   (a) Pick topic $z \sim \mathsf{Mult}(\theta_d)$,
   (b) Pick word $w_n \sim \mathsf{Mult}(\phi_z)$.

$\mathsf{Dir}()$ and $\mathsf{Mult}()$ are the Dirichlet and Multinomial distributions, respectively; α and β are hyper-parameters, and the parameters are usually estimated using collapsed Gibbs sampling. LDA inference

yields document topic distributions θ_d and word-topic probabilities ϕ_k. Both can be used for further processing, in particular θ_d encodes the topics represented in d.

One limitation of LDA and early on-line variants is that words are sampled over a fixed vocabulary. This makes sense in a batch setting, where all words are known from the collection, but not in a streaming or on-line setting, where new words may appear as documents are acquired. The on-line LDA with infinite vocabulary (oLDA$^\infty$) introduced by Zhai and Boyd-Graber (2013) addresses this by 1) processing data and doing inference in minibatches, and 2) instead of a Dirichlet distribution over a fixed vocabulary, topic distributions ϕ_k are drawn from a Dirichlet Process with a base distribution over *all* words. Dirichlet processes are a common tool from non-parametric statistics, allowing sampling and inference in finite time over unbounded and possibly countably infinite support. Details of the sampling and inference are provided by Zhai and Boyd-Graber (2013).

The original implementation of oLDA$^\infty$ updates the model for each minibatch containing a fixed *number of documents*. This does not work well in our setting where news articles or social media messages do not arrive regularly, and we want to update the model for fixed *time* increments (e.g. a few minutes or a few days). We therefore adapted oLDA$^\infty$ to process the collection in fixed-time minibatches containing variable numbers of documents according to their time stamps. We denote this variant by oLDA$^\infty_t$. Documents within the same time interval are placed in the same mini batch, and the model is updated at each time slot $t_1, t_2, \ldots t_m$. Inference yields corresponding word-topic distributions $\phi(1), \phi(2), \ldots \phi(m)$.

3.2 From Topic Models to Topic Change

The evolution of topics in on-line LDA models is usually shown using the most probable words from the word-topic distributions (Hoffman et al., 2010; Lau et al., 2012; Zhai and Boyd-Graber, 2013). However, the same top words can appear in different topics, making differences between topics hard to show. This is in particular the case when documents are related to a particular event, as words typical of that event are common in all topics. Other approaches rely on the top words in different ways: the cosine similarity of top words from the DTM Bruggermann et al. (2016), the Jensen-Shannon divergence (JSD) Lau et al. (2012), and the symmetrized Kullback-Leibler divergence of word distribution Zhou and Chen (2014) to represent topic change over time.

We propose to evaluate topic change by using the word-topic distribution $\phi(t)$ and the document-topic distribution θ_n at each time slot. Four methods were tested in our study to represent the topic change over time: cosine distance, Jensen-Shannon divergence, the word-topic distribution (WD), and the document-topic distribution (DD). For cosine distance, we used the cosine similarity between the top-word probabilities of all topics at the current time t and previous time $t-1$. From the word-topic distributions $\phi_k(t)$ and $\phi_k(t-1)$, we extract the top-word probabilities $P_W(t)$ and $P_W(t-1)$ and compute:

$$D_{\cos}(t) = 1 - \cos(P_W(t), P_W(t-1)), \tag{1}$$

$$\cos(P_W(t), P_W(t-1)) = \frac{\langle P_W(t), P_W(t-1)\rangle}{\|P_W(t)\|\|P_W(t-1)\|}$$

$D_{\cos}(t)$ is a univariate time series with $m-1$ points, which we use to detect significant changes using the CPD algorithms below.

The second method uses the sum of JSD between the top words in all topics between t and $t-1$. Normalizing the top-word distributions into $\overline{P}_W(t)$ and $\overline{P}_W(t-1)$:

$$D_{\text{JSD}}(t) = \text{JSD}(\overline{P}_W(t), \overline{P}_W(t-1)), \tag{2}$$

$$\text{JSD}(P,Q) = \frac{1}{2}\sum_i p_i \log\frac{2p_i}{p_i+q_i} + \frac{1}{2}\sum_i q_i \log\frac{2q_i}{p_i+q_i}$$

$D_{\text{JSD}}(t)$ again forms a univariate time series with $m-1$ data points.

The third method directly uses the word-topic distribution of top words. For each topic k, the probability of word w from the top L words is obtained from $\phi_k(t)$, $t = 1 \ldots m$. This produces a total of $K \times L$ time series representing evolutions in the word-topic distributions.

The last method uses the average document-topic distribution, for documents in each time interval. In the document-topic distribution matrix θ , each element θ_{kj} represents the probability of topic k in document j. Averaging over all documents from a minibatch t, we compute the average document-topic probability $\pi_k(t) = \frac{1}{|t|} \sum_{d_j \in t} \theta_{kj}$. This produces K times series, with m data points each, on which we run multivariate CPD.

3.3 Change Point Detection

In time series analysis, a change point is a location where the underlying stochastic process changes. Although it may seem superficially related, this is a different problem than anomaly detection, where the purpose is to identify observations that do not conform to an expected pattern or distribution in the data. In change point detection, we assume that data before the change point conforms to one distribution, while data after the change point comes from a second, different distribution. In our case, we are interested in identifying where the change has occurred, as soon as possible after it occurs.

Many algorithms have been proposed to detect change points. Most work on univariate time series, and use the entire time series to detect change points. We will focus on techniques that can be used with multivariate time series:

`bcp`

The Bayesian change point detection of Barry and Hartigan (1993) assumes that each block between two change points arises from a (multivariate) normal distribution. It outputs the posterior probability that a change occurred at each point in the time series. We use the implementation from the `R` package `bcp` (Erdman and Emerson, 2007), which runs in time linear in the length of the time series, and handles multivariate time series. The biggest limitations are that it is designed to detect changes in the mean of independent Gaussian observations, and that it works retrospectively, once the entire time series is available.

`ecp`

The nonparametric, hierarchical divisive algorithm of James and Matteson (2015) uses recursive bisections, identifying change points using a non-parametric divergence measure from Székely and Rizzo (2005). As the divergence measure is non-parametric, this makes `ecp` suitable to detect changes with minimal assumptions on the underlying distributions. The divisive approach by recursive bisections returns a number of consecutive *segments* between change points, without knowing the number of change points *a priori*. In addition, the implementation from the `R` package `ecp` handles multivariate time series. One remaining limitation is that it works only in retrospective mode, once the entire time series is available.

`ocp`

The Bayesian online change point detection algorithm of Adams and MacKay (2007) addresses that issue. It is designed to update the detection of change points sequentially, as new data points are acquired, rather than wait until the entire times series are available. It relies on two components: a probabilistic model $P(r_t|\mathbf{s}(1 \dots t))$ of the length of a *run* during which the underlying distribution is stable, given observations until time t; and an underlying predictive model (UPM) $P(\mathbf{s}(t+1)|\mathbf{s}(1 \dots t), r_t)$ governing the stochastic generation of new data in each run. Our basic implementation, available in the `R` package `onlineCPD`,[1] uses a multivariate Gaussian UPM.

`ocp+`

We extend the `ocp` algorithm beyond the Gaussian assumption by using a more flexible UPM, modeling linear trends within each run, using a multivariate linear regression with additive Gaussian noise. This allows modeling drifts in the time series without forcing multiple change points. Our implementation is also available in the `onlineCPD` `R` package available from github.

[1] Available at `https://github.com/cyrilgoutte/EuroGames16`

Table 1: Statistics on our two benchmark datasets.

Dataset	#Docs	#Ref timeline	Time interval	Time window
Olympics	30,115,218	89	1 hour	2 hours
Zika	31,356	31	1 day	2 days

4 Experiments

4.1 Datasets

To test the performance of our various combinations of online LDA and online CPD, we collected two datasets. One large collection of tweets related to the 2014 Sochi Winter Olympics, and a collection of public health news articles related to the 2015-2016 Zika epidemics. Statistics are shown in Table 1.

The 2014 Sochi Winter Olympics Twitter dataset was collected from the Twitter API during the whole olympic games period (Feb. 6-24th, 2014). Keywords such as "Sochi", "Olympics" were used to filter tweets. The dataset contains 3,115,218 tweets after removing re-tweets and non-English tweets. The reference events were collected from Wikipedia,[2] considering finals for most event, plus quarterfinals, semifinals, bronze and gold medal games for ice hockey, which adds up to 89 reference change points.

During the outbreak of Zika virus from 2015 to 2016, we collected news articles from November 3rd 2015 to December 31st 2016 on the GPHIN system.[3] News articles from a variety of sources were collected and translated into English, named entities and medical terms were identified, then keywords like "Zika" and "mysterious disease" were used to filter news articles (Carter, 2018). The final, filtered Zika dataset contains 31,356 news articles. The original reference events were collected from a web source.[4] Events on the same or subsequent days were combined into one, and the final gold standard contains 31 reference change points.

4.2 Evaluation

The performances of change point detection methods were measured by comparing detected change points with known reference events, using precision, recall and F-score. We count a detection as a true positive if it falls into the tolerance time window (Table 1, last column) of the reference change point, based on the order of references. The precision is computed by dividing true positives by all detections, while recall is computed by dividing true positives by total references, and the F-score is:

$$F = 2 \times \frac{\mathsf{Precision} \times \mathsf{Recall}}{\mathsf{Precision} + \mathsf{Recall}}$$

4.3 Experimental settings

In the experiments, we compare two types of LDA models, standard (LDA) and on-line (oLDA$_t^\infty$), and two categories of CPD algorithms, retrospective or off-line CPD (`bcp` and `ecp`) and real-time or on-line CPD (`ocp` and `ocp+`). We used the implementation of standard LDA from the `R` package `text2vec` (Selivanov, 2016) and adapted oLDA$_t^\infty$ from the code made available by Zhai and Boyd-Graber (2013).[5]

5 Results

We first examine whether the top words and top documents extracted from the topic models represent topic changes over time. We then compare four methods of generating time series for CPD from the on-line topic model. We also compare the performance of CPD using the on-line topic model versus the off-line topic model and a baseline using counts. Finally, we check how various CPD algorithms work in a simulated real-time experiment, and analyze the errors made by the different methods.

[2] `https://en.wikipedia.org/wiki/2014_Winter_Olympics`
[3] `https://gphin.canada.ca/cepr/aboutgphin-rmispenbref.jsp`
[4] `http://www.mapreport.com/keyword/zika.world.da.2016.html`
[5] `https://github.com/kzhai/InfVocLDA`

Topic 1	music	song	ballet	lake	swan
#Olympics music trivia: this is Tchaikovsky's \"Swan Lake\" ballet. It's about a princess turned into a swan by a sorcerer's curse.					
I am loving the music selection. Nicely played Russia... Nicely played #sochi2014 #OpeningCeremony #Olympics2014					
I hope you are all watching the #OpeningCeremony of the #Olympics in #Sochi now. What a great Swan Lake performance. Good going #Russia					
#OpeningCeremony #Sochi2014 Was incredible! Especially loved Stravinsky's and Tchaikovsky 's music especially Swan Lake(Dracula's theme) LOL					
#Olympics: Sergei Diaghilev, founder of Ballets Russes, was mentioned during the opening ceremony. Dear Russia: he was gay. #Pride #Sochi					

Topic 13	usa	team	sweater	outfits	uniforms	wearing
#Sochi2014 #Olympics2014 #TeamUSA TEAM USA TEAM USA TEAM USA TEAN USA TEAM USA TEAM USA TEAM USA TEAM USA TEAM USA TEAM USA TEAMUSA						
@NewsBreaker Ralph Lauren says sold out of sweater Team USA wore tonight @darrenrovell #Sochi2014 Great outfits tonight. Not those sweaters!						
Well #TeamUSA won't be winning gold in uniform design. @Polo wins the ugly Christmas sweater contest hands down #Olympics						
#USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #USA #Sochi2014 #USA						
Every other country is wearing cute ski/fleece jackets. United States is wearing ugly Christmas sweaters. #Olympics						

Topic 17	ring	didn	fail	snowflake	open
Looks like NBC didn't pull a Russia TV and showed the malfunctioned 5th Olympic ring. #Sochi2014 #Olympics #OpeningCeremony					
I love that Matt Lauer was actually saying how hard they worked on this to prove the #Olympics belong in #Sochi2014 when the ring failed.					
#Sochi2014 Still feel bad for the engineer whose snowflake didn't open and ruined the @Olympics rings...so close!					
LMAO WUUUUT RT @CruzanChoklate: RT @Luvvie: One ring left behind. RT @BuzzFeedSports Olympic ring fail. #Sochi2014					
That awkward moment when snowflake doesn't work & turn into an Olympic ring. Whoops! #Sochi2014 #SochiProblems					

Figure 1: Top words (colored) and top documents for three topics from the Olympics opening ceremony: the beautiful Ballet, the strange US team uniform, and the malfunctioning 5th ring.

5.1 Validating topics from LDA and oLDA$_t^\infty$

We first run a sanity check on the topic models, with qualitative and quantitative evaluations of the resulting topics.

Figure 1 shows the top words and top documents from the LDA model of the Olympics opening ceremony, with $K = 20$ topics. Top words from topics 1, 13 and 17 are shown with colored backgrounds in the top 5 documents for each topic. Top documents (i.e. tweets, in that collection) provide more context than the top words, and clearly show that the topics are related to the ballet (topic 1), the US team uniform (topic 13) and the malfunctioning fifth ring (topic 17).

For a quantitative validation, we compare top documents extracted from the LDA and oLDA$_t^\infty$ models to the reference change points. The top documents for each topic were extracted from the portion of the document-topic distribution table corresponding to the time slot for a reference sub-event, and compared to its description. For example, at 19:00 on Feb. 22nd, the sub-event is the ice hockey bronze medal game between US and Finland. Top tweet "*Ice hockey USA vs Finland now #sochi*" in one topic is counted as a correct match, whereas "*Go Finland ice hockey team! #sochi*" is not, as it does not refer to a specific game. Whenever one top document from one topic matches with the reference, this is counted as the topic model (LDA or oLDA$_t^\infty$) correctly matching a sub-event. The rate of matching references for each model is reported in Table 2.

For the Zika dataset, we only use the titles of the five top documents from five topics. For example, on Aug. 27, 2016, a reference sub-event is Singapore's Ministry of Health confirming the first case of

Table 2: Percentage of reference sub-events matched by LDA and oLDA$_t^\infty$top documents.

Dataset	K	#Top docs	#Ref changes	LDA	oLDA$_t^\infty$
Olympics	20	5	89	70.89%	79.75%
Zika	5	5	31	54.83%	74.18%

Table 3: Experimental results on Olympics (left) and Zika (right) data. Top: performance of CPD of four methods for representing topic change: cosine distance, JSD distance, word-topic distribution and document-topic distribution. Bottom: performance of CPD on LDA and oLDA$_t^\infty$.

(Olympics)	Precision	Recall	F-score
bcp count	0.346	0.888	**0.498**
bcp cosine	0.307	0.787	**0.442**
bcp JSD	0.280	0.787	0.413
bcp WD	0.196	1.000	0.327
bcp DD	0.196	1.000	0.327
ecp count	0.483	0.326	**0.389**
ecp cosine	0.478	0.124	0.196
ecp JSD	0.433	0.146	0.218
ecp WD	0.420	0.674	**0.517**
ecp DD	0.536	0.416	0.468
ocp count	0.338	0.281	**0.307**
ocp cosine	0.478	0.371	0.418
ocp JSD	0.529	0.416	0.465
ocp WD	0.438	0.629	0.516
ocp DD	0.436	0.652	**0.523**
ocp+ count	0.371	0.292	**0.327**
ocp+ cosine	0.463	0.348	0.397
ocp+ JSD	0.500	0.404	0.447
ocp+ WD	0.421	0.629	0.505
ocp+ DD	0.615	0.449	**0.519**
LDA vs oLDA$_t^\infty$ (all DD)			
bcp LDA	0.196	1.000	0.327
bcp oLDA$_t^\infty$	0.196	1.000	0.327
ecp LDA	0.552	0.596	**0.573**
ecp oLDA$_t^\infty$	0.536	0.416	0.468
ocp LDA	0.544	0.416	0.471
ocp oLDA$_t^\infty$	0.436	0.652	**0.523**
ocp+ LDA	0.557	0.438	0.491
ocp+ oLDA$_t^\infty$	0.615	0.449	**0.519**

(Zika)	Precision	Recall	F-score
bcp count	0.198	0.548	**0.291**
bcp cosine	0.259	0.710	**0.379**
bcp JSD	0.224	0.613	0.328
bcp WD	0.073	1.000	0.137
bcp DD	0.129	0.355	0.190
ecp count	0.353	0.387	**0.369**
ecp cosine	0.440	0.355	**0.393**
ecp JSD	0.343	0.387	0.364
ecp WD	0.269	0.677	0.385
ecp DD	0.278	0.161	0.204
ocp count	0.360	0.290	**0.321**
ocp cosine	0.407	0.355	0.379
ocp JSD	0.480	0.387	**0.429**
ocp WD	0.246	0.968	0.392
ocp DD	0.269	0.226	0.246
ocp+ count	0.240	0.194	**0.214**
ocp+ cosine	0.308	0.258	0.281
ocp+ JSD	0.385	0.323	0.351
ocp+ WD	0.246	0.968	**0.392**
ocp+ DD	0.308	0.258	0.281
LDA vs oLDA$_t^\infty$ (all DD)			
bcp LDA	0.176	0.484	**0.259**
bcp oLDA$_t^\infty$	0.129	0.355	0.190
ecp LDA	0.556	0.161	**0.250**
ecp oLDA$_t^\infty$	0.278	0.161	0.204
ocp LDA	0.400	0.323	**0.357**
ocp oLDA$_t^\infty$	0.269	0.226	0.246
ocp+ LDA	0.440	0.355	**0.393**
ocp+ oLDA$_t^\infty$	0.308	0.258	0.281

locally-transmitted Zika infections. A top document with title "*Singapore confirms the first case of Zika virus transmitted locally*" is counted as correct, whereas "*FDA: Screen all blood donations for Zika*" in another topic is not. Results in Table 2 show that top documents from LDA and oLDA$_t^\infty$ match the majority of reference change points in both datasets. Top documents from oLDA$_t^\infty$ clearly match better than standard LDA. As oLDA$_t^\infty$ updates the word-topic distribution on-line, adding new words as necessary, top documents can better reflect recent change.

5.2 Comparing Topic Change Time Series

We compare the different approaches to encode topic change into time series, as described in Section 3.2: cosine distance, JSD, word-topic (WD) and the document-topic (DD) distributions. We compare with a simple count-based baseline, where the time series is simply the number of document in each time slot. Table 3 (left) shows the performance obtained on the Olympics dataset. The word-topic distribution (WD) and document-topic distribution (DD) time series perform better than the cosine and JSD time series for ecp, ocp and ocp+. The reverse is true for bcp, although its best performance is lower than the other three CPD algorithms, and lower than using the count baseline. All topic change time series perform better than counts for ocp and ocp+. Overall best results are obtained by the DD time series and on-line change point detection.

Table 3 (right) shows the performance obtained on the Zika dataset. Results are not as consistent

as on the Olympics dataset, but time series derived from the on-line topic models outperform, in some configuration, the count baseline. The best performance is obtained by ocp on the JSD distance, followed by cosine distance on ecp, and the word-topic distribution on ocp and ocp+. ecp, ocp and ocp+ perform better than bcp again. Top performance (.429) is also lower than on the Olympics dataset (.523).

The Olympics and Zika datasets are quite different: the Olympics dataset has many documents in each time interval, whereas the Zika dataset is very sparse in the first several months. Documents from different topics in Zika are quite similar at first, and become much diverse later. The JSD and word-topic distribution can capture such changes of topics, but the changes in the document-topic distribution are subtle. Among these methods, only the document-topic distribution (DD) can be applied in both off-line and on-line topic models. The other three methods use the dynamic word-topic distribution at each time slot. Since the performance of the DD time series is best on the Olympic dataset and reasonable on the Zika dataset, we adopt that method in the following experiments.

5.3 Off-line vs. On-line Topic Models

We now focus on the difference between the off-line and on-line LDA. We use the document-topic distribution from LDA and oLDA$_t^\infty$ to produce time series on which we run the four CPD algorithms. Results are reported at the bottom of Table 3. On the Olympics dataset, the performance of CPD from LDA and oLDA$_t^\infty$ are much better than the baseline except on bcp. This confirms that the topic time series extracted from LDA and oLDA$_t^\infty$ help detect relevant changes. The off-line LDA degrades performance of ocp and ocp+ but improves that of ecp, which reaches the top performance on that dataset. On the Zika data set, the performance is better on the LDA time series than on oLDA$_t^\infty$. For bcp and ecp the performance stays below that of the baseline counts, suggesting that information from the topic model either does not help for these algorithms, or does not fit well with their underlying modeling assumptions. This may be due again to the smaller number of documents in the Zika dataset, and lower number of documents in the earlier time period. Note that although LDA yields better performance than oLDA$_t^\infty$ on the Zika dataset, the on-line topic model offers a key functionality: the topic model can be estimated in real-time as documents are acquired. The time series can then be build on-line and change detection can be applied on-line as well, in order to produce real-time change detection.

5.4 On-line Change Point Detection

The above change point detection experiments were performed in a off-line mode for bcp and ecp, and for all CPD algorithms using LDA: CPD algorithms read all data points at all time intervals at once. For real-time detection, however, it is necessary to test the performance in a real on-line fashion. In order to simulate that with bcp and ecp, we use a sliding window, feeding the algorithm overlapping slices of the data, step by step. Although ocp and ocp+ are built as on-line CPD algorithms, processing one new data point at a time, we used the same sliding window to get a fair comparison. On the Olympics and Zika datasets, we tested different sliding window sizes and steps to evaluate the impact of this size on performance. Table 4 shows the result obtained by the four CPD algorithms on either oLDA$_t^\infty$ topic scores or baseline message counts. On the Olympics data, oLDA$_t^\infty$ performs better than counts in all CPD algorithms except bcp. With smaller steps and window sizes, ecp performs the best. ocp performs best on the largest step and sliding window sizes (100 and 200 hours, respectively). The best performance for different settings (bold) are very close. On the Zika data, the performance of CPD algorithms on counts and oLDA$_t^\infty$ are not consistent. ocp+ reaches the best performance on counts using a 150 day window size in 75 day steps, and performs well on oLDA$_t^\infty$ in two other settings, while ecp performs well on counts in the remaining situation (100 day window, 50 day steps). Comparing the best performance obtained in off-line and on-line modes on these two datasets (Table 3 vs. Table 4), we see that the on-line CPD algorithms using sliding windows achieve better performance than using the entire data set off-line.

5.5 Error Analysis

As error analysis, we would like to understand whether all change point detection algorithms detected the same set of sub-events. We examined the sub-events detected by different change point detection

Table 4: F-score of oLDA$_t^\infty$ versus counts, using several sliding window and step sizes.

Olympics				
step (hour)	25	50	75	100
window (hour)	50	100	150	200
bcp count	0.531	0.512	0.472	0.523
bcp oLDA$_t^\infty$	0.341	0.333	0.340	0.327
ecp count	0.548	0.489	0.483	0.495
ecp oLDA$_t^\infty$	**0.582**	**0.555**	**0.557**	0.521
ocp count	0.353	0.286	0.327	0.346
ocp oLDA$_t^\infty$	0.521	0.529	0.527	**0.561**
ocp+ count	0.480	0.442	0.400	0.340
ocp+ oLDA$_t^\infty$	0.463	0.472	0.551	0.513

Zika				
step (day)	25	50	75	100
window (day)	50	100	150	200
bcp count	0.254	0.251	0.250	0.250
bcp oLDA$_t^\infty$	0.169	0.229	0.216	0.213
ecp count	0.317	**0.427**	0.392	0.414
ecp oLDA$_t^\infty$	0.327	0.190	0.218	0.185
ocp count	0.366	0.395	0.324	0.351
ocp oLDA$_t^\infty$	0.326	0.321	0.341	0.361
ocp+ count	0.409	0.383	**0.482**	0.405
ocp+ oLDA$_t^\infty$	**0.447**	0.390	0.346	**0.424**

algorithms on the Olympics dataset in the sliding window setting of 100 hours steps and 200 hours window size (Figure 2). From the top true positives (top), we observe that on oLDA$_t^\infty$ topic scores, all change-point detection algorithms agree much more on the detected sub-events (57 out of 89) than using message counts. All reference sub-events are identified by at least two of four CPD methods. This demonstrates that the document distribution from oLDA$_t^\infty$tracks topic change much better than the raw volume of tweets. In addition, combining the results of four CPD methods might lead to better performance than using any of them alone. Also, looking at the performance of the two off-line and the two off-line methods, we observe that the detected sub-events are more consistent within these two sub-groups than between them, using either counts or oLDA$_t^\infty$.

False positives, shown at the bottom of Fig. 2 are much less consistent across CPD algorithms. Few false positives are detected by all four CPD algorithms. In addition, it is clear that using either counts or oLDA$_t^\infty$ topic scores, bcp generates many more false positives than the other three methods. In particular, on oLDA$_t^\infty$ none of the other three method generate a full positive that is not also produced by bcp. This is reflected in the high recall obtained by bcp (Table 3), at the expense of a very low precision. ocp also suffers from a large number of false positives on oLDA$_t^\infty$. On the contrary, ocp+ generates few false positives, yielding hogh precision but lower recall. Overall, the combination of oLDA$_t^\infty$ and ocp+ reaches the highest true positive rate and lowest false positive rate on the Olympics dataset.

6 Summary

Our study examined real-time detection of topic changes from document streams. The main outcome from the experiments is that real-time CPD based on the topic change detected by on-line topic models can reach good performance, identifying more than half the unknown reference changes from content alone. We also show that top documents extracted from oLDA$_t^\infty$ match reference change point descriptions better than from LDA. This shows that oLDA$_t^\infty$ captures the dynamic changes inside each topic.

We compared four methods of representing topic evolution as time series and evaluated them against reference change points. In the news article stream, the univariate time series based on the JSD distance performed well, while on the large tweet collection, tracking the multivariate document-topic distribution time series performed better. This suggests that selecting the best methods of representing topic evolution for detecting topic changes may depend on the type and genre of documents and remains a challenge.

Our experiments compared off-line with on-line topic models and also off-line CPD with on-line CPD in different settings. The on-line topic model oLDA$_t^\infty$ yields better performance than off-line LDA on the Olympics dataset, but worse performance on the Zika dataset. Also, on-line CPD algorithms ocp and ocp+ did better than off-line CPD algorithms bcp and ecp on both datasets. The combination of on-line topic modeling with the on-line change point detection reaches top performance in topic change detection in our simulated real-time conditions. This demonstrates that our method is promising for real document streaming applications.

7 Conclusion

We propose a method for real-time detection of topic changes in document streams, using a combination of on-line topic modeling and Bayesian on-line change point detection algorithms. Four approaches to

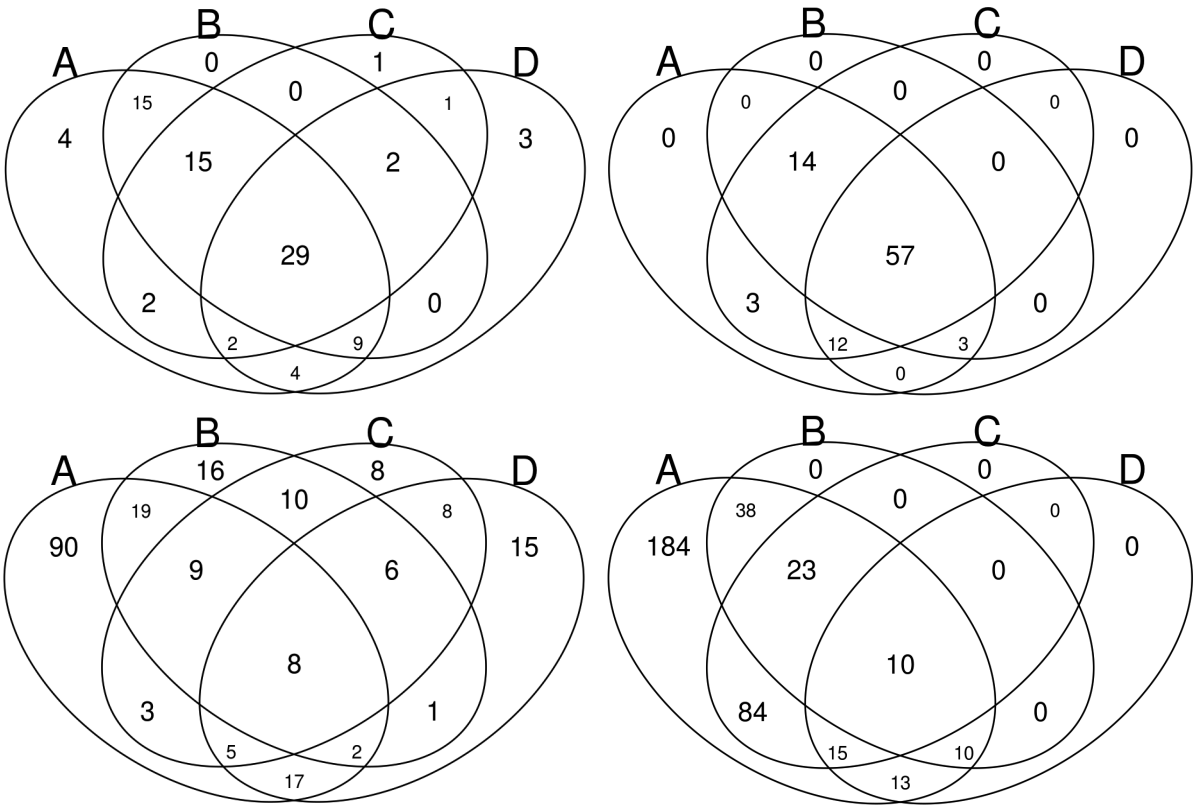


Figure 2: The number of true positives (Top) and false positives (Bottom) detected by A: `bcp`, B: `ecp`, C:`ocp`, D:`ocp+` on all sub-events of the Olympics data, using counts (Left) and oLDA$_t^\infty$(Right).

representing topic change time series were explored, and several off-line and on-line modes were explored. Extensive experiments with off-line and on-line topic models and with retrospective and on-line change point detection on one Twitter dataset and one news article dataset confirm that the combination of oLDA$_t^\infty$ and `ocp+` yield top performance, identifying up to 56% of reference changes in real-time mode on our large twitter collection. Although some off-line combinations yield similar, and sometimes slightly higher performance, the fully on-line combination is always competitive, while offering the benefit of true real-time detection. The CPD algorithms can also be applied to detect changes in other aspect of document streams such as sentiment or emotion.

Acknowledgments

We would like to thank Zachary Zanussi's contribution to develop the `onlineCPD` package and Fangming Liao's contribution to part of the experiments and for Figure 1.

References

Ryan Prescott Adams and David J.C. MacKay. 2007. Bayesian online changepoint detection. arXiv:0710.3742.

Daniel Barry and J. A. Hartigan. 1993. A Bayesian Analysis for Change Point Problems. *Journal of the American Statistical Association*, 88(421):309–319.

David M. Blei and John D. Lafferty. 2006. Dynamic topic models. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 113–120, New York, NY. ACM.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March.

Daniel Bruggermann, Yannik Hermey, Carsten Orth, Darius Schneider, Stefan Selzer, and Gerasimos Spanakis. 2016. Storyline detection and tracking using dynamic latent dirichlet allocation. In *Proceedings of the 2nd Workshop on Computing News Storylines (CNS 2016)*, pages 9–19, Austin, Texas, November. ACM.

Dave Carter. 2018. Revitalizing the Global Public Health Intelligence Network (GPHIN). *Online journal of public health informatics*, 10(1).

Chandra Erdman and John Emerson. 2007. bcp: An R package for performing a bayesian analysis of change point problems. *Journal of Statistical Software*, 23(1):1–13.

Thomas Ertl, Junghoon Chae, Ross Maciejewski, Harald Bosch, Dennis Thom, Yun Jang, and David S. Ebert. 2012. Spatiotemporal social media analytics for abnormal event detection and examination using seasonal-trend decomposition. In *Proceedings of the 2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 143–152, Washington, DC. IEEE Computer Society.

Cyril Goutte, Yunli Wang, Fangming Liao, Zachary Zanussi, Samuel Larkin, and Yuri Grinberg. 2018. Eurogames16: Evaluating change detection in online conversation. In *Proceedings of the 11th Language Resource and Evaluation Conference*.

Valery Guralnik and Jaideep Srivastava. 1999. Event detection from time series data. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 33–42, New York, NY. ACM.

Mahmud Hasan, Mehmet A. Orgun, and Rolf Schwitter. 2017. A survey on real-time event detection from the twitter data stream. *Journal of Information Science*.

Matthew D. Hoffman, David M. Blei, and Francis Bach. 2010. Online learning for latent dirichlet allocation. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems*, pages 856–864. Curran Associates Inc.

Nicholas A. James and David Matteson. 2015. ecp: An R package for nonparametric multiple change point analysis of multivariate data. *Journal of Statistical Software*, 62(1):1–25.

Nicholas A. James, A. Kejariwal, and David S. Matteson. 2016. Leveraging cloud data to mitigate user experience from "breaking bad". In *2016 IEEE International Conference on Big Data (Big Data)*, pages 3499–3508, Dec.

Jey Han Lau, Nigel Collier, and Timothy Baldwin. 2012. On-line trend analysis with topic models: #twitter trends detection topic model online. In *Proceedings of COLING 2012*, pages 1519–1534, Mumbai, India, December.

Alexei Pozdnoukhov and Christian Kaiser. 2011. Space-time dynamics of topics in streaming text. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, pages 1–8, New York, NY. ACM.

Dmitriy Selivanov, 2016. *text2vec: Modern Text Mining Framework for R*. R package version 0.4.0.

Gábor J. Székely and Maria L. Rizzo. 2005. Hierarchical clustering via joint between-within distances: Extending ward's minimum variance method. *Journal of Classification*, 22(2):151–183.

Konstantinos N. Vavliakis, Fani A. Tzima, and Pericles A. Mitkas. 2012. Event detection via LDA for the MediaEval 2012 SED task. In *Proc. MediaEval*.

Yunli Wang and Cyril Goutte. 2017. Detecting changes in twitter streams using temporal clusters of hashtags. In *Proceedings of the Events and Stories in the News Workshop*, pages 10–14. Association for Computational Linguistics.

Ke Zhai and Jordan Boyd-Graber. 2013. Online latent dirichlet allocation with infinite vocabulary. In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, pages I–561–569.

Xiangmin Zhou and Lei Chen. 2014. Event detection over twitter social media streams. *The VLDB Journal*, 23(3):381–400, June.

Automatically Creating a Lexicon of Verbal Polarity Shifters: Mono- and Cross-lingual Methods for German

Marc Schulder, Michael Wiegand
Spoken Language Systems
Saarland University
Germany
marc.schulder@lsv.uni-saarland.de
michael.wiegand@lsv.uni-saarland.de

Josef Ruppenhofer
Institute for German Language
Mannheim
Germany
ruppenhofer@ids-mannheim.de

Abstract

In this paper we use methods for creating a large lexicon of verbal polarity shifters and apply them to German. Polarity shifters are content words that can move the polarity of a phrase towards its opposite, such as the verb *"abandon"* in *"abandon all hope"*. This is similar to how negation words like *"not"* can influence polarity. Both shifters and negation are required for high precision sentiment analysis. Lists of negation words are available for many languages, but the only language for which a sizable lexicon of verbal polarity shifters exists is English. This lexicon was created by bootstrapping a sample of annotated verbs with a supervised classifier that uses a set of data- and resource-driven features. We reproduce and adapt this approach to create a German lexicon of verbal polarity shifters. Thereby, we confirm that the approach works for multiple languages. We further improve classification by leveraging cross-lingual information from the English shifter lexicon. Using this improved approach, we bootstrap a large number of German verbal polarity shifters, reducing the annotation effort drastically. The resulting German lexicon of verbal polarity shifters is made publicly available.

Title and Abstract in German

Die automatische Erstellung eines Lexikons polaritätsverschiebender Verben:
Einsprachige und sprachübergreifende Methoden für das Deutsche

In dieser Arbeit untersuchen wir Methoden zur Erstellung eines deutschsprachigen Lexikons polaritätsverschiebender Verben. Diese Verben, die vielfach auch Polaritätsshifter genannt werden, sind Inhaltswörter, die die Polarität einer Phrase zu ihrem entgegengesetzten Wert verschieben, wie z.B. das Verb *„aufgeben"* in der Verbalphrase *„alle Hoffnung aufgeben"*. Das Verhalten von Polaritätsshiftern ähnelt somit dem von Negationswörtern wie *„nicht"*. Für robuste Sentimentanalyse werden sowohl Negationswörter als auch Polaritätsshifter benötigt. Während Listen von Negationswörtern in vielen Sprachen verfügbar sind, existiert jedoch ein Polaritätsshifter-Lexikon hinreichender Größe nur für das Englische. Jene Ressource wurde mittels Bootstrapping erzeugt, indem ein überwachter Klassifikator auf einer kleinen Stichprobe von Verben trainiert wurde. Dieser Klassifikator nutzt Daten- und Ressourcen-getriebene Merkmale. Wir reproduzieren diesen Ansatz und passen ihn soweit notwendig für das Deutsche an. Somit weisen wir die Übertragbarkeit dieses Ansatzes auf andere Sprachen nach. Wir verbessern die Qualität der Klassifikation zudem weiterhin, indem wir Informationen aus dem existierenden englischen Polaritätsshifter-Lexikon nutzten. Mittels dieses verbesserten Ansatzes finden wir per Bootstrapping eine große Anzahl deutscher Polaritätsshifter und verringern somit deutlich den manuellen Annotationsaufwand. Das resultierende deutsche Lexikon polaritätsverschiebender Verben ist frei verfügbar.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2516–2528
Santa Fe, New Mexico, USA, August 20-26, 2018.

1 Introduction

Polarity shifters are content words such as verbs, nouns or adjectives that influence the sentiment polarity of an expression in ways similar to negation words. For example, the negated statement in (1) that uses the negation word *nicht* in German and *not* in English can also be expressed using the verbal shifter *unterlassen* in German and *fail* in English, as seen in (2).

(1) Peter hat ihnen **nicht** geholfen.
Peter did **not** help them.
(2) Peter hat es **unterlassen**$_{shifter}$ ihnen zu helfen.
Peter **failed**$_{shifter}$ to help them.

Polarity shifters can affect both positive and negative expressions, moving their polarity towards the opposite polarity. In (3) the shifter *verweigern/deny* affects the positive polar expression *Stipendium/scholarship*, resulting in a negative polarity for the sentence. On the other hand, the shifter *lindern/alleviate* in (4) creates a positive sentence despite the negative polar expression *Schmerz/pain*.

(3) Ihr wurde das [[Stipendium]$^{+}$ **verweigert**$_{shifter}$]$^{-}$.
She was [**denied**$_{shifter}$ the [scholarship]$^{+}$]$^{-}$.
(4) Die neue Behandlung hat ihre [[Schmerzen]$^{-}$ **gelindert**$_{shifter}$]$^{+}$.
The new treatment has [**alleviated**$_{shifter}$ her [pain]$^{-}$]$^{+}$.

As can be seen for *verhindern/prevent* in (5) and (6), the same shifter can even affect both positive and negative expressions.

(5) Seine Prinzipien [**verhinderten**$_{shifter}$ eine [Einigung]$^{+}$]$^{-}$.
His principles [**prevented**$_{shifter}$ an [agreement]$^{+}$]$^{-}$.
(6) Ihre Maßnahmen [**verhinderten**$_{shifter}$ ein [Gemetzel]$^{-}$]$^{+}$.
Their measures [**prevented**$_{shifter}$ a [slaughter]$^{-}$]$^{+}$.

We present a **reproduction** and **extension** to the work of Schulder et al. (2017), which introduced a **lexicon of verbal polarity shifters**, as well as methods to increase the size of this lexicon through bootstrapping. The lexicon lists verb lemmas and assigns a binary label (*shifter* or *no shifter*) to each. The original approach was developed on English. We apply it to **German**, validating the generality of the approach and creating a new resource, a German lexicon of 677 verbal polarity shifters. We also improve the bootstrapping process by adding features that leverage polarity shifter resources across languages.

As is the case with negation, modeling polarity shifting is important for various tasks in NLP, such as relation extraction (Sanchez-Graillet and Poesio, 2007), recognition of textual entailment (Harabagiu et al., 2006) and especially sentiment analysis (Wiegand et al., 2010). However, while there has been significant research on negation in sentiment analysis (Wiegand et al., 2010), current classifiers fail to handle polarity shifters adequately (Schulder et al., 2017). This is in part due to the lack of lexical resources for polarity shifters. Unlike negation words (*no*, *not*, *never*, etc.), of which there are only a few dozen in a language, polarity shifters are far more numerous. Among verbs alone there are many hundreds (Schulder et al., 2017). Comprehensive shifter lexicons are, therefore, considerably more expensive to create. Once available, they can be used to improve the aforementioned tasks, as has already been shown for the case of English polarity classification (Schulder et al., 2017).

To reduce the cost of creating such polarity shifter lexicons, Schulder et al. (2017) introduced methods to automatically generate a labeled list of words using either a limited amount of labeled training data or no labeled data at all. Their approach includes both features that rely on semantic resources and data-driven ones. They limited their work to English verbs, but expressed the expectation that their methods should also work for other languages. To verify that expectation, we apply their approach to German, for which all resources required to reproduce their experiments are available. Keeping in mind that this is not the case for many other languages, we focus our evaluation on differentiating between features that rely on unstructured data and those requiring rare semantic resources.

While polarity shifters are not restricted to a particular part of speech – shifter nouns (e.g. *downfall*), adjectives (*devoid*) and adverbs (*barely*) also exist – we limit ourselves to verbs. Verbs and nouns are the most important minimal semantic units (Schneider et al., 2016) and verbs are usually the main syntactic

predicates of clauses, projecting far-reaching scopes. Focusing on verbs also allows us a closer comparison with Schulder et al. (2017) and to investigate cross-lingual similarities between verbal shifters.

The **contributions** of this paper are:

(i) we introduce a German lexicon of verbal polarity shifters;
(ii) we reproduce and adapt the approach of Schulder et al. (2017) to German to extend our lexicon;
(iii) we introduce additional methods that take advantage of the existence of the English verbal polarity shifter lexicon and improve upon the current state of the art.

The focus of our work is the binary classification of verbal polarity shifters in German. The resulting German lexicon of 677 verbal polarity shifters is made **publicly available**.[1]

2 Related Work

Existing work on negation modeling focuses almost exclusively on negation words (see the survey of Wiegand et al. (2010)). One reason for this is the lack of lexicons and corpora that cover other forms of polarity shifters. Even the most complex negation lexicon for English sentiment analysis (Wilson et al., 2005) includes a mere 12 verbal shifters. So far the only larger resources for polarity shifters are the English-language verbal shifter lexicons recently introduced by Schulder et al. (2017) and Schulder et al. (2018). Schulder et al. (2017) automatically bootstrap a lexicon which covers 980 verbal shifters at the lemma level, while Schulder et al. (2018) manually annotate word senses of verbs, creating a lexicon of 2131 shifter senses across 1220 verbs. As we reproduce and extend the work of Schulder et al. (2017), all further use of and comparison to an English shifter lexicon refers to their bootstrapped lexicon as well.

To create shifter lexicons at a large scale, automation and bootstrapping techniques are required. Danescu-Niculescu-Mizil et al. (2009) propose using *negative polarity items (NPIs)* to extract downward-entailing operators, which are closely related to polarity shifters. Schulder et al. (2017) also make use of NPIs in addition to a number of other features.

Rather than using lexicons, another approach would be to learn polarity shifters from labelled corpora. In the case of negation, this has already been examined for the biomedical domain (Huang and Lowe, 2007; Morante and Daelemans, 2009; Zou et al., 2013), the review domain (Ikeda et al., 2008; Kessler and Schütze, 2012; Socher et al., 2013; Yu et al., 2016) and across domains (Fancellu et al., 2016). Unfortunately, due to the considerably higher lexical diversity of polarity shifters, far larger corpora would be required for learning shifter than for learning negation.

Available corpora that are suitable for negation learning, such as the Sentiment Treebank (Socher et al., 2013) or the BioScope corpus (Szarvas et al., 2008), are fairly small in size. Most verbs occur in them in very few instances or not at all. In the BioScope corpus, for example, there are only 6 verbal shifters (Morante, 2010). Polarity classifiers trained on such corpora, such as the state-of-the-art Recursive Neural Tensor Network tagger (Socher et al., 2013), fail to detect many instances of polarity shifting. Schulder et al. (2017) show that the explicit knowledge provided by a shifter lexicon can improve polarity classification in such cases.

3 Data

We create a gold standard for German verbal shifters, following the approach Schulder et al. (2017) used for their English gold standard. An expert annotator, who is a native speaker of German, labeled 2000 verbs, randomly sampled from GermaNet (Hamp and Feldweg, 1997), a German wordnet resource. The remaining 7262 GermaNet verbs are used to bootstrap a larger lexicon in §5.3.

Each verb is assigned a binary label of being a shifter or not. To qualify as a shifter, a verb must permit polar expressions as its dependents and cause the polarity of the expression that embeds both verb and polar expression to move towards the opposite of the polar expression. For example, in (6) *verhindern* shifts the negative polarity of its dependent *ein Gemetzel*, resulting in a positive expression. Annotation is performed at the lemma level, as word-sense disambiguation tends to be insufficiently robust.

[1] https://github.com/uds-lsv/coling2018

Resource Type	German Resource	English Resource
Wordnet	GermaNet (Hamp and Feldweg, 1997)	WordNet (Miller et al., 1990)
Text Corpus	DeWaC Web Corpus (Baroni et al., 2009)	Amazon Product Reviews (Jindal and Liu, 2008)
Polarity Lexicon	PolArt Sentiment Lexicon (Klenner et al., 2009)	Subjectivity Lexicon (Wilson et al., 2005)
Framenet	Salsa (Burchardt et al., 2006)	FrameNet (Baker et al., 1998)
Effects	EffektGermaNet (Ruppenhofer and Brandes, 2015)	EffectWordNet (Choi et al., 2014)

Table 1: Required German resources, compared with English resources used by Schulder et al. (2017).

	Frequency	Percentage
shifter	224	11.2
no shifter	1776	88.8

Table 2: Distribution of verbal shifters in annotated sample of 2000 verbs taken from GermaNet.

	Polar Verbs		Positive V.		Negative V.	
	Freq	%	Freq	%	Freq	%
shifter	81	23.1	12	11.7	69	27.9
no shifter	269	76.9	91	88.3	178	72.1

Table 3: Distribution of verbal shifters in the *PolArt Sentiment Lexicon* (Klenner et al., 2009).

Table 1 provides an overview of the German resources we use in our reproduction, compared to the resources used for the English shifter lexicon. More detailed descriptions of the resources are provided in sections discussing feature design (§4) and experiments (§5).

Table 2 shows that in our gold data 11.2% of verbs are shifters, which is a bit less than the 15.2% of the English gold standard. Table 3 shows the shifter distribution among verbs with sentiment polarity (determined using the *PolArt Sentiment Lexicon* (Klenner et al., 2009)). As was the case for the English gold data, it shows a tendency for shifter verbs to be negative rather than positive terms.

4 Feature Design

In this section we introduce the features that we will use to bootstrap our German verbal shifter lexicon in §5.3. We start by outlining the features proposed by Schulder et al. (2017) and how we adapt them for use with German (§4.1). We further separate them into data-driven features (§4.1.1) and resource-driven features (§4.1.2) to highlight their requirements when applied to a new language.

In §4.2 we introduce new methods that can either be used as stand-alone classifiers or as features for an SVM classifier. Both methods take advantage of existing knowledge about English verbal shifters. One method uses a bilingual dictionary (§4.2.1) and the other cross-lingual word embeddings (§4.2.2).

4.1 Feature Reproduction

In this section we briefly describe how we adapt the features of Schulder et al. (2017) to German language data. We distinguish between features that mainly rely on text data from a corpus (§4.1.1) and those that require complex semantic resources (§4.1.2). When working with languages with scarcer resources, it can be expected that the former will be more readily available than the latter.

4.1.1 Data-driven Features

The main requirement of the following features is a reasonably sized text corpus to detect syntactic patterns and word frequencies. The text corpus was lemmatized using the TreeTagger (Schmid, 1994) and parsed for syntactic dependency structures with ParZu (Sennrich et al., 2009).[2] For features requiring knowledge of polarities we use the *PolArt Sentiment Lexicon* (Klenner et al., 2009).[3]

[2]Lacking an appropriate parser, a part-of-speech tagger may approximate required syntactic structures (Riloff et al., 2013).

[3] We chose to consider features that use a polarity lexicon to still be data-driven features as there exist robust methods to generate them automatically from unlabeled corpora (Turney, 2002; Velikovich et al., 2010; Hamilton et al., 2016). The lexicon we use was created using bootstrapping (Clematide and Klenner, 2010).

Distributional Similarity (SIM): The distributional similarity feature assumes that words that are semantically similar to negation words are also likely to be polarity shifters. Semantic similarity is modeled as cosine similarity in a word embedding space. The word embeddings are created using Word2Vec (Mikolov et al., 2013) on the German web corpus DeWaC (Baroni et al., 2009), using the same hyperparameters as Schulder et al. (2017) and German translations of their negation seeds.

Polarity Clash (CLASH): The polarity clash feature assesses that shifting will often occur when a polar verb modifies an expression of the opposite polarity, such as in (7). The feature is further narrowed down to negative verbs that modify positive nouns, as polar verbal shifters are predominantly of negative polarity (Table 3).

(7) Er hat die [[Hoffnung]$^{+}$ [**verloren**]$^{-}$]$^{-}$.
He [[**lost**]$^{-}$ [hope]$^{+}$]$^{-}$.

Particle Verbs (PRT): Certain verb particles indicate a complete transition to an end state (Brinton, 1985). Schulder et al. (2017) hypothesize that this phenomenon correlates with shifting, which can be seen as producing a new (negative) end state. Therefore, they collect particle verbs containing relevant English particles, such as *away*, *down* and *out*. For our German data we chose the following particles associated with negative end states: *ab*, *aus*, *entgegen*, *fort*, *herunter*, *hinunter*, *weg* and *wider*.

Heuristic using 'jeglich' (ANY): *Negative polarity items (NPIs)* are known to occur in the context of negation (Giannakidou, 2008). Schulder et al. (2017) showed that the English NPI *any* co-occurs with shifters, so its presence in a verb phrase can indicate the presence of a verbal shifter. We expect the same for the German NPI *jeglich*, as seen in (8). We collect all verbs with a polar direct object that is modified by the lemma *jeglich*. The resulting pattern matches are sorted by their frequency, normalized over their respective verb frequency and then reranked using *Personalised PageRank* (Agirre and Soroa, 2009).

(8) Sie [**verwehrten**$_{shifter}$ uns jegliche [Hilfe$_{dobj}$]$^{+}$]$^{-}$.
They [**denied**$_{shifter}$ us any [help$_{dobj}$]$^{+}$]$^{-}$.

Anti-Shifter Feature (ANTI): This feature specifically targets anti-shifters, verbs that exhibit polar stability instead of causing polar shifting. These are commonly verbs indicating creation or continued existence, such as *live*, *introduce*, *construct* or *prepare*. Such verbs often co-occur with the adverbs *ausschließlich*, *zuerst*, *neu* and *extra*, as seen in (9)–(12). Accordingly, we can create a list of anti-shifters by selecting the verbs that most often co-occur with these adverbs.

(9) Im Winter **leben**$_{antiShifter}$ Schwarzbären ausschließlich von Fisch.
In winter, black bears exclusively **live**$_{antiShifter}$ on fish.
(10) Komplette Tastaturen auf Handys wurden zuerst in 1997 **eingeführt**$_{antiShifter}$.
Full keyboards on cellphones were first **introduced**$_{antiShifter}$ in 1997.
(11) Diese Gebäude wurden neu **gebaut**$_{antiShifter}$.
These buildings have been newly **constructed**$_{antiShifter}$.
(12) Sie haben extra für mich veganes Essen **zubereitet**$_{antiShifter}$.
They specially **prepared**$_{antiShifter}$ vegan dishes for me.

4.1.2 Resource-driven Features

The following features rely on advanced semantic resources which are available in only a few languages.

GermaNet: Wordnets are large lexical ontologies providing various kinds of semantic information and relations. Schulder et al. (2017) used glosses, hypernyms and supersenses taken from the English WordNet (Miller et al., 1990) as features in their work. We use GermaNet (Hamp and Feldweg, 1997), a German wordnet resource that provides all these features. In the case of glosses, called paraphrases in GermaNet, GermaNet offers two variations: the paraphrases originally written for GermaNet, and a more extensive set of paraphrases harvested from Wiktionary (Henrich et al., 2014). To improve coverage we use this paraphrase extension in our experiments.

Salsa FrameNet: Framenets provide semantic frames that group words with similar semantic behavior. Schulder et al. (2017) use the frame memberships of verbs as a feature, hypothesizing that verbal shifters will be found in the same frames. We reproduce this feature using frames from the German FrameNet project *Salsa* (Burchardt et al., 2006).

EffektGermaNet: Wiebe and colleagues (Deng et al., 2013; Choi et al., 2014) introduced the idea that events can have harmful or beneficial *effects* on their objects. These *effects* are related but not identical to polarity shifting. Choi et al. (2014) provide lexical information on *effects* in their English resource EffectWordNet. We use its German counterpart, EffektGermaNet (Ruppenhofer and Brandes, 2015), to model the *effect* feature in our data.

4.2 New Features

In §4.1 we described how we reproduce features already used for English shifter classification. Next we introduce new features that have not yet been used for the creation of a verbal shifter lexicon.

4.2.1 Bilingual Dictionary

The motivation behind the work of Schulder et al. (2017) was to introduce a large lexicon of verbal polarity shifters. Now that such a lexicon exists for English, it is an obvious resource to use when creating verbal shifter lexicons for other languages. We hypothesize that a verb with the same meaning as an English verbal shifter will also function as a shifter in its own language. All that is required is a mapping from English verbs to, in our case, German verbs. We choose to use the bootstrapped lexicon of Schulder et al. (2017), rather than the manually created one of Schulder et al. (2018), to show that bootstrapping is sufficient for all stages of the learning process.

One potential source for such a mapping is a bilingual dictionary. We use the English-German dataset by DictCC[4], as it is large (over one million translation pairs) and publicly available. It covers 76% of German verbs found in GermaNet and 77% of English verbs found in WordNet.

Mapping the shifter labels of the English verbs to German verbs is performed as follows: For each German verb, all possible English translations are looked up. Using the English verbal shifter lexicon, we confirm whether the English translations are shifters. If the majority of translations are shifters, the German word is also labeled as a shifter, otherwise as not a shifter. This approach provides explicit labels for 1368 of our 2000 gold standard verbs (68%). Less than 6% of these are tied between *shifter* and *no shifter* translations. Ties are resolved in favor of the *shifter* label. The remaining verbs are labeled with the majority label *no shifter*.

While this bilingual dictionary mapping approach makes for a promising feature, we refrain from considering it for generating a gold standard. Using a dictionary instead of annotating a random sample would introduce biases existing in the dictionary, e.g. more translation pairs being available for frequent words, which can in turn favor features that work better for frequent words. Schulder et al. (2017) also observe in their error analysis that some verbs act as shifters in only some of their word senses. As different word senses often do not translate into the same foreign word, indiscriminate translation may introduce non-shifting senses of English shifter words as false positives. Evaluating the dictionary mapping as a feature will allow us to judge its usefulness for high-precision lexicon induction in future works.

4.2.2 Cross-lingual Word Embeddings

As an alternative to using bilingual dictionaries we investigate transferring English shifter labels to German using cross-lingual word embeddings. These are word embeddings which provide a shared vector space for words from multiple languages. Similar to how the SIM feature (see §4.1.1) compares negation words to verbs in a mono-lingual word embedding, a cross-lingual word embedding allows us to compare English verbs to verbs of another language based on their distributional similarity without having labeled data for the other language. These comparisons can then be used to apply the labels of the English lexicon of verbal shifters to the other language.

Mapping shifter labels cross-lingually with a bilingual dictionary, as described in §4.2.1, requires a dictionary with good coverage for both languages. For many languages, publicly available dictionaries of adequate size are hard to come by. For instance, the second largest English dictionary on DictCC is only 40% the size of the English-German dataset and only a few others have more than 2% its size.

[4] `https://www.dict.cc`

In §5.2 we explore the effect of dictionary size on mapping performance and how cross-lingual word embeddings fare in comparison.

Methods for creating cross-lingual word embeddings can be grouped into *cross-lingual training* and *monolingual mappings*. *Cross-lingual training* learns joint embeddings from parallel corpora. However, such corpora are far smaller and rarer than monolingual corpora and, therefore, not ideal for us.[5]

Monolingual mappings take preexisting monolingual word embeddings and learn linear transformations to map both embeddings onto the same vector space. Commonly, these approaches use bilingual dictionaries to initialize this mapping, which would rather defeat our goal of using embeddings as a data-driven alternative to dictionaries. The *VecMap* framework (Artetxe et al., 2017) provides an initialization method that relies on numerals instead of a dictionary. The idea behind this is that Arabic numerals are used in most languages, even across different writing systems (e.g. Cyrillic, Chinese, etc.), and, therefore, can function as a dictionary without requiring actual bilingual knowledge.

For our experiments, we train Word2Vec word embeddings for English and German, using the Amazon Product Review (Jindal and Liu, 2008) and DeWaC (Baroni et al., 2009) corpora, respectively. Ideally, product review corpora would be used for both languages, but available German review corpora are considerably smaller than their English counterparts. For example, the German corpus Webis-CLS (Prettenhofer and Stein, 2010) contains only 33 million words, while the English-language Amazon Product Review Corpus consists of 1.2 billion words. When generating word embeddings, the size of the corpus is very important for the quality of the resulting embedding, so we choose instead to use DeWaC, a web corpus of 1.7 billion words.

Training is performed using the same hyperparameters as used by Artetxe et al. (2017).[6] We use VecMap to create a cross-lingual word embedding using the default configuration for numeral-based mappings. The resulting cross-lingual embedding covers 79% of German GermaNet verbs as well as 79% of English WordNet verbs. It covers 1598 of our 2000 gold data verbs (80%).

We use this new word embedding to apply English shifter labels to German. To achieve this, we go through our list of German verbs, look up the most similar English verb for each and apply its label. We also investigated majority voting using k-nearest neighbors, but this did not improve performance.

5 Experiments

5.1 Classifier Evaluation

We start our evaluation by reproducing the classifier evaluation of Schulder et al. (2017). The task is the classification of all verbs from the given gold standard in a 10-fold cross validation.

Analogous to Schulder et al. (2017) we evaluate a supervised SVM classifier as well as a graph-based label propagation (LP) classifier that requires no labeled training data. In addition, we evaluate our cross-lingual word embedding classifier (§4.2.2) and our dictionary classifier (§4.2.1), which both make use of the pre-existing English lexicon, but require no additional labeled German data. For an overview of the classifiers and their data requirements, see Table 4.

For the LP classifier we use the ANY features as seeds for the positive label (*shifter)* and the ANTI feature as negative label (*no shifter)* seeds. For SVM we group features into data-driven and resource-driven feature sets (see Table 6) as outlined in §4.1.1 and §4.1.2, as well as introducing the outputs of the cross-lingual word embedding and dictionary classifiers as additional separate features.

Table 5 shows the performance of our various classifiers. All classifiers clearly outperform the baseline[7] and resource-based features outperform data-based ones. This is similar to performance observed

[5] BilBOWA (Gouws et al., 2015) seeks to improve the coverage problem of parallel corpora by incorporating additional monolingual corpora into the training process. However, our experiments with it did not provide satisfactory results. This is in line with reports by Artetxe et al. (2017) and Upadhyay et al. (2016).

[6] Word2Vec configuration: CBOW, 300 dimensions, context window of 5 words, sub-sampling at $1e-05$, negative samples at 10 and vocabulary restricted to the 200,000 most frequent words. We also experimented with using the full vocabulary, but this resulted in lower quality embeddings.

[7] As in Schulder et al. (2017), accuracy proves to be a problematic measure, as it has a strong majority label bias. The *no shifter* label makes up 88.8% of our gold annotation (Table 2), which explains the strong performance of the majority baseline on this metric.

Classifier	Features	Shifter Lex	Text Corpus	Training Data
SIM	Data-driven	—	German	—
$LP_{ANY+ANTI}$	Data-driven	—	German	—
Cross-ling. Embedding	—	English	German, English	—
Dictionary	Bilingual Dictionary	English	—	—
$SVM_{data+resource}$	Data-driven, Resource-driven	—	German	German

Table 4: Classifiers used in Table 5 and their resource requirements.

Classifier	Acc	Prec	Rec	F1
$Baseline_{majority}$	88.1	44.4	50.0	47.0
SIM	70.7	58.0	67.6	62.4
$LP_{ANY+ANTI}$	87.1	67.2	65.0	66.1
Cross-lingual Embedding	85.1	67.6	74.6	$70.9^{*\dagger}$
Dictionary	86.5	69.2	77.3	$73.0^{*\dagger}$
SVM_{data}	74.6	60.8	72.6	66.2
$SVM_{resource}$	91.3	79.4	73.9	$76.4^{*\dagger}$
$SVM_{data+resource}$	91.4	79.0	76.7	$77.7^{*\circ\dagger}$
$SVM_{data+resource+embed}$	91.6	79.6	78.9	$79.2^{*\circ\dagger}$
$SVM_{data+resource+dict}$	91.3	78.0	80.9	$79.4^{*\circ\dagger}$
$SVM_{data+resource+dict+embed}$	**92.1**	**80.3**	**82.0**	$\mathbf{81.0}^{*\circ\dagger\ddagger}$

statistical significance (paired t-test with $p < 0.05$):
* better than LP; ° better than Dictionary; † better than SVM_{data}; ‡ better than $SVM_{data+resource}$

Table 5: Evaluation of classification (§5.1) on the 2000 verb gold standard (Table 2). Precision, recall and f-score are macro-averages.

Group	Features
data	$LP_{ANY+ANTI}$, SIM, CLASH, PRT
resource	GermaNet, Salsa, EffektGermaNet
embed	Cross-lingual Embedding
dict	Dictionary

Table 6: Features included in SVM feature groups in Table 5. All features in *data* and *resource* were also used in Schulder et al. (2017).

for English (Schulder et al., 2017). Cross-lingual embeddings and dictionaries as stand-alone classifiers both outperform the label propagation approach due to their better recall coverage of shifters.

Interestingly, the cross-lingual embedding classifier performs far better than SIM, despite both relying on word embeddings to judge distributional similarity. Comparing similarity among verbs, even cross-lingually, works better than across parts-of-speech, as required for negation-shifter comparisons.

Adding both cross-lingual features to the SVM classifier improves performance further. This shows that they are not only complementary to the existing features, but also to each other, as using only one cross-lingual feature does not improve performance as much. The most feature-rich SVM configuration, $SVM_{data+resource+dict+embed}$, provides a significant improvement over $SVM_{data+resource}$, the best classifier of Schulder et al. (2017). We conclude that cross-lingual shifter information is useful even when the same bootstrapping process and feature set is used in both the source and target language.

Figure 1 shows the learning curve of select SVM configurations, compared to the classifiers that work without labeled German data, i.e. LP, Embedding and Dictionary. Cross-lingual embedding and dictionary classifiers provide a stronger baseline than LP, outperforming $SVM_{data+resource}$ when training data is sparse. However, adding them as features to the SVM results in a classifier that consistently improves upon all other systems, even at small training sizes of only 20%. Combining all available sources of information as SVM features is therefore the preferred approach if any amount of training data is available.

5.2 Evaluation of Dictionary Size

The dictionary mapping approach (§4.2.1) has been shown to be a strong stand-alone classifier and SVM feature (Table 5), slightly outperforming the cross-lingual word embedding approach (§4.2.2). However, the underlying English-German dictionary by DictCC is of considerable size, consisting of over 1.1 million translation pairs. Even then, almost a quarter of WordNet and GermaNet verbs are not

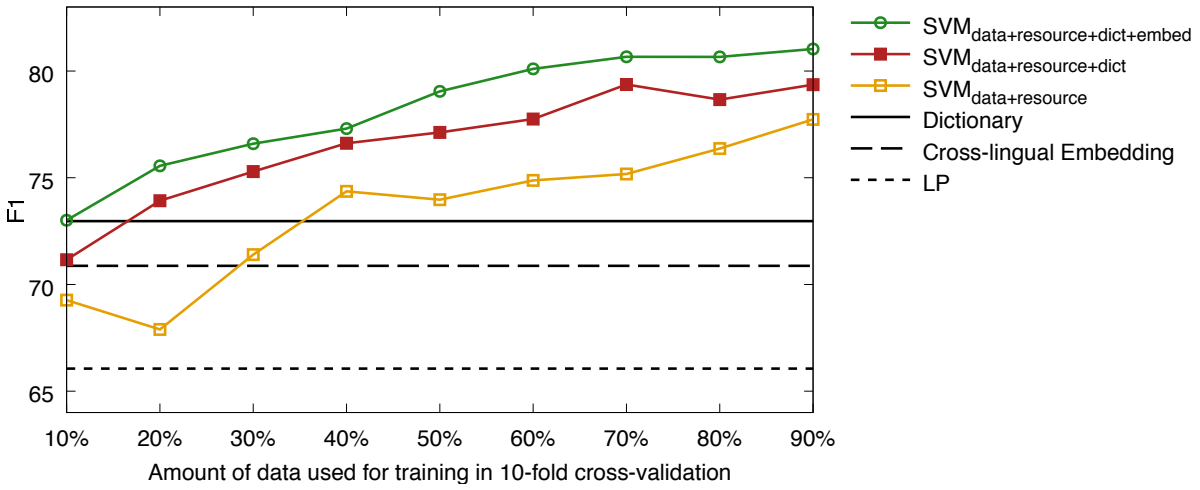


Figure 1: Learning curve on gold standard. $\text{SVM}_{\text{data+resource}}$ represents the previously best classifier (Schulder et al., 2017).

covered. For many other languages, finding a publicly available dictionary of comparable size may pose a challenge. Therefore, we investigate how smaller dictionaries may perform in our classifiers.

The English-German DictCC dictionary covers slightly over 8000 of the English verbs found in WordNet. Of the 2000 German verbs in our gold standard, DictCC covers 1368. To simulate bilingual dictionaries of smaller size, we create a version of the DictCC dictionary with half the English vocabulary by limiting it to the 4000 most frequent verbs from WordNet ($\text{Dict}_{\text{voc\_size=4k}}$). We also create even smaller versions with only the 1000 ($\text{Dict}_{\text{voc\_size=1k}}$) and 500 most frequent English verbs ($\text{Dict}_{\text{voc\_size=0.5k}}$).

As bilingual dictionaries provide a many-to-many mapping, having half the English vocabulary does not necessarily mean that we receive only half the German translations. Many German words receive multiple translations, all of which we then use to determine their shifter label via majority vote. Reducing the English vocabulary, therefore, first reduces the number of label votes for each German word, until, eventually, German words are removed as there are no more votes for them. Having fewer votes per German output label can, however, still affect the robustness of the labeling process. In our case, reducing the English vocabulary by half still provides translations for 1168 of German words in our gold data, i.e. 85% of the full dictionary. Reducing it further to 1000 English verbs drops the size of the German vocabulary to 52%. Using only the 500 most frequent English words leaves a German coverage of 33%.

Figure 2 shows the performance of the differently sized dictionaries as stand-alone classifiers, while Figure 3 shows how much they can improve the best classifier of Schulder et al. (2017), i.e. $\text{SVM}_{\text{data+resource}}$. In both cases we see that while even smaller dictionaries can still provide acceptable performance, using cross-lingual embeddings is preferable to using a dictionary of insufficient size.

5.3 Bootstrapping

In their intrinsic evaluation Schulder et al. (2017) showed that explicit knowledge of a large number of polarity shifters can improve sentiment analysis. To increase the size of our lexicon, we bootstrap additional shifters following their approach. We train our best classifier (Table 5) on the 2000 verbs from our gold standard (§3) and then use it to classify the remaining 7262 GermaNet verbs that had not been labeled so far. Of these, the classifier labels 595 verbs as shifters. A German native speaker manually checks these predicted shifters and confirms 453 to be true verbal shifters. Limiting our annotation effort to predicted shifters and discarding all others reduces the cost of annotation by 92%.

Table 7 shows the classifier precision at different confidence intervals. Like Schulder et al. (2017), we see very high performance for the first quartile, matching their observation that manual confirmation is not strictly necessary for high confidence labels. Combining the 453 bootstrapped shifters with the 224 shifters from the gold standard we produce a **novel list of 677 German verbal shifters** *(see footnote 1)*.

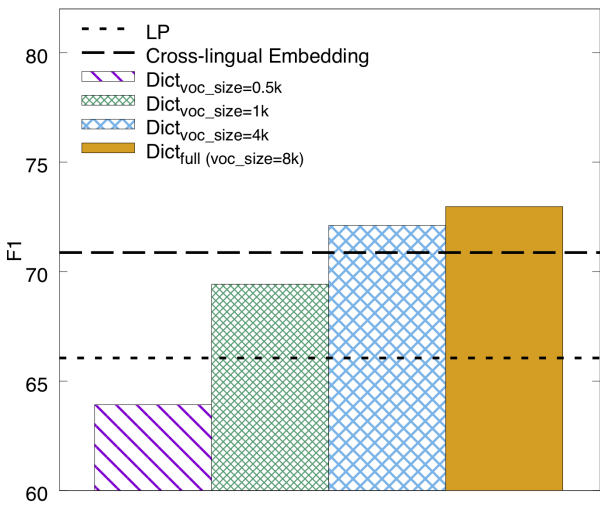


Figure 2: Comparison of dictionaries with different vocabulary sizes. Classifiers use *no labeled training data*. $\text{Dict}_{\text{full}}$ is equivalent to the dictionary shown in Table 5.

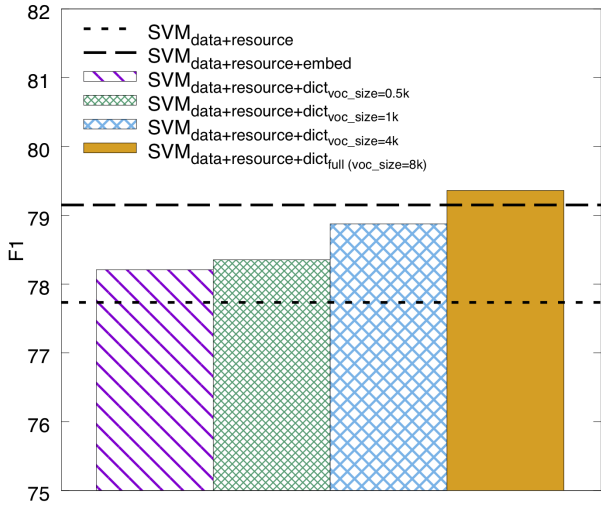


Figure 3: Comparison of SVM classifiers using dictionaries with different vocabulary sizes. $dict_{full}$ is equivalent to the SVM feature *dict* in Table 5.

Confidence Rank	1–150	151–300	301–450	451–595
Precision	97.3	83.3	70.0	53.1

Table 7: Classification of GermaNet verbs that were ***not*** part of gold standard (§3); verbs are ranked by confidence-score of classifier and evaluated at intervals by precision of *shifter* label.

6 Conclusion

We confirm that the bootstrapping process for creating a large lexicon of verbal polarity shifters can successfully be applied to German. Given appropriate resources, the effort for adjusting to a new language is minimal, mostly requiring translating seed words and adjusting syntactic patterns, while the underlying concepts of the features remain the same. Using a manually annotated sample of 2000 verbs taken from GermaNet, we train a supervised classifier with various data- and resource-driven features. Its performance is further improved by leveraging information from an existing English lexicon of verbal shifters using bilingual dictionaries and cross-lingual word embeddings. The resulting improved classifier allows us to triple the number of confirmed German shifters in our lexicon.

We differentiate features by whether they require only unlabeled data and basic linguistic tools or whether they depend on rare semantic resources that may not be available for many languages. In addition, we introduce the possibility of using cross-lingual resources to reduce the dependence on resources in the target language. This shows promise, improving performance for both unsupervised and supervised classification, especially for scenarios where only small amounts of training data are available. However, supervised learning that combines all features still provides the best results.

Our recommendation for creating shifter lexicons in new languages is to start out with cross-lingual label transfer, but to also invest in annotating a random sample of verbs if possible, especially if advanced semantic resources like a wordnet are available, as they require supervised learning to be leveraged.

In reproducing the work of Schulder et al. (2017), we limited ourselves to verbs. In the future, we would like to investigate methods to extend the shifter lexicon to also cover nouns and adjectives.

While we have shown that the same approach for classifying verbal shifters works for German and English, future work will expand the number of languages, especially to verify that these methods can also be applied to non-Indo-European languages, such as Chinese, Japanese or Arabic. In this context it will also be interesting to see whether using shifter lexicons from several languages can further improve the dictionary and cross-lingual word embedding classifiers.

Acknowledgements

The authors would like to thank Stephanie Köser for annotating the German gold standard lexicon presented in this paper. For proofreading the paper the authors would also like to thank Meaghan Fowlie and David M. Howcroft.

The authors were partially supported by the German Research Foundation (DFG) under grants RU 1873/2-1 and WI 4204/2-1.

References

Eneko Agirre and Aitor Soroa. 2009. Personalizing PageRank for Word Sense Disambiguation. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 33–41, Athens, Greece.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 451–462, Vancouver, Canada.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 86–90, Vancouver, Canada.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetti. 2009. The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-Crawled Corpora. *Language Resources and Evaluation*, 43(3):209–226.

Laurel J. Brinton. 1985. Verb Particles in English: Aspect or Aktionsart. *Studia Linguistica*, 39:157–68.

Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Padó, and Manfred Pinkal. 2006. The SALSA Corpus: a German corpus resource for lexical semantics. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 969–974, Genoa, Italy.

Yoonjung Choi, Lingjia Deng, and Janyce Wiebe. 2014. Lexical Acquisition for Opinion Inference: A Sense-Level Lexicon of Benefactive and Malefactive Events. In *Proceedings of the Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA)*, pages 107–112, Baltimore, Maryland, USA.

Simon Clematide and Manfred Klenner. 2010. Evaluation and Extension of a Polarity Lexicon for German. In *Proceedings of the Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA)*, pages 7–13, Lisbon, Portugal.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, and Richard Ducott. 2009. Without a 'doubt'? Unsupervised Discovery of Downward-Entailing Operators. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (HLT/NAACL)*, pages 137–145, Boulder, Colorado, USA.

Lingjia Deng, Yoonjung Choi, and Janyce Wiebe. 2013. Benefactive/Malefactive Event and Writer Attitude Annotation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 120–125, Sofia, Bulgaria.

Federico Fancellu, Adam Lopez, and Bonnie Webber. 2016. Neural Networks for Negation Scope Detection. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 495–504, Berlin, Germany.

Anastasia Giannakidou. 2008. Negative and Positive Polarity Items: Licensing, Compositionality and Variation. In Claudia Maienborn, Klaus von Heusinger, and Paul Portner, editors, *Semantics: An International Handbook of Natural Language Meaning*, pages 1660–1712. Mouton de Gruyter.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. BilBOWA: Fast Bilingual Distributed Representations without Word Alignments. In *International Conference on Machine Learning (ICML)*, pages 748–756, Lille, France.

William L Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing Domain-Specific Sentiment Lexicons from Unlabeled Corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 595–605, Austin, Texas, USA.

Birgit Hamp and Helmut Feldweg. 1997. GermaNet - a Lexical-Semantic Net for German. In *Proceedings of the ACL Workshop on Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 9–15, Madrid, Spain.

Sanda Harabagiu, Andrew Hickl, and Finley Lacatusu. 2006. Negation, Contrast and Contradiction in Text Processing. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 755–762, Boston, Massachusetts, USA.

Verena Henrich, Erhard Hinrichs, and Tatiana Vodolazova. 2014. Aligning Germanet Senses with Wiktionary Sense Definitions. In Zygmunt Vetulani and Joseph Mariani, editors, *Human Language Technology Challenges for Computer Science and Linguistic (LTC)*, pages 329–342. Springer.

Y. Huang and H. J. Lowe. 2007. A Novel Hybrid Approach to Automated Negation Detection in Clinical Radiology Reports. *Journal of the American Medical Infomatics Association*, 14:304–311.

D. Ikeda, H. Takamura, L. Ratinov, and M. Okumura. 2008. Learning to Shift the Polarity of Words for Sentiment Classification. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*, pages 296–303, Hyderabad, India.

Nitin Jindal and Bing Liu. 2008. Opinion Spam and Analysis. In *Proceedings of the International Conference on Web Search and Data Mining (WSDM)*, pages 219–230, Palo Alto, California, USA.

W. Kessler and H. Schütze. 2012. Classification of Inconsistent Sentiment Words using Syntactic Constructions. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 569–578, Mumbai, India.

Manfred Klenner, Angela Fahrni, and Stefanos Petrakis. 2009. PolArt: A Robust Tool for Sentiment Analysis. In *Proceedings of the Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 235–238, Odense, Denmark.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of Workshop at International Conference on Learning Representations (ICLR)*, Scottsdale, Arizona, USA.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1990. Introduction to WordNet: An On-line Lexical Database. *International Journal of Lexicography*, 3:235–244.

R. Morante and W. Daelemans. 2009. A Metalearning Approach to Processing the Scope of Negation. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, pages 21–29, Boulder, CO, USA.

R. Morante. 2010. Descriptive Analysis of Negation Cues in Biomedical Texts. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 1429–1436, Valletta, Malta.

Peter Prettenhofer and Benno Stein. 2010. Cross-Language Text Classification using Structural Correspondence Learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1118–1127, Uppsala, Sweden.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as Contrast between a Positive Sentiment and Negative Situation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 704–714, Seattle, Washington, USA.

Josef Ruppenhofer and Jasper Brandes. 2015. Extending effect annotation with lexical decomposition. In *Proceedings of the Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA@EMNLP)*, pages 67–76, Lisboa, Portugal.

Olivia Sanchez-Graillet and Massimo Poesio. 2007. Negation of protein–protein interactions: analysis and extraction. *Bioinformatics*, 23(13):i424–i432.

Helmut Schmid. 1994. Probabilistic Part-of-Speech Tagging using Decision Trees. In *Proceedings of the International Conference on New Methods in Language Processing (NeMLaP)*, pages 44–49, Manchester, United Kingdom.

Nathan Schneider, Dirk Hovy, Anders Johannsen, and Marine Carpuat. 2016. SemEval-2016 Task 10: Detecting Minimal Semantic Units and their Meanings (DiMSUM). In *Proceedings of the International Workshop on Semantic Evaluation (SemEval@NAACL-HLT)*, pages 546–559, San Diego, California, USA.

Marc Schulder, Michael Wiegand, Josef Ruppenhofer, and Benjamin Roth. 2017. Towards Bootstrapping a Polarity Shifter Lexicon using Linguistic Features. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*, pages 624–633, Taipei, Taiwan.

Marc Schulder, Michael Wiegand, Josef Ruppenhofer, and Stephanie Köser. 2018. Introducing a Lexicon of Verbal Polarity Shifters for English. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 1393–1397, Miyazaki, Japan.

Rico Sennrich, Gerold Schneider, Martin Volk, and Martin Warin. 2009. A New Hybrid Dependency Parser for German. In *Proceedings of the German Society for Computational Linguistics and Language Technology (GSCL)*, pages 115–124, Potsdam, Germany.

R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. 2013. Recursive Deep Models for Semantic Compositionality over a Sentiment Treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642, Seattle, Washington, USA.

G. Szarvas, V. Vincze, R. Farkas, and J. Csirik. 2008. The BioScope Corpus: Annotation for Negation, Uncertainty and their Scope in Biomedical Texts. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing (BioNLP@ACL-HLT)*, pages 38–45, Columbus, Ohio, USA.

Peter Turney. 2002. Thumbs up or Thumbs down?: Semantic Orientation Applied to Unsupervised Classification of Reviews. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 417–424, Philadelphia, Pennsylvania, USA.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual Models of Word Embeddings: An Empirical Comparison. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1661–1670, Berlin, Germany.

Leonid Velikovich, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan McDonald. 2010. The Viability of Web-derived Polarity Lexicons. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (HLT/NAACL)*, pages 777–785, Los Angeles, California, USA.

Michael Wiegand, Alexandra Balahur, Benjamin Roth, Dietrich Klakow, and Andrés Montoyo. 2010. A Survey on the Role of Negation in Sentiment Analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing (NeSp-NLP)*, pages 60–68, Uppsala, Sweden.

Theresa Wilson, Paul Hoffmann, Swapna Somasundaran, Jason Kessler, Janyce Wiebe, Yejin Choi, Claire Cardie, Ellen Riloff, and Siddarth Patwardhan. 2005. OpinionFinder: A System for Subjectivity Analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP) on Interactive Demos*, pages 34–35, Vancouver, Canada.

H. Yu, J. Hsu, M. Castellanos, and J. Han. 2016. Data-driven Contextual Valence Shifter Quantification for Multi-Theme Sentiment Analysis. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*, pages 939–948, Indianapolis, Indiana, USA.

B. Zou, G. Zhou, and Q. Zhu. 2013. Tree Kernel-based Negation and Speculation Scope Detection with Structured Syntactic Parse Features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 968–976, Seattle, Washington, USA.

Part-of-Speech Tagging on an Endangered Language: a Parallel Griko-Italian Resource

Antonis Anastasopoulos[♠] **Marika Lekakou**[♢] **Josep Quer**[♣] **Eleni Zimianiti**[♢]
Justin DeBenedetto[♠] **David Chiang**[♠]
[♠]Department of Computer Science and Engineering, University of Notre Dame
[♢]Department of Philology, University of Ioannina
[♣]ICREA & Department of Translation and Language Sciences, Universitat Pompeu Fabra
aanastas@nd.edu, mlekakou@cc.uoi.gr, josep.quer@upf.edu

Abstract

Most work on part-of-speech (POS) tagging is focused on high resource languages, or examines low-resource and active learning settings through simulated studies. We evaluate POS tagging techniques on an actual endangered language, Griko. We present a resource that contains 114 narratives in Griko, along with sentence-level translations in Italian, and provides gold annotations for the test set. Based on a previously collected small corpus, we investigate several traditional methods, as well as methods that take advantage of monolingual data or project cross-lingual POS tags. We show that the combination of a semi-supervised method with cross-lingual transfer is more appropriate for this extremely challenging setting, with the best tagger achieving an accuracy of 72.9%. With an applied active learning scheme, which we use to collect sentence-level annotations over the test set, we achieve improvements of more than 21 percentage points.

1 Introduction

Most natural language processing (NLP) applications have been developed for and tested on only a handful of languages. The majority of the world's languages are under-represented in the field, mostly due to the lack of proper resources. Endangered languages pose additional problems, as the lack of resources is further exacerbated by the lack of standard orthography. Therefore, there is an obvious need for both resources and technologies adapted to languages of under-represented communities. Especially in the case of endangered languages, computational approaches can be used to scale and accelerate documentation and revitalization efforts.

For the purposes of this study, we focus on Griko, a Greek dialect spoken in southern Italy, in the Grecìa Salentina area southeast of Lecce.[1] There is another endangered Italo-Greek variety in southern Italy spoken in the region of Calabria, known as Grecanico or Greco. Both languages, jointly referred to as *Italiot Greek*, were included as seriously endangered in the UNESCO *Red Book of Endangered Languages* in 1999. Griko is only partially intelligible with modern Greek, and unlike other Greek dialects, it uses the Latin alphabet. Less than 20,000 people (mostly people over 60 years old) are believed to be native speakers (Horrocks, 2009; Douri and De Santis, 2015), a number which is quite likely an overestimation (Chatzikyriakidis, 2010).

In general, the lack of annotated resources can be addressed through several directions. The obvious first one is the collection of human annotations, an expensive and time-consuming process. Another option is to collect additional monolingual data and use them in weakly-supervised approaches. Finally, one could use methods that transfer knowledge across languages, in which case they could focus on collecting translations or bilingual data. With this paper we attempt to quantify how effective each of these directions can be at the beginning of building applications for an endangered language.

We present a corpus of Griko narratives, first collected in the beginning of the 20th century, along with their Italian translations, suitable for computational research on the language.[2] We focus on the task

[1] A discussion on the possible origins of Griko can be found in the paper by Manolessou (2005).

[2] Our corpus is available for download here: https://bitbucket.org/antonis/grikoresource

Proceedings of the 27th International Conference on Computational Linguistics, pages 2529–2539
Santa Fe, New Mexico, USA, August 20-26, 2018.

	Stories	Sentences	Griko		Italian	
			Types	Tokens	Types	Tokens
train	104	9.2k	13.5k	197.6k	10.6k	169.7k
test	10	885	2.4k	14.0k	2.3k	13.1k
all	114	10.1k	14.1k	211.6k	11.0k	182.7k

Table 1: Statistics on our collected Griko-Italian resource.

of part-of-speech (POS) tagging, an important subcomponent of many downstream NLP applications, although our long-term goal is to annotate the whole corpus with morphosyntactic tags. Three linguists were tasked with creating gold-standard annotations for a portion of the resource, which we use as a test set in our experiments. Subsequently, our corpus includes human annotations as well as monolingual data and translations, allowing us to explore all approaches.

Based on another small Griko corpus with POS annotations, we explore computational methods for annotating our collected resource with POS tags. We evaluate several commonly used models for POS-tagging. We start with a traditional feature-based cross-entropy tagger, a Conditional Random Field (CRF) tagger, and a neural bi-LSTM tagger. We further experiment with methods that take advantage of additional monolingual data, and with methods that project cross-lingual tags through alignments. In addition, we are the first to combine the two latter approaches for POS-tagging in a low-resource setting, and we show that their combination achieves higher accuracy. Finally, we study how additional human annotations can be incorporated through an applied active learning scenario. In line with previous work, we show that this greatly improves the tagging accuracy.

Our contributions are three-fold: first, we aim to provide coarse insights into what type of annotations and resources are more effective at the early stages of developing a tagging tool for an endangered language. Second, we hope that the release of our resource will spark further interest in the computational community, so that previous, and future, methods are tested in actual endangered language settings. Finally, we benchmark the accuracy of several models on our dataset, and test them in a traditional setting and in a *transduction* setting where we also use the translations of the test set. We also evaluate all methods in an active learning scenario, showing that different approaches are suitable for different amounts of annotated data.

When only 360 annotated sentences are available, the best method is the one that combines them with both cross-lingual projections and monolingual data, achieving an accuracy of 72.9%. As the active learning scheme unfolds, however, we show that there is no need for either semi-supervised nor transfer learning approaches: a simpler feature-based CRF model achieves the highest scores, with more than 94% accuracy.

2 Resource

Resources in Griko are very scarce. The German scholar Gerhard Rohlfs pioneered research on Griko and composed the first grammar of the language (Rohlfs, 1977), also heavily influencing the subsequent grammar created by Karanastasis (1997). Although the language has been further studied, almost no corpora are available for linguistics research.

The only Griko corpus available online[3] (Lekakou et al., 2013) consists of about 20 minutes of speech in Griko, along with text translations into Italian. The corpus (henceforth `UoI` corpus, as it is hosted at the University of Ioannina, Greece) consists of 330 mostly elicited utterances by nine native speakers, annotated with transcriptions, morphosyntactic tags, and glossing in Italian.

The most noted Griko scholar is Vito Domenico Palumbo (1854–1928) who made the first serious attempts to create a literary Griko for the dialect of Calimera (the most populous of the nine remaining communities where Griko is still spoken), based on modified Italian orthography. Salvatore Tommasi and

[3] `http://griko.project.uoi.gr`

tag	frequency	tag	frequency	tag	frequency
V *(verb)*	24.4	Prt *(particle)*	2.2	Adv+Adv	0.4
PUNCT *(punctuation)*	18.3	P+D	1.8	X *(other)*	0.3
Pr *(pronoun)*	12.5	P *(adposition)*	1.6	V+Pr	0.3
N *(noun)*	11.6	Adj *(adjective)*	1.2	P+P	0.1
C *(complementizer)*	11.4	Num *(numeral)*	0.7	Pr+Pr	0.1
D *(determiner)*	7.2	N+Pr	0.6	C+Pr	0.1
Adv *(adverb)*	5.0	V+C	0.4		
Adv+P, Adv+Pr, Adv+Prt, Adj+Pr, Prt+N, Prt+Pt, D+N, Adv+Adv+Prt, Adv+Adv+Pr					< 0.1

Table 2: List of tags and their frequency in the annotated test part of the corpus.

Salvatore Sicuro then edited and published Palumbo's manuscripts (Palumbo, 1998; Palumbo, 1999), a part of which we now make available for computational and linguistic research.

After scraping from their website[4] 114 narratives that Palumbo had collected, along with their Italian translations, we removed all HTML markup and normalized the orthography: we substituted all curly quotes and apostrophes with simple ones, and substituted the vowels with circumflex (â, ô, û) that were used in a few contractions with the more common accented vowel–apostrophe combination (à', ò' ù'). Using the Moses tools (Koehn et al., 2007) with the Italian settings, we lowercased and tokenized our parallel dataset. For completeness purposes, we also make available the untokenized and proper-case versions of the corpus. The statistics of the resource are shown in Table 1.

We chose the first 10 narratives to be our test set, as they correspond to about 10% of all sentences. The rest of the narratives are treated as a monolingual or parallel resource to be leveraged. The test set was, in addition, hand-annotated by linguists: they corrected any tokenization errors that were introduced by the automatic process (for example, regarding the use of the apostrophe) and produced POS tags for every test sentence.

For every narrative, one of the linguists was presented with the produced output of the tagger, and proceeded to correct it. In order to ensure the quality of the annotations, a second linguist was then presented with the result of the work of the first linguist and tasked with correcting it, until all disagreements were resolved. Although it significantly slowed down the annotation process, we hope that this scheme ensured the quality of our annotations.

2.1 Differences from previous Griko resources

Orthography Griko has never had a consistent orthography. The transcriptions in the `UoI` corpus are based on orthographic conventions found in the few textual resources such as the local magazine *Spìtta*, that closely follow conventions adopted in Italian, aiming to be familiar to the speakers of the language. This non-standardization of the orthography leads to variations in the transcription of the same words.

In addition, we find that the word segmentation in our collected narratives follows more the concept of a phonological word. As a result, words that are segmented in the `UoI` corpus, in our narratives are often fused in a single token. The most common case that also appears in both Italian and Greek, is the contraction of prepositions and subsequent articles, such as the Italian *alla* or the Greek *στη (sti)* 'to the.Fem'. Other examples of word fusion that is not permitted in either Italian or Greek but appear in our narratives are nouns and possessive pronouns, or adverbs with other adverbs or prepositions. A direct result of this phenomenon is that annotating such tokens with single POS tags does not capture all of the necessary information.

Therefore, we chose to annotate such words with multiple POS-tags, effectively making our tag dictionary the superset of the universal tagset. The final tags that appear in practice in our corpus, and their respective frequencies, are listed in Table 2. Examples of fused words and their glosses and associated tags are shown in Table 3.

[4] `https://www.ciuricepedi.it`

word:	*stì*	*mànassu*	*cikau*	*ènna*	*vàleti*	*pànuti*
morphemes:	*s[e]-tì*	*màna-su*	*ci-kau*	*è-na*	*vàle-ti*	*pànu-ti*
POS tag:	P+D	N+Pr	Adv+Adv	V+C	V+Pr	Adv+Pr
gloss:	to-the.Fem.SG	mother-your.SG	there-down	have-COMP	put-her	on-her
translation:	'to the'	'your mother'	'down there'	'will'	'put her'	'on her'

Table 3: Examples of fused types that receive multiple tags in our annotation. The first example is a common preposition-determiner contraction, while the second and last example denote the common fusion of pronouns that follow nouns or adverbs. Notice the doubled consonants in the second and fourth instance, due to *raddoppiamento fonosintattico*.

Phonosyntactic Gemination One important difference is that the UoI corpus explicitly annotates the phenomenon of *raddoppiamento fonosintattico* (phonosyntactic gemination, or doubling of the initial consonant of the word in certain contexts) with a hyphen that separates the two words. The transcriptions that we collected do not mark for this phenomenon. The two words are often fused into a single token, and the doubling is not always present. For example, both following types appear in our corpus: *aderfòmmu* and *aderfòmu* 'my brother'.

Furthermore, the UoI corpus also uses apostrophes to mark word boundaries within which the *raddoppiamento fonosintattico* takes place. The use of apostrophes in our collected narratives is more loose. They are used both to mark elision/apocope, stress, as well as what it seems to be instances of raddoppiamento fonosintattico. This poses further issues that are discussed in the next paragraph.

Code Switching There are three languages present in the region of Salento: the regional variety of Italian, the Italo-Romance dialect of Salentino, and Griko.[5] In modern day all members of the Griko community are bilingual or trilingual. The generations before the Second World War are considered to have been predominantly monolingual, and our narratives were collected at that time, around the beginning of the 20th century. However, elements of Salentino do appear in the narratives, either as passing words, or as full sentences, mostly in dialogue turns. Note that resources on Salentino are also extremely scarce if not non-existent.

In order to deal with such examples, we decided to distinguish two scenarios. Tag switching or intra-sentential switching instances were fully annotated. So, any Salentino words or phrases that appear *within* a Griko sentence, are used for training and evaluation. However, in the few cases where we encounter full sentences in Salentino, we opt to not use them for training or evaluation. Such sentences are marked with distinctive tags in the released corpus. Note that the UoI corpus does not include any non-Griko words or phrases. An extensive study of the code-switching phenomena that occur in our corpus is left for future work.

The following is an example of usage of a Salentino phrase (italicized) within a Griko sentence, taken from story 4. Note that there exists a Griko word for 'olive oil', namely *alài* or *alàdi*, as well words for 'good', namely *kalòn* or *brao*. However, the Salentino phrase *oju finu* 'fine oil' is chosen:

leo	ti	vastò	*oju*	*finu*
say-1SG	COMP	hold-1SG	oil	fine
V	C	V	N	Adj

'I say that I have good olive oil'

Tokenization The UoI corpus has been carefully crafted to make sure that word boundaries are clearly denoted by spaces or hyphens. This unfortunately is not the case in our collected narratives. The "loose" use of apostrophes complicates the work of the tokenizer. We chose to tokenize all apostrophes as a single token, except for the cases of known elisions that were present in previous corpora, such as the case of the conjunction *c' (ce)* 'and'. In addition, in the manually annotated test set, the linguists corrected any clear tokenization issues regarding the apostrophe.

[5] See (Golovko and Panov, 2013) for a broader overview of the linguistic diversity in the Salento area.

Stress Marking In the UoI corpus, all words with two or more syllables have a diacritic mark to indicate the location of stress. However, the resources that we collected are not consistent in the use of such a diacritic. Its use is, besides, not standardized and not well studied. Although in most cases such a diacritic is used, there are several instances of polysyllabic words that have no stress marks.

2.2 Metadata

We further provide as much information as possible for each narrative, in the form of metadata. This includes the original url of the narrative, the title of the narrative in Griko and its translation in Italian. Whenever they were reported (more than 95% of the narratives) we include the location where the narrative was collected, and we anticipate that further analysis could possibly reveal any regional variations. The vast majority of the stories were naturally collected in Calimera, the largest village and the center of the Griko community, but the resource also includes 10 stories collected in Martano, as well as stories collected in Corigliano and Martignano, two smaller villages. We also include information about the date that a story was collected, as well as the narrator of the story. There are a total of 37 different narrators, while the 10 stories from Martano were retrieved from anonymous manuscripts. There are also 11 stories where the narrator is not known. Two thirds of the stories were narrated by women, while 15% of the narrators were male. The oldest manuscript dates back to 1883, while the most recent story was collected in 1998. We hope that this additional information will further allow us to investigate morphosyntactic phenomena in relation to their temporal or location context, but this is left as future work.

3 Related Work

POS tagging is a very well studied problem; probabilistic models like Hidden Markov Models and Conditional Random Fields (CRF) were initially proposed (Lafferty et al., 2001; Toutanova et al., 2003), with neural network approaches taking over in recent years (Mikolov et al., 2010; Huang et al., 2015).

The use of parallel data for projecting POS tag information across languages was introduced by Yarowsky and Ngai (2001), and further improved at a large scale by Das and Petrov (2011) who used graph-based label propagation to expand the coverage of labeled tokens. Täckström et al. (2013) used high-quality alignments to construct type and token level dictionaries. In the neural realm, Zhang et al. (2016) used only a few word translations in order to train cross-lingual word embeddings, using them in an unsupervised setting. Fang and Cohn (2017), on the other hand, used parallel dictionaries of 20k entries along with 20 annotated sentences.

Most of the previous approaches are rarely tested on under-represented languages, with research on POS tagging for endangered languages being sporadic. Ptaszynski and Momouchi (2012), for example, presented an HMM-based POS tagger for the extremely endangered Ainu language, based on dictionaries, 12 narratives (yukar), using one annotated story (200 words) for evaluation. To our knowledge, no other previous work has extensively tested several approaches on an actual endangered language.

The lack of high quality annotated data led to approaches that attempt to use monolingual resources in a semi-supervised setting. Notably, Garrette and Baldridge (2013) used about 200 annotated sentences along with monolingual corpora improving the accuracy of an HMM-based model. They tested their model on two low-resource African languages, Kinyarwanda and Malagasy and they found that in this time-constrained scenario type-level annotation leads to slightly higher improvements than token-level annotation, increasing the accuracy of their taggers to slightly less than 80%. Similar conclusions were reached in Garrette et al. (2013): 4 hours of annotation are more wisely spent if annotating at the type-level, provided there exist additional raw monolingual data. This line of work adequately addressed the question of what labeled data are preferable when there is (exceptionally) restricted access to annotators.

However, language documentation neither is nor needs to be restricted to such minimal amounts of annotation work. In addition, recently proposed endangered language documentation frameworks (Bird et al., 2014b) advocate the collection of translations (Bird et al., 2014a) which render the resource interpretable. In the case of our resource, we argue that the translations are enough for providing type-level supervision. Possibly, this is only feasible because the two languages belong in the same family *and* have been in contact for centuries, so care needs to be taken with the application of this claim.

Model	Data		
	no transduction		*transduction*
	UoI	+clp	+clp-all
stanford	62.90	67.10	67.11
crf	57.79	59.12	59.26
crf-mod	67.52	62.89	66.50
neural	45.27	53.24	58.50
	UoI+mono	+clp	+clp-all
G&B	71.67	**72.92**	72.07

Table 4: The best performing model is the one that combines semi-supervised learning with cross-lingual projected tags (G&B+clp). All models except for crf-mod benefit from transfer learning through alignments (+clp). Transduction does not significantly affect performance, except for the neural model.

4 Part-of-Speech Tagging

First, we construct a mapping of the tags of the UoI corpus to the Universal Part-of-Speech tagset (Petrov et al., 2012). This mapping is available as part of the complementary material of our resource.

Starting with the tagged UoI corpus, we can use several methods to train a tagger, which we use as baselines. We use the Stanford Log-linear POS-tagger (Toutanova et al., 2003) (henceforth stanford), trained and tested with the default settings. We also test a simple feature-based CRF tagger (henceforth crf), using the implementation of the nltk toolkit (Bird and Loper, 2004). We extended the implementation to also use prefix and suffix features of up to 4 characters, along with bigram and trigram features.[6] We will refer to this method as crf-mod.

Finally, we also investigate the use of a simple neural model. It uses a single bi-LSTM layer to encode the input sentence, and it outputs tags after a fully connected layer applied on the output of the recurrent encoder, as was described in Lample et al. (2016). The model is implemented in DyNet[7] (Neubig et al., 2017), with input embedding and hidden sizes of 128, and output (tag) embedding size of 32. It is trained with the Adam optimizer with an initial learning rate of 0.0002 and for a maximum of 50 epochs. We select the best model based on the performance on a small dev set of 40 sentences that we sampled randomly from the training set.

The tagging performance of all methods is shown in the first column of Table 4. We find that the crf-mod model is the best baseline model. With such few data to train on, both the crf and the neural model do not perform well. The bi- and tri-gram features that the crf-mod model uses are very sparse, while the neural model has to deal with a very large number of unknown words, as discussed below in the Analysis subsection.

In line with previous work, we find that semi-supervised training achieves better results in such low-resource settings. We exploit all the narratives that we collected by treating them as an additional monolingual corpus, used in the framework proposed by Garrette and Baldridge (2013). This approach (henceforth G&B) significantly improves upon all baselines, achieving an accuracy of 71.67% in the test set, an improvement of more than 4 percentage points.

Cross-lingual projected tags So far, our results have not used the Italian translations of our resource. We can follow a procedure similar to the one of Täckström et al. (2013), and extract word alignments from the Griko-Italian parallel data of the training set. We use a pre-trained Italian tagger[8] in order to tag the Italian side, and we map those tags to the universal tagset. We can then project the tags of the Italian tokens to the aligned Griko ones.[9] For the cross-lingual projected tags, we found that in practice

[6] Our extensions will be submitted to the nltk codebase.

[7] We will make the code available online.

[8] http://elearning.unistrapg.it/TreeTaggerWeb/TreeTagger.html

[9] The type-level projections are also provided with the Supplementary Material.

type-level predictions work better, and thus we only report results with such models. The tags of the Italian side of our resource, the Griko-Italian alignments, and the cross-lingual POS projections on Griko types are available through the complementary material of our resource.

Augmenting the training set with the type-level projected tags (`clp` in Table 4), we achieve improvements for all models, except `crf-mod`. The `crf-mod` method uses sparser features and is more prone to errors due to the noise of the projections. The best performance is achieved when we combine the projected tags, as type-level supervision, with the `G&B` method that leverages monolingual data. Their combination achieves the best overall performance, with an accuracy of 72.9%, a significant improvement over all other methods. As far as we know, this is the first time that cross-lingual projected tags are combined with the method of Garrette and Baldridge (2013).

Transduction An additional approach that needs to be studied is the transductive approach. Since we have translations both for the training and the test sets, we can extract word alignments and project POS tags also for the test set. The results of the transductive approach using cross-lingual projected tags from all the data that we have are shown in the third column of Table 4 (under `clp-all`).

We find that most methods benefit from the transductive approach, with the `stanford` and `crf` methods exhibiting minimal improvements, while the `neural` method improves significantly by about 5 percentage points as now there are even less out-of-vocabulary words in the input. The `crf-mod` method improves over the `UoI+clp` version, but still does not surpass the `UoI` only version. The only method that does not benefit from the transduction setting is the `G&B` method, where the performance drops.

An additional transductive step that can be taken with the `G&B` method is to also add the test set as part of the monolingual data that it uses. However, including the test set in the monolingual data also resulted in a drop in performance. Using all monolingual data along with the train-only cross-lingual types (`clp`) leads to accuracy around 69.9% (a drop of 3 points from the best model), while using all monolingual data with `clp-all` leads to a drop of another 1.4 points, to an accuracy of only 68.5%, which however is still better than all other taggers. These accuracy drops are probably justifiable, since `G&B` was not developed under a transductive assumption.

Analysis It is worth noting that our choice of using combined tags for fused/contracted words means that our training sets, under all settings, do not contain all tags that we encounter in the test set. The tagset of the `UoI` corpus only had 14 tags (the 12 universal ones plus `P+D` and `C+Pr`), indicative of its small size. As more narratives were annotated, the size of the necessary tagset increased to the final 29. However, the additional tags that we had to use are rather rare and do not severely affect the performance of our models. The tags that are present in the `UoI` corpus in fact account for 96.7% of all target tags in the test set, a value that could be considered as a skyline for all methods.

The explanation of our models' performance lies in vocabulary coverage. The `UoI` corpus only includes 46.6% of the test set tokens (8.9% of the test set types). The augmented training set with type-level projections increases those numbers to 48.7% of test tokens and 14.8% of test types. Even though we restrict ourselves to high quality alignments,[10] we are able to project tags to 3870 types (3911 in the transductive scenario), an amount higher than the amount of tags that a trained linguist can produce within four hours of annotation (Garrette et al., 2013).

The `G&B` method deals with the vocabulary coverage issue by introducing a tag dictionary expansion as a first step. They use a label propagation algorithm —Modified Adsorption (Talukdar and Crammer, 2009)— in order to spread labels between related items. In our framework, the cross-lingual projected tags provide labels for a subset of the types, in a way similar that an annotator would, partially alleviating the difficulty of the method's first step. This leads to less noise in the created tag dictionary, leading to increased accuracy. Note that, out of the cross-lingual projected tags that correspond to types that appear in the test set (about 10% of the test set types, in the *no transduction* setting), more than 65% were correctly projected.

[10] An alignment is used if either its probability is 1, or its probability is higher than 0.9 *and* the frequency of both tokens is higher than 5. Relaxing those conditions leads to worse performance due to noise.

Iteration	Narrative	Best accuracy without AL	Best accuracy with AL	Δ	Accuracy on story 9	Best method
1	story-1	77.89	—	0.0	78.13	G&B+clp
2	story-8	72.76	78.48	5.72	82.12	
3	story-7	75.07	85.17	10.10	83.57	
4	story-10	70.88	79.98	9.10	85.08	
5	story-5	72.26	82.34	10.08	88.21	crf-mod+clp
6	story-4	74.03	86.30	12.27	90.32	
7	story-3	72.48	89.67	17.19	92.13	crf-mod
8	story-6	74.67	91.80	17.13	93.64	
9	story-2	70.78	92.67	21.89	94.17	
10	story-9	72.97	94.17	21.20	—	

Table 5: Tagging accuracy for each test narrative with and without active learning. We obtain significant improvements (shown in the Δ column) by adding each annotated narrative to the training set before retraining and tagging the next narrative. The last two columns further outline the improvements from active learning, showing performance on the last and longest narrative of the test set (story-9) in each iteration, also showing the best method in each iteration. The impact of using monolingual data and type-level cross-lingual projections disappears when more training data are available.

5 Active learning

We further explored the use of active learning while tagging our test set. Our active learning scheme is as follows: We first sorted the test set narratives according to length, and starting only with the UoI corpus, we trained all taggers, producing annotations for the first story of the test set. After the corrections on the annotation of each narrative were completed, it was added as gold training data and the taggers were re-trained. For each subsequent story, the linguists were provided with the output of the tagger that achieved the highest accuracy in the previous iteration.

The main reason why we decided to follow this narrative-level active learning scheme instead of collecting type-level annotations is that a noisy corpus is not very helpful for linguistics research; at least some part of the resource should have to be checked for quality and accuracy by hand. In addition, the translations of the narratives can provide such information, as we already showed in the previous section. As we expand the coverage of our POS annotations over the whole corpus, we will explore other methods for selecting the types or sentences to be annotated through an active learning scheme.

The results, per narrative, with and without active learning, in the order that they were annotated by our linguists (from the shortest narrative to the longest) are outlined in Table 5. The results for each narrative in the active learning scenario ("with AL" column) report the best performing model that is trained on the concatenation of the UoI corpus and all the stories that were annotated in previous iterations. It is clear that the performance of the taggers improved continuously, as we added more training data. This is further outlined by each iteration's tagging accuracy on story 9, the last and longest narrative of the test set. Of course, when a narrative is added in the training set, it is then excluded from the test set, and the performance is reported on the rest of the narratives.

All methods display notable improvement as we added the annotated narratives to the training set. The performance trends are outlined in Figure 1. Firstly, it is notable that as the training set increases, the advantage of the model of Garrette and Baldridge (2013) that leverages monolingual data diminishes, compared to our simpler crf-mod tagger, both with or without cross-lingual projected tags. Before the first iteration, the accuracy gap is 6.4 percentage points in favor of G&B. However, after adding around 4-5 narratives so that there are around 500 training sentences, our crf-mod+clp method surpasses the G&B method and keeps improving. This is also outlined by the dashed line in Table 5. As we add more training instances, the accuracy of the G&B method plateaus around 85% and does not improve further.

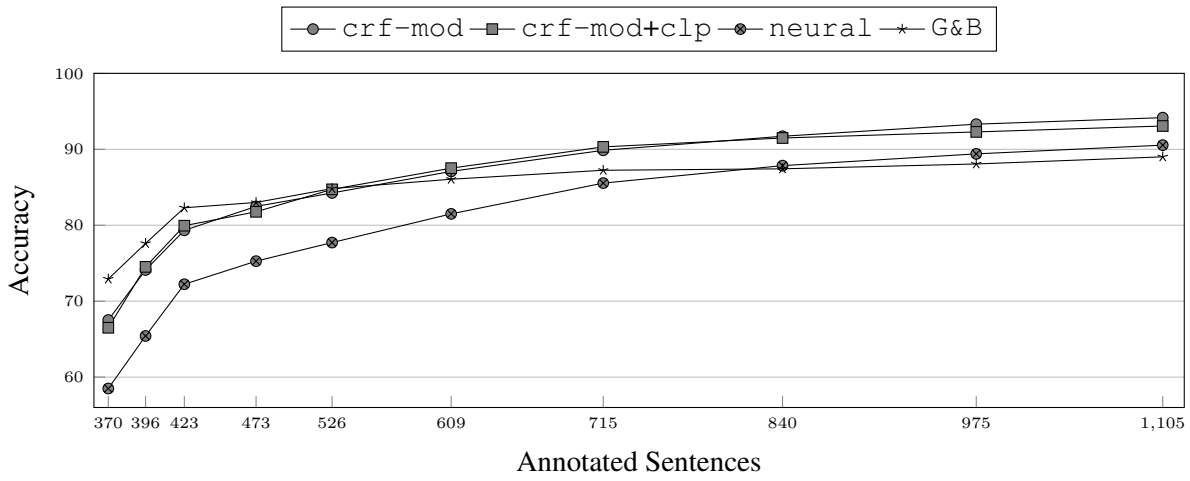


Figure 1: Accuracy on the (remaining) test set as we add annotated narratives to the training set. All methods benefit from the active learning approach, with G&B displaying better performance due to its use of monolingual data in the first iterations, but the crf-mod approach achieving the best results in the last iterations, eventually not even needing the cross-lingual type-level projections (+clp).

Furthermore, after a couple more iterations, when more than 800 annotated sentences are available for training, the crf-mod method without cross-lingual projected tags achieves higher accuracy than all others. We identify this point as the one where simple token-level supervision is efficient enough to outperform semi-supervised or transfer-learning approaches.

Finally, we observe that the accuracy of the neural bi-LSTM approach that only uses the tagged corpus without further use of monolingual data, improves significantly as the training set increases. With only 370 training sentences, the gap between the neural and the best method is more than 14 percentage points. With 1,100 training sentences, the accuracy gap diminishes to only 2 percentage points.

6 Cross-Validation

Our end goal is to annotate the whole corpus with POS tags, as well as richer annotations. Towards that direction, our gold annotated test data could be used to train a higher quality POS tagger, which we will use to annotate the rest of the corpus. In Section §5, we found that including all but one annotated narratives for training, and testing on the last one (story-9) we were able to obtain an accuracy of more than 94.17%. In order to get a better estimation of how well a tagger trained on our gold data would work, we perform a cross-validation experiment, using crf-mod, our best performing model.

For each cross-validation instance, one of the annotated narratives becomes the test set, and the rest will be included in the training set. This allows us to obtain an average performance over 10 instances. The average accuracy of the crf-mod model is about 91.9%, with a standard deviation of about 2 percentage points (minimum is 88.5% on story 5, and maximum is 94.9% on story 2).

The main obstacle to annotating the rest of the corpus with higher quality is out-of-vocabulary words. The combined vocabulary of the UoI corpus and our 10 annotated narratives covers 16% of the vocabulary of the 104 unannotated sentences (but 85% of the total tokens). As part of our future work, we plan to incorporate word-level active learning in our annotation/correction scheme, similar to the approaches proposed by Fang and Cohn (2017).

7 Conclusion

We presented a parallel corpus of 114 narratives on an endangered language, Griko, with translations in Italian. For now, a test set of 10 narratives is hand-annotated with Part-of-Speech tags, but in the future we will enrich the resource with annotations on the rest of the corpus, as well as with richer syntactic and morphological annotations. We also plan on contributing our corpus to the Universal Dependencies treebanks (Nivre et al., 2016) as Griko is absent from the supported languages.

We extensively evaluated several POS tagging approaches, and found that the method of Garrette and Baldridge (2013) can be combined with cross-lingual type-level projected tags, outperforming all other methods, with an accuracy of 72.9%, when less than 500 sentences are available. As data was added in the training set in an active learning scenario, a simple feature-based CRF approach outperforms all other models, with accuracy improvements of over 21 percentage points and over 94% accuracy on the last narrative. In fact, when more than 800 sentences are available for training, cross-lingual tag projections hurt performance.

The collected annotations from our test set could form the basis for training a high-accuracy POS tagger for Griko, so that we can expand the POS annotations to the rest of our corpus with only a small amount of noise. We aim to explore this direction in our future work, along with other active learning methods that require less human intervention. In addition, we plan to further enrich the annotations of our corpus with morphological tags similar to the `UoI` corpus, that will provide even more insight in Griko and its usage. When the full annotations of the corpus are completed, we plan to use statistical methods to study specific phenomena regarding the grammar and syntax of Griko.

Finally, and most importantly, we hope that the release of this corpus will spark further interest for computational approaches applied on endangered languages documentation and on under-represented languages in general.

Acknowledgements This work was generously partially supported by NSF Award 1464553. We are also grateful to the anonymous reviewers for their thougthful reviews and useful comments.

References

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proc. ACL*, page 31. Association for Computational Linguistics.

Steven Bird, Lauren Gawne, Katie Gelbart, and Isaac McAlister. 2014a. Collecting bilingual audio in remote indigenous communities. In *Proc. COLING*.

Steven Bird, Florian R. Hanke, Oliver Adams, and Haejoong Lee. 2014b. Aikuma: A mobile app for collaborative language documentation. In *Proc. of the 2014 Workshop on the Use of Computational Methods in the Study of Endangered Languages*.

Stergios Chatzikyriakidis. 2010. *Clitics in four dialects of Modern Greek: A dynamic account*. Ph.D. thesis, University of London.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proc. ACL*, pages 600–609. Association for Computational Linguistics.

Angeliki Douri and Dario De Santis. 2015. Griko and modern Greek in Grecìa Salentina: an overview. *L'Idomeneo*, 2015(19):187–198.

Meng Fang and Trevor Cohn. 2017. Model transfer for tagging low-resource languages using a bilingual dictionary. In *Proc. ACL*, pages 587–593. Association for Computational Linguistics.

Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *Proc. NAACL-HLT*, pages 138–147. Association for Computational Linguistics.

Dan Garrette, Jason Mielens, and Jason Baldridge. 2013. Real-world semi-supervised learning of pos-taggers for low-resource languages. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 583–592.

Ekaterina Golovko and Vladimir Panov. 2013. Salentino dialect, Griko and regional Italian: Linguistic diversity of Salento. *Working Papers of the Linguistics Circle of the University of Victoria*, 23(1):51.

Geoffrey Horrocks. 2009. *Greek: A History of the Language and its Speakers*. Wiley-Blackwell.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. arXiv:1508.01991.

Anastasios Karanastasis. 1997. *Grammatiki ton ellinikon idiomaton tis Kato Italias [Grammar of the Greek dialects of south Italy]*. Akadimia Athinon.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. ACL*, pages 177–180. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. NAACL-HLT*, pages 260–270.

Marika Lekakou, Valeria Baldiserra, and Antonis Anastasopoulos. 2013. Documentation and analysis of an endangered language: aspects of the grammar of Griko. `http://griko.project.uoi.gr`.

Ioanna Manolessou. 2005. The greek dialects of southern Italy: an overview. *KAMPOS: Cambridge Papers in Modern Greek*, 13:103–125.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proc. Interspeech*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. DyNet: The dynamic neural network toolkit. arXiv:1701.03980.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proc. LREC*.

Vito Domenico Palumbo. 1998. *Io' mia fora' - Fiabe e Racconti della Grecìa Salentina [Once upon a time - Fairy Tales and Stories from Grecìa Salentina]*. Calimera (LE): Ghetonìa. a cura di S. Tommasi.

Vito Domenico Palumbo. 1999. *'Itela na su pò - Canti popolari della Grecìa Salentina [I wanted to tell you - Folk songs of Grecìa Salentina]*. Calimera (LE): Ghetonìa. a cura di S. Sicuro.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proc. LREC*.

Michal Ptaszynski and Yoshio Momouchi. 2012. Part-of-speech tagger for Ainu language based on higher order Hidden Markov Model. *Expert Systems with Applications*, 39(14):11576–11582.

Gerhard Rohlfs. 1977. *Grammatica storica dei dialetti italogreci (Calabria, Salento) dt. Original [1949–1954] [Historical Grammar of the Italiot Greek dialects (Calabria, Salento)]*. CH Beck.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1:1–12.

Partha Pratim Talukdar and Koby Crammer. 2009. New regularized algorithms for transductive learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 442–457. Springer.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. NAACL-HLT*, pages 173–180. Association for Computational Linguistics.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual pos taggers and np brackets via robust projection across aligned corpora. In *Proc. NAACL*.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. 2016. Ten pairs to tag – multilingual pos tagging via coarse mapping between embeddings. In *Proc. NAACL-HLT*, pages 1307–1317. Association for Computational Linguistics.

One vs. Many QA Matching with both Word-level and Sentence-level Attention Network

Lu Wang[1, 2], **Shoushan Li**[1, 2, *], **Changlong Sun**[3], **Xiaozhong Liu**[3], **Luo Si**[3], **Min Zhang**[1,2], **Guodong Zhou**[1]

[1]NLP Lab, School of Computer Science and Technology, Soochow University, China
[2]Collaborative Innovation Center of Novel Software Technology and Industrialization
[3]Alibaba Group, China

[1]wanglu_1994@126.com, {lishoushan, minzhang, gdzhou}@suda.edu.cn
[3]{changlong.scl, xiaozhong.lxz, luo.si}@alibaba-inc.com

Abstract

Question-Answer (QA) matching is a fundamental task in the Natural Language Processing community. In this paper, we first build a novel QA matching corpus with informal text which is collected from a product reviewing website. Then, we propose a novel QA matching approach, namely One vs. Many Matching, which aims to address the novel scenario where one question sentence often has an answer with multiple sentences. Furthermore, we improve our matching approach by employing both word-level and sentence-level attentions for solving the noisy problem in the informal text. Empirical studies demonstrate the effectiveness of the proposed approach to question-answer matching.

1 Introduction

Question answer (QA) matching is a task to determine whether an answer is answering a given question. For instance, in Figure 1, the question "*Where is dear john filmed at?*" in **E1** has two candidate answers "*The movie was filmed in 2009 in Charleston.*" and "*The file was released on May 25, 2010 on DVD.*" The first answer is determined with the "*Matching*" category since it answers the question while the second answer is "*Non-matching*" since it could not answer the question. The past five years have witnessed a huge exploding interest in the research on QA matching, due to its widely applications, such as question answering (Yang et al., 2015; Tan et al., 2016; Wang et al., 2017) and reading comprehension (Trischler et al., 2016; Dhingra et al., 2017).

However, all existing QA matching studies only focus on formal text. In real applications, there exists many scenarios where the QA text is informal. For instance, **E2** is a question-answer pair ext-

E1: Two QA pairs in formal text	
Q_1: *Where is dear john filmed at?* Label: *Matching*	A_1: *The movie was filmed in 2009 in Charleston.*
Q_2:*Where is dear john filmed at?* Label: *Non-matching*	A_2: *The film was released on May 25, 2010 on DVD.*
E2: Three QA pairs in informal text	
Q: *Will the response time slow after updating os? What about the battery? What about the screen?*	A: *The response time is not a spark of slow, it has 4G RAM. But you must use it carefully, my phone's secreen has broke down.*
Q_1: *Will the response time slow after updating os?* Label: *Matching*	A_1: *The response ..... broke down.*
Q_2: *What about the battery?* Label: *Non-matching*	A_2: *The response ..... broke down.*
Q3: *What about the screen?* Label: *Matching*	A_3: *The response ..... broke down.*

Figure 1: Some QA examples with their matching labels

*Corresponding author.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2540–2550
Santa Fe, New Mexico, USA, August 20-26, 2018.

racted from the product reviewing platform "*Asking All*" in *Taobao*[1]. The QA text is informal where the question "*Will the response time slow after updating os? What about the battery? What about the screen?*" has more than one question and the answer contains multiple sentences which answer the first and the third sub-questions, leaving the second question un-solved. For simplicity, we split the question into three questions and generate three question-answer pairs, as shown in Figure1. In principle, for QA matching in the formal text, there are two main challenges:

First, for a particular sub-question, the answer contains multiple sentences, some of which do not answer this sub-question but other sub-questions. For instance, in **E2**, for the first question "*Will the response time slow after updating os?*", only the first sentence answers this question and the fourth sentence answers the third question "*What about the screen?*".

Second, the answer even might contain many uncorrelated sentences. For instance, in **E2**, the sentence "*It has 4G RAM*" and "*But you must use it carefully*" are uncorrelated with all three sub-questions. That is to say, there exits noisy information in some answers in informal text.

In this paper, we focus the research on QA matching with informal text. First of all, we build a novel QA matching corpus with informal text which contains many question-answer pairs from a product reviewing website. Then, we attempt to propose a novel QA matching approach for informal text. To deal with the first challenge above, we propose a novel matching approach, namely, One vs. Many Matching, to learn the matching measurement with one question sentence and multiple answer sentences. Specifically, we first adopt the BIMPM (Wang et al., 2017) to implement the matching measurement with one question sentence and one answer sentence, namely One vs. One Matching. Then, we learn the One vs. Many Matching representation by integrating all One vs. One Matching representations.

Furthermore, to deal with the second challenge, we introduce both the word-level and sentence-level attention mechanisms. Specifically, the word-level attention is applied in the One vs. One Matching learning process while the sentence-level attention is applied in the One vs. Many Matching learning process.

The remainder of this paper is organized as follows. Section 2 discusses the related work on QA matching. Section 3 presents data collection and annotation. Section 4 proposes our approach to QA matching. Section 5 reports the experimental results. Finally, section 6 gives the conclusion and our future work.

2 Related Work

2.1 Corpus construction

In previous studies for researching QA matching, there are mainly two corpora, namely TREC-QA and WikiQA. Specifically, TREC-QA is proposed by Wang et al. (2007) where the questions are a mixture of questions from both query logs and human editors, and the answers are selected from documents returned by past participating teams in TREC-QA tracks. Each question has at least one answer. WikiQA is proposed by Yang et al (2015). where the questions are sampled from real queries of *Bing* without editorial revision and the answers are from the summary section of a Wikipedia page of the topic. Different from TREC-QA, WikiQA has two-thirds questions with no matching answers.

Unlike these two corpora, this paper proposes a novel corpus for QA matching research where the question-answer pairs are real ones from "*Asking People*" in *Taobao*. Moreover, different from the above corpora, the question-answer pairs in our corpus are informal text.

2.2 Matching methods

Generally speaking, QA matching methods could be split into two categories: shallow learning methods and deep learning methods.

In shallow learning methods, some shallow learning algorithms, such as CRF, SVM and MaxEnt, are employed to train the learning models (Wang et al., 2010). Besides the learning algorithms, the related studies on shallow learning methods mainly focus on feature engineering, using linguistic tools and using external resources, such as lexical semantic resources (Yih et al., 2013), tree edit distance (Yao and Durme, 2013) and named entities (Severyn and Moschitti, 2013).

[1] https://www.taobao.com/

In deep learning methods, some neural network algorithms are employed to train learning models. Briefly, these methods could be categorized into three categories, i.e., siamense networks, attentive networks and compare-aggregate networks. In siamense networks, related studies use classic neural networks, such as LSTM and CNN, to get the representations separately and then concatenate them to classify. (Feng et al., 2015; Yang et al., 2015; Bowman et al., 2015). In attentive networks, instead of using the final time step of LSTM to represent a sentence, related studies use the attention strategy to get the weight of overall time steps and then use the weight to represent the sentence. (Tan et al., 2016; Hermann et al., 2015, Yin et al., 2015). In compare-aggregate networks, related studies use different matching strategy to get relationships within words. (He and Lin, 2016; Wang et al., 2017; Wang and Jiang, 2016; Trischler et al., 2016; Parikn et al., 2016.).

However, all above approaches are similar to our One vs. One Matching model which deals with the matching measurement between one sentence (or one piece of text) and another sentence (or another piece of text). In contrast, our approach is a One vs. Many Matching model which deals with the matching measurement between one sentence (or one piece of text) and multiple sentences (or multiple pieces of text).

3 Data Collection and Annotation

We collect 4,060 question-answer pairs from "*Asking All*" in *Taobao*, which is the most famous and biggest electronic business platform in China. The question-answer pairs are mainly from the *electronic* domain. Note that if a question contains multiple sub-questions, we split the question into several sub-questions and each sub-question and the whole answer is considered as a question-answer pair. For instance, as shown in Figure 1, the question in Example 2 contains three sub-questions, and we split it into three question-answer pairs.

To guarantee a high annotation agreement, we propose some annotation guidelines after several times of annotation processes on a limited size of data. Then, we ask more people to annotate the whole data set according to these annotation guidelines.

Generally, the two categories, "*Matching*" and "*Non-matching*" has following cases.

1) About the "*Matching*" category

(a) The answer directly answers the question. **E3** is an example of this case.
E3: Q: *Is the response time slow?*
A: *Slow.*

(b) The answer indirectly answers the question. **E4** is an example of this case.
E4: Q: *Is the response time slow?*
A: *Smooth.*

(c) The answer conditionally answers the question. **E5** is an example of this case.
E5: Q: *How long can the battery take?*
A: *If you don't play games, it can last two days.*

2) About the "*Non-matching*" category

(d) The answer is irrelevant with answer. **E6** is an example of this case.
E6: Q: *Is the microphone clear?*
A: *The battery life is decent. I like it.*

(e) The answer is about the product but doesn't talk about the aspect in the question. **E7** is an example of this case.
E7: Q: *Is the screen clear?*
A: *The quality of the screen is very good. It is a good cell phone.*

(f) The answer is an unknown response. **E8** is an example of this case.
E8: Q: *What about the performance of the phone?*
A: *I don't know, I bought it for my dad.*

For each question-answer pair, we assign two annotators to annotate the category, and the *Kappa* consistency check value of the annotation is 0.83. To deal with the question-answer pairs which are inconsistently annotated by two annotators, an expert is assigned to check them,

Category	Number
Matching	2505
Non-matching	1555

Table 1: Category distribution of the annotated data

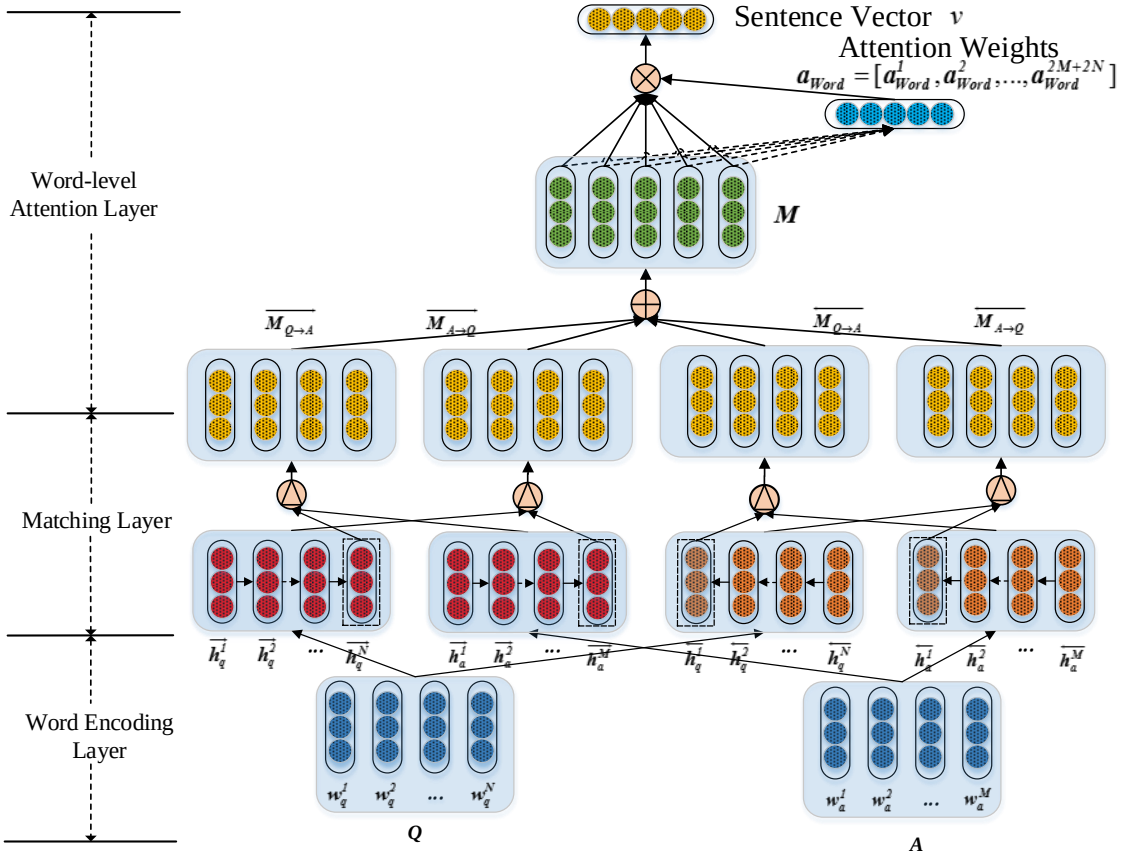


Figure 2: One vs. One Matching model with word-level attention

ensuring the quality of data annotation. Table 1 shows the category distribution of the corpus.

4 Our Approach

In this section, we propose our approach to QA matching in two steps. First, we propose the One vs. One Matching model which measures the matching between one sentence of the question text and one sentence in the answer text. Second, we propose the One vs. Many Matching model which measures the matching between one sentence of the question text and all sentences in the answer text.

4.1 One vs. One Matching Model

Figure 2 shows the overall architecture of our One vs. One Matching approach to QA matching. The involved layers are introduced in detail as following.

Word Encoding Layer. This layer has two inputs: the whole question sentence and one answer sentence. We represent the two sentences with *d*-dimensional vectors which is pre-trained with word2vec[2]. Then we utilize two bi-directional LSTM to encode the both of the sequences into contextual embeddings for each time step of the question and the sub answer sentence.

$$[\overrightarrow{h_q^1}, \overrightarrow{h_q^2}, ..., \overrightarrow{h_q^N}] = \overrightarrow{LSTM}([w_q^1, w_q^2, ..., w_q^N]) \quad (1)$$

$$[\overrightarrow{h_a^1}, \overrightarrow{h_a^2}, ..., \overrightarrow{h_a^M}] = \overrightarrow{LSTM}([w_a^1, w_a^2, ..., w_a^M]) \quad (2)$$

$$[\overleftarrow{h_q^1}, \overleftarrow{h_q^2}, ..., \overleftarrow{h_q^N}] = \overleftarrow{LSTM}([w_q^1, w_q^2, ..., w_q^N]) \quad (3)$$

$$[\overleftarrow{h_a^1}, \overleftarrow{h_a^2}, ..., \overleftarrow{h_a^M}] = \overleftarrow{LSTM}([w_a^1, w_a^2, ..., w_a^M]) \quad (4)$$

Matching Layer. This is the core layer in our One vs. One matching model. The goal of this layer is to compare the final contextual embedding (in the final time step of LSTM) of the sentence in the question against all contextual embeddings (in all time steps of LSTM) of one sentence in the answer. As

[2] https://code.google.com/archive/p/word2vec/

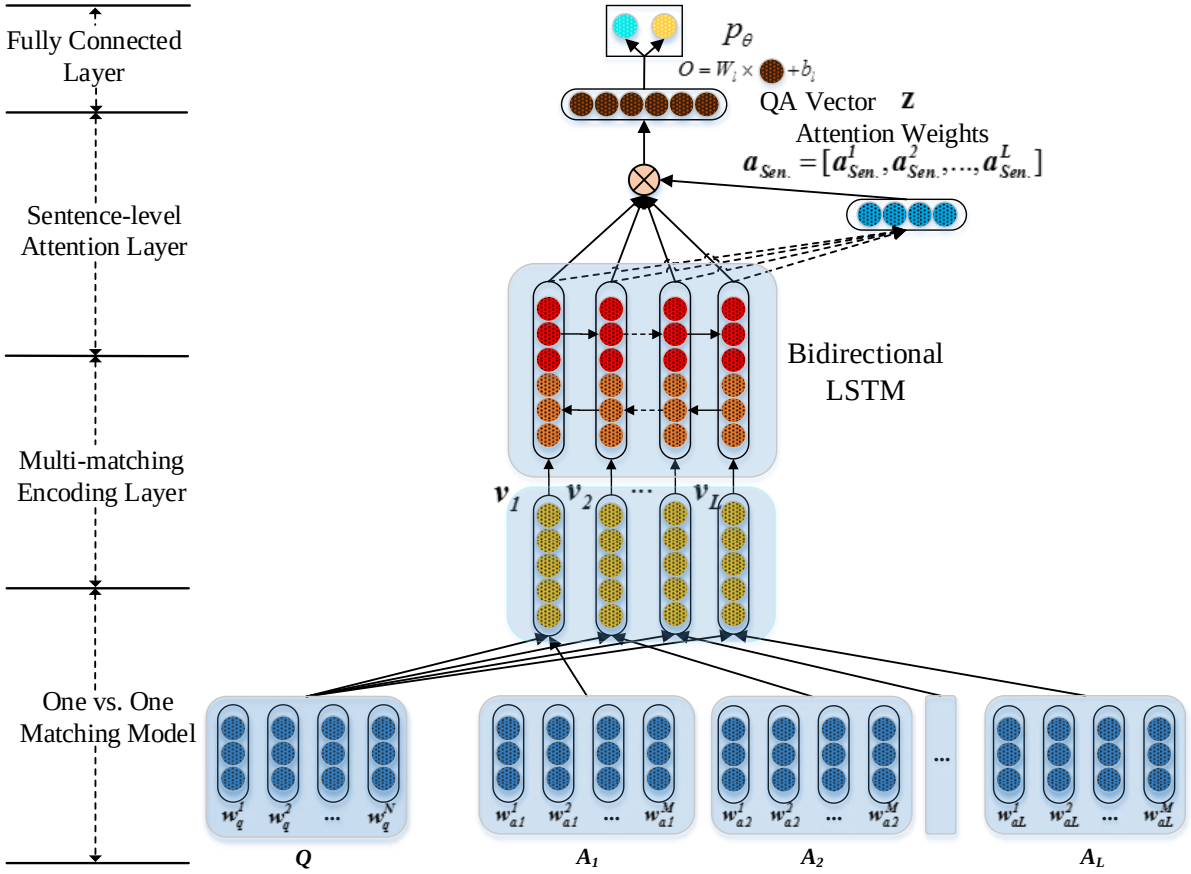


Figure 3: One vs. Many Matching model with sentence-level attention

shown in Figure 2, we match two sentences Q and A in two directions.

Formally, we use cosine function to calculate the similarity between two vectors, i.e.,

$$\overrightarrow{s_{q_N-a_i}} = cosine(\overrightarrow{h_q^N}, \overrightarrow{h_a^i}) \tag{5}$$

Inspired by Wang et al. (2017), we use the multi-perspective full matching strategy therein, i.e.,

$$\overrightarrow{s_{q_N-a_i}^k} = cosine(W^k \circ \overrightarrow{h_q^N}, W^k \circ \overrightarrow{h_a^i}) \tag{6}$$

where $\circ$ is the element-wise multiplication, and W^k is the k-th row of W, which controls the k-th perspective and assigns different weights to different dimensions of the dimension space.

Then, we obtain the similarity matrix $\overrightarrow{M_{Q\to A}} \in R^{P\times M}$, i.e.,

$$\overrightarrow{M_{Q\to A}} = \begin{bmatrix} \overrightarrow{s_{q_N-a_1}^1} & \cdots & \overrightarrow{s_{q_N-a_M}^1} \\ \vdots & \ddots & \vdots \\ \overrightarrow{s_{q_N-a_1}^P} & \cdots & \overrightarrow{s_{q_N-a_M}^P} \end{bmatrix} \tag{7}$$

where P means the number of perspectives.

Similarly, we get the matrices $\overleftarrow{M_{Q\to A}}$, $\overrightarrow{M_{A\to Q}}$ and $\overleftarrow{M_{A\to Q}}$.

Word-level Attention Layer. The goal of this layer is to assign weights to the similarity matrix and then get the most informative representation of the question sentence and one answer sentence.

First, we concatenate the $Q\to A$ and $A\to Q$ similarity matrix as the question-answer matrix.

$$M = \overrightarrow{M_{Q\to A}} \oplus \overleftarrow{M_{Q\to A}} \oplus \overrightarrow{M_{A\to Q}} \oplus \overleftarrow{M_{A\to Q}} \tag{8}$$

Second, we calculate the word attention weights a_{Word} with the following formula, i.e.,

$$a_{Word} = tanh(W_{Word} M + b_{Word}) \tag{9}$$

where W_{Word} is an intermediate matrix, b_{Word} is an offset value.

Third, we use a softmax function to normalize a_{Word}, i.e.,

$$a_{Word} = softmax(a_{Word}) \tag{10}$$

Finally, we multiply a_{Word} with M^T, and get the matching vector v, i.e.,

$$v = a_{Word} M^T \tag{11}$$

The matching vector v is the most informative representation of the question sentence and one answer sentence.

4.2 One vs. Many Matching Model

Figure 3 shows the overall architecture of our One vs. Many Matching approach to QA matching. The involved layers are introduced in detail as following.

One vs. One Matching Model. The purpose of this model is to use the question sentence and one answer sentence to calculate the most informative representation v_i. Suppose the answer contains L sentences, then we obtain the matching vectors between the question and answer, i.e., $V = [v_1, v_2, ..., v_L]$.

Multi-matching Encoding Layer. In this layer, we treat the above matching vectors as a sequence and employ a Bi-directional LSTM to learn a new representation, i.e.,

$$H = \overrightarrow{LSTM}(V) \oplus \overleftarrow{LSTM}(V) \tag{12}$$

Sentence-level Attention Layer. The goal of this layer is to get the most informative representation of the representation H.

First, we calculate the sentence attention weights $a_{Sen.}$, i.e.,

$$a_{Sen.} = tanh(W_{Sen.} H + b_{Sen.}) \tag{13}$$

where $W_{Sen.}$ is an intermediate matrix. $b_{Sen.}$ is an offset value.

Second, we use a softmax function to normalize $a_{Sen.}$, i.e.,

$$a_{Sen.} = softmax(a_{Sen.}) \tag{14}$$

Finally, we multiply $a_{Sen.}$ with H^T, and get the matching vector z, i.e.,

$$z = a_{Sen.} H^T \tag{15}$$

The matching vector z is the final representation for representing the matching measurement between the question and the whole answer.

Fully Connected Layer. The goal of this layer is to use the matching vector z to perform classification. Formally, we feed z to a softmax classifier, i.e.,

$$o = W_l z + b_l \tag{16}$$

where $o \in R^K$ is the output, W_l is the weight matrix and b_l is the bias. K is the number of categories. Then the probability of the category $k \in [1, K]$ is computed by:

$$p_\theta = \frac{exp(o_k)}{\sum_{t=1}^{K} exp(o_t)} \tag{17}$$

where θ denotes all parameters. Finally, the label with the highest probability stands for the predict label of the question-answer pair.

4.3 Model training

We use cross-entropy loss function to train our model end-to-end given a set of training data x_t, y_t, where x_t is the t-th question-answer pair to be predicted, and y_t is one-hot representation of the true category for x_t. We present this model as black-box function $\sigma(x)$ whose output is a vector representing the probability of categories. The goal of training is to minimize the loss function as following,

$$J(\theta) = -\sum_{t=1}^{N}\sum_{k=1}^{K} y_t^k \cdot \log\sigma(x_t) + \frac{l'}{2}\|\theta\|_2^2 \quad (18)$$

where N is the number of training samples and l' is a L_2 regularization to bias parameters.

We take Adam (Kingma and Ba, 2014) as the optimizing algorithm, and all the matrix and vector parameters are initialized with a uniform distribution in $\left[-\sqrt{6/(r+c)}, \sqrt{6/(r+c)}\right]$, where r and c are the rows and columns of the matrix (Glorot and Bengio, 2010).

5. Experiment

In this section, we systematically evaluate the performance of our approach to QA matching.

5.1 Experiment settings

Data Settings: As introduced in Section 3, the data contains 4,060 question-answer pairs and we randomly split the data into a training set (80% in each category), and a test set (the remaining 20% in each category). We also aside 10% data from training data as development data which is used to tune the parameters in each learning algorithm.

Word Segmentation and Embeddings: The Jieba[3] segmentation tool is employed to segment all Chinese text into words and word2vec[4] is employed to pretrain word embeddings using the data set that contains 200,000 question-answer pairs from the *electronic* domain. The dimensionality of the word vector is set to be 100 and the window size is set to be 1.

Sentence Split: We run sentence splitting with the CoreNLP[5] tool.

Hyper-parameters: The hyper-parameters values in the model are tuned according to performance in the development set. The size of units of the Bi-directional LSTM in the One vs. One Matching model is set to be 100, and the size of units of the Bi-directional LSTM in the One vs. Many Matching model is set to be 50. The default number of the sentences in an answer is set to be 5. The number of matching perspectives is set to be 20. The batch size is set to be 64.

Evaluation Measurement and Significance Test: The performance is evaluated using standard precision (*P*), recall (*R*), F-score (*F*) and *accuracy*. Furthermore, *t*-test is used to evaluate the significance of performance difference between two approaches.

5.2 Baselines

For comparison, we implement following approaches to QA matching:

- **LSTM:** A QA matching approach belonging to the siamense network, which is proposed by Bowman et al (2015). This approach employs a LSTM layer to encode the inputs.
- **Shallow Convolutional Neural Network (SCNN):** A state-of-the-art QA matching approach belonging to the siamense network, which is proposed by Zhang et al (2016). for the task of implicit discourse relation recognition.
- **Attentive LSTM:** A state-of-the-art QA matching approach belonging to an attentive network, which is proposed by Tan et al (2016).
- **MULT:** A state-of-the-art QA-matching approach belonging to the compare-aggregate network, which is proposed by Wang and Jiang (2017).
- **Bilateral Multi-Perspective Matching (BIMPM):** Another state-of-the-art QA matching approach belonging to the compare-aggregate network, which is proposed by Wang et al (2017). We use two implements where BIMPM (Full) employs the full matching strategy and BIMPM (Ensemble) employs all of the four matching strategies.

5.3 Our Approaches

Our approaches to QA matching are implemented with four different ways, i.e.,

[3] https://pypi.python.org/pypi/jieba/
[4] https://radimrehurek.com/gensim/models/word2vec.html
[5] https://stanfordnlp.github.io/CoreNLP/

	Macro-F	***Accuracy***
LSTM	0.600	0.649
SCNN	0.591	0.651
Attentive LSTM	0.632	0.697
MULT	0.651	0.691
BIMPM(Full)	0.654	0.700
BIMPM(Ensemble)	0.660	0.704
OMM	0.664	0.708
OMM+WA	0.674	0.713
OMM+SA	0.676	0.716
OMM+WA+SA	**0.689**	**0.721**

Table 2: Overall performances of different approaches to QA matching

	Matching			*Non-matching*		
	P	*R*	*F*	*P*	*R*	*F*
LSTM	0.647	0.864	0.740	0.656	0.356	0.461
SCNN	0.666	0.854	0.748	0.602	0.341	0.435
Attentive LSTM	**0.722**	0.863	0.786	0.611	0.394	0.479
MULT	0.720	0.824	0.769	0.618	0.471	0.534
BIMPM(Full)	0.712	0.860	0.779	0.663	0.442	0.531
BIMPM(Ensemble)	0.715	0.864	0.783	0.673	0.449	0.538
OMM	0.716	0.870	0.786	0.685	0.450	0.543
OMM+WA	0.709	0.883	0.786	0.725	0.460	0.563
OMM+SA	0.718	0.874	**0.789**	0.708	0.470	0.565
OMM+WA+SA	0.712	**0.883**	0.788	**0.744**	**0.488**	**0.590**

Table 3: Performances of different approaches to QA matching in each category

- **One vs. Many Matching (OMM):** This is the implementation where neither word-level attention layer nor the sentence-level attention layer is employed.
- **OMM with Word-level Attention (OMM+WA):** This is the implementation where only the word-level attention layer is employed.
- **OMM with Sentence-level Attention (OMM+SA):** This is the implementation where only the sentence-level attention layer is employed.
- **OMM with both Word-level and Sentence-level Attention (OMM+WA+SA):** This is the implementation where both the word-level attention layer and the sentence-level attention layer are employed.

5.4 Results

Table 2 and Table 3 show the overall and detailed performances of all approaches to QA matching. From this table, we can see that all our four approaches perform better than all baseline approaches and the significance test with *t*-test shows that the improvements are statistically significant (*p*-value<0.05). From the result, we can see that our approach is extremely superior in the category of *Non-matching*. For instance, our approach OMM+WA+SA outperforms the baseline approach LSTM with an improvement of 0.129 in terms of F-score. Among all our four approaches, OMM+WA+SA performs best, which highlights the importance of employing attention strategy in QA matching.

5.5 Visualization of Attention

In order to get a better understanding of the attention model and validate whether this model can capture the key information of the answer sentences, we visualize the word-level attention layer and sentence-level attention layer according to the obtained attention weights a_{Word} and $a_{Sen.}$.

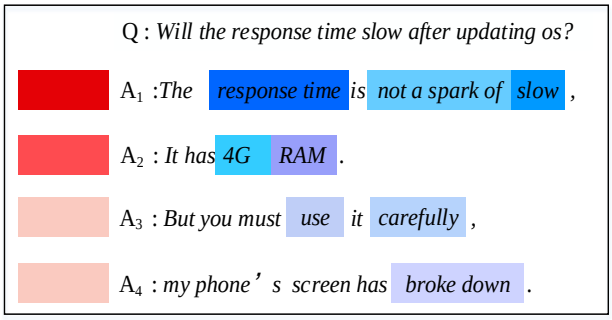


Figure 4: The attention visualization for a question answer pair

E9: Gold Label: *Non-matching*
Q: *屏幕清晰吗？*
（Is the screen clear?）
A: *屏幕质量很好。是个好手机。*
（The quality of the screen is good. It is a good phone.）
E10: Gold Label: *Matching*
Q: *有蓝牙吗？*
（Does it have bluetooth?）
A: *煤油，不要买。*
（Kerosene. Don't buy it.）
E11: Gold Label: *Matching*
Q: *系统是多少位的？*
（How many bits are there in the system?）
A: *64.*

Figure 5: Error Analysis

Figure 4 shows the attention visualizations for a question-answer pair. We normalize the word weight by the sentence weight to make sure that only informative words in informative sentences corresponding to the question are emphasized. In Figure 4, each line contains one sentence in an answer. Red denotes the sentence weight and blue denotes the word weight. The color depth indicates the importance degree of the attention weight for the question.

From the figure, we can see that the sentence-level attention function can select the informative sentences corresponding to the question, such as the first and the second sentences. In addition, the word-level attention function can select both the words and multi-word phrases carrying strong matching signals corresponding to the question, such as "*response time*", "*not a spark of*", and "*4G*".

5.6 Error Analysis

Through analyzing the classification results of our approach, we find that QA matching is still challenging and some kinds of errors are discussed as following.

First, the question and the answer share the same aspect of the product but the concerned contents of the aspect are different. For instance, in Figure 5 and **E9**, the question and the answer both have the word "*screen*". However, the question asks the clearness about the screen while the answer talks about the quality of the screen. Second, some words in the answer text have spelling mistakes. For instance, in Figure 5 and **E10**, the answer contains the word "*煤油* (*Kerosene*)" is a spelling mistake. It should actually be the word "*No*", although their pronunciations in Chinese are similar. Third, some question-answer pairs contain the answer sentence which is too short, like **E11** in Figure 5.

6. Conclusion

In this paper, we propose a new corpus for the research on QA matching in informal text. Moreover, we propose a novel approach to QA matching, namely One vs. Many Matching, to handle the difficulty in which the answer contains multiple sentences. Furthermore, we employ both the word-level and sentence-level attentions in our approach to further improve the matching performance. Empirical studies show that the proposed approach performs significantly better than several strong baseline approaches.

In our future work, we would like to enlarge the scale of the corpus by annotating some data in other domains. Furthermore, we would like to evaluate the effectiveness of our approach to QA matching in some other domains or some other languages.

Acknowledgments

The research work is partially supported by two National Nature Science Foundation of China (No.61751206 and No.61672366), the National Key R&D Program of China under Grand No.2017YFB1002104 and Collaborative Innovation Center of Novel Software Technology and Industrialization. This work is also supported by the joint research project of Alibaba and Soochow University.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 632-642.

Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang and Bowen Zhou. 2015. Applying Deep Learning to Answer Selection: A Study and An Open Task. *arXiv preprint arXiv:1508.01585.*

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249-256.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman and Phil Blunsom. 2015. Teaching Machines to Read and Comprehension. *arXiv preprint arXiv:1506.03340.*

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980.*

Hua He and Jimmy Lin. 2016. Pairwise Word Interaction Modeling with Deep Neural Networks for Semantic Similarity Measurement. In *Proceedings of NAACL-HLT 2016*. Association for Computational Linguistics, pages 937–948.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 2249-2255.

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic Feature Engineering for Answer Selection and Extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, pages 458-467.

Ming Tan, Cicero dos Santos, Bing Xiang and Bowen Zhou. 2016. Improved Representation Learning for Question Answer Matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, pages 464-473.

Adam Trischler, Zheng Ye, Xingdi Yuan, Jing He, Philip Bachman and Kaheer Suleman. 2016. A Parallel-Hierarchical Model for Machine Comprehension on Sparse Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 432-441.

Shuohang Wang and Jing Jiang. 2016. A Compare-Aggregate Model for Matching Test Sequences. *arXiv preprint arXiv:1611.01747.*

Mengqiu Wang and Christopher D. Manning. 2010. Probabilistic Tree-Edit Models with Structured Latent Variables for Textual Entailment and Question Answering. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, pages 1164-1172.

Zhiguo Wang, Wael Hamza and Radu Florian. 2017. Bilateral Multi-Perspective Matching for Natural Language Sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4144-4150.

Bingning Wang, Kang Liu and Jun Zhao. 2016. Inner Attention based Recurrent Neural Networks for Answer Selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, pages 1288-1297.

Yi Yang, Wen-tau Yih and Christopher Meek. 2015. WikiQA: A Challenge Dataset for Open-domain Question Answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, pages 2013-2018.

Xuchen Yao and Benjamin Ban Dueme. 2013. Answer Extraction as Sequence Tagging with Tree Edit Distance. In *Proceedings of NAACL-HLT 2013*. Association for Computational Linguistics, pages 858-867.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek and Andrzej Pastusiak. 2013. Question Answering Using Enhanced Lexical Semantic Models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, pages 1744-1753.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, Bowen Zhou. 2015. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *arXiv preprint arXiv:1512.05193.*

Biao Zhang, Jinsong Su, Deyi Xiong, Yaojie Lu, Hong Duan and Junfeng Yao. 2015. Shallow Convolutional Neural Network for Implicit Discourse Relation Recognition. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, pages 2230-2235.

Learning to Generate Word Representations using Subword Information

Yeachan Kim, Kang-Min Kim, Ji-Min Lee and SangKeun Lee
Korea University, Seoul, Republic of Korea
{yeachan,kangmin89,jm94318,yalphy}@korea.ac.kr

Abstract

Distributed representations of words play a major role in the field of natural language processing by encoding semantic and syntactic information of words. However, most existing works on learning word representations typically regard words as individual atomic units and thus are blind to subword information in words. This further gives rise to a difficulty in representing out-of-vocabulary (OOV) words. In this paper, we present a character-based word representation approach to deal with these limitations. The proposed model learns to generate word representations from characters. In our model, we employ a convolutional neural network and a highway network over characters to extract salient features effectively. Unlike previous models that learn word representations from a large corpus, we take a set of pre-trained word embeddings and generalize it to word entries, including OOV words. To demonstrate the efficacy of the proposed model, we perform both an intrinsic and an extrinsic task which are word similarity and language modeling, respectively. Experimental results show clearly that the proposed model significantly outperforms strong baseline models that regard words or their subwords as atomic units. For example, we achieve as much as 18.5% improvement on average in perplexity for morphologically rich languages compared to strong baselines in the language modeling task.

1 Introduction

Recently, distributed representations of words have become an inevitable component in systems of natural language processing (NLP), *e.g.,* chunking and named entity recognition (Turian et al., 2010), text classification (Kim, 2014), and question answering (Kumar et al., 2016). In such systems, word representations play a vital role by capturing useful information of words, leading to the improved performance of the systems. Most existing approaches to learning word representations are learned in an unsupervised learning manner from a large corpus and, for simplicity and effectiveness, they regard words as individual atomic units in a learning phase (Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013; Pennington et al., 2014).

However, word-based approaches typically ignore subword information in words. It prevents the approaches from taking into account the characteristic of words, *"words are lexically related when they share some morphemes or semantic elements"* (Williams, 1981). This constraint further makes representing out-of-vocabulary (OOV) words difficult. Even if lexically related words have been observed, word-based approaches fail to represent such words that are not included in a pre-defined vocabulary.

Perhaps the simplest way to sidestep this problem is to use one or more pseudo-words (*e.g.,* <unk>) to represent OOV words. However, this is not a fundamental remedy, as OOV words have their own meanings in many cases, such as variants, compounds, and misspellings. This problem is more evident when the domain of a text in which the word is learned differs from the domains to be applied to an NLP task. For example, the word *"talking"* and *"touchdown"* in the domain with generic text can be represented as *"tlking"* and *"touchdooown"* in the domain of social media. In this case, although these words share a similar sequence, representing words in other domain is difficult.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2551–2561
Santa Fe, New Mexico, USA, August 20-26, 2018.

In this paper, we present a character-based approach to generate word representations. Unlike previous works that adopt unsupervised learning from a corpus, the proposed model learns to generate word representations based on the supervised learning of pre-trained word embeddings. We use a convolutional neural network (CNN) (LeCun et al., 1998) to extract subword features over characters and utilize a highway network (Srivastava et al., 2015) to relate salient subword features to pre-trained word embeddings.

The proposed model has several benefits. As we regard characters as our atomic units for learning word representations, we have an ability to generate representations for all kinds of words, including OOV words. Moreover, we take a set of pre-trained word embeddings and generalize it to represent all words. This enables the proposed model to learn word representations efficiently without having to train with a large corpus.

To evaluate the proposed model, we perform an intrinsic and an extrinsic task which are word similarity and language modeling, respectively. We specifically conduct these tasks in various languages, as subword information is a characteristic of many languages such as French and Spanish (which have more than 40 different inflected forms for verbs). Experimental results show that the proposed model outperforms strong baselines that regard words or their subwords as atomic units, while representing OOV words quite well. In summary, we make the following contributions:

- We propose a character-based approach to generate word representations utilizing a convolutional neural network and a highway network.
- We leverage a set of pre-trained word embeddings and generalize it to all word entries. It allows us to efficiently learn word representations in a learning phase without training in a large corpus.
- We verify the efficacy of the proposed model in both an intrinsic and an extrinsic task with various languages. Our experimental results reveal that the word representations derived from our model considerably improve the performance on both of tasks in most languages.

The remainder of this paper is organized as follows. In Section 2, we discuss related works. In Section 3, we describe the proposed model. We report our performance evaluation results and analyze our methodology in detail in Sections 4 and 5, respectively. Finally, we conclude this paper in Section 6.

2 Related Works

Recently, many works have been proposed to improve the quality of word representations using subword[1] information. Some works used word and subword representations concurrently to produce better word representations. For example, (Qiu et al., 2014) proposed a model that jointly learns word and subword representations for word embeddings. They found that this joint model is useful in producing better representations for rare words. Similarly, (Mousa et al., 2013) proposed a deep neural network for language modeling, where the inputs to the network are a mixture of words and morphemes as well as their respective features. This work showed that utilizing morphemes could enhance lexical coverage and mitigate the problem of data sparseness for a n-gram language model.

In a similar manner, many works have proposed utilizing only subword information to produce word representations. For example, (Wieting et al., 2016) proposed embedding models for words and sentences using character n-gram count vectors. Although they require specific paragraph pairs for training, these models showed promising results in representing words while considering subword information. Recently, (Bojanowski et al., 2017) proposed FastText, which is an extension of the continuous skip-gram model (Mikolov et al., 2013). In this model, a word representation is replaced with the sum of character n-gram representations. This work showed that utilizing subword information allows the model to be trained efficiently by sharing the n-gram parameters between words. (Luong et al., 2013) utilized recursive neural networks in which inputs are morphemes of words. The proposed model showed better estimations of rare and complex words.

[1] We use the term "subword" for representing units smaller than a word, such as morphemes or character n-grams.

Some of the above-mentioned works (Luong et al., 2013; Mousa et al., 2013; Qiu et al., 2014), which used morphemes and normalized words, require segmentation tools for morphological segmentation. By contrast, we do not have to use segmentation tools as we use only characters to produce word representations. In addition, most works learn word representations from a large corpus using an unsupervised learning manner. However, we do not require training from a large corpus because we employ a set of pre-trained word embeddings.

Some works employed subword information, especially by using characters, to represent words in NLP tasks. For example, (Santos and Zadrozny, 2014) and (Ling et al., 2015) used a CNN and a recurrent neural network (RNN), respectively, over character representations for part-of-speech tagging. For a language modeling, (Kim et al., 2016) used a CNN and a highway network which achieved better performance than word-based models. That network shares a similar architecture with ours. However, our architecture is different from the early network (Kim et al., 2016) in that ours is designed with a simpler network architecture and a different output layer based on the pre-trained word embeddings. For neural machine translation, (Ruiz Costa-Jussà and Rodríguez Fonollosa, 2016) and (Luong and Manning, 2016) incorporated character representations to produce OOV word representations and achieved improved performance over models that used unknown tokens to represent OOV words.

Most closely related to our work is Mimick (Pinter et al., 2017), which is a bidirectional long short-term memory (LSTM)-based word representation model. It learns a function from characters to word embeddings to represent OOV words. In terms of the quality of OOV word representations, Mimick has shown promising results in syntactical properties. By contrast, we explore a CNN-based architecture to learn features from pre-trained word embeddings. As words consist of locally salient features such as prefixes, roots, and suffixes, a CNN architecture is expected to be effective at extracting these features.

3 Proposed Model

Figure 1 provides an overview of the proposed model. Our learning strategy is to *reconstruct* pre-trained word embeddings from character representations. To this end, we first extract subword features using a character-based convolution module. We then utilize a highway network to adaptively combine the subword features derived from the convolution module. In the optimization step, we make this resultant representation similar to its pre-trained word embedding.

3.1 Character-based Convolution Module

A CNN is designed to capture the most informative local aspects of a particular task (LeCun et al., 1998). CNNs have been widely used in not only computer vision but also recently in NLP (Kim, 2014; Cao and Lu, 2017) because of its ability to extract local context features such as n-grams. Accordingly, we apply this network on a character sequence to extract its subword features.

We first use a one-hot encoding to quantize the character inputs, which has a value of 1 at the index of the corresponding character, otherwise it has a value of 0. We concatenate these character representations to constitute a word, which is represented as $r \in \mathbb{R}^{|V_c| \times n}$ where V_c is a vocabulary of characters and n is the length of a word. Thus, the input representation for convolution has the following shape:

$$r = c_1 \oplus c_2 \oplus \cdots \oplus c_n = \begin{bmatrix} | & | & \cdots & | \\ c_1 & c_2 & \cdots & c_n \\ | & | & \cdots & | \end{bmatrix}$$

where c_i is a one-hot encoding for representing the i-th character in a word and $\oplus$ is a concatenation operator. We then insert zero-paddings to the side of these character representations r in order to perform the same number of convolutions for all characters. The number of zero-paddings on each side is $\frac{h-1}{2}$, where h is the width of convolution filter F. We perform the convolution operation on the padded representation r using the following equation:

$$s_i = tanh(F \cdot r_{i:i+h-1} + b) \tag{1}$$

where filter F is represented as $F \in \mathbb{R}^{|V_c| \times h}$ and $r_{i:i+h-1}$ is a sequence representation between character representation c_i to c_{i+h-1}. Then, we apply $tanh$ as the non-linear function to the convolution results.

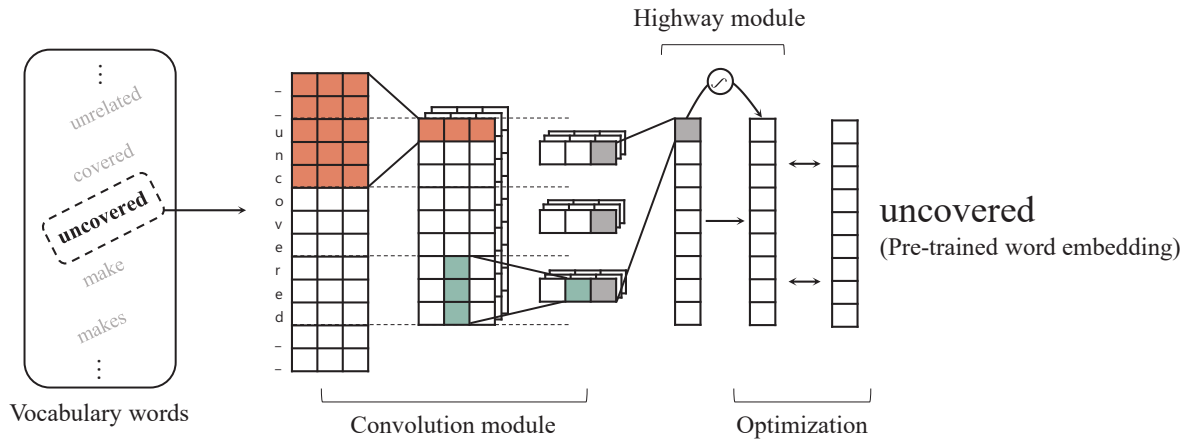


Figure 1: Overview of the learning phase in the proposed model. In this example, we process as input the word "*uncovered*" that is included in pre-trained word embeddings. In the convolution module, we use three filters with 5-width (thus, we add two zero-paddings) and other width filters, and we set the width of the stride to 3 for pooling. In the highway module, we use a single highway network. We then make this resultant representation similar to the pre-trained word embedding of "*uncovered*". After training converges, we can represent words that are lexically related to the input word but that are not included in pre-trained word embeddings, like "*uncovering*", as these words share similar character sequences.

The resultant vector s has the same length as the input word because of the zero-paddings. To extract salient features, we apply a max-pooling operation on the vector s. Specifically, we use max pooling with stride to derive locally salient features that can contain suffixes, roots, and prefixes. As a result, each feature derived from stride pooling has a summary of subword information in a character sequence. The equation for stride pooling is as follows:

$$e_i = \max\left[s_{k \cdot i : k \cdot (i+1)-1}\right] \tag{2}$$

where k is the length of a stride and $s_{k \cdot i : k \cdot (i+1)-1}$ is a sequence representation with one stride in s. As stride pooling results in features that vary in length according to the length of words, we therefore simply sum the elements of the vector e to produce fixed-sized features.

Thus far, we have described the procedure for a single filter. In our experiment, we use multiple filters with different widths to extract various subword features. We simply concatenate the results of each filter to merge the various features.

3.2 Highway Network Module

A highway network is an architecture that allows a model to train a very deep network efficiently (Srivastava et al., 2015). This network learns to route information using a gating mechanism. Recently, (Kim et al., 2016) utilized a highway network on a CNN for language modeling. It turned out that a highway network is helpful to a CNN, as it adaptively combines local features from individual filters. By following this line of work, we apply this network to adaptively combine subword features derived from a convolution module. It allows the proposed model to relate salient subword features with pre-trained word embeddings in a learning phase. The basic equation of a highway network is as follows:

$$y = T \odot H + C \odot e \tag{3}$$

where H is a non-linear transformation on its input e, T is the *transform gate* and C is the *carry gate*. In our work, we use C as a *(1-T)* for simplicity. For the function H and T, we use the following equations:

$$\begin{aligned} T &= \sigma(W_t \cdot e + b_t) \\ H &= \tanh(W_h \cdot e + b_h) \end{aligned} \tag{4}$$

where W_t, W_h and b_t, b_h are square matrices and biases[2], respectively. Finally, we set the size of resultant vector y as the pre-trained word embedding using a linear transformation for an optimization as follows:

$$w = W \cdot y + b \tag{5}$$

where w is a final word representation resulting from the proposed model, and W and b are parameters of linear transformation. Through the aforementioned procedures, we generate word representations while considering subword information.

3.3 Optimization with Pre-trained Word Embeddings

The proposed model reconstructs pre-trained word embeddings from characters. To make the word representation derived from our model learn the features of pre-trained word embeddings, we use the *squared Euclidean distance* as our objective function as follows:

$$E = \sum_{v \in V_w} \|w_v - \hat{w}_v\|^2 \tag{6}$$

where w_v is a word representation from our model and $\hat{w}_v$ is a pre-trained word embedding for a word v in the vocabulary. After training converges, we discard pre-trained word embeddings $\hat{w}$ and use w as our word representations for NLP tasks.

4 Experiments

To evaluate the proposed model, we perform both an intrinsic and an extrinsic task which are word similarity and language modeling, respectively. The word similarity evaluates the ability to capture the semantic similarity of words. The language modeling evaluates the usefulness of the proposed word representations in an NLP task. We specifically conduct both of tasks in various languages, as the subword information is a language-dependent feature.

4.1 Experimental Settings

We train the proposed model using pre-trained word embeddings. To simulate the settings having pre-trained word embeddings for various languages, we use word2vec (Mikolov et al., 2013), which is trained in Wikipedia[3] that was released on December 1, 2017 for seven languages: Arabic, Czech, German, English, Spanish, French, and Russian. The preprocessing for a corpus of each language is done using Natural Language Toolkit (NLTK) Tweet Tokenizer[4]. Based on (Lai et al., 2016), we assign to word vectors a 100 dimension, which has shown relatively better performance in both an intrinsic and an extrinsic task. In our experiments, we examine the following models:

- **word2vec** (Mikolov et al., 2013): This is a model that regards words as individual atomic units and adopts a window-based learning technique. Since this method is a word-based approach, it is incapable of representing OOV words. As a default for OOV words, we use null vectors for the word similarity task, and we randomly initialize and fine-tune them for the language modeling task. Of the two models of word2vec, CBOW and skip-gram, we use a skip-gram architecture for learning word representations, as it generally shows better performance than does CBOW (Lai et al., 2016).

- **FastText** (Bojanowski et al., 2017): This is an extension of word2vec and regards character n-grams as atomic units. The learning technique is similar to that of word2vec. We set the dimension of words to 100 that is same with other models and, for other settings, we follow the best settings described in the paper (Bojanowski et al., 2017).

[2]According to the suggestion of (Srivastava et al., 2015), we initialize b_t as -3 for making the networks initially biased towards *carry* behavior.

[3]https://www.wikipedia.org/

[4]https://www.nltk.org/

- **Mimick** (Pinter et al., 2017): This is a character-based model that learns a function from characters to word embeddings for representing words. For pre-trained word embeddings for this model, we use the previously mentioned skip-gram version of word2vec. In our experiments, we compare the best Mimick model, which uses pre-trained word embeddings for vocabulary words and generated embeddings for OOV words. We set the hidden dimension of an LSTM to 256 for better convergence, and other parameters are same as those of the paper (Pinter et al., 2017).

- **GWR** (proposed model): This is our proposed model, denoted as the generated word representation (GWR), and we use the word2vec for pre-trained word embeddings. We generate all words that are observed and use them for overall tasks.

We implemented the proposed architecture using TensorFlow frameworks (Abadi et al., 2016) and trained our model in a single machine equipped with Intel Core i5, 16GB of RAM, and an NVIDIA GeForce GTX 1080 with 8GB of RAM. The details of the training parameters are shown in Table 1. We chose these listed parameters by validating the word similarity task.

Type	value
optimizer	Adam (Kingma and Ba, 2014)
learning rate	0.001
batch size	50
the width of filter	[1,2,3,4,5,6,7]
the width of stride	3
# of filters	max(200, 50*width)
# of highway networks	2

Table 1: Experimental settings of the proposed model

4.2 Word Similarity

We first perform word similarity task to evaluate the ability of models to capture the semantic similarity among words. The performance of word similarity is measured by computing the Spearman's correlation coefficient between human judgment and the cosine similarity between word pairs.

4.2.1 Datasets

We collect word similarity datasets for six languages: Arabic (Ar), German (De), English (En), Spanish (Es), French (Fr), and Russian (Ru). For English, we compare the models on the WS353 (Finkelstein et al., 2001) and SL999 (Hill et al., 2014) datasets. For German, we conduct experiments on the GUR350 and ZG222 (Zesch and Gurevych, 2006) datasets, and we evaluate the models on the MC30 and WS353 datasets for Arabic and Spanish that are translated by (Hassan and Mihalcea, 2009). We evaluate French word vectors on the translated dataset RG65 (Joubarne and Inkpen, 2011), and for Russian, we evaluate the vectors on the HJ dataset introduced by (Panchenko et al., 2016).

4.2.2 Results

Table 2 shows the overall results. We first observe that subword-based learning methods (FastText, Mimick, GWR) show superior performance compared to that of the word-based method (word2vec) on most languages and datasets. This indicates that considering subword information is effective at representing words and useful with many languages. Among the subword-based methods, GWR outperforms Mimick in all languages except for English. We conjecture that this result can be explained by the fact that a CNN architecture is helpful in extracting local aspect features. In addition, GWR shows competitive performance with FastText that learns word representations from a large corpus (*i.e.,* GWR shows better performance in six out of nine datasets).

Language	Dataset	word2vec	FastText	Mimick	GWR$^-$	GWR
Ar	WS353	34.9	**52.1**	44.9	36.3	43.8
De	GUR350	34.6	48.7	47.6	35.4	**53.2**
	ZG222	16.2	35.4	36.5	16.4	**38.7**
En	SL999	29.5	27.7	**30.3**	28.6	29.8
	WS353	63.1	64.1	63.7	63.9	**64.6**
Es	MC30	40.6	**72.1**	66.8	41.4	69.8
	WS353	29.5	50.0	40.7	30.8	**50.2**
Fr	RG65	46.8	58.6	51.3	48.1	**60.5**
Ru	HJ	30.3	**60.1**	42.1	31.4	45.5

Table 2: Word similarity results on entire datasets (Spearman's $\rho \times 100$). GWR$^-$ implies a model using null vectors for OOV words. Best results are in bold.

To confirm that OOV word representations derived from GWR indeed improve the performance, we assess the performance of our model when used with null vectors for OOV words (GWR$^-$). The results show that the model generating OOV words (GWR) considerably outperforms the model that does not generate them (GWR$^-$). This improved performance indicates that the proposed model effectively expands the lexical coverage and generates representations of OOV words quite well. In addition, the GWR$^-$ model shows slightly better performance than that of the word2vec. It also shows its effectiveness at considering subword information in representing words. The differences between these two representations for vocabulary words are described in Section 5.2.

4.3 Language Modeling

As a second experiment, we perform language modeling as an extrinsic task to verify the usefulness of the proposed model when performing an NLP task. To evaluate the performance of word embeddings, we use an LSTM network with dropout regularization, which is used in (Bojanowski et al., 2017). Through the development set in (Botha and Blunsom, 2014), we set the hidden dimension of an LSTM to 256 and the probability of dropout to 0.5. In our experiments, we replace only the component of word embeddings to solely evaluate word embedding performance. The performance of language modeling is computed based on perplexity, which is a widely used metric in evaluating the usefulness of language modeling. Lower perplexity means a better model.

4.3.1 Datasets

We perform language modeling task using the datasets introduced by (Botha and Blunsom, 2014) on six languages: Czech (Cs), German (De), English (En), Spanish (Es), French (Fr), and Russian (Ru). Each dataset contains roughly one million tokens for training, and we follow the same pre-processing and data splits as reported in (Botha and Blunsom, 2014).

Language	$\lvert V_w \rvert$	word2vec	FastText	Mimick	GWR
Cs	46k	412.7	397.9	401.7	**352.1**
De	36k	263.8	248.2	258.6	**233.2**
En	17k	259.8	256.3	257.2	**240.7**
Es	27k	190.7	186.9	189.2	**174.7**
Fr	25k	205.2	207.7	202.4	**185.6**
Ru	62k	311.5	282.7	293.1	**243.1**

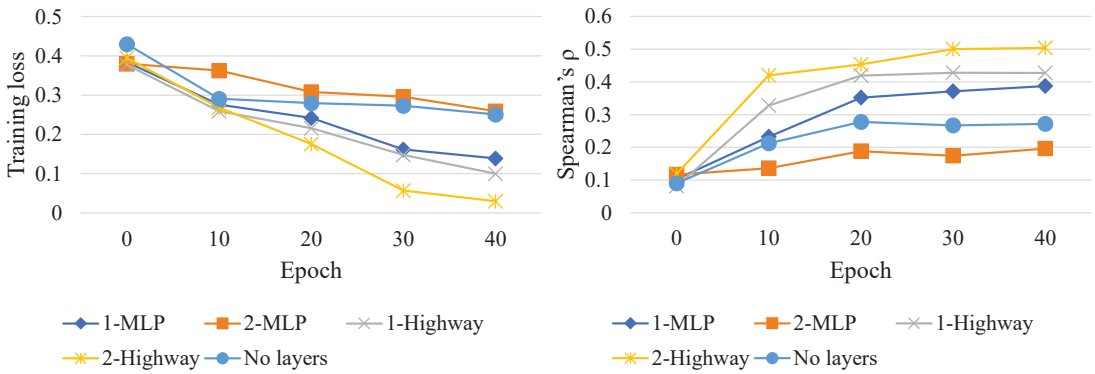
Table 3: Test perplexity on the language modeling for six languages. Best results are in bold.

4.3.2 Results

Table 3 shows the results. We observe a similar trend as with the performance of the word similarity task. The subword-based methods show better performance than the word-based method with most languages. Furthermore, the proposed GWR shows superior performances over these subword-based methods by a large margin. Based on the performance of the word2vec used in training GWR, we achieve notably improved performance in morphologically rich languages. For example, the performance on Slavic languages (Cs, Ru), which are morphologically rich, shows better perplexity reduction (15 and 22% for Czech and Russian, respectively) than other languages that are less significant languages in terms of morphology. This suggests that GWR is more useful and effective in morphologically rich languages.

5 Analysis

5.1 Effect of Highway Networks



(a) Training loss

(b) Word similarity performance (Spearman's ρ)

Figure 2: Training loss and word similarity performance with different architectures. The term of k-MLP indicates that k-layered MLP are stacked to the proposed model, and it is a same manner with a highway network.

As a component of the proposed model, we use a highway network rather than a multi-layer perceptron (MLP) that is widely used in CNNs (Kim, 2014; Cao and Lu, 2017). To show the effectiveness of a highway network in our architecture, we assess the training loss and performance of word similarity during training step when each network (highway network, MLP) is stacked to the proposed model. As a representative language in this analysis, we use the WS353 dataset in Spanish, which is a morphologically rich language. The results are shown in Figure 2.

In Figure 2a, we can observe that a highway network basically shows faster convergence than does MLP with respect to training loss. The interesting results occur when the model is deeper with each network (2-highway, 2-MLP). While 2-MLP shows slower convergence than 1-MLP, 2-Highway shows faster convergence than 1-Highway. These results are related to the property of a highway network, which allows the model to train more deeply.

Figure 2b shows the performance of word similarity task. We observe a similar trend with the results of the training loss. The highway network shows superior performance compared to MLP. Furthermore, when 2-Highway is stacked to the proposed model, it shows better performance than that of 1-Highway. In contrast, 2-MLP performs worse than 1-MLP, even when the proposed model does not have a layer (No layers).

Through this analysis, we observe that a highway network enhances the semantic similarity of words, and makes the training faster compared to a simple MLP. Furthermore, stacked highway networks enable us to train more deep model that leads the strong performance.

	In-Vocabulay				Out-of-Vocabulary			
	connect	teach	taxicab	computes	bluejacket	vehicals	globalise	computerization
word2vec	connect connecting communicate interact linking	teach learn instruct enroll educate	taxicab taxi minibus motorbike truck	computes calculates logarithmic satisfies generates	-	-	-	-
GWR	connect connecting connects connected linking	teach instruct learn teaching understand	taxicab taxi limousine minibus motorbike	computes compute calculates logarithmic computable	jacket trouser sleeve t-shirt pants	vehicles bicycles cars automobiles trailers	global worldwide globally globalisation globalization	computational computation computerized visualization computations

Table 4: Nearest neighbors of word representations for vocabulary and OOV words. Nearest words are sorted in descending order of similarity that computed by cosine similarity.

5.2 Nearest Neighbor of Words

To explore the quality of GWR, we perform a qualitative analysis by finding the nearest neighbors of word representations. Table 4 shows the nearest neighbors of word representations learned from the word2vec and GWR in English.

From nearest neighbors of OOV words, we discover the following: (i) GWR represents word-variants and compound words well (*computerization, bluejacket*); (ii) GWR captures different word styles (*globalise*) and syntactically related words (*global-globally-globalization*); (iii) GWR shows robustness on misspellings (*vehic**a**ls*); (iv) GWR captures semantically related words in most OOV words (*bluejacket-pants, vehicals-bicycles*).

We also derive some interesting results for nearest neighbors of vocabulary words. We observe that semantically or syntactically related words are located in nearest neighbors (*taxicab-minibus, teach-teaching*). Moreover, the proposed model renders lexically related word representation more similar. For example, the neighbors of "*connect*" show that the word-variants (*connects-connected-connecting*) are more closely located than are pre-trained word embeddings in vector space. This rendering satisfies the characteristic of words about *lexical relatedness* (Williams, 1981). Other words also show a similar trend (*teach-teaching*, *computes-compute-computable*). This is probably because the lexically related words share many parts of character sequences.

5.3 Training time

We measure the training time for GWR and corpus-based models. For a fair comparison, we use the same hardware resources, and record the training time when each model achieves the best performance on English data. Word2vec and FastText, which are corpus-based models, take approximately 10 and 14 hours, respectively. In contrast, GWR takes 52 minutes for learning 100,000 words in pre-trained word embeddings (*i.e.,* 16x faster than FastText). This result demonstrates that GWR produces word representations efficiently.

6 Conclusion

In this paper, we have proposed a character-based approach, denoted as GWR, to generate word representations. To this end, we have used a CNN and a highway network to extract salient subword features, and then learned features from pre-trained word embeddings. Notably, the proposed model allows us to produce high quality representations of OOV words by considering subword information. Experimental results demonstrate the efficacy of our methodology and clearly show that the proposed model works well for NLP tasks. We plan to apply the proposed model to different domains such as text classification and named entity recognition.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (number 2015R1A2A1A10052665).

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*, pages 265–283.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, pages 1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, pages 135–146.

Jan Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *Proceedings of the International Conference on Machine Learning*, pages 1899–1907.

Shaosheng Cao and Wei Lu. 2017. Improving word embeddings with convolutional feature learning and subword information. In *Proceedings of the Association for the Advancement of Artificial Intelligence*, pages 3144–3151.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the International Conference on Machine Learning*, pages 160–167.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the International Conference on World Wide Web*, pages 406–414.

Samer Hassan and Rada Mihalcea. 2009. Cross-lingual semantic relatedness using encyclopedic knowledge. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1192–1201.

Felix Hill, Kyunghyun Cho, Sébastien Jean, Coline Devin, and Yoshua Bengio. 2014. Embedding word similarity with neural machine translation. *CoRR*, abs/1412.6448.

Colette Joubarne and Diana Inkpen. 2011. Comparison of semantic similarity for different languages using the google n-gram corpus and second-order co-occurrence measures. In *Proceedings of the Canadian Conference on Artificial Intelligence*, pages 216–221.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the Association for the Advancement of Artificial Intelligence*, pages 2741–2749.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1746–1781.

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

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the International Conference on Machine Learning*, pages 1378–1387.

Siwei Lai, Kang Liu, Shizhu He, and Jun Zhao. 2016. How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6):5–14.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, pages 2278–2324.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530.

Minh-Thang Luong and Christopher D Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1054–1063.

Minh-Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. *Proceedings of the Annual Conference on Natural Language Learning*, pages 104–113.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the International Conference on Neural Information Processing Systems*, pages 3111–3119.

Amr El-Desoky Mousa, Hong-Kwang Jeff Kuo, Lidia Mangu, and Hagen Soltau. 2013. Morpheme-based feature-rich language models using deep neural networks for lvcsr of egyptian arabic. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 8435–8439.

Alexander Panchenko, Dmitry Ustalov, Nikolay Arefyev, Denis Paperno, Natalia Konstantinova, Natalia Loukachevitch, and Chris Biemann. 2016. Human and machine judgements for russian semantic relatedness. In *Proceedings of the International Conference on Analysis of Images, Social Networks and Texts*, pages 221–235.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword rnns. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 102–112.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *Proceedings of the International Conference on Computational Linguistics*, pages 141–150.

Marta Ruiz Costa-Jussà and José Adrián Rodríguez Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 357–361.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the International Conference on Machine Learning*, pages 1818–1826.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Proceedings of the International Conference on Neural Information Processing Systems*, pages 2377–2385.

Joseph P Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 384–394.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515.

Edwin Williams. 1981. On the notions "lexically related" and" head of a word". *Linguistic Inquiry*, pages 245–274.

Torsten Zesch and Iryna Gurevych. 2006. Automatically creating datasets for measures of semantic relatedness. In *Proceedings of the Workshop on Linguistic Distances*, pages 16–24.

Urdu Word Segmentation using Conditional Random Fields (CRFs)

Haris Bin Zia
ITU, Pakistan
haris.zia@itu.edu.pk

Agha Ali Raza
ITU, Pakistan
agha.ali.raza@itu.edu.pk

Awais Athar
EMBL-EBI, UK
awais@ebi.ac.uk

Abstract

State-of-the-art Natural Language Processing algorithms rely heavily on efficient word segmentation. Urdu is amongst languages for which word segmentation is a complex task as it exhibits space omission as well as space insertion issues. This is partly due to the Arabic script which although cursive in nature, consists of characters that have inherent joining and non-joining attributes regardless of word boundary. This paper presents a word segmentation system for Urdu which uses a Conditional Random Field sequence modeler with orthographic, linguistic and morphological features. Our proposed model automatically learns to predict white space as word boundary as well as Zero Width Non-Joiner (ZWNJ) as sub-word boundary. Using a manually annotated corpus, our model achieves F_1 score of 0.97 for word boundary identification and 0.85 for sub-word boundary identification tasks. We have made our code and corpus publicly available to make our results reproducible.

Title and Abstract in Urdu

اردو میں تقطیع الالفاظ بذریعہ کنڈیشنل رینڈم فیلڈز

جدید ترین نیچرل لینگویج پراسیسنگ کے الگورتھم رواں تحریر کی الفاظ میں تقسیم پر بہت انحصار کرتے ہیں۔ اردو ان زبانوں میں شامل ہے جن میں خودکار طریقے سے یہ تقطیع الالفاظ ایک پیچیدہ عمل ہے کیونکہ اردو لکھتے وقت الفاظ کے درمیان خالی جگہ اکثر حذف کر دی جاتی ہے اور کبھی غیرضروری طور پر داخل بھی کر دی جاتی ہے۔ یہ مسئلہ جزوی طور پر عربی رسم الخط کے استعمال کی وجہ سے ہے جس میں حدودِ الفاظ سے قطع نظر، حروف کی متصلہ اور غیر متصلہ، دونوں اشکال پائی جاتی ہیں ۔ اس مقالہ میں ہم اردو کے لیے ایک خود کار نظام پیش کر رہے ہیں جو کسی بھی اردو تحریر کی بآسانی تقطیع کر سکتا ہے۔ یہ نظام املائی، لسانی اور معنوی اوصاف کا استعمال کرنے والے ایک کنڈیشنل رینڈم فیلڈ ماڈل پر مبنی ہے۔ ہمارا مجوزہ ماڈل خود کار طریقے سے خالی جگہ کی لفظی حد کے طور پر پیشگوئی کرنا سیکھتا ہے اور اس کے ساتھ زیرو وڈتھ نان جائز (ZWNJ) کی ذیلی لفظ کی حد کے طور پر بھی پیشن گوئی کرتا ہے۔ ایک دستی نشان شدہ کارپس کا استعمال کرتے ہوئے، ہمارے ماڈل کو لفظی حد کی شناخت کے لیے 0.97 کا اور ذیلی لفظ کی حد کی شناخت کے لیے 0.85 کا F_1 سکور حاصل ہوتا ہے۔ اپنے نتائج کو قابل تکرار بنانے کے لیے ہم نے اپنے کوڈ اور کارپس کو عمومی طور پر دستیاب کر دیا ہے۔

1 Introduction

Word segmentation is an essential step for most Natural Language Processing (NLP) tasks. Most of the state-of-the-art NLP applications such as Machine Translation, Parts-of-Speech (POS) tagging etc. operate at word level. English and other languages that use Latin script usually rely on white spaces to mark word boundaries, with some exceptions like compound words, and exhibit less segmentation issues than Asian languages like Mandarin, Lao, and Urdu etc. that do not have consistent word boundary markings. In the absence of word boundary markers, tokenization of text for further processing is a non-trivial and challenging task.

Urdu, the official language of Pakistan, is an Indo-Aryan language with over 163 million speakers[1] worldwide. It uses Arabic script with segmental writing system. More specifically it uses an *abjad* system where consonants and (most) long vowels are necessarily written while short vowels (diacritics) are

[1] https://www.ethnologue.com/language/urd

Proceedings of the 27th International Conference on Computational Linguistics, pages 2562–2569
Santa Fe, New Mexico, USA, August 20-26, 2018.

optional. Urdu is bidirectional and characters are written from right-to-left while numerals are written from left-to-right. A sentence written in Urdu along with its English translation is given below:

اردو پاکستان کی قومی زبان ہے۔

Urdu is the national language of Pakistan.

When Urdu script is written in digital form, white space is not used for word boundary alone but also serves as a sub-word boundary marker as discussed in subsequent sections. Due to this absence of a clear word boundary marker, Urdu exhibits complex segmentation issues for natural language processing as well as information retrieval. In this paper, we present a system to solve this problem of word tokenization. **The major novel contributions of this paper are:**

1. A publicly available annotated corpus for Urdu word segmentation.
2. A Conditional Random Field (CRF) word segmentation utility for Urdu based on linguistic and orthographic features.

The remainder of this paper is structured as follows: Section 2 reviews segmentation techniques. We then present Urdu orthography and writing system in Section 3. Section 4 briefly discusses challenges in Urdu word segmentation. We present our corpus and model in section 5 and conclude in section 6.

2 Literature Review

Several techniques have been applied to solve segmentation issues for different world languages. Wong and Chan (1996) proposed a rule-based maximum-matching (max-match) dictionary look-up for Chinese word segmentation. This technique does not take into account the context of words and also never generates short valid words. Another rule-based variant of max-match exists that first generates all possible segmentations of character sequences and then selects the best one based on heuristics like minimum length words etc. (Sornlertlamvanich, 1993; Ping and Yu-Hang, 1994; Nie et al., 1994).

Statistical methods like n-grams have also been extensively used to segment character sequences into words and have proven very effective (Kawtrakul, 1997; Aroonmanakun, 2002). However, performance of statistical word segmentation methods relies heavily on the quality of training corpora and computationally expensive higher order n-grams to capture long distance dependencies. Charoenpornsawat et al. (1998) use context-based features like Winnow (Blum, 1997) for Thai word boundary identification.

Huihsin et al. (2005) proposed a conditional random field (CRF) sequence modeler for Chinese word segmentation. Their feature set consisted of simple n-gram features e.g. unigram and bigram features. Monroe et al. (2014) used the same CRF sequence modeler but with extended linguistic features for Arabic word boundary identification. They have reported F_1 score of 0.92 on Egyptian Arabic Treebank[2] and 0.98 on Arabic Treebank[3]. Our proposed model is similar to that of Monroe et al. (2014) but based on a different feature set.

More recently Cai and Zhao (2016) and Cai et al. (2017) investigated the use of neural language models with word and character-based embedding for efficient Chinese word segmentation and achieved better results than traditional segmentation algorithms. As with all deep-learning architectures, their model is data hungry and needs much training data, which is a barrier for low-resource languages.

There has been some research for Urdu word segmentation (Durrani and Hussain, 2010) using hybrid techniques such as rule-based methods relying on max-match, statistical methods such as n-grams together with the POS tags of words. Their best technique have achieved error detection rate of 85.8% for space insertion and space omission errors.

To our knowledge, there is no existing CRF based model for Urdu word segmentation along with a publicly available corpus and gold standard that can act as a reproducible reference for this task.

3 Urdu Writing System

Urdu is written using Arabic script in a cursive format (Nastaliq style) from right to left using an extended Arabic character set. The character set includes 37 basic and 4 secondary letters, seven diacritics, punctuation marks and special symbols (Hussain & Afzal, 2001; Afzal & Hussain, 2001; Hussain, 2004).

[2] LDC2012E{93,98,89,99,107,125}, LDC2013E{12,21}

[3] LDC2010T13, LDC2011T09, LDC2010T08

When characters in Urdu character-set join to form words, they can acquire different shapes. Based on context, a character may have up to four shape variants: 1) word initial 2) word medial 3) word final, and 4) isolated. Characters that can acquire all of the four shapes are known as *joiners* while one that only have two forms i.e. final and isolated are termed as *non-joiners*. Urdu joiners and non-joiners are shown in Table 1.

Joiners	ب پ ت ٹ ث ج چ ح خ س ش ص ض ط ظ ع غ ف ق ک گ ل م ن ہ ی
Non-joiners	ا د ڈ ذ ر ڑ ز ژ و ء ے

Table 1: Urdu joiners and non-joiners.

Inherently, Urdu does not have the concept of white space as a word boundary marker. During digital transcriptions, native Urdu typists use space to get accurate shape of characters instead of using it to mark word boundaries. For example, Urdu writers may insert a space within a word دولت·مند (rich) to make it visually correct, where the character · represents the ASCII space character. Omitting the space would lead to an incorrect visual form, دولتمند, for the same word. On the other hand, writers may omit a space between two separate words like اردوشاعر (Urdu poet) because the shape of characters with or without space remains identical. Due to ambiguous use of white-space in Urdu, it cannot be used as a reliable word boundary marker for NLP applications.

4 Challenges of Urdu Word Segmentation

Urdu word segmentation exhibits space omission as well as space insertion challenges. These challenges are discussed in detail by Durrani and Hussain (2010) and are summarized below:

4.1 Space Omission

Urdu words ending with non-joiner characters exhibit correct shape even without space and thus the writer may omit a space between words ending with non-joiner characters. Omission of space does not affect readability of words but raises a computation issue. An example is illustrated in Table 2.

a	اسد·قافلے·کے·صدر·کے·طور·پر·گیا۔
b	اسدقافلے کے صدر کے طورپرگیا۔
c	Asad went as the leader of caravan.

Table 2: Urdu sentence with all words ending in non-joiner characters (a) with space (shown by a ·) after each word (b) with no space after any word (c) English translation. Computationally, this sentence, which has eight tokens originally (with spaces), reduced to one token (without spaces).

4.2 Space Insertion

Another challenge of word segmentation in Urdu arises when two (or more) morphemes join to form a single word. If the first morpheme ends in a joiner character, writers may insert white space to prevent it joining with the next morpheme so that the word retains a valid visual form. If the first morpheme ends in a non-joiner, writers may (correctly) omit the space, as the shape of the word remains the same regardless. The error in this case can be avoided by using the Zero Width Non-Joiner (ZWNJ)[4] Unicode character but Urdu users are generally not aware of its existence and most Urdu keyboards also do not

[4] The standard ISO symbol for ZWNJ is ‌ which has been approximated by ‌ in this paper to avoid formatting issues

have a direct key mapping for this character. Thus an extra space within a word creates two (or more) separate tokens for a single word and creates a computational problem. The space insertion problem exists in the following cases:

- Affixation: to keep affixes separate from stem.
- Compounding: to keep words compounded together from visually merging.
- Reduplication: to keep reduplicated words from combining.
- Foreign word: to enhance readability of transliterated words.
- Abbreviation: to keep letters separate when foreign abbreviations are transliterated.

Case	a	b	c	Translation
Affixation	خوش·نصیب	خوشنصیب	خوش‌نصیب	Lucky
Compounding	نظم·و·ضبط	نظموضبط	نظم‌و‌ضبط	Discipline
Reduplication	دھوم·دھام	دھومدھام	دھوم‌دھام	Pomp & Show
Foreign word	فٹ·بال	فٹبال	فٹ‌بال	Football
Abbreviation	پی·ایچ·ڈی	پیایچڈی	پی‌ایچ‌ڈی	Ph.D.

Table 3: Example of space insertion (a) incorrect multiple tokens with space (·), correct shape (b) single token, incorrect shape (c) single token with ZWNJ (‌) as sub-word boundary, correct shape.

5 Urdu Word Segmentation Model

In order to accurately and efficiently solve space omission and insertion issues, we propose a Conditional Random Field (CRF) sequence model that uses linguistic and orthographic features to predict white space as word boundary and ZWNJ as sub-word boundary markers. Our model takes as input a concatenated sequence of characters and outputs sequence of words with space as word boundary and ZWNJ as sub-word boundary markers as shown in Figure 1. A description of our data annotation, experiments, and results is as follows.

5.1 Data Annotation

Unlike resource-rich languages such as English and Arabic that have abundant publicly accessible linguistic resources, Urdu is relatively under-resourced and does not have any segmentation benchmark corpus. To overcome this shortcoming, we manually annotated a corpus of approximately 111,000 tokens by using the Urdu-Nepali-English parallel corpus[5] as a starting point. This corpus is a parallel corpus to the common English source from PENN Treebank corpus where the source English sentences were news stories from Wall Street Journal (WSJ). We then used the CLE Urdu corpus cleaning application[6] to mark white space as word boundary and ZWNJ as sub-word boundary markers as per the rules proposed by Rehman et al. (2011). A summary of these rules is being reproduced here for the convenience of the reader:

- White space after each word ending.

[5] http://www.cle.org.pk/software/ling_resources/UrduNepaliEnglishParallelCorpus.htm
[6] http://www.cle.org.pk/software/langproc/corpuscleaningH.htm

- ZWNJ between two roots or stems of X-Y compounding e.g. انشاءاللہ
- ZWNJ between two roots or stems of X-e-Y compounding e.g. وزیراعظم
- ZWNJ before and after و in X-o-Y compounding e.g. نظم‌وضبط
- ZWNJ between reduplicated words e.g. روٹی‌ووٹی
- ZWNJ between prefix and root in case of prefixation e.g. خوش‌اخلاق
- ZWNJ between root and suffix in case of suffixation e.g. شادی‌شدہ
- ZWNJ between multiple morphemes of a single transliterated word e.g. فٹ‌بال
- ZWNJ between multiple morphemes of a transliterated abbreviation e.g. پی‌ایچ‌ڈی

The complete corpus was annotated by the first author. To calculate the inter-annotator agreement, 100 sentences (21,781 characters) were annotated by one of the co-authors and the inter-annotator agreement was found to be 0.98 as measured using Cohen's Kappa. Both annotators are native Urdu speakers with a background of computer science and NLP, and use Urdu scripts in both printed and digital form on a regular day-to-day basis.

Next, we applied Unicode text normalization to convert multiple equivalent representations of characters to their consistent underlying normal forms and removed all diacritics and multiple spaces from the text. Some of these characters, such as diacritics, are good indicators of word boundaries; however, it is a common practice of first language writers of Urdu to exclude diacritics in all informal and most formal forms of writing. The reader simply uses contextual information to interpret the correct word. Therefore, to model the real-life situation more closely we decided to omit all diacritics from our corpus. Omission of diacritics makes the segmentation task harder (as their removal loses a major word disambiguation and hence possibly word segmentation cue) but makes it unbiased at the same time. For the sake of comparison, we report results with diacritics present as well as removed in later sections.

The corpus consists of 4,325 sentences which cover most Urdu morphological and graphical constructs such as affixes, compounds, reduplicates, foreign words (transliterated from Latin to Arabic script) as well as abbreviations (transliterated from Latin to Arabic script) and words ending in both joiner and non-joiner characters. We have made this corpus publicly available on GitHub[7] along with our pipeline. For evaluation purposes, we have split the data in terms of sentences: 3,500 for training (90K tokens) and 825 for testing (21K tokens). The test set contains 20,264 word boundary spaces and 1,200 ZWNJ sub-word boundary markers.

5.2 Model

A CRF model, proposed by Lafferty et al. (2001), is defined as $P(\mathbf{Y}|\mathbf{X};\mathrm{w})$ where $\mathbf{X} = \{x_1,\ldots,x_n\}$ is a sequence of input and $\mathbf{Y} = \{y_1,\ldots,y_n\}$ is a predicted sequence of labels. We use a linear-chain model (Green and DeNero, 2012) with $\mathbf{X}$ as concatenated sequence of characters and $\hat{\mathbf{Y}}$ is chosen as per the following decision rule.

$$\hat{\mathbf{Y}} = \mathbf{argmax} \sum \mathrm{w}^{\mathrm{T}}\, \phi(\mathbf{X}, y_i);\ \mathrm{i} = 1,\ldots,\mathrm{n}$$

Where ϕ is a feature map defined in section 5.3. Our model classifies each y_i as one of the following:

- I: continuation of a word or sub-word.
- $\mathrm{B_W}$: beginning of a word.
- $\mathrm{B_S}$: beginning of a sub-word.

[7] https://github.com/harisbinzia/Urdu-Word-Segmentation

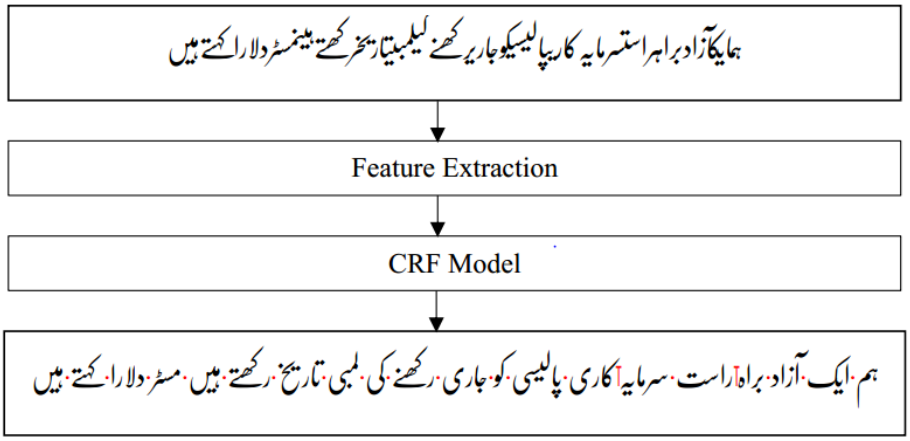


Figure 1: Urdu word segmentation pipeline. The input is a concatenated sequence of characters and the output is a sequence of words with white space (·) as word boundary marker and ZWNJ (‌) as sub-word boundary marker. In this example براہ‌راست (live) is a compound word that consists of two morphemes براہ and راست so our model predicted ZWNJ between them. Similarly, سرمایہ‌کاری (investment) is an example of affixation with ZWNJ between root سرمایہ and suffix کاری.

5.3 Feature Engineering

We experimented with different orthographic and linguistic features such as character windows of varying lengths, joining/ non-joining behavior etc. and selected the best ones for our model. Table 4 summarizes the effect of each feature on the performance of the model. Our final CRF model employs the following features:

- N-grams consisting of the current character and up to three preceding and three succeeding characters e.g. for each i^{th} character x_i, the character sequence $\{x_{i-3}, \ldots, x_i, \ldots, x_{i+3}\}$ is considered.
- Whether the current character is a digit.
- Whether the current character is a joiner.
- Unicode class of current character.
- Direction of current character. Urdu is bidirectional where basic/secondary characters and diacritics are written from right-to-left while numeric characters from left-to-right.

5.4 Results

We have used precision, recall, and F_1 measure as our evaluation metrics as they provides a more informative assessment of the performance than the word level and character level error rates. On an unseen undiacritized test set of 825 sentences (21K tokens) our model achieved F_1 score of 0.97 for word boundary and 0.85 for sub-word boundary identification. The detailed results are shown in Table 5 and 6.

Not surprisingly, the F_1 score for sub-word boundary identification is slightly higher for diacritized text as some diacritics are very indicative features of sub-word boundary e.g. ِ in compounding. Diacritized text also has high precision over undiacritized text for word boundary prediction as the diacritic ً is a clear indication of word boundary.

We also report the macro and micro F_1-measures. However the results do not show much improvement between diacritized and non diacritized corpora. One possible explanation is that the corpus is very sparsely diacritized with only 5,237 diacritics in 111K tokens. Solution to this problem via automatic diacritization is beyond the scope of the current paper.

Features	F_1 Word Boundary (Space)	F_1 Sub-word Boundary (ZWNJ)
Current character	0.59	0
+1 character window	0.77	0.07
+2 character window	0.85	0.46
+3 character window	0.92	0.72
+isDigit	0.95	0.79
+isJoiner	0.96	0.84
+Unicode class	0.97	0.85
+Direction	0.97	0.85

Table 4: Results with different feature set.

Gold \ Predicted	I	B_W	B_S
I	59,149	474	42
B_W	574	19,637	53
B_S	119	116	965

a) Undiacritized test set

Gold \ Predicted	I	B_W	B_S
I	60,254	405	44
B_W	560	19,660	44
B_S	115	88	997

b) Diacritized test set

Table 5: Confusion matrices for word and sub-word boundary identification.

	Boundary	Precision	Recall	F_1	
Without Diacritics	Word	0.97	0.97	0.97	macro-F_1= 0.91 micro-F_1= 0.96
	Sub-word	0.91	0.80	0.85	
With Diacritics	Word	0.98	0.97	0.97	macro-F_1= 0.92 micro-F_1= 0.96
	Sub-word	0.92	0.83	0.87	

Table 6: Test set results with and without diacritics.

6 Conclusion & Future work

We present a CRF sequence modeler for word and sub-word boundary identification in Urdu text using orthographic and linguistic features. Our best model achieves F_1 score of 0.97 and 0.85 for word and sub-word prediction tasks respectively. We also present handcrafted training and testing data that we have made publicly available to allow for reproducible research.

The presented system is trained using a manually crafted corpus of 4,325 sentences (111K tokens) which is relatively small compared to segmentation benchmark corpora of Arabic and Chinese. As a future direction we will extend our work to tag a much bigger corpus with space as word boundary markers and ZWNJ as sub-word boundary markers. We will explore more linguistic features for the CRF model to increase performance of sub-word boundary identification task. We also plan to explore neural models such as RNN and Bidirectional RNN for Urdu word and sub-word boundary prediction.

References

Afzal, M., & Hussain, S. (2001). Urdu computing standards: development of Urdu Zabta Takhti (UZT) 1.01. In Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International (pp. 216-222). IEEE.

Aroonmanakun, W. (2002). Collocation and Thai word segmentation. In Proceedings of the 5th SNLP & 5th Oriental COCOSDA Workshop (pp. 68-75).

Blum, A. (1995). Empirical support for winnow and weighted-majority based algorithms: results on a calendar scheduling domain. In Machine Learning Proceedings 1995 (pp. 64-72).

Cai, D., & Zhao, H. (2016). Neural word segmentation learning for Chinese. arXiv preprint arXiv:1606.04300.

Cai, D., Zhao, H., Zhang, Z., Xin, Y., Wu, Y., & Huang, F. (2017). Fast and accurate neural word segmentation for chinese. arXiv preprint arXiv:1704.07047.

Charoenpornsawat, P., Kijsirikul, B., & Meknavin, S. (1998, November). Feature-based Thai unknown word boundary identification using winnow. In Circuits and Systems, 1998. IEEE APCCAS 1998. The 1998 IEEE Asia-Pacific Conference on (pp. 547-550). IEEE.

Durrani, N., & Hussain, S. (2010, June). Urdu word segmentation. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (pp. 528-536). Association for Computational Linguistics.

Green, S., & DeNero, J. (2012, July). A class-based agreement model for generating accurately inflected translations. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1 (pp. 146-155). Association for Computational Linguistics.

Huihsin, T., Chang, P., Andrew, G., Jurafsky, D., & Manning, C. (2005). A conditional random field word segmenter. In Fourth SIGHAN Workshop.

Hussain, S., & Afzal, M. (2001). Urdu computing standards: Urdu zabta takhti (uzt) 1.01. In Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International (pp. 223-228). IEEE.

Hussain, S. (2004, August). Letter-to-sound conversion for Urdu text-to-speech system. In Proceedings of the workshop on computational approaches to Arabic script-based languages (pp. 74-79). Association for Computational Linguistics.

Kawtrakul, A. (1997). Automatic Thai unknown word recognition. In Proc. 4th Natural Language Processing Pacific Rim Symposium (NLPRS-97), Phuket, Thailand, Oct. (pp. 341-346).

Lafferty, J., McCallum, A., & Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Monroe, W., Green, S., & Manning, C. D. (2014). Word segmentation of informal Arabic with domain adaptation. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (Vol. 2, pp. 206-211).

Nie, J. Y., Jin, W., & Hannan, M. L. (1994, June). A hybrid approach to unknown word detection and segmentation of Chinese. In Proceedings of International Conference on Chinese Computing (pp. 326-335).

Ping, G., & Yu-Hang, M. (1994). The adjacent matching algorithm of Chinese automatic word segmentation and its implementation in the QHFY Chinese-English system. In Proceedings of the 1994 International Conference on Chinese Computing, Singapore (Vol. 301, p. 94).

Rehman, Z., Anwar, W., & Bajwa, U. I. (2011). Challenges in Urdu text tokenization and sentence boundary disambiguation. In Proceedings of the 2nd Workshop on South Southeast Asian Natural Language Processing (WSSANLP) (pp. 40-45).

Sornlertlamvanich, V. (1993). Word segmentation for Thai in machine translation system. Machine Translation, NECTEC, 556-561.

Wong, P. K., & Chan, C. (1996, August). Chinese word segmentation based on maximum matching and word binding force. In Proceedings of the 16th conference on Computational linguistics-Volume 1 (pp. 200-203). Association for Computational Linguistics.

ReSyf: a French lexicon with ranked synonyms

Billami, Mokhtar B.
Aix-Marseille University
LIS UMR 7020
Marseille, France
mokhtar.billami@univ-amu.fr

François, Thomas
Catholic University of Louvain
FNRS-CENTAL/IL&C
Louvain-la-Neuve, Belgium
thomas.francois@uclouvain.be

Gala, Núria
Aix-Marseille University
LPL UMR 7309
Aix-en-Provence, France
nuria.gala@univ-amu.fr

Abstract

In this article, we present ReSyf, a lexical resource of monolingual synonyms ranked according to their difficulty to be read and understood by native learners of French. The synonyms come from an existing lexical network and they have been semantically disambiguated and refined. A ranking algorithm, based on a wide range of linguistic features and validated through an evaluation campaign with human annotators, automatically sorts the synonyms corresponding to a given word sense by reading difficulty. ReSyf is freely available and will be integrated into a web platform for reading assistance. It can also be applied to perform lexical simplification of French texts.

1 Introduction

With the availability of very large corpora and the growing maturity of corpus linguistics and NLP techniques, quantitative descriptions of the lexicon have made noteworthy progress. The coverage of the resources has increased, tokenizing and part-of-speech tagging have enabled a better annotation of the lexical units and statistical models have yielded more accurate descriptions of the lexicon (in terms of frequencies, n-gram models, etc.). However, to the best of our knowledge, on-line lexical resources with a ranking of the lexical units as regards to their difficulty to be read and understood are scarse. Such information is highly relevant in communicative and educational contexts (e.g. clear and efficient writing, calibration of teaching materials, etc.) given that complex synonyms are identified and their simpler equivalents proposed.

In this paper, we present and describe such a resource for French. In ReSyf, synonyms are automatically disambiguated and ranked according to their complexity for L1 schoolchildren readers. The lexicon has been developed as part of the project ALECTOR[1] (Reading Aids to leverage Document Accessibility for Children with Dyslexia) that aims to support poor readers and dyslexic children to acquire French vocabulary and improve their reading skills in French through practise. More specifically, ALECTOR addresses the challenge of automatic text simplification for this population and illustrates one of the possible uses of ReSyf: it can be a knowledge database to carry out the substitution of complex words present in reading materials. The evaluation of the proposed ranking algorithm of the synonyms shows that, in 91% of the cases, the ranks in ReSyf correspond to the ranks given by human annotators. These results are encouraging taking into account the difficulty of the task (i.e. subjectivity, inter-annotator agreement). They are also important because the simplest synonym is always identified, which implies that ReSyf can indeed be used for reading assistance and can also be integrated into a model for automatic lexical simplification.

The paper is organized as follows. After reviewing related work at Section 2, we present the methodology applied to create ReSyf at Section 3. It includes three steps: (1) the choice of the synonym resource; (2) the disambiguation process, and (3) the ranking algorithm. Section 4 provides details on the resource itself, its availability and its evaluation by humans. Some concluding remarks and future work are discussed in Section 5.

[1] https://alectorsite.wordpress.com

Proceedings of the 27th International Conference on Computational Linguistics, pages 2570–2581
Santa Fe, New Mexico, USA, August 20-26, 2018.

2 Related work

Despite playing a crucial role in semantic applications, available machine-readable resources with synonyms are still scarce. In English, the Roget's 21st Century Thesaurus[2] (third edition) and WordNet (Miller et al., 1990) are notable examples. In French, the lexicon from CRISCO[3] (Morel and François, 2015) and the lexical network JeuxDeMots[4] (Lafourcade, 2007) are the most well-known and freely available on-line resources. Other semantic resources including synonymy relations are WOLF – the French WordNet – (Sagot and Fiser, 2008) or BabelNet (Navigli and Ponzetto, 2012). Yet, they have been automatically built from multilingual networks and therefore the granularity of their senses often lacks precision to be exploited for our purposes (not to mention wrong translations in the case of the French version of BabelNet).

With regards to graded lexicons, the idea of assessing a degree of 'difficulty' to words – although not recent – is not widespread. In the context of language learning and reading assistance (i.e. automatic lexical simplification in NLP), 'difficulty' can be understood as the level of proficieny required for a learner to read and understand a word. Beyond frequency counts, commonly used as proxys of the familiarity of a word[5], some initiatives have recently taken advantage of corpora with levels of difficulty. For French, Manulex (Lété et al., 2004) is a graded lexicon used by teachers and psycholinguists to identify the level of difficulty of words at school (in the context of learning French as mother tongue). This lexicon describes the frequency distributions of 23,812 French lemmas across three primary school levels: 1st grade, 2nd grade, and 3rd to 5th grades. The frequencies have been estimated on a corpus of pedagogical materials used at these three levels. FLELex (François et al., 2014) is a similar resource, aimed at learners of French as a foreign language. It is part of a larger project including graded resources for different European languages: the CEFRLex project[6]. All the resources in the CEFRLex project have been built based on corpora of pedagogical texts classified according to the six proficiency levels defined by the CECR (Conseil de l'Europe, 2001), ranging from A1 to C2.

The difficulty information in Manulex and FLELex can directly be used to assign difficulty levels to French words and synonyms (Gala et al., 2014). However, this approach presents some drawbacks. First, as they are both pedagogical resources, their coverage is limited. We thus have no level information for more complex synonyms if they are absent from these resources. Second, the pedagogical levels used in these resources may sometimes be too coarse-grained to make fine-grained distinctions as regards the complexity of synonyms. For instance, the word *minceur* ('thinness, slimness') is assigned the difficulty level 3 in Manulex, which is the highest difficulty level in this resource. However, in ReSyf, *minceur* is ranked at the 6th position in a list of 12 synonyms. Synonyms from position 7 to 12 are assumed to be more complex than *minceur*, but if we assign them the difficulty scores from Manulex, they would all be assigned to level 3. Such coarse-grained information does not allow to make any finer difficulty distinction between all these synonyms. We therefore believe that ranking synonyms is more efficient than assigning coarse-grained difficulty levels. Third, synonyms are extracted – and their difficulty is estimated – as word forms and not as word senses. For instance, *mince* is an adjective meaning 'thin, faint, slim'; it is given grade 1 in Manulex and A1 in FLELex, while in ReSyf it appears in 1st position with the meaning 'faint', in 3rd position with the meaning 'slim', and in 4th position with the meaning 'thin'.

As far as specific resources for automatic lexical simplification are concerned, to the best of our knowledge only the lexicon CASSAurus (Baeza-Yates et al., 2015) for Spanish is similar to ReSyf. However, the words in this lexicon are assigned only to two classes, simple and complex, which is a very coarse-grained view of lexical complexity.

[2]http://www.thesaurus.com

[3]http://www.crisco.unicaen.fr/des/

[4]http://www.jeuxdemots.org

[5]The first frequency lists appeared nearly one hundred years ago, among which: Thorndike (1921), Ogden (1930), Tharp (1939), Gougenheim (1958).

[6]http://cental.uclouvain.be/cefrlex/

3 Methodology to build the lexicon

To build ReSyf, we used the words in the lexical network JeuxDeMots (Lafourcade, 2007) linked by the synonymy relation (3.1). We then applied a disambiguation (3.2) and a ranking algorithm (3.3) to identify complexity within the word senses in a vector.

3.1 Nature of the lexical entries: synonyms

Synonymy is a lexico-semantic relation implying equivalence among two word senses. Absolute synonyms are very rare, synonyms are thus two lexical units having a semantic value close enough (by inclusion or intersection) to be replaced one by the other to convey the same meaning (Polguère, 2002). Word Sense Disambiguation (WSD) techniques are necessary to identify the meaning of the word forms to be able to identify the appropriate synonyms, i.e. the French word *grave* is synonym of *sérieux* when meaning 'serious' (i.e. a serious problem), but synonym of *profond* when meaning 'deep' (i.e. a deep voice).

3.2 Data acquisition: from word forms to word senses

3.2.1 Resources for bootstrapping

Similar to other resources where the complexity of the entries is given (see section 2), we first developped a version of ReSyf with word forms associated with grades (from 1 to 3) (Gala et al., 2013) and (Gala et al., 2015). This approach raised some important issues concerning the nature of the entries and Word Sense Disambiguation. These problems were very soon identified and we started to work in another version with refined senses[7] for each entry and with ranks dynamically assigned from 1 to n, depending of the number of synonyms in a vector, instead of static grades.

We created a second version using BabelNet (Navigli and Ponzetto, 2012), a lexical network automatically built from several resources (WordNet, Wikipedia, etc.). The problem with BabelNet is its excessive granularity of senses: for our purposes (for the targeted users of ReSyf), we needed a more accurate and simplified version. For instance, the word *toile* in French has fifteen senses in BabelNet. For schoolchildren, specially those with reading difficulties, four distinctions are enough ('computer network' , 'cloth' , 'painting' , 'spider web').

We finally decided to use the lexical network JeuxDeMots (Lafourcade, 2007) for the current version. The lexical database of this network, built by crowdsourcing, is very rich and constantly evolving. Each node in the lexical network represents a lexical unit describing a word (a simple word or a multiword expression, MWE). The relationships between the nodes are typed and weighted. Some of these relationships correspond to lexical functions related to the vocabulary itself (such as ASSOCIATED IDEA and SYNONYMY relationship) or to hierarchical semantic relations (such as HYPERNYMY and HYPONYMY). Senses are encoded with the SEMANTIC REFINEMENT relation. Each instance of this relation describes a specific sense for a given term.

JeuxDeMots is a human-based computation game, in other words, a game with a purpose (GWAP). The actors are simple players who play through an interface that has the form of an online game. The validation of the quality of the data collected for the construction of the lexical database, containing lexical items and relations between them, is provided by players. More specifically, relationships are proposed anonymously by a player and they are validated by other anonymous players. The relations between the lexical items are weighted. The weighting is carried out in the following way: the more an instance of a given relation is proposed, the more its weight increases, as long as the conditions of the game are respected.

At present, the JeuxDeMots lexical database contains:

a) 182 373 582 instances of all possible relations;

b) 3 081 091 terms having at least one outgoing relation ($term_A \rightarrow term_B$);

c) 2 497 238 terms having at least one incoming relation ($term_A \leftarrow term_B$).

[7] A refined sense $s_i(w)$ for the word w is a particular use of w that this word has no other similar refined sense to $s_i(w)$.

Synonyms are proposed for both words and word senses. However, it turns out that not all synonyms for a given word are grouped together in distinct senses. For example, *tissu* ('cloth') is a synonym for the lexical entry *toile* but it does not appear in the network as a disambiguated synonym. We thus need to include *tissu* as a disambiguated synonym aggregated to the word sense *toile* ('cloth'). To include all disambiguated synonyms into word senses, we use a disambiguation algorithm based on semantic representations for words and word senses (semantic signatures).

We previously proposed a methodology for the creation and validation of semantic signatures (Billami and Gala, 2017). In general terms, a semantic signature can be considered as a special representation form of Vector Space Model, VSM (Turney and Pantel, 2010). In the same way as VSM, the weight associated with a dimension in a semantic signature indicates relevance or importance of this dimension. The main difference is the way in which the weights are computed. VSM uses cooccurrence statistics in a given corpus whereas semantic signature uses structural properties of the used network (JeuxDeMots in our case).

The goal of our approach is to directly compare the semantic signature of each synonym with the semantic signature of each candidate word sense. The semantic similarity that we use is an activation function which takes into account the relation between two lexical units to compare (Billami and Gala, 2017). This relation checks whether the one (synonym or candidate word sense) represent a dimension in the semantic signature for the other. If it is true, the function returns a perfect similarity ($score\ of\ similarity\ =\ 1$) else the cosine similarity is estimated by using their semantic signatures.

We use the relation ASSOCIATED IDEA for creating the signatures. This relation contains the largest number of instances since it includes all terms that reflect a given lexical entry in JeuxDeMots (57% for ASSOCIATED IDEA instances). On the other hand, JeuxDeMots contains at least 100 lexico-semantic relations.

3.2.2 Word Sense Disambiguation method

The algorithm 1 described below allows to disambiguate a synonym by choosing the most suitable target word sense. For each pair of synonym syn_a and candidate word sense $Sense_i$ with $i \in 1 \ldots$ n (n : number of senses for the target word) the algorithm first checks whether the one represent a dimension in the semantic signature for the other. If it is true, the synonym is clustered directly to the $Sense_i$ else the cosine similarity is estimated by using their semantic signatures. In this case, the closest sense(s) with the best similarity are selected. We use a threshold $\varepsilon = 0.01$.

We have developed a variant of the algorithm 1 by comparing not directly a synonym with a candidate word sense, but each synonym sense (if this latter is polysemous) with each candidate word sense. This is the hypothesis that describes the similarity between two words as the similarity of their closest senses (Budanitsky and Hirst, 2006). On the other hand, if the synonym is monosemous, the algorithm 1 is applied (we note this variant algorithm 2).

In order to validate our algorithms, we used a list of manual disambiguated synonyms in JeuxDeMots as a test set (JeuxDeMots dump of December 2017, 33 039 pairs synonym – target word sense). The table 1 describes the precision results of our evaluation by applying the two clustering algorithms. In Word Sense Disambiguation, the precision is the ratio between the correct answers provided and the total answers provided, whereas recall is the ratio between the correct answers provided and the total answers to provide (Navigli, 2009).

Our semantic signatures based on associated ideas cover all synonyms and senses of the test set. Therefore, we have precision = recall = F-measure.

Not surprisingly, the results show that we have a better performance (higher precision) when we use the algorithm 1: the semantic signature of a synonym is most informative than a semantic signature for a specific synonym sense.

3.3 Data ranking method: from grades to ranks

Once all the senses were identified, we developed a ranking algorithm which is able to sort the synonyms according to their complexity (reading difficulty for target readers). For this task, we relied on an approach commonly used in the field of information retrieval to sort the results of a query by

Algorithm 1: Comparison of each candidate word sense with each synonym syn_a

Input:
$target\ word$: word to treat
$sem\_ref\ (target\ word)$: set of senses of the target word
syn_a: synonym of the target word
ε: validation threshold of the similarity
Result:
$\mathbf{\hat{Sense}_{target\ word}}$: senses of the target word with the highest score
Data:
$\mathrm{S_{a\_idea}}$: set of signatures whose dimensions are associated ideas

Initialization:
$Score_{refs\_C} = \emptyset$ // Score of the target word senses
for $Sense_i \in sem\_ref\ (target\ word)$ **do**
    $Score = \begin{cases} 1\ if\ (*) \\ Cosine\ (S_{a\_idea}(Sense_i), S_{a\_idea}(syn_a))\ otherwise \end{cases}$
    $(*) : Sense_i \in S_{a\_idea}(syn_a)\ \vee\ syn_a \in S_{a\_idea}(Sense_i)$;
    if $(Score\ \geqslant\ \varepsilon)$ **then**
        $Score_{refs\_C} \leftarrow Score_{refs\_C} \bigcup (Sense_i,\ Score)$;
$\mathbf{\hat{Sense}_{target\ word}} \leftarrow \mathbf{Best}\ (Score_{refs\_C})$

Clustering algorithms	Correct annotations	All annotations	%
Algorithm 1	32 802	33 039	**99.28**
Algorithm 2	25 307	33 039	76.6

Table 1: Evaluation results by applying the two clustering algorithms.

relevance: the pairwise approach, and more specifically the SVMRank algorithm (Herbrich et al., 2000). Such an approach requires a database of words already sorted by difficulty that will be used to create a training dataset composed of pairs of words with different levels of difficulty (section 3.3.1). We also computed, for each pair, a set of linguistic features (section 3.3.2) that can be used to predict which word of the pair is the most complex one. After model optimisation (section 3.3.3), we obtained a function that can predict, for a given input (word sense pair), which one is the more complex. This function may then be integrated into any sorting algorithm to rank a set of synonyms.

3.3.1 Training dataset

As ReSyf is mainly intended for schoolchildren, we used Manulex to obtain word difficulty annotations (see description at Section 2). As we only considered open class words (nouns, adjectives, adverbs, and verbs), we retained $19,038$ lemmas from Manulex for our training dataset.

Based on this resource, it is possible to create pairs of words in which one is more complex than the other. The pairs are then used to train the ranking model. However, this requires to transform the frequence distribution of each word into a single numerical value that can be used to compare the reading difficulty of two words. More formally, the distribution D for a word w takes the form of a vector (f_1, f_2, f_3) corresponding to the frequencies of the word at each of the three Manulex levels. Our goal is to define a function $\phi(D)$ that will output a single difficulty value l based on the values in D. Two approaches were tested. The first technique simply outputs a level value $L \in \{1, 2, 3\}$ that corresponds to the first level f_i for which $f_i > 0$. The training set based on this technique is called *Manulex-3N*. However, using only three values to represent difficulty creates a large amount of ties during the pair creation step. Therefore, we experimented a second technique to define $\phi(D)$ such as it outputs a

continuous value ranging from 1 to 3 using the formula below (Gala et al., 2013).

$$\phi(D) = L + \exp^{-r} \quad \text{where} \quad r = \frac{\sum_{i=1}^{L} f_i}{\sum_{i=L+1}^{3} f_i}$$

In this formula, L corresponds to the output of the first technique to which we add a continuous quantity that is function of the distribution D. This quantity is defined in terms of the ratio between the sum of the counts from levels 1 to L over the sum of the counts from $L + 1$ to 3. This way, we can distinguish two words such as *pomme* ('apple') et *cambrioleur* ('burglar') that both appear at level 1 ($L = 1$), but 724 times for 'apple' and only 2 times for 'burglar'. The training set based on this technique is called *Manulex-Cont*.

3.3.2 Word features

Each word sense from our dataset was first represented as a 69-feature vector capturing various linguistic and psycholinguistic properties that can be classified in the four following types:

- *Spelling features*: (1) number of letters, (2) number of phonemes, (3) number of syllables, (4) number of orthographical neighbors[8]; (5) cumulated frequencies of all orthographical neighbors; (6) number of neighbors that are more frequent as the target word; (7) transparency between the written and phonological forms; (8-13) six variables detecting specific complex graphems, namely oral vowels (e.g. *au* [o]), nasal vowels (e.g. *in* [ɛ̃]) , double consonnants (e.g. *pp*), double vowels (e.g. *ée*), other digrams (e.g. *ch* [ʃ]), or the sum of all five phenomenas; and (14-16) membership of the syllabic structure of the word to a class considered as either frequent, median, or rare.

- *Frequency features*: (17) the log-frequency of the word based on the Lexique3 (New et al., 2007) database and (18-26) the presence of the word in a list of simple words. We defined 9 lists of different sizes (1063, 2000, 3000, $\cdots$, 8000, and 8775 words), all based on the Gougenheim list (Gougenheim, 1958).

- *Semantic features*: (27) a binary variable coding whether the word is considered as polysemic in JeuxDeMots and (28) the number of synsets listed in BabelNet.

- *Morphological features*: the morphological analysis was automatically performed by systems developed by Bernhard (2006) and Bernhard (2010). The first of these systems splits words into tagged morphemes (root, prefixes, suffixes, etc.) and derives various frequency information about the identified morphemes, while the second system identifies the morphological families and can be used to extract information about a word family. The variables we used were : number of morphemes; presence of suffixes; presence of prefixes; presence of two bases or more (for compound words); the minimal frequency of the affixes of the word (i.e. number of different words in which appears the least frequent of the affix); the average frequency of all affixes in the word; the size of the morphological family; the frequency of the most frequent word in the morphological family; the mean frequency of all words in the family; and the cumulated frequency of all words in the family. The two systems include parameters that were manipulated, thus creating variants of the above variables. In total, we defined 41 morphological variables (29-69)[9].

3.3.3 Model definition and optimization

The definition of the ranking model was performed in three steps: (1) creating the training dataset (pairs of synonyms); (2) feature selecting; and (3) model training. To create the pair training dataset, we applied the following procedure: given two words w_i and w_j, each associated to a difficulty level (l_i and l_j) and to a feature vector ($\mathbf{v_i}$ and $\mathbf{v_j}$), we create a pair $< w_i, w_j >$ for which a new vector $\mathbf{v_{ij}}$ is obtained from the combination of the two vectors $\mathbf{v_i}$ and $\mathbf{v_j}$. Several arithmetic operations can be used to carry out

[8]The orthographic neighborhood of a word have been defined by Coltheart (1978) as all the words of same length and differing only by one letter (eg. FIST and GIST).

[9]For a detailed description of these parameters, see (Gala et al., 2014).

this combination, but we used substraction ($\mathbf{v_{ij}} = \mathbf{v_i} - \mathbf{v_j}$), as Tanaka-Ishii et al. (2010) showed that substraction was best for ranking texts by readability. Each pair $< w_i, w_j >$ was also assigned a new difficulty level (l_{ij}) obtained with the following rule: (1) if $l_i > l_j$, then $l_{ij} = 1$ and (2) if $l_i < l_j$, then $l_{ij} = -1$. As we had two original datasets, *Manulex-3N* and *Manulex-Cont*, we got two pair datasets.

Second, to select the best predictors of word difficulty, we computed the Spearman correlations between each of our 69 variables and the new binary difficulty variable (L_{ij}). We used only the *Manulex-3N* dataset for this aim. Table 2 displays the correlation for some of the best features in our set. At the end of this selection step, 21 variables were retained for the model.

Variable name	**Correlation (ρ)**
17 Freq. Lex3	-0.57
18 AbsGoug (6000)	-0.46
02 Nb. phon	0.35
15 Polysemy	-0.33
01 Nb. letters	0.32
03 Nb. syllables	0.32
4a Nb. neighbors	-0.23
15 Mean freq. of the morphological family	-0.27
15 Cum. freq. of the morphological family	-0.27
15 Max. freq. of the morphological family	-0.27
4b Cum. freq. of the neighbors	-0.23
16 Nb. of senses in BabelNet	-0.19

Table 2: Best variables based on Spearman correlation.

Third, we trained a SVM model with linear kernel on each pair dataset (*Manulex-3N* and *Manulex-Cont*). For each model, a grid-search was used to select the best value for the C meta-parameter. We then estimated the accuracy of pair classification by each model with a 10-fold cross-validation procedure. Table 3 summarizes the accuracy of both models and compares their scores with those of a full model (that uses all 69 features). The models including 21 variables reach similar or slighly better accuracy than the full model. More interestingly, the model trained on the *Manulex-3N* dataset (using the simple rule of first occurrence) clearly outperforms the model based on *Manulex-Cont*. As a result, we decided to retain the model based on the first way of defining $\phi(D)$ for the final version of ReSyf.

Dataset	**C**	**21 var.**	**C**	**69 var.**
Manulex-3N	0.01	77.4%	0.01	77.8%
Manulex-Cont	0.01	72.4%	0.01	71.4%

Table 3: Accuracy of the ranking models.

4 ReSyf: graded synonyms according to their difficulty

In this section we describe the data available in the lexicon (section 4.1) and an evaluation of the ranking algorithm (section 4.2).

4.1 Data available

ReSyf provides an inventory (a vector) of equivalent words ranked according to their difficulty to be read and understood[10]. For instance, *sec*(1), *léger*(2), *mince*(3), *allongé*(4) and *svelte*(5), corresponding to the meaning 'slim' . The first sense *mince (fin)* is the most general sense taking into account the weight of the relation between the word *mince* and the semantic refinement *mince (fin)* defined in JeuxDeMots. The weight of this type of relation allows to sort the senses from the most general to the most specific. Each weight is normalized by using the ratio between its value and the top level weight value.

[10]The resource is freely available for lookup and download (XML file) at http://cental.uclouvain.be/resyf

In order to distinguish words according to the four open classes (nouns, adjectives, adverbs, and verbs), we filtered all the single words with the French reference resource Lexique3 (New et al., 2007). For the multiword expressions (MWE), we used the parser Talismane[11] (Urieli, 2013) as a part-of-speech tagger. For MWE, we have considered that its POS is the POS assigned to its first open class item. Table 6 describes the distribution of entries in ReSyf (total number of entries: 57 589, 10 333 polysemic and 47 256 monosemic). The number of common nouns is greater than that of all the other parts-of-speech categories, either for the polysemic or the monosemic entries. Note that JeuxDeMots is constantly evolving and, as a result, more semantic refinements will be available in the future (more polysemic entries to be disambiguated).

The distribution of categories described in table 6 shows that the mean synonyms per polysemic entry sense is 4.95 (which is greater than what we can obtain when using other resources such as BabelNet).

	Nouns	**Verbs**	**Adjectives**	**Adverbs**	**Total**
#Polysemic entry (*Pe*)	**6 737**	1 779	1 691	126	**10 333**
#Monosemic entry (*Me*)	**30 869**	8 388	6 606	1 393	**47 256**
Single words	**21 495**	5 065	7 635	1 105	**35 300**
Multiword expressions	**16 111**	5 102	662	414	**22 289**
Mean synonyms per *Pe*	12.95	**17.97**	16.93	6.16	14.39
Mean synonyms per *Me*	4.19	6.86	**9.27**	4.71	5.39
Mean senses per *Pe*	2.95	**3.03**	2.65	2.25	2.9
Mean synonyms per *Pe* sense	4.39	5.92	**6.39**	2.73	4.95

Table 4: Distribution of ReSyf entries.

Table 5 describes the statistics as regards to the distribution of ReSyf synonym annotations (polysemic entries). As it is showed, JeuxDeMots currently proposes 27 466 associated synonyms to the semantic refinements for adjectives, adverbs, common nouns and verbs. By applying our first clustering algorithm (cf. algorithm 1), we are able to automatically disambiguate 121 182 synonyms. The automatic annotation represent a benefit of 4.4 more than what JeuxDeMots currently proposes.

	Nouns	**Verbs**	**Adjectives**	**Adverbs**	**Total**
#Automatic annotations	**69 323**	26 379	24 851	629	**121 182**
#Manual annotations	**17 954**	5 584	3 781	147	**27 466**

Table 5: Distribution of ReSyf synonym annotations (all polysemic entries).

The figure 1 shows a sample description of the Lexical Markup Framework (LMF) format used to encode the data in a XML file. Each lexical entry is encoded in the node 'LexicalEntry'. This latter contains different features such as the ambiguity (which takes 'one' if the lexical entry is polysemic or 'zero' otherwise), the lemma form to encode the lemma and the part of speech of the entry and sense nodes to encode all senses.

The figure 1 shows all the details for the second sense of *mince* ('faint, delicate'). The weight of this sense is 0.8511, which is lower than that of the first sense ('thin', 1.0). Each sense is available with its weight, its usage features and some examples which define the disambiguated synonym. Each sense example contains the annotation feature which takes a 'manually' value if the synonym is defined manually or 'automatically' value if the synonym is captured by the clustering algorithm 1. The feature 'word' encodes the lemma of the synonym. The features 'score_lemma' and 'score_sense' encode the maximal score obtained by our clustering algorithms. If these scores are greater or equal to 0.5 they are printed, else we print only '- -'.

[11] http://redac.univ-tlse2.fr/applications/talismane/talismane_en.html

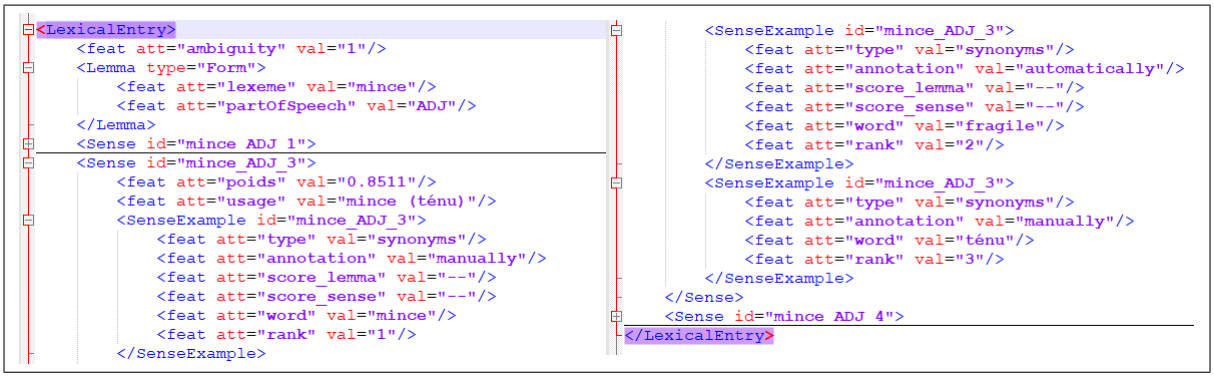


Figure 1: Lexical entry description of the word *mince* using the LMF format where the second sense is developed.

4.2 Comparison of automatic ranks with human judgments

We carried out an evaluation campaign with forty human annotators in order to obtain a gold-standard to evaluate our ranking algorithm. The annotators were asked to manually rank a list of forty senses (vectors) containing 2 to 6 synonyms (average of 3.5 synonyms/vector), with a total of 150 word forms (53 % nouns, 23 % verbs and adjectives, 1 % adverbs). The synonyms were proposed randomly (in terms of difficulty) and were not contextualized.

The ranks were manually annotated by 28 native speakers of French and 12 C1/C2 non-natives living in France for more than 5 years and having another Romance language as mother tongue. All of them were adults in the academic field: master or PhD students, assistant professors and researchers, with an average of 28,23 years old (standard deviation of 10.07).

The final reference list[12] obtained after the annotations contains 134 word forms and 36 meanings (4 senses corresponding to 16 synonyms were removed from the original list because (a) equality of the annotations or (b) presence of an unknown or irrelevant term in the vector, judged as so by most than one third of the annotators).

For each vector, the Krippendorff's alpha coefficient (α) was calculated. The global agreement obtained is 0.4, it slightly varies when it is calculated specifically for vectors with 3 or 5 synonyms. Unsurprisingly, the lesser the synonyms to rank, the higher the coefficient. These results can be compared with those obtained by Specia and collaborators at SemEval 2012 (Specia et al., 2012) (κ = 0.386 and 0.398) for a similar task.

The ranking algorithm we have developed achieves encouraging results: 83.33 % of the vectors are sorted exactly as the human annotators did, or with a slightly difference of one rank. Only 16.67 % of the vectors show a couple of synonyms ranked with more than two ranks of difference.

In terms of lexical units (synonyms), 91.04 % of them are correctly sorted or inversed with only one rank, 8.96 % have been automatically ranked with a difference of two ranks and 2.24 % (3 synonyms) have been sorted with a distance higher than two. This precise case is that of the adjectives *merveilleux, fantastique, fabuleux, formidable, splendide* ('marvellous, fantastic, fabulous, wonderful, splendid') were the annotators mostly proposed *fabuleux, formidable, fantastique, splendide, merveilleux*. This example shows the difficulty of the task for word forms with similar formal features (length, number of syllables, presence of digraphs) and corresponding to subjective senses with already very low human agreement (Krippendorff's α = 0.04).

5 Conclusion

In this paper, we have presented ReSyf, a resource for French with disambiguated synonyms that have been sorted according to readability features. The results of the ranking algorithm have been compared to human annotations and in 91% of the cases the synonyms are automatically ranked from the simplest

[12] Avalible at the end of the paper (Appendix A).

to the more complex. The lexicon can be used on-line, yet our aim is also to integrate ReSyf into a lexical simplification algorithm in order to reduce the lexical complexity of texts (lexical substitution task).

The perspectives of our work are twofold. First, we are working on the refinement of the lexicon in order to include multiword expressions (MWE). For now, the ranking of MWEs is based on an average of its content words (eg. for *faire faux bond* meaning 'to let down' , literally 'to make a false jump' , the ranking is based on feature vectors for *faux* and *bond*, which is an approximation of the reality). The issue of MWE handling is crucial for NLP applications, a more precise estimation of their complexity in reading comprehension is a real challenge.

A second perspective is that of tailoring the lexicon to the special needs of particular target audiences. By integrating ReSyf into an automatic text simplification system (lexical substitution), we aim at providing a tool to automatically adapt the words in a text to the specificities of a target reader. Roughly speaking, if avoiding long words is recommended for people with dyslexia, it might be interesting to favor them in texts tailored to adults with vision problems such as age-related macular degeneration (AMD). For instance, replace *cambrioleur* by *voleur* ('burglar' , 'thief') for dyslexic readers, but keep *cambrioleur* for AMD patients. Lexical simplification could find useful applications for different populations who struggle with reading. ReSyf could be a key component for such automatic simplification, but also for teachers, speech therapists and other professionals involved in vocabulary learning and reading assistance.

Acknowledgements

This work has been funded by the French Agence Nationale pour la Recherche, through the ALECTOR project (ANR-16-CE28-0005), and by the Belgian National Agency for Research (FNRS). The current version of the ReSyf website has been funded by Ortolang, we thank Dorian Ricci (UCL) and Brayan Delmée (UCL) for their work. We also thank Firas Hmida (LPL) and three anonymous reviewers for their comments on the previous versions of the paper.

References

R. Baeza-Yates, L. Rello, and J. Dembowski. 2015. Cassa: a context-aware synonym simplification algorithm. In *Proceedings of the 2015 NAACL:HLT Conference*, pages 1380–1385.

D. Bernhard. 2006. Unsupervised morphological segmentation based on segment predictability and word segments alignment. In *Proceedings of 2nd Pascal Challenges Workshop*, pages 19–24.

D. Bernhard. 2010. Apprentissage non supervisé de familles morphologiques: Comparaison de méthodes et aspects multilingues. *Traitement Automatique des Langues*, 51(2):11–39.

M. B. Billami and N. Gala. 2017. Creating and validating semantic signatures : application for measuring semantic similarity and lexical substitution. In *Traitement Automatique des Langues Naturelles TALN 2017*, pages 123–138, Orléans, France, June.

A. Budanitsky and G. Hirst. 2006. Evaluating WordNet-based Measures of Lexical Semantic Relatedness. *Computational Linguistics*, 32(1):13–47, Mars. MIT Press.

M. Coltheart. 1978. Lexical access in simple reading tasks. In G. Underwood, editor, *Strategies of information processing*, pages 151–216. Academic Press, London.

Conseil de l'Europe. 2001. *Cadre européen commun de référence pour les langues : apprendre, enseigner, évaluer*. Hatier, Paris.

T. François, N. Gala, P. Watrin, and C. Fairon. 2014. FLELex: a graded lexical resource for French foreign learners. In *Proceedings of LREC 2014*, Reykjavik, Island.

N. Gala, T. François, and C. Fairon. 2013. Towards a French lexicon with difficulty measures: NLP helping to bridge the gap between traditional dictionaries and specialized lexicons. In *E-lexicography in the 21st century: thinking outside the paper*, Tallin, Estonie.

N. Gala, T. François, D. Bernhard, and C. Fairon. 2014. Un modèle pour prédire la complexité lexicale et graduer les mots. In *Actes de la 21e conférence sur le Traitement Automatique des Langues Naturelles (TALN'2014)*, pages 91–102.

N. Gala, M. B. Billami, T. François, and D. Bernhard. 2015. Graded lexicons: new resources for educational purposes and much more. In *22nd Computer-assisted language learning conference (EUROCALL-2015)*, pages 204–209, Padoue, Italie, August.

G. Gougenheim. 1958. *Dictionnaire fondamental de la langue française*. Didier, Paris.

R. Herbrich, T. Graepel, and K. Obermayer. 2000. Large margin rank boundaries for ordinal regression. chapter 7, pages 115–132. MIT Press, Cambridge.

M. Lafourcade. 2007. Making people play for Lexical Acquisition with the JeuxDeMots prototype. In *SNLP'07: 7th International Symposium on NLP*, Pattaya, Chonburi, Thailande.

B. Lété, L. Sprenger-Charolles, and P. Colé. 2004. Manulex : A grade-level lexical database from French elementary-school readers. *Behavior Research Methods, Instruments and Computers*, 36:156–166.

G. A. Miller, R. Beckwith, Ch. Fellbaum, D. Gross, and K. Miller. 1990. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3:235–244.

M. Morel and J. François. 2015. Le Dictionnaire Electronique des Synonymes du CRISCO : un outil de plus en plus interactif. *Revue française de linguistique appliquée*, XX(01):9–28.

R. Navigli and S. P. Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

R. Navigli. 2009. Word Sense Disambiguation: a survey. *ACM Computing Surveys*, 41(2):1–69.

B. New, M. Brysbaert, J. Veronis, and C. Pallier. 2007. The use of film subtitles to estimate word frequencies. *Applied Psycholinguistics*, 28(04):661–677.

C. K. Ogden. 1930. *Basic English*. Paul Treber, London.

A. Polguère. 2002. *Notions de base en lexicologie*. Observatoire de Linguistique Sens-Texte, Université de Montréal, Montréal.

B. Sagot and D. Fiser. 2008. Building a free French wordnet from multilingual resources. In *Ontolex*, Marrakech.

L. Specia, Sujay K. Jauhar, and R. Mihalcea. 2012. Semeval-2012 task 1: English lexical simplification. In *Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval 2012)*, Montréal.

K. Tanaka-Ishii, S. Tezuka, and H. Terada. 2010. Sorting texts by readability. *Computational Linguistics*, 36(2):203–227.

J. Tharp. 1939. The measurement of vocabulary difficulty. *Modern Language Journal*, pages 169–178.

E. Thorndike. 1921. *The Teacher's Word Book*. Teachers College, Columbia University, New York.

Peter D. Turney and Patrick Pantel. 2010. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188, Janvier. AI Access Foundation.

Assaf Urieli. 2013. *Robust French syntax analysis: reconciling statistical methods and linguistic knowledge in the Talismane toolkit*. Ph.D. thesis, University of Toulouse II le Mirail, France.

Appendix A. List of synonyms evaluated (ranked) by human judges.

The following table shows the list of synonyms provided to the human judges for manual ranking. The words where presented decontextualized, randomly organized into a list correspondig to the same meaning. The judges had to rank them from 1 to n according to the difficulty they had to read and understand them. The table presents one line per vector of synonyms with an English translation at the end.

associer	combiner	assimiler	entrêmeler	amalgamer	to blend
bleu	azur	céruléen			blue
bleu	fromage				blue cheese
bleu	contusion	ecchymose			bruise
bleu	bizut	débutant	béjaune		beginner
brûler	cramer	incendier	cautériser	incinérer	to burn
intellectuel	cérébral				thinker
chic	élégant	huppé	aristocrate		elegant
agent	gendarme	connétable	agent de police		policemen
conte	fable	allégorie	apologue	histoire	tale
conte	narration				story
proximité	voisinage	contiguïté			nearness
noble	généreux	galant	héroïque	chevaleresque	gentle
dépouiller	apercevoir	constater	déceler	analyser	to notice
voler	piquer	dépouiller	dérober		to steal
inventer	forger	formuler			to invent
murmure	bruissement	gazouillis	gazouillement		whisper
pardon	grâce	amnistie	droit de grâce	grâce présidentielle	forgiveness
injure	affront	insulte			insult
mine	galerie	gissement	excavation	creusement	mine
mine	puits	charbonnage	huillère		pit
mine	plomb	mine de crayon			pencil lead
mine	gueule	galibot			expression
mine	mine antichar	mine antipersonnel			landmine
air	mine	manière	présence	comportement	face
parfois	tantôt	quelquefois	occasionnellement		sometimes
mémoire	rappel	réminiscence			memory
rappel	descente en rappel				abseiling
rougir	empourprer	cramoisir			to blush
fin	spirituel	mental			mental
merveilleux	fantastique	fabuleux	formidable	splendide	wonderful
maigre	osseux	squelettique			thin
sévère	rigoureux	strict	austère		strict
gémir	rugir	vagir			to roar
crier	hurler	brailler	beugler	vociférer	to yell
rugir	ronfler	bourdonner	vrombir		to hum

Table 6: ReSyf data evaluated by 40 human judges.

If you've seen some, you've seen them all: Identifying variants of multiword expressions

Caroline Pasquer
University of Tours
France

Agata Savary
University of Tours
France

Carlos Ramisch
Aix Marseille Univ,
Université de Toulon,
CNRS, LIS, Marseille, France

Jean-Yves Antoine
University of Tours
France

first.last@(univ-tours|lis-lab).fr

Abstract

Multiword expressions, especially verbal ones (VMWEs), show idiosyncratic variability, which is challenging for NLP applications, hence the need for VMWE identification. We focus on the task of variant identification, i.e. identifying variants of previously seen VMWEs, whatever their surface form. We model the problem as a classification task. Syntactic subtrees with previously seen combinations of lemmas are first extracted, and then classified on the basis of features relevant to morpho-syntactic variation of VMWEs. Feature values are both absolute, i.e. hold for a particular VMWE candidate, and relative, i.e. based on comparing a candidate with previously seen VMWEs. This approach outperforms a baseline by 4 percent points of F-measure on a French corpus.

Title and Abstract in French

Trait pour trait identiques ? Identification de variantes d'expressions polylexicales

Les expressions polylexicales (EP), et parmi elles plus particulièrement les EP verbales (EPV), se caractérisent par une grande variabilité idiosyncrasique de forme. La détection et l'identification de ces EPV variées pose ainsi un réel défi à la réalisation d'applications langagières robustes. Cet article met l'accent sur la tâche d'identification dans un corpus de variantes d'une EP verbale déjà rencontrées. Il propose une stratégie d'identification basée sur l'extraction de formes candidates à partir de patrons syntaxiques, suivie de leur classification basée sur des caractéristiques morphologiques et syntaxiques. Ces propriétés sont à la fois absolues (c.-à-d. concernent l'entité considérée) ou relatives (c.-à-d. issues de la comparaison avec des EPV déjà rencontrées). Les performances du système résultant ont été évaluées sur un corpus francophone. Elles montrent une amélioration de 4 points de F-mesure par rapport à une baseline bien établie.

1 Introduction

Multiword expressions (MWEs) such as ***red tape***, ***by and large***, *to* ***make*** *a* ***decision*** and *to* ***break*** *one's* ***heart*** are combinations of words exhibiting unexpected lexical, morphological, syntactic, semantic, pragmatic and/or statistical behavior (Baldwin and Kim, 2010). Most prominently, they are semantically non-compositional, that is, their meaning cannot be deduced from the meanings of their components and from their syntactic structure in a way deemed regular. For this reason, the presence of MWEs in texts calls for dedicated treatment, whose prerequisites include their automatic identification.

The goal of *MWE identification* is, given some input running text, to identify all MWEs' lexicalized components present in it (Schneider et al., 2016; Savary et al., 2017).[1] Such systems face three main

[1] A *lexicalized component* of a MWE, or *component* for short – marked in bold in examples – is one which is always realized by the same lexeme. For instance, the noun *decision* is lexicalized in *to* ***make*** *a* ***decision*** but the determiner *a* is not, because it can be replaced by other determiners (e.g. *to* ***make*** *this/any/the* ***decision***).

Proceedings of the 27th International Conference on Computational Linguistics, pages 2582–2594
Santa Fe, New Mexico, USA, August 20-26, 2018.

challenges: variability, ambiguity and discontinuity (Constant et al., 2017). Variability refers to the fact that the components of the same MWE can occur in different surface forms because of morphological and/or syntactic transformations, as in example (1) vs. (3) below. Variability is closely related to the issue of unseen data in machine learning – items whose surface forms differ between test and training data are usually harder to identify than those seen in identical forms (Augenstein et al., 2017). MWE ambiguity, conversely, makes the identification of seen MWEs harder, because a particular combination of words can be a MWE in some contexts, as in (1) and (2), but not necessarily all, because of *literal readings* (4) and *coincidental co-occurrence* of the MWE components (5) (Savary and Cordeiro, 2018).[2] Finally, external elements can occur in between MWEs' components, both seen and unseen (1)–(3). Discontinuity is known to be a challenge notably for methods based on sequence labeling.

(1) *He **broke** the **heart** of his class mate.*
(2) *He **broke** her young **heart** when he **let** her **down**.*
(3) *Just think of all the **hearts broken** by him.*
(4) *He broke her chocolate heart when he stepped on it.*
(5) *When her toy broke, her young heart was in sorrow.*

In this paper we address the three aforementioned challenges at a time by considering a sub-problem of MWE identification, namely *MWE variant identification*, that is, the identification of occurrences of known (seen) MWEs and their linguistic variants. For instance, suppose that we have seen the annotated MWE occurrence in (1) in some training data. We would like to correctly predict that the same MWE occurs also in (2) and (3) but not in (4) and (5). Notice that we do not aim at identifying the MWE ***let down*** in example (2) if it has not been previously seen. This focus on MWE variants has important theoretical and practical motivations discussed in Sec. 2.

MWEs are known to have specific variability profiles, e.g. the idiom in (1)–(3) admits passivization and noun inflection, while others do not, e.g. *he was **pulling** my **leg*** 'he was kidding me' vs. *he was pulling my legs*; *my leg was pulled*. Our research question is whether this MWE-specific variability profile may be captured automatically and leveraged for high-quality MWE variant identification. The idea is that defining a variability profile in terms of linguistically informed variation-oriented features should help, on the one hand, to bring different variants of the same MWE together and, on the other hand, to distinguish idiomatic occurrences of a MWE from its literal readings. We are specifically interested in morphological and syntactic variants of verbal MWEs (VMWEs). This phenomenon is particularly challenging in morphologically rich languages, notably due to the internal inflection of the VMWE components and to the relatively free word order. In this paper we are interested in VMWEs in French, which exhibits rich verbal inflection and moderate inflection of nouns and adjectives.

This paper is organized as follows. We discuss the state of the art in MWE variant modeling and identification (Sec. 2). We address aspects of VMWE variability in French (Sec. 3), and we describe the corpus used for our experiments (Sec. 4). Follows a presentation of our a baseline variant identification method (Sec. 5). We then propose linguistic features for classification and their extraction process, as well as the architecture of our VMWE variant identification system (Sec. 6). Finally, we present and analyze our results (Sec. 7), before we conclude and propose perspectives for future work (Sec. 10).

2 Related work

Variability, often referred to as flexibility, has been considered a key property of MWEs in various linguistic studies in the past. Gross (1988) analyzes French adjective-noun compounds and shows that their fixedness (the contrary of flexibility) is a matter of scale rather than a binary property. Tutin (2016) confirms this hypothesis with a corpus study of the 30 most frequent verb-noun expressions in French, and defines 6 variability levels. Nunberg et al. (1994) hypothesize a strong correlation between semantic decomposability and syntactic flexibility. They conjecture that MWEs such as *to **pull strings*** admit a large range of variants (e.g. *to **pull** political **strings**, the many **strings** he **pulled***) because their components *pull* and *strings* can be paraphrased by *use* and *influence*. This hypothesis has been criticized by Sheinfux et al. (2017), who stipulated that the degree of flexibility of Hebrew VMWEs depends on the links

[2] Literal readings and coincidental occurrences of MWEs are maked by wavy and dashed underlining, respectively.

between their literal and idiomatic readings rather than on their semantic decomposability. Inflectional and syntactic flexibility were also a major criterion for classifying MWEs into fixed, semi-fixed (subject to internal inflection only) and syntactically flexible expressions by Sag et al. (2002). This scale-wise variability challenges the traditional lexicon vs. grammar division of language modeling.

MWE variability has also been considered a major challenge in various NLP models and applications. Firstly, variants are pervasive. Jacquemin (2001) studies multiword nominal terms in French and English and shows that as many as 28% of their occurrences in texts correspond to variants of canonical forms listed in lexicons. Secondly, variants challenge automatic identification of MWEs, as discussed below. Finally, variant conflation is crucial for downstream applications such as information extraction (Savary and Jacquemin, 2003) and entity linking (Hachey et al., 2013).

Existing approaches to MWE identification partly model MWE variability to some extent. *Rule-based methods* often rely on MWE lexicons and a matching procedure. They sometimes extensively address MWE variation by morphological processors combined with rich finite-state patterns (Krstev et al., 2014) or unification meta-grammars (Jacquemin, 2001). Rule-based methods were often combined with statistical measures in the domain of multiword term extraction (Savary and Jacquemin, 2003), and explicitly addressed variation. But they can hardly distinguish literal from idiomatic readings, weakly cover discontinuous MWEs, and cannot generalize to MWEs absent from the lexicon and to new, initially uncovered, variation patterns. *Sense disambiguation methods* often focus on ambiguous known MWEs and neglect variant identification (Fazly et al., 2009). *Sequence taggers* learn identification models from annotated data based on regularities in sequences of tokens (Constant et al., 2013; Schneider et al., 2014). They are well suited for continuous seen MWEs and neutralize inflection when lemmas are available. They cope, however, badly with syntactic variation whenever it leads to discontinuities. *MWE-aware parsers* identify MWEs as a by-product of parsing a sentence (Green et al., 2013; Constant and Nivre, 2016). They often can deal with both morphological and syntactic variants, even though they are usually less accurate for highly discontinuous and variable MWEs, given that parsing is a hard problem by itself, and MWE-specific features increase data sparseness. *Parsing-based MWE identifiers* use generic parsers as providers of MWE candidates for classification (Vincze et al., 2013). They are less subject to data sparseness and can cope with discontinuities, provided that the underlying parser correctly handles long-distance dependencies. However, they may under- or overgenerate some syntactic transformations allowed by a MWE, e.g. indirect dependencies, as in Fig. 1.c and 1.f.

As a conclusion, it seems that MWE variant identification, although crucial both for corpus-based linguistic studies and for downstream NLP applications, has not yet received a satisfactory solution. Moreover, performances in solving this task are rarely explicitly reported on. Instead, the performances on seen vs. unseen data are addressed, and it is not always clear how these notions are understood. For instance, Constant et al. (2013) observe that 25% of the MWEs in their test corpus are unseen in the training data, and that at most 19% of them could be correctly identified by their system based on sequence labeling. However, the authors do not specify how unseen MWEs are defined, that is, if variants are counted as seen or unseen. The work by Maldonado et al. (2017) defines seen MWEs – similarly to our understanding – as MWEs present in the test set and sharing the same dependency structure, POS tags and lemmas with at least one annotated MWE in the training set. They find out that most systems perform better when the proportion of seen MWEs is high. This result suggests that MWE variant identification remains a hard problem and deserves being addressed explicitly by dedicated methods.

3 VMWE variation in French

VMWEs are known to have specific lexical and morpho-syntactic *variability profiles*: they are more or less variable, but usually not as variable as regular phrases with the same syntactic structure. Therefore, methods used for detecting paraphrases of regular phrases (Fujita and Isabelle, 2015) cannot be straightforwardly applied to VMWEs. Different aspects of variability may be considered.

Firstly, MWE-hood is, by nature, a lexical phenomenon, that is, a particular idiomatic reading is available only in presence of a combination of particular lexical units. Replacing one of them by a semantically close lexeme usually leads to the loss of idiomatic reading, e.g. (6) is an idiom but (7) can

only be understood literally. Lexical variability is admitted by some VMWEs, but usually with a very restricted list of equivalents only, as in (11) and (12). In this work, unlike Fazly et al. (2009), we exclude lexical variability from our scope. More precisely, two VMWE occurrences are considered *morpho-syntactic variants* of each other if they have the same (idiomatic) meaning and are *lexically identical*, that is, the multisets of the lemmas of their lexicalized components are identical. For instance, (12) is considered a separate VMWE rather than a variant of (11). Also, (16) is not a variant of (13) due to the determiner which is lexicalized in the latter but not in the former. Note that textually identical occurrences of a VMWE, like (13) vs. (15) are also considered variants.

(6) *Léa* ***tourne la page*** '*Léa turns the page*'⇒'Léa stops dealing with what occupied her'
(7) *Léa pivote la page* '*Léa rotates the page*'; *Léa tourne la feuille* '*Léa turns the sheet*'
(8) ***tournera****-t-elle* ***la page*** *de la politique?* '*turn-will-she the page of the politics?*'⇒'Will she stop dealing with politics?'
(9) ***la page*** *a été* ***tournée*** '*the page has been turned*'⇒'the subject is no longer dealt with'
(10) *elle tourne les pages de la politique* '*she turns the pages of politics*'
(11) *Ses plaintes me* ***cassent les oreilles*** '*his moans break my ears*'⇒'his moans annoy me'
(12) *Ses plaintes me* ***cassent les pieds*** '*his moans break my feet*'⇒'his moans annoy me'
(13) ***faire le tour*** *du sujet* '*make the tour of the topic*'⇒'consider all aspects of the topic'
(14) ***refaire le tour*** *du sujet* '*remake the tour of the topic*'⇒'reconsider all aspects of the topic'
(15) *on va en* ***faire le tour*** '*we will make the tour of it*'⇒'we will consider all aspects of it'
(16) ***faire*** *un* ***tour*** *en famille* '*make a tour in family*'⇒'go out with one's family'
(17) *le tour fait 3 kilomètres* '*the tour makes 3 kilometers*'⇒'the tour is 3 km long'
(18) ***filer le*** *parfait/grand.ADJ* ***amour*** '*spin the perfect/great love*'⇒'to live a perfect/great love'
(19) *Léa* ***prenait*** *souvent* ***la porte*** '*Léa took often the door*'⇒'Léa was often forced to go out'
(20) *la porte de la maison a été prise par Léa* '*the door of the house was taken by Léa*'

Another aspect of VMWE variability is morphological: some VMWE components do not admit inflection, as the noun *page* '*page*' in (8) vs. (10), while others do, as the verb *tourner* '*turn*' in (8). In rare cases, the head noun accepts derivation, often with prefixes like *re-*, to express repetition, as in (14).

Syntactic variability is also crucial for the VMWE variability profile. Firstly, idiomatic readings most often correspond to literal readings used metaphorically. Therefore, the VMWE components have to occur in a syntactic configuration which relates to the literal reading. For instance, (17) is not a variant of (13), since *tour* takes a different semantic role. Still, both the literal and the idiomatic meaning can sometimes be preserved under syntactic variation as in (6) vs. (9). Secondly, some types of syntactic variation (e.g. passivization) tend to be exhibited less frequently by VMWEs than by non-VMWEs of the same syntactic structures (Fazly et al., 2009). Thirdly, some types of syntactic dependencies can be specific to some VMWEs, e.g. (18) involves a compulsory though non-lexicalized adjectival modifier of the noun. This also means that this VMWE admits insertions of external elements between its lexicalized components. Finally, as previously shown (Pasquer et al., 2018), syntactic features are particularly strong indicators for linguistically motivated similarity and flexibility measures of (French) VWMEs.

The main challenge in modeling the variability profiles of VMWEs lies in the fact that VMWEs of the same syntactic structure may behave differently. For instance, the VMWEs in (6) and (19) both admit interrogation, insertions and inflection of the verb, and prohibit inflection of the noun. Still, modification of the noun and passivization is exhibited by the former (8)–(9) but not by the latter (20).

4 Corpus

We study VMWE variability on a corpus of French texts annotated with VMWEs for the PARSEME shared task.[3] In addition to VMWE annotation (based on universal guidelines), the corpus contains POS, lemmas, morphological features and dependency structures, which we use in our experiments. The original release of the VMWE-annotated corpus contained two sub-corpora: the French part of the Universal Dependencies v1.4 corpus (Nivre et al., 2016) and Sequoia (Candito and Seddah, 2012).

[3] `http://multiword.sf.net/sharedtask2017`, corpus at `http://hdl.handle.net/11372/LRT-2282`

Corpus	All POS patterns				All Verb-(Det-)Noun		Verb-(Det-)Noun variants	
	# Sentences	# Tokens	# VMWEs	# occ.	# VMWEs	# occ.	# VMWEs	# occ.
TrC	17,880	450,221	1,584	4,462	854	2098	n/a	n/a
TeC	1,667	35,784	291	500	177	283	86	132

Table 1: Number of different VMWE types (# VMWEs) and their token occurrences (# occ.) per POS and variability class in the training corpus (TrC) and in the test corpus (TeC).

The tagsets and dependency trees used in both sub-corpora were incompatible. We homogenized them by replacing the released annotations with their UDv2-compatible versions, while VMWE annotations remain unchanged.

The VMWEs annotated for French belong to three main categories: inherently reflexive verbs (***se plaindre*** '*pity oneself*'⇒'complain'), light-verb constructions (***prendre** une **décision*** '*take a decision*') and idioms (***tourner la page*** '*turn the page*'⇒'stop dealing with what one was occupied with'). In this study we focus only on verb-noun combinations, possibly involving a lexicalized determiner, hence only the last two VMWE categories are relevant.

Tab. 1 shows the VMWE statistics of the training corpus (TrC) and test corpus (TeC). Note that, with our focus on variants of seen VMWEs having the Verb-(Det-)Noun structure, only 86 VMWEs and their 132 occurrences from TeC are to be predicted. Among them, only 35 occurrences (26.5%) appear under identical surface forms in TeC as in TrC.

5 Baseline variant identification

The aim of our study is to automatically identify morpho-syntactic variants of VMWEs in a syntactically parsed French corpus. More precisely, given a set of VMWEs annotated in TrC, we wish to identify all their morpho-syntactic variants in TeC. Each of such variants to be predicted in TeC will be called a seen-in-train variant (STV). Note that, a given expression e in TeC can be an STV only if it has the same multiset of lemmas as a VMWE e' from TrC. But this condition is not sufficient due to the existence of literal readings and accidental co-occurrences (Sec. 1). We hypothesize that distinguishing such spurious candidates from STVs should be possible by relying on the morpho-syntactic variability profile. In other words, e has good chances to be a morpho-syntactic variant of e' if both are morpho-syntactically similar.

This idea suggests a relatively straightforward baseline STV identification approach, similar to the *BagOfDeps* strategy proposed by Savary and Cordeiro (2018). Given a dependency-parsed TeC, we extract each set of nodes e so that: (i) the multiset of lemmas in e is identical to some VMWE e' in TrC, (ii) e forms a connected dependency graph. Note that this approach neutralizes both morphological variation (via lemmatization) and syntactic variation (we disregard the labels and directions of the dependencies). As a result, the predicted STVs can be either true positives (Fig.1.a and 1.b) or false positives (Fig.1.d). False negatives include occurrences without direct connection like complex determiners (Fig.1.f). The baseline is also sensitive to syntactic annotation errors, as in Fig.1.e (*rôle* '*role*' rather than *clé* '*key*' should be the head of *rôle clé* 'key role'). Despite its simplicity, the baseline is already very strong, as shown in the first line of Tab. 2. It extracts 158 Verb-(Det-)Noun candidates, which – compared to the 132 STVs to be extracted – yield the F-score of 0.88. These results notably confirm previous observations that literal readings are rare (Waszczuk et al., 2016; Savary and Cordeiro, 2018).

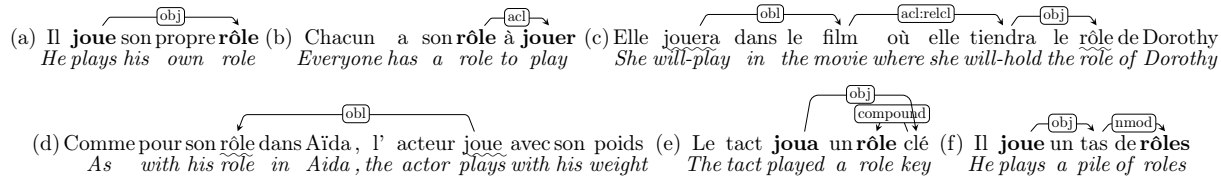


Figure 1: True positive (a,b) and true negative (c) STVs. False positive STV (d), inconsistent syntactic annotation (e), indirect dependencies (e,f)

6 Variant identification as a classification problem

We hypothesize that VMWE variability profiles can be exploited to achieve a better VMWE variant identification than the baseline. Variability profiles will be approximated by a set of morpho-syntactic features (Sec. 6.1). We will first use a more exhaustive candidate extraction method than the baseline, so as to avoid false negatives. Then, the features should help discriminate true STVs from spurious candidates (Sec. 6.2).

6.1 Variation-related features

We describe the profile of any given STV candidate expression e (hereafter called 'candidate' or e) by binary features characterized according to two dimensions:
Morphological, linear and syntactic features model characteristics discussed in Sec. 3, such as verbal/nominal inflection, linear insertions between components (e.g. adjectives, adverbs) or discrepancies between syntactic dependencies (e.g. subject vs. object).
Comparative and absolute features: Comparative features are based on the comparison of e with lexically identical VMWEs from TrC, e.g. if e's noun is found at least once with same inflection in a lexically identical VMWE in TrC, then the COMP_genderNumber(e) feature is True. Absolute features consist in contextual properties of e, e.g. if e's noun is in singular, the numberSing(e) feature is True.

6.1.1 Comparative features (COMP)

COMP features verify the correspondence of e's properties with those of VMWEs annotated in TrC. Henceforth, we say that a property of e matches if it is the same for e and for at least one lexically identical annotated VMWE in TrC. COMP features are binary (their value is False when no match is found and True otherwise) and belong to 3 types: morphological, linear and syntactic.
Morphological similarity is represented by a single feature, COMP_genderNumber, which focuses on the nominal inflection only, since previous experiments showed that it is more informative than verbal inflection (Pasquer et al., 2018). Thus, COMP_genderNumber(e) is True if e's inflection matches.
Linear similarity features account for different degrees of similarity:

- COMP_insertRaw: True if the POS sequence of insertions between lexicalized components of e matches. For instance, if e has an Adj-Adj-Det insertion, it can only match a VMWE having the identical sequence of insertions, Adj-Adj-Det.
- COMP_insertWithoutDuplicate: Same as above, ignoring duplicated POS, so Adj-Adj-Det can match both Adj-Adj-Det and Adj-Det. This feature neutralizes duplications, as we believe that MWEs have constraints on the POS of allowed insertions, but not on their number (e.g. it is rare that one adjective is tolerated but not two).
- COMP_insertPartial: True if a POS substring of more than 50% and less than 100% of insertions in e matches. This feature identifies the similarity between ***play*** *a*.DET *very*.ADV *important*.ADJ ***role*** and ***play*** *his*.DET *first*.ADJ *major* ***role***. However, linear similarity does not capture postnominal adjectives (e.g. ***jouer*** *un* ***rôle*** *important* 'play a role important'), hence the interest in syntactic similarity.

Syntactic similarity Like for morphological features, we noticed greater influence of the outgoing dependencies of the noun, as in Fig.1.b (contrary to incoming dependencies in Fig.1.a), hence the focus on the noun. These syntactic features include:

- COMP_depSynNounTotal, COMP_depSynNounPartial: True if 100%, or more than 50% and less than 100%, of the noun's outgoing dependencies match, respectively.
- COMP_distSyn : True if the syntactic distance between the verb and the noun matches. The *syntactic distance* between two components is defined as the number of elements in the syntactic dependency chain between these two components, regardless of the direction of the dependencies and excluding the components themselves. Inside known VMWEs, the syntactic distance never exceeds 2, so that the value is set to ">2" when this case occurs. In Fig. 1.c this chain is composed of *jouera-film-tiendra-rôle*. Supposing that all the VMWEs of the corpus are represented in Fig. 1, the syntactic distance is 2, hence COMP_distSyn = False since it differs from all the annotated VMWEs.

- COMP_typeDistSyn: True if the type of the syntactic distance matches. This type takes dependency directions into account, and can be serial (in Fig. 1.f, the noun *rôles* '*roles*' depends on *tas* '*pile*', which itself depends on the verb *jouer* '*play*'), parallel (in Fig. 1.e, both *jouer* '*play*' and *rôle* '*role*' depend on the non-lexicalized component *clé* '*key*') or "nonEvaluated" when the syntactic distance exceeds 2 elements.

We expect COMP features to be highly relevant. For instance, the noun in many VMWEs must remain in singular, as in (8) vs. (10), or in plural, as in (11). However, COMP features might fail in case of rare VMWEs, and cannot be calculated for hapaxes. Therefore, absolute features should also be useful.

6.1.2 Absolute features (ABS)

For each candidate, the set of absolute features includes the following.

Lemmas of the components can be considered individually or as a sequence. When considered individually, the VMWE in (6) has ABS_verbLemma = *tourner*, ABS_nounLemma = *page*, and ABS_detLemma = *le* (when there is no lexicalized determiner, ABS_detLemma = noDet). When considered as a sequence, we sort lemmas lexicographically to neutralize variable word order, as in (6) vs. (9), and obtain the *Normalized Form* (NF).[4] Thus, *le;page;tourner* 'the;page;turn' is the ABS_NF of the VMWEs in (6)–(10).

Morphological features for the noun are: ABS_numberSing, ABS_numberPlur, ABS_genderMasc, ABS_genderFem. Their value is True when the respective property is satisfied. For the verb, we only consider derivational inflection. Namely, the ABS_verbPrefix feature is True only if the verb starts with one of the most frequent repetition (*re/ré/r*) or negation prefixes (*de/dé*) and if the verb without this prefix matches the verb in any VMWE annotated in TrC, e.g. ABS_verbPrefix is True for the VMWE in (14).

Linear features correspond to insertions between lexicalized components. We add one insertion feature per possible POS, which is set to True if the e has an insertion with the given POS. For instance in ***effectuer*** *une*.DET *bonne*.ADJ *première*.ADJ ***saison*** 'make a good first season', we obtain ABS_insert_DET = True, ABS_insert_ADJ = True, ABS_insert_ADV = False, etc.

Syntactic features are defined similarly to the linear ones. We introduce one feature per possible dependency relation, and set it to True if the noun in e has at least one outgoing dependency arc with this relation (e.g. ABS_relDepNoun_amod = True for the previous example).

Syntactic distance between the verb and the noun is defined as before, i.e. as the number of elements in the syntactic dependency chain except for the components themselves.

Type of syntactic distance also follows the one in COMP features, and can be either serial or parallel. COMP features are far less numerous than ABS features (9 vs. 72 in the example detailed in App. A).

6.2 Overview of the method

Our goal is to extract STVs, i.e. variants of previously seen VMWEs. We wish to enhance over the baseline (Sec. 5), which is sensitive to indirect dependencies, to dependencies irrelevant to variation, and to annotation errors (which may cause a dramatic performance drop with automatically parsed data), as shown in Fig. 1.d–f. We thus propose an enhanced approach in two steps. First, a more relaxed variant candidate extraction is designed so as to achieve a better recall than the baseline (usually at the expense of a dramatic drop in precision), by extracting, for each annotated e' in TrC, its lexically identical candidates, disregarding syntax. Then, the extracted candidates are filtered by a binary classifier based on the features described above. Fig.2 illustrates the two most crucial phases of this process: the candidate extraction and the feature extraction. VMWE candidates from TrC are numbered (TrC1)–(TrC9), as shown in the first gray box. Those from TeC, (TeC1)–(TeC5), are shown in the first white box. Training of the classifier itself, and its application to prediction are standard and disregarded in Fig.2.

6.2.1 Variant candidate extraction

The easiest way to obtain a larger variant candidate extraction is to search, in each sentence, the simultaneous presence of all components of known VMWEs whatever their order, as described by (Savary

[4]The drawback of NF is that it ignores nominal inflection that could help distinguish between (very rare) VMWEs with different surface forms (e.g. ***fermer l'oeil*** '*close the eye*'⇒'sleep' vs. ***fermer les yeux*** '*close the eyes*'⇒'turn a blind eye').

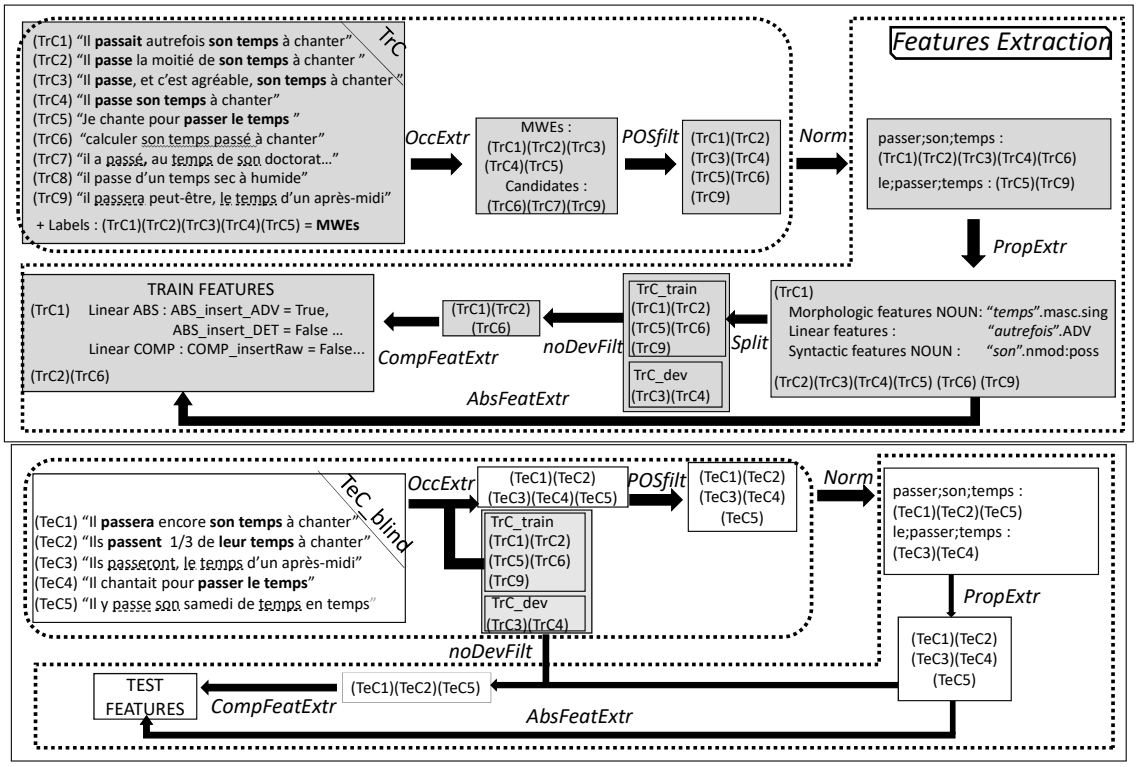


Figure 2: Candidate and feature extraction in training (above) and during prediction (below)

and Cordeiro, 2018). This phase is called *OccExtr*. Consider the first gray box in Fig.2, representing TrC, and several expressions related to two VMWEs ***passer son temps*** 'spend one's time' and ***passer le temps*** '*spend the time*'⇒'invent activities when one has nothing to do'. (TrC1)–(TrC5) are true VMWEs, (TrC6) is a literal reading, (TrC7) and (TrC9) are accidental co-occurrences, and (TrC8) is not lexically identical to any of the two VMWEs due to a missing determiner. Since *OccExtr* matches only occurrences lexically identical to known VMWE, it will extract, trivially, (TrC1)–(TrC5), and (TrC6)(TrC7)(TrC9), but not (TrC8).[5] *OccExtr* is expected to improve recall but precision should also significantly decrease. To limit possibly huge levels of noise, *OccExtr* extracts candidates in the shortest window. For instance, in (TeC5) only one candidate is extracted despite the repetition of the noun *temps* '*time*'.[6] *OccExtr* is enhanced by additional filtering (*POSfilt*) which eliminates irrelevant POS orders (i.e. incorrect in French), such as Verb-Noun-Det in (TrC7). Finally, the NFs for all candidates are produced (*Norm*).

6.2.2 *mweVIDE*: Binary classification of candidates

As a result of the previous phase, we obtain sets of candidates grouped by their NFs (cf. upper-right-hand box). Morpho-syntactic properties for these candidates are then extracted (*PropExtr*) and the whole set is randomly divided (*Split*) into two equal subsets TrC_train and TrC_dev, so as to be able to calculate COMP values. For a given candidate e from TrC_train, when no VMWE with the same NF is annotated in TrC_dev, e is deleted (*noDevFilt*), as is the case of (TrC6) and (TrC9). This may lead to a loss of almost 30% of TrC_train but is inevitable, since for each candidate we need at least one lexically identical VMWE in TrC_dev to calculate the values of the comparative features. The final set of positive and negative candidates is then represented by COMP and ABS features (*CompFeatExtr* and *AbsFeatExtr*), and fed into classifier training.

In the prediction phase (lower part of Fig. 2), the candidates are extracted from TeC on the basis of their

[5]Note that *OccExtr* in training (upper part of Fig.2) applies to TrC itself, since classification calls for both positive and negative examples. Conversely, in prediction, *OccExtr* is applied to the blind version of TeC.

[6]This may lead to both false negatives and false positives, as in *la* ***collaboration*** *que nous avons.[AUX=avoir 'have']* ***eue****.[VERB=avoir 'have']* 'the collaboration that we have had'.

lexical identity (i.e. NF-identity) with annotated VMWEs from TrC. They are *POSfilt*ered, *Norm*alized and subject to *Prop*erty *Extr*action, as in training. But no *Split* needs to be performed, since each TeC candidate can be compared to a VMWE in TrC_dev. The 3 candidates remained after *noDevFilt* are then represented as COMP and ABS features and classified into STVs or non-STVs.

We use the NLTK Naive Bayes classifier.[7] In the training phase it takes as input the set of positive and negative examples represented as feature-value pairs, together with their classes (STV/non-STV), and outputs a model. In the prediction phase, it takes as input the trained model and the candidates to classify, represented as feature-value pairs, and for each candidate outputs an STV or a non-STV label. Training and prediction are repeated 10 times with random TrC_train/TrC_dev split, corresponding to half of TrC each. The classification performance (recall, precision, F1-measure) is evaluated on the manually annotated version of TeC. For each of the 10 training and prediction turns, we also obtain the 100 most informative features associated to the (non-)STV label prediction.

7 Results

We firstly compare the performance of *OccExtr* with the baseline. Table 2 shows the performances of the systems on TrC (2098 occurrences) and TeC corpora (132 occurrences). As expected, the precision

	TrC #candidates	R	P	F1	TeC #candidates	R	P	F1
Baseline	2414	0.970	0.85	0.91	158	0.97	0.81	0.88
OccExtr	5364	0.999	0.39	0.56	484	1.00	0.27	0.43
mweVIDE	5364	n/a	n/a	n/a	484	0.97 (σ_R = 0.007)	0.87 (σ_P = 0.015)	0.92 (σ_{F1} = 0.010)

Table 2: Performance measure of the baseline, OccExtr and mweVIDE on TrC and TeC (average on 10 random TrC_train vs. TrC_dev splits)

of *OccExtr* is significantly lower than of the baseline. Conversely, both recall scores remain very close at a high level of performance. An almost perfect recall is reached by *OccExtr*, but its very low F-measure (0.56 and 0.43 on TrC and TeC, respectively) prevents its direct application to variant detection.

The classification of the candidates implemented in mweVIDE aims at improving the precision of *OccExtr* without impacting recall. As shown in Table 2, mweVIDE increases the F1-measure to 0.92 (vs. 0.88 for the baseline) with a better precision (0.87 vs. 0.81). Recall value is similar to the baseline. The low standard deviation of the F-score (0.01) we observed on the 10 experiments is a good indication of the statistical significance of this improvement. Note that, for this evaluation, we do not have access to data annotated directly with ground truth, i.e. describing which VMWEs are variants of each other. We rely instead on the comparison between the predicted and annotated VMWEs, hence a possible bias because of (i) distinct VMWEs with identical NFs, (ii) similar VMWEs with distinct NFs (due lemmatization errors).

8 Error analysis

In this section we perform a qualitative analysis of the errors performed by the baseline and mweVIDE, in order to investigate what kinds of situation are correctly handled by each system, and which leave room for improvement. The baseline cannot identify components without direct dependency (Fig.1.f) or with parallel dependencies (Fig.1.e). Other omitted cases include: (i) literal readings, (ii) overlap with another VMWE, (iii) coordination, (iv) enumeration, (v) co-reference : ***réunion*** *qui interviendra après celles* ***tenues*** 'meeting which will occur after those held' (where *those* = *meeting*), even though this specific case should not have been annotated according to the annotation guidelines.

As to mweVIDE, it may suffer from no similar VMWEs in TrC (e.g. neither similar syntactic distance, nor insertions or nominal inflection). After 10 evaluations on TeC, the predicted (non-)STV labels are

[7] NLTK (`http://www.nltk.org/`) is used with default parameter values, including ELEProbDist (Expected Likelihood Estimation based on Laplacian smoothing with alpha value = 0.5). Preliminary experiments with a linear SVM classifier (NLTK SklearnClassifier) yielded slightly lower performances than Naive Bayes. Given the low amount of the training data, Naive Bayes seems a satisfactory trade-off compared to other classification methods which require larger data (e.g. polynomial SVM).

STV label feature	$\overline{ratio}$	σ_{ratio}	non-STV label feature	$\overline{ratio}$	σ_{ratio}
COMP_insertWithoutDuplicate=True	28.7	5.3	ABS_insert_VERB=1	123.7	53.5
COMP_insertRaw=True	28.4	5.8	COMP_typeDistSyn=False	121.5	38.7
ABS_NF=jouer;rôle	14.8	3.6	ABS_typeDistSyn=parallel	71.4	39.5
ABS_nounLemma=fin	10.7	4.8	ABS_insert_CCONJ=1	70.4	30.9
ABS_nounLemma=face	9.9	2.5	COMP_distSyn=False	61.0	23.4
ABS_NF=fin;mettre	8.7	2.2	ABS_distSyn=2	56.4	20.1
ABS_distSyn=0	8.7	0.8	ABS_insert_SCONJ=1	52.5	18.3
COMP_distSyn=True	8.6	0.8	ABS_insert_PUNCT=1	29.5	7.9
ABS_NF=faire;partie	8.0	1.6	ABS_distSyn=1	27.9	8.4
ABS_verbLemma=jouer	7.6	2.5	ABS_insert_PROPN=1	25.3	8.2

Table 3: The most informative features (COMP in gray) according to the averaged likelihood ratio for (non-)STV prediction over 10 tests.

constant in 98.9% of the cases. Moreover, despite the *noDevFilt*, 99.2% of the candidates could be classified at least once. Among the 19 false positives, 5 should actually have been annotated as VMWEs. Cases of coordination, never identified by the baseline, are partially identified by mweVIDE, which seems to better handle the omission of a coordination conjunction than its addition. The better precision is due to our POS filtering which excludes irrelevant patterns and mainly to the typology of insertions in known VMWEs used by the classifier. For instance, subordinations between the components are less frequent in VMWEs than in coincidental co-occurrences as will be exposed in the next section.

9 Most informative features

Likelihood ratios associated with each feature allow us to determine which features are the most discriminative in distinguishing STV vs. non-STV candidates. For instance, if the ratio is equal to 2 for a given feature and label, this means that this feature is twice more frequently associated with this label than with the complementary label in the training set. By averaging these ratios on the 10 evaluations, we can extract the ten most informative ones for prediction (Tab. 3). The ranking presented in Tab. 3 is given with mean and standard deviation ratio values: the standard deviation is a good indication that, whatever the selected TrC_train corpus, the same features tend to favor STV (resp. non-STV) labels.

9.1 Features relevant for the STV label

As shown in Tab. 3, features which favor the STV label are both of COMP and ABS type.

COMP features The insertion-related features (COMP_insertRaw=True and COMP_insertWithoutDuplicate=True) are always ranked first. This is understandable, since with fewer insertions the number of potential inserted POS sequences is lower. In TrC, 97% of the VMWEs have less than 5 inserted elements vs. 30% of false STV candidates, which reflects how much non-STVs are susceptible to be more variable in terms of insertions. An identical syntactic distance (COMP_distSyn=True) often means that its value is 0 in a seen VMWE and its STV. False STVs tend to overpass this value because the syntactic connection between elements is often looser than inside a VMWE.

ABS features The relevance of ABS_distSyn=0 (which is the only criterion of the baseline) can be explained in the same way as COMP_distSyn=True above. The other most informative features are NFs (like *jouer;rôle* '*play role*') or isolated components (only the verb *jouer* '*play*'), which means that they are less frequently associated with coincidental co-occurrences or literal readings.

9.2 Features relevant for the non-STV label

Contrary to the prediction of the STV label, COMP features have negative values for non-STV labels, which highlights how much similarity (respectively dissimilarity) with known VMWEs is an important criterion for the STV (resp. non-STV) prediction.

COMP features The only discriminative COMP features are the syntactic distance and its type, when different from known VMWEs.

ABS features Insertions and syntactic distance are the most discriminating. All elements that tend to break the link between components are relevant. In descending order, it can be the insertion of a verb, a coordinating or subordinating conjunction, a punctuation or a proper noun. As for syntactic features, the most relevant here are the syntactic distance higher than zero or parallel syntactic distance.

10 Conclusions and future work

We developed a system for Verb-(Det-)Noun VMWE variant identification based on morphological and syntactic profiles of candidates which permit their classification as (non-)variant thanks to a set of absolute and comparative features (the latter relying on the comparison with known VMWEs). The system shows satisfactory performance (F1-measure = 0.92) and good reproducibility. This represents an improvement over the baseline based on the presence of a syntactic connection between components. Morphology does not appear as a discriminating feature to label a candidate as a STV, contrary to comparative linear features (i.e. similar POS insertions), which were rarely taken into account in previous works (Fazly et al., 2009). In general, comparative features appear to be more relevant for STV prediction. Conversely, absolute features predominate in non-STV variant prediction, since non-STVs often exhibit unreferenced variation profiles. Given the reduced size of our test corpus, we need further evaluations on larger corpora to improve our variant identification system, notably to handle cases of coordinations that are not systematically identified. We also project to include other comparative features such as orthographic similarity (***show one's true colours/colors***) or lexical similarity (***take a bath/shower***), which might help classify rare VMWEs that could not be evaluated here. A wider range of patterns than only Verb-(Det-)Noun should also be considered. Finally, future extension of our system to other languages available in the PARSEME corpus, whether Romance or not, could also be considered.

Acknowledgements

This work was funded by the French PARSEME-FR grant (ANR-14-CERA-0001). We are grateful to the anonymous reviewers for their useful comments.

References

Isabelle Augenstein, Leon Derczynski, and Kalina Bontcheva. 2017. Generalisation in named entity recognition. *Comput. Speech Lang.*, 44(C):61–83, July.

Timothy Baldwin and Su Nam Kim. 2010. Multiword expressions. In Nitin Indurkhya and Fred J. Damerau, editors, *Handbook of Natural Language Processing*, pages 267–292. CRC Press, Taylor and Francis Group, Boca Raton, FL, USA, 2 edition.

Marie Candito and Djamé Seddah. 2012. Le corpus Sequoia : annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical. In *Proceedings of TALN 2012*, juin.

Matthieu Constant and Joakim Nivre. 2016. A transition-based system for joint lexical and syntactic analysis. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 161–171, Berlin.

Matthieu Constant, Joseph Le Roux, and Anthony Sigogne. 2013. Combining compound recognition and PCFG-LA parsing with word lattices and conditional random fields. *TSLP Special Issue on MWEs: from theory to practice and use, part 2 (TSLP)*, 10(3).

Mathieu Constant, Gülşen Eryiğit, Johanna Monti, Lonneke van der Plas, Carlos Ramisch, Michael Rosner, and Amalia Todirascu. 2017. Multiword expression processing: A survey. *Computational Linguistics*, 43(4):837–892.

Afsaneh Fazly, Paul Cook, and Suzanne Stevenson. 2009. Unsupervised type and token identification of idiomatic expressions. *Computational Linguistics*, 35(1):61–103.

Atsushi Fujita and Pierre Isabelle. 2015. Expanding paraphrase lexicons by exploiting lexical variants. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 630–640. Association for Computational Linguistics.

Spence Green, Marie-Catherine de Marneffe, and Christopher D. Manning. 2013. Parsing models for identifying multiword expressions. *Computational Linguistics*, 39(1):195–227.

Gaston Gross. 1988. Degré de figement des noms composés. *Langages*, 90:57–72.

Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R. Curran. 2013. Evaluating Entity Linking with Wikipedia. *Artif. Intell.*, 194:130–150, January.

Christian Jacquemin, 2001. *Spotting and Discovering Terms through Natural Language Processing*. The MIT Press.

Cvetana Krstev, Ivan Obradovic, Milos Utvic, and Dusko Vitas. 2014. A system for named entity recognition based on local grammars. *J. Log. Comput.*, 24(2):473–489.

Alfredo Maldonado, Lifeng Han, Erwan Moreau, Ashjan Alsulaimani, Koel Dutta Chowdhury, Carl Vogel, and Qun Liu. 2017. Detection of verbal multi-word expressions via conditional random fields with syntactic dependency features and semantic re-ranking. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 114–120, Valencia, Spain, April. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), may.

Geoffrey Nunberg, Ivan A. Sag, and Thomas Wasow. 1994. Idioms. *Language*, 70:491–538.

Caroline Pasquer, Agata Savary, Jean-Yves Antoine, and Carlos Ramisch. 2018. Towards a Variability Measure for Multiword Expressions. In *NAACL*, New Orleans, United States, June.

Ivan Sag, Timothy Baldwin, Francis Bond, Ann Copestak, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proceedings of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2002)*, page 1–15, Mexico City,Mexico.

Agata Savary and Silvio Ricardo Cordeiro. 2018. Literal readings of multiword expressions: as scarce as hen's teeth. In *Proceedings of the 16th Workshop on Treebanks and Linguistic Theories (TLT 16)*, Prague, Czech Republic.

Agata Savary and Christian Jacquemin. 2003. Reducing Information Variation in Text. *LNCS*, 2705:145–181.

Agata Savary, Carlos Ramisch, Silvio Cordeiro, Federico Sangati, Veronika Vincze, Behrang QasemiZadeh, Marie Candito, Fabienne Cap, Voula Giouli, Ivelina Stoyanova, and Antoine Doucet. 2017. The PARSEME Shared Task on Automatic Identification of Verbal Multiword Expressions. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*, pages 31–47, Valencia, Spain, April. Association for Computational Linguistics.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A. Smith. 2014. Discriminative lexical semantic segmentation with gaps: running the MWE gamut. *Transactions of the ACL*, 2:193–206.

Nathan Schneider, Dirk Hovy, Anders Johannsen, and Marine Carpuat. 2016. Semeval-2016 task 10: Detecting minimal semantic units and their meanings (dimsum). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 546–559. Association for Computational Linguistics.

Livnat Herzig Sheinfux, Tali Arad Greshler, Nurit Melnik, and Shuly Wintner, 2017. *Verbal MWEs: Idiomaticity and flexibility*, pages 5–38. Language Science Press, à paraître.

Agnès Tutin. 2016. Comparing morphological and syntactic variations of support verb constructions and verbal full phrasemes in French: a corpus based study. In *PARSEME COST Action. Relieving the pain in the neck in natural language processing: 7th final general meeting*, Dubrovnik, Croatia.

Veronika Vincze, János Zsibrita, and István Nagy. 2013. Dependency parsing for identifying Hungarian light verb constructions. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 207–215.

Jakub Waszczuk, Agata Savary, and Yannick Parmentier. 2016. Promoting multiword expressions in A* TAG parsing. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 429–439.

A Absolute and comparative features for a sample STV candidate.

STV candidate in TeC:

- *Elle **fit**.VERB alors.ADV la.DET **connaissance**.NOUN de son futur mari.* '*She made then the knowledge of her future husband*'⇒'She got then acquainted with her future husband'

4 annotated VMWEs in TrC (basis for COMP features):

- *Le film [...] permet de **faire**.VERB la.DET **connaissance**.NOUN d'une kyrielle de personnages.* '*The movie allows to make the knowledge of a myriad of characters*'⇒'The movie allows one to get acquainted with a myriad of characters'
- *Ricky **fait**.VERB la.DET **connaissance**.NOUN de Junito* '*Ricky make the knowledge of Junito*'⇒'Ricky gets acquainted with Junito'.
- *On pense qu'il y **fit**.VERB la.DET **connaissance**.NOUN de Jean Heynlin et de Guillaume Fichet* 'One thinks that he made the knowledge of Jean Heynlin and Guillaume Fichet'It is believed that he got acquainted with Jean Heynlin and Guillaume Fichet
- *Lorsque Jacques exprima le souhait de **faire**.VERB la.DET **connaissance**.NOUN de les autres leaders de le groupe* 'When Jacques expressed the desire to make teh knowledge of the other leaders of the group'When Jack expressed the desire to get acquainted with the other leaders of the group

The table illustrates the calculation of the ABS and COMP features for these examples. ABS_insert_ is followed by the part-of-speech of the considered insertions, and ABS_relDepNoun_ by the outgoing dependencies related to the noun. ∅ means lacking information whereas "_" means that it is underspecified.

COMP features	**ABS features**	**ABS features**
'COMP_depSynNounPartial': True	'ABS_NF': 'connaissance;faire'	'ABS_insert_ADV': True
'COMP_depSynNounTotal': False	'ABS_detLemma': 'noDET'	'ABS_insert_AUX': False
'COMP_genderNumber': True	'ABS_nounLemma': 'connaissance'	'ABS_insert_CCONJ': False
'COMP_insertWithoutDuplicate': False	'ABS_verbLemma': 'faire'	'ABS_insert_DET': True
'COMP_insertRaw': False	'ABS_verbPrefix': 'noPrefix'	'ABS_insert_INTJ': False
'COMP_insertPartial': True	'ABS_insert_ _': False	'ABS_insert_NUM': False
'COMP_distSyn' : True	'ABS_insert_ADJ': False	'ABS_insert_PART': False
'COMP_typeDistSyn': True	'ABS_insert_ADP': False	'ABS_insert_PRON': False
	'ABS_insert_PROPN': False	'ABS_insert_PUNCT': False
	'ABS_insert_SCONJ': False	'ABS_insert_SYM': False
	'ABS_insert_VERB': False	'ABS_insert_X': False
	'ABS_genderFem': True	'ABS_genderMasc': False
	'ABS_numberPlur': False	'ABS_numberSing': True
	'ABS_relDepNoun_acl:relcl': False	'ABS_relDepNoun_acl': False
	'ABS_relDepNoun_advcl': False	'ABS_relDepNoun_advmod': False
	'ABS_relDepNoun_amod': False	'ABS_relDepNoun_appos': False
	'ABS_relDepNoun_aux:pass': False	'ABS_relDepNoun_aux': False
	'ABS_relDepNoun_case': False	'ABS_relDepNoun_cc': False
	'ABS_relDepNoun_ccomp': False	'ABS_relDepNoun_compound': False
	'ABS_relDepNoun_conj': False	'ABS_relDepNoun_cop': False
	'ABS_relDepNoun_csubj': False	'ABS_relDepNoun_dep': False
	'ABS_relDepNoun_det': True	'ABS_relDepNoun_discourse': False
	'ABS_relDepNoun_dislocated': False	'ABS_relDepNoun_expl': False
	'ABS_relDepNoun_fixed': False	'ABS_relDepNoun_flat:foreign': False
	'ABS_relDepNoun_flat:name': False	'ABS_relDepNoun_goeswith': False
	'ABS_relDepNoun_iobj': False	'ABS_relDepNoun_mark': False
	'ABS_relDepNoun_nmod:poss': False	'ABS_relDepNoun_nmod': False
	'ABS_relDepNoun_nsubj:pass': False	'ABS_relDepNoun_nsubj': False
	'ABS_relDepNoun_nummod': False	'ABS_relDepNoun_obj': False
	'ABS_relDepNoun_obl': False	'ABS_relDepNoun_orphan': False
	'ABS_relDepNoun_parataxis': False	'ABS_relDepNoun_punct': False
	'ABS_relDepNoun_reparandum': False	'ABS_relDepNoun_root': False
	'ABS_relDepNoun_vocative': False	'ABS_relDepNoun_xcomp': False
	'ABS_distSyn': False	'ABS_typeDistSyn': 'serial'

Learning Multilingual Topics from Incomparable Corpora

Shudong Hao
Computer Science
University of Colorado
Boulder, CO, USA
shudong@colorado.edu

Michael J. Paul
Information Science
University of Colorado
Boulder, CO, USA
mpaul@colorado.edu

Abstract

Multilingual topic models enable crosslingual tasks by extracting consistent topics from multilingual corpora. Most models require parallel or comparable training corpora, which limits their ability to generalize. In this paper, we first demystify the knowledge transfer mechanism behind multilingual topic models by defining an alternative but equivalent formulation. Based on this analysis, we then relax the assumption of training data required by most existing models, creating a model that only requires a dictionary for training. Experiments show that our new method effectively learns coherent multilingual topics from partially and fully *incomparable* corpora with limited amounts of dictionary resources.

1 Introduction

Multilingual topic models provide an overview of document structures in multilingual corpora, by learning language-specific versions of each topic (Figure 1). Their simplicity, efficiency and interpretability make models from this family popular for various crosslingual tasks, *e.g.,* feature extraction (Liu et al., 2015), cultural difference discovery (Shutova et al., 2017; Gutiérrez et al., 2016), translation detection (Krstovski et al., 2016; Krstovski and Smith, 2016), and others (Barrett et al., 2016; Agić et al., 2016; Hintz and Biemann, 2016).

Typical probabilistic multilingual topic models are based on Latent Dirichlet Allocation (LDA, Blei et al. (2003)), adding supervision on connections between languages. Most models achieve this by making strong assumptions on the training data—they either require a *parallel corpus* that has sentence-aligned documents in different languages (*e.g.,* EuroParl, Koehn (2005)), or a *comparable corpus* that has documents of similar content (*e.g.,* Wikipedia articles paired across languages). These training requirements limit the usage of such models: an adequately large parallel corpus is difficult to obtain, particularly for low-resource languages. For example, only 300 languages are available on Wikipedia,[1] and only 250 languages have more than 1,000 articles. Another common choice for parallel corpus in multilingual research, the Bible, is available in 2,530 languages (Agić et al., 2015).[2] However, studies show that its archaic themes and small corpus size (1,189 chapters) can limit performance (Hao et al., 2018; Moritz and Büchler, 2017). Therefore, the requirement of parallel/comparable corpora for multilingual topic models limits their usage in many situations.

Another line of research focuses on using multilingual dictionaries as supervision (Ma and Nasukawa, 2017; Gutiérrez et al., 2016; Liu et al., 2015; Jagarlamudi and Daumé III, 2010; Boyd-Graber and Blei, 2009). In contrast to parallel corpora, dictionaries are widely available and often easy to obtain. PANLEX, a free online dictionary database, for example, covers 5,700 languages and more than one billion dictionary entries (Kamholz et al., 2014; Baldwin et al., 2010).[3] Thus, a multilingual topic model built on a dictionary rather than a parallel corpus is potentially applicable to more languages.

[1] https://meta.wikimedia.org/wiki/List_of_Wikipedias

[2] Reported by United Bible Societies at https://www.unitedbiblesocieties.org/

[3] https://panlex.org/

Proceedings of the 27th International Conference on Computational Linguistics, pages 2595–2609
Santa Fe, New Mexico, USA, August 20-26, 2018.

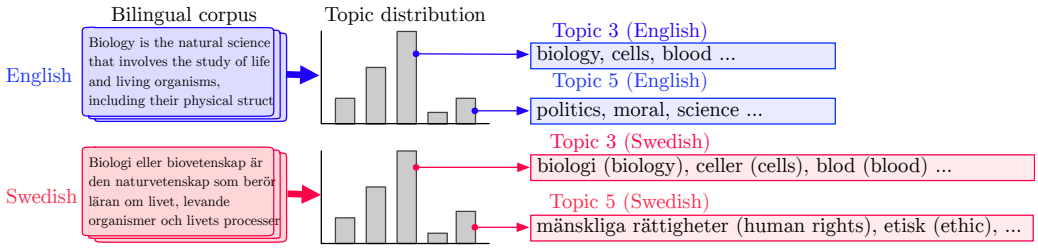


Figure 1: Multilingual topic models produce topics where each language has its own version.

A dictionary also allows for training on *incomparable* corpora—documents in different languages that are from different sources without direct connections—which have had less research on learning consistent topics. With a dictionary, a natural question is how to efficiently utilize the semantic information it carries so that a topic model can produce multilingually coherent topics. This work considers an alternative formulation of a dictionary-based topic model, one that borrows the structure of models used with comparable corpora, but uses a dictionary-based metric to learn connections between documents, instead of explicit connections from a comparable corpus. The main contributions of this work are:

- We summarize existing related work in Section 2 and propose a new formulation of multilingual topic models based on crosslingual transfer learning in Section 3. This new formulation explicitly shows the knowledge transfer mechanism during the generative process.
- Based on this new formulation, in Section 4 we generalize existing multilingual topic models and relax the assumptions of parallel/comparable datasets. Our approach requires only a dictionary, and is empirically shown to perform well even with only limited amounts of available entries.
- We evaluate our new model on five languages from different language families in Section 5. Our proposed model learns multilingually coherent topics and yields around a 25% relative improvement in crosslingual classification performance.

2 Multilingual Topic Models

Multilingual topic models generate K topics from a corpus consisting of multiple languages; each topic has a version specific to each language in the corpus (Figure 1). From a human's view, a coherent multilingual topic should talk about the same thing regardless of the language; from a machine's view, the success of multilingual topic models depends on the inferred topics being consistent across languages. For example, given an English-Swedish bilingual topic $\phi_k^{(\text{EN,SV})}$, the probability of an English word *island* and that of its translation in Swedish, *ö*, should be similar, *i.e.,* $\Pr\left(island_{\text{EN}}|\phi_k^{(\text{EN,SV})}\right) \approx \Pr\left(\ddot{o}_{\text{SV}}|\phi_k^{(\text{EN,SV})}\right)$. Most multilingual topic models extend LDA with one or both of two types of "link" information: document translations and word translations.

Document Links. The polylingual topic model (Mimno et al., 2009; Ni et al., 2009) assumes that during the generative process, a topic distribution $\theta_{\mathbf{d}}$ generates a tuple of comparable documents in different languages, *i.e.,* $\mathbf{d} = \left(d^{(\ell_1)}, \ldots, d^{(\ell_L)}\right)$ and each language ℓ has its own topic-word distributions, $\phi_k^{(\ell)}$. This model has been widely used (Vulić et al., 2013; Platt et al., 2010; Smet and Moens, 2009), but it requires a parallel/comparable corpus in order to link documents.

Vocabulary Links. Another type of model uses word translations (Jagarlamudi and Daumé III, 2010; Boyd-Graber and Blei, 2009) rather than linking documents. A multilingual dictionary is used to construct a tree structure where each internal node contains word translations, and applies hyper-Dirichlet type I distributions to generate words (Andrzejewski et al., 2009; Minka, 1999; Dennis III, 1991). For each topic k, a distribution from root r to all the internal nodes i is drawn by $\phi_{k,r} \sim \text{Dir}(\beta_r)$, and then a distribution from i to a leaf node is drawn by $\phi_{k,i}^{(\ell)} \sim \text{Dir}\left(\beta_i^{(\ell)}\right)$. A word $w^{(\ell)}$ in language ℓ is drawn from a product of the two multinomial distributions by $w^{(\ell)} \sim \text{Mult}\left(\phi_{k,r} \cdot \phi_{k,i}^{(\ell)}\right)$.

Variations. Many variations of these ideas have been proposed to deal with non-parallel corpus. Heyman et al. (2016) proposed C-BILDA, which distinguishes between shared and non-shared topics across languages, based on a document links model. The model, however, requires a comparable dataset that provides document links between languages. A variation proposed by Ma and Nasukawa (2017) deals with non-parallel corpora. This model is essentially a modified version of Jagarlamudi and Daumé III (2010) and Boyd-Graber and Blei (2009), so we consider this work to be another vocabulary links model. Other models have been proposed for very specific situations that needs additional supervision. For example, Krstovski et al. (2016) requires scientific article section alignments, and Gutiérrez et al. (2016) requires Part-of-Speech (POS) taggers, which are not always available for all languages. Without POS taggers, this model is equivalent to vocabulary links. In our work, we focus on the standard document links and vocabulary links models, which are the most generalizable models.

3 Document Links: A Crosslingual Transfer Perspective

Before we introduce our new approach, we first present an alternative understanding of the document links model from the perspective of crosslingual transfer learning. In multilingual topic models, "knowledge" refers to word distributions for a topic in a language ℓ, and we study how multilingual topic models transfer this knowledge from one language to another so that the model provides semantically coherent topics that are consistent across languages.

In the standard document links model, a "link" between a document d_{ℓ_1} in language ℓ_1 and d_{ℓ_2} in ℓ_2 indicates that they are translations or closely comparable. In this model, the topic assignments for both documents are independently generated from the same distribution, $\theta_{d_{\ell 1}, d_{\ell 2}}$. Thus, the joint likelihood of document links model is:

$$\Pr\left(\mathbf{w}_{d_{\ell 1}}, \mathbf{z}_{d_{\ell 1}}, \mathbf{w}_{d_{\ell 2}}, \mathbf{z}_{d_{\ell 2}} | \alpha, \beta\right), \tag{1}$$

where $\mathbf{w}_{d_\ell}$ and $\mathbf{z}_{d_\ell}$ are the word tokens and topic assignments of document d_ℓ. We refer this formulation as the **joint generative model**, since the topics and words of d_{ℓ_1} and d_{ℓ_2} are generated *simultaneously*.

The simultaneousness of this model formulation, in which both languages generate topics jointly, masks the knowledge transfer process. To highlight this process, and to help us generalize the model in the next section, we define an alternative formulation in which d_{ℓ_1} and d_{ℓ_2} are generated *sequentially*.

Assume the topics of d_{ℓ_1} have already been generated from $\theta_{d_{\ell 1}} \sim \mathrm{Dir}(\alpha)$, and $\mathbf{n}_{d_{\ell 1}} \in \mathbb{N}^K$ is a vector of topic counts in d_{ℓ_1}. In our alternative formulation, the generation of topics of d_{ℓ_2} depends on d_{ℓ_1} by $\theta_{d_{\ell 2}} \sim \mathrm{Dir}(\alpha + \mathbf{n}_{d_{\ell 1}})$, where the prior $\alpha + \mathbf{n}_{d_{\ell 1}}$ encourages the distribution $\theta_{d_{\ell 2}}$ to be similar to $\theta_{d_{\ell 1}}$. This formulation can go the other way, *i.e.,* generating d_{ℓ_2} first, and then d_{ℓ_1}. The combined likelihood of this formulation is:

$$\Pr\left(\mathbf{w}_{d_{\ell 1}}, \mathbf{z}_{d_{\ell 1}} | \mathbf{w}_{d_{\ell 2}}, \mathbf{z}_{d_{\ell 2}}, \alpha, \beta\right) \cdot \Pr\left(\mathbf{w}_{d_{\ell 2}}, \mathbf{z}_{d_{\ell 2}} | \mathbf{w}_{d_{\ell 1}}, \mathbf{z}_{d_{\ell 1}}, \alpha, \beta\right), \tag{2}$$

and we refer to this formulation as the **conditional generative model**.

This alternative formulation explicitly shows the knowledge transfer process across languages by shaping the topic parameters for ℓ_2 to be similar to that of the other language ℓ_1, and vice versa. In this formulation, the likelihood of the conditional generative model is different from the joint generative model. In fact, this is an instance of *pseudolikelihood* (Besag, 1975; Leppä-aho et al., 2017), where the joint likelihood of the two documents is approximated as the product of each document's conditional likelihood given the other, *i.e.,* $\Pr(d_{\ell_1}, d_{\ell_2}) \approx \Pr(d_{\ell_1} | d_{\ell_2}) \cdot \Pr(d_{\ell_2} | d_{\ell_1})$. As Leppä-aho et al. (2017) suggests, pseudolikelihood is not a numerically accurate approximation to the joint likelihood; Theorem 1 below, however, states that this formulation yields exactly the same posterior estimations of θ and ϕ.

Theorem 1. *The conditional generative model with document links yields the same posterior estimator to the joint generative model using collapsed Gibbs sampling.*

Proof. See Appendix. □

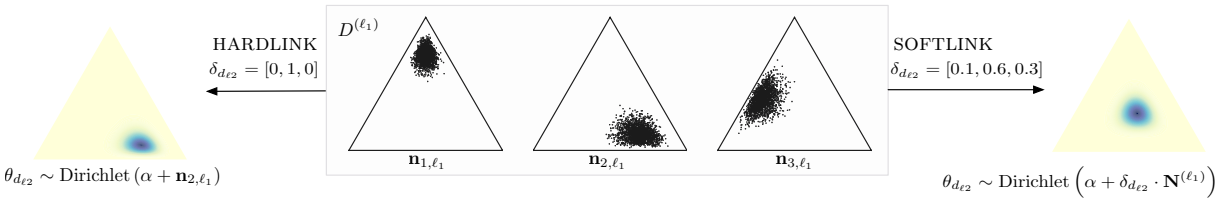


Figure 2: An illustration of how topic knowledge is transferred across languages through HARDLINK and SOFTLINK. To generate observations in d_{ℓ_2}, both models uses topics in ℓ_1 as prior knowledge to shape the Dirichlet prior for d_{ℓ_2}. This transfer happens in HARDLINK by aligned documents in a comparable corpus, while SOFTLINK uses a generalized transfer distribution δ.

4 Generalizing Document Links

Obtaining parallel corpora for training the document links model is very demanding, particularly for low-resource languages. Therefore, as the second major contribution in this paper, we generalize the document links model using the formulation described above to require only a bilingual dictionary.

4.1 From Hard Links to Soft Links

Following the above discussion, we introduce our method assuming the directionality from language ℓ_1 to ℓ_2. We generalize the model above by rewriting the generation of the distribution $\theta_{d_{\ell 2}}$:

$$\theta_{d_{\ell 2}} \quad \sim \quad \text{Dirichlet}\left(\alpha + \delta_{d_{\ell 2}} \cdot \mathbf{N}^{(\ell_1)}\right), \tag{3}$$

where $\mathbf{N}^{(\ell_1)} \in \mathbb{N}^{|D^{(\ell_1)}| \times K}$ is the matrix of topic counts per document with K topics in the corpus $D^{(\ell_1)}$ of language ℓ_1. This is equivalent to the above document links model when $\delta_{d_{\ell 2}} \in \mathbb{R}^{|D^{(\ell_1)}|}$ is an indicator vector that has value 1 for the corresponding parallel document $d_{\ell_1} \in D^{(\ell_1)}$ and 0 elsewhere. We refer this as **hard links** (HARDLINK), where each document $d_{\ell_2} \in D^{(\ell_2)}$ is informed by exactly one document d_{ℓ_1}, and this link is known *a priori* from a parallel corpus.

We create **soft links** (SOFTLINK) by relaxing the assumption that $\delta_{d_{\ell 2}}$ is an indicator vector, instead allowing $\delta_{d_{\ell 2}}$ to be any *distribution* over documents in $D^{(\ell_1)}$, a mixture of potentially multiple documents in language ℓ_1 to inform parameters for a document d_{ℓ_2} in language ℓ_2. We refer this distribution as the **transfer distribution**. The Dirichlet prior for document d_{ℓ_2} contains topic knowledge $\mathbf{N}^{(\ell_1)}$ transferred from corpus $D^{(\ell_1)}$, encouraging $\theta_{d_{\ell 2}}$ to be proportionally similar to documents in $D^{(\ell_1)}$. Figure 2 illustrates this process.

4.2 Defining the Transfer Distribution

The transfer distribution of document d_{ℓ_2} indicates how much knowledge should be transferred from every document $d_{\ell_1} \in D^{(\ell_1)}$. Intuitively, if d_{ℓ_1} and d_{ℓ_2} have a large amount of overlapping word translations, their topics should be similar as well. Therefore, we define the values of δ based on the similarity of document pairs using a bilingual dictionary. Specifically, for a document $d_{\ell_2} \in D^{(\ell_2)}$, the transfer distribution of d_{ℓ_2}, denoted as $\delta_{d_{\ell 2}}$, is a normalized vector of size $|D^{(\ell_1)}|$, *i.e.,* the size of corpus $D^{(\ell_1)}$. Each cell in $\delta_{d_{\ell 2}}$ corresponds to a document $d_{\ell_1} \in D^{(\ell_1)}$, defined as:

$$(\delta_{d_{\ell 2}})_{d_{\ell 1}} \quad \propto \quad \frac{|\,\{w_{\ell_1}\} \cap \{w_{\ell_2}\}\,|}{|\,\{w_{\ell_1}\} \cup \{w_{\ell_2}\}\,|}, \qquad \forall\, w_{\ell_1} \in d_{\ell_1} \text{ and } w_{\ell_2} \in d_{\ell_2}, \tag{4}$$

where $\{w_\ell\}$ contains all the word types that appear in document d_ℓ, and $\{w_{\ell_1}\} \cap \{w_{\ell_2}\}$ indicates all word pairs (w_{ℓ_1}, w_{ℓ_2}) that can be found in a dictionary as translations. In other words, $(\delta_{d_{\ell 2}})_{d_{\ell 1}}$ is the proportion of words in the document pair (d_{ℓ_1}, d_{ℓ_2}) that are translations of each other.

In practice, a dense transfer distribution is computationally inefficient and is less meaningful than a sparse distribution, as it becomes approximately uniform due to the large size of the corpus. The transfer

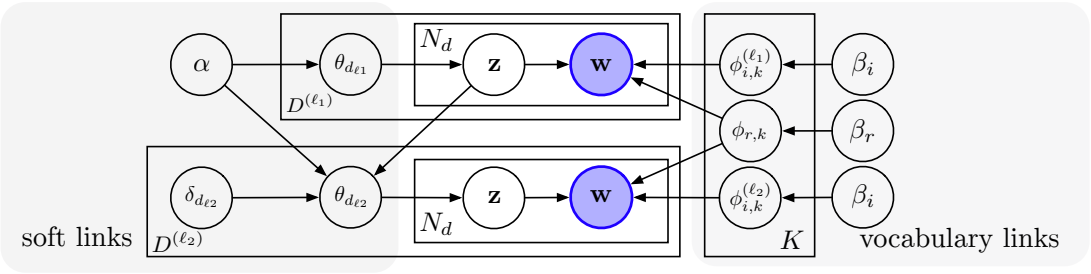


Figure 3: Plate notation of a multilingual topic model using soft links and vocabulary links.

distribution should be more heavily concentrated on documents with higher word-level translation probabilities, while reducing the noise negatively transferred from those with low probabilities. To this end, we propose two approaches to help transfer distributions more efficiently focus on specific documents.

4.2.1 Static Focusing: a Threshold Method

The first method is to focus the distribution on the highest values such that values below a threshold are set to 0, while the remaining values are renormalized to sum to 1. The modified distribution is thus:

$$\left(\widetilde{\delta}_{d_{\ell 2}}\right)_{d_{\ell 1}} \quad \propto \quad \mathbb{1}\left\{ \left(\delta_{d_{\ell 2}}\right)_{d_{\ell 1}} > \pi \cdot \max\left(\delta\right) \right\} \cdot \left(\delta_{d_{\ell 2}}\right)_{d_{\ell 1}} \tag{5}$$

where $\mathbb{1}$ is an indicator function, and $\pi \in [0, 1]$ is the **focal threshold**, a fixed parameter that adjusts the threshold. The threshold is defined with respect to the maximum value of δ. A *corpus-wise* threshold chooses $\max(\delta)$ from all the $\delta_{d_{\ell 2}}$ in $D^{(\ell_2)}$ globally, while we also consider a *document-wise* threshold for each document, $\pi \cdot \max\left(\delta_{d_{\ell 2}}\right)$. We refer these two manners as the **selection scope**.

4.2.2 Dynamic Focusing: an Annealing Method

Static focusing treats transfer distributions δ as fixed parameters during sampling, and it is difficult to decide how sparse a transfer distribution should be to achieve optimal performance. Therefore, we propose dynamic focusing, where we avoid choosing a specific focal threshold and selection scope. Specifically, we adjust the transfer distribution during inference dynamically, beginning with a dense transfer distribution and iteratively sharpening the distribution using deterministic annealing (Ueda and Nakano, 1994; Smith and Eisner, 2006; Paul and Dredze, 2015).

Assume at iteration t, the transfer distribution for a document d_{ℓ_2} is denoted as $\delta_{d_{\ell 2}}^{(t)}$. Then at iteration t', we anneal its transfer distribution by $\left(\delta_{d_{\ell 2}}^{(t')}\right)_{d_{\ell 1}} \propto \left(\delta_{d_{\ell 2}}^{(t)}\right)_{d_{\ell 1}}^{1/\tau}$ where τ is a fixed temperature, which we set to 0.9 in our experiments. We start with non-focused transfer distributions, and apply annealing at scheduling intervals during Gibbs sampling.

Designing an effective annealing schedule is critical. We propose two schedules below.

Fixed Schedule. The simplest schedule is to apply annealing for all transfer distributions every I iterations. In our experiments, we set $I = 10$ (*i.e.,* $t' = t + 10$) and stop annealing after 400 iterations. A potential problem with fixed schedule is that it can "over-anneal" the transfer distributions, *i.e.,* all the mass converges to only one document.

Adaptive Schedule. A robust multilingual topic model should produce similar distributions over topics for a pair of word translations $c = (w_{\ell_1}, w_{\ell_2})$, where we call c a concept. In other words, given a topic k, the probability of expressing a concept i in language ℓ_1 should be similar to language ℓ_2. Thus, during iteration t, we calculate $\varphi_c^{(\ell,t)}$, the distribution over K topics for each concept c for each language ℓ. Using $\varphi_c^{(\ell,t)}$ as features and its language ℓ as labels, we perform five-fold cross-validation by logistic regression for all concepts c. We define the average classification accuracy over the five folds as the **language identification score** (LIS). The lower the LIS, the better the model, since a high LIS means the

inferred distributions are inconsistent enough to discriminate between languages. This idea is related to adversarial training between languages (Chen et al., 2016).

During Gibbs sampling, we calculate LIS after each iteration, and average LIS every I iterations. We anneal all transfer distributions at iteration t only if $\overline{\text{LIS}}_{t-I:t} > \overline{\text{LIS}}_{t-2I:t-I}$. That is, if the average LIS score during iteration $t-I$ and t has been increasing since iteration $t-2I$ to $t-I$, we treat this as a warning sign of increased LIS and thus anneal the transfer distributions. As we sharpen δ by annealing, knowledge transfer between languages becomes more specific.

4.3 Modularity of Models

Multilingual topic models can include the different types of information we described in Section 2: document links, vocabulary links, or both (Hu et al., 2014), while a model with neither is equivalent to LDA. Document links can be either hard or soft, or a mix of both, as the only distinction is whether the transfer distribution is an indicator vector. A complete model with both soft document links and vocabulary links is shown in Figure 3. In Section 5, we experiment with a combination of SOFTLINK and VOCLINK.

5 Experiments

5.1 Data

We use five corpora in five languages from different language groups: Arabic (AR, Semitic), Spanish (ES, Romance), Farsi (FA, Indo-Iranian), Russian (RU, Slavic), and Chinese (ZH, Sinitic). Each language is paired with English (EN, Germanic), and we train multilingual topic models on these language pairs individually. All the corpora listed below are available at `http://opus.nlpl.eu/`. For preprocessing, we use stemmers to lemmatize and segment Chinese documents, and then remove stop words and the most frequent 100 word types for each language. Refer to the appendix for additional details.

Training corpora. As in many multilingual studies (Ruder et al., 2017), we use Wikipedia as our training corpus for multilingual topic models, and create two corpora, WIKI-INCO and WIKI-PACO for each language pair (EN, ℓ). For WIKI-INCO, we randomly select $2,000$ documents in each language without any connections, so that no documents are translations of each other (an *incomparable* corpus). We also create a *partially comparable* corpus, WIKI-PACO, which contains around 30% comparable document pairs for each language pair.

Test corpora. We create two test corpora for each language pair (EN, ℓ) from TED Talks 2013 (TED) and Global Voices (GV), which provide categories for each document that can be used as classification labels. The first one, TED+TED, contains documents from TED in both languages, while the second one, TED+GV contains English documents from TED and non-English documents from GV. After training a topic model, we use $\phi^{(\text{EN})}$ and $\phi^{(\ell)}$ to infer topics from both languages. For TED+TED, we choose the five most frequent labels in TED as the label set (*technology*, *culture*, *science*, *global issues*, and *design*); for TED+GV, we replace *global issues* and *design* with *business* and *politics*, since the label set from GV does not include *global issues* and *design*.

Dictionary. We use Wiktionary to extract word translations for VOCLINK and to calculate transfer distribution values δ for SOFTLINK. The dictionary is available at `https://dumps.wikimedia.org/enwiktionary/`.

5.2 Inference Settings

For each compared model, we set the number of topics $K = 25$. We run the Gibbs samplers for $1,000$ training iterations and 500 iterations to infer topic distributions on test corpora. We set Dirichlet priors $\alpha = 0.1$ and $\beta = 0.01$ for HARDLINK and SOFTLINK. For VOCLINK, we set $\beta_r = 0.01$ for priors from root to internal nodes, and $\beta_i = 100$ from internal nodes i to leaves, following Hu et al. (2014).

5.3 Evaluation Metrics

We evaluate each model in two ways. Experimental results below are averaged across all language pairs.

5.3.1 Intrinsic Evaluation: Multilingual Topic Coherence

Typical topic model evaluations include intrinsic and extrinsic measurements. Intrinsic evaluation focuses on topic quality or coherence of the trained topics. The most widely-used metric for measuring monolingual topic coherence is normalized pointwise mutual information (Lau et al., 2014; Newman et al., 2010). Hao et al. (2018) proposed crosslingual normalized pointwise mutual information (CNPMI) by extending this idea to multilingual settings, which correlates well with bilingual speakers' judgments on topic quality.

Given a bilingual topic k in languages ℓ_1 and ℓ_2, and a parallel reference corpus $\mathcal{R}^{(\ell_1,\ell_2)}$, the CNPMI of topic k is calculated as:

$$\text{CNPMI}(\ell_1, \ell_2, k) \quad = \quad \frac{1}{C^2} \sum_{i,j}^{C} \frac{1}{\log \Pr\left(w_i^{(\ell_1)}, w_j^{(\ell_2)}\right)} \cdot \log \frac{\Pr\left(w_i^{(\ell_1)}, w_j^{(\ell_2)}\right)}{\Pr\left(w_i^{(\ell_1)}\right) \Pr\left(w_j^{(\ell_2)}\right)} \tag{6}$$

where C is the cardinality of a topic, *i.e.,* the C most probable words in the topic-word distribution $\phi_k^{(\ell)}$. The co-occurrence probability of two words, $\Pr\left(w_i^{(\ell_1)}, w_j^{(\ell_2)}\right)$, is defined as the proportion of document pairs where both words appear. In the results below, we set $C = 20$, and average the CNPMI scores over $K = 25$ topics for each model output.

To calculate CNPMI scores, we use $10,000$ document pairs from a held-out portion of Wikipedia. CNPMI is an intrinsic evaluation, so it is only available for the training sets, WIKI-PACO and WIKI-INCO.

5.3.2 Extrinsic Evaluation: Crosslingual Classification

A successful multilingual topic model should provide informative features for crosslingual tasks. To show that our model is beneficial to downstream applications, we use crosslingual document classification to evaluate topic model performance. A high classification accuracy when testing on a different language from training indicates topic consistency across languages (Hermann and Blunsom, 2014; Klementiev et al., 2012; Smet et al., 2011).

As in other studies on multilingual topic models, we first train topic models on a bilingual corpus $D^{(\ell_1,\ell_2)}$, and then use topic-word distributions $\phi^{(\ell_1)}$ and $\phi^{(\ell_2)}$ to infer document-topic distributions on unseen documents $D'^{(\ell_1)}$ and $D'^{(\ell_2)}$. Thus, a classifier is trained on $\theta_{d_{\ell 1}}$ with corresponding labels where $d_{\ell 1} \in D'^{(\ell_1)}$, and tested on $\theta_{d_{\ell 2}}$ where $d_{\ell 2} \in D'^{(\ell_2)}$, and vice versa. In our experiments, we use WIKI-PACO and WIKI-INCO to train topic models first, and then perform inference on either TED+TED (both English and non-English documents from TED) or TED+GV (English documents from TED and non-English from GV). For each language pair, we train multi-label classifiers using support vector machines (SVM) with five-fold cross-validation on documents in one language and test on the other. The F-1 scores reported below are micro-averaged over all labels.

5.4 Baseline Comparison

We first compare SOFTLINK with other models: HARDLINK, which is expected to do well on the partially comparable corpus (WIKI-PACO) but poorly on the incomparable corpus (WIKI-INCO), and VOCLINK. We additionally combine SOFTLINK+VOCLINK.

Figure 4 shows the performance (both intrinsic and extrinsic) of all models. For the SOFTLINK models, we used the optimal hyperparameter settings, but we compare other settings in Section 5.5.

When the training corpus is partially comparable (WIKI-PACO), all models can learn comparably coherent topics based on CNPMI scores, though the CNPMI of HARDLINK is lower than all other models. When the data is completely incomparable (WIKI-INCO), HARDLINK loses all connections between languages, so as expected its topics are least coherent. Similarly, when measuring classification performance, HARDLINK is comparable to VOCLINK on WIKI-PACO, but much worse on WIKI-INCO, where it loses all information. When the test set contains mostly parallel documents (TED+TED), the F-1 scores are higher, but when the test domain changes across languages (TED+GV), the performance drops.

On the other hand, SOFTLINK consistently outperforms other models regardless of training and test sets. It seems that SOFTLINK benefits from learning new connections between documents, even when

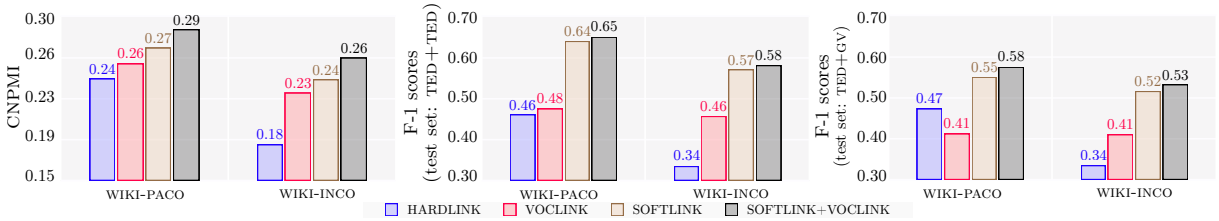
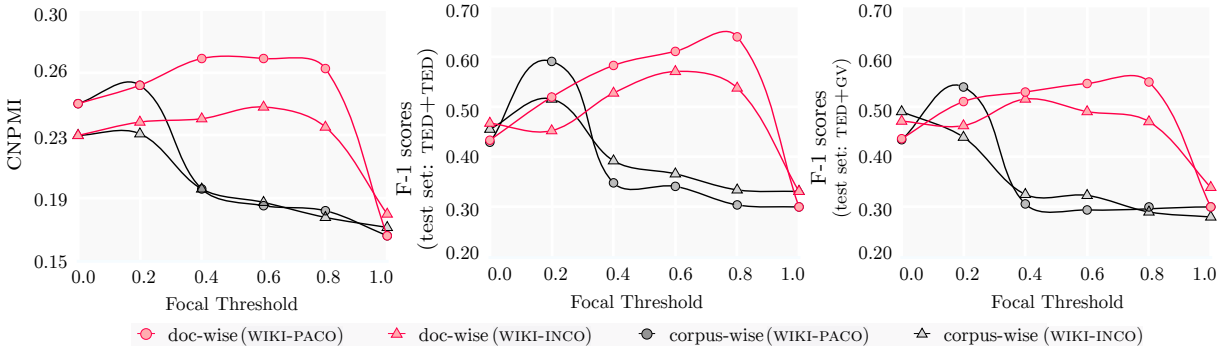
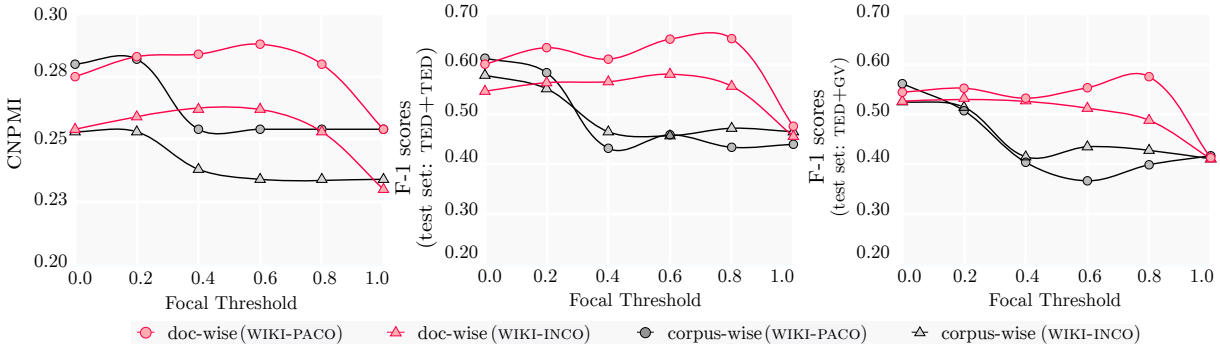


Figure 4: SOFTLINK consistently outperforms other models on both topic quality evaluation (CNPMI) and classification performance (F-1).



(a) Training SOFTLINK model.



(b) Training SOFTLINK with VOCLINK.

Figure 5: CNPMI scores and F-1 scores of crosslingual classification under different focal thresholds and selection scope of the transfer distribution for SOFTLINK and SOFTLINK+VOCLINK (Section 4.2).

part of the corpus contains direct links for training HARDLINK. It is also interesting that SOFTLINK uses the same dictionary resource as VOCLINK, but has a relative performance increase around 25%. It seems SOFTLINK can more efficiently utilize lexical information in a dictionary. We explore this relationship more in Section 5.6.

Finally, we observe that combining SOFTLINK+VOCLINK provides a performance boost over SOFTLINK in all cases, though the increase is small.

5.5 Comparison of Focusing Methods

We have shown that, when optimized, SOFTLINK can better utilize dictionary resources and outperform other models. We now focus on different training configurations for SOFTLINK, specifically, different methods of focusing the transfer distribution (Section 4.2).

Figure 5 shows how F-1 and CNPMI scores change with different static focusing methods. We vary the focal threshold and selection scope (*i.e.,* doc-wise or corpus-wise) for transfer distributions. As we increase the focal threshold π, more documents are zeroed out in the transfer distributions. When $\pi = 0.6$ or 0.8, the transfer distributions are very sparse, and we notice that document-wise selection

	F-1 scores (TED+TED)		F-1 scores (TED+GV)		CNPMI	
	LIS	Fixed	LIS	Fixed	LIS	Fixed
WIKI-PACO	0.627	0.638	0.551	0.534	0.256	0.258
WIKI-INCO	0.551	0.526	0.475	0.470	0.220	0.217

(a) Training SOFTLINK model.

	F-1 scores (TED+TED)		F-1 scores (TED+GV)		CNPMI	
	LIS	Fixed	LIS	Fixed	LIS	Fixed
WIKI-PACO	0.640	0.647	0.557	0.543	0.261	0.266
WIKI-INCO	0.546	0.517	0.459	0.465	0.242	0.233

(b) Training SOFTLINK with VOCLINK.

Table 1: Dynamically focusing transfer distributions in SOFTLINK yields competitive results on classification and topic quality evaluation. There is no significant difference between Fixed and LIS schedules.

achieves the best performance. In the extreme case that $\pi = 1$, the transfer distributions are all zero, so SOFTLINK loses its connections between ℓ_1 and ℓ_2, and thus degrades to monolingual LDA. When training SOFTLINK with VOCLINK, the change of CNPMI and F-1 scores are less obvious as we increase focal threshold, since increasing focal threshold only has an impact on the SOFTLINK component of the model. When the focal threshold is higher, fewer soft links are active, so the model is closer to a plain VOCLINK model.

Interestingly, when focal threshold π changes from 0.2 to 0.4, F-1 scores of corpus-wise selection scope trained on SOFTLINK drops drastically, in contrast to document-wise. This is because using corpus-wise selection could set a large portion of transfer distributions to zero, and only a small number of documents have non-zero transfer distributions. Since corpus-wise selection relies on the entire training corpus, it must be used with caution.

We find that using annealing to dynamically focus the distributions works well and is competitive with static focusing (Table 1). Annealing does better than the majority of settings of static focusing, though is worse than optimally-tuned focusing. We do not observe a significant difference between the two annealing schedules. When combining SOFTLINK and VOCLINK, the patterns are similar to that of SOFTLINK only.

5.6 Sensitivity to Dictionary Size

Both VOCLINK and SOFTLINK use the same dictionary resource, yet SOFTLINK produces better features for downstream tasks. To understand this behavior better, we experiment with different dictionary sizes to understand how well the models are utilizing the resource.

In Figure 6, we use different proportions (20%, 40%, ..., 80%) of the dictionary to train SOFTLINK and VOCLINK.[4] We observe that the performance of VOCLINK (both F-1 and CNPMI) increases almost linearly with the dictionary size. In contrast, SOFTLINK is already at its best performance with only 20% of the available dictionary entries. This is further confirmation that SOFTLINK is using this resource in a more efficient way.

In VOCLINK, knowledge transfer happens through internal nodes of the word distribution priors, *i.e.,* word translations pairs, and words without translations are directly connected to the Dirichlet tree's root. If the dictionary cannot cover all the word types appeared in the training set, VOCLINK will have a set of word types in ℓ_1 that cannot transfer enough topic knowledge to ℓ_2 and vice versa. The fewer entries the dictionary provides, the more VOCLINK degrades to monolingual LDA. In contrast, SOFTLINK can potentially transfer knowledge from the whole corpus. For SOFTLINK, the dictionary is not used directly for modeling, rather it is only used for linking documents. Thus, knowledge transfer does not heavily rely on the number of entries in the dictionary.

[4]We use document-wise selection scope and focal threshold $\pi = 0.6$ for training SOFTLINK; same as in Section 5.4.

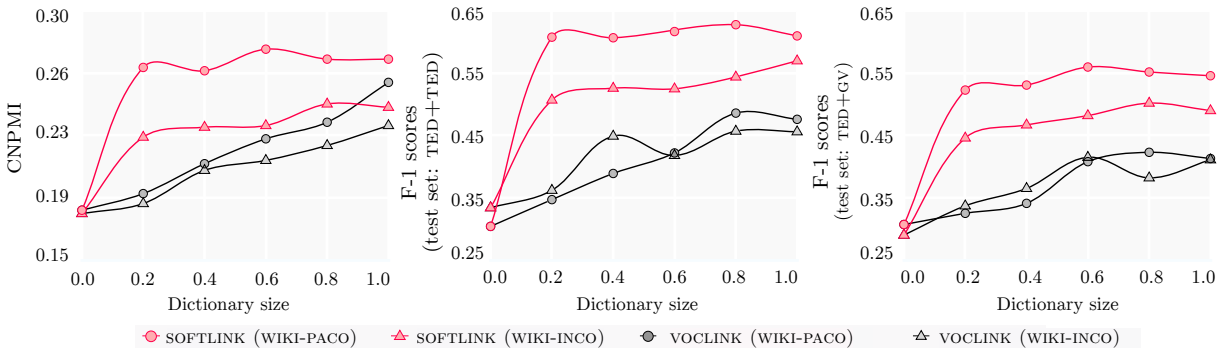


Figure 6: Performance of VOCLINK continues increasing when more dictionary entries are added, while SOFTLINK performance mostly stabilizes after using only 20% of available dictionary entries.

5.7 Discussion

The sensitivity to dictionary size is an important factor to be considered in practice. For low-resource languages, a dictionary is easier to obtain than a large parallel corpus (Section 1). Models that rely on dictionaries such as VOCLINK and SOFTLINK are therefore more applicable to low-resource languages than HARDLINK. However, there are also large variations in dictionary size among languages. For example, in Wiktionary, 57 languages have fewer than 1,000 entries, while 77 languages have more than 100,000 entries. For truly low-resource languages, dictionary size could be a limiting factor. Since SOFTLINK can outperform VOCLINK with only a limited amount of lexical information, it may be able to transfer knowledge to low-resource languages more effectively than other approaches.

In summary, SOFTLINK relaxes and generalizes HARDLINK to be adaptable to more situations, while using dictionary information more efficiently than VOCLINK.

6 Conclusions and Future Work

We have described a new formulation for multilingual topic models which explicitly shows the knowledge transfer process across languages. Based on this analysis, we proposed a new multilingual topic model that can learn multilingually coherent topics and provide consistent topic features for crosslingual tasks. Unlike existing models, our approach is flexible and adaptable to incomparable corpora with only a dictionary, which is beneficial in many situations, in particular low-resource settings.

There are many possible directions following this work. First, our formulation of the knowledge transfer process enables future work focusing on how to develop more efficient algorithms that transfer knowledge with minimal supervision. Second, for SOFTLINK we plan to explore more about characteristics of languages that can lead to better formulations and learning of the transfer distributions.

References

Željko Agić, Dirk Hovy, and Anders Søgaard. 2015. If All You Have is a bit of the Bible: Learning POS Taggers for Truly Low-Resource Languages. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 268–272.

Željko Agić, Anders Johannsen, Barbara Plank, Héctor Martínez Alonso, Natalie Schluter, and Anders Søgaard. 2016. Multilingual Projection for Parsing Truly Low-Resource Languages. *Transactions of the Association for Computational Linguistics*, 4:301–312.

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating Domain Knowledge into Topic Modeling via Dirichlet Forest Priors. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, pages 25–32.

Timothy Baldwin, Jonathan Pool, and Susan M. Colowick. 2010. PanLex and LEXTRACT: Translating All Words of All Languages of the World. In *COLING 2010, 23rd International Conference on Computational Linguistics, Demonstrations Volume, 23-27 August 2010, Beijing, China*, pages 37–40.

Maria Barrett, Frank Keller, and Anders Søgaard. 2016. Cross-lingual Transfer of Correlations between Parts of Speech and Gaze Features. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 1330–1339.

Julian Besag. 1975. Statistical Analysis of Non-lattice Data. *The statistician*, pages 179–195.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Jordan L. Boyd-Graber and David M. Blei. 2009. Multilingual Topic Models for Unaligned Text. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, pages 75–82.

Xilun Chen, Ben Athiwaratkun, Yu Sun, Kilian Q. Weinberger, and Claire Cardie. 2016. Adversarial Deep Averaging Networks for Cross-Lingual Sentiment Classification. *CoRR*, abs/1606.01614.

Samuel Y. Dennis III. 1991. On the Hyper-Dirichlet Type 1 and Hyper-Liouville Distributions. *Communications in Statistics — Theory and Methods*, 20(12):4069–4081.

E. Dario Gutiérrez, Ekaterina Shutova, Patricia Lichtenstein, Gerard de Melo, and Luca Gilardi. 2016. Detecting Cross-cultural Differences Using a Multilingual Topic Model. *Transactions of the Association for Computational Linguistics*, 4:47–60.

Shudong Hao, Jordan L. Boyd-Graber, and Michael J. Paul. 2018. Lessons from the Bible on Modern Topics: Adapting Topic Model Evaluation to Multilingual and Low-Resource Settings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1090–1100.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual Models for Compositional Distributed Semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 58–68.

Geert Heyman, Ivan Vulić, and Marie-Francine Moens. 2016. C-BiLDA: Extracting Cross-lingual Topics from Non-Parallel Texts by Distinguishing Shared from Unshared Content. *Data Mining and Knowledge Discovery*, 30(5):1299–1323.

Gerold Hintz and Chris Biemann. 2016. Language Transfer Learning for Supervised Lexical Substitution. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, pages 118–129.

Yuening Hu, Ke Zhai, Vladimir Eidelman, and Jordan L. Boyd-Graber. 2014. Polylingual Tree-Based Topic Models for Translation Domain Adaptation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1166–1176.

Jagadeesh Jagarlamudi and Hal Daumé III. 2010. Extracting Multilingual Topics from Unaligned Comparable Corpora. In *Advances in Information Retrieval, 32nd European Conference on IR Research, ECIR 2010, Milton Keynes, UK, March 28-31, 2010. Proceedings*, pages 444–456.

David Kamholz, Jonathan Pool, and Susan M. Colowick. 2014. PanLex: Building a Resource for Panlingual Lexical Translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014.*, pages 3145–3150.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing Crosslingual Distributed Representations of Words. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*, pages 1459–1474.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. *MT Summit*.

Kriste Krstovski and David A. Smith. 2016. Bootstrapping Translation Detection and Sentence Extraction from Comparable Corpora. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1127–1132.

Kriste Krstovski, David A. Smith, and Michael J. Kurtz. 2016. Online Multilingual Topic Models with Multi-Level Hyperpriors. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 454–459.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine Reading Tea Leaves: Automatically Evaluating Topic Coherence and Topic Model Quality. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*, pages 530–539.

Janne Leppä-aho, Johan Pensar, Teemu Roos, and Jukka Corander. 2017. Learning Gaussian Graphical Models with Fractional Marginal Pseudo-Likelihood. *International Journal of Approximate Reasoning*, 83:21–42.

Xiaodong Liu, Kevin Duh, and Yuji Matsumoto. 2015. Multilingual Topic Models for Bilingual Dictionary Extraction. *ACM Transactions on Asian & Low-Resource Langanguage Information Processing.*, 14(3):11:1–11:22.

Tengfei Ma and Tetsuya Nasukawa. 2017. Inverted Bilingual Topic Models for Lexicon Extraction from Non-parallel Data. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4075–4081.

David M. Mimno, Hanna M. Wallach, Jason Naradowsky, David A. Smith, and Andrew McCallum. 2009. Polylingual Topic Models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, 6-7 August 2009, Singapore, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 880–889.

Thomas Minka. 1999. The Dirichlet-tree Distribution.

Maria Moritz and Marco Büchler. 2017. Ambiguity in Semantically Related Word Substitutions: an Investigation in Historical Bible Translations. In *Proceedings of the NoDaLiDa 2017 Workshop on Processing Historical Language, Gothenburg, Sweden, May 22, 2017*, pages 18–23.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic Evaluation of Topic Coherence. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*, pages 100–108.

Xiaochuan Ni, Jian-Tao Sun, Jian Hu, and Zheng Chen. 2009. Mining Multilingual Topics from Wikipedia. In *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, pages 1155–1156.

Michael J. Paul and Mark Dredze. 2015. SPRITE: Generalizing Topic Models with Structured Priors. *Transactions of the Association for Computational Linguistics*, 3:43–57.

John C. Platt, Kristina Toutanova, and Wen-tau Yih. 2010. Translingual Document Representations from Discriminative Projections. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11 October 2010, MIT Stata Center, Massachusetts, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 251–261.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2017. A Survey of Cross-lingual Word Embedding Models. *CoRR*, abs/1706.04902.

Ekaterina Shutova, Lin Sun, E. Dario Gutiérrez, Patricia Lichtenstein, and Srini Narayanan. 2017. Multilingual Metaphor Processing: Experiments with Semi-Supervised and Unsupervised Learning. *Computational Linguistics*, 43(1):71–123.

Wim De Smet and Marie-Francine Moens. 2009. Cross-language Linking of News Stories on the Web Using Interlingual Topic Modelling. In *Proceedings of the 2nd ACM Workshop on Social Web Search and Mining, CIKM-SWSM 2009, Hong Kong, China, November 2, 2009*, pages 57–64.

Wim De Smet, Jie Tang, and Marie-Francine Moens. 2011. Knowledge Transfer across Multilingual Corpora via Latent Topics. In *Advances in Knowledge Discovery and Data Mining - 15th Pacific-Asia Conference, PAKDD 2011, Shenzhen, China, May 24-27, 2011, Proceedings, Part I*, pages 549–560.

Noah A. Smith and Jason Eisner. 2006. Annealing Structural Bias in Multilingual Weighted Grammar Induction. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*.

Naonori Ueda and Ryohei Nakano. 1994. Deterministic Annealing Variant of the EM Algorithm. In *Advances in Neural Information Processing Systems 7, [NIPS Conference, Denver, Colorado, USA, 1994]*, pages 545–552.

Ivan Vulić, Wim De Smet, and Marie-Francine Moens. 2013. Cross-language Information Retrieval Models Based on Latent Topic Models Trained with Document-aligned Comparable Corpora. *Information Retrieval*, 16(3):331–368.

Appendix A Pseudolikelihood

Theorem 1. *The conditional generative model with document links yields the same posterior estimator to the joint generative model using collapsed Gibbs sampling.*

Proof. Suppose the document links model is sampling topic of the m-th token in document d_{ℓ_2}. The sampler calculates the conditional topic distribution, and then draw a topic assignment. Using collapsed Gibbs sampling, we calculate the conditional probability of a topic k:

$$\begin{aligned}
&\Pr\left(z_{d_{\ell 2},m}=k|\mathbf{z}_{d_{\ell 2},-},\mathbf{w}_{d_{\ell 2}};\mathbf{n}_{d_{\ell 1}},\alpha,\beta\right)=\frac{\Pr\left(z_{d_{\ell 2},m}=k,\mathbf{z}_{d_{\ell 2},-},\mathbf{w}_{d_{\ell 2}};\mathbf{n}_{d_{\ell 1}},\alpha,\beta\right)}{\Pr\left(\mathbf{z}_{d_{\ell 2},-},\mathbf{w}_{d_{\ell 2}};\mathbf{n}_{d_{\ell 1}},\alpha,\beta\right)}\\
=&\frac{\Pr\left(z_{d_{\ell 2},m}=k,\mathbf{z}_{d_{\ell 2},-},\mathbf{w}_{d_{\ell 2}};\mathbf{n}_{d_{\ell 1}},\alpha\right)}{\Pr\left(\mathbf{z}_{d_{\ell 2},-};\mathbf{n}_{d_{\ell 1}},\alpha\right)}\cdot\frac{\Pr\left(\mathbf{w}_{d_{\ell 2}}|z_{d_{\ell 2},n}=k,\mathbf{z}_{d_{\ell 2},-};\beta\right)}{\Pr\left(\mathbf{w}_{d_{\ell 2}}|\mathbf{z}_{d_{\ell 2},-},\beta\right)}\\
=&\frac{\frac{\prod_{k'\neq k}\Gamma\left(n_{k'|d_{\ell 2}}+\mathbf{n}_{d_{\ell 1},k'}+\alpha\right)\cdot\Gamma\left(n_{k|d_{\ell 2}}+\mathbf{n}_{d_{\ell 1},k}+\alpha+1\right)}{\Gamma\left(n_{\cdot|d}+\mathbf{n}_{d_{\ell 1}}+\alpha+1\right)}}{\frac{\prod_{k}\Gamma\left(n_{k|d_{\ell 2}}+\mathbf{n}_{d_{\ell 1},k}+\alpha\right)}{\Gamma\left(n_{\cdot|d_{\ell 2}}+\mathbf{n}_{d_{\ell 1}}+K\alpha\right)}}\cdot\frac{\frac{\prod_{w\neq w_{d_{\ell 2},m}}\Gamma\left(n_{w|k}+\beta\right)\cdot\Gamma\left(n_{w_{d_{\ell 2},m}|k}+\beta+1\right)}{\Gamma\left(n_{\cdot|k}+V^{(\ell_2)}\beta+1\right)}}{\frac{\prod_{w}\Gamma\left(n_{w|k}+\beta\right)}{\Gamma\left(n_{\cdot|k}+V^{(\ell_2)}\beta\right)}}\\
=&\frac{\Gamma\left(n_{k|d_{\ell 2}}+\mathbf{n}_{d_{\ell 1},k}+\alpha+1\right)}{\Gamma\left(n_{k|d_{\ell 2}}+\mathbf{n}_{d_{\ell 1},k}+\alpha\right)}\cdot\frac{\Gamma\left(n_{\cdot|d_{\ell 2}}+\mathbf{n}_{d_{\ell 1}}+K\alpha\right)}{\Gamma\left(n_{\cdot|d_{\ell 2}}+\mathbf{n}_{d_{\ell 1}}+\alpha+1\right)}\cdot\frac{\Gamma\left(n_{w_{d_{\ell 2},m}|k}+\beta\right)}{\Gamma\left(n_{w_{d_{\ell 2},m}|k}+\beta+1\right)}\cdot\frac{\Gamma\left(n_{\cdot|k}+V^{(\ell_2)}\beta\right)}{\Gamma\left(n_{\cdot|k}+V^{(\ell_2)}\beta+1\right)}\\
=&\frac{n_{k|d_{\ell 2}}+\mathbf{n}_{d_{\ell 1},k}+\alpha}{n_{\cdot|d_{\ell 2}}+\mathbf{n}_{d_{\ell 1}}+K\alpha}\cdot\frac{n_{w_{d_{\ell 2},m}|k}+\beta}{n_{\cdot|k}+V^{(\ell_2)}\beta},
\end{aligned}$$

where $\mathbf{z}_{d_{\ell 2},-}$ is all the topic assignments in d_{ℓ_2} except the current one, $n_{\cdot|d_{\ell 2}}$ the number of tokens in d_{ℓ_2}, $n_{k|d_{\ell 2}}$ the number of tokens assigned to topic k in d_{ℓ_2}, $n_{\cdot|k}$ the number of tokens assigned to topic k, $n_{w|k}$ the number of word type w assigned to topic k, and $V^{(\ell_2)}$ the vocabulary size of language ℓ_2. The roles of ℓ_1 and ℓ_2 are interchangeable, so both languages use the same conditional distributions. The last equation of the derivation above gives identical posterior estimation in the original model. Thus, the alternative formulation, despite not a numerically accurate likelihood approximation, does not make a difference for parameter estimation. □

Appendix B Dataset Processing Details

B.1 Pre-Processing

For all the languages, we use existing stemmers to stem words in the corpora and the entries in Wiktionary. Since Chinese does not have stemmers, we loosely use "stem" to refer to "segment" Chinese sentences into words. We also use fixed stopword lists to filter out stop words. Table 2 lists the source of the stemmers and stopwords.

B.2 Data Source

We list the statistics in Table 3.

[1] `http://snowball.tartarus.org`;
[2] `http://arabicstemmer.com`;
[3] `https://github.com/6/stopwords-json`;
[4] `https://github.com/sobhe/hazm`;
[5] `https://github.com/fxsjy/jieba`.

Language	Family	Stemmer	Stopwords
EN	Germanic	`SnowBallStemmer` [1]	NLTK
ES	Romance	`SnowBallStemmer`	NLTK
RU	Slavic	`SnowBallStemmer`	NLTK
AR	Semitic	`Assem's Arabic Light Stemmer` [2]	GitHub [3]
FA	Indo-Iranian	`Hazm` [4]	GitHub
ZH	Sinitic	`Jieba` [5]	GitHub

Table 2: List of source of stemmers and stopwords used in experiments.

		WIKI-PACO	WIKI-INCO	TED	GV	Wikipedia (for CNPMI)	Wiktionary
AR	#docs	2,000	2,000	1,112	2,000	8,862	16,127
	#tokens	1,075,691	293,640	1,521,334	466,859	79,740	
	#types	32,843	19,900	44,982	32,468	1,533,261	
ES	#docs	2,000	2,000	1,152	2,000	9,325	31,563
	#tokens	475,234	237,561	1,228,469	493,327	1,763,897	
	#types	35,069	27,465	30,247	28,471	91,428	
FA	#docs	2,000	2,000	687	401	9,669	14,952
	#tokens	415,620	91,623	1,415,263	89,414	940,672	
	#types	18,316	9,987	36,670	9,447	46,995	
RU	#docs	2,000	2,000	1,010	2,000	9,837	33,574
	#tokens	4,368,563	766,887	1,133,098	679,217	2,356,994	
	#types	51,740	24,341	44,577	47,395	134,424	
ZH	#docs	2,000	2,000	1,123	2,000	8,222	23,276
	#tokens	3,095,977	303,634	1,428,532	745,307	1,338,116	
	#types	59,431	30,481	71,906	69,872	144,765	

Table 3: Statistics of corpora and dictionary in the five languages used in the experiments.

Wikipedia (WIKI-PACO, and WIKI-INCO). For training multilingual topic models, the dataset Wikipedia can be downloaded at `http://opus.nlpl.eu/TED2013.php`. For each language pair (EN, ℓ), we create WIKI-INCO, a completely incomparable corpus, where $2,000$ EN documents and $2,000$ non-English documents are randomly chosen but do not contain document-level translations to each other.

We also create WIKI-PACO, a partially comparable corpus. Each language has different proportions of comparable document pairs. See Table 4.

TED Talks 2013 (TED). TED Talks 2013 contains mostly parallel documents, and can be obtained from OPUS: `http://opus.nlpl.eu/TED2013.php`. Note that not all English documents have translations to another language, which is slightly different from the original assumptions in polylingual topic models.

The classification labels can be obtained from the documents. Each document has several "categories" that can be regarded as labels. Thus, we retrieve those labels, and choose the most frequent five labels for classification: *technology*, *culture*, *science*, *global issues*, and *design*.

Global Voices (GV). Global Voices can be obtained from OPUS as well: `http://opus.nlpl.eu/GlobalVoices.php`. Global Voices corpus has a large number of documents, so for efficiency, we randomly choose a sample of at most 2,000 documents for each language.

There's no label information from the corpus itself. However, the labels can be retrieved from the webpage of each document, at `https://globalvoices.org`. To make sure Global Voices have

Languages	AR	ES	FA	RU	ZH
Proportion	12.2%	9.35 %	50.85 %	50.20 %	17.90 %

Table 4: Proportions of linked document pairs in corpus WIKI-PACO.

the same label set to TED Talks, we changed the label set to: *technology*, *culture*, *science*, *business*, and *politics*.

Wiktionary. We use English Wiktionary to create bilingual dictionaries, which can be downloaded at `https://dumps.wikimedia.org/enwiktionary/`.

Using Word Embeddings for Unsupervised Acronym Disambiguation

Jean Charbonnier
Hochschule Hannover
Expo Plaza 12
D-30539 Hannover
jean.charbonnier
@hs-hannover.de

Christian Wartena
Hochschule Hannover
Expo Plaza 12
D-30539 Hannover
christian.wartena
@hs-hannover.de

Abstract

Scientific papers from all disciplines contain many abbreviations and acronyms. In many cases these acronyms are ambiguous. We present a method to choose the contextual correct definition of an acronym that does not require training for each acronym and thus can be applied to a large number of different acronyms with only few instances. We constructed a set of 19,954 examples of 4,365 ambiguous acronyms from image captions in scientific papers along with their contextually correct definition from different domains. We learn word embeddings for all words in the corpus and compare the averaged context vector of the words in the expansion of an acronym with the weighted average vector of the words in the context of the acronym. We show that this method clearly outperforms (classical) cosine similarity. Furthermore, we show that word embeddings learned from a 1 billion word corpus of scientific texts outperform word embeddings learned from much larger general corpora.

1 Motivation

Scientific papers usually contain a high number abbreviations and acronyms that might be very general or very specific for a certain domain and that might be very common in that domain or that are constructed ad hoc. For various tasks, like retrieval or information extraction we need the expanded form of the acronyms.[1]

In most cases we can find the definition for an acronym in the paper. Finding this definition automatically is not as trivial as it might sound and there is a lot of literature on this topic. However, a lot of cases remain where we cannot find such a definition, because the extraction method fails or because there is no definition in the paper. In these cases we can use a definition found elsewhere. As already noted by Chang et al. (2002) in these cases acronyms often turn out to be ambiguous and the correct definition has to be chosen by considering the context of the acronym. Exactly this task is the topic of the present paper. Thus, although the goal of our work is acronym expansion, the work is more related to word sense disambiguation (WSD) than to typical work on acronym resolution. The main difference with WSD is that we do not have dictionaries with description of possible senses. Instead the possible expansions of the acronym directly represent the different senses.

In the Project NOA (Reuse of Open Access Images, http://noa.wp-hs-hannover.de) we analyze captions from images in scientific papers to collect appropriate metadata for those images (Charbonnier et al., 2018). In order to get maximal information from the usually short image captions we need to expand the acronyms. Moreover, the acronyms are often very specific technical terms that give a lot of information on the displayed image. Since the focus of the project is on image captions, in the present paper we consider only acronyms from image captions, but this is not relevant to our method that can be used to disambiguate any acronym in a scientific paper. Since the image captions in some cases are very short, we also use the text from the sentences referring to the image to disambiguate the acronyms in the caption.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2610–2619
Santa Fe, New Mexico, USA, August 20-26, 2018.

2 Related Work

Typically definitions for acronyms are found using a number of patterns to find candidates in the context of the acronyms and a number of rules to select the most likely candidate. Patterns and rules are defined manually or learned in a supervised (Nadeau and Turney, 2005) or unsupervised (Kirchhoff and Turner, 2016) way. The work of Schwartz and Hearst (2011) focuses on finding and extracting abbreviations from medical publications by identifying <"short form", "long form"> pairs and was used in a majority of the work discussed here. They argue that many abbreviations in the biomedical domain follow a predictable pattern where the first letter of each word is a letter of its short form, but that there are cases where this is not true. Therefore they describe a simple and fast algorithm with a high recall and precision to find those pairs using string operations to verify that the acronym letters appear in the right order in the long form.

Much research on acronym resolution and disambiguation was done in the biomedical domain (Okazaki and Ananiadou, 2006; Finley et al., 2016; Vo et al., 2016; Wu et al., 2017). The most reliable approach for disambiguation is to train a classifier that chooses the right definition based on features representing the context of the acronym (Wu et al., 2015; Wang et al., 2016; Finley et al., 2016; Antunes and Matos, 2017). Typical features are the word frequencies but also morphemes and position features. Wu et al. (2015) and Antunes and Matos (2017) also include features from word embeddings. Kirchhoff and Turner (2016) learn the similarity between a possible expansion and the context in a neural network using word embedding features. Wu et al. (2015) use the sum of all pmi (pointwise mutual information) values between a definition and a context as a measure of their similarity. Thus they also use a more advanced similarity measure than just word overlap based on word co-occurrence in a large corpus.

Stevenson et al. (2009) created a biomedical evaluation corpus consisting of overall 55,655 documents using MEDLINE and MEDSTRACT. They counted the occurrence of 21 three letter abbreviations and of their expansions in the abstracts of those documents. The occurrence of these hand selected abbreviations range from $71 - 14,871$ times in the corpus. Jimeno-Yepes et al. (2011) created the widely used MSH-WSD dataset consisting of 203 ambiguous entities, where 106 are abbreviations. Each of those has a maximum of 100 instances from MEDLINE where they occur.

Disambiguation of acronyms is a special case of the more general problem of word sense disambiguation (Navigli, 2009; Yarowsky, 2010; Mihalcea, 2011). Word sense disambiguation is typically done by comparing the possible senses of an ambiguous word, expressed e.g. by their glosses in a dictionary, with the senses of the surrounding words (Lesk, 1986; Patwardhan et al., 2003). The gloss overlap can be improved, when the glosses are extended with related words (Banerjee and Pedersen, 2003). We will follow this idea below, but the extension of the glosses is done implicitly using distributional similarity, as also was done by Aga et al. (2016) and Oele and Noord (2017).

Henry et al. (2017) and Tulkens et al. (2016) specifically worked on disambiguation of acronyms. They used the MSH-WSD dataset and UMLS to disambiguate words/concepts in the domain of biomedicine. Tulkens et al. (2016) combined word representations and definitions from UMLS to create concept representations. Those concepts are then used to disambiguate terms using cosine similarity and the context of the unambiguous term. As features they used word embeddings as described by Mikolov et al. (2013) learned with MEDLINE abstracts and the MIMIC-III corpus. Henry et al. (2017) evaluate different feature extraction methods using 2-MRD (2nd Order Co-occurrence Machine Readable Dictionary) to disambiguate biomedical word senses. Among these were Singular Value Decomposition, Principal Component Analysis and Neural Word Embeddings trained with Bag of Words and Skip Gram. They used extended definitions like parent/children, narrow/broader relations and associated synonymous terms from UMLS to create those vectors. Pakhomov et al. (2016) evaluated the effect of domain specific corpora for disambiguating medical terms while using word embeddings. They showed that biomedical articles can be used to a degree to represent the semantics of clinical reports.

3 Data

Most available data sets with acronyms are quite small, or focus specifically on the biomedical domain, or do not contain ambiguous acronyms, or only ambiguous acronyms with a few very frequent resolutions. To develop a disambiguation method that is suited for our practical problem, we need a set of ambiguous

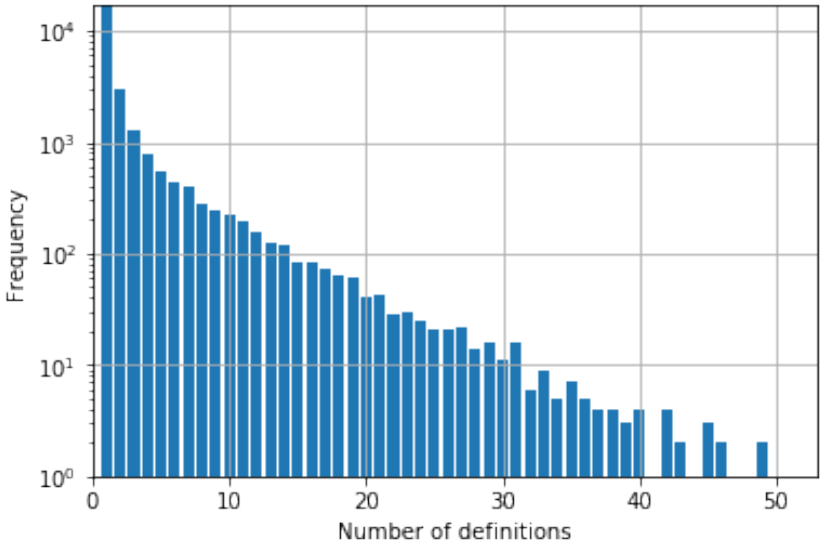


Figure 1: Number of definitions (after clustering) per acronym in the NOA corpus.

acronyms from different disciplines with possibly very infrequent and unusual definitions. In absence of such a test set, we created our own set of ambiguous acronyms.

3.1 Acronyms in the NOA Corpus

Our database contained 712,438 papers from open access journals by the end of 2017. The collection contains journals from almost all scientific disciplines with a focus on technical and biomedical research areas. We refer to Sohmen et al. (2018) for more details.

In this collection we found 2,838,713 occurrences of words written in all capitals of at least 3 letters that thus are likely to be acronyms. For 25,336 acronyms with a total of 628,470 occurrences we found a definition in the paper using a simple regular expression. For 379,509 additional acronyms we could find an unambiguous definition in another paper from the same journal. In total about 36% of all potential acronyms could be expanded. For 67 % of the remaining acronyms we have multiple possible expansions from other papers.

Since authors sometimes seem to be uncertain about the exact definition of acronyms we build clusters of definitions, such that for each pair (a, b) in a cluster either a is a permutation of b (like *annual average daily traffic* and *average annual daily traffic* for AADT) or if the Levenshtein Distance of a and b is smaller than 5 (like *anterior acetabular sector angle* and *Anterior acetabular section angle* for AASA.) Finally, we take the most frequent definition as the correct one. The average cluster size is 1.36. The average number of definitions was reduced from 3.6 to 2.7. Since we used simple heuristics for cluster building, we have a few cases in which two different definitions are put into one cluster and many cases of equivalent definitions that are still treated as different definitions: the overwhelming majority of the clusters only contain spelling variants, different capitalization and typos. The distribution of the number of definitions per acronym for all acronyms in image captions in the NOA corpus is given in Figure 1.

The acronym with the highest number of definitions is SSC with 52 definitions that is used for concepts like *Surgical Safety Checklist*, *symmetric similarity coefficients*, *secondary sclerosing cholangitis* and *Swedish Space Cooperation*.

3.2 Test Set

From our database we collected a set acronyms that have at least two and at most five definitions that are not too similar. As a criterion for similarity we used trigram overlap and we required that the Jaccard-Index of the sets of trigrams was smaller than 0.4. This gave us 4,365 acronyms. In this set we still have 140 acronyms with at least two definitions that only differ in one word, like *DNA Damage Induced Sumoylation*

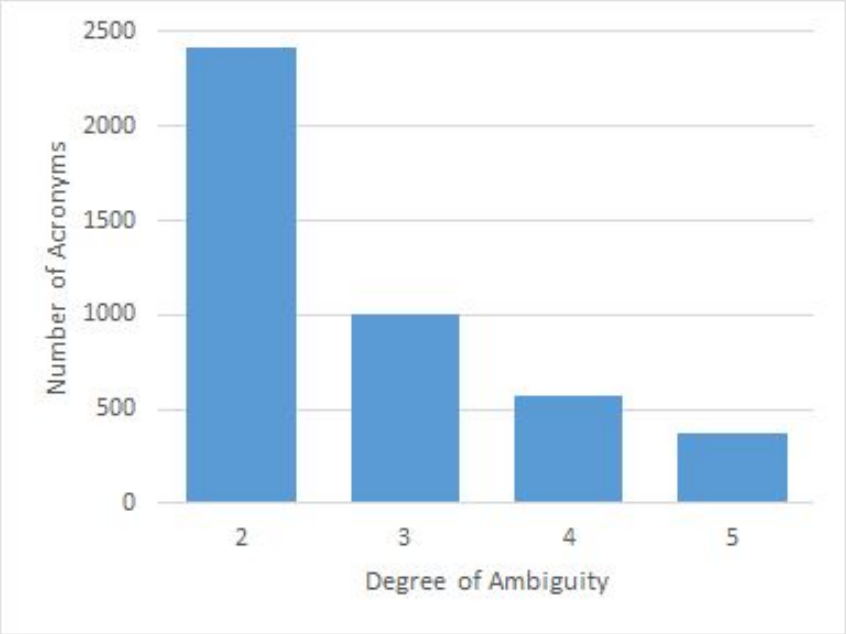


Figure 2: Number of acronyms for each degree of ambiguity in the test set. E.g. we see that there are 1000 acronyms with 3 possible expansions in the test set.

Table 1: Example of the data set. The contextual right definition is typeset in bold.

Acronym	Definitions	DOI of Source	Context/Caption
EBG	Electro-burnt Graphene **Electromagnetic Band Gap**	10.1155/2008/790358	*The EBG leaky wave antenna is fed with a patch array. It must radiate in the direction* $[\Theta = 30^o; \phi = 0^o]$.

and *DNA Damage Induced Senescence* or *very low calorie diet* and *very low carbohydrate diet*. The numbers of acronyms for each degree of ambiguity is given in Figure 2.

For each sense of an acronym we selected at least 1 and at most 5 image captions, in which that acronym was used, from papers in which it was unambiguously defined. This resulted in a set of 19,954 instances of acronyms in a specific context that have to be disambiguated. The data set is provided as supplementary material to this paper. A typical example of a such an instance is given in Table 1. The average length of the captions is 531.3 characters (105.7 tokens), the median is 385 characters (77 tokens). Figure 3 shows the number of expansions with one, two, etc. examples in the test set. Given the low number of examples, training a classifier for each ambiguous acronym is not possible.

3.3 Wordembeddings

We use the full text of all 672,157 papers in our database with a total of 1.662×10^9 tokens to compute word embeddings for all words in the definitions of the acronyms and in the image captions. For training of the word embeddings we only use the body texts and excluded the captions. The 12,002 definitions in our test set consist of 116,603 different tokens. For 92 of those tokens we do not have a word embedding. These are mainly stop words, typos and single letters (as in *Downstream of Kinases (DOK)* and *Vitamin K Antagonist (VKA)*, resp.). In the Google and GloVe dictionaries 821 and 189 tokens, respectively, are not captured. In theses dictionaries in addition many technical terms are missing.

The best results were achieved with a window size of 5 using CBOW, an embedding size of 300. We removed all tokens from our data that are in the NLTK stopword list, that have less than 2 characters or that occur less than 5 times in the corpus.

For comparison we also used two well known pre-trained general models: GoogleNews (Mikolov et al., 2013) and GloVe (Pennington et al., 2014).

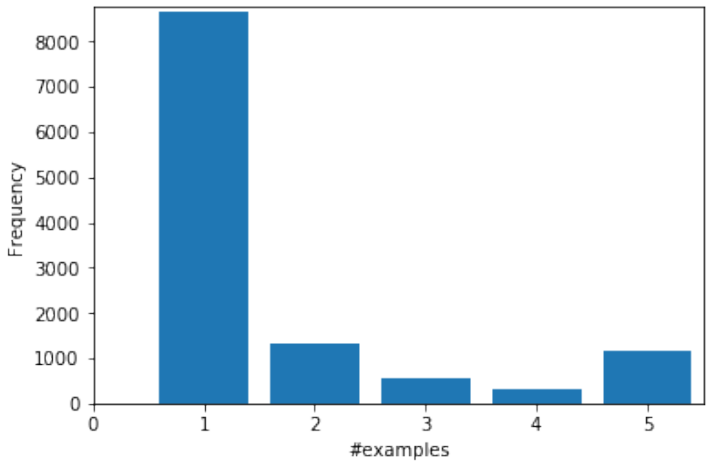


Figure 3: Frequency of each number of examples per definition: For the majority of the possible definitions of an acronym the test set only contains one example; there are about 1200 definitions for which there are 2 examples in the test set, etc.

4 Method

The general idea to decide which sense of an acronym is the contextual correct one, is to compare each definition of the acronym with its context. Now we can vary both the context and the method to compare the definition and context. As the context we take either the image caption from which the acronym was taken, or we expand the caption with sentences referring to the image.

4.1 Baselines

As a baseline we use an unsupervised classifier based on the cosine similarity between tf-idf-vectors of the definition and the context.

As a further baseline we use a majority classifier, that always assigns the definition that was found most frequently in the whole database.

4.2 Prediction

Since the simple cosine similarity used as a baseline captures only literal word overlap and not semantic similarities we also use word embeddings to represent the word in the texts and compare the average word embedding vector for both texts.

To decide which of the given candidates for a solution should be selected, we construct a representing vector for the example text. This vector is the weighted average of all word-embeddings found in the text. We simply discard any token that is not in our dictionary during the average vector calculation. We also calculate an average weighted vector representing the alternative definitions. In both cases we use idf-values as weights for the components. We calculate the cosine similarity and select the candidate with the higher similarity as the right solution.

Formally, let $t = s_0, s_1, \ldots, s_n$ be any sequence of tokens. Now we define the *context vector* of t as

$$cv(t) \quad = \quad \frac{1}{N} \sum_{i=0}^{n} idf(s_i) \cdot cv(s_i) \tag{1}$$

where $cv(s_i)$ is the word embedding or context vector of s_i. Now, for some acronym a let $\mathcal{D} = \{D^0, D^1, \ldots D^n\}$ be the set of definitions for a, with $D^i = d_0^i \ldots d_{k^i}^i$ the sequence of tokens from the definitions for each $0 \leq i \leq n$ and $C = c_0 \ldots c_l$ the tokens in the context. Now the predicted definition $\text{pred}(a, C)$ of a in context C is:

$$\text{pred}(a, C) \quad = \quad \max_{\forall D \in \mathcal{D}} \cos\left(cv(D), cv(C)\right) \tag{2}$$

Table 2: Accuracy of acronym disambiguation

Method	*small Context*	*larger Context*
Majority Classifier	0.44	
Vector Space Model (Cosine)	0.74	0.77
Distr. Sim. – GloVe	0.70	0.71
Distr. Sim. – Google News	0.71	0.73
Distr. Sim. – Own vectors	0.84	0.86

Table 3: Acronyms incorrectly resolved by our method. The contextual right definition is typeset in bold.

Acronym	**Definitions**	**DOI of Source**	**Context/Caption**
DOW	Day of Week **Deep Ocean Water**	10.3390/md15060168	..... *DOW: TAA-induced mice fed with DOW (0.603 mL/kg/day), AS: TAA-induced mice fed with adenosine (0.329 mg/kg/day), CC: TAA-induced mice fed with cordycepin (0.615 mg/kg/day).*
AIHT	**Adaptive Inverse Hyperbolic Tangent** Analysis Iterative Hard Thresholding	10.1155/2014/825169	*AIHT function image enhancement algorithm procedures.*

4.3 Adding more Context

For very short captions we can extend the context of the acronym with text from paragraphs referring to the image. To be precise, if there are less then 15 tokens in the caption, we add the referring sentence to the context and keep adding left and right neighbor sentences until we have 35 tokens.

5 Results

The results are shown in Table 2. We see that all methods using the context of an acronym outperform the majority classifier. In all cases adding more context for short captions results in a small improvement. The results of the conditions using the pre-trained vectors are slightly worse than those obtained by simple word overlap. The similarity based classifier using the corpus specific word embeddings clearly outperforms all other variants.

6 Discussion

One would expect that using word embeddings to compare the acronym definition with the context of the abbreviation always has a big advantage over a simple word overlap measure, like cosine similarity. Surprisingly, this is not the case if we use pre-trained vectors. When we train a model on the same corpus from which we have collected our examples (excluding the parts, however, used for the examples), we get much better results indeed. This shows in the first place, that unsupervised disambiguation of acronyms is feasible, if we use word embeddings to represent the alternative definitions and the context. In the second place we have shown that training word embeddings on a specialized corpus with text comparable to those in the task to be carried out, has a big advantage over using available embeddings trained on a general corpus, even if the used corpus is much smaller than the general corpus.

6.1 Error analysis

Our data set is generated automatically and not manually corrected. Therefore, we have some errors in our data set that cause misclassifications. E.g., there are two definitions for DJF: *December January February* and *Dec Jan Feb*. Obviously, the first one is the correct one, but the classifier might find either. In some cases different but equivalent definitions are found for an acronym, like *Gap Junction Channel* and *Gap Junctional Communication* for GJC, or the same acronym is used for slightly different, but not

Table 4: Accuracy of acronym disambiguation using small context for complete models and models with restricted set of word embeddings.

Model	*all words*	*intersection*
Distr. Sim. – Google News	0.71	0.64
Distr. Sim. – Own vectors	0.84	0.83

identical concepts, like IPSO for *Improved Particle Swarm Optimization* and *Iteration Particle Swarm Optimization*.

Table 3 gives two typical examples of errors. The first example has a very misleading context. While words like *water* and *ocean* are not very typical for cell experiments, at the same time the word day, that is part of the alternative definition, occurs several times in the context. The second example is a typical case for acronyms from related disciplines and a context that is too short and general to find the correct definition.

Our impression is, that about 2% to 5% of the examples are erroneous, problematic or simply do not have enough information to disambiguate the acronym. Thus the upper bound for the accuracy would be around 0.95. Our result of 0.86 is already very close to this.

6.2 Quality of embeddings vs. number of embeddings

Finally we wanted to see whether we get better results because we have word embeddings for more (corpus specific) words, or because the word embeddings trained on the specialized corpus are better suited for our purpose. To get more insight in this question, we repeated the tests in Table 2 using only words that have an embedding in our model and in the Google News model. There are only 60,940 tokens that are shared among both models. The huge drop in the amount of words in our model (originally embeddings for 1,367,648 tokens) can be explained by the fact that we computed embeddings for many number, symbols, etc. The results for the repeated experiment using only words common to both models are given in Table 4. We see, that there is hardly any change in the results for our model. Surprisingly, the Google News Model seems to loose a number of important words. Thus we can conclude that the better results can be explained by the fact that the word embeddings in the specialized model better reflect the meaning of the words in our corpus of scientific papers.

6.3 Results with high confidence

Figure 4 suggests that we can achieve an accuracy of over 0.95 if we only consider cases in which the difference between the cosine similarities is at least 0.1. (Of course the exact value should be determined using an independent data set). This way we can still classify over 76% of all examples. Thus we can resolve a large number of acronyms in our database. However, we have to keep in mind that for most definitions just one example could be found. Thus, the most likely definition of any unresolved acronym is a definition that we have not seen up to now.

7 Conclusion and Future Work

We started our experiment with a very simple method to find definitions of acronyms in scientific papers. Using definitions from other papers we can resolve many unresolved definitions. Nevertheless, an obvious way to improve the overall results of acronym expansion would be to improve the methods to find definitions.

From our point of view the presented method has two aspects that should be improved in future work. While we use advanced methods to compute word representations, for a text we simply take the (weighted) average of the word representations. We will learn text representations with neural nets in future as well. A consequent next step then would be to learn text similarity as well.

The present result encourages us to follow this way, since we have shown (1) that training word embeddings on our own corpus gives much better results than using standard word vectors and that

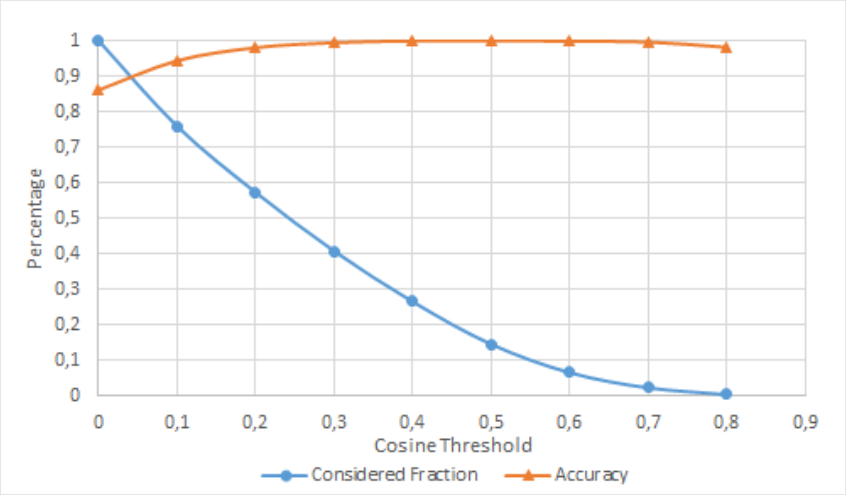


Figure 4: Fraction of classified pairs and accuracy when only examples with a difference between the cosine values above a given threshold are considered.

(2) we can obtain very good results for domain independent acronym disambiguation using these word embeddings.

Acknowledgments

The presented work was developed within the NOA Project - Automatic Harvesting, Indexing and Provision of Open Access Figures from the fields of Engineering and Technology Using the Infrastructure of Wikimedia Commons and Wikidata - funded by the DFG under grant number 315976924. NOA is a cooperative project of the Hochschule Hannover and the Technische Informationsbibliothek Hannover.

References

Rosa Tsegaye Aga, Christian Wartena, and Michael Franke-Maier. 2016. Automatic Recognition and Disambiguation of Library of Congress Subject Headings. In *Research and Advanced Technology for Digital Libraries - 20th International Conference on Theory and Practice of Digital Libraries, TPDL 2016, Hannover, Germany, September 5-9, 2016, Proceedings*, pages 442–446.

Rui Antunes and Sérgio Matos. 2017. Biomedical Word Sense Disambiguation with Word Embeddings. In Florentino Fdez-Riverola, Mohd Saberi Mohamad, Miguel Rocha, Juan F. De Paz, and Tiago Pinto, editors, *11th International Conference on Practical Applications of Computational Biology & Bioinformatics*, pages 273–279. Springer International Publishing, Cham. DOI: 10.1007/978-3-319-60816-7_33.

Satanjeev Banerjee and Ted Pedersen. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *Ijcai*, volume 3, pages 805–810.

Jeffrey T Chang, Hinrich Schütze, and Russ B Altman. 2002. Creating an online dictionary of abbreviations from medline. *Journal of the American Medical Informatics Association*, 9(6):612–620.

Jean Charbonnier, Lucia Sohmen, John Rothman, Birte Rohden, and Christian Wartena. 2018. Noa: A search engine for reusable scientific images beyond the life sciences. In Gabriella Pasi, Benjamin Piwowarski, Leif Azzopardi, and Allan Hanbury, editors, *Advances in Information Retrieval*, pages 797–800, Cham. Springer International Publishing.

Gregory P Finley, Serguei VS Pakhomov, Reed McEwan, and Genevieve B Melton. 2016. Towards Comprehensive Clinical Abbreviation Disambiguation Using Machine-Labeled Training Data. *AMIA Annual Symposium Proceedings*, 2016:560–569.

Sam Henry, Clint Cuffy, and Bridget McInnes. 2017. Evaluating feature extraction methods for knowledge-based biomedical word sense disambiguation. In *BioNLP 2017*, pages 272–281. Association for Computational Linguistics.

Antonio J. Jimeno-Yepes, Bridget T. McInnes, and Alan R. Aronson. 2011. Exploiting MeSH indexing in MEDLINE to generate a data set for word sense disambiguation. *BMC Bioinformatics*, 12(1):223, Jun.

Katrin Kirchhoff and Anne M Turner. 2016. Unsupervised resolution of acronyms and abbreviations in nursing notes using document-level context models. *EMNLP 2016*, page 52.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26. ACM.

Rada Mihalcea. 2011. Word sense disambiguation. In U. S. Springer, editor, *Encyclopedia of Machine Learning*, pages 1027–1030.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

David Nadeau and Peter D. Turney, 2005. *A Supervised Learning Approach to Acronym Identification*, pages 319–329. Springer Berlin Heidelberg, Berlin, Heidelberg.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10.

Dieke Oele and Gertjan Van Noord. 2017. Distributional lesk: Effective knowledge-based word sense disambiguation. In *IWCS 2017 — 12th International Conference on Computational Semantics — Short papers*.

Naoaki Okazaki and Sophia Ananiadou. 2006. Building an abbreviation dictionary using a term recognition approach. *Bioinformatics*, 22(24):3089–3095, December.

Serguei VS Pakhomov, Greg Finley, Reed McEwan, Yan Wang, and Genevieve B Melton. 2016. Corpus domain effects on distributional semantic modeling of medical terms. *Bioinformatics*, 32(23):3635 – 3644.

Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. 2003. Using measures of semantic relatedness for word sense disambiguation. In *Computational linguistics and intelligent text processing*, pages 241–257. Springer.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Ariel S. Schwartz and Marti A. Hearst, 2011. *A simple algorithm for identifying abbreviation definitions in biomedical text*, pages 451–462. World Scientific.

Lucia Sohmen, Jean Charbonnier, Ina Blümel, Christian Wartena, and Lambert Heller. 2018. Figures in open access scientific publications. In *Research and Advanced Technology for Digital Libraries - 22nd International Conference on Theory and Practice of Digital Libraries, TPDL 2018, Porto, Portugal, September 10-13, 2018, Proceedings*. to appear.

Mark Stevenson, Yikun Guo, Abdulaziz Alamri, and Robert Gaizauskas. 2009. Disambiguation of biomedical abbreviations. In *Proceedings of the BioNLP 2009 Workshop*, pages 71–79. Association for Computational Linguistics.

Stephan Tulkens, Simon Suster, and Walter Daelemans. 2016. Using distributed representations to disambiguate biomedical and clinical concepts. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, pages 77–82. Association for Computational Linguistics.

Thi Ngoc Chau Vo, Tru Hoang Cao, and Tu Bao Ho. 2016. Abbreviation Identification in Clinical Notes with Level-wise Feature Engineering and Supervised Learning. In Hayato Ohwada and Kenichi Yoshida, editors, *Knowledge Management and Acquisition for Intelligent Systems : 14th Pacific Rim Knowledge Acquisition Workshop, PKAW 2016, Phuket, Thailand, August 22-23, 2016, Proceedings*, pages 3–17. Springer International Publishing, Cham. DOI: 10.1007/978-3-319-42706-5_1.

Yue Wang, Kai Zheng, Hua Xu, and Qiaozhu Mei. 2016. Clinical word sense disambiguation with interactive search and classification. In *AMIA Annual Symposium Proceedings*, volume 2016, page 2062. American Medical Informatics Association.

Yonghui Wu, Jun Xu, Yaoyun Zhang, and Hua Xu. 2015. Clinical abbreviation disambiguation using neural word embeddings. In *Proceedings of the 2015 Workshop on Biomedical Natural Language Processing (BioNLP)*, pages 171–176.

Yonghui Wu, Joshua C Denny, S Trent Rosenbloom, Randolph A Miller, Dario A Giuse, Lulu Wang, Carmelo Blanquicett, Ergin Soysal, Jun Xu, and Hua Xu. 2017. A long journey to short abbreviations: developing an open-source framework for clinical abbreviation recognition and disambiguation (card). *Journal of the American Medical Informatics Association*, 24(e1):e79–e86.

David Yarowsky. 2010. Word Sense Disambiguation. In Nitin Indurkhya and Fred J. Damerau, editors, *Handbook of Natural Language Processing, Second Edition*, pages 315–338.

Indigenous language technologies in Canada: Assessment, challenges, and successes

Patrick Littell
National Research Council of Canada
1200 Montreal Road
Ottawa, ON, K1A 0R6
patrick.littell@nrc.gc.ca

Anna Kazantseva
National Research Council of Canada
5071 West Saanich Road
Victoria, BC, V9E 2E7
anna.kazantseva@nrc.gc.ca

Roland Kuhn
National Research Council of Canada
1200 Montreal Road
Ottawa, ON, K1A 0R6
roland.kuhn@nrc.gc.ca

Aidan Pine
National Research Council of Canada
5071 West Saanich Road
Victoria, BC, V9E 2E7
aidan.pine@nrc.gc.ca

Antti Arppe
University of Alberta
4-32 Assiniboia Hall
Edmonton, AB, T6G 2E7
arppe@ualberta.ca

Christopher Cox
Carleton University
1125 Colonel By Drive
Ottawa, ON, K1S 5B6
christopher.cox@carleton.ca

Marie-Odile Junker
Carleton University
1125 Colonel By Drive
Ottawa, ON, K1S 5B6
MarieOdile.Junker@carleton.ca

Abstract

In this article, we discuss which text, speech, and image technologies have been developed, and would be feasible to develop, for the approximately 60 Indigenous languages spoken in Canada. In particular, we concentrate on technologies that may be feasible to develop for most or all of these languages, not just those that may be feasible for the few most-resourced of these. We assess past achievements and consider future horizons for Indigenous language transliteration, text prediction, spell-checking, approximate search, machine translation, speech recognition, speaker diarization, speech synthesis, optical character recognition, and computer-aided language learning.

1 Introduction

There are approximately 60 Indigenous[1] languages from 10 distinct language families (Rice, 2008) currently spoken in Canada. Several of these languages have tens of thousands of speakers and are still acquired by most children, whereas others have a few tens or hundreds of mostly-elderly speakers; in all, 260,550 report speaking an Indigenous language at at least a conversational level (Statistics Canada, 2016). All of these languages are under significant pressure from English and French, but also have many young people interested in learning. The resurgent strength of community-driven Indigenous linguistic and cultural reclamation in Canada is at the heart of the growing demand for Indigenous language courses, materials and technology.

Indigenous languages are of paramount importance to the nations that speak them and the benefits associated with their use are wide-ranging (Whalen et al., 2016; Reyhner, 2010; Oster et al., 2014; Marmion et al., 2014). As a specific example, research in psychology has shown a compelling correlation

[1] In this document, "Indigenous languages" will specifically refer to Indigenous languages spoken in Canada.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2620–2632
Santa Fe, New Mexico, USA, August 20-26, 2018.

between Indigenous language use and a decrease in youth suicide rates on reserves in British Columbia (Chandler, 1998; Hallett et al., 2007). With increased awareness of these benefits has come increased interest by both Indigenous communities and federal and provincial governments in language technology development, to promote the revitalization and documentation of these languages.

However, the development of Indigenous language technologies faces many challenges: most of these languages are highly morphologically complex, there is relatively little text and speech data available, and there can be significant differences in dialects and orthographies that make it difficult to develop applications that work for all users. Well-known "flagship" language technologies that require large amounts of training data, like machine translation and automatic speech recognition, are therefore probably only feasible to develop for a few of the better-resourced languages such as Inuktitut.

There are nonetheless many practical language technologies that would be feasible to develop for a large number of these languages, and in some cases already have been developed. In this document we assess the feasibility of text (§4), speech (§5), image (§6), and educational (§7) technologies for Indigenous languages, based on past efforts in developing them for these and other low-resource languages.[2]

Disclaimer This document represents the personal opinions of the authors regarding the feasibility of certain technologies, and is not a statement of Government of Canada policy or priorities.

2 Scope and organization

This document will primarily assess user-level applications like search and spell-checking, rather than software that primarily exists to enable linguistic research. This delineation is very approximate, however, as many such applications will have benefits for both kinds of users.

This document also will not be a general inventory of digital resources such as online corpora and lexica. The collection and dissemination of these resources is, of course, highly important and is often the foundational work that makes these technologies possible; it is just that such an inventory would be outside the scope of a document of this size.[3]

In terms of organization, technologies will be clustered from the point-of-view of the practical application of a technology (e.g., spell-checking or text prediction), rather than be organized by the computational model that makes the application possible (e.g., a finite-state transducer or statistical language model). A finite-state grammar of a language (e.g. Snoek et al. (2014), Dunham (2014), Arppe et al. (2017a), Bowers et al. (2017), Harrigan et al. (2017)) can power a number of practical applications from spell-checking, to morphologically-aware search and browsing of dictionaries and corpora, to computer-aided language learning (Arppe et al., 2016).

This document will categorize technologies into five groups, according to the feasibility of developing these for a wide range of Indigenous languages, ranging from "already available" to "infeasible for most languages" (§8). It should be emphasized that these are not ratings of desirability, impact, worthiness for funding, or the relative importance of these technologies to language communities. By contrast, Krauwer (2003) proposed BLARKs - Basic Languages Resource Kits that list basic language technologies and resources needed for successful support and further research of under-represented languages in the European context. Arppe et al. (2016) extend the model to define resource and application priorities for the endangered languages of Canada - EL-BLARK (BLARK for Endangered Languages). This survey finds many similarities with the applications proposed by Arppe et al. (2016).

3 General challenges

There are many challenges that are commonly encountered during the development of Indigenous language technologies, and are encountered in almost all Indigenous languages.

[2]The inventory of existing technologies presented here is likely incomplete, as many language technologies are not published academically or publicized outside of their communities.

[3]An extensive inventory of open-source resources, in both Indigenous and other languages, is available at `github.com/RichardLitt/endangered-languages`. There are also a number of Indigenous language education and reference apps on the iOS App Store and Google Play; Animikii (`www.animikii.com`) maintains a growing list of these at `www.animikii.com/blog/apps-for-learning-an-Indigenous-language`.

3.1 Morphological complexity

Indigenous languages are typically very morphologically complex, with most being polysynthetic or agglutinative. It is commonly the case that a single word carries the meaning of what would be an entire clause in English and French.

(1) *iah th-a-etsi-te-w-ate-wistohsera-'tarih-á:t-ha-k-e'*
no NOT-WOULD-AGAIN-WE-ALL-OWN-butter-HOT-CAUSE-HABIT-CONTIN-PERF.

'We will no longer keep heating up our butter.' Mohawk (Mithun, 1996, p. 170)

(2) *Qanniqlaunngikkalauqtuqlu,* *aninngittunga*
qanniq-lak-uq-nngit-galauq-tuq-lu, ani-nngit-junga
snow-a.little-frequently-NOT-although-3.IND.S-and go.out-NOT-1.IND.S

'And even though it's not snowing a great deal, I'm not going out.'
Inuktitut (Micher, 2017, p. 102)

This complexity presents a challenge for many applications and algorithms, especially those that encode assumptions about the atomic word being the basic unit of meaning/structure, or even the assumption that concatenative morphological analysis is sufficient for finding sub-word units (Arppe et al., 2017a).

3.2 Limited training data

For most languages, there is little to no digitized text or audio available for use as training data, at least not at the scale required for modern statistical or neural NLP. Existing technologies for Indigenous languages have therefore, with a few exceptions, been exclusively rule-based.

A further problem related to the lack of training data is that available training data often comes from only a single domain. For example, the bulk of Inuktitut parallel text comes from Nunavut Legislative Assembly transcripts, but this genre is highly self-similar, and the application of a machine translation system trained on this corpus will likely have difficulties translating other genres like conversation or literature.

A promising research frontier to address limited training data is multilingual modeling; many of the least-resourced Indigenous languages are reasonably closely related to a more-resourced language. For example, automatic speech recognition in Seneca could be trained in part on other Iroquoian languages like Mohawk and Oneida (Jimerson and Prud'hommeaux, 2018), since they have similar phonetic inventories but more available speech data.

3.3 Dialectal and orthographic variation

Most Indigenous languages have a variety of dialects, but often sources and research articles only represent one dialect, or the orthographic standards were developed for a particular dialect and it is unclear how they apply to related dialects. It is sometimes the case that the roadblock to providing technology more widely in a language community is that the dialectal situation is poorly understood, and more basic research on dialectal differences is needed.

Furthermore, even within a single dialect, published works can use a variety of orthographies, and even works using the same orthography often differ in the details such as the encoding of particular diacritics or which morphemes/enclitics are written as separate words or joined. This variety can even be seen in single works, such as those with multiple contributors or transcribers.

Dialectal and orthographic variation pose a particular problem to rule-based text processing systems, since these are usually based on one relatively-well-studied dialect and use particular writing conventions that user-contributed data do not always share. A promising frontier of research to address this, as seen in Micher (2017), is to begin with an existing rule-based system and use it to bootstrap a statistical or neural system (in this case a recurrent neural network) that is more robust when faced with noisy data and unknown morphemes.

4 Text technologies

4.1 Fonts and keyboard layouts

Given the widespread adoption of Unicode and a substantial expansion of character coverage in standard Windows and MacOS fonts like Times New Roman, font coverage of Indigenous languages is currently very good so far as desktop operating systems are concerned. Both Windows and MacOS ship with the Euphemia font for Canadian Aboriginal Syllabics (§6.2), although for some languages Euphemia can display incorrect character orientations[4].

Special Roman characters, diacritics, and Syllabics are not always supported by system-installed keyboard layouts, necessitating the development of custom keyboard layouts. Fortunately, keyboard layout coverage of Indigenous languages is extensive; LanguageGeek[5] provides Windows and MacOS keyboards in almost every Indigenous language, while Tavultesoft Keyman[6] and FirstVoices[7] have developed keyboards for iOS and Android that offer complete coverage of Indigenous languages as well as support for other non-Indigenous languages.

4.2 Predictive text

One common request concerning keyboards (particularly mobile keyboards) is "predictive text" or "autocomplete", in which the keyboard offers shortcut buttons that suggest probable next words to the user depending on what they have already typed. This technology is especially desirable because it appeals to young users as well as to advanced second language learners.

The Multiling O[8] keyboard app for Android offers dictionary-based predictive text in the SENĆOŦEN language.

Maheshwari et al. (2018) examine word and character-based language models for text prediction of Mi'kmaq, based on a small web corpus.

Given the relative paucity of digital text corpora for many languages, it is likely that most predictive text systems will not be able to rely entirely on statistical models, and will instead be built on rule-based (e.g. finite state) or hybrid statistical/rule-based systems.

4.3 Orthography conversion

Almost all Indigenous languages have been written in several different orthographies. While there is a general trend towards orthographic unification in most communities, it is still common to find geographical or generational differences in how languages are written.

Conversion between modern orthographies is generally straightforward, and there exist many applications that manage these conversions[9,10,11,12,13,14]. Conversion between historical and modern orthographies can be more difficult, as historical orthographies often made different assumptions about the vowel and consonant inventories of these languages. There exists a rule-based transliterator between historical and modern Kwak'wala text[15], but the correspondences are somewhat irregular and thus the results are not completely reliable.

4.4 Spell-checking

Although Indigenous languages of Canada have a relatively short tradition of writing, it is quickly gaining steam, especially among young users and learners. However, writing—especially writing "correctly"

[4] www.eastcree.org/cree/en/resources/how-to/cree-fonts/syllabic-font-orientation/
[5] www.languagegeek.com/keyboard_general/all_keyboards.html
[6] keyman.com
[7] firstvoices.com
[8] play.google.com/store/apps/details?id=kl.ime.oh
[9] syllabics.atlas-ling.ca/
[10] www.creedictionary.com/converter/
[11] inuktitutcomputing.ca/Transcoder/
[12] mothertongues.org/#convertextract (Pine and Turin, 2018)
[13] www.eastcree.org/cree/en/resources/syllabic-convertor/
[14] www.giellatekno.uit.no/index.eng.html
[15] orth.nfshost.com/?lang=kwk&input=umista&output=boas

according to official community standards—can be particularly difficult for English- or French-dominant writers, since it requires making sound distinctions that English and French lack. Spell-checking is therefore a frequently-requested technology.

Since all Indigenous languages are morphologically complex, a purely word-list based spell-checking system is typically infeasible; a given stem can have hundreds or even millions of possible derivations/inflections. Corpus-based spell-checkers would have a similar problem; even when a digital corpus is available, only a small fraction of possible derivations/inflections will occur it. Therefore, efforts to develop spell-checkers in Indigenous languages typically concentrate on finite-state technology, since this allows the specification of very large lexicons in an efficient and succinct manner.

A Plains Cree spell-checker based on FST technology is available for system-wide use in recent versions of MacOS, and versions for Microsoft Office and Libre Office are in development (Arppe et al., 2016). The Giella infrastructure (Moshagen et al., 2013) offers an easy way to create FST-based spell-checkers that can be integrated into LibreOffice and, to a limited extent, into Microsoft Office. The spell-checkers use finite-state transducers as a backend, but it is possible to specify spelling relaxations as well as to include modules for likely or common errors. Theoretically the framework allows other types of language models as well, but they have been relatively untested.

An unexpected problem with integrating spell-checkers into mainstream office software is tokenization, since some Indigenous languages use commas, colons, and apostrophes to indicate phonetic differences, whereas many text processing systems assume internally that these are token boundaries. This points to a need for more flexible tokenization within mainstream office software to accommodate these languages.

4.5 Paradigm generation

It has generally been acknowledged that effectively teaching polysynthetic languages requires teaching morphology (Kell, 2014). Since all Indigenous languages have complex verb morphology, one frequent educational need is verb conjugators (Junker and MacKenzie, 2010; Junker and MacKenzie, 2011; Baraby and Junker, 2011; Arppe et al., 2017b), either stand-alone or integrated into an online dictionary.

For most Indigenous languages, learning morphology automatically from corpora is not a viable option. However, symbolic systems, especially those based on finite-state transducers (FSTs) have been successfully implemented for a number of languages. For example, Arppe et al. (2017b) developed an FST for East Cree by leveraging a lexical database. Arppe et al. (2016) and Arppe et al. (2017a) do not release stand-alone verb conjugators, but make verb conjugations available as a part of morphologically-aware online dictionaries for Plains Cree and Tsuut'ina languages respectively.

4.6 Approximate search

Approximate (or "fuzzy") search is a key language technology in situations where the language has not been widely written, or where a large proportion of technology users are learners. Moreover, whole-word search is problematic in highly polysynthetic/agglutinative languages, since the user's query may not use the inflectional form that appears in the dictionary or corpus. Both of these situations are common for Indigenous languages, and therefore the incorporation of approximate search is appropriate in nearly any text technology for these languages.

In general, approximate search can be done in a language-independent way—i.e., by simply counting the number of deletions, insertions, changes, and transpositions (Damerau, 1964; Levenshtein, 1966), without consideration of any language-specific properties—and can be done efficiently even on a large lexicon (Schulz and Mihov, 2002). There are three ways the user experience can be further improved for a particular language: by adapting to actual user queries, by building phonetic knowledge into the system, by making the search aware of morpheme breakdowns.

The East Cree[16] and Innu[17] dictionaries utilize relaxed search rules based on users' habits (Junker and Stewart, 2008).

[16] `dictionary.eastcree.org`
[17] `dictionnaire.innu-aimun.ca`

Mother Tongues Dictionaries[18] incorporates phonological background knowledge (e.g., that two sounds are similar and likely to be confused by users) in a finite-state approximate phonological search algorithm (Littell et al., 2017). It concentrates on Pacific Northwest languages where, due to the extensive consonant inventories and phonological complexity of these languages, approximate search is particularly important. This algorithm powers the search function in e-dictionaries for 17 Indigenous languages spoken in British Columbia, including FirstVoices'[19] mobile dictionary applications for iOS and Android, with dictionaries for 11 more languages currently in development.

Morphologically-aware search allows the user to find instances of their search query that may differ in one or more morphemes(Johnson et al., 2013). The Giella infrastructure offers morphology-aware search in dictionaries that are generated by linking a morphological model with lexical resources (and possibly with text corpora). A user can search with any inflected word form of a lemma (or root), possibly taking into account common spelling errors and spelling relaxations (Moshagen et al., 2013). Snoek et al. (2014) and Harrigan et al. (2017) use this technology to allow searching a dictionary of Plains Cree for specific lemmas. Similar capabilities exist for East Cree (Arppe et al., 2017b), Tsuut'ina (Arppe et al., 2017a), Northern Haida (Lachler et al., 2018) and Odawa (Bowers et al., 2017).[20]

4.7 Machine translation

Machine translation is one of the best-known language technologies, and receives significant attention from academia, industry, and the general public, so one of the more common queries from Indigenous groups is whether machine translation would be feasible for their languages.

The current state-of-the-art of machine translation is relatively language neutral, but requires very large amounts of parallel text, which is currently unavailable in most Indigenous languages save Inuktitut. Even then, given the complexity of Inuktitut morphology and the limited corpus available, it is probable that such systems will be, at best, aides to human translators working within that domain, rather than a general-purpose consumer technology like Google Translate.

Several prerequisite steps for Inuktitut machine translation have been achieved, including morphological segmentation (the Uqailaut analyzer[21] and its neural generalization (Micher, 2017)), and sentence and word-level alignment (Martin et al., 2003; Langlais et al., 2005). There are several Inuktitut-English machine translation systems currently under development.

The prerequisite steps can themselves power practical technology. For example, the WeBInuk translation memory system, an adaptation of the WeBiText system (Désilets et al., 2008) mines Inuktitut-English text and uses word alignments to suggest translations to Inuktitut translators.

5 Speech technologies

There has been little development of Indigenous language speech technology so far, but consultation with language communities has suggested that speech technologies are greatly desired, as these languages and cultures are traditionally oral. Text technologies typically expect the user to be able to write their language using the same conventions that the developer expects, which is a problematic expectation in languages without widespread agreement about written conventions. Speech technologies therefore offer an attractive proposition for users more accustomed to speaking and hearing their language than writing and reading it.

5.1 Automatic speech recognition

Full-vocabulary automatic speech recognition (ASR) currently requires large amounts of transcribed audio, and is therefore unlikely to be feasible in most Indigenous languages for the foreseeable future, at least not at a high degree of accuracy. However, even a low degree of accuracy can significantly assist human transcription; this technology, sometimes called Transcription Acceleration (TA), would probably be feasible for at least some languages now.

[18] mothertongues.org
[19] firstvoices.com
[20] altlab.artsrn.ualberta.ca/tools-applications/
[21] www.inuktitutcomputing.ca/Uqailaut/info.php

Jimerson and Prud'hommeaux (2018) has developed a preliminary ASR system for the Seneca language, and the Persephone ASR (Adams et al., 2018) system is being adapted to provide transcription acceleration within the Dative Online Linguistic Database interface (Dunham, 2014), which currently powers dozens of Indigenous language documentation efforts in Canada.

The frontier in speech recognition that is most promising for low-resource languages is multilingual recognition, in which a model trained on a large variety of languages can help compensate for a lack of transcribed speech data in the target language. A challenge for multilingual speech recognition is that some Indigenous languages, particularly in the Pacific Northwest, are global outliers in terms of phonological complexity, with large consonant inventories, rare consonants such as [t͡ɬ], and sometimes long sequences of consonants without the need for intervening vowels. At the very least, the development of practical multilingual recognition models would allow such languages to pool their resources, even if the difference between these and languages outside the region is too great to use a "universal" model.

5.2 Audio keyword search

The primary challenge of ASR in any language is the wide range of inputs the system might encounter—basically, anything that a person might talk about. On the other hand, ASR can also be used to find a *particular* word in an audio recording: the decision is not "what words are these?" but the simpler decision "is this part of the recording an instance of this word?".

This problem, of audio keyword search, is more tractable, but still potentially very useful for making un-transcribed speech recordings more accessible to the public, allowing users to search more quickly through long audio recordings in search of particular words or topics. The National Research Council of Canada (NRC), is collaborating with the Computer Research Institute of Montréal (CRIM) and the Pirurvik Centre on an audio keyword search project for Canadian Broadcasting Company (CBC) radio broadcasts in the Inuktitut and Cree languages.

5.3 Speech/text alignment

Even when resources are too limited to allow full, "open-vocabulary" ASR, prerequisite steps to ASR can be valuable in their own right. One of the prerequisite steps to both ASR and speech synthesis is speech/text alignment (sometimes called "forced alignment"), which involves taking a speech recording and a transcription of it and determining which segments of audio correspond to words and/or phonemes in the transcription.

This intermediate step can itself be of value for education, in creating time-aligned closed-captions from transcribed recordings, and read-along activities such as those available on the East Cree language portal[22] (Luchian and Junker, 2004; Junker et al., 2016) and in the Inuktitut-language Uqalimaarluk ("Read To Me") app for iPad[23].

5.4 Audio segmentation and speaker diarization

Even if automatic speech recognition ("what was said?") is beyond the means of current technology, speaker diarization ("who spoke when?") can be of great value, helping users to more quickly comb through large amounts of audio data in search of examples by a particular speaker or in a particular language.

The NRC-CRIM collaboration mentioned above (§5.2) will also be developing tools for automatic segmentation and speaker diarization. These are relatively language-neutral technologies that could be used in other languages as well.

5.5 Speech synthesis

The converse of automatic speech recognition, text-to-speech (TTS) is somewhat more feasible in low-resource situations. A limited-domain text-to-speech system (such as a talking clock or public transit announcement system) can be trained with just minutes or hours of total recordings, so long as the samples are adequately representative of the target domain.

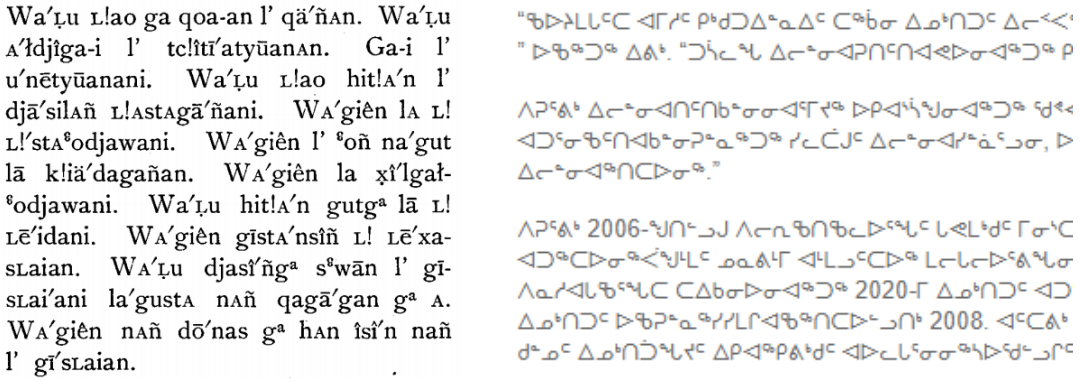
[22] www.eastcree.org/cree/en/lessons/sing-along/

[23] itunes.apple.com/ca/app/uqalimaarluk/id1348117314

Interest in text-to-speech has come from communities where interested learners outnumber fluent speakers, such that the learner might want to know how a word is pronounced but does not currently have access to a speaker to model pronunciation for them. Interest has also come from communities working on projects such as talking online dictionaries, in which the inflectional complexity of the language (§3.1) has meant that it is not feasible to record every possible inflection of a word. In such projects, TTS could allow the user to hear the pronunciation of any inflected form of the word, rather than just uninflected stems.

To our knowledge there are not yet any complete speech-synthesis project in an Indigenous language spoken in Canada, but Synscenter Refsnæs and Oqaasileriffik (the Language Secretariat of Greenland) have developed a general-purpose text-to-speech system for Kalaallisut[24] (West Greenlandic), which is closely related to Inuktitut.

6 Image technologies


Wa′ɻu ʟ!ao ga qoa-an l′ qä′ñan. Wa′ɻu ᴀ′ɬdjîga-i l′ tc!ītī′atyūanan. Ga-i l′ u′nētyūanani. Wa′ɻu ʟ!ao hit!ᴀ′n l′ djā′silañ ʟ!astagā′ñani. Wᴀ′giên lᴀ ʟ! ʟ!′stᴀ^{s}odjawani. Wᴀ′giên l′ ^{s}oñ na′gut lā k!ïä′dagañan. Wᴀ′giên la ɻî′lgał-^{s}odjawani. Wa′ɻu hit!ᴀ′n gutg^{a} lā ʟ! ʟē′idani. Wᴀ′giên gīstᴀ′nsîñ ʟ! ʟē′xa-sʟaian. Wᴀ′ɻu djasî′ñg^{a} s^{s}wān l′ gī-sʟai′ani la′gustᴀ nᴀñ qagā′gan g^{a} ᴀ. Wᴀ′giên nᴀñ dō′nas g^{a} hᴀn îsî′n nañ l′ gī′sʟaian.

"ᖃᐅᔨᒪᒪᑦᑕ ᐊᒥᓱᑦ ᑭᒃᑯᑐᐃᓐᓇᐃᑦ ᑕᖅᑲᓂ ᐃᓄᒃᑎᑐᑦ ᐃᓕᓴᐸ
" ᐅᖃᖅᑐᖅ ᐃᕕᒃ. "ᑐᑦᓚᖏᑦ ᐃᓕᓴᓂᐊᑭᑎᑎᑦᑎᐊᐸᐅᓂᐊᖅᑐᖅ ᑭ

ᐱᑭᕕᒃ ᐃᓕᓴᓂᐊᑎᑎᑎᑲᒃᓂᓂᐊᖏᓪᓘ ᐅᑭᐊᑦᓴᒍᓂᐊᖅᑐᖅ ᖃᑯᐸᓂ
ᐊᑐᑦᓂᖅᑎᐊᑲᒃᓂᑭᓇᖅᑐᖅ ᓯᓚᑕᒍᑦ ᐃᓕᓴᓂᐊᓯᓴᓪᓂᓂ, ᐅ
ᐃᓕᓴᓂᐊᖅᑎᑕᐅᓂᖅ."

ᐱᑭᕕᒃ 2006-ᖃᒍᑎᓪᓗᒍ ᐱᓕᓕᖅᑎᑕᐅᑦᖃᒪᑦ ᒪᓭᒪᒃᑯᑦ ᒥᓂᑦᑕ
ᐊᑐᖅᑕᐅᓂᖅᐸᖃᒧᑦᒪᑦ ᓄᓇᕕᒋᒥ ᐊᒻᒪᓗᑦᑕᐅᖅ ᒪᓕᓚᓕᐅᖅᕕᖃᖅᓯᓂ
ᐱᓇᓯᐊᒪᖅᑦᖃᒪᑕ ᑕᐃᑲᓂᐅᓂᐊᖅᑐᖅ 2020-ᒥ ᐃᓄᒃᑎᑐᑦ ᐊᑐ
ᐃᓄᒃᑎᑐᑦ ᐅᖃᕈᓐᓇᖅᓯᓯᒪᓕᐊᖅᖃᖅᑎᑕᐅᓪᓗᑎᒃ 2008. ᐊᑦᑕᕕᒃ
ᑯᓘᓄᑦ ᐃᓄᒃᑎᑐᖃᒪᑦ ᐃᑭᐊᖅᑭᕕᒃᑯᑦ ᐊᐅᓪᓚᑦᓂᓂᖅᓴᐅᖅᑯᓪᓗᑎᑦ

Figure 1: Example of a historical printed document in the Northern Haida language (Hubert et al., 2016), and Canadian Aboriginal Syllabics representing the Inuktitut language, courtesy of `tusaalanga.ca`.

6.1 Optical character recognition for Roman orthographies

Optical character recognition (OCR) has been successfully been applied to Indigenous language documents, including historical manuscripts printed using pre-digital presses (Fig. 1). Hubert et al. (2016) report a high degree of success on OCR with only a few pages of training data, suggesting that OCR would be feasible to implement for a wide range of Indigenous languages.

The challenge for OCR on many Roman orthographies for Indigenous languages is the proliferation of diacritics and superscript letters, particularly in languages with extensive consonant inventories. Diacritics and superscripts are difficult to differentiate from punctuation and from each other, and depending on the font resources, some letter-diacritic combinations may be very hard to distinguish. For example, British Columbian orthographies based on the Royal B.C. Museum recommendations often use underlined ⟨g⟩ for a uvular voiced plosive, and in some fonts (or in typewritten documents), this underline can overlap the descender on the ⟨g⟩.

6.2 Optical character recognition for Canadian Aboriginal Syllabics

While most Indigenous languages are written using a Roman orthography, several varieties of Inuktut, Cree, and Ojibwe[25] use a system called Canadian Aboriginal Syllabics (Fig. 1).

Canadian Aboriginal Syllabics (often simply called "Syllabics") is a "rotational syllabary" in which the shape of the glyph indicates the syllable's consonant and its rotational orientation to the vowel. There are also smaller, superscript characters that indicate consonants where no vowel follows (e.g. in a syllable

[24] `oqaasileriffik.gl/langtech/martha/`

[25] Syllabics have also historically been used for Blackfoot and some Athabascan languages such as Dakelh (Carrier), but have fallen out of use in favor of Roman orthographies.

coda). Like superscript characters in Roman orthographies, superscript characters in Syllabics can pose a problem for OCR, since due to their size and position they are easily confused for punctuation marks or with each other. Nonetheless, there have been several successful Syllabics OCR projects (e.g. Posgate and Leekam (2014)), and Inuktitut has since been included in the Tesseract OCR project[26] (Smith, 2007).

7 Computer-aided language learning

7.1 Course modules

Computer-aided language learning (CALL) course modules are widely available for Indigenous languages, particularly through the FirstVoices Language Tutor (FVLT) portal[27], which offers approximately 50 online courses covering many Indigenous languages, with exercises on listening, speaking, reading, and vocabulary development, as well as online language-learning games.

There are also language-specific CALL portals, including but not limited to:

- Tusaalanga[28] from the Pirurvik Centre, which offers exercises in five varieties of Inuktut.
- The Institut Tshakapesh learning portal[29], which offers educational games in the Innu language (Junker and Torkornoo, 2012; Junker et al., 2016). These were modeled after the eastcree.org lessons[30] for teaching syllabics, vocabulary and literacy.
- The *nêhiyawêtân* CALL portal for Plains Cree[31] fuses CALL and text technologies, in which students receive targeted feedback made possible by the integration of a finite-state morphology model (Arppe et al., 2016; Bontogon et al., 2018).
- The Yukon Native Language Centre[32] offers online audiovisual adaptations of language courses in eight Indigenous languages.

Several international CALL products have been adapted for Indigenous languages. 7000 Languages[33] adapts the Transparent Language software for low-resource and endangered languages, and offers courses on Denesuline, Dakota, and several varieties of Cree, Ojibwe, and Oji-Cree through partnerships with Grassroots Indigenous Multimedia[34] and the Manitoba First Nations Education Resource Centre[35]. Rosetta Stone[36] has also developed courses for Labrador Inuttitut and Kahnawá:ke Mohawk.

A forthcoming project headed by Dr. Marianne Ignace at Simon Fraser University presents an innovative "chat-based" UI (what is sometimes called "No-interface UI") for CALL apps, in which an AI tutor interacts with the student in a web-chat-like interface.

7.2 Phonetic tutorial

Some education applications focus more narrowly on the acquisition of speech sounds. Phonetic tutorials are particularly important in languages with rarer sounds, like lateral fricatives or ejective plosives.

The Yukon Native Language Centre[37] has developed a phonetic learning game in which players must count the number of instances of a particular sound (e.g., [t͡ɬ]) in a recording to mush a dog sled.

The Tiga Talk[38] app for iOS, originally a collection of speech-language pathology games for English, is currently being adapted to Cree to help support child acquisition.

[26] github.com/tesseract-ocr
[27] tutor.firstvoices.com
[28] tusaalanga.ca
[29] jeux.tsakapesh.ca
[30] lessons.eastcree.org
[31] oahpa.no/nehiyawetan/
[32] www.ynlc.ca
[33] 7000languages.org
[34] gim-ojibwe.org
[35] mfnerc.org
[36] www.rosettastone.com/endangered/projects
[37] ynlc.ca
[38] tigatalk.com/app/

The UBC eNunciate[39] tools use ultrasound to illustrate to students the articulatory gestures of the tongue and vocal tract that cannot ordinarily be seen, in the Upriver Halqemeylem, SENĆOŦEN, Secwepemc, and Blackfoot languages (Bliss et al., 2016).

7.3 Augmented reality and virtual reality

Augmented and virtual reality technologies are not specifically language or learning technologies, but there is a growing amount of interest in their application to Indigenous language education primarily due to their ability to be naturally integrated into popular "place-based eduction" (Sobel, 2004) practices.

The feasibility of implementing augmented and virtual reality projects is aided by the widespread interest in the technology and 3D game engines like Unity and Unreal. However, there are still very few implementations for Indigenous languages in Canada. Some examples include *Tuwitames*, an augmented reality story-telling app narrated in Secwepemctsín (Lacho, 2018), an augmented reality companion app to a Blackfoot card game (Goff et al., 2017), and *Schoolû*, a virtual reality application for teaching Cree syllabics. Yet another augmented reality app, *Wikiup*[40], is designed to take users on tours throughout Canadian cities, transforming landmarks by telling AR-enhanced Indigenous stories and histories, but not necessarily involving Indigenous languages.

8 Summary

Widely available Successful technologies that are already available for many Indigenous languages *Keyboard layouts* (§4.1), *Approximate search* (§4.6), *Computer-aided language learning* (§7).

Ready for wider implementation Technologies that have been developed for some languages, and that could feasibly be developed for most or all Indigenous languages: *Orthography conversion* (§4.3), *Optical character recognition* (§6.1, §6.2).

Awaiting implementation Technologies for which there is already a technological basis in a few languages (e.g., a finite-state analyzer has been written), or for which there exists a language-neutral technological basis, but for which practical user interfaces or language-specific implementations are not yet developed or widely available: *Spell-checking* (§4.4), *Paradigm generation* (§4.5), *Speech/text alignment* (§5.3).

Experimental Technologies that have not yet proven to be successful on Indigenous languages, but show promise in other low-resource language situations: *Predictive text* (§4.2), *Transcription acceleration* (§5.1), *Audio keyword search* (§5.2), *Audio segmentation and speaker diarization* (§5.4), *Text-to-speech* (§5.5).

Restricted feasibility Technologies that will likely be feasible only in more-resourced languages (e.g. Inuktitut, Cree): *Machine translation* (§4.7), *Automatic speech recognition* (§5.1).

From the above, it is clear that there are a number of text, speech, image, and CALL technologies that are either already available, or could be made more widely available, in many cases with relatively reasonable further investment. The boundary between the first three categories at various stages of implementability and the two last experimental and restricted ones appears to determined by the existence of technological solutions that work with typically quite sparse data resources that can be reasonably expected for Indigenous languages. Meanwhile, new developments (particularly in multilingual and finite-state/neural hybrid modeling) may make technologies possible that until recently seemed infeasible for Indigenous languages.

[39] enunciate.arts.ubc.ca
[40] wikiupedia.com

References

Oliver Adams, Trevor Cohn, Graham Neubig, Hilaria Cruz, Steven Bird, and Alexis Michaud. 2018. Evaluating phonemic transcription of low-resource tonal languages for language documentation. In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hlne Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France, May. European Language Resources Association (ELRA).

Antti Arppe, Jordan Lachler, Lene Antonsen, Trond Trosterud, and Sjur N. Moshagen. 2016. Basic language resource kits for endangered languages: A case study of Plains Cree. In *Proceedings of the 2016 CCURL Workshop. Collaboration and Computing for Under-Resourced Languages: Towards and Alliance for Digital Language Diversity, LREC 2016, May 23, 2016*, pages 1–9.

Antti Arppe, Christopher Cox, Mans Hulden, Jordan Lachler, Sjur N. Moshagen, Miikka Silfverberg, and Trond Trosterud. 2017a. Computational modeling of verbs in Dene languages: The case of Tsuut'ina. In Alessandro Jaker, editor, *Working Papers in Athabaskan (Dene) Languages*, pages 51–69.

Antti Arppe, Marie-Odile Junker, and Delasie Torkornoo. 2017b. Converting a comprehensive lexical database into a computational model: The case of East Cree verb inflection. In *Proceedings of the 2nd Workshop on the Use of Computational Methods in the Study of Endangered Languages (ComputEL-2)*, pages 43–47, Honolulu, March. Association for Computational Linguistics.

Anne-Marie Baraby and Marie-Odile Junker. 2011. Conjugaisons des verbes innus, (3e d.). http://verbe.innu-aimun.ca.

Heather Bliss, Strang Burton, and Bryan Gick. 2016. Ultrasound overlay videos and their application in indigenous language learning and revitalization. *Canadian Acoustics*, 44:136–37.

Megan Bontogon, Antti Arppe, Lene Antonsen, Dorothy Thunder, and Jordan Lachler. 2018. Intelligent computer assisted language learning (ICALL) for nêhiyawêwin: An in-depth user experience evaluation. *Canadian Modern Language Review*.

Dustin Bowers, Antti Arppe, Jordan Lachler, Sjur Moshagen, and Trond Trosterud. 2017. A morphological parser for Odawa. In *Proceedings of 2nd Workshop on Computational Methods for Endangered Languages (ComputEL-2)*, pages 2326–2330.

Michael J. Chandler. 1998. Cultural continuity as a hedge against suicide in Canada's First Nations. *Transcultural Psychiatry*, 35(4):191–219.

Fred J. Damerau. 1964. A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171–176, March.

Alain Désilets, Benoit Farley, Geneviève Patenaude, and Marta Stojanovic. 2008. WeBiText: Building large heterogeneous translation memories from parallel web content. *Proc. of Translating and the Computer*, 30:27–28.

Joel Robert William Dunham. 2014. *The online linguistic database: Software for linguistic fieldwork*. Ph.D. thesis, University of British Columbia.

Rebecca Goff, Brandon Goff, Caroline Running Wolf, Michael Running Wolf, Jesse Desrosier, Naatosi Fish, and Mizuki Miyashita. 2017. Playfully revitalizing languages and traditional knowledge through collaboration. http://hdl.handle.net/10125/41929.

Darcy Hallett, Michael J. Chandler, and Christopher E. Lalonde. 2007. Aboriginal language knowledge and youth suicide. *Cognitive Development*, 22(3):392–399.

Atticus G. Harrigan, Katherine Schmirler, Antti Arppe, Lene Antonsen, Trond Trosterud, and Arok Wolvengrey. 2017. Learning from the computational modelling of Plains Cree verbs. *Morphology*, 27(4):565–598.

Isabell Hubert, Antti Arppe, Jordan Lachler, and Eddie Antonio Santos. 2016. Training & quality assessment of an optical character recognition model for Northern Haida. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 3227–3234, Paris, France, May. European Language Resources Association (ELRA).

Robert Jimerson and Emily Prud'hommeaux. 2018. ASR for documenting acutely under-resourced indigenous languages. In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hlne Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France, May. European Language Resources Association (ELRA).

Ryan Johnson, Lene Antonsen, and Trond Trosterud. 2013. Using finite state transducers for making efficient reading comprehension dictionaries. In Stephan Oepen, Kristin Hagen, and Janne Bondi Johannessen, editors, *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013), NEALT Proceedings Series 16*, pages 59–71, Oslo University, Norway, May.

Marie-Odile Junker and Marguerite MacKenzie. 2010. East Cree (Northern dialect) verb conjugation. http://verbn.eastcree.org/.

Marie-Odile Junker and Marguerite MacKenzie. 2011. East Cree (Southern dialect) verb conjugation. http://verbs.eastcree.org/.

Marie-Odile Junker and Terry Stewart. 2008. Building search engines for Algonquian languages. In Karl S. Hele and Regna Darnell, editors, *Papers of the 39th Algonquian Conference*, pages 378–411, London, ON. University of Western Ontario Press.

Marie-Odile Junker and Delasie Torkornoo. 2012. Online language games for endangered languages: jeux.tshakapesh.ca, www.eastcree.org/lessons/. In *Proceedings of EDULEARN 12: International Conference on Education and New Learning Technologies*.

Marie-Odile Junker, Yvette Mollen, Hélène St-Onge, and Delasie Torkornoo. 2016. Integrated web tools for Innu language maintenance. In J. R. Valentine and M. MacCauley, editors, *Papers of the 44st Algonquian Conference*.

Sarah Kell. 2014. *Polysynthetic Language Structures and their Role in Pedagogy and Curriculum for BC Indigenous Languages*. BC Ministry of Education.

Steven Krauwer. 2003. The basic language resource kit (BLARK) as the first milestone for the language resources roadmap. In *Proceedings of the International Workshop Śpeech and Computer; SPECOM 2003, Moscow*, pages 8–15.

Jordan Lachler, Lene Antonsen, Trond Trosterud, Sjur N. Moshagen, and Antti Arppe. 2018. Modeling Northern Haida morphology. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2018)*, May.

David Lacho. 2018. *Developing an augmented reality app in Secwepemctsín in collaboration with the Splatsin Tsm7aksaltn (Splatsin Teaching Centre) Society*. Ph.D. thesis, University of British Columbia.

Philippe Langlais, Fabrizio Gotti, and Guihong Cao. 2005. Nukti: English-Inuktitut word alignment system description. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 75–78. Association for Computational Linguistics.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710.

Patrick Littell, Aidan Pine, and Henry Davis. 2017. Waldayu and Waldayu Mobile: Modern digital dictionary interfaces for endangered languages. In *Proceedings of the 2nd Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 141–150.

Radu Luchian and Marie-Odile Junker. 2004. Developing an on-line Cree read-along with syllabics. *Carleton University Cognitive Science Technical Report*, 2006-01.

Anant Maheshwari, Leo Bouscarrat, and Paul Cook. 2018. Towards language technology for Mi'kmaq. In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hlne Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France, May. European Language Resources Association (ELRA).

Doug Marmion, Kazuko Obata, and Jakelin Troy. 2014. *Community, identity, wellbeing: the report of the Second National Indigenous Languages Survey*. Australian Institute of Aboriginal and Torres Strait Islander Studies Canberra.

Joel Martin, Howard Johnson, Benoit Farley, and Anna Maclachlan. 2003. Aligning and using an English-Inuktitut parallel corpus. In *Proceedings of the HLT-NAACL 2003 Workshop on Building and using parallel texts: Data driven machine translation and beyond, Volume 3*, pages 115–118. Association for Computational Linguistics.

Jeffrey Micher. 2017. Improving coverage of an Inuktitut morphological analyzer using a segmental recurrent neural network. In *Proceedings of the 2nd Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 101–106. Association for Computational Linguistics.

Marianne Mithun. 1996. The Mohawk language. *MULTILINGUAL MATTERS*, pages 159–173.

Sjur N. Moshagen, Tommi A. Pirinen, and Trond Trosterud. 2013. Building an open-source development infrastructure for language technology projects. In *Proceedings of the 19th Nordic Conference of Computational Linguistics, NODALIDA 2013, May 22-24, 2013, Oslo University, Norway*, pages 343–352.

Richard T. Oster, Angela Grier, Rick Lightning, Maria J. Mayan, and Ellen L. Toth. 2014. Cultural continuity, traditional Indigenous language, and diabetes in Alberta First Nations: A mixed methods study. *International journal for equity in health*, 13(1):92.

Aidan Pine and Mark Turin. 2018. Seeing the Heiltsuk orthography from font encoding through to Unicode: A case study using convertextract. In Claudia Soria, Laurent Besacier, and Laurette Pretorius, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France, May. European Language Resources Association (ELRA).

Jess Posgate and Cathy Leekam. 2014. Digitizing Ontario's community memory: Bringing multicultural history online. In *Ontario Library Association Super Conference*, Toronto, ON, Jan. 30.

Jon Reyhner. 2010. Indigenous language immersion schools for strong Indigenous identities. *Heritage Language Journal*, 7(2):138–152.

Keren Rice. 2008. Indigenous languages in Canada. In *The Canadian Encyclopedia*.

Klaus U. Schulz and Stoyan Mihov. 2002. Fast string correction with Levenshtein-automata. *International Journal of Document Analysis and Recognition*, 5(1):67–85.

Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02*, ICDAR '07, pages 629–633, Washington, DC, USA. IEEE Computer Society.

Conor Snoek, Dorothy Thunder, Kaidi Lõo, Antti Arppe, Jordan Lachler, Sjur Moshagen, and Trond Trosterud. 2014. Modeling the noun morphology of Plains Cree. In *Proceedings of the 2014 Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 34–42, Baltimore, Maryland, USA, June. Association for Computational Linguistics.

David Sobel. 2004. *Place-based education: Connecting classroom and community*. Orion Society.

Statistics Canada. 2016. Aboriginal languages in Canada, 2016 census of population, catalogue no. 98-200-X.

Douglas H. Whalen, Margaret Moss, and Daryl Baldwin. 2016. Healing through language: Positive physical health effects of indigenous language use. *F1000Research*, 5.

Pluralizing Nouns across Agglutinating Bantu Languages

Joan Byamugisha
University of Cape Town
Cape Town
South Africa
jbyamugisha@cs.uct.ac.za

C. Maria Keet
University of Cape Town
Cape Town
South Africa
mkeet@cs.uct.ac.za

Brian DeRenzi
Dimagi
Cape Town
South Africa
bderenzi@dimagi.com

Abstract

Text generation may require the pluralization of nouns, such as in context-sensitive user interfaces and in natural language generation more broadly. While this has been solved for the widely-used languages, this is still a challenge for the languages in the Bantu language family. Pluralization results obtained for isiZulu and Runyankore showed there were similarities in approach, including the need to combine morphology with syntax and semantics, despite belonging to different language zones. This suggests that bootstrapping and generalizability might be feasible. We investigated this systematically for seven languages across three different Guthrie language zones. The first outcome is that Meinhof's 1948 specification of the noun classes are indeed inadequate for computational purposes for all examined languages, due to non-determinism in prefixes, and we thus redefined the characteristic noun class tables of 29 noun classes into 53. The second main result is that the generic pluralizer achieved over 93% accuracy in coverage testing and over 94% on a random sample. This is comparable to the language-specific isiZulu and Runyankore pluralizers.

1 Introduction

The need for generating plurals to ease the use of software tools is well known in several areas. In Natural Language Understanding, plurals are used during parsing to distinguish between atomic elements (simple nouns) and collectives (Schubert, 2015), while in Natural Language Generation, pluralization is essential during quantification (Schubert, 2015) and number agreement. For instance, to inflect properly for the grammatical number in end-user tools, like a calendar properly stating to set off an alarm "1 hour" before and not "1 hours", but also for more advanced tools, such as automatically generating patient discharge notes instructing how many pills the patient has to take. Automated pluralization may use regular expressions of the ending of the nouns, alike for English (Conway, 1998), or extensive linguistic resources to devise pluralization, as for German (Nakisa and Hahn, 1996).

Pluralization for Runyankore and isiZulu, members of the Bantu language family, showed that neither of the above approaches were feasible. It required a combined morphology with syntactic and semantic approach, where the latter was covered by using a combination of the noun and noun class to pluralize, along with a few similar refinements of Meinhof's noun class definition (Byamugisha et al., 2016). A surprising observation was that a similar approach could be used despite the regions where these languages are spoken being thousands of kilometers apart (Uganda and South Africa) and they belong to different Guthrie zones (Guthrie, 1948; Maho, 2009). This suggests there may be sufficient similarities not only for closely related languages, but also for languages in other Guthrie zones.

We thus aim to investigate the generalizability of pluralization for Bantu languages, specifically those with an agglutinating morphology. The following questions guided this investigation:

Q1: Is a combined morphology with syntactic and semantic approach needed to pluralize nouns in other agglutinating Bantu languages as well?

Proceedings of the 27th International Conference on Computational Linguistics, pages 2633–2643
Santa Fe, New Mexico, USA, August 20-26, 2018.

Q2: Are the limitations identified in Meinhof's noun class system for automated pluralization in Runyankore and isiZulu (Byamugisha et al., 2016) present in other agglutinating Bantu languages? If so, how can Meinhof's noun class system be revised systematically?

Q3: Are the same 'exceptions' to pluralization found in other agglutinating Bantu languages, and if so, are the solutions used for isiZulu and Runyankore able to pluralize the same exceptions in these languages?

Q4: In attempting to develop a generic pluralizer, how can the language-specific pluralizers be generalized whilst maintaining roughly the same accuracy?

In order to answer these questions, we investigated the generalizability of the Runyankore and isiZulu approach to agglutinating Bantu languages, both within the same Guthrie zone and across zones. Five agglutinating Bantu languages were selected: two of these, Luganda and isiXhosa, belong to the same zone as Runyankore and isiZulu, respectively, and the rest belong to other zones, being Kinyarwanda (J.10), Kikuyu (E.10), and chiShona (S.10) (Maho, 2009).

We first examined their grammatical structures (Déchaine et al., 2013; Zentz, 2016; Mberi, 2002; Jeon et al., 2015; Baertlein and Ssekitto, 2014; Kimenyi, 2004; Taraldsen, 2010), which revealed that a combined morphology with syntactic and semantic approach, where the noun and its noun class are provided as input, is also required for pluralization. This is principally because there are nouns that have the same prefix but are pluralized differently, due to belonging to different classes. Further, Meinhof's 1948 noun class tables of singular/plural pairs is underspecified as to which option is used when. Therefore, we have revised the tables so as to achieve a deterministic selection during pluralization. Further, the same exceptions to pluralization identified for Runyankore and isiZulu (Byamugisha et al., 2016) were present in each of the five languages, and the same solutions were also applicable. Finally, we developed a pluralizer that can pluralize nouns in agglutinating Bantu languages, with generic pluralization rules but language-specific resources, which achieved similar accuracies as Byamugisha et al., (2016)'s language-specific pluralizers.

This paper is arranged as follows. We briefly introduce relevant aspects of Bantu languages in Section 2. Section 3 justifies why Meinhof's noun class system should be redefined, and describes the revised noun class tables. The generic pluralizer is presented in Section 4 and it is evaluated in Section 5. We discuss in Section 6 and conclude in Section 7.

2 Basics of Bantu Languages

Bantu languages are a group of languages indigenous to Africa, from the south of Nigeria, covering most of central, east, and southern Africa (Nurse and Philippson, 2003). There are Bantu language speaking communities in 27 of the continent's 54 countries, about 240 million speakers, and number of languages ranges from 300 to 680 (depending on the criteria used (Nurse and Philippson, 2003). They are agglutinating, have concordial agreement systems, and all nouns are assigned to a noun class (NC). There are over 20 noun classes, although most languages use fewer (Nurse and Philippson, 2003; Mohlala, 2003). A NC is characterized by a class prefix, a specific singular/plural pairing, and agreement with other constituents (Zentz, 2016; Nurse and Philippson, 2003). The semantic generalizations of the type of nouns in each NC are shown in Table 1 (Keet and Khumalo, 2014; Baertlein and Ssekitto, 2014; Kimenyi, 2004; Jeon et al., 2015; Zentz, 2016; Taraldsen, 2010; Mohlala, 2003).

Bantu languages are conventionally categorized into 16 zones, referred to as Guthrie zones, which are further subdivided into decades; for example, zone J.10 contains languages from J.11 to J.19 (Nurse and Philippson, 2003; Maho, 2009). We selected five languages from five different zones, and their grammars formed the basis of whether a generic pluralizer could be developed. These 5 languages were selected because: (1) they are widely used in their native countries; (2) their linguistics are taught at university level; and (3) they are actively researched, which provides current documentation. Table 2 summarizes the findings into the grammatical features of interest.

The NC classification for each of these languages is based on Meinhof's NC definition, which places the plural form of a noun in a different class from its singular (Nurse and Philippson, 2003; Mohlala, 2003). This is deemed as a standard for defining NCs among linguists, as it facilitates cross-language

Noun Class	Description of nouns typically found in those classes
1 and 2	People and kinship
3 and 4	Plants, nature, and some parts of the body
5 and 6	Fruits, liquids, some parts of the body, loan words, and paired things
7 and 8	Inanimate objects
9 and 10	Loan words, tools, and animals
11	Long thin stringy objects, languages, and inanimate objects
12 and 13	Diminutives
14	Abstract concepts
15	Infinitive nouns
16, 17, and 18	Locative classes
19	Diminutives
20, 21, and 22	Augmentative
23	Locative class

Table 1: Classification of nouns into noun classes.

Language	Guthrie Zone	Number of NCs	Phonological Conditioning
chiShona	S.10	20	Yes, with VC, VE, and VH
isiXhosa	S.40	15	Yes, with VC and VE
isiZulu	S.40	17	Yes, with VC and VE
Kikuyu	E.50	17	Yes
Kinyarwanda	J.60	16	Yes, with VC and VE
Luganda	J.10	21	Yes, with VC and VE
Runyankore	J.10	20	Yes, with VC, VE and VC for VH

Table 2: Relevant features of the selected languages; VC: vowel coalescence; VE: vowel elision; VH: vowel harmony.

comparisons and use (Chavula and Keet, 2014). Further, as noted, the NC determines the agreement markers on the associated lexical categories such as adjectives and verbs. Moreover, there are NCs with the same class prefix but belonging to different NCs (Déchaine et al., 2013; Zentz, 2016; Mberi, 2002; Jeon et al., 2015; Baertlein and Ssekitto, 2014; Kimenyi, 2004; Taraldsen, 2010), which are highlighted in Table 3.

We identified grammatical differences among these languages, which makes the use of the exact same pluralizer impossible: (1) each language has its own lexicon of prefixes, (2) they do not have the same number of NCs; and (3) the rules for phonological conditioning and verbal morphology are language-specific (Mberi, 2002; Jeon et al., 2015; Ferrari-Bridgers, 2009; Muhirwe, 2007; Kimenyi, 2004; Taraldsen, 2010). Regarding the latter, examples include: Runyankore uses vowel coalescence and elision next to a nasal compound (such as '*nk*'), while Luganda uses vowel elision; and in Runyankore, Luganda, and isiZulu, *-u-* + vowel may become *w* (such as *ulu-* + *-azi* = *ulwazi* 'knowledge' in isiZulu), which is not the case in Kikuyu and chiShona. Nonetheless, the noted grammatical similarities among these languages are expected to be sufficient for the generalizability of the pluralization approach.

However, Meinhof's NC definition (Table 3) has several limitations when applied to computational tasks, which impede pluralization and generalizability:

1. Meinhof's NC tables lack the specification for some rules and when to apply them, especially for lesser known cases. In isiZulu, for example, the standard NC1 and NC3 singular prefix is *um*, with its plural prefix *aba* in NC2 and *imi* for NC4 (Byamugisha et al., 2016), yet for NC1 nouns whose stem commences with *a, e, i*, or *o* the plural prefix *ab* is used instead (Byamugisha et al., 2016).
2. The rules for the pluralization of some nouns deviate from those according to Meinhof's NC definition, such as mass nouns (which are neither pluralized no singularized) and prefix exceptions

NC	chiShona	isiXhosa	isiZulu	Kikuyu	Kinyarwanda	Luganda	Runyankore
1	**mu-**	**u-m-**	**u-m-/u-mu-**	**mu-**	**u-mu-**	**o-mu-**	**o-mu-**
2	va-	a-ba-	a-ba-/a-b-	a-	a-ba-	a-ba-	a-ba-
1a	**-**	**u-**	**u-**	**-**	N/A	N/A	N/A
2a	vana-	oo-	o-	**-**	N/A	N/A	N/A
2b	a-	N/A	N/A	N/A	N/A	N/A	N/A
3a	N/A	N/A	**u-**	N/A	N/A	N/A	N/A
2a	N/A	N/A	o-	N/A	N/A	N/A	N/A
3	**mu-**	**u-m-**	**u-m-/u-mu-**	**mu-**	**u-mu-**	**o-mu-**	**o-mu-**
4	mi-	i-mi-	i-mi-	mi-	**i-mi-**	**e-mi-**	**e-mi-**
5	**-**	**i-**/i-li-	**i-**/i-li-	ri-	**i-**/i-ri-	**e-**/e-li-	e-i-/e-ri-
6	ma-	a-ma-	a-ma-	ma-	a-ma-	a-ma-	a-ma-
7	chi-	i-si-	i-si-	ki-/gi-	i-ki-/i-cy-/i-gi-	e-ki-	e-ki-
8	zvi-	i-zi-	i-zi-	ci-/i-	i-bi-	e-bi-	e-bi-
9	**n-**	**i-**/i-n-	**i-**/i-n-	**n-**	**i-/i-n-/i-nz-**	**e-n-/e-m-**	**e-n-/e-m-**
10	**n-**/dzi-	ii-/i-zin-	i-zi-/i-zin-	**n-**	**i-/i-n-/i-nz-**	**e-n-/e-m-**	**e-n-/e-m-**
9a	N/A	N/A	**i-**	**-**	N/A	N/A	N/A
10a	N/A	N/A	N/A	**-**	N/A	N/A	N/A
6	N/A	N/A	a-ma-	N/A	N/A	N/A	N/A
11	ru-	**u-**/u-lu-	**u-**/u-lu-	ru-	u-ru-	o-lu-	o-ru-
10	**n-**/dzi-	ii-/i-zin-	i-zi-/i-zin-	**n-**	**i-/i-n-/i-nz-**	**e-n-/e-m-**	**e-n-/e-m-**
12	ka-	N/A	N/A	ka-/ga-	a-ka-/a-ga-	a-ka-	a-ka-
13	tu-	N/A	N/A	tu-	u-tu-	o-tu-	o-tu-
14	u-	u-bu-	u-bu-	**-**	u-bu-	o-bu-	o-bu-
6	ma-	N/A	N/A	N/A	N/A	N/A	N/A
15	**ku-**	u-ku-	u-ku-	**ku-/gu-**	u-ku-/u-du-	**o-ku-**	**o-ku-**
6	N/A	N/A	N/A	ma-	a-ma-	a-ma-	a-ma-
16	ha-	N/A	N/A	ha-	a-ha-	a-ha-	a-ha-
17	**ku-**	N/A	ku-	**ku-/gu-**	N/A	**o-ku-**	**o-ku-**
18	**mu-**	N/A	N/A	N/A	N/A	**o-mu-**	**o-mu-**
20	N/A	N/A	N/A	N/A	N/A	o-gu-	o-gu-
21	N/A	N/A	N/A	N/A	N/A	a-ga-	a-ga-
21	zi-	N/A	N/A	N/A	N/A	N/A	N/A
6	ma-	N/A	N/A	N/A	N/A	N/A	N/A
23	N/A	N/A	N/A	N/A	N/A	**e-**	N/A

Table 3: Standard NC classification by singular/plural pair (first line and the [2nd/3rd] line, respectively), highlighting the same prefixes across more than one NC for a language. The dashes between the letters in the prefix illustrate separation between the initial vowel (augment) and prefix; '-': empty prefix; 'N/A': the NC is not present in that language (none use NC19 or NC22).

(whose plural NC is different from the standard pairings). Byamugisha et al., (2016) handled this by placing an 'm' on the NC, while prefix exceptions were directly associated with their plurals.

Therefore, we investigated revising Meinhof's NC definition for other agglutinating Bantu languages within and across Guthrie zones.

3 Reclassification of Meinhof's 1948 Noun Class Specification

We investigated the presence of the limitations in Meinhof's NC definition by identifying whether a deterministic output could be achieved based on the standard NCs, and investigating whether the same limitations for mass nouns and prefix exceptions identified for Runyankore and isiZulu exist in the other five selected languages. We found that, there are also NCs that have more than one possible class prefix; e.g., NC7 has three prefixes in Kinyarwanda: *iki* pluralized with *ibi*, *icy* pluralized with *iby*, and *igi* pluralized with *ibi* and NC5 and NC9 both have *i* as a prefix. This makes the rules on pluralization non-deterministic. Additionally, each of these languages classified some mass nouns in singular NCs, which would result in them being incorrectly subjected to pluralization. We thus found it necessary to refine Meinhof's NC definition in order to ensure a deterministic output during pluralization.

There are several alternatives in which this revision can be done: (1) morphologically (and syntactically for compound nouns), where each different class prefix is reclassified as a subdivision of the general class, with the subdivision indicated with roman numerals, such as, NC5, NC5i, etc.; (2) semantically, which would account for special categories of nouns such as mass nouns, singular-only nouns, and prefix

NC	chiShona	isiXhosa	isiZulu	Kikuyu	Kinyarwanda	Luganda	Runyankore
1	mu-	u-m-	u-mu	mu-	u-mu-	o-mu-	o-mu-
2	va-	a-ba-	a-ba-	a-	a-ba-	a-ba-	a-ba-
1a	-	u-	u-	-	N/A	N/A	N/A
2a	vana-	oo-	o-	-	N/A	N/A	N/A
2b	a-	N/A	N/A	N/A	N/A	N/A	N/A
1i	mw-	N/A	u-m-	mw-	u-mw-	o-mw-	o-mw-
2i	v-	N/A	N/A	N/A	a-b-	N/A	N/A
2	N/A	N/A	a-ba-	N/A	N/A	a-ba-	a-ba-
2a	N/A	N/A	N/A	-	N/A	N/A	N/A
1ii	N/A	N/A	u-m-	m-	-	-	-
2i	N/A	N/A	a-b-	N/A	N/A	N/A	N/A
2ii	N/A	N/A	N/A	N/A	ba-	ba-	ba-
2	N/A	N/A	N/A	a-	N/A	N/A	N/A
3a	N/A	N/A	u-	N/A	N/A	N/A	N/A
2a	N/A	N/A	o-	N/A	N/A	N/A	N/A
3	mu-	u-m-	u-mu-	mu-	u-mu-	o-mu-	o-mu-
4	mi-	i-mi-	i-mi-	mi-	i-mi-	e-mi-	e-mi-
3i	mw-	N/A	u-m-	m-	u-mw-	o-mw-	o-mw-
4	mi-	N/A	i-mi-	mi-	N/A	N/A	N/A
4i	N/A	N/A	N/A	N/A	i-my-	e-my-	e-my-
5	-	i-	i-	ri-	i-	e-	e-i-
6	ma-	a-ma-	a-ma-	ma-	a-ma-	a-ma-	a-ma-
5i	N/A	i-li-	i-li-	i-	i-ri-	e-li-	e-ri-
6	N/A	a-ma-	a-ma-	ma-	a-ma-	a-ma-	a-ma-
7	chi-	i-si-	i-si-	ki-	i-ki-	e-ki-	e-ki-
8	zvi-	i-zi-	i-zi-	i-	i-bi-	e-bi-	e-bi-
7i	N/A	i-s-	i-s-	gi-	i-cy-	e-ky-	e-ky-
8i	N/A	i-z-	i-z-	N/A	N/A	e-by-	e-by-
8	N/A	N/A	N/A	i-	i-bi-	N/A	N/A
7ii	N/A	N/A	N/A	N/A	i-gi-	N/A	N/A
8	N/A	N/A	N/A	N/A	i-bi-	N/A	N/A
9a	N/A	N/A	i-	-	N/A	N/A	N/A
10a	N/A	N/A	N/A	-	N/A	N/A	N/A
6	N/A	N/A	a-ma-	N/A	N/A	N/A	N/A
9	n-	i-	i-	n-	i-	e-n-	e-n-
10	dzi-	ii-	i-zi-	n-	i-	e-n-	e-n-
9i	N/A	i-n-	i-n-	N/A	i-n-	e-m-	e-m-
10i	N/A	i-zin-	i-zin-	N/A	i-n-	e-m-	e-m-
9ii	-	N/A	N/A	N/A	-	-	-
10ii	-	N/A	N/A	N/A	-	-	-
9iii	N/A	N/A	N/A	N/A	i-zn-	N/A	N/A
10iii	N/A	N/A	N/A	N/A	i-zn-	N/A	N/A
11	ru-	u-	u-	ru-	u-ru-	o-lu-	o-ru-
10	N/A	ii-	i-zi-	n-	N/A	e-n-	e-n-
10i	N/A	N/A	N/A	N/A	i-n-	N/A	N/A
6	ma-	N/A	N/A	N/A	N/A	N/A	N/A
11i	N/A	u-lu-	u-lu-	N/A	u-rw-	o-lw-	o-rw-
10	N/A	ii-	i-zi-	N/A	i-	N/A	N/A
12	N/A	N/A	N/A	N/A	N/A	a-ka-	a-ka-
12	ka-	N/A	N/A	ka-	a-ka-	a-ka-	a-ka-
13	tu-	N/A	N/A	tu-	u-tu-	N/A	N/A
14	N/A	N/A	N/A	N/A	N/A	o-bu-	o-bu-
12i	N/A	N/A	N/A	ga-	a-ga-	a-k-	a-k-
13	N/A	N/A	N/A	tu-	N/A	N/A	N/A
13ii	N/A	N/A	N/A	N/A	u-du-	N/A	N/A
14i	N/A	N/A	N/A	N/A	N/A	o-bw-	o-bw-
13	N/A	N/A	N/A	N/A	N/A	o-tu-	o-tu-
13i	N/A	N/A	N/A	N/A	u-tw-	o-tw-	o-tw-
14	u-	u-bu-	u-bu-	-	u-bu-	o-bu-	o-bu-
6	ma-	N/A	N/A	ma-	N/A	N/A	N/A
14i	N/A	N/A	N/A	N/A	u-bw-	o-bw-	o-bw-
15	ku-	u-ku-	u-ku-	ku-	u-ku-	o-ku-	o-ku-
6	N/A	N/A	N/A	ma-	a-ma-	a-ma-	a-ma-
15i	N/A	N/A	u-kw-	gu-	u-kw-	o-kw-	o-kw-
6	N/A	N/A	N/A	ma-	a-ma-	a-ma-	a-ma-
15ii	N/A	N/A	N/A	N/A	u-gu-	N/A	N/A
6	N/A	N/A	N/A	N/A	a-ma-	N/A	N/A
16	ha-	N/A	N/A	ha-	a-ha-	a-ha-	a-ha-
17	ku-	N/A	ku-	ku-/gu-	N/A	o-ku-	o-ku-
18	mu-	N/A	N/A	N/A	N/A	o-mu-	o-mu-
20	N/A	N/A	N/A	N/A	N/A	o-gu-	o-gu-
21	N/A	N/A	N/A	N/A	N/A	a-ga-	a-ga-
20i	N/A	N/A	N/A	N/A	N/A	o-gw-	o-gw-
21	N/A	N/A	N/A	N/A	N/A	a-ga-	a-ga-
21	zi-	N/A	N/A	N/A	N/A	N/A	N/A
6	ma-	N/A	N/A	N/A	N/A	N/A	N/A
23	N/A	N/A	N/A	N/A	N/A	e	N/A

Table 4: Reclassified noun classes. The first line in each pairing is the singular and the other line(s) its plural class (if more than one line is paired, the one with the prefix is applicable, or it is N/A); '-': empty prefix; 'N/A': NC is not present in that language.

exceptions; and (3) revising the entire NC classification to account for all fine-grained details on class prefixes, concords, and agreement.

Options (2) and (3) require extensive linguistic analysis of each language, such as that presented by Taraldsen (2010) for Nguni languages. Such analyses are beyond the scope of our work and left to linguists to complete. For immediate computational needs, we thus applied a morphological with syntactic approach to the reclassification. The details on concordial agreement (subject, object, adjective, and possessive concords) are still maintained in our reclassification, enabling its application in other generation processes (such as verb conjugation) beyond pluralization. Our reclassification ensures that there is always a one-to-one singular-plural mapping, hence, ensuring a deterministic outcome. Table 4 presents the reclassified NC definition; the new classes added are those with one or more 'i' after the number.

This resultant reclassified NC definition demonstrates the pluralization rules that Meinhof's NC definition does not. For example, in Luganda, NC 1ii contains nouns that denote kinship, such as mother (*maama*) and father (*taata*); they are pluralized in NC 2ii, ***ba****maama* and ***ba****taata* respectively.

More generally, this reclassification of the NC definition clarifies the rules for pluralization, resulting in a deterministic output. We used similar prefix features among languages to guide our reclassification, based on the following:

(1) Most NCs whose prefix ends in *u* (NCs 1, 3, 11, 13, 14, 17, and 20) have nouns that have *w* instead; this then becomes the new NC, as is the case with NCs 1i, 3i, 11i, 13i, 14i, 17i, and 20i;
(2) Some NCs whose prefix ends in *i* (NCs 7 and 8) also have nouns whose prefix ends in *y*, and this forms the new NC (7i and 8i);
(3) Nouns without prefixes in the singular and/or plural, which were previously not classified in Meinhof's NC definition, are placed in a new NC with '*ii*' such as NCs 1ii, 9ii, and 10ii; NC 2ii is the pairing for 1ii; and
(4) Languages with NCs with multiple prefixes but that do not fit the above criteria, use the standard prefix in the original class, but the alternative prefix in the new class; such as NCs 7 (*ki-*), 7i (*gi-*), 12 (*ka-*), 12i (*ga-*), 15 (*ku-*), and 15i (*gu-*) in Kikuyu.

4 Implementation of Generic Pluralizer

The decision of how to implement the generic pluralizer was based on research into three approaches for multilingual noun pluralization: Grammatical Framework (GF) (Ranta, 2011), SimpleNLG (Gatt and Reiter, 2009), and the pluralizer in Byamugisha et al., (2016). GF and SimpleNLG support multiple languages from different language families, while the pluralizers by Byamugisha et al., (2016) are very accurate for two agglutinating Bantu languages, isiZulu and Runyankore.

Grammatical Framework (GF) is based on type theory and functional programming; it uses distinct functions for singulars and plurals, and linearization rules to handle the language-specific inflectional forms of nouns during pluralization (Ranta, 2011; Ranta et al., 2015). It requires a large resource grammar. An attempt to develop such a resource grammar for Kiswahili (regarded as a Bantu language, with the same NC system that determines noun pluralization, and not as computationally under-resourced) only introduced some parameter types (Ngángá, 2011), but still has no resource grammar. There is thus no existing GF resource grammar for any Bantu language.

SimpleNLG is an NLG realizer with a Java API, where pluralization is achieved through the language-specific lexicon saved as an XML file (Gatt and Reiter, 2009). The lexicon contains 'words' in their base form, as well as attributes for their 'category' (such as noun, verb, adjective, etc.), 'id' for a unique identifier, and 'plural' (Mohamood, 2016). The plural is obtained by creating an inflected word element around the base form, adding appropriate features to it (such as 'plural' or 'past participle'), and then realizing it (Mahamood, 2015).

Byamugisha et al., (2016) developed two separate pluralizers, for isiZulu and Runyankore. They are based on a combined semantic and syntactic approach with morphology, where the semantics of the noun are determined by its NC, and the appropriate plural prefix replaces the singular prefix. This approach requires that the NC, together with the noun, is passed to the pluralizer in order to obtain the correct

plural. The importance of including the NC with the noun during pluralization can be seen in the increase in accuracy for especially the isiZulu nouns, which improved from about 50% correct pluralization for the noun-only pluralizer to about 85% by passing on the NC with the noun, while it increased it from 88% to 92% for Runyankore (Byamugisha et al., 2016). This effect is due to more or less NCs having the same prefix (recall Table 3).

The GF approach requires the definition of a large resource grammar for each language (Ranta, 2011). The attempt to develop one for Kiswahili showed the difficulties with adopting the GF parameter types to fit the grammar of Bantu languages; for example, the NC was represented as the Gender type (Ngángá, 2011). As no resource grammar was derived from these definitions, there is no way to know whether this is sufficient for pluralization and other generation tasks for Bantu languages. For SimpleNLG, the use of a lexicon for pluralization requires that both the singular and plural forms be stated for each word in the lexicon (Gatt and Reiter, 2009; Mohamood, 2016). We found this approach undesirable, as our aim is to generate the plurals, not manually declare them for each noun.

We therefore decided to develop the generic pluralizer based on the approach by Byamugisha et al., (2016), because the NC reclassification in Table 4 provides one singular prefix with one plural prefix for all these languages, and is thus generalizable to other agglutinating Bantu languages, and the need for the NC-based pluralization of compound nouns in (Byamugisha et al., 2016) also holds for other agglutinating Bantu languages.

Additionally, Byamugisha et al., (2016) identified several nouns that did not conform to the NC rules on pluralization, and referred to them as 'exceptions'. These exceptions were handled in three ways: (1) mass nouns (whose NC was marked with an '**m**'); (2) prefix exceptions, which were placed in a separate look-up file with their correct plural; and (3) singular-only nouns that were also placed in a separate look-up file. These same approaches were applied to the five selected languages, and were able to correctly handle these exceptions.

The design of the generic pluralizer thus comprised the following core rules:

(1) Providing a singular noun and its NC, so as to obtain its plural prefix based on its NC;
(2) Ensuring that mass nouns and singular-only nouns are not pluralized;
(3) Obtaining the plural of the main noun, and its plural agreement, during the pluralization of compound nouns (excluding those with adjectives); and
(4) Correctly pluralizing nouns whose pluralization deviates from the NC pairings.

The language-specific NC details are made available to the pluralizer through text files, one for each language, which are read into the program, and availed to the pluralizer as variables, as illustrated in the architecture in Figure 1. The selected language and noun(s) to be pluralized are entered through the **UI**. Through **Languages**, the text files that contain the reclassified NCs, the singular/plural pairings, as well as prefix exceptions are loaded into **Pluralizer** as variables. The following then takes place: 1) If the noun is a mass noun (with an '**m**' on the NC), then it is returned without pluralization; 2) If the noun is found in **Exceptions**, then it is pluralized according to the exceptions prefix instead of the standard one; and 3) If the noun's NC is the same as that in **NCS** (the noun class system), then it is pluralized accordingly.

The generic pluralizer was implemented as a Java application. It, as well as the datasets tested for all languages are available at `https://github.com/runyankorenlg/Generic-Pluralizer`.

5 Evaluation of Generic Pluralizer

We carried out two types of evaluation for the generic pluralizer: verification and validation. The aim of verification is to ensure that all NCs were accounted for correctly, based on what literature states should be in each NC. The validation was aimed at comparing the output of the generic pluralizer to the language-specific ones.

5.1 Materials and Methods

The evaluation was carried out in two phases that used two test-sets, SetI, for verification (internal) testing, and SetC, for coverage testing. SetI was an English wordlist compiled based on what should

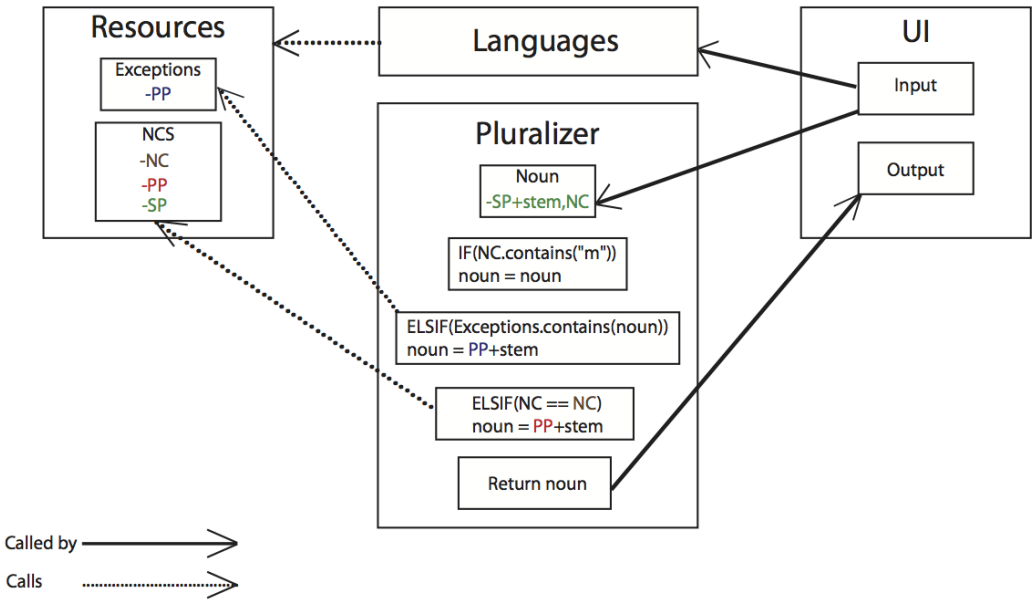


Figure 1: The architecture of the pluralizer; PP: plural prefix; SP': singular prefix; NC: noun class.

generally be in each NC (recall Table 1), and contained 81 nouns such that it includes mass nouns, abstract concepts, and compound nouns. This was done because, for several languages, no computational resources, such as dictionaries, were readily available, from which singular nouns otherwise could have been extracted randomly. The English wordlist was translated, and each noun had its NC; this formed SetI for each language. While not all nouns in the English wordlist were translatable in all languages, all NCs were accounted for in the resulting wordlists.

SetC was the same English wordlist used as Set1R in Byamugisha et al., (2016), and consists of 101 words informed by multiple ontologies. This was also translated by native speakers to chiShona, isiXhosa, Kikuyu, and Luganda, with each noun having its NC. The original Runyankore and isiZulu test-sets from (Byamugisha et al., 2016) were used, but their NCs were updated according to the new NC specification in Table 4.

The metric for evaluation is accuracy, determined as the percentage correct plurals over the test set.

5.2 Results and Analysis

Table 5 shows the accuracy of the generic pluralizer for each language, for SetI and SetC. The accuracy is based on the total number of translatable nouns. The reason for obtaining less than 100% is due to the absence of phonological conditioning. In chiShona for example, ***g**umbo* 'leg' is pluralized as *ma**k**umbo* instead of the generated *ma**g**umbo*. As explained in Section 2, the rules for phonological conditioning are language-specific, and as such could not be generalized. We propose that a separate language-specific phonological conditioning module be developed for each language, and added to that language's resources folder; we leave this for future work. Further, for isiZulu, the lower accuracy in the generic pluralizer is due to it only handling compound nouns with the possessive concord, while the isiZulu pluralizer in Byamugisha et al., (2016) additionally handles compound nouns with adjectives using the adjective concord.

Prefix Exceptions The true exceptions identified in Runyankore and isiZulu are those where pluralization does not correspond to the singular/plural pairings and there are no rules among them. They were thus placed in a separate look-up file with their corresponding plurals. In the generic pluralizer, a separate exceptions look-up file of the same structure was also created for each language (see Figure 1). Prefix exceptions were identified to be quite rare in each language; examples include: the plural of ***i**mbwa* (dog)

	SetI		SetC	
Language	**Translatable Nouns**	**Accuracy (%)**	**Translatable Nouns**	**Accuracy (%)**
chiShona	68	94.12%	74	94.59%
isiXhosa	74	97.30%	83	100%
isiZulu	71	100%	101	97.03%
Kikuyu	77	96.10%	76	94.74%
Kinyarwanda	70	97.14%	-	-
Luganda	75	93.33%	78	97.44%
Runyankore	81	93.83%	88	96.59%

Table 5: Accuracy of the generic pluralizer.

in chiShona as ***i****mbwa* instead of ***dzi****mbwa*; or *dagetari* (doctor) in Kikuyu, pluralized as ***ma****dagetari*, instead of *dagetari*; or ***i****nzu* (house) in Kinyarwanda, whose plural is ***ama****nzu* instead of ***i****nzu*.

Singular-only Nouns Singular-only nouns in isiZulu and Runyankore resulted from nouns for abstract concepts, such as 'greed' and 'thirst', found in several NCs, or infinitive nouns in NCs 15 and 15i. Abstract nouns belonging to plural NCs (for example, ***obu****rungi* 'beauty', Runyankore) were handled in the pluralization algorithm by returning them as they are. Those belonging to singular NCs were placed in a separate look-up file. Similarly, for the five selected languages, abstract concepts can either be found in singular or plural NCs; examples of the singular-only nouns identified in the wordlists are: *moto* 'fire' and *zuva* 'sun' (chiShona), *ubuhle* 'beauty' and *ubuhlobo* 'friendship' in isiXhosa, and *omululu* 'greed' and *enyoonta* 'thirst' in Luganda. The same algorithmic solution was used for those in plural NCs, while a look-up file was used for those in singular NCs. Both solutions were found to be sufficient to handle this class of exceptions.

Mass Nouns Mass nouns that belong to singular NCs are marked with an 'm', so that they can be identified and not be pluralized according to the singular/plural pairings (Byamugisha et al., 2016). This same rule was found to adequately cater for mass nouns that belonged to singular NCs; for example, *doro* (NC5, chiShona) and *umdiliya-omfaxangiweyo* (NC3, isiXhosa) 'wine', and *iria* 'milk' (NC5, Kikuyu). Mass nouns that belong to plural NCs, such as *mae* (NC6, Kikuyu), *mvura* (NC10ii, chiShona), and *amanzi* (NC6, isiXhosa) 'water' are handled algorithmically, as described in the previous section.

Compound Nouns From the translated wordlists, we identified that the main noun of a compound noun is placed before the modifier noun. When pluralizing compound nouns in isiZulu and Runyankore, the main noun was first pluralized, and the plural agreement for the modifier noun was obtained and used to associate with the modifier noun, as in (Byamugisha et al., 2016).

We further found that, similar to isiZulu and Runyankore, phonological conditioning was also required in chiShona, isiXhosa, Kinyarwanda, and Luganda; only Kikuyu does not require phonological conditioning. However, as explained in Section 2, the rules for phonological conditioning are language-specific. We omit its detail here due to space limitations.

6 Discussion

Our investigation into a generic pluralizer for agglutinating Bantu languages has shown that the approach taken by Byamugisha et al., (2016) was applicable to other Bantu languages not only within but also across Guthrie zones. For both the intra-zone and inter-zone investigations, the same approach was found to result in 100% accuracy in some datasets (recall Table 5). Moreover, the investigation into the generalizability offered a systematic way to revise the (outdated) Meinhof NC system into one that is more precise and usable for computational purposes. It clarifies the singular/plural pair rules for pluralization, resulting in a deterministic output with a higher accuracy.

Although the language-specific pluralizers achieved 100% on SetC in Byamugisha et al., (2016), the errors in the generic pluralizer were mainly found to originate from the lack of phonological conditioning. Generally, phonological conditioning is language-specific, but there may be some cases for

generalizability. For example, isiXhosa and isiZulu both have $a + i = e$ and Kinyarwanda, Luganda, and Runyankore, all have $a + vowel = ' '$. However, further research is required into under what circumstances such rules are true, in order to prevent overgeneralization. Further, we intend to extend the generic pluralizer to compound nouns with adjectives using the NC-based adjective concord, as was done in the isiZulu pluralizer (Byamugisha et al., 2016).

The processing of the resources required to add a new language to the pluralizer is quite considerable, with text files required for the noun class system (NC number, class prefixes, and genitive), NC pairings, as well as singular-only nouns and prefix exceptions if known. While the last two, singular-only nouns and prefix exceptions, can be added as errors are identified, the NC system and pairings are required for the pluralization to be done. It may be useful to use the revised NC table to speed up automated annotation of nouns, which then can facilitate resource development for pluralization. This amount of effort is still considerably less than that required by other multilingual noun pluralization implementations, notably, GF (Ranta et al., 2015) and SimpleNLG (Gatt and Reiter, 2009).

The criteria for the revised noun class table, as described in Section 3, can be used by other researchers in computational linguistics for other Bantu languages, who will undoubtably need to revise Meinhof's NC definition, especially when working with multiple languages.

7 Conclusion

We have presented a novel generic pluralizer to pluralize nouns of agglutinating Bantu languages. This pluralizer was based on a combined morphology with syntactic and semantic approach and handling of regular exceptions. The rules for pluralization are based on the classification of nouns into noun classes (NCs). To ensure deterministic outputs, i.e., one plural option for each singular noun, the standard noun class system of Meinhof (1948) had to be redefined from 29 noun classes into 53 noun classes. The generic pluralizer can pluralize simple nouns and exceptions resulting from mass nouns, singular-only nouns, prefix exceptions, and compound nouns. It achieved over 93% for all test-sets, with the remaining errors arising from the need for phonological conditioning and handling compound nouns with adjectives. Future work will center around adding support for adjectives and investigating how to implement phonological conditioning.

Acknowledgements We would like to thank Christine Wangiru Mburu, Goreth Byamugisha, Juliet Naggawa, Naissa Umutoni, Ngoni Choga, and Zola Mahlaza for providing the Kikuyu, Luganda, Kinyarwanda, chiShona, and isiXhosa translations.

This work is based on the research supported by the Hasso Plattner Institute (HPI) Research School in CS4A at UCT and the National Research Foundation of South Africa (Grant Number 93397).

References

Elizabeth Baertlein and Martin Ssekitto. 2014. Luganda nouns inflectional morphology and tests. *Linguistic Portfolios*, 3.

Joan Byamugisha, C. Maria Keet, and Langa Khumalo. 2016. Pluralizing nouns in isizulu and related languages. In *17th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2016)*, volume 9626, Konya, Turkey. Springer LNCS.

Catherine Chavula and C. Maria Keet. 2014. Is lemon sufficient for building multilingual ontologies for bantu languages? In *11th OWL Experiences and Directions Workshop (OWLED'14)*, volume 1265, pages 61–72, Riva del Garda, Italy.

D. M. Conway. 1998. An algorithmic approach to English pluralization. In C. Salzenberg, editor, *Proceedings of the Second Annual Perl Conference*. O'Reilly. San Jose, USA, 17-20 August, 1998.

Rose-Marie Déchaine, Raphaél Girard, Calisto Mudzingwa, and Martina Wiltschko. 2013. The internal syntax of shona class prefixes. *Language Sciences*, 43:18–46.

Franca Ferrari-Bridgers. 2009. Luganda verb morphology: A new analysis of the suffixes '-ye' and '-a', and their distribution across the indicative, subjunctive, and imperative moods. *Studies in African Linguistics*, 38(1).

Albert Gatt and Ehud Reiter. 2009. Simplenlg: A realization engine for practical applications. In *12th European Workshop on Natural Language Generation (ENLG'09)*, pages 90–93, Athens, Greece.

Malcolm Guthrie. 1948. *The Classification of the Bantu Languages*. Oxford University Press, London.

Lisa Jeon, Jessica Li, Samantha Mauney, Anaí Navarro, and Jonas Wittke. 2015. A basic sketch grammar of gíkúyú. *Rice Working Papers in Linguistics*, 6.

C. Maria Keet and Langa Khumalo. 2014. Towards verbalizing ontologies in isizulu. In *4th Workshop on Controlled Natural Languages (CNL'14)*, pages 78–89, Galway, Ireland.

Alex Kimenyi. 2004. Kinyarwanda morphology. In Geert Booij, Christian Lehmann, Joachim Mudgan, and Stavros Skopeteas, editors, *Morphology: An International Handbook for Inflection and Word Formation*, volume 17.2. De Gruyter.

Saad Mahamood. 2015. Simple nlg wiki: Section iv-lexicon. Accessed on 08/02/2018.

Jouni Filip Maho. 2009. Nugl online: The online version of the updated guthrie list, a referential classification of the bantu languages. http://goto.glocalnet.net/mahopapers/nuglonline.pdf.

Nhira Edgar Mberi. 2002. *The Categorical Status and Functions of Auxiliaries in Shona*. Ph.D. thesis, University of Zimbabwe.

Saad Mohamood. 2016. default-lexicon.xml. Accessed on 08/02/2018.

Linkie Mohlala. 2003. The bantu attribute noun class prefixes and their suffixal counterparts, with special reference to zulu. Master's thesis, University of Pretoria, Pretoria, South Africa.

Jackson Muhirwe. 2007. Computational analysis of kinyarwanda morphology: The morphological alternations. *International Journal of Computing and ICT Research*, 1(1):85–92.

Ramin Charles Nakisa and Ulrike Hahn. 1996. Where defaults don't help: the case of the German plural system. In *Proceedings of the 18th Annual Conference of the Cognitive Science Society*, pages 177–182. San Diego, USA, July 12-15, 1996.

Wanjiku and Ngángá. 2011. Swahili inflectional morphology for the grammatical framework. In *Human Language Technology for Development*, Alexandria, Egypt.

Derek Nurse and Gerard Philippson. 2003. *The Bantu Languages: Routledge Language Family Series 4*. Taylor and Francis Routledge, London.

Aarne Ranta, Yan Tian, and Haiyan Qiao. 2015. Chinese in the grammatical framework: Grammar, translation, and other applications. In *8th SIGHAN Workshop on Chinese Language Processing (SIGHAN'08)*, pages 100–109, Beijing, China.

Aarne Ranta. 2011. *Grammatical Framework: Programming with Multilingual Grammars*. Center for the Study of Language and Information (CSLI) Publications, Stanford.

Lenhart Schubert. 2015. Computational linguistics. In N. Edward Zalta, editor, *The Stanford Encyclopedia of Philosophy*. The Metaphysics Research Lab, Center for the Study of Language and Information, Stanford University, Stanford, California, USA. Available from: https://plato.stanford.edu/archives/spr2015/entries/computational-linguistics/.

Knut Tarald Taraldsen. 2010. The nanosyntax of nguni noun class prefixes and concords. *Lingua*, 120(6):1522–1548.

Jason Zentz. 2016. *Forming Wh-Questions in Shona: A Comparative Bantu Perspective*. Ph.D. thesis, Yale University.

Automatically Extracting Qualia Relations for the Rich Event Ontology

Ghazaleh Kazeminejad[1], Claire Bonial[2], Susan Windisch Brown[1] and **Martha Palmer[1]**
[1]University of Colorado Boulder, [2]U.S. Army Research Lab
{ghazaleh.kazeminejad, susan.brown, martha.palmer}@colorado.edu
claire.n.bonial.civ@mail.mil

Abstract

Commonsense, real-world knowledge about the events that entities or "things in the world" are typically involved in, as well as part-whole relationships, is valuable for allowing computational systems to draw everyday inferences about the world. Here, we focus on automatically extracting information about (1) the events that typically bring about certain entities (origins), (2) the events that are the typical functions of entities, and (3) part-whole relationships in entities. These correspond to the agentive, telic and constitutive qualia central to the Generative Lexicon. We describe our motivations and methods for extracting these qualia relations from the Suggested Upper Merged Ontology (SUMO) and show that human annotators overwhelmingly find the information extracted to be reasonable. Because ontologies provide a way of structuring this information and making it accessible to agents and computational systems generally, efforts are underway to incorporate the extracted information to an ontology hub of Natural Language Processing semantic role labeling resources, the Rich Event Ontology.

1 Introduction

Most adults could tell you that an umbrella is used to keep the rain off of you, that bread is baked, or that dandelions are made up of leaves, stems, and flowers. Commonsense, real-world knowledge about the events that entities or "things in the world" are typically involved in, as well as part-whole relationships, is valuable for allowing computational systems to draw everyday inferences about the world. Here, we describe our approach for automatically extracting this information from the Suggested Upper Merged Ontology (SUMO) (Niles and Pease, 2001a; Pease, 2011). We assume the theoretical framework of the Generative Lexicon (GL) (Pustejovsky, 1991), as we find GL qualia relations useful for considering the typical involvement of a particular entity in events. Thus, we focus on relationships between entities and events captured in a set of qualia associated with an entity. We find that human annotators asked to evaluate the quality and accuracy of the information extracted overwhelmingly find the origins, functions and part-whole relationships proposed to be reasonable.

One of our motivations is to add the extracted qualia relations to an ontology hub of Natural Language Processing (NLP) semantic role labeling (SRL) resources, the Rich Event Ontology (REO). We plan to exploit the qualia relations within REO to make generalizations across participant types for a given event type, thereby facilitating the addition of selectional restrictions, and to enable some shallow inferencing about how entities generally come into existence, how they are used, and what they are made up of. By structuring this information within REO we aim, for example, to allow an agent or computational system to infer that a room (whether detected visually or described in text) with a stove, pans, and a refrigerator is likely a kitchen where food preparation happens. Our approach first requires expansion of REO to include additional "things," or ENDURANTS as they are called in REO, since past efforts have focused on the inclusion of events or PERDURANTS. We again draw information from SUMO to quickly and efficiently add high-quality information on ENDURANTS to REO, and discuss the linking model we have developed to integrate the new classes from SUMO.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2644–2652
Santa Fe, New Mexico, USA, August 20-26, 2018.

2 Theoretical Framework: Generative Lexicon

The theory of GL focuses on the distributed nature of compositionality in natural language (Pustejovsky, 1991), which attributes parts of the semantic load to non-verb constituents, as opposed to approaches where verbs are at the center of the notion of compositionality, (e.g. Levin (1993)). According to GL theory, every entity has a set of qualia that plays a role in what events that entity can be involved in, and that directly affects the compositional meaning that arises from the mention of a particular entity being involved in a particular event. The set of qualia GL proposes include formal, telic, agentive, and constitutive. The formal quale represents the IS-A relation for a given entity. For example, the formal quale for *dandelion* might be *plant*. The constitutive quale is the set of other entities that are typically considered to be parts of the given entity or the material of the entity. For example, the constitutive quale for *plant* could be *leaf*, *stem*, etc., and the constitutive quale for *pottery* could be *clay.* The telic quale represents the typical purpose or function of the given entity, if there is one (e.g., the telic quale of *umbrella* might be *protect from rain*). The agentive quale represents the factors involved in how the given entity came into being (e.g., the agentive quale of *bread* could be *baking*).

Note that there is no single set of defined quale for the entities of the world, hence there is indeterminacy in how these should be labeled and at what level of granularity (i.e., is the agentive quale of *bread* better captured as *baking* or as *applying heat*?). It is in part for this reason that situating qualia relations in an ontology, as was done with SIMPLE (Lenci et al., 2000), is so appealing. An ontology provides a common vocabulary for entities and events and structures them in a hierarchy from most general to most specific.

In an ontology with GL qualia, any node would inherit all the qualia from the set of its ancestors. The deeper we go into the lower levels of the hierarchy, the more specific the qualia get, and, in some cases, they may override parts of the qualia they have inherited. For instance, any physical object has a physical form (formal quale), so any `Artifact` inherits this property because it is a child to the physical object node in the ontology. In addition to this inherited quale, `Artifact` introduces to the ontology its own set of qualia: an agentive quale of *making* and a very general telic quale of *some event,* since every artifact is 'made' and typically has 'some purpose'.

3 Resource Background

3.1 SUMO (Suggested Upper Merged Ontology)

SUMO was originally an upper level ontology (Niles and Pease, 2001a), but now includes mid-level and domain ontologies as well (Pease, 2002). These domain-specific ontologies inherit the broad conceptual distinctions of SUMO, and specify the concepts and axiomatic content of a particular domain. Niles and Pease (2001b) explain SUMO's most general concepts and the relations between them, such as the distinction between `Object` and `Process`. Pease et al. (2002) illustrates the structure of SUMO and explains that the existence of SUMO-WordNet mapping serves as a gauge of the coverage of the ontology. Niles and Pease (2003) explain a SUMO-Wordnet mapping methodology and the specifics of the language used to show the synonymy, hypernymy, and instantiation relations.

In addition to the relations between entities that ontologies typically contain, SUMO provides relations between entities and processes, and statements in higher order logic (Benzmüller and Pease, 2012) that attempt to fully define each concept. For instance, the relation *result* is a binary relation with two arguments, a `Process` and an `Entity`, where the `Entity` is the output or product of the `Process`. So both temporally and causally, the `Process` is antecedent to the `Entity`. This relation is one of a few that help us find the agentive quale. Another relation between a `Process` and an `Entity` is *instrument*, where the `Entity` is used by an agent in bringing about a `Process`. So, in this case, the `Process` is consequent to the `Entity`. We use this relation to help find the telic quale for entities.

Relations are nodes under the `Abstract` hierarchy of SUMO, but many relations have sub-relations as well as instances. In general, since relations are like functions, each relation has a domain, and SUMO provides the set of arguments each relation can take. In some cases, the argument of a relation could be a whole formula by itself, so it does not point to a certain set. Rather, it is usually a complex axiom in

itself, containing at least one `Entity`, at least one relation, and possibly some `Processes`. One of the main pointers we use to find telic quale is *hasPurpose*, which usually has such complex arguments. These arguments are themselves logical formulae, supported in the higher order logic employed for SUMO. For instance, SUMO stipulates the purpose of a Microwave using the *hasPurpose* relation, with a formula that can translate to: there is an entity `PreparedFood` and a process `Heating`, and the `Microwave` is an instrument for `Heating`, through which the `PreparedFood` gets heated.

3.2 REO (Rich Event Ontology)

The Rich Event Ontology (REO) unifies existing SRL schemas by providing an independent conceptual backbone through which they can be associated, and it augments the schemas with event-to-event causal and temporal relations. The ontology was developed in response to unsuccessful efforts to map directly between FrameNet (Fillmore et al., 2002) and the small set of disparate event types in Rich Entities, Relations and Events (ERE) (Song et al., 2015) (originally based on Automatic Content Extraction (ACE) (Doddington et al., 2004)). The difficulty was mainly due to differences in the granularity of events described by the FrameNet frames and ERE event types and inconsistencies in how the resources divided the semantic space.

FrameNet, ERE, and VerbNet (Schuler, 2005) have wide-coverage lexicons having to do with events, and they contribute annotated corpora and additional semantic and syntactic information that can be crucial to identifying events and their participants. REO serves as a shared hub for the disparate annotation schemas and therefore enables the combination of SRL training data into a larger, more diverse corpus, as well as expanding the set of lexical items associated with each ERE event type. By adding temporal and causal relational information, the ontology also facilitates reasoning on and across documents, revealing relationships between events that come together in temporal and causal chains to build more complex scenarios.

The structure of REO, illustrated in Figure 1, consists of a main "reference" ontology of generic event types and individual OWL "resource" ontologies. The relationships between the generic event types in the reference ontology and the event designations made in a particular annotated data resource are spelled out in various "linking models" that import both the reference ontology and a resource ontology. In the example shown, REO `Discharge` events map to the Releasing frame in FrameNet and the Release-Parole event type in ERE. However, other mappings between ERE and FN are necessarily more indirect. With respect to the Communication node in REO, ERE/ACE only maps to instrument_communication and statement while FN has mappings to nearly all daughters, as does VN, and VN maps to the mother Communication node as well. Additional indirect mappings are detailed in Brown et al. (2017). Individual words within the resource classes can be detected in text to find a wide variety of each event type, or one can query to view its participants and its relations to other events that is independent of the various lexical resource schemas.

4 Related Work

In considering how to best capture the type of information desired for relating ENDURANTS to PERDURANTS, we have examined the approaches of other benchmark ontologies. We find that the Basic Formal Ontology (BFO) (Smith et al., 2014) and the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) (Masolo et al., 2003) largely focus on parthood relations, qualities, and general notions of participation of an ENDURANT (or "continuant" as labeled in BFO) in a PERDURANT (or "occurrent" in BFO). The parthood relations have some overlapping information with the GL constitutive quale. Also of relevance to our work, BFO includes a class of what it calls "specifically dependent continuants," which includes functions, such as the function of a hammer to drive in nails.

The Event and Implied Situation Ontology (ESO) focuses on pre and post situations with respect to some event type, making explicit how a participant's pre and post states change in an event. Although there may be some amount of overlap in what ESO includes and the information captured in the GL agentive quale (e.g., the agentive quale for bread is baking), ESO's information will likely be much broader than the GL agentive, which essentially is limited to creation events. Although we hope to

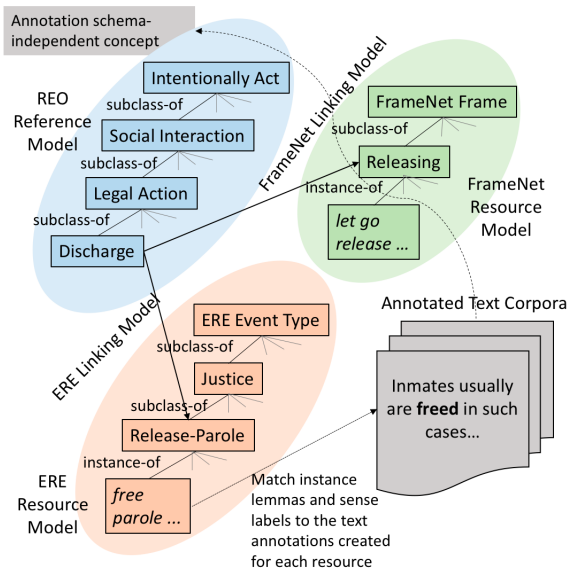


Figure 1: OWL resource models are linked to REO reference model in individual linking models (only ERE and FrameNet shown, but ACE and VerbNet linking models exist as well).

integrate or map the added and overlapping information from ESO, BFO and DOLCE in the future, here we focus on extracting information from SUMO.

5 Automatic Extraction of Qualia Relation Information

5.1 Methodology

In order to find qualia relations for entities in REO, we looked for ways to automatically extract them from SUMO. By examining around a hundred nodes in SUMO, a number of relations were found to be useful in extracting qualia. For instance, the relation *hasPurpose* in SUMO directly specifies the purpose (hence the telic quale) of a given entity. If a given entity is the second argument of the relation *instrument*, then it is a tool used to bring about an event which is the first argument of the relation *instrument* in the same axiom (e.g., (instrument ?TRANSFER ?ARTERY), where ARTERY is a tool enabling a TRANSFER event). Therefore, the first argument could be extracted as the telic quale of this entity. In the same way, a number of relations, including *part* and all its sub-relations, were found to lead us to the constitutive quale. For instance, we find that `Syllable` is a part of a `Word`, through an axiom roughly like (*part* `Syllable Word`). Though the sub-relations of *part* have fine-grained semantic differences, they all have the same weight here for our purposes. These include relations such as *initialPart, initiallyContainsPart, partTypes, typicalPart,* etc. The SUMO relation *result* leads us to the agentive quale, as its second argument is the output (or the product) of its first argument (e.g., (result ?WRITE ?TEXT) where TEXT is the output of WRITE event).

In addition to using SUMO relations, we used each node's documentation in SUMO to automatically extract telic quale using fairly straightforward regular expressions, since a number of entities were found in our initial examination to have their purpose described in their documentation, but not in their axioms. For instance, we extracted parts of the documentation that occur after terms such as 'purpose is,' 'intended to,' 'designed for,' etc. as potential candidates for the telic quale.

As the formal quale is basically an IS-A relationship, we extracted the parent of each node as its formal quale. Of course the inheritance is always at work for all the four types of qualia relations. For instance, an `Adverb` is a `Word` (parent), but is also a `LinguisticExpression` (grandparent).

The output of the function that automatically extracts qualia[1] currently can have two forms. If they are extracted from SUMO axioms, we have a set of functions to extract the actual SUMO node that

[1] `https://github.com/ghamzak/SUMOQualiaExtraction`

represents the desired quale. The second format of the output occurs when we extract qualia from the documentation. In this case, what we extract is actually plain English – a part of a sentence. In writing functions to extract sensible parts of sentences, we decided to sacrifice recall in favor of precision. At the end, we managed to extract 112 agentive, 762 telic, and 481 constitutive qualia relations. We also have a separate function that takes a SUMO entity as input and returns all possible qualia found in SUMO for that entity. For instance, `Building` has 13 total qualia found in SUMO, including 13 constitutive, 1 telic, and 1 agentive qualia relations. Table 1 illustrates the results of automatic extraction.

Agentive	112
Telic	762
Constitutive	481

Table 1: Number of Qualia found in SUMO using automatic extraction

5.2 Evaluation

In order to evaluate the automatically extracted qualia relations from SUMO, we designed an on-line annotation system, where we asked the annotators to decide whether the automatically extracted quale relation was reasonable or unreasonable, or indicate if they're not sure. We also had a comment box for each entry, and asked the annotators to provide comments. For instance, for the telic quale, our instruction for commenting was as follows:
If you feel the given function is reasonable but it's not at all how you would phrase or describe the function, then please provide your own description/phrasing of the function in the comment box.
You can also use to comment box to suggest a function for the unreasonable cases.
If you're unsure, comment on what makes you unsure.
In collecting such comments in addition to the reasonable judgments, we hope to gain insights into better alternatives for the quale relation.

We also provided SUMO documentation for the entity in question for cases where the annotator might not be familiar with the entity. The need for this was revealed in our pilot testing of the annotation system, where one of the annotators had not heard of some entities such as `Lanai`, which according to SUMO "refers to a roofed outdoor area Adjacent to a Building often furnished and used as a living room." So we decided to include SUMO documentation for each entity just in case the annotators are not familiar with it. Figure 2 shows a sample entry in our evaluation system.

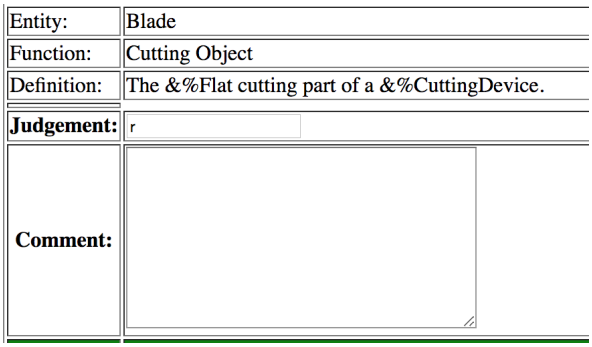


Figure 2: A sample entry in the evaluation system, asking about the telic quale.

To ensure that the annotators are not judging haphazardly and without thinking, we inserted 3 random attention tests in each page of annotation. The attention tests were completely unreasonable possibilities, in which the extracted function for one entity was paired with a totally different entity (e.g., Entity: AerobicExerciseDevice, Telic: to attach one thing to something else). So in each page, we had 25 real tasks and 3 attention tests. Examining the accuracy of each annotator, we had to throw out the data from

one of them with only 73% accuracy on the attention tasks; those tasks were re-annotated by another annotator.

It would be prohibitively difficult to have a measure of recall for this task of automatic extraction, so we limit our evaluation to the precision of the results, using human annotators' judgments as the gold standard. Table 2 shows the precision measures found. The average interannotator agreement was 84.13%.

	Agentive	Telic	Constitutive
Reasonable	93%	88.26	85.89%
Unreasonable	2%	7.87%	10.74%
Unsure	5%	3.87%	3.37%

Table 2: Precision of Qualia Relations Extracted from SUMO

5.3 Error Analysis

The 'unreasonable' judgment for the constitutive relation was mostly applied to the ones taken from SUMO axioms, where the constitutive relation was too general for our purposes. For instance, the extraction found `Object` as a component part of `WireCoil`, and `Physical` as what constitutes `Solenoid`. Despite being true, they're too general to be accepted by a human as a reasonable part-whole relation. Another reason for marking the extracted constitutive relation as unreasonable was the jargon used in particular professions with which an annotators was not familiar, such as biology or chemistry. For instance, an `AtomicGroup` is part of a `Molecule`, but it's been marked as unreasonable with the comment: "part whole switch." Some other errors were due to an ordering mistake in SUMO axioms, such as (part Penne Hole) (which means Penne is part of Hole), whereas the reverse is true. Still others were due to a bug in the extraction, which ignored negation in axioms when finding constitutive relations, such as `BloodTypeB` which does NOT contain `AntigenA` according to SUMO axioms. We extracted it as a part by ignoring the negation. Thus, the results of the annotation were illuminating: helping to pinpoint where SUMO is too general for our purposes, or where our extraction script needs refinement.

The unreasonable judgments for the telic quale were mostly due to not yet capturing and combining inherited relations from SUMO. At any particular node, SUMO underspecifies the definition because it assumes inheritance from ancestor nodes. Although we assume this as well, the qualia relations we have extracted are limited to the ones found with direct mentioning of that entity. For instance, for the entity `MilitaryVehicle`, the telic role extracted was "MilitaryOrganization uses it," which was marked as unreasonable with the suggested alternative "Provide transportation for any military organization." However, `MilitaryVehicle` has `Vehicle` as its parent and inherits from it. Therefore, the telic quale for `Vehicle`, which is extracted as "Translocation," would be inherited by `MilitaryVehicle`.

This sort of error confirms our plan to inherit qualia relations and add the quale for each entity to all its children. Not only will these types of errors be eliminated, the number of entities with qualia relations will increase significantly. Currently, many lower level entities have no specific qualia to be extracted at their SUMO node but have very informative quale that could be inherited from parent nodes. Thus, we would have a significant increase in coverage (recall), while precision is guaranteed to remain high.

Yet other instances of unreasonable telic quale were due to the wording used in SUMO. For instance, the following pairs have been judged as unreasonable.
Entity: HearingProtection – Function: protect Human from Injuring caused by RadiatingSound
Entity: PerformanceStage – Function: location of Demonstrating
Entity: Campground – Function: to have MobileResidences
These may not be how a human would describe the functions, but SUMO has tried to maximize the grounding of its definitions by using its other defined entities within them, leading to a more interconnected network of concepts. `Demonstrating`, for instance, is not the word people use to talk about the function of a performance stage, but according to the documentation of `Demonstrating` in SUMO, it would cover 'software demos, theatrical plays, lectures, dance and music recitals, museum exhibitions,

etc.' Given the connection of REO concepts to SUMO, we need not be overly concerned about these types of seemingly inaccurate results.

6 Incorporating Qualia Relations into REO

6.1 Expanding REO Endurants

As an ontology meant to provide a shared vocabulary of SRL annotation resources, the development of REO has focused on how to structure ontological relations between the events and states included in these resources (ACE, ERE, VerbNet, FrameNet). Although our intention has never been to create a detailed ontology of the entities that serve as the participants in these events/states, such information is needed to generalize and map this information across resources, as well as provide some insights into selectional restrictions generally and qualia relations specifically. Thus, we opted to integrate REO with an existing ontology containing the type of information on participants that we were interested in.

Given the detailed information on objects and accompanying information such as *hasPurpose* in SUMO, we decided that it was best suited to our needs. REO aligns with early distinctions made by DOLCE, distinguishing between ENDURANT Entities and PERDURANT Entities, where ENDURANTS are defined as "Those entities that can be observed/perceived as a complete concept, no matter which given snapshot of time," whereas PERDURANTS are defined as "Those entities for which only a part exists if we look at them at any given snapshot in time. Variously called events, processes, phenomena, or activities and states, PERDURANTS have temporal parts or spatial parts and participants." PERDURANTS largely map to what SUMO deems `Processes`, and although many ENDURANTS map to what SUMO deems `Objects`, there were also a variety of useful concepts in SUMO that seemed to fit the definition of an ENDURANT. These include: `ContentBearingPhysical`, `FinancialAsset`, `Graph`, `GraphElement`, `Model`, `PhysicalSystem`, `Quantity`, and `SetOrClass`. To flesh out the ENDURANT portion of REO, we opted to integrate all daughters of `Object` and the classes just mentioned as daughters of our ENDURANT class. Specifically, this was done in a linking model, similar to the linking models that serve to map the generic event concepts in REO's 'Reference Model' to particular groups of events in the SRL resource models (see Brown et al. (2017) for more details on the structure of REO). The linking model imports an abbreviated OWL version of SUMO (containing only the ENDURANT-compatible classes mentioned, their documentation that provides a description of what they are, and their associated WordNet sense keys), and the REO OWL reference model. The resulting model has all of the REO event ontology nested under PERDURANT, and the extracted SUMO content nested under ENDURANT (see Figure 3).

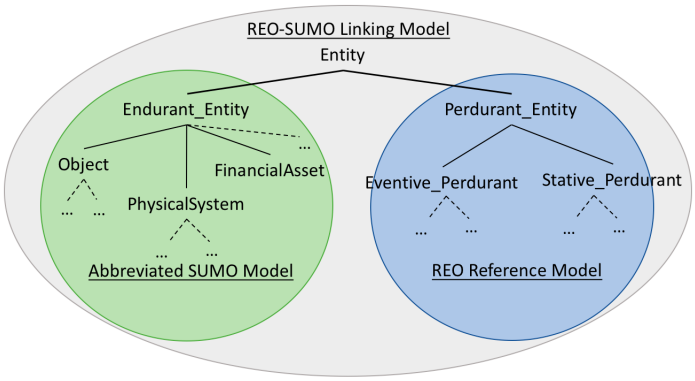


Figure 3: REO-SUMO Linking Model: This model imports the abbreviated SUMO model (containing ENDURANT-type classes only) and the REO reference model. Thus, REO specifies the PERDURANTS of the model, while SUMO specifies the ENDURANTS.

Admittedly, there are a variety of weaknesses to this model. A model that used all of SUMO would contain the many axioms found in the full version of SUMO, which OWL is not expressive enough

to handle (mathematically a description logic cannot state the content of first order logic, much less in this case, higher order logic). These axioms facilitate some of the rich reasoning potential of SUMO, which is lost in our more superficial model. We are therefore considering new ways in which to integrate the lexical resource information included in REO into the full SUMO model, enabling users to take advantage of SUMO with respect to reasoning about events included in the SRL annotations in the future.

6.2 Initial Integration of Qualia Relations

As a preliminary step in incorporating the extracted qualia relations into our ontology, we have simply added the information as annotations on the ENDURANTS drawn from the abbreviated SUMO model. We have introduced the following annotation properties: has_Quale_Agentive, has_Quale_Constitutive, and has_Quale_Telic. We do not include an explicit relation for the Formal quale, as we assume this information is encoded in the ontological structure itself. A variety of examples of these qualia incorporated are given in Table 1.

Endurant	**has_Quale**...	**Concept**
Brake	Telic	Decelerating
Building	Constitutive	Wall
Organization	Agentive	Founding

Table 3: Examples of has_Quale annotations of Endurant entities.

More general parent classes lack cohesive qualia. In fact, almost none of the nine direct daughters of ENDURANT (e.g., `Object, PhysicalSystem, Model,` etc.) have qualia annotations, but the daughters of these classes (and subclasses therein) are increasingly populated with qualia annotations, reflecting their increasing specificity.

In future work, we will incorporate the extracted qualia relation information (along with the comments acquired through annotators applied to them) into the REO/SUMO Linking Model using object properties (as opposed to the current annotation properties). These object properties will relate SUMO ENDURANTS to either REO PERDURANTS or other SUMO ENDURANTS, depending upon the relation, such that the ontology can be exploited for telic, agentive, formal, and constitutive qualia, as well as the events in which an endurant is typically involved. For example: `Meal` (ENDURANT extracted from SUMO) has_Quale_Telic CONSUMPTION (PERDURANT from REO). Unlike the annotation properties currently implemented, the object properties will better facilitate querying and reasoning using the ontology, and will effectively map the concepts extracted from SUMO (e.g., an extracted telic quale for `meal` is 'contains nutrients for humans') to the event classes in REO (labeled CONSUMPTION). We plan to do this mapping manually using a group of expert annotators who are familiar with the ontology and the linked lexical resources.

In addition, we plan to augment ongoing research evaluating the utility of REO in scene understanding in images and video (Tahmoush and Bonial, 2016). Specifically, we will be exploring how the information relating ENDURANTS and PERDURANTS for a given event type might be leveraged in putting together the pieces relating objects recognized in an image to higher-order scene understanding and perhaps even activity recognition. This is inspired in part by work on "visual semantic role labeling" using FrameNet for situation understanding (Yatskar et al., 2016), but we expect to exploit the more general qualia information to understand, for example, what type of room is encountered if a bed is recognized, versus a tooth brush, based on what we know about how those objects are typically used.

7 Conclusion

An understanding of events is not complete without understanding their participants. Similarly, an understanding of objects is incomplete without some knowledge of what events they are typically associated with. We have described efforts to extract GL qualia relations using a novel methodology exploiting particular information from SUMO relations and documentation. Our evaluation shows that the vast

majority of the automatically extracted relations are judged reasonable by human annotators. Our motivation has been to enrich REO with origin, function and part-whole information in the form of GL qualia relations. To lay the groundwork for this effort, we have first integrated REO with ENDURANTS drawn from carefully selected SUMO classes. We have completed an initial integration of the extracted qualia relation information as annotations in the updated REO. This version of the ontology will be made freely available, as will subsequent versions of the ontology that incorporate the qualia relations as object properties.

References

Christoph Benzmüller and Adam Pease. 2012. Higher-order aspects and context in sumo. *Web Semantics: Science, Services and Agents on the World Wide Web*, 12:104–117.

Susan Brown, Claire Bonial, Leo Obrst, and Martha Palmer. 2017. The rich event ontology. In *Proceedings of the Events and Stories in the News Workshop*, pages 87–97.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC*, volume 2, pages 837–840.

Charles J Fillmore, Collin F Baker, and Hiroaki Sato. 2002. The framenet database and software tools. In *LREC*.

Alessandro Lenci, Nuria Bel, Federica Busa, Nicoletta Calzolari, Elisabetta Gola, Monica Monachini, Antoine Ogonowski, Ivonne Peters, Wim Peters, Nilda Ruimy, et al. 2000. Simple: A general framework for the development of multilingual lexicons. *International Journal of Lexicography*, 13(4):249–263.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation*. University of Chicago press.

Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, and Alessandro Oltramari. 2003. Wonderweb deliverable d18, ontology library (final). *ICT project*, 33052.

Ian Niles and Adam Pease. 2001a. Origins of the ieee standard upper ontology. In *Working notes of the IJCAI-2001 workshop on the IEEE standard upper ontology*, pages 37–42.

Ian Niles and Adam Pease. 2001b. Towards a standard upper ontology. In *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*, pages 2–9. ACM.

Ian Niles and Adam Pease. 2003. Mapping wordnet to the sumo ontology. In *Proceedings of the ieee international knowledge engineering conference*, pages 23–26.

Adam Pease, Ian Niles, and John Li. 2002. The suggested upper merged ontology: A large ontology for the semantic web and its applications. In *Working notes of the AAAI-2002 workshop on ontologies and the semantic web*, volume 28, pages 7–10.

Adam Pease. 2002. Sumo.

Adam Pease. 2011. *Ontology: A practical guide*. Articulate Software Press.

James Pustejovsky. 1991. The generative lexicon. *Computational linguistics*, 17(4):409–441.

Karin Kipper Schuler. 2005. Verbnet: A broad-coverage, comprehensive verb lexicon.

Barry Smith, Mauricio Almeida, Jonathan Bona, Mathias Brochhausen, Werner Ceusters, Melanie Courtot, Randall Dipert, Albert Goldfain, Pierre Grenon, Janna Hastings, et al. 2014. Basic formal ontology 2.0 draft specification and user's guide.

Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. From light to rich ere: annotation of entities, relations, and events. In *Proceedings of the 3rd Workshop on EVENTS at the NAACL-HLT*, pages 89–98.

David Tahmoush and Claire Bonial. 2016. Ontology-based improvement to human activity recognition. In *Automatic Target Recognition XXVI*, volume 9844, page 98440U. International Society for Optics and Photonics.

Mark Yatskar, Luke Zettlemoyer, and Ali Farhadi. 2016. Situation recognition: Visual semantic role labeling for image understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5534–5542.

SeVeN: Augmenting Word Embeddings with Unsupervised Relation Vectors

Luis Espinosa-Anke and Steven Schockaert
School of Computer Science and Informatics, Cardiff University, UK
{EspinosaAnkeL,SchockaertS1}@cardiff.ac.uk

Abstract

We present SeVeN (Semantic Vector Networks), a hybrid resource that encodes relationships between words in the form of a graph. Different from traditional semantic networks, these relations are represented as vectors in a continuous vector space. We propose a simple pipeline for learning such relation vectors, which is based on word vector averaging in combination with an *ad hoc* autoencoder. We show that by explicitly encoding relational information in a dedicated vector space we can capture aspects of word meaning that are complementary to what is captured by word embeddings. For example, by examining clusters of relation vectors, we observe that relational similarities can be identified at a more abstract level than with traditional word vector differences. Finally, we test the effectiveness of semantic vector networks in two tasks: measuring word similarity and neural text categorization. SeVeN is available at bitbucket.org/luisespinosa/seven.

1 Introduction

Word embedding models such as Skip-gram (Mikolov et al., 2013a) and GloVe (Pennington et al., 2014) use fixed-dimensional vectors to represent the meaning of words. These word vectors essentially capture a kind of similarity structure, which has proven to be useful in a wide range of Natural Language Processing (NLP) tasks. Today, one of the major applications of word embeddings is their interaction with neural network architectures, enabling a kind of generalization beyond those words that were only observed during training. For example, if a classification model has learned that news stories containing words such as 'cinema', 'restaurant' and 'zoo' tend to be categorized as 'entertainment', it may predict this latter label also for stories about theme parks due to the shared semantic properties encoded in word vectors. Word embeddings thus endow neural models with some form of world knowledge, without which they would be far less effective. This has prompted a prolific line of research focused on improving word embeddings not only with algorithmic sophistication, but also via explicit incorporation of external knowledge sources such as WordNet (Faruqui et al., 2015), BabelNet (Camacho-Collados et al., 2016) or ConceptNet (Speer et al., 2016).

Regardless of how word vectors are learned, however, the use of fixed-dimensional representations constrains the kind of knowledge they can encode. Essentially, we can think of a word vector as a compact encoding of the salient attributes of the given word. For instance, the vector representation of *lion* might implicitly encode that this word is a noun, and that lions have attributes such as 'dangerous', 'predator' and 'carnivorous'. Beyond these properties, word embeddings can also encode *relational knowledge*. For instance, the embedding might tell us that the words 'lion' and 'zebra' are semantically related, which together with the attributional knowledge that lions are predators and zebras are prey may allow us to plausibly infer that 'lions eat zebras'. However, the way in which relational knowledge can be encoded in word embeddings is inherently limited. One issue is that only relationships which are sufficiently salient can affect the vector representations of their arguments; e.g. the fact that Trump has

Proceedings of the 27th International Conference on Computational Linguistics, pages 2653–2665
Santa Fe, New Mexico, USA, August 20-26, 2018.

visited France is perhaps not important enough to be encoded in the embeddings of the words 'Trump' and 'France' (i.e. there may be insufficient corpus-based evidence on this fact). Note that this is not a matter of how the embedding is learned; forcing the vector representations to encode this fact would distort the similarity structure of the embedding. From a formal point of view, there are also severe limitations to what can be encoded (Gutiérrez-Basulto and Schockaert, 2018). As a simple example, methods based on vector translations cannot model symmetric relations, and they are limited in the kind of many-to-many relations that can be encoded (Lin et al., 2015).

Like word embeddings, semantic networks such as WordNet (Miller, 1995), BabelNet (Navigli and Ponzetto, 2012) or ConceptNet (Speer et al., 2016) also encode lexical and world knowledge. They use a graph representation in which nodes correspond to words, phrases, entities or word senses. Edge labels are typically chosen from a small set of discrete and well-defined lexical and ontological relationships. Compared to word embeddings, the knowledge captured in such resources is more explicit, and more focused on relational knowledge (although attributional knowledge can be encoded as well, e.g., by using edge labels modeling the `has_property` relation). The use of discrete labels for encoding relation types, however, makes such representations too coarse-grained for many applications (e.g., a large proportion of the edges in ConceptNet are labelled with the generic 'related to' relationship). It also means that subjective knowledge cannot be modeled in an adequate way (e.g., forcing us to make a hard choice between which animals are considered to have the property 'dangerous' and which ones do not).

In this paper, we propose a hybrid representation, which we call SeVeN (Semantic Vector Networks). Similar to semantic networks, we use a graph based representation in which nodes are associated with words. In contrast to semantic networks, however, these edges are labelled with a vector, meaning that relation types are modeled in a continuous space.

To obtain a suitable *relation vector* for two given words a and b, we start by averaging the vector representations (from a pre-trained word embedding) of the words that appear in sentences that mention both a and b. The resulting vectors have two main disadvantages, however. First, they are high-dimensional, as they are constructed as the concatenation of several averaged word vectors. Second, the relation vectors are influenced by words that describe the relationship between a and b, but also by words that rather relate to the individual words a or b (as well as some non-informative words). Intuitively we want to obtain a vector representation which only reflects the words that relate to the relationship. For example, the relation vector for (paris,france) should ideally be the same as the vector for (rome,italy), but this will not be the case for the averaged word vectors, as the former relation vector will also reflect the fact that these words represent places and that they relate to France. To address both issues, we introduce an autoencoder architecture in which the input to the decoder comprises both the encoded relation vector and the word vectors for a and b. By explicitly feeding the word vectors for a and b into the decoder, we effectively encourage the encoder to focus on words that describe the relationship between a and b.

Once the semantic vector network has been learned, it can be used in various ways. For instance, the relation vectors could be used for measuring relational similarity (Jurgens et al., 2012), for identifying words that have a specific lexical relationship such as hypernyms (Vylomova et al., 2016), or complementing open information extraction systems (Delli Bovi et al., 2015). In this paper, however, we will assess the potential of SeVeN in terms of two tasks, namely using it for (1) unsupervised semantic similarity modeling, and for (2) enriching word vectors as the input to neural network architectures. The overarching idea in the latter case is that, instead of simply representing each word by its vector representation, the representation for each word position will be composed of (i) the vector representation of the word, (ii) the vector representations of the adjacent words in the semantic vector network, and (iii) the corresponding relation vectors (i.e. the edge labels).

2 Related Work

Related work broadly falls in two categories: methods which aim to improve word embeddings using relational knowledge, and methods which aim to learn relation vectors. To the best of our knowledge, there is no previous work which uses relation vectors with the aim of enriching word embeddings.

Improving Word Embeddings. One of the most notable features of word embedding models, such as

Skip-gram (Mikolov et al., 2013b) and GloVe (Pennington et al., 2014), is the fact that various syntactic and semantic relationships approximately correspond to vector translations. One limitation of vector translations is that they are not well-suited for modeling transitive relations, which is problematic among others for the is-a relationship. To this end, a number of alternative vector space representations have been proposed, which are specifically aimed at modeling taxonomic relationships (Vendrov et al., 2016; Yu et al., 2015; Nickel and Kiela, 2017). Note that while such alternative embedding spaces can solve some of the limitations of standard embeddings w.r.t. modeling taxonomic relationships, there are many other types of relations that cannot be faithfully modeled in these representations. Moreover, these alternative embeddings are not necessarily well-suited for modeling word similarity. More generally, various authors have explored the idea of adapting word embeddings to fit the needs of specific tasks, e.g. aiming to make embeddings better suited for capturing antonyms (Ono et al., 2015), hypernyms (Vulic and Mrksic, 2017) or sentiment (Tang et al., 2014).

As mentioned in the introduction, the use of semantic networks for improving word embeddings, based on the idea that words which are similar in the semantic network should have a similar embedding, has been explored by various authors (Faruqui et al., 2015; Camacho-Collados et al., 2016; Speer et al., 2016). Another possibility is to use a semantic network to decompose word embeddings into sense embeddings by imposing the constraint that the word vector is a convex combination of the corresponding sense vectors, as well as forcing similarity of the sense vector with the vector representations of its neighbors in the semantic network (Johansson and Piña, 2015). Finally, let us refer to work that learns additional embeddings that coexist in the same space as lexemes, e.g., WordNet synsets (Rothe and Schütze, 2015) or BabelNet synsets (Mancini et al., 2017).

Relation Vectors The idea of learning a relation vector for two words a and b, based on the words that appear in their context, goes back at least to the Latent Relational Analysis (LRA) method from (Turney, 2005). In that work, a matrix is constructed with one row for each considered word pair, where columns correspond to lexical patterns that have been extracted from sentences containing these words. The relation vectors are then obtained by applying Singular Value Decomposition (SVD) on that matrix. Along similar lines, in (Riedel et al., 2013) relation vectors are learned by factorizing a matrix whose rows correspond to entity pairs and whose columns correspond to properties (in this case comprising both lexical patterns from a corpus and triples from a knowledge graph). More recently, several methods have been proposed that learn a vector describing the relationship between two words by averaging the embeddings of the words that appear in between them in a given corpus (Weston et al., 2013; Hashimoto et al., 2015; Fan et al., 2015), or by learning a vector representation from PMI-like statistics on how strongly different words are associated with the considered word pair (Jameel et al., 2017). Beyond these unsupervised methods, a wide variety of supervised neural network based architectures have been proposed for learning relation vectors that are predictive of a given relation type (Zeng et al., 2014; dos Santos et al., 2015; Xu et al., 2015).

3 Constructing Semantic Vector Networks

Our aim is to construct a graph whose nodes correspond to words, whose edges indicate which words are related, and whose edge labels are vectors that encode the specific relationship between the corresponding words. We will refer to this representation as a semantic vector network. In this section, we describe our methodology for constructing such semantic vector networks. First, in Section 3.1, we provide details about the source corpus and explain how the structure of the network is chosen. In Section 3.2 we then discuss how suitable relation vectors can be constructed.

3.1 Defining the Network Structure

Our source corpus is a dump of the English Wikipedia from January 2018. We opted to keep preprocessing at a minimum to ensure that any emergent linguistic or relational regularity is captured during the network construction stages. Specifically, we applied sentence segmentation and word tokenization using *nltk*[1]. We also single-tokenized multiword expressions based on several lexicons (Schneider et al.,

[1] nltk.org

2014), and finally removed stopwords using the CoreNLP list[2]. After the above steps, we selected the 10^5 most frequent words as our vocabulary. To determine which words should be connected with an edge, we rely on Pointwise Mutual Information (PMI), which measures the strength of association between two random variables. It is commonly used in NLP as a method for identifying related words, e.g. in factorization based methods for learning word embeddings (Turney and Pantel, 2010). Specifically, we express the strength of association between words w_i and w_j as follows:

$$\textit{pmi}(w_i, w_j) = \log\left(\frac{x_{ij}x_*}{x_i x_j}\right)$$

In our case, x_{ij} is the number of times word w_i appears near word w_j, weighted by the nearness of their co-occurrences, and $x_i = \sum_j x_{ij} = \sum_j x_{ji}$ and $x_* = \sum_i \sum_j x_{ij}$. Specifically, let I_{w_i} be the word positions in the corpus at which w_i occurs, then we define:

$$x_{ij} = \sum_{p \in I_{w_i}} \sum_{q \in I_{w_j}} n(p, q)$$

where $n(p, q) = 0$ if the word positions p and q belong to a different sentence, or if $|p - q| > 10$, i.e. if there are at least 10 words in between them. Otherwise we define $n(p, q) = \frac{1}{|p-q|}$.

sorrow		**tournament**		**videogame**		**riverbank**	
ppmi	w2v	ppmi	w2v	ppmi	w2v	ppmi	w2v
contrition	sadness	scotties	tourney	lego	videogames	danube	riverbanks
lamentation	anguish	double-elimination	tournaments	consoles	videogaming	erosion	river
woe	grief	single-elimination	Tournament	villains	next_gen_consoles	laboratories	creek
savior	profound_sorrow	pre-olympic	tournment	arcade	Videogame	opposite	riverbed
everlasting	deepest_sorrow	4-day	tourament	sega	gamers	vegetation	riverside
anguish	heartfelt_sorrow	eight-team	tourneys	ea	MMOG	tales	lake
grief	profound_sadness	winnings	touranment	playstation	PS2	washed	shoreline

Table 1: Examples of the highest scoring (i.e. most strongly associated by pmi) words, as well as their nearest neighbors in the pretrained word2vec (w2v) Google news vector space.

To choose the edges of the semantic vector network, we only consider word pairs which co-occur at least 10 times in the corpus. Among such pairs, for each word w_i, we first select the 10 words w_j whose score $\textit{pmi}(w_i, w_j)$ is highest. This resulted in a total of about 900 000 pairs. Then, we added the overall highest scoring pairs (w_i, w_j) which had not yet been selected, until we ended up with a total of approximately 10^6 edges involving the initial vocabulary of 10^5 words. In the following we will write N_w for the neighbors of w, i.e. the set of words n such that $\{w, n\}$ was selected as an edge.

Note that by capturing pairs of words strongly connected by PMI, we encode a different type of relatedness than proximity in word embeddings. To illustrate this, in Table 1 we compare the most closely related words in our PMI graph, for some selected target words, with their nearest neighbors in the *word2vec* Google News word embedding space[3] (measured by cosine similarity). While the word2vec neighbors mostly consist of near-synonyms and other syntagmatic relationships, the chosen PMI pairs include a wide variety of topically related linguistic items. A semantic network based on such PMI pairs should thus capture information which is complementary to what is captured in word embeddings.

3.2 Learning relation vectors

Our general strategy for learning relation vectors is based on averaging word vectors. Specifically, for each sentence s in which w_i occurs before w_j (within a distance of at most 10), we construct three vectors, based on the words $a_1, ..., a_k$ which appear before w_i, the words $b_1, ..., b_l$ which appear in between w_i

[2] github.com/stanfordnlp/CoreNLP/blob/master/data/edu/stanford/nlp/patterns/surface/stopwords.txt

[3] https://code.google.com/archive/p/word2vec/

and w_j and the words $c_1, ..., c_q$ which appear after w_j:

$$pre^s_{w_iw_j} = \frac{1}{k}\sum_{r=1}^{k}\mathbf{v}_{a_r} \qquad mid^s_{w_iw_j} = \frac{1}{l}\sum_{r=1}^{l}\mathbf{v}_{b_r} \qquad post^s_{w_iw_j} = \frac{1}{q}\sum_{r=1}^{q}\mathbf{v}_{c_r}$$

where we write $\mathbf{v}_w$ for the vector representation of the word w. These vectors are then averaged over all sentences S_{ij} where w_i occurs before w_j:

$$pre_{w_iw_j} = \frac{1}{|S_{ij}|}\sum_{s\in S_{ij}} pre^s_{w_iw_j} \quad mid_{w_iw_j} = \frac{1}{|S_{ij}|}\sum_{s\in S_{ij}} mid^s_{w_iw_j} \quad post_{w_iw_j} = \frac{1}{|S_{ij}|}\sum_{s\in S_{ij}} post^s_{w_iw_j}$$

Since we can similarly obtain such vectors from sentences where w_j appears before w_i, we end up with a relation vector whose dimensionality is six times higher than the dimensionality of the word embedding, which would be impractical in the kind of applications we envisage (see Section 4). Another problem with these vectors is that they do not only reflect the relationship between w_i and w_j, but also the words w_i and w_j themselves. For instance, suppose we want to model the relationship between the words 'movie' and 'popcorn'. A sentence mentioning these two words could be:

Buttered popcorn is commonly eaten at movie theatres.

The most relevant words for describing the relationship are 'eaten at'. In contrast, however, 'buttered' is mostly related to the word 'popcorn' itself rather than describing its relationship with 'movie'. Similarly, 'theatres' is related to 'movie', but not relevant for characterizing the relationship.

To solve both issues, we propose to use an autoencoder architecture, in which the decoder has access to the word vectors $\mathbf{v}_{w_i}$ and $\mathbf{v}_{w_j}$, in addition to the encoded version of the relation vector. Let us write $\mathbf{z}_{w_iw_j}$ for the concatenation of $pre_{w_iw_j}$, $mid_{w_iw_j}$, $post_{w_iw_j}$, $pre_{w_jw_i}$, $mid_{w_jw_i}$, $post_{w_jw_i}$. Then the encoder is given by:

$$\mathbf{r}_{w_iw_j} = A\,\mathbf{z}_{w_iw_j} + \mathbf{b}$$

where $A \in \mathbb{R}^m \times \mathbb{R}^{6d}$ and $\mathbf{b} \in \mathbb{R}^m$, where d is the dimensionality of the word vectors, and m is the dimensionality of the encoded relation vectors m. In our experiments we experiment with different values for m. Empirically, we find that as the dimensionality of the compressed representations becomes smaller, the importance of word semantics gradually fades away in favor of their corresponding relational properties. Then, the decoder is then defined as:

$$\mathbf{z}^*_{w_iw_j} = B(\mathbf{v}_{w_i} \oplus \mathbf{r}_{w_iw_j} \oplus \mathbf{v}_{w_j}) + \mathbf{c}$$

where $\oplus$ denotes vector concatenation, $B \in \mathbb{R}^{6d} \times \mathbb{R}^{m+2d}$ and $\mathbf{c} \in \mathbb{R}^{6d}$. To train the autoencoder, we use the following L2-regularized reconstruction loss:

$$\mathcal{L} = \|\mathbf{z}_{w_iw_j} - \mathbf{z}^*_{w_iw_j}\|_2^2 + \lambda\|\mathbf{r}_{w_1n_1}\|_2^2$$

with $\lambda > 0$ a regularization parameter. This loss function balances two objectives: minimizing the reconstruction error and keeping the L2 norms of the encoded relation vectors as small as possible. Because of this latter part, we can think of the norm of the relation vectors $\mathbf{r}_{w_iw_j}$ as a measure of how strongly the words w_i and w_j are related. In particular, if sentences mentioning w_i and w_j contain few or no words that describe their relationship, we might expect $\mathbf{r}_{w_iw_j}$ to be close to the 0 vector.[4]

4 Evaluation

We propose to evaluate our semantic vector networks from three different standpoints. First, we provide a qualitative evaluation by exploring relation network spaces (both compressed and uncompressed) and discussing meaningful properties. Second, we perform experiments in word similarity where we compare

[4]We also conducted experiments without the regularization term, with slightly worse results across all evaluations.

against the standard approach of measuring the similarity of two words by means of the cosine distance between their corresponding word vectors. This evaluation serves as an illustration of how semantic vector networks could be used in an unsupervised application setting. Third, as a prototypical example of a supervised application setting, we analyze the impact of leveraging the enriched representation these networks provide in neural text classification, in particular topic categorization and sentiment analysis. In all experiments, the pretrained embeddings we use (both for baselines and for constructing the relation networks) are the *word2vec* Google News embeddings (Mikolov et al., 2013b).

4.1 Qualitative Evaluation

One of the strongest selling points of word embeddings is that they enable inference of relational properties, which can be obtained by simple vector arithmetic such as summation and subtraction (Levy et al., 2015). The basic idea is that the relationship between two words w_i and w_j is characterized by the vector difference $\mathbf{v}_{w_i} - \mathbf{v}_{w_j}$. Such vector differences, however, encode relations in a noisy way. For instance, while the differences $\mathbf{v}_{rome} - \mathbf{v}_{italy}$, $\mathbf{v}_{paris} - \mathbf{v}_{france}$ and $\mathbf{v}_{dublin} - \mathbf{v}_{ireland}$ are all rather similar, there are in fact many other word pairs (not in a capital-of relationship) whose difference is also similar to these differences (Bouraoui et al., 2018). Accordingly, it was found in (Vylomova et al., 2016) that a relation classifier which is trained on word vector differences is prone to predicting many false positives. In contrast, we can expect that our relation vectors are modeling relations in a far less ambiguous way. On the other hand, these relation vectors are limited to word pairs that co-occur sufficiently frequently. Apart from the associated sparsity issues, this also suggests that relation vectors are not suitable for characterizing syntagmatic relationships and several types of syntactic relationships. We thus view these relation vectors as complementary to word vector differences.

In this section we illustrate the semantic properties of different versions of SeVeN. To this end, we show the nearest neighbors of selected target relation vectors for a number of different representations: (1) the original 1800d SeVeN network (`original`), (2) an autoencoded 10-dimensional space (`compressed-10d`), (3) a slightly higher-dimensional version (`compressed-50d`), and finally (4) a baseline model according to which the relation between two words is modeled as the vector difference of the corresponding word vectors (`diffvec`). The five selected target relation vectors, along with their nearest neighbors, are shown in Table 2. These target relation vectors were chosen to capture a range of different types of relationships, including hypernymic ('nintendo - console' and 'gmail - email') and attributional ('roman - numerals') relations.

One immediate observation is that, in most cases, the `diffvec` neighbors remain very close to the given word pair, where each word from the given pair is either preserved or replaced by a closely related word. The `original` and `compressed-50d` relation vectors largely follow a similar trend, although a few more interesting analogies are also found in these cases (e.g. *arabic - alphabet* as a neighbor of *roman - numerals*). The results for the `compressed-10d` vectors, however, follow a markedly different pattern. For these low-dimensional vectors, our autoencoder forces the relation vectors to focus on modeling the relationship between the two words, while abstracting away from the initial domain. This leads to several interesting neighbors, although this seems to come at the cost of some added noise.

Let us now analyze more closely the results of the `compressed-10d` vectors. If we read the first example along the lines of "juice can be made from limes", similar relations are found close in the space, such as 'coconut - milk' and 'marzipan - paste'. Note that the relation 'noodles - egg' is also similar, although the two words appear in the incorrect order (i.e. noodles can be made from eggs rather than the other way around). As another example where the directionality of this pattern is not captured correctly, we also find the pair 'juice - lime'. It would be interesting to analyze in future work whether such issues can be avoided by using features from a dependency parser, e.g. following a similar strategy as in (Levy and Goldberg, 2014). Note that while all the `compressed-10d` neighbors are still related to food, these vectors have generalized beyond the domain of citrus fruits (see e.g., 'lime', 'tamarind' or 'lemon' in `diffvec`, or 'lemon' and 'orange' in `original`). A similar phenomenon occurs in some of the other examples. In the 'nintendo-console' case, after interpreting the relation as "major supplier of" or "entity which popularized", we find nearest neighbors in the `compressed-10d` space

lime_juice			
original	compressed-10d	compressed-50d	diffvec
lemon_juice	lemon_juice	lemon_juice	lime_soda
juice_lemon	coconut_milk	juice_lemon	lime_lemon
juice_lime	marzipan_paste	juice_lime	lemon_juice
lime_lemon	juice_lime	vinegar_sour	citric_juice
lemon_lime	noodles_egg	lemon_lime	tamarind_juice
pineapple_juice	lime_lemon	vinegar_sauce	lime_pie
orange_juice	marinated_beef	lime_lemon	pineapple_juice
nintendo_console			
original	compressed-10d	compressed-50d	diffvec
wii_console	wii_console	wii_console	nintendo_consoles
playstation_console	playstation_console	nintendo_nes	nintendo_handheld
nintendo_nes	nintendo_nes	playstation_console	gamecube_console
xbox_console	witcher_2	xbox_console	wii_console
nintendo_consoles	itunes_download	nintendo_consoles	dreamcast_console
famicom_console	imax_2d	sega_consoles	nintendo_switch
nintendo_64	netflix_streaming	nintendo_handheld	3ds_console
gmail_email			
original	compressed-10d	compressed-50d	diffvec
yahoo_email	renders_firefox	yahoo_email	gmail_emails
inbox_email	ie_browser	gmail_e-mail	yahoo_email
hotmail_email	infinitive_suffix	inbox_email	hotmail_email
email_yahoo	firefox_browser	gmail_emails	addy_email
gmail_e-mail	carnap_semantics	email_yahoo	imap_email
sending_email	helvetica_font	hotmail_email	smtp_email
send_email	cv_syllable	google_search	bugzilla_email
roman_numerals			
original	compressed-10d	compressed-50d	diffvec
arabic_numerals	arabic_alphabet	arabic_numerals	cyrillic_numerals
letters_numerals	greek_alphabet	letters_numerals	indic_numerals
letters_alphabet	10-inch_discs	uppercase_letters	georgian_numerals
lowercase_letters	latin_alphabet	lowercase_letters	hieratic_numerals
arabic_alphabet	yemenite_pronunciation	uppercase_characters	brahmi_numerals
latin_alphabet	standard_orthography	latin_alphabet	sinhala_numerals
symbols_numerals	wii_remote	alphabetic_numerals	quantifiers_numerals
heavy_metal			
original	compressed-10d	compressed-50d	diffvec
thrash_metal	metal_heavy	thrash_metal	heavy_metals
glam_metal	karma_dharma	doom_metal	cky_metal
doom_metal	techno_rave	glam_metal	manilla_metal
symphonic_metal	psychedelic_garage	thrash_slayer	annihilator_metal
nu_metal	cooking_recipes	punk_rock	heaviness_metal
sludge_metal	gita_yoga	hardcore_punk	doro_metal
glam_rock	post-punk_punk	sludge_metal	behemoth_metal

Table 2: Nearest neighbors (by cosine) for selected relation vectors and the three models under consideration.

where the same relation holds, but which do not belong to the video games domain, such as 'itunes-download' or 'netflix-streaming'. Next, we find that the relation holding between 'gmail' and 'email' is similar to 'ie' and 'firefox' and 'browser', and even 'helvetica' and 'font'. The relation between 'google' and 'search', found for `compressed-50d`, is also of this kind. In contrast, the `diffvec` neighbors in this case all have *email* as the second word. In the 'roman-numerals' example, likewise, the `diffvec` neighbors similarly have 'numerals' as the second word, while for `compressed-10d` we see more interesting neighbors such as 'arabic-alphabet' and 'yemenite-pronunciation'. We also find the seemingly unrelated 'wii-remote' pair, although we may consider that the Nintendo Wii console introduced a fundamentally new type of remote, which at an abstract level is similar to the fact that the Romans introduced a fundamentally new way of writing numbers. This example also suggests, however,

that the way in which relations are modeled in the 10-dimensional space might be too abstract for some applications. Finally, the 'heavy-metal' case is a paramount example of how the the relation vectors may capture information which is fundamentally different than what is encoded by word vectors. In particular, the `diffvec` vectors all express relationships from the metalwork domain (e.g., 'heavy-metals' or 'annihilator-metal'), which reflects the fact that the music-related interpretation of the word 'metal' is not its dominant sense. In contrast, since our relation vectors are exclusively learned from sentences where both words co-occur ('heavy' and 'metal' in this example), the vector for 'heavy metal' clearly captures the musical sense (see e.g., 'thrash-metal' or 'glam-metal' in the `original` space).

4.2 Modeling Similarity

The capacity to capture and *embed* nuances of word meaning is one of the most celebrated features of word embeddings. The task of semantic similarity measurement, therefore, has been adopted as a *de-facto* testbed for measuring the quality of representations of linguistic items. The standard practice is to consider a distance (or similarity) metric such as cosine similarity and compare the similarity in a given vector space model with respect to human judgement. We note, however, that there exist other similarity metrics discussed in the literature, e.g., Weighted Overlap (Pilehvar et al., 2013) or Tanimoto Distance (Iacobacci et al., 2015). Our proposed similarity measurement parts ways from the idea of improving the representation of individual words, and rather seeks to refine their meaning by incorporating complementary cues via relation vectors, as well as the corresponding neighborhood structure. There are many possible ways in which this could be done, but we restrict ourselves here to a simple strategy, based on identifying the closest neighbors of the two words w_1 and w_2. The main intuition is that when w_1 and w_2 are similar, they should also be related to similar words. Specifically, we first determine the closest match between the neighbors of w_1 and the neighbors of w_2, as follows:

$$(n_1, n_2) = \underset{(n_1,n_2)\in N_{w_1}\times N_{w_2}}{\arg\max} \cos(\mathbf{v}_{n_1}, \mathbf{v}_{n_2}) + \cos(\mathbf{r}_{w_1n_1}, \mathbf{r}_{w_2n_2})$$

Note that to identify these neighbors, we compare both their word vectors $\mathbf{v}_{n_1}$ and $\mathbf{v}_{n_2}$, and their relationships to the target words, $\mathbf{r}_{w_1n_1}$ and $\mathbf{r}_{w_2n_2}$. Once these neighbors have been identified, we compute the similarity between w_1 and w_2 as follows:

$$\textit{sim}(w_1, w_2) = \cos(\mathbf{v}_{w_1} \oplus \mu\mathbf{v}_{n_1} \oplus \mathbf{r}_{w_1n_1}, \mathbf{v}_{w_2} \oplus \mu\mathbf{v}_{n_2} \oplus \mathbf{r}_{w_2n_2})$$

where $0 < \mu \leq 1$ is a scaling factor which is aimed at reducing the impact of the neighbors n_1 and n_2 on the overall similarity computation. The fact that $\mathbf{v}_{n_1}$ is similar to $\mathbf{v}_{n_2}$ is an important indicator for the similarity between w_1 and w_2, but it should not influence the resulting similarity score as much as the similarity of the word vectors of w_1 and w_2 themselves. Rather than tuning this value, in the experiments we have fixed it as $\mu = 0.5$, which was found to give better results than $\mu = 1$ (i.e. no scaling). Note that the proposed way of computing similarities favours words of the same type. For example, we may expect 'Spain' and 'France' to be more similar than 'Spain' and 'Barcelona', when this metric is used, since 'Spain' and 'France' are associated with the neighbors 'Madrid' and 'Paris' which are similar, and which are related in a similar way to the target words. In our experiments, we also consider a variant in which the relation vectors are only used for selecting the neighbors. The similarity itself is then calculated as:

$$\textit{sim}(w_1, w_2) = \cos(\mathbf{v}_{w_1} \oplus \mu\mathbf{v}_{n_1}, \mathbf{v}_{w_2} \oplus \mu\mathbf{v}_{n_2})$$

We evaluate the proposed similarity measure on four well-known benchmarking datasets for word representation learning. These are: (1) `rg65` (Rubenstein and Goodenough, 1965); (2) `wordsim` (Finkelstein et al., 2001); (3) `mc` (Miller and Charles, 1991); and (4) the English portion of `semeval17` (Camacho-Collados et al., 2017). We restrict our experiment to single words, and do not consider multiword expressions (e.g., named entities), as this would require a different approach for compositional meaning representation. We compare against a baseline model based on cosine similarity between the vectors of the target words (`cosine`). As for our proposed models, and observing the similarity measurement described above, we consider a 10-dimensional relation space, without (10rv$_w$) and with (10rv$_r$) the

relation vector as part of the similarity computation. We also provide results stemming from using the original 1800-dimensional relation vector model. As is customary in the literature, we use Pearson's (**p**) and Spearman's (**s**) correlation coefficients as evaluation metrics, as well as their average (**avg.**). Table 3 shows that the 10rv_w variant consistently outperforms the word-level baseline. Somewhat surprisingly, the variant 10rv_r (which uses the relation vector also in the similarity computation) performs consistently worse than the variant 10rv_w. When using the original 1800-dimensional vectors, however, the situation is reversed, with 1800rv_r outperforming 1800rv_w, and achieving the best results overall (with the exception of `mc`). These results clearly show that the relation vectors capture valuable information for measuring word similarity, although the information captured by the 10-dimensional vectors may in some cases be too abstract for this purpose.

	`rg`			`wordsim`			`mc`			`semeval17`		
	p	**s**	**avg.**	**p**	**s**	**avg.**	**p**	**s**	**avg.**	**p**	**s**	**avg.**
`cosine`	77.2	76.0	76.6	64.9	69.4	67.1	79.2	80.0	79.6	69.4	70.0	69.7
10rv_w	78.1	77.0	77.5	66.0	69.6	67.8	79.7	80.7	**80.2**	70.2	70.8	70.5
10rv_r	77.4	75.5	76.4	65.8	69.5	67.6	78.8	77.9	78.3	70.0	70.7	70.3
1800rv_w	79.5	80.6	**80.0**	67.4	69.8	68.6	79.4	79.0	79.2	71.4	71.8	71.6
1800rv_r	78.9	80.2	79.5	68.1	70.1	**69.1**	79.2	79.7	79.4	72.2	73.0	**72.6**

Table 3: Correlation results for different configurations of our proposed approach and a competitor baseline based on cosine similarity of word embeddings.

4.3 Text Classification

Semantic Vector Networks may be thought of as a natural way of enriching word-level semantic representations, which may in turn be useful for informing a neural architecture with relational (e.g., commonsense or lexical) knowledge. We will focus on two well known tasks, namely text categorization and sentiment analysis. Our goal is to examine the extent to which the performance of a vanilla neural network increases by being injected vector graph information as a complement to the information encoded in each individual word embedding. The strength of our proposal lies in the fact that this information comes exclusively from corpora, and thus the need to rely on often incomplete, costly and language dependent ontological or lexical resources is avoided.

As evaluation benchmarks we use three text categorization datasets, namely `20news` (Lang, 1995), `bbc` (Greene and Cunningham, 2006) and `reuters` (Lewis et al., 2004). We also consider two polarity detection datasets (positive or negative), namely the Polarity04 (`pol.04`) (Pang and Lee, 2004) and Polarity05 (`pol.05`) (Pang and Lee, 2005) datasets, and finally a 10k document subset of the *apps for android* (`apps4and.`) corpus[5] (He and McAuley, 2016), which features reviews and associated ratings on a scale from 1 to 5. The neural network model we use for our experiments is a combination of a CNN (LeCun et al., 1998) and a bidirectional LSTM (Hochreiter and Schmidhuber, 1997). CNNs have been evaluated extensively in text classification (Johnson and Zhang, 2015; Tang et al., 2015; Xiao and Cho, 2016; Conneau et al., 2017) and sentiment analysis (Kalchbrenner et al., 2014; Kim, 2014; dos Santos and Gatti, 2014; Yin et al., 2017), and this specific model (CNN+BLSTM) has been explored in different NLP benchmarks (Kim, 2014). Finally, as evaluation metrics we use precision (**p**), recall (**r**) and f-score (**f**), as well as accuracty (**acc.**).

To use SeVeN for text classification, we keep the exact same neural network architecture, but use enriched vector representations for each word. As a proof-of-principle, in this paper, this enriched vector representation is simply obtained by concatenating the word vector of that word, with vector representations of its top-10 neighbors according to PMI (ordered by this PMI score), together with the corresponding relation vectors. For example, with word embeddings of 300 dimensions and relation vectors of 10 dimensions, the input for each word is given by a 3400-dimensional vector. We list experimental

[5]Obtained from `http://jmcauley.ucsd.edu/data/amazon/`.

	bbc			20news			reuters-r56			apps4and.			pol.04	pol.05
	p	**r**	**f**	**p**	**r**	**f**	**p**	**r**	**f**	**p**	**r**	**f**	**acc.**	**acc.**
cosine	0.95	0.95	**0.95**	0.86	0.85	0.86	0.85	0.88	0.86	0.39	0.48	0.38	0.54	**0.78**
10rv	0.95	0.95	**0.95**	0.88	0.87	0.87	0.89	0.91	**0.90**	0.40	0.44	**0.40**	0.56	0.75
20rv	0.96	0.95	**0.95**	0.89	0.89	**0.89**	0.89	0.92	**0.90**	0.38	0.48	**0.40**	0.59	**0.78**
50rv	0.94	0.94	0.94	0.88	0.87	0.88	0.89	0.91	**0.90**	0.35	0.46	0.38	**0.60**	0.77

Table 4: Experimental results on six benchmarking datasets for text classification.

results for several configurations, where the number of neighbors stays fixed, but the relation vector (rv) changes in dimensionality (10, 20 or 50). Experimental results are provided in Table 4. We can see that for the 20-dimensional vectors, the results are consistently better (or at least as good as) the baseline. The results for the 10-dimensional and 50-dimensional vectors are similar, although these configurations perform slightly worse than the baseline for pol.05.

Overall, these results show the usefulness of the relation vectors and neighborhood structure, despite the rather naive way in which this information is used. It seems plausible to assume that performance may be further improved by using network architectures which exploit the graph structure in a more direct way.

5 Conclusions and Future Work

In this paper we have presented SeVeN, a dedicated vector space model for relational knowledge. These relation vectors encode corpus-based evidence capturing the different contexts in which a pair of words may occur. An initially high-dimensional relation vector is is further "purified" thanks to a simple *ad-hoc* autoencoder architecture, designed to only retain relational knowledge. We have explored the characteristics of these vector networks qualitatively, by showing highly correlated word pairs, as opposed to, for example, difference vectors. While the latter are often assumed to capture relational properties, we found that the relational similarities they capture largely reflect the similarities of the individual words, with little relational generalization capability. In addition, we have evaluated our SeVeN vectors in terms of their usefulness in two standard NLP tasks: word similarity and text classification. In both cases we obtained better results than baselines that use standard word vectors alone.

There are several interesting avenues for future work. First, an obvious way to improve these unsupervised representations would be to leverage structured knowledge retrieved from knowledge graphs and/or Open Information Extraction systems. Such knowledge could easily be exploited by feeding any available structured knowledge as additional inputs to the autoencoder. Another way in which structured knowledge could be harnessed would simply be to label relation vectors, i.e. to identify regions in the relation vector space that correspond to particular relation types (e.g. hypernymy). Another possibility would be to improve SeVeN by aggregating relation vectors along paths in the graph. In this way, we may learn to predict missing edges (or to smooth relation vectors that were learned from too few or too uninformative sentences), similarly to the random walk based strategies that have been developed for completing traditional semantic networks and knowledge graphs (Gardner et al., 2014).

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. This work was supported by ERC Starting Grant 637277.

References

Zied Bouraoui, Shoaib Jameel, and Steven Schockaert. 2018. Relation induction in word embeddings revisited. In *Proceedings of the 27th International Conference on Computational Linguistics*.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Nasari: Integrating explicit

knowledge and corpus statistics for a multilingual representation of concepts and entities. *Artificial Intelligence*, 240:36–64.

Jose Camacho-Collados, Mohammad Taher Pilehvar, Nigel Collier, and Roberto Navigli. 2017. Semeval-2017 task 2: Multilingual and cross-lingual semantic word similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 15–26.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1107–1116.

Claudio Delli Bovi, Luis Espinosa Anke, and Roberto Navigli. 2015. Knowledge base unification via sense embeddings and disambiguation. In *Proceedings of EMNLP*, pages 726–736, Lisbon, Portugal, September. Association for Computational Linguistics.

Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78.

Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proc. ACL*, pages 626–634.

Miao Fan, Kai Cao, Yifan He, and Ralph Grishman. 2015. Jointly embedding relations and mentions for knowledge population. In *Proc. RANLP*, pages 186–191.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom M. Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 397–406.

Derek Greene and Pádraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of the 23rd international conference on Machine learning*, pages 377–384. ACM.

Víctor Gutiérrez-Basulto and Steven Schockaert. 2018. From knowledge graph embedding to ontology embedding: Region based representations of relational structures. *arXiv preprint arXiv:1805.10461*.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2015. Task-oriented learning of word embeddings for semantic relation classification. In *Proc. CoNLL*, pages 268–278.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517. International World Wide Web Conferences Steering Committee.

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

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Sensembed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 95–105.

Shoaib Jameel, Zied Bouraoui, and Steven Schockaert. 2017. Modeling semantic relatedness using global relation vectors. *CoRR*, abs/1711.05294.

Richard Johansson and Luis Nieto Piña. 2015. Embedding a semantic network in a word space. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1428–1433.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. *NAACL*.

David Jurgens, Saif Mohammad, Peter D. Turney, and Keith J. Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proceedings of the 6th International Workshop on Semantic Evaluation*, pages 356–364.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*, pages 331–339. Elsevier.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 302–308.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187.

Massimiliano Mancini, Jose Camacho-Collados, Ignacio Iacobacci, and Roberto Navigli. 2017. Embedding words and senses together via joint knowledge-enhanced training. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 100–111.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.

George A Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, pages 6341–6350.

Masataka Ono, Makoto Miwa, and Yutaka Sasaki. 2015. Word embedding-based antonym detection using thesauri and distributional information. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 984–989.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

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

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. 2013. Align, disambiguate and walk: A unified approach for measuring semantic similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1341–1351.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proc. HLT-NAACL*, pages 74–84.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1793–1803.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A Smith. 2014. Discriminative lexical semantic segmentation with gaps: running the mwe gamut. *Transactions of the Association for Computational Linguistics*, 2:193–206.

Robert Speer, Joshua Chin, and Catherine Havasi. 2016. ConceptNet 5.5: An open multilingual graph of general knowledge. *AAAI Conference on Artificial Intelligence*.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1555–1565.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Peter D. Turney. 2005. Measuring semantic similarity by latent relational analysis. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1136–1141.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ivan Vulic and Nikola Mrksic. 2017. Specialising word vectors for lexical entailment. *CoRR*, abs/1710.06371.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1366–1371.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367*.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.

Zheng Yu, Haixun Wang, Xuemin Lin, and Min Wang. 2015. Learning term embeddings for hypernymy identification. In *IJCAI*, pages 1390–1397.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *Proceedings of the International Conference on Computational Linguistics*, pages 2335–2344.

Evaluation of Unsupervised Compositional Representations

Hanan Aldarmaki
The George Washington University
aldarmaki@gwu.edu

Mona Diab
The George Washington University
mtdiab@gwu.edu

Abstract

We evaluated various compositional models, from bag-of-words representations to compositional RNN-based models, on several extrinsic supervised and unsupervised evaluation benchmarks. Our results confirm that weighted vector averaging can outperform context-sensitive models in most benchmarks, but structural features encoded in RNN models can also be useful in certain classification tasks. We analyzed some of the evaluation datasets to identify the aspects of meaning they measure and the characteristics of the various models that explain their performance variance.

1 Introduction

Distributed semantic models for words encode latent features that reflect semantic aspects and correlations among words. The goal of compositional semantic models is to induce latent semantic representations that encode the meaning of phrases, sentences, and paragraphs of variable lengths. Some neural architectures such as convolutional (Kim, 2014) and recursive networks (Socher et al., 2013) handle variable-length input by identifying shift-invariant features suitable for the classification problem at hand, which makes it possible to skip composition and work directly with the entire space of individual word embeddings. While such models can achieve excellent performance in supervised classification tasks such as sentiment analysis, we are interested in generic unsupervised fixed-length representations for variable-length text sequences so as to efficiently preserve essential semantic content for later use in various supervised and unsupervised settings.

Binary bag-of-words are simple and effective representations that serve as a strong baseline in several classification benchmarks (Wang and Manning, 2012). However, they do not exploit the distributional relationships among different words, which limits their applicability and generalization when training data are scarce. Additive compositional functions, such as word vector sum or average, are more effective in semantic similarity tasks even when compared with tensor-based compositional functions (Milajevs et al., 2014) and can outperform more complex and better tuned models based on recurrent neural architectures on out-of-domain data (Wieting et al., 2015a). Yet, averaging also has several drawbacks: unlike binary representations, the individual word identities are lost, and some words that do not carry semantic significance may end up being more prominently represented than essential words. Furthermore, additive compositional models disregard sentence structure and word order, which can lead to loss of semantic nuance. To alleviate the first issue, the weights of various words can be adjusted using word frequency statistics (Riedel et al., 2017) or by inducing context-sensitive weights using recurrent neural networks (Wieting and Gimpel, 2017), both of which have been shown to outperform vector averaging. Context-sensitive feed-forward neural models like the paragraph vector (Le and Mikolov, 2014) potentially incorporate word order, yet the training objective may not be sufficient to model deeper structure. Sequence encoder-decoder models, on the other hand, can be trained with various sentence-level objectives, such as neural machine translation (NMT) (Sutskever et al., 2014), predicting surrounding

Proceedings of the 27th International Conference on Computational Linguistics, pages 2666–2677
Santa Fe, New Mexico, USA, August 20-26, 2018.

sentences (i.e. skip-thought) (Kiros et al., 2015), or reconstruction of the input using denoising autoencoders (Hill et al., 2016). These sequential models have been evaluated and compared against other models and baselines on several supervised and unsupervised tasks in Hill et al. (2016). The denoising autoencoder model and skip-thought both performed well in supervised tasks, while the NMT model performed worse than the baselines. All three performed poorly in unsupervised settings.

To bridge some of the gaps in evaluation, we evaluated a subset of models with increasing complexity, from binary bag-of-words to RNNs, on various supervised and unupervised settings. Our objective is to evaluate compositional models against strong baselines and identify the elements that lead to performance gains. We evaluated binary vs. distributed features, weighted vs. unweighted averaging, three different word embedding models, and four context-sensitive models that optimize different objectives: the paragraph vector, the gated recurrent averaging network (Wieting and Gimpel, 2017), skip-thought, and an LSTM encoder trained on labeled natural language inference data (inferSent) (Conneau et al., 2017). We also analyzed the intrinsic structures of the various models by visual inspection and k-means clustering to gain insights into structural differences that may explain the variance in performance.

2 Background: Unsupervised Compositional Models

2.1 Baselines

The simplest way of representing a sentence is a binary bag-of-words representation, where each word is a feature in the vector space. This results in large and sparse representations that only account for the existence of individual words within a sentence, yet they have been shown to be effective in various supervised classification tasks, especially in combination with *n*-grams and Naive Bayes (NB) features (Wang and Manning, 2012). Let $\vec{x}_i$ be the binary representation of sentence i, and $y_i \in \{0, 1\}$ its label. The log-count ratio $\vec{r}$ is calculated as

$$\vec{r} = \log \frac{\vec{p}/\|\vec{p}\|}{\vec{q}/\|\vec{q}\|} \tag{1}$$

Where $\vec{p} = 1 + \sum_{i:y_i=1} \vec{x}_i$ and $\vec{q} = 1 + \sum_{i:y_i=0} \vec{x}_i$ are the smoothed count vectors for each class (i.e. the number of samples in the class that include each feature). The feature vectors are then modified using the element-wise product $\vec{x}_i \circ \vec{r}$. NB features identify the most discriminative words for each task, so using them results in task-specific rather than general representations. However, given the relative efficiency of this model, we include it as a baseline for comparison.

2.2 Word Embeddings and Composition Functions

Representations of variable-length sentences and paragraphs can be constructed by averaging the embeddings of all words within a sentence. However, simple averaging may not be the best approach since not all words within a sentence are semantically relevant. The following methods can be used to adjust the weights of words according to their frequency, assuming that frequent words have lower semantic content:

`tf-idf`-weighted Average The *term frequency-inverse document frequency* statistic measures the importance of a word to a document. We treat each sentence as a document and calculate the `idf` weight for term t as follows:

$$idf_t = \texttt{log}\frac{N}{1 + n_t} \tag{2}$$

where N is the total number of sentences and n_t the number of sentences in which the term appears. Terms that appear in more documents have lower `idf` weights.

`sif`-weighted Average The *smooth inverse frequency* (Riedel et al., 2017) is an alternative measure for discounting the weights of frequent words as follows:

$$sif_t = \frac{a}{a + p(t)} \tag{3}$$

where a is a smoothing parameter and $p(t)$ is the relative frequency of the term in the training corpus. In addition, as proposed in (Riedel et al., 2017), we subtract the projection of the vectors on the first principal component which corresponds to syntactic features associated with common words.

2.2.1 Word Embeddings

Random word projections: we generated a random vector drawn from the standard normal distribution for each word in the vocabulary. The vector sum of random word vectors is a low-dimensional projection of binary bag-of-words vectors. .

Continuous Bag of Words: `CBOW` is an efficient log-linear model for learning word embeddings using a feed-forward neural network classifier that predicts a word given the surrounding words within a fixed context window (Mikolov et al., 2013a). In Schnabel et al. (2015) word embedding evaluation, `CBOW` outperformed other word embeddings in word relatedness and analogy tasks.

Global Vectors: `GloVe` is a global log-bilinear regression model (Pennington et al., 2014) that produces word embeddings using weighted matrix factorization of word co-occurrence probabilities.

Subword Information Skip-gram `si-skip` learns representations for *n*-grams of various lengths, and words are represented as sums of *n*-gram representations (Bojanowski et al., 2017). The learning architecture is based on the continuous skip-gram model (Mikolov et al., 2013b), which is trained by maximizing the conditional probability of context words within a fixed window with negative sampling. The model exploits the morphological variations within a language to learn more reliable representations, particularly for rare morphological variants.

2.3 Neural Compositional Models

Several models have been proposed to overcome some of the weaknesses of bag-of-words and additive representations, such as lack of structure. We evaluated the following context-sensitive models:

The Paragraph Vector: `doc2vec` distributed memory model (Le and Mikolov, 2014) constructs representations for sentences and paragraphs using a neural feedforward network that maximizes the conditional probability of words within a paragraph given a context window and the paragraph embedding, which is shared for all contexts generated from the same paragraph. After learning word and paragraph embeddings for the training corpus, the model learns representations for new paragraphs by fixing the model parameters and updating the paragraph embeddings using backpropagation. This additional training at inference time considerably increases the time complexity of the model compared to all others in this study.

Gated Recurrent Averaging Network: `GRAN` has been recently introduced to combine the benefits of LSTM networks and averaging, where the weights are computed along with the word and sentence representations (Wieting and Gimpel, 2017). The model is trained using aligned sentences that are assumed to be paraphrases to maximize the similarity of their representations against negative examples. The intuition is to make the averaging operation context-sensitive, resulting in a more powerful construction than simple averaging where all words are equally important. The model was shown to outperform averaging and LSTM models in semantic relatedness tasks.

Skip-Thought: The `skip-th` model is a sequence encoder-decoder trained by projecting sentences into fixed-length vectors, which in turn are used as input to a decoder that is trained to reconstruct surrounding sentences (Kiros et al., 2015), where the encoder and decoder are RNNs with GRU activations (Chung et al., 2014). The model is trained with contiguous sentences extracted from a collection of novels. After training, the model's vocabulary is expanded by learning a linear mapping from pre-trained `CBOW` word embeddings to the vector space of the `skip-th` word embeddings.

Natural Language Inference Encoder: In `inferSent` (Conneau et al., 2017), a bidirectional LSTM encoder with max-pooling is trained jointly with an inference classifier trained on the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015), which is a large manually-annotated dataset of English sentence pairs and their inference labels: {entailment, contradition, neutral}.

Dataset	Pair-wise Similarity			Sentiment Analysis					Newsgroup				TREC
	STS	SICK	MSRP	CR	MPQA	RT-s	Subj	IMDB	REL	SPO	COM	POL	
Train	5,749	4,934	4,076	3,775	10,606	10,662	10,000	25k	1,078	1,604	1,694	1,310	5,452
Test	1,379	4,906	1,725	–	–	–	–	25k	–	–	–	–	500
pos ratio	–	–	0.66	0.64	0.31	0.50	0.50	0.50	0.58	0.51	0.51	0.52	–
l	12	10	23	21	3	21	24	262	82	82	82	92	10

Table 1: Dataset statistics. **Train**: number of samples in the training set. **Test**: number of samples in the test set, if applicable (CV is applied otherwise). **pos ratio**: ratio of positive samples in the test set (or total for datasets with no splits). l: average length of all samples.

3 Evaluation Datasets

To evaluate the text representations, we used them as features in extrinsic supervised and unsupervised tasks that reflect various semantic aspects, which can be grouped in three categories: pairwise-similarity, sentiment analysis, and categorization. A summary of the dataset statistics is in Table 1.[1]

3.1 Pairwise Similarity

Semantic Textual Similarity: the STS benchmark dataset (Cer et al., 2017) includes a collection of English sentence pairs and human-annotated similarity scores that range from 0 (unrelated sentences) to 5 (paraphrases). The dataset includes training, development, and test sets. This task can be performed without supervision by calculating the cosine similarity between two sentence vectors. We also evaluated the models in a supervised settings using linear regression, where the input vector is a concatenation of the element-wise produce $u.v$ and absolute difference $|u - v|$ of each pair $\langle u, v \rangle$.

Sentences Involving Compositional Knowledge: SICK dataset is a benchmark for evaluating compositional models (Marelli et al., 2014). We evaluated the models on the relatedness subtask, which is constructed in a similar manner as STS benchmark.

Paraphrase Detection: This is a binary classification task that involves the identification of paraphrases in similar sentence pairs using the Microsoft Research Paraphrase Corpus, MSRP (Dolan et al., 2004). We evaluated the models in two ways: calculating the cosine similarity between the sentence pairs and classifying them as paraphrases if the similarity is larger than a threshold tuned from the training set. The second approach is to learn a logistic regression classifier using a concatenation of $u.v$ and $|u - v|$.

3.2 Sentiment Analysis and Text Categorization

Sentiment Analysis: We used the following binary classification tasks: **CR** customer product reviews (Hu and Liu, 2004), **MPQA** opinion polarity subtask (Wiebe et al., 2005), **RT-s** short movie reviews (Pang and Lee, 2005), **Subj** subjectivity/objectivity classification task (Pang and Lee, 2004), and **IMDB** full-length movie review dataset (Maas et al., 2011).

Newsgroups: Following the setup in (Wang and Manning, 2012) we used the 20-Newsgroup dataset [2] to extract several binary topic categorization tasks. We processed the datasets to remove headers, forwarded text, and signatures, which results in smaller sentences and paragraphs. We used the following newsgroups for binary classification: **religion** (atheism vs. religion), **sports** (baseball vs. hockey), **computer** (windows vs. graphics), and **politics** (middle east vs. guns). We also trained multi-class classifiers on the 8 newsgroups.

Question Classification: We used the TREC 10 coarse question categorization task [3] which categorizes questions into 6 classes: human (HUM), entity (ENTY), location (LOC), number (NUM), description (DESC), and abbreviation (ABBR).

[1] Evaluation scripts and data can be downloaded from: https://github.com/h-aldarmaki/sentence_eval

[2] http://qwone.com/∼jason/20Newsgroups/

[3] http://cogcomp.org/Data/QA/QC/

4 Experimental Setup

4.1 Training Data

We trained the unsupervised word embedding models `CBOW`, `GloVe`, and `si-skip` on a set of ~ 7 million sentences extracted from the English Wikipedia and Amazon movie and product reviews (He and McAuley, 2016). We also trained the Paragraph Vector (`doc2vec`) model on this dataset, and initialized the word embeddings using the `si-skip` pre-trained word embeddings above. While better results overall could be obtained using pre-trained word embeddings trained with much larger text corpora, we used this medium-size corpus to evaluate the various models consistently and reduce model variability due to data and vocabulary coverage.

We used the publicly available pre-trained `GRAN`[4] and `skip-th`[5] models, which require training with special types of datasets: paraphrase collections, and contiguous text from books, respectively. To ensure a fair evaluation, we only compared these models against binary bag-of-words and equivalent word embeddings. The word embeddings within the `GRAN` model were initialized with `PARAGRAM-SL999` word vectors (Wieting et al., 2015b), so we used them as an evaluation baseline for `GRAN`. We compared `skip-th` against the `CBOW` embeddings that were used to expand the vocabulary, which account for most words in the final model's vocabulary. We used the pre-trained `inferSent` model [6] which uses pre-trained GloVe word embeddings [7]. We also experimented with the post-trained word embeddings for each model with similar results, so we omitted them for brevity.

4.2 Training Settings

We trained the unsupervised word embedding models using the optimal parameters recommended for each model. The hyper-parameters in `doc2vec` were set according to the recommendations in (Lau and Baldwin, 2016). For the supervised sentiment classification and text categorization tasks, we trained and tuned linear SVM models using grid search for datasets that include train/dev/test splits, and nested cross-validation otherwise. We also experimented with kernel SVMs but didn't observe notable differences in the results.

5 Evaluation Results

5.1 Pairwise Similarity Evaluation

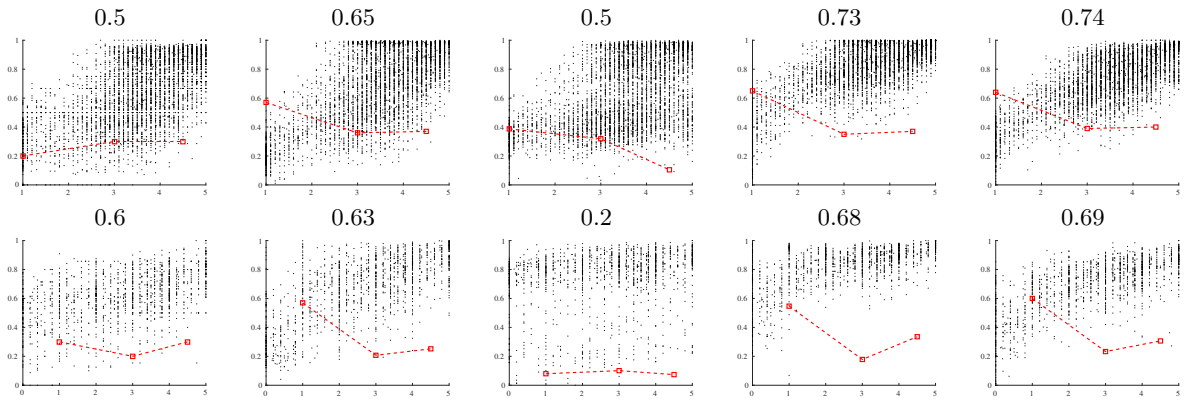


Figure 1: Scatter plots of normalized gold scores in the x axis vs. (a) word overlap (%) and (b - e) cosine similarity using various models. Top: SICK. Bottom: STS Benchmark. Pearson ρ plotted in red for $score \leq 2$, $score \in (2, 4)$, and $score \geq 4$. Overall Pearson ρ shown at the top. † sif-weighted average of pre-trained Glove vectors used in inferSent.

[4] https://github.com/jwieting/acl2017
[5] https://github.com/ryankiros/skip-ths
[6] https://github.com/facebookresearch/InferSent
[7] https://nlp.stanford.edu/projects/glove

		STS Benchmark ρ		SICK ρ		MSRP accuracy/F1	
		cosine	linear reg.	cosine	linear reg.	cosine	logistic reg.
Binary BOW		0.536	0.606	0.611	**0.761**	66.9/0.765	72.2/0.811
Paragraph Vector (doc2vec)		0.628	0.673	0.654	0.655	68.6/0.797	70.4/0.803
Random	avg	0.558	0.616	0.602	0.669	70.6/0.780	70.8/0.797
	idf	**0.668**	0.665	0.617	0.659	70.0/0.790	69.1/0.791
	sif	**0.666**	0.665	0.628	0.655	70.1/0.786	69.9/0.699
CBOW	avg	0.630	0.672	0.679	0.728	71.9/0.815	72.0/0.807
	idf	**0.697**	0.695	0.678	0.712	71.8/0.815	72.3/0.809
	sif	**0.683**	0.686	0.690	0.715	72.2/0.814	71.4/0.804
GloVe	avg	0.336	0.574	0.602	0.694	68.9/0.807	71.0/0.810
	idf	0.540	0.656	0.624	0.685	71.3/0.818	73.4/0.820
	sif	**0.685**	0.665	**0.701**	0.695	71.8/0.809	72.1/0.811
si-skip	avg	0.608	0.690	0.684	0.730	72.1/0.817	71.9/0.895
	idf	**0.683**	**0.714**	**0.702**	0.715	69.3/0.809	70.3/0.815
	sif	**0.694**	**0.721**	**0.716**	0.721	70.6/0.804	70.6/0.802
GRAN		**0.747**	**0.747**	0.715	**0.756**	71.3/0.817	72.3/0.812
Pre-trained PARAGRAM-SL999†	avg	0.564	0.690	0.694	0.746	71.8/0.818	73.2/0.816
	idf	0.711	**0.733**	**0.723**	**0.756**	72.1/0.817	73.3/0.817
	sif	0.716	**0.722**	**0.733**	**0.765**	73.4/0.822	72.0/0.809
skip-th		0.213	**0.729**	0.498	**0.811**	62.3/0.761	73.0/0.812
Pre-trained CBOW†	avg	0.631	0.695	**0.727**	0.758	70.3/0.813	73.2/0.813
	idf	**0.674**	**0.708**	**0.710**	0.731	69.6/0.809	71.3/0.807
	sif	**0.686**	**0.707**	**0.727**	0.737	69.8/0.809	70.9/0.803
inferSent		**0.692**	**0.773**	**0.744**	**0.865**	0.697/0.806	**0.746/0.827**
Pre-trained GloVe†	avg	0.497	0.655	0.687	0.753	0.711/0.818	0.732/0.817
	idf	0.606	0.688	0.696	0.736	0.688/0.809	0.711/0.804
	sif	**0.679**	0.699	0.729	0.749	0.709/0.816	0.708/0.804

Table 2: Pearson ρ for STS Benchmark and SICK relatedness, and Accuracy%/F1 for MSR Paraphrase detection. Results are shaded according to their statistical significance using the Williams test (Graham and Baldwin, 2014) with $\alpha = 0.05$. † pre-trained vectors used in the model above.

Table 2 shows the performance of the various models in the pair-wise similarity tasks. We highlight the best performance in each block; differences within the same shade are not statistically significant. Among the word embedding models, the subword skipgram `si-skip` achieved the best overall performance. For all models except `si-skip`, simple averaging performed poorly in semantic relatedness tasks, especially in the unsupervised setting, while `sif`-weighting generally outperformed `idf`-weighting (the improvement is most evident for `GloVe`). A similar trend is observed with random word vectors, which performed on par with `doc2vec`.

The binary bag-of-words model performed particularly well in the supervised SICK task. The performance of binary and random vectors can be explained by the high correlation between the percentage of overlapping words and the similarity scores as seen in Figure 1. We also highlighted the Pearson correlation coefficients for the following subsets of relatedness scores: $A = \{score \leq 2\}$, $B = \{score \in (2, 4)\}$, and $C = \{score \geq 4\}$. The overall Pearson correlation mostly reflects the performance on the most and least similar pairs, which tend to be the pairs with the highest and lowest word overlap, respectively; within the regions we highlighted, all correlations were relatively low. However, distributed models like `sif` and `inferSent` improved the correlation of A for SICK, and both A and C for STS Benchmark. Table 3 shows some examples of sentence pairs from A and C; while both `sif` and `doc2vec` vectors consistently identified similar concepts (namely food related and competition concepts) regardless of surface similarity, binary scores only reflected the lexical similarity, which resulted in inconsistent scores.

Sentence 1	Sentence 2	Score	Binary	si-sif†	doc2vec	skip-th	infer	gl-sif†
A person **is** frying some food.	There **is** no **person** peeling a potato.	0.25	0.41	0.49	0.41	0.51	0.67	0.71
A woman **is** cutting a fish.	The main **is** slicing potatos.	0.25	0.13	0.46	0.42	0.53	0.65	0.57
Two groups of people **are** playing **football**	**Two** team **are** competing in a **football** match.	0.93	0.52	0.74	0.72	0.60	0.79	0.71
Different **teams are playing** football on the field	Two **teams are playing** soccer	0.70	0.56	0.93	0.89	0.51	0.82	0.91

Table 3: Examples of sentence pairs in SICK and their relatedness scores vs. cosine similarity scores. All scores were normalized to be in [0,1]. Shared words are shown in bold and related words underlined. † sif-weighted average of `si-skipgram` and pre-trained `GloVe`

Sentence 1	Sentence 2	Label
Bashir felt he was being tried by opinion not on the **facts**, Mahendradatta told Reuters.	**Bashir** also **felt he was being tried by opinion** rather than **facts** of law, he added.	1
West Nile Virus-which **is spread** through infected **mosquitoes** -is potentially fatal.	**West Nile** is a bird **virus** that is **spread** to people by **mosquitoes**.	0
SCO says the **pricing** terms for a license **will** not be announced for week	Details on **pricing will** be announced within a few weeks , McBride said	1
Russ Britt **is** the Los Angeles **Bureau Chief** for **CBS.MarketWatch.com.**	Emily Church **is** London **bureau chief** of **CBS.MarketWatch.com.**	0

Table 4: Examples of sentence pairs in MSRP and their labels. Shared words are shown in bold and related words underlined.

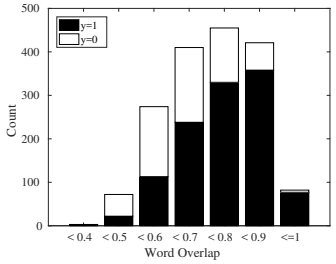


Figure 2: Ratio of paraphrases with increasing word overlap in MSRP.

GRAN outperformed simple averaging in both the STS and SICK tasks, which confirms the results in (Wieting and Gimpel, 2017), but compared with idf and sif averaging, there is no apparent improvement; it only outperformed weighted averaging in the unsupervised STS benchmark. skip-th vectors performed poorly in the unsupervised similarity tasks, but outperformed the pre-trained vectors in the supervised similarity tasks, particularly in SICK.

The low variance of the performance in the paraphrase detection task also reflects the overall correlation between word overlap and the likelihood of being a paraphrase as seen in Figure 2; for difficult cases, as in the examples in Table 4, the overall similarity is not a good indication of being a paraphrase. Significant improvements in this task may require more nuanced features as in (Ji and Eisenstein, 2013).

5.2 Evaluation on Sentiment Analysis and Categorization

		Sentiment Analysis					Text Categorization					
		CR	mpqa	RT-s	subj	imdb	rel	spo	com	pol	mult	TREC
Binary BOW		77.0/0.821	85.9/0.756	74.8	89.5	84.1	66.2/0.710	85.7	78.2	81.6	72.2	**89.8**
Unigram NBSVM		80.5	85.3	78.1	**92.4**	**88.3**	**73.2**	**93.5**	**86.7**	**91.9**	**93.4**	**89.8**
Paragraph Vector (doc2vec)		76.6/0.831	82.4/0.688	78.6	89.9	87.8	68.9/0.751	**89.6**	82.3	**90.6**	**76.1**	59.6
si-skip	avg	81.3/0.857	**87.2/0.778**	78.8	**91.6**	**88.0**	67.1/0.759	**87.8**	80.3	87.2	74.4	79.6
	idf	80.2/0.849	86.2/0.765	78.5	**91.0**	87.0	69.1/0.755	**88.3**	81.9	**89.7**	**75.0**	70.4
	sif	80.6/0.852	**86.7/0.774**	78.6	**91.0**	**88.2**	69.1/0.765	**88.8**	81.0	**89.4**	74.7	71.8
CBOW	avg	81.6/0.859	86.1/0.759	78.1	**91.0**	87.2	63.6/0.745	74.9	78.2	84.7	66.6	**82.8**
	idf	81.2/0.856	86.0/0.761	77.7	90.5	87.3	65.8/0.745	78.6	79.8	85.6	68.2	77.4
	sif	80.8/0.852	85.7/0.756	77.8	90.2	87.4	65.6/0.742	80.0	79.5	85.	68.3	76.2
GloVe	avg	80.7/0.851	85.4/0.745	77.6	**91.0**	87.3	67.1/0.748	82.1	78.8	86.5	69.3	79.8
	idf	80.7/0.851	85.8/0.756	77.8	90.8	87.5	65.5/0.725	85.2	77.7	87.6	70.9	72.4
	sif	80.6/0.850	85.4/0.751	77.6	90.7	87.4	66.8/0.736	85.8	78.5	87.5	70.9	72.0
Random	avg	70.7/0.779	74.2/0.413	61.9	77.3	74.0	55.8/0.634	71.4	72.8	69.4	47.	70.2
	idf	68.6/0.763	74.1/0.423	61.1	72.7	74.2	58.9/0.654	74.4	72.7	72.9	47.2	57.0
	sif	69.2/0.770	72.7/0.369	61.5	69.6	74.5	59.4/0.655	73.0	72.1	72.6	47.4	54.8
GRAN		78.4/0.838	86.6/0.769	75.1	88.5	83.1	66.0/0.753	90.6	80.8	88.5	73.2	60.4
pre-trained PARAGRAM-SL999†	avg	79.8/0.845	**87.8/0.794**	75.9	**89.6**	**84.5**	65.3/0.721	89.8	78.9	87.5	72.5	**83.2**
	idf	79.1/0.840	**87.4/0.791**	75.8	**89.3**	84.1	68.2/0.737	90.3	79.8	89.0	74.1	74.6
	sif	78.4/0.838	86.6/0.769	75.1	88.5	83.1	66.0/0.753	90.6	80.8	88.5	73.2	60.4
skip-th		80.4/0.851	87.0/0.78	76.4	**93.4**	81.8	65.5/0.736	70.4	69.4	81.5	60.1	**88.2**
Pre-trained CBOW †	avg	79.9/0.847	88.2/0.800	77.5	90.5	85.6	64.8/0.751	86.6	**79.5**	**85.9**	70.7	80.0
	idf	79.6/0.844	87.9/0.797	77.2	90.0	85.6	**68.4/0.766**	**87.5**	**80.5**	**85.8**	**72.6**	72.2
	sif	79.2/0.842	87.7/0.794	77.0	89.7	85.8	**67.7/0.761**	**87.8**	**81.3**	**86.9**	**72.9**	74.2
inferSent		**83.0/0.867**	88.5/0.811	77.1	91.0	**86.4**	68.5/0.731	88.6	80.9	85.2	74.4	**88.6**
Pre-trained GloVe †	avg	80.6/0.851	87.9/0.793	77.1	90.9	85.7	69.5/0.759	90.7	**83.8**	**88.3**	75.8	82.2
	idf	79.8/0.844	87.4/0.788	77.2	90.1	85.5	69.7/0.757	89.8	**83.0**	**88.8**	76.9	75.4
	sif	79.6/0.842	87.2/0.785	77.5	90.0	85.5	67.8/0.742	89.7	**83.3**	**88.7**	76.7	77.0

Table 5: Accuracy % or acuracy/F1 (for unbalanced datasets) on sentiment and topic categorization tasks. Results are shaded according to their statistical significance using a two-tailed significance test with $\alpha = 0.05$. † pre-trained word embeddings used in the model above

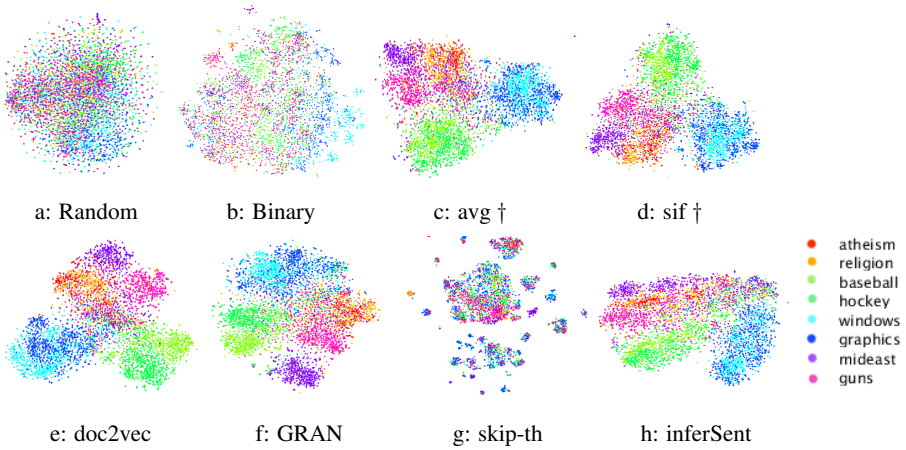


Figure 3: t-SNE visualizations of vectors on the 20-Newsgroup datasets. † `avg` and `sif` using `si-skip`.

Table 5 shows the performance in sentiment analysis and categorization tasks. Unlike in pair-wise similarity, random vectors underperformed the NBSVM and distributed models by a large margin. This underscores the importance of global and distributed features in these tasks. `si-skip` outperformed other word embedding models, but we observe no advantage for weighted vs. unweighted averaging. In TREC question classification and the subjectivity benchmarks, `avg` performed significantly better than both `idf` and `sif` weighted averaging. `skip-th` vectors also significantly outperformed the pre-trained vectors in these two tasks, and underperformed in all others. We surmise that the syntactic features conveyed in frequent function words and the overall structure encoded by the LSTM network in `skip-th` may provide useful clues for these two classification tasks. `inferSent` achieved the highest accuracy in CR sentiment task, and on par with `skip-th` and NBSVM in TREC, and it slightly underperformed the averaging models in Newsgroup categorization. In the next section, we analyze the Newsgroup and TREC datasets to shed light on intrinsic characteristics that may explain some of the performance variance.

6 Qualitative Analysis

Figure 3 shows t-SNE visualizations of the Newsgroup datasets using the various compositional models, including random and binary vectors. While random and binary vectors could identify shallow similarities between sentences as in STS tasks, they failed to do so in a globally cohesive manner. The random vectors also introduced noise in the representations, which resulted in a rather uniform vector space. All other models, except `skip-th`, clearly separated at least three regions that correspond to the categories **sport**, **computer**, and **religion/politics**. Smaller clusters with consistent labeling can also be identified with minimal separation between the clusters.

Table 6 shows examples of nearest neighbors using some of the models. `skip-th` vectors seem to be clustered more by structure than semantic content, unlike the `doc2vec` and `sif` models. To quantify these differences, we applied k-means clustering using $k = 3$ and $k = 8$, and calculated the clustering purity for each model as follows:

$$P(C, L) = \frac{1}{N} \sum_{k} \max_{\mathrm{J}} |c_k \cap \ell_j| \tag{4}$$

where $C = \{c_1, ..., c_K\}$ is the set of clusters, $L = \ell_1, ...\ell_K$ is the set of labels, and N the total number of samples. As shown in Table 7, using `doc2vec` and the averaging models, including `GRAN`, k-means successfully separated the 3 categories, with `doc2vec` and `sif` outperforming in both the fine-grained and coarse clustering. `skip-th` clusters, on the other hand, did not correspond with the correct labels, underperforming binary and random features, which explains its relatively low performance in text categorization tasks. `inferSent` achieved higher purity than binary, random and `skip-th` vectors but lower than the other models, particularly with $k = 3$.

sif †	And they work especially well when the **Feds** have **cut off your utilities.** The Dividians did not have that option after the **FBI cut off their electricity**. Not when the **power has been cut off** for weeks on end.	**What** does this **bill** do? And **Bill** James is not? Do you own " the Bill James players rating book"? **Who** has to consider it ? The being that does the action? I am still not sure I know what you are trying to say.
doc2vec	And they work especially well when the **Feds** have **cut off your utilities**. The Dividians did not have that option after the **FBI cut off their electricity**. Can the **Feds** get him on tax evasion ? I do not remember hearing about him running to the Post Office last night.	I did not **claim** that our system was **objective**. Did I **claim** that there was an absolute morality , or just an **objective** one? I have just spent two solid months **arguing** that no such thing as an **objective** moral system exists.
skip-th	**What** does this bill do ? **Where** do I get hold of these widgets ? **What** gives the United States the right to keep Washington D.C.? **What** makes you think Buck will still be in New York at year 's end with George back?	**I have just spent** two solid months arguing that no such thing as an objective moral system exists. The amount of energy being spent on one lousy syllogism says volumes for the true position of reason in this group. **I just heard** this week that he has started on compuserve flying models forum now.

Table 6: Examples of nearest neighbors in the 20-Newsgroup dataset. †`sif` using `si-skip`.

Model	Newsgroup		TREC
	k = 3, C=3	k = 8, C=8	k=6, C=6
Random	0.5563	0.2431	0.4424
Binary	0.6236	0.2963	0.444
avg †	0.8465	0.3969	0.4481
sif †	**0.8776**	0.4523	0.4037
doc2vec	**0.8625**	**0.4967**	0.3855
skip-th	0.4471	0.1854	0.3896
GRAN	0.8227	0.3553	0.3514
inferSent	0.6562	0.3801	0.4424

Table 7: Clustering purity measure with coarse categories (sports, computers, religion/politics) and the original 8 categories for the Newsgroup dataset, and 6 categories for TREC. †`avg` and `sif` using `si-skip`.

Figure 4 shows t-SNE visualizations of the questions in TREC training set. While all models identified some of the categories, like **HUM** and **LOC**, the `skip-th` and binary vectors appears to be more cohesively clustered by type than the other models. The question types are scattered in multiple smaller clusters, however, which explains why k-means clustering resulted in lower purity scores than `doc2vec` and `averaging` with $k = 6$. In Figure 5, purity results with various k are plotted. While purity is expected to increase with larger k, the rate of increase is much higher for `skip-th` than all other models, including binary features. This is consistent with the t-SNE visualization which shows several consistent clusters with `skip-th` that are larger than the binary clusters. `inferSent`'s performance was on par with `avg`, which is slightly lower than binary and `skip-th`, although the performance in the supervised setting was equivalent.

Table 8 shows nearest neighbors to the question "What country do the Galapagos Islands belong to?" using the various models. The averaging model clustered questions about islands; we observed similar behavior using weighted averaging, `doc2vec` and `GRAN`. On the other hand, `skip-th` clustered questions that start with "what country", which happens to be more suitable for identifying the **LOC** question type. Using binary vectors, questions that include the words "What" and "country" were clustered together, which do not necessarily correspond to the same question type. `inferSent` vectors seem to be clustered by a combination of semantic and syntactic features.

6.1 Discussion and Conclusions

In this study, we attempted to identify qualitative differences among the compositional models and general characteristics of their vector spaces that explain their performance in downstream tasks. Identifying the specific features that are most useful for each task may shed light on the type of information they encode and help optimize the representations for our needs. Word vector averaging performed reasonably well in most supervised benchmarks, and weighted averaging resulted in better performance in unsupervised similarity tasks outperforming all other models. Using the subword skipgram model for word embeddings resulted in better representations overall, particularly with `sif` weighting. The only model that performed on par with or slightly better than weighted averaging in unsupervised STS was

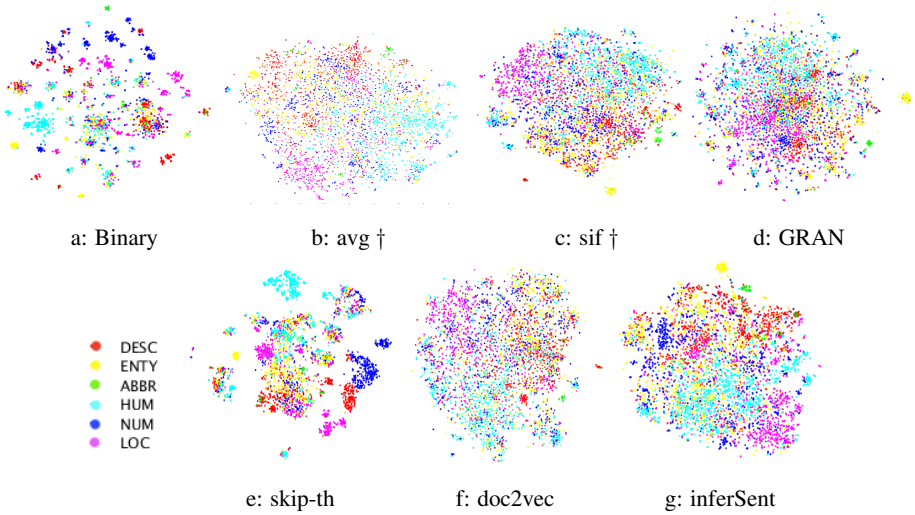


Figure 4: t-SNE visualizations of vectors on the TREC dataset. † `avg` and `sif` using `si-skip`.

avg†	What currents affect the area of the Shetland Islands and Orkney Islands in the North Sea?
	What two Caribbean countries share the island of Hispaniola?
Binary	What is a First World country?
	What is the best college in the country?
skip-th	What country is the worlds leading supplier of cannabis?
	What country did the Nile River originate in?
	What country boasts the most dams?
inferSent	What country did the ancient Romans refer to as Hibernia?
	How many islands does Fiji have?
	What country does Ileana Cotrubas come from ?

Table 8: Nearest neighbors to "What country do the Galapagos Islands belong to? " in TREC. †`avg` using `si-skip`

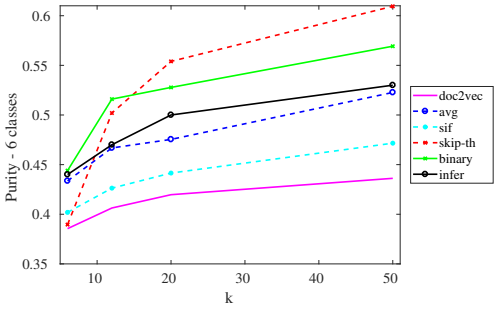


Figure 5: Clustering purity with increasing k on TREC. [`avg` and `sif` for `si-skip`]

`inferSent`. All models achieved higher correlation scores in the supervised STS evaluation, including `skip-th`, which performed poorly in the unsupervised setting. This suggests that at least some of the features in the vector space encode the semantic content and the remaining features are superfluous or encode structural information.

`doc2vec` and `GRAN` representations were qualitatively similar to `idf` and `sif` vectors where sentences/paragraphs were clustered by topic and semantic similarity. `skip-th` vectors, on the other hand, seemed to prominently represent structural rather than semantic features, which makes them more suitable for supervised tasks that rely on sentence structure rather than unsupervised similarity or topic categorization. `inferSent` vectors performed consistently well in all evaluation benchmarks, and a qualitative analysis of the vector space suggests that the vectors encode a balance of semantic and syntactic features. This makes `inferSent` suitable as a general-purpose model for sentence representation, particularly in supervised classification. For topic categorization, none of the compositional models outperformed the NBSVM baseline, which achieved significantly higher accuracies in all supervised topic categorization tasks. However, the distributional models, particularly weighted averaging, are more suitable in unsupervised or low-resource settings since sentences tend to be clustered cohesively by topic similarity and semantic relatedness.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2017)*.

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

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*.

Yvette Graham and Timothy Baldwin. 2014. Testing for significance of increased correlation with human judgment. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 172–176.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on World Wide Web*.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. pages 1367–1377.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 891–896.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, Roberto Zamparelli, et al. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *LREC*, pages 216–223.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *ICLR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. 2014. Evaluating neural word representations in tensor-based compositional settings. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271.

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

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Benjamin Riedel, Isabelle Augenstein, Georgios P Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the fake news challenge stance detection task. *arXiv preprint arXiv:1707.03264*.

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 298–307.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*.

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

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

John Wieting and Kevin Gimpel. 2017. Revisiting recurrent networks for paraphrastic sentence embeddings.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015a. Towards universal paraphrastic sentence embeddings. *ICLR*.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015b. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*.

Using Formulaic Expressions in Writing Assistance Systems

Kenichi Iwatsuki[†]
[†]The University of Tokyo
7-3-1 Hongo, Bunkyo-ku,
Tokyo, Japan
iwatsuki@nii.ac.jp

Akiko Aizawa[‡†]
[‡]National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku,
Tokyo, Japan
aizawa@nii.ac.jp

Abstract

Formulaic expressions (FEs) used in scholarly papers, such as '*there has been little discussion about*', are helpful for non-native English speakers. However, it is time-consuming for users to manually search for an appropriate expression every time they want to consult FE dictionaries. For this reason, we tackle the task of semantic searches of FE dictionaries. At the start of our research, we identified two salient difficulties in this task. First, the paucity of example sentences in existing FE dictionaries results in a shortage of context information, which is necessary for acquiring semantic representation of FEs. Second, while a semantic category label is assigned to each FE in many FE dictionaries, it is difficult to predict the labels from user input, forcing users to manually designate the semantic category when searching. To address these difficulties, we propose a new framework for semantic searches of FEs and propose a new method to leverage both existing dictionaries and domain sentence corpora. Further, we expand an existing FE dictionary to consider building a more comprehensive and domain-specific FE dictionary and to verify the effectiveness of our method.

Title and Abstract in Japanese

執筆支援システムにおける定型表現の利用

論文中に用いられる定型表現（formulaic expressions）とは，例えば'there has been little discussion about'などであり，非英語母語話者の英語論文執筆に有益である。しかしながら，定型表現辞書をその都度参照して適切な表現を探すのは容易ではない。そこで本研究では，計算機による定型表現辞書の意味検索に取り組む。まず，本タスクにおける課題として以下の2つを挙げる。既存の定型表現辞書に収録されている例文が少なく意味表現の獲得に必要な文脈情報が不足していること，多くの定型表現辞書では意味カテゴリごとに定型表現が分類されているが，ユーザの入力から意味カテゴリを予測することが困難であり，ユーザに意味カテゴリを明示する手間が発生することである。 これらの課題を解決するため，本研究では定型表現の意味検索の新たな枠組みを提案し，既存の定型表現辞書を分野毎の英文コーパスと対応づけて利用するための手法を提案する。 さらに，既存の定型表現辞書を拡張し，より網羅性が高く，かつ分野特化型の定型表現辞書の構築を検討し，有効性を検証する。

1 Introduction

Non-native English speakers writing scholarly papers often use the same expressions repeatedly or are not confident in the correctness of their usage of certain wording. Existing computer-based writing assistance systems do not always help them find better expressions than those they already know because such systems search a corpus by using simple pattern-matching based on input keywords (Chang et al., 2015; Chang and Chang, 2015; Jeong et al., 2014; Liu et al., 2016; Mizumoto et al., 2017). For instance, when a user wants to write about previous work and find expressions other than '*little research was done*

Proceedings of the 27th International Conference on Computational Linguistics, pages 2678–2689
Santa Fe, New Mexico, USA, August 20-26, 2018.

on', the only one the user might know, a keyword-based search does not present expressions such as '*there has been little discussion about*' to the user.

A reasonable solution is using dictionaries of formulaic expressions (FEs) (Ellis et al., 2008; Conklin and Schmitt, 2008). FEs used in scholarly papers are helpful for non-native English speakers writing papers in English (Peters and Pauwels, 2015). FEs have a communicative function and thus in FE dictionaries, they are classified into categories on the basis of their functions. Consequently, if a user wants to write about the fact that there is little research on a topic, the user goes to the category '*to show lack of existing research*' and finds expressions such as '*there has been little discussion about*', even if the user has not ever encountered the entire expression.

Despite the usefulness of such FE dictionaries, it is time-consuming for users to choose a category and manually find candidate expressions every time they consult the dictionary. Therefore, the aim of our research is to enable semantic searches of FE dictionaries by computers with incomplete user input. In this paper, we formulate this problem as a prediction task for categories in FE dictionaries. Note that user input can be an incomplete sentence that does not contain a FE because we suppose users who do not think of FEs or who just input some words they have in mind.

We started by identifying two salient difficulties. The first is that very few example sentences are included in FE dictionaries, resulting in the lack of context information necessary for acquiring semantic representation of FEs. The second is that writing assistance systems must automatically predict a semantic category from user input so that users no longer need to designate the category.

In this paper, we propose a method to create a new writing assistance system and verify if it works. First, we extracted example sentences from corpora for a given FE. Second, we checked if having sufficient example sentences improved the prediction of a category from user input that would be an incomplete sentence or phrase. Finally, we classified FEs into categories using context information including full sentences and section labels in addition to the FEs themselves. In our experiment, we used two corpora from different disciplines, one from ACL Anthology (computational linguistics) and the other from PubMed (life science & medicine). For an existing FE dictionary, we used Academic Phrasebank (Morley, 2018).

FEs recorded in the dictionaries are not always suitable for writing assistance systems because many of the expressions do not occur in an actual corpus and differ depending on disciplines. Thus, we aim to build a more comprehensive and more domain-specific FE dictionary by expanding an existing FE dictionary. In this study, we formulate the extraction of FEs from corpora as a sequential labelling problem, which is solved by learning *formulaicity* with an existing dictionary.

The contributions of this paper are as follows. We present a new framework for semantic searches of FEs that can be used when writing scientific papers. We propose leveraging both existing dictionaries and domain-sentence corpora and show that using context information extracted from actual corpora improves the category-prediction accuracy, compared to using the context information originally recorded in the existing FE dictionary. We reformulate FE extraction as a sequential-labelling problem and show that the quality of the FEs extracted with our method is higher than that of those with previous methods.

2 Related Work

2.1 Writing Assistance Systems

Existing writing assistance systems are classified into three types. First, the most direct approach for computer-based writing assistance is that in which user-input sentences are used to retrieve example sentences. Search results are shown with concordances (Wu et al., 2006) or dependency structures (Kato et al., 2006).

Another approach is similar to an input method in which users can input non-alphabetical languages. FLOW (Chen et al., 2012) suggests an English translation from words written in another language. WINGS (Dai et al., 2014) suggests full Chinese sentences and words from pinyin. Full sentences are suggested on the basis of searches for sentences that contain words that are the same as or similar to the input.

The third approach is combined with an authoring system. With this approach, candidate English

expressions that follow user input are listed; then the users can choose one of them (Jeong et al., 2014; Chang et al., 2015; Yen et al., 2015; Chang and Chang, 2015; Liu et al., 2016; Mizumoto et al., 2017). Some systems allow users to specify the categories of FEs. Such categories include the Introduction, Method, Results and Discussion (IMRaD) structure (Jeong et al., 2014), argumentative zone (Teufel, 1999; Chang et al., 2015) and move-step structure (Swales, 1990; Mizumoto et al., 2017). Note that users must designate which category to use.

Overall, existing writing assistance systems adopt keyword-based searches, and thus the number of suggested expressions is limited to ones that contain user-input keywords.

2.2 Definitions of Formulaic Expressions

A survey of definitions of FEs shows that there are three ways of defining them (Durrant and Mathews-Aydınlı, 2011). The first is a 'phraseological' approach. Using this approach, *formulaicity* is definable by non-compositionality of word sequences. However, this definition is not for FEs but for idioms because the semantics of FEs are often compositional. For example, '*have been explored by many researchers*' has a compositional meaning but it is nonetheless a FE. The second is a 'frequency-based' approach. In this approach, frequently co-occurring word sequences are considered FEs. However, noise such as '*is one of the*' cannot be removed. Also, FEs do not always occur frequently. The third one is a 'psychological' approach, which defines FEs as word sequences that are processed and remembered as a whole in the human brain. This seems to be a successful definition of formulaicity, but computers cannot process word sequences using this approach.

Several analyses of FEs exist. Biber et al. (2004) analysed the usage of lexical bundles (continuous word sequences) in an academic context. They defined lexical bundles as 'the most frequent recurring lexical sequences in a register'. Their results showed that lexical bundles are not always syntactically structured. In fact, they often contain some fragments such as '*is based on the*', '*I don't know if*' and '*a little bit of*'.

Along with lexical bundles, Gray and Biber (2013) specifically examined phrase frames: discontinuous word sequences with a slot ' * 'that is filled by any word. The number of lexical bundles used in corpora is larger than that of phrase frames, but examining particularly those occurring in at least five texts, phrase frames are more numerous than lexical bundles. They classified phrase frames into three types: verb-based frames, frames with other content words and function word frames.

Although there are several definitions of FEs, many dictionaries of FEs has been published in which FEs are collected on the basis of intuitive definitions.

2.3 Extraction of Formulaic Expressions

Simpson-Vlach and Ellis (2010) extracted word sequences, three to five words in length, from corpora. Then, with frequency and mutual information, the extracted word sequences were ranked. Their results show that highly frequent word sequences alone cannot be regarded as FEs because they have no distinctive function or meaning. Further, useful expressions are not always highly frequent. In fact, word sequences with high mutual information are rare in corpora because many are subject-specific.

Vincent (2013) decomposed a candidate phrase into the phrasal core and its collocates. The phrasal core is a continuous or discontinuous word sequence occurring with high frequency. Candidate phrases including the core were first identified in a corpus. Then the collocates were sought.

Brooke et al. (2015) used the technique for multi-word expression extraction (Brooke et al., 2014) to find FEs. They split a sentence into parts with a lexical predictability ratio. They pointed out that evaluating newly acquired FEs is difficult because there is no answer dataset.

Aside from studies of FE extraction, several studies have addressed phrase extraction in particular for information extraction or text mining (Zhong et al., 2012; Zhang et al., 2013; Liu et al., 2015). However, these studies specifically addressed characteristic phrases that were informative. Therefore, FEs such as '*in this paper*' were not considered target expressions.

3 Semantic FE Search for Writing Assistance System

3.1 Proposed Searching System

Most existing writing assistance systems seek candidate expressions using keyword-based searches, which causes the problem wherein expressions very different from the input are not presented to users. Consequently, we first consider a semantic searching system that searches for FEs. Then, we propose a scheme for a dictionary of FEs.

The proposed searching system is based on the way people manually use FE dictionaries. Specifically, a user chooses a category that expresses the user's intent. Then, the user picks one of the FEs belonging to that category. Consequently, the proposed searching process consists of two steps. First, the system presumes a user's intention on the basis of written words, which may be an incomplete phrase or sentence. Second, the system searches a dictionary for candidate expressions in the category corresponding to the presumed intention. To choose the correct category, the system must use information related to context that is derived from what a user is writing. Therefore, a resource must include information related to context in which an expression is used. For example, the expression '*further research should be undertaken to investigate*' is not suggested by a keyword-based search with the keywords '*it is future work to investigate*' but by category-based search with the category '*about future work*'.

For this system, the proposed dictionary is formalised as follows. The dictionary consists of several categories, and each category has multiple FEs. Each FE has one or more example sentences. Figure 1 shows an image of the whole dictionary.

In the following subsections we discuss categories and FEs.

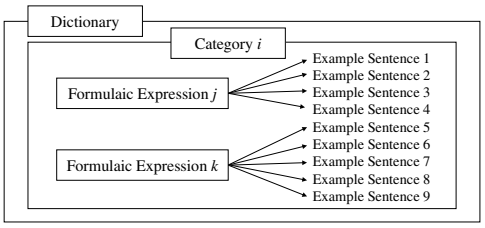


Figure 1: In the dictionary, categories have several FEs, each of which has corresponding example sentences.

3.2 Categorisation Systems

The question remains of what structures of categories are suitable for writing assistance. Presuming that FEs are classified according to their functions, writing assistance systems must anticipate what users want to write from the surrounding context. However, specifically examining scholarly papers shows that the structure of scientific documents is fixed to some degree. Consequently, as long as users write along this fixed structure, there is a good chance that the system can anticipate what the user wants to write if the categories correspond to the paper structure. Therefore, on the basis of the logical structure, categories correspond to both the functions of FEs and the user's intention.

A section-based structure is the simplest, with the Introduction, Method, Results and Discussion (IMRaD) structure adopted for many papers. However, the sections in a paper are so few that too many FEs belong to one category to choose an appropriate one easily. As described herein, we particularly examine move-step structures proposed by Swales (1990). According to Swales' analyses, a paper is composed of several sections, which include moves, each of which has steps (Table 1).

Several analyses have assessed move-step structures in scientific papers. In *Introduction* sections, the Create-A-Research-Space model was found to be adopted in many papers (Swales, 1990; Swales, 2004). Some research has specifically addressed move structures in all sections (Cotos et al., 2015) or *Abstracts* (Lorés, 2004). Other studies have emphasised examining the transition between moves (Ozturk, 2007) and differences in usage across disciplines (Peacock, 2002; Maswana et al., 2015).

Section: Introduction		
Move 1: Establishing a territory	Move 2: Establishing a niche	Move 3: Occupying the niche
Step 1: Claiming centrality	Step 1A: Counter-claiming	Step 1A: Outlining purposes
Step 2: Making topic generalization	Step 1B: Indicating a cap	Step 1B: Announcing present research
Step 3: Reviewing items of previous research	Step 1C: Question-raising	Step 2: Announcing principal findings
	Step 1D: Continuing a tradition	Step 3: Indicating RA structure

Table 1: A move-step structure for the introduction section of a research article (RA) called the Create-A-Research-Space model proposed by Swales (1990).

In our research, we use categories based on move-step structures for writing assistance systems. Specifically, we use the categorisation system that is adopted in Academic Phrasebank made by Morley (2018) because the categorisation of this resource is similar to move-step structures. In Academic Phrasebank there are six sections: Introducing Work, Referring to Sources, Describing Methods, Reporting Results, Discussing Findings and Writing Conclusions and 77 categories such as *Establishing the importance of the topic for the discipline* and *Giving reasons why a method was adopted or rejected*, which roughly correspond to steps. This is suitable for scholarly articles.

3.3 Formulaic Expressions

Considering previous work on the definitions of FEs and our semantic search model, we decide our target FEs as follows. FEs must be expressions that are helpful in writing a paper. Word sequences with an unclear function are excluded. For example, '*is the number of*' and '*the word in the*' do not have clear functions. Other expressions that should be excluded are classified into two types: overly general and overly specific expressions. The former type consists of phrases such as '*on the other hand*' and '*we would like to*'. These might appear to be useful, but they cannot be assigned one category label because '*on the other hand*' can be used in many sections of a paper. Therefore, we chose to exclude them from this research. '*Natural language processing*' or '*in the training dataset*' are examples of the latter type. Overall, in our research, we defined FEs as word sequences whose functions correspond to one category.

4 Methods

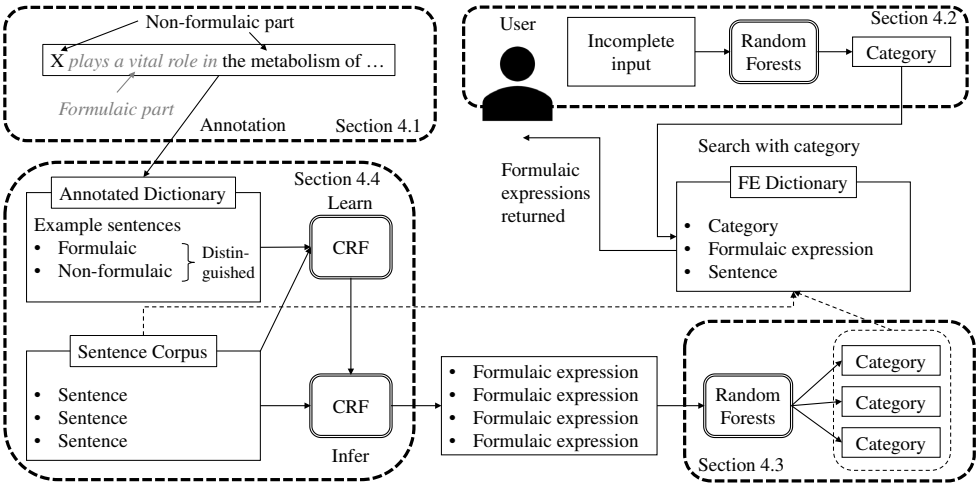


Figure 2: In Section 4.1, annotation is conducted to distinguish the formulaic from the non-formulaic part in each expression recorded in the existing dictionary to build the annotated dictionary. In Section 4.2, incomplete user input is assigned a category label. In Section 4.3, FEs and sentences are classified into categories. In section 4.4, FEs are extracted from a sentence corpus by learning formulaicity from the annotated dictionary.

4.1 Dictionary Annotation

We used two kinds of language resources: an existing dictionary of FEs and a sentence corpus extracted from a group of scientific papers. We first built an annotated dictionary from the existing dictionary.

Many expressions are recorded in the existing dictionary, but the formulaic and non-formulaic parts are not distinguished. For example, '*X plays a vital role in the metabolism of ...*' contains non-formulaic parts such as 'X', 'metabolism' and '...'. This is most problematic when using the existing dictionary because we cannot search corpora for sentences that contain FEs, using only formulaic parts. Therefore, for formulaicity to be learned from the dictionary, we first annotated the formulaic parts in the dictionary. Each word of an expression was manually labelled as either formulaic or non-formulaic (upper-left in Figure 2). Then, non-formulaic parts were removed. After annotation, we extracted sentences containing the annotated FEs from a corpus. Finally, in this annotated dictionary, each categorised FE has some example sentences derived from an actual corpus (Figure 3). All words are lemmatised to avoid inflections. Annotation guidelines are available on our GitHub repository[1].

Category: Stating the purpose of research
Formulaic expression: the objective of this research be to
Example sentences from ACL Anthology:
one of *the objectives of this research is to* make online documents more understandable by paraphrasing unknown …
Thus *the objective of this research is to* experiment with these techniques for Sinhala-Tamil, and identify the best …

Figure 3: In the annotated dictionary, each FE is classified into a category and has several example sentences extracted from a corpus[3].

As described in Section 3.2, we used Academic Phrasebank (Morley, 2018) as a dictionary because its categories are based on move-step structures.

4.2 Prediction of Categories from Incomplete Sentences

In the proposed searching system described in section 3.1, categories are automatically predicted from user input. This is formalised as a classification task from user input into categories.

The prediction is conducted with random forest[4](upper-right in Figure 2).

Each user-input is represented as a vector, which is an average of skip-gram vectors (Mikolov et al., 2013) of each word in the input. The vector representations are learned with a sentence corpus.

The annotated corpus is so divided into a training dataset and a test dataset that sentences in one category in one dataset are almost as many as those in the other dataset.

User input can be an incomplete sentence. Therefore, inputs of two types are prepared as presumed user inputs. One is a sentence not containing a FE, based on the idea that a user wants to find a FE with content words (WithoutFE). The other is a sentence with half or two thirds of the composing words removed. Specifically, we simply select words from every two or three words in a sentence. This simulates a situation where a user comes up with some words but does not compose a sentence (PickedWords). This vector can include part of a FE because users may know some of the words composing the FE.

4.3 Classification of FEs into Categories

In the proposed FE dictionary, FEs are classified into categories. In the previous section classified user input into categories; this section classifies FEs into categories. Therefore, the same methodology as the previous section can be applied to this task, but the input features are different.

We used random forests as a classifier and the input was a vector representation of a FE. The output was a category label (lower-right in Figure 2).

[1] `https://github.com/Alab-NII/FE/blob/master/annotation_guidelines.md`

[3] The example sentences were extracted from two papers: Higashinaka, R., & Nagao, K. 2002. Interactive paraphrasing based on linguistic annotation. In *COLING 2012*. and Hameed, R. A., et al. 2016. Automatic Creation of a Sentence Aligned Sinhala-Tamil Parallel Corpus. In *WSSANLP 2016*.

[4] we use an R package (`https://cran.r-project.org/web/packages/randomForest/randomForest.pdf`).

We used a sentence and a section label in which a FE is used as context information. We tested five kinds of vectors, as described below. Each vector is an average of the vectors of each composing word.

Sentence Vector (Sen) This vector is produced from every word of a sentence that includes the FE.

Sentence Vector without Functional Word Effects (Sen - FW) This vector was presented by Arora et al. (2017). To decrease the effect of functional words, each word embedding is weighed by the coefficient $\frac{a}{a+p}$ where a denotes a parameter set as 0.001 in our experiment and p is word frequency. Afterwards, singular value decomposition is applied.

Sentence Vector without Content Word Effects (Sen - CW) To decrease the content word effects, words with low frequency are removed. When calculating the average vector of each word embedding, words that occur in the sentence corpus less than once per million words are ignored.

Sentence Vector with Section Information (Sen + Sec) A simple one-hot vector which shows that the title of a section in which the FE is used contains *introduction* or *background*, *related work*, *method* or *approach*, *experiment* or *evaluation*, *result*, *discussion* or *conclusion* or *future*, is concatenated to the sentence vector. Some FEs appear in multiple sections, so we assigned the most popular section label in each FE to sentences.

Formulaic Expression Vector (FE) This vector is a baseline made solely from words that compose the FE in the annotated dictionary, which means that context information is excluded.

4.4 Acquisition of new FEs from Corpora

To exploit the intuitive definition of formulaicity in Academic Phrasebank, we implemented a supervised learning method to extract FEs. The FEs recorded in Academic Phrasebank and occurring in real corpora are few in number, so we first paraphrased all nouns, adjectives and adverbs in the annotated dictionary to expand the dataset, using PPDB2.0 (S size) (Pavlick et al., 2015). Subsequently, we attached BIO tags to each sentence in the annotated dictionary. Here, the first word of a FE is assigned a B label. Other words of a FE are tagged with an I label. The rest of the words in the sentence are labelled O. Formulaicity is learned using these sentences. We used conditional random fields (CRFs)[5] to learn and extract FEs. Features were words, part-of-speech tags and word frequency, which was discretised. Then, using the learned model, each word of every sentence in a sentence corpus was given a BIO tag (lower-left in Figure 2). This formulation facilitates the extraction of both continuous and discontinuous FEs.

5 Evaluation

5.1 Datasets

We built two kinds of datasets: an annotated dictionary and a sentence corpus. To build the annotated dictionary, we annotated Academic Phrasebank. Three annotators annotated FEs in five categories to check inter-annotator agreement. The value was 0.699 (Cohen's kappa coefficient). Then, the rest of the FEs were annotated by one annotator. Academic Phrasebank as an existing dictionary had 77 categories, but some categories in which there are fewer than two example sentences extracted from the sentence corpus or in which no FE was found after the annotation were removed. Consequently, 39 categories for ACL Anthology and 45 categories for PubMed were adopted.

We compiled a sentence corpus for which sentences were extracted from papers in ACL Anthology and PubMed. Our corpus consisted of 2,629,115 sentences from ACL Anthology and 2,894,721 sentences from PubMed. Table 2 shows the number of FEs and sentences after the paraphrasing. By paraphrasing FEs and sentences were increased. However, while the number of FEs in the annotated dictionary was originally 854, most of them did not occur in the corpora so that the number was reduced to 225 and 334, respectively.

[5]We use CRF++ (`https://taku910.github.io/crfpp/`).

ACL / PubMed	Before paraphrasing	After paraphrasing
Number of formulaic expressions	225 / 334	352 / 540
Number of example sentences	40,510 / 66,193	49,118 / 89,734

Table 2: Sentence corpora characteristics show that the number of FEs and sentences increased by paraphrasing.

5.2 Prediction of Categories from Incomplete Sentences

To show that sufficient example sentences are necessary for the writing assistance system to predict a category, we classified the presumed user input into a category with a comparison to the original Academic Phrasebank and the annotated dictionary.

Table 3 presents the accuracy, macro precision, macro recall and macro F-measure of the prediction. These results indicate that example sentences recorded in Academic Phrasebank are insufficient for the writing assistance system to predict a category from user input. Consequently, the dictionary requires example sentences extracted from an actual corpus depending on the writer domain.

Dataset	ACL Anthology				PubMed			
	Acc	P	R	F	Acc	P	R	F
WithoutFE w/AP	1.64%	3.52%	6.44%	0.64%	0.69%	2.61%	4.45%	0.52%
PickedWords w/AP	11.2%	6.04%	5.52%	3.14%	3.32%	4.91%	3.72%	2.21%
WithoutFE w/SC	55.9%	77.1%	20.6%	27.3%	47.8%	55.1%	17.8%	20.0%
PickedWords w/SC	81.7%	90.0%	43.2%	54.5%	72.1%	77.9%	37.6%	45.8%

Table 3: Accuracy (Acc), average precision (P), average recall (R) and average F-measure (F) of the classification are shown. The values in upper two rows are the results of the classification in which the training dataset was made from only example sentences in Academic Phrasebank (AP), while those in the lower two rows are the result of the proposed method trained with many more example sentences in a sentence corpus (SC).

5.3 Classification of FEs

The number of FEs was small, so we did a leave-one-out cross validation with FE vectors. The other types of vectors described in Section 4.3 were made from sentences. Eventually, the annotated corpus was divided into training and test datasets in the same way as described in Section 4.2. We also used Academic Phrasebank sentences as a training data for comparison.

The results are presented in Table 4. The results show that the classification of FEs is improved when context information such as example sentences and section information is used. Additionally, removing the effect of highly frequent word adversely affects the classification accuracy, probably because FEs usually consist of frequent words. The results demonstrate that sentences recorded in Academic Phrasebank are insufficient for the writing assistance system to classify FEs into categories.

5.4 Acquisition of new FEs

Using the learned model with CRFs, we extracted FEs from the corpora. After extraction, word sequences that occurred less than twice in the corpus, with a length of less than four words or in which the same words (excluding prepositions) were used twice or more were removed. From ACL Anthology we obtained 2,086 FEs (including 481 discontinuous FEs) with 117,889 example sentences. From PubMed we acquired 1,884 FEs (including 172 discontinuous FEs) with 184,311 example sentences. The extracted FEs are available on our GitHub repository[6].

We manually evaluated each extracted FE, checking if it met the definition we made. We picked FEs in the following way. First, we calculated the smallest edit distances between each FE and all expression

[6] https://github.com/Alab-NII/FE/

	ACL Anthology				PubMed			
Input vector	Acc	P	R	F	Acc	P	R	F
Sen	71.7%	79.0%	28.4%	35.7%	61.1%	51.1%	21.0%	23.3%
Sen - FW	65.1%	77.1%	25.3%	32.7%	54.2%	48.3%	18.6%	20.7%
Sen - CW	72.0%	78.3%	27.7%	35.0%	61.5%	51.7%	21.2%	23.4%
Sen + Sec	77.5%	78.7%	33.3%	39.8%	79.6%	89.4%	60.2%	70.2%
Sen - CW + Sec	77.9%	79.8%	34.6%	41.3%	79.8%	89.2%	60.7%	70.6%
FE	45.2%	26.3%	26.4%	24.1%	48.5%	32.9%	33.8%	32.0%
Majority (baseline)	33.3%	0.85%	2.56%	1.28%	12.3%	0.27%	2.22%	0.49%
Sen - CW + Sec (no SC)	8.95%	7.11%	9.56%	2.98%	4.72%	8.34%	5.81%	2.09%

Table 4: The upper five rows are the results of the classification with the proposed vectors. For comparison, three types of experiments were also conducted: one in which the vectors were made only from formulaic expressions (FE), another is in which they were made only from voting for majority class and the third in which Sen - CW + Sec vectors were learned only with example sentences in Academic Phrasebank, without using a sentence corpus (no SC).

	ACL Anthology		PubMed	
	Cont. FE	Discont. FE	Cont. FE	Discont. FE
Our method	54% (60/111)	32% (8/25)	67% (62/93)	26% (10/37)
High frequency	14% (15/110)	-	14% (14/102)	-
High MI	15% (16/106)	-	12% (12/100)	-

Table 5: Our method outperforms mere extraction of word sequences with high frequency or mutual information (MI).

in the annotated dictionary. Then, we randomly chose FEs in each class of the distance. This is based on the idea that FEs similar to those originally recorded in Academic Phrasebank are more likely to meet the criteria than FEs quite different from them. Three annotators conducted the evaluation with small samples to calculate inter-annotator agreement: 0.800. About 100 FEs were evaluated by one annotator. For comparison, we extracted word sequences with high frequency or high mutual information, which was adopted by Simpson-Vlach and Ellis (2010).

Table 5 presents the ratio of good FEs in each result. Comparing with previous methods, the proposed method extracted continuous FEs more successfully. Discontinuous FEs are more difficult to extract correctly. Some examples are shown in Table 6.

6 Discussion

6.1 Error Analysis of Extracted FEs

We analysed the errors in the extracted continuous FEs. We first divided errors into two categories: spanning and semantics errors. The former occur when one word is unnecessary or when the words are insufficient to make a good FE. Consequently, errors of four types occur, one in which a word to the left or right is unnecessary or missing. The remaining errors are classified into three types: too general, too specific and nonsensical. Overly general and overly specific FEs were explained in Section 3.3. Nonsensical expressions are word sequences that are unlikely to be helpful in writing.

The distribution of errors is presented in Figure 4. In any case nonsensical errors are the most frequently occurring, but the remaining errors differ across datasets and methodologies. Examining the errors made in ACL Anthology using our method, unnecessary words (right) and insufficient words (left) stand out. The former error occurs mainly when 'if' is included in a FE, such as *there be evidence to suggest that if*. The latter error includes phrases such as *be important to note that*. Most missing words are 'it'. However, 'if' is sometimes necessary and 'it' is sometimes unnecessary. For example, when 'if' can be replaced with 'whether' in expressions such as *check if*, appending 'if' to a FE is very helpful.

	ACL Anthology	PubMed
Our method	have(has) shown significant improvement(s) over is(are) a key component in it be possible to * to determine whether *to that of Figure* *has been observed that the thematic role*	previous studies have demonstrated the aim of this study is to investigate it is possible that * may have influence *the number of investigated* *to measure * suffer from*
High freq.	we can see that in this paper we propose *state of the art* *shown in Figure 2* *the words in the*	it is important to play an important role *(Figure 1)* *et al 2010)* *$p < 0.05$)*
High MI	this paper is organized as follows to the best of our knowledge *statistical machine translation (SMT)* *on the other hand* *a large number of*	it is important to play an important role *of this study is to* *et al 2011)* *($p < 0.05$)*

Table 6: Examples of extracted FEs. Expressions that are not regarded as good FEs are written in italics.

Additionally, 'it' as a formal subjective should appear along with a real subject, such as a to-infinitive or a that-clause. However, in the PubMed dataset, insufficient words (left) stand out. The words missing the most are 'it' and 'there'. For example, 'it' should be appended to *be clear from the figure that* and *be important to investigate if* and 'there' to *be significant difference between* and *be some limitation to this study*.

Specifically examining FEs extracted using the existing methods, it is apparent that there are many overly general expressions. These expressions mainly include prepositional and idiomatic phrases such as *on the basis of*, *at the time of*, *on the other hand*, *be more likely to* and *for the purpose of*. A different categorisation system is necessary to make the use of these expressions.

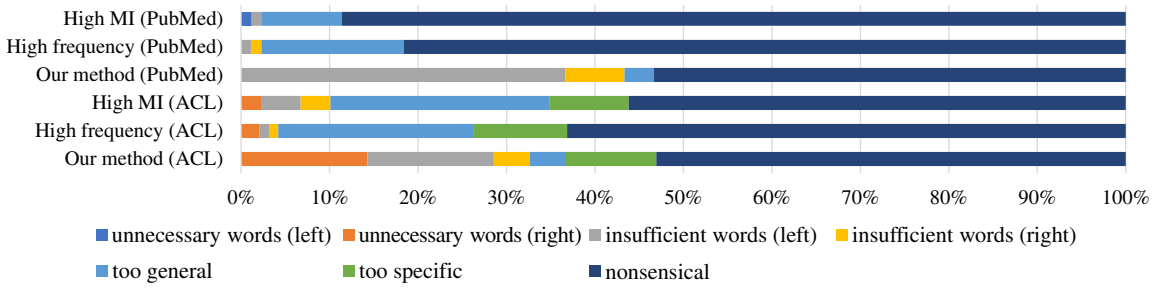


Figure 4: Distribution of error types among results.

6.2 Difference between Domains

For this work, we used two corpora, ACL Anthology and PubMed. Here we discuss the differences in the extracted FEs between the two corpora. First, there were 339 FEs occurring in both corpora (16% of ACL Anthology; 18% of PubMed).

Domain-specific technical terms, such as *natural language processing* or *reactive oxygen species*, are unlikely to be extracted using the proposed method. However, the usage of FEs is shown to differ across domains, which implies that the expressions should be re-ranked according to the users' discipline when candidate expressions are presented to users of the writing assistance system. In addition, it is interesting to note that section information seems more critical in PubMed's classification than in that of ACL.

7 Conclusion

We proposed a new framework for semantic searches of FEs with incomplete user input. We also proposed a method to classify FEs into categories and showed that context information improves the accuracy of the classification. Further, we reformulated FE extraction as a sequential labelling problem and found that this method works well to build a domain-specific FE dictionary.

On the basis of our approach, further research will be conducted to improve the classification and extraction. For classifying FEs, vector representations of a sentence should probably be improved focusing on FE. For acquiring new FEs, an advanced machine learning algorithm should be used to reduce nonsensical FEs and syntactic information would be useful to alleviate the spanning problem.

Acknowledgements

This research was supported by JSPS KAKENHI 18H03297.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*.

Douglas Biber, Susan Conrad, and Viviana Cortes. 2004. If you look at...: Lexical bundles in university teaching and textbooks. *Applied Linguistics*, 25(3):371–405.

Julian Brooke, Vivian Tsang, Graeme Hirst, and Fraser Shein. 2014. Unsupervised multiword segmentation of large corpora using prediction-driven decomposition of n-grams. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*, pages 753–761.

Julian Brooke, Adam Hammond, David Jacob, Vivian Tsang, Graeme Hirst, and Fraser Shein. 2015. Building a lexicon of formulaic language for language learners. In *Proceedings of the 11th Workshop on Multiword Expressions*, pages 96–104.

Jim Chang and Jason Chang. 2015. Writeahead2: Mining lexical grammar patterns for assisted writing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 106–110.

Jim Chang, Hsiang-Ling Hsu, Joanne Boisson, Hao-Chun Peng, Yu-Hsuan Wu, and Jason S. Chang. 2015. Learning sentential patterns of various rhetoric moves for assisted academic writing. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation: Posters*, pages 37–45.

MeiHua Chen, ShihTing Huang, HungTing Hsieh, TingHui Kao, and Jason S. Chang. 2012. Flow: A first-language-oriented writing assistant system. In *Proceedings of the ACL 2012 System Demonstrations*, pages 157–162.

Kathy Conklin and Norbert Schmitt. 2008. Formulaic sequences: Are they processed more quickly than nonformulaic language by native and nonnative speakers? *Applied Linguistics*, 29(1):72–89.

Elena Cotos, Sarah Huffman, and Stephanie Link. 2015. Furthering and applying move/step constructs: Technology-driven marshalling of swalesian genre theory for eap pedagogy. *Journal of English for Academic Purposes*, 19(Supplement C):52–72. 25 Years of "Genre Analysis".

Xianjun Dai, Yuanchao Liu, Xiaolong Wang, and Bingquan Liu. 2014. Wings:writing with intelligent guidance and suggestions. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 25–30.

Philip Durrant and Julie Mathews-Aydınlı. 2011. A function-first approach to identifying formulaic language in academic writing. *English for Specific Purposes*, 30(1):58–72.

Nick C. Ellis, Rita Simpson-vlach, and Carson Maynard. 2008. Formulaic language in native and second language speakers: Psycholinguistics, corpus linguistics, and TESOL. *TESOL Quarterly*, 42(3):375–396.

Bethany Gray and Douglas Biber. 2013. Lexical frames in academic prose and conversation. *International Journal of Corpus Linguistics*, 18(1):109–136.

Senator Jeong, Sejin Nam, and Hyun-Young Park. 2014. An ontology-based biomedical research paper authoring support tool. *Science Editing*, 1(1):37–42.

Y. Kato, S. Matsubara, and Y. Inagaki. 2006. A corpus search system utilizing lexical dependency structure. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*.

Jialu Liu, Jingbo Shang, Chi Wang, Xiang Ren, and Jiawei Han. 2015. Mining quality phrases from massive text corpora. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1729–1744.

Yuanchao Liu, Xin Wang, Ming Liu, and Xiaolong Wang. 2016. Write-righter: An academic writing assistant system. In *Thirtieth AAAI Conference on Artificial Intelligence*, pages 4373–4374.

Rosa Lorés. 2004. On RA abstracts: from rhetorical structure to thematic organisation. *English for Specific Purposes*, 23(3):280–302.

Sayako Maswana, Toshiyuki Kanamaru, and Akira Tajino. 2015. Move analysis of research articles across five engineering fields: What they share and what they do not. *Ampersand*, 2:1–11.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Atsushi Mizumoto, Sawako Hamatani, and Yasuhiro Imao. 2017. Applying the bundle–move connection approach to the development of an online writing support tool for research articles. *Language Learning*, 67(4):885–921.

John Morley. 2018. Academic phrasebank. `http://www.phrasebank.manchester.ac.uk/`.

Ismet Ozturk. 2007. The textual organisation of research article introductions in applied linguistics: Variability within a single discipline. *English for Specific Purposes*, 26(1):25–38.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430.

Matthew Peacock. 2002. Communicative moves in the discussion section of research articles. *System*, 30(4):479–497.

Elke Peters and Paul Pauwels. 2015. Learning academic formulaic sequences. *Journal of English for Academic Purposes*, 20:28–39.

Rita Simpson-Vlach and Nick C. Ellis. 2010. An academic formulas list: New methods in phraseology research. *Applied Linguistics*, 31(4):487–512.

John Swales. 1990. *Genre analysis: English in academic and research settings*. Cambridge University Press.

John Swales. 2004. *Research genres: Explorations and applications*. Cambridge University Press.

Simone Teufel. 1999. *Argumentative Zoning: Information Extraction from Scientific Text*. Ph.D. thesis, University of Edinburgh.

Benet Vincent. 2013. Investigating academic phraseology through combinations of very frequent words: A methodological exploration. *Journal of English for Academic Purposes*, 12(1):44–56.

Jien-Chen Wu, Yu-Chia Chang, Hsien-Chin Liou, and Jason S. Chang. 2006. Computational analysis of move structures in academic abstracts. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 41–44.

Tzu-Hsi Yen, Jian-Cheng Wu, Jim Chang, Joanne Boisson, and Jason Chang. 2015. Writeahead: Mining grammar patterns in corpora for assisted writing. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 139–144.

Bin Zhang, Alex Marin, Brian Hutchinson, and Mari Ostendorf. 2013. Learning phrase patterns for text classification. *IEEE transactions on audio, speech, and language processing*, 21(6):1180–1189.

Ning Zhong, Yuefeng Li, and Sheng-Tang Wu. 2012. Effective pattern discovery for text mining. *IEEE Transactions on Knowledge and Data Engineering*, 24(1):30–44.

What's in Your Embedding, And How It Predicts Task Performance

Anna Rogers, Shashwath Hosur Ananthakrishna, Anna Rumshisky
Department of Computer Science,
University of Massachusetts Lowell, Lowell, MA, USA
{arogers,sha,arum}@cs.uml.edu

Abstract

Attempts to find a single technique for general-purpose intrinsic evaluation of word embeddings have so far not been successful. We present a new approach based on scaled-up qualitative analysis of word vector neighborhoods that quantifies interpretable characteristics of a given model (e.g. its preference for synonyms or shared morphological forms as nearest neighbors). We analyze 21 such factors and show how they correlate with performance on 14 extrinsic and intrinsic task datasets (and also explain the lack of correlation between some of them). Our approach enables multi-faceted evaluation, parameter search, and generally – a more principled, hypothesis-driven approach to development of distributional semantic representations.

1 Introduction

Dense lexical embeddings are the most common distributional semantic representations in both industrial and academic natural language processing (NLP) systems (Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013a; Pennington et al., 2014; Ruppert et al., 2015). They are used in task-specific neural network models, solving such tasks as named entity recognition (Guo et al., 2014), semantic role labeling (Chen et al., 2014), syntactic parsing (Chen and Manning, 2014), and more.

Each year dozens of new models are proposed, each of them with multiple hyper-parameters that may dramatically influence performance (Lapesa and Evert, 2014; Kiela and Clark, 2014; Levy et al., 2015; Lai et al., 2016; Melis et al., 2017). Equally important are the source corpus, its domain, and the type of context (Padó and Lapata, 2007; Levy and Goldberg, 2014a; Li et al., 2017; Lapesa and Evert, 2017). This amounts to an exponential explosion of options in the quest for the best model for a given task.

Ideally, there would be a single intrinsic metric for identifying "good" embeddings – and there are many proposals for such a metric (including word relatedness and analogies). However, none of them have been shown to predict performance on a wide range of tasks, and there is evidence to the contrary (Chiu et al., 2016).

We hypothesize that *different extrinsic tasks may rely on different aspects of word representations*. In that case, the only way to reliably predict what an embedding can do is to know what aspects of language it captures, and what aspects of language are relevant for different tasks.

To that end, we propose *Linguistic Diagnostics* (LD), a new approach to automated qualitative analysis of vector neighborhoods. To the best of our knowledge, this is the first large-scale attempt to identify and quantify the factors that make word embeddings successful with different tasks. We evaluate 60 models (the popular GloVe and Word2Vec with varying vector sizes and 4 types of context), identifying 21 factors that, to varying extent, correlate with the models' performance on 14 extrinsic and intrinsic task datasets. LD scores can be used not only for evaluation, but also for model development and optimization.

LD is implemented in LDT (Linguistic Diagnostics Toolkit), an open-source Python library[1] that offers a wide range of analysis options with corpus-based statistics, psychological association norms, and dictionaries. LDT provides broad lexical coverage thanks to a combination of the English WordNet, Wiktionary, and BabelNet, and is potentially extensible to many languages.

[1] http://ldtoolkit.space

Proceedings of the 27th International Conference on Computational Linguistics, pages 2690–2703
Santa Fe, New Mexico, USA, August 20-26, 2018.

2 Related Work

Perhaps the most popular kind of intrinsic evaluation of word embeddings are the **semantic relatedness** tests (Finkelstein et al., 2002; Bruni et al., 2014; Luong et al., 2013; Radinsky et al., 2011). They rely on the idea that the distance between word vectors should correlate with human judgements of how related the two words are (e.g., *cat* should be closer to *tiger* than to *hammer*). A more sophisticated version of this task is the **semantic similarity** (Agirre et al., 2009; Hill et al., 2015), which basically restricts relatedness to synonymy and co-hyponymy.

This evaluation paradigm has come under fire for methodological reasons (Faruqui et al., 2016; Batchkarov et al., 2016), in particular, due to the unreliability of the "middle" judgments: while *cat* should be closer to *tiger* than to *hammer*, it is not clear whether it should be closer to *lion* or to *tiger* (Gladkova and Drozd, 2016). Furthermore, only 1 out of 10 datasets was a good predictor of performance on sequence labeling tasks (Chiu et al., 2016). The proposal for evaluation via coherence of semantic space (Schnabel et al., 2015) inherits all the problems with relatedness (Gladkova and Drozd, 2016).

There are multiple proposals for "subconscious intrinsic evaluation" (Bakarov, 2018) based on **correlations with psycholinguistic data** such as N400 effect (Van Petten, 2014; Ettinger et al., 2016), fMRI scans (Devereux et al., 2010; Søgaard, 2016), eye-tracking (Klerke et al., 2015; Søgaard, 2016), and semantic priming data (Lund et al., 1995; Lund and Burgess, 1996; Jones et al., 2006; Lapesa and Evert, 2013; Ettinger and Linzen, 2016; Auguste et al., 2017). However, there are no large-scale studies that would show the utility of these methods in predicting downstream task performance. It is also possible that any psychological measure would share the subjectivity problem of relatedness judgments.

The idea behind the **word analogy task** (Mikolov et al., 2013b) is that the "best" word embedding is the one that encodes linguistic relations in the most regular way: simple vector offset should be sufficient to capture semantic shifts such as $France : Paris$ to $Japan : Tokyo$. However, this view of linguistic relations (and analogical reasoning) is oversimplified, and performance on word analogies has also been shown to depend on cosine similarity between source word vectors (Rogers et al., 2017; Linzen, 2016; Levy and Goldberg, 2014b). Furthermore, the original vector offset method is underestimating the amount of semantic information captured by the embedding (Drozd et al., 2016). Last but not the least, analogies also fail to yield results consistent with downstream task performance (Ghannay et al., 2016).

One more line of research could be called **linguistically motivated evaluation**. The idea is that a "good" embedding would be somehow similar to a representation that could be constructed from a gold-standard linguistic resource (Tsvetkov et al., 2015; Tsvetkov et al., 2016; Acs and Kornai, 2016).

Crucially, all these approaches make the same core assumption: that there is *one* feature of a representation that would make it the "best" (the highest correlation with human judgements, the most regular vector offsets, the closest approximation of a linguistic resource, etc.) However, language is a multi-faceted phenomenon, and different NLP tasks may rely on its different aspects – which would doom any one-metric-to-rule-them-all approach. This is the starting point for our solution.

3 LDT: the methodology

Consider two published modifications of the word2vec model, both trained on Wikipedia: the dependency-based embeddings (DEPS) (Levy and Goldberg, 2014a) and FastText (Bojanowski et al., 2017).

Table 1 lists the first 7 nearest neighbors of *color* (as measured by cosine similarity). Both models output the British spelling of the target word (*colour*). However, DEPS also includes derivatives and synonyms, while FastText favors misspellings and compounds, as could be expected of a subword-level model.

Rank	Deps		FastText	
1	colour	0.93	$color	0.75
2	colors	0.72	color…	0.69
3	coloration	0.69	colour	0.69
4	colouration	0.68	color#ff	0.69
5	colours	0.68	color#d	0.68
6	hue	0.66	@color	0.67
7	hues	0.65	barcolor	0.67

Table 1: Top 7 neighbors of *color* in dependency-based and FastText embeddings.

Which of these models is "better"? Without the context of some application, the question is meaningless. There is no theoretical reason why plural forms of nouns would make better/worse neighbors than their synonyms or misspellings.

This is a more meaningful question: *what are the properties of embedding X that could predict its performance on tasks Y and Z?* For example, question answering would likely benefit from synonymy more than morphology induction. Consider that relatedness tests were found to poorly correlate with performance on sequence labeling tasks, but SimLex (Hill et al., 2015) performed better (Chiu et al., 2016). This could be due to its focus on a particular type of semantic relations (synonymy, co-hyponymy), which turned out to be relevant for the labeling tasks.

Our solution is based on "linguistic diagnostic" tests, achieved by large-scale automatic annotation of linguistic, psychological and distributional relations between words vectors and their neighbors. The resulting data can then be used to find what features are useful for what extrinsic tasks. This work is inspired by the BLESS categorization dataset (Baroni and Lenci, 2011) and by evaluation via a set of representative extrinsic tasks (Nayak et al., 2016).

LD analysis starts with sampling the corpus vocabulary, as will be described in Section 4.2. For each word, top n neighbor vectors are extracted from each embedding. Each neighbor undergoes spelling normalization and is paired with the source word for analysis of possible morphological, semantic, distributional and psychological relations between them, as shown in Figure 1. The annotated data is analyzed to produce direct or statistically derived measures of the degree to which a given embedding is characterized by a given factor (e.g. how many synonyms or morphologically related words are neighbors of a given word vector). The exact set of LD factors considered in this study will be described in Section 5.1.

Since linguistic relations (especially semantic relations such as hypernymy and antonymy) cannot yet be classified accurately by purely distributional means[2], LDT relies on the largest freely available lexicographic resources: WordNet (Fellbaum, 1998) and Wiktionary[3], with the option of BabelNet (Navigli and Ponzetto, 2012). Currently, only English is supported, but (thanks to Wiktionary and BabelNet) LDT can be extended to other languages.

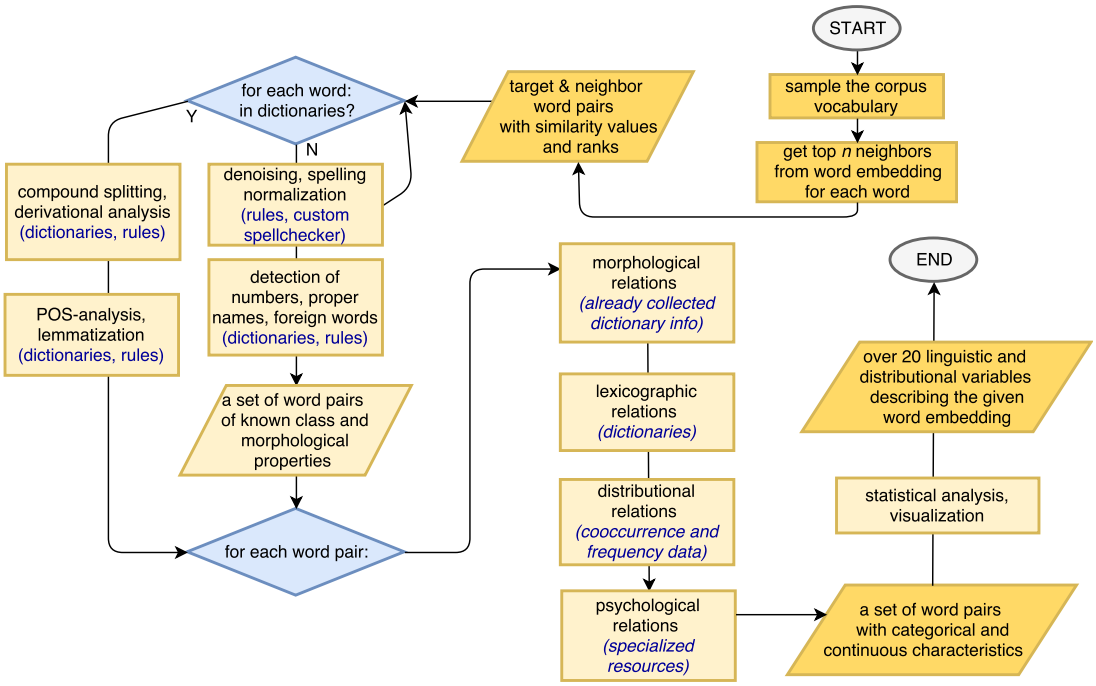


Figure 1: LDT analysis pipeline

[2]For example, the best performing system in the recent CogALex-V shared task (Santus et al., 2016) achieved only 45% accuracy in classifying only 5 semantic relations.

[3]https://en.wiktionary.org

4 Experiment set-up

4.1 Word embeddings

This work explores 3 popular word embedding algorithms: GloVe (Pennington et al., 2014), CBOW, and Skip-Gram (SG) (Mikolov et al., 2013a). The pre-trained vectors we used were published[4] by Li et al. (2017) who experimented with 4 different types of contexts on sequence labeling tasks. Additionally, they provided models with different vector sizes (25, 50, 100, 250, 500). In total, there were 60 models.

Table 2 shows that there are two types of context (linear and dependency-based), and two context representations: bound and unbound. The linear unbound context is the classic bag-of-words context (window size 2). The linear bound context is the same, except that words to the left and to the right are counted separately (Levy and Goldberg, 2014b; Ling et al., 2015). In the "bound" DEPS context (Levy and Goldberg, 2014a), the corpus is syntactically parsed, and only the words that are connected with the target word by some dependency relation are taken into account. Li et al. (2017) extended this idea into the "unbound" DEPS context, where the labels of syntactic roles are ignored.

Context Representation \ Context Type	Linear (Bag-of-Words)	DEPS
unbound	every, non-trivial, has, at	every, non-trivial, has
bound	every/-2, non-trivial/-1, has/+1, at/+2	every/+det, non-trivial+amod, has+nsubj

Table 2: Bound and unbound linear/dependency contexts for the word *program* in the sentence "*Every non-trivial program has at least one bug*". Adapted from (Li et al., 2017).

All embeddings were trained on English Wikipedia (August 2013 dump), with a minimum frequency of 100. After dependency parsing by Stanford CoreNLP (Manning et al., 2014), the corpus was lowercased. Negative sampling was set to 5 for SG and 2 for CBOW, no "dirty" sub-sampling. Distribution smoothing was set to 0.75. SG was trained for 2 epochs, CBOW - for 5, and GloVe - for 30.

4.2 Vocabulary Filtering and Sampling

Fair evaluation must take into account the amount of information that was available during training. It is possible to run LDT on any embeddings, but it yields the most information when the source corpus is available, and it is possible to estimate raw frequencies and cooccurence counts.

The source Wikipedia dump from which the embeddings were produced contains 14,404,885 token types. Only 273,229 of these occur over 100 times, but because of 4 context representations, the vocabulary of the different models is not the same (the DEPS vocabularies are particularly large, up to 5 times as many words). Since LDT methodology is based on the content of vector neighborhoods, to level the playing field for all models we filtered[5] the vocabulary down to 269,860 that were present in all models.

In this study, we focus on the general vocabulary and exclude proper nouns. We use LDT to draw a balanced sample of WordNet lemmas for four parts of speech (nouns, verbs, adjectives, adverbs) in 4 logarithmic frequency bins in the source corpus: 100, 1,000, 10,000, 100,000 (lower boundary inclusive). Following Baroni and Lenci (2011), we control for the polysemy of the words in the sample. For each part of speech at most 30 monosemous and polysemous words were drawn. Polysemy was defined as a word having over 2 meanings in WordNet[6]. The structure of the resulting sample is shown in Table 3.

Note that we also exclude the words belonging to several parts of speech (e.g. *a dog* (noun), *to dog* (verb)) to preserve the morphological class variable. This discards a lot of high-frequency vocabulary, which is why the higher frequency bins for verbs and adjectives were not populated fully. The total number of words in the sample is 908.

[4] `http://vecto.space/data/embeddings/en`

[5] Preliminary experiments showed that the filtering was beneficial to the performance on some tasks, and detrimental to others. The scope of this paper does not permit full investigation of the matter, but the effect was consistent across embeddings.

[6] This measure is not perfect, as numbers of senses in WordNet do not necessarily correspond to the number of senses in which a given word is used in Wikipedia, but it does provide a useful estimate.

4.3 Running LDT

We extract the 1,000 nearest neighbors for each word in the above sample. While most words will not have 1,000 meaningful relations, high-frequency words might have more than that. For example, a SG model with bound DEPS context has *rather* as the neighbor of *quite* at rank 920.

Frequency bin	Nouns	Verbs	Adj.	Adv.
100∼1,000	30 / 30	30 / 30	30 / 30	30 / 30
1,000∼10,000	30 / 30	30 / 30	30 / 30	30 / 30
10,000∼100,000	30 / 30	21 / 30	30 / 30	30 / 30
100,000 >	30 / 30	2 / 29	22 / 24	30 / 30

Table 3: The number of words sampled in each frequency bin per POS (monosemous / polysemous).

In total, 908,000 word pairs were processed for each of 60 embeddings. We limited the used resources to Wiktionary and WordNet, since BabelNet's maximum usage quota for research purposes (50,000 queries per day) would not be sufficient for this large-scale experiment.

The dictionaries covered 76,946 (28.51%) of all the neighbor words; another 124,511 (46.14%) were detected as proper nouns (as could be expected of a Wikipedia corpus). Thus, only 25.35% of the total vocabulary was not covered by LDT.

4.4 Extrinsic tasks

Each of 60 embeddings was evaluated on 8 extrinsic tasks[7]. The selection is similar to what Nayak et al. (2016) proposed as a representative suite of tasks for evaluation purposes. We also follow the recommendation of Nayak et al. (2016) in selecting simpler models for evaluation: more complex models often yield better accuracy, but they could smooth out the performance of different word embeddings and also raise the question of whether the gains are due to the model or the embeddings.

The morphological and syntactic information is targeted by two sequence labeling tasks: **POS-tagging** and **chunking**. We use the CoNLL 2003 shared task dataset (Tjong Kim Sang and De Meulder, 2003), following the method by Li et al. (2017). The model is a softmax classifier on the window-based concatenation of word embeddings of every training example (window size 3, 20 training epochs).

Semantic information at the word level is targeted by one more CoNLL 2003 shared task: **named entity recognition (NER)**, evaluated in the same way as POS-tagging and chunking. We also consider the task of **multi-way classification of semantic relations (Relation class.)** between pairs of nominals in the SemEval 2010 task 8 dataset (Hendrickx et al., 2010). The model we use is similar to the model by Zeng et al. (2014): a CNN equipped with word and distance embeddings.

Next, we have 3 tasks relying on how the word embeddings encode semantic information, and to what degree individual word vectors can be combined into an accurate sentence representation. The **sentence-level sentiment polarity classification (Sentiment (sent.))** task is tested with the MR dataset of short movie reviews (Pang and Lee, 2005). Binary classification is performed by a simplified version of the model proposed by Kim (2014).

We also add the **document-level polarity classification (Sentiment (text))** with the Stanford IMDB movie review dataset (Maas et al., 2011). Polarity is harder to estimate at the document than at the sentence level, because sentiment is more likely to be mixed. The task is performed with a single layer LSTM with 100 hidden units.

The **classification of subjectivity and objectivity (Subjectivity class.)** is tested on Rotten Tomato user review snippets vs official movie plot summaries (Pang and Lee, 2004). We follow the method by Li et al. (2017), employing a simple logistic regression model for the binary classification task. The input sentences are represented as a sum of their constituent word vectors.

Finally, the **natural language inference** task is represented with the SNLI dataset (Bowman et al., 2015). Similarly to the original proposal, we use two separate LSTMs to get a representation of the premise and the hypothesis using the last hidden state. The two hidden representations are merged and fed into a 50-unit dense layer, over which 3-class classification with softmax is performed.

[7] https://github.com/shashwath94/Extrinsic-Evaluation-tasks

4.5 Intrinsic tasks

Section 2 mentioned the reported lack of correlation between the performance of word embedding models on relatedness and sequence labeling tasks (Chiu et al., 2016). Ghannay et al. (2016) also report that the best-performing embedding on sequence labeling and mention detection tasks is not necessarily the embedding that performs the best on analogy and relatedness datasets. However, these studies have a limited selection of word embeddings (amounting to 9 and 5 data points, correspondingly). Crucially, they also focus on the same sequence labeling CoNLL tasks.

We explore the problem with our set of 60 embeddings, and a wider selection of extrinsic tasks. The intrinsic task datasets are WordSim353 (Finkelstein et al., 2002), together with its split into similarity and relatedness sections (Agirre et al., 2009), RareWords (Luong et al., 2013), MTurk (Radinsky et al., 2011), MEN (Bruni et al., 2014), and also the SimLex999 (Hill et al., 2015) similarity dataset. For the analogy task we use BATS dataset (Gladkova et al., 2016), which is currently the largest analogy dataset for English. We report separate scores for inflectional and derivational morphology, lexicographic and encyclopedic semantics, and the average of all categories.

The evaluation on similarity and relatedness datasets is performed as Spearman's correlation with the human judgement scores. The evaluation on analogies is performed with the state-of-the-art LRCos method (Drozd et al., 2016).

5 Results

5.1 Correlation analysis

In this study we experimented with 21 morphological, lexicographic, psychological, and distributional factors of word vector neighborhoods. For better readability, they are presented in Figure 2 together with their correlations with each other and the performance on 14 extrinsic and intrinsic task datasets (based on the data from 60 GloVe and Word2Vec embeddings described above).

Binary relations (e.g. synonymy is either detected or not) were quantified as a simple count of all cases of that relation in all target:neighbor pairs for each embedding. Directed lexicographic relations (hypernymy, hyponymy, meronymy) are counted when the target word is e.g. a hypernym of the neighbor. Continuous variables are broken down into bins, the size of which is chosen empirically: e.g. instead of frequency of the target word we count the number of low-frequency or high-frequency neighbors.

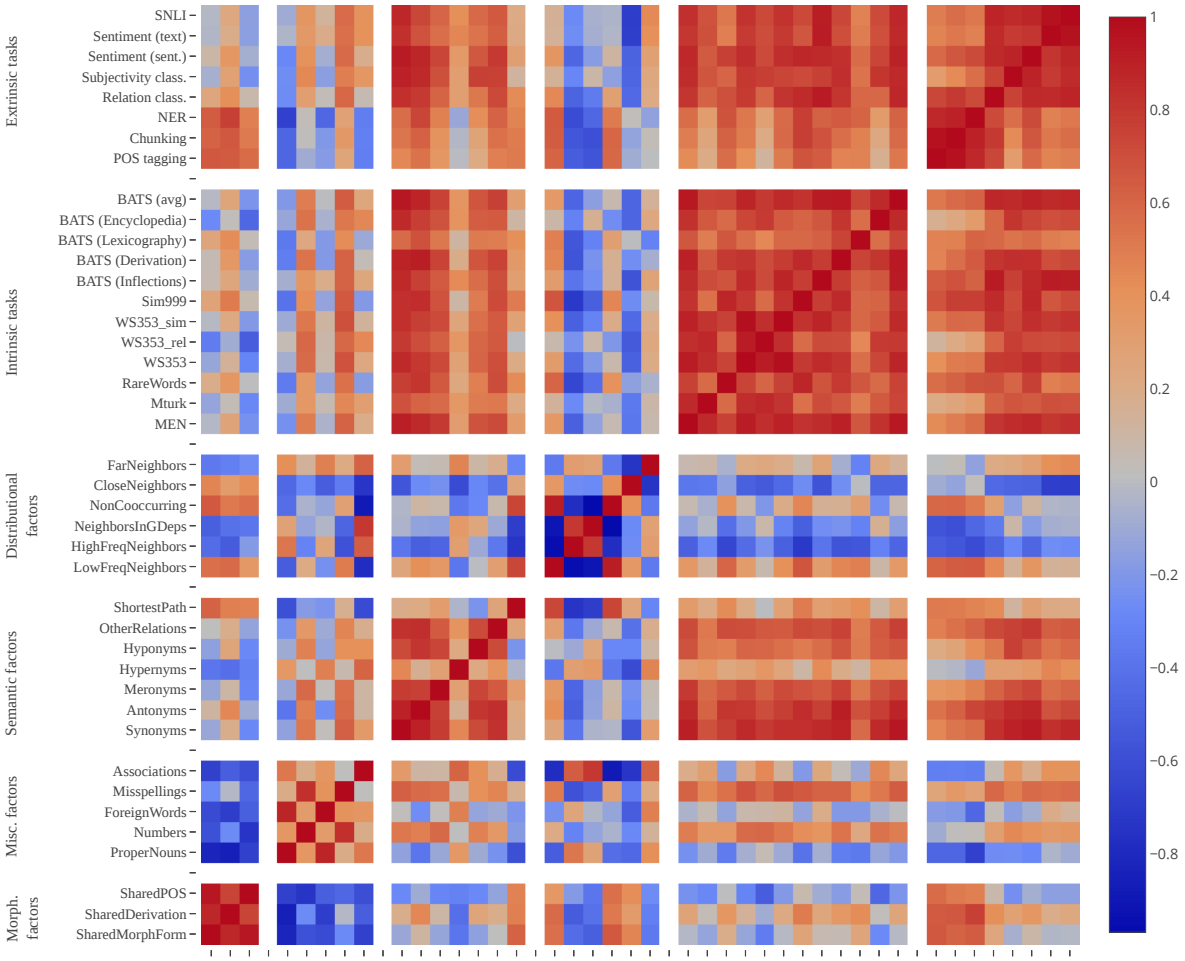
We experimented with the scores from 1,000, 5000, and 100 top neighbors of the sample words. The overall correlation patterns were similar, suggesting that it will be sufficient for future work to limit the selection to the top 100 neighbors. We thus report the results for the top 100 neighbors in this section.

The largest immediately observable pattern is the high correlation between *almost all* intrinsic tasks. The correlations are lower (but still over 0.5) for the lexicography and encyclopedic section of BATS; but the performance on these categories is generally rather low (Drozd et al., 2016) and could be unreliable. On the other hand, the high correlation between analogy and all relatedness and similarity datasets confirms the conclusion of (Rogers et al., 2017) that accuracy on analogy depends on the similarity between the source word pairs (even for LRCos method).

As for the extrinsic tasks, the immediate observation is the low correlation between all the intrinsic and 3 sequence labeling tasks. These are the same tasks that were reported by Chiu et al. (2016) and Ghannay et al. (2016) as the tasks for which higher performance does not correspond to higher performance on intrinsic tasks. However, all the non-sequence-labeling tasks in our sample *do* correlate with the intrinsic datasets.

This is a crucial finding; it shows that the traditional intrinsic tasks are after all useful for predicting performance on *some* downstream tasks. Their disadvantage is that they offer no explanation about why this is the case, and what could be expected of other extrinsic tasks not in our sample.

This is where LD methodology comes in. Figure 2 shows 4 groups of factors that we analyzed, together with their correlations with both extrinsic and intrinsic tasks. An immediate observation is that a large amount of neighbors that are in some lexicographic semantic relation with the target word is a good predictor of performance on both traditional intrinsic datasets and all the extrinsic tasks *except for the sequence labeling*. On the other hand, these particular tasks correlate highly with the three



Distributional factors
LowFreqNeighbors, HighFreqNeighbors: the frequency of the neighbor in the source Wikipedia corpus is under/above 10,000
NeighborsInGDeps: whether the two words co-occur in the Google dependency ngrams
NonCooccurring: the number of word pairs that do not co-occur in the source Wikipedia corpus (bag-of-words window size 2).
CloseNeighbors: the number of top 100 neighbors with cosine distance to the target word over 0.8.
FarNeighbors: the number of top n neighbors with cosine distance to the target word less than 0.7.

Semantic factors
Synonyms, Antonyms, Hypernyms, Hyponyms, Meronyms: the corresponding lexicographic relations established by the dictionaries.
OtherRelations: holonymy, troponymy, coordinate terms, and "otherwise related" in Wiktionary.
ShortestPath: the minimum path between synsets of two words in the WordNet ontology

Miscellaneous factors
ProperNouns: the neighbor is a proper noun.
Numbers: the neighbor is a numeral, or contains a number.
ForeignWords: the neighbor is not found in English, but found in German, French or Spanish spellechecker dictionaries.
Misspellings: the neighbor is not found in dictionaries and contains an unusual combination of letters and punctuation or numbers.
Associations: the two words constitute an associative pair (in either direction), according to EAT[8] (Wilson et al.,) and USF-FAN[9] (Nelson et al., 2004).

Morphological factors
SharedMorphForm: the two words share their morphological form (in this case, both are lemmas).
SharedDerivation: the two words share affix(es) or stem(s), or are both compounds (based on Wiktionary and custom LDT tools).
SharedPOS: the two words have the same part of speech (any overlap counts).

Figure 2: Pairwise Spearman's correlations of extrinsic and intrinsic tasks between themselves and LDT scores for top 100 neighbors. An interactive version of this chart, as well as numerical data and data for top 1000 neighbors can be found at `http://ldtoolkit.space/analysis`.

morphological factors we considered: the neighbors sharing morphological form, derivational pattern and/or part-of-speech of the target word. This finding confirms our original hypothesis: *different tasks rely on different information, making a single-number intrinsic evaluation unfeasible.*

At the same time, the border between morphology and semantics is not a stone wall. The derivational morphology factor does have weak positive correlations with all the intrinsic and extrinsic tasks, since shared derivation does indicate at least partially shared semantics. The performance on sequence labeling tasks also does correlate with the scores on lexicographic semantic relations. We attribute this to the fact that dictionaries usually store relations between words of the same part of speech, so these scores implicitly contain the *SharedPOS* factor.

The semantic factors that appear to be the least useful across all tasks are the *ShortestPath* and hypernymy. The latter is surprising in the light of such tasks as SNLI that seems to clearly rely on it.

The psychological associations turn out to be only weakly useful in the semantic extrinsic tasks (presumably to the same degree to which they correlate with relatedness tests, and relatedness tests correlate with extrinsic tasks). This is in line with Gladkova and Drozd (2016)'s suggestion that human relatedness scores depend on the psychological factors such as speed of association, rather than pure semantics.

It could be expected that, in the sample of general English vocabulary, the neighbors that are proper nouns or foreign words would be detrimental to any task. However, we observe a positive correlation with the amount of neighbors that contain numbers (presumably due to the meaningful hyponymy that they could indicate, such as model numbers, addresses etc.). A large number of misspelled neighbors is also apparently good for all tasks: since all the models in this study are word-level, this could indicate their ability to mitigate the lack of subword information.

Among the distributional factors, the clearest positive effect is observed for the models that have the highest number of low-frequency vocabulary (under 10,000 occurrences in the corpus) in the word vector neighborhoods. Since most word types fall in this range, this indicates that a "good" model should be able to populate vector neighborhoods with related words, even if they are not particularly frequent. The *NonCooccurring* factor is apparently useful for sequence labeling and some intrinsic tasks to find more *latent* relations between words that do not actually co-occur in the corpus, i.e. deduce relations on the basis of the "second-hand" rather than direct similarity between distributional patterns of words. Finally, the scores on the *FarNeighbors* factor suggest that high-level semantic tasks benefit from more neighbors that are less than 0.7 similar to the target word. This could be interpreted as follows: if a neighborhood is packed with words that are all quite similar, many of them will end up being within 0.00000001 from each other, making the margin of error very small for the models that use these representations in tasks.

One more important observation from this experiment is that all the extrinsic and intrinsic tasks have high correlations with more than one LD factor, which illustrates the point about tasks being complex *ensembles* of various linguistic features. However, it is only by breaking them down into smaller, controllable factors that we can explain and improve on them.

Note also that all the factors we considered correlate considerably with each other within their subclasses: the morphological features have mostly *negative* correlations with the lexicographic ones, while the sequence labeling tasks only weakly correlate with the high-level semantic tasks. This raises the question of what it would take for a representation to do both equally well.

5.2 Profiling embeddings with LD

As a brief demonstration of explanatory power of LD methodology, let us consider CBOW, SG, and GloVe models and their performance on the 8 above tasks in one condition: linear bag-of-words context, 500-dimensional vectors (the LD scores for top 1000 neighbors are reported). Table 4 lists some of the LD factors identified in Section 5.1 together with performance on our 8 extrinsic tasks.

We see that the 3 models are very close in most of factors; yet it is SG that is always slightly ahead in semantic, morphological and distributional LD factors and actual performance. CBOW is consistently slightly behind SG on these accounts, and slightly ahead on the scores for misspellings, foreign words, numbers and proper nouns (apparently at the cost of the meaningful relations).

The key difference between GloVe and Word2Vec appears to be the *LowFreqNeighbors* (amount of

low-frequency words as neighbors) and *NonCooccurring* words (words that end up as neighbors in spite of not co-occurring in the source corpus). This suggests that the success of SG is due to its ability to bring together related words even if they were rare, and/or did not co-occur in the corpus. This apparently outweighs even GloVe's significant advantage in sparser vector neighborhoods.

It is interesting that comparable or even superior scores on "morphological" factors did not give GloVe an advantage in POS-tagging and chunking tasks. Apparently specialized information is necessary but not sufficient for top performance, and it is successful ensembles of features that matter.

5.3 LD for parameter search

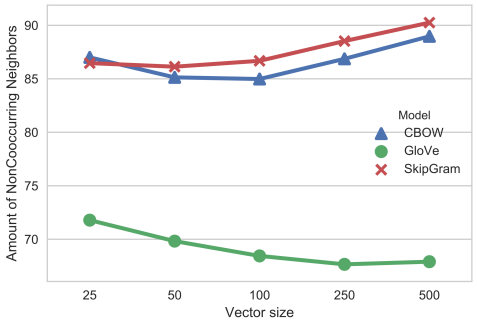
LD factors are equally useful for studying the effect of hyperparameters as well as underlying algorithms. As a brief demonstration, consider the behavior of the *NonCooccurring* factor discussed above when varying the size of SG, GloVe, and CBOW vectors (linear unbound context, top 1000 neighbors).

The larger representations are often assumed to be more informative, but Figure 3a shows that this is not the case for GloVe. The questions of why the compression effect is the smallest for the smallest vectors, and what other factors are at play here, merit a separate investigation. As in the case discussed in 5.2, Skip-Gram is consistently slightly ahead of CBOW, except for the lowest dimensionality.

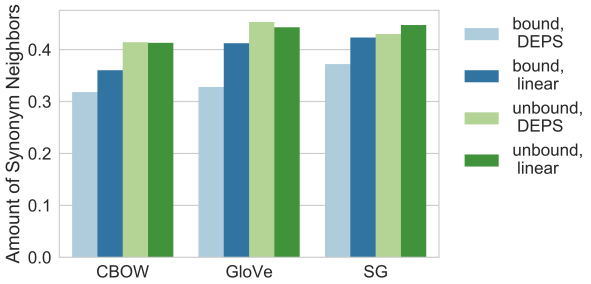
As a final example, let us take a quick look at the idea that the dependency-based contexts pack more synonyms than linear contexts. This does not seem to be the case in Fig. 3b: the positive effect is rather due to unbound vs bound representation than to dependency-based or linear context. Thus, if the goal is to maximize the number of synonyms, the effort of parsing is not justified. This result is consistent with the finding that dependency-based vector space models do not outperform the optimized window-based models on the TOEFL synonym task (Lapesa and Evert, 2017).

	CBOW	GloVe	SG
LD factors			
SharedMorphForm	51.819	52.061	**52.9**
SharedPOS	30.061	**35.507**	31.706
SharedDerivation	4.468	3.938	**5.084**
Synonyms	0.413	0.443	**0.447**
Antonyms	0.128	0.133	**0.144**
Hyponyms	0.035	0.035	**0.038**
OtherRelations	0.013	0.013	0.013
Misspellings	**13.546**	9.914	12.809
ForeignWords	**2.147**	1.976	1.793
ProperNouns	**30.442**	27.278	27.864
Numbers	**4.313**	3.147	3.64
LowFreqNeighbors	94.778	66.51	**96.109**
HighFreqNeighbors	3.421	**15.697**	2.513
NonCooccurring	88.97	67.904	**90.252**
CloseNeighbors	**3.102**	0.16	2.278
FarNeighbors	25.209	**49.934**	21.41
Downstream tasks			
POS-tagging	87.660	83.800	**87.860**
Chunking	77.530	66.100	**78.230**
NER	75.210	69.620	**75.720**
Relation class.	74.780	71.050	**74.800**
Subjectivity class.	89.800	89.160	**89.920**
Polarity (sent.)	75.900	74.600	**76.860**
Sentiment (text)	82.220	82.240	**82.730**
SNLI	69.290	69.510	**69.740**

Table 4: CBOW, GloVe and SG properties and performance



(a) Vector dimensionality effect on *NonCooccurring* factor.



(b) Amount of synonyms in models with different context types.

6 Discussion and Future Work

We have presented the LD methodology for quantitative/qualitative exploration of word embeddings. As proof of concept, our analysis of GloVe and Word2Vec showed that LD can effectively identify the linguistic and distributional factors that make word embeddings more or less successful on the downstream and traditional intrinsic tasks. We are hoping that this work will contribute to the NLP community efforts in the following directions:

- *comparison of word embedding algorithms* (e.g. different modifications of the Word2Vec);
- *hyperparameter effects* on encoding of different linguistic relations;
- a more informed, *hypothesis-driven design of new distributional representations*;
- *informed choice of word embeddings for various downstream tasks*;
- the degree to which different relations are useful for different tasks and to which they can be combined in a *generalized representation without sacrificing too much accuracy on specialist tasks*.
- interaction between *preference for different linguistic relations and performance on different tasks*.

LD methodology itself can be expanded by expanding LDT to other languages and by formulating the criteria for comparing representations of proper nouns. For example, the co-hyponymy relation between names of composers would be covered with the current implementation, but giving a higher score to a model that places *violin* closer to *Bach* than to *Beatles* would require evaluating frame-semantic, or at least topical relations, going beyond the traditional dictionaries.

A major caveat is the instability of word embeddings: different runs of the same model may yield word vector neighborhoods with significantly different lexical content. Some models are more stable than others (in particular, GloVe was found to be more stable than word2vec) Wendlandt et al. (2018), but most models published in the recent years do not explore their stability. This fact does not disqualify evaluations based on vector neighborhoods (not only LD, but also the traditional relatedness and analogy tasks), but it does highlight the absolute necessity of large-scale studies to reach any definitive conclusions about the relative (de)merits of different models. At the moment, most work on word embeddings still report experiments with less than ten models, each trained just once.

Important directions for future research also include going beyond simple word-level word embeddings. Some of the questions to investigate include the balance between semantic and morphological information in subword-level models (Bojanowski et al., 2017) and their ensembles with word-level models (Yang et al., 2017). It would also be interesting to expand LD to sense-aware embeddings (Melamud et al., 2016), particularly for contextualized representations (Peters et al., 2018).

7 Conclusion

We presented LD, a methodology for quantitative/qualitative intrinsic evaluation of word embeddings implemented in an open-source Python library. Moving away from unrealistic single-number evaluations, LD identifies precisely what kinds of information a given word embedding encodes in its vector neighborhoods. Unlike traditional intrinsic tasks, LD can also be used to *explain* the correlation between performance on different tasks, or the lack thereof.

The effectiveness of LD was shown in a large-scale experiment with 60 GloVe and Word2Vec models on 14 intrinsic and extrinsic task datasets. We have identified 21 morphological, semantic and distributional factors that are useful for predicting and interpreting the performance patterns of word embeddings. In addition to providing practical guidelines for choosing the best embeddings for a given task, LD opens new possibilities for more informed, hypothesis-driven development of distributional representations.

Acknowledgements

This work was supported in part by an NSF CAREER award to Anna Rumshisky (IIS-1652742).

References

Judit Acs and András Kornai. 2016. Evaluating embeddings on dictionary-based similarity. In *Proceedings of The 1st Workshop on Evaluating Vector Space Representations for NLP*, pages 78–82, Berlin, Germany. Association for Computational Linguistics.

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and WordNet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 19–27, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jeremy Auguste, Arnaud Rey, and Benoit Favre. 2017. Evaluation of word embeddings against cognitive processes: Primed reaction times in lexical decision and naming tasks. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*, pages 21–26.

Amir Bakarov. 2018. A survey of word embeddings evaluation methods. *arXiv:1801.09536 [cs]*, January.

Marco Baroni and Alessandro Lenci. 2011. How we BLESSed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, GEMS '11, pages 1–10. Association for Computational Linguistics.

Miroslav Batchkarov, Thomas Kober, Jeremy Reffin, Julie Weeds, and David Weir. 2016. A critique of word similarity as a method for evaluating distributional semantic models. In *Proceedings of The 1st Workshop on Evaluating Vector Space Representations for NLP*, Berlin, Germany. Association for Computational Linguistics.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5(0):135–146, June.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal Distributional Semantics. *JAIR*, 49(1-47).

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP) 2014*, pages 740–750.

Yun-Nung Chen, William Yang Wang, and Alexander I. Rudnicky. 2014. Leveraging frame semantics and distributional semantics for unsupervised semantic slot induction in spoken dialogue systems. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 584–589. IEEE.

Billy Chiu, Anna Korhonen, and Sampo Pyysalo. 2016. Intrinsic evaluation of word vectors fails to predict extrinsic performance. In *ACL 2016*, pages 1–6, Berlin, Germany. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167. ACM.

Barry Devereux, Colin Kelly, and Anna Korhonen. 2010. Using fMRI activation to conceptual stimuli to evaluate methods for extracting conceptual representations from corpora. In *Proceedings of the NAACL HLT 2010 First Workshop on Computational Neurolinguistics*, pages 70–78. Association for Computational Linguistics.

Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. 2016. Word embeddings, analogies, and machine learning: Beyond king - man + woman = queen. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3519–3530, Osaka, Japan, December 11-17.

Allyson Ettinger and Tal Linzen. 2016. Evaluating vector space models using human semantic priming results. In *Proceedings of The 1st Workshop on Evaluating Vector Space Representations for NLP*, pages 72–77, Berlin, Germany. Association for Computational Linguistics.

Allyson Ettinger, Naomi H. Feldman, Philip Resnik, and Colin Phillips. 2016. Modeling N400 amplitude using vector space models of word representation. In *Proceedings of the 38th Annual Conference of the Cognitive Science Society*, pages 1445–1450.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 30–35.

Christiane Fellbaum, editor. 1998. *Wordnet: An Electronic Lexical Database*. Language, speech, and communication series. MIT Press Cambridge.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. In *ACM Transactions on Information Systems*, volume 20(1), pages 116–131. ACM.

Sahar Ghannay, Benoit Favre, Yannick Esteve, and Nathalie Camelin. 2016. Word embedding evaluation and combination. pages 300–305, Portorož, Slovenia, May 23, 2016 – May 28, 2016. Association for Computational Linguistics.

Anna Gladkova and Aleksandr Drozd. 2016. Intrinsic evaluations of word embeddings: What can we do better? In *Proceedings of The 1st Workshop on Evaluating Vector Space Representations for NLP*, pages 36–42, Berlin, Germany. Association for Computational Linguistics. [doi:10.18653/v1/W16-2507].

Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. 2016. Analogy-based detection of morphological and semantic relations with word embeddings: What works and what doesn't. In *Proceedings of the NAACL-HLT SRW*, pages 47–54, San Diego, California, June 12-17, 2016. ACL.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Revisiting embedding features for simple semi-supervised learning. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP) 2014*, pages 110–120.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó. Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 33–38, Stroudsburg, PA, USA. Association for Computational Linguistics.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Michael N. Jones, Walter Kintsch, and Douglas J.K. Mewhort. 2006. High-dimensional semantic space accounts of priming. *Journal of Memory and Language*, 55(4):534–552, November.

Douwe Kiela and Stephen Clark. 2014. A systematic study of semantic vector space model parameters. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and Their Compositionality (CVSC) at EACL*, pages 21–30.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Sigrid Klerke, Héctor Martínez Alonso, and Anders Søgaard. 2015. Looking hard: Eye tracking for detecting grammaticality of automatically compressed sentences. *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 97–105.

Siwei Lai, Kang Liu, Shizhu He, and Jun Zhao. 2016. How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6):5–14.

Gabriella Lapesa and Stefan Evert. 2013. Evaluating neighbor rank and distance measures as predictors of semantic priming. In *Proceedings of the ACL Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2013)*, pages 66–74.

Gabriella Lapesa and Stefan Evert. 2014. A large scale evaluation of distributional semantic models: parameters, interactions and model selection. *Transactions of the Association for Computational Linguistics*, 2:531–545.

Gabriella Lapesa and Stefan Evert. 2017. Large-scale evaluation of dependency-based DSMs: Are they worth the effort? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 394–400. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 302–308.

Omer Levy and Yoav Goldberg. 2014b. Linguistic regularities in sparse and explicit word representations. *CoNLL-2014*, pages 171–180.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Bofang Li, Tao Liu, Zhe Zhao, Buzhou Tang, Aleksandr Drozd, Anna Rogers, and Xiaoyong Du. 2017. Investigating different syntactic context types and context representations for learning word embeddings. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2411–2421.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304. Association for Computational Linguistics.

Tal Linzen. 2016. Issues in evaluating semantic spaces using word analogies. In *Proceedings of the First Workshop on Evaluating Vector Space Representations for NLP*, pages 13–18. Association for Computational Linguistics.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208.

Kevin Lund, Curt Burgess, and Ruth Ann Atchley. 1995. Semantic and associative priming in high-dimensional semantic space. In *Proceedings of the 17th Annual Conference of the Cognitive Science Society*, volume 17, pages 660–665.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. *CoNLL-2013*, 104.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June. Association for Computational Linguistics.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. Context2vec: Learning Generic Context Embedding with Bidirectional LSTM. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, pages 51–61, Berlin, Germany, August 7-12, 2016. Association for Computational Linguistics.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the State of the Art of Evaluation in Neural Language Models. *arXiv:1707.05589 [cs]*, July.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *HLT-NAACL*.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, December.

Neha Nayak, Gabor Angeli, and Christopher D. Manning. 2016. Evaluating word embeddings using a representative suite of practical tasks. In *Proceedings of the 1st Workshop on Evaluating Vector Space Representations for NLP*, pages 19–23, Berlin, Germany, August 12, 2016. Association for Computational Linguistics.

Douglas L. Nelson, Cathy L. McEvoy, and Thomas A. Schreiber. 2004. The University of South Florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers*, 36(3):402–407, August.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 12, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 337–346, New York, NY, USA. ACM.

Anna Rogers, Aleksandr Drozd, and Bofang Li. 2017. The (too many) problems of analogical reasoning with word vectors. In *Proceedings of STARSEM 2017 (to Appear)*. Association for Computational Linguistics.

Eugen Ruppert, Manuel Kaufmann, Martin Riedl, and Chris Biemann. 2015. JOBIMVIZ: A web-based visualization for graph-based distributional semantic models. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 103–108, Beijing, China, July 26-31. ACL.

Enrico Santus, Anna Gladkova, Stefan Evert, and Alessandro Lenci. 2016. The CogALex-V shared task on the corpus-based identification of semantic relations. In *Proceedings of the Workshop on Cognitive Aspects of the Lexicon*, pages 69–79, Osaka, Japan, December 11-17. ACL.

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015), Lisbon, Portugal*, pages 298–307. Association for Computational Linguistics.

Anders Søgaard. 2016. Evaluating word embeddings with fMRI and eye-tracking. In *Proceedings of the 1st Workshop on Evaluating Vector Space Representations for NL*, pages 116–121, Berlin, Germany, August 12, 2016. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2049–2054, Lisbon, Portugal, 17-21 September 2015. Association for Computational Linguistics.

Yulia Tsvetkov, Manaal Faruqui, and Chris Dyer. 2016. Correlation-based intrinsic evaluation of word vector representations. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 111–115, Berlin, Germany. Association for Computational Linguistics.

Cyma Van Petten. 2014. Examining the N400 semantic context effect item-by-item: Relationship to corpus-based measures of word co-occurrence. *International Journal of Psychophysiology*, 94(3):407–419, December.

Laura Wendlandt, Jonathan K. Kummerfeld, and Rada Mihalcea. 2018. Factors influencing the surprising instability of word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2092–2102, New Orleans, Louisiana, June. Association for Computational Linguistics.

Michael Wilson, Georg Kiss, and Christine Armstrong. EAT : The Edinburgh associative corpus [Electronic resource].

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W Cohen, and Ruslan Salakhutdinov. 2017. Words or characters? Fine-grained gating for reading comprehension. In *Proceedings of ICLR*, pages 1–10.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344.

Word Sense Disambiguation Based on Word Similarity Calculation Using Word Vector Representation from a Knowledge-based Graph

Dongsuk O[*1], Sunjae Kwon[*2], Kyungsun Kim[1] and Youngjoong Ko[3]
[1]R&D Center, Diquest Inc., Seoul 08390, Republic of Korea
[2]School of Elec. & Comp. Engineering, UNIST, Ulsan 44919, Republic of Korea
[3]Dept. of Computer Engineering, Dong-A University, Busan 49315, Republic of Korea
[1]ods@diquest.com, [2]soon91jae@unist.ac.kr, [1]kksun@diquest.com,
[3]youngjoong.ko@gmail.com

Abstract

Word sense disambiguation (WSD) is the task to determine the sense of an ambiguous word according to its context. Many existing WSD studies have been using an external knowledge-based unsupervised approach because it has fewer word set constraints than supervised approaches requiring training data. In this paper, we propose a new WSD method to generate the context of an ambiguous word by using similarities between an ambiguous word and words in the input document. In addition, to leverage our WSD method, we further propose a new word similarity calculation method based on the semantic network structure of BabelNet. We evaluate the proposed methods on the SemEval-2013 and SemEval-2015 for English WSD dataset. Experimental results demonstrate that the proposed WSD method significantly improves the baseline WSD method. Furthermore, our WSD system outperforms the state-of-the-art WSD systems in the Semeval-13 dataset. Finally, it has higher performance than the state-of-the-art unsupervised knowledge-based WSD system in the average performance of both datasets.

1 Introduction

In natural language text, it is very common for a word to have more than one sense. For example, in a sentence "An airline hires new cabin crews" of Figure 1, the words 'airline,' 'hires,' 'new,' 'cabin' and 'crews' have more than two senses. In this case, we can map 'airline' to 'airline Noun#2', 'hires' to 'hire Verb#1,' 'new' to 'new Adjective#6,' 'cabin' to 'cabin Noun#3' and 'crews' to 'crew Noun #7' by the context of this sentence. The word sense disambiguation (WSD) is the task to determine the correct meaning of an ambiguous word in a given context. WSD is being used as a key step for performance improvement in many natural language processing tasks such as machine translation (Vickrey et al., 2005; Chan et al., 2007), information retrieval (Sanderson 1994; Stokoe et al., 2003) and so on.

WSD can be divided into supervised approaches and knowledge-based unsupervised approaches. In the supervised approach, machine learning models are trained by a corpus, in which the correct senses of ambiguous words are already annotated by human annotator (Weissenborn et al., 2015; Melamud et al., 2016; Raganato et al., 2017). However, constructing training corpus for all languages and words is tremendously expensive, so the supervised approaches generally have some limitations on the set of the words that can be disambiguated. On the other hand, the knowledge-based unsupervised approaches utilize lexical knowledge bases (LKBs) such as a Wordnet (Banerjee and Pederson., 2003; Chaplot et al., 2015). These approaches have performed WSD by combining contextual information and semantic knowledge on the LKBs. Thus, there is much number of words that can be disambiguated when compared to the supervised approaches. For this reason, it is known that the knowledge-based approach is more suitable than supervised approach for the practical WSD systems (Moro et al., 2014; Chaplot and Salakhutdinov 2018).

* Both authors contributed equally to this work.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2704–2714
Santa Fe, New Mexico, USA, August 20-26, 2018.

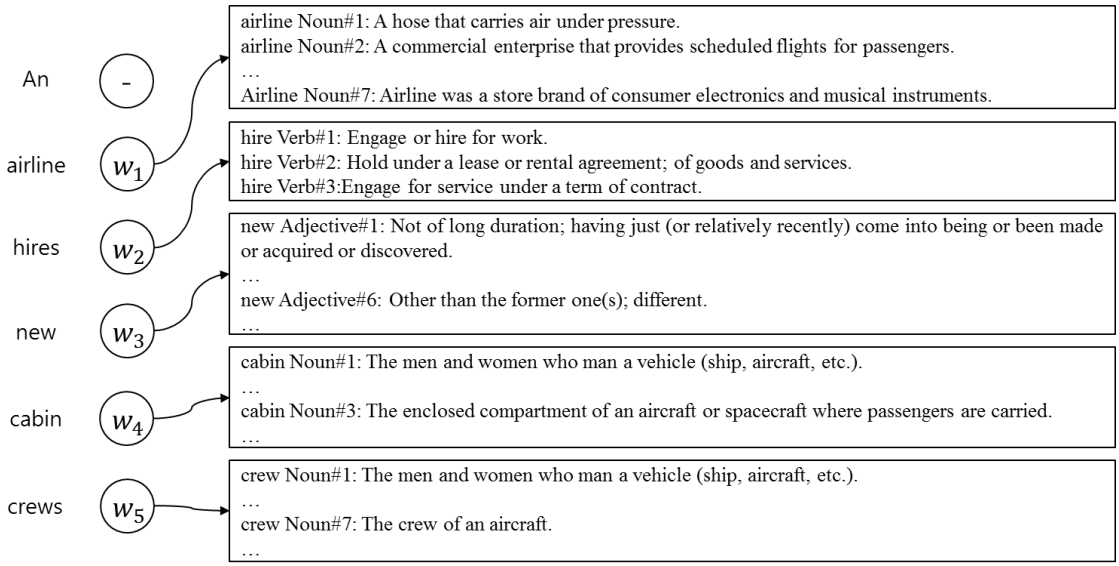


Figure 1: Example for the WSD task.

The most popular approach for the knowledge-based unsupervised ones is the graph-based WSD approach determining the answer sense by linking the senses and context of the ambiguous words on the LKBs to generate a semantic subgraph and measure the connectivity of the senses corresponding to a node in the subgraph. Therefore, establishing an efficient strategy for generating subgraphs of those senses directly affects the performance of WSD. Conventional knowledge-based unsupervised approaches tried to construct the subgraph of all the words that appear in the document and concurrently disambiguate all ambiguous words in the document (Navigli and Lapata, 2007; Navigli and Lapata, 2010). This joint optimization strategy has the advantage of being able to derive the optimal sense set for ambiguous words, but there is a limitation in that the computational complexity increases exponentially as the number of ambiguous words is increasing. To ameliorate above problem, Manion et al. (2014) proposed a subgraph construction strategy using the iterative subgraph reconstruction approach that greedily analyzes the ambiguous words according to a specific priority. However, the iterative subgraph uses entire words in the document as a context to determine the sense of each ambiguous word and it makes sometimes the subgraph of a word overcomplicated with unnecessary information.

Based on this iterative subgraph reconstruction approach, we propose a new subgraph construction strategy to avoid the abovementioned problems by selectively restricting the context when constructing a subgraph of an ambiguous word. In our proposed approach, contextual words of an ambiguous word for constructing the subgraph are selected by thresholding the word similarities to the ambiguous word. In addition, we propose a new word similarity measure method by using word vector representations, generated from the knowledge graph of LKBs and a neural network, for the efficient contextual word selection. In the experiments, we prepared the three publicly accessible English WSD datasets of SemEval-2007 (Pradhan et al., 2007), SemEval-2013 (Navigli et al., 2013) and SemEval-2015 (Moro and Navigli, 2015). We used the SemEval-2007 dataset as a development set for parameter tuning and other datasets as test sets. Experimental results show that our proposed WSD approach with the new subgraph construction strategy has significantly better performance than the baseline iterative subgraph reconstruction approach of Manion et al. (2014). Moreover, when we apply our proposed word similarity method to our WSD approach, it achieved better performance than the WSD systems using other existing word similarity methods. Eventually, the final proposed WSD system with all of the new subgraph construction and word similarity methods achieved higher performance than the state-of-the-art unsupervised knowledge-based WSD systems.

The remainder of the paper organized as follows. In section 2, we introduce previous studies for the WSD system and our sense repository BabelNet (Navigli and Ponzetto, 2012). Section 3 is allocated to introduce our proposed WSD system in detail. Experimental environments and results are described in section 4. Finally, conclusion and future work are discussed in section 5.

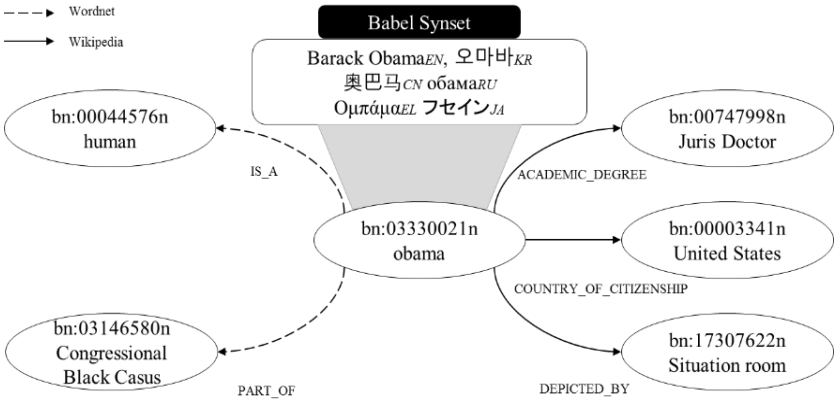


Figure 2: Babelnet example displaying the Babel synsets and the semantic relations (edges) between senses (nodes).

2 Related Work

2.1 Word Sense Disambiguation

Recently, the graph-based WSD method become the most popular a method for the knowledge-based WSD (Navigli and Velardi, 2005). The graph-based method selects the answer sense of the ambiguous word based on the semantic structure of LKBs. Generally, the answer sense is chosen from the semantic subgraph that connects the senses of the words in the input document using the semantic relationships defined in LKBs. Navigli and Lapata (2007) built a semantic subgraph of the entire words including senses and then used graph connectivity measures to determine the combination of answer senses. Agirre et al. (2014) suggested a knowledge-based WSD approach used personalized page rank (PPR) over the semantic subgraph. They calculated the relative importance of senses using PPR and the sense with the highest score was chosen as an answer sense. Babelfy (Moro et al., 2014) presented another graph-based approach that jointly selects answer of WSD and entity linking (Xiao et al., 2015). Utilizing the random walk algorithm with a restart, it extracted a dense subgraph and reweighted the edges of a BabelNet. They iteratively disambiguate words by reconstructing a semantic subgraph at each word. Based on the assumption that word with a minimum sense is an easiest word among the entire ambiguous words, Manion et al. (2014) disambiguated the ambiguous words in order of the number of their senses. Chaplot et al. (2015) maximized the joint probability of whole senses in the context using WordNet and dependency. Tripodi and Pelillo (2017) suggested to apply the idea of the evolutionary game theory to their WSD system. By exploiting the semantic similarity of the words, they formulated WSD as a constraint satisfaction problem and derived it utilizing game theorem tools.

In addition to these abovementioned methods, several methods for WSD have been proposed. Chaplot and Salakhutdinov (2018) proposed a topic modeling based WSD approach to ameliorate computational complexity of graph-based WSD. Zhong and Ng (2010) tried to disambiguate words using support vector machine (Suykens and Vandewalle, 1999) with rich linguistic features such as part-of-speech, local collocations and surrounding contextual words. Weissenborn et al. (2015) jointly optimized WSD and entity linking model in an extensible multi-objective optimization. Pasini and Navigli (2017) built a large-scale training corpus for WSD from scratch using Wordnet. Raganato et al. (2017) suggested a supervised WSD approach using bidirectional long short-term memory and attention mechanism.

Our WSD method is based on the iterative subgraph reconstruction approach (Manion et al., 2014). However, our WSD approach is crucially different in that it selectively constructs subgraphs by thresholding the contents words of the input document based on the similarity with the ambiguous words.

2.2 BabelNet

Most unsupervised WSD systems utilize LKBs such as Wordnet[1] to obtain a set of possible senses for each ambiguous word. BabelNet[2] is a multi-lingual lexicalized semantic network and ontology. It pro-

[1] https://wordnet.princeton.edu/
[2] http://babelnet.org

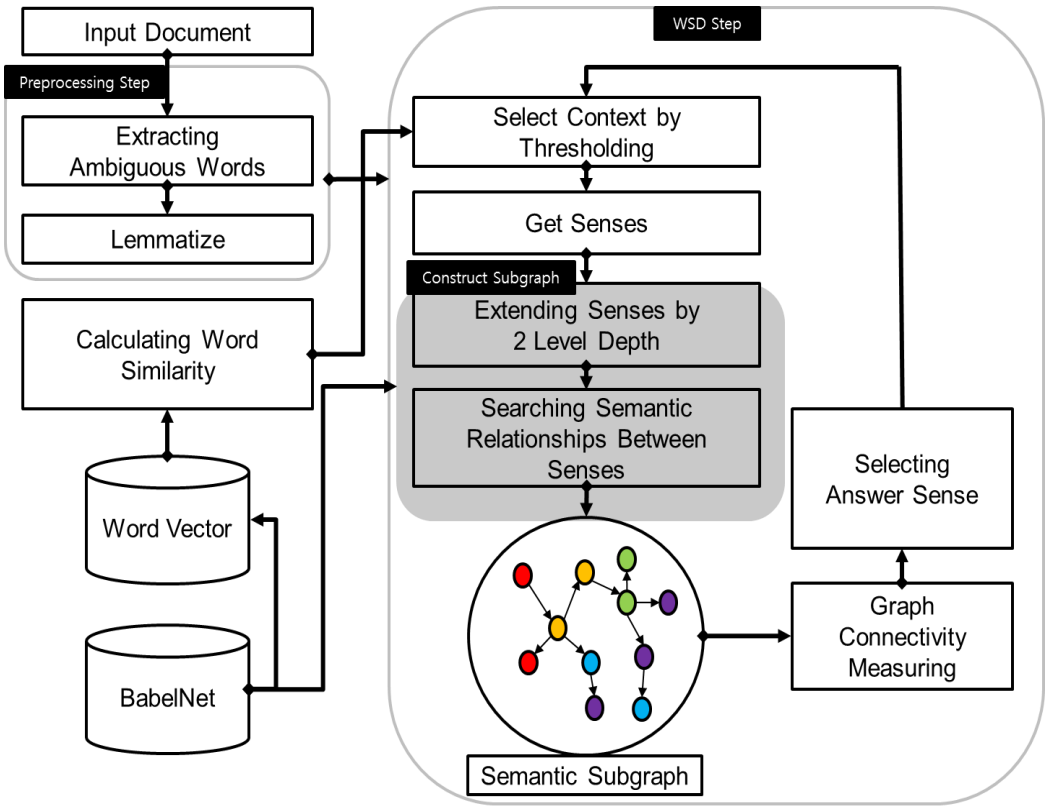


Figure 3: Overall processes of the proposed system.

vides the senses of content words[3], semantic relationship between the senses and the set of synonyms of the sense.

As shown in Figure. 2, BabelNet has a graph structure that consists of nodes and edges. A node indicates the sense of a word and an edge denotes the semantic relationship between the senses. The synonym information for a sense is accessible from Babel synset, which provides multi-lingual synonyms. Semantic relationship between senses contains the relationship defined by both Wikipedia (ACADEMIC_DEGREE, COUNTRY_OF_CITIZENSHIP, DEPICTED_BY, etc) and Wordnet (IS_A, PART_OF, etc). For example, a word 'Obama' contains six possible noun senses such as "bn:03330021n: 'Barack Hussein Obama II is an American politician who served as the 44th President of the United States from 2009 to 2017.'," and an adjective sense of "bn:13705874a: 'Of or pertaining to the political figure and 44th president of the United States of America Barack Obama.'." In addition, 'Obama' with a noun sense of 'bn: 03330021n' has an 'IS-A' semantic relationship with 'Human' with a noun sense of 'bn: 00044576n' and has a semantic relationship of 'COUNTRY_OF_CITIZENSHIP' with 'United States' with a noun sense of 'bn: 00003341n.'

As mentioned above, BabelNet provides the senses and their semantic relationships to multi-lingual, it is advantageous to extend a WSD system to other language. This makes many recent studies for WSD systems choosing BableNet as a sense repository (Moro et al., 2014; Manion et al., 2014; Tripodi and Pelillo 2017).

3 Proposed WSD System

This section describes our fundamental ideas to improve the performance of WSD and how they are integrated into our WSD system as illustrated in Figure 3. The subsection 3.1 is allocated to introduces a novel word similarity calculation method for the contextual word selection. Next, we explain our new iterative subgraph construction method that combine contextual word selection in the traditional WSD approach.

[3] Verbs, adverbs, nouns and adjectives

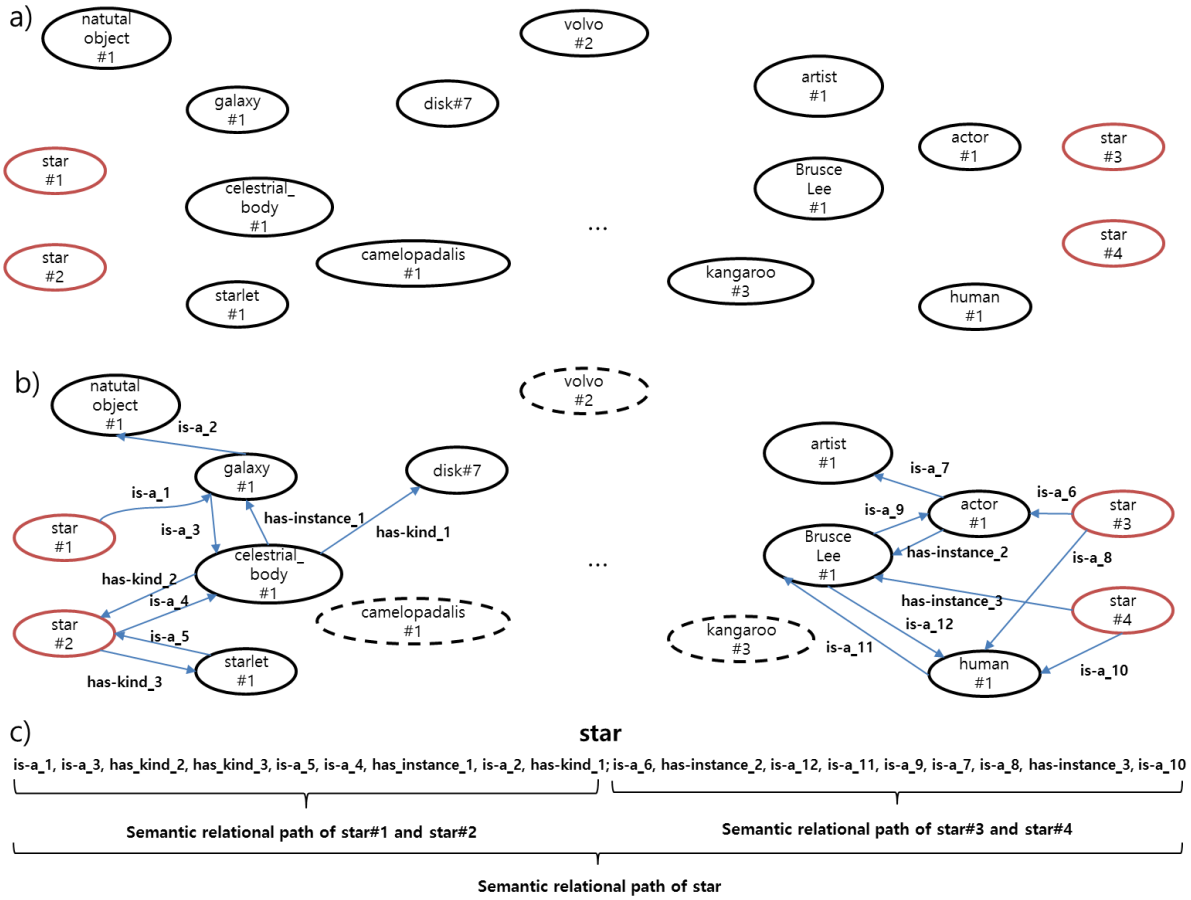


Figure 4: Example of the semantic relational path extracting process of a word 'star'. An ellipse means a sense and an arrow indicates the semantic relationship between the senses. a) Initial state, b) Extending the senses by 2 level depth and c) Generating Semantic relational path of 'star' using DFS algorithm.

3.1 Word Similarity Calculation Through the Word Vector Representation from the BabelNet Graph Structure for the Contextual Words Selection

In order to determine the correct sense of the ambiguous words in the WSD, it is very important to use the context around to find the sense of the ambiguous word. However, not all the contextual words are equally important for WSD (Karov and Edelman, 1998). If we can choose more important contextual words for each ambiguous word in the document, then we can more accurately disambiguate the sense of an ambiguous word by reducing unnecessary information. From this point of view, we assume that the higher a word has similarity to the ambiguous word, the more it can contribute to determining the sense of an ambiguous word.

In WSD, we determine the sense of a word based on context. At this time, the context in which we are interested is the theme of the entire document or sentence. For this reason, semantic information such as theme words will have a much more impact on WSD than syntactic information such as part-of-speech or sentence component information. Under this assumption, we propose a novel method of generating the word vector representations for the semantic information using knowledge graph structure as follows.

The senses of a word can be connected by a series of the semantic relationships in the semantic network. Figure 4 illustrates an example of representing a word as the sequence of the semantic relationships by connecting the senses of the word on the BabelNet knowledge graph. In Figure 4 (a), a noun word 'star' has four different senses; star#1: "A celestial body of hot gases", star#2: "Any celestial body visible from the Earth at night.", star#3: "An actor who plays a principal role" and star#4: "A widely known person." To connect these senses, we extend the senses by L level depth around each sense of the word as shown in Figure 4 (b). In this example, there are two connected graphs are generated. The

first one connects the extended senses of star#1 and star#2 and the second one does those of star#3 and star#4. In this example, each subgraph is related to the common theme of the connected senses. A graph connecting star#1 and star#2 contains common senses associated with astronomical phenomena. On the other hand, a graph connecting star#3 and star#4 consists of senses related to human or occupation. As shown in the Figure 4 (c), the DFS algorithm is applied to easily handle subgraphs that make up the meaning of a word. By the DFS algorithm, we can represent each graph connecting the sense of a word as a sequence of semantic relationships, which we refer to the semantic relational path. Finally, the concatenated semantic relational path of all subgraphs is considered as the overall representation of the word (see Figure 4 (c)).

The semantic relational path of a word from the previous paragraph consists of three structures: relations that connect senses, subgraph that is connected with the senses and sharing a theme and words that are represented as a set of subgraphs. If we match a relation to a word and a subgraph to a sentence in this structure, we can regard the semantic relation path of word as a kind of a pseudo-document. To effectively encode information of the semantic relational path of word, we used Doc2vec (Le and Mikolov, 2014). Doc2vec is an unsupervised learning algorithm that generates a document vector based on words contained in the document. The vectors of documents having a similar meaning are projected into the similar vector space. In our case, the semantic relational path, as a pseudo-document, of the word is an input and the word vector representation of the word is an output of the Doc2vec. Thus, if words sharing semantic relationships will be projected into the similar vector space, otherwise they will be projected in the totally different vector space. To measure the similarity of the words by regarding both distance and direction, the cosine similarity measure of Eq.1 is used to calculate the word similarity of the vector representation of the words $\boldsymbol{w_1}$ and $\boldsymbol{w_2}$.

$$word\_similarity(\boldsymbol{w_1}, \boldsymbol{w_2}) = \frac{\boldsymbol{w_1} \cdot \boldsymbol{w_2}}{||\boldsymbol{w_1}|| \times ||\boldsymbol{w_2}||} \quad (1)$$

3.2 Iterative Subgraph Reconstruction Approach with Word Similarity-based Context Words Selection

In order to determine the correct sense of ambiguous words in a graph-based WSD approach, it is essential to establish an efficient strategy for constructing a subgraph that connects the correct senses by taking into account the structure of the semantic network and words in the context. In our study, we suggest a new WSD strategy that constructs a set of context words that have a similarity value to the ambiguous word beyond a threshold.

Proposed Word Sense Disambiguation System

Input: input document (*Input*)
Output: disambiguated sense (*A*nswer)

1: $I \leftarrow Extracting\_ambiguous\_words(Input)$
2: $\mathcal{L} \leftarrow Lemmatize(I)$
3: $Answers \leftarrow \emptyset$
4: **For** $l_i \; in \; \mathcal{L} \; do$
5: $C_i \leftarrow SelectContext(l_i, Other\_Words)$
6: $S_{l_i} \leftarrow GetSenseSet(l_i)$
7: $S_{C_i} \leftarrow GetSenseSet(C_i)$
8: $G_i \leftarrow ConstructSubGraph(S_{l_i}, S_{C_i}, Answer)$
9: $\hat{s}_* \leftarrow argmax_{s_j \in S_{l_i}} \phi(G_i, S_{l_i})$, where ϕ *indicates graph conectivity*
10: put $\hat{s}_*$ in $Answer$
11: **Return** $Answer$

Our WSD system is made up of two steps: the preprocessing step (line 1 to 3) and the WSD step (line 4 to 11). In the preprocessing step, we extract the sequence of the ambiguous words $I = \{w_1 \ldots w_m\}$ in an input document, $Input$, and the order of sequence follows to the occurrence order of the ambiguous

words in the input document. The sequence of ambiguous words, I, is mapped to their lemmatized form, $\mathcal{L} = \{l_1 \dots l_m\}$ and the answer set of disambiguated senses from our WSD system, $Answer$, is initialized by a null set. In the WSD step, our proposed WSD method iteratively determine the answer sense in order of $\mathcal{L}$. To do this, it first selects the contextual words, C_i, of an ambiguous word, l_i, by measuring similarities between the ambiguous word and other words in the document. To do this, our system calculates all the similarities between l_i and other words. Words whose similarity to l_i exceeds a threshold are selected as C_i. If there is no word exceed the threshold, its context is created by choosing a word that has the highest similarity (line 5). Then the senses ($S_{l_i} and\ S_{C_i}$) of l_i and C_i are extracted from BabelNet (line 6 &7). Next, the whole senses of S_{l_i}, S_{C_i} and *Answer* are extended by the depth of level L and they are connected as the semantic relation by the depth-first search (DFS) algorithm (line 8). Finally, the graph connectivity of the sense, s_j, is calculated by the PPR algorithm (Gutiérrez et al., 2013) and the sense with the highest connectivity is selected as answer sense (line 9 and 10). This process is repeated until no more ambiguous words remained in the set of $\mathcal{L}$ (line 4 to 11).

4 Experiments

4.1 Datasets

We evaluated our WSD system on the three publicly available English WSD corpora: SemEval-2007, SemEval-2013 and SemEval-2015.

- The SemEval-2007 dataset consists of three documents with 465 noun and verb words annotated with Wordnet entries. In our experiment, 414 words in the BabelNet entry were selected and this dataset was used for the development set.
- The SemEval-2013 dataset consists of 13 news articles, including various domains from 2010 to 2012. All the noun words were annotated and there are 1,931 words to be disambiguated.
- The SemEval-2015 dataset consists of four documents from several heterogeneous domains. This dataset annotated WSD and entity linking tasks at the same time. The answer senses were annotated for all content words in the dataset and there are 1,261 words to be disambiguated.

4.2 Experimental Settings

Gensim Doc2vec library[4] was used to generate word vector representation introduced in the subsection 3.2. The dimension of the word vector was allocated to 200 and window size is set to 3. In addition, we set the initial learning rate of Doc2vec to 0.5. All the other Doc2vec parameters were set to default. Finally, threshold for the contextual words was set to 0.5. The resources for our word vector revector representation is available at `https://github.com/nlpbank/SRP2Vec`. Furthermore, we set up the hyperparameters of our system at the highest score on the SemEval-2007 dataset as the development set.

As a performance evaluation measure of the WSD systems, we used a F_1-score criteria of Eq.2 that is a harmonic mean of precision of Eq.3 and recall of Eq.4. Besides, to determine statistically significant difference between the performance of the system, we carried out macro student *t*-test (Yang and Liu, 1999).

$$Precision = \frac{\#\ of\ correctly\ disambiguated\ answers}{\#\ of\ words\ that\ outcome\ is\ positive} \tag{2}$$

$$Recall = \frac{\#\ of\ correctly\ disambiguated\ answers}{\#\ of\ true\ answers} \tag{3}$$

$$F_1 - score = \frac{2 \times Precision \times Recall}{Precision \times Recall} \tag{4}$$

[4] https://radimrehurek.com/gensim/models/doc2vec.html

4.3 Experimental Results

To verify the effectiveness of our proposed WSD system, we compared our WSD approach (*Wordsim_iter*) to a baseline WSD approach (*Sudoku_iter*) (Manion et al., 2014). The results of the SemEval-2013 and SemEval-2015 datasets are the same as those reported by Manion et al, (2014). On the other hand, because Manion et al, (2014) have not reported score on the SemEval-2007 dataset, we re-implemented and evaluated *Sudoku_iter* on the SemEval-2007 dataset. Furthermore, in order to verify our hypothesis that semantically similar words are more important than syntactically similar words in the WSD task, we compare our word similarity (*SRP2vSim*) calculation method with an existing word vector representation-based similarity calculation method as follows:

- *W2vSim*: It is based on the Word2vec that an unsupervised algorithm to generate word vector representation (Mikolov et al., 2013). We obtained word vector representations in the official word2vec website[5]. The word vector representation pretrained in the general text corpus has both the semantic and syntactic features of the words (Mikolov et al., 2013).

WSD Approach	Word Similarity Measurement	Dev			Test					
		SemEval-2007			SemEval-2013			SemEval-2015		
		Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1
Sudoku_iter	–	52.9	52.9	52.9	67.2	67.2	67.2	60.8	59.1	59.9
Wordsim_iter	*W2vSim*	48.5	48.5	48.5	72.1	72.1	72.1	57.8	51.1	54.2
	SRP2vSim	**56.1**	**56.1**	**56.1**	**75.0**	**75.0**	**75.0**	**69.2**	**62.6**	**65.8**

Table 1: Performance comparison of our WSD systems with the baseline WSD system (*Sudoku_iter*).

Table 1 shows that our proposed WSD system, *Wordsim_iter,* achieved the significantly improved performance compared to our baseline system, *Sudoku_iter* (p-value < 0.01), in all of the datasets. From these results, we are able to confirm that WSD performance can be improved by ignoring the words that are less similar with the ambiguous word. It also proved that WSD performance is determined by the criteria for selecting the contextual words. In case of word similarity measurement, our WSD system using *SRP2vSim* achieved significant improvement in performance compared to one using *W2vSim* (p-value < 0.05). Meanwhile, when we used *W2vSim* in the SemEval-2015 dataset, its result was significantly lower than the baseline (p-value < 0.01). On the other hand, *Wordsim_iter with SRP2vSim* has higher performances (4.5 %p and 10 %p) than *Sudoku_iter* and *Wordsim_iter* with *W2vSim,* respectively. Through these results, we think that the word vector representation from the knowledge-based graph is more adaptable than the traditional word vector representation for WSD tasks because the semantic information is more important than the syntactic information of words in the WSD tasks.

Approach	System	Semeval-2013	Semeval-2015	Macro Avg F_1
Unsupervised (Knowledge-based)	Moro 14	66.4	**70.3**	68.4
	Agirre 14	62.9	63.3	63.1
	Apidianaki 15	–	64.7	–
	Tripodi 17	70.8	–	–
	Wordsim_iter$_{SRP2vSim}$	**75.0**	65.8	**70.4**
Supervised	Zhong 10	66.3	69.7	68.0
	Weissenborn 15	71.5	**75.4**	**73.5**
	Raganato 17	66.9	71.5	69.2
	Pasini 17	65.5	68.6	67.1

Table 2: Performance comparison of our WSD system with state-of-the-art BabelNet-based unsupervised and supervised WSD systems.

In Table 2, we can compare our WSD approach using the *Wordsim_iter* method and *SRP2vSim* word similarity measurement (Wordsim_iter$_{SRP2vSim}$) to other Knowledge-based WSD systems introduced in

[5] https://github.com/mmihaltz/word2vec-GoogleNews-vectors

"[Alimta] is a [powder] that is [made_up] into a [solution] for [infusion] (drip into a [vein])."

Alimta: Chemotherapy drug manufactured and marketed by Eli Lilly and Company.
powder: A solid substance in the form of tiny loose particles.
make up: Form or compose
solution: A homogeneous mixture of two or more substances
infusion: The passive introduction of a substance into a vein or between tissues
vein: A blood vessel that carries blood from the capillaries toward the heart

Figure 5: An example sentence and definitions of the correct word senses. The term '[]' denotes an ambiguous word.

the Section 2, such as Moro 14 (Moro et al., 2014), Agirre 14 (Agirre et al., 2014), Apidianaki 15 (Apidanaki and Gong, 2015) and Tripodi 17 (Tripodi and Pelillo, 2017). In addition, we compared Wordsim_iter$_{SCP2vSim}$ to several supervised WSD systems, such as Zhong 10 (Zhong and Ng, 2010), Weissenborn 15 (Weissenborn et al., 2015), Raganato 17 (Raganato et al., 2017) and Pasini 17 (Pasini and Navigli., 2017). The results show that our WSD system surpassed all other state-of-the-art WSD systems with a large margin in the SemEval-2013 dataset. In the SemEval-2015 dataset, our WSD system has similar performance to the Agirre 14 and Apidianaki 15 and it has somewhat less performance than the state-of-the-art knowledge-based WSD system, Moro 14. This is due to the nature of the SemEval-2015 data, Moro 14 is designed to simultaneously analyzes WSD and entity linking. However, in terms of the macro average score of SemEval-2013 and SemEval-2015, Wordsim_iter$_{SRP2vSim}$ shows higher performance than the Moro 14. On the other hands, unsupervised knowledge-based approaches, including our system, generally has poorer performance than supervised approaches in the SemEval-2015 dataset. Especially, Weissenborn 15, a hybrid supervised WSD system that jointly has trained the WSD model and the entity linking model, achieved higher performance on macro average than our WSD system. Nevertheless, fewer limitation on the analyzable word set of our model makes it relatively more competitive than its counterpart state-of-the-art supervised-based WSD models.

4.4 Error Analysis

Despite the competitiveness of our system, the greedy algorithmic characteristic of the iterative subgraph-based algorithm has a negative effect on its performance. In particular, some previously analyzed words can affect other words and it makes them mis-disambiguated. For example, in the sentence of Figure 5, there are 6 ambiguous words: 'Alimta', 'powder', 'made up', 'solution', 'infusion' and 'vein'. Our WSD system wrongly determined 'made up' as "Apply make-up or cosmetics to one's face to appear prettier" because this wrong sense is more related with a previously determined sense 'powder' than correct sense in Figure 5. In addition, the mis-disambiguated 'made up' and previously analyzed words of 'powder' and 'solution' leads the meaning of 'infusion' as a wrong sense "A solution obtained by steeping a substance." If we decide the meaning of 'made up' and 'infusion' by regarding a word 'vein,' then we can disambiguate the words 'made up' and 'infusion' correctly.

5 Conclusions and Future Work

In this paper, we propose a knowledge-based WSD method that restricts contextual words based on the similarities between the ambiguous words and content words. We first measure the similarities of the words in an input document and ambiguous word and selectively create context of the ambiguous word with the words over the certain threshold. In addition, we further suggest a novel similarity calculating method suitable for our WSD method. Our WSD system significantly improves a baseline WSD system and has a higher performance than the state-of-the-art unsupervised knowledge-based WSD systems.

Our WSD system is based on an iterative subgraph reconstruction approach that determines the sense of a word in order. This method has been proposed to solve the computational complexity problem of finding the optimal combination among all possible set of senses. However, due to the nature of the greedy search, sometimes it makes hard to inference correct sense of the ambiguous word because of the error propagation from a previous result can determined answer.

For the future work, we plan to extend our WSD system to the multi-lingual system. In particular, we are going to research to generate of multi-lingual word vector representation using the Babel synset information. Another possible future work is to use the Beam search (Ow and Morton, 1988) to compensate for the drawbacks of the iterative subgraph reconstruction's greedy algorithmic characteristics.

Acknowledgements

This work was supported by the Korea Evaluation Institute of Industrial Technology grant funded by the Korea government (No. 10080681, Technical development of Korean speech recognition system in vehicle), Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (2017-0-00255, Autonomous digital companion framework and application), and this research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (NRF-2015R1D1A1A01056907).

References

Marianna Apidianaki, and Li Gong. 2015. LIMSI: Translations as Source of Indirect Supervision for Multilingual All-Words Sense Disambiguation and Entity Linking. *In the proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 298-302.

Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, Vol. 40, No. 1, pp. 57-84.

Satanjeev Banerjee and Ted Pedersen. 2003. Extended gloss overlaps as a measure of semantic relatedness. *In proceedings of the IJCAI*, pp. 805–810.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. *In proceedings of the 45th annual meeting of the association of computational linguistics*, pp. 33-40.

Devendra Singh Chaplot, Pushpak Bhattacharyya, and Ashwin Paranjape. 2015. Unsupervised Word Sense Disambiguation Using Markov Random Field and Dependency Parser. *In proceedings of the AAAI*, pp. 2217-2223.

Devendra Singh Chaplot, and Ruslan Salakhutdinov. 2018. Knowledge-based Word Sense Disambiguation using Topic Models. *arXiv preprint arXiv:1801.01900*.

Yoan Gutiérrez, et al. 2013. UMCC_DLSI: reinforcing a ranking algorithm with sense frequencies and multidimensional semantic resources to solve multilingual word sense disambiguation. *In proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pp. 241-249.

Yael Karov, and Shimon Edelman. 1998. Similarity-based Word Sense Disambiguation. *Computational Linguistics*, Vol. 24, No. 1, pp. 41-59.

Quoc Le, and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *In proceedings of the International Conference on Machine Learning*, pp. 1188-1196.

Xiao Ling, Sameer Singh, and Daniel S. Weld. 2015. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics*, Vol. 3, pp. 315-328.

Steve L. Manion, and Raazesh Sainudiin. 2014. An Iterative 'Sudoku Style' Approach to Subgraph-based Word Sense Disambiguation. *In proceedings of the Third Joint Conference on Lexical and Computational Semantics*, pp. 40-50.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. *In proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 51-61.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeffery Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *In proceedings of the 26th Advances in Neural Information Processing Systems*, pp. 3111-3119.

Andrea Moro, Alessandro Raganato. and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, Vol. 2, pp. 231-244.

Andrea Moro, and Roberto Navigli. 2015. Semeval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. *In proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 288-297.

Roberto Navigli, and Paola Velardi. 2005. Structural semantic interconnections: a knowledge-based approach to word sense disambiguation. *IEEE transactions on pattern analysis and machine intelligence*, Vol. 27, No. 7, pp. 1075-1086.

Roberto Navigli, and Mirella Lapata. 2007. Graph Connectivity Measures for Unsupervised Word Sense Disambiguation. *In proceedings of the IJCAI*, pp.1683-1688.

Roberto Navigli, and Mirella Lapata. 2010. An experimental study of graph connectivity for unsupervised word sense disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.32, No.4, pp.678-692.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, Vol. 193, pp. 217-250.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. Semeval-2013 task 12: Multilingual word sense disambiguation. *In proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pp. 222-231.

Tommaso Pasini, and Roberto Navigli. 2017. Train-O-Matic: Large-Scale Supervised Word Sense Disambiguation in Multiple Languages without Manual Training Data. *In proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 78-88.

Sameer S. Pradhan, Edward Loper, Dmitriy Dligach and Martha Palmer. 2007. SemEval-2007 Task 17: English Lexical Sample, SRL and All Words. *In proceedings of the 4th International Workshop on Semantic Evaluations*, pp. 87-92.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017. Neural sequence learning models for word sense disambiguation. *In proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1156-1167.

Mark Sanderson. 1994. Word sense disambiguation and information retrieval. *In proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 142-151.

Volker Steinbiss, Bach-Hiep Tran, and Hermann Ney. 1995. Improvements in beam search. *In proceedings of the third International Conference on Spoken Language Processing*, pp. 2143-2146.

Christopher Stokoe, Michael P. Oakes, and John Tait. 2003. Word sense disambiguation in information retrieval revisited. *In proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 159-166.

Johan AK Suykens, and Joos Vandewalle. 1999. Least squares support vector machine classifiers. *Neural processing letters*, Vol. 9, No. 3, pp. 293-300.

Egidio Terra, and Charles LA Clarke. 2003. Frequency estimates for statistical word similarity measures. *In proceedings of the 2003 NAACL-HLT*, pp. 165-172.

Rocco Tripodi, and Marcello Pelillo. 2017. A game-theoretic approach to word sense disambiguation. *Computational Linguistics*, Vol. 9, No. 3, pp. 31-70.

David Vickrey, Luke Biewald., Marc Teyssier, and Daphne Koller. 2005. Word-sense disambiguation for machine translation. *In proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pp. 771-778.

Dirk Weissenborn, Leonhard Hennig, Feiyu Xu and Hans Uszkoreit. 2015. Multi-objective optimization for the joint disambiguation of nouns and named entities. *In proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing,* pp. 596-605.

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods, *In proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 42-49.

Zhi Zhong, and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. *In proceedings of the ACL 2010 system demonstrations*, pp. 78-83.

Learning Semantic Sentence Embeddings using Pair-wise Discriminator

Badri N. Patro* **Vinod K. Kurmi*** **Sandeep Kumar*** **Vinay P. Namboodiri**
Indian Institute of Technology, Kanpur
{badri,vinodkk,sandepkr,vinaypn}@iitk.ac.in

Abstract

In this paper, we propose a method for obtaining sentence-level embeddings. While the problem of securing word-level embeddings is very well studied, we propose a novel method for obtaining sentence-level embeddings. This is obtained by a simple method in the context of solving the paraphrase generation task. If we use a sequential encoder-decoder model for generating paraphrase, we would like the generated paraphrase to be semantically close to the original sentence. One way to ensure this is by adding constraints for true paraphrase embeddings to be close and unrelated paraphrase candidate sentence embeddings to be far. This is ensured by using a sequential pair-wise discriminator that shares weights with the encoder that is trained with a suitable loss function. Our loss function penalizes paraphrase sentence embedding distances from being too large. This loss is used in combination with a sequential encoder-decoder network. We also validated our method by evaluating the obtained embeddings for a sentiment analysis task. The proposed method results in semantic embeddings and outperforms the state-of-the-art on the paraphrase generation and sentiment analysis task on standard datasets. These results are also shown to be statistically significant.

1 Introduction

The problem of obtaining a semantic embedding for a sentence that ensures that the related sentences are closer and unrelated sentences are farther lies at the core of understanding languages. This would be relevant for a wide variety of machine reading comprehension and related tasks such as sentiment analysis. Towards this problem, we propose a supervised method that uses a sequential encoder-decoder framework for paraphrase generation. The task of generating paraphrases is closely related to the task of obtaining semantic sentence embeddings. In our approach, we aim to ensure that the generated paraphrase embedding should be close to the true corresponding sentence and far from unrelated sentences. The embeddings so obtained help us to obtain state-of-the-art results for paraphrase generation task.

Our model consists of a sequential encoder-decoder that is further trained using a pairwise discriminator. The encoder-decoder architecture has been widely used for machine translation and machine comprehension tasks. In general, the model ensures a 'local' loss that is incurred for each recurrent unit cell. It only ensures that a particular word token is present at an appropriate place. This, however, does not imply that the whole sentence is correctly generated. To ensure that the whole sentence is correctly encoded, we make further use of a pair-wise discriminator that encodes the whole sentence and obtains an embedding for it. We further ensure that this is close to the desired ground-truth embeddings while being far from other (sentences in the corpus) embeddings. This model thus provides a 'global' loss that ensures the sentence embedding as a whole is close to other semantically related sentence embeddings. This is illustrated in Figure 1. We further evaluate the validity of the sentence embeddings by using them for the task of sentiment analysis. We observe that the proposed sentence embeddings result in state-of-the-art performance for both these tasks.

* Equal contribution

Proceedings of the 27th International Conference on Computational Linguistics, pages 2715–2729
Santa Fe, New Mexico, USA, August 20-26, 2018.

Our contributions are: a) We propose a model for obtaining sentence embeddings for solving the paraphrase generation task using a pair-wise discriminator loss added to an encoder-decoder network. b) We show that these embeddings can also be used for the sentiment analysis task. c) We validate the model using standard datasets with a detailed comparison with state-of-the-art methods and also ensure that the results are statistically significant.

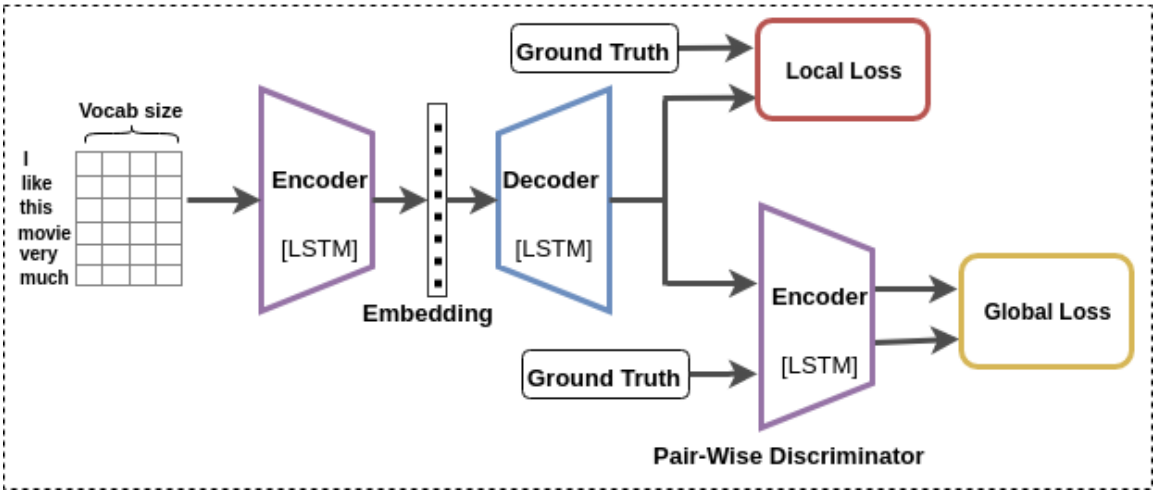


Figure 1: Pairwise Discriminator based Encoder-Decoder for Paraphrase Generation: This is the basic outline of our model which consists of an LSTM encoder, decoder and discriminator. Here the encoders share the weights. The discriminator generates discriminative embeddings for the Ground Truth-Generated paraphrase pair with the help of 'global' loss. Our model is jointly trained with the help of a 'local' and 'global' loss which we describe in section 3.

2 Related Work

Given the flexibility and diversity of natural language, it has always been a challenging task to represent text efficiently. There have been several hypotheses proposed for representing the same. (Harris, 1954; Firth, 1957; Sahlgren, 2008) proposed a distribution hypothesis to represent words, i.e., words which occur in the same context have similar meanings. One popular hypothesis is the bag-of-words (BOW) or Vector Space Model (Salton et al., 1975), in which a text (such as a sentence or a document) is represented as the bag (multiset) of its words. (Lin and Pantel, 2001) proposed an extended distributional hypothesis and (Deerwester et al., 1990; Turney and Littman, 2003) proposed a latent relation hypothesis, in which a pair of words that co-occur in similar patterns tend to have similar semantic relation. Word2Vec(Mikolov et al., 2013a; Mikolov et al., 2013b; Goldberg and Levy, 2014) is also a popular method for representing every unique word in the corpus in a vector space. Here, the embedding of every word is predicted based on its context (surrounding words). NLP researchers have also proposed phrase-level and sentence-level representations (Mitchell and Lapata, 2010; Zanzotto et al., 2010; Yessenalina and Cardie, 2011; Grefenstette et al., 2013; Mikolov et al., 2013b). (Socher et al., 2011; Kim, 2014; Lin et al., 2015; Yin et al., 2015; Kalchbrenner et al., 2014) have analyzed several approaches to represent sentences and phrases by a weighted average of all the words in the sentence, combining the word vectors in an order given by a parse tree of a sentence and by using matrix-vector operations. The major issue with BOW models and weighted averaging of word vectors is the loss of semantic meaning of the words, the parse tree approaches can only work for sentences because of its dependence on sentence parsing mechanism. (Socher et al., 2013; Le and Mikolov, 2014) proposed a method to obtain a vector representation for paragraphs and use it to for some text-understanding problems like sentiment analysis and information retrieval.

Many language models have been proposed for obtaining better text embeddings in Machine Translation (Sutskever et al., 2014; Cho et al., 2014; Vinyals and Le, 2015; Wu et al., 2016), question generation (Du et al., 2017), dialogue generation (Shang et al., 2015; Li et al., 2016b; Li et al., 2017a),

document summarization (Rush et al., 2015), text generation (Zhang et al., 2017; Hu et al., 2017; Yu et al., 2017; Guo et al., 2017; Liang et al., 2017; Reed et al., 2016) and question answering (Yin et al., 2016; Miao et al., 2016). For paraphrase generation task, (Prakash et al., 2016) have generated paraphrases using stacked residual LSTM based network. (Hasan et al., 2016) proposed a encoder-decoder framework for this task. (Gupta et al., 2017) explored a VAE approach to generate paraphrase sentences using recurrent neural networks. (Li et al., 2017b) used reinforcement learning for paraphrase generation task.

3 Method

In this paper, we propose a text representation method for sentences based on an encoder-decoder framework using a pairwise discriminator for paraphrase generation and then fine tune these embeddings for sentiment analysis task. Our model is an extension of *seq2seq* (Sutskever et al., 2014) model for learning better text embeddings.

3.1 Overview

Task: In the paraphrase generation problem, given an input sequence of words $X = [x_1, ..., x_L]$, we need to generate another output sequence of words $Y = [q_1, ..., q_T]$ that has the same meaning as X. Here L and T are not fixed constants. Our training data consists of M pairs of paraphrases $\{(X_i, Y_i)\}_{i=1}^{M}$ where X_i and Y_i are the paraphrase of each other.

Our method consists of three modules as illustrated in Figure 2: first is a Text Encoder which consists of LSTM layers, second is LSTM-based Text Decoder and last one is an LSTM-based Discriminator module. These are shown respectively in part 1, 2, 3 of Figure 2. Our network with all three parts is trained end-to-end. The weight parameters of encoder and discriminator modules are shared. Instead of taking a separate discriminator, we shared it with the encoder so that it learns the embedding based on the 'global' as well as 'local' loss. After training, at test time we used encoder to generate feature maps and pass it to the decoder for generating paraphrases. These text embeddings can be further used for other NLP tasks such as sentiment analysis.

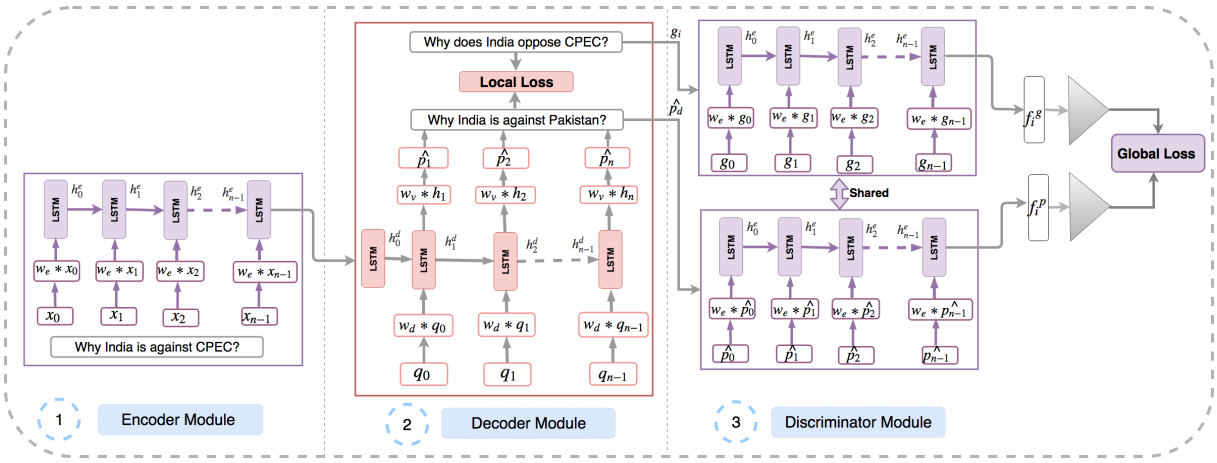


Figure 2: This is an overview of our model. It consists of 3 parts: 1) LSTM-based Encoder module which encodes a given sentence, 2) LSTM-based Decoder Module which generates natural language paraphrases from the encoded embeddings and 3) LSTM-based pairwise Discriminator module which shares its weights with the Encoder module and this whole network is trained with local and global loss.

3.2 Encoder-LSTM

We use an LSTM-based encoder to obtain a representation for the input question X_i, which is represented as a matrix in which every row corresponds to the vector representation of each word. We use a one-hot vector representation for every word and obtain a word embedding c_i for each word using a Temporal

CNN (Zhang et al., 2015; Palangi et al., 2016) module that we parameterize through a function $G(X_i, W_e)$ where W_e are the weights of the temporal CNN. Now this word embedding is fed to an LSTM-based encoder which provides encoding features of the sentence. We use LSTM (Hochreiter and Schmidhuber, 1997) due to its capability of capturing long term memory (Palangi et al., 2016). As the words are propagated through the network, the network collects more and more semantic information about the sentence. When the network reaches the last word (L_{th} word), the hidden state h_L of the network provides a semantic representation of the whole sentence conditioned on all the previously generated words $(q_0, q_1..., q_t)$. Question sentence encoding feature f_i is obtained after passing through an LSTM which is parameterized using the function $F(C_i, W_l)$ where W_l are the weights of the LSTM. This is illustrated in part 1 of Figure 2.

3.3 Decoder-LSTM

The role of decoder is to predict the probability for a whole sentence, given the embedding of input sentence (f_i). RNN provides a nice way to condition on previous state value using a fixed length hidden vector. The conditional probability of a sentence token at a particular time step is modeled using an LSTM as used in machine translation (Sutskever et al., 2014). At time step t, the conditional probability is denoted by $P(q_t|f_i, q_0, .., q_{t-1}) = P(q_t|f_i, h_t)$, where h_t is the hidden state of the LSTM cell at time step t. h_t is conditioned on all the previously generated words $(q_0, q_1.., q_{t-1})$ and q_t is the next generated word.

Generated question sentence feature $\hat{p_d} = \{\hat{p_1}, \ldots, \hat{p_T}\}$ is obtained by decoder LSTM which is parameterized using the function $D(f_i, W_{dl})$ where W_{dl} are the weights of the decoder LSTM. The output of the word with maximum probability in decoder LSTM cell at step k is input to the LSTM cell at step $k+1$ as shown in Figure 2. At $t = -1$, we are feeding the embedding of input sentence obtained by the encoder module. $\hat{Y_i} = \{\hat{q_0}, \hat{q_1}, ..., \hat{q}_{T+1}\}$ are the predicted question tokens for the input X_i. Here, we are using $\hat{q_0}$ and $\hat{q}_{T+1}$ as the special START and STOP token respectively. The predicted question token ($\hat{q_i}$) is obtained by applying Softmax on the probability distribution $\hat{p_i}$. The question tokens at different time steps are given by the following equations where LSTM refers to the standard LSTM cell equations:

$$
\begin{aligned}
&d_{-1} = \text{Encoder}(f_i) \\
&h_0 = \text{LSTM}(d_{-1}) \\
&d_t = W_d * q_t, \forall t \in \{0, 1, 2, ...T-1\} \\
&h_{t+1} = \text{LSTM}(d_t, h_t), \forall t \in \{0, 1, 2, ...T-1\} \\
&\hat{p}_{t+1} = W_v * h_{t+1} \\
&\hat{q}_{t+1} = \text{Softmax}(\hat{p}_{t+1}) \\
&\text{Loss}_{t+1} = \text{loss}(\hat{q}_{t+1}, q_{t+1})
\end{aligned}
\tag{1}
$$

Where $\hat{q}_{t+1}$ is the predicted question token and q_{t+1} is the ground truth one. In order to capture local label information, we use the Cross Entropy loss which is given by the following equation:

$$
L_{local} = \frac{-1}{T} \sum_{t=1}^{T} q_t \text{logP}(\hat{q}_t|q_0, ..q_{t-1}) \tag{2}
$$

Here T is the total number of sentence tokens, $\text{P}(\hat{q}_t|q_0, ..q_{t-1})$ is the predicted probability of the sentence token, q_t is the ground truth token.

3.4 Discriminative-LSTM

The aim of the Discriminative-LSTM is to make the predicted sentence embedding f_i^p and ground truth sentence embedding f_i^g indistinguishable as shown in Figure 2. Here we pass $\hat{p_d}$ to the shared encoder-LSTM to obtain f_i^p and also the ground truth sentence to the shared encoder-LSTM to obtain f_i^g. The discriminator module estimates a loss function between the generated and ground truth paraphrases. Typically, the discriminator is a binary classifier loss, but here we use a global loss, similar to (Reed et al.,

2016) which acts on the last hidden state of the recurrent neural network (LSTM). The main objective of this loss is to bring the generated paraphrase embeddings closer to its ground truth paraphrase embeddings and farther from the other ground truth paraphrase embeddings (other sentences in the batch). Here our discriminator network ensures that the generated embedding can reproduce better paraphrases. We are using the idea of sharing discriminator parameters with encoder network, to enforce learning of embeddings that not only minimize the local loss (cross entropy), but also the global loss.

Suppose the predicted embeddings of a batch is $e_p = [f_1^p, f_2^p, ..f_N^p]^T$, where f_i^p is the sentence embedding of i^{th} sentence of the batch. Similarly ground truth batch embeddings are $e_g = [f_1^g, f_2^g, ..f_N^g]^T$, where N is the batch size, $f_i^p \in \mathbb{R}^d$ $f_i^g \in \mathbb{R}^d$. The objective of global loss is to maximize the similarity between predicted sentence f_i^p with the ground truth sentence f_i^g of i^{th} sentence and minimize the similarity between i^{th} predicted sentence, f_i^p, with j^{th} ground truth sentence, f_j^g, in the batch. The loss is defined as

$$L_{global} = \sum_{i=1}^{N}\sum_{j=1}^{N} \max(0, ((f_i^p \cdot f_j^g) - (f_i^p \cdot f_i^g) + 1)) \tag{3}$$

Gradient of this loss function is given by

$$\left(\frac{dL}{de_p}\right)_i = \sum_{j=1, j\neq i}^{N} (f_j^g - f_i^g) \tag{4}$$

$$\left(\frac{dL}{de_g}\right)_i = \sum_{j=1, j\neq i}^{N} (f_j^p - f_i^p) \tag{5}$$

3.5 Cost function

Our objective is to minimize the total loss, that is the sum of local loss and global loss over all training examples. The total loss is:

$$L_{total} = \frac{1}{M}\sum_{i=1}^{M}(L_{local} + L_{global}) \tag{6}$$

Where M is the total number of examples, L_{local} is the cross entropy loss, L_{global} is the global loss.

Dataset	Model	BLEU1	BLEU2	BLEU3	BLEU4	ROUGE	METEOR
50K	ED-L(Base Line)	33.7	22.3	18.0	12.1	35.3	14.3
	EDD-G	40.7	28.3	21.1	16.1	39.7	19.6
	EDD-LG	40.9	28.6	21.3	16.1	40.2	19.8
	EDD-LG(shared)	**41.1**	**29.0**	**21.5**	**16.5**	**40.6**	**20.1**
100K	ED-L(Base Line)	35.1	25.4	19.6	14.4	37.4	15.4
	EDD-G	42.1	29.4	21.6	16.4	41.4	20.4
	EDD-LG	44.2	31.6	22.1	17.9	43.6	22.1
	EDD-LG(shared)	**45.7**	**32.4**	**23.8**	**17.9**	**44.9**	**23.1**

Table 1: Analysis of variants of our proposed method on Quora Dataset as mentioned in section 4.1.3. Here L and G refer to the Local and Global loss and shared represents the parameter sharing between the discriminator and encoder module. As we can see that our proposed method EDD-LG(shared) clearly outperforms the other ablations on all metrics and detailed analysis is present in section 4.1.3.

4 Experiments

We perform experiments to better understand the behavior of our proposed embeddings. To achieve this, we benchmark Encoder Decoder Discriminator Local-Global (shared) (EDD-LG(shared)) embeddings on two text understanding problems, Paraphrase Generation and Sentiment Analysis. We use the Quora question pairs dataset [1] for paraphrase generation and Stanford Sentiment Treebank dataset (Socher et al.,

[1]website: `https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs`

2013) for sentiment analysis. In this section we describe the different datasets, experimental setup and results of our experiments.

4.1 Paraphrase Generation

Paraphrase generation is an important problem in many NLP applications such as question answering, information retrieval, information extraction, and summarization. It involves generation of similar meaning sentences.

4.1.1 Dataset

We use the newly released Quora question pairs dataset for this task. The dataset consists of over 400K potential question duplicate pairs. We have IDs for each question in the pair, the full text for each question, and a binary value that indicates whether the questions in the pair are truly a duplicate of each-other. Wherever the binary value is 1, the question in the pair are not identical; they are rather paraphrases of each-other. So, for our study, we choose all such question pairs with binary value 1. There are a total of 149K such questions. Some examples of question and their generated paraphrases can be found in Table 3. More results are present in the appendix.

Dataset	Model	BLEU1	METEOR	TER
50K	Unsupervised VAE (Gupta et al., 2017)	8.3	12.2	83.7
	VAE-S (Gupta et al., 2017)	11.9	17.4	69.4
	VAE-SVG (Gupta et al., 2017)	17.1	21.3	63.1
	VAE-SVG-eq (Gupta et al., 2017)	17.4	**21.4**	61.9
	EDD-G (**Ours**)	40.7	19.7	51.2
	EDD-LG(**Ours**)	40.9	19.8	51.0
	EDD-LG(shared)(**Ours**)	**41.1**	20.1	**50.8**
100K	Unsupervised (Gupta et al., 2017)	10.6	14.3	79.9
	VAE-S (Gupta et al., 2017)	17.5	21.6	67.1
	VAE-SVG (Gupta et al., 2017)	22.5	24.6	55.7
	VAE-SVG-eq (Gupta et al., 2017)	22.9	**24.7**	55.0
	EDD-G (**Ours**)	42.1	20.4	49.9
	EDD-LG(**Ours**)	44.2	22.1	48.3
	EDD-LG(shared)(**Ours**)	**45.7**	23.1	**47.5**

Table 2: Analysis of Baselines and State-of-the-Art methods for paraphrase generation on Quora dataset. As we can see clearly that our model outperforms the state-of-the-art methods by a significant margin in terms of BLEU and TER scores. Detailed analysis is present in section 4.1.4. A lower TER score is better whereas for the other metrics, a higher score is better. Details for the metrics are present in the appendix.

4.1.2 Experimental Protocols

We follow the experimental protocols mentioned in (Gupta et al., 2017) for the Quora dataset. In our experiments, we divide the dataset into 2 parts 145K and 4K question pairs. We use these as our training and testing sets. We further divide the training set into 50K and 100K dataset sizes and use the rest 45K as our validation set. We trained our model end-to-end using local loss (cross entropy loss) and global loss. We have used RMSPROP optimizer to update the model parameter and found these hyperparameter values to work best to train the Paraphrase Generation Network: learning rate $= 0.0008$, batch size $= 150, \alpha = 0.99, \epsilon = 1e-8$. We have used learning rate decay to decrease the learning rate on every epoch by a factor given by:

$$\text{Decay\_factor} = exp\left(\frac{log(0.1)}{a * b}\right)$$

where $a = 1500$ and $b = 1250$ are set empirically.

S.No	**Original Question**	**Ground Truth Paraphrase**	**Generated Paraphrase**
1	Is university really worth it?	Is college even worth it?	Is college really worth it?
2	Why India is against CPEC?	Why does India oppose CPEC?	Why India is against Pakistan?
3	How can I find investors for my tech startup?	How can I find investors for my startup on Quora?	How can I find investors for my startup business?
4	What is your view/opinion about surgical strike by the Indian Army?	What world nations think about the surgical strike on POK launch pads and what is the reaction of Pakistan?	What is your opinion about the surgical strike on Kashmir like?
5	What will be Hillary Clinton's strategy for India if she becomes US President?	What would be Hillary Clinton's foreign policy towards India if elected as the President of United States?	What will be Hillary Clinton's policy towards India if she becomes president?

Table 3: Examples of Paraphrase generation on Quora Dataset. We observe that our model is able to understand abbreviations as well and then ask questions on the basis of that as is the case in the second example.

4.1.3 Ablation Analysis

We experimented with different variations for our proposed method. We start with baseline model which we take as a simple encoder and decoder network with only the local loss (ED-Local) (Sutskever et al., 2014). Further we have experimented with encoder-decoder and a discriminator network with only global loss (EDD-Global) to distinguish the ground truth paraphrase with the predicted one. Another variation of our model is used both the global and local loss (EDD-LG). The discriminator is the same as our proposed method, only the weight sharing is absent in this case. Finally, we make the discriminator share weights with the encoder and train this network with both the losses (EDD-LG(shared)). The analyses are given in table 1. Among the ablations, the proposed EDD-LG(shared) method works way better than the other variants in terms of BLEU and METEOR metrics by achieving an improvement of 8% and 6% in the scores respectively over the baseline method for 50K dataset and an improvement of 10% and 7% in the scores respectively for 100K dataset.

4.1.4 Baseline and State-of-the-Art Method Analysis

There has been relatively less work on this dataset and the only work which we came across was that of (Gupta et al., 2017). We further compare our method EDD-LG(shared) model with their VAE-SVG-eq which is the current state-of-the-art on Quora datset. Also we provide comparisons with other methods proposed by them in table 2. As we can see from the table that we achieve a significant improvement of 24% in BLEU score and 11% in TER score (A lower TER score is better) for 50K dataset and similarly 22% in BLEU score and 7.5% in TER score for 100K dataset.

4.1.5 Statistical Significance Analysis

We have analysed statistical significance (Demšar, 2006) for our proposed embeddings against different ablations and the state-of-the-art methods for the paraphrase generation task. The Critical Difference (CD) for Nemenyi (Fišer et al., 2016) test depends upon the given α (confidence level, which is 0.05 in our case) for average ranks and N (number of tested datasets). If the difference in the rank of the two methods lies within CD, then they are not significantly different, otherwise they are statistically different. Figure 3 visualizes the post hoc analysis using the CD diagram. From the figure, it is clear that our embeddings work best and the results are significantly different from the state-of-the-art methods.

Model	Error Rate (Fine-Grained)
Naive Bayes (Socher et al., 2013)	59.0
SVMs (Socher et al., 2013)	59.3
Bigram Naive Bayes (Socher et al., 2013)	58.1
Word Vector Averaging (Socher et al., 2013)	67.3
Recursive Neural Network (Socher et al., 2013)	56.8
Matrix Vector-RNN (Socher et al., 2013)	55.6
Recursive Neural Tensor Network (Socher et al., 2013)	54.3
Paragraph Vector (Le and Mikolov, 2014)	51.3
EDD-LG(shared) (**Ours**)	**35.6**

Table 4: Performance of our method compared to other approaches on the Stanford Sentiment Treebank Dataset. The error rates of other methods are reported in (Le and Mikolov, 2014)

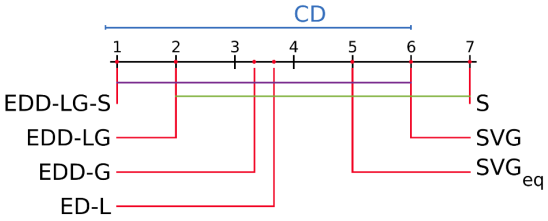


Figure 3: The mean rank of all the models on the basis of BLEU score are plotted on the x-axis. Here EDD-LG-S refers to our EDD-LG shared model and others are the different variations of our model described in section 4.1.3 and the models on the right are the different variations proposed in (Gupta et al., 2017). Also the colored lines between the two models represents that these models are not significantly different from each other. CD=5.199,p=0.0069

4.2 Sentiment Analysis with Stanford Sentiment Treebank (SST) Dataset

4.2.1 Dataset

This dataset consists of sentiment labels for different movie reviews and was first proposed by (Pang and Lee, 2005). (Socher et al., 2013) extended this by parsing the reviews to subphrases and then fine-graining the sentiment labels for all the phrases of movies reviews using Amazon Mechanical Turk. The labels are classified into 5 sentiment classes, namely {Very Negative, Negative, Neutral, Positive, Very Positive}. This dataset contains a total 126k phrases for training set, 30k phrases for validation set and 66k phrases for test set.

4.2.2 Tasks and Baselines

In (Socher et al., 2013), the authors propose two ways of benchmarking. We consider the 5-way fine-grained classification task where the labels are {Very Negative, Negative, Neutral, Positive, Very Positive}. The other axis of variation is in terms of whether we should label the entire sentence or all phrases in the sentence. In this work we only consider labeling all the phrases. (Socher et al., 2013) apply several methods to this dataset and we show their performance in table 4.

4.2.3 Experimental Protocols

We follow the experimental protocols as described in (Socher et al., 2013). To make use of the available labeled data, in our model, each subphrase is treated as an independent sentence and we learn the representations for all the subphrases in the training set. After learning the vector representations for training sentences and their subphrases, we feed them to a logistic regression to learn a predictor of the movie rating. At test time, we freeze the vector representation for each word, and learn the representations for the sentences using gradient descent. Once the vector representations for the test sentences are learned, we input them to a logistic regression model to predict the movie

Phrase ID	Phrase	Sentiment
162970	The heaviest, most joyless movie	Very Negative
159901	Even by dumb action-movie standards, Ballistic : Ecks vs. Sever is a dumb action movie.	
158280	Nonsensical, dull "cyber-horror" flick is a grim, hollow exercise in flat scares and bad acting	
159050	This one is pretty miserable, resorting to string-pulling rather than legitimate character development and intelligent plotting.	
157130	The most hopelessly monotonous film of the year, noteworthy only for the gimmick of being filmed as a single unbroken 87-minute take.	
156368	No good jokes, no good scenes, barely a moment	Negative
157880	Although it bangs a very cliched drum at times	
159269	They take a long time to get to its gasp-inducing ending.	
157144	Noteworthy only for the gimmick of being filmed as a single unbroken 87-minute	
156869	Done a great disservice by a lack of critical distance and a sad trust in liberal arts college bumper sticker platitudes	
221765	A hero can stumble sometimes.	Neutral
222069	Spiritual rebirth to bruising defeat	
218959	An examination of a society in transition	
221444	A country still dealing with its fascist past	
156757	Have to know about music to appreciate the film's easygoing blend of comedy and romance	
157663	A wildly funny prison caper.	Positive
157850	This is a movie that's got oodles of style and substance.	
157879	Although it bangs a very cliched drum at times, this crowd-pleaser's fresh dialogue, energetic music, and good-natured spunk are often infectious.	
156756	You don't have to know about music to appreciate the film's easygoing blend of comedy and romance.	
157382	Though of particular interest to students and enthusiast of international dance and world music, the film is designed to make viewers of all ages, cultural backgrounds and rhythmic ability want to get up and dance.	
162398	A comic gem with some serious sparkles.	Very Positive
156238	Delivers a performance of striking skill and depth	
157290	What Jackson has accomplished here is amazing on a technical level.	
160925	A historical epic with the courage of its convictions about both scope and detail.	
161048	This warm and gentle romantic comedy has enough interesting characters to fill several movies, and its ample charms should win over the most hard-hearted cynics.	

Table 5: Examples of Sentiment classification on test set of kaggle competition dataset.

rating. In order to train a sentiment classification model, we have used RMSPROP, to optimize the classification model parameter and we found these hyperparameter values to be working best for our case: learning rate $= 0.00009$, batch size $= 200, \alpha = 0.9, \epsilon = 1e-8$.

4.2.4 Results

We report the error rates of different methods in table 4. We can clearly see that the bag-of-words or bag-of-n-grams models (NB, SVM, BiNB) perform poorly. Also the advanced methods (such as Recursive Neural Network (Socher et al., 2013)) perform much better on the task of fine-grained sentiment analysis. Our method outperforms all these methods by an absolute margin of 15.7% which is a significant increase considering the rate of progress on this task. We have also uploaded our models to the online competition on Rotten Tomatoes dataset [2] and obtained an accuracy of 62.606% on their test-set of 66K phrases.
We also provide 5 examples for each sentiment in table 5. We can see clearly that our proposed embeddings are able to get the complete meaning of smaller as well as larger sentences. For example, our model classifies 'Although it bangs a very cliched drum at times' as Negative and 'Although it bangs a very cliched drum at times, this crowd-pleaser's fresh dialogue, energetic music, and good-natured spunk are often infectious.' as positive showing that it is able to understand the finer details of language. More results and visualisations showing the part of the phrase to which the model attends while classifying are present in the appendix. The link for the project website and code is provided here [3].

[2]website: `www.kaggle.com/c/sentiment-analysis-on-movie-reviews`
[3]Project website: `https://badripatro.github.io/Question-Paraphrases/`

5 Conclusion

In this paper we have proposed a sentence embedding using a sequential encoder-decoder with a pairwise discriminator. We have experimented with this text embedding method for paraphrase generation and sentiment analysis. We also provided experimental analysis which justifies that a pairwise discriminator outperforms the previous state-of-art methods for NLP tasks. We also performed ablation analysis for our method, and our method outperforms all of them in terms of BLEU, METEOR and TER scores. We plan to generalize this to other text understanding tasks and also extend the same idea in vision domain.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proc. of ACL workshop on Intrinsic and Extrinsic Evaluation measures for Machine Translation and/or Summarization*, volume 29, pages 65–72.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. pages 1724–1734.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. 1:1342–1352.

John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Darja Fišer, Tomaž Erjavec, and Nikola Ljubešić. 2016. Janes v0. 4: Korpus slovenskih spletnih uporabniških vsebin. *Slovenščina*, 2(4):2.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

E Grefenstette, G Dinu, Y Zhang, M Sadrzadeh, and M Baroni. 2013. Multi-step regression learning for compositional distributional semantics. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)–Long Papers*, pages 131–142.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2017. Long text generation via adversarial training with leaked information. *arXiv preprint arXiv:1709.08624*.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2017. A deep generative framework for paraphrase generation. *arXiv preprint arXiv:1709.05074*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Sadid A Hasan, Bo Liu, Joey Liu, Ashequl Qadir, Kathy Lee, Vivek Datla, Aaditya Prakash, and Oladimeji Farri. 2016. Neural clinical paraphrase generation with attention. In *Proceedings of the Clinical Natural Language Processing Workshop (ClinicalNLP)*, pages 42–53.

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

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016a. Visualizing and understanding neural models in nlp. In *Proceedings of NAACL-HLT*, pages 681–691.

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

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017a. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2017b. Paraphrase generation with deep reinforcement learning. *arXiv preprint arXiv:1711.00279*.

Xiaodan Liang, Zhiting Hu, Hao Zhang, Chuang Gan, and Eric P Xing. 2017. Recurrent topic-transition gan for visual paragraph generation. *CoRR, abs/1703.07022*, 2.

Dekang Lin and Patrick Pantel. 2001. Dirt@ sbt@ discovery of inference rules from text. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 323–328. ACM.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 899–907.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out:Proceedings of the ACL-04 workshop*.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 182–190. Association for Computational Linguistics.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International Conference on Machine Learning*, pages 1727–1736.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

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

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(4):694–707.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Aaditya Prakash, Sadid A Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual lstm networks. *arXiv preprint arXiv:1610.03098*.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. 2016. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Magnus Sahlgren. 2008. The distributional hypothesis. *Italian Journal of Disability Studies*, 20:33–53.

Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1577–1586.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

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

Peter D Turney and Michael L Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems (TOIS)*, 21(4):315–346.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. corr abs/1609.08144.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2010. Paraphrasing headlines by machine translation: Sentential paraphrase acquisition and generation using google news. *LOT Occasional Series*, 16:169–183.

Ainur Yessenalina and Claire Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 172–182. Association for Computational Linguistics.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2972–2978. AAAI Press.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858.

Fabio Massimo Zanzotto, Ioannis Korkontzelos, Francesca Fallucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1263–1271. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017. Adversarial feature matching for text generation. *arXiv preprint arXiv:1706.03850*.

A Appendix

A.1 Quantitative Evaluation

We use automatic evaluation metrics which are prevalent in machine translation domain: BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), ROUGE-n (Lin, 2004) and Translation Error Rate (TER) (Snover et al., 2006). Previous work has shown that these metrics can perform well for the paraphrase recognition task (Madnani et al., 2012) and correlate well with human judgments in evaluating generated paraphrases (Wubben et al., 2010). BLEU considers exact match between reference paraphrases and system generated paraphrases using the concept of modified n-gram precision and brevity penalty, whereas ROUGE considers recall for the same. METEOR also uses stemming and synonyms (using WordNet) while calculating the score and is based on a combination of unigram-precision and unigram-recall with the reference paraphrases. TER is based on the number of edits (insertions, deletions, substitutions, shifts) required for a human to convert the system output into one of the reference paraphrases. We provided our results using all these metrics and compared it with existing baselines.

A.2 Paraphrase Generation

Here we provide some more examples of the paraphrase generation task in table 6. Our model is also able to generate sentences which capture higher level semantics like in the last example of table 6.

S.No	**Original Question**	**Ground Truth Paraphrase**	**Generated Paraphrase**
1	How do I add content on Quora?	How do I add content under a title at Quora?	How do I add images on Quora ?
2	Is it possible to get a long distance ex back?	Long distance relationship: How to win my ex-gf back?	Is it possible to get a long distance relationship back ?
3	How many countries are there in the world? Thanks!	How many countries are there in total?	How many countries are there in the world ? What are they ?
4	What is the reason behind abrupt removal of Cyrus Mistry?	Why did the Tata Sons sacked Cyrus Mistry?	What is the reason behind firing of Cyrus Mistry ?
5	What are some extremely early signs of pregnancy?	What are the common first signs of pregnancy? How can I tell if I'm pregnant? What are the symptoms?	What are some early signs of pregnancy ?
6	How can I improve my critical reading skills?	What are some ways to improve critical reading and reading comprehension skills?	How can I improve my presence of mind ?

Table 6: Examples of Paraphrase generation on Quora Dataset.

A.3 Sentiment Analysis

We also provide visualization of different parts of the sentence on which our model focuses while predicting the sentiment in Figure 4 and some more examples of the Sentiment analysis task on SST dataset in Table 7.

A.3.1 Sentiment Visualization of the sentence

(Li et al., 2016a) have proposed a mechanism to visualize language features. We conducted a toy experiment for our EDD-LG(shared) model. Figure 4 represents saliency heat map for EDD-LG(shared) model sentiment analysis. We obtained 60 dimensional feature maps for each word present in the target sentence. The heat map captures the measure of influence of the sentimental decision. In the heat map, each word of a sentence (from top to bottom, first word at top) represents its contribution for making the

sentimental decision. For example in the first image in 4, the word ‘comic’ contributed more (2nd word, row 10-20). Similarly in the second image, first, second, and third (‘A’,‘wildly’,‘funny’) words have more influence for making this sentence have a positive sentiment.

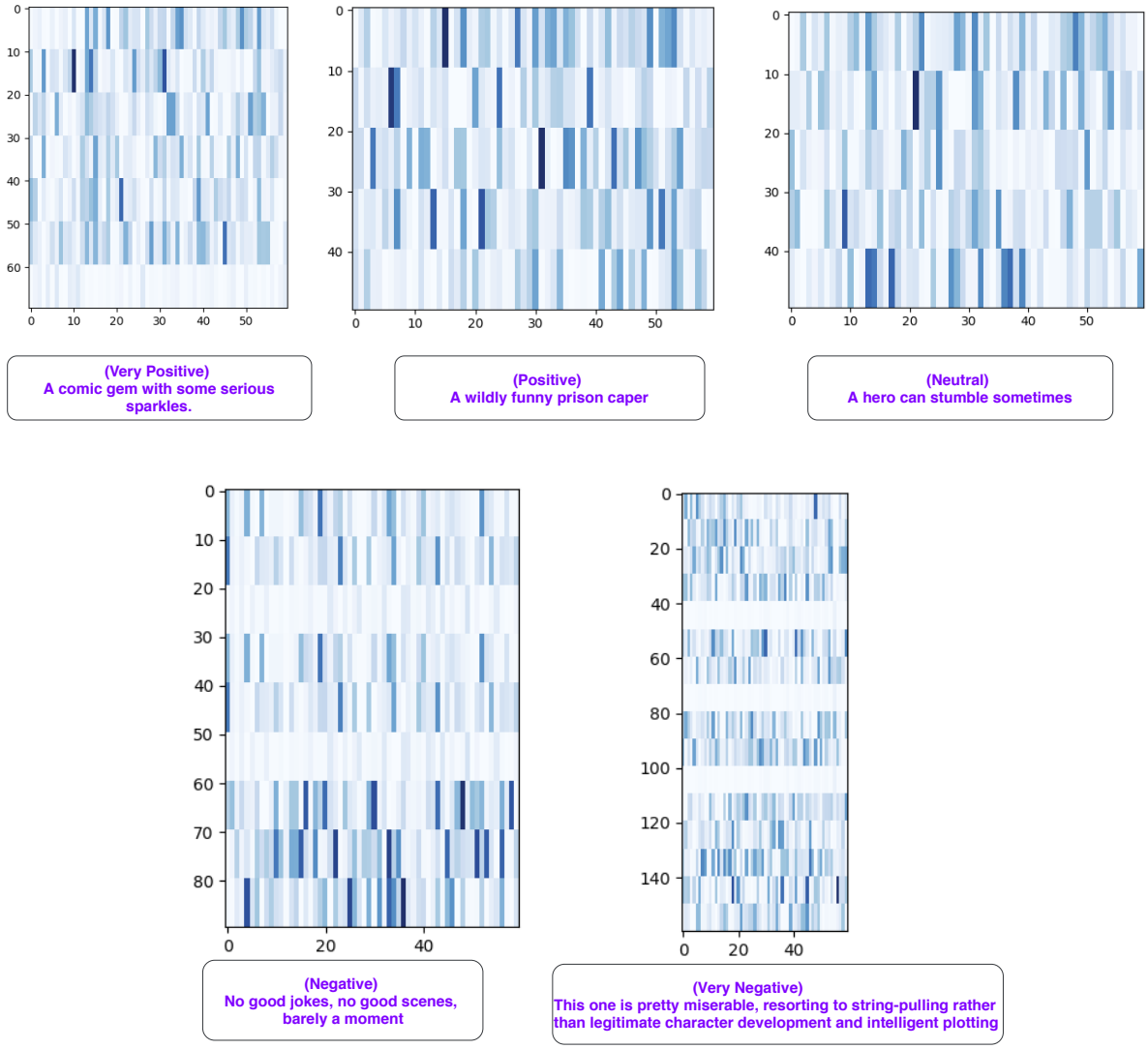


Figure 4: These are the visualisations for the sentiment analysis for some examples and we can clearly see that our model focuses on those words which we humans focus while deciding the sentiment for any sentence. In the second image, ‘wildly’ and ‘funny’ are emphasised more than the other words.

Phrase ID	Phrase	Sentiment
156628	The movie is just a plain old monster	Very Negative
157078	a really bad community theater production of West Side Story	
159749	Suffers from rambling , repetitive dialogue and the visual drabness endemic to digital video .	
163425	The picture , scored by a perversely cheerful Marcus Miller accordionharmonicabanjo abomination , is a monument to bad in all its florid variety .	
163483	lapses quite casually into the absurd	
163882	It all drags on so interminably it 's like watching a miserable relationship unfold in real time .	
164436	Your film becomes boring , and your dialogue is n't smart	
165179	Another big , dumb action movie in the vein of XXX , The Transporter is riddled with plot holes big enough for its titular hero to drive his sleek black BMW through .	
156567	It would be hard to think of a recent movie that has worked this hard to achieve this little fun	Negative
156689	A depressing confirmation	
157730	There 's not enough here to justify the almost two hours.	
157695	a snapshot of a dangerous political situation on the verge of coming to a head	
158814	It is ridiculous , of course	
159281	A mostly tired retread of several other mob tales.	
159632	We are left with a superficial snapshot that , however engaging , is insufficiently enlightening and inviting .	
159770	It 's as flat as an open can of pop left sitting in the sun .	
156890	liberal arts college bumper sticker platitudes	Neutral
160247	the movie 's power as a work of drama	
160754	Schweig , who carries the film on his broad , handsome shoulders	
160773	to hope for any chance of enjoying this film	
201255	also examining its significance for those who take part	
201371	those who like long books and movies	
221444	a country still dealing with its fascist past	
222102	used to come along for an integral part of the ride	
157441	the film is packed with information and impressions .	Positive
157879	Although it bangs a very cliched drum at times , this crowd-pleaser 's fresh dialogue , energetic music , and good-natured spunk are often infectious.	
157663	A wildly funny prison caper.	
157749	This is one for the ages.	
157806	George Clooney proves he 's quite a talented director and Sam Rockwell shows us he 's a world-class actor with Confessions of a Dangerous Mind .	
157850	this is a movie that 's got oodles of style and substance .	
157742	Kinnear gives a tremendous performance .	Very Positive
160562	The film is painfully authentic , and the performances of the young players are utterly convincing .	
160925	A historical epic with the courage of its convictions about both scope and detail.	
161048	This warm and gentle romantic comedy has enough interesting characters to fill several movies , and its ample charms should win over the most hard-hearted cynics .	
161459	is engrossing and moving in its own right	
162398	A comic gem with some serious sparkles .	
162779	a sophisticated , funny and good-natured treat , slight but a pleasure	
163228	Khouri then gets terrific performances from them all .	

Table 7: Examples of Sentiment classification on test set of kaggle dataset.

A Reassessment of Reference-Based Grammatical Error Correction Metrics

Shamil Chollampatt[1] and **Hwee Tou Ng**[1,2]
[1]NUS Graduate School for Integrative Sciences and Engineering
[2]Department of Computer Science, School of Computing
National University of Singapore
shamil@u.nus.edu,nght@comp.nus.edu.sg

Abstract

Several metrics have been proposed for evaluating grammatical error correction (GEC) systems based on grammaticality, fluency, and adequacy of the output sentences. Previous studies of the correlation of these metrics with human quality judgments were inconclusive, due to the lack of appropriate significance tests, discrepancies in the methods, and choice of datasets used. In this paper, we re-evaluate reference-based GEC metrics by measuring the system-level correlations with humans on a large dataset of human judgments of GEC outputs, and by properly conducting statistical significance tests. Our results show no significant advantage of GLEU over MaxMatch (M^2), contradicting previous studies that claim GLEU to be superior. For a finer-grained analysis, we additionally evaluate these metrics for their agreement with human judgments at the sentence level. Our sentence-level analysis indicates that comparing GLEU and M^2, one metric may be more useful than the other depending on the scenario. We further qualitatively analyze these metrics and our findings show that apart from being less interpretable and non-deterministic, GLEU also produces counter-intuitive scores in commonly occurring test examples.

1 Introduction

Grammatical error correction (GEC) refers to the task of automatically detecting and correcting grammatical, spelling, and word choice errors in written text. As newer approaches are being developed for this task, it becomes essential to have reliable means to compare them. In the related field of machine translation (MT), human judgments are used as ground truth to rank systems in the evaluation campaigns as part of the Workshop on Machine Translation (WMT) (Bojar et al., 2016a) held annually. In the absence of annual evaluations and periodic human-judged shared tasks for GEC, robust automatic evaluation metrics are necessary to reliably assess improvements. Automatic evaluation methods can also help in system development and validation.

Previous shared tasks on GEC relied on automatic evaluation metrics to evaluate the performance of participating systems (Dale and Kilgarriff, 2011; Dale et al., 2012; Ng et al., 2013; Ng et al., 2014; Rozovskaya et al., 2015). The CoNLL-2014 shared task test set and evaluation metric, M^2 (Dahlmeier and Ng, 2012), have been used as the primary benchmark for evaluating English GEC. After the CoNLL-2014 shared task, two reference-based evaluation metrics, namely I-measure (Felice and Briscoe, 2015) and GLEU (Napoles et al., 2015; Napoles et al., 2016a), were proposed for GEC. More recently, some reference-less metrics were also proposed (Napoles et al., 2016b; Asano et al., 2017). A standard way to identify the best metric among them is to compare their correlation to human judgments. In the case of MT, WMT organizes a metrics shared task each year, where the correlation of system-level rankings generated by metrics and humans is measured. There is also a segment-level evaluation subtask where the agreement of metrics is measured at the sentence level. Prior work in GEC has followed the system-level evaluation approach to compare metrics (Grundkiewicz et al., 2015; Napoles et al., 2015; Sakaguchi et al., 2016). However, there has been no prior work on sentence-level evaluation. From prior studies,

Proceedings of the 27th International Conference on Computational Linguistics, pages 2730–2741
Santa Fe, New Mexico, USA, August 20-26, 2018.

it may appear that GLEU performs better than the de-facto M^2 metric. However, previous system-level correlation studies turn out to be inadequate, primarily due to the absence of significance tests and the choice of datasets and methods used. In this paper, we re-evaluate reference-based GEC metrics using appropriate significance tests to measure if previous conclusions can be trusted. We find that there is no evidence to suggest that GLEU is significantly better than M^2. Contrary to system-level evaluation, I-measure is found to be a reasonably useful metric at the sentence level and has a positive correlation with human judgments at the sentence level. In our sentence-level evaluation, we find scenarios where M^2 outperforms GLEU and vice versa. Our qualitative assessment of these metrics on example sentences reveals the shortcomings of GLEU in comparison to the other metrics.

2 Evaluation Metrics

We study three popular reference-based evaluation measures that have been proposed for GEC, namely, MaxMatch (M^2), I-measure, and GLEU.

2.1 MaxMatch (M^2)

The M^2 metric (Dahlmeier and Ng, 2012) computes precision, recall, and F-measure by maximally matching phrase-level edits made by a system to gold-standard edits annotated by humans. A gold-standard edit used by M^2 includes the location of the error within the tokenized input sentence and the proposed correction. Although annotations about the type of errors can be included in the gold standard, they are not used while scoring. For computing scores for specific error types, a variant of M^2 has been proposed (Bryant et al., 2017).

The standard M^2 computation is defined as follows. Consider a set of input sentences $S = \{s_1, ..., s_n\}$ and their corresponding system corrected hypotheses $H = \{h_1, ..., h_n\}$. The set of gold-standard edits for each input sentence s_i is denoted by $\mathbf{g}_i$ and the set of edits made by the system to transform s_i to h_i is denoted by $\mathbf{e}_i$. Precision, recall, and F-measure are given by:

$$\text{precision} = \frac{\text{No. of correct edits made by the system}}{\text{Total no. of edits made by the system}} = \frac{\sum_{i=1}^{n} |\mathbf{e}_i \cap \mathbf{g}_i|}{\sum_{i=1}^{n} |\mathbf{e}_i|}$$

$$\text{recall} = \frac{\text{No. of correct edits made by the system}}{\text{Total no. of gold-standard edits}} = \frac{\sum_{i=1}^{n} |\mathbf{e}_i \cap \mathbf{g}_i|}{\sum_{i=1}^{n} |\mathbf{g}_i|}$$

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}.$$

A β value of 0.5 is used in standard M^2 (since CoNLL-2014 shared task), in order to weight precision twice as much as recall. This penalizes incorrect feedback more severely, which is especially important in the context of language learning, where we would like to minimize giving incorrect feedback to language learners. When several sets of annotations (by different annotators) are available for an input sentence s_i, the set that maximizes the F-measure is chosen as $\mathbf{g}_i$.

The set of system edits $\mathbf{e}_i$ for the ith sentence is obtained by first constructing a token-level edit lattice from the Levenshtein distance matrix. Additional edges are added to the lattice to represent phrase-level edits by combining adjacent edges, subject to a constraint on the maximum number of unchanged words (set to 2 in standard M^2). Edges of the lattice are appropriately weighted such that a minimum distance path computation yields the set of edits $\mathbf{e}_i$ that maximally overlaps with the gold-standard edits.

2.2 I-measure

I-measure (Felice and Briscoe, 2015) is a token-level accuracy-based metric proposed to address the inability of M^2 to distinguish between a system that does not correct any errors and a system that only makes wrong changes. Annotations for I-measure can involve non-contiguous tokens as well. The computation of I-measure begins by first aligning the input, hypothesis, and reference corrections in a three-way token-level alignment. Gaps in the alignment can be assumed to be marked by NULL (ϵ) tokens at appropriate positions (Table 1). Let the aligned input, hypothesis, and reference tokens at

POSITIONS	1	2	3	4	5	6	7	8	9	10	11	12	13
INPUT:	Thus	,	advice	from	ϵ	hospital	plays	the	important	role	for	this	.
HYPOTHESIS:	Thus	,	advice	from	ϵ	hospital	plays	an	important	role	for	this	.
REFERENCE:	Thus	,	advice	from	the	hospital	plays	an	important	role	in	this	.

Table 1: An example of three-way token-level alignments produced by I-measure.

position j be denoted by w_j^{inp}, w_j^{hyp}, and w_j^{ref}, respectively. True positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) are defined as follows:

$$\text{TP} : w_j^{\text{inp}} \neq w_j^{\text{ref}} \text{ and } w_j^{\text{hyp}} = w_j^{\text{ref}} \qquad \text{FP} : w_j^{\text{inp}} \neq w_j^{\text{hyp}} \text{ and } w_j^{\text{hyp}} \neq w_j^{\text{ref}}$$
$$\text{TN} : w_j^{\text{inp}} = w_j^{\text{ref}} = w_j^{\text{hyp}} \qquad \text{FN} : w_j^{\text{inp}} \neq w_j^{\text{ref}} \text{ and } w_j^{\text{hyp}} \neq w_j^{\text{ref}}$$

An additional class FPN is defined to balance cases that can be classified as both FP and FN ($\text{FPN} : w_j^{\text{inp}} \neq w_j^{\text{ref}} \neq w_j^{\text{hyp}}$). A weighted accuracy is computed using TP, FP, TN, and FN counts:

$$\text{WAcc} = \frac{\lambda \cdot \text{TP} + \text{TN}}{\lambda \cdot \text{TP} + \text{TN} + \lambda \cdot \left(\text{FP} - \frac{\text{FPN}}{2}\right) + \left(\text{FN} - \frac{\text{FPN}}{2}\right)}$$

The weight λ, which is set to 2, rewards correct changes and penalizes incorrect changes more than preserving erroneous input tokens. When multiple references are used, similar to M^2, the gold-standard reference that maximizes the WAcc is chosen. Then, the weighted accuracy of the input (WAcc_{inp}) is computed by considering the input sentences as the hypothesis, with the same set of references chosen to compute WAcc. I-measure is given by,

$$\text{I} = \begin{cases} \lfloor \text{WAcc} \rfloor, & \text{if WAcc} = \text{WAcc}_{\text{inp}} \\ \dfrac{\text{WAcc} - \text{WAcc}_{\text{inp}}}{1 - \text{WAcc}_{\text{inp}}}, & \text{if WAcc} > \text{WAcc}_{\text{inp}} \\ \dfrac{\text{WAcc}}{\text{WAcc}_{\text{inp}}} - 1, & \text{otherwise.} \end{cases}$$

I-measure denotes the relative improvement or degradation with respect to the input text. I-measure falls in the range[1] $[-1, 1]$, a negative value indicates degradation and a positive value indicates improvement. Unlike M^2, I measure can mix and match annotations from different annotators to produce more alternative references[2].

2.3 GLEU

Unlike M^2 and I-measure, GLEU (Napoles et al., 2015; Napoles et al., 2016a) only requires human annotators to correct by re-writing the source sentence without requiring annotations for individual errors. GLEU computes the precision of n-grams in the hypothesis that match part of the reference sentence, similar to the MT metric BLEU (Papineni et al., 2002). Additionally, GLEU penalizes n-grams in the hypotheses that match part of the input but not the reference. The original formulation (Napoles et al., 2015) included a weight parameter that had to be re-tuned according to the number of reference corrections used. Following the recommendation of Napoles et al. (2016a), we use the new formulation of GLEU[3] which does not include this weight parameter and can work for any number of references.

The computation of GLEU is done as follows. Consider a set of input sentences $S = \{s_1, ..., s_n\}$, their corresponding corrected hypotheses $H = \{h_1, ..., h_n\}$, and reference sentences, $R = \{r_1, ..., r_n\}$.

[1] I-measure may also be represented as a percentage (%) with a range $[-100, 100]$ as used in the rest of the paper.

[2] To use this ability of mixing annotations, alternative corrections for the same underlying error must be grouped together during annotation. However, since the data we use was not annotated in this manner, we disable the mixing ability of I-measure by using `-nomix` to prevent generating invalid references.

[3] This is referred to as GLEU+ in (Napoles et al., 2016a).

Here, we assume that there is a single reference sentence for each input sentence. First, a precision term p_k is computed for n-grams of size k ($k = 1, 2, ..., N$ and $N = 4$ for standard GLEU):

$$p_k = \frac{\sum_{h_i \in H} \left(\sum_{\substack{\text{k-gram in} \\ h_i \text{ and } r_i}} \text{count}_{h_i,r_i}(\text{k-gram}) - \sum_{\substack{\text{k-gram in} \\ h_i \text{ and } s_i}} \max\left[0, \text{count}_{h_i,s_i}(\text{k-gram}) - \text{count}_{h_i,r_i}(\text{k-gram})\right] \right)}{\sum_{h_i \in H} \sum_{\substack{\text{k-gram} \\ \text{in } h_i}} \text{count}_{h_i}(\text{k-gram})}$$

$\text{count}_a(\text{n-gram}) = \#$ occurrences of n-gram in a

$\text{count}_{a,b}(\text{n-gram}) = \min(\#\text{occurrences of n-gram in } a, \#\text{occurrences of n-gram in } b)$

Similar to BLEU, a brevity penalty (BP) is computed to penalize short hypotheses:

$$\text{BP} = \begin{cases} 1, & \text{if } l_h > l_r \\ \exp(1 - l_r/l_h), & \text{if } l_h \leq l_r \end{cases}$$

where l_r is the reference corpus length (sum of the number of tokens of the reference sentences) and l_h is the total hypothesis corpus length (sum of the number of tokens of all hypotheses). GLEU is given by:

$$\text{GLEU}(\text{S}, \text{H}, \text{R}) = \text{BP} \cdot \exp\left(\frac{1}{N} \sum_{k=1}^{N} \log p_k\right)$$

When multiple references are available, GLEU does not include reference n-grams from all references to compute n-gram precision like BLEU, nor picks the best like M^2 and I-measure. Instead, for each input sentence, a random reference correction is chosen from the set of reference corrections to compute the GLEU score. This makes GLEU non-deterministic, unlike M^2 and I-measure. The average GLEU score over m GLEU score computations is reported as the final score ($m = 500$ in standard GLEU).

3 Quantitative Evaluation

The three GEC metrics are evaluated by measuring their correlation with human quality judgments treated as the ground truth. Following the experimental methodology in WMT metrics shared tasks (Macháček and Bojar, 2014; Stanojević et al., 2015; Bojar et al., 2016b; Bojar et al., 2017), we evaluate system-level correlation as well as sentence-level agreement of metrics with human judgments.

We utilize the collection of human judgments of GEC outputs released in (Grundkiewicz et al., 2015). The dataset is annotated similar to the relative-ranking method adopted in the WMT metrics shared tasks in (Macháček and Bojar, 2014; Stanojević et al., 2015). Human judgments are obtained for the system outputs of the 12 participating systems from the CoNLL-2014 shared task (Ng et al., 2014) and the input text as the thirteenth system (referred to as INPUT). Each human judgment consists of a 5-way ranking of hypotheses corrections from randomly chosen systems for an input sentence. Systems that produce the same hypothesis for an input sentence are grouped together. The 5-way rankings are then converted to pairwise rankings that include ties. A total of 109,098 pairwise rankings can be obtained if systems with the same hypothesis that were grouped together in the 5-way rankings are included (henceforth, referred to as *expanded set*). Grundkiewicz et al. (2015) observed that there were a large number of ties due to the high overlap in system outputs. Trivial ties of systems that produce the same hypothesis can be removed by including only one randomly chosen system among grouped systems. This results in 20,516 pairwise rankings (henceforth, referred to as *unexpanded set*).

3.1 Measuring System-Level Correlation

System-level correlation is computed by comparing the ranking of the participating systems by humans and the ranking generated by the metric scores. Generating a ranking using metric scores is straightforward. However, human pairwise comparisons need to be converted into a ranking of systems. To do this,

two methods are employed in (Grundkiewicz et al., 2015)[4], namely *Expected Wins* (Bojar et al., 2013) and *Trueskill* (Sakaguchi et al., 2014).

The Expected Wins model computes an intuitive score for a system based on the probability that a system wins (ranks higher according to human judgments) over a randomly chosen system on a randomly chosen input sentence. This model ignores ties. The Trueskill model computes a score representing the average relative ability of the system compared to other systems, accounting for the uncertainty surrounding the system's ability as well. In the context of GEC, on this dataset, Grundkiewicz et al. (2015) empirically found that Expected Wins performs better than Trueskill. However, almost all subsequent work ignored this finding and used the Trueskill model instead. We compute correlations using both methods in order to ensure comparability to prior work.

We measure both Pearson and Spearman correlation coefficients between metric rankings and human rankings. Pearson correlation (r) compares the system-level scores produced by a metric against human, under the assumption that the two measured variables have a linear relationship:

$$r = \frac{\sum_{i=1}^{q}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{q}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{q}(y_i - \bar{y})^2}}$$

where $x_1, ..., x_q$ are metric scores for q systems for a particular metric with $\bar{x}$ as the mean metric score, and $y_1, ..., y_q$ are human scores computed by Trueskill or Expected Wins with $\bar{y}$ as the mean human score. On the other hand, Spearman correlation (ρ) computes a correlation coefficient based on ranks instead of scores:

$$\rho = 1 - \frac{6\sum_{i=1}^{q}(d_i)^2}{q(q^2-1)}$$

where d_i represents the difference in the human rank and metric rank of the ith system. Spearman correlation is more relaxed compared to Pearson correlation in terms of the assumptions made about variables and also less sensitive to outliers in the sample.

In order to determine if a metric outperforms another, it is inadequate to measure differences in correlation alone. Following the recommendations in (Graham and Baldwin, 2014), we evaluate for significance of differences of correlation between metrics using William's test (Williams, 1959). Note that prior work in human evaluation of GEC systems has not reported significance tests that account for the dependence between two metrics and hence the derived conclusions are not justified.

3.2 Measuring Sentence-Level Agreement

Since system-level evaluation is done on a few systems (13 in our case), we also compute a fine-grained sentence-level agreement of metrics to human pairwise rankings. For this, sentence-level metric scores are computed[5]. Agreement to humans is computed in a similar manner as segment-level evaluation in the WMT metrics shared task and variants of the Kendall's Tau (τ) are used:

$$\tau = \frac{|\text{Concordant}| - |\text{Discordant}|}{\text{Total \# Pairwise Comparisons}} \quad (1)$$

where |Concordant| refers to the number of times a metric agrees with the sentence-level human pairwise comparisons, and |Discordant| refers to the number of times they disagree. The variants of τ are related to the way in which human ties are handled. In the variant used in the WMT metrics shared task from WMT14 (Macháček and Bojar, 2014) onwards (henceforth referred to as *NoTies*), human ties are ignored completely and metric ties are added to the denominator alone, without contributing to Concordant or Discordant sets. For GEC, since there is a large number of ties, particularly in the expanded set, we include another variant that incorporates human ties as well (henceforth, referred to as *HTies*). In this

[4]We use the scripts released by Grundkiewicz et al. (2015) to compute human ranking of systems.

[5]Similar to BLEU, sentence-level GLEU needs smoothing to avoid zero n-gram counts. We use the sentence-level scores produced by GLEU using the `-d` flag, where smoothing is performed by replacing a zero count with one.

Metric	Expected Wins		Trueskill	
	r	ρ	r	ρ
GLEU	0.691	0.407	0.733	0.478
I-MEASURE	–0.250	–0.385	–0.316	–0.423
M^2	0.623	0.687	0.672	0.720

Table 2: Results of system-level Pearson (r) and Spearman (ρ) correlations of GEC metrics using the Expected Wins model and Trueskill model of human rankings.

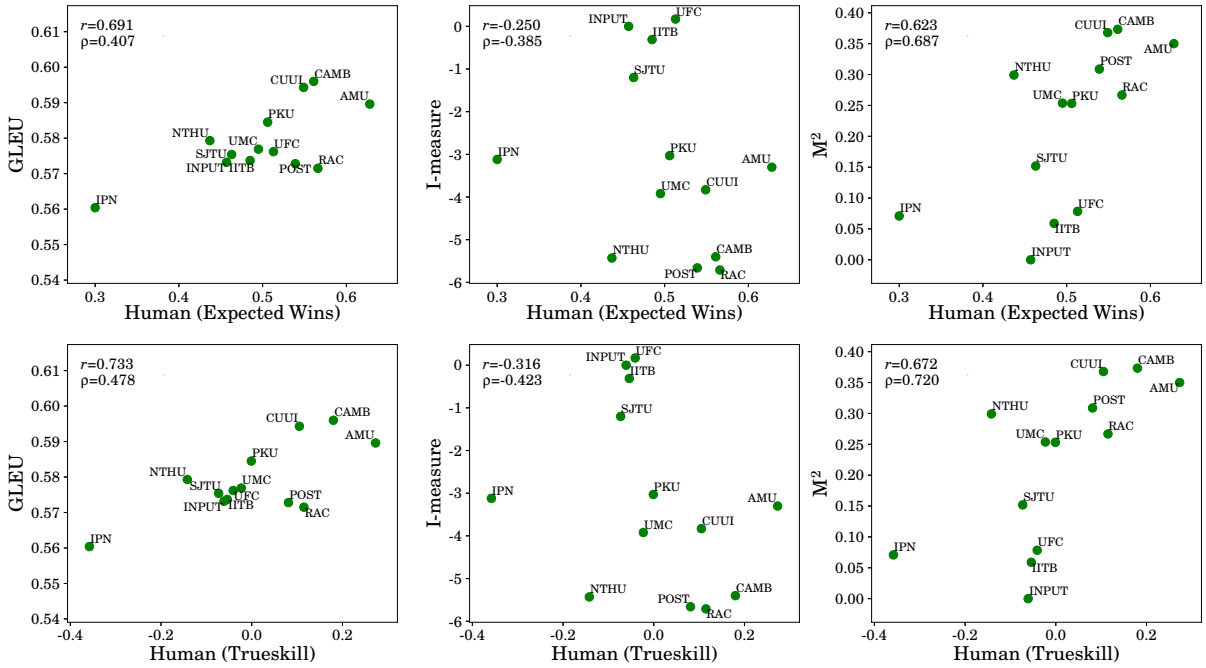


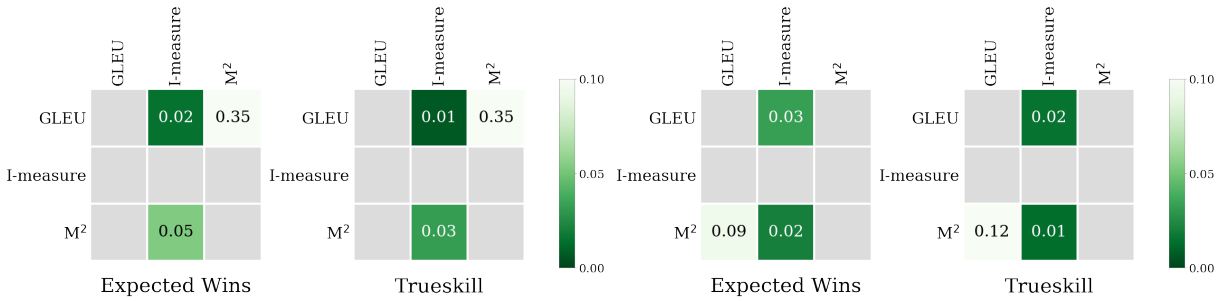
Figure 1: Scatter plots showing the metric scores and human scores of CoNLL-2014 shared task systems.

variant, when a human judges a tie and metric predicts a tie as well for a given pair of hypotheses, it is treated as a concordant pair. However, when a human predicts a tie and a metric does not, or vice versa, it contributes only to the denominator and not to the Discordant set. We test for significance similar to (Bojar et al., 2017), by employing bootstrap resampling over 1,000 samples and those metrics with non-overlapping 95% confidence intervals are treated as having statistically significant improvements compared to the lower performing metrics.

3.3 Results

3.3.1 System-Level Evaluation

The results of the system-level correlation tests are given in Table 2. In both Expected Wins and Trueskill methods of human ranking, GLEU achieves a higher Pearson correlation compared to the rest. M^2 achieves moderately high Pearson correlations and the highest Spearman correlations using both methods of human ranking. As noted in prior studies (Grundkiewicz et al., 2015; Napoles et al., 2016a), I-measure achieves a negative correlation. In order to decide which correlation measure is more suitable and to understand the cause of disagreement in both correlation measures, it may be worth looking at the bivariate scatter plot of metric and human scores (Figure 1). The presence of an outlier (the leftmost point – IPN) indicates that Pearson correlation may not be the ideal choice as it is sensitive to outliers in the data. To demonstrate this, we remove the system IPN from the ranking. Pearson correlation of GLEU then becomes lower than that of M^2 (0.500 for GLEU vs 0.593 for M^2 using Expected Wins and



(a) p-values for Pearson correlation.

(b) p-values for Spearman correlation

Figure 2: Results of William's significance tests for correlations between metrics. A green non-empty cell denotes that the row metric has a higher correlation coefficient compared to the column metric with the specified p-value. A gray empty cell indicates that the row metric has equal or lower correlation coefficient compared to the column metric.

0.584 for GLEU vs 0.638 for M^2 using Trueskill). The relative performance under Spearman correlation, however, remains the same (0.245 for GLEU vs 0.643 for M^2 using Expected Wins and 0.336 for GLEU vs 0.692 for M^2 using Trueskill). The high Spearman correlation for M^2 indicates that it is a good metric for ranking systems, which is the primary goal of automatic evaluation metrics.

To understand if one metric significantly outperforms another, we perform William's significance tests (Figure 2). As expected, we find that GLEU is not significantly better than M^2 in terms of Pearson correlation (p-value of 0.35 for both Expected Wins and Trueskill). In the case of Spearman correlation for Expected Wins, M^2 is significantly better than GLEU at the level of significance 0.10, but does not significantly outperform GLEU for the Trueskill model. The significance test results show that neither GLEU nor M^2 can be conclusively shown to be better than the other.

3.3.2 Sentence-Level Evaluation

Table 3 shows the results of sentence-level agreement of metrics to human pairwise rankings when human ties are considered (HTies) or ignored (NoTies). We also consider the unexpanded set where systems that produce the same hypothesis for an input sentence are grouped together. This removes the trivial ties (two identical hypotheses judged to be a tie) from the pairwise rankings. In the HTies variant, M^2 achieves statistically significant improvements compared to the other two metrics in both expanded and unexpanded sets. However, for NoTies variant, GLEU achieves the best result. The results show that M^2 may be better at judging ties and performs well overall when complete pairwise rankings are considered. On the other hand, GLEU does a better job at discriminating hypotheses, likely due to its ability to judge fluency like humans do (Sakaguchi et al., 2016). This difference in behavior of GLEU and M^2 between the two variants, HTies and NoTies, is expected given a large number of ties for GEC (Grundkiewicz et al., 2015) as indicated by the difference in the number of pairwise comparisons (Table 3) and that M^2 generally assigns more ties compared to GLEU. Surprisingly, I-measure achieves a positive correlation at the sentence level as opposed to a negative correlation at the system level, suggesting that it can be a useful metric at the sentence level.

4 Qualitative Assessment

We illustrate the strengths and weaknesses of the three metrics with the help of two examples (Table 4). We compare the metrics based on two criteria, interpretability and intuitiveness, which we believe are necessary for good GEC metrics.

Interpretability: The scores produced by a good GEC metric should be interpretable as a measure of a system's ability to correct errors and improve the text. M^2 and I-measure are based on minimal annotations and their results are interpretable. For example, M^2 measures the precision and recall of a system, in terms of the number of phrase-level errors that it corrects in a given input text. Similarly, I-measure is computed based on the relative weighted accuracy between a system and a do-nothing

Metric	Expanded		Unexpanded	
	HTies (109098)	NoTies (49981)	HTies (20516)	NoTies (14584)
GLEU	0.567	0.388*	0.237	0.321
I-MEASURE	0.564	0.368	0.242	0.293
M^2	0.617*	0.300	0.348*	0.266

Table 3: Results of sentence-level agreement in terms of two variations of Kendall's τ (HTies and NoTies) on expanded and unexpanded sets. The number of human pairwise comparisons used is given in parentheses. * indicates that the 95% confidence interval of the metric does not overlap with those of the other metrics.

		(GLEU, I%, M^2)
	Example 1	
INP:	The weekly quizzes in this course makes it challenging and fun .	
HYP1:	The weekly quizzes in this course makes it challenging and fun .	(0.392, 0.00, 0.000)
HYP2:	The weekly quizzes in this course ***making*** it challenging and fun .	(0.735, –4.00, 0.000)
REF:	The weekly quizzes in this course ***make*** it challenging and fun .	
	Example 2	
INP:	The senior student who failed have to retake the course next year .	
HYP1:	The senior student who failed ***has*** to retake the course next year .	(0.661, 100.00, 1.000)
HYP2:	The senior ***students*** who failed have to retake the course next year .	(0.656, 100.00, 1.000)
HYP3:	The senior ***students*** who failed ***has*** to retake the course next year .	(0.776, –6.11, 0.556)
REF1:	The senior student who failed ***has*** to retake the course next year .	
REF2:	The senior ***students*** who failed have to retake the course next year .	

Table 4: Illustrating examples and scores produced by the metrics.

baseline. In contrast, GLEU measures the precision of n-grams of the output text similar to BLEU. The scores that it produces have no clear relation to the system's ability to correct or improve a text. For example, in Example 1, Hypothesis 1 (Table 4), when the system hypothesis is exactly the input sentence itself, there is no evidence about the system's ability to correct errors. While I and M^2 give a zero score, GLEU gives a non-zero score (GLEU = 0.392). In the case of its counterpart BLEU, if some n-grams of the hypothesis and reference match, it rightly assigns a non-zero score as the system shows some ability to perform translation. However, for GEC, simply copying the input can result in matching several n-grams from the reference despite the system showing zero ability to perform correction. Moreover, the GLEU score will vary with the length of the input sentence without the system having to fix any error or even modify the input. Similarly, for Example 2, Hypothesis 1 and 2 (Table 4), despite being grammatical and matching one of the reference corrections exactly, GLEU gives a non-perfect score for both hypotheses, whereas I and M^2 correctly gives perfect scores. This is an artefact of GLEU randomly selecting one among the two references for scoring and averaging over multiple iterations. Also, due to its non-deterministic behavior, the scores for Hypothesis 1 and 2 differ despite having the same n-gram statistics compared to the references.

Intuitiveness: In Example 1, GLEU produces a non-zero score for a hypothesis that is exactly the same as the input sentence (Hypothesis 1). When an incorrect change is introduced (Hypothesis 2), the score becomes even higher (GLEU=0.735). This is counter-intuitive. If a change results in an ungrammatical sentence, the score should remain the same or decrease as in the case of M^2 and I. This unintuitive behavior of GLEU is due to the additional term in GLEU that penalizes preservation of n-grams in the

input sentence that were supposed to be changed according to the reference. It can be argued that GLEU intends to reward GEC systems for detecting errors by assigning a partial credit to systems that make spurious changes at locations where corrections are deemed necessary by human annotators. However, this will encourage building GEC systems that provide inaccurate feedback and potentially mislead the end users (primarily language learners). Hence, it is better to build and evaluate grammatical error detection systems separately (Rei and Yannakoudakis, 2016). I-measure, on the other hand, assigns a negative score for Hypothesis 2 as it is considered to 'degrade' the input, although it is arguable that both hypotheses 1 and 2 are equally ungrammatical. In Example 2, when multiple references are used, GLEU gives a higher score to Hypothesis 3 (ungrammatical) than Hypothesis 1 and 2, both of which are grammatical and each matches one of the two references exactly. On the other hand, both M^2 and I-measure assign a lower score to Hypothesis 3 and conform to our intuition.

5 Related Work

5.1 GEC Evaluation

When GEC was restricted to specific error types, measures such as accuracy, precision, recall, and F-score were employed (Chodorow et al., 2012) as done in the earlier shared tasks (Dale and Kilgarriff, 2011; Dale et al., 2012). Dahlmeier and Ng (2012) identified weaknesses in evaluation methods used in these shared tasks in extracting phrase-level edits that correctly match the reference and proposed MaxMatch scoring (Dahlmeier and Ng, 2012), which can also work for sentence correction over all error types. M^2 does not distinguish between a do-nothing baseline and a system that changes the input incorrectly, and hence I-measure (Felice and Briscoe, 2015) was proposed. Later, GLEU (Napoles et al., 2015) was proposed. It relied on whole sentence rewrites instead of span-based error annotations. However, our analysis shows that GLEU has several weaknesses in its formulation and has a non-deterministic behavior that makes it an unreliable alternative to prior metrics. A few reference-less evaluation methods have been proposed of which (Napoles et al., 2016b) considers the grammaticality of the output sentences alone to evaluate GEC systems. Recently, Asano et al. (2017) proposed improvements to (Napoles et al., 2016b) by additionally accounting for fluency and meaning preservation. Sakaguchi et al. (2017) suggested future directions of improving GEC evaluation such as considering whole document rewrites.

5.2 Human Judgments and Metric Quality

Inspired from WMT, two collections of human judgments of GEC system outputs were released (Grundkiewicz et al., 2015; Napoles et al., 2015). We use the former in our study as it has a much higher number of human pairwise comparisons (109,098) compared to the latter (28,146). Also, (Napoles et al., 2015) includes references as one of the compared systems in order to act as a control measure to ensure quality of human judgments. However, this is unfair to systems that compare more often against the reference as noted by Callison-Burch et al. (2012), since human raters may prefer the reference corrections more often. This bias is further worsened by explicitly displaying the reference corrections to the human judges in (Napoles et al., 2015). Reference translations are neither included in rankings nor shown to judges in (Grundkiewicz et al., 2015). Also, all subsequent studies used the judgments released by Grundkiewicz et al. (2015) for comparing GEC metrics. Grundkiewicz et al. (2015), however, did not compare to GLEU as it was not available at the time. Later work which measured system-level correlation using GLEU failed for a number of reasons which motivate this paper. Apart from using different variations of GLEU, there was a mistake[6] in GLEU computation that produced different results and incorrect conclusions in earlier studies. Moreover, no proper significance tests to compare the differences in correlations between metrics were done. Significance tests that reject the null hypothesis of having no correlation to humans is not adequate for comparing differences in correlations between metrics (Graham and Baldwin, 2014). Also, (Napoles et al., 2016b; Asano et al., 2017) use 18 references for CoNLL-2014 sentences for generating metric rankings, of which 2 are the references used in the shared task, 8 are from (Bryant and Ng, 2015), and another 8 are annotated by both experts and non-experts as sentence-rewrites (Sakaguchi

[6]The brevity penalty was incorrectly implemented as $\exp(1-l_h/l_r)$ instead of $\exp(1-l_r/l_h)$. This was fixed in the version that we use (fixed on 10 June, 2017: https://github.com/cnap/gec-ranking/commit/50b503)

et al., 2016). Automatically constructing M^2 and I-measure annotations from sentence rewrites will be sub-optimal. Also, we observed that the non-determinism of GLEU scores is more pronounced when more number of references are used, resulting in large differences in correlation values. In the end, we decided to use the standard two references for CoNLL-2014 as used in the original shared task and continues to be the standard benchmark used to evaluate GEC systems to date. Also, prior studies had not conducted sentence-level agreement with human judgments, which we did in this paper. Sakaguchi et al. (2017) qualitatively compared metrics using contrived, gamed examples, which are rather irrelevant in practice (such as a system that outputs a dummy hypothesis "a", "a a", or "a a a" for any input sentence) and showed that M^2 under-penalized such systems. On the contrary, our qualitative assessment is based on naturally occurring examples and highlights the practical strengths and weaknesses of current GEC metrics.

6 Conclusion

Through carefully designed experiments and significance tests, we find no evidence of GLEU being a better metric than M^2 for ranking systems as claimed in prior work (Napoles et al., 2015; Sakaguchi et al., 2016; Napoles et al., 2016b). In fact, at the sentence level, when correctly predicting human ties are rewarded, M^2 works better than GLEU. We believe that GEC metrics should be interpretable and must provide intuitive scores. In our qualitative assessment, we find that GLEU is less interpretable and produces absurd scores in common scenarios. Our analysis suggests that GLEU cannot be considered as a replacement of existing GEC metrics. The code to replicate the evaluations in this paper is available at `https://github.com/nusnlp/gecmetrics`.

Acknowledgements

We thank the three anonymous reviewers for their useful feedback.

References

Hiroki Asano, Tomoya Mizumoto, and Kentaro Inui. 2017. Reference-based metrics can be replaced with reference-less metrics in evaluating grammatical error correction systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*.

Ondřej Bojar, Christian Federmann, Barry Haddow, Philipp Koehn, Matt Post, and Lucia Specia. 2016a. Ten years of WMT evaluation campaigns: Lessons learnt. In *Proceedings of LREC Workshop Translation Evaluation: From Fragmented Tools and Data Sets to an Integrated Ecosystem*.

Ondřej Bojar, Yvette Graham, Amir Kamran, and Miloš Stanojević. 2016b. Results of the WMT16 metrics shared task. In *Proceedings of the First Conference on Machine Translation*.

Ondřej Bojar, Yvette Graham, and Amir Kamran. 2017. Results of the WMT17 metrics shared task. In *Proceedings of the Second Conference on Machine Translation*.

Christopher Bryant and Hwee Tou Ng. 2015. How far are we from fully automatic high quality grammatical error correction? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 Workshop on Statistical Machine Translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*.

Martin Chodorow, Markus Dickinson, Ross Israel, and Joel Tetreault. 2012. Problems in evaluating grammatical error detection systems. In *Proceedings of the 24th International Conference on Computational Linguistics*.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Robert Dale and Adam Kilgarriff. 2011. Helping our own: The HOO 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. HOO 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*.

Mariano Felice and Ted Briscoe. 2015. Towards a standard evaluation method for grammatical error detection and correction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Yvette Graham and Timothy Baldwin. 2014. Testing for significance of increased correlation with human judgment. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Edward Gillian. 2015. Human evaluation of grammatical error correction systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Matouš Macháček and Ondřej Bojar. 2014. Results of the WMT14 metrics shared task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2016a. GLEU without tuning. *arXiv preprint arXiv:1605.02592*.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2016b. There's no comparison: Reference-less evaluation metrics in grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*.

Marek Rei and Helen Yannakoudakis. 2016. Compositional sequence labeling models for error detection in learner writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Alla Rozovskaya, Houda Bouamor, Nizar Habash, Wajdi Zaghouani, Ossama Obeid, and Behrang Mohit. 2015. The second QALB shared task on automatic text correction for Arabic. In *Proceedings of the Second Workshop on Arabic Natural Language Processing*.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2014. Efficient elicitation of annotations for human evaluation of machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Keisuke Sakaguchi, Courtney Napoles, Matt Post, and Joel Tetreault. 2016. Reassessing the goals of grammatical error correction: Fluency instead of grammaticality. *Transactions of the Association for Computational Linguistics*, 4:169–182.

Keisuke Sakaguchi, Courtney Napoles, and Joel Tetreault. 2017. GEC into the future: Where are we going and how do we get there? In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*.

Miloš Stanojević, Amir Kamran, Philipp Koehn, and Ondřej Bojar. 2015. Results of the WMT15 metrics shared task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*.

Evan James Williams. 1959. *Regression Analysis*. Wiley.

Information Aggregation via Dynamic Routing for Sequence Encoding

Jingjing Gong, Xipeng Qiu*, Shaojing Wang, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
{jjgong15, xpqiu, sjwang17, xjhuang}@fudan.edu.cn

Abstract

While much progress has been made in how to encode a text sequence into a sequence of vectors, less attention has been paid to how to aggregate these preceding vectors (outputs of RNN/CNN) into fixed-size encoding vector. Usually, a simple max or average pooling is used, which is a bottom-up and passive way of aggregation and lack of guidance by task information. In this paper, we propose an aggregation mechanism to obtain a fixed-size encoding with a dynamic routing policy. The dynamic routing policy is dynamically deciding that *what* and *how much* information need be transferred from each word to the final encoding of the text sequence. Following the work of Capsule Network, we design two dynamic routing policies to aggregate the outputs of RNN/CNN encoding layer into a final encoding vector. Compared to the other aggregation methods, dynamic routing can refine the messages according to the state of final encoding vector. Experimental results on five text classification tasks show that our method outperforms other aggregating models by a significant margin. Related source code is released on our github page[1].

1 Introduction

Learning the distributed representation of text sequences, such as sentences or documents, is crucial to a wide range of important natural language processing applications. A primary challenge is how to encode the variable-length text sequence into a fixed-size vector, which should fully capture the semantics of text.

Many successful text encoding methods usually contain three key steps: (1) converting each word in a text sequence into its embedding; (2) taking as input the sequence of word embeddings, and computing the context-aware representation for each word with a recurrent neural network (RNN) (Hochreiter and Schmidhuber, 1997; Chung et al., 2014) or convolutional neural network (CNN) (Collobert et al., 2011; Kim, 2014); (3) summarizing the sentence meaning into a fixed-size vector by an aggregation operation. Then, these models are trained by combining a downstream task in a supervised or unsupervised way.

Currently, much attention is paid to the first two steps, while the aggregation step is less emphasized on. Some simple aggregation methods, such as max (or average) pooling, is used to sum the RNN hidden states or convolved vectors, computed in the previous step, into a single vector. This kind of methods aggregate information in a bottom-up and passive way and are lack of the guide of task information. Recently, several works employ self-attention mechanism (Lin et al., 2017; Yang et al., 2016) on top of the recurrent or convolutional encoding layer to replace simple pooling. A basic assumption is that the words (or even sentences) are not equally important. One or several task-specific context vectors are used to assign a different weight to each word and select task-specific encodings. The context vectors are parameters learned jointly with other parameters during the training process. These attentive aggregation can select task-dependent information. However, the context vectors are fixed once learned.

* Corresponding Author

[1] https://github.com/FudanNLP/Capsule4TextClassification

Proceedings of the 27th International Conference on Computational Linguistics, pages 2742–2752
Santa Fe, New Mexico, USA, August 20-26, 2018.

In this paper, we regard the aggregation as a routing problem of how to deliver the messages from source nodes to target nodes. In our setting, the source nodes are the outputs of a recurrent or convolutional encoding layer, and the target nodes are one or several fixed-size encoding vectors to represent the meaning of the text sequence.

From this viewpoint, both the pooling and attentive aggregations are a *fixed* routing policy without considering the state of the final encoding vectors. For example, the final encoding vectors could receive some redundancy information from different words. The fixed routing policy cannot avoid this issue. Therefore, we wish for a new way to aggregate information according to the state of the final encoding.

In recent promising work of capsule network (Sabour et al., 2017), a dynamic routing policy is proposed and proven to be more effective than the max-pooling routing. Inspired by their idea, we introduce a text sequence encoding model with dynamic routing mechanism. Specifically, we propose two kinds of dynamic routing policies. One is the standard dynamic routing policy same as the capsule network, in which the source node decides what and how many messages are sent to different target nodes. The other is the reversed dynamic routing policy, in which the target node decides what and how many messages may be received from different source nodes.

Experimental results on five text classification tasks show that the dynamic routing policy outperforms other aggregation methods, such as max pooling, average pooling, and self-attention by a significant margin.

2 Background: general sequence encoding for text classification

In this section, we are going to introduce a general text classification framework. It consists of an Embedding Layer, Encoding Layer, Aggregation Layer and Prediction Layer.

2.1 Embedding Layer

Given a text sequence with words $S = w_1, w_2, \cdots, w_L$. Since the words are symbols that could not be processed directly using prominent neural architectures, so we first map each word into a d dimensional embedding vector,

$$X = [\mathbf{x}_1, \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_L]. \tag{1}$$

In order to transfer knowledge from a vast unlabeled corpus, the embeddings can be taken from the pre-trained word embedding, such as Glove (Pennington et al., 2014).

2.2 Encoding Layer

However, each word representation in X is still independent with each other. To gain some dependency between adjacent words, we then build a bi-directional LSTM (BiLSTM) layer (Hochreiter and Schmidhuber, 1997) to incorporate forward and backward context information of a sequence. Then we can get phrase-level encoding $\mathbf{h}_t$ of a word by concatenating forward $\mathbf{h}_t^f$ and backward output vector $\mathbf{h}_t^b$ correspond to the target word.

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

$$\mathbf{h}_t^b = \text{LSTM}(\mathbf{h}_{t+1}^b, \mathbf{x}_t), \tag{3}$$

$$\mathbf{h}_t = [\mathbf{h}_t^f; \mathbf{h}_t^b]. \tag{4}$$

Thus, the outputs of BiLSTM encoder are a sequence of vectors

$$H = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_L]. \tag{5}$$

2.3 Aggregation Layer

Encoding layer only models dependency between adjacent words, but the final prediction of the text requires a fix-length vector. Therefore we need aggregate information from variable length sequence to a single fix-length vector. There are several different ways of aggregation such as max or average pooling, and context-attention.

Max or Average Pooling Max or Average pooling is a simple way of aggregating information, which does not require extra parameters and is computationally efficient (Kim, 2014; Zhao et al., 2015; Lin et al., 2017). In the process of modeling natural language, max or average pooling is performed along the time dimension.

$$\mathbf{e}^{max} = \max([\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_L]), \tag{6}$$

$$\mathbf{e}^{avg} = \frac{1}{L}\sum_{i=1}^{L}\mathbf{h}_i, \tag{7}$$

For example,in Equation 6 the $\max$ operation is performed on each dimension of $\mathbf{h}$ along time dimension. And in Equation 7 the average operation is performed along time dimension.

Max pooling is empirically better at aggregating long sentences than average pooling. We assume it's because that, the actual word that contributes to the classification problem is far less than the number of words that contain in a long sentence. Information from important words is weakened by a large population of "boring" words.

Self-Attention As has been stated previously, average pooling is prone to weaken important words when the sentence is longer. Self-Attention assigns each word a weight to indicate the importance of a word depending on the task on hand. A few words that are crucial to the task will be emphasized while the "boring" words are ignored. The self-attention process is formulated as follows:

$$u_i = \mathbf{q}^T\mathbf{h}_i, \tag{8}$$

$$a_i = \frac{\exp(u_i)}{\sum_k \exp(u_k)}, \tag{9}$$

$$\mathbf{e}^{attn} = \sum_{i=1}^{L} a_i \cdot \mathbf{h}_i \tag{10}$$

First, we need a task-specific trainable query $\mathbf{q} \in \mathbb{R}^d$ to calculate similarity weight between query and each contextually encoded word. Then the corresponding weights are normalized across time dimension using softmax normalization function Eq. 9, after that the aggregated vector is simply a weighted sum of the input sequence in Eq. 10.

2.4 Prediction Layer

Then we feed the encoding $\mathbf{e}$ to the input of a multi-layer perceptron (MLP), followed by a softmax classifier.

$$\mathbf{p}(\cdot|\mathbf{e}) = \mathrm{softmax}(\mathrm{MLP}(\mathbf{e}))$$

where $\mathbf{p}(\cdot|\mathbf{e})$ is the predicted distribution of different classes given the representation vector $\mathbf{m}$.

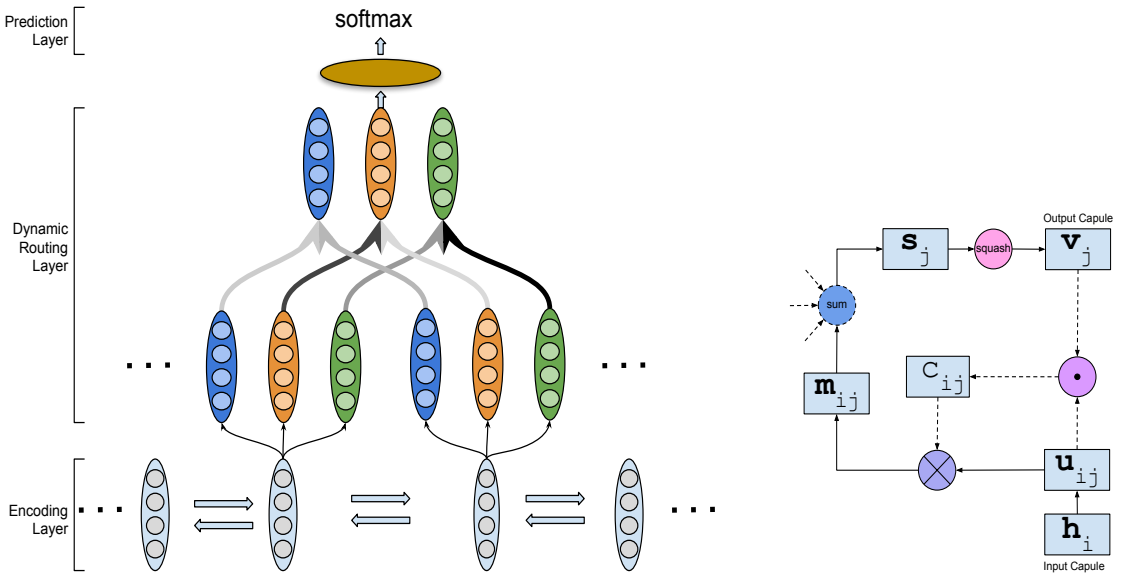
3 Aggregation via Dynamic Routing

In this section, we will formally introduce dynamic routing in detail. The goal of dynamic routing is to encode the meaning of X into M fix-length vectors

$$V = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_M]. \tag{11}$$

To transfer information from a variable number of representation H to a fixed number of vectors V, a key problem we need to slove is to properly design a routing policy of information transfer. In other words, *what* and *how much* information is to be transferred from $\mathbf{h}_i$ to $\mathbf{v}_j$.

Although self-attention has been applied in aggregation, the notion of summing up elements in the attention mechanism is still very primitive. Inspired by the capsule networks (Sabour et al., 2017), we propose a dynamic routing aggregation (DR-AGG) mechanism to compute the final encoding of text sequence.



(a) Aggregation via Dynamic Routing

(b) Detialed dynamic routing process

Figure 1: Diagram of dynamic routing for sequence encoding. (a) is the overall dynamic routing diagram, the width of edges between capsules indicate the amount of information transfered, which is refined iteratively. (b) is a detailed iterative process of transferring information from capsule $\mathbf{h}_i$ to capsule $\mathbf{v}_j$, where $\odot$ is a inner product operation and $\otimes$ is a element-wise product.

Following the definition of capsule networks, we call each encoding vector, or a group of neurons, as a capsule. Thus, H denotes the input capsules, and V denotes the output capsules.

A message vector $\mathbf{m}_{i\to j}$ denotes the information to be transferred from $\mathbf{h}_i$ to $\mathbf{v}_j$.

$$\mathbf{m}_{i\to j} = c_{ij} f(\mathbf{h}_i, \theta_j), \tag{12}$$

where c_{ij} indicates proportionally how much information is to be transferred, and $f(\mathbf{h}_i, \theta_j)$ is a one-layer fully-connected network parameterized by θ_j, indicating which aspect of information is to be transferred.

The output capsule $\mathbf{v}_j$ first aggregates all the incoming messages

$$\mathbf{s}_j = \sum_{i=1}^{L} \mathbf{m}_{i\to j}, \tag{13}$$

and then squashes $\mathbf{s}_j$ to confine $|\mathbf{s}_j| \in (0, 1)$ to a probability,

$$\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1 + \|\mathbf{s}_j\|^2} \frac{\mathbf{s}_j}{\|\mathbf{s}_j\|} \tag{14}$$

Dynamic Routing Process The dynamic routing process is implemented by an iterative process of refining the coupling coefficient c_{ij}, which define proportionally how much information is to be transferred from $\mathbf{h}_i$ to $\mathbf{v}_j$.

The coupling coefficient c_{ij} is computed by

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})}, \tag{15}$$

$$b_{ij} \leftarrow b_{ij} + \mathbf{v}_j^{\mathrm{T}} f(\mathbf{h}_i, \theta_j), \tag{16}$$

where b_{ij} is the log probabilities, initialized with 0.

The coefficients c_{ij} is computed using a softmax function, and $\sum_{j=1}^{M} c_{ij} = 1$. Therefore, the total amount of information transferred from capsule h_i is proportionally summed to one.

When an output capsule $\mathbf{v}_j$ receives the incoming messages, its state will be updated and the coefficient c_{ij} is also re-computed for each input capsule. Thus, we iteratively refine the route of information passing, towards an instance dependent and context aware encoding of a sequence. After the text sequence is encoded into M capsules, We map these capsules into vector representation by simply concatenating all capsules,

$$\mathbf{e} = [\mathbf{v}_1; \ldots; \mathbf{v}_M]. \tag{17}$$

Figure 1 gives an illustration of the dynamic routing mechanism. The detailed dynamic routing algorithm is further described in detail in Algorithm 1.

Algorithm 1: Dynamic Routing Algorithm

Data: Input Capsules: $\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_L$, Maximum number of Iterations: T
Result: Output Capsules: $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_M$
Initialize $b_{ij} \leftarrow 0$;
for $t = 1$ **to** T **do**
  Compute the routing coefficients c_{ij} for all $i \in [1, L], j \in [1, M]$; /* Eq.15 */
  Update all the output capsule $\mathbf{v}_j, j \in [1, M]$; /* Eq. 13 and 14 */
  Update b_{ij} for all $i \in [1, L], j \in [1, M]$; /* Eq. 16 */
end for
return $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_M$

Reversed Dynamic Routing Process In standard DR-AGG, an input capsule decides what proportion of information can be transferred to an output capsule. We also explore a reversed dynamic routing, in which the output capsule decides what proportion of information should be received from an input capsule. The only difference between reversed dynamic routing and standard dynamic routing is how the softmax function was applied to the log probabilities $[b_{ij}]_{L\times M}$. Instead of normalizing each row of $[b_{ij}]_{L\times M}$ as is done in standard DR-AGG, reverse dynamic routing normalizes each column of $[b_{ij}]_{L\times M}$,

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{kj})} \tag{18}$$

Other detail of reversed dynamic routing is the same as the standard dynamic routing.

The reversed DR-AGG works like the multi-hop memory network in iteratively aggregating information (Sukhbaatar et al., 2015; Kumar et al., 2015).

3.1 Analysis

The DR-AGG is somewhat like attention mechanism (Bahdanau et al., 2014; Vaswani et al., 2017).however, there are differences.

In standard DR-AGG, each input capsule (encoding of each word) is employed as query vector to assign a proportion weight to each output capsule, and then sends messages to the output capsules in proportion. Thus, for all input capsules the total amount of messages sent from an input capsule are the same.

In reversed DR-AGG, each output capsule is used as query vector to assign a proportion weight to each input capsule and then receives messages from the input capsules in proportion. Thus, for all output capsules the total amount of message received by an output capsule is same.

The major difference between DR-AGG and self-attention (Lin et al., 2017; Yang et al., 2016) is that the query vector of self-attention is task dependent trainable parameters learned during the training phase, while the query vector of DR-AGG is each input or output capsule which is instance dependent and dynamically updated.

Dataset	Type	Train Size	Dev. Size	Test size	Classes	Averaged Length	Vocabulary Size
Yelp 2013	Document	62522	7773	8671	5	189	29.3k
Yelp 2014	Document	183019	22745	25399	5	197	49.6k
IMDB	Document	67426	8381	9112	10	395	61.1k
SST-1	Sentence	8544	1101	2201	5	18	16.3k
SST-2	Sentence	6920	872	1821	2	19	14.8k

Table 1: Statistics of the five datasets used in this paper

Additionally, the self-attention aggregation collects information in a bottom-up way, without considering the state of the final encoding. It is hard to avoid the problems of information redundancy and information loss. While in the standard DR-AGG, each word can iteratively decide *what* and *how much* information is to be sent to the final encoding.

4 Hierarchical Dynamic Routing for Long Text

The dynamic routing mechanism can aggregate the text sequence with any length, therefore it is able to handle long texts directly, such as the whole paragraphs or documents.

To further enhance the efficiency and scalability of information aggregation, we adopt a hierarchical dynamic routing mechanism to handle the long text. The hierarchical routing strategy can exploit more parallelization and speed up training and inference process. A similar strategy is also used in (Yang et al., 2016).

Concretely, we split a document into sentences, and apply the proposed dynamic routing mechanism on word and sentence levels separately. We first encode each sentence into a fixed-length vector, then convert the sentence encodings into document encoding.

5 Experiment

We test the empirical performance of our proposed model on 5 benchmark datasets for document and sentence level classification and compare our proposed model to other competitor models.

5.1 Datasets

To evaluate the effectiveness of our proposed aggregation method, we have conducted experiments on 5 datasets, the statistics of experimented datasets are shown in Table 1. As shown in the table, Yelp-2013, Yelp-2014, and IMDB are document level datasets, while SST-1 and SST-2 are sentence level datasets. Note that we use the same document level datasets provided in (Tang et al., 2015).

Yelp reviews Yelp-2013 and Yelp-2014 are reviews from Yelp, each example consists of several review sentences and a rating score range from 1 to 5 (higher is better).

IMDB is a movie review dataset extracted from IMDB website. It is a multi-sentence dataset that for each example there are several review sentences. A rating score range from 1 to 10 is also associated with each example.

SST-1 Stanford Sentiment Treebank is a movie review dataset which has been parsed and further splited to train/dev/test set (Socher et al., 2013). For each example in the dataset, there exists only one sentence and a label associated with it. And the labels can be one of {negative, somewhat negative, neutral, somewhat positive, positive}.

SST-2 This dataset is a binary-class version of SST-1, with neutral reviews removed and the remaining reviews categorized to either negative or positive.

	Yelp-2013	Yelp-2014	IMDB	SST-1	SST-2
Embedding size	300	300	300	300	300
LSTM hidden unit	200	200	200	200	200
Capsule dimension	200	200	200	200	200
Capsule number	5	5	5	5	5
Iteration number	3	3	3	3	3
Regularization rate	1e-5	1e-5	1e-6	1e-6	1e-5
Initial learning rate	0.0001	0.0002	0.0001	0.0001	0.0003
learning rate decay	0.9	0.9	0.95	0.95	0.95
learning rate decay steps	1000	1000	1000	500	500
Initial Batch size	32	32	32	64	64
Batch size low bound	32	32	32	16	16
Dropout rate	0.2	0.2	0.2	0.2	0.5

Table 2: Detailed hyper-parameter settings

5.2 Training

Given a training set $\{x^{(i)}, t^{(i)}\}_{i=1}^{N}$, where $x^{(i)}$ is an example of the training set and $t^{(i)}$ is the corresponding label, the goal is to minimize the cross-entropy loss $\mathcal{J}(\theta)$:

$$\mathcal{J}(\theta) = -\frac{1}{N}\sum_{i} \log p(t^{(i)}|x^{(i)};\theta) + \lambda||\theta||_2^2, \tag{19}$$

where θ represents all of the parameters.

The Adam optimizer is applied to update the parameters (Kingma and Ba, 2014). Table 2 displays the detailed hyper-parameter settings. To prevent overfitting, the L2 regularization term is introduced to our loss function. We also adopt early stop strategy, The training process will be stopped after seven epochs of no improvement on development set is observed.To further avoid overfitting, dropout is applied before the biLSTM encoder and hidden layer of classifier **MLP**.

The mini-batch size is set to 32 for document level dataset, 64 for sentence level dataset, examples are sampled from a sliding bucket to speed up the training process. Data is sorted by the length of sentence, and we first sample a window on the sorted data, we call the window "sliding bucket" and then sample a batch of examples from the sliding bucket, we double the window size after an epoch of no improvement on development set, through such a strategy, we are able to considerably speed up training while retaining randomness. Also, batch size is halved after an epoch of no improvement on development set until it reaches the low bound batch size. We also utilize a data preparation queue to parallelize data preparation and training.

Word embedding is initialized from pre-trained Glove (Pennington et al., 2014). We randomly initialize word vectors for words that doesn't appear in Glove. Network weights are initialized with Xavier Normalization (Glorot and Bengio, 2010). A more detailed hyper-parameter setting can be referred to hyper-parameter Table 2. And hyper-parameters are determined using grid search strategy.

5.3 Experimental Results

We evaluate several aggregation methods on five text classification datasets, in which Yelp-2013, Yelp-2014 and IMDB are document-level datasets, and SST-1 and SST-2 are sentence-level datasets. Since max pooling, average pooling and self-attention are most related to our proposed DR-AGG, we mainly compare DR-AGG to these three methods.

Table 3 gives the results for different methods, the last two rows are our model (standard DR-AGG and reversed DR-AGG), the table shows that our proposed dynamic routing performed the best on all datasets. In document-level text classification, specifically Yelp 2013 Yelp 2014 and IMDB, DR-AGG outperforms previous models best results by 2.5%, 3.0% and 1.6% respectively. In sentence-level text

	Yelp-2013	Yelp-2014	IMDB	SST-1	SST-2
RNTN+Recurrent (Socher et al., 2013)	57.4	58.2	40.0	-	-
CNN-non-static (Kim, 2014)	-	-	-	48.0	87.2
Paragraph-Vec (Le and Mikolov, 2014)	-	-	-	48.7	**87.8**
MT-LSTM (F2S) (Liu et al., 2015)	-	-	-	49.1	87.2
UPNN(np UP) (Tang et al., 2015)	57.7	58.5	40.5	-	-
UPNN(full) (Tang et al., 2015)	59.6	60.8	43.5	-	-
Cached LSTM (Xu et al., 2016)	59.4	59.2	42.1	-	-
Max pooling	61.1	61.2	41.1	48.0	87.0
Average pooling	60.7	60.6	39.1	46.2	85.2
Self-attention	61.0	61.5	43.3	48.2	86.4
Standard DR-AGG	**62.1**	**63.0**	**45.1**	**50.5**	87.6
Reverse DR-AGG	61.6	62.5	44.5	49.3	87.2

Table 3: Experimental result comparison on five datasets. For the document-level datasets, hierarchical aggregation is used for both self-attention and DR-AGGs.

(1) so relentlessly wholesome it made me want to swipe something .
(2) so relentlessly wholesome it made me want to swipe something .
(3) so relentlessly wholesome it made me want to swipe something .

Table 4: A visualization to show the perspective of a sentence from 3 different upper level capsule. A deeper color indicates more information of the associated word is routed to the corresponding capsule.

classification, such as SST-1 SST-2, our model also achieves better results. Compared to max pooling, average pooling and self-attention, which are closely related to our model, DR-AGGs significantly improves the performance. For example the standard DR-AGG outperforms the max pooling approach by 1%, 1.8%, 4%,2.5% and 0.4% on Yelp 2013,Yelp 2014, IMDB, SST-1 and SST-2. It empirically shows that our proposed dynamic routing policy is the most effective method on aggregating information.

It is worth to note the reversed DR-AGG is inferior to the standard DR-AGG by a small margin, although it has also achieved better results than the other aggregation methods and SOTA approaches. As discussion before, the reversed DR-AGG have much resemblance with the attention using output capsule as query vector. Not all of the input capsules would be selected by the reversed DR-AGG, while in the standard DR-AGG, the information of all the input capsules need be sent to the output capsules.

Effects of Iterative Routing We also study how the iteration number affect the performance of aggregation on the SST-2 dataset. Figure 2 shows the comparison of 1 - 5 iterations in the standard DR-AGG. The capsule number is set to 1, 2, 3 and 4 for each comparison respectively. We found that the performances on several different capsule number setting reach the best when iteration is set to 3. The results indicate the dynamic routing is contributing to improve the performance.

Visualization Additionally, we visualize how much information each input capsule sends to the output capsules. As shown in Table 4, the visualization experiment was conducted with the setting on three output capsules. The i-th column represents the i-th input capsule, while the j-th row is the j-th output capsule. The color density of each word denotes the proportion c_{ij} in equation 15. A deeper color indicates more information of the concerned word is routed to the output capsule.

Intuitively, the different part of the sentence is routed to three different capsules. In another word, each capsule has a different perspective or focus of the sequence. Therefore, DR-AGG can avoid the problem of information redundancy and information missing.

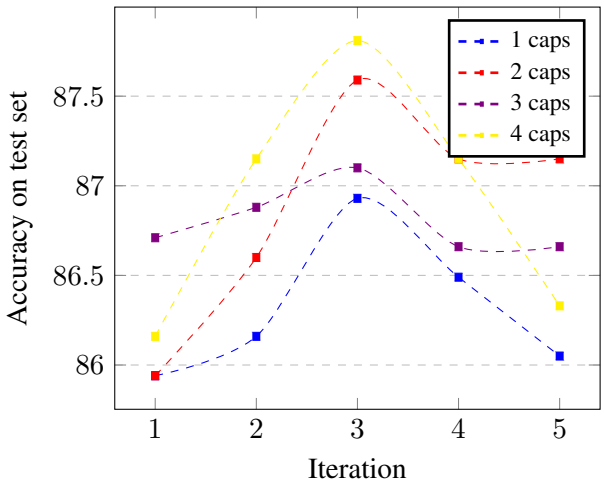


Figure 2: Relationship between test accuracy and routing iteration, where the vertical axis denotes test accuracy and the horizontal axis denotes routing iteration. When the iteration is set to 3 performance peaks on several different capsule number setting

6 Related Work

Currently, much attention has been paid to how developing a sophisticated encoding models to capture the long and short term dependency information in a sequence. Specific to text classification task, most of the models cannot deal with the texts of several sentences (paragraphs, documents), such as MV-RNN (Socher et al., 2012), RNTN (Socher et al., 2013), CNN (Kim, 2014), AdaSent (Zhao et al., 2015), and so on. The simple neural bag-of-words model can deal with long texts, but it loses the word order information. PV (Le and Mikolov, 2014) works in an unsupervised way, and the learned vector cannot be fine-tuned on the specific task. There are also many works (Liu et al., 2015; Xu et al., 2016; Cheng et al., 2016) to improve LSTM's ability to carrying information for a long distance.

A line of orthogonal researches (Lin et al., 2017; Yang et al., 2016; Shen et al., 2018a; Shen et al., 2018b) is to introduce attention mechanism (Vaswani et al., 2017) to weighted average the outputs of CNN/RNN layer. The attention mechanism can effectively reduce the burden of CNN/RNN. The CNN/RNN encoding layer is only expected to extract local context information for each word, while the global semantics of text sequence can be aggregated from the local encoding vectors.

The attention based aggregation collects information in a bottom-up way, without considering the state of the final encoding. It is hard to avoid the problems of information redundancy or information lost. An improved idea is to use multi-hop attention, like memory network (Sukhbaatar et al., 2015; Kumar et al., 2015), to iterative aggregate information. This idea is equivalent to our proposed reversed dynamic routing mechanism.

Different from the attention based aggregation methods, aggregation via dynamic routing is iteratively deciding that *what* and *how much* information need be transfer to the final encoding of each word.

7 Conclusion

In this paper, we focus on how to obtain a fixed-size encoding of text sequence by aggregating the encodings of each word. Although we use LSTM hidden states as word encoding in this paper, the other word encodings, such as convolved n-gram, could be alternatively used. We introduced a fixed-size encoding of text sequence with dynamic routing mechanism. Experimental results of five text classification tasks show that the model outperforms other encoding models by a significant margin.

In the future, we would like to investigate more sophisticated routing policy for better encoding the text sequence. Besides, dynamic routing should also be useful to improve the encoder in the sequence-to-sequence tasks (Sutskever et al., 2014).

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. The research work is supported by the National Key Research and Development Program of China (No. 2017YFB1002104), Shanghai Municipal Science and Technology Commission (No. 17JC1404100 and 16JC1420401), and National Natural Science Foundation of China (No. 61672162 and 61751201).

References

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. *ArXiv e-prints*, September.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 551–561.

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

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

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of ICML*.

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

Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 2326–2335.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. 2017. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 3859–3869.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018a. DISAN: Directional self-attention network for RNN/CNN-free language understanding. In *AAAI Conference on Artificial Intelligence*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. 2018b. Bi-directional block self-attention for fast and memory-efficient sequence modeling.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, pages 1201–1211.

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

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

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

Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1014–1023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Cached long short-term memory neural networks for document-level sentiment classification. *CoRR*, abs/1610.04989.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. *arXiv preprint arXiv:1504.05070*.

A Full End-to-End Semantic Role Labeler, Syntax-agnostic Over Syntax-aware?

Jiaxun Cai[1,2], **Shexia He**[1,2], **Zuchao Li**[1,2], **Hai Zhao**[1,2*]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, China
{caijiaxun, heshexia, charlee}@sjtu.edu.cn,
zhaohai@cs.sjtu.edu.cn

Abstract

Semantic role labeling (SRL) is to recognize the predicate-argument structure of a sentence, including subtasks of predicate disambiguation and argument labeling. Previous studies usually formulate the entire SRL problem into two or more subtasks. For the first time, this paper introduces an end-to-end neural model which unifiedly tackles the predicate disambiguation and the argument labeling in one shot. Using a biaffine scorer, our model directly predicts all semantic role labels for all given word pairs in the sentence without relying on any syntactic parse information. Specifically, we augment the BiLSTM encoder with a non-linear transformation to further distinguish the predicate and the argument in a given sentence, and model the semantic role labeling process as a word pair classification task by employing the biaffine attentional mechanism. Though the proposed model is syntax-agnostic with local decoder, it outperforms the state-of-the-art syntax-aware SRL systems on the CoNLL-2008, 2009 benchmarks for both English and Chinese. To our best knowledge, we report the first syntax-agnostic SRL model that surpasses all known syntax-aware models.

1 Introduction

Semantic role labeling (SRL) is a shallow semantic parsing, which is dedicated to identifying the semantic arguments of a predicate and labeling them with their semantic roles. SRL is considered as one of the core tasks in the natural language processing (NLP), which has been successfully applied to various downstream tasks, such as information extraction (Bastianelli et al., 2013), question answering (Shen and Lapata, 2007; Berant et al., 2013), machine translation (Xiong et al., 2012; Shi et al., 2016).

Typically, SRL task can be put into two categories: constituent-based (i.e., phrase or span) SRL and dependency-based SRL. This paper will focus on the latter one popularized by CoNLL-2008 and 2009 shared tasks (Surdeanu et al., 2008; Hajič et al., 2009). Most conventional SRL systems relied on sophisticated handcraft features or some declarative constraints (Pradhan et al., 2005; Zhao et al., 2009a), which suffers from poor efficiency and generalization ability. A recently tendency for SRL is adopting neural networks methods attributed to their significant success in a wide range of applications (Bai and Zhao, 2018; Zhang and Zhao, 2018). However, most of those works still heavily resort to syntactic features. Since the syntactic parsing task is equally hard as SRL and comes with its own errors, it is better to get rid of such prerequisite as in other NLP tasks. Accordingly, Marcheggiani et al. (2017) presented a neural model putting syntax aside for dependency-based SRL and obtain favorable results, which overturns the inherent belief that syntax is indispensable in SRL task (Punyakanok et al., 2008).

Besides, SRL task is generally formulated as multi-step classification subtasks in pipeline systems, consisting of predicate identification, predicate disambiguation, argument identification and argument classification. Most previous SRL approaches adopt a pipeline framework to handle these subtasks one

*Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100), National Natural Science Foundation of China (No. 61672343 and No. 61733011), Key Project of National Society Science Foundation of China (No. 15-ZDA041), The Art and Science Interdisciplinary Funds of Shanghai Jiao Tong University (No. 14JCRZ04).

Proceedings of the 27th International Conference on Computational Linguistics, pages 2753–2765
Santa Fe, New Mexico, USA, August 20-26, 2018.

after another. Until recently, some works (Zhou and Xu, 2015; He et al., 2017) introduce end-to-end models for span-based SRL, which motivates us to explore integrative model for dependency SRL.

In this work, we propose a syntactic-agnostic end-to-end system, dealing with predicate disambiguation and argument labeling in one model, unlike previous systems that treat the predicate disambiguation as a subtask and handle it separately. In detail, our model contains (1) a deep BiLSTM encoder, which is able to distinguish the predicates and arguments by mapping them into two different vector spaces, and (2) a biaffine attentional (Dozat and Manning, 2017) scorer, which unifiedly predicts the semantic role for argument and the sense for predicate.

We experimentally show that though our biaffine attentional model remains simple and does not rely on any syntactic feature, it achieves the best result on the benchmark for both Chinese and English even compared to syntax-aware systems. In summary, our major contributions are shown as follows:

- We propose an accurate syntax-agnostic model for neural SRL, which outperforms the best reported syntax-aware model, breaking the long-held belief that syntax is a prerequisite for SRL.
- Our model gives state-of-the-art results on the CoNLL-2008, CoNLL-2009 English and Chinese benchmarks, scoring 85.0% F_1, 89.6% F_1 and 84.4% F_1, respectively.
- Our work is the first attempt to apply end-to-end model for dependency-based SRL, which tackles the predicate disambiguation and the argument labeling subtasks in one shot.

2 Semantic Structure Decomposition

SRL includes two subtasks: predicate identification/disambiguation and argument identification/labeling. Since the CoNLL-2009 dataset provides the gold predicates, most previous neural SRL systems use a default model to perform predicate disambiguation and focus on argument identification/labeling. Despite nearly all SRL work adopted the pipeline model with two or more components, Zhao and Kit (2008) and Zhao et al. (2013) presented an end-to-end solution for the entire SRL task with a word pair classifier. Following the same formulization, we propose the first neural SRL system that uniformly handles the tasks of predicate disambiguation and argument identification/labeling.

In semantic dependency parsing, we can always identify two types of words, semantic head (predicate) and semantic dependent (argument). To build the needed predicate-argument structure, the model only needs to predict the role of any word pair from the given sentence. For the purpose, an additional role label *None* and virtual root node $<VR>$ are introduced. The *None* label indicates that there is no semantic role relationship inside the word pair. We insert a virtual root $<VR>$ in the head of the sentence, and set it as the semantic head of all the predicates. By introducing the *None* label and the $<VR>$ node, we construct a semantic tree rooted at the $<VR>$ node with several virtual arcs labeled with *None*. Thus, the predicate disambiguation and argument identification/labeling tasks can be naturally regarded as the labeling process over all the word pairs. Figure 1 shows an example of the semantic graph augmented with a virtual root and virtual arc, and Table 1 lists all the corresponding word pair examples, in which two types of word pairs are included, $<VR>$ followed by predicate candidates[1] and a known predicate collocated with every words in the sentence as argument candidates. Note that since the nominal predicate sometimes takes itself as its argument, the predicate itself is also included in the argument candidate list.

3 Model

Our model contains two main components: (1) a deep BiLSTM encoder that takes each word embedding $\boldsymbol{e}$ of the given sentence as input and generates dense vectors for both words in the to-be-classified word pair respectively, (2) a biaffine attentional scorer which takes the hidden vectors for the given word pair as input and predict a label score vector. Figure 2 provides an overview of our model.

[1]Note that there is a key difference between CoNLL 2008 and 2009 shared task for English, the latter has specified the predicate in the data so that here we have only one sample for $<VR>$-predicate pair to make predicate disambiguation. For the former, predicate candidates should be every words in the given sentence. More details are seen in Section 4.4.

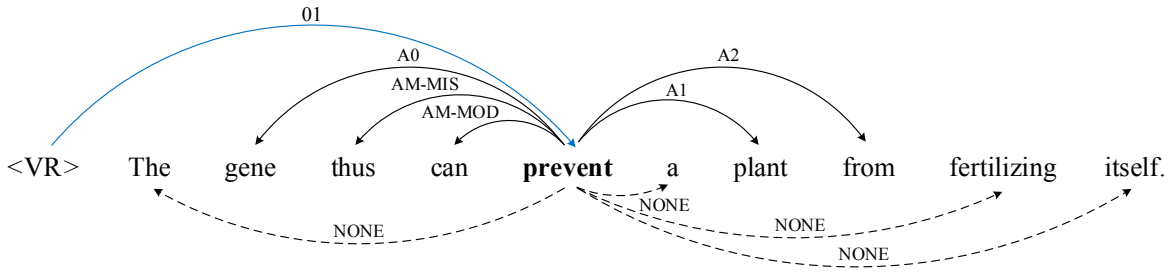


Figure 1: The dependency graph with a virtual node <VR> and virtual arcs with *None* label.

Head	dependent (word pair)	Label	Head	dependent (word pair)	Label
<VR>	prevent	01	<VR>	fertilizing	01
prevent	The	None	fertilizing	The	None
prevent	gene	A0	fertilizing	gene	None
prevent	thus	AM-MIS	fertilizing	thus	None
prevent	can	AM-MOD	fertilizing	can	None
prevent	prevent	None	fertilizing	prevent	None
...	...	...	...	...	...
prevent	itself	None	fertilizing	itself	A1

Table 1: Word pairs for semantic role label classification.

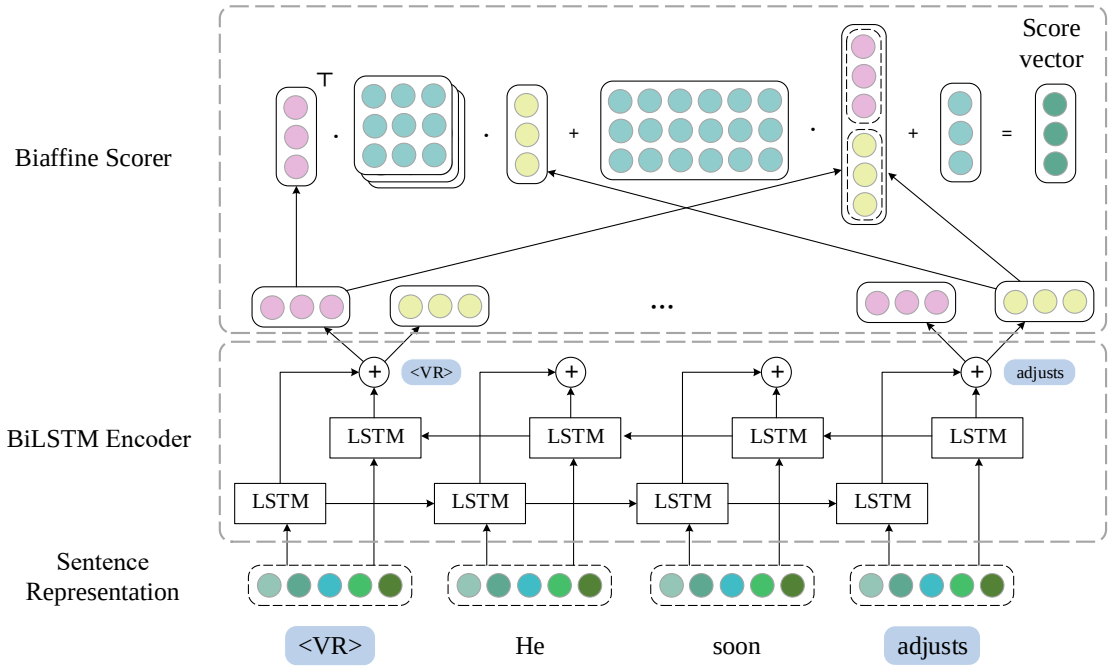


Figure 2: An overview of our model.

3.1 Bidirectional LSTM Encoder

Word Representation The word representation of our model is the concatenation of several vectors: a randomly initialized word embedding $\boldsymbol{e}^{(r)}$, a pre-trained word embedding $\boldsymbol{e}^{(p)}$, a randomly initialized part-of-speech (POS) tag embedding $\boldsymbol{e}^{(pos)}$, a randomly initialized lemma embedding $\boldsymbol{e}^{(l)}$. Besides, since previous work (He et al., 2018) demonstrated that the predicate-specific feature is helpful in promoting the role labeling process, we employ an indicator embedding $\boldsymbol{e}^{(i)}$ to indicate whether a word is a predicate when predicting and labeling the arguments for each given predicate. The final word representation is given by $\boldsymbol{e} = \boldsymbol{e}^{(r)} \oplus \boldsymbol{e}^{(p)} \oplus \boldsymbol{e}^{(l)} \oplus \boldsymbol{e}^{(pos)} \oplus \boldsymbol{e}^{(i)}$, where $\oplus$ is the concatenation operator.

Encoder As commonly used to model the sequential input in most NLP tasks (Wang et al., 2016; He et al., 2018), BiLSTM is adopted for our sentence encoder. By incorporating a stack of two distinct LSTMs, BiLSTM processes an input sequence in both forward and backward directions. In this way, the BiLSTM encoder provides the ability to incorporate the contextual information for each word.

Given a sequence of word representation $S = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_N\}$ as input, the i-th hidden state $\boldsymbol{g}_i$ is encoded as follows:

$$\boldsymbol{g}_i^f = LSTM^{\mathcal{F}}\left(\boldsymbol{e}_i, \boldsymbol{g}_{i-1}^f\right), \quad \boldsymbol{g}_i^b = LSTM^{\mathcal{B}}\left(\boldsymbol{e}_i, \boldsymbol{g}_{i+1}^b\right), \quad \boldsymbol{g}_i = \boldsymbol{g}_i^f \oplus \boldsymbol{g}_i^b,$$

where $LSTM^{\mathcal{F}}$ denotes the forward LSTM transformation and $LSTM^{\mathcal{B}}$ denotes the backward LSTM transformation. $\boldsymbol{g}_i^f$ and $\boldsymbol{g}_i^b$ are the hidden state vectors of the forward LSTM and backward LSTM respectively.

3.2 Biaffine Attentional Role Scorer

Typically, to predict and label arguments for a given predicate, a role classifier is employed on top of the BiLSTM encoder. Some work like (Marcheggiani et al., 2017) shows that incorporating the predicate's hidden state in their role classifier enhances the model performance, while we argue that a more natural way to incorporate the syntactic information carried by the predicate is to employ the attentional mechanism. Our model adopts the recently introduced biaffine attention (Dozat and Manning, 2017) to enhance our role scorer. Biaffine attention is a natural extension of bilinear attention (Luong et al., 2015) which is widely used in neural machine translation (NMT).

Nonlinear Affine Transformation Usually, a BiLSTM decoder takes the concatenation $\boldsymbol{g}_i$ of the hidden state vectors as output for each hidden state. However, in the SRL context, the encoder is supposed to distinguish the currently considered predicate from its candidate arguments. To this end, we perform two distinct affine transformations with a nonlinear activation on the hidden state $\boldsymbol{g}_i$, mapping it to vectors with smaller dimensionality:

$$\boldsymbol{h}_i^{(pred)} = ReLU\left(\boldsymbol{W}^{(pred)}\boldsymbol{g}_i + \boldsymbol{b}^{(pred)}\right), \quad \boldsymbol{h}_i^{(arg)} = ReLU\left(\boldsymbol{W}^{(arg)}\boldsymbol{g}_i + \boldsymbol{b}^{(arg)}\right),$$

where $ReLU$ is the rectilinear activation function, $\boldsymbol{h}_i^{(pred)}$ is the hidden representation for the predicate and $\boldsymbol{h}_i^{(arg)}$ is the hidden representation for the candidate arguments.

By performing such transformations over the encoder output to feed the scorer, the latter may benefit from deeper feature extraction. First, ideally, instead of keeping both features learned by the two distinct LSTMs, the scorer is now enabled to learn features composed from both recurrent states together with reduced dimensionality. Second, it provides the ability to map the predicates and the arguments into two distinct vector spaces, which is essential for our tasks since some words can be labeled as predicate and argument simultaneously. Mapping a word into two different vectors can help the model disambiguate the role that it plays in different context.

Biaffine Scoring In the standard NMT context, given a target recurrent output vector $h_i^{(t)}$ and a source recurrent output vector $h_j^{(s)}$, a bilinear transformation calculates a score s_{ij} for the alignment:

$$s_{ij} = \boldsymbol{h}_i^{\top(t)}\boldsymbol{W}\boldsymbol{h}_j^{(s)},$$

However, considering that in a traditional classification task, the distribution of classes is often uneven, and that the output layer of the model normally includes a bias term designed to capture the prior probability $P(y_i = c)$ of each class, with the rest of the model focusing on learning the likelihood of each class given the data $P(y_i = c|x_i)$, (Dozat and Manning, 2017) introduced the bias terms into the bilinear attention to address such uneven problem, resulting in a biaffine transformation. The biaffine transformation is a natural extension of the bilinear transformation and the affine transformation. In SRL task, the distribution of the role labels is similarly uneven and the problem comes worse after we introduce the additional <*VR*> node and *None* label, directly applying the primitive form of bilinear attention would fail to capture the prior probability $P(y_i = c_k)$ for each class. Thus, the biaffine attention introduced in our model would be extremely helpful for semantic role prediction.

It is worth noting that in our model, the scorer aims to assign a score for each specific semantic role. Besides learning the prior distribution for each label, we wish to further capture the preferences for the label that a specific predicate-argument pair can take. Thus, our biaffine attention contains two distinguish bias terms:

$$\boldsymbol{s}_{ij} = \boldsymbol{h}_i^{\top(arg)} \boldsymbol{W}^{(role)} \boldsymbol{h}_j^{(pred)} \tag{1}$$

$$+ \boldsymbol{U}^{(role)} \left(\boldsymbol{h}_i^{(arg)} \oplus \boldsymbol{h}_j^{(pred)} \right) \tag{2}$$

$$+ \boldsymbol{b}^{(role)}, \tag{3}$$

where $\boldsymbol{W}^{(role)}$, $\boldsymbol{U}^{(role)}$ and $\boldsymbol{b}^{(role)}$ are parameters that will be updated by some gradient descent methods in the learning process. There are several points that should be paid attention to in the above biaffine transformation. First, since our goal is to predict the label for each pair of $\boldsymbol{h}_i^{(arg)}$, $\boldsymbol{h}_j^{(pred)}$, the output of our biaffine transformation should be a vector of dimensionality N_r instead of a real value, where N_r is the number of all the candidate semantic labels. Thus, the bilinear transformation in Eq. (1) maps two input vectors into another vector. This can be accomplished by setting $\boldsymbol{W}^{(role)}$ as a $(d_h \times N_r \times d_h)$ matrix, where d_h is the dimensionality of the hidden state vector. Similarly, the output of the linear transformation in Eq. (2) is also a vector by setting $\boldsymbol{U}^{(role)}$ as a $(N_r \times 2d_h)$ matrix. Second, Eq. (2) captures the preference of each role (or sense) label condition on taking the j-th word as predicate and the i-th word as argument. Third, the last term $\boldsymbol{b}^{(role)}$ captures the prior probability of each class $P(y_i = c_k)$. Notice that Eq. (2) and (3) capture different kinds of bias for the latent distribution of the label set.

Given a sentence of length L (including the <*VR*> node), for one of its predicates w_j, the scorer outputs a score vector $\{\boldsymbol{s}_{1j}, \boldsymbol{s}_{2j}, \cdots, \boldsymbol{s}_{Lj}\}$. Then our model picks as its output the label with the highest score from each score vector: $y_{ij} = \arg\max_{1 \le k \le N_r} (\boldsymbol{s}_{ij}[k])$, where $\boldsymbol{s}_{ij}[k]$ denotes the score of the k-th candidate semantic label.

4 Experiments

4.1 Dataset and Training Detail

We evaluate our model[2] on English and Chinese CoNLL-2009 datasets with the standard split into training, test and development sets. The pre-trained embedding for English is trained on Wikipedia and Gigaword using the GloVe (Pennington et al., 2014), while those for Chinese is trained on Wikipedia. Our implementation uses the DyNet[3] library for building the dynamic computation graph of the network.

When not otherwise specified, our model uses: 100-dimensional word, lemma, pre-trained and POS tag embeddings and 16-dimensional predicate-specific indicator embedding; and a 20% chance of dropping on the whole word representation; 3-layer BiLSTMs with 400-dimensional forward and backward LSTMs, using the form of recurrent dropout suggested by (Gal and Ghahramani, 2016) with an 80% keep probability between time-steps and layers; two 300-dimensional affine transformation with the ReLU non-linear activation on the output of BiLSTM, also with an 80% keep probability.

[2]The code is available at `https://github.com/JiaxunCai/Dynet-Biaffine-SRL`

[3]https://github.com/clab/dynet

Syntax-aware system (single)	P	R	F_1
Zhao et al. (2009a)	–	–	86.2
Zhao et al. (2009c)	–	–	85.4
Björkelund et al. (2010)	87.1	84.5	85.8
Lei et al. (2015)	–	–	86.6
FitzGerald et al. (2015)	–	–	86.7
Roth and Lapata (2016)	88.1	85.3	86.7
Marcheggiani and Titov (2017)	89.1	86.8	88.0
He et al. (2018)	**89.7**	**89.3**	**89.5**
Syntax-aware system (ensemble)	P	R	F_1
Roth and Lapata (2016)	90.3	85.7	87.9
Marcheggiani and Titov (2017)	90.5	87.7	89.1
Syntax-agnostic system	P	R	F_1
Marcheggiani et al. (2017)	88.7	86.8	87.7
He et al. (2018)	89.5	87.9	88.7
This work	**89.9**	**89.2**	**89.6**

Table 2: Results on English in-domain (WSJ) test set.

The parameters in our model are optimized with Adam (Kingma and Ba, 2015), which keeps a moving average of the L2 norm of the gradient for each parameter throughout training and divides the gradient for each parameter by this moving average, ensuring that the magnitude of the gradients will on average be close to one. For the parameters of optimizer, we follow the settings in (Dozat and Manning, 2017), with $\beta_1 = \beta_2 = 0.9$ and learning rate 0.002, annealed continuously at a rate of 0.75 every $5,000$ iterations, with batches of approximately $5,000$ tokens. The maximum number of epochs of training is set to 50.

4.2 Results

Tables 2 and 3 report the comparison of performance between our model and previous dependency-based SRL model on both English and Chinese. Note that the predicate disambiguation subtask is unifiedly tackled with arguments labeling in our model with precisions of 95.0% and 95.6% respectively on English and Chinese test sets in our experiments[4]. The proposed model accordingly outperforms all the SRL systems so far on both languages, even including those syntax-aware and ensemble ones. The improvement grows even larger when comparing only with the single syntax-agnostic models.

For English, our syntax-agnostic model even slightly outperforms the best reported syntax-aware model (He et al., 2018) with a margin of 0.1% F_1. Compared to syntax-agnostic models, our model overwhelmingly outperforms (with an improvement of 0.9% F_1) the previous work (He et al., 2018).

Although we used the same parameters as for English, our model substantially outperforms the state-of-art models on Chinese, demonstrating that our model is robust and less sensitive to the parameter selection. For Chinese, the proposed model outperforms the best previous model (He et al., 2018) with a considerable improvement of 1.5% F_1, and surpasses the best single syntax-agnostic model (He et al., 2018) with a margin of 2.5% F_1.

Table 4 compares the results on English out-of-the-domain (Brown) test set, from which our model still remains strong. The proposed model gives a comparable result with the highest score from syntax-aware model of (He et al., 2018), which affirms that our model does well learn and generalize the latent semantic preference of the data.

Results on both in-domain and out-of-the-domain test sets demonstrate the effectiveness and the robustness of the proposed model structure—the non-linear transformation after the BiLSTM serves to distinguish the predicate from argument while the biaffine attention tells what to attend for each candidate argument. In Section 4.3.2, we will get an insight into our model and explore how each individual

[4]Note that we give comparable predicate disambiguation results with He et al. (2018), with 95.01% and 95.58% F_1 on development and test sets, respectively.

Syntax-aware system	P	R	F_1
Zhao et al. (2009a)	80.4	75.2	77.7
Björkelund et al. (2009)	82.4	75.1	78.6
Roth and Lapata (2016)	83.2	75.9	79.4
Marcheggiani and Titov (2017)	84.6	80.4	82.5
He et al. (2018)	**84.2**	**81.5**	**82.8**
Syntax-agnostic system	P	R	F_1
Marcheggiani et al. (2017)	83.4	79.1	81.2
He et al. (2018)	84.5	79.3	81.8
This work	**84.7**	**84.0**	**84.3**

Table 3: Results on Chinese in-domain test set.

Syntax-aware system	P	R	F_1
Björkelund et al. (2010)	75.7	72.2	73.9
Lei et al. (2015)	–	–	75.6
FitzGerald et al. (2015)	–	–	75.2
Roth and Lapata (2016)	76.9	73.8	75.3
Marcheggiani and Titov (2017)	78.5	75.9	77.2
He et al. (2018)	**81.9**	**76.9**	**79.3**
Syntax-agnostic system	P	R	F_1
Marcheggiani et al. (2017)	79.4	76.2	77.7
He et al. (2018)	81.7	76.1	78.8
This work	**79.8**	**78.3**	**79.0**

Table 4: Results on English out-of-the-domain (Brown) test set.

	AL			**PD**
System	P	R	F_1	P
without POS	89.5	89.1	89.3	94.9
without lemma	89.5	89.3	89.4	94.9
without indicator	89.1	88.5	88.8	95.0
This work	**89.9**	**89.2**	**89.6**	**95.0**

Table 5: Contribution of the input representation. Acronyms used: **AL**-argument labeling, **PD**-predicate disambiguation.

component impacts the model performance.

4.3 Ablation Analysis

4.3.1 Word Representation

To learn how the input word representation choice impacts our model performance, we conduct an ablation study on the English test set whose results are shown in Table 5. Since we deal with the two subtasks in a single model, the choice of word representation will simultaneously influence the results of both of them. Besides the results of argument labeling, we also report the precision of predicate disambiguation.

The results demonstrate that the multiple dimensional indicator embedding proposed by (He et al., 2018) contributes the most to the final performance of our model. It is consistent with the conclusion in (Marcheggiani et al., 2017) which argue that encoding predicate information promotes the SRL model. It is interesting that the impact of POS tag embedding (about 0.3% F_1) is less compared to the previous works, which possibly allows us to build an accuracy model even when the POS tag label is unavailable.

4.3.2 Into the Model

In this section, we get insight into the proposed model, exploring how the deep BiLSTM encoder and the biaffine attention affect the labeling results respectively. Specifically, we present two groups of results on the CoNLL-2009 English test set. 1) Shallow biaffine attentive (SBA) labeler. Instead of mapping the output of the BiLSTM into two distinct vector spaces, we apply a single non-linear affine transformation on the output. The single transformation just serves to reduce the dimensionality and does not differ the predicates from the arguments. 2) Deep bilinear attentive (DBA) labeler. We apply the primitive form of bilinear attention in the scorer by removing the two bias terms of the biaffine transformation. By this means, we learn to what extent can the bias terms fit the prior distribution of the data. Results of the two experiments are shown in Table 6.

The results show that the bias terms in biaffine attention play an important role in promoting the model performance. Removal of the bias terms dramatically declines the performance by 1.7% F_1. Thus we can draw a conclusion that the bias term does well in fitting the prior distribution and global preference

System	AL			PD
	P	R	F_1	P
SBA-labeler	89.5	88.8	89.1	94.7
DBA-labeler	88.1	87.7	87.9	94.3
This work	**89.9**	**89.2**	**89.6**	**95.0**

Table 6: Contribution of the model components.

System	AL			PD
	P	R	F_1	P
with pruning	88.2	85.6	86.9	95.0
without pruning	**89.9**	**89.2**	**89.6**	**95.0**

Table 7: Comparison of results with and without argument candidate pruning.

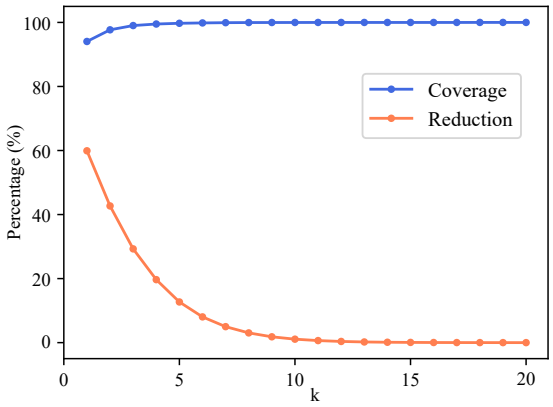


Figure 3: Coverage rate of true arguments and reduction rate of argument candidates against the pruning order k on the English training set.

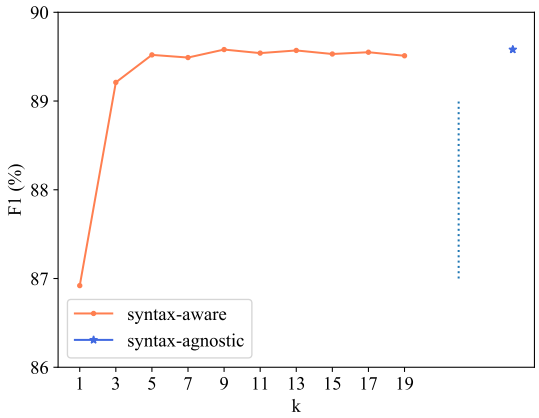


Figure 4: F1 scores against the pruning order k on English test set.

of the data. The bilinear attentional model behaves more poorly since it struggles to learn the likelihood of each class on an uneven data set without knowledge about the prior distribution. Though the deep encoder contributes less to the performance, it also brings an improvement of 0.5% F_1. Note that the only difference of SBA-labeler of our standard model is whether the hidden representations of the arguments and the predicates lay in different vector spaces. Such a result confirms that distinguishing the predicates from the arguments in encoding process indeed enhances the model to some extent.

4.3.3 Syntax-aware and Syntax-agnostic

Noting that the work (Zhao and Kit, 2008) and (Zhao et al., 2013) are similar to ours in modeling the dependency-based SRL tasks as word pair classification, and that they successfully incorporate the syntactic information by applying argument candidate pruning, we further perform empirical study to explore whether employing such pruning method enhance or hinder our model. Specifically, we use the automatically predicted parse with moderate performance provided by CoNLL-2009 shared task, with the LAS score about 86%.

The pruning method is supposed to work since it can alleviate the imbalanced label distribution caused by introducing the *None* label. However, as shown in Table 7, the result is far from satisfying. The main reason might be the pruning algorithm is so strict that too many true arguments are falsely pruned. To address this problem, He et al. (2018) introduced an extended k-order argument pruning algorithm. Figure 3 shows the curves of coverage and reduction rate against the pruning order k on the English training set following (He et al., 2018). Following this work, we further perform different orders of pruning and obtain the F_1 scores curve shown in Figure 4. However, the k-order pruning does not boost the performance of our model. Table 8 presents the performance gap between syntax-agnostic and syntax-aware settings of the same models. Unlike the other two works, the introduction of syntax information fails to bring about bonus for our model. Nevertheless, it is worth noting that even when running without the syntax information, our mode still show a promising result compared to the other syntax-aware models.

System	syntax-agnostic	syntax-aware	Δ F_1
Marcheggiani and Titov (2017)	87.7	88.0	0.3
He et al. (2018) (CoNLL-2009 predicted)	88.7	89.5	0.8
He et al. (2018) (gold syntax)	88.7	90.3	1.6
This work (CoNLL-2009 predicted)	89.6	89.6	$\approx$0

Table 8: The absolute performance gaps between syntax-agnostic and syntax-aware settings. Both (He et al., 2018) and our models use 10-order pruning according to syntactic parse tree.

4.4 CoNLL 2008: Augment the Model with Predicate Identification

Though CoNLL-2009 provided the gold predicate beforehand, the predicate identification subtask is still indispensable for a real world SRL task. Thus, we further augment our model with the predicate identification ability.

Specifically, we first attach all the words in the sentence to the virtual root $<VR>$ and label the word which is not a predicate with the *None* role label. It should be noting that, in CoNLL-2009 settings, we just attach the predicates to the virtual root, since we do not need to distinguish the predicate from other word. The training scheme still keeps the same as that in CoNLL-2009 settings, while in testing phase, an additional procedure is performed to find out all the predicates of a given sentence.

First, our model is fed the representations of the virtual root and each word of the input sentence, identifying and disambiguating all the predicates of the sentence. Second, it picks each predicate predicted by the model with each word of the sentence to identify and label the semantic role in between, which remains the same as the model does on CoNLL-2009. The second phase is repeated until all the predicates have got its arguments being identified and labeled. We evaluate our model on CoNLL-2008 benchmark using the same hyperparameters settings mentioned in Section 4.1 except that we remove the predicate-specific indicator feature.

The F_1 scores on predicates identification and labeling of our model is 89.43%, which remain comparable with the most recent work (He et al., 2018) (90.53% F_1). As shown in Table 9, though tackling all the subtasks of CoNLL-2008 SRL unifiedly in a full end-to-end manner, our model outperforms the best reported results with a large margin of about 1.7% semantic F_1.

		Predicate Labeling			**Semantic Labeling**		
System	LAS	P	R	F_1	P	R	F_1
Johansson and Nugues (2008)	90.13	–	–	–	–	–	81.75 (80.37)
Zhao and Kit (2008)	87.52	–	–	–	80.57	74.97	77.67
Zhao et al. (2009b)	88.39	–	–	–	–	–	82.1 (80.53)
	89.28	–	–	–	–	–	82.5 (80.94)
Zhao et al. (2013)	88.39	–	–	(87.15)	–	–	82.5 (80.91)
	89.28	–	–	(86.47)	–	–	82.4 (80.88)
He et al. (2018) (syntax-aware)	86.0	89.73	91.35	90.53	83.9	82.7	83.3
He et al. (2018) (syntax-agnostic)	–	89.73	91.35	90.53	83.5	82.4	82.9
Ours (syntax-agnostic)	–	**88.9**	**90.0**	**89.4 (87.9)**	**84.7**	**85.2**	**85.0 (83.6)**

Table 9: Results on the CoNLL-2008 test set (WSJ). The results enclosed with parenthesis are evaluated on WSJ + Brown test set, following the official evaluation setting of CoNLL-2008 shared task.

5 Related Work

Semantic role labeling was pioneered by Gildea and Jurafsky (2002). Most traditional SRL models heavily rely on complex feature engineering (Pradhan et al., 2005; Zhao et al., 2009a; Björkelund et al., 2009). Among those early works, Pradhan et al. (2005) combined features derived from different syntactic parses based on SVM classifier, while Zhao et al. (2009a) exploited the abundant set of language-specific features that were carefully designed for SRL task.

In recent years, applying neural networks in SRL task has gained a lot of attention due to the impressive success of deep neural networks in various NLP tasks (Zhang et al., 2016; Cai et al., 2017; Qin et al., 2017; Cai and Zhao, 2017). Collobert et al. (2011) initially introduced neural networks into the SRL task. They developed a feed-forward network that employed a convolutional network as sentence encoder and a conditional random field as a role classifier. Foland and Martin (2015) extended their model to further use syntactic information by including binary indicator features. FitzGerald et al. (2015) exploited a neural network to unifiedly embed arguments and semantic roles, similar to the work (Lei et al., 2015) which induced a compact feature representation applying tensor-based approach. Roth and Lapata (2016) introduced the dependency path embedding to incorporate syntax and exhibited a notable success, while Marcheggiani and Titov (2017) employed the graph convolutional network to integrate syntactic information into their neural model.

Besides the above-mentioned works who relied on syntactic information, several works attempted to build SRL systems without or with little syntactic information. Zhou and Xu (2015) came up with an end-to-end model for span-based SRL and obtained surprising performance putting syntax aside. He et al. (2017) further extended their work with the highway network. Simultaneously, Marcheggiani et al. (2017) proposed a syntax-agnostic model with effective word representation for dependency-based SRL.

However, almost all of previous works treated the predicate disambiguation as individual subtasks, apart from (Zhao and Kit, 2008; Zhao et al., 2009a; Zhao et al., 2009c; Zhao et al., 2013), who presented the first end-to-end system for dependency SRL. For the neural models of dependency SRL, we have presented the first end-to-end solution that handles both semantic labeling subtasks in one single model. At the same time, our model enjoys the advantage that does not rely any syntactic information.

This work is also closely related to the attentional mechanism. The traditional attention mechanism was proposed by Bahdanau et al. (2015) in the NMT literature. Following the work (Luong et al., 2015) that encouraged substituting the MLP in the attentional mechanism with a single bilinear transformation, Dozat and Manning (2017) introduced the bias terms into the primitive form of bilinear attention and applied it for dependency parsing. They demonstrate that the bias terms help their model to capture the uneven prior distribution of the data, which is again verified by our practice on SRL in this paper.

Different from the latest strong syntax-agnostic models in (Marcheggiani and Titov, 2017) and (He et al., 2018) which both adopted sequence labeling formulization for the SRL task, this work adopts word pair classification scheme implemented by LSTM encoder and biaffine scorer. Compared to the previous state-of-the-art syntax-agnostic model in (He et al., 2018) whose performance boosting (more than 1% absolute gain) is mostly due to introducing the enhanced representation, namely, the CNN-BiLSTM character embedding from (Peters et al., 2018), our performance promotion mainly roots from model architecture improvement, which results in quite different syntax-aware enhanced impacts. Using the same latest syntax-aware k-order pruning, the syntax-agnostic backbone in (He et al., 2018) may receive about 1% performance gain, while our model is furthermore enhanced little. This comparison also suggests the possibility that maybe our model can be further improved by incorporating with the same character embedding as (He et al., 2018) does[5].

6 Conclusion and Future Work

This paper presents a full end-to-end neural model for dependency-based SRL. It is the first time that a SRL model shows its ability to unifiedly handle the predicate disambiguation and the argument labeling subtasks. Our model is effective while remains simple. Experiments show that it achieves the best scores on CoNLL benchmark both for English and Chinese, outperforming the previous state-of-the-art models even with syntax-aware features. Our further investigation by incorporating the latest syntax-aware pruning algorithm shows that the proposed model is insensitive to the input syntactic information, demonstrating an interesting performance style for the SRL task. Of course, we cannot exclude the possibility that the proposed model can be furthermore improved by other syntactic information integration ways, which is left for the future work.

[5]Such an attempt may be hindered by too luxurious computational resource requirement, as there comes extremely high graphic memory prerequisite when integrating both biaffine scorer and the ELMo character embedding.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of 3rd International Conference on Learning Representations (ICLR)*.

Hongxiao Bai and Hai Zhao. 2018. Deep enhanced representation for implicit discourse relation recognition. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*.

Emanuele Bastianelli, Giuseppe Castellucci, Danilo Croce, and Roberto Basili. 2013. Textual inference and meaning representation in human robot interaction. In *Proceedings of the Joint Symposium on Semantic Processing. Textual Inference and Structures in Corpora*, pages 65–69.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1533–1544, Seattle, Washington, USA, October.

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 43–48, Boulder, Colorado, June.

Anders Björkelund, Bohnet Bernd, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of the 23rd International Conference on Computational Linguistics (CoLING 2010)*, pages 33–36, Beijing, China, August.

Deng Cai and Hai Zhao. 2017. *Pair-Aware Neural Sentence Modeling for Implicit Discourse Relation Classification*. IEA/AIE 2017, Part II, LNAI 10351.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and accurate neural word segmentation for Chinese. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 608–615.

Ronan Collobert, Jason Weston, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(1):2493–2537.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of 5th International Conference on Learning Representations (ICLR)*.

Nicholas FitzGerald, Oscar Tckstrm, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 960–970.

William Foland and James Martin. 2015. Dependency-based semantic role labeling using convolutional neural networks. In *Joint Conference on Lexical and Computational Semantics*, pages 279–288.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: representing model uncertainty in deep learning. In *International Conference on International Conference on Machine Learning (ICML)*, pages 1050–1059.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado, June.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 473–483, Vancouver, Canada, July.

Shexia He, Zuchao Li, Hai Zhao, Hongxiao Bai, and Gongshen Liu. 2018. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic-semantic analysis with propbank and nombank. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL)*, pages 183–187.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization.

Tao Lei, Yuan Zhang, Llus Mrquez, Alessandro Moschitti, and Regina Barzilay. 2015. High-order low-rank tensors for semantic role labeling. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL: HLT)*, pages 1150–1160.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421, Lisbon, Portugal, September.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1506–1515, Copenhagen, Denmark, September.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 411–420, Vancouver, Canada, August.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL: HLT)*, New Orleans, Louisiana.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 581–588, Ann Arbor, Michigan, June.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric P. Xing. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1006–1017.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1192–1202, Berlin, Germany, August.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 12–21.

Chen Shi, Shujie Liu, Shuo Ren, Shi Feng, Mu Li, Ming Zhou, Xu Sun, and Houfeng Wang. 2016. Knowledge-based semantic embedding for machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2245–2254, Berlin, Germany, August.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning - Shared Task (CoNLL)*, pages 159–177, Manchester, England, August.

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. 2016. Learning distributed word representations for bidirectional lstm recurrent neural network. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2016)*, pages 527–533.

Deyi Xiong, Min Zhang, and Haizhou Li. 2012. Modeling the translation of predicate-argument structure for smt. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 902–911, Jeju Island, Korea, July.

Zhuosheng Zhang and Hai Zhao. 2018. One-shot learning for question-answering in gaokao history challenge. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*.

Zhisong Zhang, Hai Zhao, and Lianhui Qin. 2016. Probabilistic graph-based dependency parsing with convolutional neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1382–1392.

Hai Zhao and Chunyu Kit. 2008. Parsing syntactic and semantic dependencies with two single-stage maximum entropy models. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL)*, pages 203–207.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009a. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning - Shared Task (CoNLL)*, pages 61–66, Boulder, Colorado, June.

Hai Zhao, Wenliang Chen, and Chunyu Kit. 2009b. Semantic dependency parsing of NomBank and PropBank: An efficient integrated approach via a large-scale feature selection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 30–39, Singapore, August.

Hai Zhao, Wenliang Chen, and Guodong Zhou. 2009c. Multilingual dependency learning: A huge feature engineering method to semantic dependency parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning - Shared Task (CoNLL)*, pages 55–60, Boulder, Colorado, June.

Hai Zhao, Xiaotian Zhang, and Chunyu Kit. 2013. Integrative semantic dependency parsing via efficient large-scale feature selection. *Journal of Artificial Intelligence Research*, 46:203–233.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1127–1137, Beijing, China, July.

Authorship Attribution By Consensus Among Multiple Features

Jagadeesh Patchala
University of Cincinnati
Cincinnati
OH, USA
jagadeesh.patchala@gmail.com

Raj Bhatnagar
University of Cincinnati
Cincinnati
OH, USA
raj.bhatnagar@uc.edu

Abstract

Most existing research on authorship attribution uses various lexical, syntactic and semantic features. In this paper we demonstrate an effective template-based approach for combining various syntactic features of a document for authorship analysis. The parse-tree based features that we propose are independent of the topic of a document and reflect the innate writing styles of authors. We show that the use of templates including sub-trees of parse trees in conjunction with other syntactic features result in improved author attribution rates. Another contribution is the demonstration that Dempster's rule based combination of evidence from syntactic features performs better than other evidence-combination methods. We also demonstrate that our methodology works well for the case where actual author is not included in the candidate author set.

1 Introduction

In the past decade authorship analysis has become a significant need due to its various application areas such as identification of authors of anonymous posts in blogs (Koppel et al., 2011; Koppel and Yeron, 2014), of emails (Patchala et al., 2015), for copyright issues, plagiarism detection, and authentication of electronic documents (Abbasi and Chen, 2005; Chaski, 2005). Typically we encounter two types of author attribution problems. The first kind is the *closed author set problem* where the test-document is written by someone within the set of candidate authors being examined. The second kind is the *semi-closed authorship problem* where the test-document may or may not be written by someone in the set of candidate authors.

The main hypothesis underlying authorship attribution methods is that the grammatical style of an author remains the same across topics and is fairly unique to each author. This uniqueness of an author is easily identifiable when considering small sets of authors, and it expectedly dilutes as the author set becomes large. It is, therefore, desirable to discover from texts more such features that are unique to an author and help increase the discriminability of authors based on their writings. The focus of this paper is on identification and integration of features to enable improved authorship attribution.

We use template based classification and an evidence combination method for exploiting information embedded in different types of syntactic features of documents. We build templates using the frequencies of sub-trees of various heights derived from the parse trees of sentences in documents. Our intuitive basis for using these new features is that they capture the grammatical style very well. We consider the frequencies of sub-trees of heights two, three, and four, as

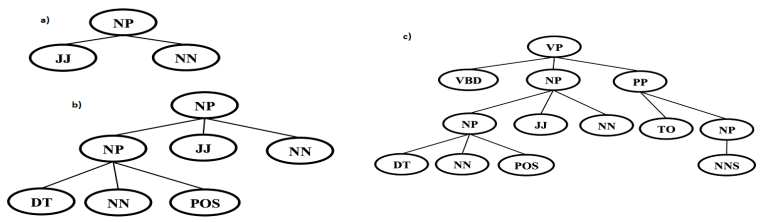


Figure 1: Height two, height three and height four sub-trees of a parse tree.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2766–2777
Santa Fe, New Mexico, USA, August 20-26, 2018.

possible features to describe an author's writings. Examples of sub-trees of various heights derived from a syntactic parse tree are shown in Figure 1. In addition to these we also include three other feature types, namely, character n-grams, function words, and Part-of-Speech (POS) tags' n-grams.

We evaluate the effectiveness of each individual feature type for author attribution and then show that combining the evidence from multiple feature types enhances the performance. We show that the consensus formation approach of Dempster-Shafer (DS) framework (Dempster, 1967; Shafer, 1976) performs better than other evidence combination methods. We have used the consensus formation rule of the Dempster-Shafer theory because the six feature types that we use capture very similar and overlapping underlying information about the syntax of the text. Each feature type also has some unique insights that are not provided by the other features. We cannot combine the six features as independent pieces of evidence, and therefore seek only the consensus among individual inferences based on them. Such consensus among sources of evidence is highlighted by the Dempster's rule of combination.

Using the example in Figure 2, we illustrate the intuitive differences between the DS consensus approach, voting, and aggregation (combination) approaches. Figure 2 shows three features (columns) and three authors 'a', 'b', and 'c' (rows). Given a test document the $mass\ values$ assigned to the three authors by each feature type are shown in their respective columns.

	Height two sub-trees	Height three sub-trees	Height four sub-trees	
NULL	0	0	0	
Author 'a'	0.34	0.3	0.3	Dempster rule
Author 'b'	0.2	0.27	0.5	Combined
Author 'c'	0.36	0.33	0.1	Voting
'a', 'b', 'c'	0.1	0.1	0.1	

Figure 2: An example of Dempster's rule.

The fourth row, including all the authors, captures the uncertainty of the evidence source and this much mass is not assigned to any individual author. When an author is selected based on voting, author 'c' is elected because two out of three feature types declare author 'c' as the winner. When we use a combination approach ($0.2 + 0.27 + 0.5 = 0.97$), author 'b' is selected. When Dempster's rule is used, author 'a' is selected because the extent to which all the different features "agree" on an author is highest for author 'a'.

2 Related work

Any framework for authorship attribution has two main tasks: 1) Feature selection, and 2) A classification technique to do authorship attribution using the selected features.

2.1 Feature Selection

The crux of the authorship attribution problem is the selection of a set of features that remain stable across a large number of writings created by an author. A large variety of features built from lexical, syntactic, semantic, content, and structural properties of the texts have been used as style markers in the past (Juola, 2006; Stamatatos, 2009).

The work published in Kjell et al. (1994) is the first to use the character bigrams and trigrams for authorship analysis of $Federalist$ papers. Later, the work in (Keelj et al., 2003; Houvardas and Stamatatos, 2006; Stamatatos, 2007; Stamatatos, 2012) used character n-grams of different sizes and reported an accuracy of up to 72% with 50 authors.

Syntactic features explain the grammatical structure of sentences and are considered to be reliable style markers as they are not under the conscious control of the author (Bayyen et al., 1996). There are many studies (Argamon and Levitan, 2005; Garcia and Martin, 2007; Kestemont, 2014) that showed the effectiveness of function words as features in authorship attribution tasks. Using rewrite rules as features - Gamon (2004) reported an accuracy of 86% (author set size 3), and Patchala et al. (2015) reported 74% accuracy (30 authors). The work in Eder (2013) studied the effectiveness of POS trigrams with a set of 21 authors and reported an accuracy of 65%. The problem with POS n-grams is that, they don't provide much information about how an author forms the phrases in sentences. The features generated with partial parsing (Luyckx and Dalemens, 2005) capture only the dependency between phrases but not how each phrase is formed. Our intuition is that considering various levels of sub-trees will capture dependency between phrases and also how a phrase is formed. The authors of Kim et al. (2011) identified

- For each author
  1. Combine all the articles in training text set for each author.
  2. Generate the character trigrams from raw text and compute their frequencies.
  3. Tokenize the text into sentences.
  4. Using a parser (Klein and Manning, 2003), parse the sentences and extract the POS trigrams, function words, and height two, three, four level sub-trees, and compute the frequencies for each feature type.
  5. Sort the features within each feature type by decreasing frequencies.
  6. Choose the top n discriminating features using the entropy metric for each feature.

Figure 3: Steps to generate the features

the most discriminating sub-trees in syntactic parse-trees and reported an accuracy of 84% (for 7 authors). Even though the sub-trees we derived from syntactic trees are in the same spirit as in Kim et al. (2011), we use them in building author templates, and combine the information of these features with other syntactic feature types in a completely different and novel way.

2.2 Attribution Techniques

There are two types of classifiers used in authorship attribution: 1) Template-matching classifiers, and 2) Machine-learning based classifiers (such as SVMs).

In template-matching approaches, all the training documents by an author are merged to form a single text collection and is used to generate a feature template for each author's writing style. A test document's signature is then compared with each author's template and authorship is assigned to the author with the closest match. Using character n-grams as features and entropy as similarity measure, the work in Peng et al. (2003) has reported an accuracy of 90% (for 8 authors). Instead of using a single template for each author, the work in (Jankowska et al., 2013; Layton, 2014; Koppel et al., 2011) created multiple templates for each author by selecting subsets of features and the result from each template is combined to find the best match using voting. The work in Koppel et al. (2011) has reported precision and recall values of 90% and 43% for 1000 authors and proved that this template based approach works well even in the case of large author sets.

In machine learning based approaches each document is treated as a representative data point of an author's style in a feature space and a classifier is trained. The work in Abbasi and Chen (2005) uses a combination of lexical, structural and syntactic features, employs Decision Trees as model, and reports an accuracy of 90% (for 20 authors). A number of studies in authorship analysis (Gamon, 2004; Zheng et al., 2006; Stamatatos, 2007; Luyckx and Daelemans, 2008; Luyckx, 2011; Silva et al., 2011) have used various Support Vector Machine (SVM) based classifiers. The work in Luyckx and Daelemans (2008) uses SVM and POS n-grams to assign authorship of essays written by 145 students and reported an accuracy of 55%. Work in (Stamatatos, 2007; Stamatatos, 2012) reported an accuracy of 75% (50 authors) by using character n-grams and SVM.

3 Methodology

We address the authorship attribution problem using the template-based methodology. The main reasons for this choice include stability in the face of changing sets of authors, ability to handle semi-closed authorship problems, and ability to more effectively combine different types of syntactic feature sets.

Each syntactic feature type captures a different mix of properties of the text in a document. This necessitates inclusion of a number of different feature types to design a robust classification system. We consider six types of features: character trigrams, function words, POS trigrams, and sub-trees of height two, three and four.

The steps to generate author's template are summarized in Figure 3. Instead of including thousands of feature values into an author template, we find the most frequent and then within them the most

informative feature values of each feature type. That is, we may include some character trigrams by first selecting the most frequent ones and then within those the most informative ones as measured by their informational entropy across all the authors. Typically 800 to 1500 of each feature's values and their frequencies are included in a template. The most frequent feature items used by an author are likely to persist in their higher relative frequencies across their multiple writings. One such template for each of the six feature types is created for each author. Let us say the resulting six templates for an author are: T_1, T_2, T_3, T_4, T_5, and T_6. Each template contains the relative frequencies of occurrence of individual feature items of a feature type for an author. For example, a template may contain relative frequencies of selected 1000 character trigrams, or 1000 POS trigrams, or 1000 height two sub-trees of parse trees.

Given a test document, six signatures referred to as: S_1, S_2, S_3, S_4, S_5, and S_6, which are similar in structure to the six templates generated for authors are generated. We use the Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951) to compute the dissimilarity values between a test document's six signatures and the corresponding six templates for an author. This gives us six different divergence values for each document-author pair.

We now consider the decision rules for assigning an author to a test document. We examine the cases of making a decision for: (i) the closed author set using only one feature type (ii) the semi-closed author set using only one feature type; (iii) the closed author set using all the six feature types; and (iv) the semi-closed author set using all the six feature types. Our decision making strategies for these four cases are as follows.

Case-1: Compute the divergence value between the test document's selected feature signature, S_m, and each author's template for the m^{th} feature type, T_m. Let us say, D(S_m , T_m(i)) is the divergence value for the m^{th} feature type for the i^{th} author's template. We find the k^{th} author for whom this divergence value is the smallest and assign him/her the authorship.

Case-2: We use the intuitive idea that if the k^{th} author is the true author then the value D(S_m , T_m(k)) must be significantly smaller than the average of D(S_m , T_m(i)) values for all the authors in the pool. We capture this notion by requiring that the k^{th} author is assigned to a document only when the z-score of D(S_m , T_m(k)) is -2.0 or smaller when compared to the values D(S_m , T_m(i)), for all authors else we announce that the true author is not in the candidate set.

Case-3: In this case we seek to combine information obtained from all six feature types and obtain consensus among the inferences arrived by each of them individually. We use Dempster's rule from Theory of Evidence to combine the evidences. Let us say there are k authors in the pool of known authors and for the m^{th} feature type we compute the k divergence values as Div(m,i) = D(S_m , T_m(i)) where i takes the value from 1 to k. Now such a divergence vector is generated for each of the six feature types.

To apply the Dempster's Rule we need to convert these divergence values for authors (for a fixed feature type) to a $mass\ assignment$. As part of this conversion process we have to group all those authors who have very similar divergence values for a feature type into a single group. Given a vector of k divergence values (one feature type), we first normalize the divergence values by scaling them as follows:

$$NormDiv(i,j) = \frac{(max_j(Div(i,j)) - Div(i,j))}{(max_j(Div(i,j)) - (min_j(Div(i,j)))}. \quad (1)$$

Here, i represents the author number, j represents the feature type and $Div(i,j)$ denotes the divergence value of i^{th} author for j^{th} feature type. These scaled values are such that the author with the template closest to the document signature gets a value of 1 and the author with the template farthest from the document gets a value of 0. If two or more authors have very similar scaled scores (say 1.0 and 0.98) then they are equally likely to be the authors of the document (as per the feature type under consideration).

Therefore, as the next step of this process we form groups of authors from the perspective of this feature type. For each author in the candidate set, we identify the authors whose scaled scores are within the threshold δ and include all these authors to form an author set. The score of the author group is set equal to the average of the scaled values of the authors contained in the group. We then remove the duplicate author groups, if any are formed, and normalize all the group scores (for the author sets,

for each feature type) so that all the scores (for each feature type) add up to 1.0 as per the DS theory requirement of a mass assignment. We call these scores as mass values. Now, we have generated one mass assignment function for each feature type.

For example, say, for eight authors (a, b, c, d, e, f, g, h) the scaled values are: (1, 0.97, 0.94, 0.92, 0.87, 0.81, 0.73, 0). The mass assignment for these eight authors is shown in Table 1.

Author groups	Scaled scores	Average	mass values
<a,b >	<1, 0.97 >	0.985	0.157
<a, b, c, d >	<1, 0.97, 0.94, 0.92 >	0.957	0.153
<b, c, d >	<0.97, 0.94, 0.92 >	0.943	0.151
<b, c, d, e >	<0.97, 0.94, 0.92, 0.87 >	0.925	0.148
<d, e >	<0.92, 0.87 >	0.895	0.143
<f >	<0.81 >	0.81	0.129
<g >	<0.73 >	0.73	0.116
<h >	<0 >	0	0

Table 1: Mass assignment process using eight authors and δ value 0.05

We repeat this process for the divergence vector of each feature type and generate mass assignments for groups of authors. Then, the mass assignments for each feature type are combined using Dempster's rule of combination. Given two mass assignments m_1, m_2, the combined mass assignment m_{12} is computed as follows:

$$m_{12}(A) = \frac{1}{1-K} \sum_{B \cap C = A \neq \emptyset} m_1(B) m_2(C), \quad K = \sum_{B \cap C = \emptyset} m_1(B) m_2(C) \tag{2}$$

Here, K is the measure of the amount of conflict between two mass assignments and A, B, C represent subsets of authors.

After computing the consensus mass assignment from six feature types we compute the *plausibility* values of each candidate author being the true author of the test document.

The plausibility pl(A) is the sum of all the masses of sets 'B' that intersect the set of interest-'A' and is defined as follows:

$$pl(A) = \sum_{B | B \cap A \neq \emptyset} m(B) \tag{3}$$

We then select the author with the highest plausibility value as the true author of the test document.

Case-4: For this situation we follow the same process as in case-3 and then determine if the author with the highest plausibility has a plausibility value greater than 0.5, and its z-score among plausibility values for all authors is larger than 2.0. If not, we say that the true author is not in the given author set.

4 Dataset Description

News Articles' Dataset: We selected 7 columnists from the New York Times newspaper and 3 from the Guardian newspaper and extracted 50 articles for each author. To make each author's writings topic independent we selected the topics of sports, education, immigration, elections, health and politics and made sure that each author has written articles on at least 4 of these topics in our collection. These articles consists of, on an average, 800 words per document.

Reuter_50_50 Dataset: *Reuter_50_50* dataset (Houvardas and Stamatatos, 2006; Stamatatos, 2007; Bache and Lichman, 2013) has 50 authors and 5,000 documents. The training corpus has 2,500 documents and test corpus has 2,500 documents (50 documents for each author in train and test) with an average of 500 words per document.

Blogs Dataset: We used the *blogs* dataset (Schler et al., 2006; Koppel et al., 2011) and selected the top 100 authors based on the number of posts. We selected 10 posts for each author as test documents and rest as training documents. Each posting has around 200 words on average.

5 Results and Analysis

We have used the traditional metrics - accuracy, precision, and recall for measuring the performance of our author attribution study.

We have used the Naïve Bayes, Support Vector Machine (SVM), Voting, and the method proposed in Koppel et al. (2011) which we call as 'Koppel 2011' to compare the performance of proposed features types and methodology. Matlab's Naïve Bayes classifier with multinomial distribution and LIBSVM one-vs-one classifier with RBF kernel are used for Naïve Bayes and SVM based classification, and parameters are optimized in each case.

In case of attributing an author based on all six feature types, the method labeled 'Combined (C)' is performed by concatenating all the six feature templates of an author into one large template and then making the decisions using the divergence of similarly concatenated test signatures. The voting method labeled 'Voting' is executed by assigning the authorship to the author selected by maximum number of feature types. The method with label 'Koppel 2011' uses the approach discussed in Koppel et al. (2011). The purpose of these methods is to contrast their result with proposed Dempster's combination rule based method (DS combined).

5.1 Closed author set using only one feature type

The main reason to assign authorship considering each feature type (section 3: case-1) separately is to compare and contrast their individual performances. For each feature type, we kept the number of features - 'n' to be constant at 1,500 and reported the accuracies in Figure 4. This figure shows the accuracy of author attribution using four different classifiers: our proposed method KL-Divergence, Naïve Bayes, SVM, and Koppel 2011.

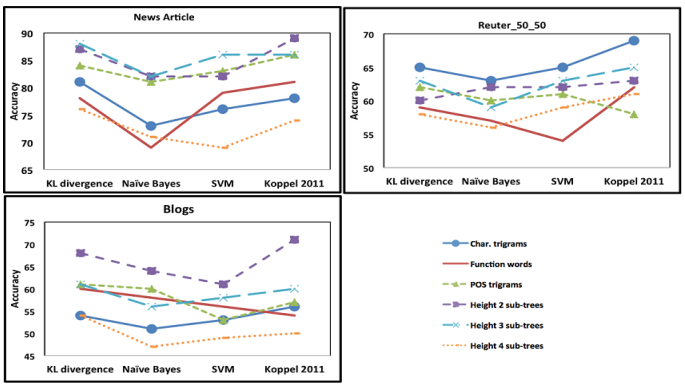


Figure 4: Accuracy values for closed-author set using one feature type.

Here, we can see that the frequencies of sub-trees of height two, and three, and POS trigrams - yielded greater accuracy in *news articles* and *blogs* datasets irrespective of the classification algorithm used. In *Reuter_50_50* dataset, character trigrams gave a slightly better accuracy compared to other features. A common intuition with character n-grams is that they capture the thematic information along with the author style (Stamatatos, 2012). Many authors in *Reuter_50_50* dataset have written articles on a particular subtopic with a high occurrence of a set of words specific to the topic. This makes it easy for character trigrams to distinguish different authors. Parse trees frequencies work better for the News Articles dataset because each author has written articles on different topics and the feature is topic independent.

5.1.1 Effect of feature set size

For this analysis (Figure 5), we kept the training and test document sets fixed, and gradually increased the number of best feature items (number of best character 3-grams, or height 2 sub-trees etc.) included in each template. Most of the feature types attained maximum accuracy when 1000 to 2000 feature items are included and are stable in performance when few hundred feature items are added.

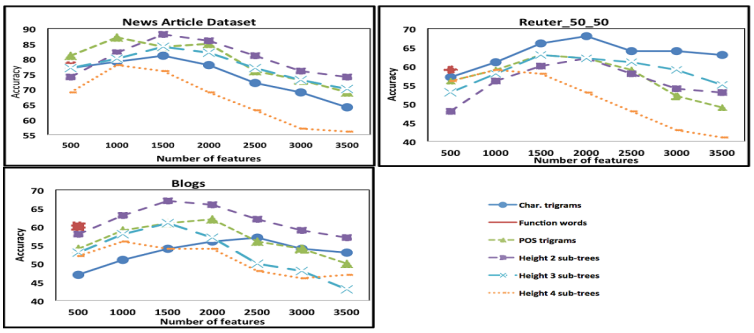


Figure 5: Effect of feature set size.

Afterward, the performance decreased as more feature items are added. The results presented in (Stamatatos, 2007; Stamatatos, 2012) suggest considering at least a feature set size of 3000 - 5000 for character trigrams in *Reuter_50_50* dataset. In their formulation they include the 5000 most frequent character trigrams in the feature set whereas in our formulation we select the top '*n*' discriminating features based on informational entropy. From Figure 5, we can observe that there is no overall consensus on ideal size for feature sets. However, one can achieve a reasonable accuracy by considering the most discriminating features numbering between 1000 and 2000. This is a very wide range. For larger numbers it seems the curse of dimensionality sets in and the extra feature items, that are less discriminating, add noise to the author attribution process. Fewer feature items have lower accuracy because they may not have enough information in them to make best decisions.

5.1.2 Effect of training data size

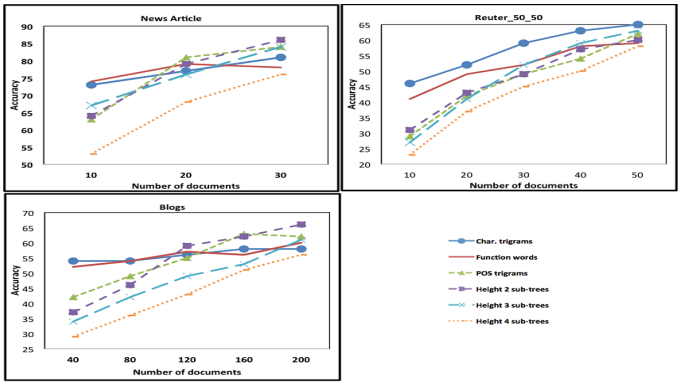


Figure 6: Effect of training data size.

To analyze the effect of limited training data, we started with ten documents and increased the number of documents in multiples of ten for *news article* and *Reuter_50_50* dataset. In *blogs* dataset, we started with 40 documents and increased the size in multiples of 40. Each document in *news article* dataset is approximately twice the size of the document in *Reuter_50_50* and *blogs* dataset. The test set is kept constant for each author and the attribution accuracy results are shown in Figure 6. There is a large difference in accuracy between the character trigrams and other syntactic features especially with limited training data and this difference narrows once we start increasing the number of training documents. The function words and character 3-grams templates have the best performance when the training dataset is very small. So, these features are very effective even with very small training set sizes. We also notice that, expectedly, as the number of training documents increases the performance for each feature type improves consistently. At the higher end of the number of training documents, subtrees of height two became the best feature types for the News Articles and the Blogs datasets and kept improving their performance even for the Reuters dataset.

5.2 Semi-closed author set using only one feature type

Here, we use the approach described in Case-2 of the methodology section (Section 3). In addition to the test documents in the *news articles* dataset, we added 200 randomly selected documents from the *Reuter_50_50* dataset to the *news article* test set. Similarly, we added 200 test documents from *news article* dataset to *Reuter_50_50* dataset. For *blogs* dataset, we randomly selected 500 documents written by the authors who are not in the selected sample dataset and added them to the test set. The feature set size '*n*' is kept constant at 1500 items and z-score threshold for making a decision is selected as -1.5 for all the datasets.

In Table 2, we show the precision and recall values, measured individually for each feature type. We can observe the similar behavior as in the closed author set case. That is, grammar based features gave better performance in *news article* and *blogs* datasets and character trigrams gave higher performance in *Reuter_50_50* dataset. We observe that as the number of authors increases the precision and recall values decline. They are the highest for the News articles dataset (10 authors), followed by Reuters (50 authors) and then the lowest for the Blogs dataset (100 authors). Also, the recall is consistently smaller than the precision because for the semi-closed case an author is announced only if he/she stands significantly apart compared to other authors.

		Character trigrams	Function words	POS trigrams	Height 2 sub-trees	Height 3 sub-trees	Height 4 sub-trees
News article	prec.	69	72	71	77	74	59
	rec.	51	47	56	54	58	43
Reuter_50_50	prec.	64	66	57	54	57	51
	rec.	56	53	49	53	51	49
Blogs	prec.	41	43	46	47	46	44
	rec.	36	41	43	41	41	37

Table 2: Precision and Recall for semi-closed authorship using individual features (Case-2)

		Combined (C)	Voting	Koppel 2011	SVM	DS combined(3)	DS combined(6)
News Article	prec.	77±4.4	69±2.9	80±3.4	76±3.8	78±4.5	88±3.0
	rec.	73±5.1	71±2.7	83±2.9	72±4.4	73±3.2	82±2.0
Reuter_50_50	prec.	56±3.9	64±3.2	64±4.2	63±3.5	67±4.0	73±3.7
	rec.	61±4.8	59±3.6	62±2.6	64±3.1	62±4.6	67±5.0
Blogs	prec.	58±4.1	54±3.8	62±3.9	57±3.6	57±3.4	68±2.4
	rec.	61±3.2	52±3.4	63±3.4	54±3.9	56±2.2	65±1.1

Table 3: Precision and Recall values using all feature types (Case-3)

5.3 Closed author set using all the six feature types

We consider feature item set sizes between 500 - 3000, in multiples of 500, and the average values for precision and recall along with their standard deviations are show in Table 3. Here, the threshold δ for forming the groups of authors for the Dempster's rule combination method, is kept constant at 0.05. We observed that for δ values between 0.025 and 0.075 there is negligible change in performance in all the datasets. To show that the sub-tree features derived from the parse trees capture some additional author discriminating information that is not captured by the char. trigrams, func. words, and POS trigrams, we performed the following experiment. First we used the three feature types (func. words, char. trigrams, POS trigrams) and calculated the precision and recall values using our DS combined approach (*DS combined(3)*). Then we considered all six feature types, and calculated the precision and recall values (*DS combined(6)*). There is a significant increase in precision and recall values when we include the sub-tree features for consensus formation compared to only combining the three non-tree feature types. This validates our hypothesis that the sub-tree feature types capture some additional information that is not captured by the other three feature types. We also can see that the performance improves significantly when inferences from multiple features types are combined using various aggregation methods; and the best performance for all datasets is obtained when Dempster's rule is used.

Robustness Analysis:

The main aim of our test described in this section is to analyze how our *DS combined* method performs when small changes are introduced in the test documents. For each test document in an author's test set we

		Original	Adding 50 words	Adding 100 words
News Article	prec.	88±3.0	86±4.3	83±3.8
	rec.	82±2.1	80±3.9	77±5.7
Reuter_50_50	prec.	73±3.7	71±2.8	65±2.7
	rec.	66±5.0	64±3.7	60±3.6
Blogs	prec.	68±2.4	61±2.1	54±2.4
	rec.	65±1.1	58±2.4	51±2.3

Table 4: Precision and Recall values with slight changes to test documents

added 50 random words of text taken from a different author within the same dataset. We repeated this in all the three datasets and used feature set sizes between 500 - 3000, in multiples of 500. The results obtained are shown in Table 4. The δ value is kept constant at 0.05 for all the datasets. The *Blogs* dataset is very sensitive to changes in test documents as the test document size is very small (200 words). There is very small drop in precision and recall values in other datasets when we added 50 random words to a test document. However, when we added 100 random words to each test document, performance decreased more in *Reuter_50_50* dataset but not that much in *news article* dataset. This is because the average test document size is small (500 words) in the former dataset and the average size in *news article* dataset is 800 words.

To study the effect of candidate author set size on *DS combined* approach, we kept the feature set size constant at 1500 items and analyzed the performance by gradually increasing the author set size. For each author set size, we repeated the experiment ten times by randomly selecting the authors and the average precision and recall values are reported in Figure 7. As expected, both the precision and recall values slowly decrease with an increase in the author set size.

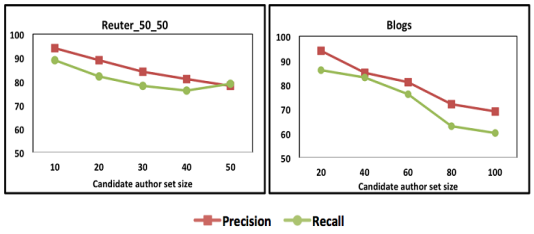


Figure 7: Precision and recall values for different candidate author set sizes.

5.4 Semi-closed author set using all the six feature types

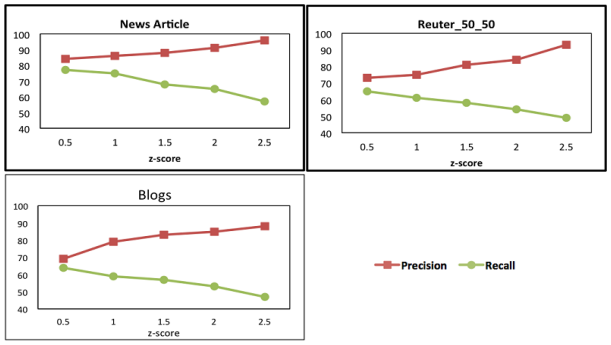


Figure 8: Precision and recall values for different z-score cutoff values.

For this test, we have used the same datasets as described in Section 5.2 and used the approach described in case-4 of the methodology section. The precision and recall values for various z-score cut off values are shown in Figure 8. In all datasets the precision values do not decline, but rise slightly, as the z-score cutoff goes up to 2.5. This is because a larger z-score cutoff means that the winning author must have a significantly higher plausibility value compared to all the other authors. Therefore, an author is announced for a document only if he/she is the clear winner compared to all the others. The recall values reduce somewhat with increasing z-score cutoff and this is because the authors who do not stand out very strongly compared to the other authors do not get attributed to the documents.

6 Conclusion

In the above discussion we have shown two main contributions of our proposed methodology for author attribution of documents. The first is that templates formed from the sub-tree frequencies in the parse tree of an author's text provide very valuable insights about one's writing style. The second main contribution is the demonstration that information embedded in different types of syntactic features is best combined using Dempster's rule compared to some other methods of information fusion. We have also shown successfully that our proposed approach works well for both - 'closed' and 'semi-closed' cases. We have shown the robustness of our proposed approach from various perspectives and the best results obtained using Dempster's Rule based combination of all six features are of significantly high quality.

References

Abbasi, A., and Chen, H. 2005. Applying Authorship Analysis to Extremist-Group Web Forum Messages. *IEEE Intelligent Systems* 20(5): 67-75.

Argamon, S., Koppel, M., and Avneri, G. 1998, Routing documents according to style, *First International workshop on innovative information systems*, pp. 85-92.

Argamon, S., Levitan, S. 2005, Measuring the usefulness of function words for authorship attribution. *In Proceedings of the 2005 ACH/ALLC Conference*.

Argamon, S., Whitelaw, C., Chase, P., Raj, H. S., Garg, N., and Levitan, S. 2007. Stylistic text classification using functional lexical features: Research Articles. *Journal of American Society for Information Science and Technology*, 58(6): 802-822.

Axelsson, M. W., 2000. USE-the Uppsala Student English corpus: an instrument for needs analysis. *ICAME journal* 24:155-157.

Bache, K. and Lichman, M. 2013. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Baayen, H., Van Halteren, H., and Tweedie, F. (1996). Outside the cave of shadows: Using syntactic annotation to enhance authorship attribution. *Literary and Linguistic Computing*, 11(3): 121-132.

Bayyen, R. H., Halteren, H. V., Neijt, A., and Tweedie, F. J. 2002. An experiment in authorship attribution. *In Proceedings of the 6th International Conference on Statistical Analysis of Textual Data*, pp. 29-37.

Chaski, C.E. 2005, Who's At The Keyboard? Authorship Attribution in Digital Evidence Investigations. *International Journal of Digital Evidence*, 4(1): 1-13.

Dempster, A. P. 1967. Upper and lower probabilities induced by a multivalued mapping. *The Annals of Mathematical Statistics*, 38(2): 325-339.

Eder, M. 2013 Does size matter? Authorship attribution, small samples, big problem. *Literary and Linguistic Computing*: fqt066.

Galitsky, B. 2013. Machine learning of syntactic parse trees for search and classification of text. *Engineering Applications of Artificial Intelligence*, 26(3): 1072-1091.

Gamon, M. 2004. Linguistic correlates of style: authorship classification with deep linguistic analysis features. *In Proceedings of the 20th international conference on Computational Linguistics (COLING '04). Association for Computational Linguistics*, Stroudsburg, PA, USA, Article 611.

Garca, A. M., and Martin, J. C. (2007). Function words in authorship attribution studies. *Literary and Linguistic Computing*, 22(1), 49-66.

Houvardas, J., and Stamatatos, E. 2006. N-gram feature selection for authorship identification. *In Artificial Intelligence: Methodology, Systems, and Applications*. Springer Berlin Heidelberg, pp. 77-86.

Jankowska, M., Keelj, V., and Milios, E. 2013. Ensembles of Proximity-Based One Class Classifiers for Author Verification ? Notebook for PAN at CLEF 2014. *In Cappellato, L., Ferro, N., Halvey, M., and Kraaij, W. (eds.). CLEF 2014 Labs and Workshops, Notebook Papers. CEUR Workshop Proceedings (CEUR-WS.org)*,ISSN 1613-0073.

Juola, P. 2006. Authorship attribution. *Found. Trends Inf. Retr*, 1(3): 233-334.

Keelj, V., Peng, F., Cercone, N., and Thomas, C. 2003. N-gram-based author profiles for authorship attribution. *In Proceedings of the conference pacific association for computational linguistics*, PACLING, Vol. 3, pp. 255-264.

Kestemont, M. 2014. Function Words in Authorship Attribution From Black Magic to Theory?. *EACL 2014*, pp. 59-66.

Kim, S., Kim, H., Weninger, T., Han, J., and Kim, H. D. 2011. Authorship classification: a discriminative syntactic tree mining approach. *In Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. pp. 455-464.

Kjell, B., Woods, W. A., and Frieder, O. 1994. Discrimination of authorship using visualization. *Information processing and management*, 30(1), 141-150.

Klein, D., and Manning, D. C. 2003. Accurate Unlexicalized Parsing. *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pp. 423-430.

Koppel, M., and Schler, J. 2003. Exploiting stylistic idiosyncrasies for authorship attribution. *In Proceedings of IJCAI'03 Workshop on Computational Approaches to Style Analysis and Synthesis*, pp. 69-72.

Koppel, M., Schler, J., and Zigdon, K. 2005. Automatically determining an anonymous author's native language. *In Proceedings of the 2005 IEEE international conference on Intelligence and Security Informatics* (ISI'05), Paul Kantor, Gheorghe Muresan, Fred Roberts, Daniel D. Zeng, and Fei-Yue Wang (Eds.). Springer-Verlag, Berlin, Heidelberg, pp. 209-217.

Koppel, M., Akiva, N., and Dagan, I. 2006. Feature instability as a criterion for selecting potential style markers: Special Topic Section on Computational Analysis of Style. *Journal of American Society for Information Science and Technology*, 57(11):1519-1525.

Koppel, M., Schler, J., Argamon, S. 2011. Authorship attribution in the wild. *Language Resources and Evaluation*, 45(1): 83-94.

Koppel, M., and Yaron, W. 2014. Determining if two documents are written by the same author. *Journal of the Association for Information Science and Technology*, 65(1): 178-187.

KULLBACK, S. and LEIBLER, R.A. 1951. On Information and Sufficiency, *Ann. Math. Statist.*, 22: 79-86.

Layton, R. 2014. A Simple Local n-gram Ensemble for Authorship Verification ? Notebook for PAN at CLEF 2014. *In Cappellato, L., Ferro, N., Halvey, M., and Kraaij, W. (eds.). CLEF 2014 Labs and Workshops, Notebook Papers. CEUR Workshop Proceedings (CEUR-WS.org)*,ISSN 1613-0073.

Luyckx, K., and Daelemans, W. 2005. Shallow text analysis and machine learning for authorship attribution. *In Proceedings of the 15th meeting of Computational Linguistics in the Netherlands*, pp. 149-160. Utrecht, Netherlands: LOT.

Luyckx, K., and Daelemans, W. 2008. Authorship attribution and verification with many authors and limited data. *In Proceedings of the 22nd International Conference on Computational Linguistics* -Volume 1, pp. 513-520. Association for Computational Linguistics.

Luyckx, K. 2011. Scalability issues in authorship attribution. *ASP/VUBPRESS/UPA*.

McCarthy, P.M., Lewis, G.A., Dufty, D.F., and McNamara, D.S. 2006. Analyzing writing styles with coh-metrix. *In Proceedings of the Florida Artificial Intelligence Research Society International Conference*, pp. 764-769.

Mosteller, F., and Wallace, D. L. 1963. Inference in an authorship problem: A comparative study of discrimination methods applied to the authorship of the disputed Federalist Papers. *Journal of the American Statistical Association*, 58(302), 275-309.

Nektaria, P., and Stamatatos, E. 2014. A Profile-Based Method for Authorship Verification. *Artificial Intelligence: Methods and Applications*. Springer International Publishing, pp. 313-326.

Patchala, J., Bhatnagar, R., and Gopalakrishnan, S. 2015. Author Attribution of Email Messages Using Parse-Tree Features. *In Machine Learning and Data Mining in Pattern Recognition*. Springer International Publishing, pp. 313-327.

Peng, F., Schuurmans, D., Wang, S., and Keselj, V. 2003. Language independent authorship attribution using character level language models. *In Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume 1* (EACL '03), Vol. 1. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 267-274.

Raghavan, S., Kovashka, A., and Mooney, R. 2010. Authorship attribution using probabilistic context-free grammars. *In Proceedings of the ACL 2010 Conference Short Papers (ACLShort '10)*. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 38-42.

Schler, J., Koppel, M., Argamon, S., and Pennebaker, J. 2006. Effects of age and gender on blogging. *In AAAI Spring Symposium*, 06(03): 191-197.

Shafer, G. 1976. *A Mathematical Theory of Evidence*, Princeton: Princeton University Press.

Shane B., Matt, P., and David, Y. 2012. Stylometric analysis of scientific articles, *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics*: Human Language Technologies, June 03-08, Montreal, Canada.

Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A., and Chanona-Hernndez, L. 2014. Syntactic N-grams as machine learning features for natural language processing. *Expert Systems with Applications*, 41(3): 853-860.

Silva, R. S., Laboreiro, G., Sarmento, L., Grant, T., Oliveira, E., and Maia, B. 2011. twazn me!!! ;(automatic authorship analysis of micro-blogging messages. *In Proceedings of the 16th international conference on Natural language processing and information systems*, Springer-Verlag, Berlin, Heidelberg, pp. 161-168.

Stamatatos, E. 2007. Author Identification Using Imbalanced and Limited Training Texts, *In Proc. of the 4th International Workshop on Text-based Information Retrieval*, September 3-7; Regensburg, Germany.

Stamatatos, E. 2009. A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology*, 60(3): 538-556.

Stamatatos, E. 2012. On the robustness of authorship attribution based on character n-gram features. *Journal of Law and Policy*, 21: 421.

Stamatatos, E., Daelemans, W., B., Verhoeven, B., Stein, B., Potthast, M., Juola, P., Sanchez-Perez, M. A., and Barrn?Cedeo, A. 2014. Overview of the Author Identification Task at PAN 2014.*Working Notes for CLEF 2014 Conference*, pp. 877-897.

Tan, R. H. R., and Tsai, F. S. 2013. Authorship Identification for Online Text, *In 2013 International Conference on Cyberworlds*, pp.155-162.

Zečević, A. 2011. N-gram Based Text Classification According To Authorship. *In Student Research Workshop*, pp. 145-149.

Zhao, Y., and Zobel, J. 2005. Effective and scalable authorship attribution using function words. *In Information Retrieval Technology*, pp. 174-189, Springer Berlin Heidelberg.

Zhao, Y. and Zobel, J. 2007. Searching with style: authorship attribution in classic literature. *In Proceedings of the thirtieth Australasian conference on Computer science* (ACSC '07), Gillian Dobbie (Ed.). Australian Computer Society, Inc., Darlinghurst, Australia, 62: 59-68.

Zheng, R., Li, J., Chen, H., and Huang, Z. 2006. A framework for authorship identification of online messages: Writing-style features and classification techniques. *Journal of American Society for Information Science and Technology*, 57(3): 378-393.

Modeling with Recurrent Neural Networks for Open Vocabulary Slots

Jun-Seong Kim
Samsung Electronics Co., Ltd /
Suwon, South Korea
Js087.kim@samsung.com

Junghoe Kim, SeongUn Park,
Kwangyong Lee, Yoonju Lee
Samsung Electronics Co., Ltd /
Suwon, South Korea
{kjh94, seongun.park,
ky4you.lee,
yo0nju.lee}@samsung.com

Abstract

Dealing with 'open-vocabulary' slots has been among the challenges in the natural language area. While recent studies on attention-based recurrent neural network (RNN) models have performed well in completing several language related tasks such as spoken language understanding and dialogue systems, there has been a lack of attempts to address filling slots that take on values from a virtually unlimited set. In this paper, we propose a new RNN model that can capture the vital concept: Understanding the role of a word may vary according to how long a reader focuses on a particular part of a sentence. The proposed model utilizes a long-term aware attention structure, positional encoding primarily considering the relative distance between words, and multi-task learning of a character-based language model and an intent detection model. We show that the model outperforms the existing RNN models with respect to discovering 'open-vocabulary' slots without any external information, such as a named entity database or knowledge base. In particular, we confirm that it performs better with a greater number of slots in a dataset, including unknown words, by evaluating the models on a dataset of several domains. In addition, the proposed model also demonstrates superior performance with regard to intent detection.

1 Introduction

In spoken dialogue systems and goal-oriented dialogue systems, slot filling and intent detection are primarily involved in a process of understanding user utterances based on pre-designed semantic frames. Slot filling is to identify a sequence of tokens and extract semantic constituents from the utterances, and intent detection is to classify the intent of the utterances. Two examples relevant to these problems are shown in Figure 1 for the domains of flight and message.

In recent years, a significant amount of interest has built up around slot filling and intent detection due in part to the competition among commercial artificial intelligence assistants, such as Samsung Bixby, Apple Siri, Google Assistant, Microsoft Cortana, and Amazon Alexa. As a result, a multitude of studies regarding recurrent neural network (RNN) models for slot filling and intent detection has been completed and has resulted in outcomes that generally outperform most other past approaches. In particular, there have been various attempts to improve the performance of RNNs for spoken dialogue systems such as label sampling (Liu and Lane, 2015), the hybrid type of Elman and Jordan (Mesnil et al., 2015), Deep LSTM (Yao et al., 2014), external memory (Peng et al., 2015), bidirectional LSTM (Hakkani-Tur et al., 2016), encoder-labeler and Deep LSTM (Kurata et al., 2016), attention (Liu and Lane, 2016), bidirectional LSTM and CRF (Huang et al., 2015), word hashing (Ravuri and Stolcke, 2015a; Ravuri and Stolcke, 2015b) CNN and bidirectional LSTM (Chiu and Nichols, 2015), external word embeddings (Kim et al., 2016), multi-task learning of a word-based language model (Shi et al., 2015), and multi-task learning of intent and domains (Shi et al., 2015; Hakkani-Tur et al., 2016; Bapna et al., 2017).

Proceedings of the 27th International Conference on Computational Linguistics, pages 2778–2790
Santa Fe, New Mexico, USA, August 20-26, 2018.

Sentence	find	flights	to	new	york	tomorrow
Slots	O	O	O	B-toloc	I-toloc	B-date
Intent	find_flight					

Sentence	text	to	mary	that	sad	puppy	Noise
Slots	O	O	B-recipient	O	B-text	I-text	I-text
Intent	send_message						

Figure 1: Two examples of a flight domain and a message domain with intent and slot annotation, following the IOB (in-out-begin) slot representation

Although most previous studies that exploited RNN and attention methods (Liu and Lane, 2016; Vaswani et al., 2017) were able to achieve state-of-the-art performance across a wide range of natural language research areas, they failed to deal with 'open-vocabulary' slots, which is one of the most important problems in the context of end-user experience using voice assistant. 'Open-vocabulary' slots signify a slot type that can take on values from a virtually unlimited set, such as file name, album name, text body, or schedule title. Therefore, words that are not seen in the training set or are employed differently from the meaning inherent in them can be included in sentences because of the fact that they have no constraints on the length and specific patterns of content. For slot filling of 'open-vocabulary' slots with such characteristics, the goal of this study was to capture the vital concept in which the meaning or role of words could vary according to how long a reader focuses on a particular part of a sentence. For example, in the utterance "Send how about dinner tonight to him" in the message domain, if a reader focuses on 'how about dinner tonight,' the word 'tonight' signifies time. However, it should rather focus on the entire utterance that starts with "Send how …" and recognize the word 'tonight' as a part of the text body. Thus, it is necessary to understand the holistic semantics at the level of the whole sentence, not only the vicinity of a word.

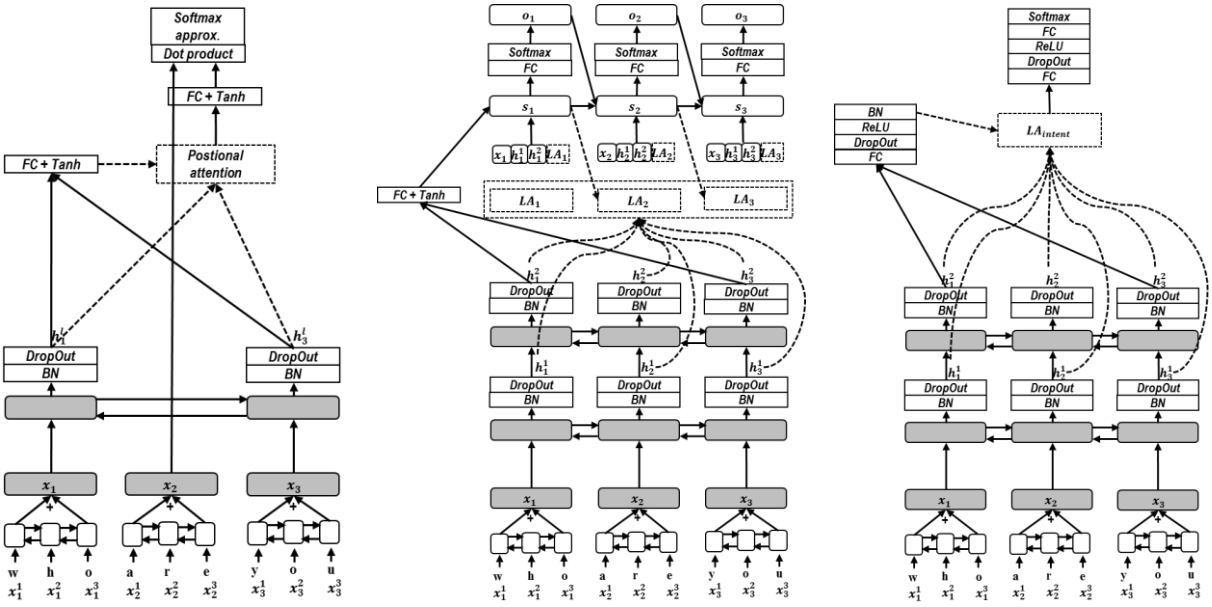
Our model embodies the above concept by introducing a long-term aware attention structure and positional encoding primarily intended for the relative distance between words. Long-term aware attention structure utilizes and stacks two layers of an RNN and treats them as a pair of short-term and long-term meanings. The lower layer denotes short-term or relatively local information corresponding to the vicinity of a word, and the upper layer denotes long-term or relatively global information corresponding to the whole sentence. A novel positional encoding, called the light-house encoding, considers the relative distance between current word and the word being compared in relation to the current word. In addition, it is completely based on characters rather than words, which is adequate for unknown words of 'open-vocabulary' slots. It is also trained with multi-task learning of a character-based language model and an intent detection model to improve the stability and utility of character-based representation and to obtain additional sentence-level information. The language model serves as a regularizer, and the intent detection model is beneficial in the presence of 'open-vocabulary' slots because it allows the proposed model to refer to the global information of a sentence. Lastly, to focus on the effectiveness of modeling, this study only uses the lexical input of language without the help of any external information, such as a named-entity or knowledge base.

We evaluated our model on slot filling and intent detection with the ATIS corpus (Hemphill et al., 1990) and five in-house domains. For the ATIS corpus, the proposed model achieved a state-of-the-art result on intent detection and a result comparable to the state-of-the-art result on slot filling. The proposed model also outperformed the existing RNN models for both tasks on the dataset of five domains. In particular, we demonstrate that our model has outstanding performance on 'open-vocabulary' slots and its efficacy is increased as more data in a domain contain 'open-vocabulary' slots including unknown words.

The main contribution of this study is a new RNN-based model that captures global information of words for discovering 'open-vocabulary' slots. To achieve this, we propose a long-term aware attention structure and a novel positional encoding with help from character-based modeling and multi-task learning.

2 Slot Filling and Intent Detection

Slot filling is a sequence labeling problem and intent detection can be treated as a classification problem that has multiple output labels. In the example in Figure 1, the flight domain may contain *find_flight*, *find_airfare*, and *find_distance* as intent list and *toloc*, *fromloc*, *date*, and *flight_number* as



(a) (b) (c)

Figure 2. Structures of our proposed model (a) Structure of our language model (b) Structure of our slot filling model (c) Structure of our intent detection model

the slot list. The task of slot filling is to find an explicit alignment between slots and a given user query, and the task of intent detection is to instantiate one or more intents from the intent list for a given user query.

More formally, we want to learn two functions for the $f_{\text{slot}} : \mathrm{X} \rightarrow \Upsilon$ and $f_{\text{intent}} : \mathrm{X} \rightarrow \Upsilon'$ that maps an input sequence $\mathrm{X} = x_1, x_2, \ldots, x_{T_x}$ to the corresponding label sequence $\Upsilon = y_1, y_2, \ldots, y_{T_x}$ and intent labels $\Upsilon' = y'_1, \ldots, y'_n$ where T_x is the length of the input sequence and n is the number of intents. The sequence of slots is in the form of IOB labels that have three outputs corresponding to 'B' as the beginning of a slot, 'I' as the continuation of a slot, and 'O' as the absence of a slot.

3 Model Description

The structure of our proposed model is shown in Figure 2. The input of the network is a sequence of characters x_i^j where i is a word index and j is a character index inside a word. Our model includes task models that include both the slot filling model (SFM) in (b) of Figure 2 and the intent detection model (IDM) in (c) of Figure 2, as well as language model (LM) in (a) of Figure 2. All three models share character embedding and a word-related layer to build word representation x_i, and task models (SFM and IDM) share two encoding layers as well to build sentence-level representations which are h_i^1 as the output of the 1st encoding layer and h_i^2 as the output of the 2nd encoding layer. The encoding layer of LM is defined separately as h_i^l because it is an independent encoding layer only for LM, which is unlike h_i^1 and h_i^2 which are shared between SFM and IDM. In addition, the slot filling model utilizes sequential outputs as input of the next decoder in the shape of a slot label embedding o_i.

Basically, we use a bi-directional GRU (bGRU) as the RNN of the encoder and decoder in all models (Chung et al., 2014). The locations of dropout (Srivastava et al., 2014) and batch normalization (Ioffe and Szegedy, 2015), selection of the activation function, connection between representations of the encoder and inputs of the decoder, and choices of summation and concatenation are obtained from extensive experiments.

3.1 Character-based Word Representation

A layer of bGRU to build word representation x_i, depicted in the bottommost layer of Figure 2, encodes word information into hidden representations from character inputs. Word representation x_i is the sum of two final hidden states of the bGRU. The first one is a vector concatenating forward pass and backward pass at the first time step of characters. The other one is the same type vector at the last time step of characters. All parameters including character embeddings in the bGRU are shared in SFM, IDM, and LM.

We chose character-based embeddings without word-based embeddings for two reasons. Firstly, for large vocabulary tasks, the size of the word embeddings overwhelms the number of other parameters if word embedding is used. Secondly, the character-based approach could give us better robustness in terms of unknown and rare words than the word-based approach, which is an important aspect for 'open-vocabulary' slots. The larger the number of words we model, the more the sparseness problem is exacerbated. Furthermore, Kim et al., (2015) said that word representation based on character embedding is able to better detect orthographic forms and semantic features and better learn to differentiate between morphemes, such as prefixes and suffixes, than word embedding. Moreover, Verwimp et al. (2017) claimed that it is possible to reveal structural similarities between words and be used when a word is out-of-vocabulary, unknown and infrequent.

3.2 Long-term Aware Attention Structure

Basically, the existing attention mechanism in alignment-based RNN models has the following form (Mnih et al., 2014; Bahdanau et al., 2015; Firat et al., 2016).

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{1}$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{2}$$

$$e_{ij} = f(s_{i-1}, h_j) \tag{3}$$

where j is the sequential index of a word, s_{i-1} is the decoder state at step $i-1$, c_i is a corresponding attention vector, α_{ij} is position-wise weights related to s_{i-1} which is obtained by the function f.

In this formula, the query of attention is the previous decoder state s_{i-1}, and the keys and values of attention are each encoding information h_j. An attention weight α_{ij} is obtained from some operations including softmax and neural network f based on query and keys, and final attention vector c_i is made by weighted sum of attention weights and values.

To address the challenge presented at the beginning of the paper, we took two ideas which embody the concept where the meaning or role of words can vary according to how long a reader focuses on a particular part of a sentence. The first idea is to build multiple RNNs hierarchically and each upper layer uses final forward and backward hidden states of a layer right below it as initial hidden states in opposite directions. For example, SFM and IDM in our model share two layers of the bGRU as a sentence-level encoder as follows:

$$\overrightarrow{h_i^1} = \mathrm{GRU}_1\left(\overrightarrow{h_{i-1}^1}, x_i\right), \overleftarrow{h_{i-1}^1} = \mathrm{GRU}_1\left(\overleftarrow{h_i^1}, x_i\right), i = 1, \ldots, T_x \tag{4}$$

$$\overrightarrow{h_0^1} = 0_d, \overleftarrow{h_{T_x+1}^1} = 0_d \tag{5}$$

$$h_i^1 = \left[\overrightarrow{h_i^1}; \overleftarrow{h_i^1}\right] \tag{6}$$

$$\overrightarrow{h_i^2} = \mathrm{GRU}_2\left(\overrightarrow{h_{i-1}^2}, h_i^1\right), \overleftarrow{h_{i-1}^2} = \mathrm{GRU}_2\left(\overleftarrow{h_i^2}, h_i^1\right), i = 1, \ldots, T_x \tag{7}$$

$$\overrightarrow{h_0^2} = \overleftarrow{h_1^1}, \overleftarrow{h_{T_x+1}^2} = h_{T_x}^1 \tag{8}$$

$$h_i^2 = \left[\overrightarrow{h_i^2}; \overleftarrow{h_i^2}\right] \tag{9}$$

where $\overrightarrow{\ }$ and $\overleftarrow{\ }$ are the forward pass and backward pass of the bGRU, $\overrightarrow{h_0^1}$ and $\overrightarrow{h_0^2}$ are the initial hidden states of the forward pass for the 1st layer and the 2nd layer, $\overleftarrow{h_{T_x+1}^1}$ and $\overleftarrow{h_{T_x+1}^2}$ are the initial hidden states of the backward pass for the 1st layer and the 2nd layer, and [;] is the concatenation operator.

The intuition behind equation (8) is that each pass in an upper bGRU layer provides more global information than a lower one and updates the states of each word time step to redeem correlations between distant words with the help of sentence-level information in the opposite direction of lower bGRU layer. In other words, we stack two layers of the bGRU and treat them as a pair of short-term and long-term meanings. The first layer output h_i^1 of two bGRU layers denotes short-term or relatively local information and the second layer output h_i^2 denotes long-term or relatively global information.

The second idea is that, in the attention mechanism of the decoder, the output h_i^1 of the lower layer is used as both a value and a key and output h_i^2 of the upper layer is used only as a key. We intended that the model would keep short-term information for the concrete meaning of a word, and combine it

using additional sentence-level information for actual meaning or role in a sentence learned from training data. More formally,

$$LA_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j^1 \tag{10}$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{11}$$

$$e_{ij} = f\left(s_{i-1}, h_j^1, h_j^2\right) \tag{12}$$

where LA_i is an attention vector at each decoding step i, h_j^1 is a hidden state of the 1st bGRU layer at word index j, h_j^2 is the hidden states of the 2nd bGRU layer at word index j.

We expect these simple changes allow the model to compensate for missing points or defects at the level of the whole sentence, and also disentangle the local and global requisite information as appropriate for the tasks. The key features of the ideas above are the ability to avoid insufficient tackling of long range dependencies and the ability to offer assistance in learning the actual meaning or role within a sentence. Although the LSTM (Gers and Schmidhuber, 2000) and GRU are well designed to solve the problem of long range dependencies that results in the model becoming unable to capture correlations between words separated by many time steps as a distance between the current word and the relevant word grows, the long range dependencies may still be a lingering issue, especially for 'open-vocabulary' slots, from an approximated learning method like truncated backpropagation through time (truncated BPTT), noisy and meaningless interim inputs, and ambiguity.

In practice, we constitute the function f in (12) used for computing weights by fully connected neural networks.

$$f\left(s_{i-1}, h_j^1, h_j^2\right) = v_a^T \left(Tanh\left(FC(s_{i-1})\right) + Tanh\left(FC\left(h_j^1\right)\right) + Tanh\left(FC\left(h_j^2\right)\right)\right) \tag{13}$$

where FC is a fully connected neural network with a linear matrix and a bias vector to be projected into attention dimension d_a and $v_a \in R^{d_a}$ is a vector to compute one scalar weight logit for each decoding step i in the decoder and word index j of hidden states in the encoder.

3.3 Light-house Positional Encoding

Positional encoding conveys positional information in a form of vectors, which is related to where words are located within a sentence. To appropriately impose the second idea of the long-term aware attention structure on our model, positional encoding in the attention mechanism of the decoder is necessary because the encoder is able to focus more on the meaning in a sentence as intended by the fact that applying it to the attention mechanism separately conveys interactions depending on the distance between words to the decoder.

We introduce a novel positional encoding, called light-house positional encoding, which considers the relative distance between the current word and the word being compared in relation to the current word. Our scheme has two differences from the previous positional encodings (Sukhbaatar et al., 2015; Vaswani et al., 2017; Gehring et al., 2016; Gehring et al., 2017). One is that it defines only one trained embedding, which is unlike position-wise encoding and fixed encoding, and the other is that it is not placed in the inputs of the encoder and decoder but on the inside of attention computation. The formula is as follows:

$$p_i^b = \beta * |b - i| * D \tag{14}$$

where i is a word index, b is the current word index as a basis, $D \in R^d$ is the distance embedding which has positional dimension d, β is a parameter which controls the norm of the encoding increased by the distance.

The existing positional encodings are mostly fixed encoding like sinusoid type or learned position-wise encoding which is set up per each word independently. However, light-house positional encoding defines only one distance embedding D with respect to one time step between words. It configures positional encodings p_i^b for each word index i according to time step distance $|b - i|$ from the current word index b multiplied by a parameter β. Thus, it has fewer position-related parameters than the position-wise encoding. Additionally, the further the distance between words being compared, the larger the norm of positional encoding it has, which makes deeper layers easily notice distance-related information from positive and negative values of encoding and weights. In our experiments, the distance embeddings of models for several domains have positive and negative values in a 50/50 ratio on aver-

age. In practice, although we are able to deal with β as a learned parameter, we treat it as a hyper-parameter, which is closely connected to the average length of sentences. $\beta = 1$ showed stable results in our experiments.

As for placement, we place it only inside of the attention computation function f of (13) as follows:

$$f(s_{i-1}, h_j^1, h_j^2) = v_a^T \left(Tanh(FC(s_{i-1})) + Tanh\left(FC([h_j^1; p_j^i])\right) + Tanh\left(FC([h_j^2; p_j^i])\right)\right) \quad (15)$$

where [;] is concatenation operator and p_j^i is the positional encodings for each word index j of hidden states in the encoder based on each decoding step i in the decoder.

The reason for this placement is also for the sake of using positional encoding only where it is absolutely necessary because the other layers are composed of the bGRU and it exhibits dynamic temporal behavior.

3.4 Configuration for Models and Multi-task Learning

The SFM in (b) of Figure 2 is composed of a layer of the bGRU to build word representation x_i, two layers of the bGRU in the encoder, and a layer of the bGRU including the attention mechanism in decoder. The encoder and decoder of the SFM are based on the long-term attention structure and light-house positional encoding that we explained above. In addition, we utilize both ends of the hidden states of the 2nd encoder layer, h_1^2 and $h_{T_x}^2$, to make an initial state of decoder s_0 by concatenating them and using a layer of the fully connected neural network with a hyperbolic tangent 'tanh'. That means that the initial state of the decoder is built from states with the broadest information in the encoder.

The IDM in (c) of Figure 2 is composed of encoder layers shared with the SFM and two layers of the fully connected neural network in the decoder. It is also based on the long-term attention structure and light-house positional encoding. The difference with the STM in relation to using two ideas is that there is only one attention vector LA_{intent}, its positional encoding is computed by treating the last word time step as the current word index b in (14), and a query of attention is made by a layer of the fully connected neural network with a rectified linear unit 'ReLU', not 'tanh'.

The LM in (a) of Figure 2 is composed of a layer of the bGRU for word representation x_i shared with the SFM and IDM, a separate layer of the bGRU in the encoder, and a layer of the fully connected neural network in the decoder. It adopts a conventional attention structure like (1) to (3) and light-house positional encoding, and it also has only one attention vector called 'Positional attention' like the IDM. The LM was trained by randomly removing a word in a sentence, inserting other words within window 10 ($\pm$5) of the removed word as an input, and using the removed word as an output. Thus, positional encoding in the LM treats the index of the removed word as the current word index b = 5 in (14), and its 10 indexes of input are 0, …,4,6, …,10. Finally, the formula of the decoder is as follows:

$$Positional\ attention = \sum_j \alpha_j h_j^l, j = 0, \ldots, 4, 6, \ldots, 10 \quad (16)$$

$$\alpha_j = \frac{\exp(e_j)}{\sum_k \exp(e_k)}, k = 0, \ldots, 4, 6, \ldots, 10 \quad (17)$$

$$e_j = f(s, h_j^l) \quad (18)$$

$$f(s, h_j^l) = v_a^T \left(Tanh(FC(s)) + Tanh\left(FC([h_j^l; p_j^5])\right)\right) \quad (19)$$

$$\mathrm{s} = Tanh\left(FC([h_0^l; h_{10}^l])\right) \quad (20)$$

where h_j^l is the outputs of a separate encoding layer for $j = 0, \ldots, 4, 6, \ldots, 10$, s is a query of attention, FC is a fully connected neural network with a linear matrix and a bias vector to be projected into attention dimension d_a and $v_a \in R^{d_a}$ is a vector to compute one scalar weight logit for each word index j of hidden states in the encoder, p_j^b is a positional encoding for each word index j of the hidden states in the encoder based on the word index 5 corresponding to the removed word.

In the LM, the decoder formulates one representation from 'Positional attention' via a layer of the fully connected neural network with 'tanh' and we use the dot product between the word representation for each word candidate and the final output to compute the final probability, which indicates how well each word fits into the position of the removed word. In addition, the LM utilizes infrequent normalization softmax approximation, combining the strengths of Noise Contrastive Estimation (NCE) and self-normalization, which results in a higher speed-up factor (Andreas and Klein, 2015).

Domains	# of train	# of test	# of slot labels	# of intent types
Calculator	3401	680	10	2
Calendar	9062	1812	70	24
Camera	9425	1885	63	61
Gallery	61885	12377	54	197
Message	18571	3714	49	74

Table 1. The number of training sets, test sets, slot labels, and intent types in 5 Samsung Bixby domains

	ATIS	Gallery	Calendar	Message
Ratio of "open vocabulary slots" to "close vocabulary slots" in the test set	59.58% (1689/2835)	15.13% (5305/35045)	34.56% (1064/3079)	79.44% (3655/4601)
Ratio of "open vocabulary slots including unknown words" to "the other open vocabulary slots" in the test set	2.42% (41/1689)	4.79% (254/5305)	13.82% (147/1064)	45.42% (1660/3655)

Table 2. Percentages of 'open-vocabulary' slots, including unknown words in the test set for each domain

In practice, all words existing in a training batch are treated as a result of down-sampling for approximating the normalization term.

We institute multi-task learning for the above three models with two types called 'Slot only' and 'Joint'. The first one is a combination of the SFM and the LM and the other is a combination of all three models. It also follows that joint learning of intent detection has a positive effect on performance of slot filling as in Liu and Lane (2016). A final cost function to be maximized is as follows:

$$\mathcal{L} = \log\big(p(\mathrm{Y}|\mathrm{X})\big) + \log\big(p(Y'|X)\big) + \lambda\Big(\log\big(p(x_b|X \setminus x_b)\big) + \textstyle\sum_{w_i \setminus embeddings} \| w_i \|_2^2\Big) \quad (21)$$

$$p(\mathrm{Y}|\mathrm{X}) = \textstyle\prod_{t=1}^{T_x} p\big(y_t \big| x_1, \dots, x_{T_x}, y_1, \dots, y_{t-1}\big) \quad (22)$$

$$p(Y'|X) = \textstyle\prod_{k=1}^{n} p\big(y'_k \big| x_1, \dots, x_{T_x}\big) \quad (23)$$

where λ is a weight-decay hyper-parameter for the LM and l2-norm, T_x is the length of the input sequence, n is the number of intents, and w_i is all the parameters learned from training data. Thus, the LM is used for stability and utility of character-based word representation and can be construed as a regularizer for the SFM and IDM.

4 Experiments

4.1 Data

The ATIS corpus (Hemphill et al., 1990) is the most commonly used dataset for the research on spoken language understanding. In this study, we used the ATIS corpus from Hakkani-Tur et al. (2016).[1]

The dataset consists of sentences of people making flight reservations. The training set contains 4978 utterances from the ATIS-2 and ATIS-3 corpora, and the test set contains 893 utterances from the ATIS-3 NOV93 and DEC94 datasets. There are in total 127 distinct slot labels and 18 different intent types.

For this study we collected English in-house real data of five different Bixby domains. We adopted domains and corresponding intent types to show the performance of our model in terms of various levels of difficulty. Table 1 describes the number of training sets, test sets, slot labels, and intent types in the five domains. The calculator and camera domains have no 'open-vocabulary' slots, and 'open-vocabulary' slots of the others, including the ATIS corpus, are shown in Table 5 of Appendix B. We did not utilize any external information, such as a named entity database or knowledge base with entities like city, airport, person, etc., to focus on the performance only using the lexical input of language. Thus, we treated named entity slots as 'open-vocabulary' slots. Additionally, we defined the difficulty of each domain associated with 'open-vocabulary' slots on the basis of how many values of 'open-vocabulary' slots in each domain included unknown words in the test set. Table 2 shows that ATIS, gallery, calendar, and message had a high level of difficulty in an increasing order.

For the five Bixby domains, we compared our model to three previous models: the hybrid RNN (Mesnil et al., 2015), the encoder-labeler deep LSTM (Kurata et al., 2016), and the attention biRNN for both 'Slot only' and 'Joint' (Liu and Lane, 2016).

[1] https://github.com/yvchen/JointSLU/tree/master/data

Model	Best F1 score (Slot filling)	Best error rate (Intent detection)
RNN with Label Sampling (Liu and Lane, 2015)	94.89	-
Hybrid RNN (Mesnil et al., 2015)	95.06	-
Deep LSTM (Yao et al., 2014)	95.08	-
RNN-EM (Peng et al., 2015)	95.25	-
bLSTM (Hakkani-Tur et al, 2016)	95.48	-
Encoder-labeler Deep LSTM (Kurata et al., 2016)	95.66	-
Word embeddings updated and bLSTM (Kim et al., 2016)	-	2.69
LSTM (Ravuri and Stolcke, 2015a)	-	2.45
Attention Encoder-Decoder NN (Slot only) (Liu and Lane, 2016)	**95.78**	
Attention BiRNN (Slot only) (Liu and Lane, 2016)	95.75	
Attention Encoder-Decoder NN (Joint) (Liu and Lane, 2016)	95.87	**1.57**
Attention BiRNN (Joint) (Liu and Lane, 2016)	**95.98**	1.79
Proposed model (Slot only)	**95.93**	-
Proposed model (Joint)	**95.93**	**1.46**

Table 3. Best F1 score of slot filling and best error rate of intent detection for ATIS in comparison with previous approaches

Model	Slot filling					Intent detection				
	Gallery	Calculator	Calendar	Message	Camera	Gallery	Calculator	Calendar	Message	Camera
Hybrid RNN (Mesnil et al., 2015)	88.41	97.18	71.57	74.17	87.44	-	-	-	-	-
Encoder-labeler Deep LSTM (Kurata et al., 2016)	94.03	98.2	87.34	88.42	93.51	-	-	-	-	-
Attention BiRNN (Slot only) (Liu and Lane, 2016)	98.48	99.73	93.94	91.07	99.44	-	-	-	-	-
Attention BiRNN (Joint) (Liu and Lane, 2016)	97.31	99.82	91.98	90.50	99.34	1.35	**0.00**	3.20	2.53	**0.53**
Proposed model (Slot only)	99.19	99.83	95.34	94.70	**99.58**	-	-	-	-	-
Proposed model (Joint)	**99.24**	**99.93**	**96.13**	**95.00**	**99.58**	**1.32**	**0.00**	**2.76**	**1.46**	**0.53**

Table 4. Best F1-score of slot filling and best error rate of intent detection for five Bixby domains in comparison with previous approaches

4.2 Training details

For the training of our model, we used a mini-batch stochastic descent method named Adadelta (Zeiler, 2012), a method with a relatively simple configuration, and set its momentum related parameter to a value of 0.95, but we used default settings and methods specified in published codes and papers for the comparison models.[2] We also utilized l2-norm regularization for parameters and gradient clipping for stable training.

For tuning hyper-parameters, we split the training set into 90% training and 10% development. After choosing the hyper-parameters, we re-trained the model with all of the training data. For evaluation, we evaluated the performance on slot filling using an F1 score and the performance on intent detection using the classification error rate, and each result was recorded from the best one among 10 different initializations.

We did not utilize any kind of pre-training or external data for any of the experiments. Details of the preprocessing procedures and the finally determined model parameters are given in Appendix A.

4.3 Results

For the ATIS corpus, Table 3 shows that our models for both 'Joint' and 'Slot only' achieved comparable results to the state-of-the-art result of slot filling. In particular, our 'Slot only' model provided improvement from 95.78 to 95.93 in comparison to the previous best 'Slot only' model. For intent detection, Table 3 also shows that we provided the state-of-the-art performance with a 1.46 error rate using the 'Joint' model. In addition, learning with additional intent information in our model had no positive effect on the performance of slot filling.

On the other five domains of Bixby, we compared our model with three previous methods for slot filling and also compared it with only one previous method for intent detection. The Hybrid RNN

[2] Attention BiRNN : https://github.com/HadoopIt/rnn-nlu and Hybrid RNN : https://github.com/mesnilgr/is13

(Mesnil et al., 2015) applied the RNN to slot filling for the first time, the Encoder-labeler Deep LSTM (Kurata et al., 2016) applied the encoder-decoder concept to slot filling, and the Attention BiRNN (Liu and Lane, 2016) applied attention and multi-task learning to slot filling and intent detection and achieved the state-of-the-art performance among the previous approaches for both. Thus, for slot filling, we intended to check change of the performance as additional ideas are introduced such as the RNN, encoder-decoder, attention, and long-term aware attention and light-house positional encoding. Not to our surprise, adding ideas improved the performance and our model demonstrated the best performance for both slot filling and intent detection. In particular, as the difficulty of domains associated with 'open-vocabulary' slots increased, like calendar and message, a larger gap in performance emerged to the amount of about a 5% absolute gain for slot filling and about a 1% error rate for intent detection. However, multi-task learning with the intent detection model did not have much effect on the performance of slot filling. The results are summarized in Table 4.

To see more detailed results in terms of 'open-vocabulary' slots, we divided the results of slot filling into an F1 score on 'closed-vocabulary' slots and an F1 score on 'open-vocabulary' slots as shown in Table 6 of Appendix C. We highlight some observations.

First, results from our models of both 'Joint' and 'Slot only' had better F1 scores over the previous best model with respect to 'open-vocabulary' slots for domains having them. In accordance with the results of all the slots, our model experienced more benefit in the performance of 'open-vocabulary' slots as the difficulty of the domains we defined increased. Furthermore, the first three samples of Table 7 in Appendix D and all the samples of Table 8 in Appendix D confirm that our model tends to draw conclusions by referring to the whole sentence as we intended.

Second, there was a trade-off to some degree between 'closed-vocabulary' slots and 'open-vocabulary' slots as was revealed in the ATIS corpus case. That is why the performance of our model could not exceed the state-of-the-art result in Table 3. If the distinction or boundary between slots is not clear like 'Boston late night' in a sentence "ground transportation that I could get in Boston late night", the whole can be treated as an 'open-vocabulary' slot due to a shortage of training data about the '*period_of_day*' slot corresponding to 'late night'. We have to recognize it depending only on the content of the sentence without any information about the city name, and moreover, 'boston late night' may be construed as one of city names. In addition, our model tends to open up more possibilities for various conclusions. For an example of "flight ## from jfk to lax", our model considers 'lax' as a '*city_name*' slot despite the fact that '*airport_code*' slot consists of three letters. Samples of slot filling results for the ATIS corpus are in Table 7 of Appendix D.

Third, multi-task learning of our 'Joint' model generally had a higher performance than our 'Slot only' model regarding 'open-vocabulary' slots on all datasets, including the ATIS corpus. Therefore, it is safe to say that the performance of 'open-vocabulary' slots in our approach benefits more from the joint training with intent detection and the help of sentence-level information. On the contrary, unlike the result of Liu and Lane (2016) the 'Joint' model of them had a lower performance than the 'Slot only' model of them for 'closed-vocabulary' slots as well as for 'open-vocabulary' slots.

5 Conclusion and Future Work

In this study, we presented a new modeling with RNNs and successfully dealt with 'open-vocabulary' slots. It focuses more on relatively global information within a sentence using a long-term aware attention structure and light-house positional encoding with the help of multi-task learning of a character-based language model and intent detection model. Compelling experimental results on several domains demonstrated the advantages of our model, especially for 'open-vocabulary' slots.

In this study, we did not utilize additional methods for generalization, such as variational dropout (Gal and Ghahramani, 2016), attention weights dropout, label smoothing, and dropout in decoder (Vaswani et al., 2017; Merity et al., 2017; Melis et al., 2017), and we did not search extensively hyper-parameter space and choices of optimizers extensively as in Merity et al. (2017) and Melis et al. (2017). These could enhance the performance of our RNN model. Furthermore, we will conduct not only a delicate analysis of weights of the long-term aware attention structure but also experiments using models based on architectures other than RNN and hierarchically deeper structures.

Reference

Jacob Andreas and Dan Klein. 2015. *When and why are log-linear models self-normalizing?* in Proc. NAACL.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. *Neural Machine Translation by jointly learning to align and translate.* in ICLR.

Ankur Bapna, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2017. *Towards Zero-Shot Frame Semantic Parsing for Domain Scaling.* in Proc. Interspeech.

Jason PC Chiu and Eric Nichols. 2015. *Named Entity Recognition with Bidirectional LSTM-CNNs.* arXiv preprint arXiv:1511.08308.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.* in Proc. NIPS.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. *Multi-Way, Multilingual Neural Machine Translation with a Shared Attention Mechanism.* in Proc. NAACL.

Yarin Gal and Zoubin Ghahramani. 2016. *A Theoretically Grounded Application of Dropout in Recurrent Neural Networks.* in Proc. NIPS.

Jonas Gehring, Michael Auli, David Grangier, and Yann N. Dauphin. 2016. *A Convolutional Encoder Model for Neural Machine Translation.* arXiv preprint arXiv:1611.02344.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. *Convolutional Sequence to Sequence Learning.* arXiv preprint arXiv:1705.03122.

Felix A. Gers and Jurgen Schmidhuber. 2000. *Recurrent Nets that Time and Count.* In Neural Networks. in Proc. IJCNN.

Dilek Hakkani-Tur, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. *Multi-Domain Joint Semantic Frame Parsing using Bi-directional RNN-LSTM.* in Proc. Interspeech.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. *Bidirectional LSTM-CRF Models for Sequence Tagging.* arXiv preprint arXiv:1508.01991

Sergey Ioffe and Christian Szegedy. 2015. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* in Proc. ICML.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. *Character-Aware Neural Language Models.* CoRR, abs/1508.06615.

Joo-Kyung Kim, Gokhan Tur, Asli Celikyilmaz, Bin Cao, and Ye-Yi Wang. 2016. *Intent detection using semantically enriched word embeddings.* in Proc. IEEE SLT.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. *Leveraging Sentence-level Information with Encoder LSTM for Natural Language Understanding.* arXiv preprint arXiv:1601.01530.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. *On the state of the art of evaluation in neural language models.* arXiv preprint arXiv:1707.05589.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. *Regularizing and Optimizing LSTM Language Models.* arXiv preprint arXiv:1708.02182.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, and Geoffrey Zweig. 2015. *Using Recurrent Neural Networks for Slot Filling in Spoken Language Understanding.* in Proc. IEEE/ACM.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. *Recurrent Models of Visual Attention.* in Proc. NIPS.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington.1990. *The ATIS Spoken Language Systems Pilot Corpus.* in Proc. DARPA Speech and natural language workshop.

Bing Liu and Ian Lane. 2015. *Recurrent Neural Network Structured Output Prediction for Spoken Language Understanding.* in Proc. NIPS.

Bing Liu and Ian Lane. 2016. *Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling.* in Proc. Interspeech.

Baolin Peng, Kaisheng Yao, Li Jing, and Kam-Fai Wong. 2015. *Recurrent Neural Networks with External Memory for Spoken Language Understanding.* Natural Language Processing and Chinese Computing, pp. 25–35, Springer.

Suman Ravuri and Andreas Stolcke. 2015a. *A comparative study of neural network models for lexical intent classification.* in Proc. IEEE ASRU.

Suman Ravuri and Andreas Stolcke. 2015b. *Recurrent Neural Network and LSTM Models for Lexical Utterance Classification.* in Proc. Interspeech.

Yangyang Shi, Kaisheng Yao, Hu Chen, Yi-Cheng Pan, Mei-Yuh Hwang, and Baolin Peng. 2015. *Contextual spoken language understanding using recurrent neural networks.* in Proc. ICASSP.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting.* JMLR, 15:1929–1958.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. *End-to-End Memory Networks*. In NIPS.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. *Attention Is All You Need.* arXiv preprint arXiv:1706.03762.

Lyan Verwimp, Joris Pelemans, Hugo Van hamme, and Patrick Wambacq. 2017. *Character-Word LSTM Language Models.* in Proc. EACL.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. *Spoken Language Understanding using Long Short-Term Memory Neural Networks.* in Proc. IEEE SLT.

Matthew D. Zeiler. 2012. *ADADELTA: An adaptive learning rate method.* arXiv preprint arXiv:1212.5701.

Appendix A. Experimental Details

We preprocessed the datasets as follows for all datasets. To deal with unseen or unknown inputs in the test set, we replaced all 'words' with at most three occurrences in the training data by <UNK> for comparison models and all 'characters' with at most ten occurrences in the training data also by <UNK> for our model. We also changed all sequences of numbers to sequences of '#', for example, '2017' is converted to '####'. For comparison models, we applied basic tokenizer before training, which is related to apostrophe and a combination of number and word.

For the training of our model, we used the Adadelta with 0.95 momentum related parameter and 5 gradient clipping. Parameters related to batch normalization are initialized by 1, parameters related to the hidden of RNN are initialized by orthogonal values obtained from the singular vector decomposition and the standard normal distribution with scale 1, bias parameters are initialized by zeros, and all other parameters are initialized by the standard normal distribution with scale 0.01. All dimension settings are simply 100 except 50 dimension of a slot label embedding o_i.

Hyper-parameters are 0.5 dropout, $\beta = 1$ hyper-parameter of the light-house positional encoding, 0.02 noise probability of character replacement for each characters in the inputs, 0.1 weight decay for the l2-norm of parameters learned from the training data and the language model cost, and 5 to 12 batch size according to the domains.

Appendix B. Lists of 'Open-vocabulary' Slots

Domains	'Open-vocabulary' slots
ATIS	{*airport_name, airline_name, city_name, state_name, fromloc.airport_name, fromloc.city_name, fromloc.state_name, stoploc.airport_name, stoploc.city_name, toloc.airport_name, toloc.city_name, toloc.country_name, toloc.state_name*}
Gallery	{*album_name, app_name, recipient, contact, file_name, location, poi, recipient, story_name, tag_name, title*}
Calendar	{*app_name, recipient, save_invitee, save_location, save_notes, save_title, search_keyword*}
Message	{*appname, blocked_phrase, search_keyword, text_body, recipient, contact*}

Table 5. List of 'open-vocabulary' slots for ATIS, gallery, calendar, and message.

Appendix C. Slot Filling for 'Open-vocabulary' Slots

Models	Slot types	ATIS	Gallery	Calendar	Messages
Attention BiRNN (Joint) (Liu and Lane, 2016)	Closed vocabulary slots	94.64	97.91	93.71	95.07
	Open vocabulary slots	96.68	93.97	88.69	89.31
Attention BiRNN (Slot only) (Liu and Lane, 2016)	Closed vocabulary slots	**95.06**	98.85	94.82	95.64
	Open vocabulary slots	96.58	96.42	90.88	89.89
Proposed model (Joint)	Closed vocabulary slots	94.38	**99.39**	**97.07**	**97.30**
	Open vocabulary slots	**96.98**	**98.36**	**94.34**	**94.40**
Proposed model (Slot only)	Closed vocabulary slots	94.68	99.34	96.79	96.89
	Open vocabulary slots	96.77	**98.38**	92.57	94.13

Table 6. F1 score on 'open-vocabulary' slots and 'closed-vocabulary' slots for ATIS, gallery, calendar, and message in comparison with the previous best approach.

Appendix D. Slot Filling Samples

Samples	Ground Truth	Slot results of Attention BiRNN (Slot only) (Liu and Lane, 2016)	Slot results of the proposed model
Show me flights from montreal to orlando and **"long beach"**	toloc.city_name	fromloc.city_name	**toloc.city_name**
find me a flight from cincinnati to any airport in the "**new york"** city area	toloc.city_name	fromloc.city_name	**toloc.city_name**
please find all the flights from cincinnati to any airport in the "**new york city"** area that arrive next saturday before # pm	toloc.city_name	fromloc.city_name	**toloc.city_name**
and now show me ground transportation that i could get in "**boston late night"**	city_name (boston) & period_of_day (late night)	**city_name (boston)** & state_code (late) & **period_of_day (night)**	city_name (Boston late night)
list the airfare for american airlines flight ## from jfk to **"lax"**	toloc.airport_code	**toloc.airport_code**	toloc.city_name

Table 7. Samples of the slot filling results for the ATIS corpus. The bold and underlined letters in samples are targets for analysis. The slots in bold means that they are correct and the letters in parentheses are the outputs corresponding to the slots.

Samples	Ground Truth	Slot results of Attention BiRNN (Slot only) (Liu and Lane, 2016)	Slot results of the proposed model
get rid of all the sms with **"bolt from the blue"** in them	search_keyword (bold from the blue)	search_keyword (the blue)	**search_keyword (bolt from the blue)**
Deliver tony jones and nine six zero four a message with **"say it til you believe it"**	text_body (say it til you believe it)	text_body (it til you believe it)	**text_body (say it til you believe it)**
snap a selfie camera picture also share with **"ronnie"**	recipient	search_keyword	**recipient**
Text to lou **"i just finished the last of my exams"**	text_body (I just finished the last of my exams)	text_body (just finished) & search_keyword (exams)	**text_body (I just finished the last of my exams)**
For phone charging on 12 december invite **"kayla green"**	save_invitee (kayla green)	search_keyword (kayla) & save_title (green)	**save_invitee (kayla green)**
i've gotta go **"guatemala city"** with resendez today so place that a new event	save_location	save_title	**save_location**

Table 8. Samples of slot filling results for message and calendar. The bold and underlined letters in samples are targets for analysis. The slots in bold means that they are correct and the letters in parentheses are the outputs corresponding to the slots.

Challenges and Opportunities of Applying Natural Language Processing in Business Process Management

Han van der Aa
Humboldt University
Berlin, Germany
vanderah@hu-berlin.de

Josep Carmona
Univ. Politècnica de Catalunya
Barcelona, Spain
jcarmona@cs.upc.edu

Henrik Leopold
Vrije Univ. Amsterdam
Amsterdam, The Netherlands
h.leopold@vu.nl

Jan Mendling
Vienna Univ. of Economics and Business
Vienna, Austria
jan.mendling@wu.ac.at

Lluís Padró
Univ. Politècnica de Catalunya
Barcelona, Spain
padro@cs.upc.edu

Abstract

The Business Process Management (BPM) field focuses in the coordination of labor so that organizational processes are smoothly executed in a way that products and services are properly delivered. At the same time, NLP has reached a maturity level that enables its widespread application in many contexts, thanks to publicly available frameworks. In this position paper, we show how NLP has potential in raising the benefits of BPM practices at different levels. Instead of being exhaustive, we show selected key challenges were a successful application of NLP techniques would facilitate the automation of particular tasks that nowadays require a significant effort to accomplish. Finally, we report on applications that consider both the process perspective and its enhancement through NLP.

1 Introduction

In the last decade, the maturity achieved by Natural Language Processing (NLP) technologies, together with the explosion of *big data* and *deep learning* techniques, have turned the spotlight to the possibilities offered by NLP approaches for a variety of novel applications. Many of these applications are situated in a business setting where documents and textual data is extensively used to manage production, logistics, accounting, and procurement, to give just a few examples. These application areas have in common that they relate to various business processes that are executed inside a company and beyond.

Organizing business processes in an efficient and effective manner is the overarching objective of business process management. Classically, business process management has been many concerned with the quantitative analysis of key performance dimensions such as time, cost, quality, and flexibility (Dumas et al., 2013) without taking the automatic processing of textual data too much into account. Recent research highlights the potential of NLP-based analysis techniques (Leopold, 2013; Mendling et al., 2015; Mendling et al., 2017) to support many business process management tasks in a scalable fashion.

In this paper, we describe the application of NLP techniques to BPM, where the focus rests on the processes that an organization must carry out on its daily activities, and on how are they modeled, updated, optimized, and shared with the relevant stakeholders. We believe that the NLP community has much to offer to BPM, as well as many interesting challenges to address, and we aim to display the most relevant in this paper.

The rest of the paper is structured as follows. Section 2 describes the background of our research. Section 3 provides an outline how NLP could inform business process management tasks in the future. Section 4 highlights the potential impact of NLP-supported business process management in various domains. Finally, Section 5 concludes the paper.

2 Background

In this section, we describe the essential ideas of business process management and its major design and analysis artifacts, which are process representations and event logs. Likewise, we provide a current

Proceedings of the 27th International Conference on Computational Linguistics, pages 2791–2801
Santa Fe, New Mexico, USA, August 20-26, 2018.

perspective on NLP and how it can be oriented towards improving BPM in a general setting.

2.1 Requirements of Business Process Management

Business process management is concerned with various management activities that are related to business processes (Dumas et al., 2013). In line with Mendling et al. (2017), we describe three levels of business process management as illustrated in Figure 1. The top level called multi process management is concerned with the identification of the major processes of an organization and the prioritization of these. The middle level is concerned with the management of a single process along the classical BPM lifecycle (Dumas et al., 2013) including the steps of discovery, analysis, redesign, implementation and controlling. The level of process instance management deals with planning the tasks of a process, executing them, monitoring them, and potentially adapting the instance if required. All the three layers make use of business process models and event data.

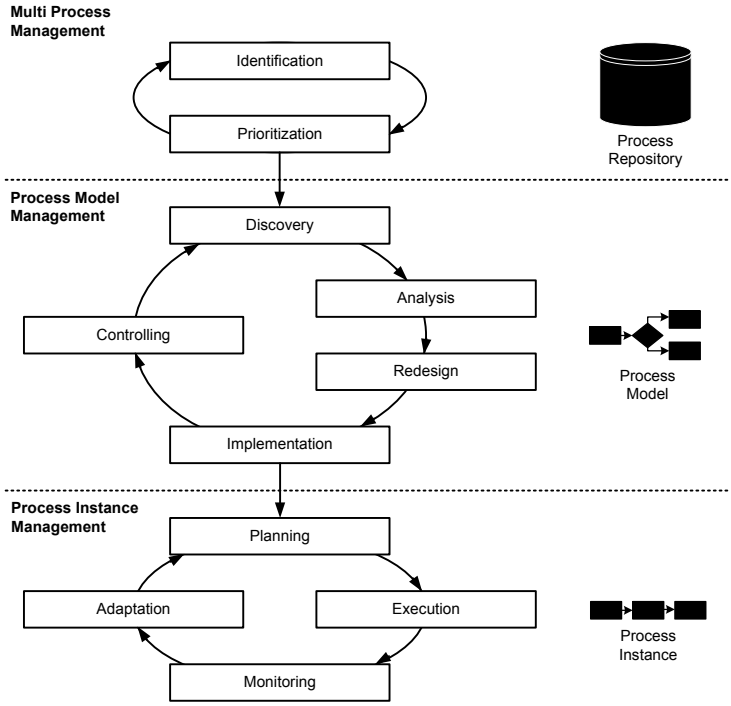


Figure 1: Three Levels of Business Process Management (taken from (Mendling et al., 2017)).

2.2 Process Representations

A variety of representation formats can be used to capture process information in informational artifacts (Wolter and Meinel, 2010), including process models (Davies et al., 2006), natural language descriptions (Phalp et al., 2007), spreadsheets (Krumnow and Decker, 2010), and checklists (Reijers et al., 2017). The representation format used to provide process information to users should be well-suited for its particular purpose, in two ways: A format should convey its informational content in a useful manner and the intended users should be able to appropriately understand the received format (van der Aa, 2018).

Representation formats emphasize different aspects of business processes. This means that the choice for a certain representation format depends on the intended focus of an informational artifact. For instance, *natural language text* can be very useful to provide process participants with detailed insights on how to perform complex tasks (Baier and Mendling, 2013). However, for a process participant who needs to be sure that all necessary steps are performed, a *checklist* might be more useful. This later format could be more suitable because it emphasizes the information that is of primary importance for that purpose. Furthermore, process models have been found to be better suited to express complex execution logic of a process in a more comprehensive manner than natural language (Mendling, 2008, p.23).

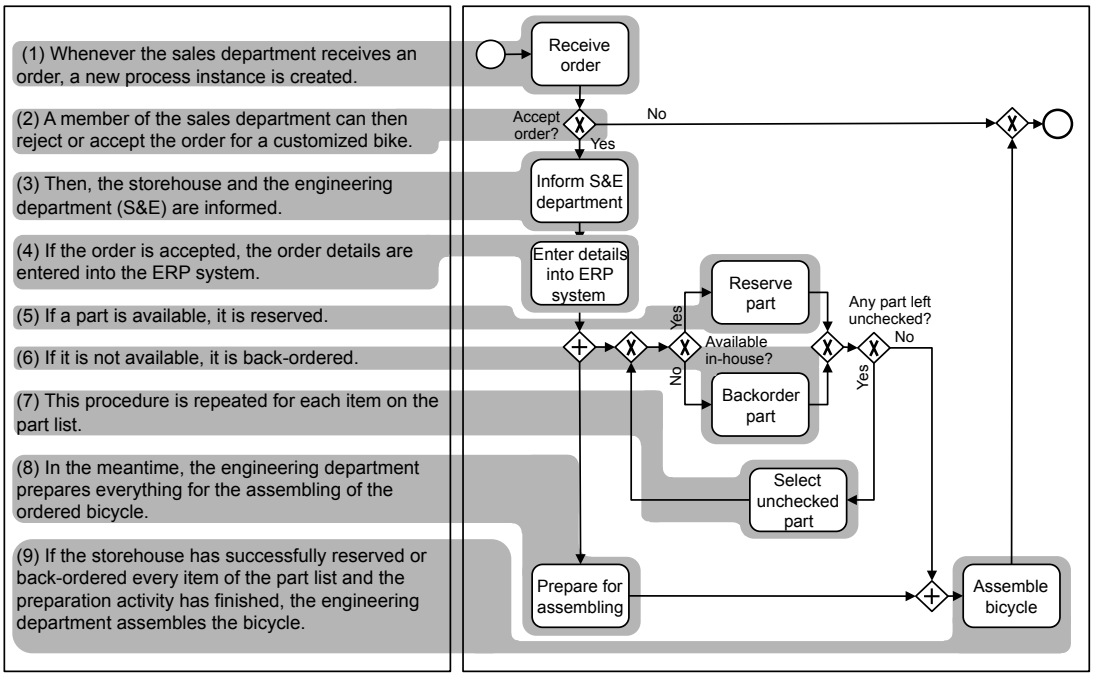


Figure 2: A textual and a model-based description of a bicycle manufacturing process.

It is also important that users of an informational artifact are able to work well with the employed representation format. The ability of users to do so can depend on their familiarity and preferences with respect to different formats. Research by Figl and Recker (2016) shows that people prefer different process representation formats depending on the application purpose and on the *cognitive style* of the user. For example, some participants were found to prefer textual descriptions over process models, whereas others preferred models over text for the same purpose. The influence of user preferences on the choice for model or text-based process representations is also recognized by Recker et al. (2012) and Chakraborty et al. (2010).

Figure 2 shows an example of two different process representations, namely a process model and a textual process description. On the left- hand side, we observe a textual description, which comprises nine sentences. On the right-hand side, a corresponding model-based description can be seen, modeled using the Business Process Model and Notation (BPMN). The model contains eight activities, which are depicted using boxes with rounded edges. The diamond shapes that contain a plus symbol indicate concurrent streams of action; the diamond shapes containing a cross represent decision points. The gray shades suggest correspondences between the sentences and the activities of the process model.

2.3 Event Data for the Footprints of Process Executions

Information systems supporting processes in organizations enable the persistent storage of process executions, in large log files denoted *event logs*. Event logs can be seen as a tabular representation of all the necessary events for a process execution (denoted as *case*) to be accomplished. Table 1 provides an excerpt of an event log for the bicycle manufacturing process used in this paper. Three important elements (columns in the table) that identify an event (row) in an event log are: i) the case ID (to identify the process instance or case executed), ii) the timestamp (to identify when the event was executed), and iii) the activity name (to identify the task performed). Extra information in form of additional columns may exist, to provide contextual information to the execution of events. For instance, in the table three cases appear: case 1 denotes an order that was rejected, case 2 denotes an order that was completely executed till the bicycle assembly, and case 3 is ongoing.

In contrast to the process representations described in the previous section, event logs describe the reality, i.e., the recorded events witnessing all process instances stored in the information system. We will refer to event logs in this paper in particular parts, to highlight the importance to relate observed and

Event	Case ID	Activity	Timestamp	Dept.	Add. Data
1	1	`Receive Order`	10-04-2015 9:08am	Sales	Reject
2	2	`Receive Order`	10-04-2015 10:03am	Sales	Accept
3	2	`Inform S&E dept.`	10-04-2015 10:05am	Sales	–
4	2	`Enter details ERP`	10-04-2015 10:09am	Sales	–
5	2	`Prepare for assemb.`	10-04-2015 10:10am	Eng.	–
6	2	`Reserve part`	10-04-2015 10:12am	Storeh.	Wheel
7	2	`Sel uncheck. part`	10-04-2015 11:12am	Storeh.	Breaks
8	2	`Backorder part`	10-04-2015 12:06am	Storeh.	Breaks
9	3	`Receive Order`	10-04-2015 13:18am	Sales	Accept
10	2	`Assemble bycicle`	11-04-2015 10:03am	Eng.	–

Table 1: Part of an event log for the bicycle manufacturing process.

modeled behavior.

2.4 NLP Capabilities and Frameworks

There are several existing NLP technologies that can be more or less straightforwardly applied to BPM. As illustrated in Fig. 2, a textual description of a bussiness process is a text where several actors are mentioned, and their actions and interactions described. Thus, out-of-the-box analyzers can be used to structure the content of the text. Tools such as Stanford Core (Manning et al., 2014), FreeLing (Padró and Stanilovsky, 2012), Apache OpenNLP[1], or NLTK (Bird et al., 2009) can be used to extract predicates and involved actors via SRL, perform WSD to identify domain objects that may be mentioned using different words, solve coreferences to decide which of the mentioned actors correspond to the same entity, decide which order the described actions must follow or whether there are choices or loops, etc.

However, existing analysis pipelines are still below optimal performance, specially in the more semantic tasks (WSD, coreference, SRL, temporal relation extraction, etc.) and on an unrestricted domain (process descriptions may report organization processes in a huge range of sectors –health, education, banking, manufacturing, etc).

So, several challenges for NLP researchers can be found in the application to BPM:

- Improvement of the performance of individual analyzers, specially at the semantic/pragmatic level (e.g. in Figure 2, the mentions *the sales department* in sentence (1), and *a member of the sales department* in sentence (2) are actually referring to the same actor in the process, but a coreference system would consider them two different entities).
- Domain adaptation methods to tune generic NLP processors to deal with process descriptions in a specific organization or sector. This may require the creation/acquisition of tailored ontologies that help specifying with the right terms important parts of the process and relations among these relevant domain concepts.
- Definition of new tasks, such as the detection of exclusivity, parallelism/concurrency, decision points, or iteration of tasks described in the text (e.g. the phrase *this procedure* or the marker *in the meantime* in sentences (7) and (8) in Figure 2 are ambiguous with respect their antecedent, and each interpretation leads to different formal models).
- Use of world knowledge to improve the results (often some steps or relations among tasks are omitted because the reader is assumed to be able to understand it using common sense or domain knowledge, such as that a payment happens always after an invoice is emitted, or that a bicycle can not be assembled until all required parts have been obtained).

[1] `https://opennlp.apache.org`

- Information extraction from event logs. Among others, interesting directions is the elicitation through NLP techniques of process steps for loosely specified processes, projecting event logs for events at a given granularity level, etc.

3 Expanding BPM Capabilities through NLP

In this section, we provide interesting research directions based on incorporating/extending NLP capabilities in the three levels described in Fig. 1.

3.1 Multi-Process Management

Multi process management is concerned with the identification of the major business processes of a company and the prioritization of these processes. The overall landscape of business processes is often represented as a process architecture, which is stored in a business process repository. Large multinational corporations often maintain several thousand process models in such repositories.

Prior research on multi process management has focused on repository management, building on techniques for determining process similarity (Dijkman et al., 2011), automatically matching business process models (Weidlich et al., 2010), and other querying techniques (Leopold et al., 2017). Several concepts have been proposed on top including automatic refactoring (Weber et al., 2011) including harmonization of terminology (Pittke et al., 2015), automatic service derivation (Leopold et al., 2015), semantic search (Thomas and Fellmann, 2009) and merging business process models (Rosa et al., 2013).

Several challenges remain in this area including the following (Mendling et al., 2015). First, the availability of a business process repository bears the potential to discover an overarching formal ontology that captures the full spectrum of operations of a company. Second, content captured in the repository might be automatically categorized, for example with respect to documentation standards such as ISO:9001. Such tasks require the application and adaptation of existing NLP techniques in this specific context.

3.2 Process Management

Several important challenges need to be tackled so that tasks in the process model management stage of Fig. 1 such as discovery, analysis, redesigning and controlling of processes are excelled at organizations. In this section we focus on providing examples of key enablers for tackling these challenges.

The discovery of processes concerns the identification of the main processes in an organization, and the corresponding elicitation into a (graphical) notation such as BPMN, which facilitates the communication between the stakeholders involved in a process. In practice, this phase is often implemented using workshop meetings, requiring quite significant efforts to materialize a process model from the conversations arising in these meetings. An alternative is the use of process mining techniques, which enable, for instance, the automatic discovery of process models from event logs (van der Aalst, 2016).

NLP techniques can be applied at very different granularities to boost the discovery of precise process model descriptions of a process. Among the available techniques, we highlight the most disruptive ones:

- *Transform textual descriptions into process models*: The creation of process models consumes up to 60% of the time spent on process management projects. This is a paradox, because there are often extensive textual process documentations available in organizations. Therefore, automatically transforming textual process descriptions to process models represents a particularly attractive use case. Several techniques have been developed for this purpose (Friedrich et al., 2011). These use tailored NLP techniques in order to identify actions, e.g. "*The sales department receives an order*", and their inter-relations, e.g. "*If the order is accepted, the order details are entered into the system*. These extracted components represent the foundation for the generation of a process model from a text. However, a number of challenges still remain for this endeavor. For instance, techniques must be able to identify sentences that provide contextual information, rather than describe process steps. Furthermore, the inherent ambiguity of natural language can lead to different interpretations regarding the process that is described (Van der Aa et al., 2016).

- *Translate process models*: Sometimes the same process (e.g., the admission process to enroll in a university) is defined multiple times (e.g., several universities in several countries, or a worldwide

company that applies a given process across several countries). In these contexts, it is crucial to have multilingual support for translating a given process model description (in any form) into a target language. For the case of process model graphical descriptions (e.g., BPMN), translation of individual model elements may be sufficient to obtain a translated description with a certain quality, since the structure of the process is retained. In contrast, translating textual descriptions of a process may become more challenging, due to the particularities each language has in describing certain constructs.

- *Text Annotation and Inference*: In the scope of the two previous use cases (from text to process models, process model translation), the difficulty of generating the output would be significantly alleviated if the text was correctly annotated, reducing the noise introduced by automatic NLP tools. One can envision that, in the narrow scope of the description of a process one could define a limited set of template annotations that address particular perspectives (control-flow, roles, data, among others) which can help a user to (partially) annotate a textual description of a process. Nowadays, there exist advanced tools for text annotation such as Brat (Stenetorp et al., 2012). From annotated textual descriptions of a process model (e.g., control-flow annotations establishing relations between two sentences), inferences can elicit new relations that can be used to have a more precise and refined descriptions (e.g., transitive closure of the relations). The same annotations could be used as training data to improve or adapt automatic languages analyzers tuned to this particular tasks.

The analysis phase is oriented towards finding weakness of current process model candidates. This phase can also be significantly improved by a tailored used of NLP techniques in particular situations. We describe here some examples of challenges to face in order to materialize such improvements:

- *Verify semantic correctness and completeness*: The semantic quality of process models is crucial to the proper understanding of business processes. A number of techniques aim to verify this quality in an automated manner. These techniques achieve this, for example, by detecting and/or correcting inconsistent use of terminology (Koschmider and Blanchard, 2007) or violations of labeling conventions (Becker et al., 2009; Leopold et al., 2013a; Van der Vos et al., 1997). Others aim to improve modeling quality by detecting common modeling errors (Gruhn and Laue, 2011) or ambiguously labeled activities (Pittke et al., 2015; Pittke et al., 2014).

- *Calculate consistency between process model and text*: As mentioned before, keeping different process descriptions helps improving the knowledge about processes across an organization. However, as processes evolve continuously, it is necessary to detect inconsistencies between process descriptions in order to ensure the expectations for process outcomes are the same for every actor (van der Aa et al., 2017). The challenge here is to find correspondences between sentences in the text and elements in the process model, and warn in case important parts are not mapped. This requires a respective NLP analysis in both process descriptions and computation of similarities while considering the different discourse level, ambiguities, anaphora/coreference in the text, among others. Fig. 2 shows an example of a possible mapping between the textual and the model description of the process.

In the redesign phase, issues detected in the previous phase are amended by a refactoring of the process model, so that the to-be model is produced. NLP-based techniques can also be oriented towards this goal:

- *Fine-tuning of semantic abstraction levels*: Larger sets of process models, so-called, process model collections, are typically hierarchically organized. This means, that process models on higher level provide a rather course-grained view of a process, while process model on a lower level provide a more detailed view. One of the major challenges in this context is to make sure that process models on the same level in the hierarchy also have a comparable level of abstraction. First works that addressed this problem have used rather simple linguistic measures, such as the specificity of individual words, to determine whether two process models provide a comparable level of abstraction (Leopold et al., 2013b). However, a solution that assesses the abstraction level based on the conveyed semantics is still missing.

- *Process description semantic auto-completion*: Process descriptions may be semantically incomplete. Ideally, a process description should be semantically complete before it is used. There are different situations where a process description could be auto-completed. For instance, imagine that the second sentence in the process description of Fig. 2 is: *A member of the sales department must check the order for a customized bike and decide its acceptance*. Semantically, this sentence suggests that there is also the possibility to reject the order of a bike. However, it is not explicit in the textual description what to do if the result of the check is negative. Warning about this incompleteness of the textual description may help improving it, by reducing its ambiguity. A similar situation may occur in the graphical description in the right, i.e., if the arc labeled *No* (and the target gateway) are missing in the process model description in Fig. 2. Some techniques are available for this task (Hornung et al., 2007), but there are still several challenges, mainly on the automation of the problem.

In the controlling phase, apart from the traditional monitoring techniques that focus on checking the performance and conformance requirements, Natural Language Generation techniques can be applied to keep different descriptions of the process available:

- *Transform process model to textual descriptions*: Not all stakeholders are able to understand a process model descriptions like the one shown at the right of Fig. 2. However, virtually everyone can understand the textual description shown at the left. Recent studies on process understandability advocate for the use of several descriptions in order to boost the understanding (Ottensooser et al., 2012). Hence, generating textual descriptions of processes that complement formal ones helps ensuring different stakeholders will have the same expectations for a certain process (Leopold et al., 2014), and NLG is a good fit for this task. There are two main challenges associated with this task. First, we need to properly infer which words from the short process model labels refer to verbs and which words refer to nouns, and due to the shortness of the labels and the lack of a proper sentences structure (e.g., consider the label *Order reservation*) this is a non-trivial task which may require domain knowledge. Second, parallel behavior as well as choices from the process model have to be communicated using a sequence of sentences, without compromising the ability of the reader to comprehend the process semantics.

3.3 Instance Management

Managing the execution of a single process instance (e.g., a particular order for manufacturing a bicycle in the process of Fig. 2, e.g., case 2 of Table 1) is the primary objective of the bottom level of BPM.

There exists a significant amount of research from the last years about *conversational systems* (Mott et al., 2004), ultimately implemented as conversational bots or chatbots. Now let us assume that conversational systems are trained to support the execution of business processes. The ultimate goal would be to allow stakeholders of the process to navigate through it by means of querying a dialog system. For instance, for the process described in Fig. 2, a new worker may have doubts on what to do after a new order is received. A tailored conversation system may come into rescue, by first describing her what are the formal requirements for an order to be accepted. Then, when these requirements have been evaluated by the worker, it will communicate back to the dialogue system the outcome of the evaluation (accepted or not), and the dialogue system will provide the next step correspondingly (in case of the order being accepted, to inform the engineering department; in case of the order being rejected, finish the process). The challenge here is how to build useful conversational systems when a description of the process is available.

In general, BPM tailored conversational systems may incorporate important NLP features like semantic understanding, context resolution, NLG, among others.

4 Applications

4.1 Education

The possibility of transforming a formal process description (e.g., a BPMN like the one on the right in Fig. 2) into a text (Leopold et al., 2014), or vice-versa, as well as the ability to align a textual and a formal

description of a process (Sànchez-Ferreres et al., 2017) opens the door to a range of educational applications for modeling students (e.g. Computer Science or Business students that need to learn to formalize a process): For instance, the model created by a student can be automatically compared to the text given as statement, not only for automatic grading, but also to provide feedback regarding missing/redundant tasks or inconsistent paths in the control flow. Also, an initial model automatically generated from text can be given to the student to complete or correct.

4.2 Troubleshooting

An important application domain for ensuring the correct execution of a process in a organization is through tailored dialogue systems. An example of this is chatbot-aided troubleshooting (Acomb et al., 2007), where artificial agents are used to complement human operators in contact centers. So far, knowledge representation in such chatbots comprises several components like natural language understanding and/or generation, together with a planner that encompasses the possible reactions to provide (Thorne, 2017). The incorporation of process descriptions may help into attaining a more precise dialogue system, so that the planning of the dialogue is done under the constraints of the process, and that the agent can better assist the human in following the appropriate steps.

4.3 Regulations

Non-compliance represents a risk for many organizations. According to a recent study by Thomson Reuters, non-compliance may even represent a possible cause of bankruptcy, also for the so-called "behemoths" in the financial sector (English and Hammond, 2014). Recognizing the risk that is associated with non-compliance, organizations in a wide range of domains are stepping up their spending in order to ensure their compliance with laws, regulations, and procedures. In this context, automated compliance checking techniques that consider the process model and the event log play a crucial role thanks to their ability to automatically identify compliance violations (Accorsi and Stocker, 2012; Van der Aalst et al., 2010). For this reason, numerous approaches have been developed to perform this task (Van der Aalst et al., 2012; Weidlich et al., 2011; Awad et al., 2008). While the majority of such techniques require the allowed process behavior to be specified in formal models, such as process models or business rules, recent advances of NLP in the context of BPM overcome this restriction. In particular, a technique (van der Aa et al., 2018) has been developed that can perform compliance checks on the basis of textual process documentation. This technique employs probabilistic methods in order to deal with the inherent ambiguity of natural language, which can cause uncertainty about the truly allowed process behavior specified in a text.

4.4 Healthcare

Electronic health records (ERHs) contain detailed information of patients. They can and have been used for monitoring adherence to clinical guidelines. There has been several studies on how using these ERHs can lead to improving the management of patients in the healthcare domain. NLP tooling related to ERH processing can also provide a significant step towards improving the efficiency in the treatment of certain diseases (Garvin et al., 2018).

Both clinical guidelines and ERHs can be the source for elicitating formal process descriptions, using a combination of the techniques listed in the Sect. 3.2. Moreover, recent techniques for comparing process models with event logs stored in Healthcare Information Systems (HIS) have proven to be successful in improving the analysis of patients (Mans et al., 2015). We therefore envision the connection of both disciplines: on the one hand, NLP-based techniques to formalize into models ERHs or clinical guidelines, that can be then used to accurately analyze patients with the event data stored in a HIS.

5 Conclusions

In this paper, we have highlighted research directions and prospective applications for NLP in the area of business process management. A more intensive exchange between the two fields has the potential to significantly enhance the tool set of business process management and to fruitfully provide both practical challenges and industrial application scenarios to the NLP community.

Acknowledgements

This work supported by the Spanish Ministerio de Economía y Competitividad, under projects TIN2017-86727-C2-1-R and TIN2016-77820-C3-3-R.

References

Rafael Accorsi and Thomas Stocker. 2012. On the exploitation of process mining for security audits: the conformance checking case. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 1709–1716. ACM.

Kate Acomb, Jonathan Bloom, Krishna Dayanidhi, Phillip Hunter, Peter Krogh, Esther Levin, and Roberto Pieraccini. 2007. Technical support dialog systems: Issues, problems, and solutions. In *Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*, NAACL-HLT-Dialog '07, pages 25–31, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ahmed Awad, Gero Decker, and Mathias Weske. 2008. Efficient compliance checking using bpmn-q and temporal logic. In *International Conference on Business Process Management*, pages 326–341. Springer.

Thomas Baier and Jan Mendling. 2013. Bridging abstraction layers in process mining by automated matching of events and activities. In *International Conference on Business Process Management*, pages 17–32. Springer.

Jörg Becker, Patrick Delfmann, Sebastian Herwig, L. Lis, and Armin Stein. 2009. Formalizing linguistic conventions for conceptual models. In *Conceptual Modeling - ER 2009*, LNCS, pages 70–83. Springer Berlin Heidelberg.

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

Suranjan Chakraborty, Saonee Sarker, and Suprateek Sarker. 2010. An exploration into the process of requirements elicitation: A grounded approach. *J. AIS*, 11(4).

Islay Davies, Peter Green, Michael Rosemann, Marta Indulska, and Stan Gallo. 2006. How do practitioners use conceptual modeling in practice? *Data & Knowledge Engineering*, 58(3):358–380.

Remco M. Dijkman, Marlon Dumas, Boudewijn F. van Dongen, Reina Käärik, and Jan Mendling. 2011. Similarity of business process models: Metrics and evaluation. *Information Systems*, 36(2):498–516.

Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. 2013. *Fundamentals of Business Process Management*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Stacey English and Susannah Hammond. 2014. *The rising costs of non-compliance: from the end of a career to the end of a firm*. Thomson Reuters.

Kathrin Figl and Jan Recker. 2016. Exploring cognitive style and task-specific preferences for process representations. *Requirements Engineering*, 21(1):63–85.

Fabian Friedrich, Jan Mendling, and Frank Puhlmann. 2011. Process model generation from natural language text. In Haralambos Mouratidis and Colette Rolland, editors, *Advanced Information Systems Engineering*, pages 482–496, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hornung Jennifer Garvin, Youngjun Kim, Temple Glenn Gobbel, E. Michael Matheny, Andrew Redd, E. Bruce Bray, Paul Heidenreich, Dan Bolton, Julia Heavirland, Natalie Kelly, Ruth Reeves, Megha Kalsy, Kane Mary Goldstein, and M. Stephane Meystre. 2018. Automating quality measures for heart failure using natural language processing: A descriptive study in the department of veterans affairs. *JMIR Med Inform*, 6(1):e5, Jan.

V. Gruhn and R. Laue. 2011. Detecting Common Errors in Event-Driven Process Chains by Label Analysis. *Enterprise Modelling and Information Systems Architectures*, 6(1):3–15.

Thomas Hornung, Agnes Koschmider, and Andreas Oberweis. 2007. Rule-based autocompletion of business process models. In *CAiSE'07 Forum, Proceedings of the CAiSE'07 Forum at the 19th International Conference on Advanced Information Systems Engineering, Trondheim, Norway, 11-15 June 2007*.

Agnes Koschmider and Emmanuel Blanchard. 2007. User assistance for business process model decomposition. In *Proceedings of the 1st IEEE International Conference on Research Challenges in Information Science*, pages 445–454.

Stefan Krumnow and Gero Decker. 2010. A concept for spreadsheet-based process modeling. In *International Workshop on Business Process Modeling Notation*, pages 63–77. Springer.

Henrik Leopold, Rami-Habib Eid-Sabbagh, Jan Mendling, Leonardo Guerreiro Azevedo, and Fernanda Araujo Baião. 2013a. Detection of naming convention violations in process models for different languages. *Decision Support Systems*, 56:310–325.

Henrik Leopold, Fabian Pittke, and Jan Mendling. 2013b. Towards measuring process model granularity via natural language analysis. In *Business Process Management Workshops - BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers*, pages 417–429.

Henrik Leopold, Jan Mendling, and Artem Polyvyanyy. 2014. Supporting process model validation through natural language generation. *IEEE Trans. Software Eng.*, 40(8):818–840.

Henrik Leopold, Fabian Pittke, and Jan Mendling. 2015. Automatic service derivation from business process model repositories via semantic technology. *Journal of Systems and Software*, 108:134–147.

Henrik Leopold, Han van der Aa, Fabian Pittke, Manuel Raffel, Jan Mendling, and Hajo A Reijers. 2017. Searching textual and model-based process descriptions based on a unified data format. *Software & Systems Modeling*, pages 1–16.

Henrik Leopold. 2013. *Natural language in business process models*. Ph.D. thesis, Springer.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Ronny Mans, Wil M. P. van der Aalst, and Rob J. B. Vanwersch. 2015. *Process Mining in Healthcare - Evaluating and Exploiting Operational Healthcare Processes*. Springer Briefs in Business Process Management. Springer.

Jan Mendling, Henrik Leopold, and Fabian Pittke. 2015. 25 challenges of semantic process modeling. *International Journal of Information Systems and Software Engineering for Big Companies*, 1(1):78–94.

Jan Mendling, Bart Baesens, Abraham Bernstein, and Michael Fellmann. 2017. Challenges of smart business process management: An introduction to the special issue. *Decision Support Systems*, 100:1–5.

Jan Mendling. 2008. *Metrics for process models: empirical foundations of verification, error prediction, and guidelines for correctness*, volume 6. Springer Science & Business Media.

Bradford W. Mott, James C. Lester, and Karl Branting. 2004. Conversational agents. In *The Practical Handbook of Internet Computing.*

Avner Ottensooser, Alan Fekete, Hajo A. Reijers, Jan Mendling, and Con Menictas. 2012. Making sense of business process descriptions: An experimental comparison of graphical and textual notations. *Journal of Systems and Software*, 85(3):596–606.

Lluís Padró and Evgeny Stanilovsky. 2012. Freeling 3.0: Towards wider multilinguality. In *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*, Istanbul, Turkey, May. ELRA.

Keith Thomas Phalp, Jonathan Vincent, and Karl Cox. 2007. Improving the quality of use case descriptions: empirical assessment of writing guidelines. *Software Quality Journal*, 15(4):383–399.

Fabian Pittke, Henrik Leopold, and Jan Mendling. 2014. When language meets language: Anti patterns resulting from mixing natural and modeling language. In *Business Process Management Workshops*, pages 118–129. Springer.

Fabian Pittke, Henrik Leopold, and Jan Mendling. 2015. Automatic detection and resolution of lexical ambiguity in process models. *IEEE Transactions on Software Engineering*, 41(6):526–544.

Jan Recker, Norizan Safrudin, and Michael Rosemann. 2012. How novices design business processes. *Information Systems*, 37(6):557–573.

Hajo A Reijers, Henrik Leopold, and Jan Recker. 2017. Towards a science of checklists. In *Proceedings of the 50th Hawaii International Conference on System Sciences*.

Marcello La Rosa, Marlon Dumas, Reina Uba, and Remco M. Dijkman. 2013. Business process model merging: An approach to business process consolidation. *ACM Transactions on Software Engineering and Methodology*, 22(2):11:1–11:42.

Josep Sànchez-Ferreres, Josep Carmona, and Lluís Padró. 2017. Aligning textual and graphical descriptions of processes through ILP techniques. In *Advanced Information Systems Engineering - 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings*, pages 413–427.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations Session at EACL 2012*, Avignon, France, April. Association for Computational Linguistics.

Oliver Thomas and Michael Fellmann. 2009. Semantic process modeling - design and implementation of an ontology-based representation of business processes. *Business & Information Systems Engineering*, 1(6):438–451.

Camilo Thorne. 2017. Chatbots for troubleshooting: A survey. *Language and Linguistics Compass*, 11(10).

Han Van der Aa, Henrik Leopold, and Hajo A Reijers. 2016. Dealing with behavioral ambiguity in textual process descriptions. In *International Conference on Business Process Management*, pages 271–288. Springer.

Han van der Aa, Henrik Leopold, and Hajo A. Reijers. 2017. Comparing textual descriptions to process models - the automatic detection of inconsistencies. *Inf. Syst.*, 64:447–460.

Han van der Aa, Henrik Leopold, and Hajo A Reijers. 2018. Checking process compliance against natural language specifications using behavioral spaces. *Information Systems*.

Han van der Aa. 2018. *Comparing and Aligning Process Representations*. Ph.D. thesis, Springer.

Wil M P Van der Aalst, Kees M van Hee, Jan Martijn van Werf, and Marc Verdonk. 2010. Auditing 2.0: using process mining to support tomorrow's auditor. *Computer*, 43(3):90–93.

Wil M P Van der Aalst, Arya Adriansyah, and Boudewijn van Dongen. 2012. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):182–192.

Wil M. P. van der Aalst. 2016. *Process Mining: Data Science in Action*. Springer.

Bram Van der Vos, Jon Atle Gulla, and Reind van de Riet. 1997. Verification of conceptual models based on linguistic knowledge. *Data & Knowledge Engineering*, 21(2):147 – 163.

Barbara Weber, Manfred Reichert, Jan Mendling, and Hajo A. Reijers. 2011. Refactoring large process model repositories. *Computers in Industry*, 62(5):467–486.

Matthias Weidlich, Remco M. Dijkman, and Jan Mendling. 2010. The icop framework: Identification of correspondences between process models. In Barbara Pernici, editor, *Advanced Information Systems Engineering, 22nd International Conference, CAiSE 2010, Hammamet, Tunisia, June 7-9, 2010. Proceedings*, volume 6051 of *Lecture Notes in Computer Science*, pages 483–498. Springer.

Matthias Weidlich, Jan Mendling, and Mathias Weske. 2011. Efficient consistency measurement based on behavioral profiles of process models. *IEEE Transactions on Software Engineering*, 37(3):410–429.

Christian Wolter and Christoph Meinel. 2010. An approach to capture authorisation requirements in business processes. *Requirements engineering*, 15(4):359–373.

Novelty Goes Deep. A Deep Neural Solution To Document Level Novelty Detection

Tirthankar Ghosal†, Vignesh Edithal†, Asif Ekbal†, Pushpak Bhattacharyya†,
George Tsatsaronis‡, Srinivasa Satya Sameer Kumar Chivukula‡
†Indian Institute of Technology Patna, Bihar, India
‡Elsevier, Amsterdam, Netherlands
†(tirthankar.pcs16, edithal.cs14, asif, pb)@iitp.ac.in
‡(g.tsatsaronis, s.chivukula)@elsevier.com

Abstract

The rapid growth of documents across the web has necessitated finding means of discarding redundant documents and retaining novel ones. Capturing redundancy is challenging as it may involve investigating at a deep semantic level. Techniques for detecting such semantic redundancy at the document level are scarce. In this work we propose a deep Convolutional Neural Network (CNN) based model to classify a document as novel or redundant with respect to a set of relevant documents already seen by the system. The system is simple and does not require manual feature engineering. Our novel scheme encodes relevant and relative information from both source and target texts to generate an intermediate representation for which we coin the name Relative Document Vector (RDV). The proposed method outperforms the existing benchmark on two document-level novelty detection datasets by a margin of $\sim$5% in terms of accuracy. We further demonstrate the effectiveness of our approach on a standard paraphrase detection dataset where the paraphrased passages closely resembles semantically redundant documents.

1 Introduction

Document-level novelty detection implies the categorization of a document as *novel* if it contains sufficient *new* information with respect to whatever *relevant* is previously known or seen. Existing literature on novelty detection from texts mostly followed the detection of novelty at the sentence level, i.e., to extract the sentences that carry new information with respect to some relevant source texts (c.f. Section 2). *The current work is aimed at automatically classifying an incoming document as* ***novel*** *or* ***non-novel*** *on the basis of documents already seen by the system*. We view the problem as a two-class classification problem in machine learning with the judgment that whether an incoming document bears sufficiently new information to be labeled as novel with respect to a set of source documents. The source document set could be seen as the memory of the reader which stores known information. Document-level novelty detection is a crucial problem and finds application in diverse domains of language processing such as information retrieval, document summarization, predicting impact of scholarly articles, etc. It is a complex problem that comprehends *lexical, syntactic, semantic and pragmatic* levels of texts in conjunction with certain characteristics like *relevance, diversity, relativity, and temporality* (Ghosal et al., 2018). Literature methods for novelty detection based on lexical similarity, divergence features, and information retrieval measures, although proved effective for sentence level but could not address the document-level comprehension needs. Our understanding of the problem led us to an assertion that a deep neural network might be able to extract the text subtleties involved in understanding a document's *novelty*. To the best of our knowledge this is the very first attempt to address document-level novelty detection without the involvement of any hand-crafted features. Our approach achieves significant performance improvement over the existing measures on novelty detection from texts on three different datasets, thus establishing our contention.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2802–2813
Santa Fe, New Mexico, USA, August 20-26, 2018.

1.1 Motivation

Motivation behind the current work stems from *the eternal quest of the inquisitive mind* : **new information**.

- The exponential dump of redundant information in documents across the web, which does not add anything new to the knowledge, yet consumes valuable space and time (of readers) accentuates identifying novel documents and discarding non-novel documents in the current context.
- Plagiarism detection in scholarly articles at the semantic level is another ambitious vision that demands research attention. Methods for document-level novelty detection can aid in the investigation.
- Our survey reveals that in spite of having potential in various applications, document-level novelty detection has not garnered deserved attention from the community. Also it was intriguing to investigate how deep neural architectures could distinguish between novel and non-novel texts.

1.2 Contributions

The principal offerings of the present work could be outlined as :

1. *Proposing an effective deep learning strategy for document-level novelty detection: a first of its kind*
2. *Presenting an effective semantic vector representation to model information from both source and target documents.*

2 Related Works

Research in novelty mining could be traced back to the Topic Detection and Tracking (TDT) (Wayne, 1997) evaluation campaigns where the concern was to detect new events or First Story Detection (FSD) with respect to online news streams. Techniques mostly involved grouping the news stories into clusters and then measuring the belongingness of an incoming story to any of the clusters based on some preset similarity threshold. Some notable contributions from TDT are (Allan et al., 1998; Yang et al., 2002; Allan et al., 2000; Yang et al., 1998).

The task gained prominence in the novelty tracks of Text Retrieval Conferences (TREC) from 2002 to 2004 (Soboroff and Harman, 2005; Harman, 2002; Soboroff and Harman, 2003; Clarke et al., 2004) although the focus was sentence-level novelty detection. The goal of these tracks was to highlight the relevant sentences that contain novel information, given a topic and an ordered list of relevant documents. Some interesting works in TREC were based on the sets of terms (Zhang et al., 2003a; Zhang et al., 2003b), term translations (Collins-Thompson et al., 2002), Principal Component Analysis (PCA) vectors (Ru et al., 2004), Support Vector Machine (SVM) classification (Tomiyama et al., 2004) etc. Similar works relied on named entities (Gabrilovich et al., 2004; Li and Croft, 2005; Zhang and Tsai, 2009), language models (Zhang et al., 2002; Allan et al., 2003), contexts (Schiffman and McKeown, 2005), etc. Next came the novelty sub tracks of Recognizing Textual Entailment-Text Analytics Conference (RTE-TAC) 6 and 7 (Bentivogli et al., 2011) where Textual Entailment was viewed as one close neighbor to sentence level novelty detection.

At the document level an interesting work was carried out by (Yang et al., 2002) via topical classification of on-line document streams and then detecting novelty of documents in each topic exploiting the named entities. Another work by (Zhang et al., 2002) viewed novelty as an opposite characteristic to redundancy and proposed a set of five redundancy measures ranging from the set difference, geometric mean, distributional similarity in order to calculate the novelty of an incoming document with respect to a set of memorized documents. They also presented the first publicly available Associated Press-Wall Street Journal (APWSJ) news dataset for document level novelty detection. (Tsai and Zhang, 2011) applied a document to sentence level framework to calculate the novelty of each sentence of a document which aggregates to detect novelty of the entire document. (Karkali et al., 2013) computed novelty score based on the inverse document frequency scoring function. Another work by (Verheij et al., 2012) presents a comparison study of different novelty detection methods evaluated on news articles

where language model based methods perform better than the cosine similarity based ones. More recently (Dasgupta and Dey, 2016) conducted experiments with information entropy measure to calculate innovativeness of a document. Each of these works evaluated their methods on separate datasets and to the best of our knowledge except APWSJ none of these is publicly available.

Novelty detection is also studied in works related to diversity in information retrieval literature. Idea is to retrieve relevant yet diverse documents in response to user query. The work on Maximal Marginal Relevance (Carbonell and Goldstein, 1998) was the first to explore diversity and relevance for novelty. Some other notable works along this line are (Chandar and Carterette, 2013; Clarke et al., 2011; Clarke et al., 2008).

The work that we present here significantly differs from the existing literature as we provide a deep neural network solution to the problem which does not depend on hand crafted features, and learns the notion of novelty and non-novelty from the data itself.

3 Document Novelty : Challenges and Observations

Deciding the novelty of an entire document or what amount of new information makes a document appear novel are very challenging tasks. Even more challenging since the task inherently exhibits several text characteristics (*relevance, diversity, relativity, temporality*) to be addressed simultaneously. A novel document should be *relevant* to context, should exhibit *diversity* in information content, should have *relatively* more new relevant yet diverse information and is usually deemed as a *temporal* update over the existing knowledge. The difficulty associated with the task is further exaggerated when there are a large number of history documents against which a target document is to be judged for novelty. We propose a possible solution that could address most of these challenges and also reduce the need of manual feature engineering to an extent. We analyze the available benchmark sentence-level novelty detection datasets like TREC/RTE-TAC and come up with certain observations :
(1) Named Entities (NEs) play a vital role in adjudging the *relevance* of a test document against its source document(s).
(2) Semantic features play a crucial role in identifying *redundancy* or *non-novelty*. Lexical level features (such as *n-gram* match) do well when there is a word-by-word match or when two texts share a fair amount of tokens. But it fails in cases where surface form is very different yet conveys the same meaning (*paraphrases*).
(3) Lexical features are sometimes efficient to detect document level new information content. For e.g., if the source document(s) consists of information regarding the facts of a certain accident and the target document reports the police investigation carried out afterwards-although the two documents would share a fair amount of NEs, yet would differ in contextual information content which could be captured to a large extent via lexical level features.

4 Dataset Description

Since our focus is on document-level novelty detection we consider only document level datasets. Existing benchmark datasets on sentence-level novelty detection are not suited to our experimental framework. We also evaluate our approach on the standard Webis Crowd Paraphrase Corpus (Webis CPC) (Burrows et al., 2013) whose inherent semantic richness sets a competitive benchmark for our method.

4.1 APWSJ Corpus

The APWSJ data consists of news articles from Associated Press (AP) and Wall Street Journal (WSJ) corpus covering the same time period (1988 to 1990) and many on the same topics, guaranteeing some redundancy in the document stream. Similar to (Zhang et al., 2002) we use the documents within the designated 33 topics[1] with redundancy judgments by the assessors. The dataset was meant to filter superfluous documents in a retrieval scenario and deliver only the documents having *redundancy score* below a calculated threshold. Documents for each topic were delivered in a chronological manner and

[1] http://www.cs.cmu.edu/ yiz/research/NoveltyData/

the assessors provided two degrees of judgments on the non-novel documents: *absolute redundant* or *somewhat redundant* based on prior documents. The unmarked documents were treated as *novel.*

4.2 Webis CPC

The Webis Crowd Paraphrase Corpus 2011 (Webis-CPC-11) contains 7,859 candidate paraphrases obtained from the Mechanical Turk crowdsourcing. The corpus[2] is made up of 4,067 accepted paraphrases, 3,792 rejected non-paraphrases, and the original texts. For our experiment we assume the original texts as the source document and the corresponding candidate paraphrase/non-paraphrase as the target document. We hypothesize that a paraphrased document would not contain any new information and hence could be treated as *non-novel* whereas a non-paraphrased candidate could be taken as *novel*. Basically we seek to investigate the viability of our proposed method to discover the semantically redundant documents (paraphrases) from this dataset.

4.3 TAP-DLND 1.0

We experiment with this recent benchmark resource for document-level novelty detection (DLND) (Ghosal et al., 2018). The dataset[3] is balanced and consists of 2,736 novel documents and 2,704 non-novel documents. For each novel/non-novel document there are three source documents against which the target document is annotated. The state of *novelty* for each target document is to be measured against those source documents i.e. once the system has already seen the designated source documents for a particular event, it is to judge whether an incoming on-topic document is novel or not. The semantic nature of the dataset makes it an ideal candidate for our experiments.

5 Proposed Model

Having an acceptable benchmark dataset avaialable at our end, instead of handcrafting the similarity/divergence based features,we try to learn feature representations from a target document with respect to the source document(s) using a Convolutional Neural Network (CNN). Our proposed model is based on a recent sentence embedding paradigm proposed by (Conneau et al., 2017). We leverage their idea and create a representation of the *relevant* target document *relative* to the designated source document(s) and call it as the *Relative Document Vector (RDV)*[4]. We then train a Convolutional Neural Network (CNN) with the RDV of the target documents in the similar line of (Kim, 2014), and finally classify a document as *novel* or *non-novel* with respect to its source documents (Figure 1). Although there are document embedding models available, our method is specifically tailored to address the *relativity* and *diversity* criteria which is fundamental to the definition of *novelty*. Here T_1 is the *target* document whose state of *novelty* is to be determined against the source document(s) $S_1, S_2, ...S_M$ i.e. to say the objective is to automatically figure out whether T_1 is *novel* or not once the machine has already seen/scanned $S_1, S_2, ...S_M$. Our model assumes that the documents are relevant to a context. We reserve *Temporality* to be explored in a later work.

5.1 Embedding and Sentence Encoder

The task of *Novelty Detection* requires high-level understanding and reasoning about semantic relationships within texts. Textual Entailment or Natural Language Inference is one such task which exhibits such complex semantic interactions. Following from (Conneau et al., 2017) we therefore employ a sentence encoder based on a bi-directional Long Short Term Memory (LSTM) architecture with max pooling, trained on the large-scale Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015). SNLI entries supposedly captures rich semantic associations between the text pairs (entailment/inference relationships between premise and hypothesis). The output of our sentence encoder is

[2]https://www.uni-weimar.de/en/media/chairs/computer-science-department/webis/data/corpus

[3]http://www.iitp.ac.in/ ai-nlp-ml/resources.html

[4]We call our document vector *relative*, as we desire to encode the relative *new* information of a target document w.r.t. its relevant source document(s)

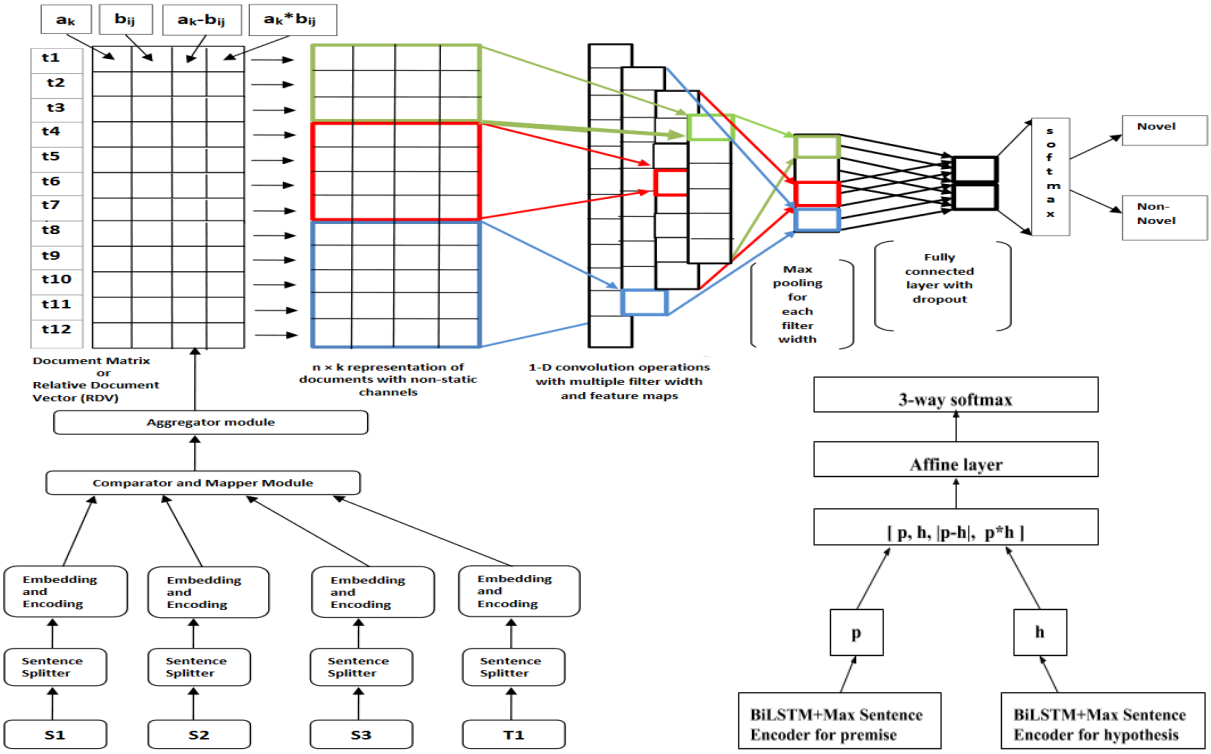


Figure 1: RDV-CNN framework for Novelty Detection. Generic SNLI Training (Conneau et al., 2017). The sentence encoder is trained on SNLI. The RDV-CNN is trained with the respective novelty datasets.

a fixed sized (2048 dimension) sentence embedding of each of the sentences[5] of the input document (source or target).

5.1.1 BiLSTM with max pooling

For a sequence of T words $\{w_t\}_{t-1,...,T}$, a bidirectional LSTM computes a set of T vectors $\{h_t\}_t$. For $t \in [1, ..., T]$, h_t, is the concatenation of a forward LSTM and a backward LSTM that read the sentences in two opposite directions. We combine the varying number of $\{h_t\}_t$ to form a fixed-size vector by selecting the maximum value over each dimension of the hidden units (max pooling).

$$\overrightarrow{h_t} = \overrightarrow{LSTM}_t(w_1, ..., w_T)$$

$$\overleftarrow{h_t} = \overleftarrow{LSTM}_t(w_1, ..., w_T)$$

$$h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$$

5.1.2 Training the sentence encoder

We follow the generic SNLI training scheme of (Conneau et al., 2017). The semantic nature of SNLI corpus makes it a good candidate for learning universal sentence embeddings in a supervised way to capture universally useful features. The training[6] on SNLI is done using the shared sentence encoder that outputs a representation for the premise p and hypothesis h (Figure 1). The representations are further concatenated as:

$$[p, h, |p - h|, p * h]$$

to form a resulting vector, which captures information from both the premise and hypothesis, and is fed into a 3-class classifier[7] (multi-layer perceptron with 1 hidden layer of 512 units) consisting of multiple fully-connected layers culminating into a softmax layer.

[5] split using NLTK

[6] SGD with a learning rate of 0.1 and a weight decay of 0.99 are used. At each epoch, the learning rate is divided by 5 if the development accuracy decreases. Training is done with a mini-batch size of 64. GloVe vectors trained on Common Crawl 840B (https://nlp.stanford.edu/projects/glove/) with 300 dimensions are used as fixed word embeddings.

[7] SNLI has 3 classes of sentence-pair judgments: neutral, contradiction, entailment

5.2 Comparator Module

This module finds the closest sentence in the source document(s) with respect to each of the target sentences. Thus each sentence (t_k) in a target document (T) is mapped to its nearest source sentence (s_{ij}) where i denotes the source document and j signifies the sentence position within that document. The encoder module outputs a fixed-length vector representation a_k for the target sentence t_k and b_{ij} for any source sentence s_{ij}. We define closeness as the maximum of cosine similarity between the vectors a_k and b_{ij}.

$$s_{cosine}(\overrightarrow{a_k}, \overrightarrow{b_{ij}}) = \frac{\overrightarrow{a_k}.\overrightarrow{b_{ij}}}{|\overrightarrow{a_k}||\overrightarrow{b_{ij}}|}$$

Thus we have a mapping relationship for each sentence in the target document with one of the source sentences.

$$b_{ij} \rightarrow a_k$$

where a_k has the max. cosine similarity with b_{ij}.

5.3 Aggregator module

This module aggregates the mappings produced in the comparator module to generate a document matrix. The mapping of a target sentence t_k to its closest source sentence s_{ij} is rendered by constructing a feature vector that captures the relation between the source and the target. This feature vector consists of the concatenation of the two sentence embeddings corresponding to t_k and s_{ij}, their absolute element-wise difference, and their element-wise product (Mou et al., 2016). The first heuristic follows the most standard procedure of the Siamese architecture, while the latter two are certain measures of "similarity" or "closeness". Thus, the *Relative Sentence Vector* (RSV) corresponding to a target sentence t_k is represented as :

$$RSV_k = [a_k, b_{ij}, |a_k - b_{ij}|, a_k * b_{ij}]$$

where comma (,) refers to the column vector concatenation. This representation is inspired from the word embedding studies (Mikolov et al., 2013) where the linear offset of vectors is seen to capture semantic relationships between the two words. (Mou et al., 2016) successfully leveraged this idea for modeling sentence-pair relationships which we alleviate to model documents. Thus for each target sentence t_k we compute the RSV and aggregate them to form the *Relative Document Vector* (RDV) of target document T_j with respect to the source documents(s) S_i. Aggregation is realized as a *slot filling task* to shape the document matrix[8] or RDV of dimension N X $4D$ where N is the number of sentences in a target document (padded when necessary) and D is the sentence embedding dimension produced by the encoder module. *Our rationale behind the RDV-CNN is: The operators:* ***absolute element-wise difference*** *and* ***product*** *would result in such a vector composition for* ***non-novel*** *sentences which would manifest '****closeness****' whereas for* ***novel*** *sentences would manifest '****diversity****'; the aggregation of which would aid in the interpretation of document level* ***novelty*** *or* ***redundancy*** *by a deep neural network. We chose CNN due to its inherent ability to automatically extract features from distinct representations.*

5.4 CNN module

The document matrix or the RDV now becomes the input to a CNN[9] for training and subsequent classification of a target document as *novel* or *non-novel* with respect to its source document(s). The CNN component is similar to the one as used in (Kim, 2014) for sentence classification. The notable difference is in the usage of word embedding. Instead of *word2vec* embeddings we use here the relative sentence embeddings of dimension $4D$ (k^{th} sentence in the document is represented by an embedding vector $RSV_k \in \mathbb{R}^D$). We experiment with the NON-STATICTEXT channel variant (embeddings gets updated during training) of the CNN.

For each possible input channel, a given document is transformed into a tensor of fixed length N (padded

[8] each row in the document matrix is one *relative sentence vector* corresponding to each target sentence t_k

[9] *tanh* as the activation function, filter windows (h) of 3,4,5 with 100 feature maps each, dropout rate (p) of 0.5 on the penultimate layer with a constraint on l_2-norms of the weight vectors. $ADADELTA$ (Zeiler, 2012) optimizer with a learning rate set to 0.1. Training via Stochastic Gradient Descent (SGD). Input batch size : 50, number of training iterations (epochs): 250. 10% of the training data for validation.

with *zero-tensors* wherever necessary to tackle variable sentence lengths) by concatenating the relative sentence embeddings.

$$RSV_{1:N} = RSV_1 \oplus RSV_2 \oplus RSV_3 \oplus ..... \oplus RSV_N$$

where $\oplus$ is the concatenation operator. To extract *local features*[10], convolution operation is applied. Convolution operation involves a *filter*, $W \in \mathbb{R}^{HD}$, which is convolved with a window of H embeddings to produce a local feature for the H target sentences. A local feature, c_k is generated from a window of embeddings $RSV_{k:k+H-1}$ by applying a non-linear function (such as hyperbolic tangent) over the convoluted output. Mathematically,

$$c_k = f(W.RSV_{k:k+H-1} + b)$$

where $b \in \mathbb{R}$ is the *bias* and f is the non-linear function. This operation is applied to each possible window of H target sentences to produce a feature map (c) for the window size H.

$$c = [c_1, c_2, c_3, ..., c_{N-H+1}]$$

A global feature is then obtained by applying *max-pooling* operation (Collobert et al., 2011) over the feature map. The idea behind *max-pooling* is to capture the most important feature-one with the highest value -for each feature map. We describe the process by which one feature is extracted from one filter (red bordered portions in Figure 1 illustrate the case of $H = 4$). The model uses multiple filters for each filter size to obtain multiple features representing the text. These features form the penultimate layer and are passed to a fully connected feed forward layer (with number of hidden units set to 100) followed by a *SoftMax* layer whose output is the probability distribution over the labels (*novel* or *non-novel*).

6 Results and Discussion

We proceed with the intuition: *redundancy* recognition would eventually lead us to *novelty* detection. The three datasets we use represent the most comprehensive resources that are publicly available and could be effectively used for document-level novelty detection.

6.1 On APWSJ: a document-level novelty detection dataset

We take upon the results reported by (Zhang et al., 2002) on APWSJ and use similar metrics to report the performance of our approach. Table 1 exhibits the effect of different redundancy measures (taken from (Zhang et al., 2002)) on APWSJ considering both *absolutely redundant* and *somewhat redundant* documents as redundant or non-novel. Our RDV-CNN approach delivers comparable performance in a 10-*fold* cross-validation classification setting. Although the system suffers in identifying the *redundant* documents but succeeds to minimize the errors committed. It signifies the affinity of our architecture towards detecting *novel* documents. It is to be noted here that (Zhang et al., 2002) conducted their experiments in an information filtering scenario using some thresholding scheme, where the **redundant or non-novel** documents were to be filtered from being delivered (*Not Delivered*). Only the *novel* documents were to be *Delivered* by the retrieval system. No learning was involved. Even APWSJ corpus was developed to support this Information Retrieval (IR) perspective of retrieving novel documents. Hence, there is a huge class imbalance. The presence of a relatively less number of *non-novel* documents in APWSJ (only 9.07%) and also that *somewhat redundant* documents are considered as *redundant*[11] in our experiment may have hindered the learning of *redundant* patterns by our system. However, the superiority of our approach is established when we experiment with two other balanced datasets that closely approximates the semantic level redundancy we aim to capture.

6.2 On Webis-CPC-11: a corpus for paraphrase detection (simulating non-novelty)

As stated earlier, we deem paraphrase detection as one close simulation of semantic level redundancy (non-novelty) detection. With this view we subject our model to detect passage-level paraphrase pairs

[10] features specific to a region in case of images or window of target sentences

[11] as originally considered in (Zhang et al., 2002). We replicate the same experimental conditions for fair comparison.

Measure	Recall	Precision	Mistake
Set Distance	0.52	0.44	43.5%
Cosine Distance	0.62	0.63	28.1%
LM:Shrinkage	0.80	0.45	44.3%
LM:Dirichlet Prior	0.76	0.47	42.4%
LM:Mixture Model	0.56	0.67	27.4%
Proposed Approach (RDV-CNN)	0.58	**0.76**	**22.9%**

Table 1: Results for Redundant class on APWSJ, $LM \rightarrow$ Language Model, $Mistake \rightarrow$100-Accuracy. Except for RDV-CNN, all other numbers are taken from (Zhang et al., 2002)

from Webis-CPC-11. We investigate similar evaluation systems for novelty detection as carried out in (Ghosal et al., 2018). The three novelty detection measures (Set Difference, Geometric Distance, Language Model), originally formulated by (Zhang et al., 2002) and another based on *Inverse Document Frequency (IDF)* by (Karkali et al., 2013) are our benchmarks for evaluation. Instead of setting a fixed threshold[12] as in these works we train a Logistic Regression (LR) classifier based on those measures to automatically determine the decision boundary.

Baseline 1: As a baseline we take *state-of-the-art* document embedding (*Paragraph Vector*) technique by (Le and Mikolov, 2014) for document representation; concatenate the source and target document vectors and pass it to LR.

Baseline 2: Both the target and source sentence encodings[13] are passed to separate BiLSTM layers; the two resultant vectors are concatenated and passed to a Multi Layered Perceptron (MLP) with hidden dim of 2048 followed by classification via softmax.

Table 2 clearly shows that our RDV-CNN outperforms all the measures and baselines and maintains a significant supremacy to detect semantic-level redundant passages.

Evaluation System	Description	P	R	F_1	A
Baseline 1	Paragraph Vector+LR	0.72	0.58	0.64	66.94%
Baseline 2	BiLSTM+MLP	0.71	0.73	0.72	70.91%
Novelty Measure 1(Zhang et al., 2002)	Set Difference + LR	0.71	0.52	0.60	64.75%
Novelty Measure 2(Zhang et al., 2002)	Geometric Distance + LR	0.69	0.75	0.71	70.23%
Novelty Measure 3(Zhang et al., 2002)	LM: Dirichlet Prior + LR	0.74	0.77	0.75	74.34%
Novelty Measure 4(Karkali et al., 2013)	IDF + LR	0.65	0.55	0.59	61.72%
Proposed Approach	**RDV-CNN**	**0.75**	**0.84**	**0.80**	78.02%

Table 2: Results for Paraphrase class on Webis-CPC (in %), IDF$\rightarrow$ Inverse Document Frequency, LR$\rightarrow$ Logistic Regression

6.3 On TAP-DLND 1.0: a corpus for document-level novelty detection

We experiment with the recently published TAP-DLND 1.0 corpus for our *document level novelty detection* experiments. TAP-DLND 1.0 is well balanced and consists of a fair share of different levels (lexical as well as semantic) of textual views for both novel and non-novel documents. Since our objective here is to identify both novel and non-novel documents we report results for each class. We take the same baseline as we use for Webis-CPC in the previous section. Our approach outperforms the popular semantic document embedding technique *Paragraph Vector* and another deep neural baseline (BiLSTM+MLP) by a considerable margin(12% and 6% in terms of accuracy, respectively). We also re-execute the other novelty detection measures by (Zhang et al., 2002) (Set Difference, Geometric Distance, Language Model) and (Karkali et al., 2013) (IDF) on TAP-DLND 1.0 and report the results. Our RDV-CNN even surpasses

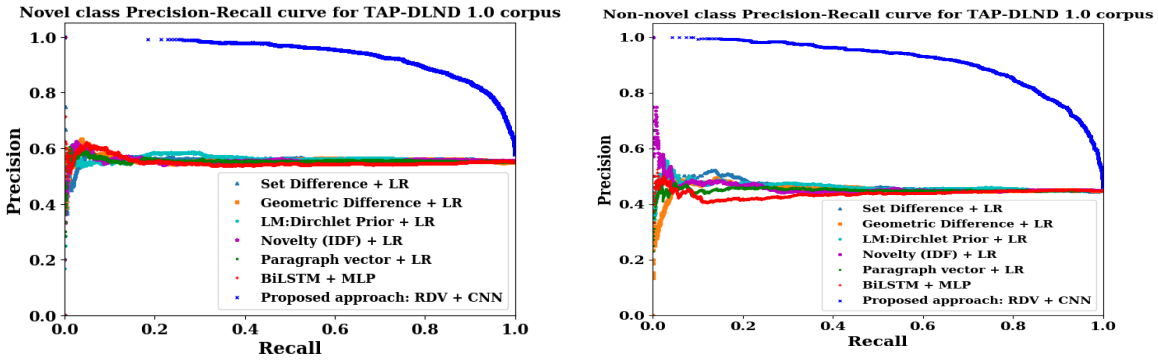
[12] the weak thresholding algorithm reported in these works yield poor results

[13] trained on SNLI as in section 5.1.2

Evaluation System	Description	P(N)	R(N)	F_1(N)	P(NN)	R(NN)	F_1(NN)	A
Baseline 1	Paragraph Vector+LR	0.75	0.75	0.75	0.69	0.69	0.69	72.81%
Baseline 2	BiLSTM+MLP	0.78	0.84	0.80	0.78	0.71	0.74	78.57%
Novelty Measure 1 (Zhang et al., 2002)	Set Difference+LR	0.74	0.71	0.72	0.72	0.74	0.73	73.21%
Novelty Measure 2 (Zhang et al., 2002)	Geometric Distance+LR	0.65	0.84	0.73	0.84	0.55	0.66	69.84%
Novelty Measure 3 (Zhang et al., 2002)	LM:Dirichlet Prior+LR	0.73	0.74	0.74	0.74	0.72	0.73	73.62%
Novelty Measure 4 (Karkali et al., 2013)	Novelty (IDF)+LR	0.52	0.92	0.66	0.66	0.16	0.25	54.26%
(Ghosal et al., 2018)	Supervised features	0.77	0.82	0.79	0.80	0.76	0.78	79.27%
Proposed Approach	**RDV-CNN**	**0.86**	0.87	**0.86**	**0.84**	**0.83**	**0.83**	**84.53%**

Table 3: Results on TAP-DLND 1.0 , P→ Precision, R→ Recall, A→ Accuracy, R→ Recall, MLP→ Multi Layer Perceptron, N→ Novel, NN→ Non-Novel, IDF→ Inverse Document Frequency

the current benchmark (Ghosal et al., 2018) on TAP-DLND 1.0 by a considerable margin in identifying both novel and non-novel documents. Figure 2 shows the consistency and edge of our approach over other measures. This behavior we attribute to our customized architecture of mapping each target sen-



(a) PRC for Novel class

(b) PRC for Non-Novel Class

Figure 2: Precision-Recall Curve for TAP-DLND 1.0

tence to the nearest source sentence to produce a *relative* document representation being input to a CNN. The CNN then extracts the features from the *relative* document representation and finally classifies the incoming document. Results signify that our network is able to learn the complex semantic interactions between source and target information necessary to conclude upon the state of novelty of a document. It is to be noted that except (Ghosal et al., 2018) and our baselines, all other measures that we consider for comparison were developed with an information retrieval perspective. The *p-values* for F_1 score produced by 20 runs of our system against the baseline are less than 0.05 and hence the improvement is statistically significant and unlikely to be observed by chance in 95% confidence interval.

6.4 Observations and Analysis

We scrutinize the results and arrive to the following observations:
(1) The RDV-CNN customized architecture of mapping each target sentence to the nearest source sentence for producing a *relative* document representation tackles the *relativity* criterion required for the problem. It facilitates CNN to extract the relevant features from the *relative* document representation which accounts for the better performance across the two datasets.
(2) Lexical approaches perform closer to our approach in detecting paraphrase pairs. This is due to the higher number of Named Entities (NEs) shared between those literary texts in Webis-CPC-11.
(3) Poor recall for non-novel class by IDF measure (Karkali et al., 2013) is due to the existence of many

The associated codes of this paper is available at: https://github.com/edithal-14/A-Deep-Neural-Solution-To-Document-Level-Novelty-Detection-COLING-2018-

new entity terms in the target documents for TAP-DLND 1.0.
(4) Lexical overlap based measures performs poorly in identifying *non-novel* documents in TAP-DLND 1.0. This behavior approves our discussion in Section 3 that we need semantic flair to address *non-novelty*. RDV-CNN bridges the gap.
(5) The deep baselines did not appear as promising to capture the semantic interactions between source and target sentences as did RDV-CNN (Section 5.3).
(6) We thoroughly examined our gold and predicted class labels. Errors committed by our system are mostly due to presence of multiple source premise sentence for a target sentence. Our RDV-CNN could capture only one premise for a target sentence and map them. We intend to accommodate multiple premises in RDV definition in our future work.
(7) The universal sentence encoding generated by BiLSTM trained on SNLI are sometimes unable to capture the complex semantic interactions between source and target sentences. This mostly occurred for new NEs and out-of-vocabulary (OOV) words.

7 Conclusion and Future work

In this work we put forward a deep learning solution for document-level novelty detection. Our deep neural network succeeds in extracting the text subtleties essential for this complex task. The system displays comparable potential on an existing resource, significant performance gain over the high-level *paraphrase detection* task and outperforms the state-of-the-art on an apt dataset. Analysis of results validates our claim that semantic features play a vital role in identifying *semantic level redundancy* aka *non-novelty* whereas relevant lexical divergence is a good indicator of *novelty*. However, we agree that given a large number of documents as source our architecture would be computationally expensive which we intend to mitigate. Also we desire to incorporate *relevance* judgment as an essential component our architecture. Our future agenda includes investigating the role of attention mechanism in creating the *Relative Document Vector* as an intermediate representation of target documents for document-level novelty detection.

Acknowledgements

The first author, Tirthankar Ghosal, acknowledges Visvesvaraya PhD Scheme for Electronics and IT, an initiative of Ministry of Electronics and Information Technology (MeitY), Government of India for fellowship support. Asif Ekbal acknowledges Young Faculty Research Fellowship (YFRF), supported by Visvesvaraya PhD scheme for Electronics and IT, Ministry of Electronics and Information Technology (MeitY), Government of India, being implemented by Digital India Corporation (formerly Media Lab Asia). We thank the anonymous reviewers for their valuable feedback and Prof. Donia Scott, University of Sussex for her advice in the Writing Mentoring Program as part of COLING 2018. We also thank Elsevier Center of Excellence for Natural Language Processing, Indian Institute of Technology Patna for adequate help and support to carry out this research.

References

James Allan, Ron Papka, and Victor Lavrenko. 1998. On-line new event detection and tracking. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 37–45. ACM.

James Allan, Victor Lavrenko, and Hubert Jin. 2000. First story detection in tdt is hard. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 374–381. ACM.

James Allan, Courtney Wade, and Alvaro Bolivar. 2003. Retrieval and novelty detection at the sentence level. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 314–321. ACM.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2011. The seventh PASCAL recognizing textual entailment challenge. In *Proceedings of the Fourth Text Analysis Conference, TAC 2011, Gaithersburg, Maryland, USA, November 14-15, 2011*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 632–642.

Steven Burrows, Martin Potthast, and Benno Stein. 2013. Paraphrase Acquisition via Crowdsourcing and Machine Learning. *Transactions on Intelligent Systems and Technology (ACM TIST)*, 4(3):43:1–43:21, June.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336. ACM.

Praveen Chandar and Ben Carterette. 2013. Preference based evaluation measures for novelty and diversity. In *SIGIR*.

Charles LA Clarke, Nick Craswell, and Ian Soboroff. 2004. Overview of the trec 2004 terabyte track. In *TREC*, volume 4, page 74.

Charles L. A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. 2008. Novelty and diversity in information retrieval evaluation. In *SIGIR*.

Charles L. A. Clarke, Nick Craswell, Ian Soboroff, and Azin Ashkan. 2011. A comparative analysis of cascade measures for novelty and diversity. In *WSDM*.

Kevyn Collins-Thompson, Paul Ogilvie, Yi Zhang, and Jamie Callan. 2002. Information filtering, novelty detection, and named-page finding. In *TREC*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Tirthankar Dasgupta and Lipika Dey. 2016. Automatic scoring for innovativeness of textual ideas. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*.

Evgeniy Gabrilovich, Susan Dumais, and Eric Horvitz. 2004. Newsjunkie: providing personalized newsfeeds via analysis of information novelty. In *Proceedings of the 13th international conference on World Wide Web*, pages 482–490. ACM.

Tirthankar Ghosal, Amitra Salam, Swati Tiwary, Asif Ekbal, and Pushpak Bhattacharyya. 2018. TAP-DLND 1.0 : A corpus for document level novelty detection. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018.*

Donna Harman. 2002. Overview of the TREC 2002 novelty track. In *Proceedings of The Eleventh Text REtrieval Conference, TREC 2002, Gaithersburg, Maryland, USA, November 19-22, 2002*.

Margarita Karkali, François Rousseau, Alexandros Ntoulas, and Michalis Vazirgiannis. 2013. Efficient online novelty detection in news streams. In *WISE (1)*, pages 57–71.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Xiaoyan Li and W Bruce Croft. 2005. Novelty detection based on sentence level patterns. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 744–751. ACM.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, volume 13, pages 746–751.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*.

Liyun Ru, Le Zhao, Min Zhang, and Shaoping Ma. 2004. Improved feature selection and redundance computing-thuir at trec 2004 novelty track. In *TREC*.

Barry Schiffman and Kathleen R McKeown. 2005. Context and learning in novelty detection. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 716–723. Association for Computational Linguistics.

Ian Soboroff and Donna Harman. 2003. Overview of the TREC 2003 novelty track. In *Proceedings of The Twelfth Text REtrieval Conference, TREC 2003, Gaithersburg, Maryland, USA, November 18-21, 2003*, pages 38–53.

Ian Soboroff and Donna Harman. 2005. Novelty detection: the trec experience. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 105–112. Association for Computational Linguistics.

Tomoe Tomiyama, Kosuke Karoji, Takeshi Kondo, Yuichi Kakuta, Tomohiro Takagi, Akiko Aizawa, and Teruhito Kanazawa. 2004. Meiji university web, novelty and genomic track experiments. In *TREC*.

Flora S Tsai and Yi Zhang. 2011. D2s: Document-to-sentence framework for novelty detection. *Knowledge and information systems*, 29(2):419–433.

Arnout Verheij, Allard Kleijn, Flavius Frasincar, and Frederik Hogenboom. 2012. A comparison study for novelty control mechanisms applied to web news stories. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2012 IEEE/WIC/ACM International Conferences on*, volume 1, pages 431–436. IEEE.

Charles L Wayne. 1997. Topic detection and tracking (tdt). In *Workshop held at the University of Maryland on*, volume 27, page 28. Citeseer.

Yiming Yang, Tom Pierce, and Jaime Carbonell. 1998. A study of retrospective and on-line event detection. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 28–36. ACM.

Yiming Yang, Jian Zhang, Jaime Carbonell, and Chun Jin. 2002. Topic-conditioned novelty detection. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 688–693. ACM.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Yi Zhang and Flora S Tsai. 2009. Combining named entities and tags for novel sentence detection. In *Proceedings of the WSDM'09 Workshop on Exploiting Semantic Annotations in Information Retrieval*, pages 30–34. ACM.

Yi Zhang, Jamie Callan, and Thomas Minka. 2002. Novelty and redundancy detection in adaptive filtering. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 81–88. ACM.

Min Zhang, Chuan Lin, Yiqun Liu, Leo Zhao, and Shaoping Ma. 2003a. THUIR at TREC 2003: Novelty, robust and web. In *Proceedings of The Twelfth Text REtrieval Conference, TREC 2003, Gaithersburg, Maryland, USA, November 18-21, 2003*, pages 556–567.

Min Zhang, Ruihua Song, Chuan Lin, Shaoping Ma, Zhe Jiang, Yijiang Jin, Yiqun Liu, Le Zhao, and S Ma. 2003b. Expansion-based technologies in finding relevant and new information: Thu trec 2002: Novelty track experiments. *NIST SPECIAL PUBLICATION SP*, (251):586–590.

What constitutes "style" in authorship attribution?

Kalaivani Sundararajan
Florida Institute for Cybersecurity
University of Florida
kalaivani.s@ufl.edu

Damon L. Woodard
Florida Institute for Cybersecurity
University of Florida
dwoodard@ece.ufl.edu

Abstract

Authorship attribution typically uses all information representing both content and style whereas attribution based only on stylistic aspects may be robust in cross-domain settings. This paper analyzes different linguistic aspects that may help represent style. Specifically, we study the role of syntax and lexical words (nouns, verbs, adjectives and adverbs) in representing style. We use a purely syntactic language model to study the significance of sentence structures in both single-domain and cross-domain attribution, *i.e.* cross-topic and cross-genre attribution. We show that syntax may be helpful for cross-genre attribution while cross-topic attribution and single-domain may benefit from additional lexical information. Further, pure syntactic models may not be effective by themselves and need to be used in combination with other robust models. To study the role of word choice, we perform attribution by masking all words or specific topic words corresponding to nouns, verbs, adjectives and adverbs. Using a single-domain dataset, IMDB1M reviews, we demonstrate the heavy influence of common nouns and proper nouns in attribution, thereby highlighting topic interference. Using cross-domain Guardian10 dataset, we show that some common nouns, verbs, adjectives and adverbs may help with stylometric attribution as demonstrated by masking topic words corresponding to these parts-of-speech. As expected, it was observed that proper nouns are heavily influenced by content and cross-domain attribution will benefit from completely masking them.

1 Introduction

Authorship attribution attributes a text sample to a person based on either content or their writing style. On the other hand, stylometry involves authorship attribution only based on writing style and needs to be independent of topic or genre. Though authorship attribution has various applications, some important ones are security-related. eg. cybercrime investigation, digital forensics, countering identity deception in social networks, etc. In today's connected world, most of our cyber-interactions are text-based on various platforms like messages, emails, blogs, tweets, forum posts etc. With rampant cybercrime attacks, it is important to perform robust authorship attribution or verification across these platforms. This calls for stylometric attribution approaches that model style and work well in cross-domain scenarios. Cross-domain scenarios include both cross-topic (same genre but different topics) and cross-genre (different genres) scenarios.

Various authorship attribution approaches have been surveyed in literature (Juola and others, 2008; Stamatatos, 2009; Neal et al., 2017). However, only some of these approaches focus specifically on cross-domain attribution. Function words have been touted as content-independent and hence reliable as a style marker (Juola and others, 2008). Syntactic information, both shallow (Luyckx and Daelemans, 2008) and deep (Baayen et al., 1996; Raghavan et al., 2010; Feng et al., 2012; Fuller et al., 2014; Björklund and Zechner, 2017), have also been proposed for ensuring topic or genre independence. Few other approaches, perform preprocessing to prevent topic/genre dependency when using lexical or character features (Stamatatos, 2012; Sapkota et al., 2014; Markov et al., 2017; Stamatatos, 2018).

Proceedings of the 27th International Conference on Computational Linguistics, pages 2814–2822
Santa Fe, New Mexico, USA, August 20-26, 2018.

In this paper, we study the role of different linguistic aspects on style, specifically sentence syntax and word choice. We use a language model to represent syntax and words as against using vector representations with machine learning approaches. We do so for two reasons: i) Vector representation of syntax has not been effective in past efforts (Stamatatos, 2009), ii) Vector representation of words or character n-grams mostly consist of function words and may not be helpful to study the role of lexical parts-of-speech (POS).

To study the role of syntax, we perform authorship attribution using a purely syntactic language model. The syntactic model is constructed using the Probabilistic Context-free grammars (PCFG) obtained from an author's training data. While the approach is similar to other syntactic approaches (Raghavan et al., 2010), we keep our model purely syntactic by removing any lexical information. We also add context to the rewrite rules by vertical and horizontal Markovization. To handle previously unseen rules, we also incorporate smoothing where the language model can back-off to lower order models.

To study the role of word choice, we perform attribution by masking out all words or specific topic words corresponding to different lexical POS. We do so because character n-gram based approaches have largely outperformed function word based approaches (Kestemont, 2014) indicating that lexical words may also help with authorship attribution. This analysis would help understand which of these lexical words may help represent style. Other approaches (Stamatatos, 2018) also mask out infrequent words (mostly lexical words) but we perform an in-depth analysis as to which of these lexical words are useful for representing style.

In this paper, we attempt to answer the following questions:

- Does sentence structure or syntax help with stylometric attribution in cross-domain settings?
- What are effects of words corresponding to different lexical POS on cross-domain attribution?
- Does masking topic words corresponding to various lexical POS help with stylometric attribution in cross-domain settings?

2 Methodology

In this section, we explain the approaches taken to answer the above questions pertaining to stylometric attribution. This could help us come up with better style representations that would work well in cross-domain scenarios.

2.1 Role of syntax

We use a purely syntactic language model to study the role of sentence structure or syntax in cross-domain stylometric attribution. A syntactic language model is obtained by constructing the probabilistic context-free grammar (PCFG) for each author using the constituency parse trees of sentences in their training posts. While some approaches (Raghavan et al., 2010) use the rewrite rules directly to construct PCFGs, we apply both vertical and horizontal Markovization (Klein and Manning, 2003) to the parse trees before constructing PCFGs. This helps to incorporate some context into the rewrite rules and improves parsing accuracy. To keep the approach purely syntactic, we remove the leaf nodes which contain the sentence words (Fuller et al., 2014; Björklund and Zechner, 2017). A test sample is attributed to the author whose PCFG yields the highest likelihood score. In order to account for unseen rules during test time, we also incorporate smoothing that allows the model to backoff to lower order syntactic language models. The rewrite rules of a sample sentence corresponding to different Markovization orders used in our model are shown in Figure 1.

We compare the syntactic language model with an analogous character language model (Teahan and Harper, 2003). This approach uses a lossless text compression method called Prediction by Partial Matching (PPM) (Cleary and Witten, 1984). With PPM, individual characters are encoded using the context provided by the preceding characters thus representing each author A using a separate language model p_A. To attribute an unknown sample $\mathbf{u}$ of length L, we compute its cross-entropy with respect to an

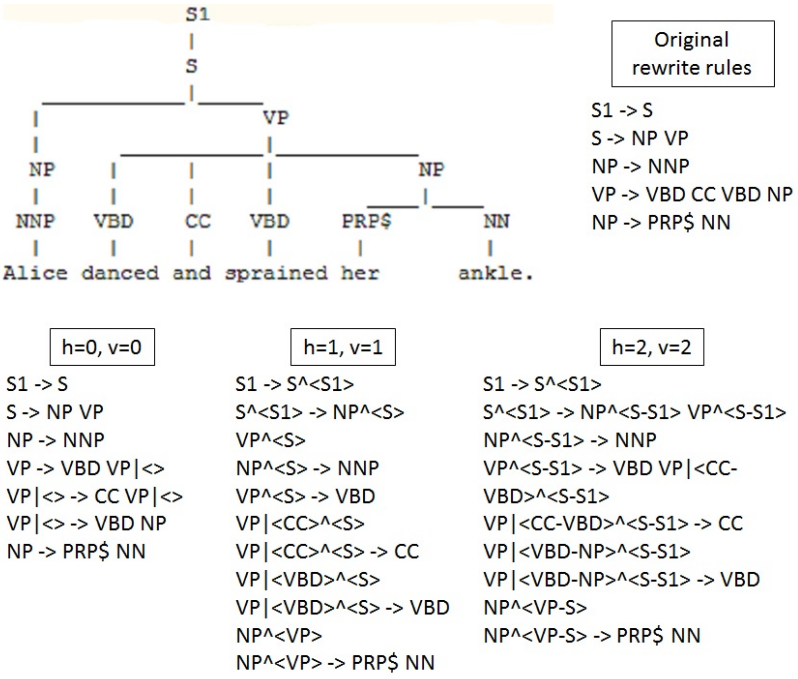


Figure 1: Parse tree of a sample sentence and rewrite rules under different orders of vertical (v) and horizontal (h) Markovization. The leaf nodes are excluded from the rewrite rules.

author model p_A as

$$H(p_A, u) = -\frac{1}{L}\, log_2\, p_A(u)$$
$$= -\frac{1}{L}\sum_{i=1}^{L} log_2\, p_A(x_i|context_i)$$

where $context_i$ of character x_i is $x_1, x_2, ..., x_{i-1}$. The unknown sample is attributed to the author whose model yields least cross-entropy. For computational efficiency, the context is typically truncated to preceding n-1 characters i.e. $x_{i-n}, ..x_{i-1}$.

2.2 Role of word choice

Besides syntax, the choice of words used by a person also plays an important role in stylometric attribution. These words may have a primary purpose of being lexical or grammatical. Lexical words have meaning by themselves and are mostly content-specific. Such words typically include nouns, verbs, adjectives and adverbs. On the other hand, grammatical words do not have any meaning by themselves and specify the relationships between lexical words. Hence, they are independent of content and typically include determiners, prepositions, pronouns, etc. Conventionally, grammatical words, especially function words, have been proposed for stylometric authorship attribution since they are independent of content. However, character n-gram based approaches have largely outperformed function word based approaches (Kestemont, 2014) indicating that some lexical words may also help with authorship attribution. In order to understand which lexical words may help with stylometric attribution, we study the effects of masking all lexical words and certain topic words corresponding to different POS on authorship attribution.

2.2.1 Role of lexical POS

To analyze the effects of different lexical POS on stylometric attribution, we preprocess the posts to replace words corresponding to different POS with a predefined string $<T>$. We use the set of Penn TreeBank POS tags in our experiments. The following effects are analyzed using a character language model (Teahan and Harper, 2003) for both single-domain and cross-domain datasets:

- **orig:** This utilizes the original posts and are used for benchmarking.
- **no_NNP:** This utilizes posts in which all proper nouns (NNP, NNPS) are replaced by $<T>$.
- **no_NN:** This utilizes posts in which all common nouns (NN, NNS) are replaced by $<T>$.
- **no_VB:** This utilizes posts in which all verbs (VB, VBD, VBG, VBN, VBP, VBZ) except function words are replaced by $<T>$.
- **no_ADJ:** This utilizes posts in which all adjectives (JJ, JJR, JJS) are replaced by $<T>$.
- **no_ADV:** This utilizes posts in which all adverbs (RB, RBR, RBS) are replaced by $<T>$.

2.2.2 Role of topic words

The approach in Section 2.2.1 assumes that all words corresponding to a lexical POS may be content-specific. This may not be the case as some of these words may actually help with stylometric attribution. Hence, in this section, we analyze the effects of masking only the topic words corresponding to different lexical POS on attribution. We use Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to obtain the topic words for each lexical POS. Since each author may focus on different topics, we perform LDA for each author using their training posts. The topic words of all authors are then collated and used for masking. We analyze the same effects as Section 2.2.1 - *orig, no_topic_NNP, no_topic_NN, no_topic_VB, no_topic_ADJ* and *no_topic_ADV*.

3 Datasets

We use two datasets for our experiments as follows:

- **IMDB1M reviews** (Seroussi et al., 2011): This single-domain dataset consists of 204,809 posts and 66,816 reviews written by 22,116 users. For our experiments, we use only a subset of 674 users with more than 50 posts per person. The chosen data subset contains 160,482 posts across 674 users. On an average, the dataset consists of 238 posts/author, 347 characters/post, 73 words/post and 4 sentences/post.
- **Guardian10 corpus** (Stamatatos, 2012): This cross-domain dataset consists of opinion articles and book reviews written by 13 authors and up to 10 posts per topic/genre. On an average, this dataset consists of 34 posts/author, 6234 characters/post, 1202 words/post and 52 sentences/post. Cross-topic experiments are performed only using the opinion articles which are on four different topics - Politics, UK, World and Society. Cross-genre experiments involve training author models using opinion articles on all four topics and evaluating authorship attribution performance on book reviews.

4 Results

In this section, we report the results of our analysis detailed in Section 2.

4.1 Role of syntax

We use the BLLIP parser (Charniak and Johnson, 2005) trained on Wall Street Journal (WSJ) corpus to treebank an author's training data. We use vertical and horizontal Markovization order of two *(h=2, v=2)* since higher orders did not seem to improve performance. The leaf nodes are removed from the parse trees to keep the analysis purely syntactic. To test stylometric attribution, we evaluate the syntactic language model (Syntactic LM) on both single-domain IMDB1M and cross-domain Guardian10 datasets. We repeat these experiments using the syntactic language model with lexical information in its leaf nodes (Syntactic LM + Lexical) and also using the character language model (Character LM) (Teahan and Harper, 2003) for comparison. The single-domain and cross-domain attribution performances are reported in Tables 1 and 2 respectively.

Table 1: Single-domain performance on IMDB1M using syntactic language model

Scenarios	Precision(%)	Recall(%)	F-score(%)	Accuracy(%)
Syntactic LM	30.96	27.71	28.28	27.71
Syntactic LM + Lexical	44.31	43.52	42.26	43.53
Character LM	64.41	64.77	63.42	64.77

Table 2: Cross-domain performance on Guardian10 using syntactic language model

Domain	Scenarios	Precision(%)	Recall(%)	F-score(%)	Accuracy(%)
Cross-topic	Syntactic LM	21.85	22.22	18.40	24.14
	Syntactic LM + Lexical	27.30	25.15	21.27	26.56
	Character LM	76.89	67.41	67.02	70.41
Cross-genre	Syntactic LM	20.44	26.96	21.90	33.33
	Syntactic LM + Lexical	18.92	23.15	18.99	28.57
	Character LM	72.32	56.90	59.79	69.84

4.1.1 Discussion

We studied the role of sentence structure or syntax in stylometric attribution using a purely syntactic language model. For single-domain IMDB1M dataset, it can be inferred from Table 1 that using a purely syntactic model (*Syntactic LM*) seems to be ineffective as compared to using both syntax and lexical information (*Syntactic LM + Lexical*). This reaffirms the fact that single-domain attribution benefits largely from lexical information some of which may be topic-dependent. For cross-domain settings, it can be inferred from Table 2 that purely syntactic attribution (*Syntactic LM*) may be more helpful for cross-genre data than cross-topic data. Adding lexical information to purely syntactic model (*Syntactic LM + Lexical*) seems to improve cross-topic attribution while it deteriorates cross-genre attribution.

Nevertheless, purely syntactic models seem to perform poorly compared to character language models as is evident from in Tables 1 and 2. Character n-gram based approaches have been highly successful in authorship attribution because they represent character, lexical and syntactic information to varying extents (Luyckx, 2011; Kestemont, 2014) and are robust to morphological variations in language use (Sapkota et al., 2015). However, from a purely quantitative perspective, their success can be attributed to the fact that character n-grams have much more data points than function words or syntactic rules thereby yielding superior performance (Kestemont, 2014). Character n-grams span the same word or word combinations multiple times depending on the n-gram order and hence amplify the presence of possibly unique word/phrase choices by an author. PCFG-based syntactic language models do not have this advantage as the set of all possible syntactic rewrite rules is much lesser than the set of all possible character n-grams. Hence, the low performance of purely syntactic models does not necessarily imply that they are not useful in representing style. One can use these in combination with character language models to improve cross-domain performance.

4.2 Role of word choice

To analyze the role of word choice in stylometric attribution, we perform experiments on both single-domain and cross-domain datasets by masking out all lexical words and topic words corresponding to different POS.

4.2.1 Role of lexical POS

To study the effects of lexical words on authorship attribution, we mask all words corresponding to specific lexical POS as described in Section 2.2.1. We use *NLTK* toolkit (Bird and Loper, 2004) for POS tagging. We perform author identification on both single-domain and cross-domain datasets using the character language model (Teahan and Harper, 2003). We perform 4-fold cross-validation and report the average precision, recall, F-score and accuracy across all cross-validation folds.

For single domain IMDB1M dataset, the performance measures reflecting the effects of masking different lexical POS are reported in Table 3. Similarly, the performance measures for cross-topic and cross-genre experiments on Guardian10 dataset are reported in Table 4.

Table 3: Effect of lexical POS on single-domain authorship attribution using IMDB1M

Scenarios	Precision(%)	Recall(%)	F-score(%)	Accuracy(%)
orig	64.41	64.77	63.42	64.77
no_NN	53.78	50.47	50.73	50.47
no_topic_NN	62.19	62.33	61.12	62.33
no_NNP	59.66	58.38	57.83	58.38
no_topic_NNP	63.20	63.25	62.12	63.25
no_VB	62.78	62.99	61.71	62.99
no_topic_VB	63.23	63.48	62.17	63.48
no_ADJ	63.12	63.25	62.04	63.25
no_topic_ADJ	63.48	63.71	62.45	63.71
no_ADV	63.42	63.91	62.53	63.91
no_topic_ADV	64.02	64.41	63.07	64.41

Table 4: Effect of lexical POS on cross-domain authorship attribution using Guardian10

Domain	Scenarios	Precision(%)	Recall(%)	F-score(%)	Accuracy(%)
Cross-topic	orig	77.79	70.41	69.50	70.41
	no_NN	80.46	69.10	69.35	69.10
	no_topic_NN	84.13	73.13	72.17	73.13
	no_NNP	85.86	80.30	79.98	80.30
	no_topic_NNP	86.25	75.48	74.68	75.48
	no_VB	81.12	67.33	66.77	67.33
	no_topic_VB	80.13	71.41	70.21	71.41
	no_ADJ	78.18	69.49	68.26	69.49
	no_topic_ADJ	83.25	72.73	71.07	72.74
	no_ADV	76.31	67.92	67.13	67.92
	no_topic_ADV	84.17	73.22	72.24	73.22
Cross-genre	orig	78.99	69.84	69.94	69.84
	no_NN	79.18	71.43	71.52	71.43
	no_topic_NN	82.92	73.02	72.80	73.02
	no_NNP	84.52	82.54	80.97	82.56
	no_topic_NNP	81.68	71.43	71.44	71.43
	no_VB	81.82	74.60	74.02	74.60
	no_topic_VB	80.37	73.02	72.34	73.02
	no_ADJ	76.37	69.84	69.61	69.84
	no_topic_ADJ	79.45	71.43	71.14	71.43
	no_ADV	78.71	73.02	72.42	73.02
	no_topic_ADV	80.89	74.60	73.93	74.60

4.2.2 Role of topic words

In this section, we hypothesize that not all words corresponding to lexical POS are content-specific. Hence, we experiment with masking certain topic words corresponding to different lexical POS. The topic words are chosen using the LDA implementation in *gensim* Python toolkit. Only words corresponding to specific lexical POS are input to LDA. We experiment by varying the number of topics (t = 2,10,50) and number of words per topic (w=10,100). The topic words obtained from each author's training data are collated and used for masking. We perform 4-fold cross-validation experiments on both single-domain and cross-domain datasets. We report the performance measures for the best performing configuration using both IMDB1M (t=50, w=10) and Guardian datasets (t=10, w=10) in Tables 3 and 4.

4.2.3 Discussion

It is known that function words are independent of content and are useful for representing style. However, the success of character n-gram approaches in authorship attribution indicate that some lexical words may also be useful for authorship attribution. Besides, character n-gram approaches do not necessarily decouple style and content and using these approaches as such may deteriorate attribution in cross-domain settings. Hence, we studied the effect of masking all words or topic words corresponding to different lexical POS on both single-domain and cross-domain attribution as explained in Sections 2.2.1

and 2.2.2. For all experiments, attribution with the character language model without masking any lexical POS ($orig$) is considered as the baseline.

For single-domain IMDB1M dataset, it can be observed from Table 3 that excluding all common nouns (no_NN) and proper nouns (no_NNP) affects the attribution performance drastically. In contrast, masking only topic words corresponding to common nouns (no_topic_NN) and proper nouns (no_topic_NNP) seems to improve attribution performance compared to masking them completely (no_NNP, no_NN). Nevertheless, this performance is lesser than that of $orig$. This shows the heavy influence of nouns and proper nouns in single-domain attribution. Masking all words or topic words corresponding to other lexical POS like verbs, adjectives or adverbs do not seem to impact the performance even though POS like verbs do contribute significantly to attribution as seen in Figure 2a. This could possibly be due to the heavy dependence on common nouns and proper nouns for attribution under those scenarios.

In contrast, for cross-domain Guardian10 dataset, it can be observed from Table 4 that excluding proper nouns (no_NNP) completely yields a marked improvement in performance for both cross-topic and cross-genre experiments. In fact, excluding only topic words corresponding to proper nouns (no_topic_NNP) performs poorly compared to no_NNP though it performs marginally better than $orig$. Hence, while performing cross-domain attribution, it would be wise to exclude proper nouns from style representations. With respect to common nouns, masking topic words (no_topic_NN) improves performance as compared to completely masking them (no_NN) for both cross-topic and cross-genre scenarios. This implies that some common nouns also help represent style as shown by their distribution in Figures 2b,2c,2d,2e and 2f. With respect to verbs, in cross-topic settings, masking topic words (no_topic_VB) marginally improves performance while completely masking them (no_VB) reduces performance. In cross-genre settings, masking all verbs completely (no_VB) improves performance significantly while masking only topic verbs (no_topic_VB) improves performance marginally compared to $orig$. This implies that choice of verbs have a significant influence in cross-genre settings as compared to cross-topic settings. For both adjectives and adverbs, masking certain topic words (no_topic_ADJ and no_topic_ADV) improves attribution performance as against completely masking them (no_ADJ and no_ADV) for both cross-topic and cross-genre settings. This suggests that some of these adverbs and adjectives may help represent style. The top 20 non-topic words corresponding to different lexical POS for both datasets are shown in Table 5.

Table 5: Top 20 non-topic lexical POS

Guardian10				**IMDB1M**			
Nouns	**Verbs**	**Adjectives**	**Adverbs**	**Nouns**	**Verbs**	**Adjectives**	**Adverbs**
nothing	given	real	probably	opinion	done	little	generally
something	found	better	instead	world	give	overall	somewhat
one	got	past	alone	kind	said	modern	also
end	asked	big	first	sense	put	nice	deeply
point	used	modern	enough	others	thought	new	together
work	came	large	especially	scene	do	fine	just
day	tell	true	soon	day	going	third	hopefully
moment	wants	recent	indeed	times	guess	impressive	perhaps
year	let	less	easily	man	watched	realistic	fairly
kind	turned	right	hardly	thing	getting	long	completely
fact	allowed	hard	perhaps	place	seeing	glad	definitely
question	told	obvious	only	someone	use	fair	first
things	seen	best	surely	story	call	enjoyable	seriously
example	making	wrong	though	reason	look	similar	long
place	seemed	easy	later	chance	saying	less	later
course	thought	full	less	plot	given	easy	therefore
story	mean	foreign	simply	mind	let	huge	best
evidence	run	private	certainly	death	gets	next	personally
business	trying	general	seriously	idea	enjoy	likely	obviously
nation	happen	long	instead	experience	released	happy	only

(a) IMDB1M (b) Guardian10 - Books (c) Guardian10 - Politics

(d) Guardian10 - Society (e) Guardian10 - UK (f) Guardian10 - World

Figure 2: POS distribution on single-domain and cross-domain datasets. POS with $<1\%$ contribution have been suppressed

One of the most pressing problems in this line of research is that there seems to be a dearth for publicly available large-scale cross-domain datasets. Authorship attribution approaches in literature largely demonstrate their results using either small datasets or large-scale single-domain datasets. This does not provide a clear picture of the factors that contribute to style which may be robust in cross-domain settings. The role of syntax or word choice or other aspects in representing style can be better ascertained with experiments on large-scale cross-domain datasets.

5 Conclusion

Authorship attribution approaches in literature focus mostly on single-domain attribution where content and style are highly entangled. This does not provide a clear picture of linguistic aspects that are robust to cross-domain settings. In this paper, we studied the role of sentence structure and word choice in representing style. We evaluated the role of syntax using a purely syntactic language model and show that syntax may be useful with cross-genre attribution while cross-topic attribution and single-domain attribution may be benefit from both syntax and lexical information. However, syntactic language models are not discriminative by themselves and need to be used in conjunction with more successful character n-gram models. We evaluated the role of word choice by masking off all words or certain topic words corresponding to different lexical POS. For common nouns, verbs, adjectives and adverbs, masking off certain topic words yield better performance suggesting that the remaining words corresponding to these lexical POS may help represent style. However, proper nouns seem to be heavily influenced by topic and cross-domain attribution may benefit from completely masking them.

References

Harald Baayen, Hans Van Halteren, and Fiona Tweedie. 1996. Outside the cave of shadows: Using syntactic annotation to enhance authorship attribution. *Literary and Linguistic Computing*, 11(3):121–132.

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

Johanna Björklund and Niklas Zechner. 2017. Syntactic methods for topic-independent authorship attribution. *Natural Language Engineering*, 23(5):789–806.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

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

John Cleary and Ian Witten. 1984. Data compression using adaptive coding and partial string matching. *IEEE transactions on Communications*, 32(4):396–402.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Characterizing stylistic elements in syntactic structure. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1522–1533. Association for Computational Linguistics.

Simon Fuller, Phil Maguire, and Philippe Moser. 2014. A deep context grammatical model for authorship attribution.

Patrick Juola et al. 2008. Authorship attribution. *Foundations and Trends® in Information Retrieval*, 1(3):233–334.

Mike Kestemont. 2014. Function words in authorship attribution. from black magic to theory? In *CLfL@ EACL*, pages 59–66.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st annual meeting of the association for computational linguistics*.

Kim Luyckx and Walter Daelemans. 2008. Authorship attribution and verification with many authors and limited data. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 513–520. Association for Computational Linguistics.

Kim Luyckx. 2011. *Scalability issues in authorship attribution*. ASP/VUBPRESS/UPA.

Ilia Markov, Efstathios Stamatatos, and Grigori Sidorov. 2017. Improving cross-topic authorship attribution: The role of pre-processing. In *Proceedings of the 18th International Conference on Computational Linguistics and Intelligent Text Processing. CICLing*.

Tempestt Neal, Kalaivani Sundararajan, Aneez Fatima, Yiming Yan, Yingfei Xiang, and Damon Woodard. 2017. Surveying stylometry techniques and applications. *ACM Comput. Surv.*, 50(6):86:1–86:36, November.

Sindhu Raghavan, Adriana Kovashka, and Raymond Mooney. 2010. Authorship attribution using probabilistic context-free grammars. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 38–42. Association for Computational Linguistics.

Upendra Sapkota, Thamar Solorio, Manuel Montes, Steven Bethard, and Paolo Rosso. 2014. Cross-topic authorship attribution: Will out-of-topic data help? In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1228–1237.

Upendra Sapkota, Steven Bethard, Manuel Montes, and Thamar Solorio. 2015. Not all character n-grams are created equal: A study in authorship attribution. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 93–102.

Yanir Seroussi, Fabian Bohnert, and Ingrid Zukerman. 2011. Personalised rating prediction for new users using latent factor models. In *Proceedings of the 22nd ACM conference on Hypertext and hypermedia*, pages 47–56. ACM.

Efstathios Stamatatos. 2009. A survey of modern authorship attribution methods. *Journal of the Association for Information Science and Technology*, 60(3):538–556.

Efstathios Stamatatos. 2012. On the robustness of authorship attribution based on character n-gram features. *JL & Pol'y*, 21:421.

Efstathios Stamatatos. 2018. Masking topic-related information to enhance authorship attribution. *Journal of the Association for Information Science and Technology*, 69(3):461–473.

William J Teahan and David J Harper. 2003. Using compression-based language models for text categorization. In *Language modeling for information retrieval*, pages 141–165. Springer.

Learning Target-Specific Representations of Financial News Documents For Cumulative Abnormal Return Prediction

Junwen Duan[†]*, **Yue Zhang**[‡], **Xiao Ding**[†], **Ching-Yun Chang**[‡], **Ting Liu**[†]
[†]Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, Harbin, China
{jwduan,xding,tliu}@ir.hit.edu.cn
[‡]Singapore University of Technology and Design
yue_zhang@sutd.edu.sg,chang.frannie@gmail.com

Abstract

Texts from the Internet serve as important data sources for financial market modeling. Early statistical approaches rely on manually defined features to capture lexical, sentiment and event information, which suffers from feature sparsity. Recent work has considered learning dense representations for news titles and abstracts. Compared to news titles, full documents can contain more potentially helpful information, but also noise compared to events and sentences, which has been less investigated in previous work. To fill this gap, we propose a novel target-specific abstract-guided news document representation model. The model uses a target-sensitive representation of the news abstract to weigh sentences in the news content, so as to select and combine the most informative sentences for market modeling. Results show that document representations can give better performance for estimating cumulative abnormal returns of companies when compared to titles and abstracts. Our model is especially effective when it used to combine information from multiple document sources compared to the sentence-level baselines.

1 Introduction

Texts from the Internet was shown to be statistically correlated with stock market trends (Antweiler and Frank, 2004). Natural language processing (NLP) techniques have been applied to extract information from company filings (Lee et al., 2014), financial news articles (Xie et al., 2013) and social media texts (Bollen et al., 2011) in order to gain understandings of financial markets. In particular, traditional methods exploited statistical signals, such as lexical and syntactic features (Wang and Hua, 2014; Schumaker and Chen, 2009), sentiment (Bollen et al., 2011) and event structures (Ding et al., 2014; Ding et al., 2015; Ding et al., 2016) from these text sources, which suffers from feature sparsity problems.

With the recent trends in deep learning for NLP, neural networks have also been leveraged to learn dense representations for text elements, which can address the sparsity of discrete features in statistical models. Such representations can implicitly represent sentiment, event and factual information, which can be extremely challenging for sparse indicator features. In particular, Ding et al. (2015) show that deep learning representations of event structures yield better accuracies for stock market prediction compared to discrete event features. Chang et al. (2016), Duan et al. (2018) use neural networks to directly learn representations of news abstracts, showing that it is effective for predicting the cumulative abnormal returns of public companies.

One limitation of Ding et al. (2015) and Chang et al. (2016), however, is that these methods only model news titles and abstract texts, which are typically single sentences. Ding et al. (2015) show that a model that uses only news content gives inferior results compared to one trained using news titles only, and that adding news content information to a title-driven model does not significantly improve the results. Intuitively, news content can contain richer information that is not directly relevant to the title message, or the most important event, and hence can lead to noise in predictive modeling. On the other hand, news

* This work was done while the first author was visiting Singapore University of Technology and Design. Yue Zhang is the corresponding author.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2823–2833
Santa Fe, New Mexico, USA, August 20-26, 2018.

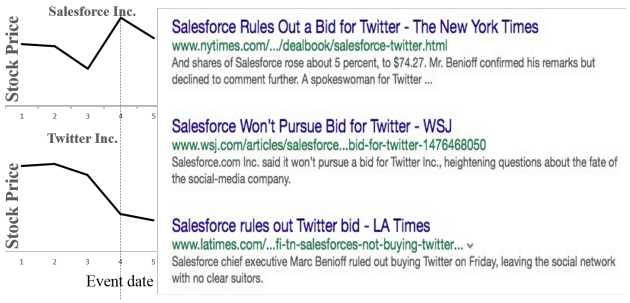


Figure 1: Event "Salesforce rules out Twitter bid" showing different impacts to differnt firms. Twitter fell 5.12%, while Salesforce rose 5.15% on the event day.

content can also contain useful information for making informed decisions. For example, given the news abstract passage *"Amicus Inc (arcs.o), a provider of health-care IT services, said it agreed to be bought by an affiliate of private firm them bravo llc for $217 million in an all-crash deal."*, it would be difficult to tell whether the acquisition is beneficial to investors. However, a sentence in the news content states that *"the deal offers Amicus Inc shareholders $5.35 for each share, a premium of 21 percent over the stock's close on December 24 on NASDAQ"*, which explicitly indicates a positive return.

We aim to exploit such useful information from the news content for making more informed decisions in stock market prediction. A main challenge is how to automatically identify the most useful parts of the news content, while disregarding noise, which prevents naive utilization of news contexts (aka Ding et al. (2015)). Another challenge, as Chang et al. (2016) suggest, is that information must be selected with regard to a certain stock of interest. As shown in Figure 1, the same event "*Salesforce rules out Twitter bid*" can lead to different influences on different stocks, with Salesforce benefiting from it yet Twitter suffering from it.

To address the above challenges, we build a neural model that selects and represents relevant sentences from a full news document with respect to a specific firm of interest. In particular, we leverage conditional encoding (Rocktäschel et al., 2015) to encode information of a given stock into the dense representation of the news abstract. Using this target-specific news abstract representation, we apply neural attention (Bahdanau et al., 2014) over each sentence in the news content to automatically learn its relative importance with regard to the target company. The attention weights are learned automatically towards a final predictive goal. The model is full data-driven, which do not rely on an external syntactic parser as the first step to obtain its target-specific linguistic structures.

In addition to Chang et al. (2016) model, which uses only abstract information, we also compare with several state-of-the-art baselines for learning document representations, such as paragraph vector (Le and Mikolov, 2014) and hierarchical attention network (Yang et al., 2016), giving the best reported performances. The advantage of our approach over sentence-level baseline is especially obvious when it is used to combine information from multiple news document sources. In addition, a case study shows that our model can select the sentences that most intuitively help predict stock returns from a full news document. Our contributions can be summarized as follows:

- We propose a target-specific document representation model, which leverages the abstracts as evidences to select informative sentences from the documents while disregarding noise.
- We are the first, to our knowledge, to build a neural model that can effectively leverage full news document for stock market prediction.

  Resources of this work can be found at http://github.com/sudy/coling2018

2 Problem Definition

Cumulative Abnormal Return The task that we attack in this paper is *Cumulative Abnormal Return Prediction (**CAR**)*. Formally, the abnormal return AR_{jt} of a firm j on a date t is the difference between its actual return R_{jt} and the expected return $\hat{\mathrm{R}}_{jt}$, $\mathrm{AR}_{jt} = \mathrm{R}_{jt} - \hat{\mathrm{R}}_{jt}$. The expected return $\hat{\mathrm{R}}_{jt}$ can be

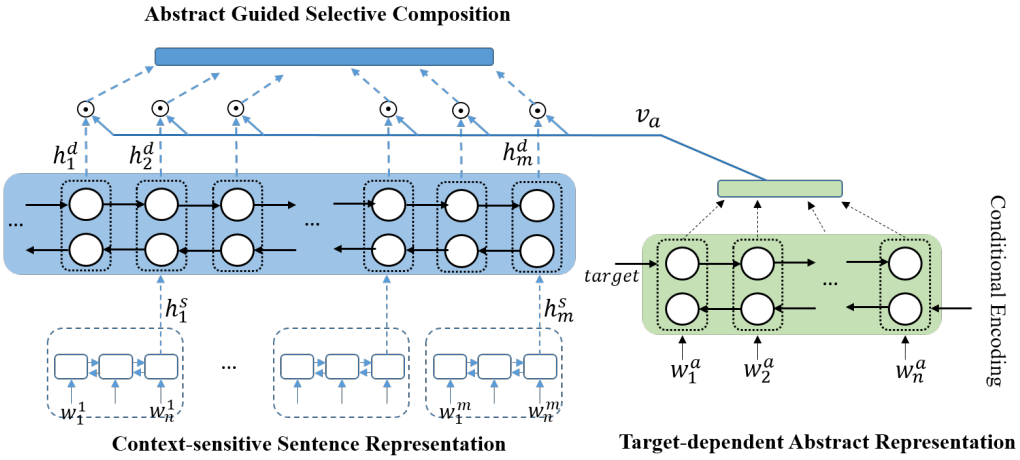


Figure 2: The architecture of proposed method

estimated by an asset price model based on historical prices, or approximated by the market return in a short-term event window (Kothari and Warner, 2004). The cumulative abnormal return CAR_j of the firm j in an n-day time window is calculated by summing up the daily abnormal returns in the period (Eq 1). In this paper, we adopt the commonly used three-day window (-1,0,1), which we denote as CAR_3 and day 0 is the day when the current news documents are released.

$$\text{CAR}_j = \sum_{t=1}^{n} \text{AR}_{jt} \tag{1}$$

Cumulative Abnormal Return Prediction The ultimate goal is to build a model that predicts the cumulative abnormal return CAR_3 of a public company based on a set of related news documents released on day 0. Given a document D and associated firm f, we learn a representation d_f for D that is specified to firm f based on our proposed approach (introduced in Section 3). With the firm-specific document representation d_f, we predict the CAR_3 direction $y \in \{-1, 1\}$ using a parameterized softmax function (Eq 2).

$$p(y|d_f) = \text{softmax}(W \cdot d_f + b), y \in \{-1, 1\}, \tag{2}$$

where $p(y = -1|d_f)$ and $p(y = 1|d_f)$ indicate the probability of CAR_3 being negative and positive, respectively. In the next section, we describe our method for learning d_f.

3 Model

The challenges for learning a target-sensitive document representation are two-fold. On the one hand, we must encode firm-specific information into the dense document representations so as to make them different across targets. On the other hand, we must identify the most informative sentences while disregarding noise for the prediction. To remedy this, we propose to first learn a target-specific representation for the abstract, which is most relevant to the market movement among all the sentences and then leverage the abstract to guide the sentence selection. The architecture of our proposed approach is illustrated in Figure 2, which consists of three key modules. We give details of each module in this section.

3.1 Target-dependent Abstract Representation

As the first step, we learn a target-dependent representation for the abstract of a news document by encoding target information into it. We use a bidirectional Long-short memory network as the basic model. In order to allow information of a target company to influence the semantic representation, we apply conditional encoding (Rocktäschel et al., 2015), using an embedding vector of the target firm of concern $e^t(c)$ as the initial state vector for the sentence-level Bi-LSTMs. We average the hidden states of each word in the sentence to obtain the target-dependent news abstract representation v_a. The vector

$e^t(c)$ for the company of interest is initialized by averaging the words of its constituents and are fine-tuned during training.

3.2 Context-sensitive Sentence Representation

To preserve the semantic structures of documents and make the sentence representations aware of their contexts, we leverage a hierarchical structure (Li et al., 2015) to encode sentences (the abstracts are not considered) in a document. A sentence-level LSTM is first used to encode words into hidden state vectors, and then a document-level LSTM is applied to encode sentences into hidden state vectors.

At the sentence level, given a sentence $\{w_1, w_2, \cdots, w_n\}$, we obtain their embedding forms $\{\vec{e}(w_1), \vec{e}(w_2), \cdots, \vec{e}(w_n)\}$ via a lookup table. A Bi-LSTM is used to capture sentence-level context from $\{\vec{e}(w_1), \vec{e}(w_2), \cdots, \vec{e}(w_n)\}$, yielding two sequences of hidden states $\{\overrightarrow{h_1^w}, \overrightarrow{h_2^w}, \cdots, \overrightarrow{h_n^w}\}$ and $\{\overleftarrow{h_1^w}, \overleftarrow{h_2^w}, \cdots, \overleftarrow{h_n^w}\}$, respectively. We average $\overleftarrow{h_i^w}$ and $\overrightarrow{h_i^w}$ into a single vector h_i^w for the word w_i, and use average pooling over the sentence to obtain a single sentence embedding vector h^s.

In the document level, given the sentence embedding $\{h_1^s, h_2^s, \cdots, h_m^s\}$ for sentences $\{s_1, s_2, \cdots, s_m\}$ in a document D, respectively, the same Bi-LSTM structure (with a different set of model parameters) is applied to obtain hidden states $\{\overrightarrow{h_1^d}, \overrightarrow{h_2^d}, \cdots, \overrightarrow{h_m^d}\}$ and $\{\overleftarrow{h_1^d}, \overleftarrow{h_2^d}, \cdots, \overleftarrow{h_m^d}\}$, respectively. For each sentence s_i, the forward and backward hidden vectors are averaged to give a single hidden state embedding h_i^d. h_i^d contains both internal information from the sentence by semantic composition of words, and document level context by bi-directional recurrent composition.

We use this vector form of each sentence as their representation for abstract and sentence composition. We apply the same conditional encoding to sentence-level and document-level Bi-LSTMs in context-sensitive sentence representation.

3.3 Abstract Guided Selective Composition

As mentioned in Section 1, some sentences can give background information that can support decision making, which is too lengthy to include in the abstract. To address the challenge of informative sentence selection, we consider the target-specific representation obtained as well as the sentence relevances to the abstract in our model.

To compose context-sensitive sentence representations into a final document representation, we use the attention mechanism (Bahdanau et al., 2014), taking the embedding of the abstract to guide calculation of attention weights. Another benefit of using the attention mechanism is the prediction is made interpretable, since interpretability is crucial in financial prediction, as investors have to verify the underlying basis for the decisions.

Formally, given the target-specific representation of the abstract v_a from Section 3.1 and the context-sensitive sentence representations $\{h_1^d, h_2^d, \cdots, h_m^d\}$ from Section 3.2, we concatenate v_a and h_i^d and feed it to a single-layer, feed-forward neural network (Eq 3) to get the score for each sentence h_i^d in the document,

$$u_i = v^\top \tanh(W_a[v_a, d_i] + b), \tag{3}$$

where W_a is the weight matrix, b is the bias, v is a projection vector that maps the output to a scalar and u_i is the weight score showing how much attention should be put on current sentence s_i in the composition of the whole document. The attention score α_i are computed from u_i by Eq 4.

$$\alpha_i = \frac{\exp(u_i)}{\sum_i \exp(u_i)}, \tag{4}$$

$$d = \sum_i \alpha_i h_i^d. \tag{5}$$

Here α_i is the normalized weight score, where $\sum_i \alpha_i = 1$. The final output d is a weighted sum of all the hidden states h_i^d (Eq 5). We concatenate the vector of the abstract v_a and d as the final representation for the document $d' = [v_a, d]$.

	Training	Development	Test
+CAR$_3$	9,674	493	995
-CAR$_3$	9,643	507	1,005

Table 1: Number of CAR$_3$ in the datasets

3.4 Multi-Document Composition

There can be more than one news document mentioning a firm of interest in a certain event window. We adopt a self-attentive neural network (Bahdanau et al., 2014) to assign different weights to the news documents $\{d'_1, d'_2, \cdots, d'_n\}$ (Eq 6) with respect to our prediction goal.

$$u_i = v^\top \tanh(W^{(s)} d'_i + b^{(s)}), \tag{6}$$

where $W^{(s)}$ is a weight matrix and $b^{(s)}$ is a bias, u_i is a scalar showing how much attention should be put on document d_i. The normalization process is the same with Eq 5 but with different parameters. We denote the final representation as $\hat{d}$, and use it as d_f for CAR$_3$ prediction in Section 2.

4 Training

Given a set of training instances T with documents and their corresponding CAR$_{\mathcal{3}}$ as discussed in Section 2, the overall training objective is to minimize the cross-entropy loss with `L2` regularization (Eq 7),

$$\min_{\Theta} - \sum_{T} \sum_{j} t_j \log p(y_j|d_f) + \sum_{\theta \in \Theta} \lambda_\Theta ||\theta_2^2||, \tag{7}$$

where $p(y_j|d_f)$ is the predicted distribution for class j and t_j is the ground-truth distribution. Θ represents all the parameters and the λ_Θ are the regularization parameters.

We pre-train continuous bag-of-word (CBOW) word embeddings with dimension 100 on a collection of Reuters and Bloomberg financial news articles, which is released by Ding et al. (2014). The overall document size is over 400K. The open source toolkit *word2vec*[1] is used to train the word embeddings, which are then fine tuned during the training of the prediction model to help boost the model performance. Out-of-vocabulary (OOV) words are replaced by a `UNK` token. We adopt `Adam` (Kingma and Ba, 2014) as our optimization algorithm, with the initial learning rate being set to 0.0005, `L2` regularization at the strength of 10^{-6}. We use mini-batch size of 32, the model that achieved the best micro-F1 performance on the development set is kept for final test.

5 Experiments

5.1 Data

We collect publicly available financial news articles from Reuters from October 2006 to December 2015. In our preliminary experiments, we find that a news document is more likely to be relevant to a firm only if it is mentioned in the news abstract. Thus, we only include news documents mentioned at least one public listed firm in the U.S. security market. We group the news documents per firm per event date. If the news is released during a trading hour, day 0 is the current day, otherwise day 0 is the next trading day. We compute the expected return $\hat{\mathrm{R}}_{jt}$ by the return of equally-weighted market index including all the stocks on NYSE, Amex, NASDAQ.

We follow Chang et al. (2016) and split the dataset into training, development and test sets, with 1000 instances for development, 2000 instances for testing and the rest for training. The numbers of positive and negative CAR$_{\mathcal{3}}$ in the dataset are listed in Table 1, in which the positive and negative cases are fairly balanced.

[1] https://code.google.com/archive/p/word2vec/

5.2 Evaluation Metrics

Our model is designed for stock recommendation, where companies with high absolute cumulative abnormal return expectations can be given to traders for their information. As a result, it can be useful to have a confidence threshold $beta$ in the model, so that only companies with $p(y = 1|d_f) > \beta$ or $p(y = -1|d_f) > \beta$ are recommended. With increasing β values, the number of recommended stocks is expected to decrease, while the precision to increase. As a result, the model performance is evaluated by the area under the precision-recall curve (AUC). The precision and recall are calculated on both the positive class and negative class. This metric offers a trade-off between precision and recall when varying the prediction confidence threshold. Following Chang et al. (2016), we also evaluate the model on test instances with $|\text{CAR}_3| > 2\%$, namely the event windows with high impacts.

5.3 Baselines

We compare our method with a set of baseline representation learning methods for d_f learning including *target-dependent abstract* representation method of Chang et al. (2016), and state-of-the-art *target-independent document* representations.

Target-dependent News Abstract representation (TGT-CTX-LSTM) Chang et al. (2016) give the current state-of-the-art accuracies for CAR prediction by using the abstract only. They used a syntactic parser to analyze the dependency structure of the abstract (as a sentence), and then transform it to a target-specific dependency tree form. They leverage a tree structured LSTM (Tai et al., 2015) to learn abstract representation according to the target-specific tree form.

Paragraph Vector We take the paragraph vector embedding of Le and Mikolov (2014) as an unsupervised full document representation baseline. This method gives remarkable performances on tasks such as document classification (Yang et al., 2016) and sentiment analysis (Tang et al., 2015).

Target-dependent sentence combination (TD-AVG) This model first learns a context-sensitive representation for all sentences using Bi-LSTM and conditional encoding (Rocktäschel et al., 2015). Average pooling is then applied on the sentence representations to obtain the final document representation. This baseline is equivalent to a conditionally encoded version of the target-specific model of Li et al. (2015), but with a different training objective (i.e., classification instead of auto-encoder). We refer to it as TD-AVG. We use conditional encoding (Rocktäschel et al., 2015) on the sentence-level and document-level Bi-LSTMs.

Target-dependent hierarchical neural network (TD-HAN) This model is similar to TD-AVG except that it adopts an attention model (Bahdanau et al., 2014) over the context-sensitive sentence representations. Intuitively, similar to abstract, some parts of the document are more related to CAR compared to others. Such sentences should be given more weights in composition for document representations. This baseline is adapted from Yang et al. (2016), who applied attention on both the word level and the sentence level in a hierarchical LSTM network for document representation. In the implementation, it is slightly different in not using attention on the word level. We refer to it as TD-HAN.

5.4 Results

Table 2 summarizes the AUCs of both positive and negative classes of different embedding models on the test dataset. As shown in Table 2, our model gives AUCs of 0.65 for both positive and negative classes on the test dataset, outperforming all the baselines. Following Chang et al. (2016), we also make comparisons on cases that give higher CAR, with threshold $|\text{CAR}_3| > 2\%$. This subset covers a total of 1021 cases. Our model gives AUCs of 0.75 and 0.73 on $+\text{CAR}_3$ and $-\text{CAR}_3$, respectively. The improvement is statistically significant at a 1% significance level using T-test. Figure 3 presents the precision-recall curve of *Paragraph Vector*, *TGT-CTX-LSTM* (Chang et al., 2016) and our proposed method. We make more detailed analysis of our approach and relevant models.

	+CAR$_3$	-CAR$_3$	+CAR$_3$ $> 2\%$	-CAR$_3$ $< -2\%$
Paragraph Vector (Le and Mikolov, 2014)	0.51	0.53	0.54	0.55
TGT-CTX-LSTM (Chang et al., 2016)	0.63	0.62	0.70	0.68
TD-AVG (Tang et al., 2015)	0.61	0.61	0.71	0.68
TD-HAN (Yang et al., 2016)	0.63	0.63	0.72	0.71
Our Approach (without target)	0.64	0.63	0.73	0.72
Our Approach	**0.65****	**0.65****	**0.75****	**0.73****

Table 2: Final *AUC* on the test set, ** means that the result is better than the best baseline at 1% significance level.

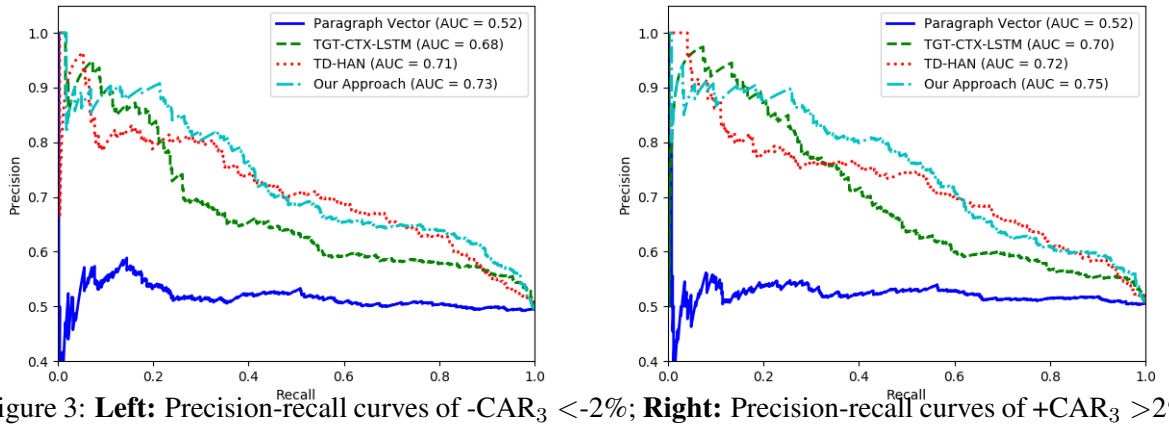


Figure 3: **Left:** Precision-recall curves of -CAR$_3$ $<$-2%; **Right:** Precision-recall curves of +CAR$_3$ $>$2%

The Usefulness of Supervision The performance of *Paragraph Vector* falls far behind other approaches. The primary cause is that its document representation is both target-independent and unsupervised, which neither benefit from the end task nor information from specific targets. This demonstrates the importance of supervisions in capturing the deep semantic meaning of documents for our prediction task.

The Usefulness of Modeling Full Document As shown in Table 2, simple averaging schema TD-AVG gives inferior performances compared to the target-specific abstract representation TGT-CTX-LSTM. This indicates the informativeness of abstract for predicting the stock market movements. However, when the simple average pooling of TD-AVG is replaced by a simple feed-forward attention over the context-sensitive sentence representation, i.e., TD-HAN, the model achieves comparable performance with state-of-the-art model TGT-CTX-LSTM. The results indicate the importance of choosing the right strategy to compose the sentences in document-level compositions, since a document is more sparse and contains noise. It also implies that if the document-level background information is properly modeled, document-level models have obvious advantages over sentence-level model, since short text and simple structure may not fully represent the information conveyed in the document.

Our proposed method, which incorporates firm-specific information as well as the hierarchical structures of news documents, achieves the best performances, showing that target-specific supervisions and fine-grained information from the main contents of news can help gain understanding about the market fluctuations.

The Impact of Target The performances of our model with and without target information are also compared in Table 2. Our model without target information achieves 0.64 and 0.63 AUC on positive and negative classes, respectively, which is comparable with the best baselines. Encoding the target information into the representation did not bring a significant improvement to the model performance. An explanation for the relatively small difference between our method with and without target-specific

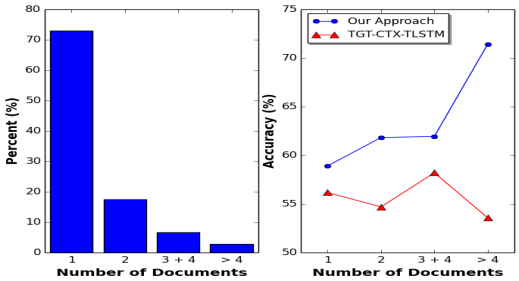


Figure 4: **Left**: The proportion of instances with different number of financial news documents in our test dataset. For example, over 70 percent of instances in our test dataset have information collected from one document in the event window; **Right**: The accuracy of our approach and baseline TGT-CTX-TLSTM against the number of news documents in the event window.

encoding is that there is a relatively low ratio (10.2%) of news abstracts in our dataset mentioning more than one company.

Accuracy with Respect to the Number of Documents As mentioned in the previous section, there can be more than one news documents associated with a firm of interest in a particular event window. More than 25 percent of CAR_3 instances in our dataset have information gathered from more than one documents. Extra documents can provide richer background information about the firm of interest, but may also bring more noise. Figure 4 (Right) shows the accuracies of our model and the baseline TGT-CTX-LSTM with respect to the number of news documents. It is worth noting that our model and TGT-CTX-LSTM use the same attention strategy to combine information from multiple documents. The performance of our approach improves as the number of documents to model grows, showing its effectiveness in utilizing more non-overlapping information sources. Compared to sentence-level baseline TGT-CTX-LSTM, our model can better capture information from multiple document sources.

5.5 Case Study

To illustrate the attention mechanism for weighing sentences in the news documents, Table 3 shows the details of three news documents. The target firm, the actual CAR_3 and the prediction probability of our method and the baseline method TGT-CTX-LSTM are shown in the first rows of each news. The first line of each document is the abstract. The automatically learned weights are shown in the front of each sentence. For case 1, since the most informative parts of the document are not in the abstract, TGT-CTX-LSTM, which only exploits the abstract, yields the incorrect prediction. Our model puts more weights on the last two sentences, which suggest the value of the acquired properties and the voting rights for the target Lennar Corp.. This meets human expectation, since they have shown the potential of the future prospect of the firm.

For case 2, though the ground-truth CAR_3 outperforms the market by a large margin, the baseline TGT-CTX-LSTM shows less confidence for the prediction compared with our method. The abstract gives little knowledge about the purchase event, which make the system that rely on the abstract difficult to make a decision. The second sentence in the news explicitly state the impact of the news to the shareholders, the target firm stock price rose by more than 20 percent. We observed similar patterns in other cases, where the model prefer to put more weights on the sentences which explicitly mention the stock returns. An explanation is that the market follows the trends and it takes time for the market to absorb all information.

For case 3, both methods present similar confidence for the prediction. However, our model puts more weight on the third sentence instead of the abstract. The two sentences can be treated as paraphrases, because they are very close in the meaning, both mentioning future profit expectations of the firm and a quarterly profit decline. It is worth noting that, for documents with a large number of long sentences, our model tends to give evenly-distributed weights for all the sentences in such documents. This is a limitation of our attention model, which we leave for future work. Above all, the result demonstrates that our model is able to automatically capture the most informative parts in the documents.

Our Method	TD-HAN	Case: No. 1; Target: Constellation Brands Inc. ; CAR_3:3.3% ; Our method:0.57; TGT-CTX-LSTM:0.45
0.05	0.07	**Abstract**: builder lennar_corp (len.n) said on friday it formed an investment venture with morgan stanley real estate and sold the venture $ 525 million in properties .
0.07	0.28	the properties acquired by the new entity consist of about 11,000 homesites in 32 communities across the united states , it said .
0.07	0.30	lennar_corp and morgan stanley real estate , an affiliate of morgan stanley & co. (ms.n) , formed the venture to acquire , develop , manage and sell residential real estate .
0.34	0.24	lennar_corp acquired a 20 percent ownership interest and 50 percent voting rights in the venture .
0.47	0.09	as of september 30 , the acquired properties had a net book value of $ 1.3 billion , it said .
Our Method	**TD-HAN**	**Case**: No. 2; **Target**: Amicas Inc.; CAR_3:23.2%; **Our method**:0.77; **TGT-CTX-LSTM**:0.53
0.06	< 0.01	**Abstract**: amicas_inc amcs.o , a provider of healthcare it services , said it agreed to be bought by an affiliate of private firm thoma bravo llc for $ 217 million in an all-cash deal .
0.67	< 0.01	the deal offers amicas_inc shareholders $ 5.35 for each share , a premium of 21 percent over the stock 's close on december 24 on nasdaq .
0.06	< 0.01	amicas_inc expects to close the transaction in the first quarter of 2010 , it said in a statement .
0.21	0.99	shares of the boston , massachusetts-based company closed at $ 4.42 thursday .
Our Method	**TD-HAN**	**Case**: No. 3; **Target**: Canon Inc.; CAR_3:4.7%; **Our method**:0.65; **TGT-CTX-LSTM**:0.71
0.05	0.01	**Abstract:** canon_inc (7751.t) posted a 30.9 percent decline in quarterly operating profit on monday , hurt by production halts due to parts shortages after the march 11 earthquake , but it raised its full-year forecast due to a faster recovery than expected .
0.2	0.03	canon_inc 's april-june figure came to 78.4 billion yen ($ 1 billion) , which is higher than an expected profit of 55.9 billion yen , the average of six analysts polled by thomson_reuters_corp i/b/e/s , but lower than the 113.4 billion yen it booked for the same quarter last year .
0.53	0.11	the world 's biggest maker of digital cameras also lifted its annual forecast to 380 billion yen , after slashed its full-year operating profit forecast following the devastating march earthquake and tsunami to 335 billion yen from 470 billion yen .
0.22	0.85	market expectations are for an annual profit of 365 billion yen , based on the average of 18 forecasts by analysts polled by thomson_reuters_corp i/b/e/s .

Table 3: Weights learned by our method and TD-HAN for each sentence in news documents. The actual CAR_3, the predicted probability of our method and TGT-CTX-LSTM are shown in the first row of each news document. The first sentence is the abstract.

6 Related Work

Our work falls into the area of mining financial text for stock market prediction. Rich linguistic features in financial text have been studied. Wang and Hua (2014) and Schumaker and Chen (2009) exploit shallow lexical and syntactic features, such as n-grams, noun-phrases and named entities. Such representations ignore the word orders and the important semantic relations between sentences, which thus can not fully represent the information conveyed in a document. With the rise of open information extraction techniques, Xie et al. (2013) exploit semantic frames and Ding et al. (2014) use event tuples to represent the news events. Such representation suffers from the problem of discreteness and data sparsity, which makes it difficult to generalize. The main difference between our method and this line of work is that they use discrete representations of the title or abstract of the financial news. Instead, we propose a fixed-length, continuous representations for the entire documents.

Our work also aligns with recent work on using deep neural models to learn embedding vectors for sentences and documents. Le and Mikilov (2014) extend the skip-gram algorithm (Mikolov et al., 2013) by regarding documents as contexts, training their embeddings together with training word embeddings. Kingma and Ba (2014) train sentence embeddings by using LSTMs to compose word vectors in sentences, leveraging neighbor sentences. Kenter et al. (2016) take a similar basis by using neighbor sentences to guide sentence vector training, but use a simpler network structure by calculating sentence embeddings as the sum of its word embeddings. Li et al. (2015) use a hierarchical auto-encoder structure to learn sentence and document embeddings. Yang et al. (2016) extends the hierarchical LSTM network of Li et al. (2015), applying attention for weighting different words and sentences, giving state-of-the-art accuracies for document classification. In contrast to the existing methods, we propose learning a target-specific representation of documents for a specific end task, namely news-driven market prediction.

7 Conclusion

We investigated target-specific document representations of financial news for cumulative abnormal return prediction in this paper, comparing a set of document embedding models. Empirical studies showed that a combination of target-specific news abstract representation and contextual sentence representation via the attention mechanism can give better results compared to several alternative sentence-level and document-level methods. Our final model demonstrated the usefulness of modeling a full document, as compared to only titles and abstracts, for obtaining more accurate results.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments and suggestions to help improve this paper. This work was partly supported by the National Key Basic Research Program of China via grant 2014CB340503, the National Natural Science Foundation of China (NSFC) via grant 61472107 and 61702137.

References

Werner Antweiler and Murray Z Frank. 2004. Is all that talk just noise? the information content of internet stock message boards. *The Journal of Finance*, 59(3):1259–1294.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *JCS*, 2(1):1–8.

Ching-Yun Chang, Yue Zhang, Zhiyang Teng Teng, Zahn Bozanic, and Bin Ke. 2016. Measuring the information content of financial news. In *26th Coling*.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2014. Using structured events to predict stock price movement: An empirical investigation. In *EMNLP*, pages 1415–1425.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep learning for event-driven stock prediction. In *IJCAI*, pages 2327–2333.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2016. Knowledge-driven event embedding for stock prediction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2133–2142.

Junwen Duan, Xiao Ding, and Ting Liu. 2018. Learning sentence representations over tree structures for target-dependent classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 551–560.

Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese cbow: Optimizing word embeddings for sentence representations. *arXiv preprint arXiv:1606.04640*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

SP Kothari and Jerold B Warner. 2004. The econometrics of event studies. *Available at SSRN 608601*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.

Heeyoung Lee, Mihai Surdeanu, Bill MacCartney, and Dan Jurafsky. 2014. On the importance of text analysis for stock price prediction. In *LREC*, pages 1170–1175.

Jiwei Li, Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1106–1115.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Robert P Schumaker and Hsinchun Chen. 2009. Textual analysis of stock market prediction using breaking financial news: The azfin text system. *TOIS*, 27(2):12.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1556–1566.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, pages 1422–1432.

William Yang Wang and Zhenhao Hua. 2014. A semiparametric gaussian copula regression model for predicting financial risks from earnings calls. In *ACL*, pages 1155–1165.

Boyi Xie, Rebecca J Passonneau, Leon Wu, and Germán G Creamer. 2013. Semantic frames to predict stock price movement. In *ACL*, pages 873–883.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of NAACL-HLT*, pages 1480–1489.

Model-Free Context-Aware Word Composition

Bo An[2,3], **Xianpei Han**[1,2], **Le Sun**[1,2]
[1]Beijing Advanced Innovation Center for Language Resources, Beijing, China
[2]State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences, Beijing, China
[3]University of Chinese Academy of Sciences, Beijing, China
{anbo, xianpei, sunle}@iscas.ac.cn

Abstract

Word composition is a promising technique for representation learning of large linguistic units (e.g., phrases, sentences and documents). However, most of the current composition models do not take the ambiguity of words and the context outside of a linguistic unit into consideration for learning representations, and consequently suffer from the inaccurate representation of semantics. To address this issue, we propose a model-free context-aware word composition model, which employs the latent semantic information as global context for learning representations. The proposed model attempts to resolve the word sense disambiguation and word composition in a unified framework. Extensive evaluation shows consistent improvements over various strong word representation/composition models at different granularities (including word, phrase and sentence), demonstrating the effectiveness of our proposed method.

1 Introduction

Recent development in natural language processing (NLP) has seen a rise of continuous representation learning of linguistic units, which has been successfully applied to various tasks such as contextual word similarity (Iacobacci et al., 2015; Huang et al., 2012), machine translation (Devlin et al., 2014), paraphrase detection (Socher et al., 2011a) and sentiment classification (Tang et al., 2015). The continuous representation of linguistic units bring many benefits. For example, we can easily calculate the semantic relatedness between two words *'dog'* and *'cat'* using cosine distance of their continuous representations. Specifically, a representation learning method aims at mapping a linguistic unit with k words $S = [w_1, w_2, ..., w_k]$ into a continuous vector in a low dimensional feature space.

Currently, most of the representation learning methods focus on learning word embeddings, mostly based on the distributional hypothesis (Harris, 1954), i.e., *words in similar contexts tend to have similar meanings*. In most cases, a word is represented by summarizing all its contexts in a large text corpus, either by dimension reduction from co-occurrence matrix (Levy and Goldberg, 2014), or by neural network models (Collobert and Weston, 2008; Mikolov et al., 2013; Huang et al., 2012).

Intuitively, a word may express distinct meanings in different contexts, so its representations may be distinct in various linguistic units. For example, the word *'bank'* should have different representations in *'commercial bank'* and in *'river bank'*, because they correspond with two distinct senses of *'bank'*: the former with the meaning of *'financial institution'*, and the latter with *'riverside'* meanings. To address this issue, some multi-prototype word embeddings models are proposed to learn multiple representations for individual words. For instance, MSSG (Neelakantan et al., 2015) utilizes local context to infer the accurate semantic of a word, and the global context is incorporated by TWE (Liu et al., 2015) to learn accurate representations of a word.

The representations of linguistic units larger than words (e.g., phrases, sentences), unfortunately, usually cannot be learned using the method as word representations. The main reason is the context sparsity problem, i.e., most of the large linguistic units will not occur frequently even in a large text corpus, which leads to insufficient context statistics for accurate representations learning.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2834–2845
Santa Fe, New Mexico, USA, August 20-26, 2018.

To effectively learn the representations of large linguistic units, a promising approach is word composition, which is based on the Frege's principle of composition: (Pelletier, 2001) – *the meaning of a complex expression is determined by the meanings of their constituent expressions and the rules used to combine them*. Specifically, given a linguistic unit S with $[w_1, w_2, ..., w_k]$ words, a composition method first infers the vectors of all words in S based on embedding matrix. After that, a composition model is utilized to transform the word vectors into a single representation of unit *S*. For example, to represent the phrase *'the dog outside'*, a composition method will first infer the representations of *'the'*, *'dog'* and *'outside'*, and then feed those word vectors into a composition model, e.g., element-wise addition, element-wise multiplication (Mikolov et al., 2013), recursive neural network (Socher et al., 2012), gated recurrent neural network, Long-Short Term Memory network (Le and Zuidema, 2015) or convolutional neural network (Xu et al., 2015). There are also approaches that jointly learn the representations of words and larger units, e.g Le and Mikolov (2014).

However, the meaning of a large linguistic unit not only depends on its constituent words but also affected by the context around it. For example, the meaning of word *'going'* in sentence *'I am going to the bank.'* can be disambiguated based on the local context (the other words in the sentence). But the meaning of word *'bank'* in this sentence can not be inferred without context outside this sentence. Unfortunately, deriving contextual information outside of a given sentence is not trivial due to the data sparsity problem. What's worse, in some practical NLP applications, there is no context for a given sentence, such as question answering or information retrieval.

To address this issue, this paper proposes a model-free context-aware word composition model to learn proper representations for linguistic units at different granularity levels by incorporating latent semantic information. The proposed model attempts to address the word sense disambiguation and word composition in a unified architecture. Specifically, inspired by TWE (Liu et al., 2015), this paper utilizes topic distribution as the global context of a linguistic unit. Each topic is utilized to derive accurate meanings for all the word occurrences in the linguistic unit and to learn the topic-specific representation of the unit. After that, the context-aware representation of the linguistic unit is inferred by summarizing all its representations under different topics based on the topic distribution. In this way, our method can make use of the topic information of a word to learn its accurate topic-specific representation, and the topic distribution of the a unit is employed as a cue to guide the process of word composition to learn meaningful representation.

Note that, our context-aware composition framework is model-free in that it can employ any existing composition model as the base composition model. We assess our method on multiple text similarity tasks based on various base composition models. Experimental results verify the effectiveness of our model on learning the representations of linguistic units at different granularity level, and show consistently improvements over various base composition models. The main contribution are threefold: (1) It regards the linguistic unit as a whole, and utilizes the latent semantic information to learn the accurate representations for both the linguistic unit and the word occurrences in it; (2) This paper verifies that the topic information is benefit for both the accurate word representation and word composition. (3) We proposed a model-free framework that can enhance various kinds of methods.

2 Related Work

This section briefly reviews related work, including context representation models, word/phrase embeddings leanings methods and word composition methods.

2.1 Context Representation Model

How to represent contextual information is a key topic in natural language processing research area. Currently, a main trend is to represent context via latent variable models (Szpektor and Dagan, 2008), e.g., the Latent Dirichlet Allocation (LDA) (Blei et al., 2003). Using LDA models, a context will be represented as a topic distribution. Frank et al. (2014) used topic models to provide context for a vowel categorisation task in child directed speech.

2.2 Word/Phrase Embeddings

Currently, most of the word embeddings methods use either neural network or co-occurrence matrix. Several well-known models include C&W (Collobert and Weston, 2008), word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). The main drawback of these methods is that they represent a word using a single vector, such a uniform representation method cannot accurately represent polysemy.

There are also various methods which try to learn multi-prototype word embeddings to solve the polysemy problem, with each vector corresponding to a sense instead of word, such as the Multi-Prototype model (Huang et al., 2012), the MSSG (Neelakantan et al., 2015), the EHModel (Tian et al., 2014), the SenseEmb (Guo et al., 2014) and the SAMS model (Cheng and Kartsaklis, 2015). The main drawback of these models is that they need to induce the senses of a word, which is a very challenging task.

Salant and Berant (2017) verified the importance of the contextual information for word representation. Recently, a number of methods are proposed to learn contextual word representations based on neural network (Melamud et al., 2016; Peters et al., 2018) without considering word composition.

(Liu et al., 2015) proposed a TWE model which employs the topic assignments to produce topic-specific word embeddings to avoid the challenging task of word sense induction, and generates the representation of a larger unit by adding the vectors of topical word embeddings weighted by TFIDF of them. Fadaee et al. (2017) labels a word with topic distribution in both hard and soft ways, and learns topic-specific representations and general representation for a word simultaneously. Shi et al. (2017) introduces a method to enhance the word representation and topic model with each other. By contrast, our proposed model aims at utilizing topic distribution to enhance the word composition models for learning representations of linguistic units of different granularity levels, including word, phrase and sentence.

2.3 Word Composition

Due to the context sparsity problem, word composition is a promising technique to learn representations of linguistic units larger than words. A lot of models have been developed for word composition, e.g., simple models such as element-wise addition/multiplication (Mitchell and Lapata, 2010; Mikolov et al., 2013) and neural network based models such as recurrent neural network (Paulus et al., 2014), gated recurrent neural network, Long-Short Term Memory network (Le and Zuidema, 2015), convolution neural network (Xu et al., 2015) and attention-based model (Ling et al., 2015). These composition models could be used as the base composition model in our model-free framework.

Skip-Thought (Kiros et al., 2015) learns word and sentence representations based on sentences around it simultaneously. Doc2vec (Le and Mikolov, 2014) jointly trains the word sentence vectors by predicting the words in it. Recently, (Mao et al., 2017) introduced a topic-aware model to enhance the word composition model based on topic embeddings, which verifies the benefit of topic information for word composition. But despite its apparent success, there remains a major drawback: this method suffers from the limitations that learning topic embeddings and the uniform representation of words. By contrast, our method generates different vectors for a word in various context and only depends on the topic distributions of linguistic units.

3 Model-Free Context-Aware Word Composition Framework

In this section, we present our model-free context-aware word composition framework, which contains two main components: context-aware word representation and context-aware word composition. We first describe the method of learning a context-aware word embeddings. After that, we propose a model-free context-aware word composition framework to learn contextual representation of a linguistic unit.

3.1 Context-Aware Word Embeddings

This section describes how to learn the context-aware word representation of a contextual word based on contextual Information.

Contextual Information. In this paper, we represent contextual information using the most prevalent topic model (Blei et al., 2003), i.e., the context of a linguistic unit is represented as a topic distribution $C = \{p(Topic_1), p(Topic_2), ..., p(Topic_K)\}$. For example, the phrase context

'river bank' of the word *'bank'* may be represented as $\{Geography^{0.8}, Financial^{0.01}, ..., Sport^{0.1}\}$, by contrast the phrase context *'commercial bank'* of the word *'bank'* may be represented as $\{Geography^{0.02}, Financial^{0.85}, ..., Sport^{0.03}\}$. Note that, the proposed method can also use other context representation model, such as Melamud et al. (2016; Peters et al. (2018).

Using the above context representation model, we estimate the context-aware representation as follows: (1) We train a topic model on a large-scale text corpus using topic model; (2) Given the context of a word occurrence $S = [w_1, w_2, ..., w_k]$, where w_i represents a word, we employ collapsed Gibbs sampling algorithm (Griffiths and Steyvers, 2004) to infer the topic assignments for all words in S, and the topic distribution of S is utilized as contextual information.

Context-Aware Word Embedding Learning: Based on the above contextual information, a word can express distinct meanings under different topic assignments, and the unit representation can be learned separately for each individual topic, before being combined to constitute the final context-aware unit representation. For example, the word *'bank'* with topic assignments *'Geography'* and *'Financial'* respectively express two distinct senses of *'bank'*: *'riverside'* and *'financial institution'*. We employ the topic assignment as the contextual cue of a word in a specific context. In this way, we need to learn the embeddings of all $\langle word : topic \rangle$ pairs – we refer them topic-specific word embeddings. Formally, we learn a set of vectors for each word, with each vector corresponding to a specific topic. For instance, the word *'bank'* will be represented as:

$$\vec{V}_{Geo}(bank) = [0.628, 0.093, 0.051, ..], \vec{V}_{Fin}(bank) = [0.034, 0.016, 0.320, ..]$$

where $\vec{V}_{Geo}(bank)$ and $\vec{V}_{Fin}(bank)$ correspond to two distinct senses of *'bank'* under topics of *'Geography'* and *'Financial'*, respectively.

To learn the above topic-specific word embeddings from a text corpus, we first assign each word in the text corpus with a topic using the topic models; secondly we treat each $\langle word : topic \rangle$ pair as an individual unit; finally we learn the embeddings of all $\langle word : topic \rangle$ pairs using word embedding model, such as Word2Vec and Glove.

3.2 Context-aware Word Composition

In this section, we propose a model-free context-aware word composition framework, which represents a linguistic unit based on contextual information and context-aware word representation. Our method employs topic distribution of a linguistic unit as its contextual information. Specifically, given a linguistic unit $S = [w_1, w_2, ..., w_n]$, its topic distribution $C = \{p(topic_1|S), p(topic_2|S), ...\}$ and the topic-specific embeddings of all words, we construct the representation of S as follows:

1) We calculate the topic-biased representation of S under topic t by composing the topic-specific word embeddings as follows:

$$\vec{V}_{TopicBiased}(S, t) = f(\vec{V}_t(w_1), ..., \vec{V}_t(w_n)) \quad (1)$$

where f is a base composition model, which composes a sequence of vectors into a single representation. Notice that we can utilize any context-unaware word composition model as our base composition model, such as recurrent neural network, convolutional neural network, element-wise addition/multiplication, etc. For example, given the sentence *'we run along the bank'*, the topic-biased representation of the sentence under topic *'Geography'* is learned as:

$$\vec{V}_{TopicBiased}(S, Geo) = f(\vec{V}_{Geo}(we), ..., \vec{V}_{Geo}(bank))$$

2) We construct the context-aware representation of S by the weighted average of all topic-biased representations of S based on its topic distribution:

$$\vec{V}_{Contextual}(S) = \sum_{t \in T} p(t|S) \cdot \vec{V}_{TopicBiased}(S, t) \quad (2)$$

where $p(t|S)$ is the topic probability of topic t of S. For instance, using the inferred topic distribution of the sentence S as $\{Geography^{0.03}, Financial^{0.01}, .., Sport^{0.73}\}$, we generate the context-aware representation of S by averaging all the topic-biased vectors of S according to its topic distribution as:

$$\vec{V}_{Contextual}(S) = 0.03 \cdot \vec{V}_{TopicBiased}(S, Geo) + ... + 0.73 \cdot \vec{V}_{TopicBiased}(S, Sport)$$

To learn the context-aware representation for a word in a specific sentence,the latent topic distribution of the sentence and the topic-specific word embeddings are needed. Specifically, a word in a specific context can be learned by averaging the topic-specific word embeddings based on the topic distribution:

$$\vec{V}_{Contextual}(w|S) = \sum_{t \in T} p(t|S) \cdot \vec{V}_t(w) \tag{3}$$

where $\vec{V}_{contextual}(w|S)$ represents the context-aware representation of word w in context S, $p(t|S)$ is the topic probability of S under topic t, $\vec{V}_t(w)$ represents the topic-specific embedding of word w and T represents the set of topics. For example, given a sentence *S='Bank is a financial institution that accepts deposits'*, we first infer its topic distribution of the sentence as $C = \{Geography^{0.01}, Financial^{0.8}, ..., Sport^{0.03}\}$. Secondly we learn the context-aware representation of *'bank'* as follows:

$$\vec{V}_{Contextual}(bank|S) = 0.01 \cdot \vec{V}_{Geo}(bank) + 0.8 \cdot \vec{V}_{Fin}(bank) + 0.03 \cdot \vec{V}_{Sport}(bank) \tag{4}$$

where $\vec{V}_{sport}(bank)$ represents the topic-specific representation of *'bank'* under topic *'Sport'*.

4 Experiments

In this section, we assess our method on text similarity tasks at different granularities. And we conduct experiments on paraphrase detection task to evaluate the improvements of our method over various base word composition models.

4.1 Model Pre-training

In our experiments, we employ two different corpora for a comprehensive comparison to the various baseline models. We train the LDA model and the topic-specific word embeddings on the British National Corpus (BNC) (Consortium and others, 2007), which contains more than 93 million terms, and a bigger corpus – a snapshot of the English Wikipedia corpus[1] with about 990 million tokens.

Specifically, we use GibbsLDA++ (Phan and Nguyen, 2007) to estimate the topic distribution and infer the topic assignment for each word in the corpus. The parameters of GibbsLDA++ are empirically tuned as follows: $\alpha = 0.5, \beta = 0.1$, the number of topics 50, the number of iterations 400. For the topic-specific word embeddings, we use the SkipGram algorithm, with the dimension of word vector 300, the windows size 5, the number of iterations 5 and 10 negative samples per occurrence.

4.2 Overall Results

The proposed model focuses on learning the context-aware representations of large linguistic units. To evaluate the learned representations of linguistic units, we conduct experiments on text semantic similarity tasks at different granularities, including contextual word similarity, phrase similarity and sentence similarity. In this section, we calculate the similarity of two linguistic units based on cosine similarity of their representations, and we assess different systems using the Spearman's correlation between system outputs and gold standards manually labelled.

Contextual Word Similarity Results

To assess the quality of our context-aware word representations, we conduct experiments on the Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012), which contains 2003 manually labelled word pair similarities, with each word is paired with a sentence context.

We compare our context-aware word representation model with several baseline word embeddings models, including C&W (Collobert and Weston, 2008), CBOW and SkipGram (SG) (Mikolov et al., 2013). We also compare with two multi-prototype word embeddings models, including MSSG model (Neelakantan et al., 2015) and SAMS model (Cheng and Kartsaklis, 2015), both of which predict the

[1] https://www.wikipedia.org/

sense of a word based on its local context. All of the above models are trained on BNC. In addition, we compare with CBOW and SkipGram (SG) (Mikolov et al., 2013), Huang (Huang et al., 2012), Tian (Tian et al., 2014), TWE (Liu et al., 2015), STD-dif (Shi et al., 2017) and HTLE (Fadaee et al., 2017) models based on Wikipedia corpus for fair comparison, which have achieved state-of-the-art performances. And we report their best published result. The overall results are presented in Table 1.

Corpus	CBOW	SK	C&W	MSSG	SAMS	Tian	Huang	TWE(best)	STE-Dif	HTLE	Our
BNC	59.0	61.0	55.0	56.0	58.0	-	-	-	-	-	**63.2**
Wikipedia	65.3	65.7	-	-	-	65.4	65.3	68.1	68.0	63.0	**68.3**

Table 1: The Spearman's correlation ρ of different methods on SCWS dataset.

From Table 1, we can see that: (1) Our context-aware word representation had achieved the best performance on both the BNC and Wikipedia corpora. Compared with the three context-unaware word representation baselines CBOW, SkipGram and C&W, our method correspondingly achieves 6.8%, 3.3% and 14.5% ρ improvements. (2) The latent topic model based context representation can effectively capture the proper meanings of a contextual word. Compared with the multi-prototyped word embeddings baselines MSSG and SAMS, our method achieved 12.5% and 8.6% ρ improvements, respectively. We believe the topic distribution of the entire sentence is a beneficial cue for accurately representing polysemy words. (3) The results verify that the context-aware way of learning word representation as Formula (3) may provide a better way of utilizing topic distribution than hard label schema as in TWE, because both of the models implemented on the dataset.

Phrase Similarity Results

To assess the performance of our method on phrase representation learning, we conduct phrase similarity experiments on the ML2010 dataset (Mitchell and Lapata, 2010), which contains 108 pairs of phrases for adjective-noun (AN), verb-object (VO) and compound-noun (NN) respectively.

In our experiments, we employ the element-wise addition as base composition model, which turns out to be both robust and effective in many tasks. We compare our system with two baselines: the element-wise addition models correspondingly using the context-unaware word embeddings learned by CBOW and SkipGram. We also compare our system with state-of-the-art results from ML Original (Mitchell and Lapata, 2010), PAS(Hashimoto et al., 2014), PARAGRAM (Wieting et al., 2015b), DeepCCA (Lu et al., 2015) and vecDCS (Tian et al., 2016). The results of the models are taken from the original papers. The overall results are shown in Table 2.

	ML Original	PAS	PARAGRAM	DeepCCA	vecDCS	SkipGram	CBOW	Our
AN	0.46	0.46	0.50	0.45	0.41	0.47	0.45	**0.60**
NN	0.37	0.49	0.51	0.45	0.51	0.51	0.49	**0.54**
VO	0.45	0.45	0.40	0.47	0.49	0.39	0.40	**0.47**
ALL	0.44	0.47	0.47	0.46	0.47	0.41	0.43	**0.54**

Table 2: The Spearman's correlation ρ of different methods on ML2010 datasets.

From Table 2, we can see that: (1) By incorporating the topic distributions of phrases, our context-aware word composition model achieves the best results on all three types of phrases: compared with SkipGram and CBOW baselines, our method achieves 22.2% and 20.8% ρ improvements on average. (2) Compared with state-of-the-art systems, our method achieves the best performances on all types of phrases. The result verified the effectiveness of contextual information in phrase representation.

Sentence Similarity Results

To assess our method on sentence representations, we conduct experiments on the semantic textual similarity dataset (STS2015) from SemEval 2015, which contains five datasets of different genres: answers-students, answers-forums, belief, headlines and images.

In our experiments, the element-wise addition method is employed as the base composition model for our context-aware word composition model. We compare our method with two baselines: the element-

wise addition composition model using word embeddings learned from CBOW and SkipGram. All of the models in this experiment is trained based on Wikipedia corpus. We also compare to several strong baselines: skip-thought vector (ST) (Kiros et al., 2015) and average Glove vector (Pennington et al., 2014). Because our proposed model aims at improving word composition model based on topic distribution, we don't compare with the methods which incorporate syntax information or other resources (Wieting et al., 2015a). The overall results are showed in Table 3.

From Table 3, we can note that: our context-aware word composition method significantly improved the performances on all of the datasets by incorporating the topic distribution information. Compared with the baselines CBOW and SkipGram, our method achieved 35.5% and 19.2% ρ improvements on average. And the topic distribution is a beneficial resource for modelling the context of a sentence for learning better representation of the sentence.

Model	Answers students	Answers forums	Beliefs	Headlines	Images	Overall
ST	0.361	0.330	0.246	0.436	0.177	0.310
GloVe	0.305	0.630	0.405	0.618	0.675	0.527
SkipGram	0.413	0.475	0.392	0.581	0.621	0.496
CBOW	0.477	0.620	0.467	0.573	0.684	0.564
Our	**0.632**	**0.631**	**0.657**	**0.681**	**0.761**	**0.672**

Table 3: The Spearson's correlation ρ for different methods on STS2015 datasets.

All of the previous experiments have verified that the context-aware combination model is effective and can achieve better results than the baseline models. Furthermore, the proposed context-aware model achieved the greatest improvements over base models at sentence level, and made the least progress on word level. We suspect that because the local contextual information is utilized in learning word representation, which weakens the value of global context. By contrast, the contextual information is rarely taken into consideration by word composition models. As a result, we believe the global contextual information is beneficial for learning representations of larger linguistic units, such as sentence.

4.3 The Effect of Using Different Base Composition Models

Our method is model free that it can employ any context-unware composition models as its base word composition models. In this section, we intend to clarify whether the context-aware method could enhance the representations on different base composition models. Concretely, we conduct experiments on four of the most commonly used word composition models as base models: element-wise addition (Mikolov et al., 2013), recurrent neural network (RNN) (Mikolov et al., 2010; Socher et al., 2011b), gated recurrent neural network (GNN) (Tang et al., 2015) and Long-Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997; Le and Zuidema, 2015; Ghosh et al., 2016) on paraphrase detection task.

Paraphrase Detection

Paraphrase detection is a binary classification task which is used to decide whether two sentences express the same meanings, which is an appropriate task for evaluating our method. In this paper, we employ MSRPC datasets (Dolan et al., 2004).

Model	**C-unware**		**C-aware**	
	Acc	F	Acc	F
Addition	0.658	0.765	0.699	0.801
RNN	0.682	0.80	0.732	0.822
GNN	0.735	0.832	0.763	0.841
LSTM	0.739	0.838	**0.776**	**0.852**
ST	0.642	0.739	-	-
MSSG	-	-	0.692	0.793
SAMS	-	-	**0.786**	**0.853**

Table 4: Accuracies and F-Scores of paraphrase detection on MSRPC datasets.

For each base composition model, we compare our method with its context-unaware version. In addition, we compare with Skip-Thought (ST) vector as a strong baseline, MSSG and SAMS are used

Topic	Top 4 Frequent Words of Topic
Financial	money, million, cost, tax
Geography	river, lake, mountain, island
Information	user, systems, ibm, software
Sport	play, team, season, ball

Table 5: Top 4 topics of the word *'bank'*.

⟨ word:topic ⟩	Neighbors
bank:Geography	longitude:Geography, inland:Geography, coasts:Geography, southward:Geography
bank:Financial	bankers: Financial, debt: Financial, loans: Financial, finance: Financial

Table 6: The top 4 nearest neighbors of word *'bank'* with topics *'Geography'* and *'Financial'*.

as the multi-prototype word embeddings models for comparison. In this experiment, we implement a two layers RNN/GNN/LSTM with a max pooling layer as base word composition model. We randomly select 500 sentence pairs in training data as validation data to find the best parameters. We learn the best hyper-parameters based on validation data as follows: the hidden units for RNN/GNN/LSTM is 300, the mini-batch size as 100, the dropout rate is 0.4, the learning rate of SGD is 0.01. All of the word embeddings are trained on BNC (Consortium and others, 2007) for fair composition with SAMS. Accuracy and F-Score are used as the metrics. The results are shown in Table 4.

From Table 4, we can see that:(1) The context-aware representation consistently improved the performances on all the base word composition models. (2) Compared with Skip-thought vector model and MSSG model, our proposed model had achieved better performance, we believe the latent semantic information of a sentence may be the main reason that leads to the results. (3) SAMS achieved better results than our model, however, it utilized external knowledge resource (PPDB), which is a larger dataset for enhancing the performance on paraphrase detection task. The paraphrase detection results demonstrate that our context-aware representation brings beneficial information for different base word composition models and improves the performances on this task. The proposed model may achieve better results by utilizing more advanced base word composition models, we leave it for future work.

5 Detailed Analysis

To better understand the way of the proposed method works, this section provides detailed analysis on the quality of the topic-specific word embeddings and context-aware representations.

5.1 The Quality of Topic-Specific Word Representations

Topic-specific word representations are essential to our method, which is the basic unit for composing representations of larger linguistic units. In this section, we analyze the quality of topic distributions and topic-specific word embeddings.

For the reason that the proposed context-aware model is based on an assumption that a word can express distinct meanings with different topic assignments, so it is important to estimate whether topics can distinguish different senses of a word. Table 5 shows the top 4 topics of word *'bank'* and the most frequent words of these topics. From Table 5, we can infer that, *'bank'* with topic of *'Financial'* associates with the meanings of *'money'* or *'tax'*, and *'bank'* with *'Geography'* topic associates with the meanings of *'river'* or *'lake'*. In addition, We list the top 4 $\langle word : topic \rangle$ neighbors for *'bank:Geography'* and *'bank:Financial'* in Table 6. The conclusion can be drawn that the topic information can effectively distinguish distinct senses of a word.

5.2 The Quality of Context-Sensitive Word Composition

We analyze the quality of linguistic unit representations using the *'river bank'* and *'commercial bank'* as examples. We first present the top 3 most frequent topics of their words in Table 7.

Word	Topic Probabilities
bank	Financial(62.7%), Geography(23.4%), Information(7.3%)
river	Geography(89.7%), Information(0.07%) , Sport(0.01%)
commercial	Financial(86.3%), Information (6.5%), Law(5.0%)

Table 7: The top 3 most frequent topics of *'river'*, *'bank'* and *'commercial'*.

From Table 7, we can see that, both the word *'commercial'* and *'bank'* have high probability on topic *'Financial'*, so *'commercial bank'* will have a high probability belonging to topic *'Financial'*. In this way, the representation of *'commercial bank'* will mainly depend on the topic-biased representation under *'Financial'* topic, which is composed from topic-specific embeddings of *'commercial:Finanial'* and *'bank:Financial'*. By contrast, the representation of *'river bank'* will mainly depend on the embeddings of *'river:Geography'* and *'bank:Geography'*. In a word, the proposed model is capable of correctly identifying the important topics for a specific phrase, and generate proper representation for it.

In addition, we list top 5 nearest $\langle word : topic \rangle$ pairs of the phrases in Table 8. As shown in Table 8, all of the nearest neighbors of *'river bank'* are labelled with *'Geography'* topic and all of the nearest neighbors of *'commercial bank'* are related to its *'Financial'* topic. This result demonstrates the effectiveness of our method on learning accurate representations of phrases.

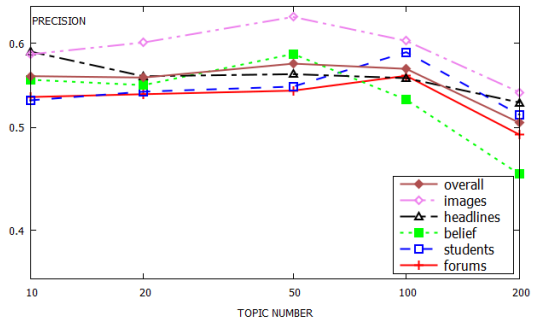


Figure 1: The precisions on STS2015 datasets with different numbers of topics in LDA model.

Phrase	$\langle word : topic \rangle$ **neighbors**
river bank	river:Geography, valley:Geography, park:Geography, north:Geography, boat: Geography
commercial bank	bank:Financial, leasing:Financial, merchandising: Financial, retailing: Financial, financing: Financial

Table 8: The top 5 nearest $\langle word : topic \rangle$ pairs for *'river bank'* and *'commercial bank'*.

The experimental results reveal that the topic distribution information could be used as the cue for learning proper representations of a linguistic unit in different contexts. And the proposed context-aware model can generate accurate representations for phrases.

5.3 The Number of Topics

One important parameter of our method is the number of topics of the LDA model. In order to detect the impact of the number of topics on our model, we conduct experiments using different topic numbers (including 10, 20, 50, 100 and 200) on STS2015 datasets with element-wise addition as base composition model, the results are shown in Figure 1.

From Figure 1, we can see that: (1) Our method is not very sensitive to the topic number, which achieved stable performances when using topic numbers range from 10 to 100. But the precision drops quickly when the number of topics is set as 200, we believe such large number of topics inevitably causes insufficient occurrences for learning topic-specific word embeddings, which is crucial basic units for our compositional method. (2) Our method achieved the best precision when the topic number is 50. We believe this is actually a trade-off between distinguishability and data sparsity.

6 Conclusions

In this paper, we have proposed a model-free context-aware word composition framework for text representation learning, which can effectively utilize the latent semantic information of the whole linguistic unit for learning context-aware representations for linguistic units. The proposed model is model-free in that our method can employ various context-unaware word composition models as the base model within our proposed framework. Experimental results demonstrated the effectiveness of our method on text representation learning at different granularities, including word, phrase and sentence, and the improvements on various base word composition models. In future work, to further improve the learned representations of linguistic units, we want to also take the non-compositional units into consideration, i.e., to solve the representations of multiword expressions like idioms or name entities, whose meanings

cannot be composed from the meanings of its constituent parts, such as *kill the goose that lays the golden eggs* and *speak of the devil*.

Acknowledgments

This work is supported by the Project of Beijing Advanced Innovation Center for Language Resources (451122512), the National Natural Science Foundation of China under Grants no. 61433015, 61772505 and 61572477, and the Young Elite Scientists Sponsorship Program no. YESS20160177. Moreover, we sincerely thank the reviewers for their valuable comments.

References

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Jianpeng Cheng and Dimitri Kartsaklis. 2015. Syntax-aware multi-sense word embeddings for deep compositional models of meaning. *arXiv preprint arXiv:1508.02354*.

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

British National Corpus Consortium et al. 2007. British national corpus version 3 (bnc xml edition). *Distributed by Oxford University Computing Services on behalf of the BNC Consortium. Retrieved February*, 13:2012.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *ACL (1)*, pages 1370–1380. Citeseer.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In *International Conference on Computational Linguistics*, page 350.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Learning topic-sensitive word representations.

Stella Frank, Naomi H Feldman, and Sharon Goldwater. 2014. Weak semantic context helps phonetic learning in a model of infant language acquisition. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1073–1083.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck. 2016. Contextual lstm (clstm) models for large scale nlp tasks. *arXiv preprint arXiv:1602.06291*.

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National academy of Sciences*, 101(suppl 1):5228–5235.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning sense-specific word embeddings by exploiting bilingual resources. In *COLING*, pages 497–507.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2014. Jointly learning word representations and composition functions using predicate-argument structures. In *Conference on Empirical Methods in Natural Language Processing*, pages 1544–1555.

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

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Sensembed: Learning sense embeddings for word and relational similarity. In *ACL (1)*, pages 95–105.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *Computer Science*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *Computer Science*, 4:1188–1196.

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. *arXiv preprint arXiv:1503.02510*.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Wang Ling, Yulia Tsvetkov, Silvio Amir, Ramon Fermandez, Chris Dyer, Alan W Black, Isabel Trancoso, and Chu Cheng Lin. 2015. Not all contexts are created equal: Better word representations with variable attention. In *Conference on Empirical Methods in Natural Language Processing*, pages 1367–1372.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *AAAI*, pages 2418–2424.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep multilingual correlation for improved word embeddings. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 250–256.

Kezhi Mao, Kezhi Mao, Kezhi Mao, and Kezhi Mao. 2017. Topic-aware deep compositional models for sentence classification. *IEEE/ACM Transactions on Audio Speech & Language Processing*, 25(2):248–260.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *Signll Conference on Computational Natural Language Learning*, pages 51–61.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

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

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2015. Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv preprint arXiv:1504.06654*.

Romain Paulus, Richard Socher, and Christopher D Manning. 2014. Global belief recursive neural networks. In *Advances in Neural Information Processing Systems*, pages 2888–2896.

Francis Jeffry Pelletier. 2001. Did frege believe frege's principle? *Journal of Logic, Language and information*, 10(1):87–114.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Xuan-Hieu Phan and Cam-Tu Nguyen. 2007. Gibbslda++: Ac/c++ implementation of latent dirichlet allocation (lda). *URL: http://gibbslda. sourceforge. net*.

Shimi Salant and Jonathan Berant. 2017. Contextualized word representations for reading comprehension. *arXiv preprint arXiv:1712.03609*.

Bei Shi, Wai Lam, Shoaib Jameel, Steven Schockaert, and Kwun Ping Lai. 2017. Jointly learning word embeddings and latent topics.

Richard Socher, Eric H Huang, Jeffrey Pennington, Andrew Y Ng, and Christopher D Manning. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*, volume 24, pages 801–809.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*, pages 151–161. Association for Computational Linguistics.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Idan Szpektor and Ido Dagan. 2008. Learning entailment rules for unary templates. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 849–856. Association for Computational Linguistics.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, pages 1422–1432.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *COLING*, pages 151–160.

Ran Tian, Naoaki Okazaki, and Kentaro Inui. 2016. Learning semantically and additively compositional distributional representations. In *Meeting of the Association for Computational Linguistics*, pages 1277–1287.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015a. Towards universal paraphrastic sentence embeddings. *Computer Science*.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015b. From paraphrase database to compositional paraphrase model and back. *Computer Science*, pages 98–104.

Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. 2015. Short text clustering via convolutional neural networks. In *Proceedings of NAACL-HLT*, pages 62–69.

Learning Features from Co-occurrences: A Theoretical Analysis

Yanpeng Li
IBM T. J. Watson Research Center
Yorktown Heights, New York 10598, USA
liyanpeng.lyp@gmail.com

Abstract

Representing a word by its co-occurrences with other words in context is an effective way to capture the meaning of the word. However, the theory behind remains a challenge. In this work, taking the example of a word classification task, we give a theoretical analysis of the approaches that represent a word X by a function $f(P(C|X))$, where C is a context feature, $P(C|X)$ is the conditional probability estimated from a text corpus, and the function f maps the co-occurrence measure to a prediction score. We investigate the impact of context feature C and the function f. We also explain the reasons why using the co-occurrences with multiple context features may be better than just using a single one. In addition, based on the analysis, we propose a hypothesis about the conditional probability on zero probability events.

1 Introduction

In natural language processing (NLP) and information retrieval (IR), representing a word by its co-occurrences with contexts is an effective way to learn the meaning of the word and lead to significant improvement in many tasks (Deerwester et al., 1990; Brown et al., 1992; Mikolov et al., 2013). The underlying intuition known as distributional hypothesis (Harris, 1954) can be summarized as: "a word is characterized by the company it keeps" (Firth, 1957). However, there is a lack of theory to justify why it works, even for a simple task such as word classification or clustering. For example, to determine if the word "phycoerythrin" is a gene name, we found it was effective to use the ratio of the count of "phycoerythrin gene" to the count of "phycoerythrin" as features (Li et al., 2009), denoted by

$$\hat{P}(\text{“}Xgene\text{”}|X = phycoerythrin) = \frac{Count(phycoerythrin\ gene)}{Count(phycoerythrin)}$$

The ratio is equivalent to the estimation of the conditional probability $P(\text{“}Xgene\text{”}|X = phycoerythrin)$ from a large text corpus. The pattern "$Xgene$" is a context feature that indicates if the word "gene" appears right next to the word X. Assuming $Y \in \{0, 1\}$ is the gold standard label such as "being a gene name" and $C \in \{0, 1\}$ is a binary context feature such as "$Xgene$". The nature of the approach is to predict $P(Y = 1|X)$ by $P(C = 1|X)$ based on the intuition that there could be correlation between these two functions of X. Also $P(Y = 1|X)$ tends to be more difficult to estimate than $P(C = 1|X)$, since the gold standard labels are more expensive to obtain than the context patterns. We need to know why and when $P(C = 1|X)$ is effective to predict $P(Y = 1|X)$. Obviously, this study is beyond computational linguistics only but of general interest of probability and statistical theory.

In this work, we first show that in a word classification task, simple co-occurrence with a single context feature can achieve perfect performance under some assumptions. Then we investigate the impact of context features and different co-occurrence measures without the assumptions. We also explain the reasons why using co-occurrences with multiple context features, e.g., vector representation can be better than just using a single one. In addition, we give a further analysis of the first theorem for the case of continuous random variables and discuss some hypothesis that may open the door to a new area in probability theory.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2846–2854
Santa Fe, New Mexico, USA, August 20-26, 2018.

2 Notions

We consider a word classification task that assigns each word in a text corpus one or more class labels by the co-occurrences of the word with one or more context patterns. By going through the occurrence of each word in the corpus, we define each example as a tuple $(word, context, labels)$ denoted by $(X, \mathbf{c}, \mathbf{y})$. The component *X* is a discrete random variable taking the values from the set $\{x_1, ..., x_m\}$, where each element refers to a different word. The component $\mathbf{c} = (C_1, ..., C_n)$ is a vector of features, where each component C_k(*k* is from 1 to *n*) is a random variable that takes the value 1 or 0, indicating if a word or pattern appears in the context. Similarly we define $\mathbf{y} = (Y_1, ..., Y_t)$ as a vector of binary labels such as concepts. We aim to investigate how well we can assign each example a set of labels **y** by the co-occurrence information of *X* and C_k with insufficient or without **y** at training stage.

Note that for a label variable Y, there are two cases. It may depend on both *X* and **c** or depend on *X* only. In the tasks such as word sense disambiguation or part-of-speech tagging, the same word can be assigned to different labels in different contexts. In some tasks that investigate the meaning of words independently of contexts, e.g., building WordNet (Miller, 1995) or calculating word-word similarity, the same word X can only be assigned to an unique label 0 or 1 for each *Y*. In other words, *Y* is function of *X*, that is, for each *X*, $P(Y|X)$ equals 0 or 1. We consider both cases in our analysis, but we find that we can get much simpler results for the second case. Moreover, the word classification task for the second case can be generalized to the task of annotating everything with words. For example, describing an image with several words or a sentence is closely related to the task of classifying words into the class of description for the given image.

In the following sections, when we aim to investigate the impact of a single context feature C_k or a single task Y_t, for simplicity, we use a random variable C or Y to denote C_k or Y_t respectively. So the tuple $(X, \mathbf{c}, \mathbf{y})$ can be simplified as (X, C, Y) in the setting of single context feature and single task. To avoid "dividing by zero", we assume that $P(X) \neq 0$, $P(Y) \neq 0$ and $P(C) \neq 0$ in all the sections except Section 6 where we discuss some open problems about the conditional probability on zero probability events.

3 Conditional independence assumption

We show that in a special case if a context feature C is conditionally independent with the word feature X on label Y, the Pearson correlation coefficient between $P(Y = 1|X)$ and $P(C = 1|X)$ equals 1 or -1.

Theorem 1. *Given random variables* $X \in \{x_1, ..., x_m\}$, $C \in \{0, 1\}$ *and* $Y \in \{0, 1\}$, *if*

1. $P(C = 1, Y = 1) \neq P(C = 1)P(Y = 1)$
2. $P(C = 1, X = x_i|Y = 1) = P(C = 1|Y = 1)P(X = x_i|Y = 1)$ *and* $P(C = 1, X = x_i|Y = 0) = P(C = 1|Y = 0)P(X = x_i|Y = 0)$ *for every i from 1 to m*

, we have:

$$Corr(P(Y = 1|X), P(C = 1|X))^2 = 1$$

Proof.

$$\begin{aligned}
P(C = 1|X) &= \frac{1}{P(X)}(P(C = 1, X|Y = 1)P(Y = 1) + P(C = 1, X|Y = 0)P(Y = 0)) \\
&= \frac{1}{P(X)}(P(C = 1|Y = 1)P(X|Y = 1)P(Y = 1) \\
&+ P(C = 1|Y = 0)P(X|Y = 0)P(Y = 0)) \\
&\qquad \text{(using the conditional independence assumption)} \\
&= \frac{1}{P(X)}(\frac{P(C = 1, Y = 1)P(X, Y = 1)}{P(Y = 1)} + \frac{P(C = 1, Y = 0)P(X, Y = 0)}{P(Y = 0)})
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{P(X)}\left(\frac{P(C=1,Y=1)P(X,Y=1)}{P(Y=1)}\right. \\
&\left.+ \frac{(P(C=1)-P(C=1,Y=1))(P(X)-P(X,Y=1))}{P(Y=0)}\right) \\
&= \left(\frac{P(C=1,Y=1)}{P(Y=1)} - \frac{P(C=1)-P(C=1,Y=1))}{P(Y=0)}\right)P(Y=1|X) \\
&+ \frac{P(C=1)-P(C=1,Y=1)}{P(Y=0)} \\
&= \frac{P(C=1,Y=1)-P(C=1)P(Y=1)}{P(Y=1)P(Y=0)}P(Y=1|X) + P(C=1|Y=0)
\end{aligned}
$$

Since $P(C=1,Y=1) \neq P(C=1)P(Y=1)$, $P(C=1|X)$is a linear function of $P(Y=1|X)$, so we have $Corr(P(Y=1|X), P(C=1|X))^2 = 1$ □

This theorem indicates that a perfect context feature for a given task is not necessarily the class label itself. Therefore, even if we have little information about Y, we still have the chance to know the information of $P(Y|X)$ by $P(C|X)$. Some previous works also reported similar findings in different tasks (Blum and Mitchell, 1998; Abney, 2002; Veeramachaneni and Kondadadi, 2009; Li, 2013). However, in practice it is almost impossible to get the case with exact conditional independence in different tasks. Even if it exists, it is difficult to find it because in order to calculate the conditional dependence we still need to know the joint probability $P(X,C,Y)$. Therefore, we need a theory to describe the performance in the situation with certain degree of conditional dependence. We also need to know the impact of different functions that convert simple co-occurrence measures e.g., $P(C=1|X)$ to feature values or prediction scores. In the following sections, we will show the results for the cases.

4 Co-occurrences with a single context feature

In this section, we study the case of co-occurrence with a single context feature for a single task. Although it seems simple, there is still no systematic theoretical framework for it. Based on the theory about the function of discrete random variables, we analyze the impact of context features and the functions that convert co-occurrence measures to prediction scores. Note that the cases discussed in this section are not under the conditional independence assumption introduced in Section 3.

4.1 Function of a discrete random variable

Since the co-occurrence based learning actually converts the word variable $X \in \{x_1, ..., x_m\}$ to another variable $S \in \{s_1, ..., s_u\}$ by certain co-occurrence measure such as $P(X, C=1)$ or $P(C=1|X)$, or equivalently, we can say S is a function of X. Our goal is to find the function that can achieve the best performance. Therefore, we investigate some general principles about the function of a discrete variable. In this work, we use Pearson correlation coefficient of conditional probabilities as the measure of performance, because it tends to make the complicated analysis simpler.

Lemma 1. *Let g be a function that maps $X \in \{x_1, ..., x_m\}$ to another random variable $S \in \{s_1, ..., s_u\}$. For any binary random variable $Y \in \{0,1\}$ and any function f that assigns S a real number, we have the following results:*

(1) $Cov(P(Y=1|X), f(S)) = Cov(P(Y=1|S), f(S))$

(2) $Cov(P(Y=1|X), P(Y=1|S)) = Var(P(Y=1|S))$

(3) $Corr(P(Y=1|X), P(Y=1|S)) = \frac{\sqrt{Var(P(Y=1|S))}}{\sqrt{Var(P(Y=1|X))}}$

(4) $Corr(P(Y=1|X), f(S)) = Corr(P(Y=1|S), f(S))Corr(P(Y=1|X), P(Y=1|S))$

Proof. (1).

$$E(P(Y=1|X)) = \sum_{i=1}^{m} \frac{P(X=x_i)P(Y=1,X=x_i)}{P(X=x_i)} = \sum_{i=1}^{m} P(Y=1,X=x_i) = P(Y=1)$$

Similarly, we can prove $E(P(Y=1|S)) = P(Y=1) = E(P(Y=1|X))$

$$Cov(P(Y=1|X), f(S)) = Cov(P(Y=1|X), f(g(X))) \qquad \text{(the definition of } g\text{)}$$
$$= \sum_{i=1}^{m} P(X=x_i)(P(Y=1|X=x_i) - E(P(Y=1|X)))(f(g(x_i)) - E(f(g(X))))$$
$$= \sum_{j=1}^{u} (f(s_j) - E(f(S))) \sum_{i\in\{i|g(x_i)=s_j\}} P(X=x_i)(P(Y=1|X=x_i) - E(P(Y=1|X)))$$
$$= \sum_{j=1}^{u} (f(s_j) - E(f(S)))P(S=s_j)(P(Y=1|S=s_j) - E(P(Y=1|X)))$$
$$= \sum_{j=1}^{u} P(S=s_j)(f(s_j) - E(f(S)))(P(Y=1|S=s_j) - E(P(Y=1|S)))$$
$$\text{(by } E(P(Y=1|X) = E(P(Y=1|S))\text{)}$$
$$= Cov(P(Y=1|S), f(S))$$

(2). Since $P(Y=1|S)$ is a function of S, let $f(S)$ be $P(Y=1|S)$ and using (1), and we have $Cov(P(Y=1|X), P(Y=1|S)) = Cov(P(Y=1|S), P(Y=1|S)) = Var(P(Y=1|S))$

(3). Based on (2), we have

$$Corr(P(Y=1|X), P(Y=1|S)) = \frac{Cov(P(Y=1|X), P(Y=1|S)))}{\sqrt{Var(P(Y=1|X))}\sqrt{Var(P(Y=1|S))}}$$
$$= \frac{Var(P(Y=1|S)))}{\sqrt{Var(P(Y=1|X))}\sqrt{Var(P(Y=1|S))}} = \frac{\sqrt{Var(P(Y=1|S)))}}{\sqrt{Var(P(Y=1|X))}}$$

(4). By combining (1) and (3), we have

$$Corr(P(Y=1|X), f(S)) = \frac{Cov(P(Y=1|X), f(S)))}{\sqrt{Var(P(Y=1|X))}\sqrt{Var(f(S))}}$$
$$= \frac{Cov(P(Y=1|S), f(S)))}{\sqrt{Var(P(Y=1|X))}\sqrt{Var(f(S))}} = \frac{Cov(P(Y=1|S), f(S)))}{\sqrt{Var(P(Y=1|S))}\sqrt{Var(f(S))}} \frac{\sqrt{Var(P(Y=1|S))}}{\sqrt{Var(P(Y=1|X))}}$$
$$= Corr(P(Y=1|S), f(S))Corr(P(Y=1|X), P(Y=1|S))$$

□

The first equation plays a fundamental role, which indicates that if S is function of X, to calculate the covariance between $P(Y=1|X)$ and $f(S)$, we don't need the appearance of X but just S, as if X is safely hidden. It simplifies the analysis by avoiding explicit analysis of the divergence between $P(Y=1|X)$ and $f(S)$ such as $|P(Y=1|X) - f(S)|$ or $P(Y=1|X)f(S)$. Therefore, the correlation coefficient can also be written in a much simpler form. In the word classification task, the term $f(S)$ can be viewed as a classifier based on the new features S. The $Corr(P(Y=1|X), f(S))$ measures its correlation with the best possible classifier $P(Y=1|X)$ based on the word features X (single view) only. If Y is a function of X (the second case discussed in Section 2), we have $Corr(P(Y=1|X), f(S)) = Corr(Y, f(S))$, which measures almost exactly the performance of the word classification task. Based on these results, we are able to analyze the performance of the new features generated by co-occurrences.

4.2 An upper bound of any function

Theorem 2. *Given random variables $X \in \{x_1, ..., x_m\}$, $C \in \{0, 1\}$ and $Y \in \{0, 1\}$, for any function f that maps $P(C = 1|X)$ to a real number, there is:*

$$Corr(P(Y = 1|X), f(P(C = 1|X)))^2$$
$$= Corr(P(Y = 1|P(C = 1|X)), f(P(C = 1|X)))^2 Corr(P(Y = 1|X), P(Y = 1|P(C = 1|X)))^2$$

Proof. Since $P(C = 1|X)$ is a function of X, let S be the random variable that takes the value $P(C = 1|X)$ and based on Lemma 1 (4), we have:

$$Corr(P(Y = 1|X), f(S))^2 = Corr(P(Y = 1|S), f(S))^2 Corr(P(Y = 1|X), P(Y = 1|S))^2$$

Equivalently, we can write the equation as:

$$Corr(P(Y = 1|X), f(P(C = 1|X)))^2$$
$$= Corr(P(Y = 1|P(C = 1|X)), f(P(C = 1|X)))^2 Corr(P(Y = 1|X), P(Y = 1|P(C = 1|X)))^2$$

□

It shows that the performance of the classifier $f(P(C = 1|X))$ is determined by the product of two parts. The first part depends on both the context feature C and the function f. The second depends on the context feature only but not on f. Therefore, if we fix the context feature C to select the function f, using $Corr(P(Y = 1|X), f(P(C = 1|X)))^2$ (correlation with the gold standard $P(Y = 1|X)$) and $Corr(P(Y = 1|P(C = 1|X)), f(P(C = 1|X)))^2$ (correlation with the "silver" standard $P(Y = 1|P(C = 1|X))$) produce the same result. Similar to Lemma 1(1), we don't need the appearance of $P(Y = 1|X)$ for function selection. In addition, since any correlation coefficient ranges from -1 to 1, we have:

$$Corr(P(Y = 1|X), f(P(C = 1|X)))^2 \leq Corr(P(Y = 1|X), P(Y = 1|P(C = 1|X)))^2$$

It indicates that if we want to improve the performance by fixing a certain context feature C and changing different functions f , e.g., from $P(C = 1|X)$ to $log(P(C = 1, X)/(P(C = 1)P(X)))$ (the point wise mutual information), the performance cannot be arbitrarily high but upper bounded by the squared correlation coefficient between $P(Y = 1|X)$ and $P(Y = 1|P(C = 1|X))$. Given fixed word feature set X and label Y, the upper bound is determined only by the context feature C but not relevant to the function f. Actually, the upper bound is a special case of maximal correlation coefficient (Rényi, 1959; Hall, 1967). In order to build a high performing classifier $f(P(C = 1|X))$, we must find a way to improve the upper bound $Corr(P(Y = 1|X), P(Y = 1|P(C = 1|X)))^2$, in other words, to select a good context feature C. However, it is not easy to do it from this formula directly. Therefore, we relate the upper bound to the special case introduced in Section 3. Since $P(C = 1|X)$ is a function of itself, based on Theorem 2, we have:

$$Corr(P(Y = 1|X), P(C = 1|X))^2 \leq Corr(P(Y = 1|X), P(Y = 1|P(C = 1|X)))^2$$

The squared correlation coefficient $Corr(P(Y = 1|X), P(C = 1|X))^2$ not only reflects the performance of a special co-occurrence measure $P(C = 1|X)$, but also provide a way to improve the upper bound (for other co-occurrence measures). Based on Theorem 1, under the conditional independence assumption, $Corr(P(Y = 1|X), P(C = 1|X))^2$ equals 1, so that $Corr(P(Y = 1|X), P(Y = 1|P(C = 1|X)))^2$ equals 1 as well. As what we discussed in Section 3, it is important to know more about the cases out of the conditional independence assumptions, but it is difficult to analyze $Corr(P(Y = 1|X), P(C = 1|X))^2$ based on the conditional dependence directly without any other assumptions. In the following section, we show that we are able to obtain much simpler results under the assumption that the label variable $Y \in \{0, 1\}$ is a function of X.

4.3 Assuming that *Y* is a function of *X*

In Section 2, we have pointed out that there are many cases in practice that assume *Y* is a function of *X*. In addition, if we can explain everything about this simple task, we may naturally find the cues to move to more advanced tasks such as sentence classification.

Theorem 3. *If* $Y \in \{0,1\}$ *is a function of* $X \in \{x_1, ..., x_m\}$*, for a context feature* $C \in \{0,1\}$ *we have:*

$$Corr(P(Y=1|X), P(C=1|X))^2$$
$$= Corr(P(Y=1|Y), P(C=1|Y))^2 Corr(P(C=1|X), P(C=1|Y))^2$$

Proof. Since *Y* and *C* are both binary random variables, by Lemma 1 (4) we replace Y by C without loss of generality, and we have:

$$Corr(C=1|X), f(S))^2 = Corr(P(C=1|S), f(S))^2 Corr(P(C=1|X), P(C=1|S))^2$$

Since *Y* is a function of *X* and $P(Y=1|Y)$is a function of Y, in the above equation replace *S* by *Y* and $f(S)$ by $P(Y=1|Y)$, and we have:

$$Corr(C=1|X), P(Y=1|Y))^2 = Corr(P(C=1|Y), P(Y=1|Y))^2 Corr(P(C=1|X), P(C=1|Y))^2$$

Since *Y* is a function of *X*, for any X, there is $P(Y=1|X) = P(Y=1|Y) = Y$. So, we have:

$$Corr(C=1|X), P(Y=1|X))^2 = Corr(C=1|X), P(Y=1|Y))^2$$
$$= Corr(P(Y=1|Y), P(C=1|Y))^2 Corr(P(C=1|X), P(C=1|Y))^2$$

□

This result describes what a good context feature is and can give guidance for the selection of context features for a particular task. It is much simpler and more practical than what we can get by the direct analysis of the conditional dependence, since we just need to know the joint distribution of (X, C) and (C, Y) rather than (X, C, Y). This result can also be written as a simpler form as follows.

Corollary 1. *If* $Y \in \{0,1\}$ *is a function of* $X \in \{x_1, ..., x_m\}$*, for a context feature* $C \in \{0,1\}$ *we have:*

$$Corr(P(Y=1|X), P(C=1|X))^2 = \frac{(P(C=1, Y=1) - P(C=1)P(Y=1))^2}{P(Y=1)(P(Y=0))Var(P(C=1|X))}$$

Proof.

$$\begin{aligned}
&Corr(P(Y=1|X), P(C=1|X))^2 \\
&= Corr(P(Y=1|Y), P(C=1|Y))^2 Corr(P(C=1|X), P(C=1|Y))^2 && \text{(Theorem 3)} \\
&= \frac{Cov(P(Y=1|Y), P(C=1|Y))^2}{Var(P(Y=1|Y))Var(P(C=1|Y))} \frac{Var(P(C=1|Y))}{Var(P(C=1|X)} && \text{(Lemma 1 (3))} \\
&= \frac{(E(P(Y=1|Y)P(C=1|Y)) - E(P(Y=1|Y))E(P(C=1|Y)))^2}{Var(P(Y=1|Y))} \frac{1}{Var(P(C=1|X)} \\
&= \frac{(P(C=1, Y=1) - P(C=1)P(Y=1))^2}{P(Y=1)(P(Y=0))Var(P(C=1|X))}
\end{aligned}$$

□

In the equation, the term $(P(C=1, Y=1) - P(C=1)P(Y=1))^2$ describes the dependency between context feature C and the class label Y. The term $Var(P(C=1|X))$ addresses the dependency between context feature C and each word feature X. We can see more clearly in another equivalent form:

$$Corr(P(Y=1|X), P(C=1|X))^2 = \frac{P(Y=1)}{P(Y=0)} \frac{(\frac{P(C=1,Y=1)}{P(C=1)P(Y=1)} - 1)^2}{E((\frac{P(C=1,X)}{P(C=1)P(X)} - 1)^2)}$$

From this mutual information style form, we can conclude that a good context feature for a particular task should be a trade-off between high dependence with label Y and low dependence with every word feature X. The conclusion is similar to our prevsious work (Li, 2013), but the result here is more general, under weaker assumptions, and more accurate (e.g., equations rather than inequalities). During the engineering work of context feature selection, intuitively, people tend to emphasize the numerator (tight connection with the label Y), but ignore the denominator (loose connection with each word X). However, the theory indicates that we should consider both factors to make a good co-occurrence. For example, some experiments showed that the co-occurrences with simply high frequency words such as "the" and "of" chould be used as good features (Li and Yu, 2014). Obviously, for these function words, in the equation both numerator and denominator tend to be smaller than the class-indicative words such as "$Xgene$" mentioned in Section 1, but the ratio of them may be larger. Therefore, it will be interesting to investigate what happen if both numerator and denominator approach to zero. Actually, the scope of candidates of context features is far beyond context words, since any binary pattern such as letters or sound that appears in the context can be a context feature.

5 Co-occurrences with multiple context features

So far, we have investigated the co-occurrences with a single context feature only. In practice, we usually find representing a word by a vector of co-occurrences with multiple context features tends to perform better. In the following theorem, we show part of the reason by analyzing the performance of the vector from multiple context features.

Theorem 4. *If we represent each X by a r-dimensional vector* $\mathbf{x} = (P(C_1 = 1|X), ..., P(C_r = 1|X))$*, where* $C_1, ..., C_r$ *are r context features (*$r \leq n$*), for the label* $Y \in \{0, 1\}$ *and any funcation f that maps each veoctor* $\mathbf{x}$ *to a real number, we have:*

(1) $Corr(P(Y=1|X), f(\mathbf{x}))^2 = Corr(P(Y=1|\mathbf{x}), f(\mathbf{x}))^2 Corr(P(Y=1|X), P(Y=1|\mathbf{x}))^2$

(2) $Corr(P(Y=1|X), P(Y=1|\mathbf{x}))^2 = \frac{Corr(P(Y=1|X), P(Y=1|P(C_k=1|X)))^2}{Corr(P(Y=1|\mathbf{x}), P(Y=1|P(C_k=1|X)))^2}$ *for every k from 1 to n*

Proof. (1) Since X is a discrete random variable and $\mathbf{x} = (P(C_1 = 1|X), ..., P(C_r = 1|X))$, we know that $\mathbf{x}$ is a discrete random variable and a function of X as well. Therefore, based on Lemma 1 (4), we come to the conclusion directly:

$$Corr(P(Y=1|X), f(\mathbf{x}))^2 = Corr(P(Y=1|\mathbf{x}), f(\mathbf{x}))^2 Corr(P(Y=1|X), P(Y=1|\mathbf{x}))^2$$

(2) Since for each $\mathbf{x} = (P(C_1 = 1|X), \ldots, P(C_r = 1|X))$ with the a different value, the value of $P(C_k = 1|X)$is unique, the discrete random variable $P(C_k = 1|X)$ is a function of $\mathbf{x}$. In Lemma 1 (3), replace X by $\mathbf{x}$, and S by $P(C_k = 1|X)$ without loss of generality, and we have

$$Corr(P(Y=1|\mathbf{x}), P(Y=1|P(C_k=1|X)))^2 = \frac{Var(P(Y=1|P(C_k=1|X)))}{Var(P(Y=1|\mathbf{x}))}$$

Since $\mathbf{x}$ is a function of X and $P(C_k = 1|X)$ is a function of X, according to Lemma 1 (3), we have

$$Corr(P(Y=1|X), P(Y=1|\mathbf{x}))^2 = \frac{Var(P(Y=1|\mathbf{x}))}{Var(P(Y=1|X))}$$

$$Corr(P(Y=1|X), P(Y=1|P(C_k=1|X)))^2 = \frac{Var(P(Y=1|P(C_k=1|X)))}{Var(P(Y=1|X))}$$

By combining the three equations above, we have:

$$Corr(P(Y=1|X), P(Y=1|\mathbf{x})) = \frac{Corr(P(Y=1|X), P(Y=1|P(C_k=1|X)))}{Corr(P(Y=1|\mathbf{x}), P(Y=1|P(C_k=1|X))^2}$$

□

Similar to Theorem 2, the performance of the vector-style representation can also be written as the products of two parts. The second part $Corr(P(Y=1|X), P(Y=1|\mathbf{x}))^2$ is an upper bound of the performance of any function, which is determined only by the set of context features $C_1, ..., C_r$. The result (2) shows that the upper bound from the vector is always better than the performance from any individual context feature, since we can prove $Corr(P(Y=1|X), P(Y=1|\mathbf{x})) \geq Corr(P(Y=1|X), P(Y=1|P(C_k|X)))^2$. Similar to the proof of (2), it can be proved that $Corr(P(Y=1|X), P(Y=1|\mathbf{x}))^2$ always increases when more context features are introduced. However, $Corr(P(Y=1|\mathbf{x}), f(\mathbf{x}))^2$ may decrease when the vector become longer, so its real performance $Corr(P(Y=1|X), f(\mathbf{x}))^2$ may increase or decrease depending on the first term. Therefore, we need to find the factors that determine the combined performance of the two terms in the future.

6 Assuming that the probability of *X* is zero

In the previous sections, we discussed the impact of context feature C. In addition, we should also consider the impact of the content feature X. We found an interesting case that satisfies exactly the conditional independence assumption in Theorem 1, although it is beyond the word classification task. If we assume that X is a continuous random variable such as $X \in \mathbb{R}$, we have $P(X=x)=0$, $P(X=x, C|Y)=0$, and $P(X=x|Y)=0$ for every real number $x \in \mathbb{R}$, which fits exactly the conditional independence assumption. The trouble is that we have the conditional probability $P(C=1|X)$ and $P(Y=1|X)$ in the format of $0/0$. It is well known that the conditional probability on an event with zero probability is undefined in the current probability theory, but there is no proof that it cannot be defined logically. If we assume that in some condition $P(C=1|X)$ and $P(Y=1|X)$ exist and are between 0 and 1, for example, as if we may define $f(x)=\frac{sin(x)}{x}$ when $x=0$ as $f(0)=\lim_{x\to 0}\frac{sin(x)}{x}$, the open question is: does the following equation (from Section 3) still hold?

$$P(C=1|X) = \frac{P(C=1,Y=1)-P(C=1)P(Y=1)}{P(Y=1)P(Y=0)}P(Y=1|X) + P(C=1|Y=0)$$

If the answer is "Yes", it suggests that given every specific point $X=x$, the possibility of appearance of any two non-independent events (e.g., $C=1$ and $Y=1$) has a linear relationship. One underlying philosophy is that two things with low dependence (e.g., low semantic similarity) could have strong correlation in some case, since $Corr(P(C=1|X), P(Y=1|X))$ is always 1 or -1 even if $|\frac{P(C=1,Y=1)-P(C=1)P(Y=1)}{P(Y=1)P(Y=0)}|$ is very small but not zero. Based on it we may have even more interesting findings. If the answer is "No", we would have a more complicated result, e.g., two or more different equations: one for $P(X)\neq 0$ and one or more for $P(X)=0$. An interesting fact is that for Bayes' theorem the answer of the similar question is "Yes", since if we assume $P(X)=0$ and $P(C=1|X)\in[0,1]$, we will have $P(X|C=1)=P(C=1|X)P(X)/P(C=1)=0$.

We also believe that such probability is more useful in practice because it gives to more exact knowledge and prediction. For example, assuming that $X=x$ is a specific time or position represented by a real number, $P(Y=1|X=x)$ (where $P(X=x)=0$) could mean the possibility of the occurrence of Y at the point of time or position x, which is more exact than the probability $P(Y=1|X\in[x1,x2])$, where $P(X\in[x1,x2])\neq 0$. Interestingly, such model is closely related to physics, and we will give a further analysis in the future.

7 Conclusions and future works

In this paper, we give a theoretical study of the approaches that learn the representation of word by the approaches like $f(P(C=1|X))$ or $f(P(C_1=1|X), ..., P(C_r=1|X))$. The theoretical framework is able to explain a set of approaches based on distributional semantics and give guidance for algorithm

design such as the selection of context feature and the co-occurrence metrics. In the next steps, we are going to give a deeper analysis of each formula that determines the performance, such as the diversity between multiple context features, find more principles that are constructive to algorithm design in practice and extend the theory to analyze other advanced tasks such as sentence classification and semantic similarity. Moreover, it will be interesting to see if we can verify the hypothesis in Section 6 and get inspiration from it.

References

[Deerwester et al.1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. and Harshman, R. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6), 391.

[Brown et al.1992] Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D. and Lai, J. C. 1992. Class-based N-gram Models of Natural Language. *Computational linguistics*, 18(4), 467-479.

[Mikolov et al.2013] Mikolov, T., Chen, K., Corrado, G. and Dean, J. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint*, 1301.3781.

[Harris1954] Harris, Z. S. 1954. Distributional Structure. *Word*, 10(2-3), 146-162.

[Firth1957] Firth, J. R. 1957. A Synopsis of Linguistic Theory. *Studies in Linguistic Analysis*, 1930-1955.

[Li et al.2009] Li, Y., Lin, H. and Yang, Z. 2009. Incorporating Rich Background Knowledge for Gene Named Entity Classification and Recognition. *BMC Bioinformatics*, 10(1), 223

[Miller1995] Miller, G. A. 1954. WordNet: a Lexical Database for English. *Communications of the ACM*, 38(11), 39-41.

[Blum and Mitchell1998] Blum, A. and Mitchell, T. 1998. Combining Labeled and Unlabeled Data with Co-training *Proceedings of the eleventh annual conference on Computational learning theory*, 92-100

[Abney2002] Abney, S. 2002. Bootstrapping. *Proceedings of the 40th Annual Meeting of the Association for Computa- tional Linguistics*, 360–367.

[Veeramachaneni and Kondadadi2009] Veeramachaneni, S. and Kondadadi, R. K. 2009. Surrogate Learning: from Feature Independence to Semi-supervised Classification *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*, 10-18.

[Li2013] Li, Y. 2013. Reference Distance Estimator. *arXiv preprint*, 1308.3818.

[Rényi1959] Rényi, A. 1959. On Measures of Dependence. *Acta mathematica hungarica*, 10(3-4), 441-451.

[Hall1967] Hall, W. J. 1967. On Characterizing Dependence in Joint Distributions. University of North Carolina, Department of Statistics.

[Li and Yu2014] Li, Y. and Yu, H. 2014. A Robust Data-driven Approach for Gene Ontology Annotation. *Database*.

Towards a unified framework for bilingual terminology extraction of single-word and multi-word terms

Jingshu Liu[1,2], Emmanuel Morin[1], and Sebastián Peña Saldarriaga[2]

[1]LS2N - UMR CNRS 6004, Université de Nantes, France
[2]Dictanova, Nantes, France
[1]{jingshu.liu, emmanuel.morin}@ls2n.fr
[2]{jingshu, spenasaldarriaga}@dictanova.com

Abstract

Extracting a bilingual terminology for multi-word terms from comparable corpora has not been widely researched. In this work we propose a unified framework for aligning bilingual terms independently of the term lengths. We also introduce some enhancements to the context-based and the neural network based approaches. Our experiments show the effectiveness of our enhancements over previous works and that the system can be adapted in specialized domains.

Title and Abstract in French

Vers un système unifié pour l'extraction terminologique bilingue de termes simples et complexes

L'extraction d'une terminologie bilingue pour les termes complexes à partir de corpus comparables n'a pas été beaucoup étudiée. Dans ce travail, nous proposons un système unifié pour l'alignement des termes bilingues indépendamment de la longueur des termes. De plus nous introduisons également des améliorations aux approches basées sur l'alignement de contexte et sur un réseau neuronal. Nos expériences montrent l'efficacité de nos améliorations sur les travaux antérieurs, et le fait que le système peut être adapté en domaines de spécialité.

1 Introduction

Bilingual terminology extraction from comparable corpora has aroused a lot of attention since the 1990s (Fung, 1995; Rapp, 1999). Two classes of approach have been developed depending on the nature of the term to be aligned. The first class concerns the alignment of single-word terms using context-based or neural network approaches while the second attempts to align multi-word terms relying on compositional approaches. Few studies have focused on providing a unified framework for aligning single-word and multi-word terms, apart from Delpech et al. (2012) and Taslimipoor et al. (2016). The first requires some specific linguistic information like the morpheme translation table, which makes it difficult to employ for two languages from different linguistic families such as English and Chinese. The second incorporates word embeddings into their collocation alignment system but it is limited to several types of mapping that must match a set of pre-defined syntactic patterns. Our objective is to provide such a unified framework for aligning terms of variable length in specialized domains without specific linguistic knowledge of the source or target language.

Large size comparable corpora in specialized domains are not always available. Consequently, many data driven systems cannot learn from enough information. A possible solution to this problem is to associate external data such as general domain corpora (Hazem and Morin, 2016) to the specialized corpora. Our work adapts this method in order to improve the system performance.

Besides the enrichment of the data, our work focuses on studying and improving different state-of-the-art approaches both for single-word term and multi-word term alignments, which are finally unified into a single framework for alignment of terms of any length. We first describe the context-based projection

Proceedings of the 27th International Conference on Computational Linguistics, pages 2855–2866
Santa Fe, New Mexico, USA, August 20-26, 2018.

approach, and based on discussions in previous works, we introduce two enhancements which improve our final results. Secondly, we cover the bilingual word embedding approach which depends on vector representations of words that can be learned on large text collections using different neural networks (Bengio et al., 2003; Mnih and Hinton, 2008; Collobert and Weston, 2008; Mikolov et al., 2013b), then we also propose an enhancement that re-normalizes the word embedding vector length in order to improve the results.

After studying the approaches for single-word terms which are our fundamental components for processing multi-word terms, following the idea in Morin and Daille (2012), we propose a new system called Compositional Approach with Word Embedding Projection (CMWEP) to combine the advantages of the traditional compositional approach and the bilingual word embedding approach. Our final results show considerable improvements over the state-of-the-art approach for bilingual multi-word term extraction. Moreover, the proposed method is able to align variable length terms in a single process.

2 Single-Word Term Alignment

In this section, we describe the two principal state-of-the-art approaches used for bilingual lexicon extraction from comparable corpora. Furthermore we introduce our enhancements in order to overcome various limitations of these approaches. The reason why we want to study these approaches for single-word terms (SWTs) is that they are fundamental low-level elements in our approaches for multi-word terms (MWTs).

The two approaches for SWTs are known as distributional and distributed semantics (Hermann and Blunsom, 2014). Both use word vector representations. The vectors in the first approach are sparse, high dimensional and explicit (Levy and Goldberg, 2014). The vectors in the second one are dense, low dimensional and generalized. They both rely on the distributional hypothesis (Harris, 1968) which assumes that a word and its translation tend to appear in the same lexical contexts.

2.1 Context-Based Projection Approach

The historical context-based projection approach, also known as the standard approach (SA) has been studied in a variety of works (Fung, 1995; Rapp, 1999; Chiao and Zweigenbaum, 2002; Bouamor et al., 2013; Hazem and Morin, 2016; Jakubina and Langlais, 2017). To implement this approach, we first build a co-occurrence matrix for the source and target languages, where each line represents a context vector in an n-word window. These vectors are then normalized using the Mutual Information (MI (Fano, 1961)) for instance. Then we get the word in the target language vector space by projecting each element in the context vector via a bilingual seed lexicon. Finally, the candidate translations are ranked by calculating the similarity of the projected context vector with all the context vectors into the target language. We use the Cosine similarity measure as it is the one most used in previous works. In addition it enables us to parallelize the process of similarity comparison.

Due to the small size of specialized domain corpora, occurrences of words are not always statistically reliable. In order to improve word co-occurrence counts, Hazem and Morin (2016) show that using a general language corpus can significantly improve the standard approach results. They suggest two methods for exploiting external resources. The first adaptation called Global Standard Approach (GSA) consists in building the context vectors from a comparable corpus composed of the specialized and the general comparable corpora. We implement the second adaptation called Selective Standard Approach (SSA) which gives the best results in their experiments. Elements of this adaptation are defined as follows: Let S be the vocabulary of the specialized corpus, G the vocabulary of the general corpus, w the word to represent and c a context word that appears in the window around w such that :

$$\forall w \in S \cap G, \forall c \in S \cap G,\ cooc(w,c) = cooc_S(w,c) + cooc_G(w,c) \quad (1)$$

Distance-Sensitive Co-occurrence

In the standard approach, we note that some context words in the window are not effectively related to the central word. Usually the further the latter is away from a context word, the less they are semantically related. This effect is more obvious especially after stop word filtering. A word originally far from the

central word can appear in the context window. This makes the context vector less relevant as a representation for the central word. To reduce this effect, we propose a weighted co-occurrence depending on the distance between the two words, denoted by Distance-Sensitive Co-occurrence (DSC):

$$DSC(w,c) = g(c|w) \times cooc(w,c) \quad \text{where} \quad g(c|w) = \Delta(w,c)^{-\lambda}, \;\; \lambda \in [0,1] \tag{2}$$

where w and c respectively denote the central word and the context word, $g(c|w)$ the weight that is distributed to c as the context of w, Δ is the distance between the two words and λ a hyper-parameter that determines the degree of penalization for distant word pairs. Note that λ = 0 is equivalent to a uniform distribution.

Weighted Mutual Information

Another limitation in the standard approach with MI is that MI overestimates low counts and underestimates high counts. In order to overcome this drawback, we propose the Weighted Mutual Information (WMI) inspired by the work of Pennington et al. (2014) where they introduce a function to smooth word co-occurrences. The original function is a weight function which prevents the overestimation of word co-occurrences. We also use it as a weight function for MI:

$$WMI(w,c) = f(cooc(w,c)) \times MI(w,c) \tag{3}$$

$$f(x) = \begin{cases} (x/x_{max})^{\alpha}, \alpha = 3/4, x_{max} = 20, \text{ if } x < x_{max} \\ 1 \text{ otherwise} \end{cases} \tag{4}$$

We have kept the same value for the hyper-parameter α. Concerning the x_{max}, since our corpora size is much smaller than the one in their work, we decide to make it correspondingly smaller (20). By adding the weight function, the output value for low co-occurrence counts is in fact reduced and the high co-occurrence counts are not impacted because their weight coefficient is always 1.

2.2 Neural Network-Based Approach

The neural network based approach uses neural network models to obtain word representation in low dimensional and dense vectors. These word vectors are also called word embeddings (Mikolov et al., 2013b). In the case of bilingual word embedding, Mikolov et al. (2013a) propose a method to learn a linear transformation from the source language to the target language for the task of lexicon extraction from bilingual corpora. Much research has been focused on this area since then. Faruqui and Dyer (2014) introduce canonical correlation analysis (CCA) to project the embeddings in both languages to a shared vector space. Xing et al. (2015) propose the orthogonal transformation and the vector length normalization during the learning phase. Artetxe et al. (2016) generalize these works and explain the equivalence of different objective functions under orthogonality and different normalization procedures. They show that combining these models effectively improves the results for both monolingual and bilingual tasks. Finally Smith et al. (2017) point out that the mapping should be orthogonal in order to be self-consistent.

Normalization, Mean Centering and Orthogonal Mapping

The method of Artetxe et al. (2016)[1] combines several related studies in this particular order:

1. Normalizing each word vector to unit length.
2. Dimension-wise mean centering for source and target matrices.
3. Learning the transformation matrix by mathematical analysis while constraining the orthogonality of the transformation matrix. In the original work of Mikolov et al. (2013a), the matrix is learned by gradient descent which is an on-line method.

Renormalization After Mean Centering

We observe in the implementation above that once the dimension-wise averages are subtracted, each word vector is no longer unit normalized. As a consequence, this could lead to a violation against the

[1] https://github.com/artetxem/vecmap

initial intuition that each word should have the same importance for learning the transformation matrix. Therefore we propose a second normalization after the mean centering. Although this would make the expected random product of any dimension not strictly zero, our experiment results show that it is still more beneficial for the bilingual task. Intuition could be that the mean centering was more in consideration of monolingual tasks, whereas in bilingual tasks, the length normalization of each word vector outweighs the mean centering.

Concatenation for Usage of External Data

Word embedding systems usually need a large amount of data in order to obtain reliable word vectors. However the size of domain-specialized comparable corpora is generally very modest (fewer than one million words). The word embedding models trained on our small size specialized corpora are not capable of generalizing meaningful features. To overcome this problem, the idea exploited in Hazem and Morin (2016) which seeks external general data to enrich the training phase can be useful, but it occasionally makes the specialized word representation biased by the general corpus. This is especially the case when the specialized corpus contains some infrequent or ambiguous words that have different meanings in different corpora. Considering these factors, we propose to use a concatenated vector of the one trained on the specialized corpora and on the general corpora as our new word vector. More specifically, we want to concatenate a relatively small size word vector from the specialized corpora and a relatively large size one from the general corpora. In our experiment, the word vector size for the word embedding model trained on the specialized corpus is set to be 100 and the size for the model trained on the general corpus is set to be 300. Hence we preserve the specialized corpus features albeit with less corresponding weight features in the transformation matrix. Consequently the new concatenated word vector carries self-contained information from both corpora (Figure 2). Another advantage is that by doing the concatenation, it is not necessary to retrain our word embedding models over large corpora because we can use existing pre-trained models available. This could save a great deal of time in practice while improving efficiency.

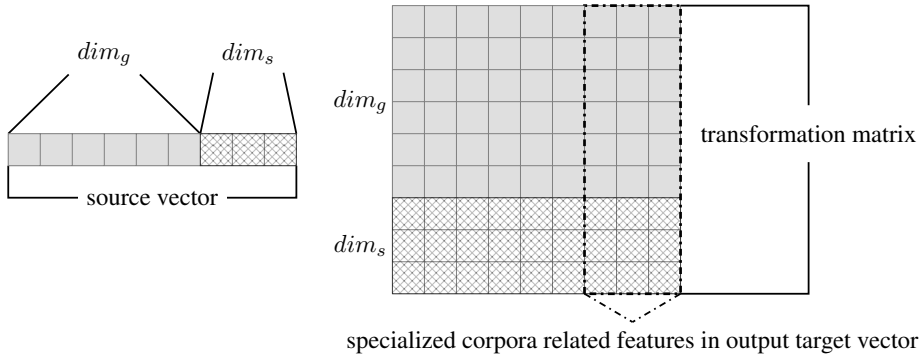


Figure 1: Let dim_g be the dimension size of the word vector from general corpora, and dim_s the size from specialized corpora. The transformation matrix will have a total number of $(dim_g + dim_s)^2$ weight features. Among these, $dim_g \times (dim_g + dim_s)$ will be source or target general corpora related features and only $dim_s \times (dim_g + dim_s)$ source or target specialized corpora related ones.

3 Multi-Word Term Alignment

Regarding MWTs, we first describe the compositional approach and an adapted version which combines the traditional compositional approach and the context-based projection approach. We also introduce a unified representation for MWTs which enables the mapping of variable length terms. Finally we propose a new approach in this family which combines the traditional compositional approach and the neural network approach. To the best of our knowledge, this is the first time that word embedding vectors are used in a compositional approach.

3.1 Compositional Approach

The compositional approach (CA) (Grefenstette, 1999; Tanaka, 2002; Robitaille et al., 2006) is a simple and direct approach that consists in translating each element of an MWT via a dictionary and generating all possible combinations and permutations. The ranking of the candidates is done by their frequency in the target corpora.

Compositional Approach with Context-Based Projection

The main limitation of the traditional compositional approach is the inability to translate a term when one of its composing words is not in the dictionary. To solve this problem, Morin and Daille (2012) propose the Compositional Approach with Context-Based Projection (CMCBP), where the objective is to combine the advantages of the standard and compositional approaches by substituting non-dictionary words with their context vectors obtained by the standard approach. The CMCBP begins by building the co-occurrence matrix as in the standard approach. Then it applies a direct translation reinforced by context alignment. If a word of a term to be translated is not present in the dictionary, it uses the context vector obtained by the standard approach and projects it into the target language, otherwise it takes the context vector of the target language directly. The next step is the generation of all combinations of possible translation representations for a source language term. Finally, the candidate terms are ranked according to their similarity with terms of the same length in the target language, and the final score for each possible translation is defined by the arithmetic or geometric mean of each similarity score.

MWT Representation

CMCBP does not, however, take the mapping of MWTs of variable lengths into account. For example, the English term "*wind vane*" can be translated as "*girouette*" in French and the English term "*wind energy*" by "*Windenergie*" in German. In order to take these cases into account, we propose to modify the representation of the MWTs, inspired by the works of Blacoe and Lapata (2012) in which the representation of a sentence is the sum of the distributional representations of each composing word:

$$vector(term) = \frac{1}{n}\sum_{i}^{n}\frac{vector(w_i)}{||vector(w_i)||}, \quad \text{where } n \text{ is the term length} \tag{5}$$

Notice that this is different from the mean vector introduced in the original work, here the mean vector is calculated from the normalized vectors because we want each component word to have the same impact rather than having the whole meaning influenced by the random vector length which could lead to some unpredictable bias. The MWT representation is then stored in a single vector, giving the ability to handle translations of different lengths while reducing the calculation time. In fact, in CMCBP, aligning an MWT requires calculating all possible permutations, so we must compare a factorial number of vectors for sorting candidates against a single vector with this new representation.

Compositional Approach with Word Embedding Projection

Following the idea of CMCBP, we propose a new method called Compositional Approach with Word Embedding Projection (CMWEP). It can be implemented by applying the following steps:

1. Prepare two word embedding models for the source and the target language with the same vector size.
2. Learn the transformation matrix using the approach we mentioned in 2.2.
3. Translate each word in an MWT via a bilingual seed lexicon. If a word is not in the dictionary, we return the embedding vector projected by the transformation matrix, and if the word is in the lexicon, we return the averaged vector of a list of embedding vectors for each possible translation. Giving an MWT "ABC" for instance, if "A" and "B" are in the lexicon and have their respective translations: {"a_1", "a_2", "a_3"}, {"b_1", "b_2"} and "C" is not in the lexicon, then for "A" and "B", we return the averaged vector of the word embeddings list of {"a_1", "a_2", "a_3"} and {"b_1", "b_2"} directly by the lookup table of the target language word embedding system, for "C" we return its projected embedding vector using the learned transformation matrix. In this way each word in the MWT in the source language is represented by a single vector in the target language space.

4. Generate the representation vector for the whole MWT by applying the method in the section above since each composing word has been represented by one vector.
5. Compare the translated vector to each candidate in the target language using a similarity measure such as Cosine. The candidate translations are ranked according to the scores of the similarity measure.

Here we show an overview which illustrates how the CMWEP turns a MWT into a vector. In addition, as it shares the same idea as in CMCBP, this diagram also represents the process in CMCBP.

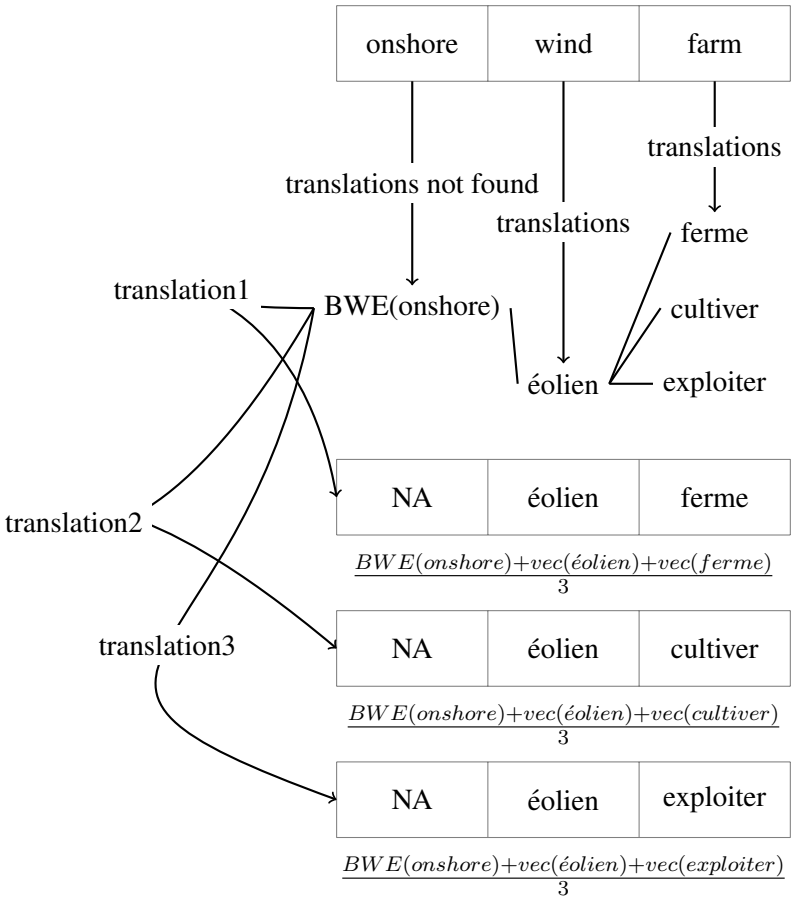


Figure 2: An example for translating the MWT *onshore wind farm* into French. The translations for each composing word are limited to those that exist in the comparable corpus and the bilingual dictionary. The final similarity between the MWT and a candidate is the maximum of the three similarities between each of the three possible representation vectors and the candidate vector.

4 Experiment Data and Resources

In this section, we outline the different textual resources used for our experiments: the comparable corpora, the reference lists, the bilingual seed lexicon and the pre-trained word embedding models.

4.1 Comparable Corpora

For our experiments, we used two specialized comparable corpora that have been used in previous works for bilingual terminology extraction in technical domains:

Breast Cancer Corpus (BC) is composed of documents collected from the Elsevier website[2]. The documents were taken from the medical domain within the subdomain of breast cancer.

[2] http://www.elsevier.com

Wind Energy Corpus (WE) has been released by the TTC project. The corpus has been crawled using the Babouk crawler (Groc, 2011) based on several keywords such as "*wind*", "*energy*", "*rotor*" in English and their translations in French.[3].

News Commentary (NC) consists of political and economic commentaries crawled from the web[4]. We use this corpus as our external data.

4.2 Gold Standard

Our reference lists for SWTs in BC and WE corpora are the same as used in Hazem and Morin (2016) which consist of 248 SWTs for BC and 139 for WE. The reference list for MWTs in WE is built based on the term list provided. Finally this list contains 73 MWT pairs but each pair has multiple variant translations and in our settings, we consider them to be also the gold translations[5]. If we deploy the 73 MWTs to a list where each pair contains only one translation, the list would have a size of 277 pairs. Because some MWTs have far more possible translations than others, we decide not to use the deployed version list to prevent the result from being biased by these MWTs. The reference list for the Italian/English task is the same as in Artetxe et al. (2016) which contains 1,500 entries. We would like to mention that the candidate list for one MWT includes all the words in the vocabulary plus all the MWTs extracted by a symbolic terminology extraction system, which generally extracts three times as many MWTs as words in the vocabulary. The terms are extracted following some pre-defined syntactic patterns, for example the pattern *NOUN NOUN* could lead to the extraction of *shop assistant*. So the one-to-many or many-to-one mapping is theoretically findable. Moreover, our reference list for the MWT task contains only out-of-dictionary MWTs. Table 1 presents the principal characteristics of the data.

Corpus	**# distinct words**		**# content words**		**Reference List**
	FR	EN	FR	EN	
BC	521,262	525,934	6,630	8,821	SWT: 248
WE	314,549	313,943	6,038	7,134	MWT: 73, SWT: 139
NC	5.7 M	4.7 M	23,597	29,489	

Table 1: Characteristics of comparable corpora in our experiments

4.3 Bilingual Lexicon

For our French/English experiments, we use the French/English dictionary ELRA-M00337[6] (243,539 entries). From this dictionary we select a subset of 3,007 entries from the BC corpora and a subset of 2,745 entries from the WE corpora based on a word frequency threshold of 5. These two subsets are used as the training data in our word embedding mapping experiments. For our Italian/English experiments, we only use the same seed lexicon[7] as used in Artetxe et al. (2016) where 5,000 entries are manually selected.

4.4 Pre-trained Word Embedding Models

To be fully comparable, in the Italian/English experiments we use the same model as in Artetxe et al. (2016). This could be retrieved by following the instructions they provide. The English embeddings were trained on a 2.8 billion word general corpus (UKWAC + WIKIPEDIA + BNC)[8], while a 1.6 billion word general corpus ITWAC was employed to train the Italian embeddings. As for the French/English embeddings, the vectors in dimension 300 were obtained using the skip-gram model described in Bojanowski et al. (2016) with default parameters[9].

[3]Corpus available here: `http://www.lina.univ-nantes.fr/?Reference-Term-Lists-of-TTC.html`
[4]`http://opus.lingfil.uu.se`
[5]The reference list and evaluation software are available here: `https://github.com/Dictanova/term-eval`
[6]`http://catalog.elra.info/product\_info.php?products\_id=666`
[7]`https://github.com/artetxem/vecmap`
[8]`http://clic.cimec.unitn.it/georgiana.dinu/down/`
[9]`http://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md`

5 Experiment Settings and Results

We conducted two sets of experiments. The first one aims at validating our enhancements to the state-of-the-art approaches by comparing the results of the bilingual SWT extraction in general and specialized domains. The second experiment aims at studying the application of the proposed approaches of the bilingual MWT extraction focusing on a specialized domain which is our main interest. This kind of task often lacks specialized data. As pointed out in Mikolov et al. (2013a), applications to low resource domains is a very interesting topic with much to be explored.

5.1 Standard Approach

In the standard approach, the window size is 3 (a total of 7 words are considered), which is the same as in Hazem and Morin (2016). The distance weight parameter λ in distance-sensitive co-occurrence is empirically set to be 0.25.

5.2 Word Embedding on Specialized Corpora

We use the implementation of *word2vec* from *deeplearning4j*[10] to train the word embedding model on our specialized corpora. Considering the relative low frequency of some words in specialized corpora, we apply the skip-gram algorithm which is supposed to work better with infrequent words[11]. We set the negative sample to 20, the window size to 5 and the training epoch to 20.

5.3 Results on SWT Task

Table 2 shows the results of the bilingual SWT extraction using the standard approach (SA) and its variant, the Selective Standard Approach (SSA) for the breast cancer corpus (BC) and the wind energy corpus (WE). It also includes the results of our two enhancements: Weighted Mutual Information (WMI) and Distance-Sensitive Co-occurrence (DSC).

Standard Approach	**BC**	**WE**
Hazem and Morin (2016)†	25.9	15.6
SA + WMI	28.9	21.4
SA + DSC	27.4	15.8
SA + WMI + DSC	**29.5**	**21.8**

(a) Result (MAP%) of standard approaches for the bilingual SWT extraction

Selective Standard Approach	**BC**	**WE**
Hazem and Morin (2016)†	56.5	44.4
SSA + WMI	55.0	33.9
SSA + DSC	**57.3**	**45.3**
SSA + WMI + DSC	55.8	35.7

(b) Result (MAP%) of selective standard approaches with NC corpus as the external data for the bilingual SWT extraction

Table 2: Context-Based Projection approaches results for bilingual SWT extraction. † indicates results obtained by our implementation of the approach.

We observe in Table 2a that WMI alone improves the results compared to those obtained by Hazem and Morin (2016) with MI but also compared to those when using the *Discounted Odds Ratio* (Evert, 2005) as the normalization method, where the MAP is 0.270 for BC and 19.4 for WE. This shows the interest of penalizing small occurrences to compensate the overestimation of the original MI. The second observation is that DSC alone also improves the results. This confirms our intuition that the further a context word is from its central word, the less relevant it is. Finally, we see that combining both enhancements gives the best result. So we could consider that the two enhancements are not mutually exclusive. Another observation is that the improvement of DSC for WE is indeed poorer than for BC, but the tendency is always improving. So we think our hypothesis above holds true for the WE data. Again we can see the same tendency when combining WMI. Note that, despite the difference in improvement between the two datasets, the enhancement that our approach offers is still relevant.

The results in Table 2b shows that WMI is less efficient when the data is enriched because the overestimation of small occurrences is smoothed by the addition of the enlarged overall data as discussed in

[10] http://deeplearning4j.org/

[11] https://code.google.com/archive/p/word2vec/

Hazem and Morin (2016). Moreover, since some term elements to be translated are quite infrequent or even non-existent in the general corpus, it is possible that penalizing all the small occurrences reduces discriminative features in the general corpus. Besides, although the results of SSA are different the two corpora share again the same tendency: they both reach their best when using DSC without WMI. However DSC always improves the results whether it is applied alone or combined with WMI. Again, it shows that the two enhancements are not mutually exclusive with external data.

Bilingual Word Embedding	**Accuracy**
Mikolov et al. (2013a)‡	34.93
Artetxe et al. (2016)‡	39.27
Our method (Renorm.)	**39.60**

(a) Accuracy % (p@1) of bilingual word embedding approaches for the Italian/English general word mapping task. The word embedding vectors are trained on wiki corpora mentioned in 4.4.

Bilingual Word Embedding	**BC**	**WE**
Hazem and Morin (2017)‡	82.3	-
L^2 + MC + Orth.	27.4	21.8
Concat + L^2 + MC + Orth.	82.4	**83.1**
Concat + L^2 + MC + Orth. + Renorm.	**83.2**	**83.1**

(b) Result (MAP%) of bilingual word embedding approaches for bilingual SWT extraction

Table 3: Bilingual Word Embedding approaches results for bilingual SWT extraction. ‡ indicates results reported by the authors.

Table 3 shows the results of the bilingual word embedding approaches on SWTs of the specialized BC and WE corpora along with the general WIKIPEDIA corpus. Our final proposed method is in the last line in each table. We can see that without external data, the results are not better than the traditional approach.

In Table 3a the three factors (denoted by L^2*, MC, Orth.*) resumed in Artetxe et al. (2016) improve the original results of Mikolov et al. (2013a). By adding an additional operation between the mean centering and the orthogonal mapping, our method brings a small but consistent improvement that can also be found in the special domain task in Table 3b. Other than that, the external data information stored in the pre-trained word embeddings significantly improves the results when carefully concatenated to those trained on special corpora.

5.4 Results on MWT Task

Model	**Accuracy**	**MAP**
CA	59.0	61.5
CMCBP	49.3	61.4
CMCBP + WMI + DSC	46.6	63.2
CMCBP + SSA	50.7	66.0
CMCBP + SSA + WMI + DSC	53.4	66.3
CMWEP + L^2 + MC + Orth.	52.1	67.8
CMWEP + L^2 + MC + Orth. + Concat.	**61.6**	73.3
CMWEP + L^2 + MC + Orth. + Concat. + Renorm.	**61.6**	**73.4**

Table 4: Accuracy (p@1) and MAP % of different approaches for MWTs in the WE corpus

Table 4 shows results with different approaches on MWTs in the special domain WE corpus. The traditional compositional approach (CA) has better accuracy than CMCBP because the candidates are ranked by their frequency in the target language corpus while the candidates in CMCBP are ranked by the similarity measure, so theoretically those translated by CA are also translated by CMCBP but with the same similarity score (1.0). Therefore, the final rank in CMCBP is to some degree randomized as multiple top candidates have the same score. Nevertheless, when using external data (+SSA), the MAP is considerably improved by CMCBP, this shows that the CMCBP can effectively find many out-of-dictionary translations that can not be found by CA. The same problem also exists in CMWEP.

Another remarkable point is that unlike the SWT bilingual extraction with SSA, combining WMI and DSC improves the results in MAP with or without external data.

Our word embedding methods (CMWEP) notably improve the results when using information carried by external data (+ *concat*) by almost 12 points in MAP and 2.6 points in accuracy. Combining the usage of external data and the renormalization gives the best result. The characteristics of the WE corpus determine the fact that for most MWTs, each component word are included in the dictionary. So CA already provides a high result especially for the accuracy, if we took a less academic or less cleaned corpus for example the Amazon Reviews where there are more out-of-vocabulary words because of spelling mistakes and web languages, we expect the result of CA to be much lower while the result of other approaches would be less impacted. This is also why the renormalization improves the result very slightly, since most of the translations are projected by a CA-like process.

5.5 Error analysis

The error analysis is made by manual observation: we first extract the 28 MWTs for which no translation has been found in the in the top@1 of our best performing model (the last line in Table 4), then we study these MWTs one by one. Among the 28 MWTs, we observe three categories:

1. **Weak compositionality**. The translation of the whole is not a combination of all the translations for each element (Melamed, 2001). 10 MWTs (35.7%) belong to this type. For example "*pitch angle*" is translated as "*angle d'inclinaison* (lit. *angle of tilt*)" by the system while the gold one should be "*angle de calage* (lit. *angle of sitting*)" where the word "*calage*" is not the translation of any word in the bilingual dictionary. Similarly, the gold translation for "*rotor shaft*" is a single word "*arbre*" which is basically the translation of one composing word "*shaft*". This problem could be even worse between two further languages, Baldwin and Tanaka (2004) report that at least 50% of the Japanese *NN* compounds are not translated through a compositional strategy into English.
2. **Ambiguous translations**. Multiple possible translations have the same similarity at the top@n list because each composing word has multiple translations in the dictionary and some combinations exist in the terminology candidate list. 12 MWTs (42.9%) belong to this type. For example, "*low frequency*" is translated by "*rotation inférieure* (lit. *inferior rotation*)" while it should be "*basse fréquence* (lit. *low frequency*)". Both "*inférieure*" and "*basse*" are translation of "*low*", and "*rotation*" and "*fréquence*" are translation of "*frequency*". This problem also relates to the monolingual word disambiguation.
3. **Different orders with same words**. Since our method ignores the word order, two sequences of the same set of words have the same representation vector. 6 MWTs (21.4%) belong to this type. For example, "*power installation*" is translated as "*puissance d'installation* (lit. *power of installation*)" while it should be "*installation de puissance* (lit. *installation of power*)". We could have taken the word order into consideration but it would increase a huge amount of calculation time. Moreover, the alignment of variable length terms would not be manageable.

6 Conclusion and future work

This work proposes a unified framework for bilingual terminology extraction of multi-word terms, improving on several previous works. Our experiments demonstrate the effectiveness of our enhancements of previous works. Generally speaking, the bilingual word embedding method with external data information plus the three factors resumed in Artetxe et al. (2016) and the renormalization outperforms previous works and brings us the best result. Moreover we observe that without external information, the traditional approach with our proposed enhancements still has very competitive results. For future work, we would like to restructure the representation of MWTs which is now based on a naive hypothesis that all the notional words contribute equally to the whole meaning of an MWT. In addition, as we mentioned in section 5.5, the performance for two further languages would be interesting to see and the disambiguation for words with multiple possible translations could be very useful.

Acknowledgments

The authors would like to thank Joseph Lark for feedback on earlier version of this paper, and the anonymous reviewers for their valuable comments. Jingshu Liu is partially supported by the CIFRE grant N 2016/0659.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*, pages 2289–2294, Austin, Texas.

Timothy Baldwin and Takaaki Tanaka. 2004. Translation by machine of complex nominals: Getting it right. In Takaaki Tanaka, Aline Villavicencio, Francis Bond, and Anna Korhonen, editors, *Second ACL Workshop on Multiword Expressions: Integrating Processing*, pages 24–31, Barcelona, Spain.

Yoshua Bengio, R'ejean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, pages 1137—1155.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP'12)*, pages 546–556, Jeju Island, Korea, July. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Dhouha Bouamor, Nasredine Semmar, and Pierre Zweigenbaum. 2013. Context vector disambiguation for bilingual lexicon extraction from comparable corpora. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'13)*, pages 759–764, Sofia, Bulgaria.

Yun-Chuang Chiao and Pierre Zweigenbaum. 2002. Looking for candidate translational equivalents in specialized, comparable corpora. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, pages 1–5, Stroudsburg, PA, USA.

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

Estelle Delpech, Béatrice Daille, Emmanuel Morin, and Claire Lemaire. 2012. Extraction of domain-specific bilingual lexicon from comparable corpora: Compositional translation and ranking. In *Proceedings of the 24rd International Conference on Computational Linguistics (COLING'12)*, pages 745–762, Mumbai, India.

Stefan Evert. 2005. *The statistics of word cooccurrences : word pairs and collocations*. Ph.D. thesis, University of Stuttgart.

Robert M. Fano. 1961. *Transmission of Information: A Statistical Theory of Communications*. MIT Press, Cambridge, MA, USA.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL'14)*, pages 462–471, Gothenburg, Sweden.

Pascale Fung. 1995. Compiling bilingual lexicon entries from a non-parallel english-chinese corpus. In *Proceedings of the 3rd Annual Workshop on Very Large Corpora (VLC'95)*, pages 173–183, Cambridge, MA, USA.

Gregory Grefenstette. 1999. The world wide web as a resource for example-based machine translation tasks. In *Proceedings of the ASLIB Conference on Translating and the Computer 21*, London, UK.

Clément De Groc. 2011. Babouk: Focused web crawling for corpus compilation and automatic terminology extraction. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 497–498.

Zellig Sabbettai Harris. 1968. *Mathematical structures of language*. Interscience Publishers.

Amir Hazem and Emmanuel Morin. 2016. Efficient data selection for bilingual terminology extraction from comparable corpora. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING'16)*, pages 3401–3411, Osaka, Japan.

Amir Hazem and Emmanuel Morin. 2017. Bilingual word embeddings for bilingual terminology extraction from specialized comparable corpora. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (IJCNLP'17)*, pages 685–693, Taipei, Taiwan.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL'14)*, pages 58–68, Baltimore, Maryland.

Laurent Jakubina and Phillippe Langlais. 2017. Reranking translation candidates produced by several bilingual word similarity sources. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL'17)*, pages 605–611, Valencia, Spain.

Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning (ACL'14)*, pages 171–180, Ann Arbor, Michigan.

I. Dan Melamed. 2001. *Empirical Methods for Exploiting Parallel Texts*. MIT Press, Cambridge, MA, USA.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIP'13)*, pages 3111–3119, Lake Tahoe, Nevada, USA.

Andriy Mnih and Geoffrey Hinton. 2008. A scalable hierarchical distributed language model. In *Proceedings of the 21st International Conference on Neural Information Processing Systems (NIPS'08)*, pages 1081–1088, Vancouver, British Columbia, Canada.

Emmanuel Morin and Béatrice Daille. 2012. Revising the compositional method for terminology acquisition from comparable corpora. In *Proceedings of the 24rd International Conference on Computational Linguistics (COLING'12)*, pages 1797–1810, Mumbai, India.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*, pages 1532–1543, Doha, Qatar.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated english and german corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, pages 519–526, College Park, Maryland, USA.

Xavier Robitaille, Yasuhiro Sasaki, Masatsugu Tonoike, Satoshi Sato, and Takehito Utsuro. 2006. Compiling French-Japanese terminologies from the web. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'06)*, pages 225–232, Trento, Italy.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *5th International Conference on Learning Representations (ICLR'17)*.

Takaaki Tanaka. 2002. Measuring the similarity between compound nouns in different languages using non-parallel corpora. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, pages 1–7, Stroudsburg, PA, USA.

Shiva Taslimipoor, Ruslan Mitkov, Gloria Corpas Pastor, and Afsaneh Fazly. 2016. Bilingual contexts from comparable corpora to mine for translations of collocations. In *Proceedings of the 17th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing'16)*, Konya, Turkey.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'15)*, pages 1006–1011, Denver, Colorado.

Neural Activation Semantic Models: Computational lexical semantic models of localized neural activations

Nikos Athanasiou
School of ECE NTUA
Athens, Greece
el12074@central.ntua.gr

Elias Iosif *
IFF, University of Nicosia
Nicosia, Cyprus
iosif.e@unic.ac.cy

Alexandros Potamianos
School of ECE NTUA
Athens, Greece
apotam@gmail.com

Abstract

Neural activation models that have been proposed in the literature use a set of example words for which fMRI measurements are available in order to find a mapping between word semantics and localized neural activations. Successful mappings let us expand to the full lexicon of concrete nouns using the assumption that similarity of meaning implies similar neural activation patterns. In this paper, we propose a computational model that estimates semantic similarity in the neural activation space and investigates the relative performance of this model for various natural language processing tasks. Despite the simplicity of the proposed model and the very small number of example words used to bootstrap it, the neural activation semantic model performs surprisingly well compared to state-of-the-art word embeddings. Specifically, the neural activation semantic model performs better than the state-of-the-art for the task of semantic similarity estimation between very similar or very dissimilar words, while performing well on other tasks such as entailment and word categorization. These are strong indications that neural activation semantic models can not only shed some light into human cognition but also contribute to computation models for certain tasks.

1 Introduction

The human ability of translating concepts into words and back depends on the ability of mind to decode and encode meaning. This mental process, which is not currently completely understood, has captivated the interest of both neuroscientists and computational linguists (Haxby et al., 2001; Ishai et al., 1999; Cree and McRae, 2003; G. Kanwisher et al., 1997; Just et al., 2010). Specifically, when a person experiences a visual stimulus of a concept, reads, speaks or writes a word, particular neuronal regions in the brain are activated (Pulvermller, 2001).

Various studies have been carried out to explore brain encoding and decoding mechanisms when a stimulus is present, as detailed next. For visual stimuli, studies have shown that is feasible to discriminate and reconstruct images using patterns of neural activity, mainly found in the visual cortex (Kay et al., 2008; Thirion et al., 2006; Nishimoto et al., 2011; Naselaris et al., 2009; Miyawaki et al., 2008), the part of brain responsible for visual information processing. Other works have demonstrated the relationship between cognitive perception and speech (Benson et al., 2001; Formisano et al., 2008). Regarding textual stimuli, researchers have shown distributed semantic maps of words are present in our brains (Huth et al., 2012; Sudre et al., 2012).

Lexical semantics are based on the assumption that similar words appear in similar contexts (Harris, 1954). Based on that assumption, two different approaches for building semantic models have been proposed. The first approach is to encode word semantics, by applying dimensionality reduction of context-word occurrence matrix which was computed using large corpora (Deerwester et al., 1990; Bullinaria and Levy, 2012). The second approach replaces thse "counting" by predictive models (Baroni et al., 2014) based on neural networks (Bengio et al., 2000; Mnih and Hinton, 2007; Mikolov et al., 2013;

*The research was performed when the author was a postdoctoral researcher at School of ECE, NTUA in Athens, Greece.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2867–2878
Santa Fe, New Mexico, USA, August 20-26, 2018.

Turian et al., 2010; Collobert and Weston, 2008). Counting models calculate and weight context vectors, while predictive models learn word vectors by guessing the context in which these words tend to appear.

In pursuance of enriching such lexical semantic models with cognitive information, as well as discovering the cognitive representation of word semantics, a number of studies have attempted to examine the mapping between semantic representation of computational and cognitive models. In prior work, it has been shown that semantic of words are related to activation potentials in regions of the brain and that decoding between neural activations and semantic content (Mitchell et al., 2008; Palatucci et al., 2009; Sudre et al., 2012; Murphy et al., 2012; Pulvermller, 2001) is possible. Furthermore, neural activations are shown to have predictive power with respect to semantics at the word (Mitchell et al., 2008; Jelodar et al., 2010) and sentence (Jing et al., 2017; Anderson et al., 2017) level. Computational studies that aim to explore the influence of neural activations in word representations have shown that by incorporating neural activations when training lexical semantic models can improve their generalization ability despite the small amount of neural activation data used (Fyshe et al., 2014; Ruan et al., 2016). These works show that a strong relationship exists between computational semantic models and neural representations. However, it remains to be seen how cognitive semantic representations, including localized neural activation patterns can help improve the performance of computational semantic models, especially for complicated classification and recognition tasks.

Motivated by the aforementioned studies that show correlation between localized neural activations and word semantics, we propose a computational model for semantic similarity that utilizes predicted neural activations learned from a small set of concrete nouns. The proposed model is applied to a variety of natural language processing tasks. The neural activation prediction model used here for lexical expansion is that proposed in (Mitchell et al., 2008). In our list of experiments, we first compare the performance of the proposed neural activation model for a concrete noun semantic similarity task and show that for certain word pairs it outperforms the state-of-the-art. Then we evaluate the performance of neural activation vectors for a word classification, sensory modality (sense) classification and textual entailment task. The fusion of neural and traditional word embedding vectors are shown to outperform the state-of-the-art. To our knowledge, this is the first time brain imaging data are successfully used for the aforementioned tasks.

2 Related Work

A significant body of literature investigates neural activations by mapping word semantics to fMRI data. Most of them have in common a basic idea published in (Mitchell et al., 2008). In this work, a model is introduced that maps low dimensional word coocurence vectors to neural activations. The approach is validated in a neural activation-based word classification task. This work shows that the mapping between lexical semantic spaces constructed via computational lexical semantic algorithms and 3D neural activations representations measured via fMRIs is possible.

A first variant of the aforementioned model was introduced in (Jelodar et al., 2010), where the use of WordNet features was investigated for constructing the lexical semantic space. Word classification results reported showed similar performance to (Mitchell et al., 2008), however, by fusion of the two lexical semantic models improved classification results where achieved.

A second approach, introduced in (Levy and Bullinaria, 2012), extends the work in (2008) by increasing the number of fMRI voxels used in the neural activation vectors and the number of featurs (dimension) of the lexical semantic model showing additional performance improvement.

Algorithms that count word coocurences and utilize hand-crafted features for constructing lexical semantic models can be found in (Bullinaria and Levy, 2012; Baroni and Lenci, 2010).Moreover, various lexical semantic models that predict a word based on its context have also elaborated (Bengio et al., 2000; Mikolov et al., 2013; Collobert and Weston, 2008). Word prediction models tend to perform better in natural language processing tasks such as analogy, similarity, synonym detection, concept taxonomy (Baroni et al., 2014) and sentiment analysis (Socher et al., 2011; Socher et al., 2013). However, their relationship with cognitive lexical representation is not yet well understood, at least to a degree that would allow us to improve current computation lexical semantic models. Along these lines, there have

been two main lines of work. In (Fyshe et al., 2014), neural activations were integrated in the training procedure of lexical semantic models in order to learn word embeddings that include latent neural information. Although a small number of words was used to bootstrap the neural activation representations, it have been shown that their model can predict unseen words and generalizes well across different topics. Ruan et al. (2016), have shown that neural activations for different parts of the brain are correlated with word embeddings especially skip-grams. A semantic model was also proposed for training word embeddings as a first step towards including cognitive information in a word vector representation.

3 Approach

3.1 Neural activations prediction

The neural activation prediction model used here is that proposed in Mitchell (2008)[1] First activation potentials are measured from fMRI images. We consider voxels to be 3D pixels created by MRI scanning software depicting brain state. Every voxel v is associated with a $TN \times V'$ array, M, of neural activation values (blood flow), where V' is the number of voxels, T is the number of trials, N is the number of stimuli, in our case different nouns. The first step is to select the most stable (salient) voxels, V to include in the neural activation model. The stability score s_v for voxel v is computed as the average pairwise Pearson correlation ϱ for all the different row combinations of M, as follows:

$$s_v = \frac{2}{TN(TN-1)} \sum_{i=1}^{TN} \sum_{j=i+1}^{TN} \varrho(M_{i,:}, M_{j,:}), \quad \forall\, v = 1 \cdots V' \tag{1}$$

where $M_{i,:}$ is the ith row of matrix M. High stability scores, s_v, as described in Eq. 1, indicate that corresponding voxels have consistent representations across different trials and nouns.

Next, the neural activation predictive model proposed in (Mitchell et al., 2008) is defined.

For this purpose we identify a set of m seed words s_1, s_2, ..., s_m, and a function $f_i(w)$ that estimates the association between seed word s_i and word w. The core assumption of the model is that words that are closely associated have similar neural activation patterns, thus the mapping from the associative (semantic) space to the neural activation (voxel) space is estimated as follows:

$$y_v(w) = \sum_{i=1}^{m} c_{v,i}\, f_i(w), \quad \forall\, v = 1 \cdots V, \tag{2}$$

where $y_v(w)$ is the activation of voxel v for word w, $f_i(w)$ is a scalar value that reflects the association between the i^{th} seed word s_i and the word w, m is the number of seed words (semantic features), V is the total number of voxels, and $c_{v,i}$ is a learned weight ranging between 0 and 1. A set of 25 verbs (seed words) was identified in (Mitchell et al., 2008) as semantic features s_i; seed words were manually selected according to psycholinguistic criteria. The similarity function $f_i(w)$ was set to the (normalized) co-occurrence frequency of the i^{th} seed word and w, estimated on a corpus. The weights $c_{v,i}$ were estimated using the fMRI data on words w using linear or ridge regression estimation. Once $c_{v,i}$ have been estimated, Eq. 2 can be used to predict the neural activation of unseen words[2].

3.2 From neural activations to similarity

Based on the hypothesis that similar words have similar neural activations, we propose a model to estimate word similarities based on neural activations predicted using Eq. 2. We evaluated various metrics for computing semantic similarity from neural activations. We present only a top performing metric

[1]The features in (2008) are attractive because of their simplicity and low-dimensionality, and generalize well for lexical expansion to a large lexicon compared to other works described in Section 2 that potentially perform slightly better.

[2]Although Eq. 3 can be used to perform lexical expansion an any token w for which $f_i(w)$ can be computed, the proposed framework (choice of $f()$ and associated fMRI data) is meant for concrete words and typically only neural activations for concrete words are reported in the literature.

formulated as the weighted square distance, namely:

$$S(w_1, w_2) = \sum_{v=1}^{V} b_v \ (y_v(w_1) - y_v(w_2))^2, \tag{3}$$

where $S(w_1, w_2)$ is the semantic similarity between words w_1 and w_2, V represents the number of voxels used in the predicted neural image, $y_v(w)$ is the activation of a voxel for word w, and b_v is a learned weight of the contribution of a particular voxel to the similarity metric. In Figure 1 the predictions

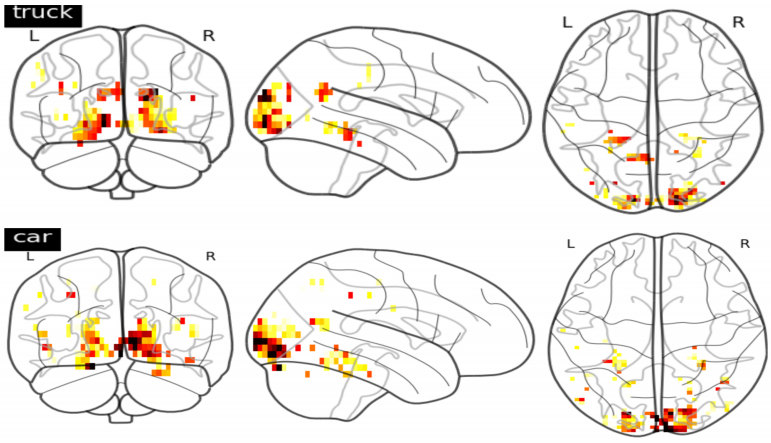


Figure 1: Neural activation images for two similar concrete nouns including the 500 most stable voxels (participant P1). This figure shows just one horizontal slice in Montreal Neurological Institute (MNI) space of the three-dimensional image.

of neural activations for two highly similar nouns in fMRI dataset are presented. Visualizations were created from 500 voxels to gather insight for our computational model. Observe that both brain images have similar neural activations both in terms of which parts of the brain are activated and the activation values. Although, we don't utilize localization information, our weighting schema implicitly detects activation patterns by variations in b_i coefficients.

4 Experimental Data and Preprocessing

We built the neural activations prediction model as in (Mitchell et al., 2008) using fMRI data for 60 concrete nouns. We calculate the semantic features $f_i(w)$ for each concrete noun, w, by counting its co-occurences with 25 manually selected verbs in a large corpus created in (Iosif et al., 2016) by aggregating results of web queries to Yahoo. The fMRI data used in our experiments were collected and processed by (Mitchell et al., 2008) and are publicly available. In this dataset, each one of the participants was presented 60 concrete nouns (for 6 times each) through a line drawing which was labeled with the corresponding noun. Each out of nine subjects, was asked to think about properties of the presented noun during scanning procedure. Finally, a vector representation of the whole cortex neural activation was extracted. Further details about the dataset can be found in (Mitchell et al., 2008) and its supplementary website[3]. Prior to training both training and test data are averaged across trials and the final neural activations of each noun are mean normalized. The following datasets have been used for semantic similarity, taxonomy creation, sense classification and entailment tasks. The code of the present work is publicly available[4].

MEN: For the semantic similarity task we train and evaluate our model on the MEN dataset (Bruni et al., 2014) which consists of 3000 word pairs (2000 for training set and 1000 pairs for test set). Each word pair is associated with a similarity score. This score was computed by averaging the similarities

[3] http://www.cs.cmu.edu/~tom/science2008/
[4] https://github.com/athn-nik/neural_asm

that provided by human annotators. We hand-labeled the dataset to keep only concrete nouns because the neural activation prediction model is trained only on concrete nouns. This resulted in 1114 pairs (562 unique words) in the training set and 524 pairs (438 unique words) in the test set. The similarity scores were normalized between zero and one. We also created 2 subsets of MEN of 39 highly similar and 79 highly dissimilar word pairs using a thresholding technique, where pairs with similarity score over 0.85 and under 0.1 belong in the first and second subset respectively.

ESSLLI: For the taxonomy creation task, we evaluate our model on the ESSLLI dataset (Baroni et al., 2008). It consists of a three-level hierarchy (2-3-6 classes). The lowest level of hierarchy contains 6 classes of concrete nouns (birds, land animals, fruit, greens, vehicles, tools), the middle 3 classes (vegetables, artifacts, animals) while the upper class is distinguished between living beings and objects.

Sensicon: For sense classification, we use the Sensicon (Tekiroglu et al., 2014) dataset. Sensicon is a dictionary which contains 22684 English words and associates each word with 5 numerical scores and a part of speech annotation. The scores correspond to the relevance of the word to each of the 5 senses, namely vision, hearing, taste, smell and touch. In order to use these scores for the sense classification task we selected nouns who have non-zero scores in only one sense results in 1011 words.

SNLI dataset: For the entailment task, we used the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015) which contains around 570K sentence pairs with three labels: entailment, contradiction and neutral. We preprocessed the initial dataset to keep only training and testing examples that have at least two or three concrete nouns that are also in the MEN dataset for both premise and hypothesis. This resulted in 30,498 training and 592 test samples for the case of at least three common words and 171,528 training and 3201 test samples for the case of at least two common words with MEN.

5 Experiments on Semantic Similarity

In this section we present baseline results on the neural activation prediction model of (Mitchell et al., 2008) and investigate the performance of the proposed neural activation semantic model of Eq. 3 for a semantic similarity task on the MEN dataset. Our goal here is first to reproduce the results in (Mitchell et al., 2008) and then to investigate the properties of neural activation embeddings semantic models compared to traditional embedding models, e.g., word2vec in (Mikolov et al., 2013).

5.1 Neural activations prediction

As proposed in (Levy and Bullinaria, 2012), we choose the 500 most stable voxels from the fMRI images. Then, $c_{v,i}$ is estimated as in Eq. 2 by applying linear regression per voxel across different words with regularization. To evaluate the neural activation prediction model we used cosine similarity in order to evaluate if our prediction for the possible pair of test words is correct or not. Correct prediction means that sum of the cosine similarities of the correct matched pairs is greater than the false matched pair as shown next:

$$cos(\vec{i_1}, \vec{p_1}) + cos(\vec{i_2}, \vec{p_2}) > cos(\vec{i_2}, \vec{p_1}) + cos(\vec{i_1}, \vec{p_2}), \tag{4}$$

where $\vec{i}$ is the actual image and $\vec{p}$ is the predicted image of 500 voxels. The dataset, which consists of 60 nouns, is split in train set (the rest 58 nouns) and test set (2 nouns) for all possible 1770 combinations using cross-validation.That process is followed for every one of 9 participants. The results are shown reported in Table 1 when using linear and ridge regression to estimate $c_{v,i}$ in Eq. 2. Results in Table 1 are on average consistent with the baseline results reported in (Mitchell et al., 2008). We achieved higher performance for some participants and lower for others. This can be attributed to the different tools we used to implement the system (as we used scikit-learn (Pedregosa et al., 2011) and (Mitchell et al., 2008) used Matlab). Moreover, we experimented with the number of voxels and our results agree with the findings reported in (Levy and Bullinaria, 2012).

Participant ID	Linear Regression	Ridge Regression	Mitchell et. al
1	0.79	0.84	0.83
2	0.75	0.82	0.76
3	0.63	0.76	0.78
4	0.63	0.79	0.72
5	0.61	0.78	0.78
6	0.58	0.65	0.85
7	0.58	0.75	0.73
8	0.65	0.68	0.68
9	0.57	0.68	0.82
Mean	0.64	0.75	0.77

Table 1: Baseline Model Results

5.2 Similarity task

For the semantic similarity task, we applied Eq. 3 for the word pairs of the MEN dataset. The $y_v(.)$s of Eq. 3 were computed using Eq. 2 utilizing up to 250 voxels. We exploited the training subset of MEN for learning the b weights of Eq. 3 using linear regression. Those weights were used for computing the similarities for the test subset of MEN. The Spearman correlation coefficient between the human similarity scores (ground truth) and the similarity scores computed by Eq. 3 was used as evaluation metric. The results are presented in Table 2, where we compare the performance of the proposed neural model (averaged across participants) against the performance yielded by the w2vec word embeddings (Mikolov et al., 2013) trained on the GoogleNews corpus.

Subset	**Number of voxels**	**Neural model**	**w2vec**
All Concrete nouns	50	0.43	0.76
	100	0.47	0.76
	150	0.48	0.76
	200	0.48	**0.76**
Most & Least similar	50	0.58	0.73
	100	0.82	0.73
	150	0.82	0.73
	200	**0.88**	0.73
Least similar	50	0.43	0.43
	100	0.44	0.43
	150	0.47	0.43
	200	**0.63**	0.43
Most similar	50	0.28	0.14
	100	0.63	0.14
	150	0.68	0.14
	200	**0.83**	0.14

Table 2: Evaluation results on the concrete nouns subset of the MEN test set, and on most and least similar concrete word subsets.

Overall, the w2vec model outperforms the neural model achieving 0.76 correlation on all concrete nouns. For the neural model, performance increases as more voxels are exploited reaching 0.48 correlation is obtained for at least 150 voxels. In Table 2, the performance is also shown for three subsets of the MEN test set, namely, "Most & Least similar", "Least similar" and "Most similar" concrete nouns.

For all three subsets, the performance achieved by the neural model exceeds[5] the performance of w2vec when at least 100 voxels are used. The performance improvement becomes more pronounced as the number of voxels increases. The best correlation score achieved is 0.88 for the case of the "Most & Least similar" subset for 200 voxels, outperforming the w2vec model (0.73 correlation). Especially for the case of the "Most similar" evaluation subset, we observe a remarkable difference between the two models, i.e., 0.83 vs. 0.14 for the neural and the w2vec model, respectively.

6 Classification and Entailment Experiments

Next we present the performance of the neural activation semantic model for a taxonomy creation (semantic class classification), sensory modality (sense) classification, and lexical entailment task. Neural vectors are averaged across participants and evaluated both standalone and in combination (early or late fusion) with traditional word embedding models.

6.1 Taxonomy Creation

The performance of similarities computed by Eq. 3 were also evaluated on a taxonomy creation task on the ESSLLI dataset (Baroni et al., 2008). Taxonomy creation is performed using the neural activation vectors $\vec{y}(w)$ estimated from Eq. 2 and the coefficient vectors $\vec{b}$ defined in Eq. 3 trained using linear regression on the whole of the MEN dataset. Then, the similarity matrix $S(w_i, w_j)$ is estimated for all pairs in the dataset using Eq. 3 and then the spectral clustering algorithm proposed in (Ng et al., 2001) is applied to obtain the lexical classes. In this work, neural fusion refers to early fusion (vector concatenation) of word vectors and neural activation vectors. We used a purity-based metric for evaluating the quality of the automatically created clusters. The purity P of the taxonomy is defined as in (Baroni and Lenci, 2010):

$$P = \frac{1}{d} \sum_{i=1}^{k} \max_{j}(e_{ij}), \tag{5}$$

where e_{ij} is the number of nouns assigned to the ith cluster that belong to the jth groundtruth class, k is the number of clusters, and d is the total number of concrete nouns included in the dataset. Purity expresses the fraction of nouns that belong to the true class, which is most represented in the cluster, taking values in the range [0,1].

Dataset	Neural Model	w2vec	Neural Fusion
ESSLI (6 classes)	0.61	0.70	**0.71**
ESSLI (3 classes)	0.77	0.95	**0.95**
ESSLI (2 classes)	0.66	**0.77**	0.72

Table 3: Evaluation results for taxonomy creation.

The evaluation results are presented in Table 3 for the neural activation model, the w2vec word embeddings (Mikolov et al., 2013) trained on the GoogleNews corpus and the late fusion of the two models (denoted as neural fusion) with equal weighting of their similarity matrices S. All results shown are computed on $V = 250$ voxels. The neural model performs worse that the w2vec model in all three subtasks (6, 3 or 2 classes), however, the proposed neural fusion achieves the best purity scores for 6 and 3 classes, at 0.71 and 0.95 purity, respectively.

6.2 Sense Classification

For the sensory modality (sense) classification task we use the Sensicon dataset to evaluate the performance of our model regarding sense discrimination. By definition all nouns in Sensicon are concrete nouns since they are associated with a real-world sensory stimulus. Sense classification is performed as

[5]The differences between the similarity scores estimated by our model and the baseline (i.e., w2vec) were found to be statistically significant at 99% level according to paired-sample t-test.

described in the previous section, i.e., the similarity matrix in Eq. 3 is calculated using the weight vector $\vec{b}$ trained on the MEN dataset and then the spectral clustering (Ng et al., 2001) is applied for the five sense categories. The resulting clusters are used for sense classification either between two senses, one versus all or among all five senses.

Classes	Neural Model	w2vec	Neural Fusion
Vision, Audition	0.55	0.55	**0.57**
Vision, Touch	0.68	0.66	**0.69**
Vision, Taste	0.60	0.60	**0.61**
Audition, Touch	0.59	0.58	**0.59**
Audition, Taste	0.57	0.55	**0.57**
Taste, Touch	0.54	0.54	0.54
Vision, Other	0.68	0.68	0.68
Audition, Other	0.74	0.74	0.74
Touch, Other	0.81	0.81	0.81
Taste, Other	0.78	0.78	**0.79**
Audition, Vision, Smell, Touch, Taste	0.37	0.33	**0.39**

Table 4: Evaluation results for two-way, one-vs-all, and five-way sense classification.

The evaluation results[6] are presented in Table 4 for the neural model, the w2vec model (same as the one used in the previous section) and the late fusion of the two (neural fusion) using equal weighting on the similarity matrices S. The evaluation metric used is the purity of clusters defined in Eq. 5. All results shown are computed on $V = 250$ voxels.

The neural and w2vec models achieve very similar results for two-way classification tasks, with the neural model performing better 0.37 vs 0.33 for five-way sense classification. The neural fusion model outperforms both neural and w2vec models for the majority of the two-way classification tasks and also achieves top performance for the five-way classification task reaching 0.39 purity score. Overall, the neural and neural fusion models show strong performance for this task, which is reasonable given the localization of sensory representations in the human cortex.

These results are also consistent with neuroscientific research (Heed, 2010; Marieb and Hoehn, 2007; Kobayashi, 2006; Ungerleider, 1982; Pickles, 1988; Buck and Axel, 1991).

6.3 Entailment Task

Next, we applied the neural activations to an entailment classification task. We used a Bi-LSTM model proposed in (Rocktäschel et al., 2015) featuring contextual attention (see Figure 2) as our baseline model. Word embeddings for concrete nouns were estimated using GloVe as detailed in (Pennington et al., 2014) and used as input to the Bi-LSTM network. The neural activations vectors were then combined via early fusion (vector concatenation) with GloVe embeddings (Pennington et al., 2014). Evaluation results for GloVe vectors and neural fusion are shown in Table 5 in terms of prediction accuracy. The results are

Dataset(SNLI)	Dimensions (GloVe, neural)	GloVe	Neural Fusion
3-common	(300,250)	68.2	**68.7**
2-common	(300,250)	76.6	**77.7**

Table 5: Entailment task accuracy for GloVe and neural fusion vector input to the Bi-LSTM.

reported for two subsets of the SNLI dataset, namely, 3-common and 2-common (see section 4).

We observe that the top accuracy is achieved by the fusion scheme for both the 3-common and 2-common subsets ($68.7 \pm 0.9\%$ and $77.7 \pm 0.9\%$).

[6]Note that the sense smell is not always present in Table 4 because smell has only four nouns compared in to other senses that contain more than 100 nouns each.

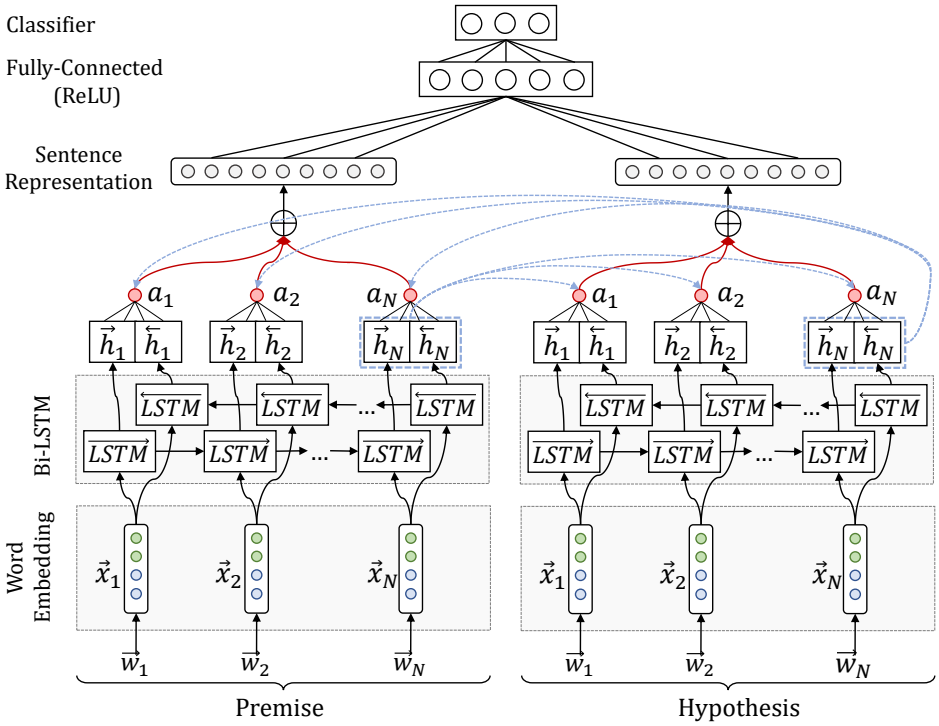


Figure 2: Bi-LSTM with context attention used in our experiments. Words' representations are either pretrained word embeddings or fusion of neural activations and word embeddings.

7 Conclusion

We proposed a simple neural activation semantic model extending the work of (Mitchell et al., 2008). The performance of the neural model was investigated for the tasks of word semantic similarity, taxonomy creation, sensory modality classification and concept entailment.

The analysis of the neural model word performance showed that the proposed model can differentiate between very similar and very dissimilar concrete nouns better than state-of-the-art word embeddings semantic models, while it performs worse overall for the word semantic similarity task. This is a strong indication that the semantic discriminability of neural activation vectors is of a different flavor than that of traditional word embedding vectors, and thus neural vectors can be used to augment state-of-the-art semantic representations. Results on the taxonomy classification, sense classification and entailment task indeed verify the different flavor of neural embeddings. For certain tasks, e.g., sense classification, neural models provide state-of-the-art performance. For other tasks, the fusion of neural and word2vec embeddings provides significant improvement. Overall (predicted) localized neural activation vectors can also be used in conjunction with other semantic representations and deep architectures to improve the results in challenging tasks, like concept entailment.

Although the collection of neuroimaging data has many limitations such as variation across participants, high signal-to-noise ratio and the need of expensive equipment for data capture, it provides an alternative view of how lexical and sensory information in localized in the human brain. Despite the very small dataset used in our experiments, results are encouraging about the value of neural activations patterns for computational tasks. In future work, we will investigate the relative performance of neural activations for a variety of natural language processing tasks, how to build neural vector predictors for the whole dictionary via lexical expansion, as well as more efficient fusion algorithms of neural and traditional word embedding models.

Acknowledgements. This work has been partially supported by EU H2020 BabyRobot project (grant #687831). Also, the authors would like to thank Christos Baziotis and Georgios Paraskevopoulos for many useful discussions.

References

[Anderson et al.2017] Andrew James Anderson, Jeffrey R. Binder, Leonardo Fernandino, Colin J. Humphries, Lisa L. Conant, Mario Aguilar, Xixi Wang, Donias Doko, and Rajeev D. S. Raizada. 2017. Predicting neural activity patterns associated with sentences using a neurobiologically motivated model of semantic representation. *Cerebral Cortex*, 27(9):4379–4395.

[Baroni and Lenci2010] Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

[Baroni et al.2008] M Baroni, S Evert, and A Lenci. 2008. Bridging the gap between semantic theory and computational simulations. In *Proc. of ESSLLI Distributional Semantic Workshop*.

[Baroni et al.2014] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland, June. Association for Computational Linguistics.

[Bengio et al.2000] Y Bengio, Rjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:932–938, 01.

[Benson et al.2001] Randall R. Benson, D.H. Whalen, Matthew Richardson, Brook Swainson, Vincent P. Clark, Song Lai, and Alvin M. Liberman. 2001. Parametrically dissociating speech and nonspeech perception in the brain using fmri. *Brain and Language*, 78(3):364 – 396.

[Bowman et al.2015] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September. Association for Computational Linguistics.

[Bruni et al.2014] Elia Bruni, N Tram, Marco Baroni, et al. 2014. Multimodal distributional semantics. *The Journal of Artificial Intelligence Research*, 49:1–47.

[Buck and Axel1991] Linda Buck and Richard Axel. 1991. A novel multigene family may encode odorant receptors: A molecular basis for odor recognition. *Cell*, 65(1):175 – 187.

[Bullinaria and Levy2012] John A. Bullinaria and Joseph P. Levy. 2012. Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and svd. *Behavior Research Methods*, 44(3):890–907, Sep.

[Collobert and Weston2008] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

[Cree and McRae2003] George S Cree and Ken McRae. 2003. Analyzing the factors underlying the structure and computation of the meaning of chipmunk, cherry, chisel, cheese, and cello (and many other such concrete nouns). *Journal of Experimental Psychology: General*, 132(2):163.

[Deerwester et al.1990] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

[Formisano et al.2008] Elia Formisano, Federico De Martino, Milene Bonte, and Rainer Goebel. 2008. "who" is saying "what"? brain-based decoding of human voice and speech. *Science*, 322(5903):970–973.

[Fyshe et al.2014] Alona Fyshe, Partha P. Talukdar, Brian Murphy, and Tom M. Mitchell. 2014. Interpretable semantic vectors from a joint model of brain- and text- based meaning. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 489–499, Baltimore, Maryland, June. Association for Computational Linguistics.

[G. Kanwisher et al.1997] N G. Kanwisher, Josh Mcdermott, and Marvin M. Chun. 1997. The fusiform face area: A module in human extrastriate cortex specialized for face perception. 17:4302–11, 07.

[Harris1954] Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

[Haxby et al.2001] James V. Haxby, M. Ida Gobbini, Maura L. Furey, Alumit Ishai, Jennifer L. Schouten, and Pietro Pietrini. 2001. Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science*, 293(5539):2425–2430.

[Heed2010] Tobias Heed. 2010. Touch perception: How we know where we are touched. *Current Biology*, 20(14):R604 – R606.

[Huth et al.2012] AlexanderG. Huth, Shinji Nishimoto, AnT. Vu, and JackL. Gallant. 2012. A continuous semantic space describes the representation of thousands of object and action categories across the human brain. *Neuron*, 76(6):1210 – 1224.

[Iosif et al.2016] Elias Iosif, Spiros Georgiladakis, and Alexandros Potamianos. 2016. Cognitively motivated distributional representations of meaning. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

[Ishai et al.1999] Alumit Ishai, Leslie G Ungerleider, Alex Martin, Jennifer L Schouten, and James V Haxby. 1999. Distributed representation of objects in the human ventral visual pathway. *Proceedings of the National Academy of Sciences*, 96(16):9379–9384.

[Jelodar et al.2010] Ahmad Babaeian Jelodar, Mehrdad Alizadeh, and Shahram Khadivi. 2010. Wordnet based features for predicting brain activity associated with meanings of nouns. In *Proceedings of the NAACL HLT 2010 First Workshop on Computational Neurolinguistics*, CN '10, pages 18–26, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Jing et al.2017] Wang Jing, Cherkassky Vladimir L., and Just Marcel Adam. 2017. Predicting the brain activation pattern associated with the propositional content of a sentence: Modeling neural representations of events and states. *Human Brain Mapping*, 38(10):4865–4881.

[Just et al.2010] Marcel Adam Just, Vladimir L. Cherkassky, Sandesh Aryal, and Tom M. Mitchell. 2010. A neurosemantic theory of concrete noun representation based on the underlying brain codes. *PLOS ONE*, 5(1):1–15, 01.

[Kay et al.2008] Kendrick Kay, Thomas Naselaris, Ryan J Prenger, and Jack Gallant. 2008. Identifying natural images from human brain activity. 452:352–5, 04.

[Kobayashi2006] Masayuki Kobayashi. 2006. Functional organization of the human gustatory cortex. *Journal of Oral Biosciences*, 48(4):244–260.

[Levy and Bullinaria2012] Joseph P Levy and John A Bullinaria. 2012. Using enriched semantic representations in predictions of human brain activity. *Connectionist Models of Neurocognition and Emergent Behavior: From Theory to Applications. Singapore: World Scientific*, pages 292–308.

[Marieb and Hoehn2007] Elaine Nicpon Marieb and Katja Hoehn. 2007. *Human anatomy & physiology*. Pearson Education.

[Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mitchell et al.2008] Tom M. Mitchell, Svetlana V. Shinkareva, Andrew Carlson, Kai-Min Chang, Vicente L. Malave, Robert A. Mason, and Marcel Adam Just. 2008. Predicting human brain activity associated with the meanings of nouns. *Science*, 320(5880):1191–1195.

[Miyawaki et al.2008] Yoichi Miyawaki, Hajime Uchida, Okito Yamashita, Masa aki Sato, Yusuke Morito, Hiroki C. Tanabe, Norihiro Sadato, and Yukiyasu Kamitani. 2008. Visual image reconstruction from human brain activity using a combination of multiscale local image decoders. *Neuron*, 60(5):915 – 929.

[Mnih and Hinton2007] Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM.

[Murphy et al.2012] Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012. Selecting corpus-semantic models for neurolinguistic decoding. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 114–123, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Naselaris et al.2009] Thomas Naselaris, Ryan J Prenger, Kendrick Kay, Michael Oliver, and Jack Gallant. 2009. Bayesian reconstruction of natural images from human brain activity. 63:902–15, 09.

[Ng et al.2001] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856. MIT Press.

[Nishimoto et al.2011] Shinji Nishimoto, AnT. Vu, Thomas Naselaris, Yuval Benjamini, Bin Yu, and JackL. Gallant. 2011. Reconstructing visual experiences from brain activity evoked by natural movies. *Current Biology*, 21(19):1641 – 1646.

[Palatucci et al.2009] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. 2009. Zero-shot learning with semantic output codes. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1410–1418. Curran Associates, Inc.

[Pedregosa et al.2011] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.

[Pickles1988] James O Pickles. 1988. *An introduction to the physiology of hearing*, volume 2. Academic press London.

[Pulvermller2001] Friedemann Pulvermller. 2001. Brain reflections of words and their meaning. *Trends in Cognitive Sciences*, 5(12):517 – 524.

[Rocktäschel et al.2015] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

[Ruan et al.2016] Yu-Ping Ruan, Zhen-Hua Ling, and Yu Hu. 2016. Exploring semantic representation in brain activity using word embeddings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 669–679, Austin, Texas, November. Association for Computational Linguistics.

[Socher et al.2011] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 151–161, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Socher et al.2013] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October. Association for Computational Linguistics.

[Sudre et al.2012] Gustavo Sudre, Dean Pomerleau, Mark Palatucci, Leila Wehbe, Alona Fyshe, Riitta Salmelin, and Tom Mitchell. 2012. Tracking neural coding of perceptual and semantic features of concrete nouns. *NeuroImage*, 62(1):451 – 463.

[Tekiroglu et al.2014] Serra Sinem Tekiroglu, Gözde Özbal, and Carlo Strapparava. 2014. Sensicon: An automatically constructed sensorial lexicon. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1511–1521, Doha, Qatar, October. Association for Computational Linguistics.

[Thirion et al.2006] Bertrand Thirion, Edouard Duchesnay, Edward Hubbard, Jessica Dubois, Jean-Baptiste Poline, Denis Lebihan, and Stanislas Dehaene. 2006. Inverse retinotopy: Inferring the visual content of images from brain activation patterns. *NeuroImage*, 33(4):1104 – 1116.

[Turian et al.2010] Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Ungerleider1982] Leslie G Ungerleider. 1982. Two cortical visual systems. *Analysis of visual behavior*, pages 549–586.

Folksonomication: Predicting Tags for Movies from Plot Synopses Using Emotion Flow Encoded Neural Network

Sudipta Kar **Suraj Maharjan** **Thamar Solorio**
Department of Computer Science
University of Houston
Houston, TX 77204-3010
{skar3, smaharjan2, tsolorio}@uh.edu

Abstract

Folksonomy of movies covers a wide range of heterogeneous information about movies, like the genre, plot structure, visual experiences, soundtracks, metadata, and emotional experiences from watching a movie. Being able to automatically generate or predict tags for movies can help recommendation engines improve retrieval of similar movies, and help viewers know what to expect from a movie in advance. In this work, we explore the problem of creating tags for movies from plot synopses. We propose a novel neural network model that merges information from synopses and emotion flows throughout the plots to predict a set of tags for movies. We compare our system with multiple baselines and found that the addition of emotion flows boosts the performance of the network by learning ≈18% more tags than a traditional machine learning system.

1 Introduction

User generated tags for online items are beneficial for both of the users and content providers in modern web technologies. For instance, the capability of tags in providing a quick glimpse of items can assist users to pick items precisely based on their taste and mood. On the other hand, such strength of tags enables them to act as strong search keywords and efficient features for recommendation engines (Lambiotte and Ausloos, 2006; Szomszor et al., 2007; Li et al., 2008; Borne, 2013). As a result, websites for different medias like photography[1], literature[2], film[3], and music[4] have adopted this system to make information retrieval easier. Such systems are often referred as Folksonomy (Vander Wal, 2005), social tagging, or collaborative tagging.

In movie review websites, it is very common that people assign tags to movies after watching them. Tags for movies often represent summarized characteristics of the movies such as emotional experiences, events, genre, character types, and psychological impacts. As a consequence, tags for movies became remarkably convenient for recommending movies to potential viewers based on their personal preferences and user profiles. However, this situation is not the same for all of the movies. Popular movies usually have a lot of tags as they tend to reach a higher number of users in these sites. On the other hand, low profile movies that fail to reach such an audience have very small or empty tagsets. In an investigation, we found that ≈34% of the movies among the top ≈130K movies of 22 genres[5] in IMDB do not have any tag at all. It is very likely that lack of descriptive tags negatively affects chances of movies being discovered.

An automatic process to create tags for movies by analyzing the written plot synopses or scripts could help solve this problem. Such a process would reduce the dependency on humans to accumulate tags

[1] http://www.flickr.com
[2] http://www.goodreads.com
[3] http://www.imdb.com
[4] http://www.last.fm
[5] http://www.imdb.com/genre/

Proceedings of the 27th International Conference on Computational Linguistics, pages 2879–2891
Santa Fe, New Mexico, USA, August 20-26, 2018.

for movies. Additionally, learning the characteristics of a movie plot and possible emotional experiences from the written synopsis is also an interesting problem by itself from the perspective of computational linguistics. As the attributes of movies are multi-dimensional, a tag prediction system for movies has to generate multiple tags for a movie. The application of predicting multiple tags from textual description is not necessarily limited to the domain of movie recommendation but also appropriate in other domains, such as video games and books, where storytelling is relevant. In this paper, we explore the problem of analyzing plot synopses to generate multiple plot-related tags for movies. Our key contributions in this paper are as follows:

- We create a neural system for predicting tags from narrative texts and provide a robust comparison against traditional machine learning systems. Table 1 shows examples of predicted tags by our system for four movies.
- We propose a neural network model that encodes flow of emotions in movie plot synopses. This emotion flow helps the model to learn more attributes of movie plots.
- We release our source code and a live demo of the tag prediction system at `http://ritual.uh.edu/folksonomication-2018`.

IMDB ID	Movie Title	Predicted Tags
tt0133093	The Matrix	though-provoking, action, sci-fi, suspenseful, mystery
tt0233298	Batman Beyond: Return of the Joker	action, good versus evil, suspenseful, humor, thought-provoking
tt0309820	Luther	murder, melodrama, intrigue, historical fiction, christian film
tt0163651	American Pie	adult comedy, cute, feel-good, prank, entertaining

Table 1: Example of predicted tags from the plot synopses of four movies. Blue and red labels indicate true positives and false positives respectively.

2 Related Work

Automatic tag generation from content-based analysis has drawn attention in different domains like music and images. For example, creating tags for music has been approached by utilizing lyrics (van Zaanen and Kanters, 2010; Hu et al., 2009), acoustic features from the tracks (Eck et al., 2008; Dieleman and Schrauwen, 2013), categorical emotion models (Kim et al., 2011), and deep neural models (Choi et al., 2017).

AutoTag (Mishne, 2006) and TagAssist (Sood et al., 2007), which utilize the text content to generate tags, aggregate information from similar blog posts to compile a list of ranked tags to present to the authors of new blog posts. Similar works (Katakis et al., 2008; Lipczak, 2008; Tatu et al., 2008) focused on recommending tags to users of BibSonomy[6] upon posting a new web page or publication as proposed systems in the ECML PKDD Discovery Challenge 2008 (Hotho et al., 2008) shared task. These systems made use of some kind of out of content resources like user metadata, and tags assigned to similar resources to generate tags.

Computational narrative studies deal with representing natural language stories by computational models that can be useful to understand, represent, and generate stories computationally. Current works attempt to model narratives using the character's personas and roles (Valls-Vargas et al., 2014; Bamman et al., 2013), interaction information between the characters (Iyyer et al., 2016; Chaturvedi et al., 2016; Chaturvedi et al., 2017) and events taking place throughout the stories (Goyal et al., 2010; Finlayson, 2012; McIntyre and Lapata, 2010). Other works try to build social networks of the characters (Agarwal et al., 2013a; Agarwal et al., 2013b; Agarwal et al., 2014; Krishnan and Eisenstein, 2015). Only a few works explored the possible type of impressions narrative texts can create on their consumers. For instance, different types of linguistic features have been used for success prediction for books (Ganjigunte Ashok et al., 2013; Maharjan et al., 2017) and tag prediction of movies from plot synopses (Kar et al., 2018). The tag prediction system predicts a fixed number of tags for each movie. But the tag space created by the system for the test data covers only 73% tags of the actual tagset as the system could capture a small portion of the multi-dimensional attributes of movie plots.

[6] `https://www.bibsonomy.org`

3 Dataset

We conduct our experiments on the Movie Plot Synopses with Tags (MPST) corpus (Kar et al., 2018), which is a collection of plot synopses for 14,828 movies collected from IMDb and Wikipedia. Most importantly, the corpus provides one or more fine-grained tags for each movie. The reason behind selecting this particular dataset is two-fold. First, the tagset is comprised of manually curated tags. These tags express only plot-related attributes of movies (e.g. suspenseful, violence, and melodrama) and are free of any tags foreign to the plots, such as metadata. Furthermore, grouping semantically similar tags and representing them by generalized tags helped to reduce the noise created by redundancy in tag space. Second, the corpus provides adequate amount of texts in the plot synopses as all the synopses have at least ten sentences. We follow the same split provided with the corpus, using 80% for training and 20% for test set. Table 2 gives statistics of the dataset.

Split	#Plot Synopses	#Tags	#Tags per Movie	#Sentence per Synopsis	#Words per Synopsis
Train	11862	71	2.97	42.36	893.39
Test	2966	71	3.04	42.61	907.96

Table 2: Statistics of the MPST corpus.

4 Encoding Emotion Flow with a Neural Network

Our proposed model simultaneously takes the emotion flow throughout the storyline and the text-based representation of the synopsis to retrieve relevant tags for a movie. Figure 1 shows the proposed architecture. The proposed neural architecture has three modules. The first module uses a convolutional neural network (CNN) to learn plot representations from synopses. The second module models the flow of emotions via a bidirectional long short-term memory (Bi-LSTM) network. And the last module contains hidden dense layers that operate on the combined representations generated by the first and second modules to predict the most likely tags for movies.

(a) Convolutional Neural Network (CNN): Recent successes in different text classification problems motivated us to extract important word level features using convolutional neural networks (CNNs) (dos Santos and Gatti, 2014; Kim, 2014; Zhang et al., 2015; Kar et al., 2017; Shrestha et al., 2017). We design a model that takes word sequences as input, where each word is represented by a 300-dimensional word embedding vector. We use randomly initialized word embeddings but also experiment with the FastText[7] word embeddings trained on Wikipedia using subword information. We stack 4 sets of one-dimensional convolution modules with 1024 filters each for filter sizes 2, 3, 4, and 5 to extract word-level n-gram features (Kim, 2014; Zhang et al., 2015). Each filter of size c is applied from window t to window $t + c - 1$ on a word sequence $x_1, x_2, \ldots, x_n$. Convolution units of filter size c calculate a convolution output using a weight map W_c, bias b_c, and the ReLU activation function (Nair and Hinton, 2010). The output of this operation is defined by:

$$h_{c,t} = ReLU(W_c x_{t:t+c-1} + b_c) \tag{1}$$

The ReLU activation function is defined by:

$$ReLU(x) = max(0, x) \tag{2}$$

Finally, each convolution unit produces a high-level feature map h_c.

$$h_c = [h_{c,1}, h_{c,2}, ..., h_{c,T-c+1},] \tag{3}$$

On those feature maps, we apply max-over-time pooling operation and take the maximum value as the feature produced a particular filter. We concatenate the outputs of the pooling operation for four filter sets that represent the feature representations for each plot synopsis.

(b) CNN with Flow of Emotions (CNN-FE): Stories can be described in terms of emotional shapes (Vonnegut, 1981), and it has been shown that the emotional arcs of stories are dominated by six

[7] https://fasttext.cc/docs/en/english-vectors.html

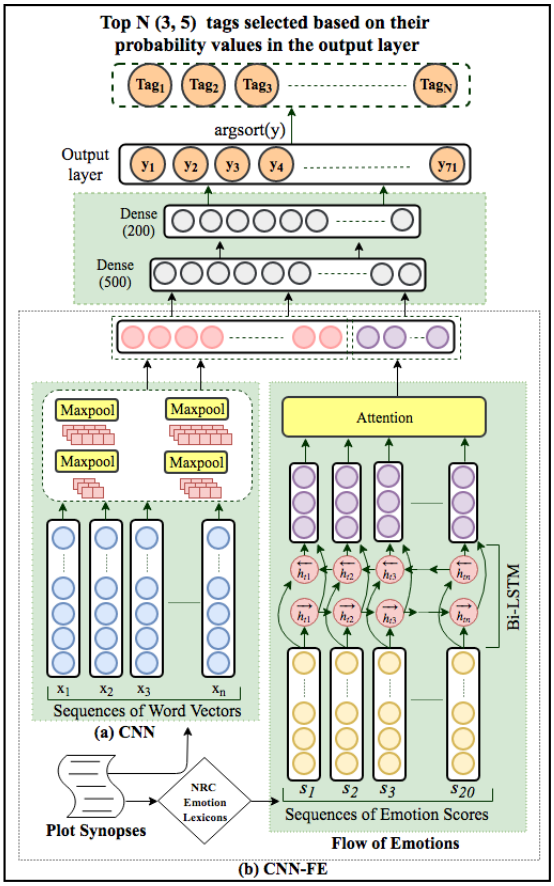


Figure 1: Convolutional Neural Network with Emotion Flow. The entire model is a combination of three modules. Module (a) learns feature representations from synopses using convolutional neural network. Module (b) incorporates emotion flows with module (a) to generate a combined representation of synopses. Module (c) uses these representations to predict the likelihood of each tag.

different shapes (Reagan et al., 2016). We believe that capturing the emotional ups and downs throughout the plots can help better understand how the story unfolds. This will enable us to predict relevant tags more accurately. So we design a neural network architecture that tries to learn representations of plots using the vector space model of words combined with the emotional ups and downs of plots.

Human emotion is a complex phenomenon to define computationally. The Hourglass of Emotions model (Cambria et al., 2012) categorized human emotions into four affective dimensions (*attention, sensitivity, aptitude, and pleasantness*), which started from the study of human emotions by Plutchik (2001). Each of these affective dimensions is represented by six different activation levels that make up to 24 distinct labels called 'elementary emotions' that represent the total emotional state of the human mind. NRC[8] emotion lexicons (Mohammad and Turney, 2013) is a list of 14,182 words[9] and their binary associations with eight types of elementary emotions from the Hourglass of Emotions model (*anger, anticipation, joy, trust, disgust, sadness, surprise,* and *fear*) with polarity. These lexicons have been used effectively in tracking the emotions in literary texts (Mohammad, 2011) and predicting success of books (Maharjan et al., 2018).

To model the flow of emotions throughout the plots, we divide each synopsis into N equally-sized segments based on words. For each segment, we compute the percentage of words corresponding to each emotion and polarity type (positive and negative) using the NRC emotion lexicons. More precisely, for a synopsis $x \epsilon X$, where X denotes the entire collection of plot synopses, we create N sequences of emotion vectors using the NRC emotion lexicons as shown below:

$$x \rightarrow s_{1:N} = [s_1, s_2, ..., s_N] \tag{4}$$

[8]National Research Council Canada

[9]Version 0.92

where s_i is the emotion vector for segment i. We experiment with different values of N, and $N = 20$ works better on the validation data.

As recurrent neural networks are good at encoding sequential data, we feed the sequence of emotion vectors into a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) with 16 units as shown in Figure 1. This bidirectional LSTM layer tries to summarize the contextual flow of emotions from both directions of the plots. The forward LSTMs read the sequence from s_1 to s_N, while the backward LSTMs read the sequence in reverse from s_N to s_1. These operations will compute the forward hidden states $(\overrightarrow{h_1}, \ldots, \overrightarrow{h_N})$ and backward hidden states $(\overleftarrow{h_1}, \ldots, \overleftarrow{h_N})$. For input sequence s, the hidden states h_t are computed using the following intermediate calculations:

$$
\begin{aligned}
i_t &= \sigma(W_{si}s_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{sf}s_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
c_t &= f_t c_{t-1} + i_t \tanh(W_{sc}s_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{sc}s_t + W_{hc}h_{t-1} + b_c) \\
h_t &= o_t \tanh(c_t)
\end{aligned}
$$

where, W and b denote the weight matrices and bias, respectively. σ is the sigmoid activation function, and i, f, o, and c are *input gate, forget gate, output gate*, and *cell* activation vectors, respectively. The annotation for each segment s_i is obtained by concatenating its forward hidden states $\overrightarrow{h_i}$ and backward hidden states $\overleftarrow{h_i}$, i.e. h_i=$[\overrightarrow{h_i}; \overleftarrow{h_i}]$. We then apply attention mechanism on this representation to get a unified representation of the emotion flow.

Attention models have been used effectively in many problems related to computer vision (Mnih et al., 2014; Ba et al., 2014) and have been successfully adopted in problems related to natural language processing (Bahdanau et al., 2014; Seo et al., 2016). An attention layer applied on top of a feature map h_i computes the weighted sum r as follows:

$$r = \sum_i \alpha_i h_i \tag{5}$$

and the weight α_i is defined as

$$\alpha_i = \frac{\exp(score(h_i))}{\sum_{i'} \exp(score(h_{i'}))}, \tag{6}$$

where, $score(.)$ is computed as follows:

$$score(h_i) = v^T tanh(W_a h_i + b_a) \tag{7}$$

where, W, b, v, and u are model parameters. Finally, we concatenate the representation of the emotion flow produced by the attention operation and the output vector with the vector representation generated from the CNN module.

The concatenated vector is then fed into two hidden dense layers with 500 and 200 neurons. To improve generalization of the model, we use dropout with a rate of 0.4 after each hidden layer. Finally, we add the output layer $\hat{y}$ with 71 neurons to compute predictions for 71 tags. To overcome the imbalance of the tags, we weight the posterior probabilities for each tag using different weight values. Weight value CW_t for tag $t \epsilon T$ is defined by,

$$CW_t = \frac{|D|}{|T| \times M_t} \tag{8}$$

where, $|D|$ is the size of the training set, $|T|$ is the number of classes, and M_t is the number of movies having tag t in the training set. We normalize the output layer by applying a softmax function defined by,

$$softmax(\hat{y}) = \frac{exp(\hat{y})}{\sum_{k=0}^{70} exp(\hat{y}_k)} \tag{9}$$

Based on the ranking for each tag, we then select top N (3/5/10) tags for a movie.

5 Experimental Setup

Data Processing and Training: As a preprocessing step, we lowercase the synopses, remove stop-words and also limit the vocabulary to top 5K words to reduce noise and data sparsity. Then we convert each synopsis into a sequence of 1500 integers where each integer represents the index of the corresponding word in the vocabulary. For the sequences longer than 1500 words, we truncate them from the left based on experiments on the development set. Shorter sequences are left padded with zeros.
During training, we use 20% of the training data as validation data. We tune various deep model parameters (dropouts, learning rate, weight initialization schemes, and batch size) using early stopping technique on the validation data. We use the Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951) to compute the loss between the true and predicted tag distributions and train the network using the RMSprop optimization algorithm (Tieleman and Hinton, 2012) with a learning rate of 0.0001. We implemented our neural network using the PyTorch deep learning framework[10].
Baselines: We compare the model performance against three baselines: majority baseline, random baseline, and traditional machine learning system. The majority baseline method assigns the most frequent three or five or ten tags in the training set to all the movies. Similarly, the random baseline assigns randomly selected three or five or ten tags to each movie. Finally, we compare our results with the benchmark system reported in Kar et al. (2018). This benchmark system used different types of hand-crafted lexical, semantic, and sentiment features to train a OneVsRest approach model with logistic regression as the base classifier.
Evaluation Measures: We try to follow the same evaluation methodology as described in Kar et al. (2018). We create two sets of tags for each movie by choosing the most likely three and five tags by the system. Additionally, we report our results on a wider range of tags, where we select top ten predictions. We evaluate the performance using the number of unique tags learned by the system (TL), micro averaged F1, and tag recall (TL). Tags learned (TL) computes how many unique tags are being predicted by the system for the test data (size of the tag space created by the model for test data). Tag recall represents the average recall per tag and it is defined by the following equation:

$$TR = \frac{\sum_{i=1}^{|T|} |R_i|}{|T|} \tag{10}$$

Here, $|T|$ is the total number of tags in the corpus, and R_i is the recall for the i^{th} tag.

6 Results and Discussions

	Top 3			Top 5			Top 10		
Methods	**TL**	**F1**	**TR**	**TL**	**F1**	**TR**	**TL**	**F1**	**TR**
Baseline: Most Frequent	3	29.7	4.23	5	28.4	14.08	10	28.4	13.73
Baseline: Random	71	4.2	4.21	71	6.4	15.04	71	6.6	14.36
Baseline: Kar et al. (2018)	47	**37.3**	**10.52**	52	**37.3**	**16.77**	—	—	—
CNN without class weights	24	36.8	7.99	26	36.7	12.62	27	**31.3**	24.52
CNN with class weights	49	34.9	9.85	55	35.7	14.94	67	30.8	**26.86**
CNN-FE	**58**	36.9	9.40	**65**	36.7	14.11	**70**	31.1	24.76
CNN-FE + FastText	53	**37.3**	10.00	59	36.8	15.47	63	30.6	26.45

Table 3: Performance of tag prediction systems on the test data. We report results of two setups using three matrices (TL: Tags learned, F1: Micro f1, TR: Tag recall).

Table 3 shows our results for Top 3, Top 5, and Top 10 settings. We will mainly discuss the results achieved by selecting top five tags as it allows us to compare with all the baseline systems and more tags to discuss about. As the most frequent baseline system assigns a fixed set of tags to all the movies, it fails to exhibit diversity in the created tag space. Still it manages to achieve a micro-F1 score around 28%. On the other hand, the random baseline system creates the most diverse tag space by using all of the possible tags. However its lower micro-F1 score of 6.30% makes it impractical to be used in real world scenario.

[10] https://pytorch.org

At this point, we find an interesting trade-off between accuracy and diversity. It is expected that a good movie tagger will be able to capture the multi-dimensional attributes of the plots that allows to generalize a diverse tag space. Tagging a large collection of movies with a very small and fixed set of tags (e.g. majority baseline system) is not useful for either a recommendation system or users. Equally important is the relevance between the movies and the tags created for those movies. The hand-crafted features based approach (Kar et al., 2018) achieves a micro-F1 around 37%, which outperforms the majority and random baselines. But the system was able to learn only 52 tags, which makes 73% of the total tags.

Our approach achieves a lower micro-F1 score than the traditional machine learning one, but it performs better in terms of learning more tags. We observe that the micro-F1 of the CNN model with only word sequences is very close (36.7%) to the hand-crafted features based system. However, it is able to learn only around 37% of the tags. By utilizing class weights in this model (see Eq. 8), we improve the learning for under-represented tags yielding an increase in *tag recall* (TR) and *tags learned* (TL). But the micro-f1 drops to 35.7%. With the addition of emotion flows to CNN, the CNN-FE model learns significantly more tags while micro-F1 and tag recall do not change much. Initializing the embedding layer with pre-trained embeddings made a small improvement in micro-F1 but the model learns comparatively lesser tags. If we compare the CNN-FE model with the hand-crafted feature based system, micro-F1 using CNN-FE is slightly lower ($\approx 1\%$) than the feature based system. But it provides a strong improvement in terms of the number of tags it learns (TL). CNN-FE learns around 91% tags of the tagset compared to 73% with the feature based system. It is an interesting improvement, because model is learning more tags and it is better at assigning relevant tags to movies. We observe similar pattern for the rest of the two sets of tags where we select top three and ten tags. For all the sets, CNN-FE model learns the highest number of tags compared to the other models. In terms of micro-F1 and tag recall, it does not achieve the highest numbers but performs very closely.

Incompleteness in Tag Spaces: One of the limitations of folksonomies is the incompleteness in tag spaces. The fact that users have not tagged an item with a specific label does not imply that that label does not apply to the item. Incompleteness makes learning challenging for computational models as the training and evaluation process penalizes the model for predicting a tag that is not present in the ground truth tags, even though in some cases it may be a suitable tag. For example, ground truth tags for the movie *Luther (2003)*[11] are *murder, romantic,* and *violence* (Table 1). And the predicted tags from our proposed model are *murder, melodrama, intrigue, historical fiction,* and *christian film*. The film is indeed a Christian film[12] portraying the biography of Martin Luther, who led the Christian reformation during the 16th century. According to the Wikipedia, "*Luther is a 2003 American-German epic historical drama film loosely based on the life of Martin Luther*"[13]. Similarly, Edtv[14] (Table 6) has tags *romantic* and *satire* in the dataset. Our system predicted *adult comedy* and this tag is appropriate for this movie.

In these two cases, the system will get lower micro-F1 since the relevant tags are not part of the ground truth. Perhaps a different evaluation scheme could be better suited for this task. We plan to work on this issue in our future work.

Significance of the Flow of Emotions: The results suggest that incorporating the flow of emotions helps to achieve better results by learning more tags. Figure 2 shows some tags with significant improvements in recall after incorporating the flow of emotions. We notice such improvements for around 30 tags. We argue that for these tags (e.g. *absurd, cruelty, thought-provoking, claustrophobic*) the changes in specific sentiments are adding new information helpful for identifying relevant tags. But we also notice negative changes in recall for around 10 tags, which are mostly related to the theme of the story (e.g. *blaxploitation, alternate history, historical fiction, sci-fi*). It will be an interesting direction of future work to add a mechanism that can also learn to discern when emotion flow should contribute more to the prediction task.

In Figure 3, we inspect how the flow of emotions looks like in different types of plots. Emotions like *joy* and *trust* are continuously dominant over *disgust* and *anger* in the plot of *Arthur (1981)*, which is

[11] http://www.imdb.com/title/tt0309820

[12] https://www.christianfilmdatabase.com/review/luther-2

[13] https://en.wikipedia.org/wiki/Luther_(2003_film)

[14] http://www.imdb.com/title/tt0131369/

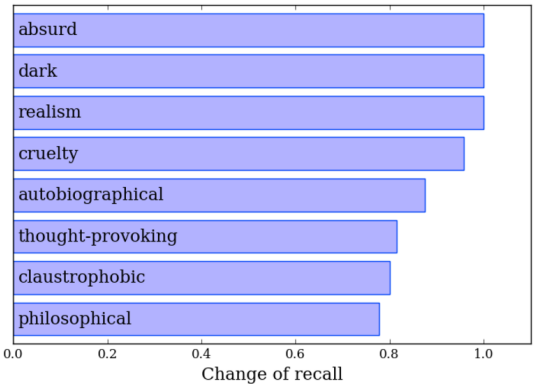


Figure 2: Tags with higher change of recall after adding the flow of emotions in CNN.

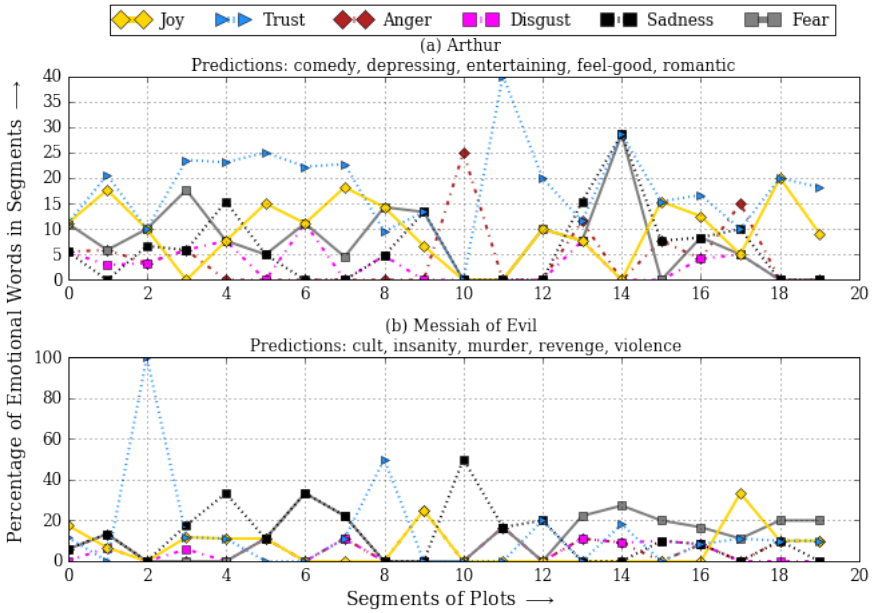


Figure 3: The flow of emotions in the plots of 2 different types of movies. Each synopsis was divided into 20 segments based on the words, and percentage of the emotions for each segment was calculated using the NRC emotion lexicons. The y axis represents the percentage of emotions in each segment; whereas, the x axis represents the segments.

a comedy film. We can observe sudden spikes in *sadness* and *fear* at segment 14, which is the possible reason for triggering the tag *depressing*. We observe a different pattern in the flow of emotions in *Messiah of Evil (1973)*, which is a horror film. Here the dominant emotions are *sadness* and *fear*. Such characteristics of emotions are helpful to determine the type and possible experiences from a movie. Our model seems to be able to leverage this information that is allowing it to learn more tags; specifically tags that are related to feelings.

Learning or Copying? We found that only 11.8% of the 14,830 predicted tags for the ∼3K movies in the test data were found in the synopses themselves. 12.7% of the total 9,022 ground truth tags appear in the plot synopses. These numbers suggest that the model is not dependent on the occurrences of the tags in the synopses to make predictions, rather it seems it is trying to understand the plots and assign tags based on that. We also found that all the tags that were present in the synopses of the test data are also present in the synopses of the training data. Then we investigate what type of tags appear in the synopses and which ones do not. Tags present in the synopses are mostly genre or event related tags like *horror, violence, historical*. On the other hand, most of the tags that do not appear in the synopses are the tags that require a more sophisticated analysis of the plots synopses (e.g. *thought-provoking, feel-good,*

suspenseful). It is not necessarily bad to predict tags that are in the synopses, since they are still useful for recommender systems. However, if this was the only ability of the proposed models, their value would be limited. Luckily this analysis, and the results presented earlier show that the model is able to infer relevant tags, even if they have not been observed in the synopses. This is a much more interesting finding.

Learning Stories from Different Representations: Movie scripts represent the detailed story of a movie, whereas the plot synopses are summaries of the movie. The problem with movie scripts is that they are not as readily available as plot synopses. However, it is still interesting to evaluate our approach to predict tags from movie scripts. For this purpose, we collected movie scripts from our test set. We

	Top 3			Top 5		
	F1	**TR**	**TL**	**F1**	**TR**	**TL**
Plot Synopses	29.3	8.04	28	38.7	15.70	35
Scripts	29.8	5.16	19	37.0	9.27	26

Table 4: Evaluation of predictions using plot synopses and scripts

were able to find 80 movie scripts using the ScriptBase corpus (Gorinski and Lapata, 2015).

In table 4, we show the evaluation of tags generated using plot synopses and scripts. Despite having similar micro-f1 scores, *tag recall* and *tags learned* are lower when we use the scripts. A possible explanation for this is the train/test mismatch since the model was trained using summarized versions of the movie, while the test data contained full movies scripts. Additional sources of error could come from the external info included in scripts (such as descriptions of actions from the characters or settings).

Percentage of Match	**Percentage of Movies**
>=80%	40%
>=40% & <80%	47.5%
>=20% & <40%	11.25%

Table 5: Percentage of the match between the sets of top five tags generated from the scripts and plot synopses.

Table 5 shows that for most of the movies we generate very similar tags using the scripts and plot synopses. For 40% movies, at least 80% tags are the same. While the predictions are not identical, these results show a consistency in the learned tags from our system. An interesting direction for future work would be to study what aspects in a full movie script are relevant to predict tags.

Title: A Nightmare on Elm Street 5: The Dream Child **Ground Truths:** cult, good versus evil, insanity, murder, sadist, violence **Synopsis:** cult, murder, paranormal, revenge, violence **Script:** murder, violence, flashback, cult, suspenseful
Title: EDtv **Ground Truths:** romantic, satire **Synopsis:** adult comedy, comedy, entertaining, prank, satire **Script:** comedy, satire, prank, entertaining, adult comedy
Title: Toy Story **Ground Truths:** clever, comedy, cult, cute, entertaining, fantasy, humor, violence **Synopsis:** comedy, cult, entertaining, humor, psychedelic **Script:** psychedelic, comedy, entertaining, cult, absurd
Title: Margot at the Wedding **Ground Truths:** romantic, storytelling, violence **Synopsis:** depressing, dramatic, melodrama, queer, romantic **Script:** psychological, murder, mystery, flashback, insanity

Table 6: Example of ground truth tags of movies from the test set and the generated tags for them using plot synopses and scripts.

Challenging Tags: We found that these seven tags: *stupid, grindhouse film, blaxploitation, magical realism, brainwashing, plot twist,* and *allegory*, were not assigned to any movies in the test set. One reason might be that these are very infrequent (around 0.06% of movies have them assigned as their tags). This

will obviously make them difficult to learn. Again, these are subjective as well. We believe that tagging a plot as stupid or brainwashing is complicated and depends on perspectives of a tagger. We plan to investigate such type of tags in the future.

7 Conclusions and Future Work

In this paper we explore the problem of automatically creating tags for movies using plot synopses. We propose a model that learns word level feature representations from the synopses using CNNs and models sentiment flow throughout the plots using a bidirectional LSTM. We evaluated our model on a corpus that contains plot synopses and tags of 14K movies. We compared our model against a majority and random baselines, and a system that uses traditional hand-crafted linguistic features. We found that incorporating emotion flows boosts prediction performance by improving the learning of tags related to feelings as well as increasing the overall number of tags learned.
Predicting tags for movies is an interesting and complicated problem at the same time. To further improve our results, we plan to investigate more sophisticated architectures and explore ways to tackle the problem of incompleteness in the tag space. We also plan to evaluate the quality of predicted tags using a human study evaluation and experiment on predicting tags in other storytelling related domains.

Acknowledgements

This work was partially supported by the National Science Foundation under grant number 1462141 and by the U.S. Department of Defense under grant W911NF-16-1-0422.

References

Apoorv Agarwal, Anup Kotalwar, and Owen Rambow. 2013a. Automatic extraction of social networks from literary text: A case study on alice in wonderland. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1202–1208, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Apoorv Agarwal, Anup Kotalwar, Jiehan Zheng, and Owen Rambow. 2013b. Sinnet: Social interaction network extractor from text. In *The Companion Volume of the Proceedings of IJCNLP 2013: System Demonstrations*, pages 33–36, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Apoorv Agarwal, Sriramkumar Balasubramanian, Anup Kotalwar, Jiehan Zheng, and Owen Rambow. 2014. Frame semantic tree kernels for social network extraction from text. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 211–219, Gothenburg, Sweden, April. Association for Computational Linguistics.

Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. 2014. Multiple object recognition with visual attention. *CoRR*, abs/1412.7755.

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

David Bamman, Brendan O'Connor, and Noah A. Smith. 2013. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 352–361, Sofia, Bulgaria, August. Association for Computational Linguistics.

Kirk Borne. 2013. Collaborative annotation for scientific data discovery and reuse. *Bulletin of the American Society for Information Science and Technology*, 39(4):44–45.

Erik Cambria, Andrew Livingstone, and Amir Hussain. 2012. The hourglass of emotions. In *Proceedings of the 2011 International Conference on Cognitive Behavioural Systems*, COST'11, pages 144–157, Berlin, Heidelberg. Springer-Verlag.

Snigdha Chaturvedi, Shashank Srivastava, Hal Daumé III, and Chris Dyer. 2016. Modeling evolving relationships between characters in literary novels. In Dale Schuurmans and Michael P. Wellman, editors, *AAAI*, pages 2704–2710. AAAI Press.

Snigdha Chaturvedi, Mohit Iyyer, and Hal Daumé III. 2017. Unsupervised learning of evolving relationships between literary characters.

K. Choi, G. Fazekas, M. Sandler, and K. Cho. 2017. Convolutional recurrent neural networks for music classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2392–2396, March.

Sander Dieleman and Benjamin Schrauwen. 2013. Multiscale approaches to music audio feature learning. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*, November 4-8. http://www.ppgia.pucpr.br/ismir2013/wp-content/uploads/2013/09/69_Paper.pdf.

Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Douglas Eck, Paul Lamere, Thierry Bertin-mahieux, and Stephen Green. 2008. Automatic generation of social tags for music recommendation. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 385–392. Curran Associates, Inc.

Mark Alan Finlayson. 2012. *Learning narrative structure from annotated folktales*. Ph.D. thesis, Massachusetts Institute of Technology.

Vikas Ganjigunte Ashok, Song Feng, and Yejin Choi. 2013. Success with style: Using writing style to predict the success of novels. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1753–1764, Seattle, Washington, USA, October. Association for Computational Linguistics.

Philip John Gorinski and Mirella Lapata. 2015. Movie script summarization as graph-based scene extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1066–1076, Denver, Colorado, May–June. Association for Computational Linguistics.

Amit Goyal, Ellen Riloff, and Hal Daumé, III. 2010. Automatically producing plot unit representations for narrative text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 77–86, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Andreas Hotho, Dominik Benz, Robert Jáschke, and Beate Krause. 2008. Ecml pkdd discovery challenge 2008 (rsdc'08). In *Workshop at 18th Europ. Conf. on Machine Learning (ECML'08)/11th Europ. Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'08)*, volume 32.

Xiao Hu, J. Stephen Downie, and Andreas F. Ehmann. 2009. Lyric text mining in music mood classification. In *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009*, pages 411–416.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1534–1544. Association for Computational Linguistics.

Sudipta Kar, Suraj Maharjan, and Thamar Solorio. 2017. RiTUAL-UH at semeval-2017 task 5: Sentiment analysis on financial data using neural networks. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 877–882.

Sudipta Kar, Suraj Maharjan, A. Pastor López-Monroy, and Thamar Solorio. 2018. MPST: A corpus of movie plot synopses with tags. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France, May. European Language Resources Association (ELRA).

Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2008. Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD 2008 Discovery Challenge*.

JungHyun Kim, Seungjae Lee, SungMin Kim, and Won Young Yoo. 2011. Music mood classification model based on arousal-valence values. In *Advanced Communication Technology (ICACT), 2011 13th International Conference on*, pages 292–295. IEEE.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.

Vinodh Krishnan and Jacob Eisenstein. 2015. "You're Mr. Lebowski, I'm the Dude": Inducing address term formality in signed social networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1616–1626, Denver, Colorado, May–June. Association for Computational Linguistics.

S. Kullback and R. A. Leibler. 1951. On Information and Sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 03.

Renaud Lambiotte and Marcel Ausloos, 2006. *Collaborative Tagging as a Tripartite Network*, pages 1114–1117. Springer Berlin Heidelberg, Berlin, Heidelberg.

Xin Li, Lei Guo, and Yihong Eric Zhao. 2008. Tag-based social interest discovery. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 675–684, New York, NY, USA. ACM.

Marek Lipczak. 2008. Tag recommendation for folksonomies oriented towards individual users. In *In: Proc. of the ECML PKDD Discovery Challenge*.

Suraj Maharjan, John Arevalo, Manuel Montes, Fabio A González, and Thamar Solorio. 2017. A multi-task approach to predict likability of books. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1217–1227.

Suraj Maharjan, Sudipta Kar, Manuel Montes, Fabio A. Gonzalez, and Thamar Solorio. 2018. Letting emotions flow: Success prediction by modeling the flow of emotions in books. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 259–265, New Orleans, Louisiana, June. Association for Computational Linguistics.

Neil McIntyre and Mirella Lapata. 2010. Plot induction and evolutionary search for story generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1562–1572, Uppsala, Sweden, July. Association for Computational Linguistics.

Gilad Mishne. 2006. Autotag: A collaborative approach to automated tag assignment for weblog posts. In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, pages 953–954, New York, NY, USA. ACM.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. Recurrent models of visual attention. *CoRR*, abs/1406.6247.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a word-emotion association lexicon. 29(3):436–465.

Saif Mohammad. 2011. From once upon a time to happily ever after: Tracking emotions in novels and fairy tales. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, LaTeCH '11, pages 105–114, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA. Omnipress.

Robert Plutchik. 2001. The nature of emotions human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice. *American scientist*, 89(4):344–350.

Andrew J. Reagan, Lewis Mitchell, Dilan Kiley, Christopher M. Danforth, and Peter Sheridan Dodds. 2016. The emotional arcs of stories are dominated by six basic shapes. *CoRR*, abs/1606.07772.

Paul Hongsuck Seo, Zhe Lin, Scott Cohen, Xiaohui Shen, and Bohyung Han. 2016. Hierarchical attention networks. *CoRR*, abs/1606.02393.

Prasha Shrestha, Sebastian Sierra, Fabio Gonzalez, Manuel Montes, Paolo Rosso, and Thamar Solorio. 2017. Convolutional neural networks for authorship attribution of short texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 669–674, Valencia, Spain. Association for Computational Linguistics.

Sanjay C. Sood, Kristian J. Hammond, Sara H. Owsley, and Larry Birnbaum, 2007. *TagAssist: Automatic tag suggestion for blog posts*.

Martin Szomszor, Ciro Cattuto, Harith Alani, Kieron O'Hara, Andrea Baldassarri, Vittorio Loreto, and Vito D.P. Servedio. 2007. Folksonomies, the semantic web, and movie recommendation.

M. Tatu, M. Srikanth, and T. D'Silva. 2008. *RSDC'08: Tag Recommendations using Bookmark Content*. Workshop at 18th Europ. Conf. on Machine Learning (ECML'08) / 11th Europ. Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'08).

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Josep Valls-Vargas, Jichen Zhu, and Santiago Ontanón. 2014. Toward automatic role identification in unannotated folk tales. In *Proceedings of the Tenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 188–194. AAAI Press.

Menno van Zaanen and Pieter Kanters. 2010. Automatic mood classification using tf*idf based on lyrics. In *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010, Utrecht, Netherlands, August 9-13, 2010*, pages 75–80.

Thomas Vander Wal. 2005. Folksonomy definition and wikipedia. *vanderwal. net*.

Kurt Vonnegut. 1981. Palm sunday: An autobiographical collage.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.

Emotion Representation Mapping for Automatic Lexicon Construction (Mostly) Performs on Human Level

Sven Buechel & Udo Hahn
{sven.buechel|udo.hahn}@uni-jena.de
Jena University Language & Information Engineering (JULIE) Lab
Friedrich-Schiller-Universität Jena, Jena, Germany
http://www.julielab.de

Abstract

Emotion Representation Mapping (ERM) has the goal to convert existing emotion ratings from one representation format into another one, e.g., mapping Valence-Arousal-Dominance annotations for words or sentences into Ekman's Basic Emotions and vice versa. ERM can thus not only be considered as an alternative to Word Emotion Induction (WEI) techniques for automatic emotion lexicon construction but may also help mitigate problems that come from the proliferation of emotion representation formats in recent years. We propose a new neural network approach to ERM that not only outperforms the previous state-of-the-art. Equally important, we present a refined evaluation methodology and gather strong evidence that our model yields results which are (almost) as reliable as human annotations, even in cross-lingual settings. Based on these results we generate new emotion ratings for 13 typologically diverse languages and claim that they have near-gold quality, at least.

1 Introduction

From its inception, researchers in the field of sentiment analysis aimed at predicting the affective state that is typically associated with a given word based on a list of linguistic features, a problem referred to as *word emotion induction* (WEI) (Hatzivassiloglou and McKeown, 1997; Turney and Littman, 2003). Early research activities have focused on *semantic polarity* (the positiveness or negativeness of a feeling) for quite a long time. But more recently this focus on binary representations has been replaced by more expressive *emotion representation formats* such as Basic Emotions or Valence-Arousal-Dominance. In the meantime, WEI has become an active area of research, regularly featured in shared tasks (Rosenthal et al., 2015; Yu et al., 2016b). Based on these achievements, WEI techniques have become a natural methodological choice for the automatic construction of emotion lexicons (Köper and Schulte im Walde, 2016; Shaikh et al., 2016).

Yet, only very recently, a radically different approach to automatic emotion lexicon construction has been proposed. Instead of relying on linguistic features (such as similarity with seed words or word embeddings), the goal of *emotion representation mapping* (ERM) is to derive new emotional word ratings *in one format* based on known ratings of the same words *in another format* (Buechel and Hahn, 2017a). For example, ERM could use empirically gathered ratings for Basic Emotions and convert them into a Valence-Arousal-Dominance representation scheme, with greater precision than currently achievable by WEI algorithms. As a much appreciated side effect, one of the promises of ERM is to make otherwise incompatible resources (lexicons or annotated corpora, as well as tools) compatible, and incomparable systems comparable. Thus, this approach has the potential to mitigate some of the negative effects that arise from not having a community-wide standard for emotion annotation and representation (Calvo and Mac Kim, 2013; Buechel and Hahn, 2018a).

We here want to contribute to this endeavor by providing a large-scale evaluation of previously proposed ERM approaches for four typologically diverse languages and report evidence that ERM clearly

Proceedings of the 27th International Conference on Computational Linguistics, pages 2892–2904
Santa Fe, New Mexico, USA, August 20-26, 2018.

outperforms current state-of-the-art WEI algorithms. Furthermore, we present our own deep learning model which performs even better against all competitors. Most importantly, however, we propose a new methodology for comparing the reliability of ERM against human annotation reliability, a major shortcoming of previous work. As a result, we find that our proposed model performs competitive to a reasonably large group of human raters, *even in cross-lingual settings*. Based on this evidence, we automatically construct emotion lexicons for 13 languages and claim that they have (near) gold quality. These lexicons as well as our experimental code base and results are publically available.[1]

2 Related Work

Psychological Models of Emotion. Models of emotion typically fall into two main groups, namely *discrete* (or *categorical*) and *dimensional* ones (Stevenson et al., 2007; Calvo and Mac Kim, 2013). Discrete models are built around particular sets of emotional categories deemed fundamental and universal. Ekman (1992), for instance, identifies six *Basic Emotions* (Joy, Anger, Sadness, Fear, Disgust and Surprise). In contrast, dimensional models consider emotions to be composed out of several influencing factors (mainly two or three). These are often referred to as *Valence* (corresponding to the concept of polarity), *Arousal* (a calm–excited scale), and *Dominance* (perceived degree of control over a (social) situation)—the VAD model. The last dimension, Dominance, is quite often omitted, thus constituting the VA model. For convenience, both will be jointly referred to as VA(D). An illustration of VAD and its relationship to Basic Emotions is given in Figure 1.

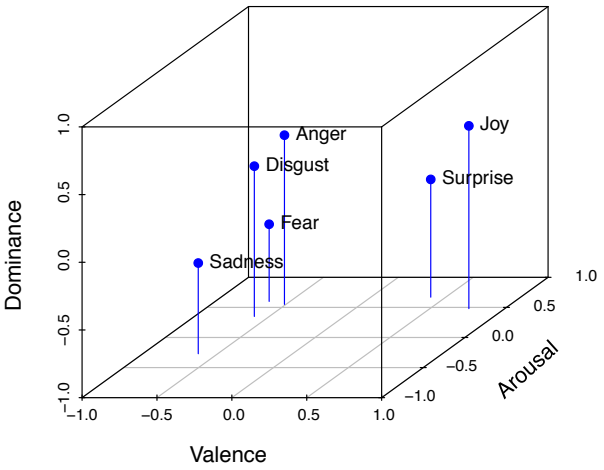


Figure 1: Affective space spanned by the Valence-Arousal-Dominance (VAD) model, together with the position of six Basic Emotions. Adapted from Buechel and Hahn (2016).

Lexical Data Sets. In contradistinction to NLP where many different representation formats for emotions are being used, lexical resources originating from psychology labs almost exclusively subscribe either to VA(D) or Basic Emotions models (typically omitting Surprise; the BE5 format). Over the years, a considerable number of resources built on these premises have emerged from psychological research for various languages.[2] In more detail, these lexical ratings have been gathered via questionnaire studies by collecting individual ratings from a large number of subjects for each lexical item under consideration (typically between 20 to 30 individual ratings per item). These individual assessments are then averaged to yield aggregated scores on which we base our experiments. The emotion values we deal with must thus be understood as an average emotional reaction when presenting a lexical stimulus to a group of human judges.

In this paper, we restrict ourselves to the VA(D) and BE5 format. Following the conventions of the emotion lexicons used in our experiments (Table 1), each VA(D) dimension receives a value from the interval $[1, 9]$ where '1' means "most negative/calm/submissive", '9' means "most positive/excited/dominant" and '5' means "neutral". Conversely, values for BE5 categories range in the interval $[1, 5]$ where '1' means "absence" and '5' means "most extreme" expression of the respective emotion.[3] Consequently, the VA(D) and BE5 formats are conceptually different from one another insofar as VA(D) dimensions are bi-polar, whereas BE5 categories are uni-polar.

[1] `https://github.com/JULIELab/EmoMap`

[2] See, e.g., Tables 1 and 6. An enhanced list of these and similar data sets is provided in Buechel and Hahn (2018a).

[3] Although these intervals are fairly well established conventions, in some data sets different rating scales were used, nevertheless. In these cases, we linearly transformed the ratings so that they match the defined intervals.

Abbrev.	VA(D)	BE5	Dom?	Overlap
en_1	Bradley and Lang (1999)	Stevenson et al. (2007)	✓	1,028
en_2	Warriner et al. (2013)	Stevenson et al. (2007)	✓	1,027
es_1	Redondo et al. (2007)	Ferré et al. (2017)	✓	1,012
es_2	Hinojosa et al. (2016b)	Hinojosa et al. (2016a)	✓	875
es_3	Stadthagen-Gonzalez et al. (2017b)	Stadthagen-González et al. (2017a)	✗	10,491
de_1	Võ et al. (2009)	Briesemeister et al. (2011)	✗	1,958
pl_1	Riegel et al. (2015)	Wierzba et al. (2015)	✗	2,902
pl_2	Imbir (2016)	Wierzba et al. (2015)	✓	1,272

Table 1: Data sets used in our experiments; with abbreviation (including language code according to ISO 639-1), the bibliographic sources of the VA(D) and BE5 ratings, information on whether Dominance is included and the number of overlapping entries.

Word Emotion Induction. Automatically constructing such word-level emotion data sets has been a focus of NLP-based sentiment analysis studies from the beginning. In fact, the problem to automatically predict polarity or emotion scores for a given word based on some linguistic features—often referred to as Word Emotion Induction (WEI)—is already dealt with in the seminal work of Hatzivassiloglou and McKeown (1997). At first, the features taken into account were typically derived from co-occurrence or terminology-based similarity with a small set of *seed word* with known emotional scores (Turney and Littman, 2003; Esuli and Sebastiani, 2005). Nowadays, these features are almost completely replaced by *word embeddings*, i.e., dense, low-dimensional vector representations of words that are trained on large volumes of raw text in an unsupervised manner. WORD2VEC (Mikolov et al., 2013), GLOVE (Pennington et al., 2014) and FASTTEXT (Bojanowski et al., 2017) are among today's most popular algorithms for generating embeddings.

WEI algorithms constitute a natural baseline for ERM because, first, they produce the same output (emotion ratings for words according to some emotion representation format), yet their predictions are based on expressively weaker features (word embeddings instead of emotion ratings for the same word but in another format), thus constituting a harder task. Second, they form the currently prevailing paradigm for the automatic construction of emotion lexicons (Köper and Schulte im Walde, 2016; Shaikh et al., 2016), a problem for which ERM offers a promising alternative.

Emotion Representation Mapping. In contrast to WEI, ERM is based on the condition that the pairs of data sets in Table 1 are complementary in the sense that, when combining these lexicons, a subset of their entries are then encoded in *both* emotion formats, i.e., VA(D) *and* BE5. This condition is illustrated for three lexical items in Table 2.

Word	V	A	D	J	A	S	F	D
sunshine	8.1	5.3	5.4	4.3	1.2	1.3	1.3	1.2
terrorism	1.6	7.4	2.7	1.1	3.0	3.4	4.1	2.5
orgasm	8.0	7.2	5.8	4.3	1.3	1.3	1.4	1.2

Table 2: Three lexical items and their emotion values in VAD (second column group) and BE5 (third column group) format. VAD scores are taken from Warriner et al. (2013), BE5 scores were automatically derived (see Section 4.4).

Although such complementary data sets have been available for quite some time, ERM has only recently been introduced to NLP by Buechel and Hahn (2016) in order to compare a newly proposed VAD-based prediction system against previously established results on Basic Emotion gold standards. In a follow-up study, Buechel and Hahn (2017b) devised EMOBANK, a VAD-annotated corpus which, in part, also bears BE5 ratings on the *sentence* level. They found that both kinds of annotation were highly predictive for each other using a k-Nearest-Neighbor approach. In later studies, they examined the potential of ERM as a substitute for manual annotation of *lexical* items, also in cross-lingual settings (Buechel and Hahn, 2017a; Buechel and Hahn, 2018a). Although their evaluation was limited in expressiveness, they already found evidence that ERM may be comparable to human performance in terms of the quality of the resulting ratings.

Similar work has, to the best of our knowledge, only been done in the psychology domain. However, related work from this area does not target the goal of predictive modeling (Stevenson et al., 2007; Pinheiro et al., 2017). In both contributions, linear regression models were fitted to predict VAD di-

mensions given BE5 categories and vice versa. Yet, this was mainly done to inspect the respective slope-coefficients as an indicator of the relationship of dimensions and categories. Thus, the overall goodness of the fit was *not* in the center of interest and was not even reported by Stevenson et al. (2007).

3 Methods

Let $L := \{w_1, w_2, ..., w_n\}$ be a set of words. Let s,t denote two distinct *emotion representation formats* such that *both* $emo^s(w_i) \in \mathbb{R}^{|s|}$ and $emo^t(w_i) \in \mathbb{R}^{|t|}$ describe the emotion vector associated with w_i relative to s and t, respectively, where $|s|$, $|t|$ denote the number of variables which each format employs (e.g., 3 for VAD and 5 for BE5). The task we address in this paper is to predict the *target emotion ratings* $T := \{emo^t(w_i)|\ w_i \in L\}$ given the set L and the corresponding *source emotion ratings* $S := \{emo^s(w_i)|\ w_i \in L\}$. Performance will be measured as Pearson correlation r between the predicted values and human gold ratings (one r-value per element of the target representation). In general, the Pearson correlation between two data series $X := x_1, x_2, ..., x_n$ and $Y := y_1, y_2, ..., y_n$ takes values between $+1$ (perfect positive correlation) and -1 (perfect negative correlation) and is computed as

$$r_{xy} := \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\ \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{1}$$

where $\bar{x}$ and $\bar{y}$ denote the mean values for X and Y, respectively.

3.1 Reference Methods

The first method against which we will compare our proposed model is linear regression (LR) as used by Stevenson et al. (2007) in their early study. LR predicts an emotion value in the target representation t as the affine transformation

$$emo^t_{\text{LR}}(w_i) := W\ emo^s(w_i) + b \tag{2}$$

where W is a $|t| \times |s|$ matrix and b is a $|t| \times 1$ vector. The model parameters are fitted using ordinary least squares. In contrast, Buechel and Hahn (2017b) proposed the use of *k*-Nearest-Neighbor Regression (KNN) for ERM. This simple supervised approach predicts the target value as

$$emo^t_{\text{KNN}}(w_i) := \frac{1}{k} \sum_{w_i' \in \text{NEAREST}(w_i,k,S)} emo^t(w_i') \tag{3}$$

where NEAREST yields the k nearest neighbors of w_i in the training set (determined by the Euclidean distance between the source representations of two words). The k parameter was fixed to 20 based on a pilot study.[4] We used the `scikit-learn.org` implementation for both LR and KNN.

3.2 Proposed Model: A Multi-Task Feed-Forward Neural Network for ERM

Despite the fact that the above set-ups already perform quite well for ERM (see Section 4), both LR and KNN are rather basic types of models lacking deeper sophistication. As a consequence, we here propose the use of Feed-Forward Neural Networks[5] (FFNNs) for ERM which have been shown to be capable of approximating arbitrary functions, in theory at least (Hornik, 1991). In general, an FFNN consists of an *input layer* with activation $a^{(0)} := emo^s(w_i) \in \mathbb{R}^{|s|}$ followed by multiple hidden layers with activation $a^{(l+1)} := \sigma(W^{(l+1)}a^{(l)} + b^{(l+1)})$ where $W^{(l+1)}$ and $b^{(l+1)}$ are the weights and biases for layer $l+1$ and σ is a nonlinear activation function. Since the emotion formats under scrutiny capture affective states as real-valued vectors, the activation on the output layer a^{out} (where out is the number of non-input layers in the network) is computed as the affine transformation

$$emo^t_{\text{FFNN}}(w_i) := a^{(out)} := W^{(out)}a^{(out-1)} + b^{(out)} \tag{4}$$

[4] In contrast, Buechel and Hahn (2017a) determined k for each lexicon *individually* based on a dev set. Now, we deviate from this approach since it is inapplicable for the cross-lingual lexicon construction presented in Section 4.4.

[5] Note that applying neural architectures currently popular for other NLP tasks is not advisable because of the simplicity of our input data (feature vectors of length 2 to 5). These more complex architectures are instead designed for, e.g., *sequential* data (such as the RNN family) or *spatially arranged* data (such as CNNs).

Consequently, our model differs from the other approaches presented in this section by *sharing* model parameters (weights and biases of the hidden layers) across the different dimensions/categories of the target format with only the last layer having parameters which are uniquely associated to one of the outputs (see Equation 4). This can be considered as a mild form of multi-task learning (Caruana, 1997), a machine learning technique which has been shown to strongly decrease the risk of overfitting (Baxter, 1997) and also speeds up computation by greatly decreasing the number of tunable parameters compared to training individual layers for each affective dimension/category.

The remaining specifications of our model are as follows. We train two-hidden layer FFNNs (both with 128 units), ReLU activation, .2 dropout on the hidden layers (none on the input layer)[6] and Mean-Squared-Error loss. Each model was trained for $10,000$ iterations (well beyond convergence, independently of the size of the training set) using the ADAM optimizer (Kingma and Ba, 2015). `Keras.io` was used for implementation.

3.3 Baseline: Word Emotion Induction

As a natural baseline for ERM, we will use a recent state-of-the-art method for word emotion induction (WEI) by Du and Zhang (2016).[7] They propose Feed-Forward Neural Networks (similar to our proposed model for ERM) in combination with a boosting algorithm. The authors used FFNNs with a single hidden layer of 100 units and ReLU activation. The boosting algorithm ADABOOST.R2 (Drucker, 1997) was used to train the ensemble (one per target variable). We implemented this approach with `scikit-learn` using exactly the same settings as in the original publication.[8] As for the word embeddings this method needs as input, we used the pre-trained FASTTEXT embeddings that Facebook Research makes available for a wide range of languages trained on the respective Wikipedias.[9] This way, we hope to achieve a particularly high level of comparability across languages because, for each of them, embeddings are trained on data from the same domain and of a similar order of magnitude.[10]

3.4 Comparison to Human Reliability

Since common metrics for Inter-Annotator Agreement (IAA), such as Cohen's Kappa, are not applicable for real-valued emotion scores (Carletta, 1996), we will now discuss how to compare our own results against human assessments in order to put their reliability on a safe ground.

One possible point of comparison that has been used in previous work (Buechel and Hahn, 2017a; Buechel and Hahn, 2018a) is *inter-study reliability* (ISR), i.e., the correlation between the ratings of common words in different data sets. However, this procedure comes with a number of downsides. First, the number of pairs of data sets with substantially overlapping entries is rather small since researchers focus mainly on acquiring ratings for *novel* words instead of gathering annotations anew for ones already covered. Thus, employing ISR comparison with human performance is only possible on few data sets. In particular, we are not aware of any pair of data sets with significantly overlapping BE5 ratings. Second, ISR is sensitive to differences in acquisition methodologies (e.g., alternative sets of instructions or rating scales) and may thus vary substantially between different pairs of data sets.

As an alternative, these shortcomings lead us to propose *split-half reliability* (SHR) as a new basis for our comparison. SHR is computed by splitting all individual ratings for each of the items into two groups. These individual ratings are then averaged for both groups and the Pearson correlation between the group averages is computed. The whole processes is repeated (typically 100 times) with random splits before averaging the results from each iteration (Mohammad and Bravo-Marquez, 2017). Thus, an

[6] We found the usual recommendation of .2 on input and .5 on hidden layers (Srivastava et al., 2014) too high given the small number of features in our task (2 to 5).

[7] In our most recent contribution featuring a large-scale evaluation of many current WEI approaches on numerous data sets, we found that among the existing ones the model proposed by Du and Zhang (2016) performs best, only beaten by our own, newly proposed model (Buechel and Hahn, 2018b). Note that even compared to this more advanced approach to WEI, the performance figures we report here for ERM still remain much higher (see Section 4). Hence, the claim of this paper that ERM is superior to WEI, remains valid even despite most recent achievements for the latter task.

[8] Publicly available at: `https://github.com/StevenLOL/ialp2016_Shared_Task`

[9] `https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md`

[10] For English, much larger embedding models are publicly available, yet not for the other languages under consideration; cf. Buechel and Hahn (2018b).

important difference between SHR and ISR is that the former is computed on a *single* data set whereas the latter requires two *different* data sets with overlapping items. On the other hand, ISR can be computed on the final ratings alone, whereas SHR requires knowledge of the judgments of the individual raters. Most often, these individual ratings are not distributed. Yet, luckily, SHR values are commonly reported when publishing emotion lexicons (see below).

Still, both SHR and ISR—as well as other popular approaches to reliability estimation for numerical emotion scores, e.g., the leave-one-out approach presented by Strapparava and Mihalcea (2007)—are heavily influenced by the number of participants of a study. For SHR, this is intuitively clear because with enough subjects, both groups should yield reliable estimates of the true population mean ratings, leading to very high correlation values between the groups. As a result, by splitting the number of raters into two groups for the SHR estimate, this technique will on average produce lower correlation values than if the study was repeated with the full number of participants and correlation between the first and second study had been computed (test-retest reliability). To counterbalance this effect, when reporting SHR values, authors often turn to *Spearman-Brown adjustment* (SBA; Vet et al. (2017)), a technique which estimates the reliability r^* of a study if the number of subjects was increased by the factor k:

$$r^* := \frac{k\, r}{1 + (k-1)\, r} \tag{5}$$

were r is the *empirically measured* SHR and k is set to 2 for the use case discussed above (virtually doubling the number participants).

Since some authors of the data sets in Table 1 apply SBA while others do not, the reported SHR values must be normalized to guarantee a consistent evaluation. Going one step further, we can even apply SBA to normalize the reported values with respect to the number of participants in a given study, thus establishing an even more consistent ground for evaluation.

We chose the *normalized number of participants* to be 20, i.e., the adjusted scores (reported in Table 3) estimate the *empirical* SHR values, if the given study was conducted with 20 participants (the average correlation between two randomly assigned groups of 10 raters). Normalization was conducted by applying Equation (5) to the reported values with $k := N^*/N$, if SBA was not already applied, or $k := N^*/(2 \times N)$, if SBA was already applied to the reported values; N being the actual number of participants and $N^* := 20$ being the normalized number of participants.

	Val	Aro	Dom	Joy	Ang	Sad	Fea	Dsg
en_1	—	—	—	—	—	—	—	—
en_2	.914	.689	.770	—	—	—	—	—
es_1	—	—	—	.915	.889	.915	.889	.864
es_2	.839	.730	.730	.915	.915	.915	.889	.889
es_3	.880	.750	—	.754	.786	.818	.802	.739
de_1	—	—	—	—	—	—	—	—
pl_1	.928	.630	—	.884	.802	.821	.821	.802
pl_2	.935	.679	.725	.884	.802	.821	.821	.802

Table 3: Normalized split-half reliabilities for VAD and BE5 for the data sets used in our experiments. "—" indicates that reliability has not been reported.

It is important to note that the decision for $N^* = 20$ is necessarily arbitrary, to some degree, with higher SHR estimates arising from higher values of N^*. However, 20 raters are often used in psychological studies (Warriner et al., 2013; Stadthagen-Gonzalez et al., 2017b), while being way higher than the number of raters typically used in NLP for emotion annotation, both for the word and sentence level (Yu et al., 2016a; Strapparava and Mihalcea, 2007). Thus, we argue that this choice constitutes a rather challenging line of comparison for our system.

Since model performance will be measured in terms of Pearson correlation (see above), the performance figures achieved on the gold data can be compared with the adjusted SHR (also based on correlation). We can interpret cases where the former outperforms the latter as *the model agreeing more with the gold data than two random groups of ten annotators would agree with each other*. Thus, for these cases we say our model achieves *super-human* performance, as it cannot be expected that a well-conducted annotation study leads to more reliable results.

4 Results

4.1 Ablation Experiments on Affective Dimensions and Categories

Previous work has limited itself to data sets comprising all three VAD dimensions with the implicit belief that Dominance provides valuable affective information which is important for ERM. However, since only about half of the data sets developed in psychology labs (and even less provided by NLP groups) actually *do* comprise Dominance, this decision massively decreases the amount of data sets at hand. To resolve this dilemma, the following experiment aims at quantifying the relative importance of the different affective variables of the VAD and the BE5 format.

Our set-up works as follows: For each data set from Table 1 that includes the Dominance dimension, we trained one LR model[11] (Section 3.1) to map VAD to BE5 and another one to map BE5 to VAD ('dim2cat' and 'cat2dim' for short) applying 10-fold cross-validation. The resulting performance measurements were averaged over all data sets.

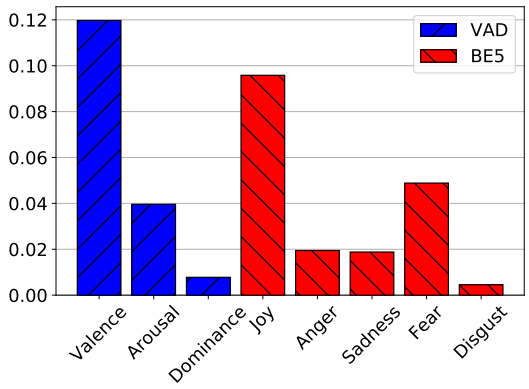


Figure 2: Relative importance of the affective variables of VAD and BE5 for predicting the alternative format, respectively; measured in drop of Pearson r when using all variables vs. omitting the one under scrutiny.

We then repeated this procedure once for each VAD dimension (when mapping dim2cat) and each BE5 category (when mapping cat2dim), omitting one of the dimensions/categories from the source representation in every iteration, thus constituting a kind of ablation experiment. Next, for each of the "incomplete" models, we computed the difference between its performance and the performance of the "complete" model (not lacking any of the variables). Now, we can use this loss of performance as an estimate of the *relative importance* of the respective left-out dimension or category. The results of this experiment are depicted in Figure 2.

As can be seen, regarding VAD, Valence is by far the most important dimension with a performance drop of .12 when ablating it. In turn, Arousal, the second-best dimension only increases performance by .04, whereas Dominance contributes to less than .01 of the performance. Similarly, for Basic Emotions, Joy is the most important category, although BE5 seems to distribute the affective information more equally across its variables (with the exception of Disgust which contributes far less than .01 to the performance).

Since our data suggest that Dominance plays only a minor role within the VAD framework, we will *not* limit our further experiments to data sets including this dimension—as it was done in previous work (Section 2)—but rather include the large variety of bi-representational data sets which leave it out (see Table 1).

4.2 Monolingual Representation Mapping

In this experiment, we compared the performance of the WEI baseline, the LR- and KNN-based reference methods for ERM and our newly proposed FFNN model. For each of these methods and data sets in Table 1, we trained one model to map cat2dim and another one to map dim2cat (for the ERM methods) or to predict VA(D) ratings and BE5 ratings based on word embeddings for the WEI baseline. The whole process was conducted using 10-fold cross-validation where we used identical train/test splits for all methods.[12] The results of this experiment are displayed in Table 4a, only showing the average values over VA(D) and BE5, respectively, but allowing for an easy comparison between the different approaches.

[11] Linear regression was used because it does not comprise any hyperparameters that might heavily influence the outcome of this experiment (thus leading to greater generality of the results).

[12] This procedure constitutes a more direct comparison than using different splits for each method and allows *paired* t-tests.

As can be seen, all of the ERM approaches (LR, KNN, FFNN) perform more than 10%-points better than the state of the art in word emotion induction (WEI) for VAD prediction and at least about 5%-points better for BE5 predictions (on average over all data sets and affective variables). This finding already strongly suggests that ERM is the superior approach for automatic lexicon creation, given that the required data are available. This might be especially useful in situations where, say, large VAD but only small BE5 lexicons are available for a given language (see Section 4.4). Regarding the ordering of the ERM approaches, KNN outperforms LR in almost all cases. The advantage is more pronounced for mapping dim2cat (2.5%-points difference on average) than cat2dim (.4%-points difference). On top of that, our proposed FFNN model outperforms KNN by a 1.2%-point margin for cat2dim and a .8%-point margin for dim2cat (again as average over all data sets) performing best on each single data set. Regarding the 16 cases of Table 4a (8 data sets times two mapping directions), the performance gain of FFNN compared to the respective second best system is statistically significant[13] in all but 2 cases. The differences between the individual ERM approaches might appear quite small, yet become a lot more meaningful considering the proximity to human annotation capabilities as discussed in the following paragraphs.

	cat2dim				dim2cat			
	WEI	LR	KNN	FFNN	WEI	LR	KNN	FFNN
en_1	.685	.841	.840	**.853****	.818	.844	.868	**.877***
en_2	.741	.827	.828	**.843*****	.821	.829	.852	**.858*****
es_1	.709	.856	.855	**.869*****	.775	.804	.849	**.853**
es_2	.600	.823	.828	**.844*****	.797	.863	.882	**.889***
es_3	.713	.799	.796	**.804*****	.743	.776	.820	**.826*****
de_1	.758	.819	.827	**.837****	.701	.669	.698	**.712**
pl_1	.681	.858	.870	**.875****	.707	.844	.848	**.855*****
pl_2	.619	.803	.814	**.825****	.697	.820	.834	**.839****
Avg.	.688	.828	.832	**.844**	.757	.806	.831	**.839**

(a) Results of the monolingual experiment for the WEI baseline, two reference methods (LR and KNN) as well as our FFNN model in Pearson r. Best result per data set and emotion format in bold, second best result underlined; significant difference (paired two-tailed t-test) over the second best system marked with "*", "**", or "***" for $p < .05$, .01, or .001, respectively.

	Val	Aro	Dom	Joy	Ang	Sad	Fea	Dsg
en_1	.969	.741	.848	.962	.876	.871	.873	.805
en_2	.964	.704	.861	.942	.868	.821	.860	.799
es_1	.974	.771	.863	.957	.854	.833	.869	.752
es_2	.986	.828	.720	.977	.913	.867	.878	.807
es_3	.915	.692	—	.846	.839	.857	.842	.744
de_1	.929	.745	—	.894	.778	.644	.785	.461
pl_1	.963	.787	—	.946	.872	.826	.805	.826
pl_2	.947	.768	.760	.935	.844	.805	.790	.819
Avg.	.956	.754	.810	.932	.855	.816	.838	.752

(b) Results of the monolingual experiment per affective dimension in Pearson r. Color indicates outperforming human SHR (blue), being outperformed (red) or SHR not being reported (white; "—" meaning that the respective variable is not included).

Table 4: Results of the monolingual experiment.

Table 4b displays the performance figures of the FFNN model relative to each affective variable. As can be seen, among VAD, Valence is the easiest dimension to predict ($r = .956$ on average over all data sets) whereas for Arousal the performance is worst . Similarly, for BE5, Joy obtains the best values ($r = .932$) and Disgust is the hardest to predict. Interestingly, the overall ordering of performance within the two formats is consistent with the ordering of human reliability (see Table 3).

Comparing our system performance against human SHR (based on 20 participants per study; see Section 3.4), again our approach seems to be highly reliable (color coding of Table 4b). In particular, ERM using the FFNN model outperforms SHR in over half of the applicable cases (25 of 38). For mapping cat2dim it surpasses human reliability in all but 2 cases whereas when mapping dim2cat the reported SHR is surpassed in over half of the cases (14 out of 25).

This result, astonishing as it might appear, is yet consistent with findings from previous work which, in turn, were based on ISR (not on SHR) data (Buechel and Hahn, 2017a; Buechel and Hahn, 2018a). We conclude that in the monolingual set-up, ERM using the FFNN model substantially outperforms current capacities in word emotion induction and is even more reliable than a medium sized human rating study. Thus these automatically produced ratings should be cautiously attributed gold standard quality.

[13] Paired two-tailed t-tests based on the 10 train/test splits during cross-validation; $p < .05$.

4.3 Crosslingual Representation Mapping

In the crosslingual set-up, we make use of the fact that our model does not rely on any language-specific information, since the categories/dimensions describe supposedly universal affective states rather than linguistic entities. Thus, models trained on one language could, in theory, be applied to another one without any need for adaptation. This capability comes in handy when only data sets according to *one* emotion format exist for a given language. In such cases we could still train our model on data available for other languages and use it to produce new ratings for the language in focus. This section aims at estimating the performance of lexicons derived in this manner.

	Val	Aro	Joy	Ang	Sad	Fea	Dsg
en_1	.966	.683	.955	.858	.838	.817	.781
en_2	.956	.642	.934	.855	.810	.791	.800
es_1	.973	.692	.951	.786	.802	.782	.682
es_2	.985	.735	.974	.881	.860	.835	.787
es_3	.908	.548	.839	.821	.850	.807	.728
de_1	.927	.708	.889	.767	.618	.760	.458
pl_1	.957	.666	.937	.848	.784	.745	.801
pl_2	.938	.720	.932	.816	.785	.751	.809
Avg.	.951	.674	.926	.829	.793	.786	.731

Table 5: Results of crosslingual experiment in Pearson r. Color indicates outperforming human SHR (blue), being outperformed (red) or SHR not being reported (white).

For each of the data sets in Table 1, we trained FFNN models to map cat2dim and dim2cat, respectively. We trained on each gold lexicon that did not cover the language of the data set under scrutiny (e.g., for testing on en_1, the models were trained on all Spanish, Polish and German data sets, but not on en_2). Since this set-up leads to fixed train and test sets, we did not perform cross-validation. For comparability between data sets, the Dominance dimension was excluded for this experiment.

Overall, the results remained astonishingly stable compared to the monolingual set-up, with performance figures for Valence and Joy dropping by less than 1%-point on average over all data sets (see Table 5). Also, Anger, Sadness, Fear and Disgust only suffer a moderate decrease of about 5%-points at most—only the performance of Arousal decreased more than that.

A possible explanation for these strong results is the marked increase in the amount of training data that comes along with training on the majority of the available data (independent of language). This circumstance seems to counterbalance much of the negative effects that may arise in this crosslingual applications.

In comparison to SHR, the ERM approach still turns out to work quite well. Regarding VA, we outperform human reliability in 8 of 10 cases. Concerning BE5, SHR was beaten in about half of the cases (11 of 25). We conclude that, although the capability of our mapping approach suffers a bit in the crosslingual set-up, it still produces very accurate predictions and can thus be attested *near* gold quality, at least.

4.4 Automatic Lexicon Construction for Diverse Languages

After the positive evaluation of the FFNN model for ERM, the last bit of our contributions is to apply the created models to a wide variety of data sets which so far bear emotion ratings for *one* format only (either VA(D) or BE5). Based on the experiments reported so far, we claim that these have gold quality (for the monolingual approach, Section 4.2) or near-gold quality (for the crosslingual approach, Section 4.3).

For the monolingual approach, we train our model on the data set on which we achieved the highest performance in Section 4.2 for the respective language (assuming this hints at particularly "clean" data). In contrast, in the crosslingual set-up, training data are acquired by concatenating *all* the available data sets from Table 1 (consequently ignoring Dominance for compatibility).

Table 6 lists the emotion lexicons constructed in this manner together with their most important characteristics. The number of new ratings ranges from almost 13,000 (for English) and 10,500 (for Spanish), over several thousands (for Dutch, Chinese and Polish,) and around 1,500–1,000 (for Indonesian, Italian, Portuguese, Greek, French and German) to 200–100 (for Finnish and Swedish). For illustration, Table 2 displays three entries of the English BE5 lexicon, the largest one we constructed.

5 Conclusion

In this paper, we addressed the relatively new task of *emotion representation mapping*. It aims at transforming emotion ratings for lexical units from one emotion representation format into another one, e.g., mapping from Valence-Arousal-Dominance representations to Basic Emotion ones. Based on a large-scale evaluation we gathered solid empirical evidence that the proposed neural network model consistently outperforms the previous state-of-the-art performance figures in both word emotion induction and emotion representation mapping. Hence, the approach we propose currently constitutes the best-performing method for automatic emotion lexicon creation.

Mth	Lng	Format	Source	#Words
m	en	BE5	Warriner et al. (2013)	12,884
m	es	VAD	Stadthagen-González et al. (2017a)	10,489
m	de	BE5	Võ et al. (2009)	944
m	pl	BE5	Imbir (2016)	3,633
c	it	BE5	Montefinese et al. (2014)	1,121
c	pt	BE5	Soares et al. (2012)	1,034
c	nl	BE5	Moors et al. (2013)	4,299
c	id	BE5	Sianipar et al. (2016)	1,487
c	zh	BE5	Yu et al. (2016a); Yao et al. (2017)	3,797
c	fr	BE5	Monnier and Syssau (2014)	1,031
c	gr	BE5	Palogiannidi et al. (2016)	1,034
c	fn	BE5	Eilola and Havelka (2010)	210
c	sv	BE5	Davidson and Innes-Ker (2014)	99

Table 6: Overview of automatically constructed emotion lexicons; mapping methodology (monolingual or crosslingual), language (codes according to ISO 639-1), target emotion format, source lexicon of the mapping process and number of previously unknown ratings (excluding those present in other lexicons).

We also proposed a novel methodology for comparison against human rating capabilities based on normalized split-half reliability scores. For the first time, this allows for a large-scale evaluation against human performance. Our experimental data suggest that our models perform competitive relative to human assessments, even in cross-lingual applications, thus producing (near) gold quality data. We take this as a strong hint towards the reliability of the methods we propose.

Finally, we used these models to produce new emotion lexicons for 13 typologically diverse languages which are publicly available along with our code and experimental data (see Footnote 1).

Acknowledgements

We thank the anonymous reviewers for their thoughtful comments and suggestions.

References

Jonathan Baxter. 1997. A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28(1):7–39.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Margaret M. Bradley and Peter J. Lang. 1999. Affective Norms for English Words (ANEW): Stimuli, instruction manual and affective ratings. Technical Report C-1, The Center for Research in Psychophysiology, University of Florida, Gainesville, Florida, USA.

Benny B. Briesemeister, Lars Kuchinke, and Arthur M. Jacobs. 2011. Discrete Emotion Norms for Nouns: Berlin Affective Word List (DENN–BAWL). *Behavior Research Methods*, 43(2):441.

Sven Buechel and Udo Hahn. 2016. Emotion analysis as a regression problem: Dimensional models and their implications on emotion representation and metrical evaluation. In *ECAI 2016 — Proceedings of the 22nd European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 1114–1122, The Hague, The Netherlands, August 29 – September 2, 2016.

Sven Buechel and Udo Hahn. 2017a. A flexible mapping scheme for discrete and dimensional emotion representations: Evidence from textual stimuli. In *CogSci 2017 — Proceedings of the 39th Annual Meeting of the Cognitive Science Society*, pages 180–185, London, UK, July 26–29, 2017.

Sven Buechel and Udo Hahn. 2017b. EMOBANK: Studying the impact of annotation perspective and representation format on dimensional emotion analysis. In *EACL 2017 — Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, volume 2, short papers, pages 578–585, Valencia, Spain, April 3–7, 2017.

Sven Buechel and Udo Hahn. 2018a. Representation mapping: A novel approach to generate high-quality multilingual emotion lexicons. In *LREC 2018 — Proceedings of the 11th International Conference on Language Resources and Evaluation*, pages 184–191, Miyazaki, Japan, May 7–12, 2018.

Sven Buechel and Udo Hahn. 2018b. Word emotion induction for multiple languages as a deep multi-task learning problem. In *NAACL-HLT 2018 — Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, long papers, pages 1907–1918, New Orleans, Louisiana, USA, June 1–6, 2018.

Rafael A. Calvo and Sunghwan Mac Kim. 2013. Emotions in text: Dimensional and categorical models. *Computational Intelligence*, 29(3):527–543.

Jean C. Carletta. 1996. Assessing agreement on classification tasks: The Kappa statistic. *Computational Linguistics*, 22(2):249–254.

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

Per Davidson and Åse Innes-Ker. 2014. Valence and arousal norms for Swedish affective words. Technical Report Volume 14, No. 2, Lund University, Lund, Sweden.

Harris Drucker. 1997. Improving regressors using boosting techniques. In *ICML 1997 — Proceedings of the 14th International Conference on Machine Learning*, pages 107–115, Nashville, Tennessee, USA, July 8–12, 1997.

Steven Du and Xi Zhang. 2016. Aicyber's system for IALP 2016 Shared Task: Character-enhanced word vectors and boosted neural networks. In *IALP 2016 — Proceedings of the 2016 International Conference on Asian Language Processing*, pages 161–163, Tainan, Taiwan, November 21–23, 2016.

Tiina M. Eilola and Jelena Havelka. 2010. Affective norms for 210 British English and Finnish nouns. *Behavior Research Methods*, 42(1):134–140.

Paul Ekman. 1992. An argument for basic emotions. *Cognition & Emotion*, 6(3-4):169–200.

Andrea Esuli and Fabrizio Sebastiani. 2005. Determining the semantic orientation of terms through gloss classification. In *CIKM 2005 — Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 617–624, Bremen, Germany, October 31 – November 05, 2005.

Pilar Ferré, Marc Guasch, Natalia Martínez-García, Isabel Fraga, and José Antonio Hinojosa. 2017. Moved by words: Affective ratings for a set of 2,266 Spanish words in five discrete emotion categories. *Behavior Research Methods*, 49(3):1082–1094.

Vasileios Hatzivassiloglou and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *ACL-EACL 1997 — Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics & 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 174–181, Madrid, Spain, July 7–12, 1997.

José Antonio Hinojosa, Natalia Martínez-García, Cristina Villalba-García, Uxia Fernández-Folgueiras, Alberto Sánchez-Carmona, Miguel Angel Pozo, and Pedro R. Montoro. 2016a. Affective norms of 875 Spanish words for five discrete emotional categories and two emotional dimensions. *Behavior Research Methods*, 48(1):272–284.

José Antonio Hinojosa, Irene Rincón-Pérez, M. Verónica Romero-Ferreiro, Natalia Martínez-García, Cristina Villalba-García, Pedro R. Montoro, and Miguel Angel Pozo. 2016b. The Madrid Affective Database for Spanish (MADS): Ratings of dominance, familiarity, subjective age of acquisition and sensory experience. *PLoS One*, 11(5):e0155866.

Kurt Hornik. 1991. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251 – 257.

Kamil K. Imbir. 2016. Affective Norms for 4900 Polish Words Reload (ANPW_R): Assessments for valence, arousal, dominance, origin, significance, concreteness, imageability and, age of acquisition. *Frontiers in Psychology*, 7:#1081.

Diederik Kingma and Jimmy Ba. 2015. ADAM: A method for stochastic optimization. In *ICLR 2015 — Proceedings of the 3rd International Conference on Learning Representations*, pages 1–15, San Diego, California, USA, May 7–9, 2015.

Maximilian Köper and Sabine Schulte im Walde. 2016. Automatically generated affective norms of abstractness, arousal, imageability and valence for 350,000 German lemmas. In *LREC 2016 — Proceedings of the 10th International Conference on Language Resources and Evaluation*, pages 2595–2598, Portorož, Slovenia, May 23–28, 2016.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS 2013 — Proceedings of the 27th Annual Conference on Neural Information Processing Systems*, pages 3111–3119, Lake Tahoe, Nevada, USA, December 5–10, 2013.

Saif M. Mohammad and Felipe Bravo-Marquez. 2017. Emotion intensities in tweets. In **SEM 2017 — Proceedings of the 6th Joint Conference on Lexical and Computational Semantics*, pages 65–77, Vancouver, British Columbia, Canada, August 3–4, 2017.

Catherine Monnier and Arielle Syssau. 2014. Affective norms for French words (FAN). *Behavior Research Methods*, 46(4):1128–1137.

Maria Montefinese, Ettore Ambrosini, Beth Fairfield, and Nicola Mammarella. 2014. The adaptation of the Affective Norms for English Words (ANEW) for Italian. *Behavior Research Methods*, 46(3):887–903.

Agnes Moors, Jan De Houwer, Dirk Hermans, Sabine Wanmaker, Kevin van Schie, Anne-Laura Van Harmelen, Maarten De Schryver, Jeffrey De Winne, and Marc Brysbært. 2013. Norms of valence, arousal, dominance, and age of acquisition for 4,300 Dutch words. *Behavior Research Methods*, 45(1):169–177.

Elisavet Palogiannidi, Polychronis Koutsakis, Elias Iosif, and Alexandros Potamianos. 2016. Affective lexicon creation for the Greek language. In *LREC 2016 — Proceedings of the 10th International Conference on Language Resources and Evaluation*, pages 2867–2872, Portorož, Slovenia, 23–28 May 2016.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP 2014 — Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar, October 25–29, 2014.

Ana P. Pinheiro, Marcelo Dias, João Pedrosa, and Ana P. Soares. 2017. Minho Affective Sentences (MAS): Probing the roles of sex, mood, and empathy in affective ratings of verbal stimuli. *Behavior Research Methods*, 49(2):698–716.

Jaime Redondo, Isabel Fraga, Isabel Padrón, and Montserrat Comesaña. 2007. The Spanish adaptation of ANEW (Affective Norms for English Words). *Behavior Research Methods*, 39(3):600–605.

Monika Riegel, Małgorzata Wierzba, Marek Wypych, Łukasz Żurawski, Katarzyna Jednoróg, Anna Grabowska, and Artur Marchewka. 2015. Nencki Affective Word List (NAWL): The cultural adaptation of the Berlin Affective Word List–Reloaded (BAWL–R) for Polish. *Behavior Research Methods*, 47(4):1222–1236.

Sara Rosenthal, Preslav I. Nakov, Svetlana Kiritchenko, Saif M. Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. SemEval 2015 Task 10: Sentiment analysis in Twitter. In *SemEval 2015 — Proceedings of the 9th International Workshop on Semantic Evaluation @ NAACL-HLT 2015*, pages 451–463, Denver, Colorado, USA, June 4–5, 2015.

Samira Shaikh, Kit Cho, Tomek Strzalkowski, Laurie Feldman, John Lien, Ting Liu, and George Aaron Broadwell. 2016. ANEW+: Automatic expansion and validation of affective norms of words lexicons in multiple languages. In *LREC 2016 — Proceedings of the 10th International Conference on Language Resources and Evaluation*, pages 1127–1132, Portorož, Slovenia, 23–28 May 2016.

Agnes Sianipar, Pieter van Groenestijn, and Ton Dijkstra. 2016. Affective meaning, concreteness, and subjective frequency norms for Indonesian words. *Frontiers in Psychology*, 7:#1907.

Ana Paula Soares, Montserrat Comesaña, Ana P. Pinheiro, Alberto Simões, and Carla Sofia Frade. 2012. The adaptation of the Affective Norms for English Words (ANEW) for European Portuguese. *Behavior Research Methods*, 44(1):256–269.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Hans Stadthagen-González, Pilar Ferré, Miguel A. Pérez-Sánchez, Constance Imbault, and José Antonio Hinojosa. 2017a. Norms for 10,491 Spanish words for five discrete emotions: Happiness, disgust, anger, fear, and sadness. *Behavior Research Methods*. Available: `https://doi.org/10.3758/s13428-017-0962-y`.

Hans Stadthagen-Gonzalez, Constance Imbault, Miguel A. Pérez-Sánchez, and Marc Brysbært. 2017b. Norms of valence and arousal for 14,031 Spanish words. *Behavior Research Methods*, 49(1):111–123.

Ryan A. Stevenson, Joseph A. Mikels, and Thomas W. James. 2007. Characterization of the Affective Norms for English Words by discrete emotional categories. *Behavior Research Methods*, 39(4):1020–1024.

Carlo Strapparava and Rada Mihalcea. 2007. SemEval 2007 Task 14: Affective text. In *SemEval 2007 — Proceedings of the 4th International Workshop on Semantic Evaluations @ ACL 2007*, pages 70–74, Prague, Czech Republic, June 23–24, 2007.

Peter D. Turney and Michael L. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems*, 21(4):315–346.

Melissa L. H. Võ, Markus Conrad, Lars Kuchinke, Karolina Urton, Markus J. Hofmann, and Arthur M. Jacobs. 2009. The Berlin Affective Word List Reloaded (BAWL–R). *Behavior Research Methods*, 41(2):534–538.

Henrica C. W. de Vet, Lidwine B. Mokkink, David G. Mosmuller, and Caroline B. Terwee. 2017. Spearman-Brown prophecy formula and Cronbach's alpha: Different faces of reliability and opportunities for new applications. *Journal of Clinical Epidemiology*, 85:45–49.

Amy Beth Warriner, Victor Kuperman, and Marc Brysbært. 2013. Norms of valence, arousal, and dominance for 13,915 English lemmas. *Behavior Research Methods*, 45(4):1191–1207.

Małgorzata Wierzba, Monika Riegel, Marek Wypych, Katarzyna Jednoróg, Paweł Turnau, Anna Grabowska, and Artur Marchewka. 2015. Basic emotions in the Nencki Affective Word List (NAWL BE): New method of classifying emotional stimuli. *PLoS One*, 10(7):e0132305.

Zhao Yao, Jia Wu, Yanyan Zhang, and Zhenhong Wang. 2017. Norms of valence, arousal, concreteness, familiarity, imageability, and context availability for 1,100 Chinese words. *Behavior Research Methods*, 49(4):1374–1385.

Liang-Chih Yu, Lung-Hao Lee, Shuai Hao, Jin Wang, Yunchao He, Jun Hu, K. Robert Lai, and Xuejie Zhang. 2016a. Building Chinese affective resources in valence-arousal dimensions. In *NAACL-HLT 2016 — Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 540–545, San Diego, California, USA, June 12–17, 2016.

Liang-Chih Yu, Lung-Hao Lee, and Kam-Fai Wong. 2016b. Overview of the IALP 2016 Shared Task on dimensional sentiment analysis for Chinese words. In *IALP 2016 — Proceedings of the 2016 International Conference on Asian Language Processing*, pages 156–160, Tainan, Taiwan, November 21–23, 2016.

Emotion Detection and Classification in a Multigenre Corpus with Joint Multi-Task Deep Learning

Shabnam Tafreshi
George Washington University
Department of Computer Science
GWU NLP Lab
shabnamt@gwu.edu

Mona Diab
George Washington University
Department of Computer Science
GWU NLP Lab
mtdiab@gwu.edu

Abstract

Detection and classification of emotion categories expressed by a sentence is a challenging task, due to subjectivity of emotion. To date, most of the models are trained and evaluated on single genre and when used to predict emotion in different genre, their performance drops by a large margin. To address the issue of robustness, we model the problem within a joint multi-task learning framework. We train this model with a multigenre emotion corpus to predict emotions across various genres. Each genre is represented as a separate task, we use soft parameter shared layers across the various tasks. our experimental results show that this model improves the results across the various genres, compared to a single genre training in the same neural net architecture.

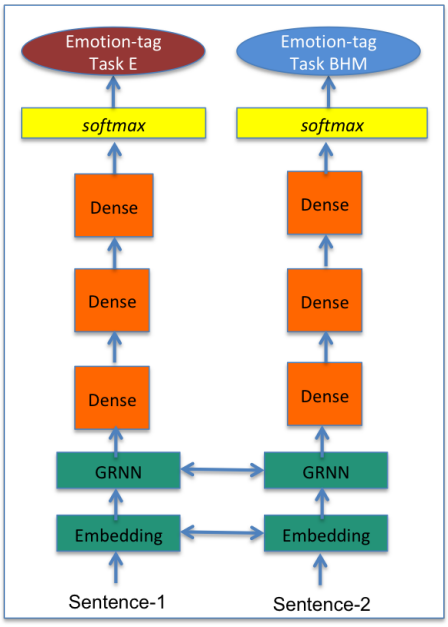


Figure 1: Joint Multi-Task Emotion neural net architecture. In this multi-task framework, Task-E and Task-BHM (explained in section 3) demonstrate different genres.

1 Introduction

Sentence-level emotion detection is garnering a lot of attention recently. To date, most systems are trained and evaluated on a single genre resource. However, the problem is robustness, when such models are applied to new genres, the performance expectedly drops significantly. In this paper, we propose a model to address the genre robustness issue.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2905–2913
Santa Fe, New Mexico, USA, August 20-26, 2018.

Dataset	joy	trust	anti	surprise	sad	fear	anger	disgust
BLG	689	43	260	150	312	132	192	255
HLN	106	6	56	31	83	68	28	55
MOV	4875	26	119	255	258	63	20	5145
EmoNet	5670	75	435	436	653	263	240	5455
BHMT	11340	150	870	872	1306	526	480	10910

Table 1: Data statistics illustrating the distributions of the various emotion tags from PL8 across the different genres.

We frame the problem as a joint multi task learning problem, where each of the genres is represented as a separate task that shares information with other tasks (genres). Our model, the Joint Multi-Task Emotion (JMTE), illustrated in figure 1, trains on emotions in four genres of data: Blog Posts, News Headlines, Movie Reviews, and Tweets. Experimental results show that using JMTE achieves better results, over a single model trained on a single of these different genres. We also study the impact of specifically adding Twitter data that is distantly supervised to the training models.
The rest of this paper is organized as follows: We describe our multigenre corpus in section 2; We present our devised JMTE model in section 3; Experimental conditions and results are presented in section 4; Discussion is presented in section 5; We review related studies in section 6; Conclusions are described in section 7.

2 Data

We create a unified multigenre data set annotated on the sentence and clause level using the 8 emotions from Plutchik (Plutchik, 1962), which includes the following 8 emotions: *joy, trust, anticipation, surprise, anger, fear, sadness, disgust* (PL8) and a *no-emotion* category (Tafreshi and Diab, 2018). [1]
Our combined multigenre corpus includes the following data sets: The emotional blog post (BLG) (Saima and Stan, 2007) comprising 4,115 sentences; The headlines dataset (HLN) (Carlo and Rada, 2007) comprising 1,250 sentences; a movie review dataset (MOV) (Bo and Lillian, 2005) where people express their opinions about movies, sound tracks, and casts. The MOV dataset contains 11,855 sentences. Both BLG and HLN were originally annotated using the 6 basic emotion categories from (Paul, 1992), *happiness, sadness, fear, anger, surprise and disgust* (EK6), while the MOV data set was annotated for sentiment and sentiment intensity. The three corpora resulted in 17,220 sentences annotated with the PL8 emotion tag set, from which 3993 sentences are annotated with *no-emotion*. That corpora annotated using the crowd sourcing platform CrowdFlower[2]. The inter-annotator agreement (IAA) achieved was 79.95%. We refer to this data set as BHM corresponding to BLG, HLN, and MOV, respectively.
We further experiment with a fourth dataset from Twitter. We added 13,227 randomly selected tweets (to match the BHM collection size) from the EmoNet (Muhammad and Ungar, 2017) tweet collection, which has a total of 547,555 tweets tagged with PL8 and its extension into 16 fine grained emotion tags. EmoNet is labeled using distant supervision, namely relying on hashtag information to render the emotion tag. We only selected for our corpus tweets that had emotion tags corresponding to the PL8 emotions. This collection has no *no-emotion* tag. We refer to this data set as TweetEN. Accordingly, our annotated multigenre corpus includes: BLG, HLN, MOV, and TweetEN, comprising a total of 26,454 annotated sentences with PL8 labels. We refer to this combined corpus as BHMT. Table-1 shows data statistics.

3 Proposed Approach

We model the problem as a joint multitask learning architecture. We use a Gated Recurrent Neural Network architecture (GRU), inspired by (Muhammad and Ungar, 2017). We create two tasks in JMTE,

[1] You can download BHM dataset and JMTE python code from `https://github.com/shabnamt/jointMultitaskEmo`

[2] `https://www.crowdflower.com`

one trained on the TweetEN data set (Task-E) and the other trained on the BHM data (Task-BHM). The reason to model them separately is that they are annotated in very different ways: TweetEN is annotated using distant supervision relying on hashtags, vs. BHM which is annotated completely manually. We balance the distribution of labeled data per emotion across the two tasks.

Recurrent Neural Network (RNN)- has been widely used in the literature to model sequential problems. RNN applies the same set of weights recursively as follow:

$$h_t = f(W_{x_t} + Uh_{t-1} + b) \tag{1}$$

RNN input vector $x_t \in R^n$ at time step t is calculated based on a hidden state and an input from the current state based on Eq. 1. The function f is a nonlinearity such as tanh or ReLU. Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Neural Nets- (Cho, 2014; Chung, 2015) are implementations of RNNs that circumvent some of the major issues with RNNs as they are much better at capturing long-term dependencies, and dealing with the vanishing gradient problem (Bengio et al., 1994; Pascanu et al., 2013).

Gated Recurrent Neural Nets- (Cho, 2014; Chung, 2015) is very similar to LSTM with the following equations:

$$r_t = \sigma(W^r_{x_t} + U^r h_{t-1} + b^r) \tag{2}$$

$$z_t = \sigma(W^z_{x_t} + U^z h_{t-1} + b^z) \tag{3}$$

$$\hat{h_t} = \tanh(W_{x_t} + r_t \times U^{\hat{h}} h_{t-1} + b^{\hat{h}}) \tag{4}$$

$$h_t = z_t \times h_{t-1} + (1 - z_t) \times \hat{h_t} \tag{5}$$

GRU has two gates, a reset gate r_t, and an update gate z_t. Intuitively, the reset gate determines how to combine the new input with the previous memory, and the update gate defines how much of the previous memory to keep around. We use Keras[3] GRNN implementation to setup our experiments. We note that GRU units are a concatenation of GRU layers in each task.

Shared layers - these two tasks, Task-E and Task-BHM share the word embedding (*i.e.* Keras embedding) layer. We setup this layer in two different ways: a) initiating random weights for each word vector and tune these weights using the training set, b) initiating the weights from pre-trained word embedding model and let the embedding layer to tune the weights using the training set. The task specific GRU layers share soft parameters between each of the two layers, we observed that soft parameter sharing creates better weights for this layer. Shared GRU takes advantage of the underlying emotion cues structure shared among the genres (*e.g.* presence of adjective and adverbs) as well as semantic and syntactic emotion features that improve emotion detection and classification in different genres.

Task Specific Layers - each task has 3 hidden dense layers and a *softmax* layer for prediction, this setup allows freedom for each task to have auxiliary input layers and be optimized per task. We optimize these layers per task.

3.1 Training JMTE Model

We set the dimensionality of the input embedding layer to 300 and the hidden GRU to 70. We concatenate clause feature, explained at 4.2 to Task-BHM embedding layer. At each training epoch, we train the model in the following order: shared embedding, shared soft parameters of the hidden GRU, 3 specified hidden dense layers for each task. We experimented with different number of units in dense layers and our results on dev set suggest the following setup for dense layers: 300 units for Task-E and 200 units for Task-BHM. We use a *softmax* layer for predicting emotion tags. Further, we use an input maximum length of 70, 10 epochs, and Adam (Kingma and Ba, 2014) optimizer with a learning rate 0.001. We use

[3] https://keras.io/

Condition	Training set	TweetEN	BLG+HLN	MOV
LL1	TweetEN	*21.9%*	6.9%	35.8%
LL2	BLG+HLN	37.0%	***38.3%***	62.4%
LL3	MOV	25.2%	17.0%	*32.1%*
LL4	BHMT	**43.9%**	21.7%	**79.0%**

Table 2: LIBLINEAR weighted F1 scores for different experimental conditions where we train on various training data set combinations and test within and across genres. Within genre is marked in italics. It should be noted that we did not balance the size of the training data across the different experimental conditions.

dropout (Graves et al., 2013) for regularization, with a dropout rate: 0.3 for both tasks. The loss function is a categorical-cross-entropy function. We use a mini batch (Cotter, 2011) of size 65.

4 Experiments and Results

4.1 Data

We split the data representing each emotion category per genre into 70%,10%,20% for train, dev, and test, respectively. We added dev set to training set after tuning our model parameters on dev set.

4.2 Baseline Models

We compare our proposed models to two baselines: a feature engineering architecture LIBLINEAR from the SVM family. We leverage LIBLINEAR architecture implementation in Weka.[4]; and a single GRU model architecture where we use all the data from both TweetEN and BHM in a single model.

Feature Engineering Baseline: For the LIBLINEAR setting, we build our model combining a number of features: character and word n-grams (uni-gram and bi-gram); POS: presence of POS tags taken from PennTreebank; syntactic features like presence of adjective, adverbs, or negation; and semantic features like presence of emotion words based on EmoNet lexicon (Mohammad, 2012), and clause feature, which we explain below. We report weighted F1 scores across the 8 PL8 emotion tags.
Clause feature - for this feature, we study the distribution of clauses emotion tags in multi-clausal sentences. We note that the majority of those sentences with multiple clauses tend to have clauses with specific emotion labels (e.g. sentence emotion tag *joy*, have clauses with tags *trust, anticipation, no-emotion, and surprise*). We model this feature as an 8-dimension vector, where each dimension represent one emotion tag with a binary value: 1 indicates the presence of sub-sentential emotion clause tag and 0 otherwise.

Table 2 illustrates the results of the LIBLINEAR models. We combine the test sets for BLG and HLN as they are relatively small. In LL1, training with TweetEN yields the worst results on all 3 data sets even on the TweetEN test set (within genre setting). In LL3, training with MOV yields the lowest within genre results compared to the other sets. In LL4, training with a combination of BLG+HLN+MOV and TweetEN yields the best results on TweetEN and MOV data sets. In addition, in this condition we combine gold annotated set, BHM, with distant supervision set TweetEN. In LL2, training with BLG+HLN, which is the smallest training set yields the best results on BLG+HLN, and compare to LL1, yields better results on TweetEN and MOV. Hence, size is not a factor in the performance and the feature engineering approach is not robust towards genre variation.
Single Task NN Learning Baseline: We experiment with a GRU architecture as a single task. The architecture is similar to the JMTE framework with an embedding layer, followed by a GRU, then 3 dense layers followed by a softmax classification layer. Both the GRNN and JMTE are implemented using Keras and Tensorflow[5] in the backend. Table 3 present the results for single task GRU baseline. We mentioned earlier about embedding layer setup in our model, we experimented with two different

[4] https://www.cs.waikato.ac.nz/ml/weka/
[5] https://www.tensorflow.org/

Condition	training set	TweetEN	BLG+HLN	MOV
GRNN1	TweetEN	61.9%	27.2%	36.5%
GRNN1-pt		65.5%	29.9%	45.7%
GRNN2	BLG+HLN	33.7%	31.2%	27.4%
GRNN2-pt		40.2%	37.2%	38.2%
GRNN3	MOV	45.3%	26.4%	66.2%
GRNN3-pt		46.6%	28.02%	69.6%
GRNN5	BHMT	76.2%	81.2%	89.8%
GRNN5-pt		**78.1%**	**83.6%**	**91.0%**

Table 3: Single task GRNN F-score results on all the test data sets. Using pre-train (GRNN-pt) word embedding to initiate the weights for embedding layers, creates better results across all conditions. Colored results are out-of-genre evaluations.

Model	TweetEN	BLG+HLN	MOV
LL5	43.9%	21.7%	79.0%
GRNN5-pt	78.1%	83.6%	91.0%
JMTE	78.5%	82.3%	92.2%
JMTE-pt	**80.0%**	**84.0%**	**92.6%**

Table 4: Weighted macro F1-scores yielded by baseline models using all the training data BHMT compared against the JMTE model.

setup, a) we initiate random weights for word vectors b) we initiate the weights using pre-trained word embedding; in both of these settings we tuned the word vectors using the training set. We experimented with different word embedding models, mainly to have a better coverage for all these 4 genres in our corpus. Common training set for word embedding models are wiki+news, news, tweets, and common crawl, and the methods are Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and fasText (Bojanowski et al., 2016). Our results indicate that common crawl corpus with 2 million words, trained using fastText model has the most word coverage among these genres. [6] We experimented with google news (trained using word2vec), wikipedia+Gigaword (trained using GloVe), Twitter (trained using GloVe).[7]

The best results are obtained using all the data in condition GRNN5-pt which trains on all the labeled data, and we used pre-trained word embedding to initialize the wights for embedding layer. The results yielded by the Deep learning model surpass those of the LIBLINEAR baseline by a significant margin comparing GRNN5-pt (F1 scores of: 78.1%, 83.6%, 91.0%, for TweetEN, BLG+HLN, MOV, respectively) compared with LL5 (F1 scores of: 43.9%, 21.7%, 79%, for TweetEN, BLG+HLN, MOV, respectively). GRNN1-pt and GRNN3-pt indicate that within genre training and testing yields the best results. Even when there is more data available for training, comparing GRNN1-pt (more training data) compared to GRNN3-pt condition, GRNN3-pt yields higher results on the MOV (within genre) test data at 69.9% vs. 45.7% F1 score as yielded from GRNN1-pt condition. GRNN1-pt consistently beats LL1 across all test sets, likewise for GRNN3-pt and LL3. The pattern is consistent.

Joint MultiTask Learning of Emotion Model: Table 4 shows the performance of the JMTE model proposed in this paper against the two baselines GRNN (GRNN5-pt) and LIBLINEAR (LL5) when using all the data for training. Overall, the JMTE-pt yields the best results across all test data sets, significantly outperforming the LIBLINEAR baseline as well as beating the single task GRNN architecture.

5 Discussion

It is worth noting that we could not compare our results against other systems in the literature since available systems are typically trained on the EK6 tag set. The only system we know that is trained and tested on the PL8 tag set is that of Muhammad and Ungar (2017) but the test set is twitter data which is

[6] https://fasttext.cc/docs/en/english-vectors.html
[7] https://nlp.stanford.edu/projects/glove/

Test set	Model	joy	trust	anti	surprise	sad	fear	anger	disgust
TweetEN	JMTE-pt	86.2%	31.3%	57.5%	52.1%	69.8%	29.2%	26.3%	86.6%
	GRNN5-pt	85.2%	22.1%	54.9%	48.9%	53.2%	39.9%	39.9%	85.6%
	LIBLINEAR	42.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	63.0%
MOV	JMTE-pt	93.5%	29.0	68.8%	50.4%	66.2%	64.2%	0.0%	93.2%
	GRNN5-pt	92.5%	29.0	68.1%	50.4%	65.8%	63.8%	0.0%	93.1%
	LIBLINEAR	88.8%	0.01%	34.8%	39.0%	48.8%	20.0%	0.0%	90.3%
BLG+HLN	JMTE-pt	88.7%	53.2%	83.6%	87.3%	84.2%	86.4%	83.3%	83.7%
	GRNN5-pt	87.7%	52.2%	83.6%	87.3%	84.2%	86.4%	82.1%	82.7%
	LL5	78.4%	28.6%	55.8%	45.5%	54.5%	51.9%	66.7%	50.0%

Table 5: F-score results with LIBLINEAR and JMTE across different emotion tags, per genre. Emotion tags *anger* and *trust* are minority in MOV, hence, the model could not predict correct instances in these two emotion categories.

not a stable data set.[8] Furthermore, in an attempt to understand the performance of the various models per emotion tag. Table 5 compares F-score per emotion tag between LIBLINEAR and JMTE, which indicates JMTE is able to learn each emotion tag better than LIBLINEAR and GRNN models. We built JMTE to generalize emotion detection and classification across different genres.
Certainly, adding more data would create better results, but this is not the aim of our study. We observe that LIBLINEAR has shortcoming when we add TweetEN to the training set, and when we train LIBLINEAR with different genre than TweetEN. JMTE and GRNN overcome this shortcoming across all genres, and for some genres the improvement is significant (*i.e.* BLG+HLN and TweetEN).
GRNN produces close results to JMTE, however JMTE has the advantage to be specialized per genre by adding auxiliary layers and genre specific layers.

LIBLINEAR trained on BLG+HLN+MOV performed poorly on tweetsEN, this model can only classify *joy* and *disgust*, with f-score 42.0% and 63.0% respectively and these two emotion tags are the most populated tags in this set.
Although JMTE is able to generalize emotion classification across genres and create the best results, however, GRNN trained on BLG+HLN+MOV created better results for HLN, which indicates JMTE needs auxiliary (*i.e.* specific) layer for different genres, particularly if the genre has lower amount of test and train data (HLN) or genre is very different compare to other genres (MOV). In our setup HLN has the minimum number of instances in multigenre corpus (training: 825 and test: 425) and MOV is very different compare to tweets, BLG, and HLN, hence, adding specific layers for these genres can improve the results.
Other challenge here is pre-trained word embedding model. We observed that pre-trained word embedding creates better results, however, none to the best of our knowledge are trained using different genres to correspond to our need. We observe the results in Table 3 using pre-trained word embedding, and the significance of better coverage for all the genres. Comparing GRNN4 vs GRNN4-pt on TweetEN 31.7% vs. 32.9%, which has only 1% improvement, vs., GRNN2 vs. GRNN2-pt on TweetEN 33.7% vs. 40.2% which has 7% improvement, which, is an indication that an effective word embedding can improve the results by a large margin, as GRNN2-pt has less training data compare to GRNN-4.
Further, we observed that most of the instances that are misclassified in BLG, HLN, and MOV, are the ones with lower confidence score annotation, as these sets are manually annotated. In addition, observing the confusion matrix in JMTE suggests that in both tasks data points are separable, since misclassified data points are not skewed towards particular or the most popular classes. However, emotion tag *surprise* is mainly misclassified with *joy* and *discussed*, we further observed the data points to address this issue and noticed most of these data points imply surprise and even as human it was difficult for us to categorize them as surprise.

[8]Unfortunately due to licensing issues, exact tweets can't be shared, only IDs, hence when retrieving the actual tweets, we noted that we can only retrieve 75% of the exact tweets.

6 Related Work

6.1 Emotion Detection and Classification

Emotion detection has attracted several NLP applications like chatbots, stock market, and human personality analysis. Several studies investigated the problem in various genres. We present some of the studies most relevant to this paper. In the literature, emotion detection is cast as a classification problem, hence, the objective is to effectively learn emotion cues. Among the feature engineering approaches, we review the following works:Gilad (2005) collected a set of blog posts - online diary entries - which include an indication of the writers' mood. Carlo and Rada (2007) collected and manually labeled 1,250 headlines (HLN) for emotion classification and valence (*i.e.* positive, negative) using the 6 basic emotions identified by Ekman (Paul, 1992) (EK6) tags. Saima and Stan (2007) collected and labeled a blog posts corpus (BLG) using EK6 tags on both the sentence and the phrase levels, they annotated emotion categories, emotion intensity, and identifying emotion phrases in blog posts. Diman et al. (2010) experimented with hierarchical classification for emotion analysis which considers the relation between neutrality, polarity and emotion of a text. Diana and Carlo (2010) presented a categorical model and dimensional model for recognition of affective states (emotion cues). Mohammad (2012) investigated word-level affect lexicons features on sentence-level emotion detection. Özbal and Daniele (2013) showed the effect of incorporating different levels of syntactic and semantic information on sentence level emotion detection.
In recent years, unlimited access to social media data such as *Twitter and Facebook*, enabled the community to have access to large amount of data. In these works researchers have used supervised learning model trained on lexical, semantic, and stylistic features to classify emotion in Twitter (Wenbo et al., 2012; Roberts et al., 2012; Ashequl and Ellen, 2013; Yan, 2014; Saif and Svetlana, 2015; Yan and Howard, 2016; Svitlana and Yoram, 2016). Muhammad and Ungar (2017) proposed a gated recurrent neural network architecture to classify emotion in tweets.

6.2 Multi-Task Learning in Deep Neural Net

Multi-Task learning is inspired by human learning. As human, when we learn new tasks, often we apply the knowledge we have gathered from related tasks. In the literature, multi-task learning comes in different forms: joint learning, learning to learn, and learning with auxiliary tasks. The following studies show the benefit of multi-task learning in closely-related or different type of tasks (R, 1993; Ronan and Jason, 2008; Collobert et al., 2011; Xiao et al., 2011; Seltzer and Droppo, 2013; Devries et al., 2014; Xia and Liu, 2015; Luong et al., 2015; Anders and Goldberg, 2016; Kazuma et al., 2016; Andor et al., 2016; Jonathan et al., 2016; Makoto and Bansal, 2016).
To the best of our knowledge multi-task learning has not been studied to detect emotion in multigenre text input. The closest work to ours is the work of (Xia and Liu, 2015). In the latter study of (Xia and Liu, 2015), they proposed a multi-task learning framework that leverages activation and valence information for acoustic emotion recognition.
Our work contributes the following: a) we empirically illustrate that emotion cues can be learned robustly across genres by framing the problem as a Joint Multi-Task learning problem.

7 Conclusion

Combination of different genre datasets can improve and generalize emotion detection in sentences. We showed multi-task deep neural net models are able to successfully classify emotion in multigenre and across genres. We showed that unified annotation is beneficial for emotion detection through combining different genres to augment and create larger training sets. We discuss the impact of pre-trained word embedding in emotion classification and the challenges involve finding a proper word embedding that has the most coverage among different genres in our corpus.
Our future direction is to increase number of instances for minority genres and experiment with formal and informal text to represent different tasks. We aim to experiment with more robust tuning method to create better pre-trained word embedding. Further, we aim to add genre specific layer to improve results across different genres.

References

Sgaard Anders and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. Vol. 2*.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042*.

Qadir Ashequl and Riloff Ellen. 2013. Bootstrapped learning of emotion hashtags hashtags4you. *WASSA, NAACL-HLT*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks 5.2*, pages 157–166.

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

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Strapparava Carlo and Mihalcea Rada. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 70–74. Association for Computational Linguistics.

Kyunghyun Cho. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Junyoung Chung. 2015. Gated feedback recurrent neural networks. *ICML*, pages 2067–2075.

Ronan Collobert, Jason Weston, Lon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, pages 2493–2537.

Andrew Cotter. 2011. Speech recognition with deep recurrent neural networks. *Better mini-batch algorithms via accelerated gradient methods.*

Terrance Devries, Kumar Biswaranjan, and Graham W. Taylor. 2014. Multi-task learning of facial landmarks and expression. *Computer and Robot Vision (CRV), 2014 Canadian Conference on. IEEE*.

Inkpen Diana and Strapparava Carlo. 2010. Evaluation of unsupervised emotion models to textual affect recognition. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 62–70. Association for Computational Linguistics.

Ghazi Diman, Inkpen Diana, and Szpakowicz Stan. 2010. Hierarchical versus flat classification of emotions in text. In *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*, pages 140–146. Association for Computational Linguistics.

Mishne Gilad. 2005. Experiments with mood classification in blog posts. In *Proceedings of ACM SIGIR 2005 workshop on stylistic analysis of text for information access*, volume 19, pages 321–327.

Alex Graves, Abdel rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. *Acoustics, speech and signal processing (icassp), ieee international conference on. IEEE*.

Sepp Hochreiter and Jorgen Schmidhuber. 1997. Long short-term memory. *Neural computation 9.8*, pages 1735–1780.

Godwin Jonathan, Stenetorp Pontus, and Riedel Sebastian. 2016. Deep semi-supervised learning with linguistically motivated sequence labeling task hierarchies. *arXiv preprint arXiv:1612.09113*.

Hashimoto Kazuma, Xiong Caiming, Tsuruoka Yoshimasa, and Socher Richard. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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

Miwa Makoto and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv preprint arXiv:1601.00770*.

Tomas Mikolov, Chen Kai, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Saif Mohammad. 2012. Portable features for classifying emotional text. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 587–591. Association for Computational Linguistics.

Abdul-Mageed Muhammad and Ungar Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. *ACL*.

Gözde Özbal and Pighin Daniele. 2013. Evaluating the impact of syntax and semantics on emotion recognition from text. In *Computational Linguistics and Intelligent Text Processing*, pages 161–173. Springer.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *International Conference on Machine Learning*.

Ekman Paul. 1992. An argument for basic emotions. *Cognition and emotion*, 6.3-4:169–200.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*.

Robert Plutchik. 1962. The emotions: Facts, theories, and a new model. *New York: Random House*.

Caruna R. 1993. Multitask learning: A knowledge-based source of inductive bias. *Machine Learning: Proceedings of the Tenth International Conference*.

Kirk Roberts, Michael A. Roach, Joseph Johnson, Josh Guthrie, and Sanda M. Harabagiu. 2012. Empatweet: Annotating and detecting emotions on twitter. *Vol. 12. LREC*.

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

Mohammad Saif and Kiritchenko Svetlana. 2015. Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence 31.2*, pages 301–326.

Aman Saima and Szpakowicz Stan. 2007. Identifying expressions of emotion in text. In *International Conference on Text, Speech and Dialogue*, pages 196–205. Springer.

Michael L. Seltzer and Jasha Droppo. 2013. Multi-task learning in deep neural networks for improved phoneme recognition. *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE*.

Volkova Svitlana and Bachrach Yoram. 2016. Inferring perceived demographics from user emotional tone and user-environment emotional contrast. *ACL (1)*.

Shabnam Tafreshi and Mona Diab. 2018. Sentence and clause level emotion annotation, detection, and classification in a multi-genre corpus. In *Proceedings of LREC 2018*, pages 1246–1251.

Wang Wenbo, Chen Lu, Thirunarayan Krishnaprasad, and Sheth Amit. 2012. Harnessing twitter" big data" for automatic emotion identification. *Privacy, Security, Risk and Trust (PASSAT), International Conference on and 2012 International Confernece on Social Computing (SocialCom). IEEE*.

Rui Xia and Yang Liu. 2015. A multi-task learning framework for emotion recognition using 2d continuous space. *IEEE Transactions on Affective Computing*.

Li Xiao, Ye-Yi Wang, and Gokhan Tur. 2011. Multi-task learning for spoken language understanding with shared slots. *Twelfth Annual Conference of the International Speech Communication Association*.

Liew Jasy Suet Yan and Turtle Howard. 2016. Exploring fine-grained emotion detection in tweets. *NAACL-HLT*.

Liew Jasy Suet Yan. 2014. Expanding the range of automatic emotion detection in microblogging text. *EACL*.

How emotional are you? Neural Architectures for Emotion Intensity Prediction in Microblogs

Devang Kulshreshtha*, Pranav Goel*, and Anil Kumar Singh

Indian Institute of Technology (Banaras Hindu University) Varanasi
Varanasi, Uttar Pradesh, India
{devang.kulshreshtha.cse14, pranav.goel.cse14, aksingh.cse}@iitbhu.ac.in

Abstract

Social media based micro-blogging sites like Twitter have become a common source of real-time information (impacting organizations and their strategies), and are used for expressing emotions and opinions. Automated analysis of such content therefore rises in importance. To this end, we explore the viability of using deep neural networks on the specific task of emotion intensity prediction in tweets. We propose a neural architecture combining convolutional and fully connected layers in a non-sequential manner - done for the first time in context of natural language based tasks. Combined with lexicon-based features and transfer learning, our model achieves state-of-the-art performance, outperforming the previous best system by 4.4% Pearson correlation on the WASSA'17 EmoInt shared task dataset. We investigate the performance of deep multi-task learning models trained for all emotions at once in a unified architecture and get encouraging results. Experiments performed on evaluating correlation between emotion pairs offer interesting insights into the relationship between them. The code for our experiments is publicly available.

1 Introduction & Related Work

One can use text in any language not only to express a variety of emotions but also to convey the associated 'intensity' of the emotion. *Emotion intensity* is the degree of an emotion (like anger or fear) expressed by the speaker or author. For example, *'When you lose somebody close to your heart you lose yourself as well. Crying my heart out!!!'* expresses *more* sadness than, say, *'All friends are out of town. Feeling a bit lonely...'*. Automatic detection of emotion intensity can be useful for natural language based systems. An e-commerce firm seeking to publicize positive reviews of its products would want to use the statements expressing high amount of joy or happiness. The intensity of anger expressed in a grievance can be used to automatically decide the priority of addressing complaints. A possible example scenario: based on the tweet *'X box one controller jockey update is the fucking worst! #fuming'* expressing a higher intensity of anger than the tweet *'New madden bundle code for x box one is stupid, takes forever to download lol'*, may point to the need of prioritizing addressing issues with the new jockey stick before looking into the madden bundle of the same product (an Xbox One).

The WASSA'17 workshop conducted the EmoInt shared task (Mohammad and Bravo-Marquez, 2017b) where given a tweet and the emotion it exhibits, systems had to predict the intensity of this given emotion as a real valued score between 0 and 1. We use the shared task setting and the dataset they provide to develop and evaluate all our approaches. The main contributions of our work are:

a) A neural architecture for emotion intensity detection which combines convolutional layers with fully connected layers in a non-sequential or 'parallel' fashion. Such a combination of these neural models or layers has been explored in computer vision (Antol et al., 2015; Vijayanarasimhan et al., 2017) but, to the best of our knowledge, our work is the first to utilize this in Natural Language Processing based tasks. Our results will motivate the NLP community to take an increased interest in such architectures. We also rely on lexicon-based linguistic information in our network, and use transfer learning by incorporating

* The two first authors have equal contributions to the paper

Proceedings of the 27th International Conference on Computational Linguistics, pages 2914–2926
Santa Fe, New Mexico, USA, August 20-26, 2018.

activations of a pre-trained convolutional neural network which uses emojis to learn representations for related tasks of sentiment and emotion detection in tweets (Felbo et al., 2017). Our model achieves *state of the art performance* on the EmoInt dataset.

b) Two multi-task neural network based models which successfully handle all emotions jointly, as opposed to having a separate network for each individual emotion. Our best multi-task learning based approach performs relatively well when compared to other approaches and previous systems while benefiting from better generalization, speed and lesser trainable network parameters.

c) A comprehensive set of ablation tests to show how the various components within our models contribute, to help guide the design of future systems for our task as well as other, related tasks.

d) Experiments to gauge the correlation between different pairs of emotion and an effort to link our observations with past findings in linguistic and psychological experiments.

e) Error analysis to assess reasons for erroneous intensity prediction in tweets.

The code for all our approaches is publicly available [1].

Most datasets and systems focus on emotion classification or detecting the presence or absence of an emotion (Brooks et al., 2013; Alm et al., 2005; Aman and Szpakowicz, 2007; Bollen et al., 2011; Wang et al., 2016). To the best of our knowledge, apart from the dataset used for conducting the EmoInt shared task, there is only one resource annotated for the intensity of emotion in text, which was released by Strapparava and Mihalcea (2007) as part of SemEval 2007. Annotators gave a score between 0 and 100 for the degree of emotion in newspaper headlines using a slide bar in a web interface.

We perform our experiments on a Twitter dataset (Mohammad and Bravo-Marquez, 2017a) (Table 4). Users from many backgrounds express emotions via tweets. Increasing activity of bots on Twitter (Gilani et al., 2017) and companies aiming to accurately respond to tweets about their products or services makes Twitter an attractive domain for sentiment analysis (Fan and Gordon, 2014; Nakov et al., 2016).

Twenty two systems participated in the EmoInt shared task. Approaches included random forest regressors, neural approaches and lexicon-based methods (Mohammad and Bravo-Marquez, 2017b).

2 Proposed Neural Framework: LE-PC-DNN

We propose a novel neural network based architecture that combines - (i) Convolutional layers, (ii) Fully-connected layers, (iii) Linguistic features, and (iv) Pretrained CNN activations in a non-sequential fashion (detailed below). The architecture and the hyperparameter setting is consistent across all emotions.

2.1 Embedding Layer ($l^{(1)}$)

The input to our network's first layer is a sequence of tokens $\{w_1, w_2, ...w_n\}$ (padded when necessary). $l^{(1)}$ begins by associating each word w with a feature vector $\mathbf{e}_w$, also called *word embeddings* (Bengio et al., 2003). The obtained embedding matrix $\mathbf{E} \in R^{n*d}$ serves as the input to next layer.

The model consists of two different types of layers - *Parallelly-connected layers* and *Sequential fully-connected layers* (Figure 1).

2.2 Parallelly-connected layers

The output E of the embedding layer ($l^{(1)}$) above is fed to two parallel layers - the **CNN layer** ($l_a^{(2)}$), which applies convolutional operation on the embedding matrix $\boldsymbol{E}$ followed by a max-pooling-over-time operation to get the $l_a^{(2)}$ layer representation (Figure 1) and the **Average Embedding Layer** ($l_b^{(2)}$), which is calculated by taking the mean of word embeddings $\boldsymbol{E}$ across the length of the tweet, giving us a d-dimensional feature vector. $l_b^{(2)}$ is supposed to capture the global context of the tweet. The CNN layer and the average embedding layer are both fully connected to layers $l_a^{(3)}$ and $l_b^{(3)}$ respectively after applying Dropout (Srivastava et al., 2014) in each case to control over-fitting of parameters.

The network consists of two more parallel layers ($l_c^{(2)}$, $l_d^{(2)}$), which process the input tweet $\boldsymbol{T}$ directly. For emotion intensity prediction task, the **Linguistically Informed Layer** ($l_c^{(2)}$) uses the *TweetToLexiconFeatureVector* filter developed in the Affective Tweets[2] package to get 43 tweet-level features, cal-

[1] https://github.com/Pranav-Goel/Neural_Emotion_Intensity_Prediction

[2] https://github.com/felipebravom/AffectiveTweets

culated using several lexicons (details in Mohammad and Bravo-Marquez (2017b)). Sentiment-based lexicons and other resources of linguistic knowledge can capture some aspects of data that are different than those learned by CNN/LSTM relying on word embeddings (Chen et al., 2017) and boost performance (Ebert et al., 2015). There are different ways to use linguistic features with neural architectures (Ebert et al., 2015; Park et al., 2016). We incorporate these external features in a simple manner (Figure 1). The input to the layer $l_c^{(2)}$ is a tweet T. Applying TweetToLexiconFeatureVector filter gives a 43-dimensional feature vector.

We also use **Pretrained CNN features** ($l_d^{(2)}$) by leveraging the activations of a Convolutional Neural Network (CNN) pre-trained on emoji prediction task called DeepMoji (Felbo et al., 2017) [3]. This CNN was trained on 1.3 billion emoji-containing tweets and tested on eight benchmark datasets within emotion, sarcasm and sentiment detection. Since emoji prediction is closely related to emotion intensity prediction, we hypothesize that transferring knowledge via pre-trained CNN activations could help improve performance for our task. Each tweet is converted into a 2304-dimensional feature vector $l_d^{(2)}$ by feeding the tweet into the DeepMoji-CNN and extracting activations of the last hidden layer.

2.3 Sequential Fully-Connected Layers ($l^{(4)}, l^{(5)}, l^{(6)}$)

The layer ($l^{(4)}$) is obtained by concatenating the feature activations of layers $l_a^{(3)}, l_b^{(3)}, l_c^{(2)}, \& l_d^{(d)}$. $l^{(4)}$ is connected to $l^{(5)}$, which is a fully-connected layer. Additionally, a linear hidden layer ($l^{(6)}$) with lesser number of neurons is introduced on top of $l^{(5)}$.

2.4 Output Layer ($\hat{y}$)

Finally, we use a sigmoid neuron after $l^{(6)}$ to compute the intensity score $\hat{y}$ of between 0 and 1.

It is quite common to have fully connected layers or small feedforward neural networks connected in a sequential or hierarchical manner to sequence-to-sequence models like CNNs/LSTMs (Lee and Dernoncourt, 2016; Plahl et al., 2013). The feedforward networks in this case take in the pooled output of CNNs/LSTMs as input. Our architecture (Figure 1), however, combines feedforward, convolutional layers and manually extracted features in a non-hierarchical manner (layer $l_a^{(3)}$) as well as in hierarchical fashion (layer $l_b^{(3)}$). Such a non-sequential combination of fully-connected and convolutional networks is *novel* for NLP tasks, to the best of our knowledge (though this has been used by the vision community (Antol et al., 2015; He et al., 2017; Vijayanarasimhan et al., 2017)). We call this neural framework with a non-hierarchical or *parallel* combination of CNNs/LSTMs, fully-connected layers, lexical features and pretrained CN activations as **PC-DNN** (Parallel Combination of Deep Neural Networks). As it incorporates **L**inguistic features (layer $l_c^{(2)}$) and **E**moji-based pretrained CNN activations (layer $l_d^{(2)}$) as well, we will refer to the network discussed in this section as **LE-PC-DNN** (Figure 1).

3 Deep Multi-Task Learning (DMTL): Handling all Emotions in a Unified Architecture

Multi-task Learning (Caruna, 1993; Caruana, 1998) has resulted in successful systems for various NLP tasks (Collobert and Weston, 2008), especially in cross-lingual settings (Huang et al., 2013). MTL was first applied to Emotion Intensity prediction in (Goel et al., 2017). In their neural architecture, the initial network layers are shared across multiple emotions upto a certain point, after which the architecture gets diverged for each emotion. The objective is to jointly train on different emotion datasets such that initial layers increase generalization and the individual final layers can learn task specific features. The network input features are the concatenation of average word embeddings across the length of the tweet and linguistic features (using *Tweet2LexiconFeatureExtractor*), and the neural layers were fully-connected.

3.1 LE-PC-DMTL for Emotion Intensity Prediction

The above DMTL model only uses feed-forward layers as the input is a 1-dimensional representation of the tweet. Our proposed LE-PC-DNN network, on another note includes convolutional layers and pre-trained CNN activations in the architecture. The model however is separate for each emotion. We

[3] https://github.com/bfelbo/DeepMoji

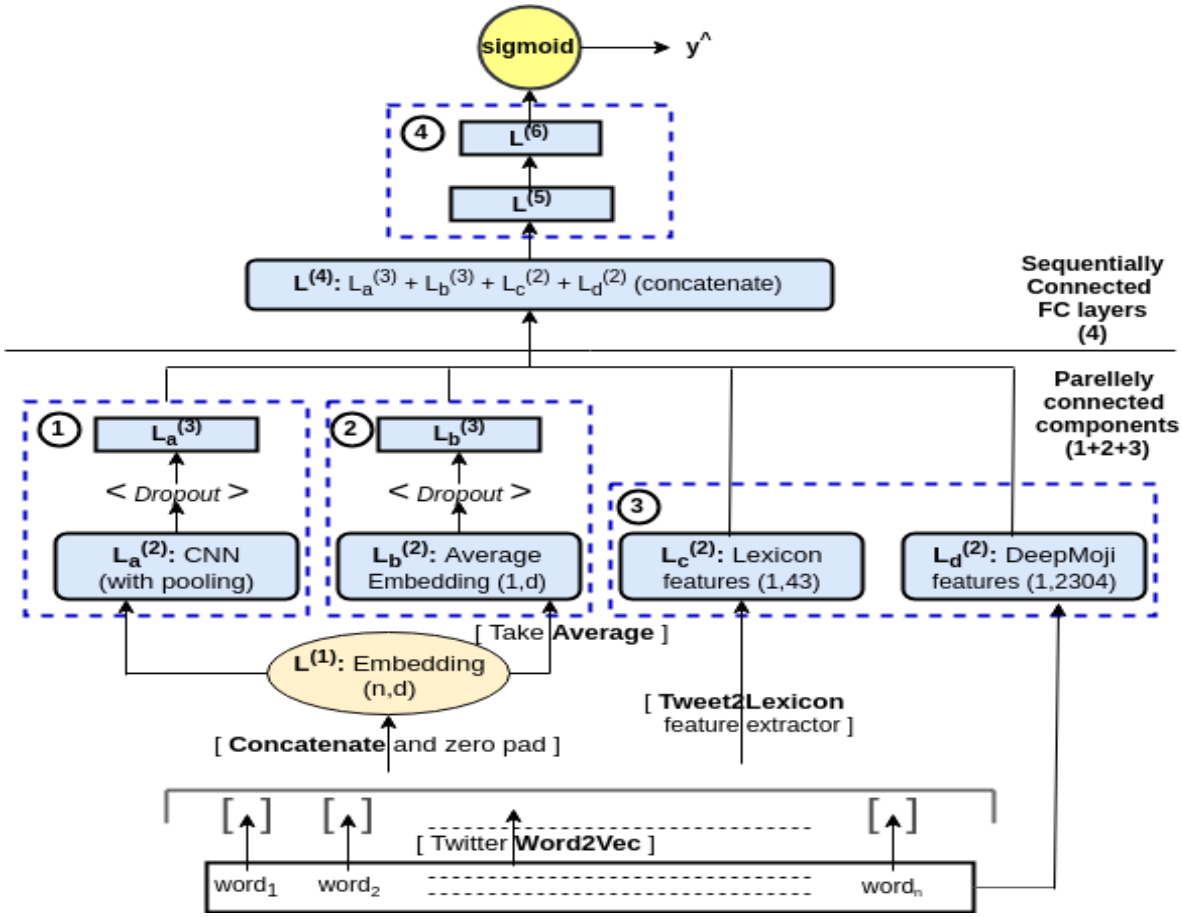


Figure 1: Network architecture for the **LE-PC-DNN** model: Various components - ***L***exicon-based features, an ***E***moji detecting CNN's pre-trained activations, fully connected layers, and a CNN/LSTM undergo a ***P***arallel ***C***ombination (parallelly-connected) to form a ***D***eep ***N***eural ***N***etwork (Section 2).

can combine the different emotion models of LE-PC-DNN architecture into a unified network by using the idea of Multi-Task learning proposed in (Goel et al., 2017). In order to that, we adopt the initial architecture of LE-PC-DNN and then the final fully-connected layers are different for each emotion.

Our basic model (Figure 2a) consists of two different types of layers: *shared representation layers* and *task-specific representation layers*. The network receives a tweet-emotion pair (T, e) and produces an intensity score between 0 and 1.

3.1.1 Shared Representation Layers

These layers receive tweets from all emotions which helps in achieving better generalization as the layers are forced to learn features which work for multiple subtasks. Similar to LE-PC-DNN (Figure 2a), there are four such levels of layers - $l^{(1)}$ (Embedding layer), $l^{(2)}$ (CNN layer, Avg. Embedding layer, Linguistic layer, Pretrained CNN features), $l^{(3)}$ (Fully-connected layers), and $l^{(4)}$ (concatenation layer).

3.1.2 Task-specific Representation Layers

The last two layers are allowed to be different across the different sub-tasks to learn task-specific features.

1. **Fully Connected Layers ($l^{(5)}, l^{(6)}$):** For every emotion, a ReLu transformation will map shared layer $l^{(4)}$ to task-specific layers $l^{(5)}$ ($l_a^{(5)}, l_f^{(5)}, l_j^{(5)}, l_s^{(5)}$ - separate for each emotion), with the separate outputs going to another hidden layers - $l_a^{(6)}, l_f^{(6)}, l_j^{(6)}, l_s^{(6)}$.

2. **Output Layer ($\hat{y}$) :** $l^{(6)}$ is connected to a single sigmoid output neuron – $\hat{y}$, which gives the predicted intensity score between 0 and 1.

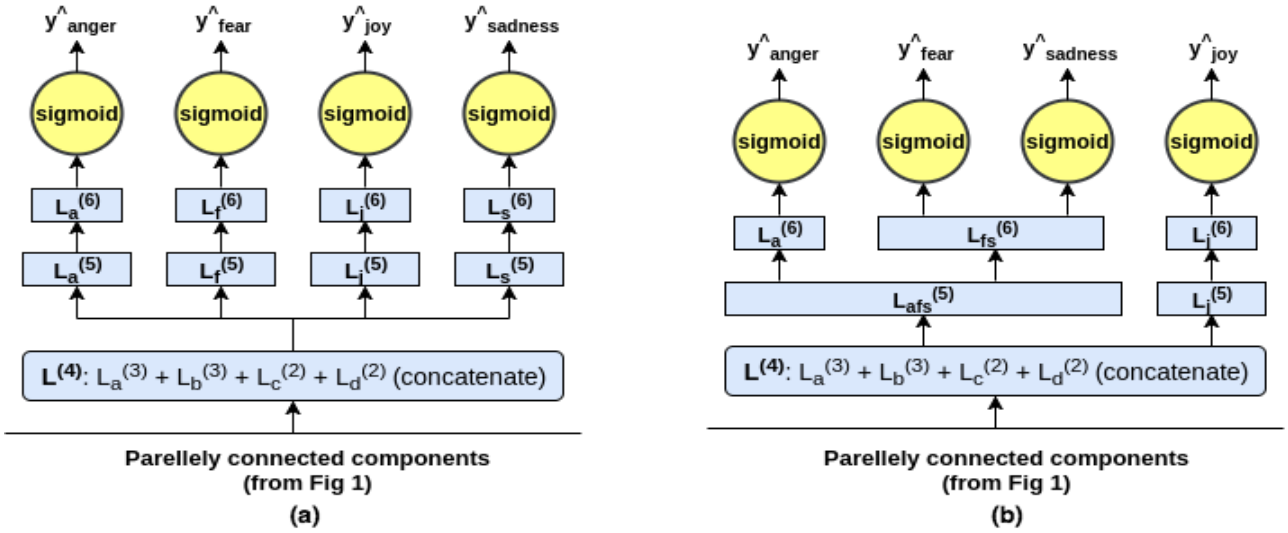


Figure 2: **(a) LE-PC-DMTL**: A deep multi-task learning neural network that uses parallelly connected components - a CNN/LSTM, fully connected layers, a feature vector derived from linguistic lexicons and pre-trained activations of a network trained for Emoji detection (DeepMoji). So, a ***L***exicon and ***E***moji detection based, ***P***arallelly ***C***onnected ***D***eep ***M***ulti-***T***ask ***L***earning neural network and **(b) LE-PC-DMTL-EI**: This network branches differently to exploit the different pairwise correlations between the different emotions (some emotion pairs sharing more layers than others), to optimize performance. So, a ***L***exicon and ***E***moji detection based, ***P***arallelly ***C***onnected ***D***eep ***M***ulti-***T***ask ***L***earning neural network, optimized specifically for the task of ***E***motion ***I***ntensity detection. Both these models handle all the four emotions together in a single architecture (Section 3).

Figure 2a clearly demonstrates that replacement of individual bottom layers with shared layers reduces the network's parameters significantly, less hyper-parameters required to be fine-tuned and speedup during training the network compared to the scenario of having separate networks for each emotion.

3.2 LE-PC-DMTL-EI: A Task-Optimized DMTL Model

The architecture of our proposed LE-PC-DMTL model is that the shared layers will be common across all emotions, *irrespective* of the degree of correlation between any particular pair. The correlations between emotions are investigated via experiments described in section 6. We would like the model to incorporate or *learn* relationships between different emotions and not treat all emotions or subtask in a similar manner since we cannot expect all emotions to be related to each other in the same way.

Inspired by Lu et al. (2016) and some results discussed in section 6, we propose to learn a network architecture which *allows grouping of similar emotions and separate branching of unrelated tasks*. Here we remove the constraint of feature sharing up to a certain level and allow sharing of more parameters between correlated emotions.

Learning the Network Architecture: We initialize the network with our basic LE-PC-DMTL model (Figure 2a). Our objective is to learn an architecture where similar emotions are grouped together and uncorrelated emotions are branched out. After initialization, we refine the architecture step-by-step by experimenting with various combinations. Since manual exploration of such a combinatorially large space is not possible, we follow two heuristics:

1. A model is favored over other if it gives a higher Pearson correlation on the cross-validation set.

2. Models where emotion pairs having high positive correlation (Section 6) share more parameters.

The refinement process is repeated manually for fixed iterations or until no further gain in Pearson correlation is observed (on development set). Since we start from our basic multi-task model, the final model would definitely be an improved version. We apply this heuristic-based algorithm to the EmoInt shared

Emotion	Train	Dev	Test	Total
Anger	857	84	760	1701
Fear	1147	110	995	2252
Joy	823	74	714	1611
Sadness	786	74	673	1533
Total	3613	342	3142	7097

Table 1: The number of instances (tweets) in the EmoInt shared task dataset

Model /	**LE-PC-DNN**					**LE-PC-DMTL**		**LE-PC-DMTL-EI**	
Emotion	$L_a^{(2)}$	$L_a^{(3)}$	$L_b^{(3)}$	$L^{(5)}$	$L^{(6)}$	$L_x^{(5)}$	$L_x^{(6)}$	$L_x^{(5)}$	$L_x^{(6)}$
Anger	CNN (250,Max)	128	256	128	32	64	32	256	64
Fear						75	40		75
Sadness						64	32		
Joy						40	20	125	50

Table 2: Summary of network hyperparameters for our proposed models (Figures 1 and 2).

task dataset (Mohammad and Bravo-Marquez, 2017b). The resulting architecture is shown in Figure 2b. Though the algorithm is general, the resulting architecture is meant to be optimal for the task to which it is applied. We refer to this resulting architecture as **LE-PC-DMTL-EI** (**LE-PC-DMTL** for **E**motion **I**ntensity prediction). Note that the pairwise correlations (heuristic 2) is only a guiding principle - the multi-task model discovers the optimal branching or sharing of layers based on the performance across a lot of variations. The captions of Figures 1 and 2 also serve to summarize the three models we propose.

4 Experimental Setting

4.1 Data

We use the *training*, *development*, and *testing* datasets provided within the WASSA EmoInt shared task (Mohammad and Bravo-Marquez, 2017b) to train and evaluate the various approaches (Table 4). The data files include the tweet ID, the tweet, the emotion of the tweet and the emotion intensity. Details regarding creation and annotation can be found in Mohammad and Bravo-Marquez (2017a).

4.2 Implementation Details

Word Embeddings: We use publicly available pre-trained word embeddings called the Twitter word2vec model (Godin et al., 2015). These were trained on 400 million tweets for the ACL-WNUT 2015 shared task (Baldwin et al., 2015) using the word2vec approach (Mikolov et al., 2013). The large number of tweets in training data makes it a better choice than others available. We choose it over other pre-trained models like Google word2vec (Mikolov et al., 2013) and the GloVe (Pennington et al., 2014) Twitter model. This was backed by cross validation results (not shown for brevity).

Preprocessing: Before forming word vector based embeddings of the Twitter tweets, we employed some preprocessing steps including removal of URLs and user mentions, stripping punctuations from word boundaries, hashtag segmentation ('#wearethebest' to 'we are the best' using the Word Segment [4] module in Python) and elongation removal ('goooooood' to 'good') (inspired by the work in Akhtar et al. (2017)). Such preprocessing helped in improving cross-validation performance.

Network Hyper-Parameters and Architecture Settings: We show the final hyper-parameter values for the LE-PC-DNN (section 2), and the multi-task models LE-PC-DMTL and LE-PC-DMTL-EI (section 3) in Table 2. These values were chosen on the basis of 7-fold cross validation results on the combined training and validation sets. We changed various network hyper-parameters like number of layers, output dimensionality, the type of pooling for CNNs (max versus average) and dropout value.

[4] https://github.com/grantjenks/wordsegment

Training: The network parameters are learned by directly minimizing the Mean Absolute Error between the actual and predicted intensity values. We optimize the above function by back-propagating through layers via Mini-batch Gradient Descent. We use a batch size of 8, 25-30 training epochs and Adam optimization algorithm (Kingma and Ba, 2014) with the parameters set as $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999\ and\ \epsilon = 10^{-9}$. These settings are common across all our models.

4.3 Evaluation

We evaluate the performance of all systems using the Pearson correlation coefficient (r) between gold ratings and predicted output intensities. This was the metric used in the EmoInt shared task. The Pearson coefficient measures linear correlation between two variables and is always between -1 and 1. A high (positive) value indicates strong (positive) correlation and a value of 0 indicates no correlation. We use following baselines to benchmark our results:

1. *Weka*: The Weka system (Mohammad and Bravo-Marquez, 2017a) served as the baseline for EmoInt shared task (Mohammad and Bravo-Marquez, 2017b) . It uses an SVM regressor on pre-trained word embeddings and lexicon based features.

   The other two baselines are the two top performing systems in the shared task:

2. *Prayas*: (Goel et al., 2017) used weighted ensembling technique to combine 3 different models - feedforward neural network on average word embeddings and sentiment features, CNN/LSTM+max-pooling on concatenated word embeddings, and deep multi-task learning model (described in Section 3). Note that they combine only the outputs (predicted intensities) of their different models, but the fact that combining outputs of different models helped with performance indicates that the architectures are able to complement each other. Our model (Figure 1) exploits this observation by letting the network learn how to best combine different neural models.

3. *IMS*: (Köper et al., 2017) use a random forest regressor on features based on manually created resources, automatically extended lexicons and the output of CNNs/LSTMs.

5 Results and Discussion

We show the Pearson correlation scores for all our proposed neural models and baselines (section 4.3) on the test set in Table 3. The major takeaways from these results and the ablation tests (Figure 3) are:

- **State-of-the-Art Performance** The proposed LE-PC-DNN model (section 2) outperforms all the baselines for all emotions (except for sadness in which our proposed multi-task architecture LE-PC-DMTL-EI fares better), setting a new state of the art score of **0.791** on average. It significantly outperforms the previous best model (Prayas) by 4.4% Pearson score on average.

- **A Solid Case for Multi-Task Learning for Emotion Intensity Prediction**: While the multi-task architecture used in the system Prayas (Goel et al., 2017) used just fully-connected layers, our proposed multi-task models uses CNNs, lexicon-based features and activations of a pre-trained CNN (section 3, Figure 1) to improve performance by a comprehensive 9.6-11.0% Pearson score (the last 3 columns of Table 3) and also gives competitive scores when compared to our state-of-the-art

Emotion	LE-PC-DNN	Prayas	IMS	Weka	LE-PC-DMTL-EI	LE-PC-DMTL	MTL (Prayas)
Anger	**0.767**	0.765	0.767	0.639	0.735	0.731	0.645
Fear	**0.791**	0.732	0.705	0.652	0.760	0.755	0.677
Joy	**0.803**	0.762	0.726	0.654	0.785	0.758	0.654
Sadness	0.803	0.732	0.690	0.648	**0.808**	0.786	0.672
Average	**0.791**	0.747	0.722	0.648	0.772	0.758	0.662

Table 3: Pearson correlations (r) obtained by the systems on the full test sets.

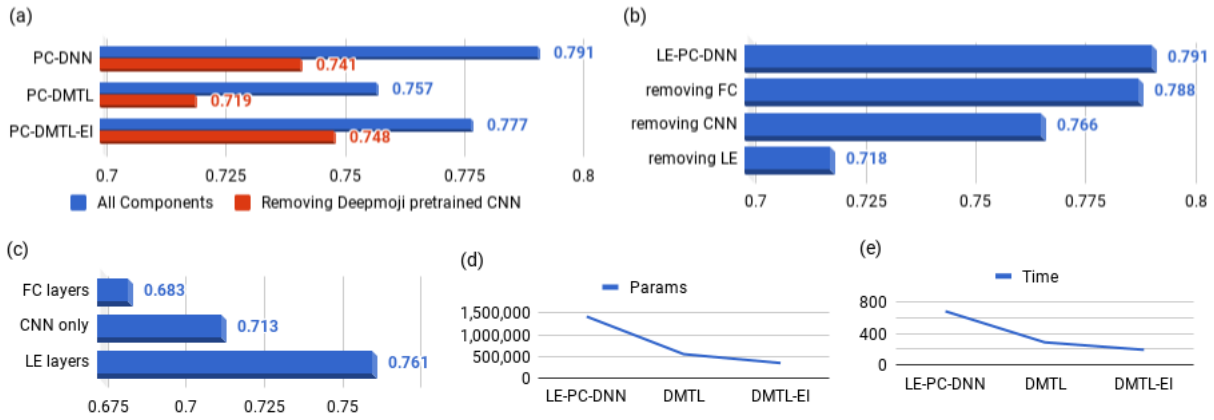


Figure 3: **Ablation test visualizations (scores are average Pearson correlation**: a) Effect of Transfer Learning, b) Effect of removing one of the parallel connected components from LE-PC-DNN architecture (Figure 1), c) Comparing individual parallel components of LE-PC-DNN (section 2.2), d) Comparing number of trainable parameters and e) training time with proposed multi-task learning models.

LE-PC-DNN model trained on and used for each emotion individually. Out of our two multi-task models, LE-PC-DMTL-EI (section 3.2) gives better performance by 1.4% Pearson score. It benefits from being optimized for our task by exploiting the correlation among certain pairs of emotions. The major advantage of using the multi-task models offer is the use of a unified architecture instead of separate models for each emotion, which means *less computational complexity*. This is established in Figures 3d and 3e, which show that our LE-PC-DMTL and LE-PC-DMTL-EI architectures (section 3) have less than one-third number of trainable parameters and take less than one-third the training time compared to our LE-PC-DNN model (section 2). This established the viability of our multi-task approach, which will only increase with the number of different emotions!

- **Effect of Transfer Learning**: Removing the deepmoji *pre-trained CNN activations* (section 2.2) has a significant effect of performance of all our proposed models including the multi-task ones (Figure 3a), with the drop of 5% Pearson score in case of the PC-DNN model (section 2) being the most notable. Transferring features from a related task (like emoji detection) trained on a much larger dataset (1.3 billion tweets compared to about just 1,000 in our case) proves to be very useful in terms of prediction performance. This also means that the representations learned by the Deepmoji model (Felbo et al., 2017) work well for the emotion intensity detection task.

- **Impact of the Different Non-Sequential Components in our Neural Architecture**: From Figures 3b and 3c, it is clear that all of our parallelly connected components (components 1, 2 and 3 in Figure 1) have a positive role in the overall performance. Lexicon-based features and pre-trained activations of a CNN trained for Emoji detection task (section 2; component 3 in Figure 1) contribute the most followed by CNN and then the fully-connected layers.

Note: For the training time results shown in Figure 3e, the experiments were run on a CPU with 4 GB RAM and 1.7 GHz Intel core i5 processor. Also, the time for LE-PC-DNN is the sum of the times taken for training of the individual models (since the architecture is run separately for each emotion unlike multi-task models which handle all emotions in one architecture).

6 Cognitive Implications of Emotions' Pairwise Similarity

To quantify various correlations that different pairs of emotions exhibit in social media microblogs, we perform two experiments. First, we train the best-performing LE-PC-DNN model on one emotion and test on another emotion (Table 4) (also explored in Mohammad and Bravo-Marquez (2017a)). Then, we train the model on the combined datasets of two emotions and see the performance on the test sets of individual emotions (Table 5) to further test how well different pairs of emotions complement each other.

Emotion i/j	anger	fear	joy	sadness
anger	0.766	0.447	-0.526	0.443
fear	0.533	0.791	-0.57	0.712
joy	-0.592	-0.387	0.803	-0.537
sadness	0.586	0.628	-0.567	0.803

Table 4: Pearson correlation scores (trained on emotion *j* (row); tested on emotion *i* (column)).

Emotion i/j	anger	fear	joy	sadness
anger	-	0.779	0.66	0.79
fear	0.73	-	0.687	0.802
joy	0.637	0.675	-	0.728
sadness	0.745	0.787	0.677	-

Table 5: Pearson correlation scores (trained on combined data of emotion *i*, *j*; tested on *i*).

6.1 Training on One Emotion and Testing on Another

Table 4 shows that there exists significant correlations between all the emotions, ranging from absolute Pearson scores of 0.387 (between fear and joy) to 0.803. This indicates that *all emotions are at least somewhat correlated*. This is backed in part by observations made in Wilson-Mendenhall et al. (2013), where the authors claim that 'all human emotions share core affective properties' (note that they relied on arousal and valence of emotions).

The negative emotions ('anger', 'fear' and 'sadness') are *positively correlated with each other* and *negatively correlated with 'joy'*. The *correlations are asymmetric*, *i.e.*, a pair of emotions does not show similar predictive power in either direction. For example, training on anger predicts intensity of sadness with 0.443 correlation with actual sadness intensities (a drop of about 0.36 Pearson score from when training happens on the sadness training set), whereas training on sadness predicts intensity of anger with 0.586 correlation (a drop of about 0.22 Pearson score). These last two observations match well with the results of the experiments carried out in Rutherford et al. (2008). In that work, visual aftereffects of certain emotional scenarios on the participants' faces were observed in various settings. They found that the results of their functional approach suggests (in their own language): "Negative emotions are many and specific. In contrast, positive emotions are few and less specific." The results further backed their hypothesis that "emotions share an asymmetric relationship as a group, in which numerous negative emotions together oppose the relatively small set of positive emotions". Schwartz and Weinberger (1980) also hint at the asymmetric nature of emotions (for fear and anger in particular).

6.2 Training on a Combined Dataset of Two Emotions and Testing on Both of them

Table 5 investigates the impact of combining the datasets of two emotions for training. All negative emotions (anger, fear and sadness) exhibit good performance in this scenario. The Pearson correlation suffers a drop of 0.02-0.03 in case of anger, and training on combined datasets of fear and sadness results in very similar performance for the individual emotions as compared to using just the individual emotion's dataset for training (the results in the diagonal of Table 4). The idea for using combined emotion datasets was motivated in Mohammad and Bravo-Marquez (2017a), who hinted at an increase in performance when combining fear and sadness training sets (but we do not see an increase, although the performance remains basically unaffected). High correlation among the negative emotions was also established in Barrett (2006) and Feldman (1993). However, the specific relationship between fear and sadness seems to be unexplored. It is possible that this relationship is a characteristic of the domain of Twitter tweets or our dataset. These observations comply with our LE-PC-DMTL-EI architecture (Figure 2b), where the emotion 'joy' is the first to get separated while the emotions 'fear' and 'sadness' are learned jointly for many layers in the architecture which gave optimal performance.

7 Error Analysis

We present some examples from the test dataset (Table 6) where the predictions were off by about 0.3 in intensity in either direction. Note that intensities are real-values scores between 0 and 1. To help future systems better predict emotion intensity in tweets, we present plausible explanations for the errors:

- **Incorrect Modeling of the Full Tweet Context**: It can become necessary to correctly model the overall context of the tweet. In Tweet #2, words like 'angry','anger' and 'hurt' indicate high intensity of anger. But the tweet is more of a moral preaching than an angry outburst, which explains an

#	Tweet	Emotion	Actual Intensity	Predicted Intensity
1	Oi @THEWIGGYMESS you've absolutely fucking killed me.. 30 mins later im still crying with laughter.. Grindah.. Grindah...hahahahahahaha	Joy	0.846	0.417
2	People are #hurt and #angry and it's hard to know what to do with that #anger Remember, at the end of the day, we're all #humans #bekind	Anger	0.25	0.580
3	T minus 10 hours till I meet with a designer who wants me to model his new fashion line !!!	Fear	0.667	0.286
4	Ibiza blues hitting me hard already wow	Sadness	0.833	0.533

Table 6: Examples of tweets with absolute difference in predicted and actual intensity greater than 0.3

actual anger intensity of 0.25. In Tweet #1, the latter half of the tweet makes it clear that 'absolutely fucking killed me' is a strong expression of joy and not some negative emotion.

- **Context Outside the tweet**: Tweet #3 requires common sense or world knowledge to know that an important meeting drawing near is generating anxiety with high intensity. This fear or anxiety is not obvious from the text itself which might explain the predicted intensity of 0.291 when the actual intensity is 0.667. Tweet #4 suffers from a similar problem. World knowledge is required to know the relationship between 'Ibiza blues' and sadness.

- **Metaphors**: Metaphorical expressions, like 'crying with laughter' in Tweet #1 or 'Ibiza blues' in Tweet #4, pose a significant challenge to computational models (Ghosh et al., 2015).

We carried out the above analysis after concluding all the experiments. The number of tweets where the predicted intensity was higher than actual and vice-versa were almost equal for any emotion. Hence, our system is not biased towards predicting higher or lower intensities than the annotated ones.

8 Conclusion & Future Work

We proposed various deep neural architectures for the task of emotion intensity prediction on micro-blogging data, specifically, Twitter data. This problem was recently brought to light by the EmoInt shared task (Mohammad and Bravo-Marquez, 2017b). Combining fully-connected layers with CNN in a parallel or non-sequential manner helped performance, and while we see such combinations in works tackling Vision tasks, the NLP community should also try this in their experiments. Transfer learning proved very effective in terms of performance since it helped us utilize the pre-trained activations of a CNN trained on more than a billion tweets for the related task of emoji detection in the same domain of Twitter tweets. These techniques when combined allow us to set the new state-of-the-art performance in emotion intensity detection, considerably outperforming all previous systems. We also proposed two deep neural network based multi-task learning methods with one of them learning to group similar emotions together using a heuristic based algorithm. The models compare well with the best performing system, and give significant gains in terms of computational complexity when compared to having a separate architecture for each emotion. Our multi-task model will be even more useful when the number of emotions are large. The optimized multi-task model and focused correlation tests brought out an especially strong correlation between 'fear' and 'sadness' for our dataset. Our error analysis on the test set using predictions from the best performing system will help future systems aiming to predict emotion intensity in tweets.

For future work, we would like to exploit topic modeling to find out correlations between certain topics and high intensity emotional outbursts, and to see if knowing the topic of a tweet can help with intensity prediction. We also plan to exhibit our models' effectiveness in real-world applications by using them, for example, to rank product reviews in terms of priority on basis of the emotion intensity. Another planned direction is to extend the four emotions used in this task to Ekman's six basic emotions (anger, happiness, surprise, disgust, sadness, and fear) (Ekman and Friesen, 1971).

References

Md Shad Akhtar, Palaash Sawant, Asif Ekbal, Jyoti Pawar, and Pushpak Bhattacharyya. 2017. Iitp at emoint-2017: Measuring intensity of emotions using sentence embeddings and optimized features. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 212–218.

Cecilia Ovesdotter Alm, Dan Roth, and Richard Sproat. 2005. Emotions from text: machine learning for text-based emotion prediction. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 579–586. Association for Computational Linguistics.

Saima Aman and Stan Szpakowicz. 2007. Identifying expressions of emotion in text. In *Text, speech and dialogue*, pages 196–205. Springer.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

Timothy Baldwin, Marie Catherine De Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text (WNUT 2015), Beijing, China*.

Lisa Feldman Barrett. 2006. Are emotions natural kinds? *Perspectives on psychological science*, 1(1):28–58.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Johan Bollen, Huina Mao, and Alberto Pepe. 2011. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. *ICWSM*, 11:450–453.

Michael Brooks, Katie Kuksenok, Megan K Torkildson, Daniel Perry, John J Robinson, Taylor J Scott, Ona Anicello, Ariana Zukowski, Paul Harris, and Cecilia R Aragon. 2013. Statistical affect detection in collaborative chat. In *Proceedings of the 2013 conference on Computer supported cooperative work*, pages 317–328. ACM.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, pages 95–133. Springer.

R Caruna. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Machine Learning: Proceedings of the Tenth International Conference*, pages 41–48.

Ruey-Cheng Chen, Evi Yulianti, Mark Sanderson, and W. Bruce Croft. 2017. On the benefit of incorporating external features in a neural architecture for answer sentence selection. In *Proceedings of SIGIR '17*, pages 1017–1020. ACM.

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

Sebastian Ebert, Ngoc Thang Vu, and Hinrich Schütze. 2015. A linguistically informed convolutional neural network. In *WASSA@ EMNLP*, pages 109–114.

Paul Ekman and Wallace V Friesen. 1971. Constants across cultures in the face and emotion. *Journal of personality and social psychology*, 17(2):124.

Weiguo Fan and Michael D Gordon. 2014. The power of social media analytics. *Communications of the ACM*, 57(6):74–81.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625.

Lisa A Feldman. 1993. Distinguishing depression and anxiety in self-report: Evidence from confirmatory factor analysis on nonclinical and clinical samples. *Journal of consulting and clinical psychology*, 61(4):631.

Aniruddha Ghosh, Guofu Li, Tony Veale, Paolo Rosso, Ekaterina Shutova, John Barnden, and Antonio Reyes. 2015. Semeval-2015 task 11: Sentiment analysis of figurative language in twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 470–478.

Zafar Gilani, Reza Farahbakhsh, and Jon Crowcroft. 2017. Do bots impact twitter activity? In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 781–782. International World Wide Web Conferences Steering Committee.

Fréderic Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van de Walle. 2015. Multimedia lab@ acl w-nut ner shared task: named entity recognition for twitter microposts using distributed word representations. *ACL-IJCNLP*, 2015:146–153.

Pranav Goel, Devang Kulshreshtha, Prayas Jain, and Kaushal Kumar Shukla. 2017. Prayas at emoint 2017: An ensemble of deep neural architectures for emotion intensity prediction in tweets. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 58–65.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE.

Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng, and Yifan Gong. 2013. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7304–7308. IEEE.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Maximilian Köper, Evgeny Kim, and Roman Klinger. 2017. Ims at emoint-2017: Emotion intensity prediction with affective norms, automatically extended resources and deep learning. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 50–57.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. In *Proceedings of NAACL-HLT*, pages 515–520.

Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. 2016. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. *arXiv preprint arXiv:1611.05377*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Saif M Mohammad and Felipe Bravo-Marquez. 2017a. Emotion intensities in tweets.

Saif M. Mohammad and Felipe Bravo-Marquez. 2017b. WASSA-2017 shared task on emotion intensity. In *Proceedings of the Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA)*, Copenhagen, Denmark.

Preslav Nakov, Sara Rosenthal, Svetlana Kiritchenko, Saif M Mohammad, Zornitsa Kozareva, Alan Ritter, Veselin Stoyanov, and Xiaodan Zhu. 2016. Developing a successful semeval task in sentiment analysis of twitter and other social media texts. *Language Resources and Evaluation*, 50(1):35–65.

Eunbyung Park, Xufeng Han, Tamara L Berg, and Alexander C Berg. 2016. Combining multiple sources of knowledge in deep cnns for action recognition. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–8. IEEE.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Christian Plahl, Michael Kozielski, Ralf Schlüter, and Hermann Ney. 2013. Feature combination and stacking of recurrent and non-recurrent neural networks for lvcsr. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6714–6718. IEEE.

MD Rutherford, Harnimrat Monica Chattha, and Kristen M Krysko. 2008. The use of aftereffects in the study of relationships among emotion categories. *Journal of Experimental Psychology: Human Perception and Performance*, 34(1):27.

Gary E Schwartz and Daniel A Weinberger. 1980. Patterns of emotional responses to affective situations: Relations among happiness, sadness, anger, fear, depression, and anxiety. *Motivation and emotion*, 4(2):175–191.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 70–74. Association for Computational Linguistics.

Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. 2017. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*.

Bo Wang, Maria Liakata, Arkaitz Zubiaga, Rob Procter, and Eric Jensen. 2016. Smile: Twitter emotion classification using domain adaptation. In *CEUR Workshop Proceedings*, volume 1619, pages 15–21. Sun SITE Central Europe.

Christine D Wilson-Mendenhall, Lisa Feldman Barrett, and Lawrence W Barsalou. 2013. Neural evidence that human emotions share core affective properties. *Psychological science*, 24(6):947–956.

Expressively vulgar: The socio-dynamics of vulgarity and its effects on sentiment analysis in social media

Isabel Cachola[*‡] **Eric Holgate**[*†] **Daniel Preoţiuc-Pietro**[◇] **Junyi Jessy Li**[†]
[‡]Department of Mathematics, [†]Department of Linguistics,
The University of Texas at Austin
{isabelcachola@,holgate@,jessy@austin.}utexas.edu
[◇]Computer and Information Science, University of Pennsylvania
danielpr@sas.upenn.edu

Abstract

Vulgarity is a common linguistic expression and is used to perform several linguistic functions. Understanding their usage can aid both linguistic and psychological phenomena as well as benefit downstream natural language processing applications such as sentiment analysis. This study performs a large-scale, data-driven empirical analysis of vulgar words using social media data. We analyze the socio-cultural and pragmatic aspects of vulgarity using tweets from users with known demographics. Further, we collect sentiment ratings for vulgar tweets to study the relationship between the use of vulgar words and perceived sentiment and show that explicitly modeling vulgar words can boost sentiment analysis performance.

1 Introduction

Vulgarity is common in language use, with vulgar words appearing with a frequency estimated between 0.5% and 0.7% in day-to-day conversational speech (Jay, 2009; Mehl et al., 2007) and 1.15% in Twitter (Wang et al., 2014). Vulgarity can employed for multiple goals: as an intensifier for subjective opinions, as a way to offend or express hate speech towards others, to describe vulgar activities or as a way to signal an informal conversation. Understanding the expression of vulgarity in naturally occurring text is thus important for several disciplines such as linguistics, which aims to better understand the pragmatics of vulgarity, for computer scientists which can explicitly model vulgarity in downstream NLP applications and for psychologists who study the socio-cultural factors of profanity.

Social media offers researchers access to vast volumes of naturally occurring user-generated content, with language use on social media containing a high level of expression of thoughts, opinions and emotions (Java et al., 2007; Kouloumpis et al., 2011). Furthermore, it is a platform for observing written interactions and conversations between users (Ritter et al., 2010). Thus, social media is a ideal medium to observe and study the expression of vulgar words and, in addition, allows us to study the socio-cultural context of this phenomenon.

Given the fact that most thoughts can be rephrased to not include vulgarity, the use of vulgar words indicates a purposeful attempt of performing a specific function. Table 1 includes examples of tweets containing vulgar words that perform different functions.

Vulgarity is often employed used to express emotion in language and can be used either to express negative sentiment or emotions or to intensify the sentiment present in the tweet (Wang, 2013). In one of the examples, 'I am stupid as f*ck' conveys more intense anger, while 'I am stupid' conveys a less emotional expression of irritation. Hence, understanding vulgar words is expected to have practical implications in sentiment analysis on social media. Furthermore, vulgar word usage is dependent on the user socio-cultural context with demographic and social information shown to improve sentiment analysis performance (Volkova et al., 2013; Yang and Eisenstein, 2017).

Hence, this study aims to address the following research questions:

- Is the expression of vulgarity and its function different across author demographic traits?

[*] Equal contribution.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2927–2938
Santa Fe, New Mexico, USA, August 20-26, 2018.

Tweet	Function
I have made the awful decision to stay up all night. Why? Because I'm stupid as fuck.	emphasize
It's annoying when a program does the "you sure you want to quit?" thing but I'd be lying if I said it hasn't saved my ass a number of times	neutral/idiomatic
U got some shitty weather Illinois.... I miss texas heat	express negative sentiment
<USER> I had that game it was the shit I believe it was something like baseball 2121	express positive sentiment

Table 1: Examples of tweets with vulgar words and their function.

- Does vulgarity impact perception of sentiment?
- Does modeling vulgarity explicitly help sentiment prediction?

To this end, we collect a new data set of 6.8K tweets labeled for sentiment on a five-point scale by nine annotators. We found that the expression of vulgarity interacts with many key demographic variables, including gender, age, education, income, religiosity and political ideology. Vulgarity is also used to convey different sentiment polarities by users that differ in age and religiosity. When studying the impact of vulgarity on sentiment perception, we show that vulgarity is most often used as an intensifier for the original sentiment. Finally, we show that explicitly informing sentiment analysis systems about vulgarity usage helps boost predictive performance.

This study aims to develop a quantitative understanding of the way in which social demographics interact with how vulgarity is actively and intentionally employed and show that this can be used to improve end-user applications such as sentiment analysis. We release our novel annotated corpus as well as the accompanying code to the community at `https://github.com/ericholgate/vulgartwitter`.

2 Related Work

Vulgarity The study of vulgar language – also referred to as profanity or use of swear/curse words despite the fact that they can be used for different goals – is of interest to linguists, psychologists and computer scientists. Vulgar words are very versatile, with a vulgar word being able to perform different interpersonal functions according to different contexts (Andersson and Trudgill, 1990). Several studies have examined these different uses and pragmatic functions. Pinker (2007) groups use of swear words into five roles: abusive (intention to offend or cause psychological harm), cathartic (used in response to pain), dysphemistic (intention to convey negative sentiment), emphatic (intention to draw attention) and idiomatic (used for no purpose or to signal informality). Wang (2013) identifies four pragmatic roles of swear words: express emotion, express emphasis (either in presence of additional sentiment or not), to signal group identity and for aggression – in this latter case, swearing is a face-threatening act (Brown and Levinson, 1987). Overall, the function and effects of vulgar word usage are highly dependent on the context and the social factors (e.g., inter-personal relationships, setting), which we aim to further study and capitalize on in this study.

Recently, there has been a surge in interest in studying hate speech online (Warner and Hirschberg, 2012). A better understanding of vulgar words and the context they are used in can aid models that aim to automatically detect and categorize hate speech, which is harder to distinguish when profanity is employed (Malmasi and Zampieri, 2018).

Vulgar words have been quantitatively studied in social media and online communities. Liu et al. (2010) identified and filtered vulgar content distributed via peer-to-peer applications. Wang et al. (2014) presents a quantitative analysis of the frequency of curse word usage on Twitter, their variation with time and location, their gender usage and use in conversations across genders. An analysis of swear word usage across gender and age on Twitter was performed in Gauthier et al. (2015). Vulgar words are obfuscated in mediums where messages are censored or for other goals, and an analysis of these techniques is presented in Laboreiro and Oliveira (2014). Our paper expands the scope of this research by studying several other demographic traits beyond gender, aiming to identify different uses of swear

words through eliciting sentiment annotations and uses these to improve sentiment prediction methods.

Sentiment analysis Sentiment analysis on Twitter is a very popular research topic, anchored in a wide range of industry applications and with several shared tasks over the past years (Rosenthal et al., 2017). Several studies have looked at the effects of intensifiers and words that switch their polarity in sentiment dictionaries (Flekova et al., 2015). Vulgar words are commonly part of popular sentiment and emotion analysis lexicons (Mohammad and Turney, 2013).

Demographics & Vulgarity Use of vulgar or swear words is influenced by pragmatic or contextual factors such as the inter-personal relationship between the speakers or their social factors such as gender, occupation or social status (Jay and Janschewitz, 2008). The most studied social factor was gender, with several studies finding that males employ profanity much more often than females (Selnow, 1985; Wang et al., 2014). Other social factors such as age, religiosity or social status were found to be related with the rate of using vulgar words (McEnery, 2004). In studying the task of user trait prediction from social media texts, several vulgar words were shown to be characteristic of demographic traits such as political ideology (Preoţiuc-Pietro et al., 2017).

Explicitly modeling demographic (Lynn et al., 2017) and other social factors such as community membership (Yang and Eisenstein, 2017) have to date been successfully used for improving accuracy of several tasks including sentiment analysis (Volkova et al., 2013) or document classification (Hovy, 2015).

3 Data

In exploring the role of vulgarity in written interactions, social media and Twitter in particular stands out as an ideal medium for collecting data. Twitter provides vast volumes of naturally occurring text, which are less affected by formal and curated text such as newswire, and can be studied together with socio-demographic attributes of their authors.

Data Collection We collect a novel corpus of tweets, all of which contain vulgar words, and annotate this for sentiment, as to the best of our knowledge, no such corpus focusing on vulgar expressions exists in past research. This data is used in the sentiment analysis section of this study.

In order to enable the analysis of demographics and vulgarity in the analysis section of this study, the above tweets were deliberately sampled from those posted by a set of 4,132 users with known socio-demographic traits obtained through asking the users to self-report them in an online survey. For the demographic analysis, we have downloaded and used all most recent 3,200 tweets (per Twitter API limitations) from these users. This data set was used in prior research and for a full description of the data collection process and quality control processes, we refer the interested reader to Preoţiuc-Pietro et al. (2017).

Demographic Variables and Coding The Twitter users self-reported through a survey the following demographic variables: gender, age, education level, annual income level, faith and political ideology. All users solicited for data collection were from the United States in order to limit cultural variation.

- Gender was considered as binary[2] and coded with Female – 1 and Male – 0. All other variables are treated as ordinal variables.
- Age is represented as a integer value in the 13–90 year old interval.
- Education level is coded as an ordinal variable with 6 values representing the highest degree obtained, with the lowest being 'No high school degree' (coded as 1) and the highest being 'Advanced Degree (e.g. PhD)' (coded as 6).
- Income level is coded as on ordinal variable with 8 values representing the annual income of the person, randing from '<20,000$' to '>200,000$').
- Faith is coded as a ordinal variable with 6 levels, representing the average number a time a user attends religious service. The answers range from 'Never' (coded as 1) through to 'Multiple times a week' (coded as 6).

[2]We asked users to report gender as either 'Female', 'Male' or an open-ended field, and removed the few users which did not select 'Male' or 'Female'

- Political ideology is measured on the conservative–liberal spectrum, the common way of representing ideology in the US (Ellis and Stimson, 2012). Ideology option are: Very conservative (1), Conservative (2), Moderately conservative (3), Moderate (4), Moderately liberal (5), Liberal (6), Very liberal (7); Other (8) and Apathetic (9). In order to convert this to a ordinal scale, we only perform political ideology analysis with users with a ideology in the set 1, 2, 3, 5, 6, 7 (1290 users removed in total, similar to (Preoţiuc-Pietro et al., 2017)).

Identifying vulgar tweets In order to identify tweets containing vulgarity, we start with the vulgarity lexicon available at `www.noswearing.com`. This lexicon comprises 349 items including general vulgarities, slurs, and sex-related terms.[3] We manually removed a total of 82 items from this list that were deemed not to be unambiguously vulgar.[4] Additionally, we employed a manually crafted list of regular expressions to identify common intentional spelling variations in vulgar terms (e.g., vowel reduplication such as *fuuuuu*k* or sibilant reduplication such as *asssss*).

We identified 261,592 tweets containing at least one vulgar word, from 3,626 users featuring 222 unique vulgar tokens, many of which are either compounds of vulgar tokens with non-vulgar words or are morphological derivations (e.g., *assbag* or *f*ck* vs. *f*cking*). Of these, 149 words occur in the data less than 100 times, while the top 20 most frequent words account for 90% of total vulgar word occurrences. The median frequency of the vulgarity lexicon is 27 in our larger data set. These results are in line with previous analyses of vulgar word frequencies and their distribution (Wang et al., 2014).

Data Processing To preserve anonymity, we replace URL's by a <URL> token and usernames are masked by a <USER> token. Further, punctuation is removed and all words are lowercased.

Sentiment Annotation Human annotations of sentiment were solicited for 7,800 vulgar tweets (excluding retweets) via Amazon Mechanical Turk (MTurk) platform. These tweets were sampled uniformly from a randomly sampled set of 2,000 users in our corpus that posted at least one vulgar tweet. We perform this sampling of users in order to have a broad range of users, but have more than one sample tweet for each user.

The task setup and guidelines follow the settings of SemEval 2016 Task 4 subtask C (Nakov et al., 2016). Annotators evaluated tweet sentiment on a five-point scale: (1) *very negative*, (2) *somewhat negative*, (3) *neutral*, (4) *somewhat positive*, and (5) *very positive*. The numeric coding for this scale follows a gradual movement from more negative to more positive, allowing sentiment to be treated as a continuous variable. A sixth option, *not applicable*, was available to workers to cover instances in which there was missing contextual information or a tweet was illegible. Each tweet was annotated by nine different and independent annotators to allow for sufficient data to generalize across individual sentiment perception boundaries.[5]

Annotation results were checked for inter-annotator agreement by comparing individual user responses to the average annotation of their peers across all tweets that they annotated (Li et al., 2016). Annotations from users with a Spearman correlation coefficient less than 0.3 were removed from computing consensus labels. Tweets with less than five remaining annotations were excluded. We also excluded tweets whose majority ratings are *not applicable*. This resulted in a final data set consisting of 6,791 annotated vulgar tweets. Following the procedure for SemEval, Sentiment annotations for these tweets were consolidated by majority vote. If a majority rating was not present, the consolidated score was set to the mean of all ratings (ratings of *not applicable* were excluded).

The consolidated sentiment ratings were distributed as follows: 1.94% *very positive*, 14.57% *somewhat positive*, 26.51% *neutral*, 46.62% *somewhat negative*, 10.14% *very negative*.

[3]While this initial method we use is not fully accurate (Sood et al., 2012a), it reaches high accuracy levels of more than 90% (Sood et al., 2012b)

[4]These terms were largely anatomical words or general verbs like *penis*, *vagina*, and *blow*, but some identity descriptors like *gay*, *queer*, and *lesbian* were also excluded after manual review of a large sampling of uses revealed they were not overwhelmingly employed as slurs.

[5]We asserted the following qualifications on MTurk: locale=US, approval rate >90%, number of HITs approved >100.

Demographic Trait	Pearson r	p-value
Gender	-0.077	1.61^{-4}
Age	-0.233	6.64^{-31}
Education	-0.100	7.62^{-07}
Income	-0.087	1.73^{-05}
Faith	-0.187	2.74^{-20}
Political Ideology	0.124	8.69^{-10}

Table 2: Relationship between user demographic traits and percentage of vulgar tweets of total tweets sent by the user. Results are computed using Pearson correlation (point-biserial correlation for gender), with gender and age as controls in partial correlation.

4 Analysis

4.1 Demographics & Frequency of Vulgarity

We first analyze the differences in frequency of employing vulgarity and how this relates to demographic traits. We use partial Pearson correlation where the dependent variable is the fraction of vulgar tweets from the total number of tweets sent by a user. For all analyses, we consider gender and age basic traits and control for data skew by introducing both variables as controls in partial correlation, as done in prior work (Schwartz et al., 2013; Preoţiuc-Pietro et al., 2016). When studying age and gender, we use only the other trait as the control. We have experimented with percentage of vulgar words as an alternative outcome, but the results were very similar, hence we exclude them for brevity. We also experimented with log-scaling the age variable, but found no major differences.

Gender results show that females are less likely to post vulgar tweets than males. Several previous studies have reported a similar effect, in written text in the BNC (McEnery, 2004), on social networks (Wang et al., 2014), speech (Jay, 1980; Jay, 1992) or as self-reported using questionnaires (Selnow, 1985; Pilotti et al., 2012). In our data, the average vulgar tweet percentages per user are 3.332% for males and 3.060% for females.

Age exhibits the largest effect on posting vulgar tweets, with younger users much more likely than older to post, which aligns to analysis of speech (Jay, 1992) and tweets as previously performed in Gauthier et al. (2015).

Both higher education and income are anti-correlated with usage of vulgarity on social media even when controlling for gender, with education being slightly more strongly associated with lack of vulgarity than income. Previously, McEnery (2004) suggested that social rank, which is related to both education and income, is anti-correlated to use of swear words. In a study on perceptions of user traits from tweets, raters that were exposed to vulgar words were more likely to consider the user as being less educated than in reality (Carpenter et al., 2016).

Faith is anti-correlated with use of vulgar words with the second strongest effect, a correlation also previously identified by Jay (1992).

Finally, liberal users are more likely to use vulgarity on social media, an association on Twitter also uncovered by Sylwester and Purver (2015) and Preoţiuc-Pietro et al. (2017).

4.2 Demographics & Sentiment Perception

Next, we analyze the relation between the demographic traits of the tweet authors to the perceived (annotated) sentiment of the tweet. We perform this analysis in order to uncover potentially different types of uses of vulgar words. If vulgar words are used as an intensifier, they are used with both positive and negative sentiment, while if expressing emotion they are expressed mainly with negative sentiment. If vulgar words are used in neutral tweets, then their role is more likely to be idiomatic.

We test this using partial Pearson correlation with user demographics as the independent variable and with the polarity of the rating (positive, negative or neutral) as a dependent indicator variable. Again, we consider gender and age as basic demographic traits and introduce these variables as controls. Similarly, for age and gender analysis, we the other basic trait as the only control. The results of these analyses are

	Positive vs. All		Negative vs. All		Neutral vs. All	
Demographic Trait	r	p	r	p	r	p
Gender	0.024	0.334	-0.006	0.797	-0.030	0.222
Age	-0.107	$\mathbf{2.11}^{-5}$	0.061	**0.015**	0.093	$\mathbf{2.305}^{-4}$
Education	-0.049	0.052	-0.018	0.470	0.076	**0.002**
Income	-0.024	0.324	0.016	0.524	0.001	0.966
Faith	-0.108	$\mathbf{1.640}^{-5}$	0.008	0.741	0.128	$\mathbf{3.539}^{-7}$
Political Ideology	-0.032	0.276	0.057	0.051	0.008	0.765

Table 3: Relationship between user demographic traits and perceived sentiment rating by MTurk workers. Results are computed once more using Pearson correlation (point-biserial correlation for gender), with gender and age as controls in partial correlation.

presented in Table 3.

Positive Sentiment With positive sentiment as an indicator variable, we see that both age and faith have significant correlations. This indicates that younger and less religious users are more likely to express vulgar words in presence of positive sentiment, which represents use of vulgarity as an intensifier for sentiment.

Negative Sentiment With negative sentiment as an indicator variable, we again see a weaker (yet still significant) correlation with age. This shows that older users are prefer to use vulgar words to deliberately express negative emotions and sentiment, with other functions being less frequent.

Neutral Sentiment Finally, vulgar words are present with neutral sentiment when this is used idiomatically for signaling informality or other functions. Neutral sentiment is most often employed by religious, older and more educated users. The latter correlation is especially relevant, as higher education leads to better social adaptation.

Overall, gender, income and political ideology exhibit no significant correlations with sentiment ratings of vulgar tweets.

4.3 Vulgar vs. Censored Tweet Sentiment Perception

To test whether vulgarity impacts the *perceived* sentiment of tweets, we removed the vulgar terms in the tweets from the vulgar corpus, arriving at an *original-censored* parallel dataset. We then crowdsourced the sentiment judgments of the censored tweets using the same method as before.

Among the censored tweets, annotators indicated that 355 —around 5%—of the tweets were rated by more than 5 (out of 9) workers to be unintelligible after censorship. The distribution of sentiment in this subset was originally 21.40% positive, 21.97% neutral, 56.62% negative. This loss of intelligibility is expected and an additional indication that vulgarity bears semantic content and by censoring these words, the resulting tweet looses coherence.

To determine the extent to which the use of vulgar terms impacts the perception of tweet sentiment, we calculated the direction and magnitude of the change in sentiment between the vulgar and censored pairs. For original tweets that are not neutral, we consider three situations: (a) **intensify**, where the magnitude of the sentiment towards a tweet intensifies after censorship, e.g. with score changes from 2 to 1 (for negative tweets) or from 4 to 5 (for positive tweets); (b) **weaken**, where the magnitude changes in the other direction, e.g., with score changes from 1 to 2 (for negative tweets) and from 5 to 4 (for positive tweets); (c) **flip**, where the censored sentiment is opposite from the original sentiment, i.e., negative becomes positive or positive becomes negative. For neutral tweets, we consider situations where the censored sentiment becomes positive (**to-pos**), or negative (**to-neg**), or stays neutral (**same**).

In Table 4, we present the numbers and magnitudes of changes observed in these 6,436 tweets (this number reflects the size of the censored corpus after excluding the 593 which were rated unintelligible and those which did not have at least five ratings after removing annotators with low inter-annotator agreement scores). Clearly, for original positive or negative tweets, censoring has a significant weakening

	Positive			Negative			Neutral		
	intensify	weaken	flip	intensify	weaken	flip	to-pos	to-neg	same
% changed	15.17	34.09	3.69	5.53	40.34	7.44	24.07	19.34	56.59
mean $\|\Delta\|$	0.591	0.785	1.597	0.521	0.936	1.569	0.817	0.825	—

Table 4: Change in perceived sentiment score with censored vulgarity. Results are grouped by initial perceived sentiment polarity, scored from 1 (*very negative*) to 5 (*very positive*). The *intensify* subcondition indicates that the censored tweet had the same polarity, but with greater magnitude (i.e., original 2, censored 1; or original 4, censored 5). The *weaken* subcondition describes the opposite effect. The *flip* subcondition indicates that the polarity of the perceived sentiment inverted. All of the changes are statistically significant ($p < 0.01$)

effect for sentiment. Censored tweets that are originally negative also have a stronger tendency than originally positive ones to flip sentiment. For neutral tweets, most of the time the sentiment does not change.

Kruskal-Wallis H-tests were conducted to determine if changes in sentiment rating were significant, and all results were shown to be statistically significant ($p < 0.01$). The strongest effect was seen in the *to-pos* and *negative-flip* conditions, indicating that vulgarity is frequently employed by users to emphasize or introduce a negative sentiment.

About half of the tweets in the corpus did not exhibit a change in sentiment. Of those that didn't change, 21.31% were positive, 21.11% were neutral, and 56.14% were negative. Vulgar tweets which were initially positive or negative were slightly more likely to exhibit some change in perceived sentiment rating than not, while vulgar tweets that were initially neutral were 8% more likel y to exhibit no change at all than they are to exhibit change.

5 Modeling Vulgarity

We now consider whether explicitly inserting knowledge about vulgarity into sentiment models will help sentiment prediction. To do this, we build on top of a base model that has been shown to be effective in Twitter sentiment prediction (Rosenthal et al., 2017; Cliche, 2017), and propose three ways of inserting vulgar features into the system.

Base architecture The basis of our system is a bidirectional Long-Short Term Memory Network (LSTM) (Hochreiter and Schmidhuber, 1997), which utilizes memory cells to capture long-term dependencies. This architecture is similar to the LSTM models in BB_twtr (Cliche, 2017), the best system participated in SemEval 2017 task 4 subtask C (Rosenthal et al., 2017).

For a given tweet of n words, we encode each word w_t into an embedding vector x_t; we then pass the embedding matrix to a bidirectional LSTM, arriving at a sentence representation where each hidden state h_t for word w_t encodes the context from both words before w_t (i.e., w_1 to w_t) and after w_t (i.e., w_t to w_n). The hidden state is the concatenation of two directions: $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$.

The bidirectional LSTM sentence representation is then passed into a dense layer with a RELU activation function $RELU(x) = max(0, x)$. The dense layer is then followed by a dropout layer, and a softmax classification layer.

Vulgarity features We propose and compare three different ways to account for vulgarity:

- **Masking**: Mask all vulgar words with a vulgar token, e.g., *This is the <VG>*. This method essentially considers all vulgar words to be the same token and informs the model of such, but removes lexical variances and different expressions of vulgarity.
- **Token insertion**: Insert a vulgar token after each vulgar word, e.g, *This is the shit <VG>*. This method allows the curse word to retain its original meaning, while giving it the same representation as other curse words.

System	Overall	Very positive	Positive	Neutral	Negative	Very negative
Bi-LSTM	0.791	3.000	1.978	1.008	0.049	1.084
+ Masking	0.898	**2.000**	**1.030**	**0.028**	1.000	1.979
+ Token Insertion	0.761	3.000	2.000	0.996	**0.002**	**1.000**
+ Concatenation	**0.759**	3.000	1.993	0.992	**0.002**	**1.000**

Table 5: MAE for each system on Vulgar-Tweet dataset. The best per-class results are significantly better than the baseline Bi-LSTM with the Wilcoxon signed-rank test.

- **Concatenation**: Count the number of vulgar words n_v in a tweet, and use it as a feature in concatenation with the tweet representation from Bi-LSTM h_t, before feeding it to the next layer, whose new input would be $[h_t, n_v]$. Effectively, this informs the classifier of the number of vulgar words, but removes contextual information of where a curse word appears.

6 Experiments

The goal of our evaluation is to assess how explicit knowledge of vulgarity can help in sentiment prediction. Hence we focus on tweets with vulgarity present, and we train and evaluate on our Vulgar-Tweet dataset. We further compare the aforementioned three ways to encode vulgarity features.

6.1 Systems and training

Systems We train and compare the following systems: *Bi-LSTM*, i.e., the base architecture without explicit knowledge about vulgarity; and the base architecture with each of the vulgarity features stated: *Bi-LSTM + Masking*, *Bi-LSTM + Token insertion*, and *Bi-LSTM + Concatenation*.

Data We develop, train and evaluate our system on our Vulgar-Tweet dataset, from which we carved out a development set of 500 tweets, a test set of 1,000 tweets. The remaining 5,291 tweets were used for training.

It is worth noting that the SemEval data is not viable for evaluating the effects of modeling vulgarity. There are only 1,235 (4%) of vulgar tweets in the 2016 data, and 339 (2.75%) in the 2017 testing data. For the sake of comparison, we also train a system with SemEval 2016 training data, and report on the 2017 test set where there is at least one vulgar word in each tweet.

Settings We used 200 dimensional embeddings trained on a corpus of 50 million tweets (Astudillo et al., 2015). We do not tune the embeddings. Embeddings for unknown words are randomly initialized. The LSTM dimension is 128 and we use a batch size of 256. We use accuracy as our metric and optimize using the Adam optimizer (Kingma and Ba, 2014). The dropout layer uses a dropout rate of 0.4. We train until the validation accuracy stops improving with a learning rate of 0.001. All hyperparameters are tuned on the SemEval development set. The model is implemented in Keras (Chollet and others, 2015).

6.2 Evaluation

Metrics Since our data is annotated on a 5-point ordinal scale, we used mean absolute error (MAE), which asserts more penalty when the predicted label is further away from the true label, i.e., if the system predicts 1, and the true label is -2, the error will be 3 (instead of just "incorrect"). On a test set Te where the true class of tweet i is c_i, the mean absolute error (MAE) is:

$$MAE = \frac{1}{|Te|} \sum_{x_i \in Te} |y_{pred} - y_{true}| \quad (1)$$

We report the overall MAE as well as the MAE values for each sentiment class.

Results Table 5 tabulates the MAE values for each system. Due to the fact that our data set is largely negative, our systems in general are better at predicting negative and very negative tweets. Except masking, vulgarity features improves the overall performance of the system. Concatenating a vulgar token

Bi-LSTM	+ Masking	+ Token Insertion	+ Concatenation
1.320	1.666	**1.068**	1.148

Table 6: MAE on SemEval 2017 task 4 subtask C test set, vulgar tweets only.

Text	**True Label**	**Baseline Prediction**	**Insertion Prediction**	**Concatenation Prediction**
the bitch is back jamies new tumblr page	Negative	Neutral	Negative	Negative
welcome to my personal hell	Negative	Neutral	Negative	Negative
the simpsons donut hell apptrailers 0 likes	Neutral	Neutral	Negative	Negative
so fucking excited	Very Positive	Neutral	Negative	Negative

Table 7: Example Tweets and their predictions.

produced the lowest MAE overall, followed by insertion. The best per-class results are significantly better than the baseline Bi-LSTM with the Wilcoxon signed-rank test.

Across sentiment labels, masking improves the prediction of positive tweets; however, the prediction of negative tweets suffer. With masking, the actual vulgar word is replaced by a special token, stripping its meaning. One reason for this behavior could be that for positive tweets, vulgarity is more often used as an intensifier than other sentiment classes (Table 4, positive-weaken), so there is less of an impact on the loss of meaning, vs. the benefit of explicit vulgarity information. Token insertion and concatenation improves the prediction of negative tweets, and the prediction of non-negative tweets remain stable. This shows that retaining the meaning of vulgar expressions is in general more helpful and especially for negative sentiment.

Finally, we report on systems trained on SemEval and tested on the 339 vulgar-only tweets in the SemEval test set, shown in Table 6. Again, we see that inserting and concatenating vulgar word count both improves performance.[6]

Qualitative Analysis In Table 7, we show example tweets along with their labels and predictions. The first two are correct predictions, and the two others are incorrect predictions.

In the first tweet, "is back" is often used in positive sentences. For example, the tweet "pretty little liars is back in action hell yes" is rated very positive. However, "bitch" is often used as a negative word, making the baseline predict the sentence is neutral. The addition of vulgar features gave greater significance to the word "bitch," pulling the sentiment in the negative direction. The baseline may have rated the second tweet as neutral because the combination of the positive word "welcome" and the negative word "hell" gives the tweet a neutral prediction. Again, the vulgarity features give "hell" more significance, allowing the system to pick up on the negativity of the tweet.

In the third sentence, the presence of the word "hell" pushed the prediction to be negative, although in this case it made the prediction incorrect. In the last sentence, the systems with vulgar features predicted negative although the sentence is very positive. The baseline predicted negative, indicating that it did not have enough power to counterbalance "fucking" and "excited". However, our systems pulled the sentiment in the wrong direction. This may also be because vulgarity is often used in a negative form in this context (e.g. "so fucking angry").

7 Conclusions

We have introduced a new data set of 6.8K vulgar tweets labeled for sentiment on a five-point scale by nine annotators. Analysis of the ratings revealed that the expressiveness of vulgarity interacts with a number of demographic features, including age, gender, education, income, religiosity, and political ideology. Vulgarity is used to different ends by users who differ in age and faith. An examination of the impact of vulgarity on tweet sentiment perception showed that, in cases where the presence of vulgarity

[6]The MAE of Bi-LSTM on all the SemEval 2017 test set is 0.52, comparable with submitted systems (Rosenthal et al., 2017).

influences sentiment perception, it is most often used to intensify existing sentiment. Finally, we have shown that utilizing vulgarity-centric features yields increased sentiment analysis system performance. Future work will look directly at modeling the functions of vulgar words.

References

Lars-Gunnar Andersson and Peter Trudgill. 1990. *Bad language*. Penguin.

Ramón Astudillo, Silvio Amir, Wang Ling, Mário Silva, and Isabel Trancoso. 2015. Learning word representations from scarce and noisy data with embedding subspaces. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1074–1084.

Penelope Brown and Stephen C Levinson. 1987. *Politeness: Some Universals in Language Usage*, volume 4. Cambridge University Press.

Jordan Carpenter, Daniel Preoţiuc-Pietro, Lucie Flekova, Salvatore Giorgi, Courtney Hagan, Margaret Kern, Anneke Buffone, Lyle Ungar, and Martin Seligman. 2016. Real Men don't say 'cute': Using Automatic Language Analysis to Isolate Inaccurate Aspects of Stereotypes. *Social Psychological and Personality Science*, 8.

François Chollet et al. 2015. Keras. `https://github.com/keras-team/keras`.

Mathieu Cliche. 2017. BB_twtr at SemEval-2017 Task 4: Twitter Sentiment Analysis with CNNs and LSTMs. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, pages 573–580.

Christopher Ellis and James A Stimson. 2012. *Ideology in America*. Cambridge University Press.

Lucie Flekova, Eugen Ruppert, and Daniel Preoţiuc-Pietro. 2015. Analysing domain suitability of a sentiment lexicon by identifying distributionally bipolar words. In *Proceedings of the Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA)*.

Michael Gauthier, Adrien Guille, A Deseille, and Fabien Rico. 2015. Text mining and twitter to analyze british swearing habits. *Handbook of Twitter for Research*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 752–762.

Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. 2007. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 56–65.

Timothy Jay and Kristin Janschewitz. 2008. The pragmatics of swearing. *Journal of Politeness Research. Language, Behaviour, Culture*, 4(2):267–288.

Timothy B Jay. 1980. Sex roles and dirty word usage: A review of the literature and a reply to Haas. *Psychological Bulletin*, 88:614–621.

Timothy Jay. 1992. *Cursing in America: A Psycholinguistic Study of Dirty Language in the Courts, in the Movies, in the Schoolyards, and on the Streets*. John Benjamins Publishing.

Timothy Jay. 2009. The utility and ubiquity of taboo words. *Perspectives on Psychological Science*, 4(2):153–161.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations*.

Efthymios Kouloumpis, Theresa Wilson, and Johanna D Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, pages 538–541.

Gustavo Laboreiro and Eugénio Oliveira. 2014. What we can learn from looking at profanity. In *International Conference on Computational Processing of the Portuguese Language*, pages 108–113.

Junyi Jessy Li, Bridget O'Daniel, Yi Wu, Wenli Zhao, and Ani Nenkova. 2016. Improving the annotation of sentence specificity. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*.

Xiangtao Liu, Xueqi Cheng, Jingyuan Li, Haijun Zhai, and Shuo Bai. 2010. Identifying vulgar content in emule network through text classification. In *IEEE International Conference on Intelligence and Security Informatics*, pages 168–168.

Veronica Lynn, Youngseo Son, Vivek Kulkarni, Niranjan Balasubramanian, and H Andrew Schwartz. 2017. Human centered nlp with user-factor adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1146–1155.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30(2):187–202.

Tony McEnery. 2004. *Swearing in English: Bad language, purity and power from 1586 to the present*. Routledge.

Matthias R Mehl, Simine Vazire, Nairán Ramírez-Esparza, Richard B Slatcher, and James W Pennebaker. 2007. Are women really more talkative than men? *Science*, 317(5834):82–82.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a Word-Emotion Association Lexicon. *Computational Intelligence*, 29(3):436–465.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, pages 1–18.

Maura Pilotti, Jennifer Almand, Salif Mahamane, and Melanie Martinez. 2012. Taboo words in expressive language: Do sex and primary language matter. *American International Journal of Contemporary Research*, 2(2):17–26.

Steven Pinker. 2007. *The stuff of thought: Language as a window into human nature*. Penguin.

Daniel Preoţiuc-Pietro, Jordan Carpenter, Salvatore Giorgi, and Lyle Ungar. 2016. Studying the Dark Triad of Personality using Twitter Behavior. In *Proceedings of the 25th ACM Conference on Information and Knowledge Management*, CIKM, pages 761–770.

Daniel Preoţiuc-Pietro, Ye Liu, Daniel Hopkins, and Lyle Ungar. 2017. Beyond binary labels: political ideology prediction of twitter users. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 729–740.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, pages 502–518.

H. Andrew Schwartz, Johannes C. Eichstaedt, Margaret L. Kern, Lukasz Dziurzynski, Stephanie M. Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin E. P. Seligman, and Lyle H. Ungar. 2013. Personality, Gender, and Age in the Language of Social Media: The Open-vocabulary Approach. *PLoS ONE*, 8(9):1–16.

Gary W Selnow. 1985. Sex differences in uses and perceptions of profanity. *Sex Roles*, 12(3-4):303–312.

Sara Sood, Judd Antin, and Elizabeth Churchill. 2012a. Profanity use in online communities. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1481–1490.

Sara Owsley Sood, Judd Antin, and Elizabeth F Churchill. 2012b. Using crowdsourcing to improve profanity detection. In *AAAI Spring Symposium: Wisdom of the Crowd*, volume 12, page 06.

Karolina Sylwester and Matthew Purver. 2015. Twitter Language Use Reflects Psychological Differences between Democrats and Republicans. *PLoS ONE*, 10(9), 09.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring demographic language variations to improve multilingual sentiment analysis in social media. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1815–1827.

Wenbo Wang, Lu Chen, Krishnaprasad Thirunarayan, and Amit P Sheth. 2014. Cursing in English on Twitter. In *Proceedings of the 17th ACM conference on Computer Supported Cooperative Work & Social Computing*, pages 415–425.

Na Wang. 2013. An analysis of the pragmatic functions of swearing in interpersonal talk. *Griffith Working Papers in Pragmatics and Intercultural Communication*, 6:71–79.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26.

Yi Yang and Jacob Eisenstein. 2017. Overcoming language variation in sentiment analysis with social attention. *Transactions of the Association for Computational Linguistics*, pages 295–307.

Clausal Modifiers in the Grammar Matrix

Kristen Howell
University of Washinton
Department of Linguistics
kphowell@uw.edu

Olga Zamaraeva
University of Washington
Department of Linguistics
olzama@uw.edu

Abstract

We extend the coverage of an existing grammar customization system to clausal modifiers, also referred to as adverbial clauses. We present an analysis, taking a typologically-driven approach to account for this phenomenon across the world's languages, which we implement in the Grammar Matrix customization system (Bender et al., 2002, 2010). Testing our analysis on testsuites from five genetically and geographically diverse languages that were not considered in development, we achieve 88.4% coverage and 1.5% overgeneration.

Title and Abstract in Russian

Обстоятельственные придаточные в системе LinGO Grammar Matrix

В данной работе мы представляем новую библиотеку для системы LinGO Grammar Matrix. LinGO Grammar Matrix—это платформа для автоматического построения грамматик, где под грамматикой подразумевается программа, которая принимает на вход строку и возвращает соответствующие синтаксическую и семантическую структуру, построенные на основе синтаксической теории HPSG и семантического формализма MRS. Пользователь системы заполняет типологический опросник (отвечает на вопросы о конкретном языке, для которого требуется построить грамматику) и автоматически получает спецификацию, которую затем можно загрузить в различные приложения и получить собственно программу (грамматику). Ранее LinGO Grammar Matrix не поддерживала никаких придаточных предложений; в данной работе мы описываем новую библиотеку для обстоятельственных придаточных. Библиотека включает в себя новую страницу типологического опросника, набор признаковых структур HPSG, которые, с учетом уже имеющихся, позволяют составить корректные грамматики с обстоятельственными придаточными, и, наконец, собственно логику программы, которая и составляет эти грамматики в ответ на конкретный запрос пользователя. В процессе разработки мы проверяем, что система работает корректно в отношении списка возможных (coverage; охват грамматики) и невозможных (overgeneration; избыточные порождения грамматики) предложений из некоторого набора релевантных искусственных псевдоязыков (которые мы определяем как комбинации возможных выборов пользователя), а также из четырех естественных языков, отобранных из типологической литературы, посвященной обстоятельственным придаточным. Для оценки качества нашей библиотеки мы смотрим на охват и избыточные порождения грамматик, полученных нами для пяти естественных языков из разных языковых семей, отобранных уже после окончания разработки.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2939–2952
Santa Fe, New Mexico, USA, August 20-26, 2018.

1 Introduction

The value of large-scale implemented precision grammars to linguists is twofold. First, by including analyses for linguistic phenomena that interact with each other, they are useful for verifying the consistency of linguistic theories (Bender, 2008; Müller, 2015); and second, they facilitate linguistic hypothesis testing by allowing the comparison of multiple analyses for a single phenomenon (Bender, 2010; Fokkens, 2014). Precision grammars parse and generate grammatical strings, emphasizing the linguistic correctness of an analysis by prioritizing *precision* (the number of inputs correctly parsed) over *recall* (the total number of inputs parsed), and in doing so allow linguists to test hypotheses over corpora. While the development of such grammars is time consuming, precision grammar starter kits can speed up the process by using stored analyses to create customized grammars (Bender et al., 2002).

Our present work is situated within the LinGO Grammar Matrix, a precision grammar customization system in which users answer typological questions about their language and a precision grammar fragment in the Head-driven Phrase Structure Grammar formalism (HPSG; Pollard and Sag, 1994) is produced. These starter grammars have given rise to such broad coverage grammars as the Spanish Resource Grammar (Marimon, 2010). We build on previous work in the Grammar Matrix by Trimble (2014), Poulson (2011) and others, following the methodology set forth by Drellishak (2009).

While language-specific analyses for clausal modifiers exist in the English Resource Grammar (ERG; Flickinger, 2000, 2011), the Jacy Grammar of Japanese (Siegel et al., 2016) and the Spanish Resource Grammar (Marimon, 2010), we are not aware of an implemented cross-linguistic analysis that has been applied to a broad range of typologically diverse languages. As subordinate clauses of any kind were not previously supported by the Grammar Matrix customization system and they are common in natural speech, their addition will extend the coverage of grammar fragments produced by this system. Restricting our focus to subordinate clauses that modify verbal projections, we present a library for clausal modifiers (or adverbial clauses) that supports numerous subordination strategies.

Languages typically employ multiple clausal modifier strategies which are marked by a wide range of characteristics. Supporting this variety poses a challenge for customization due to the range of complexity within each clausal modifier strategy, the multiplicity of syntactic phenomena that these strategies interact with and the possibility of multiple strategies within each language. We accomplish this with minimal additions to the Grammar Matrix, adding only two new lexical types and two phrase structure rules and defining strategy-specific subtypes to capture fine-grained complexity.

We begin with a description of related grammar engineering frameworks and annotation schemes (§2) followed by an overview of clausal modifiers, as seen in the typological literature (§3). Next we present an HPSG analysis of the phenomena (§4) and our contribution to the grammar customization system (§5). We describe our development data and evaluation on languages not considered during development, providing error analysis (§6) and conclude with a discussion of the uses for this library (§7).

2 Background

A variety of frameworks have been developed to support grammar engineering. While the Grammar Matrix uses the DELPH-IN Joint Reference Formalism (Copestake, 2002), a Head-driven Phrase Structure Grammar (HPSG; Pollard and Sag, 1994) based formalism that balances expressive power with computational efficiency, CoreGram (Müller, 2015) uses another HPSG-based formalism – TRALE (Meurers et al., 2002; Penn, 2004). Other frameworks include ParGram (Butt et al., 2002), the MetaGrammar project (de La Clergerie, 2005) and Grammatical Framework (Ranta, 2004). However, the Grammar Matrix provides a particularly suitable framework for our analyses because it requires its libraries to be typologically robust. As a result, multiple analyses must be developed for a particular phenomenon to account for its cross-linguistic distribution, and these analyses must interact with other phenomena in the customization system in order to produce customized grammars for any given language.

The grammars created by the Grammar Matrix produce syntactic annotations in the HPSG formalism and semantic annotations using Minimal Recursion Semantics (MRS; Copestake et al., 2005) for the strings they parse. These representations can capture information akin that captured by the Universal Dependencies annotation scheme and annotations in PropBank. On the syntactic side, the Universal

Dependencies annotation scheme (McDonald et al., 2013) uses the SCONJ and ADV part of speech tags for subordinators, corresponding to our adposition lexical type in §4.1 and adverb lexical type in §4.2, respectively. These are dependents of the subordinate verb, via a MARK dependency relation, which corresponds to our *basic-head-comp-phrase* in §4.1 and *isect-mod-phrase* in §4.2. In UD, the subordinate verb is a dependent of the matrix verb, via the ADVCL relation, which corresponds to our *scopal-mod-phrase*, described in §4.1. On the semantic side, PropBank (Kingsbury and Palmer, 2002; Palmer et al., 2005) marks different semantic classes of clausal modifiers, such as temporal (ARGM-TMP) and purposive (ARGM-PURP). One key difference is that in PropBank the subordinate clause is a dependent of the matrix verb, whereas in our MRS representations the matrix verb is an argument of the subordinator.

3 Typological Patterns

The typological literature describes a number of strategies for clausal modifiers and, in any given language, multiple strategies may be employed. Drawing on the review in Thompson et al. (2007), we can describe the range of clausal modification strategies in terms of the subordinator on the one hand, and additional characteristics of the subordinate clause on the other. A clausal modifier may be marked by a subordinator,[1] as in (1) from Japanese [jpn]; a subordinator pair comprising a subordinator in the embedded clause and an adverb in the matrix clause, as in (2) from Mandarin [cmn]; or special verbal morphology, as in (3) from Luiseño [lui].

(1) Ame ga agaru **to**, Gon wa hotto shite ana kara haidemashita
rain NOM stop when Gon TOP relief perform.INF hole from sneak.out.PST
'When the rain stopped, Gon got relieved and came out of the hole' [jpn]
(adapted from Thompson et al. 2007)

(2) **Yīnwèi** tiān hēi le, **suǒyǐ** wǒ méi chū - qu
because sky black CRS so 1SG NEG exit - go
'Because it had gotten dark, I didn't go out.' [cmn] (adapted from Li and Thompson 1989)

(3) Yaʔáʃ ŋéŋi ʂuŋá·l kí·ʃ pu-wá·qi-**pi**
man leave.remote woman house.ACC her-sweep-PURP
'The man left in order for the woman to sweep the house' [lui]
(Davis 1973 in Thompson et al. 2007)

Subordinators can occur either before or after the verb phrase or sentence in the subordinate clause. While in some languages their position is restricted to only before or only after the clause, in others the subordinator may attach freely before or after the clause, attaching at either the verb phrase (VP) or sentence (S) level. Clausal modifier strategies with subordinator pairs have an adverb in the matrix clause, which must co-occur with a particular subordinator, as in (2), and the position of that adverb may also be strict or free, attaching before or after the VP or S (Li and Thompson, 1989). Cross-linguistically, the subordinator pair strategy is particularly common for *if ... then* constructions.

The clausal modifier itself shows variation in both its external distribution and internal characteristics. The clausal modifier may attach before or after the matrix clause at the either the VP or S level. This distribution might also be strict or free.

Internal characteristics of the subordinate clause include constraints on the subject, word order and verbal morphology. In some strategies the subject of the modifying clause is shared with the matrix clause. In this case the subject of the subordinate verb and the matrix verb are co-referential and it is unexpressed in the subordinate clause. Additionally, some languages have a distinct word order in the subordinate clause. For example, German is a verb second language with verb final word order in embedded clauses (Thompson et al., 2007).

Finally, the clausal modifier may also be marked by special verbal morphology. Certain subordinators often occur with a particular morphological form or, if the clausal modifier strategy does not involve a subordinator, the morphological form itself may be associated with a particular semantic predication, as in (3). Many Turkic and Austronesian languages, require subordinate clauses to be nominalized.

[1] These are sometimes called subordinating conjunctions or clause linkers.

As described in Noonan (2007), this generally involves a nominalization morpheme on the verb and a change in the clause's internal distribution from that typical of verbal projections to that typical of nominal projections. Furthermore, this often includes an alternate case frame for nominalized verbs.

Our review of the typological literature reveals that clausal modifiers can be marked with a subordinator, a subordinator paired with a matrix adverb or no suborinator at all, illustrated in (1)–(3), and that the clause may bear a number of additional characteristics. We also find that it is common for languages to employ different strategies for different classes of clausal modifiers. In the next section we will develop an analysis to account for these typological patterns.

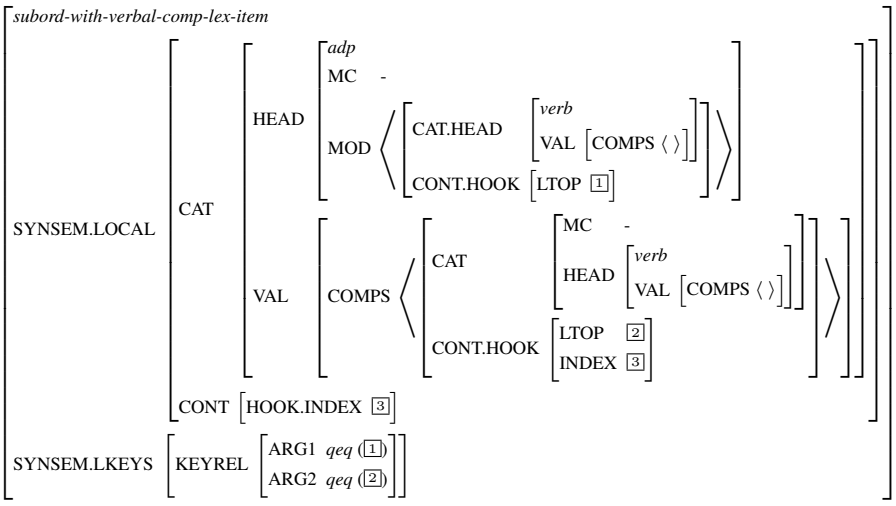
4 Analysis

We present an analysis for clausal modifiers using the HPSG formalism (Pollard and Sag, 1994) for syntactic structure and Minimal Recursion Semantics (MRS; Copestake et al., 2005) for semantic representation. HPSG uses lexical entries, lexical rules and phrase structure rules, encoded as feature structures, to model language. Central to HPSG is a notion of headedness, such that most phrases have a distinguished head daughter and syntactic features are passed up from the head daughter to the mother. Additionally, HPSG describes its features using a multiple inheritance hierarchy, allowing supertypes to capture broad generalizations and subtypes to add fine-grained detail. While there is little theoretical HPSG literature on a cross-linguistic analysis of clausal modifiers, we do find target semantic representations in the ERG (Flickinger, 2000, 2011), and take advantage of existing type definitions in the Grammar Matrix (Bender et al., 2002) on which to build our analysis.

Whereas we described the typology of clausal modifiers in terms the surface forms possible in each strategy, we develop our analysis with respect to underlying lexical type of the subordinator. We posit two types of subordinators: an adposition (§4.1), which is the head of its clause, and therefore must attach to to the subordinate clause at the sentence (S) level, and whose position (before or after the sentence) is strict; and an adverb (§4.2), whose distribution is more free—it may attach to the subordinate clause at either the S or verb phrase (VP) level and may attach either before or after the S or VP constituent.[2] If there is no subordinator, we analyze subordination as morphological (§4.3). For each of these three analyses, we define a new lexical type and/or unary rule for subordination. Finally, we add additional constraints, or feature specifications, to subtypes of the lexical types and unary rules to capture the variation described in §3.

4.1 The Adposition Subordinator

The adposition subordinator is the head of the subordinate clause, defined in (4) as a scopal modifier that takes a clause as its complement.[3] The subordinator is a type of adposition ([HEAD *adp*]) whose comp-

(4)


[2]While we suggest the adposition analysis unless there is evidence that the subordinator is an adverb, the strict S-attachment associated with adpositions is also possible under our adverb analysis.

[3]Due to limited space, we use the notation *qeq*() as a stand-in for the way we build the handle constraint. For a more detailed account of the handle constraint, see Copestake et al. 2005.

lement is [HEAD *verb*].[4] The complement clause is constrained with a boolean feature MC, indicating that it is not compatible with the characteristics of matrix clauses in the language, and has an empty complement list ([COMPS ⟨ ⟩]), requiring that the verb's complement requirements have been satisfied. Similarly, the subordinator modifies a clause (specified on the MOD list), which is [HEAD *verb*] and [COMPS ⟨ ⟩].[5]

Abstracting away from some of the details of semantic composition, the main semantic contribution of this type is a predication (the value of KEYREL), whose first semantic argument (ARG1) is the matrix clause and whose second argument (ARG2) is the subordinate clause. The result of these constraints is an MRS representation, like that in (5) for the sentence *Kim left when Pat arrived*, such that the predication _when_subord_rel[6] has two arguments which point (via *qeq*) to the matrix and subordinate verbs.

(5)
$$\left\langle h_1, e_3, \left| \begin{array}{l} h_4{:}\text{proper\_q}\langle 0:3\rangle(\text{ARG0 } x_6\{\text{PERS 3, NUM sg, IND +}\}, \text{ RSTR } h_5, \text{ BODY } h_7), \\ h_8{:}\text{named}\langle 0:3\rangle(\text{ARG0 } x_6, \text{ CARG } \textit{Kim}), \\ h_9{:}\text{\_leave\_v\_1}\langle 4:8\rangle(\text{ARG0 } e_3\{\text{SF prop, TENSE past}\}, \text{ ARG1 } x_6, \text{ ARG2 } p_{10}), \\ h_2{:}\text{\_when\_x\_subord}\langle 9:13\rangle(\text{ARG0 } e_{11}\{\text{SF prop}\}, \text{ ARG1 } h_{12}, \text{ ARG2 } h_{13}), \\ h_{14}{:}\text{proper\_q}\langle 14:17\rangle(\text{ARG0 } x_{16}\{\text{PERS 3, NUM sg, IND +}\}, \text{ RSTR } h_{15}, \text{ BODY } h_{17}), \\ h_{18}{:}\text{named}\langle 14:17\rangle(\text{ARG0 } x_{16}, \text{ CARG } \textit{Pat}), \\ h_{19}{:}\text{\_arrive\_v\_1}\langle 18:25\rangle(\text{ARG0 } e_{20}\{\text{SF prop, TENSE past}\}, \text{ ARG1 } x_{16}) \end{array} \right| \\ \{\, h_{15}\ qeq\ h_{18}, h_{13}\ qeq\ h_{19}, h_{12}\ qeq\ h_9, h_5\ qeq\ h_8, h_1\ qeq\ h_2 \,\} \right\rangle$$

We take advantage of existing phrase structure rules in the Grammar Matrix for complement and modifier attachment. The *basic-head-comp-phrase* phrase structure rule attaches the adposition to its complement (the subordinate clause) and the *scopal-mod-phrase* attaches the subordinate clause to the constituent it modifies (the matrix clause).

4.2 The Adverb Subordinator

In contrast with the adposition subordinator, the adverb subordinator is not a syntactic head, and therefore has more flexibility with respect to its position in the subordinate clause. As described in this section, because adverbs are not syntactic heads, many features will not be passed up the tree. Thus, rather than adding constraints on the matrix clause to the adverb's lexical type, we add them to the clausal modifier via a non-branching rule and use a new feature, SUBORDINATED to constrain the non-branching rule's daughter to contain the appropriate adverb.

The Lexical Type We define the adverb subordinator type as a non-scopal adverb. It has one element on its MOD list, the subordinate clause, which it can attach to at the VP or S level via the *isect-mod-phrase* rule, already in the Grammar Matrix. The element on the adverb's MOD list, like the complement of the adposition subordinator, is [HEAD *verb*] and [MC −] to prevent the adverb subordinator from occurring in matrix clauses.

The SUBORDINATED Feature Due to the constraints on composition in MRS (as codified, for example, in Copestake et al. 2001), we cannot introduce the subordinate predication on the adverb. MRS composition is done locally at each level in the tree, and though the adverb has access to the semantic head of the subordinate clause (the verb), it doesn't have access to any information about the matrix clause. Therefore, we add the predication and related semantic constraints with a unary rule, after the clause is fully formed. To link each adverb subordinator to a unary rule that contributes the corresponding predication, we introduce the SUBORDINATED feature, which is passed up by the various phrase structure rules. For each adverb, a feature value is created under SUBORDINATED and the unary rule that adds the predication will select for a daughter with the right SUBORDINATED value. To prevent multiple subordinators in the same sentence, all lexical entries for verbs are [SUBORDINATED none] as is the element on the adverb subordinator's MOD list. Once an adverb subordinator attaches, changing the clause's SUBORDINATED value, it will not be compatible with any other adverb subordinator.

[4]Note that this type will not extend to nominalized clauses. A type with a nominalized complement is described in §4.4.

[5]Additional constraints regarding the subjects of these clauses, which will determine whether or not the subject is expressed in the subordinate clause and if the subordinate clause modifies a VP or S, are added to subtypes, as described in §4.4.

[6]The specific predication symbol is specified on the lexical entry for each subordinator.

The Unary Rule We define the *adv-marked-subord-clause-phrase* such that it constrains its daughter to be [HEAD *verb*], [MC −] and, to prevent the rule from applying more than once, [MOD ⟨ ⟩]. The valence features must be fully satisfied on the mother and daughter, with the exception of the SUBJ list, which is identified between daughter and mother in order to accommodate subject sharing.

The main contribution of this rule is the addition of the matrix clause to the subordinate clause's MOD list as well as the subordinating predication and corresponding constraints. The element added to the MOD list is the same as that on the adposition subordinator's MOD list. The unary rule adds an *arg-12-ev-relation*, which has two arguments that are identified with the semantic content of the matrix and subordinate clauses respectively. The unary rule's daughter is identified with ARG2 and the matrix clause is identified with ARG1. Subtypes of this rule include a specific predication value, and we use the SUBORDINATED feature to identify the daughter of the unary rule with a clause marked by the appropriate subordinator for that predication. The resulting MRS is the same as that produced by the adposition subordinator in (5).

4.3 Morphological Subordination

Finally, morphological subordination involves a unary rule that selects a clause with particular morphological features and adds an element to the modifier list and a predication with the appropriate semantic identities. The *morphological-subord-clause-phrase* is identical to the *adv-marked-subord-clause-phrase* with one key difference: It selects a daughter with one or more syntactic or semantic features that are specific to the strategy, as described in §4.4, rather than using the SUBORDINATE feature.

4.4 Additional Constraints

For each clausal modifier strategy, we create a subtype of the appropriate lexical type and/or unary rule described in §4.1–4.3, adding additional constraints that capture the variation described in §3. Table 1 presents the additional features that constrain each phenomenon and indicates whether those features are expressed on the lexical type or unary rule.

Table 1: Features for Clausal Modifier Strategies

Constraints	Adposition Subordinator	Adverb Subordinator	No Subordinator
Clause Position	{POSTHEAD $+, -, bool$} (lexical type)	{POSTHEAD $+, -, bool$} (unary rule)	{POSTHEAD $+, -, bool$} (unary rule)
Clause Attachment	{MOD.SUBJ $\langle\,\rangle, \langle[\,]\rangle, none$} (lexical type)	{MOD.SUBJ $\langle\,\rangle, \langle[\,]\rangle, none$} (unary rule)	{MOD.SUBJ $\langle\,\rangle, \langle[\,]\rangle, none$} (unary rule)
Subordinator Position	{INIT $+, -$} (lexical type)	{POSTHEAD $+, -, bool$} (lexical type)	
Subordinator Attachment	{COMPS.SUBJ $\langle\,\rangle$} (lexical type)	{MOD.SUBJ $\langle\,\rangle, \langle[\,]\rangle, none$} (lexical type)	
Matrix Pair	{SUBPAIR} (lexical type)	{SUBPAIR} (lexical type)	
Special Morphology	{COMPS.FEATURE} (lexical type)	{MOD.FEATURE} (lexical type)	{DTR.FEATURE} (unary rule)
Nominalization	{COMPS.NMZ +} (lexical type)		{DTR.NMZ +} (unary rule)
Shared Subject	{COMPS.SUBJ $\boxed{0}\langle unexpressed\rangle$} {MOD.SUBJ $\boxed{0}$} (lexical type)	{DTR.SUBJ $\boxed{0}\langle unexpressed\rangle$} {MOD.SUBJ $\boxed{0}$} (unary rule)	{DTR.SUBJ $\boxed{0}\langle unexpressed\rangle$} {MOD.SUBJ $\boxed{0}$} (unary rule)

Clause Position and Attachment The first constraints we add govern the external distribution of the clausal modifier. Head-modifier rules are sensitive to the feature POSTHEAD, so if a clausal modifier strategy occurs strictly before the matrix clause, we add the constraint [POSTHEAD −]; if it occurs strictly after, [POSTHEAD +]; and if it can occur in either position, we leave this feature underspecified. We constrain clause attachment with the subject list of the matrix clause. If the modifier attaches to a sentence, the matrix clause must already have a subject, signified by an empty subject list ([SUBJ ⟨ ⟩]). On the other hand, if it attaches to a verb phrase, we constrain this list to be non-empty ([SUBJ ⟨ [ ] ⟩]) and if it can attach to either the VP or S, we leave this constraint underspecified. These constraints go directly on the lexical type of the adposition subordinator and on the unary rule for the adverb subordiantor.

Subordinator Position and Attachment If the subordinator is an adverb, the subordinator position and attachment are constrained the same way as the clausal modifier's, except that these constraints are

specified on the lexical type, so that they will govern the adverb's distribution in the subordinate clause. If the subordinator is an adposition, it is the head of its clause and attaches strictly to a sentence, so the element on its COMPS list is necessarily [SUBJ $\langle\ \rangle$]. It can attach at the beginning or end of the clause (but only one or the other), which is constrained with the INIT feature. We add INIT to the head-complement rules to constrain the order of the head and complement. If an adposition attaches at the beginning of the clause, it is [INIT +] and if it attaches at the end it is [INIT −].[7]

Subordinator Pairs Both adverbial and adpositional subordinators can require a adverb in the matrix clause (as illustrated in (2)), which we treat as a scopal modifier, constraining its position and attachment the same way as the adverb subordinator. We introduce the SUBPAIR feature, and for each pair of subordinators in the language, we create a unique value, which is added to the matrix adverb's lexical type. The head-modifier rule passes the SUBPAIR value up from the non-head daughter and this feature is propagated up through the head daughter by the other phrase structure rules. In the subordinate clause, this feature is specified on the MOD list of the adposition subordinator, or the lexical type of the adverb subordinator.[8] Finally the *scopal-head-mod-phrase* identifies the SUBPAIR value of the non-head-daughter's MOD list with that of the head-daughter.

Special Morphology and Nominalization Many subordination strategies, whether they include a subordinator or not, require special morphology on the embedded verb. Certain morphological forms can be associated with with features in the morphology library (O'Hara, 2008; Goodman, 2013). For adverb and adposition subordinators, morphological constraints are specified on the lexical type, and if there is no subordinator, these constraints are specified on the unary rule. Supported morhpological features include FORM, ASPECT, MOOD, NOMINALIZATION and user-defined syntactic features.

If NOMINALIZATION is among the specified features, we use a different set of lexical types and unary rules. The nominalized clauses library (Howell et al., to appear) allows users to define nominalization strategies in which nominalization occurs at the verb, verb phrase or sentence level and either adds a nominalized predication or not. This library changes clauses to [HEAD *noun*], adds the feature [NMZ +] and may add a nominalization predication that scopes over the event. Therefore, we define new lexical types and unary rules that select a clause that is [HEAD *noun*] [NMZ +].[9] Lexical nouns are constrained to [NMZ –], preventing the subordinator from selecting a lexical noun rather than a nominalized clause.

Shared Subjects If the subject is shared between the matrix and subordinate clauses, the embedded clause is constrained with an *unexpressed* element on its SUBJ list. The XARG (external argument; Copestake et al., 2005) of the subordinate clause is identified with the XARG of the subordinator's modifier, creating a semantic identity between the subjects of the subordinate and matrix verbs.

Special Word Order We use the analysis of Fokkens (2014) to accommodate verb second word order in the matrix clause and verb final in the subordinate clause. Under this analysis, a clause with subordinate word order is [MC −], which is compatible with our adposition lexical entry and unary rules.

Summary The analyses presented in this section build on the existing customization system, interaction with many different libraries, including word order, nominalization, aspect and mood, among others. We have accounted for the distribution of clausal modifiers, as described in §3, with the addition of two new lexical types and two unary rules to the Grammar Matrix. We create subtypes of those lexical types and unary rules that are further constrained according to Table 1 to account for the diversity of clausal modifier strategies in the typological literature.

[7] These rules are added by the word order library (Bender and Flickinger, 2005). In the rare case that the word order library does not add the necessary rule (for example, an otherwise head-final language has a head-initial subordinator), we add a special head-complement rule that selects for an adposition head daughter. Further detail is provided by Zamaraeva et al. (2018).

[8] This feature is then copied to the modifier list of the clausal modifier by the unary rule.

[9] If the nominalization strategy adds a nominalized_rel predication, the lexical type for adposition subordinators and unary rule for adverb or morphological subordination identify the subordinator's ARG2 with the local top of the _nominalized_rel, rather than the INDEX of the verb.

5 Customized Grammar Development

The Grammar Matrix customization system (Bender and Flickinger, 2005; Bender et al., 2010; Drellishak, 2009) uses a web-based questionnaire to elicit user input regarding the typological characteristics of their language on the front end and a customization script that produces a grammar that handles these phenomena on the back end. The two are linked via a 'choices' file, containing values that correspond to the user's input and can be read by the customization script. Thus to integrate the analysis described in §4, we first developed a clausal modifiers page in the questionnaire[10] to elicit the user's choices and then modified the customization script to add the appropriate analyses to the output grammar.

The clausal modifiers questionnaire uses an iterator so that users can define any number of clausal modifier strategies found in their language. For each strategy the user is asked about the clausal modifier's position and attachment and if the subordinating predication is contributed by a subordinator, a subordinator pair or if it does not involve a subordinator, as illustrated in (1), (2) and (3), respectively. Subsequent questions are based on this first choice.

If the user selects subordinator or subordinator pair, they are asked if the subordinator is an adverb or adposition. If they choose adposition, they are asked if it occurs at the beginning or end of the clause. If it's an adverb, they are asked whether it attaches strictly before, strictly after or freely before or after a VP, S or either a VP or S. If they select subordinator pair, they are also asked about the position of the matrix adverb. The user may enter any number of subordinators or subordinator-matrix adverb pairs, including an orthographic representation and a predicate symbol for each. If there is no subordinator, the user may enter only one predication per strategy. The user may add any number of verbal features associated with the clausal modifier strategy and can check a box to indicate subject sharing.

The choices file generated by the questionnaire goes through a script which validates if the choices are 'legal' and will result in a working grammar. If validation fails, the user is prompted to make changes; otherwise, the customization script adds a basic lexical type and/or unary rule to the grammar, based on the subordinator type, and creates a subtype of that lexical type and/or unary rule that is specific to the strategy. Constraints corresponding to the other choices are added to the subtypes, according to §4.4.

6 Testing, Evaluation and Error Analysis

We use two types of testing during development, which we follow with held out evaluation. Each test involves a testsuite, a choices file and a grammar produced by the customization system. The testsuites are small and designed to be representative of the relevant contrasts in the language, including the full range of possible grammatical sentences for each clausal modifier strategy as well as ungrammatical sentences that are each ungrammatical in one specific way. While our testsuites are small, they are robust in that they contain an example that targets each feature in Table 1 as well as any relevant interacting phenomena individually. We create choices files that define each clausal modifier strategy and account for interacting phenomena and load the resulting grammars into the LKB (Copestake, 2002) to parse each sentence. We inspect the parse trees and semantic representations to verify their correctness. [11]

6.1 Pseudo Languages

During development we tested our analyses on pseudo languages—artificial languages with a minimal lexicon which exhibit each of the phenomena outlined in Table 1 as well as known interacting phenomena. The typological space is large, including 1008 possible combinations for the features in Table 1.[12] Rather than test all combinations exhaustively, we created tests by sampling each constraint from the first column with each subordinator type, rather than each value, and added additional tests for interacting phenomena, including word order, verbal features and nominalization. To test the interaction between different strategies (which may result in overgeneration if under-constrained), we included multiple strategies in some tests. This resulted in 16 pseudo languages, containing a total of 33 distinct

[10] http://matrix.ling.washington.edu/customize/matrix.cgi?subpage=clausalmods

[11] Our code and testsuites can be checked out from svn://lemur.ling.washington.edu/shared/matrix/trunk at revision 41464.

[12] This is a conservative estimate, bundling many features and suppressing interacting phenomena, such as special word order, matrix adverb position, nominalization strategy and different types of verbal features. In reality, the space is much larger.

strategies. We refined our analysis during testing to achieve full coverage over these languages.

6.2 Development Languages

Next we tested our system on natural languages that illustrate particular phenomena in our library. We selected four languages, based on the characteristics their clausal modifiers exhibit and developed a testsuites of ten sentences for each, illustrating which characteristics of clausal modifiers are exhibited in the language and which would result in ungrammatical sentences, according to descriptive grammars.

Mandarin Mandarin [cmn] has a number of subordinator pairs, of which some subordinators can occur without their matrix adverb and some matrix adverbs are homophonous with conjunctions (Li and Thompson, 1989). We illustrated this range with one such pair *yīnwèi...suǒyǐ*, defining two clausal modifier strategies, one with the pair and one with just *yīnwèi* as a subordinator, and one coordination strategy,[13] with *suǒyǐ* as a conjunction.

Wambaya Wambaya [wmb] expresses purposive and prior clauses with a special purposive or prior morpheme on the verb (Nordlinger, 1998). Purposive can also be expressed with the infinitive suffix. These morphological strategies exhibit subject sharing and require dative instead of absolutive case on the object. 'When or because' and 'right after' clauses are finite clauses with no subordinator.[14] 'When or because' clauses attach strictly after the matrix clause, whereas 'right after' clauses attach strictly before. The inherent ambiguity from having both of these strategies in the same language is captured by our grammar. Our typological survey did not reveal case change outside of nominalized clauses, so it was not accounted for in our analysis. For this reason, our Wambaya testsuite has one grammatical sentence that does not parse, a purposive clause with dative case on the object, and one ungrammatical sentence that does parse, a purposive clause with absolutive case on the object.

Rukai We used Rukai [dru], as described by Zeitoun (2007), to test nominalization as a primary strategy for clausal modifiers. We tested two strategies, one in which the nominalization morpheme is also associated with 'reason' and another in which a generic nominalization morpheme is paired with a subordinator, meaning 'reason'. To capture the distinction between these morphemes, we defined a feature in customization that is suitable for both verbs and nouns (so that it will mark the clause both before and after nominalization), and associated it with each morpheme and each clausal modifier strategy.

German Finally German [deu] has verb final word order in the subordinate clause, while matrix clauses are verb second. We used a strategy with a clause initial subordinator from Thompson et al. (2007) to test this word order variation.

Table 2: Development Languages

Language	Family	Test Items	Coverage	Overgeneration
Wambaya [wmb]	Mirndi	10	5/6	1/4
German [deu]	Indo-European	10	2/2	0/8
Rukai [dru]	Austronesian	10	2/2	0/8
Mandarin [cmn]	Sino-Tibetan	10	6/6	0/4

We summarize our results on development languages in Table 2. The two errors are the result of case-frame changes, which are more closely related to verbal morphology than subordination. As a this is an interacting phenomenon and not specific to clausal modifiers we leave this out of scope for our library.

6.3 Evaluation with Held-out Languages

We evaluate on languages that are genetically and geographically diverse from the languages considered in development. These languages are chosen at random from the set of descriptive grammars available at the University of Washington library, and discarded if (a) they come from the same language family as an illustrative language or previous held-out language, (b) they do not contain (or the grammar does

[13] Using the coordination library contributed by (Drellishak and Bender, 2005)

[14] While this could be analyzed as juxtaposition, we illustrate the user's analytic freedom to treat these as subordinate clauses.

not describe) clausal modifiers or (c) the data does not contain a sufficient level of annotation to reliably construct additional examples. From the descriptive grammars, we develop testsuites that capture the range of clausal modifiers according to the linguist's description, constructing grammatical and ungrammatical sentences from examples in the descriptive grammar to contrast each characteristic of the clausal modifier strategy.[15] For each language we include up to four clausal modifier strategies, as described by the author, which may translate to between three and ten strategies in the choices file, depending on the type of variation therein.[16]

Ma'di Ma'di [mhi] has a wide range of clausal modifier strategies, including postposition subordinators, adverb subordinators (some requiring the subordinate clause to occur first, and others allowing it in either position) and subordinate clauses marked by the directive mood (Blackings and Fabb, 2003). Our analysis of adverb subordinators, based on the typological literature and a generalization that adverbials related to time and modality tend to attach higher in the tree (Cinque, 1999), only allows VP and S attachment. In Ma'di, however, adverb subordinators may intervene between the verb and object, suggesting V attachment as well. Since we do not support this, one sentence in our testsuite is not parsed.

Mosetén Strategies in Mosetén [cas], include subordinators in finite clauses, subordinators that require special morphology and strictly morphological subordination. Sakel (2004) states that the clausal modifier can occur in any position that an adverb can, but only shows S attachment in examples. The adverb chapter does not discuss the distribution of adverbs, but shows examples of S and VP attachment, so we infer that clausal modifiers may attach at the S and VP level. This testsuite revealed a bug in the customization code, that was not sampled in pseudo language evaluation. The unary rule supertype for morphological subordination is constrained to be [SUBJ $\langle\ \rangle$], barring VP attachment for morphologically subordinated clauses, and resulting the failure of two strings to parse.

Lavukaleve Subordination in Lavukaleve [lvk] is primarily morphological and various clausal modifier strategies position the subordinate clause differently (Terrill, 2003). Although Lavukaleve is a nominative-accusative language in general, it has a split ergative system in subordinate clauses, such that third person subjects take canonical object marking, but first and second person subjects take the usual subject marking. Alternate case frames in subordinate clauses are not modeled in the Grammar Matrix or by our library, so this results in one ungrammatical string parsing and one grammatical string failing to parse. Another interesting phenomenon in Lavukaleve, as described by Terrill (2003), is an adverb that occurs in the matrix clause when the subordinate clause is marked by special morphology, but no subordinator. This is not captured in our analysis of subordinator pairs, which require a subordinator, so another sentence in our testsuite fails to parse.

Basque Hualde and de Urbina (2003) provide an extremely thorough description of clausal modifiers in Basque [eus], including numerous clausal modifier strategies with different morphological forms that combine with various adpositions and adverbs. We do our best to select a representative set of combinations, including two sets of adpositions that co-occur with different morphemes, one adverb that co-occurs with another morpheme and a nominalization strategy whose meaning differs based on the case of the nominalized clause. This results in 10 clausal modifier strategies in the choices file and a special feature with 7 values (such as purposive, conditional, etc.) that is suitable for both verbal and nominal projections. Our test sentences contained dropped subjects which revealed that the nominalized clauses library does not provide a phrase structure rule for subject dropping in nominalized clauses. This results in three sentences failing to parse. However, upon constructing such a rule, we confirmed that if this rule were in the grammar, those sentences would parse correctly.

Uranina Finally, Uranina [ura] primarily uses subordinators (a clause final clitic and a number of postpositions) for clausal modifiers (Olawsky, 2006). The majority require a finite verb, but in the

[15] This is necessary for robust testing, as the descriptive grammar will often include only grammatical examples, but explain the full paradigm in prose.

[16] For example the choices file requires separate strategies for each morphologically contributed predication, even if they might be grouped under one strategy in the descriptive grammar.

innovative dialect (spoken by younger members of the community), some of these postpositions co-occur with non-finite verbs. There is also a subordinator pair for conditional clauses. We created three clausal modifier strategies, with multiple subordinators to successfully capture this variation.

Table 3: Held-out Language Evaluation

Language	Family	Test Items	Coverage	Overgeneration
Ma'di [mhi]	Central Sudanic	23	16/17	0/6
Mosetén [cas]	Mosetenan	26	13/15	0/11
Lavukaleve [lvk]	Solomons East Papuan	23	8/10	1/13
Basque [eus]	Basque	26	13/16	0/10
Uranina [ura]	Uranina	21	12/12	0/9

The average coverage over 5 test suites was 88.4% and overgeneration was 1.5%, as detailed in Table 3. With the exception of case frame change on embedded verbs in Wambaya and Lavukaleve, our library does a good job of preventing the parsing of ungrammatical strings, upholding our emphasis on precision. Our recall is lower, as we are not able to parse 8 strings, due to 5 errors, which we can classify in 3 groups. The first group includes bugs: the inability to parse VP attachment of clausal modifiers in Mosetén is due to a mistake in the supertype of the non-branching rule, and the lack of a subject dropping rule for nominalized verbs for Basque is due to an oversight in the nominalized clauses library. These bugs are easily fixed, now that they have been identified. The next group is out-of-scope phenomena: valence change (Wambaya and Lavukaleve) is more closely associated with verb form than clausal modifier strategy and should be handled in a library for valence change. The third type of error comes from phenomena that were not brought to light in the typological survey, but were found during held-out evaluation. First, we expected that adverb subordinators would be high-attaching, based on a cross linguistic generalization about adverb classes and the absence of any attested low-attaching adverbs in our literature review. Ma'di brings to light an interesting contradiction to this assumption and we will add the option of V attachment in the matrix clause now that it has been attested. Second, the typological literature did not suggest that matrix "pair" adverbs could occur without a subordinator in the embedded clause. If this is in fact the case in Lavukaleve, this poses a interesting direction for future work.

7 Conclusion

We presented a cross-linguistic HPSG analysis of clausal modifiers, which we implemented in the LinGO Grammar Matrix. We demonstrated the effectiveness of this library over the known typological space in §6.1, tested the application of specific clausal modifier strategies on real languages in §6.2, and evaluated its generalizability on 5 held-out languages in §6.3. Because our analysis is implemented within a larger grammar engineering environment, we were able to test its interaction with relevant phenomena. We contributed all testsuites and choices files back to the project so that they can be added to regression testing to ensure that downstream changes will not impact the coverage of this library and future developers can test their phenomena with clausal modifiers. The grammars produced by the Grammar Matrix with the addition of the clausal modifiers library are useful starting points to linguists who wish to develop broad coverage grammars, as the starter grammar can now include not only matrix clause phenomena, but analyses for subordinate clausal modifiers. In addition to developing broad coverage grammars, the grammars produced by the Grammar Matrix can be used to teach grammar engineering and for linguistic hypothesis testing, as our approach allows user-linguists analytic freedom in their implementation, so that multiple analyses can be explored.

Acknowledgements

We would like to thank Emily Bender for helpful discussion and Jiahui Huang and Paul Buechsenmann for grammaticality judgments for examples in our testsuites.

This material is based upon work supported by the National Science Foundation under Grant No. BCS-1561833 (PI Bender).

References

Emily M. Bender. 2008. Grammar engineering for linguistic hypothesis testing. In *Proceedings of the Texas Linguistics Society X conference: Computational linguistics for less-studied languages*, pages 16–36. Citeseer.

Emily M. Bender. 2010. Reweaving a grammar for Wambaya. *Linguistic Issues in Language Technology*, 3(1).

Emily M. Bender and Dan Flickinger. 2005. Rapid prototyping of scalable grammars: Towards modularity in extensions to a language-independent core. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing IJCNLP-05 (Posters/Demos)*, Jeju Island, Korea, 2005.

Emily M. Bender, Dan Flickinger, and Stephan Oepen. 2002. The Grammar Matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In John Carroll, Nelleke Oostdijk, and Richard Sutcliffe, editors, *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, pages 8–14, Taipei, Taiwan, 2002.

Emily M. Bender, Scott Drellishak, Antske Fokkens, Laurie Poulson, and Safiyyah Saleem. 2010. Grammar customization. *Research on Language & Computation*, pages 1–50. ISSN 1570-7075. URL `http://dx.doi.org/10.1007/s11168-010-9070-1`. 10.1007/s11168-010-9070-1.

Mairi John Blackings and Nigel Fabb. 2003. *A Grammar of Ma'di*, volume 32. Walter de Gruyter.

Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The parallel grammar project. In *Proceedings of COLING-2002 Workshop on Grammar Engineering and Evaluation*, pages 1–7.

Guglielmo Cinque. 1999. *Adverbs and functional heads: A cross-linguistic perspective*. Oxford University Press on Demand.

Ann Copestake. 2002. Definitions of typed feature structures. In Stephan Oepen, Dan Flickinger, Junichi Tsujii, and Hans Uszkoreit, editors, *Collaborative Language Engineering*, pages 227–230. CSLI Publications, Stanford, CA.

Ann Copestake, Alex Lascarides, and Dan Flickinger. 2001. An algebra for semantic construction in constraint-based grammars. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 140–147. Association for Computational Linguistics.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3(2-3):281–332.

John Frederick Davis. 1973. *A partial grammar of simplex and complex sentences in Luiseño*. PhD thesis, University of California.

Eric Villemonte de La Clergerie. 2005. From metagrammars to factorized tag/tig parsers. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 190–191. Association for Computational Linguistics.

Scott Drellishak. 2009. *Widespread but not universal: Improving the typological coverage of the Grammar Matrix*. PhD thesis, University of Washington.

Scott Drellishak and Emily Bender. 2005. A coordination module for a crosslinguistic grammar resource. In *International Conference on Head-Driven Phrase Structure Grammar*, volume 12, pages 108–128. URL `http://web.stanford.edu/group/cslipublications/cslipublications/HPSG/2005/drellishak-bender.pdf`.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6 (1) (Special Issue on Efficient Processing with HPSG):15 – 28.

Dan Flickinger. 2011. Accuracy v. robustness in grammar engineering. In Emily M. Bender and Jennifer E. Arnold, editors, *Language from a Cognitive Perspective: Grammar, Usage and Processing*, pages 31–50. CSLI Publications, Stanford, CA.

Antske Sibelle Fokkens. 2014. *Enhancing Empirical Research for Linguistically Motivated Precision Grammars*. PhD thesis, Department of Computational Linguistics, Universität des Saarlandes.

Michael Wayne Goodman. 2013. Generation of machine-readable morphological rules from human-readable input. *Seattle: University of Washington Working Papers in Linguistics*, 30.

Kristen Howell, Olga Zamaraeva, and Emily M. Bender. to appear. Nominalized clauses in the Grammar Matrix. In *The 25th International Conference on Head-Driven Phrase Structure Grammar*.

José Ignacio Hualde and Jon Ortiz de Urbina. 2003. *A grammar of Basque*, volume 26. Walter de Gruyter.

Paul Kingsbury and Martha Palmer. 2002. From Treebank to PropBank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, pages 1989–1993, Las Palmas, Spain, 2002. Citeseer.

Charles N Li and Sandra A Thompson. 1989. *Mandarin Chinese: A functional reference grammar*. University of California Press.

Montserrat Marimon. 2010. The Spanish resource grammar. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*, pages 700–704, Valletta, Malta, 2010.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 92–97.

W Detmar Meurers, Gerald Penn, and Frank Richter. 2002. A web-based instructional platform for constraint-based grammar formalisms and parsing. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics*, pages 19–26. Association for Computational Linguistics.

Stefan Müller. 2015. The CoreGram project: Theoretical linguistics, theory development and verification. *Journal of Language Modelling*, 3(1):21–86. URL `https://hpsg.hu-berlin.de/~stefan/Pub/coregram.html`.

Michael Noonan. 2007. Complementation. In Timothy Shopen, editor, *Language typology and syntactic description*, volume 2: Complex constructions, pages 52–150. Cambridge University Press.

Rachel Nordlinger. 1998. *A Grammar of Wambaya, Northern Australia*. Pacific Linguistics.

Kelly O'Hara. 2008. A morphotactic infrastructure for a grammar customization system. Master's thesis, University of Washington.

Knut J. Olawsky. 2006. *A grammar of Uranina*, volume 37. Walter de Gruyter.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Gerald Penn. 2004. Balancing clarity and efficiency in typed feature logic through delaying. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 239–246. Association for Computational Linguistics.

Carl Pollard and Ivan A Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.

Laurie Poulson. 2011. Meta-modeling of tense and aspect in a cross-linguistic grammar engineering platform. *University of Washington Working Papers in Linguistics (UWWPL)*, 28.

Aarne Ranta. 2004. Grammatical framework. *Journal of Functional Programming*, 14(2):145–189.

Jeanette Sakel. 2004. *A grammar of Mosetén*, volume 33. Walter de Gruyter.

Melanie Siegel, Emily M Bender, and Francis Bond. 2016. *Jacy: An implemented grammar of Japanese*. CSLI Publications.

Angela Terrill. 2003. *A grammar of Lavukaleve*, volume 30. Walter de Gruyter.

Sandra A Thompson, Robert E Longacre, and Shin Ja J Hwang. 2007. Adverbial clauses. In Timothy Shopen, editor, *Language typology and syntactic description*, volume 2: Complex constructions, pages 237–269. Cambridge University Press.

Thomas James Trimble. 2014. Adjectives in the LinGO Grammar Matrix. Master's thesis, University of Washington.

Olga Zamaraeva, Kristen Howell, and Emily M. Bender. A cross-linguistic account of subordinator and subordinate clause position. Poster to be presented at The 25th International Conference on Head-Driven Phrase Structure Grammar, 2018.

Elizabeth Zeitoun. 2007. *A grammar of Mantauran (Rukai)*. Institute of Linguistics, Academia Sinica Taipei.

Sliced Recurrent Neural Networks

Zeping Yu
School of Eclectronic Information and Electrical Engineering
Shanghai Jiao Tong University
zepingyu@foxmail.com

Gongshen Liu*
School of Eclectronic Information and Electrical Engineering
Shanghai Jiao Tong University
lgshen@sjtu.edu.cn

Abstract

Recurrent neural networks have achieved great success in many NLP tasks. However, they have difficulty in parallelization because of the recurrent structure, so it takes much time to train RNNs. In this paper, we introduce sliced recurrent neural networks (SRNNs), which could be parallelized by slicing the sequences into many subsequences. SRNNs have the ability to obtain high-level information through multiple layers with few extra parameters. We prove that the standard RNN is a special case of the SRNN when we use linear activation functions. Without changing the recurrent units, SRNNs are 136 times as fast as standard RNNs and could be even faster when we train longer sequences. Experiments on six large-scale sentiment analysis datasets show that SRNNs achieve better performance than standard RNNs.

1 Introduction

Recurrent neural networks (RNNs) have been widely used in many NLP tasks, including machine translation (Cho et al., 2014; Bahdanau et al., 2015; Luong et al., 2015; Bradbury and Socher, 2016), question answering (Xiong et al., 2016; Chen et al., 2017), image caption (Xu et al., 2015; Karpathy and Li, 2015), and document classification (Tang et al., 2015; Yang et al., 2016; Zhou et al., 2016). RNNs have the ability to obtain the order information of the input sequences. The two most popular recurrent units are long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) (Cho et al., 2014), both of which could store previous memory in hidden states and use a gating mechanism to determine how much previous memory should be combined with the current input. Unfortunately, because of the recurrent structure, RNNs cannot be computed in parallel. Therefore, training RNNs takes much time, which limits academic research and industrial applications.

To solve this problem, several scholars try to use convolutional neural networks (CNNs) (Lecun et al., 1998) instead of RNNs in the field of NLP (Kim, 2014; Kalchbrenner et al., 2014; Gehring et al., 2017). However, CNNs may not obtain the order information of the sequences, which is very important in NLP tasks.

Some scholars tried to increase the speed of RNNs by improving the recurrent units and they have achieved good results. Quasi-recurrent neural networks (QRNNs) (Bradbury et al., 2017) got up to 16 times faster speeds by combining CNNs with RNNs. Lei et al. (2017) proposed the simple recurrent unit (SRU), which is 5-10 times faster than LSTM. Similarly, strongly-typed recurrent neural networks (T-RNN) (Balduzzi and Ghifary, 2016) and minimal gated unit (MGU) (Zhou et al., 2016) are also methods that could change the recurrent units.

Although RNNs have achieved faster speeds in these researches with the recurrent units improved, the recurrent structure among the entire sequence remains unchanged. As we still have to wait for the output of previous step, the bottleneck of RNNs still exists. In this paper, we introduce sliced recurrent neural networks (SRNNs), which are substantially faster than standard RNNs without changing the recurrent units. We prove that when we use linear activation functions, the standard RNN is a special case of the SRNN, and the SRNN has the ability to obtain high-level information of the sequences.

* Corresponding author

Proceedings of the 27th International Conference on Computational Linguistics, pages 2953–2964
Santa Fe, New Mexico, USA, August 20-26, 2018.

In order to compare our model with the standard RNN, we choose GRU as the recurrent unit. Other recurrent units could also be used in our structure, because we improve the overall RNN structure among the whole sequence rather than just changing the recurrent units. We complete experiments on six large-scale datasets and SRNNs perform better than standard RNNs on all the datasets. We open source our implementation in Keras (François et al, 2015).[1]

2 Model Structure

2.1 Gated Recurrent Unit

The GRU (Bahdanau et al., 2014) has the reset gate *r* and the update gate *z*. The reset gate decides how much of the previous memory is combined with the new input, and the update gate determines how much of the previous memory is retained.

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{1}$$

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{2}$$

where x is the input and h is the hidden state.

$$\tilde{h}_t = \tanh(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h) \tag{3}$$

The candidate hidden state $\tilde{h}_t$ is controlled by the reset gate. When the reset gate is 0, the previous memory is ignored.

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t \tag{4}$$

When the update gate is 1, the hidden state could copy the previous memory to the current moment and ignore the current input.

2.2 The Standard RNN Structure

The standard RNN structure is shown in Figure 1, where A denotes the recurrent units.

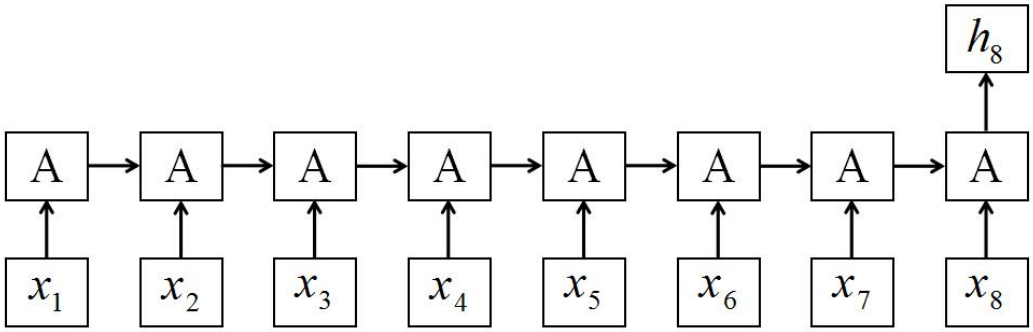


Figure 1: The standard RNN structure. Each step waits for the output of its previous step, which is computed by the recurrent unit A.

The length of the input sequence *X* is *T*, and here we assume $T = 8$ as Figure 1 shown. The standard RNN uses the last hidden state as the representation of the whole sequence, and then add a softmax classifier to predict the labels. In addition to GRU and LSTM, QRNN and SRU could be seen as one form of recurrent unit A as well. However, the overall RNN structure has not been improved, at each step we need to wait for the output of the previous step:

$$h_t = f(h_{t-1}, x_t) \tag{5}$$

where *h* is the previous hidden state. This standard RNN structure in which every two adjacent cells are connected causes the bottleneck: the longer the input sequence is, the longer it takes.

[1] https://github.com/zepingyu0512/srnn

2.3 Sliced Recurrent Neural Networks

We construct a new RNN structure called sliced recurrent neural networks (SRNNs), which is shown in Figure 2. In Figure 2 the recurrent unit is also referred to as A.

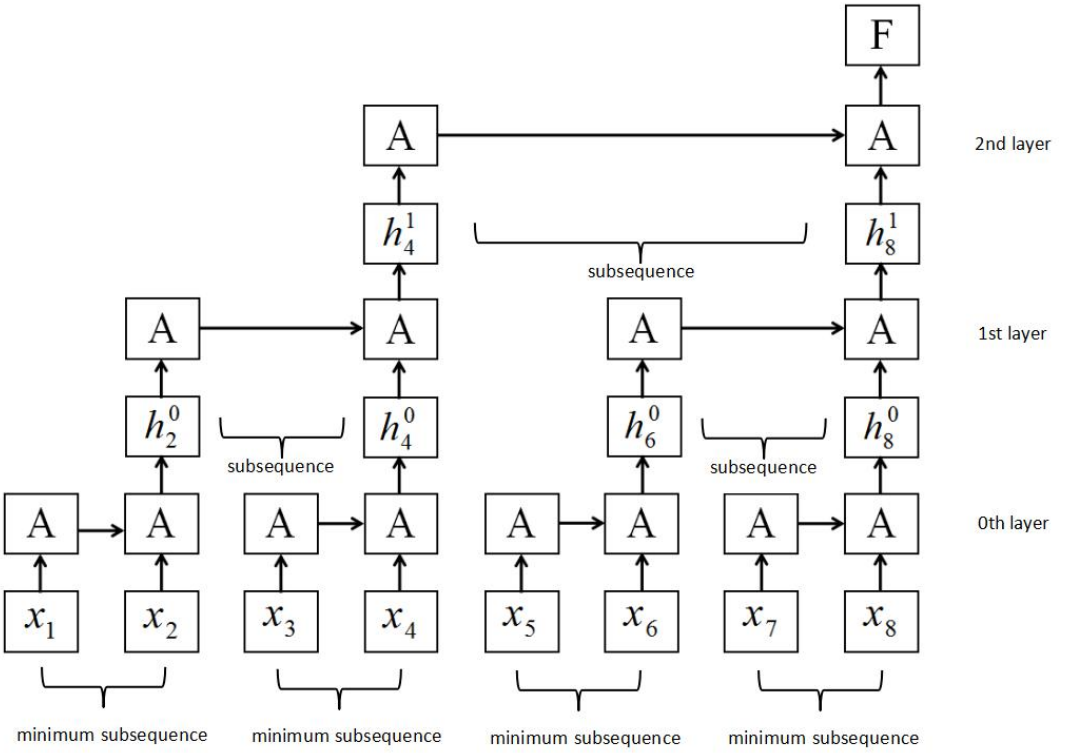


Figure 2: The SRNN structure. It is constructed by slicing the input sequence into several minimum subsequences with equal length. The recurrent units could work on each subsequence simultaneously on each layer, and the information could be transmitted through multiple layers.

The length of input sequence X is T, and the input sequence is:

$$X = [x_1, x_2, ..., x_T] \tag{6}$$

where x is the input at each step and it may have multiple dimensions such as word embeddings. Then we slice X into n subsequences of equal length, and the length of each subsequence N is:

$$t = \frac{T}{n} \tag{7}$$

where n is the slice number, and the sequence X could be represented as:

$$X = [N_1, N_2, ..., N_n] \tag{8}$$

where each subsequence is:

$$N_p = [x_{(p-1)\times t+1}, x_{(p-1)\times t+2}, ..., x_{p\times t}] \tag{9}$$

Similarly, we slice each subsequence N into n subsequences of equal length again, and then repeat this slice operation k times until we have an appropriate minimum subsequence length on the bottom layer (we call it 0th layer, which is shown in Figure 2), and k+1 layers are obtained by slicing k times. The minimum subsequence length of 0th layer is:

$$l_0 = \frac{T}{n^k} \tag{10}$$

and the number of the minimum subsequences on 0th layer is:

$$s_0 = n^k \tag{11}$$

Because every parent sequence on pth layer (p>0) is sliced into n parts, the number of the subsequences on pth layer is:

$$s_p = n^{k-p} \tag{12}$$

and the subsequence length of pth layer is:

$$l_p = n \tag{13}$$

Take Figure 2 as an example. The sequence length T is 8, the slice operation times k is 2, and the slice number n of each pth layer is 2. After slicing the sequence twice, we get four minimum subsequences on 0th layer, and the length of each minimum subsequence is 2. If the length of the sequence or the length of its subsequences cannot be divided by n, we may exploit padding method or choose different slice number on each layer. Different k and n may be used on different tasks and datasets.

The difference between the SRNN and the standard RNN is that the SRNN slices the input sequence into many minimum subsequences and utilizes the recurrent units on each subsequence. In this way, the subsequences could be easily parallelized. On 0th layer, the recurrent units are acted on each minimum subsequence through the connection structure. Next, we obtain the last hidden states of each minimum subsequence on 0th layer, which are used as the input of their parent sequences on 1st layer. And then we use the last hidden state of each subsequence on (p-1)th layer as the input of their parent sequence on pth layer and compute the last hidden states of the subsequences on pth layer.

$$h_t^1 = \overrightarrow{GRU}^0(mss^0_{(t-l_0+1)\sim t}) \tag{14}$$

$$h_t^{p+1} = \overrightarrow{GRU}^p(h^p_{t-l_p} \sim h_t^p) \tag{15}$$

where h_l^p is the number l hidden state on pth layer, mss denotes minimum subsequences on 0th layer, and different GRUs could be used on different layers. This operation is repeated between each sub-parent sequence on each layer until we get the final hidden state F of the top layer (kth layer).

$$F = \overrightarrow{GRU}^k(h^k_{t-l_k} \sim h_t^k) \tag{16}$$

2.4 Classification

Similar to the standard RNN, the softmax layer is added after the final hidden state F to classify the labels:

$$p = \text{softmax}(W_{\text{F}}F + b_F) \tag{17}$$

and the loss function is negative log-likelihood:

$$loss = -\sum \log p_{dj} \tag{18}$$

where d is each document of the dataset with label j.

2.5 Speed Advantage

The reason why SRNNs could be parallelized is that SRNNs improve the traditional connection structure. In SRNNs, not every current input is connected with its previous moment, but the entire sequence is connected together by a sliced method. SRNNs could also obtain the sequence order by the recurrent units in each subsequence, and transmit the information through multiple layers. We assume that the time spent in each recurrent unit is r, then the time spent in the standard RNN is:

$$t_{RNN} = T \times r \tag{19}$$

where T is the input sequence length. In the SRNN, each minimum subsequence could be parallelized, so the time spent on 0th layer is:

$$t_0 = (\frac{T}{n^k}) \times r \tag{20}$$

and similarly, the time spent on pth layer (not including 0th layer) is:

$$t_p = n \times r \tag{21}$$

so the total time in the SRNN is:

$$t_{SRNN} = (n \times k + \frac{T}{n^k}) \times r \tag{22}$$

At last we could compute the speed advantage of the SRNN:

$$R = \frac{t_{SRNN}}{t_{RNN}} = \frac{1}{n^k} + \frac{n \times k}{T} \tag{23}$$

where R is how much faster the SRNN gets. We could choose different n and k to get different speed advantage.

2.6 Relations between the SRNN and the Standard RNN

In this part we describe the relations between the SRNN and the standard RNN. In the standard RNN structure, each step is related to the input and the previous step, which could be computed by:

$$h_t = f(Ux_t + Wh_{t-1} + b) \tag{24}$$

where x is the input and h is the hidden state. The function f could be a nonlinear activation function such as sigmoid, or a linear activation function such as rectified linear units (Le et al., 2015). To simplify the question, here we discuss the case when we use a linear function:

$$f(x) = x \tag{25}$$

and we set the bias b and h_0 to be zero. When we use the standard RNN, we could get the last hidden state:

$$h_T = Ux_T + Wh_{T-1} = Ux_T + W(Ux_{T-1} + Wh_{T-2}) = ... = Ux_T + WUx_{T-1} + W^2Ux_{T-2} + ... + W^{T-1}Ux_1 \tag{26}$$

where T is the sequence length. And then we construct the SRNN structure. When $T = n^{k+1}$, we choose SRNN (n,k), which means slicing k times with the slice number n. The SRNN has k+1 layers, on each layer the length of each subsequence is n. We could compute the last hidden state of each minimum subsequence on 0th layer:

$$h_n^0 = U_0x_n + W_0U_0x_{n-1} + W_0^2U_0x_{n-2} + ... + W_0^{n-1}U_0x_1$$

$$h_{2n}^0 = U_0x_{2n} + W_0U_0x_{2n-1} + W_0^2U_0x_{2n-2} + ... + W_0^{n-1}U_0x_{n+1}$$

...

$$h_T^0 = U_0x_T + W_0U_0x_{T-1} + W_0^2U_0x_{T-2} + ... + W_0^{n-1}U_0x_{T-n+1} \tag{27}$$

where h_l^p is the number l hidden state on pth layer. There are n^k last hidden states on 0th layer. Similarly, we could take the hidden states on (p-1)th layer as the input, and compute the last hidden states of the subsequences on pth layer (p>0).

$$h_{n^{p+1}}^p = U_ph_{n^{p+1}}^{p-1} + W_pU_ph_{n^{p+1}-n^p}^{p-1} + W_p^2U_ph_{n^{p+1}-2n^p}^{p-1} + ... + W_p^{n-1}U_ph_{n^p}^{p-1}$$

$$h_{2n^{p+1}}^p = U_ph_{2n^{p+1}}^{p-1} + W_pU_ph_{2n^{p+1}-n^p}^{p-1} + W_p^2U_ph_{2n^{p+1}-2n^p}^{p-1} + ... + W_p^{n-1}U_ph_{n^{p+1}+n^p}^{p-1}$$

...

$$h_T^p = U_ph_T^{p-1} + W_pU_ph_{T-n^p}^{p-1} + W_p^2U_ph_{T-2n^p}^{p-1} + ... + W_p^{n-1}U_ph_{T-(n-1)n^p}^{p-1} \tag{28}$$

There are n^{k-p} last hidden states on pth layer. And this operation is repeated from 0th layer to kth layer. At last we get the final hidden state F of kth layer.

$$F = U_kh_T^{k-1} + W_kU_kh_{T-n^k}^{k-1} + W_k^2U_kh_{T-2n^k}^{k-1} + ... + W_k^{n-1}U_kh_{n^k}^{k-1} \tag{29}$$

When we compute equation (29) using each hidden state calculated by equation (27) and (28), we could get:

$$F = U_kU_{k-1}...U_0x_T + U_kU_{k-1}...W_0U_0x_{T-1} + U_kU_{k-1}...W_0^2U_0x_{T-2} + ... + W_k^{n-1}U_kW_{k-1}^{n-1}U_{k-1}...W_0^{n-1}U_0x_1 \tag{30}$$

When we compare equation (30) with equation (26), we could find that the two equations compute the same results when:

$$U_0 = U$$

$$U_k = U_{k-1} = \ldots = U_1 = I$$

$$W_k = W^{n^k} \tag{31}$$

where I is the identity matrix, U and W are the parameters in equation (26). This means that SRNNs could have the same output as standard RNNs when equation (31) is satisfied. For example, when $T = 4$, $k = 1$ and $n = 2$, we get a SRNN (2,1) with two layers and compare it with the standard RNN. The last hidden state of the standard RNN is:

$$h_4 = Ux_4 + WUx_3 + W^2Ux_2 + W^3Ux_1$$

In SRNN (2,1), the hidden states on 0th layer are:

$$h_2^0 = U_0x_2 + W_0U_0x_1$$

$$h_4^0 = U_0x_4 + W_0U_0x_3$$

and the final hidden state of 1st layer is:

$$F = U_1h_4^0 + W_1U_1h_2^0 = U_1U_0x_4 + U_1W_0U_0x_3 + W_1U_1U_0x_2 + W_1U_1W_0U_0x_1$$

When we use equation (31) to set $W_0 = W$, $W_1 = W^2$, $U_0 = U$ and $U_1 = I$, we could get:

$$F = Ux_4 + WUx_3 + W^2Ux_2 + W^3Ux_1$$

which is equal to h_4 above.

We have proved that when the function f is linear, the output of the SRNN is equal to the output of the standard RNN when the parameters satisfy equation (31), so the standard RNN is a special case of the SRNN. Furthermore, the SRNN may get high-level information when they have different parameters on different layers, so the SRNN is able to obtain more information from the input sequences than the standard RNN.

3 Experiments

3.1 Datasets

We evaluate SRNNs on six large-scale sentiment analysis datasets. Table 1 shows the information of the datasets. We choose 80% data for training, 10% for validation and 10% for testing.

Dataset	Classes	Documents	Max words	Average words	Vocabulary
Yelp 2013	5	468,608	1060	129.2	202,058
Yelp 2014	5	670,440	1053	116.1	210,353
Yelp 2015	5	897,835	1092	108.3	228,715
Yelp_P	2	598,000	1073	139.7	308,028
Amazon_F	5	3,650,000	441	82.7	1,274,916
Amazon_P	2	4,000,000	257	80.9	1,348,126

Table 1: Dataset information. Max words denotes the max sequence length, and Average words denotes the average length of the sentences in each dataset.

Yelp reviews: The Yelp reviews datasets are obtained from the Yelp Dataset Challenge, which has 5 sentiment labels (the higher, the better). This dataset consists of 4,736,892 documents, and we extract three subsets Yelp 2013, 2014, 2015 containing 468,608, 670,440 and 897,835 documents separately. Zhang et al. (2015) created the polarity dataset including 598,000 documents with two sentiment labels, and we obtain the polarity dataset from them.

Amazon reviews: The Amazon reviews dataset is a commentary dataset containing 34,686,770 reviews on 2,441,053 products from 6,643,669 users (He and McAuley, 2016). One review has a review title, a review content and a sentiment label, and we combine the title and content into one

document. The dataset is also constructed into a full dataset with 3,650,000 documents and a polarity dataset with 4,000,000 documents, which is also obtained from Zhang et al. (2015).

3.2 Baseline

We compare SRNNs with standard RNNs and take GRU as the recurrent unit. The output of the last hidden state is the representation of each document, and then we add a softmax layer on it to predict the sentiment labels. In order to compare SRNNs with convolutional structures, we also build a stack of dilated casual convolutional layers as a baseline, which is proposed in wavenet (Oord et al., 2016). The dilated casual convolutional structure could maintain the order information of the sequences. The dilation is 1, 2, 4, ... , 256 for Yelp datasets, and 1, 2, 4, ... , 128 for Amazon datasets. The number of filters is 50, and the activation function after each layer is rectified linear units.

3.3 Training

We use the sequence preprocessing tool of Keras (François et al, 2015) to pad sequences into the same length *T*. Sequences shorter than *T* are padded with zero at the end, and sequences longer than *T* are truncated. In this work, *T* is set to be 512 on Yelp datasets, and 256 on Amazon datasets. Different *n* and *k* values, which separately denotes the slice number and the slice times, are used on the experiments. For each dataset, we retain the top 30,000 words of the vocabulary. The pre-trained GloVe embeddings (Pennington et al., 2014) are utilized to initialize the word embeddings.

We set GRU as the recurrent unit of the SRNN. We have discussed the relations between the SRNN and the standard RNN in section 2.6 and have proved that the standard RNN is a special case of the SRNN when we use linear activation function between the recurrent units. However, it does not mean only linear activation function could be used in the SRNN. Both linear activation function such as hard sigmoid, and nonlinear activation function such as hyperbolic tangent could be used in or after each layer in the SRNN. In this paper, the recurrent activation function in GRU is sigmoid, and the activation function after each layer is linear function $f(x) = x$.

The dimension of GRU on each layer is set to be 50 and the word embedding dimension is 200. In SRNNs, the final state *F* also has 50 dimensions. We set the mini-batch size to be 100 for training and use Adam (Kingma and Ba, 2014) with $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$. We tune the hyper parameters on the validation set and select the best model to predict the sentiment labels on the test set. We train the models with an NVIDIA GTX 1080 GPU, and record the training time per epoch on each dataset.

3.4 Results and Analysis

The results on each dataset are shown on Table 2. We choose different *n* and *k* values and get different SRNNs. For example, SRNN (16,1) means *n*=16 and *k*=1, which could get a 32-length minimum subsequence when *T* is 512 or a 16-length minimum subsequence when *T* is 256. We compare four SRNNs with the standard RNN. For each dataset, we use bold words to label the highest-performing model and the fastest model.

Dataset	Model	Parameters	Validation	Test	Time/Epoch
Yelp 2013	GRU	5.76M	66.56	66.12	3172s
	SRNN (16,1)	5.77M	67.18	**67.03**	270s
	SRNN (8,2)	5.79M	67.11	66.80	**145s**
	SRNN (4,3)	5.80M	67.26	66.72	164s
	SRNN (2,8)	5.85M	66.30	66.41	204s
	DCCNN	5.78M	64.91	64.79	67s
Yelp 2014	GRU	5.76M	70.37	70.63	4142s
	SRNN (16,1)	5.77M	70.53	70.70	388s
	SRNN (8,2)	5.79M	70.35	**70.76**	**201s**
	SRNN (4,3)	5.80M	70.25	70.48	238s
	SRNN (2,8)	5.85M	69.50	69.70	284s
	DCCNN	5.78M	68.46	68.66	96s

Yelp 2015	GRU	5.76M	72.52	72.89	4434s
	SRNN (16,1)	5.77M	73.09	**73.50**	510s
	SRNN (8,2)	5.79M	72.84	73.30	319s
	SRNN (4,3)	5.80M	72.98	73.29	**309s**
	SRNN (2,8)	5.85M	72.37	72.75	367s
	DCCNN	5.78M	70.69	70.94	131s
Yelp_P	GRU	5.76M	96.02	95.96	3170s
	SRNN (16,1)	5.77M	95.83	95.92	401s
	SRNN (8,2)	5.79M	95.87	95.99	**205s**
	SRNN (4,3)	5.80M	95.90	**96.04**	236s
	SRNN (2,8)	5.85M	95.69	95.88	297s
	DCCNN	5.78M	95.03	95.26	98s
Amazon_F	GRU	5.76M	61.54	61.36	8953s
	SRNN (16,1)	5.77M	61.65	**61.65**	1584s
	SRNN (8,2)	5.79M	61.58	61.41	**1147s**
	SRNN (4,3)	5.80M	61.50	61.40	1166s
	SRNN (2,7)	5.85M	61.04	60.88	1344s
	DCCNN	5.78M	59.64	59.60	401s
Amazon_P	GRU	5.76M	95.27	95.22	11062s
	SRNN (16,1)	5.77M	95.29	**95.26**	2144s
	SRNN (8,2)	5.79M	95.21	95.18	**1309s**
	SRNN (4,3)	5.80M	95.12	95.12	1567s
	SRNN (2,7)	5.85M	94.98	95.02	1886s
	DCCNN	5.78M	94.72	94.69	448s

Table 2: The accuracy and training time on validation and test sets of the models on each dataset. Four different structures of SRNNs are constructed. DCCNN is dilated casual convolutional neural network, which is described in section 3.2.

The results show that SRNNs achieve better performance and attain much faster speeds than standard RNNs on all the datasets with few extra parameters. Different SRNNs have achieved the best performance on different datasets. SRNN (16,1) gets the highest accuracy on Yelp 2013, Yelp 2015, Amazon_F and Amazon_P; SRNN (8,2) performs best on Yelp 2014; SRNN (4,3) is the best on Yelp_P. SRNNs with k more than 1 could get nearly 15 times faster than standard RNNs on the Yelp datasets, and the speed advantage depends on k, n and T. In this work, SRNN (4,3) has the fastest speed on Yelp 2015, while SRNN (8,2) is the fastest on the rest (except DCCNN).

When we focus on the results of SRNN (2,8) on Yelp datasets and SRNN (2,7) on Amazon datasets, we could find that even if they did not achieve the best performance, they did not lose much accuracy. This means that SRNNs are able to transmit information through multiple layers, and because of this, SRNNs may achieve remarkable results when we train very long sequences. When n is 2, SRNN has the same number of layers as DCCNN, and it has much higher accuracy than DCCNN. So it means that the recurrent structure in SRNN is better than the dilated casual convolutional structure.

When we go back to equation (23), we may find that when n and k are not set to be too small, we could get much faster. In this work, we set $n = 8$, $k = q - 1$ when $T = 8^q$. We use an NVIDIA GTX 1080 GPU to train the models on 5120 documents, since the standard RNN would take too much time if we use more data. The training time is shown on Table 3.

We could get the amazing results from Table 3: the longer the sequence length is, the bigger speed advantage the SRNN achieves. When the sequence length is 32768, SRNN would take only 52s while the standard RNN would take nearly 2 hours. The SRNN is 136 times as fast as the standard RNN, and the speed advantage may be even bigger when longer sequences are used. Therefore, SRNNs may achieve much faster speeds on long-sequence tasks such as speech recognition, character-level text classification and language modeling.

Model	Sequence length		
	$8^3 = 512$	$8^4 = 4096$	$8^5 = 32768$
SRNN (8,2)	4s	-	-
SRNN (8,3)	-	9s	-
SRNN (8,4)	-	-	52s
GRU	54s	476s	7074s
Speed advantage	13.5x	52.9x	136.0x

Table 3: The training time and speed advantage on different sequence length. For each sequence length we choose a different SRNN structure.

3.5 Why SRNN

In this part we will discuss the advantages and importance of the SRNN. With the success of RNNs in many NLP tasks, many scholars have proposed different structures to increase the speed of RNNs. Most of the researches get faster by improving the recurrent units. However, the traditional connection structure has scarcely been questioned, in which each step is connected to its previous step. It is this connection structure that limits the speed of RNNs. The SRNN has improved the traditional connection method. Instead, a sliced structure is constructed to implement the parallelization of RNNs. The experimental results on six large-scale sentiment analysis datasets show that SRNNs achieve better performance than the standard RNN. The reasons are as follows:

(1) When we use the standard RNN connection structure, recurrent units with gated structures such as GRU and LSTM are useful, but they are not able to store all the important information when the sequences are very long. The SRNN, however, could divide the long sequence into many short subsequences and obtain the important information in short sequences. SRNNs are able to transmit the important information through the multiple-layers structure from 0th layer to the top layer.

(2) SRNNs have the ability to obtain high-level information from the sequences, instead of just the word-level information. When we use SRNN (8,2) in a document with 512 words, 0th layer may get the sentence-level information from the word embeddings, 1st layer may gain the paragraph-level information from the sentence-level information and 2nd layer could generate the final document-level representation from the paragraph-level information. The standard RNN, however, could only get the word-level information. Although it is impossible to have 8 paragraphs in each document, 8 sentences in each paragraph and 8 words in each sentence, the overall order information and structure information is uniform. Take the paragraph information as an example. People always express their opinions at the beginning or end of an article, and show examples in the middle of the article to explain their views. Compared to standard RNN, it is much easier for SRNNs to gain this information on the top layer.

(3) In terms of handling sequences, the SRNN is akin to the human brain mechanism. For example, if we, as humans, are given an article and asked to answer some questions about it, we usually do not need to read the whole article intensively. To answer the questions correctly, we try to locate the paragraph which mentions the specific information, and then find sentences and words in this paragraph that can answer the question. The SRNN could easily do this through multiple layers.

In addition to the improvement of accuracy, the most significant advantage of SRNNs is that SRNNs can be computed in parallel and achieve much faster speeds. Equation (23) computes the speed advantage of SRNN, and experiments of different sequence length also show that SRNNs could run much faster than standard RNNs. SRNNs could be even faster on longer sequences. As the Internet develops, hundreds of millions of data are generated every day, and SRNNs have devised new ways for us to handle these data.

4 Related Work

In order to increase the speed of RNNs, many scholars tried to improve the traditional RNN and achieved great results. Kim (2014), Kalchbrenner et al. (2014), and Gehring et al. (2017) tried to use CNNs in NLP tasks, which are usually used in computer vision (Lecun et al., 1998). Several structures get faster with the recurrent units changed (Greff et al., 2015; Balduzzi and Ghifary, 2016; Bradbury et al., 2017; Lei et al., 2017). As an overall structure improvement, the SRNN are related to these models,

because different types of recurrent units could be used in SRNNs. In this work we choose GRU (Cho et al., 2014), but other recurrent units are able to work in SRNNs as well.

The SRNN is related to the hierarchical structure proposed by Tang et al. (2015) and Yang et al. (2016). Tang et al. (2015) use CNN or LSTM to obtain the sentence representations, and then exploit gated RNN to generate the document representations. Yang et al. (2016) build a hierarchical network on both word-level and sentence-level, then use attention mechanism on both level. The difference between the SRNN and the hierarchical structure is that the documents do not need to be split into sentences when we use the SRNN, and the SRNN could have multiple layers. The hierarchical structure could be viewed as a special case of the SRNN, where k is 1 and all the sentences have equal length.

Several other architectures have been proposed to improve the connection structure of RNNs (Sutskever and Hinton, 2010; Koutnik et al., 2014; Chang et al., 2017). The SRNN is different from those architectures in the connection structure. The SRNN could be computed in parallel by slicing the sequences into many subsequences, but these models may still limit parallelization.

Also, the SRNN structure is similar to the overall structure of wavenet (Oord et al., 2016), which is used for audio generation. The difference between the SRNN and wavenet, which is also the most important innovation of the SRNN, is that we use recurrent units on each layer. For sequences, recurrent structure has its inherent advantages than convolutional structure.

5 Conclusion

In this paper we present the sliced recurrent neural network (SRNN), which is an overall structure improvement of RNN. The SRNN could reach a remarkably faster speed than the standard RNN and achieve better performance on six large-scale sentiment datasets.

6 Future Work

The SRNN has been successful in text classification. In future work, we hope to promote it to other NLP applications, such as question answering, text summarization, and machine translation. In sequence to sequence model, the SRNN can be used as the encoder, and the decoder may be improved by using an inverse SRNN structure. Also, we hope to use the SRNN in several long sequence tasks, such as language model, music generation and audio generation. And we want to explore more about variants of the SRNN. For example, bidirectional structure and attention mechanism could be added.

In section 2.6, we have discussed the relations between the SRNN and the standard RNN when choosing the linear activation function. In future work, we will try to research the situation of using nonlinear recurrent activation functions by mathematics, though it may be harder.

Acknowledgements

Special thanks to Linjuan Wu for her help in polishing the language. Also, we would like to thank the anonymous reviewers for their suggestions. This work is supported by the National Natural Science Foundation of China (Grant No. 61772337, 61472248, and U1736207) and the SJTU-Shanghai Songheng Information Analysis Joint Lab.

Reference

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

David Balduzzi and Muhammad Ghifary. 2016. Strongly-typed recurrent neural networks. In *Proceedings of 33th International Conference on Machine Learning*.

James Bradbury and Richard Socher. 2016. MetaMind neural machine translation system for WMT 2016. In *Proceedings of the First Conference on Machine Translation: Shared Task Papers-Volume 2*, pages 264-267. Association for Computational Linguistics.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. Quasi-recurrent neural networks. In *Proceedings of the International Conference on Learning Representations*.

Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark Hasegawa-Johnson, and Thomas S. Huang. 2017. Dilated recurrent neural networks. In *Advances in Neural Information Processing Systems,* pages 76-86.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*.

Kyunghyun Cho, Bart van Merriënboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Chollet François and others. 2015. Keras. *https://github.com/keras-team/keras.* GitHub.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2015. Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069*.

Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *Proceedings of the 25th international conference on world wide web*, pages 507-517.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735-1780. ISSN 0899-7667.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1.* Association for Computational Linguistics.

Andrej Karpathy and Fei-Fei Li. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128-3137.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the Empiricial Methods in Natural Language Processing*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jan Koutnik, Klaus Greff, Faustino Gomez, and Jurgen Schmidhuber. 2014. A clockwork rnn. In *Proceedings of International Conference on Machine Learning*, pages 1863-1871.

Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 86(11):2278-2324.

Tao Lei, Yu Zhang, and Yoav Artzi. 2017. Training RNNs as Fast as CNNs. *arXiv preprint arXiv:1709.02755*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attentionbased neural machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499.*

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing*, pages 1532-1543.

Ilya Sutskever and Geoffrey Hinton. 2010. Temporal-kernel recurrent neural networks. In *Neural Networks*, 23(2):239-243.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on Empirical Methods in Natural Language Processing*, pages 1422-1432.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *Proceedings of 33th International Conference on Machine Learning*, pages 2397-2406.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2048-2057.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical Attention Networks for Document Classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480-1489.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649-657.

Guo-Bing Zhou, Jianxin Wu, Chen-Lin Zhang, and Zhi-Hua Zhou. Minimal gated unit for recurrent neural networks. 2016. In *International Journal of Automation and Computing*, 13(3):226-234.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 207-212. Association for Computational Linguistics.

Multi-Task Learning for Sequence Tagging: An Empirical Study

Soravit Changpinyo, Hexiang Hu, and **Fei Sha**
Department of Computer Science
University of Southern California
Los Angeles, CA 90089
schangpi,hexiangh,feisha@usc.edu

Abstract

We study three general multi-task learning (MTL) approaches on 11 sequence tagging tasks. Our extensive empirical results show that in about 50% of the cases, jointly learning all 11 tasks improves upon either independent or pairwise learning of the tasks. We also show that pairwise MTL can inform us what tasks can benefit others or what tasks can be benefited if they are learned jointly. In particular, we identify tasks that can always benefit others as well as tasks that can always be harmed by others. Interestingly, one of our MTL approaches yields embeddings of the tasks that reveal the natural clustering of semantic and syntactic tasks. Our inquiries have opened the doors to further utilization of MTL in NLP.

1 Introduction

Multi-task learning (MTL) has long been studied in the machine learning literature, cf. (Caruana, 1997). The technique has also been popular in NLP, for example, in (Collobert and Weston, 2008; Collobert et al., 2011; Luong et al., 2016). The main thesis underpinning MTL is that solving many tasks together provides a shared inductive bias that leads to more robust and generalizable systems. This is especially appealing for NLP as data for many tasks are scarce — shared learning thus reduces the amount of training data needed. MTL has been validated in recent work, mostly where auxiliary tasks are used to improve the performance on a target task, for example, in sequence tagging (Søgaard and Goldberg, 2016; Bjerva et al., 2016; Plank et al., 2016; Alonso and Plank, 2017; Bingel and Søgaard, 2017).

Despite those successful applications, several key issues about the effectiveness of MTL remain open. Firstly, with only a few exceptions, much existing work focuses on "pairwise" MTL where there is a target task and one or several (carefully) selected auxiliary tasks. However, *can jointly learning many tasks benefit all of them together*? A positive answer will significantly raise the utility of MTL. Secondly, *how are tasks related such that one could benefit another?* For instance, one plausible intuition is that syntactic and semantic tasks might benefit among their two separate groups though cross-group assistance is weak or unlikely. However, such notions have not been put to test thoroughly on a significant number of tasks.

In this paper, we address such questions. We investigate learning jointly multiple sequence tagging tasks. Besides using independent single-task learning as a baseline and a popular shared-encoder MTL framework for sequence tagging (Collobert et al., 2011), we propose two variants of MTL, where both the encoder and the decoder could be shared by all tasks.

We conduct extensive empirical studies on 11 sequence tagging tasks — we defer the discussion on why we select such tasks to a later section. We demonstrate that there is a benefit to moving beyond "pairwise" MTL. We also obtain interesting pairwise relationships that reveal which tasks are beneficial or harmful to others, and which tasks are likely to be benefited or harmed. We find such information correlated with the results of MTL using more than two tasks. We also study selecting only benefiting tasks for joint training, showing that such a "greedy" approach in general improves the MTL performance, highlighting the need of identifying with whom to jointly learn.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2965–2977
Santa Fe, New Mexico, USA, August 20-26, 2018.

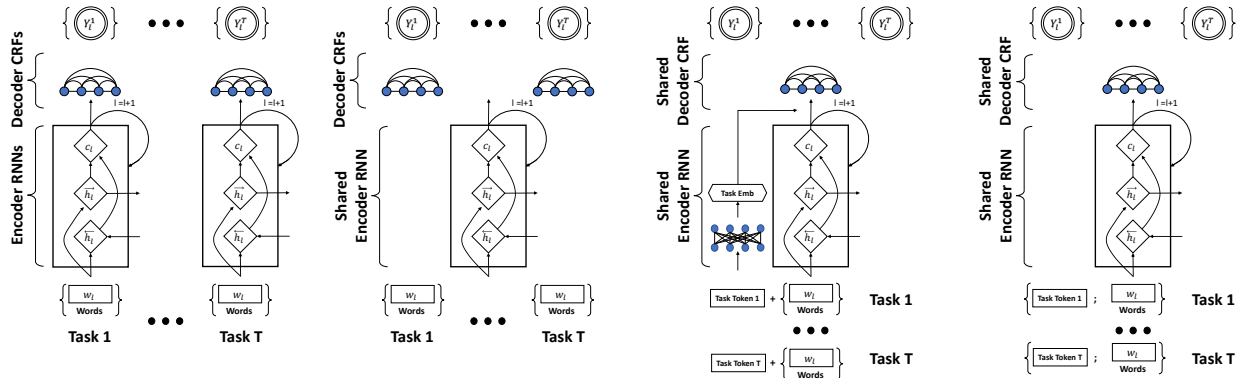
The rest of the paper is organized as follows. We describe different approaches for learning from multiple tasks in Sect. 2. We describe our experimental setup and results in Sect. 3 and Sect. 4, respectively. We discuss related work in Sect. 5. Finally, we conclude with discussion and future work in Sect. 6.

2 Multi-Task Learning for Sequence Tagging

In this section, we describe general approaches to multi-task learning (MTL) for sequence tagging. We select sequence tagging tasks for several reasons. Firstly, we want to concentrate on comparing the tasks themselves without being confounded by designing specialized MTL methods for solving complicated tasks. Sequence tagging tasks are done at the word level, allowing us to focus on simpler models while still enabling varying degrees of sharing among tasks. Secondly, those tasks are often the first steps in NLP pipelines that come with extremely diverse resources. Understanding the nature of the relationships between them is likely to have a broad impact on many downstream applications.

Let T be the number of tasks and $\boldsymbol{D}^t$ be training data of task $t \in \{1, \ldots, \mathsf{T}\}$. A dataset for each task consists of input-output pairs. In sequence tagging, each pair corresponds to a sequence of words $w_{1:\mathsf{L}}$ and their corresponding ground-truth tags $y_{1:\mathsf{L}}$, where L is the sequence length. We note that our definition of "task" is not the same as "domain" or "dataset." In particular, we differentiate between tasks based on whether or not they share the label space of tags. For instance, part-of-speech tagging on weblog and that on email domains are considered the same task in this paper.

Given the training data $\{\boldsymbol{D}^1, \ldots, \boldsymbol{D}^{\mathsf{T}}\}$, we describe how to learn one or more models to perform all the T tasks. In general, our models follow the design of state-of-the-art sequence taggers (Reimers and Gurevych, 2017). They have an encoder e with parameters θ that encodes a sequence of word tokens into a sequence of vectors and a decoder d with parameters ϕ that decodes the sequence of vectors into a sequence of predicted tags $\hat{y}_{1:\mathsf{L}}$. That is, $c_{1:\mathsf{L}} = e(w_{1:\mathsf{L}}; \theta)$ and $\hat{y}_{1:\mathsf{L}} = d(c_{1:\mathsf{L}}; \phi)$. The model parameters are learned by minimizing some loss function $\mathcal{L}(\hat{y}_{1:\mathsf{L}}, y_{1:\mathsf{L}})$ over θ and ϕ. In what follows, we will use superscripts to differentiate instances from different tasks.



Single-task learning | MTL (**MULTI-DEC**) | MTL (**TE⊕DEC**) | MTL (**TE⊕ENC**)

Figure 1: Different settings for learning from multiple tasks considered in our experiments

In **single-task learning (STL)**, we learn T models independently. For each task t, we have an encoder $e^t(\cdot; \theta^t)$ and a decoder $d^t(\cdot; \phi^t)$. Clearly, information is not shared between tasks in this case.

In **multi-task learning (MTL)**, we consider two or more tasks and train an MTL model *jointly* over a combined loss $\sum_t \mathcal{L}(\hat{y}^t_{1:\mathsf{L}}, y^t_{1:\mathsf{L}})$. In this paper, we consider the following general frameworks that are different in the nature of how the parameters of those tasks are shared.

Multi-task learning with multiple decoders (Multi-Dec) We learn a shared encoder $e(\cdot; \theta)$ and T decoders $\{d^t(\cdot; \theta^t)\}_{t=1}^{\mathsf{T}}$. This setting has been explored for sequence tagging in (Collobert and Weston, 2008; Collobert et al., 2011). In the context of sequence-to-sequence learning (Sutskever et al., 2014), this is similar to the "one-to-many" MTL setting in (Luong et al., 2016).

Multi-task learning with task embeddings (TE) We learn a shared encoder $e(\cdot; \theta)$ for the input sentence as well as a shared decoder $d(\cdot; \phi)$. To equip our model with the ability to perform one-to-many mapping (i.e., multiple tasks), we augment the model with "task embeddings." Specifically, we additionally learn

Dataset	# sentences	Token/type	Task	# labels	Label entropy
Universal Dependencies v1.4	12543/16622	12.3/13.2	UPOS XPOS	17 50	2.5 3.1
CoNLL-2000	8936/10948	12.3/13.3	CHUNK	42	2.3
CoNLL-2003	14041/20744	9.7/11.2	NER	17	0.9
Streusle 4.0	2723/3812	8.6/9.3	MWE SUPSENSE	3 212	0.5 2.2
SemCor	13851/20092	13.2/16.2	SEM SEMTR	75 11	2.2 1.3
Broadcast News 1	880/1370	5.2/6.1	COM	2	0.6
FrameNet 1.5	3711/5711	8.6/9.1	FRAME	2	0.5
Hyper-Text Corpus	2000/3974	6.7/9.0	HYP	2	0.4

Table 1: Datasets used in our experiments, as well as their key characteristics and their corresponding tasks. / is used to separate statistics for training data only and those for all subsets of data.

a function $f(t)$ that maps a task ID t to a vector. We explore two ways of injecting task embeddings into models. In both cases, our f is simply an embedding layer that maps the task ID to a dense vector.

One approach, denoted by **TE⊕DEC**, is to incorporate task embeddings into the decoder. We concatenate the task embeddings with the encoder's outputs $c^t_{1:\mathsf{L}}$ and then feed the result to the decoder.

The other approach, denoted by **TE⊕ENC**, is to combine the task embeddings with the input sequence of words at the encoder. We implement this by prepending the task token (`<<upos>>`, `<<chunk>>`, `<<mwe>>`, etc.) to the input sequence and treat the task token as a word token (Johnson et al., 2017).

While the encoder in **TE⊕DEC** must learn to encode a general-purpose representation of the input sentence, the encoder in **TE⊕ENC** knows from the start which task it is going to perform.

Fig. 1 illustrates different settings described above. Clearly, the number of model parameters is reduced significantly when we move from STL to MTL. Which MTL model is more economical depends on several factors, including the number of tasks, the dimension of the encoder output, the general architecture of the decoder, the dimension of task embeddings, how to augment the system with task embeddings, and the degree of tagset overlap.

3 Experimental Setup

3.1 Datasets and Tasks

Table 1 summarizes the datasets used in our experiments, along with their corresponding tasks and important statistics. Table 2 shows an example of each task's input-output pairs. We describe details below. For all tasks, we use the standard splits unless specified otherwise.

We perform universal and English-specific POS tagging (UPOS and XPOS) on sentences from the English Web Treebank (Bies et al., 2012), annotated by the Universal Dependencies project (Nivre et al., 2016). We perform syntactic chunking (CHUNK) on sentences from the WSJ portion of the Penn Treebank (Marcus et al., 1993), annotated by the CoNLL-2000 shared task (Tjong Kim Sang and Buchholz, 2000). We use sections 15-18 for training. The shared task uses section 20 for testing and does not designate the development set, so we use the first 1001 sentences for development and the rest 1011 for testing. We perform named entity recognition (NER) on sentences from the Reuters Corpus (Lewis et al., 2004), consisting of news stories between August 1996-97, annotated by the CoNLL-2003 shared task (Tjong Kim Sang and De Meulder, 2003). For both CHUNK and NER, we use the IOBES tagging scheme.

We perform multi-word expression identification (MWE) and supersense tagging (SUPSENSE) on sentences from the reviews section of the English Web Treebank, annotated under the Streusle project (Schneider and Smith, 2015)[1]. We perform supersense (SEM) and semantic trait (SEMTR) tagging on SemCor's sentences (Landes et al., 1998), taken from a subset of the Brown Corpus (Francis and Kučera, 1982), using the splits provided by (Alonso and Plank, 2017) for both tasks[2]. For SEM, they are annotated with supersense tags (Miller et al., 1993) by (Ciaramita and Altun, 2006)[3]. For SEMTR, (Alonso and Plank, 2017) uses the EuroWordNet list of ontological types for senses (Vossen et al., 1998) to convert supersenses into coarser semantic traits.

[1] https://github.com/nert-gu/streusle

[2] https://github.com/bplank/multitasksemantics

[3] We consider SUPSENSE and SEM as different tasks as they use different sets of supersense tags.

Task	Input/Output
UPOS	once again , thank you all for an outstanding accomplishment . ADV ADV PUNCT VERB PRON DET ADP DET ADJ NOUN PUNCT
XPOS	once again , thank you all for an outstanding accomplishment . RB RB , VBP PRP DT IN DT JJ NN .
CHUNK	the carrier also seemed eager to place blame on its american counterparts . B-NP E-NP S-ADVP S-VP S-ADJP B-VP E-VP S-NP S-PP B-NP I-NP E-NP O
NER	6. pier francesco chili (italy) ducati 17541 O B-PER I-PER E-PER O S-LOC O S-ORG O
MWE	had to keep in mind that the a / c broke , i feel bad it was their opening ! B I B I I O O B I I O O O O O O O O O
SUPSENSE	this place may have been something sometime ; but it way past it " sell by date " . O n.GROUP O O v.stative O O O O O O p.Time p.Gestalt O v.possession p.Time n.TIME O O
SEM	a hypothetical example will illustrate this point . O adj.all noun.cognition O verb.communication O noun.communication O
SEMTR	he wondered if the audience would let him finish . O Mental O O Object O Agentive O BoundedEvent O
COM	he made the decisions in 1995 , in early 1996 , to spend at a very high rate . KEEP KEEP DEL KEEP DEL DEL DEL DEL DEL DEL DEL KEEP KEEP KEEP KEEP DEL KEEP KEEP KEEP
FRAME	please continue our important partnership . O B-TARGET O B-TARGET O O
HYP	will this incident lead to a further separation of civilizations ? O O O O O O O B-HTML B-HTML B-HTML O

Table 2: Examples of input-output pairs of the tasks in consideration

For sentence compression (COM), we identify which words to keep in a compressed version of sentences from the 1996 English Broadcast News Speech (HUB4) (Graff, 1997), created by (Clarke and Lapata, 2006)[4]. We use the labels from the first annotator. For frame target identification (FRAME), we detect words that evoke frames (Das et al., 2014) on sentences from the British National Corpus, annotated under the FrameNet project (Baker et al., 1998). For both COM and FRAME, we use the splits provided by (Bingel and Søgaard, 2017). For hyper-link detection (HYP), we identify which words in the sequence are marked with hyperlinks on text from Daniel Pipes' news-style blog collected by (Spitkovsky et al., 2010)[5]. We use the "select" subset that correspond to marked, complete sentences.

3.2 Metrics and Score Comparison

We use the span-based micro-averaged F1 score (without the O tag) for all tasks. We run each configuration three times with different initializations and compute mean and standard deviation of the scores. To compare two scores, we use the following strategy. Let μ_1, σ_1 and μ_2, σ_2 be two sets of scores (mean and std, respectively). We say that μ_1 is "higher" than μ_2 if $\mu_1 - k \times \sigma_1 > \mu_2 + k \times \sigma_2$, where k is a parameter that controls how strict we want the definition to be. "lower" is defined in the same manner with $>$ changed to $<$ and $-$ switched with $+$. k is set to 1.5 in all of our experiments.

3.3 Models

General architectures We use bidirectional recurrent neural networks (biRNNs) as our encoders for both words and characters (Irsoy and Cardie, 2014; Huang et al., 2015; Lample et al., 2016; Ma and Hovy, 2016). Our word/character sequence encoders and decoder classifiers are common in literature and most similar to (Lample et al., 2016), but we use two-layer biRNNs (instead of one) with Gated Recurrent Unit (GRU) (Cho et al., 2014) (instead of with LSTM (Hochreiter and Schmidhuber, 1997)).

Each word is represented by a 100-dimensional vector that is the concatenation of a 50-dimensional *embedding* vector and the 50-dimensional output of a *character* biRNN (whose hidden representation dimension is 25 in each direction). We feed a sequence of those 100-dimensional representations to a *word* biRNN, whose hidden representation dimension is 300 in each direction, resulting in a sequence of 600-dimensional vectors. In **TE⊕DEC**, the token encoder is also used to encode a task token (which is then concatenated to the encoder's output), where each task is represented as a 25-dimensional vector. For decoder/classifiers, we predict a sequence of tags using a linear projection layer (to the tagset size) followed by a conditional random field (CRF) (Lafferty et al., 2001).

Implementation and training details We implement our models in PyTorch (Paszke et al., 2017) on top of the AllenNLP library (Gardner et al., 2018). Code is to be available at https://github.

[4] http://jamesclarke.net/research/resources/

[5] https://nlp.stanford.edu/valentin/pubs/markup-data.tar.bz2

com/schangpi/.

Words are lower-cased, but characters are not. Word embeddings are initialized with GloVe (Pennington et al., 2014) trained on Wikipedia 2014 and Gigaword 5. We use strategies suggested by (Ma and Hovy, 2016) for initializing other parameters in our networks. Character embeddings are initialized uniformly in $[-\sqrt{3/d}, \sqrt{3/d}]$, where d is the dimension of the embeddings. Weight matrices are initialized with Xavier Uniform (Glorot and Bengio, 2010), i.e., uniformly in $[-\sqrt{6/(r+c)}, \sqrt{6/(r+c)}]$, where r and c are the number of of rows and columns in the structure. Bias vectors are initialized with zeros.

We use Adam (Kingma and Ba, 2015) with default hyperparameters and a mini-batch size of 32. The dropout rate is 0.25 for the character encoder and 0.5 for the word encoder. We use gradient normalization (Pascanu et al., 2013) with a threshold of 5. We halve the learning rate if the validation performance does not improve for two epochs, and stop training if the validation performance does not improve for 10 epochs. We use L2 regularization with parameter 0.01 for the transition matrix of the CRF.

For the training of MTL models, we make sure that each mini-batch is balanced; the difference in numbers of examples from any pair of tasks is no more than 1. As a result, each epoch may not go through all examples of some tasks whose training set sizes are large. In a similar manner, during validation, the average F1 score is over all tasks rather than over all validation examples.

3.4 Various Settings for Learning from Multiple Tasks

We consider the following settings: (i) "STL" where we train each model on one task alone; (ii) "Pairwise MTL" where we train on two tasks jointly; (iii) "All MTL" where we train on all tasks jointly; (iv) "Oracle MTL" where we train on the Oracle set of the testing task jointly with the testing task; (v) "All-but-one MTL" setting where we train on all tasks jointly except for one (as part of Sect. 4.4.)

Constructing the Oracle Set of a Testing Task The Oracle set of a task t is constructed from the pairwise performances: let $\mu(A,t), \sigma(A,t)$ be the F1 score and the standard deviation of a model that is jointly trained on a set of tasks in the set A and that is tested on task t. Task s is considered "beneficial" to another (testing) task t if $\mu(\{s,t\},t)$ is "higher" than $\mu(\{t\},t)$ (cf. Sect. 3.2). Then, the "Oracle" set for a task t is the set of its all beneficial (single) tasks. Throughout our experiments, we compute μ and σ by averaging over three rounds (cf. Sect. 3.2, standard deviations can be found on the arXiv version.)

4 Results and Analysis

4.1 Main Results

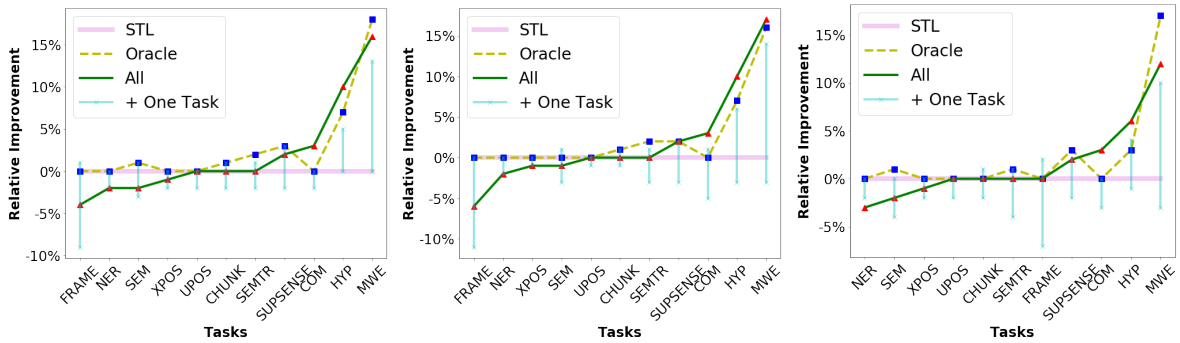


Figure 2: Summary of our results for MTL methods **MULTI-DEC** (left), **TE⊕DEC** (middle), and **TE⊕ENC** (right) on various settings with different types of sharing. The vertical axis is the relative improvement over STL. See texts for details. Best viewed in color.

Fig. 2 summarizes our main findings. We compare relative improvement over single-task learning (STL) between various settings with different types of sharing in Sect. 3.4. Scores from the pairwise setting ("+One Task") are represented as a vertical bar, delineating the maximum and minimum improvement over STL by jointly learning a task with one of the remaining 10 tasks. The "All" setting (red triangles) indicates the joint learning all 11 tasks. The "Oracle" setting (blue rectangles) indicates the joint learning using a subset of 11 tasks which are deemed beneficial, based on corresponding performances in pairwise MTL, as defined in Sect. 3.4.

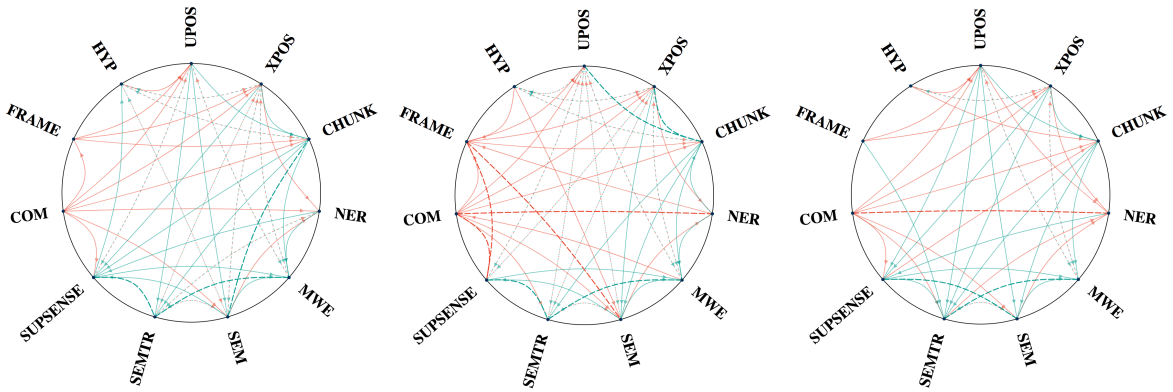


Figure 3: Pairwise MTL relationships (benefit vs. harm) using **MULTI-DEC** (left), **TE⊕DEC** (middle), and **TE⊕ENC** (right). **Solid** green (red) directed edge from s to t denotes s benefiting (harming) t. **Dashed** Green (Red) edges between s and t denote they benefiting (harming) *each other*. **Dotted** edges denote asymmetric relationship: benefit in one direction but harm in the reverse direction. Absence of edges denotes neutral relationships. *Best viewed in color and with a zoom-in.*

We observe that (1) [STL vs. Pairwise/All] Neither pairwise MTL nor All always improves upon STL; (2) [STL vs. Oracle] Oracle in general outperforms or at least does not worsen STL; (3) [All/Oracle vs. Pairwise] All does better than Pairwise on about half of the cases, while Oracle almost always does better than Pairwise; (4) [All vs. Oracle] Consider when both All and Oracle improve upon STL. For **MULTI-DEC** and **TE⊕ENC**, Oracle generally dominates All, except on the task HYP. For **TE⊕DEC**, their magnitudes of improvement are mostly comparable, except on SEMTR (Oracle is better) and on HYP (All is better). In addition, All is better than Oracle on the task COM, in which case Oracle is STL.

In the arXiv version, we compare different MTL approaches: **MULTI-DEC**, **TE⊕DEC**, and **TE⊕ENC**. There is no significant difference among them.

4.2 Pairwise MTL results

Summary The summary plot in Fig. 3 gives a bird's-eye view of patterns in which a task might benefit or harm another one. For example, MWE is always benefited from jointly learning any of the 10 tasks as the *incoming edges* are green, so is SEMTR in most cases. On the other end, COM seems to be harming any of the 10 as the *outgoing edges* are almost always red. For CHUNK and U/XPOS, it generally benefits others (or at least does not do harm) as most of their *outgoing* edges are green.

In Table 3-5, we report F1 scores for **MULTI-DEC**, **TE⊕DEC**, and **TE⊕ENC**, respectively. In each table, rows denote settings in which we train our models, and columns correspond to tasks we test them on. We also include "Average" of all pairwise scores, as well as the number of positive (↑) and negative (↓) relationships in each row or each column.

Which tasks are benefited or harmed by others in pairwise MTL? MWE, SUPSENSE, SEMTR, and HYP are generally benefited by other tasks. The improvement is more significant in MWE and HYP. UPOS, XPOS, NER, COM, and FRAME (**MULTI-DEC** and **TE⊕DEC**) are often hurt by other tasks. Finally, the results are mixed for CHUNK and SEM.

Which tasks are beneficial or harmful? UPOS, XPOS, and CHUNK are universal helpers, beneficial in 16, 17, and 14 cases, while harmful only in 1, 3, and 0 cases, respectively. Interestingly, CHUNK never hurts any task, while both UPOS and XPOS can be harmful to NER. While CHUNK is considered more of a syntactic task, the fact that it informs other tasks about the boundaries of phrases may aid the learning of other semantic tasks (task embeddings in Sect. 4.4 seem to support this hypothesis).

On the other hand, COM, FRAME, and HYP are generally harmful, all useful in 0 cases and causing the performance drop in 22, 10, 12 cases, respectively. One factor that may play a role is the training set sizes of these tasks. However, we note that both MWE and SUPSENSE (Streusle dataset) has smaller training set sizes than FRAME does, but those tasks can still benefit some tasks. (On the other hand, NER has the largest training set, but infrequently benefits other tasks, less frequently than SUPSENSE does.) Another potential cause is the fact that all those harmful tasks have the smallest label size of 2. This combined with small dataset sizes leads to a higher chance of overfitting. Finally, it may be possible that harmful

tasks are simply unrelated; for example, the nature of COM, FRAME, or HYP may be very different from other tasks — an entirely different kind of reasoning is required.

Finally, NER, MWE, SEM, SEMTR, and SUPSENSE can be beneficial or harmful, depending on which other tasks they are trained with.

	UPOS	XPOS	CHUNK	NER	MWE	SEM	SEMTR	SUPSENSE	COM	FRAME	HYP	#↑	#↓
+UPOS	95.4	95.01	94.18 ↑	87.68	59.99 ↑	73.23 ↑	74.93 ↑	68.25 ↑	72.46	62.14	48.02	5	0
+XPOS	95.38	95.04	93.97 ↑	87.61 ↓	58.87 ↑	73.34 ↑	74.91 ↑	67.78 ↑	72.83	60.77	48.81 ↑	6	1
+CHUNK	95.43	95.1	93.49	87.96	59.18 ↑	73.16 ↑	74.79 ↑	67.39 ↑	72.44	62.67	47.85 ↑	5	0
+NER	95.38	94.98	93.47	88.24	55.4 ↑	72.88	74.34 ↑	68.06 ↑	70.93	62.39	47.9	3	0
+MWE	95.15 ↓	94.7 ↓	93.54	88.15	53.07	72.75	74.51 ↑	66.88	71.31	61.75	47.32	1	2
+SEM	95.23	94.77 ↓	93.63 ↑	87.35 ↓	60.16 ↑	72.77	74.73 ↑	68.29 ↑	72.72	61.74	48.15 ↑	5	2
+SEMTR	95.17	94.86 ↓	93.61	87.34 ↓	58.84 ↑	72.5 ↓	74.02	68.6 ↑	71.96	62.03	47.74	2	3
+SUPSENSE	95.08 ↓	94.75	93.2	87.9	58.81 ↑	72.81	74.61 ↑	66.81	72.24	61.94	49.23 ↑	3	1
+COM	93.04 ↓	93.19 ↓	91.94 ↓	86.62 ↓	53.89	70.39 ↓	72.6	65.57 ↓	72.71	56.52 ↓	47.41	0	7
+FRAME	94.98 ↓	94.64 ↓	93.22 ↓	88.15	53.88	72.76	74.18	66.59	72.47	62.04	47.5	0	3
+HYP	94.84 ↓	94.46 ↓	92.96 ↓	87.98	53.08	72.47 ↓	74.23	66.47	71.82	61.02	46.73	0	4
#↑	0	0	3	0	7	3	7	6	0	0	4		
#↓	5	6	3	4	0	3	0	1	0	1	0		
Average	94.97	94.65	93.37	87.67	57.21	72.63	74.38	67.39	72.12	61.3	47.99		
All	95.04 ↓	94.31 ↓	93.44	86.38 ↓	61.43 ↑	71.53 ↓	74.26 ↑	68.1 ↑	74.54	59.71	51.41 ↑	4	4
Oracle	95.4	95.04	94.01 ↑	88.24	62.76 ↑	73.32 ↑	75.23 ↑	68.53 ↑	72.71	62.04	50.0 ↑	6	0

Table 3: F1 scores for **MULTI-DEC**. We compare STL setting (blue), with pairwise MTL (+⟨task⟩), All, and Oracle. We test on each task in the columns. Beneficial settings are in green. Harmful setting are in red. The last two columns indicate how many tasks are helped or harmed by the task at that row.

	UPOS	XPOS	CHUNK	NER	MWE	SEM	SEMTR	SUPSENSE	COM	FRAME	HYP	#↑	#↓
+UPOS	95.4	94.99	94.02 ↑	87.99	60.28 ↑	73.17 ↑	74.87 ↑	67.8 ↑	72.86	61.54	49.36 ↑	6	0
+XPOS	95.4	95.04	94.18 ↑	87.65 ↓	60.32 ↑	73.21 ↑	74.84 ↑	68.3 ↑	72.87	61.44	49.23 ↑	6	1
+CHUNK	95.57 ↑	95.21 ↑	93.49	88.11	57.61 ↑	73.02 ↑	74.73 ↑	67.29	73.3	61.39	48.43 ↑	6	0
+NER	95.32	95.09	93.64 ↑	88.24	55.17 ↑	72.77	74.01	67.25	71.08 ↓	59.25 ↓	48.24	2	2
+MWE	95.11 ↓	94.8 ↓	93.59	87.99	53.07	72.66	74.63 ↑	66.88	70.93 ↓	56.77	45.83	1	3
+SEM	95.2 ↓	94.82	93.45	87.27 ↓	58.21 ↑	72.77	74.72 ↑	68.46 ↑	73.14	60.09 ↓	47.95	3	3
+SEMTR	95.21 ↓	94.8 ↓	93.47	87.75	58.55 ↑	72.5 ↓	74.02	68.18 ↑	71.74	59.77	46.96	2	3
+SUPSENSE	95.05 ↓	94.81 ↓	93.25	87.94	58.75 ↑	72.71	74.52 ↑	66.81	69.13 ↓	55.68 ↓	47.29	2	4
+COM	94.03 ↓	93.94 ↓	92.29 ↓	86.59 ↓	51.72	70.37 ↓	71.76 ↓	64.98 ↓	72.71	55.25 ↓	45.24	0	8
+FRAME	94.79 ↓	94.66 ↓	93.23 ↓	88.02	53.05	72.26 ↓	74.21	66.2 ↓	72.89	62.04	46.0	0	5
+HYP	94.35 ↓	94.56 ↓	92.86 ↓	87.91	52.98	72.15 ↓	74.19	66.52	70.47	55.35 ↓	46.73	0	5
#↑	1	1	3	0	7	3	6	4	0	0	3		
#↓	7	6	3	3	0	4	1	2	3	5	0		
Average	95.0	94.77	93.4	87.72	56.67	72.48	74.25	67.19	71.84	58.65	47.45		
All	94.95 ↓	94.42 ↓	93.64	86.8 ↓	61.97 ↑	71.72 ↓	74.36 ↑	67.98 ↑	74.61 ↑	58.14 ↓	51.31 ↑	5	5
Oracle	95.57 ↑	95.21 ↑	94.07 ↑	88.24	61.74 ↑	73.1 ↑	75.24 ↑	68.22 ↑	72.71	62.04	50.15 ↑	8	0

Table 4: F1 scores for **TE⊕DEC**. We compare STL setting (blue), with pairwise MTL (+⟨task⟩), All, and Oracle. We test on each task in the columns. Beneficial settings are in green. Harmful setting are in red. The last two columns indicate how many tasks are helped or harmed by the task at that row.

4.3 All MTL Results

In addition to pairwise MTL results, we report the performances in the All and Oracle MTL settings in the last two rows of Table 3-5. We find that their performances depend largely on the trend in their corresponding pairwise MTL. We provide examples and discussion of such observations below.

How much is STL vs. Pairwise MTL predictive of STL vs. All MTL? We find that the performance of pairwise MTL is predictive of the performance of All MTL to some degree. Below we discuss the results in more detail. Note that we would like to be predictive in both the performance direction and magnitude (whether and how much the scores will improve or degrade over the baseline).

When pairwise MTL improves upon STL even slightly, All improves upon STL in all cases (MWE, SEMTR, SUPSENSE, and HYP). This is despite the fact that jointly learning some pairs of tasks lead to performance degradation (COM and FRAME in the case of SUPSENSE and COM in the case of SEMTR). Furthermore, when pairwise MTL leads to improvement in all cases (all pairwise rows in MWE and HYP), All MTL will achieve even better performance, suggesting that tasks are beneficial in a complementary manner and there is an advantage of MTL beyond two tasks.

The opposite is almost true. When pairwise MTL does not improve upon STL, most of the time All MTL will not improve upon STL, either — with one exception: COM. Specifically, the pairwise MTL performances of UPOS, XPOS, NER and FRAME (**TE⊕DEC**) are mostly negative and so are their All

	UPOS	XPOS	CHUNK	NER	MWE	SEM	SEMTR	SUPSENSE	COM	FRAME	HYP	#↑	#↓
+UPOS	95.4	94.94	94.0 ↑	87.43 ↓	57.61 ↑	73.11 ↑	74.85 ↑	67.76 ↑	72.09	62.27	48.27	5	1
+XPOS	95.42	95.04	93.98 ↑	87.71 ↓	58.26 ↑	73.04	74.66 ↑	67.77 ↑	72.41	61.62	48.06 ↑	5	1
+CHUNK	95.4	95.1	93.49	88.07	58.06 ↑	73.13 ↑	74.77 ↑	67.36	72.88	62.98	47.13	3	0
+NER	95.29	95.05	93.54	88.24	53.4	72.91	74.04	67.57 ↑	70.78 ↓	63.02	48.64	1	1
+MWE	95.05 ↓	94.66 ↓	93.33	88.02	53.07	72.83	74.66 ↑	66.26	71.36	60.61	46.71	1	2
+SEM	95.27	94.93	93.52	87.49 ↓	58.62 ↑	72.77	74.41 ↑	68.1 ↑	72.25	62.17	47.12	3	1
+SEMTR	95.23	94.97	93.45	87.29 ↓	58.31 ↑	72.17 ↓	74.02	67.64	72.15	62.79	46.1	1	2
+SUPSENSE	95.27	95.0	93.13 ↓	87.92	58.05 ↑	73.09 ↑	74.94 ↑	66.81	72.12	61.96	47.24	3	1
+COM	93.6 ↓	93.12 ↓	91.86 ↓	86.75 ↓	51.71	70.18 ↓	71.35 ↓	65.55 ↓	72.71	57.65	47.81	0	7
+FRAME	95.0 ↓	94.55 ↓	93.29	87.99	53.3	72.49	74.63 ↑	66.75	72.1	62.04	46.66	1	2
+HYP	94.43 ↓	94.26 ↓	93.13 ↓	87.82	52.59	71.95	74.14	66.16	72.79	61.14	46.73	0	3
#↑	0	0	2	0	6	3	7	4	0	0	1		
#↓	4	4	3	5	0	2	1	1	1	0	0		
Average	95.0	94.66	93.32	87.65	55.99	72.49	74.24	67.09	72.09	61.62	47.37		
All	94.94 ↓	94.3 ↓	93.7 ↑	86.01 ↓	59.57 ↑	71.58 ↓	74.35	68.02 ↑	74.61 ↑	61.83	49.5 ↑	5	4
Oracle	95.4	95.04	93.93 ↑	88.24	61.92 ↑	73.14 ↑	75.09 ↑	69.04 ↑	72.71	62.04	48.06 ↑	6	0

Table 5: F1 scores for **TE⊕ENC**. We compare STL setting (blue), with pairwise MTL (+⟨task⟩), All, and Oracle. We test on each task in the columns. Beneficial settings are in green. Harmful setting are in red. The last two columns indicate how many tasks are helped or harmed by the task at that row.

	UPOS	XPOS	CHUNK	NER	MWE	SEM	SEMTR	SUPSENSE	COM	FRAME	HYP	#↑	#↓
All	95.04	94.31	93.44	86.38	61.43	71.53	74.26	68.1	74.54	59.71	51.41		
All - UPOS		94.03	93.59	86.03	61.28	70.87	73.54	68.27	74.42	58.47	51.13	0	0
All - XPOS	94.57 ↓		93.57	86.04	61.91	71.12	74.03	67.99	74.36	60.16	51.65	0	1
All - CHUNK	94.84 ↓	94.46		86.05	61.01	71.07	73.97	68.26	74.2	60.01	50.27	0	1
All - NER	94.81 ↓	94.3	93.59		62.69	70.82	73.51 ↓	68.16	74.08	59.17	50.86	0	2
All - MWE	94.93 ↓	94.45	93.71	86.21		71.01	73.61 ↓	68.18	74.7	59.23	50.83	0	2
All - SEM	94.82	94.34	93.63	85.81	61.17		71.97 ↓	67.36	74.31	58.73	50.93	0	1
All - SEMTR	94.83	94.35	93.58	86.11	63.04	69.72 ↓		68.17	74.2	59.49	51.27	0	1
All - SUPSENSE	94.97	94.54	93.67	86.43	60.51	71.22	73.86 ↓		74.24	59.23	50.86	0	1
All - COM	95.19 ↑	94.69 ↑	93.67	86.6	61.95	72.38 ↑	74.75 ↑	68.67		62.37 ↑	50.28	5	0
All - FRAME	95.15	94.57	93.7	85.9	62.62	71.48	74.24	68.47	75.03		50.89	0	0
All - HYP	94.93	94.53	93.78 ↑	86.31	62.04	71.22	74.02	68.46	74.62	59.69		1	0
#↑	1	1	1	0	0	1	1	0	0	1	0		
#↓	4	0	0	0	0	1	4	0	0	0	0		

Table 6: F1 scores for **MULTI-DEC**. We compare All with All-but-one settings (All - ⟨TASK⟩). We test on each task in the columns. Beneficial settings are in green. Harmful setting are in red.

MTL performances. Furthermore, tasks can also be harmful in a complementary manner. For instance, in the case of NER, All MTL achieves the lowest or the second lowest score when compared to any row of the pairwise MTL settings. In addition, SEM's pairwise MTL performances are mixed, making the average score about the same or slightly worse than STL. However, the performance of All MTL when tested on SEM almost achieves the lowest. In other words, SEM is harmed more than it is benefited but pairwise MTL performances cannot tell. This suggests that harmful tasks are complementary while beneficial tasks are not.

Our results when *tested on* COM *are the most surprising*. While none of pairwise MTL settings help (with some hurting), the performance of All MTL goes in the opposite direction, outperforming that of STL. Further characterization of task interaction is needed to reveal why this happens. One hypothesis is that instances in COM that are benefited by one task may be harmed by another. The joint training of all tasks thus works because tasks *regularize* each other.

We believe that our results open the doors to other interesting research questions. While the pairwise MTL performance is somewhat predictive of the performance direction of All MTL (except COM), the magnitude of that direction is difficult to predict. It is clear that additional factors beyond pairwise performance contribute to the success or failure of the All MTL setting. It would be useful to automatically identify these factors or design a metric to capture that. There have been initial attempts along this research direction in (Alonso and Plank, 2017; Bingel and Søgaard, 2017; Bjerva, 2017), in which manually-defined task characteristics are found to be predictive of *pairwise* MTL's failure or success.

Oracle MTL Recall that a task has an "Oracle" set when the task is benefited from some other tasks according to its pairwise results (cf. Sect. 3.4). In general, our simple heuristic works well. Out of 20 cases where Oracle MTL performances exist, 16 are better than the performance of All MTL. In SEM, UPOS and XPOS (**TE⊕DEC**, Oracle MTL is able to reverse the negative results obtained by All MTL to the positive ones, leading to improved scores over STL in all cases. This suggests that pairwise MTL performances are valuable knowledge if we want to go beyond two tasks. But, as mentioned previously,

pairwise performance information fails in the case of COM; All MTL leads to improvement but we do not have an Oracle set in this case.

Out of 4 cases where Oracle MTL does not improve upon All MTL, 3 is when we test on HYP and one is when we test on MWE. These two tasks are not harmed by any tasks. This result seems to suggest that sometimes "neutral" tasks can help in MTL (but not always, for example, in **MULTI-DEC** and **TE⊕ENC** of MWE). This also raises the question of whether there is a more effective way to construct an oracle set.

4.4 Analysis

Task Contribution in All MTL How much does one particular task contribute to the performance of All MTL? To investigate this, we remove one task at a time and train the rest jointly. Results are shown in Table 6 for the method **MULTI-DEC**– results for other two methods are in the arXiv version as they are similar to **MULTI-DEC** qualitatively. We find that UPOS, SEM and SEMTR are in general sensitive to a task being removed from All MTL. Moreover, at least one task significantly contributes to the success of All MTL at some point; if we remove it, the performance will drop. On the other hand, COM generally negatively affects the performance of All MTL as removing it often leads to performance improvement.

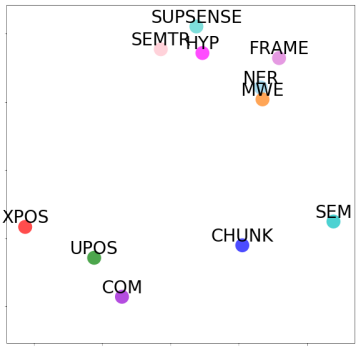


Figure 4: t-SNE visualization of the embeddings of the 11 tasks that are learned from **TE⊕DEC**

Task Embeddings Fig. 4 shows t-SNE visualization (Van der Maaten and Hinton, 2008) of task embeddings learned from **TE⊕DEC**[6] in the All MTL setting. The learned task embeddings reflect our knowledge about similarities between tasks, where there are clusters of syntactic and semantic tasks. We also learn that sentence compression (COM) is more syntactic, whereas multi-word expression identification (MWE) and hyper-text detection (HYP) are more semantic. Interestingly, CHUNK seems to be in between, which may explain why it never harms any tasks in any settings (cf. Sect. 4.2).

In general, it is not obvious how to translate task similarities derived from task embeddings into something indicative of MTL performance. While our task embeddings could be considered as "task characteristics" vectors, they are not guaranteed to be interpretable. We thus leave a thorough investigation of information captured by task embeddings to future work.

Nevertheless, we observe that task embeddings disentangle "sentences/tags" and "actual task" to some degree. For instance, if we consider the locations of each pair of tasks that use the same set of sentences for training in Fig. 4, we see that SEM and SEMTR (or MWE and SUPSENSE) are not neighbors, while XPOS and UPOS are. On the other hand, MWE and NER are neighbors, even though their label set size and entropy are not the closest. These observations suggest that hand-designed task features used in (Bingel and Søgaard, 2017) may not be the most informative characterization for predicting MTL performance.

5 Related Work

For a comprehensive overview of MTL in NLP, see Chapter 20 of (Goldberg, 2017) and (Ruder, 2017). Here we highlight those which are mostly relevant.

[6]We observed that task embeddings learned from **TE⊕ENC** are not consistent across multiple runs.

MTL for NLP has been popular since a unified architecture was proposed by (Collobert and Weston, 2008; Collobert et al., 2011). As for sequence to sequence learning (Sutskever et al., 2014), general multi-task learning frameworks are explored by (Luong et al., 2016).

Our work is different from existing work in several aspects. First, the majority of the work focuses on two tasks, often with one being the main task and the other being the auxiliary one (Søgaard and Goldberg, 2016; Bjerva et al., 2016; Plank et al., 2016; Alonso and Plank, 2017; Bingel and Søgaard, 2017). For example, POS is the auxiliary task in (Søgaard and Goldberg, 2016) while CHUNK, CCG supertagging (CCG) (Clark, 2002), NER, SEM, or MWE+SUPSENSE is the main one. They find that POS benefits CHUNK and CCG. Another line of work considers language modeling as the auxiliary objective (Godwin et al., 2016; Rei, 2017; Liu et al., 2018). Besides sequence tagging, some work considers two high-level tasks or one high-level task with another lower-level one. Examples are dependency parsing (DEP) with POS (Zhang and Weiss, 2016), with MWE (Constant and Nivre, 2016), or with semantic role labeling (SRL) (Shi et al., 2016); machine translation (TRANSLATE) with POS or DEP (Niehues and Cho, 2017; Eriguchi et al., 2017); sentence extraction and COM (Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011; Almeida and Martins, 2013).

Exceptions to this include the work of (Collobert et al., 2011), which considers four tasks: POS, CHUNK, NER, and SRL; (Raganato et al., 2017), which considers three: word sense disambiguation with POS and coarse-grained semantic tagging based on WordNet lexicographer files; (Hashimoto et al., 2017), which considers five: POS, CHUNK, DEP, semantic relatedness, and textual entailment; (Niehues and Cho, 2017; Kiperwasser and Ballesteros, 2018), which both consider three: TRANSLATE with POS and NER, and TRANSLATE with POS and DEP, respectively. We consider as many as 11 tasks jointly.

Second, we choose to focus on model architectures that are generic enough to be shared by many tasks. Our structure is similar to (Collobert et al., 2011), but we also explore frameworks related to task embeddings and propose two variants. In contrast, recent work considers stacked architectures (mostly for sequence tagging) in which tasks can supervise at different layers of a network (Søgaard and Goldberg, 2016; Klerke et al., 2016; Plank et al., 2016; Alonso and Plank, 2017; Bingel and Søgaard, 2017; Hashimoto et al., 2017). More complicated structures require more sophisticated MTL methods when the number of tasks grows and thus prevent us from concentrating on analyzing relationships among tasks. For this reason, we leave MTL for complicated models for future work.

The purpose of our study is relevant to but different from transfer learning, where the setting designates one or more target tasks and focuses on whether the target tasks can be learned more effectively from the source tasks; see e.g., (Mou et al., 2016; Yang et al., 2017).

6 Discussion and Future Work

We conduct an empirical study on MTL for sequence tagging, which so far has been mostly studied with two or a few tasks. We also propose two alternative frameworks that augment taggers with task embeddings. Our results provide insights regarding task relatedness and show benefits of the MTL approaches. Nevertheless, we believe that our work simply scratches the surface of MTL. The characterization of task relationships seems to go beyond the performances of pairwise MTL training or similarities of their task embeddings. We are also interested in exploring further other techniques to MTL, especially when tasks become more complicated. For example, it is not clear how to best represent task specification as well as how to incorporate them into NLP systems. Finally, the definition of tasks can be relaxed to include domains or languages. Combining all these will move us toward the goal of having a *single* robust, generalizable NLP agent that is equipped with a diverse set of skills.

Acknowledgments This work is partially supported by USC Graduate Fellowship, NSF IIS-1065243, 1451412, 1513966/1632803/1833137, 1208500, CCF-1139148, a Google Research Award, an Alfred P. Sloan Research Fellowship, gifts from Facebook and Netflix, and ARO# W911NF-12-1-0241 and W911NF-15-1-0484.

References

[Almeida and Martins2013] Miguel B. Almeida and André F. T. Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *ACL*. 10

[Alonso and Plank2017] Héctor Martínez Alonso and Barbara Plank. 2017. Multitask learning for semantic sequence prediction under varying data conditions. In *EACL*. 1, 3, 8, 10

[Baker et al.1998] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *COLING-ACL*. 4

[Berg-Kirkpatrick et al.2011] Taylor Berg-Kirkpatrick, Daniel Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *ACL*. 10

[Bies et al.2012] Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English web treebank. Technical Report LDC2012T13, Linguistic Data Consortium, Philadelphia, PA. 3

[Bingel and Søgaard2017] Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *EACL*. 1, 4, 8, 9, 10

[Bjerva et al.2016] Johannes Bjerva, Barbara Plank, and Johan Bos. 2016. Semantic tagging with deep residual networks. In *COLING*. 1, 10

[Bjerva2017] Johannes Bjerva. 2017. Will my auxiliary tagging task help? Estimating Auxiliary Tasks Effectivity in Multi-Task Learning. In *NoDaLiDa*. 8

[Caruana1997] Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28:41–75. 1

[Cho et al.2014] Kyunghyun Cho, Bart van Merrienboer, aglar Gülehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*. 4

[Ciaramita and Altun2006] Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *EMNLP*. 3

[Clark2002] Stephen Clark. 2002. Supertagging for combinatory categorial grammar. In *Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks*. 10

[Clarke and Lapata2006] James Clarke and Mirella Lapata. 2006. Constraint-based sentence compression: An integer programming approach. In *ACL*. 4

[Collobert and Weston2008] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*. 1, 2, 10

[Collobert et al.2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537. 1, 2, 10

[Constant and Nivre2016] Matthieu Constant and Joakim Nivre. 2016. A transition-based system for joint lexical and syntactic analysis. In *ACL*. 10

[Das et al.2014] Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40:9–56. 4

[Eriguchi et al.2017] Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *ACL*. 10

[Francis and Kučera1982] Winthrop Nelson Francis and Henry Kučera. 1982. Frequency analysis of english usage: Lexicon and grammar. *Journal of English Linguistics*, 18(1):64–70. 3

[Gardner et al.2018] Matt A. Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*. 4

[Glorot and Bengio2010] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 5

[Godwin et al.2016] Jonathan Godwin, Pontus Stenetorp, and Sebastian Riedel. 2016. Deep semi-supervised learning with linguistically motivated sequence labeling task hierarchies. *arXiv preprint arXiv:1612.09113*. 10

[Goldberg2017] Yoav Goldberg. 2017. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309. 9

[Graff1997] David Graff. 1997. The 1996 broadcast news speech and language-model corpus. In *Proceedings of the 1997 DARPA Speech Recognition Workshop*. 4

[Hashimoto et al.2017] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A Joint Many-Task Model: Growing a neural network for multiple NLP tasks. In *EMNLP*. 10

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–80. 4

[Huang et al.2015] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*. 4

[Irsoy and Cardie2014] Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *EMNLP*. 4

[Johnson et al.2017] Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *TACL*, 5:339–351. 3

[Kingma and Ba2015] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*. 5

[Kiperwasser and Ballesteros2018] Eliyahu Kiperwasser and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. *TACL*, 6:225–240. 10

[Klerke et al.2016] Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. In *HLT-NAACL*. 10

[Lafferty et al.2001] John D. Lafferty, Andrew D. McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*. 4

[Lample et al.2016] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *HLT-NAACL*. 4

[Landes et al.1998] Shari Landes, Claudia Leacock, and Randee I. Tengi. 1998. Building semantic concordances. *WordNet: An electronic lexical database*, 199(216):199–216. 3

[Lewis et al.2004] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397. 3

[Liu et al.2018] Liyuan Liu, Jingbo Shang, Frank F. Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. In *AAAI*. 10

[Luong et al.2016] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *ICLR*. 1, 2, 10

[Ma and Hovy2016] Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *ACL*. 4, 5

[Marcus et al.1993] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330. 3

[Martins and Smith2009] André F. T. Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the NAACL-HLT Workshop on Integer Linear Programming for NLP*. 10

[Miller et al.1993] George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Proceedings of the workshop on Human Language Technology*. 3

[Mou et al.2016] Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in nlp applications? In *EMNLP*. 10

[Niehues and Cho2017] Jan Niehues and Eunah Cho. 2017. Exploiting linguistic resources for neural machine translation using multi-task learning. In *WMT*. 10

[Nivre et al.2016] Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *LREC*. 3

[Pascanu et al.2013] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *ICML*. 5

[Paszke et al.2017] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *Proceedings of the NIPS Workshop on the future of gradient-based machine learning software and techniques*. 4

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*. 5

[Plank et al.2016] Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *ACL*. 1, 10

[Raganato et al.2017] Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017. Neural sequence learning models for word sense disambiguation. In *EMNLP*. 10

[Rei2017] Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *ACL*. 10

[Reimers and Gurevych2017] Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *EMNLP*. 2

[Ruder2017] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*. 9

[Schneider and Smith2015] Nathan Schneider and Noah A. Smith. 2015. A corpus and model integrating multiword expressions and supersenses. In *HLT-NAACL*. 3

[Shi et al.2016] Peng Shi, Zhiyang Teng, and Yue Zhang. 2016. Exploiting mutual benefits between syntax and semantic roles using neural network. In *EMNLP*. 10

[Søgaard and Goldberg2016] Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL*. 1, 10

[Spitkovsky et al.2010] Valentin I. Spitkovsky, Daniel Jurafsky, and Hiyan Alshawi. 2010. Profiting from mark-up: Hyper-text annotations for guided parsing. In *ACL*. 4

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*. 2, 10

[Tjong Kim Sang and Buchholz2000] Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *CoNLL*. 3

[Tjong Kim Sang and De Meulder2003] Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *CoNLL*. 3

[Van der Maaten and Hinton2008] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85. 9

[Vossen et al.1998] Piek Vossen, Laura Bloksma, Horacio Rodriguez, Salvador Climent, Nicoletta Calzolari, Adriana Roventini, Francesca Bertagna, Antonietta Alonge, and Wim Peters. 1998. The EuroWordNet Base Concepts and Top Ontology. Technical Report LE2-4003, University of Amsterdam, The Netherlands. 3

[Yang et al.2017] Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. In *ICLR*. 10

[Zhang and Weiss2016] Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *ACL*. 10

Using J-K-fold Cross Validation to Reduce Variance When Tuning NLP Models

Henry B. Moss
STOR-i Centre for Doctoral Training, Lancaster University

David S. Leslie
Department of Mathematics and Statistics, Lancaster University

Paul Rayson
School of Computing and Communications, Lancaster University

initial.surname@lancaster.ac.uk

Abstract

K-fold cross validation (CV) is a popular method for estimating the true performance of machine learning models, allowing model selection and parameter tuning. However, the very process of CV requires random partitioning of the data and so our performance estimates are in fact stochastic, with variability that can be substantial for natural language processing tasks. We demonstrate that these unstable estimates cannot be relied upon for effective parameter tuning. The resulting tuned parameters are highly sensitive to how our data is partitioned, meaning that we often select sub-optimal parameter choices and have serious reproducibility issues.

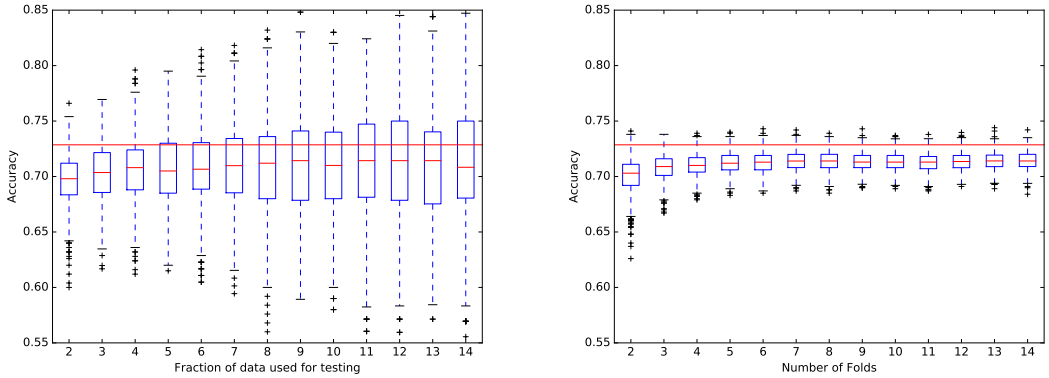
Instead, we propose to use the less variable J-K-fold CV, in which J independent K-fold cross validations are used to assess performance. Our main contributions are extending J-K-fold CV from performance estimation to parameter tuning and investigating how to choose J and K. We argue that variability is more important than bias for effective tuning and so advocate lower choices of K than are typically seen in the NLP literature, instead use the saved computation to increase J. To demonstrate the generality of our recommendations we investigate a wide range of case-studies: sentiment classification (both general and target-specific), part-of-speech tagging and document classification.

1 Motivation

In recent years, the main focus of the machine learning community has been on model performance. We cannot, however, hope to improve our model's predictive strength without being able to confidently identify small (but genuine) improvements in performance. We require an accurate measurement of how well our model will perform once in practical use, known as prediction or generalisation error; the model's ability to generalise from its training set (see Friedman et al (2001) for an in depth discussion). The accurate estimation of model performance is crucial for selecting between models and choosing optimal model parameters (tuning).

Estimating prediction error on the same data used to train a model can lead to severe under-estimation of the prediction error and is unwise. Simple alternatives use a random splitting of data into training and testing sets, or training, validation and testing sets, with the model trained using the training set, tuned on the validation set and performance on the testing set used to report the quality of the fitted model. More sophisticated approaches are based on re-sampling and make more efficient use of the data; including bootstrapping (Efron and Tibshirani, 1994) and K-fold cross validation (CV) (Kohavi, 1995). Since bootstrapping has prohibitive computational cost and is prone to underestimating prediction error, the machine learning community have coalesced around CV as the default method of estimating prediction error. For a snapshot into prediction error techniques currently used in NLP, we look at the proceedings from COLING 2016, focusing on papers that include estimation of model performance. While some submissions use the more sophisticated K-fold CV, the majority use a single data split. Between those that use K-fold CV, 19 use 10-fold, 14 use 5-fold and a further 2 use 3-fold.

Proceedings of the 27th International Conference on Computational Linguistics, pages 2978–2989
Santa Fe, New Mexico, USA, August 20-26, 2018.



(a) Prediction error estimates based on leaving $\frac{1}{K}$ of the data for testing

(b) Prediction error estimates based on K-fold CV for different choices of K

Figure 1: The substantial variability in performance estimates for IMDB sentiment classification. The horizontal line represents the true performance of the model. Each box-plot summarises 1,000 estimations of prediction error based on a different random shuffle of our $1,000$ reviews.

Each of these estimation methods involves making one or more random partitions of the data. This partitioning means the estimated prediction error is a random quantity, even when conditioned on the data available to the algorithm. This random splitting of the data leads to variation in our prediction estimates, which we define as internal variability. Although this internal variability has been discussed before (Jiang and Wang, 2017; Rodriguez et al., 2010; Bengio and Grandvalet, 2004), it is poorly understood for which datasets and models this is a problem. Note that we are referring to variation between separate estimations by K-fold CV and not variability between the K prediction error estimates that make up a single round of K-fold CV. The estimates are also biased (with their expected value not equal to the truth). Since the model is only trained on a subset it cannot achieve as high performance as if it had access to all the data. Zhang and Yang (2015) argue that evaluating performance, model selection and parameter tuning have distinct requirements in terms of the bias and variance of our estimators. In particular, as long as bias is approximately constant across different models/parameters, it will have little effect on the selected model. However, variability is critical. If our estimates have variance that swamps the real differences in model performance, we cannot tell the difference between genuinely superior parameters and noise. Reducing the internal variance of cross-validation is the main focus of this paper.

To motivate the need for the paper and demonstrate the typical size of this variability, we now introduce a simple NLP task; classifying the sentiment of IMDB movie reviews using a random forest (described in detail in Section 2). Figure 1 shows that for K-fold CV and, to an even greater extent single train-test splits, the prediction error estimates have variability substantially larger than performance differences (less than 1% accuracy improvements between models) identified using the same prediction error estimation methods at COLING 2016. Without analysing the variability of each prediction error estimate it is impossible to say whether we have a genuinely improved model or are just looking at noise. Just because one model outperforms another by a small amount on a particular data-split there is no guarantee that it is genuinely superior and, by changing the partitioning, we may in fact see the opposite. Despite the potentially large variability of the performance estimate, it is common to ignore the stochasticity and just use the results from a single partitioning. The following arguments and experiments are all with respect to the internal variability of K-fold CV. However, as single train-test splits produce less stable prediction error estimates, our arguments are still valid for single data-splits but to an even greater degree.

In the machine learning literature the usual method for reducing the internal variability of prediction error estimates is stratification (Kohavi, 1995), used five times at COLING 2016. This keeps the proportions of classes constant across partitioning and so forces each partition to be more representative of the whole data. Stratification reduces variability due to unbalanced classes. However, as demonstrated in

Figure 1, NLP tasks still suffer from large variability even when the classification classes are perfectly balanced. For natural language, representativity is a much more complicated concept than just matching class proportions. Although having been discussed in the corpus linguistics literature (Biber, 1993), representativity remains a sufficiently abstract task to not have a clear definition that can be used in NLP. We wish to split our data in a way that avoids under-representing the diversity and variability seen in the whole data, however, complicated structural properties like discourse and syntactic rules make it very difficult to measure this variability.

We present, to the best of the authors' knowledge, the first investigation into prediction error variability specifically for NLP and the first in the machine learning literature to investigate the effect of this variability on parameter tuning, questioning both reproducibility and the ability to reliably select the best parameter value. We argue that variability in our prediction error estimates is often significantly larger than the small performance gains searched for by the NLP community, so these estimates are unsuitable for model selection and parameter tuning (Section 2). We instead propose using repeated K-fold CV (Kohavi, 1995) to reduce this variability (Section 3). This is not a new idea, yet has failed to be taken up by the NLP community. Repeated CV was used only twice at COLING 2016 and both times only used for improving the stability of prediction error estimates for a chosen model: 10 repetitions of 10-fold CV by Bhatia (2016) and 2 repetitions of 5-fold CV by Collel (2016). Our main contribution is to extend this method to parameter tuning, alongside investigating guidelines for choosing the number of repetitions and K. Finally, we demonstrate that stable prediction error estimates lead to more effective parameter tuning across a wide range of typical NLP tasks (Section 4).

2 Current Practice: Tuning by K-fold CV

K-fold CV consists of averaging the prediction estimates of K train-test splits, specifically chosen such that each data point is only used in a single test set (for more information see Kohavi (1995)). The data is randomly split into K folds of roughly equal size (known as partitioning the data) before each fold is held out to evaluate a model trained on the remaining $K-1$ folds. Increasing K improves stability by averaging over more models. However, each evaluation is performed on a smaller subset of the available data and so the evaluations themselves become less stable. The combination of these competing variabilities makes up the total internal variance which, as confirmed by Figure 1, is nevertheless smaller than with single train-test splits. If we have enough data that using $1/K^{th}$ of the data still provides stable evaluations then we can see a reduction in internal variability as we increase K. However, this is not a general statement (Bengio and Grandvalet, 2004), as we will demonstrate in Section 4. It is common to choose $K = 5$ or $K = 10$, based on a study by Kohavi (1995) where empirical evidence is presented for these choices producing a reasonable trade-off between bias and variance for some specific statistical examples. We will see that the optimal choice of K is problem-specific so there is no guarantee that Kohavi's studies are comparable to modern NLP tasks. As K-fold CV requires the training of K models, there is a clear incentive to choose the smallest suitable value of K. This idea is developed in Section 3.

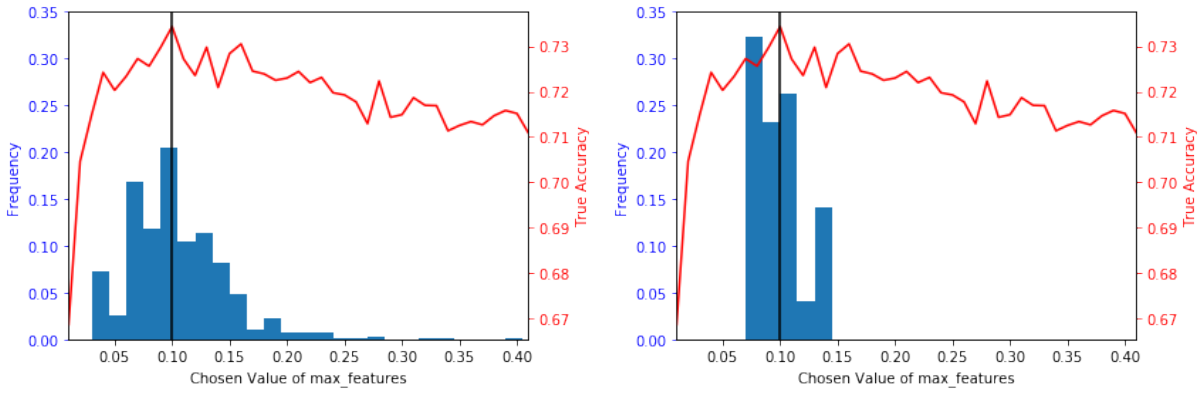
In this paper, we consider an exhaustive method used to tune parameters; grid search. Here we calculate K-fold CV performance estimation across a set of possible parameter values (our grid) and select the value that gives us the best estimated performance. Other popular tuning procedures from the machine learning literature include random search (Bergstra and Bengio, 2012) and Bayesian optimisation (Snoek et al., 2012). Grid search is often not the most efficient approach as it scales poorly with dimensionality. However, its simplicity and interpretability means that it is the standard tuner in NLP and a simple scenario for clearly demonstrating the impact of internal variability on the effectiveness of parameter tuning. Note that all tuning procedures rely on individual prediction error estimates, so our analysis of the problems of grid search is indicative of similar problems with more sophisticated tuning procedures.

To demonstrate the unsuitability of K-fold CV for tuning a typical NLP task, we train a sentiment classifier on a bag-of-word features model using a random forest (Breiman, 2001), a common set-up in the NLP literature (Gokulakrishnan et al., 2012; Da Silva et al., 2014; Fang and Zhan, 2015). We use a large corpus of $25,000$ positive and $25,000$ negative IMDB movie reviews (originally used to train word

embeddings (Maas et al., 2011)) and randomly choose a fixed $1,000$ to act as our training set (on which we need to train and tune our model). The purpose of this contrived task is simply to demonstrate the variability that can result from the random partitioning in K-fold CV. Thus "true performance", consists of the global score on the held-out reviews. We acknowledge that the exact scores depend on which $1,000$ reviews are used for training, so should not be taken as the ground truth. They do, however, allow us to roughly quantify the sub-optimality of our tuned parameters and so measure the effectiveness of different tuning procedures. Even without these scores, the variability of the tuned parameter values raises concerns regarding reproducibility (as decisions cannot be reproduced without exact knowledge of the partitioning).

Using Python's [1] random forest implementation, we consider tuning the maximum proportion of features ($max\_features$) considered at a single time by our model; a parameter whose accurate tuning is crucial to prevent over-fitting. We initially focus on $K = 5$, as Figure 1 shows a lack of significantly improved bias or variance for the increased computational cost of higher K. For each of 1,000 random 5-fold partitions of our training set, we estimate the performance for each parameter value in the set $\{0.01, 0.02, ..., 0.5\}$ and choose the value that gives us the best performance estimate. Aside from maximum depth and number of trees, which we set to 4 and 100 respectively, we keep all other parameters as their SKLEARN defaults. We limit ourselves to the 300 features with the highest tf-idf score (Salton and Buckley, 1988). Note that random forests are stochastic, with a sub-sampling step used to fit the individual tree classifiers. However, as this additional variation cannot be fixed between successive performance estimates (with the sampling being dependent on how our data is partitioned) this should be considered as part of the internal variability.

Figure 2(a) shows that partitioning can have a large effect on the chosen parameter, leading to choices of $max\_features$ across the wide range of 0.03 to 0.4 (almost covering the whole search space). This variability means that our tuning procedure is often failing to choose the parameter that gives the best global performance and so produces sub-optimal models. Depending on the initial partitioning, our tuning procedure can lead to tuned models with global performance varying from 71% to 73.5% accuracy. In Section 4, we will see that the variability is even larger when tuning multiple parameters at once.



(a) Tuning of $max\_features$ by 5-fold CV.

(b) Tuning of $max\_features$ by 10-5-fold CV.

Figure 2: Distribution of our chosen $max\_features$ across $1,000$ random data partitions. We compare the industry standard (a) with our proposed solution (b). The global performance of a model for a given value of $max\_features$ is superimposed in red, with the global optimal choice represented by the vertical line.

3 Proposed Approach: Tuning By Repeated J-K-fold Cross Validation

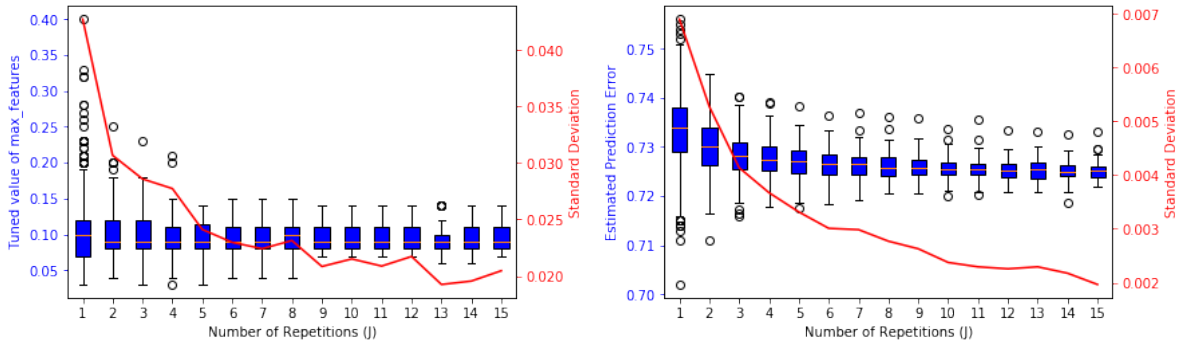
Unfortunately, only considering a single partitioning cannot give us information about the amount of variability present in our performance estimates. To produce Figure 2, we had to look at the tuned model resulting from $1,000$ different partition choices. This observation motivates the use of repeated K-fold

[1] http://scikit-learn.org/stable/index.html

CV, also known as J-K-fold CV; averaging the K-fold CV estimate from J different partition choices. It has been shown empirically that repeated CV reduces internal variability and so stabilises prediction error (Chen et al., 2012; Vanwinckelen and Blockeel, 2015; Jiang and Wang, 2017), especially for smaller datasets (Rodriguez et al., 2010). These authors, however, only investigate the use of J-K-fold CV to reduce the variability of individual performance estimates and do not consider its use for tuning.

We propose the following novel extension to grid search, which extends the improved stability of J-K-fold CV to parameter tuning. For each of J random partitionings, we calculate a K-fold CV estimate for every parameter choice on our grid. The average over these partitions produces performance estimates for each parameter choice and we choose the parameter value that maximises these stable estimates of prediction error. For the rest of the paper we will refer to this procedure as tuning by J-K-fold CV. We believe that using J-K-fold CV will also improve the effectiveness of the more sophisticated parameter tuning procedures, however, this requires further investigation.

Returning to our IMDB example, Figure 2(b) shows that using information from multiple partitioning choices greatly reduces variability in the chosen parameter value, with standard deviation dropping from 0.0427 to 0.0221 when tuning by 10-5-fold CV (Figure 2(b)) instead of vanilla 5-fold CV (Figure 2(a)). This corresponds to the worse performing tuned model over 1,000 random partition choices improving from 71.3% to 72.1% accuracy. We also see a significant reduction in the standard deviation of our performance estimate of the tuned model from 0.700% to 0.233%, greatly improving our ability to discern between closely performing models.



(a) Variability of the tuned $max\_features$ value.

(b) Variability of the performance estimate for the tuned model.

Figure 3: Diminishing reductions in variance as we increase the number of repetitions J for tuning by J-5-fold CV. As before, for each value of J we retune our model over $1,000$ different random partitioning.

	SD of chosen $max\_features$ value	Range of chosen $max\_features$ value	SD of the accuracy estimate of tuned model
1-10-fold CV	0.0396	0.03-0.38	0.550 %
2-5-fold CV	0.0307	0.04-0.25	0.528 %
1-20-fold CV	0.0337	0.03-0.24	0.483 %
2-10-fold CV	0.0301	0.03-0.23	0.400 %
4-5-fold CV	0.0278	0.03-0.21	0.367 %

Table 1: Standard deviation (SD) performance of J-K-fold tuning on the IMDB dataset over $1,000$ different partitions (3 s.f).

We now discuss how to choose the values of J and K in terms of computational cost and effective tuning, a concept we define as selecting near-optimal parameter values irrespective of how the data is partitioned. At a first approximation the cost of J-K-fold CV is proportional to $J * K$ (the number of models trained and evaluated). If, as is the case for sophisticated models, training and evaluation costs are more than linear in the amount of data, then the computational cost grows faster in K, but are still linear in J. So for fixed computational resources we have a trade-off between increasing either J or K.

First, consider increasing K. As previously mentioned there is no clear general relationship between K and the internal variability of K-fold CV. As long as $K \geq 3$ (to guarantee overlapping training sets), we can have comparable variance across K (see Section 4). This means that the only reliable reason to increase K is to reduce bias, however, there is no clear argument for how the bias of each individual performance estimate relates directly to bias in our tuning procedure (the difference between the expected value of our tuned parameter and the true optimal parameter choice). As long as the bias of the estimate for each parameter choice is roughly constant across the parameter grid, then we would expect this bias to have a limited effect on our tuned parameter value (as is assumed for tuning by vanilla K-fold CV). A further research interest is to investigate exactly when this assumption is violated and the effect that this can have on the effectiveness of parameter tuning.

In contrast, increasing J has no effect on bias but does significantly reduce the internal variability. For fixed data, the estimations from each repetition are independent and identically distributed, so prediction error estimates from vanilla K-fold CV have J times as much variability as J-K-fold CV but the same bias. Figure 3 demonstrates that we see similar variance reduction returns in the choice from tuning and also the performance estimate of the tuned model as we increase J. This means we can access the largest drops in variability within the first few repetitions (Kim, 2009).

We can therefore consider our choices of K and J in isolation. K is increased to reduce bias whereas J reduces the internal variability. We commented earlier that effective parameter tuning is more sensitive to variance than bias. This is confirmed by our IMDB task, where we see that although the tuned value from 5-fold CV has a mean close to the best performing parameter choice (calculated for our held-out $49,000$ reviews), the tuned value has significant variability. Sub-optimal tuning is a consequence of this large variance, which overshadows any potential problems from the bias. Therefore we first need to reduce internal variability before our tuning can benefit from reduced bias.

Table 1 summarises the tuning performance of five different J and K choices. We see that between 1-10-fold CV and 2-5-fold CV, and between 2-10-fold, 4-5-fold and 1-20-fold (choices of equivalent computational cost), we have the least variability when reducing K and using the saved computation for increasing J. Note that the naive approach of using all available computation to increase K (1-20-fold CV) produces a wider range of parameter choices than 2-10-fold and 4-5-fold CV. This is despite any bias reductions from choosing a higher K and so we do not recommend 1-$J * K$-fold CV. This analysis questions the rule of thumb selection of $K = 10$ common in NLP, as 2-5 CV seems to produce superior tuning at the same cost, a hypothesis we now test across a range of NLP tasks.

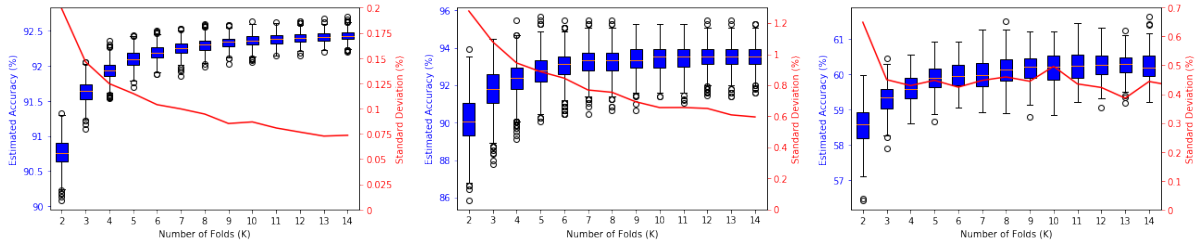
4 Case Studies

We now widen our investigation into the suitability of K-fold CV by analysing three further NLP tasks, chosen to cover a wide range of typical models and datasets. Note that we are not trying to improve the modelling of these standard models, just showing that ineffective tuning due internal variability is a general issue across NLP. We look at a simple part-of-speech tagger using logistic regression, document classification with support vector machines, and target-dependent sentiment classification with an LSTM. Each task is treated the same as our IMDB example; comparing the performances of tuning by 1-10-fold CV (the industry standard) against the computationally equivalent 2-5-fold CV. We also investigate the situation where we have two and four times the computational budget by comparing 2-10-fold with 4-5-fold and 4-10-fold with 8-5-fold. Our first task shows a case where tuning by vanilla K-fold is still reasonably effective. For task two we tune multiple parameters at once, seeing substantial variation in our tuning and so a real need for J-K-fold CV. Our final example shows that our criticisms still hold for more sophisticated models, where it can be necessary to have even more than 8 repetitions for reliable tuning. All code is available [2].

Before investigating tuning the different models, we use these examples to provide empirical evidence for our recommendation of using lower K than the common choice of ten. We argued that increasing K has no clear effect on the internal variance of individual performance estimates from K-fold CV and diminishing bias reduction returns. These are presented in Figure 4. Note that although our POS

[2] https://github.com/henrymoss/COLING2018

tagger and topic classifier do in fact become less variable for larger K, this is not a general rule, with our LSTM providing a counter-example (with comparable variance for choices of K above three). Even when we do see reductions in variability when increasing K beyond five, the following experiments show that reductions in internal variability from increasing J dominate those gained from higher K, failing to provide justification for choosing K as large as ten. Note also that the accuracy estimates for all three tasks show significant variation and so predictions by vanilla K-fold CV are unsuitable for the comparison of even relatively closely performing models.



(a) K-fold CV for logistic regression part-of-speech tagger.

(b) K-fold CV for support vector machine topic classifier

(c) K-fold CV for LSTM target-dependent sentiment classification.

Figure 4: Prediction error estimates based on K-fold CV. The variation is calculated across 1000 random partitionings for (a) and (b)) and 100 for (c).

4.1 Part-of-speech tagger

For our first task we are not directly recreating a method from a particular paper, just demonstrating that J-K-fold CV can improve the tuning of even the most common NLP models. To find such a task we consulted Jurafsky and Martin (2000), choosing logistic regression and part-of-speech tagging. We train and evaluate our model on a fixed $10,000$ word subset of the Brown Corpus (as available in Python's NLTK package (Bird, 2006)). As with our IMDB example, we use the remaining $90,000$ words as an independent global test set to check the validity of our parameter tuning. We classify with respect to the Penn Treebank tagging guidelines (Santorini, 1990) based on simple intuitive features including information about prefixes, suffixes, capital letters, position in sentence and adjacent words. We tune the amount of l2 regularisation (described by C in SKLEARN) on the grid $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100\}$.

Table 2 shows that larger choices of J with smaller K provide the most effective tuning at each computational budget. Our most unstable tuning is by 1-10-fold CV. However when checked on the held-out population, the tuned model with the worst true performance only loses 0.05% accuracy from the parameter choice with best global performance. So although increasing J does reduce variability, it is not always necessary for effective tuning, as vanilla 10-fold CV produced consistently near-optimal models. The size of variability, however, is still a concern for reproducibility, as different random partitions can lead to choosing C as 10, 50 or 100.

4.2 Document Classification

We now consider a task where tuning by vanilla K-fold CV is inadvisable; topic classification on the Reuters-21578 dataset (Lewis, 2006) via support vector machines (SVM) (Cortes and Vapnik, 1995). This exact task is well-studied in the NLP literature (Joachims, 1998; Tong and Koller, 2001; Leopold and Kindermann, 2002) however no details are provided regarding the tuning of model parameters. Note that SVM are still commonly used for document classification. This dataset and a specific train-test split (known as the ApteMod version) are available in NLTK. For ease of explanation, we focus on just the corn and wheat categories, producing a binary classification task with 334 instances. We tune two parameters; the flexibility of the decision boundary and the RBF kernel coefficient (denoted in SKLEARN as C and $gamma$ respectively). We search for C in $\{1, 5, 10, 50, 100, 500, 1000, 5000, 10000\}$ and $gamma$ in $\{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45\}$.

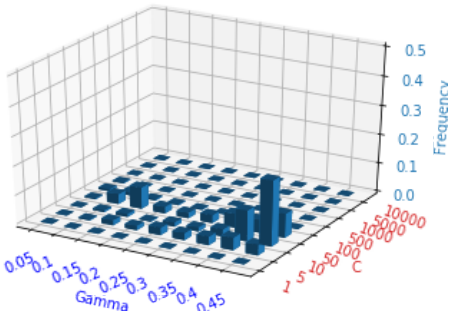
	SD of chosen C value	SD of the accuracy estimate of the tuned model
1-10-fold CV	34.6	0.0915 %
2-5-fold CV	31.0	0.0874 %
2-10-fold CV	29.0	0.0653 %
4-5-fold CV	27.3	0.0653 %
4-10-fold CV	23.2	0.0472 %
8-5-fold CV	24.4	0.0439 %

Table 2: Performance of different J-K-fold tuning procedures for a logistic regression part-of-speech tagger over $1,000$ different partitions (3 s.f).

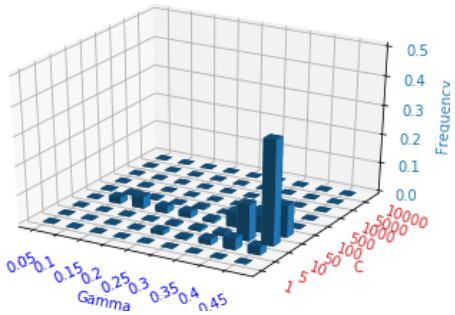
SD of chosen C value	SD of chosen $gamma$ value	SD of the accuracy estimate of the tuned model
28.7	0.121	0.807 %
30.7	0.0987	0.615 %
21.3	0.105	0.580 %
20.4	0.0766	0.445 %
20.0	0.0708	0.407 %
17.5	0.0563	0.330 %

Table 3: Performance of different J-K-fold tuning procedures for a support vector machine topic-classifier (3 s.f).

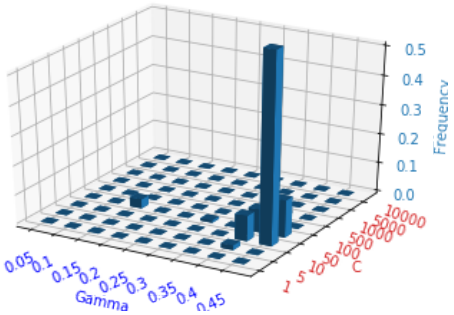
As summarised in Table 3, this task suffers from much larger variability than the part-of-speech tagger. Once again, at each computational cost, the most effective tuning (in terms of $gamma$ and variability of the performance estimate of the tuned model) is from the lower choice of K and higher J. Note that for C variability, there is an slight increase when moving from 1-10-fold to 2-5-fold CV, however, this corresponds to a large drop in the other variabilities and so 2-5-fold CV still provides more effective tuning overall. Also note that we see a larger reduction in $gamma$ variability than C variability for larger J choices. This is explained in Figure 5, where we see substantially more variation in the tuned value of $gamma$ than in C when tuning with no repetitions (Figure 5(a)). Note that effective parameter tuning corresponds to a single predominately large bin, i.e. selecting a single parameter choice irrespective of the partitioning. This is achieved by increasing J (Figures 5(b) and 5(c)). Also note that the accuracy estimate for the model tuned by 1-10-fold CV is not stable enough to allow comparison with other models of close performance. To be able to reliably distinguish between our tuned model and alternatives with performance differing by only a couple of percentage points, we require higher choices of J.



(a) Tuning SVM by vanilla 5-fold CV



(b) Tuning SVM by 2-5-fold CV



(c) Tuning SVM by 8-5-fold CV

Figure 5: Distribution of the tuned parameter values across $1,000$ random partitionings.

	SD of chosen widths	SD of the accuracy estimate of the tuned model
1-10-fold CV	19.9	0.293 %
2-5-fold CV	19.8	0.240 %
2-10-fold CV	19.3	0.213 %
4-5-fold CV	18.4	0.205 %
4-10-fold CV	17.3	0.169 %
8-5-fold CV	16.0	0.166 %

Table 4: Performance of J-K-fold tuning when choosing the width of the LSTM (3 s.f).

SD of chosen bias regularisation	SD of chosen input regularisation	SD of the accuracy estimate of the tuned model
0.0404	0.0160	0.365 %
0.0285	0.00705	0.284 %
0.0358	0.0166	0.254 %
0.0195	0.00995	0.206 %
0.0280	0.0185	0.196 %
0.00815	0.000500	0.148 %

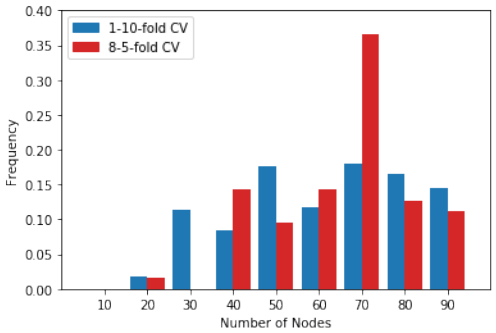
Table 5: Performance of J-K-fold tuning on the IMDB dataset over $1,000$ different partitions (3 s.f).

4.3 Target-Dependent Sentiment Classification

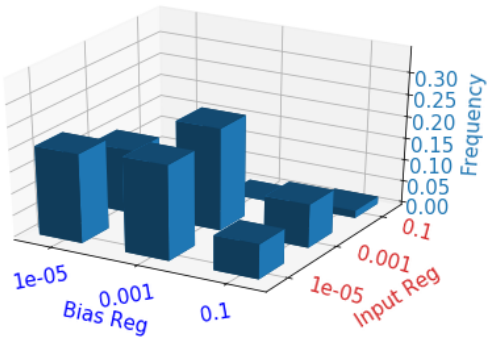
For our final task we consider a much more sophisticated model using LSTMs (with Python's KERAS [3]) to perform target-dependent sentiment classification (Tang et al., 2016) using twitter-specific sentiment word vectors (Tang et al., 2014) on the benchmark twitter dataset collected by Dong et al (2014). Our implementation relies heavily upon the reproduction study of Moore et al (2018). Each of $6,248$ sentences are annotated with a target element and the task is to predict the sentiment regarding that element (positive, negative or neutral). Unfortunately little information is provided about model architecture or parameter tuning. However, we can still investigate the effectiveness of tuning by J-K-fold CV. Throughout these experiments we fix the maximum number of epochs as 100 and stop training when we see no improvements in validation set performance for five successive epochs. For each of our $J * K$ models, the validation set is a random 20% of that model's training data, and so can be thought of just another part of the random partitioning. We use an ADAM optimiser with the default learning parameters and a batch size of 32.

We perform two experiments: tuning the number of nodes in the LSTM layer (known as width) across the grid $\{10, 20, 30, .., 90\}$, and separately tuning the amount of l2 regularisation on the inputs and biases (for a fixed width of 50) each across $\{0.00001, 0.001, 0.1\}]$. Tables 4 and 5 provide further support for our claim that the most effective tuning for a specified computational budget comes from lower K and higher J. Note that, as shown in Figure 6(a), vanilla 1-10-fold CV is not at all suitable for tuning the width of our LSTM, as it produces almost uniform values between 30 and 90. It also fails to consistently select a single choice for the regularisation scheme (Figure 6(b)). In contrast, tuning with 8-5-fold produces much more consistent choices, the majority of the time choosing 70 for the optimal width (Figure 6(a)) and 0.001 for both the input and bias regularisation (Figure 6(c)). Although variability in our chosen LSTM parameters does reduce as we increase J, it remains significantly higher than the gaps in our parameter grid. When coupled with the relative stability of the accuracy estimates of this tuned model, this suggests that the performance differences between the most frequent choices by 8-5-fold CV are small. However, to consistently tune the model to this specificity we require larger choices of J than 8.

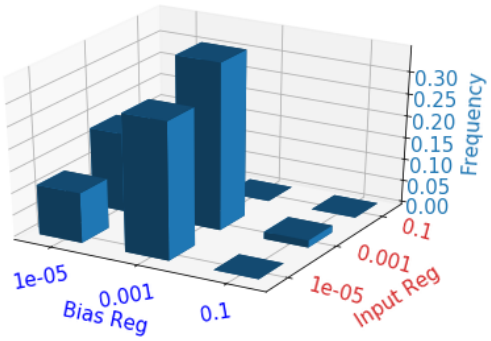
[3] `https://keras.io/`



(a) Tuning width by 1-10-fold CV and 8-5-fold CV



(b) Tuning regularisation by 10-fold CV



(c) Tuning regularisation by 8-5-fold CV

Figure 6: Distribution of tuned LSTM values over random partitionings.

5 Discussion and Conclusions

The aim of this paper has been to demonstrate the significant variability of current performance estimation techniques when applied to NLP tasks. We argue that the size of this variability is poorly understood by practitioners, which has serious implications for both model selection and parameter tuning. We show that the variability in estimates can often be larger than the performance improvements sought by the NLP community; questioning conclusions regarding the comparison of techniques. For parameter tuning, the variability can lead to unstable, irreproducible and significantly sub-optimal parameter choices.

We advocate the use of J-K-fold CV, which we extend to parameter tuning. By using information from multiple estimations we stabilise our tuning procedure. To counteract the computational cost of increasing J, we suggest lower choices of K, as effective tuning is more reliant on variability than bias.

Although we have shown the effectiveness of some specific choices of J and K on our NLP examples, there is still work to be done regarding choosing their optimal configuration, which is problem dependent. Unlike K, J can be chosen adaptively, allowing practitioners to respond to the amount of observed variability and efficiently manage computational resources. We would also like to analyse the current common practice of early stopping (as in our LSTM example), which requires evaluations on another held-out set to prevent over-fitting. This is likely to suffer from the concerns outlined in this report and it is poorly understood how best to incorporate early stopping into a wider parameter tuning framework.

6 Acknowledgements

The authors are grateful to reviewers, whose comments and advice have greatly improved this paper. The research was supported by EPSRC and the STOR-i Centre for Doctoral Training.

References

Yoshua Bengio and Yves Grandvalet. 2004. No unbiased estimator of the variance of k-fold cross-validation. *Journal of Machine Learning Research*, 5:1089–1105.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305.

Shraey Bhatia, Jey Han Lau, and Timothy Baldwin. 2016. Automatic labelling of topics with neural embeddings. In *The 26th International Conference on Computational Linguistics*, pages 953–963.

Douglas Biber. 1993. Representativeness in corpus design. *Literary and Linguistic Computing*, 8:243–257.

Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45:5–32.

Cuixian Chen, Yishi Wang, Yaw Chang, and Karl Ricanek. 2012. Sensitivity analysis with cross-validation for feature selection and manifold learning. *Advances in Neural Networks–ISNN 2012*, pages 458–467.

Guillem Collell and Marie-Francine Moens. 2016. Is an image worth more than a thousand words? On the fine-grain semantic differences between visual and linguistic representations. In *The 26th International Conference on Computational Linguistics*, pages 2807–2817.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20:273–297.

Nadia FF Da Silva, Eduardo R Hruschka, and Estevam R Hruschka. 2014. Tweet sentiment analysis with classifier ensembles. *Decision Support Systems*, 66:170–179.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 49–54.

Bradley Efron and Robert J Tibshirani. 1994. *An Introduction to the Bootstrap*. CRC press.

Xing Fang and Justin Zhan. 2015. Sentiment analysis using product review data. *Journal of Big Data*, 2:5–19.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The Elements of Statistical Learning*, volume 1. Springer Series in Statistics New York.

Balakrishnan Gokulakrishnan, Pavalanathan Priyanthan, Thiruchittampalam Ragavan, Nadarajah Prasath, and AShehan Perera. 2012. Opinion mining and sentiment analysis on a twitter data stream. In *Advances in ICT for emerging regions (ICTer), 2012 International Conference on*, pages 182–188. IEEE.

Gaoxia Jiang and Wenjian Wang. 2017. Error estimation based on variance analysis of k-fold cross-validation. *Pattern Recognition*, 69:94–106.

Thorsten Joachims. 1998. Text categorization with support vector machines: learning with many relevant features. In *European Conference on Machine Learning*, pages 137–142. Springer.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing*. Prentice Hall.

Ji-Hyun Kim. 2009. Estimating classification error rate: repeated cross-validation, repeated hold-out and bootstrap. *Computational Statistics & Data Analysis*, 53(11):3735–3745.

Ron Kohavi. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1137–1145.

Edda Leopold and Jörg Kindermann. 2002. Text categorization with support vector machines. How to represent texts in input space? *Machine Learning*, 46:423–444.

David Lewis. 2006. Reuters-21578. *http://www.daviddlewis.com/resources/testcollections/reuters21578*, Last accessed: 14th February 2018.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150.

Andrew Moore and Paul Rayson. 2018. Bringing replication and reproduction together with generalisability in nlp: Three reproduction studies for target dependent sentiment analysis. In *Proceedings of COLING 2018, the 27th International Conference on Computational Linguistics: Technical Papers*. The COLING 2018 Organizing Committee.

Juan D Rodriguez, Aritz Perez, and Jose A Lozano. 2010. Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:569–575.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24:513–523.

Beatrice Santorini. 1990. Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision). *Technical Reports (CIS)*, page 570.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1555–1565.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016. Effective LSTMs for target-dependent sentiment classification. In *The 26th International Conference on Computational Linguistics*, pages 3298–3307.

Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66.

Gitte Vanwinckelen and Hendrik Blockeel. 2015. Look before you leap: some insights into learner evaluation with cross-validation. In *Statistically Sound Data Mining*, pages 3–20.

Yongli Zhang and Yuhong Yang. 2015. Cross-validation for selecting a model selection procedure. *Journal of Econometrics*, 187:95–112.

Incremental Natural Language Processing: Challenges, Strategies, and Evaluation

Arne Köhn
Natural Lanuguage Systems Group
Department of Informatics
Universität Hamburg
koehn@informatik.uni-hamburg.de

Abstract

Incrementality is ubiquitous in human-human interaction and beneficial for human-computer interaction. It has been a topic of research in different parts of the NLP community, mostly with focus on the specific topic at hand even though incremental systems have to deal with similar challenges regardless of domain. In this survey, I consolidate and categorize the approaches, identifying similarities and differences in the computation and data, and show trade-offs that have to be considered. A focus lies on evaluating incremental systems because the standard metrics often fail to capture the incremental properties of a system and coming up with a suitable evaluation scheme is non-trivial.

Title and Abstract in German

Inkrementelle Sprachverarbeitung:
Herausforderungen, Strategien und Evaluation

Inkrementalität ist allgegenwärtig in Mensch-Mensch-Interaktiton und hilfreich für Mensch-Computer-Interaktion. In verschiedenen Teilen der NLP-Community wird an Inkrementalität geforscht, zumeist fokussiert auf eine konkrete Aufgabe, obwohl sich inkrementellen Systemen domänenübergreifend ähnliche Herausforderungen stellen. In diesem Überblick trage ich Ansätze zusammen, kategorisiere sie und stelle Ähnlichkeiten und Unterschiede in Berechnung und Daten sowie nötige Abwägungen vor. Ein Fokus liegt auf der Evaluierung inkrementeller Systeme, da Standardmetriken of nicht in der Lage sind, die inkrementellen Eigenschaften eines Systems einzufangen und passende Evaluationsschemata zu entwickeln nicht einfach ist.

1 Introduction

Interaction using language is incremental in many forms: In a dialogue, understanding takes place continuously and participants are able to interrupt each other or signal whether they understand the currently ongoing utterance. Simultaneous interpreters translate speeches as they are spoken. Texts are (partially) understood before they are fully read. Incremental processing exploits this property by starting to compute before all input is available, allowing a system to already act on partial input. Without incremental processing, an NLP system is unable to perform any of these tasks as it is bound to wait for complete utterances or a finished text and only afterwards it can compute how to (re-)act, leading to deficiencies in human-computer interaction.

First, I want to delimit this notion of incrementality from other notions: Systems that work on a complete input but generate output layer by layer, e.g. shallow to deep syntax, are sometimes called incremental, e.g. by Aït-Mokhtar et al. (2002). These systems are not discussed in this survey. Anytime algorithms are incremental in the sense that they iteratively improve their output for a fixed input, given more and more processing time. They can play an important role in incremental systems as they allow to

Proceedings of the 27th International Conference on Computational Linguistics, pages 2990–3003
Santa Fe, New Mexico, USA, August 20-26, 2018.

perform a trade-off between processing time – i.e. system responsiveness – and quality. The question of whether to employ anytime algorithms is however orthogonal to the approaches discussed in this survey[1].

In an incremental system, all processors need to work incrementally. A prime example is the Verbmobil project, which set out to develop a portable simultaneous interpreter (Kay et al., 1994). The project developed speech recognition and synthesis components, syntactic and semantic parsers, self-correction detection, dialogue modeling and of course machine translation, showing that incrementality is an aspect that touches nearly all topics of NLP. This project also exemplifies that building incremental systems is not easy, even with massive funding[2]: Only one of the many components ended up being incremental and the final report makes no mention of *simultaneous* interpretation (Wahlster, 2000).

The incremental nature is more obvious in speech than in written language because language processing often processes already-written text in bulk whereas interactive systems are a major research focus for speech-based applications. As the benefits of incrementality are more prominent in processing dialogues compared to text, research on incrementality in NLP has been primarily driven by speech-based research questions, such as understanding based on partial speech recognition (Sagae et al., 2009), determining when to respond during ongoing utterances (DeVault et al., 2009), training actor policies for rapid task-based dialogue (Paetzel et al., 2015), continuous understanding and acting (Stoness et al., 2005), incremental repair detection (Hough and Purver, 2014), or incremental reference resolution (Schlangen et al., 2009).

2 Psycholinguistic evidence

Humans still pose the gold standard for language processing, especially if the processing does not happen on large scale but in an interactive setting. When speaking, they perform several tasks incrementally and in parallel, from conceptualization to articulation (Levelt, 1989). Machines mimicking human behavior need to be able to perform similar computations as humans in such settings to be seen as a competent partner, and psycholinguistic research gives insight into the processes that humans perform. Psycholinguistic research is often carried out by timing experiments, i.e. they make use of the incremental processing by humans to gain insights into linguistic processes, mostly by eye-tracking (Tanenhaus et al., 1995; Sturt and Lombardo, 2005; von der Malsburg and Vasishth, 2011), but also by timing word-by-word reading of sentences (Gibson and Warren, 2004).

Language is perceived incrementally and this incremental processing is influenced by other modalities even while speech is perceived (Tanenhaus et al., 1995). Humans seem to create fully connected structures for sentence prefixes, even for coordinate structures, and a continuation of a sentence that does not match the predicted structure can be measured by increased reading times (Sturt and Lombardo, 2005). They perform syntactic re-analysis and use various strategies to rescan sentences (von der Malsburg and Vasishth, 2011).

3 A typology for incremental problems

When working on an incremental processor, it is helpful to know which other already existing processors had to deal with similar challenges despite being designed for completely different NLP tasks. In this section, I will examine properties relevant for classifying processors. These properties describe the input and output data as well as the relation between input and output and help classifying concrete tasks in Section 4. For an in-depth discussion of properties relevant to incremental processing, see Chapter 5 of Guhe (2007)[3].

3.1 Data types

Data can be *structured* (e.g. syntactic or semantic structures) or *sequential*. Sequential data can be *discrete* (e.g. words) or *continuous* (e.g. speech signals) along the time axis[4]. Structured data is always discrete on

[1] An anytime algorithm could be seen as a simply another non-monotonic processor, as described in Section 3.4.

[2] Verbmobil had a funding of 116 million DM, (~60 million €, or 78 million € when adjusted for inflation).

[3] The properties discussed by Guhe (2007) are mostly complementary to the ones discussed here.

[4] Data that is discrete along the time axis can of course include continuous data such as word embeddings.

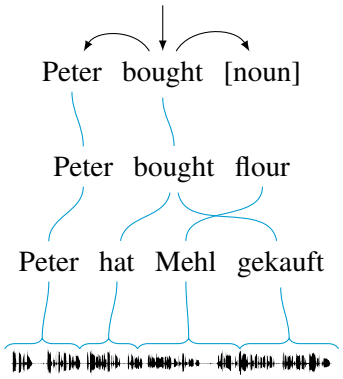
Process	Data	**Data** / *alignment* properties
		**structured, including prediction**
Parsing		*vertices partially grounded, edges not grounded*
		discrete, sequential
Machine translation		*reordering, fuzzy mapping*
		discrete, sequential
Speech recognition		*order-preserving, clear mapping*
		continuous

Figure 1: Different types of data, groundings, and processors. Note that for parsing only the prefix "Peter bought" is processed to exemplify intermediate structure generated for incomplete input. Incremental systems can take many forms, and this example is not meant to perform a specific task.

the time axis. A processor can take one type of data as an input and create another one as output; some examples are depicted in Figure 1.

Sequential data can usually be subdivided along a time axis, i.e. there exists a total ordering between the elements of the input (or output respectively). Structured data does not exhibit this property: Given, for example, the dependency tree produced by parsing in Figure 1, the words are ordered, but the dependencies between the words can't be clearly attributed to a word: They could be attributed to the head or the dependent; this poses an additional challenge for evaluation.

3.2 Granularity

The *granularity* determines the size into which the input and output is subdivided. Assuming coarse enough granularity, every system can be seen as incremental: A normal syntax parser works non-incremental inside a sentence, but incrementally if processing a paragraph with the basic units being sentences. Grapheme to phoneme conversion is incremental when processing text with the basic unit of words. When considering a larger incremental system, a processor usually seen as non-incremental might be incremental enough: If a language generation system works on the level of words, the grapheme to phoneme conversion does not need to be able to process sub-word input. On the other hand, if input typed by a user should be vocalized, sub-word granularity might be needed. The granularity of a pipeline is determined by its most coarse-grained component. In general, fine-grained processing is harder than coarse-grained processing; a system can always process data fine-grained internally while having coarse-grained interfaces, whereas the opposite is not possible.

3.3 Grounding

Grounding describes the alignment from the generated output to the elements of the input that yielded evidence for this output (Schlangen and Skantze, 2009). Grounding allows to reason about which part of the output can be reasonably generated given only partial input. In some cases, this alignment is explicit in both test and training data, e.g. in sequence labeling tasks where each element of the input is assigned a label. In other cases such as machine translation, there is no gold standard word alignment[5] and even a human-generated alignment would not create a one-to-one mapping between input and output. In addition, the alignments may or may not be *order-preserving*: A tagging task preserves the ordering, whereas in translation reordering takes place (cmp. "hat Mehl gekauft" → "bought flour" in Figure 1).

[5] At least not on the word level, but alignments can be generated automatically, see e.g. Och and Ney (2003)

3.4 Monotonicity

A system is *non-monotonic* if it is allowed to retract output it has previously produced. For example, an incremental sequence labeler that is free to re-assign labels can change its mind about every element for a sentence once the sentence is complete. A *monotonic system* on the other hand is required to only extend previously generated output without retracting information. Some components are inherently monotonic because their output is not fed to another processor of the system but to the outside world; e.g. a speech synthesizer is inherently monotonic as it cannot retract sound waves realized through a loudspeaker.

Monotonicity limits the quality a component can produce as it can not revert a decision that turns out to be wrong later on in light of additional available input. In contrast, a non-monotonic component can always achieve the same non-incremental output as a non-incremental component by simply replacing all intermediate output with the one of the non-incremental component once all input is available.

The precise meaning of monotonicity needs to be defined for each component. For sequential output, the most common definition is to only allow appending to the output. For structured output, the structure of an increment could be required to be a super-set of its predecessor.

Non-monotonic output can only be generated sensibly if the consumers of the output can deal with non-monotonic input. Otherwise these consumers might ignore the revisions made to previous output and end up with an inconsistent input or have to restart their computation in light of new input.

3.5 Timeliness

Each NLP processor has to optimize *what* to output given an input. An incremental system also needs to decide *when* to provide output. Discrete input provides specific anchors for this decision, continuous input does not and new output can be generated continuously[6]. Such decisions also need to be made by human interpreters while performing simultaneous translation; they need to buffer input until they are able to produce additional output. The characteristics of this (human) process varies by language, e.g. the delay is relatively long when translating from German to English because the verb in the input tends to be delayed (Goldman-Eisler, 1972).

3.6 Trade-Off between properties

Incremental components have to make a trade-off between timeliness (i.e. the amount of delay introduced between input and output), output quality, and the amount of non-monotonicity (Beuck et al., 2011a; Baumann et al., 2009). High-quality, monotonic output can be obtained by delaying all output. This strategy taken to the extreme results in a non-incremental system that only produces one complete output once all input is available. To reduce the delay, non-monotonicity via output revisions can be allowed, or compromises with respect to the quality of the output can be made. Gradual trade-offs can also be performed: allowing infrequent revisions and/or mild delays can lessen the negative impact on accuracy. These trade-offs are universal to all incremental processors.

4 Incremental processors

This section discusses three tasks that can be performed incrementally to exemplify strategies to "incrementalize" a processor: first, incremental speech recognition, a continuous sequence labeling problem that is usually implemented as a non-monotonic processor, then incremental machine translation, which is a sequence to sequence task with reordering implemented monotonically, and third parsing, a sequence to structure task, which is implemented both monotonic an non-monotonic.

4.1 Speech recognition

Speech recognition lends itself to incremental processing because it is used in all interactive spoken dialogue systems and the decoding happens incrementally even for non-incremental use-cases. It is therefore possible to look into a speech recognizer (SR) to obtain the most probable hypothesis at each point in time without modifying the recognizer (Baumann et al., 2009). Because the SR is not changed

[6]While continuous input is made discrete before reaching a processor, the discretization is usually in the order of milliseconds and can be seen as continuous for all practical purposes.

from the non-incremental one, the only optimization point is when to let new output through, i.e. to implement a *gatekeeper*. Its policy can be guided by observing the time that a hypothesis survived without being discarded (Baumann et al., 2009) or based on the internal state of the recognizer (Selfridge et al., 2011). McGraw and Gruenstein (2012) show that even sophisticated stability estimation only slightly improves upon the simple age-based estimation by Baumann et al. (2009). Given that the SR is the same for the incremental case as for the non-incremental one, no trade-off is performed against the accuracy. In addition, Selfridge et al. (2011) noted that if during decoding all beams in the beam search pass through the same state, the prefix up to that state can be declared stable because future decoding will not change the most probable path up to that state; this observation essentially makes use of the Markov property and is primarily useful if grammar-based recognition is performed.

4.2 Machine translation

Translation can be performed as batch processing (e.g. for websites) or incrementally, to facilitate human-human interaction. Modern approaches to machine translation, i.e. neural machine translation, employ a sequence to sequence model where the input sequence is encoded into a representation and then decoded again. This can be performed using recurrent networks which represent the input as a single fixed-length vector and possibly modeling attention to the input when generating the output sequence (Bahdanau et al., 2014) or even without a recurrent network, using only attention to model the influences from the input sequence to the output sequence to generate (Vaswani et al., 2017). In all these cases, all input is consumed before translation happens.

4.2.1 Incremental Neural Machine Translation (NMT) by using a gatekeeper

As already discussed, machine translation is a sequence to sequence problem where the output ordering does not conform to the input ordering and the ground truth for the grounding often is missing. As the incremental machine translation systems found in the literature are monotonic, the systems need to decide at which point they have enough information from the input to generate the next output token with high confidence. Gu et al. (2017) propose a system where an NMT processor repeatedly proposes an output token based on the currently available input and the generated output to a gatekeeper. The gatekeeper either accepts this output, resulting in a write operation to the output, or rejects it, resulting in a read operation on the input. The gatekeeper can work based on different policies, yielding different trade-offs between timeliness and accuracy: rejecting all output until the input is complete means falling back to a non-incremental behavior; performing alternating read and write actions eliminates delay but results in bad translations. Gu et al. (2017) train the policy using reinforcement learning with the reward based on the resulting BLEU score and the delay incurred; weighting them differently results in different trade-offs. Humans have different preferences regarding this trade-off, depending on whether the output is speech or subtitles (Mieno et al., 2015). The underlying NMT model is trained on complete sentences and therefore not adapted to incremental processing. An example of an incremental translation can be seen in Figure 2.

4.2.2 Reordering output

Reordering phenomena are a major hindrance to timely translation, as exemplified in Figure 2. Grissom II et al. (2014) deal with this by training a classifier to predict verbs needed for the output but not yet seen in the input, allowing the MT system to produce a verb without having seen its counterpart on the input. He et al. (2015) instead propose to edit the gold standard translations to better fit the source ordering for training the MT system by trying to imitate the transformations a human translator would perform. This approach tackles the problem that the training data is only available for the non-incremental case, which is not optimal for simultaneous translations. Human simultaneous translators produce sentences that systematically deviate from the "normal" target language but such data is not readily available for training (He et al., 2016). The training data is adapted by generating phrase-structure trees for the target sentences and applying (manually written) syntax-based reordering rules. The system then checks whether the reordering has reduced the delay based on an automatically computed alignment and if so uses the reordered version instead of the original one. As in the approach by Gu et al. (2017), the average delay induced by the system can be tuned, see Figure 2. Because the processor was not trained on gold-standard

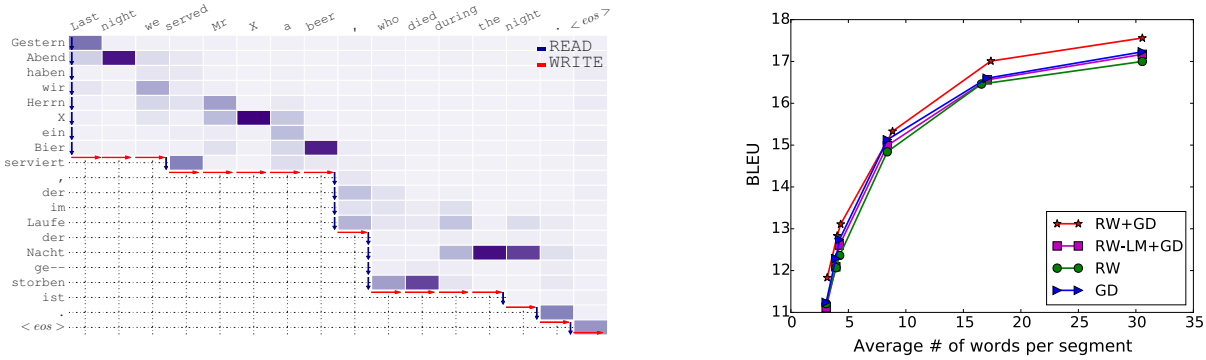


Figure 2: Left: Incremental NMT with attention, from Gu et al. (2017). Colors denote the attention given to each input token when generating an output token. Note the delay induced by the reordering for "serviert" (*served*) and "gestorben" (*died*). Right: trade-off decisions between delay (x-axis, measured in words translated at once) and accuracy (y-axis, measured in BLEU) from He et al. (2015).

data, it might yield sub-optimal results on gold-standard test data. He et al. (2015) evaluate on both gold-standard and transformed data and show that the system in fact performs better on both targets.

4.3 Parsing

Many state-of-the-art parsers work incrementally internally, both for semantic and for syntactic parsing: they use a transition system with a scorer and optionally combine that with beam search to find the best parse (Nivre, 2008; Huang and Sagae, 2010; Dyer et al., 2015; Swayamdipta et al., 2016; Kiperwasser and Goldberg, 2016; Zhou et al., 2016; Damonte et al., 2017, *inter alia*). While they build a structure incrementally going from left to right, they don't produce intermediate structures meant for incremental consumption; the intermediate states consist of several unconnected sub-structures. In addition, they usually employ look-ahead which delays the processing; approaches using Bi-LSTMs (such as Kiperwasser and Goldberg (2016)) effectively make use of the whole sentence during each step, making the computation non-incremental as it depends on the complete input being available. Noji and Miyao (2014) propose a transition system based on the ones discussed by Nivre (2008) that introduces dummy nodes which denote an expectation of upcoming words. This transition system still produces disconnected trees for sentence prefixes but is able to predict processing difficulties humans have when reading sentences with center embeddings. The other approaches described in this section all produce connected structures for sentence prefixes.

4.3.1 Incremental parsing with monotonic expansions

One approach to incremental parsing that produces connected structures at each step is to monotonically extend a connected structure and to employ a beam of possible structures to prevent being stuck with a structure that does not fit the continuation of the sentence. Using beam search, a parser does not provide monotonic output but still guarantees that one of the beam entries will be selected for a continuation.

Hassan et al. (2009) show why either a beam or delay is necessary if performing incremental parsing with monotonic extensions: They experiment with a parser based on Combinatory Categorial Grammar (Steedman, 2000). Their parser achieves an accuracy of 86% when using lookahead and performing greedy parsing (i.e. it does not use a beam). This accuracy drops significantly to 56% without lookahead because the parser often commits to a structure incompatible with the continuation of a sentence.

Roark (2001) presents a top-down phrase structure parser that performs beam-search to generate connected intermediate structures for every sentence prefix. As the parser is based on a probabilistic generative model, it can be used for language modeling and beats trigram models on the Penn Treebank (Marcus et al., 1994) (but not other sequence models, see e.g. Merity et al. (2017)). Demberg and Keller (2008) describe the PLTAG formalism, which is based on Tree Adjoining Grammar (Joshi and Schabes, 1997) and strives for psycholinguistic plausibility. It not only predicts upcoming structure needed for connectedness, but also structure required by e.g. valencies not yet filled in the prefix. The

predictions of the second type are automatically extracted from the treebank during lexical induction by learning the distinction between modifiers and arguments. The PLTAG formalism provides better predictions for reading-time experiments than predictions based on Roark's parser (Demberg et al., 2013). The combination of only extending entries in the beam and prediction leads to some sentences being unparseable for both parsers because the structures stored in the beam can all become incompatible with the input to be consumed. A larger beam reduces the number of unparseable sentences at the cost of additional memory and processing cost, but can not eliminate the problem.

4.3.2 Incremental gold standards for parsing

Köhn and Menzel (2014) go in the opposite direction and perform incremental parsing using restart-incrementality, i.e. performing a complete new parse for each sentence prefix without reusing previously generated output. This approach gives no guarantees that a certain structures will still be present in subsequent outputs but on the other hand is able to react on any upcoming input without constraints, yielding – in contrast to the approaches described in § 4.3.1 – the same accuracy for complete sentences as its non-incremental counterpart. The underlying parser performs a graph-based optimization (Martins et al., 2013) and has no notion of incrementality; instead, the gold standard the parser is trained on is adapted to consist of syntactic structures for sentence prefixes which contain prediction nodes as stand-ins for upcoming words. The partial dependency structure for a prefix is created by a rule based system that works on the complete dependency structure for the sentence. It keeps all dependencies between words in the prefix and delexicalizes words that are needed to connect the words in the prefix to the sentence root as well as arguments that are deemed to be predictable. Words outside the prefix that are neither needed for connection nor deemed to be predictable are deleted. The downside of this approach (in contrast to Demberg and Keller (2008), but similar to the reordering of output discussed in § 4.2.2) is that a rule set needs to be manually created for each language and that the linguistic intuition encoded into this rule set might be a bottleneck for the parser as structures not envisioned by the rule writer might be a better fit than the one created by the rules. Köhn and Baumann (2016) show that using the delexicalized predictions to augment 5-gram language models improves perplexity on the Billion Word Corpus (Chelba et al., 2013).

4.4 Natural language generation

Skantze and Hjalmarsson (2013) compare a non-incremental dialogue system and an incremental one, with which language learners interact to buy items at a flea market. The speech recognition component was performed manually, i.e. a human manually transcribed the speech. As the manual transcription takes time, the system response is noticeably delayed in a non-incremental system where the dialogue system only starts to plan its response once transcription is complete. In contrast, the incremental system constructs a response as soon as possible, based on partial input. The response is recomputed on changed input, which can have three effects: If the update is consistent with what has been said already, the continuation of what to say is simply changed (a *covert change*). If the system has to retract information already uttered, an explicit repair has to be produced (an *overt change*). If not all information to produce a complete utterance is available, fillers are inserted to avoid silence. The system is faster and preferred by users even though it has to explicitly correct itself, resulting in longer responses. Even though its output is monotonic – as it cannot erase information from the hearer's ears – its explicit repairs allow to act with low delay (a similar strategy to the one employed by human speakers (Levelt, 1989, ch. 12)). As the NLG component is rule-based, it is not constrained by the (non-) availability of data suitable for learning incremental language generation.

5 Evaluating incremental systems

An incremental system can be evaluated just as a non-incremental one. Evaluation schemata exist for all established tasks such as speech recognition (quality measured in word error rate), PoS tagging (measured in accuracy), phrase structure parsing (measured in precision/recall), or machine translation (BLEU). Additional evaluation allows to examine the behavior more closely, such as compiling error confusion matrices. These schemata enable a comparison between components in a standardized way.

	non-monotonic (1)	non-monotonic (2)	delayed (3)	erroneous (4)
input: a	a/y	a/y		a/y
b	a/x b/y	a/y b/y	a/x b/y	a/y b/y
c	a/x b/y c/z	a/x b/y c/z	a/x b/y c/z	a/y b/y c/z
inc_acc(i)	2: 1/1; 1: 2/2; 0: 2/3	2: 1/1; 1: 1/2 ; 0: 2/3	2: 1/1; 1: 2/2 ; 0: 2/3	2: 0/1; 1: 1/2 ; 0: 2/3
EO	1/4	1/2	0	0
accuracy	3/3	3/3	3/3	2/3

Table 1: Examples for characteristics of incremental output that need to be captured for evaluation. Correct output: a/x b/y c/z. (1), (2): output for *a* changed; (3): output for *a* held back until input "b" is available; (4): incorrect assignment to "a" stays in output. Dashed: exemplifies output used to compute inc_acc (incremental accuracy). EO: edit overhead. Accuracy: measured on complete output.

The downside of standardized evaluation is the lack of insight for incremental properties. When building incremental systems, not only the final output but also the intermediate behavior is of importance but using evaluations tailored to non-incremental systems fails to give insight into the incremental properties of the system. An incremental system should provide as much correct information as early as possible but using non-incremental evaluation hides all differences in this respect.

Table 1 shows abstract input/output patterns for a sequence labeling task, where the correspondence between input and output is given and no reordering effects take place. A standard accuracy-based evaluation on the complete output would yield a perfect score for the three first systems although their behavior over time is quite different: In addition to the non-incremental evaluation for the complete output, the non-monotonicity in (1) and (2) as well as the delay in (3) need to be covered. It is impossible to boil down all these differences to a single number. We will therefore have a look at distinct metrics for timeliness, monotonicity, and quality.

5.1 Measuring timeliness

Cho and Esipova (2016) propose to measure timeliness by counting for each output element t of an output sequence Y how many input elements from the input sequence X have been consumed before its production ($s(t)$). $\tau(X, Y)$ then computes the translation delay:

$$0 < \tau(X, Y) = \frac{1}{|X||Y|} \sum_{t=1}^{|Y|} s(t) \leq 1$$

This computation is helpful when there is no gold standard alignment between output and input which one could use to obtain the output timing that could be achieved under optimal conditions. $\tau = 0$ means all output was made without consuming input, $\tau = 1$ means all input was read before generating output.

Grissom II et al. (2014) introduce latency-BLEU, a metric that averages the BLEU scores of the outputs corresponding to each input prefix. The complete translation is weighed higher than all other partial translations to penalize incorrect translation. If the resulting translation is the same, a processor with less delay will obtain a higher score. It has to be noted that due to the averaging the sentence-initial output has more influence on the score than output created near the end of a sequence. It is also not possible to completely distinguish between the quality and the timeliness because quality and timeliness are measured in a single metric. In the MT system by He et al. (2015), the timeliness can be (indirectly) tuned by adjusting a threshold at which all yet untranslated words should be translated. Figure 2 (right) shows a plot of the resulting BLEU score against the average number of words translated at once, i.e. the delay. This way, potential users can see the trade-offs that can be made using a system (RW+GD is the proposed architecture, beating the other approaches at each trade-off point).

If an explicit alignment exists between the input and the output – such as in speech recognition – the difference between when an output is made (i.e. which amount of input data has been consumed) and the timing of the corresponding input can be measured to obtain an anchored timeliness measure (Baumann

et al., 2011). Both the relation to the first occurrence of an output (FO) and the relation to the last change of an output (final decision, FD) can be measured. E.g. if a word ends at 2.5 seconds of the input audio, was first recognized after consuming 3 seconds and was part of all outputs produced after consuming 4 seconds, its FO would be 0.5 seconds and its FD would be 1.5 seconds. To separate the timeliness from the quality, these measures can be computed against the final output of the system instead of the gold standard, but then a reliable alignment between the input and the generated output is needed. The advantage to other methods is its interpretability: A FO of 100ms for a speech recognizer means that it produces, on average, an output 100ms after it has consumed input that carries evidence for this output. Obviously, FO and FD only differ for non-monotonic processors; FD measures what delay is necessary on average to rely on an output produced by the processor.

5.2 Measuring incremental quality

When dealing with monotonic output, the incremental quality can be assessed using the non-incremental quality metrics as the processor can't correct previously made output. If a processor can be tuned to provide more or less timely output, the quality can be plotted against delay, as in Figure 2.

If a system is non-monotonic, it makes sense to measure the accuracy not only based on the final output but also on the intermediate output. Averaging the quality for each increment has two downsides: Output for early input is weighed more heavily than for later input, and it is unclear how the non-monotonicity affects the quality. Beuck et al. (2011b) perform an evaluation for incremental dependency parsing by computing the accuracy for the *n*-th word to the right of the frontier in every prefix, with n between zero (measuring the newest words) and five (the accuracy of the sixth-newest word). This way, an accuracy curve relative to the age of the input is generated, Table 1 shows incremental accuracy (*inc_acc*) measures relative to the age of the input; the examples (1) and (2) obtain different incremental accuracy measures even though the non-incremental accuracy is the same.

Incremental structured output might include predictions, which are not accounted for in the metrics discussed up to now. If an incremental gold standard is available (such as the one used for training discussed in § 4.3.2), the precision and recall of those predictions can be computed with respect to the predictions in the gold standard (Beuck et al., 2013).

5.3 Measuring the degree of non-monotonicity

Evaluating non-monotonicity can be viewed from two (similar) standpoints: first, how much intermediate output will be retracted again? And second: how sure can we be that a certain output is reliable, i.e. will also be part of the final output of a processor?

Baumann et al. (2009) and Baumann (2013) tackle the first question by defining the *edit overhead* generated by a non-monotonic processor producing sequential output by defining three edit operations on an output sequence: *add* (append an element to the output), *revoke* (remove the last element from the output), and *substitute* (revoke and then append). The difference $\mathit{diff}(o_i, o_j)$ between two outputs o_i and o_j can then be defined as the minimal number of edits needed to change o_i into o_j. Note that to change the first element of an output of length n, $2(n-1)+1$ operations are needed whereas changing the last element only yields a difference of one. This notation allows to compute the edit overhead: Let $N_{optimal} = |\mathit{diff}(o_0, o_{tmax})|$ be the number of edits necessary to obtain the final output and $N_{actual} = \sum_{t=1}^{tmax} \mathit{diff}(o_{t-1}, o_t)$ be the amount of edit operations actually performed. The edit overhead is then defined as the proportion of unnecessary edits produced by the processor:

$$EO = (N_{actual} - N_{optimal})/N_{actual}$$

Table 1 shows that EO can distinguish between the different levels of non-monotonicity. The edit operations can be adapted to the problem at hand, e.g. if structured output is produced or edits at the start of the sequence should not be penalized heavier than edits at the end.

Regarding the second question, Beuck et al. (2011b) propose to compute the accuracy against the output generated based on the complete input instead of computing against the gold standard to obtain a stability measure. Using the gold standard as reference measures consistency with the ground truth,

i.e. the quality, and using the complete output measures the consistency with that, i.e. the stability. This approach obviously only works if an incremental accuracy is defined for the problem at hand and that accuracy can be measured with respect to a non-incremental gold standard.

6 Combining multiple incremental processors

A NLP system usually consist of several processors. In a non-incremental system, all processors can simply form a pipeline with each processor working on the output of the previous one. This is non-trivial in an incremental system because for a given input, each processor may produce several outputs which may even be contradictory due to non-monotonicity. Therefore, a system either needs to use restart-incrementality throughout the pipeline or track changes to perform partial recomputation. Wirén (1992, ch. 5) introduces the notion of dependencies to track which parts of the input a certain output is based on in a chart parser. This way, only parts of the chart need to be recomputed given a non-monotonic change. Schlangen and Skantze (2009) vastly extend this notion to a general computation model in which multiple processors work on data which is organized in incremental units (IU). An IU stands for a minimal unit of information, such as a recognized word. IUs are grounded in other IUs from a previous level, and linked to other IUs on the same level, e.g. to describe a sequential relationship. Processors create new IUs based on their input and may revoke IUs already generated. A processor can commit to an IU to signify that this IU will not be revoked and other processors may rely on it.

7 Conclusion and outlook

Building incremental processors poses problems that are similar regardless of the domain. Decisions have to be made regarding the amount of delay acceptable, whether non-monotonic output is acceptable for downstream processors, and if so, to what extent. For all these decisions, the relation between input and output is important: Is there a clear mapping between input and output, maybe even a one-to-one mapping, and does reordering happen? Several techniques have been discussed for implementing incremental processors: Training a *gatekeeper* to either perform a trade-off between non-monotonicity and delay (§ 4.1) or – for monotonic processors – between delay and quality (§ 4.2.1), *Beam search* to create non-monotonic output with monotonic extension (§ 4.3.1), and *restart incrementality* for unrestricted non-monotonicity (§ 4.3.2). To evaluate an incremental system, ideally three characteristics should be measured: timeliness (§ 5.1), quality (§ 5.2), and non-monotonicity (§ 5.3). If a system can be tuned in regard to these characteristics, different trade-off points between these properties can be measured (§ 3.6).

There are still open problems for building incremental NLP systems: Processors generating connected structured output need incremental training data to learn intermediate structures not visible in the non-incremental gold standards; as the quality of the training data generation influences the quality of the processor, data-driven approaches producing high-quality incremental training data are needed. Explicit corrections in spoken output are preferred by users to delay (§ 4.4), which could transfer to incremental MT. However, lack of automatic evaluation likely requires an expensive human end-to-end evaluation due to the large deviation from non-incremental gold standard translations. All data-driven processors discussed are able to produce non-monotonic output, but are not able to consume non-monotonic input, a problematic discrepancy for building incremental systems out of multiple processors. To bridge the gap between certainty (monotonic output) and uncertainty (non-monotonic output), the likelihood of an output being stable could be attached to the output (see Selfridge et al. (2011) § 5). Modern NLP processors heavily rely on sub-symbolic representation and use attention mechanisms to obtain the relevant information. With this approach, an explicit grounding for partial recomputation as discussed in Section 6 does not work anymore and would need to be replaced with some notion of soft grounding.

Acknowledgments

I would like to thank Christine Köhn, Sebastian Beschke, and Timo Baumann for valuable feedback, as well as the anonymous reviewers for helpful remarks.

References

S. Aït-Mokhtar, J.-P. Chanod, and C. Roux. 2002. Robustness beyond shallowness: incremental deep parsing. *Natural Language Engineering*, 8(2-3):121–144.

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

Timo Baumann, Michaela Atterer, and David Schlangen. 2009. Assessing and improving the performance of speech recognition for incremental systems. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 380–388, Boulder, Colorado, June. Association for Computational Linguistics.

Timo Baumann, Okko Buß, and David Schlangen. 2011. Evaluation and optimisation of incremental processors. *Dialogue & Discourse*, 2(1):113–141. Special Issue on Incremental Processing in Dialogue.

Timo Baumann. 2013. *Incremental Spoken Dialogue Processing: Architecture and Lower-level Components*. Ph.D. thesis, Universität Bielefeld, Germany.

Niels Beuck, Arne Köhn, and Wolfgang Menzel. 2011a. Decision strategies for incremental pos tagging. In Bolette Sandford Pedersen, Gunta Nešpore, and Inguna Skadina, editors, *Proceedings of the 18th Nordic Conference of Computational Linguistics NODALIDA 2011*, volume 11 of *NEALT Proceedings*, pages 26–33. Northern European Association for Language Technology (NEALT).

Niels Beuck, Arne Köhn, and Wolfgang Menzel. 2011b. Incremental parsing and the evaluation of partial dependency analyses. In *Proceedings of the 1st International Conference on Dependency Linguistics*. Depling 2011.

Niels Beuck, Arne Köhn, and Wolfgang Menzel. 2013. Predictive incremental parsing and its evaluation. In Kim Gerdes, Eva Hajičová, and Leo Wanner, editors, *Computational Dependency Theory*, volume 258 of *Frontiers in Artificial Intelligence and Applications*, pages 186 – 206. IOS press.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005.

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation? *CoRR*, abs/1606.02012.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. An incremental parser for abstract meaning representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 536–546, Valencia, Spain, April. Association for Computational Linguistics.

Vera Demberg and Frank Keller. 2008. A psycholinguistically motivated version of tag. In *Proceedings of the Ninth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+9)*, pages 25–32, Tübingen, Germany, June.

Vera Demberg, Frank Keller, and Alexander Koller. 2013. Incremental, predictive parsing with psycholinguistically motivated tree-adjoining grammar. *Computational Linguistics*, 39(4):1025–1066.

David DeVault, Kenji Sagae, and David Traum. 2009. Can i finish? learning when to respond to incremental interpretation results in interactive dialogue. In *Proceedings of the SIGDIAL 2009 Conference*, pages 11–20, London, UK, September. Association for Computational Linguistics.

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

Edward Gibson and Tessa Warren. 2004. Reading-time evidence for intermediate linguistic structure in long-distance dependencies. *Syntax*, 7(1):55–78.

Frieda Goldman-Eisler. 1972. Segmentation of input in simultaneous translation. *Journal of Psycholinguistic Research*, 1(2):127–140, Jun.

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Dońt until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1342–1352, Doha, Qatar, October. Association for Computational Linguistics.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O.K. Li. 2017. Learning to translate in real-time with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1053–1062, Valencia, Spain, April. Association for Computational Linguistics.

Markus Guhe. 2007. *Incremental Conceptualization for Language Production*. Lawrence Erlbaum Associates, Inc.

Hany Hassan, Khalil Sima'an, and Andy Way. 2009. Lexicalized semi-incremental dependency parsing. In *Proceedings of the International Conference RANLP-2009*, pages 128–134, Borovets, Bulgaria, September. Association for Computational Linguistics.

He He, Alvin Grissom II, John Morgan, Jordan Boyd-Graber, and Hal Daumé III. 2015. Syntax-based rewriting for simultaneous machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 55–64, Lisbon, Portugal, September. Association for Computational Linguistics.

He He, Jordan Boyd-Graber, and Hal Daumé III. 2016. Interpretese vs. translationese: The uniqueness of human strategies in simultaneous interpretation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 971–976, San Diego, California, June. Association for Computational Linguistics.

Julian Hough and Matthew Purver. 2014. Strongly incremental repair detection. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 78–89, Doha, Qatar, October. Association for Computational Linguistics.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086, Uppsala, Sweden.

Aravind K. Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages: Volume 3 Beyond Words*, pages 69–123. Springer Berlin Heidelberg, Berlin, Heidelberg.

Martin Kay, Jean Mark Gawron, and Peter Norvig. 1994. *Verbmobil: a translation system for face-to-face dialog*. Number 33 in CSLI lecture notes. CSLI.

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

Arne Köhn and Timo Baumann. 2016. Predictive incremental parsing helps language modeling. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 268–277. The COLING 2016 Organizing Committee.

Arne Köhn and Wolfgang Menzel. 2014. Incremental predictive parsing with turboparser. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 803–808, Baltimore, Maryland, June. Association for Computational Linguistics.

Wilem J. M. Levelt. 1989. *Speaking: From Intention to Articulation*. The MIT Press.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 114–119, Stroudsburg, PA, USA. Association for Computational Linguistics.

André Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 617–622, Sofia, Bulgaria, August.

Ian McGraw and Alexander Gruenstein. 2012. Estimating word-stability during incremental speech recognition. In *Interspeech*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing LSTM language models. *CoRR*, abs/1708.02182.

Takashi Mieno, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Speed or accuracy? a study in evaluation of simultaneous speech translation. In *16th Annual Conference of the International Speech Communication Association (InterSpeech 2015)*, Dresden, Germany, September.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Hiroshi Noji and Yusuke Miyao. 2014. Left-corner transitions on dependency parsing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2140–2150, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Maike Paetzel, Ramesh Manuvinakurike, and David DeVault. 2015. "so, which one is it?" the effect of alternative incremental architectures in a high-performance game-playing agent. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 77–86, Prague, Czech Republic, September. Association for Computational Linguistics.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Kenji Sagae, Gwen Christian, David DeVault, and David Traum. 2009. Towards natural language understanding of partial speech recognition results in dialogue systems. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 53–56, Boulder, Colorado, June. Association for Computational Linguistics.

David Schlangen and Gabriel Skantze. 2009. A general, abstract model of incremental dialogue processing. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 710–718, Athens, Greece, March. Association for Computational Linguistics.

David Schlangen, Timo Baumann, and Michaela Atterer. 2009. Incremental Reference Resolution: The Task, Metrics for Evaluation, and a Bayesian Filtering Model that is Sensitive to Disfluencies. In *Proceedings of SigDial 2009*, London, UK.

Ethan Selfridge, Iker Arizmendi, Peter Heeman, and Jason Williams. 2011. Stability and accuracy in incremental speech recognition. In *Proceedings of the SIGDIAL 2011 Conference*, pages 110–119, Portland, Oregon, June. Association for Computational Linguistics.

Gabriel Skantze and Anna Hjalmarsson. 2013. Towards incremental speech generation in conversational systems. *Computer Speech & Language*, 27(1):243 – 262. Special issue on Paralinguistics in Naturalistic Speech and Language.

Mark Steedman. 2000. *The syntactic process*. MIT Press, Cambridge, MA, USA.

Scott C. Stoness, James Allen, Greg Aist, and Mary Swift. 2005. Using real-world reference to improve spoken language understanding. In *AAAI Workshop on Spoken Language Understanding*, pages 38–45.

Patrick Sturt and Vincent Lombardo. 2005. Processing coordinated structures: Incrementality and connectedness. *Cognitive Science*, 29:291–305.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Greedy, joint syntactic-semantic parsing with stack lstms. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, pages 187–197. Association for Computational Linguistics, June.

MK Tanenhaus, MJ Spivey-Knowlton, KM Eberhard, and JC Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268(5217):1632–1634.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6000–6010. Curran Associates, Inc.

Titus von der Malsburg and Shravan Vasishth. 2011. What is the scanpath signature of syntactic reanalysis? *Journal of Memory and Language*, 65(2):109 – 127.

Wolfgang Wahlster, editor. 2000. *Verbmobil: Foundations of Speech-to-Speech Translation*. Springer-Verlag Berlin Heidelberg.

Mats Wirén. 1992. *Studies in Incremental Natural-Language Analysis*. Ph.d. thesis, Linköping University.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang QU, Ran Li, and Yanhui Gu. 2016. Amr parsing with an incremental joint model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 680–689, Austin, Texas, November. Association for Computational Linguistics.

Gold Standard Annotations for Preposition and Verb Sense with Semantic Role Labels in Adult-Child Interactions

Lori Moon
University of Illinois at Urbana-Champaign
aralluna@illinois.edu

Christos Christodoulopoulos
Amazon Research
chrchrs@amazon.co.uk

Cynthia Fisher
University of Illinois at Urbana-Champaign
clfishe@illinois.edu

Sandra Franco
Intelligent Medical Objects
Northbrook, IL USA
sfranco@imo-online.com

Dan Roth
University of Pennsylvania
danroth@seas.upenn.edu

Abstract

This paper describes the augmentation of an existing corpus of child-directed speech. The resulting corpus is a gold-standard labeled corpus for supervised learning of semantic role labels in adult-child dialogues. Semantic role labeling (SRL) models assign semantic roles to sentence constituents, thus indicating who has done what to whom (and in what way). The current corpus is derived from the Adam files in the Brown corpus (Brown, 1973) of the CHILDES corpora, and augments the partial annotation described in Connor et al. (2010). It provides labels for both semantic arguments of verbs and semantic arguments of prepositions. The semantic role labels and senses of verbs follow Propbank guidelines (Kingsbury and Palmer, 2002; Gildea and Palmer, 2002; Palmer et al., 2005) and those for prepositions follow Srikumar and Roth (2011). The corpus was annotated by two annotators. Inter-annotator agreement is given separately for prepositions and verbs, and for adult speech and child speech. Overall, across child and adult samples, including verbs and prepositions, the κ score for sense is 72.6, for the number of semantic-role-bearing arguments, the κ score is 77.4, for identical semantic role labels on a given argument, the κ score is 91.1, for the span of semantic role labels, and the κ for agreement is 93.9. The sense and number of arguments was often open to multiple interpretations in child speech, due to the rapidly changing discourse and omission of constituents in production. Annotators used a discourse context window of ten sentences before and ten sentences after the target utterance to determine the annotation labels. The derived corpus is available for use in CHAT (MacWhinney, 2000) and XML format.

1 Introduction

The study of human language acquisition has greatly benefited from the availability of corpora of language use to, by, and around young children. The CHILDES project (MacWhinney, 2000) makes available transcribed corpora of adult-child dialogue in English and in a growing set of other languages.[1] In recent years, annotations have been added to some CHILDES corpora, including part-of-speech tagging, syntactic parsing, and the identification of grammatical roles (Pearl and Sprouse, 2013; Sagae et al., 2010; Sagae et al., 2007).

In the present paper, we describe an ongoing project that adds a new layer of annotation to selected CHILDES corpora, a hand-checked corpus of semantic role labels that provides a shallow semantic analysis of sentences' predicate-argument structure. Our goal is to support the development of computational models of language acquisition that explore how children come to interpret sentences, assigning semantic roles to sentence constituents to determine who does what to whom in each sentence. An additional goal is to provide new resources for testing the ability of trained NLP systems to generalize to new domains, in this case the challenging linguistic environment of dialogues between toddlers, who are in incomplete stages of language acquisition, and adults.

[1] Corpora and documentation are available at https://childes.talkbank.org.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3004–3014
Santa Fe, New Mexico, USA, August 20-26, 2018.

Semantic role labeling (SRL) is a common task in NLP. For each predicate, an SRL system identifies sentence constituents and assigns to them argument (e.g., agent, patient) or adjunct (e.g., locative, manner) roles. Usually SRL refers to labeling verb semantic roles, but it has been extended to nominal predicates (Meyers et al., 2004), as well as prepositions (Srikumar and Roth, 2011; Srikumar, 2013; Schneider, 2016). SRL has proven useful in areas such as question answering and textual entailment. Annotated data sets for training and evaluating the performance of SRL systems are time-consuming to construct, but new types of annotated data are important for modeling early language acquisition, and for testing the ability of SRL systems to generalize across varieties of language use.

The corpus described in this paper augments portions of an existing partial annotation of child-directed speech corpora, as described in Connor et al. (2010), with additional verb semantic role labeling and preposition semantic role labeling of the adult and child speech. This annotation project involved the Brown corpus (Brown, 1973) from the CHILDES corpora, a classic study of interactions between young children and their caregivers and other adults. The Brown corpus contains data collected in natural conversational contexts at home, and includes multiple sessions with each of three children, called 'Adam', 'Eve', and 'Sarah', learning English as their first language. The transcribed corpora are freely available on the CHILDES site, and have been involved in many analyses of the input for first language acquisition. The Brown corpus was chosen for this project in part because it has been the focus of some of the morphosyntactic annotations mentioned above. The already-released partial annotation of the Brown corpus included verb semantic role labels for some of the parental speech in the Adam, Eve, and Sarah corpora, as described below. In the present extension, we annotate all speech containing a preposition or a verb, spoken by adults or by the child, in Adam files 01-23 for verb or preposition sense and semantic role labels. This project used trained annotators, who followed existing specifications for semantic-role annotation, and where necessary, developed additional specifications for the task.

The previously released partial annotation of child-directed speech was used to train and evaluate a 'BabySRL' model that learned to interpret sentences based on simple representations of syntactic structure, derived from a constrained distributional analysis of child-directed speech, amplified by simple built-in expectations about predicate-argument structure (Connor et al., 2010; Connor, 2011; Connor et al., 2013). This system demonstrated that simple syntactic features based on the set of nouns in a sentence can guide early steps in language learning. The BabySRL learned to label the first of two nouns as an agent in simple sentences with invented verbs (e.g., *'Adam krads Mommy'*), replicating the linguistic behavior of toddlers, as shown in previous experimental work with children (Gertner et al., 2006). The model also made striking errors with two-noun intransitive sentences with invented verbs (e.g., *'Adam and Mommy krad'*), as do toddlers learning English (Gertner and Fisher, 2012). These errors diminished as the model learned to find verbs, gaining data about the importance of verb position in English sentences. These results showed that partial sentence structures grounded in sets of nouns are useful for learning in natural corpora; the model's constrained distributional learning component offers one account of where these partial sentence structures might come from during early acquisition. It also appeals to powerful evidence that infants detect distributional cues that are relevant to discovering grammatical categories (e.g., (Lany and Saffran, 2010; Shi and Melancon, 2010)).

In addition to supporting models of human language development, annotated corpora of adult-child dialogue can provide a useful context in which to evaluate the robustness of NLP learning models. Toddlers producing their early word combinations often omit the function words that support high-accuracy part-of-speech tagging, parsing and semantic-role assignment (e.g., Brown, 1973). Despite these omissions, however, they are often understood by their adult interlocutors. Understanding speech with missing elements requires a flexible knowledge of language.

The partially annotated corpus of child-directed speech that supported the development of the BabySRL model had two main limitations. First, it annotated only verbs' arguments. Second, it annotated only parental speech, leaving out the children's own contributions to the conversation. This work complements the previous corpus, and corpora such as that used in Fernald et al. (2009), which model discourse environments with child-directed speech. Our corpus does not have visual information, but it is compatible with semantic role labeling methods that are standardly used with other corpora.

The corpus labels both adult and child speech in the dialogue, adding to the noisiness of the data and providing a realistic model for the speech of young children. Naturally occurring noise in data is an interesting theoretical problem. Testing existing automatic semantic role labelers on these data provides an engineering challenge for improving tools in a new domain.

The corpus is of dialogue, so semantic arguments of a verb can appear across interlocutors. Though preposition semantic role labels are only given at the sentence level and not at the discourse level (Srikumar 2013:7), they can be used in a way that facilitates identifying semantic arguments across sentences. For example, if one person says, *'Where did the toy fall?'*, and the next line of dialogue is *'on the floor'*, this corpus is annotated in such a way that could support recovery of semantic roles across sentences in a dialogue. That is, the first sentence is annotated with *'fall'* as the main verb. The word *'where'* is the location argument of *'fall'*. The next utterance *'on the floor'* is annotated with semantic role labels for the head preposition *'on'* with the sense of *location*.

These gold labels also provide a new domain for training an automated preposition semantic role labeler. The usefulness of labeling prepositions was demonstrated in the earlier example of finding answers to questions in a dialogue. Preposition roles also add structure to verb semantic role labelers because the prepositions are often contained in an argument of the main predicate. With this corpus, if a verb has a semantic role assigned to a prepositional phrase, it follows that the preposition takes the same verb as a governor. This information can be used to tie prepositions to governing verbs occurring in the discourse.

In section 2, we explain semantic roles in more detail, discussing Propbank and the automatic preposition semantic roles of Srikumar and Roth (2011). Section 3 describes the annotation tool that we used along with modifications added to the tool for our project. In section 4 we discuss special problems presented by the CHILDES data set (Brown, 1973), explaining why it serves to meet an existing need in the community and foster more scientific discovery. Section 5 provides a break-down of our IAA measures and what they mean in terms of accuracy of the labels. Section 5.1 describes the ratings on the held-out data and what users can expect in the final version of the corpus. We note that this data set contains inherent noise and show that annotator scores reflect noise in the areas that have below 90κ scores, and achieve scores above 90κ in other areas. Section 6 discusses the availability and licensing of the corpus, and section 7 concludes the paper.

2 Semantic Role Labels

2.1 Verb Semantic Role Labeling with Propbank

Propbank (Gildea and Palmer, 2002; Kingsbury and Palmer, 2002; Palmer et al., 2005) provides resources for labeling semantic roles for verbs. The original Propbank corpus included a large hand-annotated corpus of semantic verb-argument relations, and extensive guidelines for annotating verb semantic roles in new corpora (Bonial et al., forthcoming). Propbank added semantic role labels to sentences parsed according to Penn Treebank Guidelines (Mitchell et al., 1993). Other corpora have been annotated with Propbank verb senses and semantic roles, including discourses and SMS.[2]

For each verb, Propbank lists a set of senses for the verb and the licit semantic arguments for that sense. The list of semantic arguments includes core arguments like the agent and patient of transitive verbs as well as directional and locational phrases that commonly occur with the verb. Propbank annotations involve annotating the span of each of a verb's arguments. For example, for the verb *'put'*, sense number 01, has the licit semantic roles of *Arg0, Arg1,* and *Arg2*, which are *putter*, *thing put*, and *where put*, respectively. It is not the case that all of these are present in every use of the verb *put* in a corpus.[3]

2.2 Preposition Semantic Role Labeling

Labeling preposition semantic roles helps with NLP tasks by providing additional shallow semantic information about prepositions and their semantic relation to other words in a sentence.

[2]For current information on corpora with Propbank annotations, see `https://propbank.github.io`.

[3]Senses are from the index of Propbank and FrameNet (Baker et al., 1998) for English at `http://verbs.colorado.edu/propbank/framesets-english-aliases/`.

The preposition semantic role labeler is the one described in Srikumar and Roth (2011) and Srikumar (2013). This role labeler was used due to the relatively small number of observed preposition semantic roles, and its integration with the CogComp NLP pipeline used in the current project.[4] Each preposition has the potential semantic roles associated with it of GOVERNOR and GOVERNED. The governed argument in the phrase *'to the store'* is *'the store'*. It is generally a noun phrase that follows the preposition.

The governor of the preposition can be a verb that takes a preposition as an argument. For example, in the sentence *'Take the cart to the store'*, the preposition *'to'* has the verb *'take'* as a governor. The governor can also be a noun phrase. For example, in the sentence *'Give me the horse with the blue mane.'* the governor of *'with'* is *'the horse'*.

3 Jubilee Annotation Tool

The annotation tool was based on the Jubilee tool by (Choi et al., 2010)[5] and the modified version is available at `https://gitlab-beta.engr.illinois.edu/babysrl-group/jubilee`. The original annotation tool used the Penn Treebank annotations (Mitchell et al., 1993) and Propbank's framesets,[6] and after an initial automatic SRL annotation phase, it allowed the annotators to modify the predicate sense and assign the associated semantic roles to constituents of the sentence.

We extended the tool in several ways to accommodate the annotation of children's utterances and prepositional SRL, and provided other improvements for the convenience of the annotators. A summary of the changes is as follows: We used a JSON format for the annotation files and stored the syntactic trees internally, instead of relying on a separate treebank; we added the ability to edit syntax trees with bracket highlighting; we added the ability to delete entire annotations if they had no real predicate, and we allowed the creation of new predicates when an entry was missing; we extended the context window to show more dialogue context; we added an inter-annotator agreement (IAA) calculator; we added the ability to view predicate information via a link to the Propbank website;[7] we added a bookmarking ability to allow annotators to save difficult annotation cases for future discussion.

3.1 Data Sources

The BabySRL Corpus (Connor, 2011; Connor et al., 2013) annotated a portion of the Adam, Eve, and Sarah files from the Brown corpus (Brown, 1973) with verb semantic role labels.[8] The project only annotated adult speech and omitted uses of the copula verb *'to be'*. We imported data from this corpus into the tool. The imported data did not include verb senses, but it had verb semantic role labels.

Utterances that were not part of the previous derived corpus were automatically parsed and labeled for verb and preposition semantic roles and verb and preposition sense using the NLP pipeline tools available through the Cognitive Computation Group.[9] Files were preprocessed from CHAT format to JSON format in order to annotate the data in the Jubilee tool.

After the first five Adam files were annotated, there was additional preprocessing on the xml files. Most of this preprocessing was for the purpose of working with transcriptions. There are several conventions used in the transcription that were changed. Adam's use of interdental fricatives, represented orthographically as *'th'* (phonetically [ð] and [θ] in many adult dialects of North American English), were transcribed orthographically in the CHILDES data with *'d'*, resulting in *'dat'* for *'that'*, for example. Because the entire discourse was not phonetically transcribed, we found the use of orthography

[4] An anonymous reviewer mentioned that we should explain why we did not use the more recent senses of Schneider (2016). Although there is more coverage with the preposition SRL of Schneider (2016), it uses fine-grained relations that are rarely observed. Due to the high ambiguity of preposition use in toddler speech, even the small number of distinctions in Srikumar and Roth (2011) required large amounts of discussion and calibration between annotators.

[5] The Jubilee tool is described at `https://code.google.com/archive/p/propbank/`

[6] Framesets contain a verb's sense, the associated semantic roles, and examples of uses of the intended sense in corpora. For examples, see `https://verbs.colorado.edu/propbank/framesets-english/`

[7] `http://verbs.colorado.edu/propbank/framesets-english-aliases/`

[8] The derived corpus is available through TalkBank derived corpora `https://childes.talkbank.org/derived/`

[9] The relevant tools are available at `https://cogcomp.org/page/software/` under 'NLP Tools'. The specific set of tools and instructions for replicating the results are available at `https://gitlab-beta.engr.illinois.edu/babysrl-group/babysrl-corpus`.

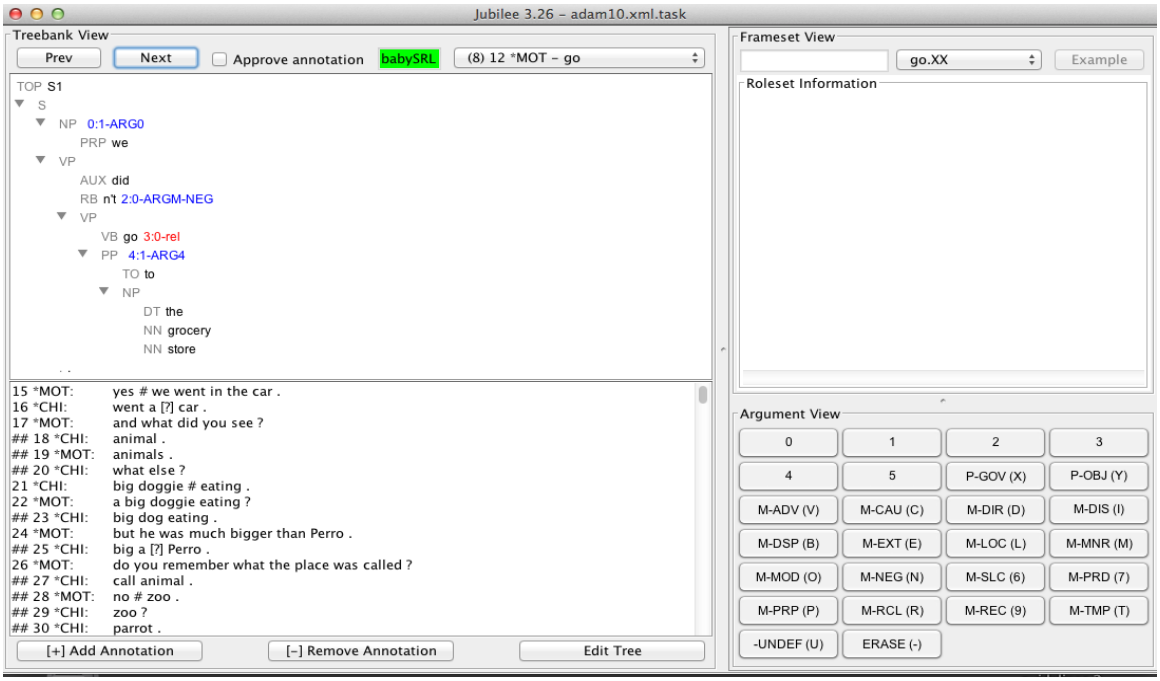


Figure 1: Example from the previous derived BabySRL corpus. 'babySRL' is highlighted in green, indicating that the labels come from the previous corpus. The context window at the bottom shows the surrounding discourse.

to represent the pronunciation inconsistent with the way in which the majority of the corpus was transcribed, so we preprocessed these examples to have standard orthography. Unknown words, transcribed as *'xxx'* were removed, as were symbols for spelling out loud, such as *'@c'*, which indicated someone speaking the name of the letter 'c'. All indicated pauses, which used the symbol *'#'* were changed to commas. Characters following underscores, along with the underscores, were removed.[10]

For utterances that were annotated in the previous round, annotators saw the gold-standard annotation without the verb sense listed. An example is given in Figure 1. Near the center-top of the bar in the Jubilee tool, there is a window that says 'babySRL', highlighted in green. Annotators could see from this window whether they were annotating utterances from the previous corpus, the automatic annotation, or viewing utterances that they had already annotated. Below the window containing the parsed and labeled utterance, there is a context window that shows some of the surrounding discourse.

To the right of the main windows, there is a window in which role set information for a sense appears. In the previous BabySRL example, no sense was imported. Annotators choose the appropriate sense. They could use semantic role labels to help determine the intended sense. The 'View in Browser' button takes the annotator directly to the Propbank online index, where annotators can check senses.

For utterances new to this round of annotation, annotators saw the automated parse and semantic role labeling. Figure 2 shows the output of the NLP pipeline, as the annotator would see the data. Rather than a green window labeled 'babySRL' there is a red window labeled 'auto', indicating that the annotator is viewing an automatic parse and SRL. In this example, the tool assigns the most likely sense, and the roles associated with the sense are shown in the Frameset View window.

4 Annotation Guidelines

The syntactic parses follow the Penn Treebank Guidelines (Santorini, 1990; Mitchell et al., 1993) as outlined in Bies et al. (1995) and the modified guidelines in Warner et al. (2012). Annotators were

[10]These changes are in BrownXMLReader.java at `https://gitlab.engr.illinois.edu/babysrl-group/babysrl-corpus/blob/master/src/main/java/edu/illinois/cs/cogcomp/babySRL/corpus/xml/BrownXMLReader.java`

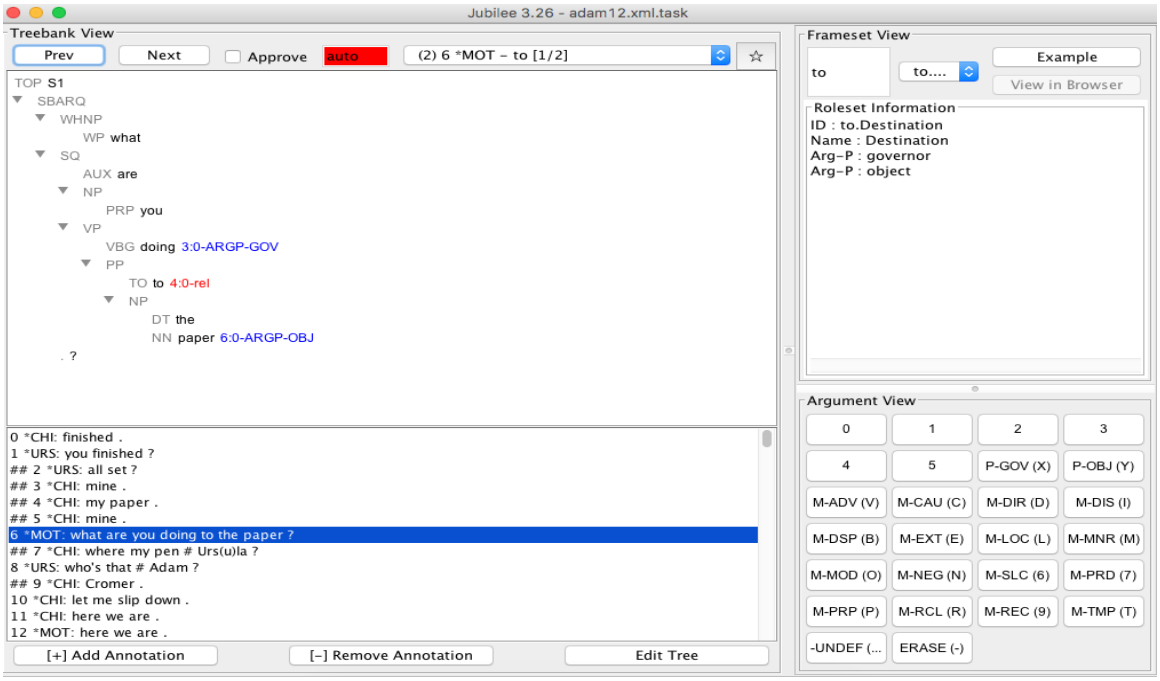


Figure 2: Example from the output of the NLP pipeline parse and semantic role labeler. The label 'auto', is in red, indicating that the labels are the output of the NLP pipeline.

instructed to consult these guidelines for all decisions. Additional decisions that came up are listed in the specifications.[11] Annotators were instructed to make a single-pass of the data. If the output parse interfered with labeling semantic roles, annotators were instructed to change the parse according to the Penn Treebank Guidelines.

In order to check verb senses and semantic role labels, an abbreviated version of Propbank (Kingsbury and Palmer, 2002; Gildea and Palmer, 2002; Palmer et al., 2005) labels appeared in the Jubilee window, and annotators were able to use a link to the Propbank entry in the Unified Verb Index (Bonial et al., 2014; Bonial et al., forthcoming), if additional information was needed.

If a verb sense did not appear to be present in Propbank, annotators consulted a list of previous decisions and, if it was present there, used that decision on semantic role labels. If the verb sense was neither in Propbank nor the list of previous decisions, annotators made note of the sense. However, annotators were instructed to try to use the previous senses as much as possible.[12] The preposition semantic roles, based on those in Srikumar and Roth (2011), had brief example descriptions in the Frameset View window.

4.1 Special Issues in Annotation

The most common issues that came up in annotation involved labeling partial expressions and ambiguous arguments. Because the child speech was labeled, it was often unclear whether expressions were filling semantic roles of a predicate or not and, if they were, it was not clear which role they were filling.

Annotators were instructed to only give a semantic role if it was clear what it was. They were instructed to use only the visible context window at the bottom of the Jubilee tool, which showed about twelve sentences with the target sentence in the middle of the context window, highlighted in blue (see Figure 2) to help determine verb or preposition senses that were uncertain.

[11]The full specifications are available at `https://github.com/CogComp/child-discourse-SRL/tree/master/specs`.

[12]Additional decisions that came up are listed in the specifications available at `https://github.com/CogComp/child-discourse-SRL/tree/master/specs`.

5 Inter-Annotator Agreement

The following four files were held-out data: Adam11, Adam15, Adam17, and Adam19. The data in these files were not discussed. Annotators were instructed to annotate according to the same methods that they used in other non-held-out data. There were 27,380 total sentences for annotation, and the held-out data total 5,804, so nearly one-fifth of the data was held-out for inter-annotator agreement. The files were annotated sequentially according to their numbers. After the IAA scores were measured on Adam 11, the results were discussed and used for calibration on subsequent held-out corpora.[13] The data was annotated by two annotators. Annotator X was an undergraduate student in linguistics and a bi-lingual speaker of English and Spanish. Annotator Y was a graduate student in linguistics and a native speaker of English.

File	total annotator X	total annotator Y	original sentences
Adam 11	1159	1133	1403
Adam 15	909	884	1187
Adam 17	1402	1379	1453
Adam 19	1704	1695	1761

Table 1: Table of total annotations of held-out data for each annotator (annotator *'X'* and annotator *'Y'*). The total annotations for each annotators are the annotations in the final annotated file. The original sentences total is the number of sentences that appeared in the Jubilee window to be annotated. Some were removed due to errors in automatic detection of prepositions and verbs or because they were instances of the child repeating one verb many times.

Table 1 shows the total annotations for each annotator and the starting number of annotations in the task. In the course of annotation, the number of annotations in the automatically annotated corpus and previous BabySRL corpus changes. Annotations are added when there is a preposition or verb in the sentence that the automated annotator missed entirely. Annotations are removed for reasons given in the specifications. For example, if a child utterance repeats the verb *'jump'* seven times, the automatic annotation will give seven entries. We decided to only use one verb in a set of repetitions. Annotations are also removed due to the fact that automatic tagging of modal auxiliaries and auxiliaries *'do'* and *'be'* occurred, but the specifications said not to annotate them.

In the subsequent sub-sections, the IAA is broken down into component parts.

Table 2 provides the inter-annotator agreement measures (raw score and Cohen's *kappa*). Each of the four held-out files are presented, followed by an average of the scores across all of the held-out data. For each file, the data is separated by verb SRL and preposition SRL, followed by the agreement on both, labeled *'predicate'*. For each of the three categories, the scores are further separated into the scores on child utterances and the scores on adult utterances, with the scores on child and adult utterances combined provided as well.

Column labels of Table 2 are the various annotations that were measured. The label *'sense'* refers to the Propbank sense, in the case of verbs, and the Srikumar and Roth (2011) sense in the case of prepositions. The label *'# args'* represents the measure of annotator agreement on how many semantic role arguments were present in the sentence. The *'label id'* measures how often annotators provided the same label name to an argument, and the *'span'* column checks annotator agreement on the span of the expression to which a label was assigned.

The syntactic structures are not explicitly compared, however, if the trees are significantly different between annotators, then it is reflected in the span of the arguments, in some cases.

[13] Non-held-out data was done up to Adam 7 at this point, and annotators were calibrating non-held-out data during the process. Adam 1-5 were later re-annotated due to a post-processing error. The work-flow time-line can be found at `https://docs.google.com/spreadsheets/d/19wJtQjt6D4CtoQH4drrpdsw4jiSkzj24U9L4W7Ob4vc/edit?usp=sharing`

Adam 11		sense	sense κ	# args	# args κ	label id	label id κ	span	span κ
Verb	Adult	*	*	91.4	86.7	93.5	91.8	96.8	96.7
	Child	90.4	59.2	86.2	76	88.1	83.1	92.4	92.3
	Total	90.4	59.2	88.8	81.35	90.8	87.45	94.6	94.5
Preposition	Adult	72.8	69.1	94.3	75.3	89.3	84.7	88.3	88.1
	Child	78.8	74.7	84.4	69.7	91.1	87.1	90.2	89.9
	Total	75.8	71.9	89.35	72.5	90.2	85.9	89.25	89
Predicates	Total	92.5	71.8	89	76.9	90.5	86.7	91.9	91.8
Adam 15		sense	sense κ	# args	# args κ	label id	label id κ	span	span κ
Verb	Adult	*	*	92.7	88.5	95	93.6	96.5	96.4
	Child	92.2	73.9	93.6	89.5	95.6	93.8	95.7	95.6
	Total	92.2	73.9	93.2	89	95.3	93.7	96.1	96
Preposition	Adult	73.4	70	96.8	80.6	96.3	94.6	96.3	96.2
	Child	72.4	67	86.2	62.4	92.6	89.3	90.8	90.5
	Total	72.9	68.5	91.5	71.5	94.5	92	93.6	93.4
Predicates	Total	82.6	71.2	92.3	80.3	94.9	92.8	94.8	94.7
Adam 17		sense	sense κ	# args	# args κ	label id	label id κ	span	span κ
Verb	Adult	92.6	71.8	91.4	86.2	93.7	92.1	95.9	95.8
	Child	92.9	63.9	88.4	81.5	93.2	90.5	93.5	93.4
	Total	92.8	67.9	90	83.9	93.5	91.3	94.7	94.6
Preposition	Adult	80.9	78.4	92.9	71	94.4	91.8	93.9	93.7
	Child	85.4	82.6	82.2	54.5	92.8	89.6	90.8	90.6
	Total	83.2	80.5	87.6	62.8	93.6	90.7	92.4	92.2
Predicates	Total	88	74.2	88.7	73.3	93.5	91	93.5	93.4
Adam 19		sense	sense κ	# args	# args κ	label id	label id κ	span	span κ
Verb	Adult	92.6	77.6	90.2	85.2	94.8	93.5	95.7	95.6
	Child	94.2	78.3	92.8	87.4	95.2	93.4	96.5	96.5
	Total	93.4	78	91.5	86.3	95	93.5	96.1	96.1
Preposition	Adult	77.6	74.2	95.6	75.1	96.1	94.3	95.3	95.2
	Child	73.6	70.6	92.4	69.4	95.8	93.8	95.5	95.4
	Total	75.6	72.4	94	72.3	96	94.1	95.4	95.3
Predicates	Total	84.5	75.2	92.8	79.3	95.5	93.8	95.8	95.7
All Files		sense	sense κ	# args	# args κ	label id	label id κ	span	span κ
Verb	Adult	92.6	74.7	91.4	86.7	94.3	92.8	94.5	94.5
	Child	92.4	68.8	90.3	83.6	93	90.2	96.2	96.1
	Total	92.5	71.8	90.8	85.1	93.6	91.5	95.4	95.3
Preposition	Adult	76.2	72.9	94.9	75.5	94	91.4	93.5	93.3
	Child	77.6	73.7	86.3	64	93.1	90	91.9	91.6
	Total	76.9	73.3	90.6	69.8	93.6	90.7	92.6	92.5
Predicates	Total	84.7	72.6	90.7	77.4	93.6	91.1	94	93.9

Table 2: For each of the held-out data sets, the IAA is presented for the verb SRL and the preposition SRL separately and for all SRL predicates (verbs and preposition SRL combined). The IAA is presented for adult language data, for child language data, and for both combined. The chart gives the raw IAA score followed by Cohen's κ for each of the following categories: *sense* gives the agreement on the Propbank sense, in the case of verbs, and the Srikumar and Roth (2011) sense, in the case of prepositions. *# arg* gives the IAA on the number of arguments annotators assigned to each predicate. The *label id* measures the agreement on the name of the label given, provided both annotators gave an expression a semantic role label. The *span* gives agreement on the span of a semantic role argument. The asterisks indicate sense data that we did not count due to a misunderstanding in the specifications, which was corrected.

5.1 Analysis

The IAA reports show a high overall agreement on the held-out dataset.[14]

The verb sense agreement κ scores for adults are much higher in the later half of the annotations. The scores for Adam 11 and 15 (not included) were 24.4 and 25.5, going up in Adam 17 and 19 to 71.8 and 77.6. The initial low scores were due to the two annotators following different specifications. The initial specifications said not to add a Propbank sense to expressions imported from the previous corpus. One annotator was adding senses anyway, and this fact was not discovered until the annotators discussed the IAA results of Adam 11. At that time, the annotator who was not adding senses had already completed annotating Adam 15. At meetings, it was decided that the specifications should be changed to say that we would add senses to verbs imported from the previous corpus. We corrected this in annotations that were finished.

The sense agreement scores for child speech and for prepositions are entirely new to this corpus, as are some of the verbs in adult speech, including the copula verb. Overall, the scores reflect high annotator agreement, given the new data. The overall sense agreement κ score is 72.6.

The argument number is simply a calculation of how many arguments the annotators assigned roles to, regardless of whether or not they assigned those arguments the same label or the same span. The effects of the κ score are more drastic on prepositions because the prepositions can only have at most two arguments, whereas verbs, in contrast, can have five. The κ scores for prepositions also tend to be the lowest due to the frequent absence of arguments and ambiguity of expressions. For example, in the child language phrase *'dog on the floor'* the preposition *'on'* governs *'the floor'*, but *'dog'* may or may not be a governor, depending on how the annotator interprets the child's phrase, given the context window. The noun *'dog'* can be considered to be a governor, similar to the sentence *'I meant the dog on the floor'*, or it can be read as being like the sentence *'The dog is on the floor'*, in which case, the governor is *'is'*, and that verb is missing in the child utterance. Because of such problems, a lower agreement occurs on prepositions overall 69.8κ, as opposed to verbs at 85.1κ.

The label identification matching has very high agreement scores, staying over 90κ throughout most of the annotation. The measure checks how often annotators assigned the same label, given that they assigned a label to an argument. Because this project was concerned with child language acquisition and identification of agents and patients in child speech, this was a welcome result. The agent and patient roles, defined in Propbank guidelines, based on a prototype view of semantic roles (Dowty, 1991), are the most common semantic roles associated with verbs in Propbank. Seeing that annotators consistently agreed on how to assign them in a corpus with high ambiguity attests to the specification development in this area and the feasibility of labeling semantic roles in child speech.

The argument span labels also have scores over 90κ in most of the data. Annotators were instructed to alter the parse according to Penn Treebank guidelines only when the parse was incorrect in a way that altered the span of a semantic role argument. Indirectly, the high agreement on argument span indicates a high agreement on decisions regarding changes to the automatic parser output.

Among the areas that had lower agreement, the agreement scores reflect reasonable variation in the interpretation of data that is frequently ambiguous. For example, when the child says *'go mommy'*, it can be read as an imperative telling his mother to leave or as a statement of the child's intention to go to his mother (e.g., 'I am going to mommy.').

In comparison to previous work, according to `http://cogcomp.org/Data/BabySRL.html`, 15 of the 133 files were held-out for measuring IAA. Across all of the files, annotators agreed on an average of 96.57% of the annotated arguments for span and label. Our span and label average is 93.8%. Considering the increased difficulty of SRLs in child speech, the results are comparable.

[14]The code used to calculate IAA is available at `https://gitlab-beta.engr.illinois.edu/babysrl-group/jubilee/blob/master/src/jubilee/agreement/AgreementCalculator.java`.

6 Availability and Licensing of the Corpus

All uses of the CHILDES corpus have general licensing requirements.[15] For this derived corpus, also cite this document.

The xml version is available for download at `http://cogcomp.org/page/resource_view/115`. The CHAT version is available through TalkBank at `https://childes.talkbank.org/derived/`.

7 Conclusion

In this work, we have described a new resource for NLP studies. It applies a commonly used gold standard for shallow semantic labeling, Propbank, as well as a preposition SRL to adult-child interactions. We explained the methods and tools needed for completing the annotation project, and provided support for the quality of the annotation through inter-annotator agreement measures on held-out data.

Acknowledgements

The BabySRL project is funded by NIH grant R01-HD054448 and NSF grant BCS-060257. We would like to thank the anonymous mentor who gave us excellent feedback on our earlier draft, and the three anonymous reviewers who gave helpful comments.

References

Collin F Baker, Charless J Fillmore, and John B Low. 1998. The Berkeley FrameNet project. *COLING-ACL '98: Proceedings of the Conference*, Montreal, Canada.

Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. Bracketing guidelines for Treebank II style Penn Treebank Project. Technical report, Linguistic Data Consortium.

Claire Bonial, Julia Bonn, Kathryn Conger, Jena Hwang, and Martha Palmer. 2014. Propbank: Semantics of new predicate types. *The 9th Edition of Language Resources and Evaluation Conference*, Reykjavik, Iceland.

Claire Bonial, Kathryn Conger, Jena Hwang, Aous Mansouri, Yahya Aseri, Julia Bonn, Tim O'Gorman, and Martha Palmer. forthcoming. Current directions in English and Arabic Propbank. In *The Handbook of Linguistic Annotations*.

R. Brown. 1973. *A First Language: The Early Stages*. Cambridge: Harvard University Press.

Michael Connor, Yael Gertner, Cynthia Fisher, and Dan Roth. 2010. Starting from scratch in semantic role labeling. *Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL)*, Uppsala, Sweden.

Michael Connor. 2011. *Minimal Supervision for Language Learning: Bootstrapping Global Patterns from Local Knowledge*. PhD thesis, University of Illinois at Urbana-Champaign.

Michael Connor, Cynthia Fisher, and Dan Roth. 2012. Starting from Scratch in Semantic Role Labeling: Early Indirect Supervision. *Cognitive Aspects of Computational Language Acquisition*.

Jino D. Choi, Claire Bonial, and Martha Palmer. 2010. Propbank Instance Annotation Guidelines Using a Dedicated Editor, Jubilee *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, 2010.

David Dowty. 1991. Thematic Proto-roles and Argument Selection. *Language* 67, 547-619.

Anne Fernald, Michael Frank, Noah Goodman, and Joshua Tenenbaum. 2009. Continuity of discourse provides information for word learning. *Proceedings of the Annual Meeting of the Cognitive Science Society*.

Cynthia Fisher, Yael Gertner, Rose M Scott, and Sylvia Yuan. 2010. Syntactic bootstrapping. *Wiley Interdisciplinary Reviews: Cognitive Science*, 1(2):143–149, March.

Yael Gertner and Cynthia Fisher. 2012. Predicted errors in children's early sentence comprehension. *Cognition. 2012 July; 124(1):85-94*.

[15] The general usage guidelines are found at `https://talkbank.org/share/rules.html`.

Yael Gertner, Cynthia Fisher, J. Eisengart. 2006. Learning words and rules. *Psychological Science*, 17.

Dan Gildea and Martha Palmer. 2002. The necesity of parsing for predicate argument recognition. In *Proceedings of the ACL 2002*, Philadelphia, Pennsylvania.

Paul Kingsbury and Martha Palmer. 2002. From Treebank to Propbank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, Las Palmas, Spain.

Jill Lany and Jenny Saffran. 2010 From statistics to meaning: Infant's acquisition of lexical categories. *Psychological Science* Feb 21(2).

Brian MacWhinney. 2000. *The CHILDES Project: Tools for Analyzing Talk*. Lawrence Erlbaum Associates, Inc., 3rd edition.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. Annotating Noun Argument Structure for NomBank. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC-2004)*.

Marcus P. Mitchell, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2).

L. Pearl and J. Sprouse. 2013. Syntactic islands and learning biases: Combining experimental syntax and computational modeling to investigate the language acquisition problem. *Language Acquisition*, 20, 23-68.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The Proposition Bank: A corpus annotated with semantic roles. *Computational Linguistics Journal*, 31(1).

K. Sagae, E. Davis, A. Lavie, B. MacWhinney, and S. Wintner. 2010. Morphosyntactic annotation of CHILDES transcripts. *Journal of child language* 37(3), 705-729.

K. Sagae, E. Davis, A. Lavie, B. MacWhinney, and S. Wintner. 2007. High-accuracy annotation and parsing of CHILDES transcripts. In *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition* 25-32. Association for Computational Linguistics.

Beatrice Santorini. 1990. Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision).

Nathan Schneider, Jena D. Hwang, Vivek Srikumar, Meredith Green, Kathryn Conger, Tim O'Gorman, and Martha Palmer. 2016. A corpus of preposition supersenses in English web reviews In *Computing Research Repository (CoRR)*

Rushen Shi and Andreane Melancon. 2010. Syntactic categorization in French-learning infants. *Infancy*, 15(5).

Vivek Srikumar and Dan Roth. 2011. A joint model for extended semantic role labeling. In *EMNLP 2011 - Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 129-139.

Vivek Srikumar 2013. The semantics of role labeling. Unpublished dissertation. University of Illinois at Urbana-Champaign.

Colin Warner, Arrick Lanfranchi, Tim O'Gorman, Amanda Howard, Kevin Gould, and Michael Regan. 2012. Bracketing biomedical text: An addendum to Penn Treebank II Guidelines. Technical report, Institute of Cognitive Science, University of Colorado at Boulder.

Multi-layer Representation Fusion for Neural Machine Translation

Qiang Wang[†], **Fuxue Li**[†,‡], **Tong Xiao**[†*], **Yanyang Li**[†], **Yinqiao Li**[†], **Jingbo Zhu**[†]
[†]Natural Language Processing Lab., Northeastern University
[‡]YingKou Institute of Technology
wangqiangneu@gmail.com, lifuxue119@163.com
xiaotong@mail.neu.edu.cn, blamedrlee@outlook.com
li.yin.qiao.2012@hotmail.com, zhujingbo@mail.neu.edu.cn

Abstract

Neural machine translation systems require a number of stacked layers for deep models. But the prediction depends on the sentence representation of the top-most layer with no access to low-level representations. This makes it more difficult to train the model and poses a risk of information loss to prediction. In this paper, we propose a multi-layer representation fusion (MLRF) approach to fusing stacked layers. In particular, we design three fusion functions to learn a better representation from the stack. Experimental results show that our approach yields improvements of 0.92 and 0.56 BLEU points over the strong Transformer baseline on IWSLT German-English and NIST Chinese-English MT tasks respectively. The result is new state-of-the-art in German-English translation.

1 Introduction

Neural models that use the encoder-decoder architecture to capture the translation equivalence relation between languages have been widely adopted over the last few years. The simplest of these relies on one recurrent neural network layer on both the encoder and decoder sides (Bahdanau et al., 2015), whereas others have successfully explored the high-level representation of language via deeper models (Wu et al., 2016; Gehring et al., 2017b; Vaswani et al., 2017). It has been noted that increasing the network depth is one of the factors contributing to the success of neural machine translation (NMT). To this end, one can stack a number of layers for an enriched sentence representation. E.g., popular NMT systems require 4 stacked layers or more for state-of-the-art results on large-scale translation tasks (Wu et al., 2016). Unfortunately, a straightforward implementation of deep neural networks has been found to underperform shallow models due to the poor convergence. One solution to this problem is to add identity mapping (or shortcut connection) between layers. A popular method is the residual network (He et al., 2016a), which has been used in several successful systems, e.g., GNMT and Transformer.

This model offers a good way to ease the information flow in the stack. But they do not make full use of the multi-level representations of the deep networks. E.g., word predictions depend on the sentence representation of the top-most layer with no access to low-level representations. Recent evidence supports that it helps when features from lower-level layers are involved in prediction (Xiong et al., 2017; Gehring et al., 2017b). A good instance is DenseNet that applies a fully-connected convolutional network to computer vision (Huang et al., 2017a). But it is still rare to see studies on this issue in machine translation.

In this paper, we take a further step along the line of this research. Drawing on previous successful attempts to create shortcut connections between adjacent layers, we propose a multi-layer representation fusion (MLRF) approach that connects one layer to all its predecessors. It learns a fused representation by accessing all lower-level representations of sentence, rather than learning from the limited information flow in one residual connection. The contributions of this work are three-fold.

- We propose an approach to learning better sentence representations by accessing lower-level layers. To our knowledge, it is the first time to use densely connected networks in NMT.

* Corresponding author

Proceedings of the 27th International Conference on Computational Linguistics, pages 3015–3026
Santa Fe, New Mexico, USA, August 20-26, 2018.

- We investigate methods to learn functions that fuse different levels of sentence representations. In particular, we propose to learn a 2D representation by the self-attention mechanism.
- We demonstrate the effectiveness of the approach on top of a strong system (Transformer) which has already used residual networks. It achieves a new state-of-the-art result on the IWSLT German-English MT task.

Our approach is applicable to arbitrary neural architectures and is easy to be implemented. In experiments on IWSLT German-English and NIST Chinese-English translation, it yields improvements of 0.92 and 0.56 BLEU points over the baseline respectively. More interestingly, we find that our approach has a regularizing effect and reduces the risk of over-fitting.

2 The Transformer System

In this work, all discussions and experiments are based on the Transformer system. We choose Transformer because it is one of the most successful NMT systems in recent MT evaluations. Unlike usual NMT models, Transformer does not require any recurrent units for modeling word sequences of arbitrary length. Instead, it resorts to self-attention and standard feed-forward networks for both encoder and decoder.

On the encoder side, there are L identical stacked layers. Each of them is composed of a self-attention sub-layer and a feed-forward sub-layer. The attention model used in Transformer is scaled dot-product attention[1]. Its output is fed into a fully connected feed-forward network. To ease training, the output of each sub-layer is defined as $\text{LayerNorm}(x + \text{sublayer}(x))$, where $\text{LayerNorm}(\cdot)$ is layer normalization (Ba et al., 2016) and $\text{sublayer}(x)$ is the output of the sub-layer. The identical mapping of input x represents the residual connection. To facilitate description, we use $H = \{h^1, \ldots, h^L\}$ to denote the outputs of source-side layers in this paper [2].

Likewise, the decoder has another stack of L identical layers (denoted as the function $\text{Layer}(\cdot)$). It has an encoder-decoder attention sub-layer in addition to the two sub-layers used in each encoder layer. Moreover, because the model is auto-regressive, the decoder attends a target position to all positions up to it. Let z_j^l be the output vector of target position j in the l-th layer, and $z_{\leq j}^{l-1}$ be the vector sequence $\{z_1^{l-1}, \ldots, z_j^{l-1}\}$. The output of the layer can be described as $z_j^l = \text{Layer}(z_{\leq j}^{l-1}, H)$.

The auto-regressive property also persists in the output layer. Given a source sentence $\mathbf{x} = (x_1, \ldots, x_M)$, and a target sentence $\mathbf{y} = (y_1, \ldots, y_N)$, the translation model is defined to be:

$$P(\mathbf{y}|\mathbf{x}) = \prod_{j=1}^{N} \Pr(y_j|y_{<j}, \mathbf{x}) \tag{1}$$

where $\Pr(y_j|y_{<j}, \mathbf{x})$ is the probability of generating a target word y_j given the previously generated words $y_{<j}$ and the source sentence $\mathbf{x}$. To model $\Pr(y_j|y_{<j}, \mathbf{x})$, we have

$$\Pr(y_j|y_{<j}, \mathbf{x}) = \text{Softmax}(W_o \cdot z_j^L + b_o) \tag{2}$$

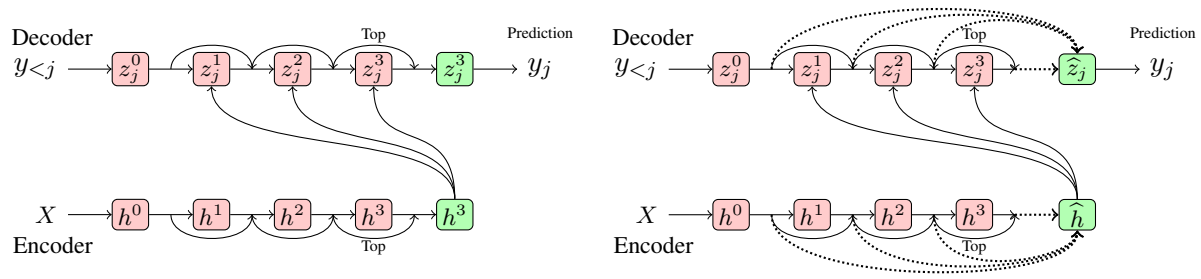
where W_o and b_o are the model parameters of the output layer. Obviously, the word prediction model here depends only on the output of the top-most layer in the stack (i.e., z_j^L).

3 The Approach

The model presented in Section 2 shows a standard way of word prediction where the output layer is connected to the top-most hidden layer only. This is fine if the system is composed of one hidden layer or two. But it may be problematic when the network goes deeper. First, this method has difficulties

[1] Given a sequence of vectors and a position i, the self-attention model computes the dot-product of the input vectors for each pair of positions (i, j), followed by a rescaling operation and Softmax. In this way, we have an attention score (or weight) for each (i, j). It is then used to generate the output by a weighted sum over all input vectors.

[2] We regard the word embedding layer as a special layer at the bottom of the stack, and denote it as h^0.



(a) Standard Architecture (b) The MLRF Architecture

Figure 1: Network Architectures. Dotted lines denote multi-layer fusion connections. Green rectangles denote the representations used in the downstream components.

in training deep models. To learn model parameters, gradients have to be propagated through a single channel of several stacked layers. Previous work has pointed out that it is not easy for lower-level layers to find good parameters due to the vanished or exploded gradients (Srivastava et al., 2015; He et al., 2016a). The situation is even worse when we use many non-linear transformations that are hard to learn. The residual connection between adjacent layers can alleviate the problem, but it is still a long distance for bottom layers to have direct feedbacks from the prediction. Second, there is potential information loss when we feed a single layer to the output layer. Top-level layers may forget previous features, especially when the stacked layers have the same capacity (e.g., the same size of layer output). For a simple solution, one can enlarge the layer size on the top. But it in turn drastically increases the number of parameters and is obviously not efficient for practical systems.

In this work, we propose a multi-layer representation fusion method to address these problems. It learns a fused representation of the sentence by direct access to all the layers in the stack. In this way, the prediction can benefit from the fused representation which has less information loss. As another bonus, this method makes it easier for the deep network to learn parameters because of the efficient information flow in connections from bottom to top.

3.1 Multi-layer Representation Fusion

The idea of representation fusion is pretty simple. We introduce a fusion layer into the network, and connect it to all the stacked layers. This layer can learn a refined representation from different levels of the stack. Let $Z = \{z^1, \ldots, z^L\}$ be the output sequence of the stacked layers on the decoder side. We define $\phi(Z)$ to be the fusion function that fuses $\{z^1, \ldots, z^L\}$ into a single representation. More specifically, given a target word position j, $\widehat{z}_j = \phi(Z_j)$ is defined to be the fused representation of all layer representations $Z_j = \{z_j^1, \ldots, z_j^L\}$ in the stack for position j. Then Eq. (2) can be rewritten as:

$$\Pr(y_j|y_{<j}, \mathbf{x}) = \text{Softmax}(W_o \cdot \phi(Z_j) + b_o) \tag{3}$$

There are many choices to design the fusion function $\phi(Z_j)$. E.g., we can cast the usual method as a special case of our model and do exactly the same thing as in Section 2, i.e., $\phi(Z_j) = z_j^L$. Alternatively, we can fuse more layers with another neural network. Moreover, we can do representation fusion on either the encoder side, the decoder side, or both. See Figure 1 for an illustration of the method.

Another note on representation fusion. The method is applicable to arbitrary neural network architectures with multiple levels of layers. Although we restrict ourselves to Transformer for our experiments, Eq. (3) actually shows a more general case. In a sense, our model is doing something similar to fully connected networks in computer vision (Huang et al., 2017a; Liu et al., 2018). Representation fusion is essentially a process that creates a densely connected network on top of the stack. In this work, we describe the problem in the more general framework, and will show that NMT systems can benefit from better design of fusion functions.

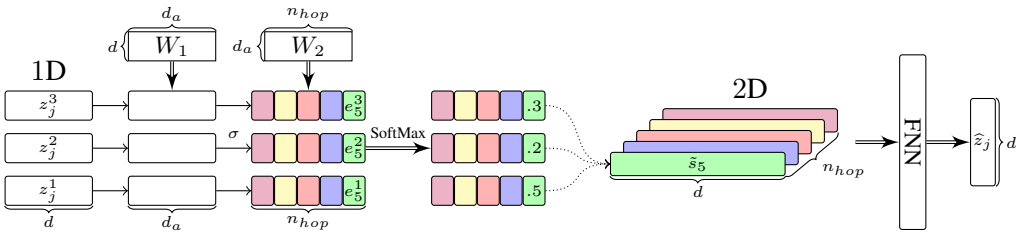


Figure 2: An example of 3-layer representation fusion based on self-attention with 5 hops.

3.2 Fusion Functions

Fusion by Pooling The simplest way of representation fusion is pooling. Here we choose average pooling (avg-pooling) because it shows better results than other pooling methods in our preliminary experiments. Given a sequence of layer outputs at position j, the output of average pooling-based fusion is $\phi(Z_j) = \frac{1}{L}\sum_{l=1}^{L} z_j^l$. In general, this way can be seen as the natural generalization of residual connections (He et al., 2016b; He et al., 2016a). It accumulates all layer features at once when going through the stack, rather than performing identity mapping between adjacent layers at each time. Also, this method is nonparametric and simply assigns an equal weight to each layer in fusion.

Fusion by Feed-forward Neural Network A more sophisticated method is to use feed-forward neural networks (FNNs) for fusion. To this end, we first concatenate all layer outputs to form a big vector. Then, we feed the vector into a single hidden layer FNN with d_f hidden units. Let d be the size of the layer output vector (i.e., $d = |z_j^l| \ \ \forall j, l$). The FNN transforms an $L * d$ dimensional vector to a d dimensional vector. This is good because we do not need to change the size of the downstream components. Moreover, we can learn a better fusion function by the non-linear activation functions used in FNNs. It should be noted that this method can be seen as a special case of dense connection, that is we only concatenate all the features of predecessors in the topmost layer rather than every intermediate layer. As a result, it can reduce the connection complexity from $\mathcal{O}(L^2)$ to $\mathcal{O}(L)$.

Fusion by Multi-hop Self Attention In addition to the pooling and FNN-based methods, we design another fusion function based on the self-attention model with hops [3] (Figure 2). The motivation is straightforward. Different layers reflect different aspects of sentence representation. An ideal way is to take correlations between layers into account, and model the correlation strength (or weight) in generating the fused result. This agrees with the nature of the attention mechanism well (Bahdanau et al., 2015). Beyond this, self-attention with hops can generate a 2D output that explains different meanings of the input vectors. Previous work has reported that 2D matrix-based representations outperform 1D vector-based counterparts in sentence embedding (Lin et al., 2017). So it is worth a study on this issue in NMT.

The idea is that we apply self-attention with multiple hops vertically over the layer sequence (along the axis of depth), and extract a part of layer information by each hop. The new problem that arises here is that we need to perceive the temporal order information in sequence (Shen et al., 2018). Unlike Vaswani et al. (2017)'s work, we do not resort to a fixed positional encoding generated by sine and cosine functions of different frequencies. Instead, we learn an embedding of the layer index, as used in positional embedding (Gehring et al., 2017a). The output of layer embedding is of size d, so it is compatible with the layer representation vector z_j^l. Also, we use a shared layer embedding for the encoder and the decoder for simple implementation.

More formally, let $E^l \in \mathbb{R}^d$ be the embedding of layer l. We inject layer embedding into the original representation of the layer, like this:

$$\tilde{z}^l = z^l + E^l \tag{4}$$

[3] In this work, the term "hop" denotes an independent attention computation, which has the same meaning as Lin et al. (2017). More specifically, one hop can generate a vector as the attention distribution over the layers. And different hops may generate the distinguished distributions.

Entry	Fusion Function	Low-level accessible	Parametric	2D represent.
Baseline	$\phi(Z_j) = z_j^L$	No	No	No
Avg-pooling	$\phi(Z_j) = \frac{1}{L}\sum_{l=1}^{L} z_j^l$	Yes	No	No
FNN-based	$\phi(Z_j) = \text{FNN}([z_j^1, ..., z_j^L])$	Yes	Yes	No
Self-att.-based	$\phi(Z_j) = \text{FNN}(\text{SelfAtt}([z_j^1, ..., z_j^L]))$	Yes	Yes	Yes

Table 1: Comparison of the fusion functions used in this work.

where $\tilde{z}^l \in \mathbb{R}^d$ is the new representation that encodes both the content and index information of layer l. Then the self-attention method is performed over $(\tilde{z}^1, ..., \tilde{z}^L)$. For hop p, the energy of attention weight is computed by an FNN with a single hidden layer,

$$e = W_2(\sigma(W_1 \cdot [\tilde{z}^1, ..., \tilde{z}^L]^T)) \tag{5}$$

where $[\tilde{z}^1, ..., \tilde{z}^L] \in \mathbb{R}^{d \times L}$ is the 2D layout of layer representations, $W_1 \in \mathbb{R}^{d \times d_a}$ and $W_2 \in \mathbb{R}^{d_a \times n_{hop}}$ are model parameters, σ is the activation function, and d_a is the size of inner hidden state. In this model, each hop has an independent parameter vector corresponding to one column of W_2, whereas W_1 is shared across different layers to make efficient use of memory [4]. After that, we obtain the attention weight vector $\mathrm{a}_p = (a_p^1, ..., a_p^L)$ of hop p by Softmax. For each layer index l, we have

$$a_p^l = \frac{\exp(e_p^l)}{\sum_{l'=1}^{L} \exp(e_p^{l'})} \tag{6}$$

The self-attention model generates an intermediate representation by dot-production ($\odot$) of the weights and the layer representations.

$$\tilde{s}_p = a_p \odot [\tilde{z}^1, ..., \tilde{z}^L] \tag{7}$$

In this way, all the hops compose a 2D matrix $\tilde{s} = [\tilde{s}_1, ..., \tilde{s}_{n_{hop}}] \in \mathbb{R}^{d \times n_{hop}}$. This matrix is fed into another single hidden layer FNN with d_f hidden units for the final output. It is worth noting that the model presented here can be enhanced by using an independent weight $a_{p;k}^l$ for each dimension k of $\tilde{z}^l$ (call it feature-wise fusion). But this method is computationally expensive. In our experiments we do not observe improvements by this method. Therefore, we choose the version described in Eqs. (6-7) for the empirical study.

Table 1 shows a comparison of the methods used in this work. For good convergence, we use layer normalization after the fusion layer in this work.

4 Experiments

We evaluate our proposed approach on German-English and Chinese-English translation tasks.

4.1 Setup

For German-English translation, we use the data of the IWSLT 2014 German-English track (Cettolo et al., 2014). We follow Ranzato et al. (2015)'s work for preprocessing. We use a joint source and target byte-pair encoding with 10k merge operations (Sennrich et al., 2016). The source and target vocabulary sizes are 8,389 and 6,428 respectively. We remove the sentences with more than 175 words or 100 sub-word units. This results in 160K sentence pairs for training. We randomly sample 7K sentences from the training data for held-out validation, and concatenate dev2010, dev2012, tst2010, tst2011, and tst2012 for test.

For Chinese-English translation, we use parts of the bitext provided within NIST12 OpenMT[5]. We choose NIST 2006 (MT06) as the validation set, and 2004 (MT04), 2005 (MT05), 2008 (MT08) as the

[4] W_1 can be unique for each layer. We discuss this issue in Section 4.4.

[5] LDC2000T46, LDC2000T47, LDC2000T50, LDC2003E14, LDC2005T10, LDC2002E18, LDC2007T09, LDC2004T08

System		#Param.	Valid.	Test
Existing Systems	RNN-MIXER (Ranzato et al., 2015)	-	-	20.73
	RNN-BSO (Wiseman and Rush, 2016)	-	-	26.36
	RNN-AC(Bahdanau et al., 2016)	-	-	28.53
	RNN-NPMT (Huang et al., 2017b)	-	-	28.96
	RNN-NPMT + LM (Huang et al., 2017b)	-	-	29.16
	ConvSeq2Seq-MLE (Edunov et al., 2017)	-	32.96	31.74
	ConvSeq2Seq-Risk (Edunov et al., 2017)	-	33.91	32.85
Baselines	Transformer-MLE	10.97M	33.58	31.75
	+RestartAdam	10.97M	34.14	32.67
	+RestartAdam-4Layers	12.82M	34.20	32.57
	+RestartAdam-6Layers	16.50M	34.35	32.97
MLRF Systems	Enc-AVG	10.97M	34.34	32.71
	Enc-FNN	11.63M	34.55	33.31
	Enc-SA (n_{hop}=4)	11.90M	34.48	33.06
	Enc-SA (n_{hop}=6)	12.16M	34.61	33.40
	Dec-AVG	10.97M	34.02	32.80
	Dec-FNN	11.63M	34.38	32.93
	Dec-SA (n_{hop}=4)	11.90M	**34.83**	33.29
	Dec-SA (n_{hop}=6)	13.47M	**34.73**	33.54
	Both-FNN	12.29M	34.30	32.66
	Both-SA ($n_{hop} = 4$)	12.82M	34.59	33.46
	Both-FNN-SA (n_{hop}=4)	12.55M	34.55	**33.59**

Table 2: BLEU scores [%] on IWSLT German-English translation.

test sets. All Chinese sentences are word segmented using the tool provided within NiuTrans (Xiao et al., 2012). All sentences of more than 50 words are removed. The resulting training corpus consists of 1.85M sentence pairs, with 39.42M Chinese words and 44.92M English words on each language side. We limit the vocabularies to the most frequent 30K words, covering approximately 98.16% and 99.17% of the Chinese and English words respectively. All out-of-vocabulary words are replaced with <UNK>. We report results without any UNK-replacement techniques (Luong et al., 2015).

4.2 Implementation Details

For German-English systems, the model consists of a 3-layer encoder and a 3-layer decoder with $d = 256$ and 1024 hidden units in the FNN sub-layer. For our approach, we set $d_a = 1024$ and $d_f = 512$. Dropout (rate$= 0.1$) is used for regularization. We initialize all word/sub-word embedding matrices using a normal distribution $\mathcal{N}(0, d^{-0.5})$, while initialize the layer embedding matrix using a uniform distribution $\mathcal{U}(-0.1, 0.1)$. The weight and bias in layer-normalization are initialized to constants 1 and 0. All other parameters are initialized from $\mathcal{U}(-\frac{1}{\sqrt{fan_{in}}}, \frac{1}{\sqrt{fan_{in}}})$, where fan_{in} is the input size of the parameter matrix. We use negative Maximum Likelihood Estimation (MLE) as loss function, and train all the models using Adam (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$. We run training for 40 epochs with a mini-batch of 80. The learning rate is scheduled as described in (Vaswani et al., 2017): $lr = d^{-0.5} \cdot min(t^{-0.5}, t \cdot 16\text{k}^{-1.5})$, where t is the step number. After that, we restart Adam and continue the training for additional 20 epochs with a fixed learning rate $5e^{-5}$ and a smaller mini-batch of 32 (Denkowski and Neubig, 2017). At test time, translations are generated by beam search with length normalization. By tuning on the validation set, we use a beam of width 8 and a length normalization weight of 1.6.

For Chinese-English systems, we use a 6-layer encoder and a 6-layer decoder, with $d = 512$ and 2048 hidden units in the FNN sub-layer. We restart Adam after 10 epochs and train the model for 5 additional epochs. A beam of width 12 and a length normalization weight of 1.3 are employed.

Note that the network equipped with our fusion layer increases a fraction of the computation cost than original one. E.g., in Chinese-English translation, the training speed reduces from 2.6 batches/second

System		Valid.	Test			
		MT06	MT04	MT05	MT08	Ave.
Open Source Systems	Nematus-4Layers	40.49	46.83	39.40	32.73	39.65
	NMTTutorial-GNMT	41.29	47.43	40.18	33.67	40.43
	T2T	41.87	47.42	41.09	34.39	40.97
Baselines	Transformer-MLE	40.60	47.70	40.33	34.70	40.91
	+RestartAdam	41.82	48.09	40.76	34.75	41.20
MLRF Systems	Enc-AVG	42.17	48.35	41.24	34.93	41.51
	Enc-FNN	40.81	47.44	40.03	33.55	40.34
	Enc-SA ($n_{hop} = 4$)	40.17	46.79	39.55	32.80	39.71
	Dec-AVG	41.43	48.38	41.19	34.66	41.49
	Dec-FNN	41.97	48.14	41.36	34.38	41.23
	Dec-SA ($n_{hop} = 4$)	42.13	48.48	**41.51**	34.79	41.59
	+ indep. W_1	**42.26**	**48.75**	41.44	**35.08**	**41.76**
	Both-AVG-SA ($n_{hop} = 4$)	41.61	47.65	40.69	34.52	40.95

Table 3: BLEU scores [%] on NIST Chinese-English translation.

(baseline) to 2.4 batches/second (Dec-SA, ref. row 11 in Table 3).

4.3 Results on German-English Translation

Table 2 shows the BLEU scores of various NMT systems[6]. For comparison, we list previous results on the same data set. First of all, our baseline is good. The base setting of our baseline system can lead to a similar score to ConvSeq2Seq which is trained for more epochs (Edunov et al., 2017). The baseline is stronger when Adam restart is adopted. It outperforms most of previous systems on this task (only lower than the best reported system by 0.2 BLEU points).

Then, we test different fusion functions (AVG, FNN and SA) on the encoder side (Enc), the decoder side (Dec) and both of them (Both). We see that avg-pooling does not yield promising improvements due to the relatively low expressive power. In addition, the representation fusion on the decoder side is more effective than that on the encoder side. A possible reason is that the prediction can benefit more from the lower-level layers in the decoder due to the "shorter" distance from fusion to prediction. This agrees with the result that representation fusion on both sides (Both-SA) does not improve Enc-SA and Dec-SA significantly. The best result is achieved when we use FNN-based fusion on the encoder side and self-attention-based fusion on the decoder side (Both-FNN-SA). It outperforms the 3-layer baseline by 0.92 BLEU points. Also, we increase the number of layers to 4 and 6 to examine whether the improvement comes from the additional model parameters introduced by the added fusion layer. It results in stronger baselines, but they still underperform the Both-FNN-SA system which has fewer parameters.

4.4 Results on Chinese-English Translation

For Chinese-English translation, we compare our systems with three open source systems: Nematus (Sennrich et al., 2017), NMTTutorial (Luong et al., 2017) and T2T [7]. The first two are based on RNN, while the last one is based on Transformer. Table 3 shows that our baseline with restarting Adam is a bit better than T2T. Like in German-English translation, fusion on the decoder side is superior than that on the encoder side. But both Enc-FNN and Enc-SA are worse than the baseline. We suspect that adding more layers with non-linear transformations makes it more difficult to train the deep network (6 layers in Chinese-English translation) than the shallow counterpart (3 layers in German-English translation). This problem is more evident here because the Chinese-English systems have many more parameters than the German-English systems and are more difficult to optimize. Interestingly, we see that the simple avg-pooling fusion method works well in both encoder and decoder. In addition, using independent W_1 (in

[6] All BLEU results are case-insensitive. For a fair comparison with previous work, we run *multi-bleu.perl* for German-English translation, and run *mteval-v13a.pl* for Chinese-English translation.

[7] https://github.com/tensorflow/tensor2tensor

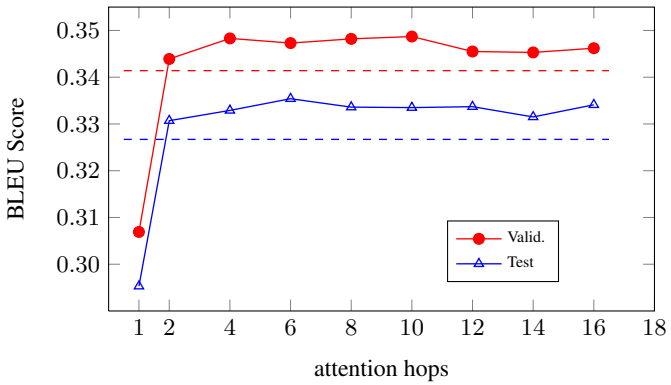


Figure 3: BLEU scores under various attention hops in Dec-SA on German-English translation. Dashed lines denotes the baseline.

System	Word Emb.	Layer Emb.	Test
Enc-FNN	√	√	33.31
	√		33.17
		√	33.07*
Dec-SA	√	√	33.29
	√		33.06*
		√	33.21
Both-FNN-SA	√	√	33.59
	√		33.55
		√	33.16*

Table 4: BLEU scores on German-English translation when fusing with (denoted by √)/without word embedding and layer embedding in encoder, decoder, and both of them. * denotes significant performance reduction (> 0.2 BLEU) than baseline (with both word embedding and layer embedding).

self-attention) for each layer is helpful for better fitting. It indicates that the MLRF method can benefit from more sophisticated design of the self-attention model.

4.5 Effect on Attention Hops

Figure 3 shows the BLEU curves by varying n_{hop} on German-English translation with the Dec-SA approach. Clearly, increasing the number of hops improves the system. The improvement is significant when $n_{hop} < 6$ thanks to the 2D representations extracted by multiple hops. Note that when $n_{hop} = 1$, the 2D sentence representation is degraded into the 1D representation. It results in a dramatic decrease in BLEU. However, larger n_{hop} does not make further improvements due to the redundant information in hops and potential overlaps between them.

4.6 Importance of Word Embedding and Layer Embedding

Then, we study the model behavior with/without word embedding or layer embedding on German-English translation. Table 4 shows that removing either word embedding or layer embedding harms the Enc-FNN system. The impact of word embedding is a bit more than that of layer embedding. As a contrast, Dec-SA is more sensitive to the remove of layer embedding. We attribute it to the unawareness of the temporal order information in the self-attention mechanism. More interestingly, when we fuse representations on both sides, almost no BLEU reduction is observed in Both-FNN-SA. It is true even if layer embedding is removed. This result indicates that the layer indexing information can be delivered from the FNN fusion layer in the encoder to the decoder in an implicit way. Also, a significant improvement can be achieved by fusing with the original word embedding features. It agrees with the result reported in (Xiong et al., 2017).

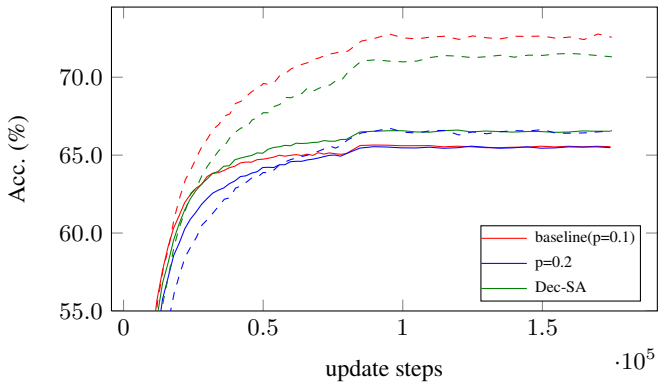
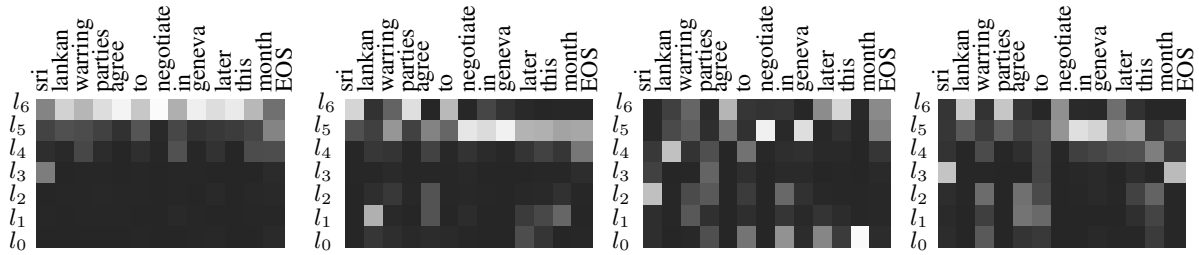


Figure 4: Curve of Accuracy in training-set (dashed lines) and validation-set (solid lines) along with update steps. p is dropout rate.



(a) Hop 1 (b) Hop 2 (c) Hop 3 (d) Hop 4

Figure 5: A visualization example of Dec-SA with 4 hops in Chinese-English translation. Colors change between white (probability of 1) to black (probability of 0). The source sentence (pinyin) is *Sīlǐlánkǎ jiāozhàn shuāngfāng tóngyì běnyuè xiàxún zài rìnèiwǎ tánpàn*.

4.7 Regularization

Also, we plot the curves of accuracy on the training and validation sets in Figure 4 (German-English translation). It is obvious that the baseline with dropout rate $p = 0.1$ (red) has a significant lower BLEU score when we switch from training to validation, implying that the model somehow over-fits the data. When we use a larger $p = 0.2$ (blue) for dropout, the accuracy on the training set decreases dramatically, but no promising improvement is observed on the validation set. On the contrary, Dec-SA seems to have a better regularization effect. Although Dec-SA has lower accuracy in training than the baseline, it outperforms the baseline on the validation set. It might be due to the introduced direct connections from lower-level layers to the top of the stack. These connections make the model easier to fit the data because they have no dependence on the non-linear transformations in the stacked layers.

4.8 Layer Attention Visibility

A visualization example of layer attention with multiple hops is presented in Figure 5. The attention result is generated from an example of Chinese-English translation by Dec-SA with 4 hops. The x-axis is the target word sequence, and the y-axis is the indexing of layers (including the word embedding layer denoted by l_0). We can see that most attention weights focus on the high-level layers like hops 1 and 2, while hops 3 and 4 have more dispersive attentions over different layers. It indicates the different aspects of the sentence encoded in different hops. An interesting finding is that those large probability points in the lower-level layers are easier to appear when a noun or pronoun is fed into the decoder. For example, when we feed *sri* into the decoder to generate *lankan*, the first layer is obviously more important in hop 2. Similar cases can be observed for words *geneva* and *this* in hop 3, where the word embedding layer has a larger weight. This is reasonable because most of these words have specific meanings and do not need high-level representations for modeling large context in disambiguation.

5 Related Work

Training neural networks with multiple stacked layers is challenging. It has been observed that introducing direct connections between layers can drastically improve the performance of deep neural models. Methods include highway networks (Srivastava et al., 2015), residual connections (He et al., 2016a), dense connections (Huang et al., 2017a), and fast-forward connections (Zhou et al., 2016). E.g., in machine translation, residual networks have been a popular way to address the issue due to its simplicity (Wu et al., 2016; Gehring et al., 2017a; Gehring et al., 2017b; Vaswani et al., 2017). Another related study is Wang et al. (2017). They introduce linear associative units to reduce the length of gradient propagation in recurrent neural networks (RNNs), and demonstrate promising improvements on their RNN-based NMT systems. But previous studies all focus on using the top-level sentence representation for prediction, and ignore the access to the representations encoded in lower-level layers.

The next obvious step is toward models that make full use of all stacked layers for prediction (call it representation fusion). Some research groups have been aware of this and explored solutions. E.g., Gehring et al. (2017b) find that NMT systems can benefit from shortcut connections from the source word embedding layer to the attention layer. Perhaps the most related work is Xiong et al. (2017). They propose a multi-channel encoder (MCE) which uses an external memory module (Graves et al., 2014) to compose word embeddings and hidden states in their RNN-based encoder. But this model is applied to the shallow network on the encoder side. In this work we instead propose a more general method to do representation fusion in either the encoder, the decoder, or both. In addition, we present a 2D representation of sentence which has not been well studied in machine translation.

6 Conclusion

We have proposed a multi-layer representation fusion approach that densely connects all the stacked layers to a fusion layer for encoder or/and decoder in NMT. Also, we have developed three fusion functions to learn a better representation of sentence. Experimental results on German-English and Chinese-English translation show that our method outperforms the strong Transformer baseline significantly, and achieves new state-of-the-art on the IWSLT German-English MT task. More interestingly, it is observed that our approach has a regularizing effect and reduces the risk of over-fitting.

Acknowledgements

This work was supported in part by the National Science Foundation of China (No. 61672138, 61432013 and 61572120), the Fundamental Research Funds for the Central Universities. The authors would like to thank anonymous reviewers and Chunliang Zhang for their comments.

References

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. CoRR, abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In In Proceedings of the 3rd International Conference on Learning Representations.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. arXiv e-prints, abs/1607.07086, July.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014. In Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam.

Michael Denkowski and Graham Neubig. 2017. Stronger baselines for trustable results in neural machine translation. In Proceedings of the First Workshop on Neural Machine Translation, pages 18–27. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2017. Classical structured prediction losses for sequence to sequence learning. arXiv preprint arXiv:1711.04956.

Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017a. A convolutional encoder model for neural machine translation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 123–135. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017b. Convolutional Sequence to Sequence Learning. ArXiv e-prints, May.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016a. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016b. Identity mappings in deep residual networks. In European Conference on Computer Vision, pages 630–645. Springer.

Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. 2017a. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, volume 1, page 3.

Po-Sen Huang, Chong Wang, Dengyong Zhou, and Li Deng. 2017b. Neural phrase-based machine translation. arXiv preprint arXiv:1706.05565.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

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

Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. 2018. Path aggregation network for instance segmentation. arXiv preprint arXiv:1803.01534.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 11–19. Association for Computational Linguistics.

Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. 2017. Neural machine translation (seq2seq) tutorial. https://github.com/tensorflow/nmt.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics, pages 65–68, Valencia, Spain, April. Association for Computational Linguistics.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In AAAI Conference on Artificial Intelligence.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28, pages 2377–2385. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems, pages 6000–6010.

Mingxuan Wang, Zhengdong Lu, Jie Zhou, and Qun Liu. 2017. Deep neural machine translation with linear associative unit. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers, pages 136–145.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016, pages 1296–1306.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

Tong Xiao, Jingbo Zhu, Hao Zhang, and Qiang Li. 2012. Niutrans: an open source toolkit for phrase-based and syntax-based machine translation. In Proceedings of the ACL 2012 System Demonstrations, pages 19–24. Association for Computational Linguistics.

Hao Xiong, Zhongjun He, Xiaoguang Hu, and Hua Wu. 2017. Multi-channel encoder for neural machine translation. arXiv preprint arXiv:1712.02109.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. Transactions of the Association of Computational Linguistics, 4:371–383.

Toward Better Loanword Identification in Uyghur Using Cross-lingual Word Embeddings

Chenggang Mi[†‡], **Yating Yang**[†‡], **Lei Wang**[†‡], **Xi Zhou**[†‡], **Tonghai Jiang**[†‡]
[†]Xinjiang Technical Institute of Physics & Chemistry
of Chinese Academy of Sciences, China
[‡]Key Laboratory of Speech Language Information Processing of Xinjiang, China
{micg,yangyt,wanglei,zhouxi,jth}@ms.xjb.ac.cn

Abstract

To enrich vocabulary of low resource settings, we proposed a novel method which identify loanwords in monolingual corpora. More specifically, we first use cross-lingual word embeddings as the core feature to generate semantically related candidates based on comparable corpora and a small bilingual lexicon; then, a log-linear model which combines several shallow features such as pronunciation similarity and hybrid language model features to predict the final results. In this paper, we use Uyghur as the receipt language and try to detect loanwords in four donor languages: Arabic, Chinese, Persian and Russian. We conduct two groups of experiments to evaluate the effectiveness of our proposed approach: loanword identification and OOV translation in four language pairs and eight translation directions (Uyghur-Arabic, Arabic-Uyghur, Uyghur-Chinese, Chinese-Uyghur, Uyghur-Persian, Persian-Uyghur, Uyghur-Russian, and Russian-Uyghur). Experimental results on loanword identification show that our method outperforms other baseline models significantly. Neural machine translation models integrating results of loanword identification experiments achieve the best results on OOV translation(with 0.5-0.9 BLEU improvements).

1 Introduction

Almost every natural language processing (NLP) task suffers from data sparseness. This situation is even worse for low resource languages in cross lingual tasks, such as neural machine translation(NMT).Lexical borrowing happens in every language. It is a phenomenon of cross-linguistic influence. A loanword is a word adopted from one language (the donor language) and incorporated into another language (the recipient language) without translation. One reason we choose Uyghur as the example in this study is that Uyghur has been influenced by many languages, both oriental and western languages; another reason is that Uyghur language is low-resource and lack of research in NLP field. If loanwords in a resource poor language can be identified effectively, we can use the loanword and its corresponding donor word to extend the bilingual dictionary (Tsvetkov et al., 2015).

Some researchers take loanword identification as a string similarity problem. They have applied several commonly used methods to detect loanwords (Mi et al., 2014). However, these methods are usually rule-based, so these methods are difficult to deal with ambiguity. Machine learning based models have been proposed to overcome the shortcomings that exist in previous work. Lack of annotated corpora weakens the performance of these methods (Mi et al., 2016). Although some studies combine rule based and machine learning based methods to overcome data sparsity to a certain extent, challenges still exist due to semantic issues.

A loanword usually shares the same/similar semantic space with its corresponding source word in donor language. Word embeddings were first proposed to learn representations where words that have the same meaning have a similar representation (Mikolov et al., 2013). Cross-lingual word embedding is one way to learn word embeddings using information from multiple languages (Klementiev et al., 2012). This inspired us to introduce the cross-lingual embedding into the loanword identification. Additionally,

Proceedings of the 27th International Conference on Computational Linguistics, pages 3027–3037
Santa Fe, New Mexico, USA, August 20-26, 2018.

the loanword often has the similar pronunciation with its source word, the string similarity and character-level language model of receipt and donor languages can reflect this characteristic.

This paper proposes a novel loanword identification model based on both shallow features and semantically-related features. We first use a cross-lingual word embedding based model to generate the loanword candidates; then, an optimized loanword identification model based on shallow features such as pronunciation similarity and hybrid language model is built to output the final results. We conduct experiments on loanword identification in Uyghur and OOV translation in several low-resource language pairs separately.[1] Experimental results show that our proposed approach outperforms other baseline models significantly. The output of our loanword identification model can improve translation results by at least 0.5 BLEU points.

2 Related Work

Our study is mainly inspired by following two fields in NLP: loanword identification in NLP and cross-lingual word embedding.

2.1 Loanword Identification in NLP

Loanwords were mainly studied by linguists (Hoffer et al., 2005; Peperkamp, 2004; Kang et al., 2014; Shinohara, 2015) in the early period; there have been relatively few papers in loanwords research in natural language processing (NLP). (Tsvetkov and Dyer, 2016; Tsvetkov and Dyer, 2016a)proposed a morph-phonological transformation model, where features are based on optimality theory; experiments proved that with a few training examples, this model can obtain good performance at predicting donor forms from borrowed forms. (Tsvetkov et al., 2015a) suggest an approach that uses the lexical borrowing as a model in SMT framework to translate OOV words in a low-resource language. (Mi et al., 2014; Mi et al., 2016) use shallow features such as string similarity to detect loanwords in Uyghur.

2.2 Cross-lingual Word Embedding

Recent advances in cross-lingual word embedding have shown its effectiveness in several cross-lingual NLP tasks (Gouws et al., 2015; Ruder et al., 2017; Duong et al., 2017; Ammar et al., 2016), especially in some under-resourced situations (Adams et al., 2017; Fang and Cohn, 2017; Wei and Deng, 2017). Cross-lingual word embedding has the capacity to learn high quality embeddings even in the absence of bilingual corpora by exploiting bilingual lexicons. This is very useful when creation of high quality lexicons on some low-resource languages.

Previous work mainly focused on one specific language or some context free features. These methods are difficult to adopt in situations such as when the receipt and donor language belong to very different language families or low-resource settings. In this study, we propose a novel loanword identification approach, which is not only based on some shallow features, but also cross-lingual word embeddings.

3 Background

3.1 Loanwords in Uyghur

Uyghur is an official language of the Xinjiang Uyghur Autonomous Region in Western China, and is spoken by 10 to 25 million people. Uyghur is an agglutinative language. Due to the different kinds of language contact through the history of the Xinjiang region, Uyghur has also adopted many loanwords (most of them are nouns) from Persian. Additionally, many words of Arabic origin have also entered the language directly through Islamic literature after the introduction of Islam. Among all languages Uyghur borrowed, Russian and Chinese have the greatest influence in recent years. Due to the globalization, a number of loanwords of European origin have also reached Uyghur through Russia's influence over the region. Loanwords in Uyghur not only include named entities such as person and location names, but also some daily used words (Kontovas, 2008; Sugar, 2017) (Figure 1).

[1] Actually there are two methods to overcome OOV problems in neural machine translation in low-resource settings: 1) use different translation granularities (Nguyen and Chiang, 2017; Sennrich et al., 2016); 2) extend the vocabulary. Our experiments on OOV translation belong to the second.

Donor language	Source word	Uyghur word	English
Persian	افسوس	ئەپسۇس	pity
	گوشت	گۆش	meat
Arabic	ساعة	سائەت	hour
Russian	велосипед	ۋېلىسىپىت	bicycle
	доктор	دوختۇر	doctor
	поезд	پويىز	train
	область	ئوبلاست	oblast
	телевизор	تېلېۋىزور	television
Chinese	凉粉	لەڭپۇڭ	agar-agar jelly
	豆腐	دۇفۇ	tofu

Figure 1: Examples of loanwords in Uyghur.

3.2 Challenges in Loanword Identification

Intuitively, loanwords can be simply detected according to pronunciation similarity. However, the words may have several changes to adopt the recipient language:1) Changes in meaning: words are occasionally changed with a different meaning than that in the donor language; 2) Changes in spelling: although words taken into different recipient languages are sometimes spelled as in the donor language, sometimes borrowed words retain their original (or near-original) pronunciation, but undergo a spelling change to represent the orthography of the recipient language; 3) Changes in pronunciation: in cases where a new loanword has a very unusual sound, the pronunciation of the word can be radically changed.

All above challenges shown that the lexical borrowing is very complicated, so it is difficult to achieve expected results without further semantic information, especially some cross-lingual features.

4 Methodology

In this paper, we propose a novel method to identify loanwords in Uyghur. Inspired by state-of-the-art achievements of word embedding techniques in cross-lingual NLP tasks, we first train a cross-lingual embedding model based on comparable corpora (Uyghur-one donor language) and a small bilingual lexicon; then, given a Uyghur word, we obtain its most sematic related source words (and its semantic distance) in donor language based on cross-lingual embedding model; we set a threshold ρ for each donor language, if a Uyghur word has a semantic distance (the first source word of donor language) lower than this threshold, it will be removed. With the candidate list, we use a log-linear model with pronunciation similarity and hybrid language model features to predict the final results (Figure 2).

4.1 Problem Description

Assume we have a low resource language named $Lang_A$ (receipt language) and a high resource language named $Lang_B$ (donor language). The goal of loanword identification is to find out words w_i in the $Lang_A$ corpus which is originally borrowed from $Lang_B$. In this study, we mainly focus on loanwords that share the same or similar meaning with its corresponding word in the donor language.

4.2 Loanword Candidate List Generation

In this section, we first introduce cross-lingual word embedding model based on comparable data, which is used in our proposed approach as an important feature to represent the semantic relationships between words in donor language and receipt language; then, we present the generation of loanword candidate list.

Cross-lingual Word Embedding Model.

Cross-lingual embedding models learn cross-lingual representations of words in a joint embedding space. They enable knowledge transfer between languages, which is very important between resource-rich language and resource-poor language. Many previous studies on cross-lingual word embedding critically

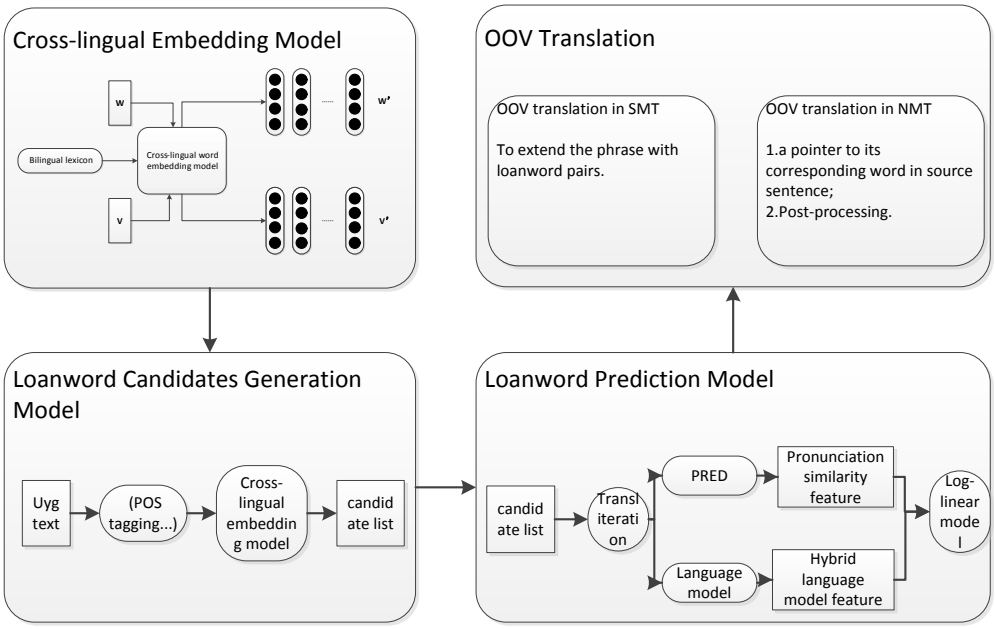


Figure 2: Framework of our proposed model.

require large sentence-aligned parallel data or dictionaries to induce bilingual word emebddings that closely aligned over languages in the same semantic space. However, these bilingual resources are very expensive for low-resource languages like Uyghur. Therefore, in this study we follow the work described by (Vulić and Moens, 2015) to obtain cross-lingual word embeddings with comparable data.

Let $W = \{w_1, w_2, ..., w_{|w|}\}$ be the vocabulary of a language $lang_A$ with $|W|$ words, and $\mathbf{W} \in \mathbb{R}^{|W| \times l}$ be the corresponding word embeddings of length l. Let $V = \{v_1, v_2, ..., v_{|V|}\}$ be the vocabulary of another language $lang_B$ with $|V|$ words, and $\mathbf{V} \in \mathbb{R}^{|V| \times m}$ the corresponding word embeddings of length m. Let d_u(Uyghur) and d_f (donor language) denote a pair of comparable documents with length in words p and q ($p > q$). The comparable data based cross-lingual embeddings (**CdbCLE**) method merges d_u and d_f as one single pseudo-bilingual document exploiting a deterministic strategy based on length ratio of two documents $R = \lfloor \frac{p}{q} \rfloor$ firstly. We sequentially pick word from d_f and put it into the merged document as the Rth word. Then, we train a skip-gram model on merged documents to generate word vectors for all words in $\mathbf{W} \cup \mathbf{V}$. The most important step is to instantiate CdbCLE algorithm. We use the similar objective function with word2vec skip-gram based on pseudo-lingual document described above.

The CdbCLE method can be described as:

$$C(\mathbf{W}, \mathbf{V}) = -\sum_{s \in \mathbf{W} \cup \mathbf{V}} \sum_{t \in NBR(s)} \log P(t|s) \tag{1}$$

where $NBR(s)$ is the context of s in pseudo-lingual document, $P(ts) \propto \exp(t^T s)$. Note that $t, s \in \mathbf{W} \cup \mathbf{V}$.

Candidate List Generation

In this part, we treat the loanword candidate list generation in Uyghur as a procedure to find out the most semantic similar source words in donor language corpora with current Uyghur word.

According to the CdbCLE algorithm described above, to identify loanwords in Uyghur, we must ensure that the current Uyghur word should share the same meaning with the corresponding source word in donor language. This can be indicates that the loanword and its corresponding word in the donor language share the same word embedding space.

To simplify this step and filter some unrelated words in Uyghur, we annotate the Uyghur texts with a part-of-speech tagger firstly. The goal of part-of-speech tagging is to assign to each word in a sentence

its morph-syntactic category. Since loanwords in Uyghur are mainly nouns, we can filter Uyghur word list with POS tags.

We use the CdbCLE algorithm to train a cross-lingual embedding model, the source part is the Uyghur language corpus, and the target parts are the donor language corpora (Chinese, Arabic, Persian and Russian). When testing, the pre-trained model gives more than one sematic similar words in donor languages based on the given Uyghur word.

$$\{u_i, [w_{i1} : v_{i1}, w_{i2} : v_{i2}, ..., w_{ik} : v_{ik}]\} \tag{2}$$

here, k is the length of returned source word list in donor language based on cross-lingual embeddings.

Based on the distance v_i between the embeddings of a Uyghur word u_i and the first source word w_{i1} returned by the CdbCLE, we discard the Uyghur word and source words item when the distance is little than a certain threshold ρ (this value is obtained through experiments)[2]. After that, the loanword candidate list is generated.

4.3 Loanword Prediction Model

Our cross-lingual word embedding based loanword identification approach generates some candidates which are similar semantically. To predict the final outputs, we propose a log-linear model which integrate several features into our approach.

Transliteration

The string similarity feature and hybrid language model features are all rely on writing system heavily. If the receipt and donor languages belong to different writing systems, we can' t compute these two features directly. In this study, we use a transliteration model to generate a character mapping table between Uyghur and donor languages (especially for language has a different writing system with Uyghur) which make sure the receipt and donor language belong to the same writing system.

The transliteration model used in this paper can be define as

$$t_{best} = arg\max_{t} \prod_{i=1}^{I} \phi(s_i|t_i) \prod_{i=1}^{|\mathbf{t}|} p_{LM}(t_i|t_1, t_2, ..., t_{i-1}) \tag{3}$$

this model is actually a phrase-based machine translation model without reordering part. Here, we first segment the training corpus into characters and learn a phrase-based translation system over character pairs (source part:$\{s_0, s_1, s_2, ..., s_{I-1}\}$, target part:$\{t_0, t_1, t_2, ..., t_{I-1}\}$. $\phi(s_i|t_i)$ indicates the phrase translation table, $p_{LM}(t_i|t_1, t_2, ..., t_{i-1})$ means the language model probabilities (built from the target-side if mined transliteration corpus). The transliteration model assumes that source and target characters are generated monotonically. Therefore, we dont use any reordering model.

Pronunciation Similarity Feature

Usually, a loanword has the similar pronunciation to its corresponding word in the donor language. A straightforward way to identify loanword is to compute pronunciation similarity between the current word and its corresponding loanword candidates in the donor language. In this study, we use the edit distance algorithm as a feature in our prediction model.

Edit distance (ED), also known as the Levenshtein distance, is a string metric for measuring the difference between two sequences. Informally, the edit distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other. However, if two words belong to different word formation systems (Uyghur and Chinese), the original ED algorithm does not perform well. For example, due to the suffixes of a Uyghur word, the number of deletions according to the ED algorithm is equal or even greater than the length of donor word. Intuition might suggest that stemming of a Uyghur word can avoid such problems. However, this approach depends heavily on the performance of stemming algorithm. We propose a position-related edit distance (PRED) method , which ignores deletions at the end of word b. That is,

[2]A word with the semantic similarity lower than a certain threshold is not a loanword or is a loanword changed its meaning; we cant apply this kind of word pairs in low-resource NMT.

$$PRED_{a,b}(i,|b|) = \min\{PRED_{a,b}(i-1,|b|), ED_{a,b}(i,|b|)\} \tag{4}$$

here, a is a Uyghur word and b is a donor word, ED is the traditional edit distance algorithm.

We can define the pronunciation similarity feature function as

$$f(u,t,V) = \begin{cases} \frac{\sum_{i=1}^{k}\psi_i PRED(u,v_i)}{\sum_{i=1}^{k} len(v_i)} & if \frac{\sum_{i=1}^{k}\psi_i PRED(u,v_i)}{\sum_{i=1}^{k} len(v_i)} \leq \frac{1}{2}len(u), \\ 0 & else. \end{cases} \tag{5}$$

here, V is a set of words have similar sematic space in donor language with one candidate loanword in Uyghur u, t is the possible tag of u, $len(v_i)$ indicates the length of word v_i, ψ_i is a parameter.

Hybrid Language Model Feature

When borrowing, a word in the donor language may adopt the receipt languages pronunciation system. Therefore, the pronunciation of a loanword belongs to two different pronunciation systems (receipt and donor). Each pronunciation system can be represented by a character-level language model. This inspired us to combine two language models of pronunciation system to simulate the pronunciation of loanwords.

$$p_{hlm}(c_1,c_2,...,c_l) = (1-\lambda)p_{rec}(c_1,c_2,...,c_l) + \lambda p_{don}(c_1,c_2,...,c_l) \tag{6}$$

where $\{c_1,c_2,...,c_l\}$ is a character sequence of a candidate loanword, p_{rec} is the character level language model probability of a given character sequence in receipt language, p_{don} is the character level language model probability of a given character sequence in the donor language. λ is the prior probability of donor language model.

Accordingly, we can formulize the hybrid language model feature as:

$$f(u,t) = \begin{cases} p_{don}(u), & if p_{don}(u) \geq p_{rec}(u) \\ p_{hlm}(u), & else. \end{cases} \tag{7}$$

Loanword Prediction Model

To integrate above two features together, we use a log-linear model. One important reason is that the log-linear model allows a very rich set of features to be used in a single model, arguably much richer representations than the other machine learning algorithms.

Assume we have a candidate loanword u, and a set of possible tags T (LW/O). LW means loanword and O means not loanword. The goal of our task is to model the conditional probability $p(t|u)$. So we can describe the prediction model as:

$$p(t|u;v) = \frac{exp(v \cdot f(u,t))}{\sum_{t' \in t} exp(v \cdot f(u,t'))} \tag{8}$$

where $f(.)$ indicates a feature function, which maps (u,t) pair to a feature vector $f(u,t)$, and **v** is a parameter vector. Note that the number of features and parameters should be the same.

We use the maximum-likelihood estimation (MLE) to estimate the parameters of the log-linear model. To overcome the overfitting, we follow the common solution that modifies the objective function to include a regularization term.

4.4 Applications in OOV Translation

In this study, we integrate the results of our proposed Uyghur loanword identification model in OOV translation in NMT with following two methods:

Training data extend: we extend the training data with loanword pairs (as parallel sentences) directly.

Bilingual dictionary extend: we first extend the bilingual dictionary with our loanword pairs; then, we follow (Luong et al., 2014) to annotate the training data with explicit information that to enables the NMT system to emit, for each OOV word, a pointer to its corresponding word in the source sentence. The information is later used in post-processing to translate the OOVs words using the extended bilingual dictionary.

Type	Name	Size(Sent./Tok.)		
		Train set	Dev set	Test set
BilCorp4MT	UyAr(MT)	0.10M/2.35M	1K/23K	1.5K/30K
	UyCn(MT)	0.25M/4.79M	1K/25K	1.5K/32K
	UyRu(MT)	0.09M/2.01M	1K/19K	1.5K/28K
	UyPr(MT)	0.15M/3.08M	1K/24K	1.5K/30K
CompCorp4Train	UygMono	22.50M/450.10M(UygChn), 26.30M/478.02M(UygAra), 19.84M/405.37M(UygRus), 23.96M/500.58M(UygPer).		
	ChnMono	29.76M/502.42M	N/A	N/A
	AraMono	25.50M/469.23M	N/A	N/A
	RusMono	20.96M/402.65M	N/A	N/A
	PerMono	23.19M/486.29M	N/A	N/A

Table 1: Statistic of Corpus.

5 Experiments

To evaluate the effectiveness of our proposed approach, we conducted two groups of experiments: loanword identification and OOV translation. In loanword identification experiments, we try to detect loanwords in Uyghur in four donor languages: Chinese, Persian, Russian, and Arabic; the details can be found in Table 2. In OOV translation experiments, we trained several NMT models on eight translation directions, all of them with serious OOV problems due to the limited training data sets. We integrated the results of the loanword identification experiments into OOV translation.

5.1 Datasets

We applied two types of data in our experiments: comparable and bilingual. The details of these corpora can be found in Table 1:

BilCorp4MT:This set includes bilingual corpora for training and developing NMT translation models. We collected these corpora from Wikipedia, websites, government documents and several other resources. Also, we generated bilingual lexicons on training data in this sets. The test set of the Uyghur part was annotated manually, which include loanword information in four donor languages. Therefore, we used the test set both for evaluation of loanword identification and NMT.

CompCorp4Train:We used these data sets to train word embedding models, and apply them in cross-lingual word embedding based loanword candidate list generation approach. We extract some loanword pairs (100) to train transliteration models.

5.2 Settings

Loanword identification experiments We compared our method with other three previous proposed models for loanword identification. Before that, we finished transliteration based on Moses[3] with default settings.

MutilShlFeats:(Multiple shallow features based model): which was proposed by (Mi et al., 2014), which use pronunciation similarity, common substring to build a loanword detection model in Uyghur.

NNBased: (Neural Network based model): This method was described by (Mi et al., 2016), which relied on pronunciation similarity and a small annotated corpus to build a loanword identification model.

Ours: We applied the open source toolkit word2vec[4] to train our cross-lingual word embeddings. We set the window size as 5 and negative sampling parameter as 5. We merged the documents based on method described in section 4.2. We implemented the PRED and language model tools by ourselves.

[3]https://github.com/dav/word2vec.

[4]http://www.statmt.org/moses/.

Donor Language	Model	Results (%)		
		Precision	Recall	F1-value
Arabic	MutilShlFeats	77.45	72.59	74.94
	NNBased	79.31	73.52	76.31
	Ours	**80.82**	**74.29**	**77.42**
Chinese	MutilShlFeats	75.20	70.45	72.75
	NNBased	77.09	71.13	73.99
	Ours	**79.33**	**73.56**	**76.34**
Persian	MutilShlFeats	78.26	72.97	75.52
	NNBased	79.50	73.95	76.62
	Ours	**80.77**	**74.60**	**77.56**
Russian	MutilShlFeats	76.45	70.57	73.39
	NNBased	78.65	70.94	74.60
	Ours	**79.90**	**71.34**	**75.38**

Table 2: Evaluation on different loanword identification methods in four donor languages (ρ = 0.70).

Lang	Precision									
	ρ**=0.50**	**0.55**	**0.60**	**0.65**	**0.70**	**0.75**	**0.80**	**0.85**	**0.90**	**0.95**
Arabic	0.52	0.64	0.69	0.76	**0.80**	0.58	0.53	0.49	0.41	0.36
Chinese	0.58	**0.79**	0.75	0.72	0.67	0.63	0.56	0.54	0.40	0.38
Persian	0.43	0.52	0.65	0.74	**0.80**	0.72	0.62	0.48	0.32	0.24
Russian	0.49	0.67	0.70	0.71	**0.79**	0.64	0.55	0.51	0.45	0.39

Table 3: Evaluation the affection of threshold value ρ (in **CdbCLE** model) on the precision of Uyghur loanword identification model.

The differences between our proposed model and previous studies are: 1) we used a transliteration model to obtain character mapping table for pronunciation similarity computation between two writing systems and a fixed mapping rule table was used in previous work; 2) previous loanword identification models mainly relied on large-scale annotated corpora and some rules. Our model based on a cross-lingual embedding model and loanword candidates were extracted from monolingual corpora.

OOV translation experiments

We conducted OOV translation experiments on NMT. Based on loanword identification results, we built loanword dictionaries for Uyghur-Chinese, Uyghur-Arabic, Uyghur-Russian and Uyghur-Persian. We extended bilingual corpus with these dictionaries.

NMT:We conducted NMT experiments using the open source Nematus toolkit[5]. Nematus has been used to build top-performing submissions to shared translation tasks at WMT and IWSLT. In this study, we use the default settings: the word embedding dimension was 620 and the size of a hidden layer was 1000, the vocabulary size was 30K, the batch size was 80, the maximum sequence length was 50, and the beam size for decoding was 10. Default dropout was applied. Each NMT model was trained for 80 batches by using AdaDelta optimizer (Zeiler, 2012).

5.3 Results and Analysis

Corpora	Size (word pair)			
	Uyghur-Arabic	Uyghur-Chinese	Uyghur-Persian	Uyghur-Russian
Dictionary	62,820	77,592	54,902	80,367
Dictionary (extend)	**63,652** (+832)	**78,342** (+750)	**55,408**(+506)	**81,065**(+698)

Table 4: Size of bilingual dictionaries (before and after extend).

[5]https://github.com/EdinburghNLP/nematus.

Language pair	BLEU			OOV rate		
	sys1	**sys2**	**sys3**	**sys1**	**sys2**	**sys3**
Uyghur-Russian	15.94	16.71	**16.93**	18.25	17.08	**16.94**
Uyghur-Persian	17.88	18.25	**18.47**	18.63	17.27	**17.10**
Uyghur-Chinese	22.50	22.94	**23.15**	18.05	17.48	**17.29**
Uyghur-Arabic	19.79	20.03	**20.31**	18.69	18.25	**17.84**
Russian-Uyghur	13.51	13.84	**14.07**	19.07	18.69	**18.52**
Persian-Uyghur	16.14	16.59	**16.80**	18.92	18.73	**18.56**
Arabic-Uyghur	17.82	18.23	**18.52**	19.97	19.45	**19.29**
Chinese-Uyghur	19.26	19.85	**20.03**	19.32	18.74	**18.61**

Table 5: Experiment results on OOV translation of different language pairs. **sys1** indicates the BPE based model (baseline), **sys2** means BPE based model with loanword pairs in training data, **sys3** means BPE based model with loanword pairs in post-processing step (Section 4.4).

In Table 2, we present loanword identification results in Uyghur. We find that across the four donor languages, our proposed method achieves the best performance compares with other three models. One important reason is that our method combines both semantic similarity (cross lingual word embedding) and pronunciation similarity (edit distance and language model). Among different donor languages, experimental results show that Persian loanwords in Uyghur can be identified more effectively compared with Arabic, Chinese and Russian. This is because Uyghur and Persian share much more words and has the most similar word formation. The loanword identification of Arabic also performs well compare with Russian and Arabic, the most possible reason is that many loanwords entered the language directly through Islamic literature. Although many loanwords of Europe origin have reached Uyghur through Russia's influence, the results of Russian loanword identification cannot outperform others. We believe that relatively fewer corpora lead to this situation. Due to the significant different word formation, the Chinese loanword detection achieves the worst results.

We find that the precision of our proposed model rely on threshold value heavily (Table 3). There are two possible reasons lead to this situation: size of data and diversity of languages. Since we have different size of data for each language pairs and the performance of cross-lingual embedding model usually based on the data, our model achieves best precision of four donor languages in different thresholds (0.55 and 0.70). Another reason is diversity of languages. We can easily find that Arabic, Persian and Russian are all achieves best results when ρ is **0.70**, only Chinese achieves its best precision score when ρ is **0.55**. After analysis, we know that Russian, Arabic and Persian have the similar word formation system with Uyghur, which is very different from Chinese.

We evaluate translation quality both with the BLEU score and OOV rate in NMT experiments (Table 5). We find that in all NMT tasks, models optimized by our loanword identification experiments received significant improvements compared with baseline methods. After extending the bilingual corpus by our loanword identification results, the OOV rates decrease significantly. Since BPE has achieved many state-of-the-art results in recent studies, we choose the BPE based model as the baseline (**sys1**). Due to the extend size of data sets, systems with loanword pairs are all outperform the baseline model, but with different BLEU improvements. For **sys2** (BPE+loanwordpairs+train), we use the loanword pairs in model training as other parallel sentences. Therefore, some OOV may not translate due to data spareness. **sys3** overcome this problem through translate OOV words in post-processing step with extended dictionaries Table 4. So the sys3 achieves best performance among all three translation systems in all language pairs.

6 Conclusion

In this study, we try to identify loanwords in low-resource languages (Uyghur) to extend bilingual corpus. In our proposed method, we first use the cross-lingual word embeddings as the core feature to generate semantically related candidates based on large comparable corpora and a small bilingual lexicon; then,

a log-linear model which combines several shallow features is explored to predict the final results. Results of loanword identification suggest that our proposed model outperforms other baseline methods significantly. Also, we conduct experiments on OOV translation in NMT. Experimental results show that loanword pairs can help reduce OOV rates in several low-resource language pairs.

In future work, we plan to import transfer learning into our model to further improve the performance.

Acknowledgements

We are grateful to the mentor of this paper for his meaningful feedback. Thanks Rui Wang and Kehai Chen for helpful discussions and three anonymous reviewers for their insightful comments and suggestions. This work is supported by the West Light Foundation of The Chinese Academy of Sciences under Grant No.2015-XBQN-B-10, Xinjiang Science and Technology Major Project under Grant No.2016A03007-3 and National Natural Science Foundation of China (NSFC) under Grant No.U1703133.

References

Oliver Adams,Adam Makarucha,Graham Neubig,Steven Bird and Trevor Cohn. 2017. Cross-lingual word embeddings for low-resource language modeling. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics,* pages 937–947, Valencia, Spain.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, Noah A. Smith. 2016. Massively multilingual word embeddings. *arXiv 2016.*

Long Duong,Hiroshi Kanayama,Tengfei Ma,Steven Bird and Trevor Cohn. 2017. Multilingual Training of Crosslingual Word Embeddings. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics,* pages 894–904, Valencia, Spain.

Meng Fang and Trevor Cohn. 2017. Model transfer for tagging low-resource languages using a bilingual dictionary. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics,* pages 587–593, Vancouver, Canada.

Stephan Gouws, Yoshua Bengio and Greg Corrado. 2015. BilBOWA: Fast Bilingual Distributed Representations without Word Alignments. *Proceedings of the 32nd International Conference on Machine Learning,* pages 748–756, Lille, France.

Bates L. Hoffer. 2005. Language borrowing and the indices of adaptability and receptivity. *Intercultural communication studies*, 14(2):53–72.

Yoonjung Kang, Andrea H Phạm, and Benjamin Storme. 2014. French loanwords in Vietnamese: the role of input language phonotactics and contrast in loanword adaptation. *Proceedings of the Annual Meetings on Phonology,* MIT.

Alexandre Klementiev, Ivan Titov and Binod Bhattarai. 2012. Inducing Crosslingual Distributed Representations of Words. *Proceedings of COLING 2012,* pages 1459–1474, Mumbai, India.

Nicholas Kontovas. 2008. An analysis of recent loans into the Standard Uyghur lexicon. *University of Chicago,* Chicago.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals and Wojciech Zaremba. 2014. Addressing the Rare Word Problem in Neural Machine Translation. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing,* pages 11–19, Beijing, China.

Chenggang Mi, Yating Yang, Lei Wang, Xiao Li and Kamali Dalielihan. 2014. Detection of Loan Words in Uyghur Texts. *Proceedings of the 3rd International Conference on Natural Language Processing and Chinese Computing,* pages 103–112, Shenzhen, China.

Chenggang Mi, Yating Yang, Xi Zhou, Lei Wang, Xiao Li and Tonghai Jiang. 2016. Recurrent Neural Network Based Loanwords Identification in Uyghur. *Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation,* pages 209–217, Seoul, Korea.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems 26,* pages 3111–3119, Lake Tahoe, California.

Toan Q. Nguyen and David Chiang. 2017. Transfer Learning across Low-Resource, Related Languages for Neural Machine Translation. *Proceedings of the Eighth International Joint Conference on Natural Language Processing,* pages 296–301, Taipei, Taiwan.

Sharon Peperkamp. 2004. A psycholinguistic theory of loanword adaptations. *Proceedings of the 30th Annual Meeting of the Berkeley Linguistics Society,* pages 341–352, Berkeley, CA.

Sebastian Ruder, Ivan Vulić and Anders Søgaard. 2017. A Survey Of Cross-lingual Word Embedding Models. *arXiv 2017.*

Rico Sennrich, Barry Haddow and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics,* pages 1715–1725, Berlin, Germany.

Shigeko Shinohara. 2015. Loanword-specific grammar in Japanese adaptations of Korean words and phrases. *Journal of East Asian Linguistics*, 24(2): 149-191.

Alexander Sugar. 2017. Mandarin Chinese Verbs as Verbal Items in Uyghur Mixed Verbs. *Languages*, 2(1), 1.

Yulia Tsvetkov, Waleed Ammar and Chris Dyer. 2015. Constraint-Based Models of Lexical Borrowing. *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* pages 598–608, Denver, Colorado.

Yulia Tsvetkov, Sunayana Sitaram, Manaal Faruqui,Guillaume Lample,Patrick Littell,David Mortensen,Alan W Black,Lori Levin and Chris Dyer. 2016. Polyglot Neural Language Models: A Case Study in Cross-Lingual Phonetic Representation Learning. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* pages 1357–1366, San Diego, California.

Yulia Tsvetkov and Chris Dyer. 2016. Cross-lingual bridges with models of lexical borrowing . *Journal of Artificial Intelligence Research*, 55(1): 63-93.

Yulia Tsvetkov and Chris Dyer. 2016. Lexicon Stratification for Translating Out-of-Vocabulary Words. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing,* pages 125–131, Beijing, China.

Ivan Vulić and Marie F Moens. 2015. Bilingual Word Embeddings from Non-Parallel Document-Aligned Data. Applied to Bilingual Lexicon Induction.. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing,* pages 719–725, Beijing, China.

Liangchen Wei and Zhihong Deng. 2017. A Variational Autoencoding Approach for Inducing Cross-lingual Word Embeddings. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence,* pages 4165–4171, Melbourne, Australia.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *arXiv 2012.*

Adaptive Weighting for Neural Machine Translation

Yachao Li[1,2], **Junhui Li**[1], **Min Zhang**[1,*]
[1]Institute of Artificial Intelligence,
School of Computer Science and Technology, Soochow University, China
[2]Key Laboratory of China's Ethnic Languages and Information Technology
of Ministry of Education, Northwest Minzu University, China
liyc7711@gmail.com, {lijunhui, minzhang}@suda.edu.cn

Abstract

In the popular sequence to sequence (seq2seq) neural machine translation (NMT), there exist many weighted sum models (WSMs), each of which takes a set of input and generates one output. However, the weights in a WSM are independent of each other and fixed for all inputs, suggesting that by ignoring different needs of inputs, the WSM lacks effective control on the influence of each input. In this paper, we propose adaptive weighting for WSMs to control the contribution of each input. Specifically, we apply adaptive weighting for both GRU and the output state in NMT. Experimentation on Chinese-to-English translation and English-to-German translation demonstrates that the proposed adaptive weighting is able to much improve translation accuracy by achieving significant improvement of 1.49 and 0.92 BLEU points for the two translation tasks. Moreover, we discuss in-depth on what type of information is encoded in the encoder and how information influences the generation of target words in the decoder.

1 Introduction

Recent advances in neural machine translation (NMT) have achieved remarkable success over the state-of-the-art of statistical machine translation (SMT) on various language pairs (Bahdanau et al., 2015; Jean et al., 2015; Luong et al., 2015; Wu et al., 2016; Vaswani et al., 2017). In the neural networks of seq2seq models, either RNN-based (Bahdanau et al., 2015), CNN-based (Gehring et al., 2017), or full attention-based (Vaswani et al., 2017), there exist many scenarios in both encoder and decoder where a weighted sum model (WSM) takes a set of inputs and generate one output. As shown in Eq. 1, the WSM first combines k inputs $(x_1, \cdots, x_k)$ with k respective weights $(w_1, \cdots, w_k)$ and then non-linearizes it through an activation function f, such as $\tanh$, *sigmoid function*, $ReLU$, and so on. In this paper we omit bias terms to make the equations less cluttered.

$$o = f\left(\sum_{i=1}^{k} w_i x_i\right) \tag{1}$$

Note that the above weights $(w_1, \cdots, w_k)$ are independent of each other and once the model is tuned, the weights are fixed for all inputs, suggesting that by ignoring different needs of the inputs, the WSM lacks effective control on the influence of each input.

Let us take a concrete scenario in seq2seq model as an example. Figure 1(a) shows a typical illustration of generation of t-th target word where the decoder takes three inputs, i.e., source context c_t, previous target word y_{t-1} and current target state s_t while generating one output y_t via output state o_t. However, the study in Tu et al. (2017) suggests that different target words require inconsistent contributions from source context (c_t) and target context (i.e., y_{t-1} and s_t). For example, to generate translation *Xinhua News Agency , Hong Kong*, the first word y_1 *xinhua* is highly related to its source context c_1 while the

*Min Zhang is Corresponding Author.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3038–3048
Santa Fe, New Mexico, USA, August 20-26, 2018.

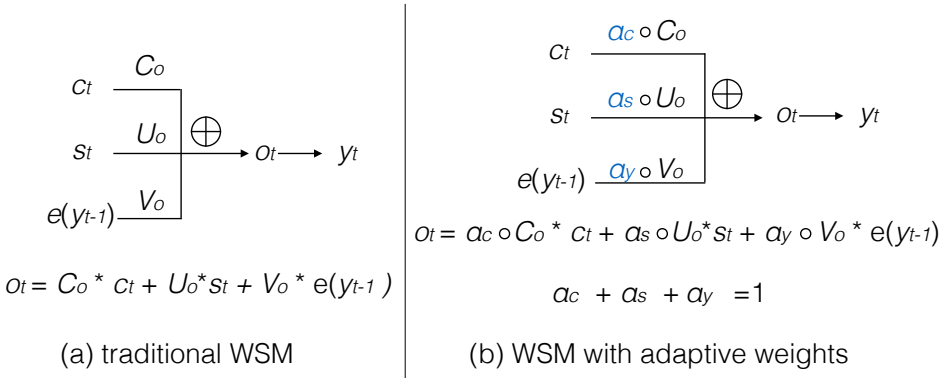


Figure 1: Illustration of generation of t-th translation word.

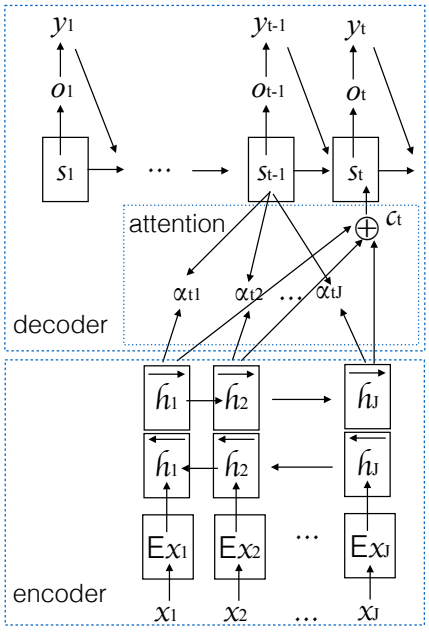


Figure 2: NMT model with attention mechanism.

second and third words y_2 *news* and y_3 *agency* are mainly influenced by target context due to the well-formed saying *Xinhua News Agency*. Similarly, y_5 *Hong* and y_6 *Kong* are mainly influenced by source context and target context, respectively.

In this paper, we propose adaptive weighting to dynamically control the contribution of each input in a WSM that has a set of inputs. Unlike the conventional weights that are independent of each other, adaptive weights are learned to be dependent on each other and more importantly be able to dynamically select the amount of input information. Specifically, we use gate mechanism to incorporate adaptive weights in GRU and in computing the output states. Experimentation on both Chinese-to-English and English-to-German translation tasks demonstrates that NMT systems with adaptive weighting are able to much improve the translation accuracy. Moreover, through adaptive weights we discuss in-depth on what type of information encoded in the encoder and how information influences the generation of a target word.

2 NMT with Attention Mechanism

In this section, we review NMT model with attention mechanism. Encoder-decoder model with attention mechanism is one of the most popular frameworks for NMT (Bahdanau et al., 2015), which consists of

an encoder and a decoder, as shown in Figure 2. In the following, m refers to the embedding size of both source and target sides, and n the hidden state size of the two sides. E_{x_j} returns the word embedding for source word x_j while $e(y_t)$ returns the word embedding for target word y_t.

Encoder: Given a source sentence $\mathbf{x} = (x_1, \cdots, x_J)$, the encoder first converts each word x_j into a real-valued m-dimensional vector E_{x_j} via the embedding matrix $E \in \mathbb{R}^{m \times |V_s|}$, where V_s is the source-side vocabulary. Then the encoder uses bidirectional RNN: the forward RNN reads the input sequence from left to right and outputs a forward sequence of hidden states $(\overrightarrow{h_1}, \cdots, \overrightarrow{h_J})$ by $\overrightarrow{h_j} = RNN(\overrightarrow{h_{j-1}}, E_{x_j})$; likewise the backward RNN operates from right to left and outputs a backward sequence $(\overleftarrow{h_1}, \cdots, \overleftarrow{h_J})$. Each source word x_j is represented as h_j (also referred to as word annotation vector): the concatenation of hidden states $\overrightarrow{h_j}$ and $\overleftarrow{h_j}$. Such bidirectional RNN encodes not only the word itself but also its left and right context, which can provide important evidence for its translation.

Decoder: The decoder is also an RNN that predicts a target sequence $\mathbf{y} = (y_1, \cdots, y_T)$. It defines a probability over the translation $\mathbf{y}$ by decomposing the joint probability into the ordered conditionals:

$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t | y_1, \cdots, y_{t-1}, \mathbf{x}) \tag{2}$$

$$p(y_t | y_1, \cdots, y_{t-1}, \mathbf{x}) = softmax\left(\mathbf{W}_o f(o_t)\right) \tag{3}$$

where $\mathbf{W}_o \in \mathbb{R}^{|V_t| \times m}$ is a weight matrix, and V_t is the target-side vocabulary. o_t is the **output state**, defined as:

$$o_t = \mathbf{U}_o s_t + \mathbf{V}_o e(y_{t-1}) + \mathbf{C}_o c_t \tag{4}$$

$$s_t = RNN(s_{t-1}, e(y_{t-1}), c_t) \tag{5}$$

where $\mathbf{U}_o \in \mathbb{R}^{n \times n}$, $\mathbf{V}_o \in \mathbb{R}^{n \times m}$, $\mathbf{C}_o \in \mathbb{R}^{n \times 2n}$ are weight matrices. s_t is the hidden state of the decoder. c_t is the context vector, which is calculated as the summation vector weighted by a_{tj}:

$$c_t = \sum_{j=1}^{J} a_{tj} h_j \tag{6}$$

where a_{tj} is the weight of h_j, defined as

$$a_{tj} = \frac{exp(e_{tj})}{\sum_{k=1}^{J} exp(e_{tk})} \tag{7}$$

where $e_{tj} = a(s_{t-1}, h_j)$ is an alignment model, measuring the degree of matching between the *j*-th source hidden unit h_j and the *t*-th target hidden unit s_t.

Training: The whole model is jointly trained to maximize the conditional log-likelihood of the training data containing I sentence pairs.

$$L(\Theta) = \max_{\Theta} \frac{1}{I} \sum_{i=1}^{I} \log p_\Theta(\mathbf{y}_i | \mathbf{x}_i) \tag{8}$$

where Θ is the set of all parameters, and $(\mathbf{x}_i, \mathbf{y}_i)$ is the i-th sentence pair in the training set.

3 Adaptive Weighting for NMT

In this section, we propose adaptive weighting for NMT. Our goal is to enable a WSM dynamically selects the amount of input information when there exist two or more inputs. Specifically, we apply the above idea into the attention-based seq2seq model from two perspectives: (1) adaptive weighting for GRU; and (2) adaptive weighting for the output state o_t as in Figure 1.

3.1 Gated Recurrent Unit

Gated recurrent unit (GRU) (Cho et al., 2014a; Cho et al., 2014b) is a type of hidden unit of RNN which makes each recurrent unit to adaptively capture dependencies of different time scales. A GRU has two gates called reset gate r_t and update gate z_t, which control the amount of information will "flow through" the network. r_t and z_t are computed as follows:

$$r_t = \sigma(\mathbf{W}_r Ex_t + \mathbf{U}_r h_{t-1}) \tag{9}$$

$$z_t = \sigma(\mathbf{W}_z Ex_t + \mathbf{U}_z h_{t-1}) \tag{10}$$

where $\sigma(\cdot)$ is a logistic sigmoid function. Ex_t and h_{t-1} are the input and the previous hidden state. $\mathbf{W}_r, \mathbf{W}_z \in \mathbb{R}^{n\times m}$, $\mathbf{U}_r, \mathbf{U}_z \in \mathbb{R}^{n\times n}$ are model parameters which are learned. The new hidden unit h_t is computed by

$$\tilde{h_t} = tanh(\mathbf{W}Ex_t + \mathbf{U}(r_t \circ h_{t-1})) \tag{11}$$

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h_t} \tag{12}$$

where $tanh(\cdot)$ is a hyperbolic tangent activation function, and $\circ$ is an element-wise multiplication. $\mathbf{W} \in \mathbb{R}^{n\times m}$, $\mathbf{U} \in \mathbb{R}^{n\times n}$ are model parameters.

3.2 Adaptive Weighting for GRU

As shown in Section 3.1, recurrent unit is critical in GRU. Both the reset gate r_t and the update gate z_t control the amount of previous state h_{t-1} and current input Ex_t being carried over to the current hidden state h_t. However, the gate itself either r_t or z_t, is uniformly set via fixed weights $\mathbf{W}_r$ (or $\mathbf{W}_z$) and $\mathbf{U}_r$ (or $\mathbf{U}_z$) for all inputs of Ex_t and all previous states of h_{t-1}. To dynamically select the amount of Ex_t and h_{t-1}, we define a hyper-gate g_t to control the flow of information between Ex_t and h_{t-1} explicitly, as shown in Eq. 13.

$$g_t = \sigma(\mathbf{W}_g Ex_t + \mathbf{U}_g h_{t-1}) \tag{13}$$

where $\mathbf{W}_g \in \mathbb{R}^{n\times m}$ and $\mathbf{U}_g \in \mathbb{R}^{n\times n}$ are parameters to be learned. Then we add g_t to the standard GRU to control how much information from previous hidden state is preserved at current time step explicitly. Accordingly, we update Eq. 9 $\sim$ 12 with the following equations.

$$r_t = \sigma((1 - g_t) \circ \mathbf{W}_r Ex_t + g_t \circ \mathbf{U}_r h_{t-1}) \tag{14}$$

$$z_t = \sigma((1 - g_t) \circ \mathbf{W}_z Ex_t + g_t \circ \mathbf{U}_z h_{t-1}) \tag{15}$$

$$\tilde{h_t} = tanh((1 - g_t) \circ \mathbf{W}Ex_t + g_t \circ \mathbf{U}(r_t \circ h_{t-1})) \tag{16}$$

$$h_t = g_t \circ z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h_t} \tag{17}$$

As shown in Eq.14, for example, the input Ex_t and the previous state h_{t-1} are now controlled by weights $(1-g_t)\circ\mathbf{W}_r$ and $g_t\circ\mathbf{U}_r$, respectively, which are dependent on each other and further controlled by the hyper-gate g_t.

Note that the NMT model adopts GRU in both encoder and decoder to generate the source side hidden states (i.e., h_t) and the target side hidden states (i.e., s_t), respectively. Therefore, we incorporate adaptive weights for GRUs in both encoder and decoder.

3.3 Adaptive Weighting for Output State

As shown in Figure 1 (b), the decoder iteratively takes three inputs, i.e., source context c_t, previous target word y_{t-1} and current target state s_t and generates one output y_t via intermediate state o_t. In order to dynamically select the amount of c_t, y_{t-1}, and s_t, we update Eq. 4 with the following:

$$o_t = \alpha_s \circ \mathbf{U}_o s_t + \alpha_y \circ \mathbf{V}_o e(y_{t-1}) + \alpha_c \circ \mathbf{C}_o c_t \tag{18}$$

where α_s, α_y, α_c can be either scalars or vectors. Together with $\mathbf{U}_o$, $\mathbf{V}_o$ and $\mathbf{C}_o$, they control the amount of s_t, y_{t-1} and c_t being carried forward, respectively. In this work, we define them as vectors, and α_s, for example, is computed as:

$$\alpha_s = \frac{exp(e_s)}{exp(e_s) + exp(e_y) + exp(e_c)} \tag{19}$$

$$e_s = f(\tilde{o}_t, s_t) \tag{20}$$

$$\tilde{o}_t = \mathbf{U}_c s_t + \mathbf{V}_c e(y_{t-1}) + \mathbf{C}_c c_t \tag{21}$$

where f is a feed-forward neural network. $\mathbf{U}_c \in \mathbb{R}^{n \times n}$, $\mathbf{V}_c \in \mathbb{R}^{n \times m}$, and $\mathbf{C}_c \in \mathbb{R}^{n \times 2n}$ are parameters to be learned. Similarly, e_y and e_c can be computed, and consequently α_y and α_c.

4 Experiments

To test our approach, we carry out experiments on the tasks of Chinese-to-English (ZH-EN) and English-to-German (EN-DE) machine translations. All source code is available on github. [1]

4.1 Dataset and Evaluation Metrics

ZH-EN Translation. Our training data for ZH-EN translation consists of 1.25M sentence pairs extracted from LDC corpora, with 27.9M Chinese words and 34.5M English words respectively. NIST MT 06 (1664 sentence pairs) is chosen as the development set while NIST MT 02, 03, 04, 05, and 08 datasets (878, 919, 1788, 1082 and 1357 sentence pairs, respectively) are used as our test sets. We use the case-insensitive 4-gram NIST BLEU score (Papineni et al., 2002) for validation and evaluation, measured by mteval-v11b.pl script.

EN-DE Translation. The EN-DE training data is provided by the standard benchmark ACL WMT 2017 [2], which consists of 5.85M sentence pairs with 141.42M English words and 134.83M German words respectively. We combine news-test-2012 and news-test-2013 as development set (6003 sentence pairs), and use news-test-2014 (News14), news-test-2015 (News15) and news-test-2016 (News16) as test sets (3003, 2169, and 2999 sentence pairs, respectively). Following Barone et al. (2017), we use the validation cross-entropy to choose the best model on the development set and use the case-sensitive 4-gram BLEU score for evaluation on test sets, measured by multi-bleu.perl script.

4.2 Training Details and Systems

We train each model with sentences of length up to 50 words for ZH-EN and 60 words (tokens) for EN-DE. The source and target word embedding dimension is 620. The size of the hidden layer is 1000. We use Adam (Kingma and Ba, 2014) to optimize model parameters with a learning rate of 0.0002, and the mini-batch size of 80.

For efficient training the neural networks, in ZH-EN translation we limit the source and target vocabularies to the most frequent 30K words, covering approximately 97.7% and 99.3% of the data in the two languages respectively. All out of vocabulary words are mapped to a special token *UNK*. For EN-DE translation, we apply byte-pair encoding (BPE) [3] (Sennrich et al., 2016) for better handling unknown words and set the vocabulary size as 30K.

We compare the performance of the following NMT systems:

- baseNMT: We use the open source toolkit dl4mt as our baseline attention-based NMT system (Bahdanau et al., 2015) [4] with most default parameter settings kept the same. For translation, a beam search with size 10 is employed.
- +Adaptive GRU: On baseNMT, we leverage adaptive weighting for GRU, as described in Section 3.2.

[1] https://github.com/liyc7711/weighted-nmt
[2] http://data.statmt.org/wmt17/translation-task/preprocessed/de-en/
[3] https://github.com/rsennrich/subword-nmt
[4] https://github.com/nyu-dl/dl4mt-tutorial

- +Adaptive Output: On baseNMT, we leverage adaptive weighting for the output state, as described in Section 3.3.
- +Both: On baseNMT, we leverage adaptive weighting for both GRU and the output state.

System	MT06	MT02	MT03	MT04	MT05	MT08	All	Params (M)
baseNMT	35.29	39.28	36.69	38.68	36.13	25.69	35.46	89.7
+Adaptive GRU	36.29‡	40.14†	36.99	39.96‡	36.91†	**27.15**‡	36.62‡	97.5
+Adaptive Output	35.72	40.38‡	37.29	39.31†	36.47	25.98	35.93‡	93.1
+Both	**36.52**‡	**40.87**‡	**38.04**‡	**40.41**‡	**37.42**‡	27.07‡	**36.95**‡	100.9

Table 1: BLEU scores of ZH-EN translation. †/‡: significant over baseline at 0.05/ 0.01, tested by bootstrap resampling (Koehn, 2004).

4.3 Experimental Results: ZH-EN Translation

Table 1 shows the performance of ZH-EN translation measured in BLEU score. From the table, we have the following observations.

- NMT models are benefited from adaptive weighting for either GRU or the output state. Moreover, the system of Adaptive GRU outperforms the system of Adaptive Output (i.e., 36.62 vs. 35.93 on all test sets).
- Fortunately, the improvements from adaptive weighting for GRU and the output state have little overlap. Combining the two types of adaptive weighting leads to more improvement on all test sets with the only exception of MT 08. Compared to baseNMT, the system *+Both* yields significant improvement of 1.49 BLEU scores, suggesting the positive effect of dynamically controlling the amount of input information being carried forward.

4.4 Experimental Results: EN-DE Translation

The results on English-German translation are presented in Table 2. It shows that leveraging adaptive weighting for GRU and the output state leads to significant improvement of 0.92 BLEU scores over all test sets. Overall, the performance trend over the proposed systems is consistent with that of ZH-EN translation.

System	News14	News15	News16	All
baseNMT	23.61	25.22	28.79	25.97
+Adaptive GRU	24.26‡	25.71†	29.63‡	26.53 ‡
+Adaptive Output	23.65	25.33	29.34†	26.11
+Both	**24.26**‡	**26.15**‡	**29.98**‡	**26.89**‡

Table 2: BLEU scores of EN-DE translation.

4.5 Parameters and Training Speed

As shown in Table 1, the proposed models introduce new parameters in different ways. [5] The baseline system has 89.8M parameters. The Adaptive GRU system introduces additional 7.8M parameters. The Adaptive output system introduces additional 3.4M parameters.

When running on a single GPU GeForce GTX 1080, the baseline model speeds 1.1 second per batch with 150K updates for ZH-EN, and 600K updates for EN-DE tasks, while the improved system (+Both) only increases the training time by about 12%.

[5]The parameters of the systems for EN-DE translation tasks are same as those for ZH-EN.

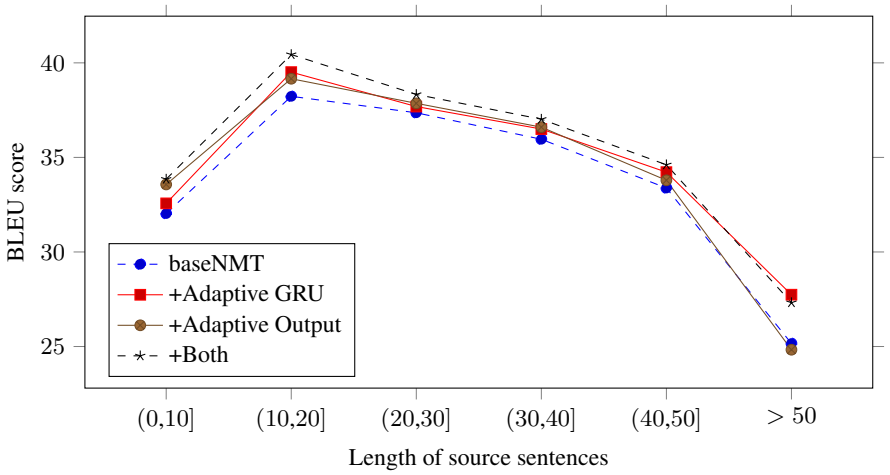


Figure 3: BLEU scores of translation with respect to the lengths of the input sentences.

5 Discussion

In this section, we further look at the *+Both* system and the *baseNMT* system to explore more on how adaptive weighting help in ZH-EN translation.

5.1 Analysis on Word Alignment

In the encoder, the Adaptive GRU uses adaptive weights to control the information flow of previous state and the current input. Thus, it has a direct impact on word representation vectors (i.e., hidden states) of source words. We conjecture that better word representation will help the decoder be able to attend to appropriate source words in decoding. To test this hypothesis, we carry out experiments of the word alignment task on the evaluation dataset from Liu and Sun (2015), which contains 900 manually aligned Chinese-English sentence pairs. We force the decoder to output reference translations, as to get automatic alignments between input sentences and their reference translations. To evaluate alignment performance, we report the alignment error rate (AER) (Och and Ney, 2003) and the soft AER (SAER) (Tu et al., 2016) in Table 3. It shows that adaptive weighting improves the attention model.

System	AER	SAER
baseNMT	43.0	55.7
+Both	**40.9**	**54.6**

Table 3: Evaluation of word alignment for ZH-EN translation. The lower the AER or SAER, the better the alignment quality.

5.2 Effects on Long Sentences

Following Bahdanau et al. (2015), we group sentences of similar lengths together and compute BLEU scores. Figure 3 presents the BLEU scores over different lengths of input sentences. It shows that systems with adaptive weighting consistently outperform baseNMT on all sentence lengths. It also shows that the performance drops substantially when the length of input sentences increases from 20. This performance trend over the length is consistent with the translation output in (Wang et al., 2017; Tu et al., 2016; Li et al., 2017).

5.3 Quantitative Analysis

Due to the continuous representations and non-linearity of neural networks. It is difficult to interpret the internal workings of NMT model (Ding et al., 2017). Fortunately, adaptive weights provide a mechanism to analyze what inputs of a WSM are more important than others. Next, we make insights on what type of

information encoded in the encoder and what types of information has a greater impact on the generation of a target word.

	Chinese POS	**Frequency**	g_t**-F (%)**	g_t**-B (%)**
	All	38,349	41.09	50.20
Content words	verb	8,176	37.84	46.96
	adjective	1,345	38.28	51.50
	adverb	2,104	38.40	50.93
	geographical name	1,537	36.14	50.70
	temporal noun	891	36.85	45.47
	general noun	7,636	39.15	45.57
	person name	587	39.90	44.56
Function words	preposition	1,553	43.59	53.84
	punctuation	4,956	47.91	52.07
	pronoun	1,449	43.01	54.55
	conjunction	1,015	46.27	43.20
	auxiliary	2,891	52.13	57.28

Table 4: Average g_t values groped by part of speech (POS) tags on development set MT06. g_t-F and g_t-B indicate the average g_t values in the forward GRU and the backward GRU, respectively.

Words	α_s **(%)**	α_c **(%)**	α_y**(%)**
All	60.61	29.06	10.34
EOS	72.02	22.40	5.58
.	72.10	20.20	7.7
,	70.00	22.84	7.16
of	66.34	21.34	12.32
to	65.45	22.58	11.96
on	65.72	23.02	11.25
is	71.67	22.32	6.00
has	68.05	26.12	5.84
US	55.42	36.08	8.50
China	56.36	35.01	8.63
president	57.00	34.49	8.52
Kong	42.94	19.30	37.77
York	42.14	26.55	31.31
agency	54.11	11.10	34.79

Table 5: Averaged α_s, α_c and α_y scores in computing the intermediate state o_t for different target words.

Analysis on encoder. In GRU, g_t in Eq. 14 $\sim$ 16 indicates that the importance of h_{t-1}. The lower g_t is, the more important the word content itself x_t and the less important the previous state h_{t-1}. To obtain word representation vectors, the encoder uses a forward GRU and a backward GRU to read an input sentence in two directions. Table 4 presents the average g_t values in the forward GRU and the backward GRU. Interestingly, as shown in the table, the two GRUs behave differently in choosing amount of input word x_t and previous hidden state h_{t-1}. There also exists a consistent trend in the two GRUs that as expected, content words usually have lower g_t than function words, indicating that the encoder focuses more on the words themselves while encoding content words.

Analysis on decoder. In the output states, we assign α_s to control the current hidden state s_t, α_y to the previously generated word y_{t-1} and α_c to the source context c_t. Table 5 lists their average values for a few typical target words in the development set MT06. From it, we observe that:

- Generally speaking, the decoder pays more attention to the current hidden states s_t than either source context c_t or previous target word y_{t-1}.
- As expected, the decoder works differently in selecting how much source context to generate different types of target words. For example, *comma*, *period* and the end of sentence token *EOS* have the lowest α_c scores in that they are mainly decided by the translation content itself and less influenced by source context. This explains why it is surprising that target side *EOS* aligns to source side *EOS* in low frequency (e.g., 20%).
- Similarly, function words, including prepositions (e.g., *of*, *to*, *on*) and auxiliary words (e.g., *is*, *has*) have low α_c scores too, indicating that their generation is less decided by source context.
- Compared to function words, content words (e.g., *US*, *China*, *president*) are opt to have higher α_c scores, suggesting the decoder considers more on source context to generate target side content words.
- However, there are also scenarios that some target content words are less influenced by source context. In 1-to-many translation where a source content word aligns to multiple target content words (i.e., 新华社/Xinhua News Agency, 香港/Hong Kong, and 纽约/New York), the first target content word usually has higher α_c score while the others have lower α_c scores. Figure 4 illustrates the changes of α_s, α_c, and α_y in a real translation *Xinhua News Agency, Hong Kong*. Apart from α_s, it shows a very obvious trend that *Xinhua* and *Hong* have higher α_c scores while *News*, *Agency*, and *Kong* have lower ones.

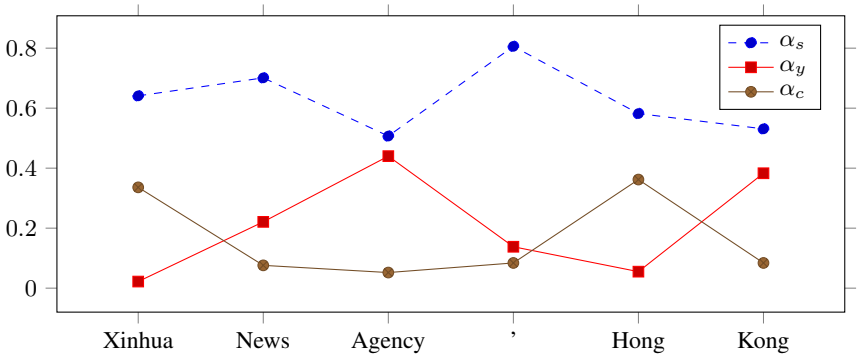


Figure 4: α_s, α_y and α_c scores in a real translation "新华社 香港 / Xinhua News Agency , Hong Kong". In it, there exist two 1-to-many correspondences, i.e., 新华社/Xinhua News Agency, and 香港/Hong Kong.

6 Related Work

We describe related work from two perspectives.

Gate mechanism. Our work is partially inspired by studies of gate mechanism for neural networks. Following the success of LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014a; Cho et al., 2014b), the gate mechanism has become standard components in RNN. Recently, Srivastava et al. (2015) employ gate units to regulate information flow, called highway networks. The most relevant work to ours is Tu et al. (2017), in which they propose context gate to control the ratios of the source context (i.e., c_t) and target context (i.e., y_{t-1} and s_{t-1}) for computing next target state s_t. On the contrary, we use gate units to regulate information flow in computing the output state o_t. Moreover, we propose adaptive weighting for GRU through gate units.

Interpretation for neural networks. Attention mechanism (Bahdanau et al., 2015; Lin et al., 2017; Vaswani et al., 2017) offers a way of understanding the contribution of every source words to the generation of a target word. Ding et al. (2017) propose to use layer-wise relevance propagation (LRP) to

interpret the internal workings of NMT and analyze translation errors. Moreover, Karpathy et al. (2015) and Li et al. (2016) propose to visualize and understand RNNs for natural language processing. In this work, we use the proposed gates in both encoder and decoder to analyze what types of information encoded in the encoder and what types of information influences the generation of a target word.

7 Conclusion

In this paper, we present an approach to regulate the information flow in neural machine translation model explicitly. This is done by employing adaptive weights through gate units. We apply adaptive weighting for both GRU and the output intermediate state. Experimental results on Chinese-to-English and English-to-Germany translation tasks show that the proposed approach achieves better translation performance and alignment quality over baseline NMT system.

Acknowledgements

The authors would like to thank anonymous reviewers for providing helpful comments, and also acknowledge Deyi Xiong, Xing Wang, Mingming Yang, Xiangyu Duan, Zhengxian Gong for useful discussions. This work was supported by National Natural Science Foundation of China (Grant No. 61525205, 61432013). Junhui and Yachao are also partially supported by the Opening Project of Key Laboratory of China's Ethnic Languages and Information Technology of Ministry of Education (Grant No. KFJJ201605).

References

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

Antonio Valerio Miceli Barone, Jindrich Helcl, Rico Sennrich, Barry Haddow, and Alexandra Birch. 2017. Deep architectures for neural machine translation. In *Proceedings of WMT 2017*, pages 99–107.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machinetranslation: Encoder-decoder approaches. In *Proceedings of SSST 2014*, pages 103–111.

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

Yanzhuo Ding, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Visualizing and understanding neural machine translation. In *Proceedings of ACL 2017*, pages 1150–1159.

Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017. A convolutional encoder model for neural machine translation. In *Proceedings of ACL 2017*, pages 123–135.

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

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. Montreal neural machine translation systems for wmt'15. In *Proceedings of WMT 2015*, pages 134–140.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. In *arXiv:1506.02078*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *arXiv:1412.6980*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*, pages 388–395.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *Proceedings of NAACL 2016*, pages 681–691.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of ACL 2017*, pages 688–697.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *Proceedings of ICLR 2017*.

Yang Liu and Maosong Sun. 2015. Contrastive unsupervised word alignment with non-local features. In *Proceedings of AAAI 2015*, pages 857–868.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP 2015*, pages 1412–1421.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL 2002*, pages 311–318.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of ACL 2016*, pages 1715–1725.

Rupesh Kumar Srivastava, Klaus Greff, and J ü rgen Schmidhuber. 2015. Highway networks. In *Proceedings of ICML 2015 Deep Learning workshop*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of ACL 2016*, pages 76–85.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2017. Context gates for neural machine translation. *Transactions of the Association for Computational Linguistics*, 5:87–99.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS 2017*, pages 6000–6010.

Xing Wang, Zhengdong Lu, Zhaopeng Tu, Hang Li, Deyi Xiong, and Min Zhang. 2017. Neural machine translation advised by statistical machine translation. In *Proceedings of AAAI 2017*, pages 3330–3336.

Yonghui Wu, Mike Schuster, Zhifeng Chen, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. In *arXiv preprint arXiv:1609.08144*.

Generic refinement of expressive grammar formalisms with an application to discontinuous constituent parsing

Kilian Gebhardt
Department of Computer Science
Technische Universität Dresden
D-01062 Dresden, Germany
kilian.gebhardt@tu-dresden.de

Abstract

We formulate a generalization of Petrov et al. (2006)'s split/merge algorithm for interpreted regular tree grammars (Koller and Kuhlmann, 2011), which capture a large class of grammar formalisms. We evaluate its effectiveness empirically on the task of discontinuous constituent parsing with two mildly context-sensitive grammar formalisms: *linear context-free rewriting systems* (Vijay-Shanker et al., 1987) as well as *hybrid grammars* (Nederhof and Vogler, 2014).

1 Introduction

Probabilistic grammars are a standard model for language processing tasks. Their fundamental principle is a rewriting process in which nonterminals are repeatedly unfolded in accordance to rewrite rules until a structure consisting solely of terminals is obtained. Context-free independence assumptions imply that the applicability as well as the probability of a rewrite step depend only on the nonterminal that is unfolded but not on the context or history in which the nonterminal occurs.

The independence assumptions allow for tractable algorithms when processing data with these grammars. Then again, the expressiveness of such a grammar is constrained by the number of its nonterminals. This is why it was found beneficial to refine naturally emerging sets of nonterminals (such as syntactic categories). Strategies of refinement of *context-free grammars* (CFGs) involve for instance *Markovization*, i.e., the encoding of limited context into the nonterminals (Collins, 1999; Klein and Manning, 2003), and automatic state-splitting by means of latent annotations (Matsuzaki et al., 2005; Petrov et al., 2006). An important observation is that these refinements are *latent*, i.e., they are not observed in the predictions that the CFG is supposed to provide. Automatic state-splitting has been successfully applied also to tree-substitution grammars (Shindo et al., 2012) and tree-adjoining grammars (Ferraro et al., 2012).

Koller and Kuhlmann (2011) proposed *interpreted regular tree grammars* (IRTGs) as a uniform way to describe a large class of grammar formalisms that share the context-free rewriting mechanism. IRTGs decouple the derivational process, in which derivation trees are generated by a (probabilistic) *regular tree grammar* (RTG), from the interpretation of derivation trees in one or multiple algebras. IRTGs enable the development of generic algorithms for binarization (Büchse et al., 2013), parsing and decoding (Groschwitz et al., 2016; Teichmann et al., 2017), and estimation techniques (Teichmann et al., 2016).

The central hypothesis of this article is that Petrov et al. (2006)'s *split/merge algorithm* (a) can be transferred from CFGs to a large class of grammar formalisms and (b) that its application improves the probabilistic behavior of a given grammar for parsing and decoding tasks.

The first contribution of our paper is a generic version of Petrov et al. (2006)'s *split/merge algorithm* in the IRTG framework. We choose IRTGs because the separation of the derivational process in an RTG from the interpretation in algebras implies that (i) only one algorithm needs to be formulated to capture a large class of grammar formalisms and that (ii) the nonterminals cannot be observed in the generated structures. Because of (ii) nonterminals of IRTGs may be viewed as already being latent, i.e., with IRTGs latent annotations *come for free*. We also transfer objectives for efficient parsing and decoding as proposed

Proceedings of the 27th International Conference on Computational Linguistics, pages 3049–3063
Santa Fe, New Mexico, USA, August 20-26, 2018.

by Matsuzaki et al. (2005) and Petrov and Klein (2007) into the IRTG framework. An implementation of the generic algorithms is provided.

Then, for a case study of (b) we apply the generalized split/merge algorithm and the different parsing objectives to both *linear context-free rewriting systems* (LCFRSs) (Vijay-Shanker et al., 1987; Kallmeyer and Maier, 2013) and *hybrid grammars* (Nederhof and Vogler, 2014) on the task of discontinuous constituent parsing. This choice is relevant because the application of the split/merge algorithm to either grammar formalism has been supposed by Evang and Kallmeyer (2011) and Gebhardt et al. (2017), respectively, but to our knowledge not yet been performed. We find that the split/merge algorithm improves the parsing accuracy of the grammars by up to 14.5 points in labeled F1. However, the grammars do not reach the accuracy of recent transition-based discriminative parsers.

2 Preliminaries

Let A, B, and C be sets and $f\colon A \to B$ and $g\colon B \to C$ be functions. The *powerset of* A is denoted by $\mathcal{P}(A)$. We extend f in the natural way to $f\colon \mathcal{P}(A) \to \mathcal{P}(B)$ and $f^{-1}\colon \mathcal{P}(B) \to \mathcal{P}(A)$. We call f *surjective*, if for each $b \in B$, there exists $a \in A$ with $f(a) = b$. We let $g \circ f\colon A \to C$ denote the composition of f and g. We identify a singleton set $\{a\}$ with its element a.

2.1 Interpreted regular tree grammars

An interpreted regular tree grammar generates an object a of some domain A in two phases: firstly a derivation tree ξ is generated and secondly ξ is interpreted in an algebra $\mathcal{A}$ to a. Figure 1 shows two derivation trees ξ_1 and ξ_2 over *operator symbols* f_0, f_1 and f_2. Each operator symbol admits a fixed number of arguments called its *rank*, e.g., the ranks of f_0, f_1, and f_2 are 0, 1, and 2, respectively. A finite, non-empty set of operator symbols constitutes a *signature* Σ. The set of *derivation trees over* Σ, denoted by T_Σ, is the smallest set U where for any $f \in \Sigma$ of rank k and $\xi_1, \ldots, \xi_k \in U$ we have $f(\xi_1, \ldots, \xi_k) \in U$. Let $\xi = f(\xi_1, \ldots, \xi_k)$ be in T_Σ. The set of *positions* of ξ is $\mathrm{pos}(\xi) = \{\varepsilon\} \cup \{i\pi \mid 1 \le i \le k, \pi \in \mathrm{pos}(\xi_i)\}$. The *operator symbol at the position* π *in* ξ, denoted by $\xi(\pi)$, is f if $\pi = \varepsilon$ and $\xi_i(\pi')$ if $\pi = i\pi'$.

Next, we describe the interpretation of a derivation tree from T_Σ by a Σ-algebra. A Σ-*algebra* $\mathcal{A}$ consists of a set A (*domain*) and, for each operator symbol f in Σ of rank k, an *operation* $f^{\mathcal{A}}\colon A^k \to A$. Each derivation tree $f(\xi_1, \ldots, \xi_n)$ in T_Σ can be *evaluated in* $\mathcal{A}$ to an element $[\![f(\xi_1, \ldots, \xi_k)]\!]_{\mathcal{A}} = f^{\mathcal{A}}([\![\xi_1]\!]_{\mathcal{A}}, \ldots, [\![\xi_k]\!]_{\mathcal{A}})$ in A. Let $\Sigma_{\mathrm{ex}} = \{f_0, f_1, f_2\}$. Figure 1 shows the operations of the Σ_{ex}-algebras $\mathcal{A}_s$ and $\mathcal{A}_t$ with the set of strings and parse trees as domains, respectively. Also, it shows the evaluation of $\xi_1, \xi_2 \in T_{\Sigma_{\mathrm{ex}}}$ in $\mathcal{A}_s$ to the string $s = bbb$ and the evaluation of ξ_1 and ξ_2 in $\mathcal{A}_t$ to the parse trees t_1 and t_2, respectively.

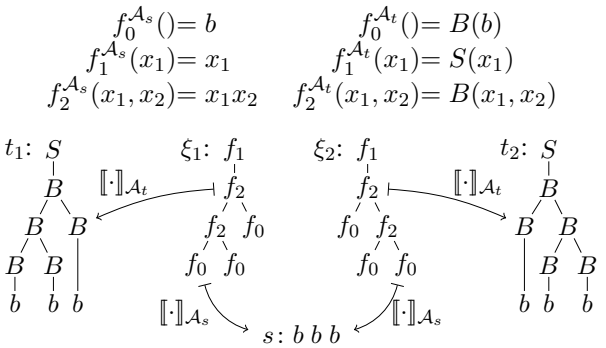


Figure 1: Operations of the Σ_{ex}-algebras $\mathcal{A}_s$ and $\mathcal{A}_t$, derivation trees ξ_1 and ξ_2, and evaluation of ξ_1 and ξ_2 in $\mathcal{A}_s$ and $\mathcal{A}_t$.

Regular tree grammars are useful to describe formal languages of derivation trees. A *regular tree grammar* (RTG) (Brainerd, 1969) over Σ is a tuple $G = (N_G, S_G, R_G)$. The finite set N_G is disjoint from Σ and contains *nonterminals*. S_G in N_G is the *start nonterminal*. The set R_G is a finite subset of the set of prototypical rules $R[N_G, \Sigma]$. $R[N_G, \Sigma]$ contains each *rule* of the form $B \to f(B_1, \ldots, B_k)$ where $f \in \Sigma$ is of rank k and $B, B_1, \ldots, B_k$ are in N_G. Figure 2a shows the rules of $G_{\mathrm{ex}} = (\{S, B\}, S, R_G)$ over Σ_{ex}.

An RTG G generates a derivation tree $\xi \in T_\Sigma$ if it has a valid run on it: A *run of* G *on* ξ is a mapping $r\colon \mathrm{pos}(\xi) \to N_G$. The *rule at position* π *of* r is $\mathrm{rule}_r^\pi = r(\pi) \to \xi(\pi)(r(\pi 1), \ldots, r(\pi k))$ where k is the rank of $\xi(\pi)$. We call r *valid* if $r(\varepsilon) = S_G$ and r is *consistent with* R_G, i.e., for every $\pi \in \mathrm{pos}(\xi)$, we require that $\mathrm{rule}_r^\pi \in R_G$. We denote the set of all valid runs of G on ξ by $\mathrm{runs}_G^v(\xi)$. The *language of* G is $L(G) = \{\xi \in T_\Sigma \mid \mathrm{runs}_G^v(\xi) \neq \emptyset\}$. Moreover, we let $\mathrm{runs}_G^v = \{(\xi, r) \mid \xi \in T_\Sigma, r \in \mathrm{runs}_G^v(\xi)\}$.

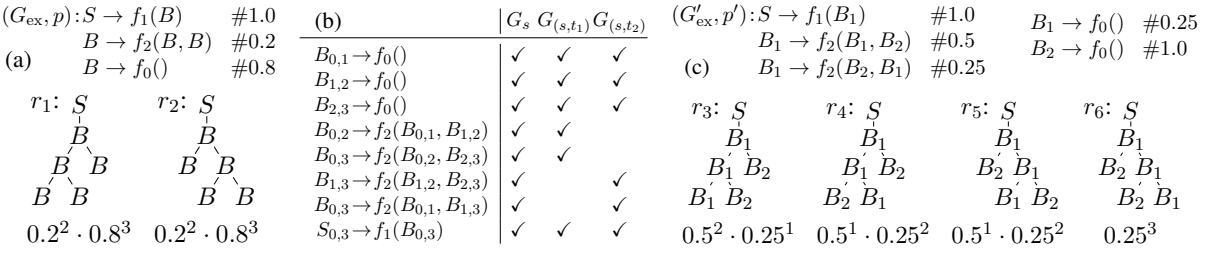


Figure 2: (a) and (c): Prob. RTGs (G_{ex}, p) and (G'_{ex}, p') with runs and probabilities (below). (b): Charts.

Figure 2a shows the only valid runs r_1 and r_2 of G_{ex} on the derivation trees ξ_1 and ξ_2, respectively, in tree-like form.

Interpreted RTGs. An *interpreted RTG* (IRTG) (Koller and Kuhlmann, 2011) is a tuple $\mathbb{G} = (G, \mathcal{A}_1, \ldots, \mathcal{A}_l)$ where G is an RTG over Σ and each $\mathcal{A}_j$ is a Σ-algebra[1]. Given an object $a = (a_1, \ldots, a_l)$ from the *domain* $A = A_1 \times \ldots \times A_l$, the parsing problem for IRTG is to compute $\text{parses}_{\mathbb{G}}(a) = \{\xi \in L(G) \mid \forall j : [\![\xi]\!]_{\mathcal{A}_j} = a_j\}$. This set is equal to the intersection of all sets $D_{a_j} = \{\xi \in T_\Sigma \mid [\![\xi]\!]_{\mathcal{A}_j} = a_j\}$ and $L(G)$. If, for each $a_j \in A_j$, there is an RTG whose language is D_{a_j}, then $\mathcal{A}_j$ is called *regularly decomposable*. In the following we consider only such algebras. Since the languages of RTGs are closed under intersection, we can construct RTGs G_{a_j} with $L(G_{a_j}) = D_{a_j} \cap L(G)$ and the RTG G_a with $L(G_a) = \bigcap_j L(G_{a_j}) = \text{parses}_{\mathbb{G}}(a)$. We call G_{a_j} and G_a the *chart of* a_j and the *chart of* a, respectively.

As a running example we extend the RTG G_{ex} to an IRTG $\mathbb{G}_{\text{ex}} = (G_{\text{ex}}, \mathcal{A}_s, \mathcal{A}_t)$. Then $\mathbb{G}_{\text{ex}}$ is equivalent to a CFG where $\mathcal{A}_s$ specifies the generation of strings and $\mathcal{A}_t$ describes the corresponding parse trees. Figure 2b depicts the rules of the charts G_s, $G_{(s,t_1)}$, and $G_{(s,t_2)}$ where the nonterminals were annotated with spans as usual and the start nonterminal is $S_{0,3}$.

Grammar morphisms. In order to relate two RTGs G' and G (e.g., G' could be a chart G_a), we consider mappings $\varphi\colon N_{G'} \to N_G$ between the respective sets of nonterminals. We lift φ to $\varphi\colon R[N_{G'}, \Sigma] \to R[N_G, \Sigma]$ by setting $\varphi(B' \to f(B'_1, \ldots B'_n)) = \varphi(B') \to f(\varphi(B'_1), \ldots, \varphi(B'_n))$. If $\varphi^{-1}(S_G) = \{S_{G'}\}$ and $\varphi(R_{G'}) \subseteq R_G$, then we call φ a *grammar morphism from G' to G*. Intuitively, φ has to establish a correspondence between the grammars' start nonterminals and rules.

For instance, we can choose any set M and any mapping $\varphi\colon N_G \to M$ that satisfies $\varphi^{-1}(\varphi(S_G)) = \{S_G\}$ and construct a new RTG $\varphi(G)$ from G: we set $\varphi(G) = (\varphi(N_G), \varphi(S_G), \varphi(R_G))$. Obviously, φ is a grammar morphism from G to $\varphi(G)$.

For each chart G_a we assume a grammar morphism φ_a from G_a to G that gives rise to a one-to-one correspondence between the valid runs of G_a and the valid runs of G for derivations trees of a. Formally, for each $\xi \in \text{parses}_{\mathbb{G}}(a)$ we require that $\bar{\varphi}_a\colon \text{runs}^v_{G_a}(\xi) \to \text{runs}^v_G(\xi)$ where $\bar{\varphi}_a(r) = \varphi_a \circ r$ is a bijection. For instance, φ_s, $\varphi_{(s,t_1)}$, and $\varphi_{(s,t_2)}$ strip the spans from each nonterminal, e.g., $\varphi_s(B_{i,j}) = B$.

2.2 Extending IRTGs with probabilities

RTGs and IRTGs can be equipped with probabilities in the *standard* way: A *weight assignment for an RTG* G is a mapping $p\colon R_G \to [0, 1]$. We define the weight of a run r on a derivation tree ξ as $W_{(G,p)}(\xi, r) = \prod_{\pi \in \text{pos}(\xi)} p(\text{rule}^\pi_r)$ and the weight of a derivation tree ξ as $W_{(G,p)}(\xi) = \sum_{r \in \text{runs}^v_G(\xi)} W_{(G,p)}(\xi, r)$. We call p *proper* if, for every $A \in N_G$, we have $1 = \sum_{\varrho = (A \to f(B_1, \ldots, B_k)) \text{ in } R_G} p(\varrho)$. If $1 = \sum_{\xi \in T_\Sigma} W_{(G,p)}(\xi)$, then we call p *consistent*. In this case we may write $P(\xi, r \mid G, p)$ and $P(\xi \mid G, p)$ for $W_{(G,p)}(\xi, r)$ and $W_{(G,p)}(\xi)$, respectively, and call (G, p) *probabilistic RTG*. Let G' be an RTG and φ be a grammar morphism from G' to G. Then $p \circ \varphi\colon R_{G'} \to [0, 1]$ is a weight assignment for G'.

Given an IRTG $\mathbb{G} = (G, \mathcal{A}_1, \ldots, \mathcal{A}_l)$ and a proper and consistent weight assignment p for G, we define a probability distribution on A by $P(a \mid \mathbb{G}, p) = \sum_{\xi \in \text{parses}_{\mathbb{G}}(a)} W_{(G,p)}(\xi)$. Using the chart G_a the same quantity can be obtained: $P(a \mid \mathbb{G}, p) = \sum_{\xi \in L(G_a)} W_{(G_a, p \circ \varphi_a)}(\xi)$ (see Appendix A.1).

[1] For ease of notation, we omit the homomorphism that is originally associated to each algebra. We note that the methods we develop in the following are agnostic of the algebras and, thus, applicable to the original definition as well.

Considering $\mathbb{G}_{\text{ex}}$ and the probability assignment p for G in Figure 2a, we get that $P((s,t_1) \mid \mathbb{G}_{\text{ex}}, p) = P((s,t_2) \mid \mathbb{G}_{\text{ex}}, p)$ because r_1 and r_2 have the same probability. In contrast, G'_{ex} in Figure 2c has two runs (r_3/r_4 and r_5/r_6, respectively) on each of ξ_1 and ξ_2. Taking the sum of their probabilities yields $P((s,t_1) \mid \mathbb{G}'_{\text{ex}}, p') > P((s,t_2) \mid \mathbb{G}'_{\text{ex}}, p')$ for the IRTG $\mathbb{G}'_{\text{ex}} = (G'_{\text{ex}}, \mathcal{A}_s, \mathcal{A}_t)$.

Inside and outside weights are a tool for efficient calculation of probabilities during training and parsing. The *inside weight* $\beta(B)$ and the *outside weight* $\alpha(B)$ of a nonterminal $B \in N_G$ are defined as

$$\beta(B) = \sum_{\varrho = B \to f(B_1,\ldots,B_k) \in R_G} p(\varrho) \cdot \beta(B_1) \cdot \ldots \cdot \beta(B_k) \quad \text{and} \quad \alpha(B) = \delta_B^S + \sum_{\substack{\varrho = C \to f(B_1,\ldots,B_k) \text{ in } R_G \\ 1 \le i \le k\colon B_i = B}} \alpha(C) \cdot p(\varrho) \cdot \prod_{j \neq i} \beta(B_j) \ ,$$

respectively, where $\delta_B^S = 1$ if $B = S$ and 0 otherwise.

The sum of the probabilities of all runs of an RTG G equals $\beta(S_G)$. Hence, an efficient way to obtain $P(a \mid \mathbb{G}, p)$ is computing $\beta(S_{G_a})$. The expected frequency with which a nonterminal B occurs in a run of G is obtained by $\alpha(B) \cdot \beta(B)/\beta(S_G)$ (Nederhof and Satta, 2004). For any rule ϱ of the form $B \to f(B_1, \ldots, B_k)$, we let $\alpha(\varrho) = \alpha(B)$ and $\beta(\varrho) = \beta(B_1) \cdot \ldots \cdot \beta(B_k)$.

2.3 A parsing or decoding problem

Suppose we want to employ $\mathbb{G}_{\text{ex}}$ for syntactic parsing (for clarity we write G instead of G_{ex}). This can be framed as a decoding problem: for a given string s, the chart G_s is computed, from which we obtain the set $T = [\![L(G_s)]\!]_{\mathcal{A}_t}$ of parse trees of s (i.e., $T = \{t_1, t_2\}$ in our example). If the IRTG is equipped with a probability assignment, then, alternatively, one can ask for the *best* parse tree $\hat{t}$, which may be formalized as

$$\hat{t} = \arg\max_{t \in T} \quad \sum_{\xi \in L(G_s)\colon [\![\xi]\!]_{\mathcal{A}_t} = t} \quad P(\xi \mid G, p) \ . \tag{1}$$

Computing this expression turns out to be infeasible in general because maximizing the sum over the potentially infinite set of derivation trees and the sum over the exponential number of runs over each derivation tree resists dynamic programming. A tractable option is to compute the Viterbi run $\hat{r}$ of the grammar (Knuth, 1977; Nederhof, 2003), defined as

$$(\hat{\xi}, \hat{r}) = \arg\max_{(\xi,r) \in \text{runs}^v_{G_s}} P(\xi, r \mid G, p) \tag{2}$$

or, with a small overhead, the n-best runs (Huang and Chiang, 2005).

For a given derivation tree $\xi \in T_\Sigma$, we can efficiently compute or approximate $P(\xi \mid G, p)$ by restricting G to ξ, which yields an RTG G_ξ, and computing $\beta(S_{G_\xi})$. Likewise, for a given parse tree t, we can compute $P(t \mid s, G, p) = \beta(G_{(s,t)})/\beta(G_s)$.

2.4 Expectation/Maximization training

The expectation/maximization (EM) algorithm (Dempster et al., 1977) in the inside-outside variant (Lari and Young, 1990) carries over to IRTGs. We recall it for sake of completeness. During the *expectation* step, we compute a corpus $c\colon R_G \to \mathbb{R}_{\ge 0}$ over rules given a corpus $c_A\colon A \to \mathbb{R}_{\ge 0}$ over the domain such that

$$c(\varrho) = \sum_{a \in A} c_A(a) \cdot \sum_{\varrho' \in \varphi_a^{-1}(\varrho)} \frac{\alpha(\varrho') \cdot p(\varrho') \cdot \beta(\varrho')}{\beta(S_{G_a})} \ .$$

In the *maximization* step the probability assignment p is updated to match the empirical distribution of $c(\varrho)$. Both steps are iterated until p changes only slightly or until the likelihood of a validation set drops.

3 Refinement of IRTGs with the split/merge algorithm

Probabilistic context-free grammars with latent annotation (PCFG-LAs) were introduced by Matsuzaki et al. (2005) as a way to tackle the too strong independence assumptions of probabilistic CFG. Instead of assigning each CFG rule just a single probability, different probabilities are assigned depending on the substate which is annotated to each nonterminal. These substates are *latent*, i.e., when calculating

the probability of a parse tree, any assignment of substates to nonterminals is considered. The work of Petrov et al. (2006) extends the PCFG-LA approach by a procedure that adaptively refines the latent state set. Petrov et al. (2006) start from a binarized PCFG-LA where each nonterminal has just one substate. In multiple cycles each such substate is split in two and the resulting grammar is trained with the EM algorithm. Then 50% of the splits are undone depending on how much likelihood gets lost and the grammar is trained with EM again. Finally, the rule weights are smoothed and trained once more.

The idea of latent annotated grammar states can be easily formalized in the IRTG framework: The probabilistic behavior of the IRTG depends only on the nonterminals and the applied rules of its RTG G. However, in the derivation trees in $L(G)$ and their interpretations, the nonterminals of G are no longer visible. To calculate the probability of a derivation tree ξ, we have to consider any valid run r on ξ and its weight. Thus, the latent states of a PCFG-LA naturally correspond to the nonterminals of the RTG G.

We reformulate the split/merge algorithm by Petrov et al. (2006) for an arbitrary IRTG $\mathbb{G} = (G, \Sigma, \mathcal{A}_1, \ldots, \mathcal{A}_l)$. In the process a (fine) probabilistic RTG (G', p') is constructed from the (coarse) probabilistic RTG (G, p), while Σ and the algebras are not changed. The refinement from N_G to $N_{G'}$ allows defining a more subtle probabilistic behavior in (G', p'). Thus, $L(G) = L(G')$, however $W_{(G,p)}$ and $W_{(G',p')}$ may differ. In analogy to the original algorithm, each nonterminal is split in two by an inverse grammar morphism μ_{sp}^{-1} yielding an intermediate grammar G^f. The probabilities of the rules of G^f are tuned by EM training. Afterwards, splits that turn out less useful are reversed by a grammar morphism μ_Δ yielding the grammar G'. The probabilities of this grammar are trained again and smoothed.

The grammar morphisms between the grammars G, G^f, and G', also imply the existence of grammar morphisms between the charts of these grammars. Consequently, charts can be easily transformed (sparing recomputation from scratch) which increases the efficiency of EM training and parsing (cf. Section 3.4).

3.1 Splitting and merging

We define the splitting and merging of nonterminals in a generic way by (inverse) grammar morphisms. Let (G, p) be a probabilistic RTG. For splitting the nonterminals of (G, p) we consider a surjective mapping $\mu\colon N' \to N_G$ where N' is a finite set (fine nonterminals) and $\mu^{-1}(S_G)$ is a singleton set. Splitting corresponds to applying the *inverse* of μ to G, i.e., $\mu^{-1}(G) = (N', \mu^{-1}(S_G), R')$ where $R' = \mu^{-1}(R) = \{\varrho' \in R[N', \Sigma] \mid \mu(\varrho) \in R\}$. The corresponding weight assignment for $\mu^{-1}(G)$ is $p \circ \mu$, which may be normalized to obtain a genuine probability assignment.

For merging the nonterminals of (G, p) we consider a surjective mapping $\mu\colon N_G \to M$ where M is a finite set (merged nonterminals) and $\mu^{-1}(\mu(S_G)) = \{S_G\}$. Merging is as simple as applying μ to G, i.e., computing $\mu(G) = (M, \mu(S_G), \mu(R))$. In order to construct a probability assignment $\mu(p)$ for $\mu(G)$, we let for every $\hat{\varrho} \in \mu(R)$:

$$(\mu(p))(\hat{\varrho}) = \textstyle\sum_{\varrho \in \mu^{-1}(\hat{\varrho})} p(\varrho) \cdot \dfrac{\alpha(\varrho) \cdot \beta(\varrho)}{\sum_{\varrho' \in \mu^{-1}(\hat{\varrho})} \alpha(\varrho') \cdot \beta(\varrho')} \quad ,$$

where α and β are computed with respect to (G, p).

Two instances. We give two instances for grammar morphisms in reminiscence of Petrov et al. (2006). For splitting we consider a grammar morphism μ_{sp} that splits every nonterminal B of G but the start nonterminal into B_1 and B_2. Formally, $\mu_{\mathrm{sp}}\colon N' \to N_G$ where $N' = \{B_q \mid B \in N_G, B \neq S, q \in \{1, 2\}\} \cup \{S\}$ and

$$\mu_{\mathrm{sp}}(B') = \begin{cases} B & \text{if } B' = B_1 \text{ or } B' = B_2 \\ B' & \text{if } B' = S \end{cases}$$

In order to partially reverse the split of μ_{sp}, we define the grammar morphism μ_Δ that merges each pair B_1 and B_2 back to B based on a utility measure $\Delta(B_1, B_2)$. Formally, $\mu_\Delta\colon N' \to M$ where

$$\mu_\Delta(B') = \begin{cases} B & \text{if } B' = B_q \text{ for } B \in N_G,\ q \in \{1, 2\},\ \text{and } \Delta(B_1, B_2) > \eta \\ B' & \text{otherwise.} \end{cases}$$

We chose M to be the largest subset of $N_G \cup N'$ such that μ_Δ is surjective.

The function Δ is meant to approximate the quotient of likelihood after and before merging. This approximation, introduced by Petrov et al. (2006), uses inside and outside weights of charts, which were precomupted during EM training, to simulate merging of single instances of B_1 and B_2 in one chart. A generalization of Δ can be defined for arbitrary IRTG as long as each nonterminal of some chart occurs at most once in a run (App. A.2). The parameter η is set dynamically such that 50% of the splits are merged.

3.2 The complete split/merge cycle

Splitting, merging, the EM training of Section 2.4, and a tie-breaking and smoothing step, yet to be defined, is composed to a complete *split/merge cycle* in Algorithm 3.1. Multiple split/merge cycles are iteratively applied to a *base grammar* G_0 until the resulting grammar G_i reaches the desired level of refinement. For every refined grammar G_i, there exists a grammar morphism μ_i from G_i to G_0. During *tie-breaking* each rule's probability obtains a small random perturbation. During *smoothing* the probability of a rule ϱ is set proportional to $\gamma \cdot p(\varrho) + (1-\gamma) \cdot u$ where u is the sum of probabilities of rules $\varrho' \in \mu_i^{-1}(\mu_i(\varrho))$. Intuitively, the probability of different refinements of the same rule from the base grammar is slightly aligned. Following Petrov et al. (2006), we set γ to 0.9 for rules without nonterminals on the right-hand side. Otherwise, we set γ to 0.99.

Algorithm 3.1 Split/merge cycle

Input: IRTG $\mathbb{G} = (G, \mathcal{A}_1, \ldots, \mathcal{A}_l)$, prob. ass. p
corpus $c_A \colon A \to \mathbb{R}_{\geq 0}$
Output: IRTG $\mathbb{G}'$, prob. assignment p'

1: $(G^f, p^f) \leftarrow (\mu_{\text{sp}}^{-1}(G), p \circ \mu_{\text{sp}})$
2: $p^f \leftarrow \textsc{BreakTies}(p^f)$
3: $p^f \leftarrow \textsc{EM-Training}(G^f, p^f, \mathcal{A}_1, \ldots, \mathcal{A}_l, c_A)$
4: $(G', p') \leftarrow (\mu_\Delta(G^f), \mu_\Delta(p^f))$
5: $p' \leftarrow \textsc{EM-Training}(G', p', \mathcal{A}_1, \ldots, \mathcal{A}_l, c_A)$
6: $p' \leftarrow \textsc{Smooth}(G', p')$
7: $p' \leftarrow \textsc{EM-Training}(G', p', \mathcal{A}_1, \ldots, \mathcal{A}_l, c_A)$
8: **output** $(G', \mathcal{A}_1, \ldots, \mathcal{A}_l), p'$

3.3 Efficient refinement of a chart

Let G^c be a coarse grammar, $a \in A$ be in the domain, and the chart G_a^c be already computed. We assume that G^c was refined to $G^f = \mu^{-1}(G^c)$ and that we want to compute the chart G_a^f. Due to the definition of splitting and merging, we do not need to compute G_a^f from scratch. Instead we construct (a grammar that is isomorphic to) G_a^f via the grammar morphisms φ'_a and μ' such that the diagram on the right commutes. Let $N' = \{(B, q) \mid B \in N_{G_a^c}, q \in \mu^{-1}(\varphi_a(B))\}$, and for every $(B, q) \in N'$, set $\varphi'_a(B, q) = q$ and $\mu'(B, q) = B$. We define $G_a^f = {\mu'}^{-1}(G_a^c)$.

$$\begin{array}{ccc} G_a^f & \xrightarrow{\varphi'_a} & G^f \\ {\scriptstyle \mu'}\downarrow & & \downarrow{\scriptstyle \mu} \\ G_a^c & \xrightarrow[\varphi_a]{} & G^c \end{array}$$

3.4 Parsing objectives and weight projections

In order to apply a refined IRTG to parsing, we return to the question on how to substitute Equation 1. We consider alternative parsing objectives inspired by Matsuzaki et al. (2005). In the following we refer to the base RTG and the one resulting from several iterations of Algorithm 3.1 by G^c and G^f, respectively. Also, we consider charts of a string s and assume grammar morphisms φ_s, φ'_s, μ, and μ' as in Section 3.3.

The $\hat{t}$ of Equation 1 is sometimes called *most probable parse*. A subproblem that is in general still infeasible is finding the parse tree corresponding to the *most probable derivation tree*, i.e.,

$$\hat{t} = [\![\arg\max_{\xi \in L(G_s^f)} P(\xi \mid G^f, p^f)]\!]_{\mathcal{A}_t} \ .$$

For usual PCFG(-LA) this objective coincides with the most probable parse because derivation trees and parse trees are in a one-to-one correspondence.

The parse corresponding to the *viterbi derivation tree* is $\hat{t} = [\![\hat{\xi}]\!]$ where $(\hat{\xi}, \hat{r})$ is computed according to Equation 2 for $(G_s^f, p^f \circ \varphi'_s)$. This objective is tractable but reported to yield suboptimal parses for PCFG-LA in terms of the usual bracket scoring metric (Matsuzaki et al., 2005; Petrov et al., 2006).

A combination of coarse-to-fine parsing with n-best parsing yields the *base-n-rerank* objective: n-best runs of the base grammar are computed and the corresponding derivation trees are reranked according to the refined grammar. Formally, we compute $\hat{t} = [\![\hat{\xi}]\!]_{\mathcal{A}_t}$ with

$$(\hat{\xi}, \hat{r}) = \arg\max_{(\xi, r) \in n\text{-best-runs}(G_s^c, p^c \circ \varphi_s)} P(\xi \mid G^f, p^f) \ .$$

$\text{SQ} \to \text{yf}_1(\text{MD}, \text{VP}, \text{NP})$	$\text{yf}_1^{\mathcal{A}_s}(x_1, (x_2, x_3), x_4) = (x_2 x_1 x_3 x_4)$	$\text{yf}_1^{\mathcal{A}_t}(x_1, x_2, x_3) = \text{SQ}(x_1, x_2, x_3)$
$\text{VP} \to \text{yf}_2(\text{WHNP}, \text{VB})$	$\text{yf}_2^{\mathcal{A}_s}((x_1), (x_2)) = (x_1, x_2)$	$\text{yf}_2^{\mathcal{A}_t}(x_1, x_2) = \text{VP}(x_1, x_2)$
$\text{WHNP} \to \text{yf}_3()$	$\text{yf}_3^{\mathcal{A}_s}() = (\text{What})$	$\text{yf}_3^{\mathcal{A}_t}() = \text{WHNP}(\text{What})$
$\text{MD} \to \text{yf}_4()$	$\text{yf}_4^{\mathcal{A}_s}() = (\text{should})$	$\text{yf}_4^{\mathcal{A}_t}() = \text{MD}(\text{should})$
$\text{NP} \to \text{yf}_5()$	$\text{yf}_5^{\mathcal{A}_s}() = (\text{I})$	$\text{yf}_5^{\mathcal{A}_t}() = \text{NP}(\text{I})$
$\text{VB} \to \text{yf}_6()$	$\text{yf}_6^{\mathcal{A}_s}() = (\text{do})$	$\text{yf}_6^{\mathcal{A}_t}() = \text{VB}(\text{do})$

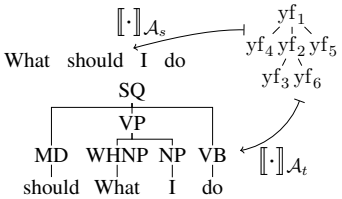


Figure 4: Rules of an LCFRS and the evaluation of a derivation tree in $\mathcal{A}_s$ and $\mathcal{A}_t$.

Parsing by weight projection. Matsuzaki et al. (2005) propose an alternative parsing objective, where a new weight assignment q for G_s^c is computed based on $(G_s^f, p^f \circ \varphi'_s)$ such that the KL-divergence of $P(\xi \mid G_s^c, q)$ to $P(\xi \mid s, G^f, p^f)$ is minimized. Precisely, $q = \mu'(p^f \circ \varphi'_s)$. Subsequently the parse tree $\hat{t}$ corresponding to the Viterbi derivation tree is computed using q, i.e., $\hat{t} = [\![\hat{\xi}]\!]_{\mathcal{A}_t}$ with

$$(\hat{\xi}, \hat{r}) = \arg\max_{(\xi,r) \in \text{runs}^v_{G_s^c}} P(\xi, r \mid G_s^c, q) \ .$$

An empirically superior way to define q called *max-rule-product* is due to Petrov and Klein (2007). They intent to optimize for "the tree with greatest chance of having all rules correct, under the (incorrect) assumption that the rules correctness are independent." To this end, each rule is assigned the sum of the expected frequencies of its refinements, i.e., $q(\varrho)$ is set to $\sum_{\varrho' \in \mu'^{-1}(\varrho)} \alpha(\varrho') \cdot (p^f \circ \varphi'_s)(\varrho') \cdot \beta(\varrho') / \beta(S_{G_s^f})$. Hence, the value $q(\varrho)$ does not have to be in the interval $[0, 1]$ and max-rule-product (as well as *max-rule-sum* – where the weight of a run is the sum of rule weights rather than the product) is in theory a potentially non-monotonic and, thus, ill-defined objective (cf. Appendix A.3).

4 Grammars for discontinuous parsing

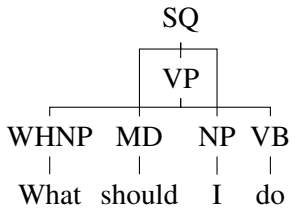


Figure 3: A discontinuous phrase structure.

String-rewriting *linear context-free rewriting systems* (LCFRSs) (Vijay-Shanker et al., 1987) generalize CFGs and can account for discontinuous constituent trees such as the one in Figure 3. Each nonterminal B derives instead of a single string an m_B-tuple of strings. The integer $m_B \geq 1$ is called *fanout* of B. Each rule of an LCFRS consists of a left-hand side nonterminal B, any number of right-hand side nonterminals $B_1, \ldots, B_k$, and a *yield function* yf. The latter specifies how the components of the string tuples generated by $B_1, \ldots, B_k$ are concatenated with new terminal symbols to form the string tuple of B. The rewriting is *linear*, that is, each input to yf is used at most once. Parsing of binary LCFRS is in $\mathcal{O}(n^{3m})$ where n is the sentence length and m is the maximum fanout of nonterminals of the LCFRS (Seki et al., 1991).

We use the following straightforward representation of LCFRSs as IRTGs: Σ is the set of yield function symbols, A_s is the set of tuples over strings, and the string algebra $\mathcal{A}_s$ interprets each yield function symbol by the respective yield function. To obtain the corresponding parse tree, the derivation tree is interpreted in an algebra $\mathcal{A}_t$. Figure 4 depicts an LCFRS with start nonterminal SQ and the interpretation of a derivation tree.

In a *LCFRS/sDCP-hybrid grammar* (Nederhof and Vogler, 2014; Gebhardt et al., 2017) (short: hybrid grammar), an LCFRS is coupled with a tree generating device called *simple definite clause program* (sDCP) (Deransart and Małuszynski, 1985). An object in the domain of a hybrid grammar is a *hybrid tree*, i.e., a string and a tree together with a linking between sentence positions and tree positions. Hybrid trees (and hybrid grammars) are suitable to model (formal languages of) discontinuous constituent structures. Figure 5 depicts a hybrid grammar in IRTG notation and the evaluation of the derivation tree ξ_3 in the algebra $\mathcal{A}_h$ to the hybrid tree h. The algebra $\mathcal{A}_s$ is a copy of the string component of $\mathcal{A}_h$ and evaluates ξ_3 to "What should I do". We observe that the structure of ξ_3 deviates notably from the structure of the tree component of h. In fact, the hybrid grammar generates a discontinuous tree although its string component is equivalent to a CFG.

Nederhof and Vogler (2014) present an algorithm that induces an LCFRS/sDCP-hybrid grammar $\mathbb{G}$ given a corpus of phrase structure trees. The algorithm is parametrized by an integer $k \geq 1$ that limits the maximum fanout of the LCFRS component of $\mathbb{G}$ to k. A second parameter of the algorithm is one of two nonterminal labeling schemes called *child labeling* and *strict labeling* of which the former is coarser. Drewes et al. (2016) present an algorithm that computes the chart G_h in time polynomial in the size of some hybrid tree h. Computing G_s, given a string s, inherits the parsing complexity of LCFRSs.

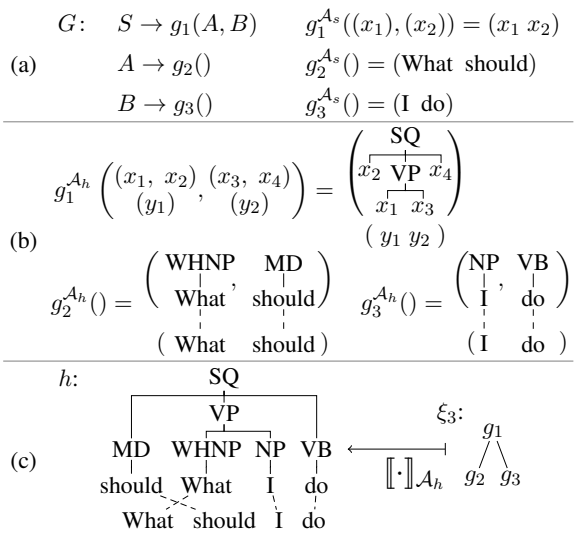


Figure 5: (a,b): An LCFRS/sDCP-hybrid grammar $(G, \mathcal{A}_s, \mathcal{A}_h)$ in IRTG notation. (b): $\mathcal{A}_h$ interprets each function symbol g_i by a function on synchronized tuples over trees (top) and tuples over strings (bottom). (c): Interpretation of ξ_3 in $\mathcal{A}_h$ to a hybrid tree h.

5 Experimental evaluation

We implemented[2] the generic split/merge algorithm and the parsing objectives in `C++` with bindings to `python3`. We evaluate it with LCFRSs and hybrid grammars for discontinuous phrase structure parsing. We use the TiGer corpus (Brants et al., 2004) and employ the split of the SPMRL shared task (Seddah et al., 2014) (TiGerSPMRL) and the one of Hall and Nivre (2008) (TiGerHN08). TiGer contains ca. 50k annotated sentences of German news text, exhibits discontinuity, and is predominantly used for evaluation in recent literature on discontinuous parsing. Evaluation with other languages is subject of further research. Part-of-speech (POS) tags for the TiGerHN08 test set are predicted using the `MATE` tagger (Björkelund et al., 2010), which we trained on the training and development section of TigerHN08.

The base LCFRS is induced from the treebank following Maier and Søgaard (2008). For each rule we remember the grammatical function symbol of the left-hand side nonterminal to its parent in the algebra $\mathcal{A}_t$. We binarize the LCFRS either right-to-left ($\text{LCFRS}_{\text{r2}\ell}$) or head-outward ($\text{LCFRS}_{\text{ho}}$) (cf. Kallmeyer and Maier, 2013) and apply Markovization ($v = 1$, $h = 1$). The base LCFRS/sDCP-hybrid grammar is induced according to a modified version of the algorithm by Nederhof and Vogler (2014) where we include only the POS tag in the sDCP component of a lexical rule but no additional unary categories. Also, we include syntactic function labels into the rules' sDCP component. We restrict the fanout to 2 and use strict and child labeling (abbreviated $\text{hybrid}_{\text{strict}}$ and $\text{hybrid}_{\text{child}}$, respectively). The numbers of nonterminals and all/lexical rules of either grammar, and the coverage of trees from the development set are given in the table on the right for TiGerHN08.

	nont.	rules / lex.	cover.
LCFRS_{ho}	767	50,153 / 28,080	78.3%
$\text{LCFRS}_{\text{r2}\ell}$	817	49,297 / 28,080	76.5%
$\text{hybrid}_{\text{child}}$	288	39,123 / 28,080	82.9%
$\text{hybrid}_{\text{strict}}$	32,281	108,957 / 28,080	50.0%

The LCFRSs and hybrid grammars are refined by 5 and 4 split/merge cycles, respectively. We stop EM training if the likelihood of the covered trees in the development set decreases. Then we apply the grammars in some of the ways outlined in Section 3.4 for parsing unseen sentences. Each sentence is a tokenized string s over pairs of words and (gold) POS tags. For both LCFRSs and hybrid grammars we have to carry out an LCFRS parsing step on s, for which we employ the implementation[3] of van Cranenburgh et al. (2016, see Sec. 6.4). Precisely, a pruned version of the chart G_s is computed via a coarse-to-fine pipeline that utilizes a PCFG approximation of the probabilistic LCFRS. Once a best parse has been selected, we compare it to the gold tree by computing F1 and exact match (EM) for labeled brackets, F1 for discontinuous labeled brackets, and F1 including function tags (F1-fun) using *disco-dop*[3].

[2]The implementation is freely available at `https://github.com/kilian-gebhardt/panda-parser`.

[3]*disco-dop* can be obtained from `https://github.com/andreasvc/disco-dop`. For evaluation on TiGerHN08 we use the `proper.prm` parameters. For TiGerSPMRL we also report F1 with `spmrl.prm`.

Objective	F1 (disc)	EM	F1-fun	F1 (disc)	EM	F1-fun	F1 (disc)	EM	F1-fun	F1 (disc)	EM	F1-fun
	LCFRS head-outward			LCFRS right-to-left			$\text{hybrid}_{\text{child}}$			$\text{hybrid}_{\text{strict}}$		
base-Viterbi	68.29 (22.55)	28.21	41.73	70.36 (23.00)	30.06	43.15	63.19 (15.04)	23.89	39.22	69.86 (29.34)	29.63	43.24
fine-Viterbi	76.59 (29.01)	35.87	63.45	77.32 (30.94)	36.83	65.48	76.56 (29.66)	39.27	65.03	73.34 (34.47)	33.95	61.06
variational	79.09 (33.17)	41.30	67.23	79.04 (34.32)	40.85	68.74	77.48 (30.53)	40.79	66.96	73.94 (33.75)	35.28	62.34
max-rule-prod.	**79.44** (33.74)	**41.73**	67.51	79.21 (**34.54**)	40.95	**68.83**	77.69 (30.45)	41.18	67.05	73.99 (34.02)	35.48	62.37
base-500-rerank	74.09 (29.31)	36.77	55.65	74.52 (28.82)	36.49	56.15	69.30 (25.61)	31.82	52.16	72.53 (32.98)	33.68	55.41

Table 1: Results on the TiGerHN08 development set with gold POS tags for sentences up to length 40.

The results on the TiGerHN08 development set are depicted in Table 1. The refined grammars notably improve over the respective base grammars by up to 14.50 points in F1. If we fix a base grammar but alter the parsing objectives, then we observe the following variations in F1 and EM: Reranking the 500-best derivations of the base grammar gives worse results than the Viterbi objective on the refined grammar. The weight projection approaches again improve over the Viterbi objective by up to 2.85 and 1.13 points in F1 for LCFRSs and hybrid grammars, respectively, where max-rule-product is consistently superior to variational. We do not observe an instance where max-rule-product is ill-defined in our experiments. Figure 6 shows that the F1 decreases for longer sentences by the example of LCFRS_{ho}/max-rule-product. $\text{LCFRS}_{\text{r2}\ell}$ is superior to LCFRS_{ho} with the Viterbi objective but the opposite holds for the projection-based objectives. $\text{Hybrid}_{\text{strict}}$, which suffers from 165 parse failures, produces worse results than $\text{hybrid}_{\text{child}}$ (12 parse failures) except for the reranking objective. For the discontinuous F1 and the F1-fun we find that $\text{LCFRS}_{\text{r2}\ell}$ performs better than LCFRS_{ho} in all cases but one. Also $\text{hybrid}_{\text{strict}}$ outperforms $\text{hybrid}_{\text{child}}$ with respect to discontinuous F1 but not for F1-fun. Overall, LCFRSs outperform hybrid grammars in terms of F1. The best F1 of 79.44 and 77.69 are obtained by LCFRS_{ho} and $\text{hybrid}_{\text{child}}$, respectively, with max-rule-product.

We parse the test set with LCFRS_{ho} and $\text{hybrid}_{\text{child}}$ using the max-rule-product objective. We present the results and compare to other discontinuous constituent parsers of the recent literature in Table 2. Most of these parsers are discriminative. The parsers by Hall and Nivre (2008), Fernández-González and Martins (2015), and Corro et al. (2017) employ different forms of dependency representation which are converted into constituent structures (dep2const). In contrast, Maier (2015), Maier and Lichte (2016), Coavoux and Crabbé (2017), and Stanojević and Garrido Alhama (2017) employ transition systems (SR-swap, SR-gap, SR-adj-gap) that can produce discontinuous constituent structures directly. Lastly, the chart-based parser of van Cranenburgh et al. (2016) is a generative model that enhances LCFRSs to discontinuous tree substitution grammars where tree fragments are learned according to the *data-oriented parsing* (DOP) paradigm. The results on the HN08 test set are close to the one on the development set. The F1 on the SPMRL test set is more than 4 points lower than in the HN08 split. Other parsers exhibit the same phenomenon that is probably caused by a shift in the distribution to longer sentences. Using predicted POS tags decreases the F1 by 2.38 and 2.00 points for the LCFRS and the hybrid grammar, respectively.

Discussion. The results indicate a strong influence of the granularity of the base grammar's nonterminals. A low granularity results in a higher coverage but also decreases the performance of the base grammar. For instance, for hybrid grammars we see that, despite many parse failures, strict labeling outperforms child labeling with the reranking objective. The drastically lower scores of the reranking objective for $\text{LCFRS}_{\text{r2}\ell}$, LCFRS_{ho} and $\text{hybrid}_{\text{child}}$ in light of the small difference with $\text{hybrid}_{\text{strict}}$ are likely caused by the base grammars being too coarse and, thus, assigning higher probabilities to bad candidates. Moreover, we suppose that including some context in the base grammars' nonterminals helps to guide the split/merge algorithm and avoids overfitting: For $\text{hybrid}_{\text{child}}$ the accuracy drops if we run more than 4 split/merge cycles. Also, in early experiments we observed worse performance if the conditioning context of Markovization for LCFRSs is further restricted. It will be interesting to study hybrid grammars whose base grammars have slightly finer nonterminals than with child labeling.

The EM algorithm is prone to overfitting to the training corpus. In fact, in our experiments we observed that the validation likelihood decreased after some epochs of training whereas the smoothing step counteracted this trend. To improve robustness, we plan to investigate changes in the training regime,

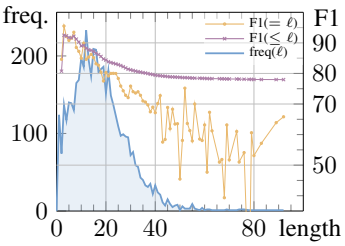


Figure 6: Frequency of sentence length ℓ, F1 for sentences of length ℓ, and F1 of sentences of length $\leq \ell$ on TiGerHN08 dev. set with LCFRS_{ho}/max-rule-product.

	Method	TiGerSPMRL F1 `spmrl/proper`	TiGerHN08 $\ell \leq 40$ F1	EM	F1-fun
Hall and Nivre (2008)	dep2const	- / -	79.93	-	-
Fernández-González and Martins (2015)	dep2const	80.62 / -	**85.53**	-	-
Corro et al. (2017)	dep2const	- / **81.63**	-	-	-
Maier (2015)	SR-swap	- / -	79.52	44.32	-
Maier and Lichte (2016)	SR-swap	- / 76.46	80.02	45.11	-
Coavoux and Crabbé (2017)	SR-gap	**81.50** / **81.60**	85.11	-	-
Stanojević and Garrido Alhama (2017)	SR-adj-swap	- / **81.64**	84.06	-	-
here	LCFRS: head-outward/max-rule-product	75.00 / 75.08	79.29	42.55	67.25
here	hybrid grammar: child/max-rule-product	72.91 / 72.98	77.68	41.28	66.72
† van Cranenburgh et al. (2016)	DOP	- / -	78.2	40.0	68.1
† **here**	LCFRS: head-outward/max-rule-product	- / -	76.91	39.22	64.91
† **here**	hybrid grammar: child/max-rule-product	- / -	75.66	38.40	64.66

Table 2: Evaluation on the test sets. Rows with † use predicted POS tags.

e.g., adding a prior on probability assignments or merging a higher percentage of nonterminal splits.

It is not surprising that the best results are obtained with the projection-based parsing objectives: the loss function of EM training does not guarantee that the probability mass of one derivation is concentrated in a single run. That max-rule-product outperforms the variational approach in terms of F1 and EM is in accordance with the findings and interpretations of Petrov and Klein (2007). For hybrid grammars the improvements of the projection-based methods are smaller than for LCFRSs which may be explained by the additional layer of spurious ambiguity (i.e., multiple derivation trees for one hybrid tree) they exhibit.

The F1 on discontinuous brackets is much lower than the overall F1 where the discontinuous recall is particularly low. Each grammar predicts only between 631 and 989 discontinuous brackets where there are 1837 in the gold standard. This could be affected by the way we approximate the PCFG in the coarse-to-fine pipeline which penalizes discontinuous rules. Most discontinuous brackets are predicted with the reranking objective but, as the lower discontinuous F1 indicates, these brackets are often incorrect.

Table 2 shows that the systems proposed in the literature exhibit a higher F1 than our grammars. This holds in particular for the systems that employ discriminative (and sometimes global) features, which are not available in our system. On the other hand, also van Cranenburgh et al. (2016)'s DOP-based parser is superior to our system in the predicted POS tag scenario. One reason might be error propagation due to wrong POS tags predicted by the external tagger that we use. In contrast, in van Cranenburgh et al. (2016) POS tags are predicted during parsing. Also, the probability assignment that we obtained by EM training may be less robust than the rule probabilities obtained according to the DOP paradigm.

Conclusion. The state refinement method considerably improves over the baseline grammars, which confirms our hypothesis that the usefulness of the split/merge algorithm goes beyond parsing with CFGs. However, at least with the used version of EM training, the initial granularity of the nonterminal set has a decisive impact on the quality of the resulting grammar and should be chosen carefully. When considering the task of discontinuous parsing, the results fall behind recent advances in the literature. This holds in particular for the discriminative deterministic transition-based systems where the representable discontinuity is not restricted by grammar constants, non-local features may be considered, and the parsing is much faster. One remedy to the lower accuracy and speed could be the enhancement of our chart-based method with a discriminative classifier to guide the pruning of the chart (Vieira and Eisner, 2017). In the future one may advance the understanding of the split/merge algorithm by applying it to other IRTGs such as synchronous hyperedge replacement grammars for graph parsing (Peng et al., 2015) or in a task different from parsing like syntax-based machine translation with synchronous grammars (Chiang, 2007).

Acknowledgements

The author thanks the anonymous reviewers for many helpful comments, Markus Teichmann for contributing to the implementation, Heiko Vogler and Tobias Denkinger for feedback on a draft of this paper, and Andreas van Cranenburgh and Toni Dietze for helpful discussions. The implementation utilizes *disco-dop* by Andreas van Cranenburgh, *treetools* by Wolfgang Maier, and the *Eigen* template library.

References

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Coling 2010: Demonstrations*. Beijing, China, pages 33–36. https://www.aclweb.org/anthology/C10-3009.

Walter S. Brainerd. 1969. Tree generating regular systems. *Information and Control* 14(2):217 – 231. https://doi.org/10.1016/S0019-9958(69)90065-5.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. TIGER: Linguistic interpretation of a German corpus. *Research on Language and Computation* 2(4):597–620. https://doi.org/10.1007/s11168-004-7431-3.

Matthias Büchse, Alexander Koller, and Heiko Vogler. 2013. General binarization for parsing and translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria, pages 145–154. https://www.aclweb.org/anthology/P13-1015.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics* 33(2):201–228. https://doi.org/10.1162/coli.2007.33.2.201.

Maximin Coavoux and Benoit Crabbé. 2017. Incremental discontinuous phrase structure parsing with the gap transition. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain, pages 1259–1270. https://www.aclweb.org/anthology/E17-1118.

Michael John Collins. 1999. Head-driven Statistical Models for Natural Language Parsing. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA. http://www.cs.columbia.edu/∼mcollins/papers/thesis.ps.

Caio Corro, Joseph Le Roux, and Mathieu Lacroix. 2017. Efficient discontinuous phrase-structure parsing via the generalized maximum spanning arborescence. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark, pages 1644–1654. https://www.aclweb.org/anthology/D17-1172.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* 39(1):1–38. https://www.jstor.org/stable/2984875.

Pierre Deransart and Jan Małuszynski. 1985. Relating logic programs and attribute grammars. *Journal of Logic Programming* 2(2):119–155. https://doi.org/10.1016/0743-1066(85)90015-9.

Frank Drewes, Kilian Gebhardt, and Heiko Vogler. 2016. EM-training for weighted aligned hypergraph bimorphisms. In *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata*. Berlin, Germany, pages 60–69. https://www.aclweb.org/anthology/W16-2407.

Kilian Evang and Laura Kallmeyer. 2011. PLCFRS parsing of English discontinuous constituents. In *Proceedings of the 12th International Conference on Parsing Technologies*. Dublin, Ireland, pages 104–116. https://www.aclweb.org/anthology/W11-2913.

Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, pages 1523–1533. https://www.aclweb.org/anthology/P15-1147.

Francis Ferraro, Benjamin Van Durme, and Matt Post. 2012. Toward tree substitution grammars with latent annotations. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*. Montreal, Canada, pages 23–30. https://www.aclweb.org/anthology/W12-1904.

Kilian Gebhardt, Mark-Jan Nederhof, and Heiko Vogler. 2017. Hybrid grammars for parsing of discontinuous phrase structures and non-projective dependency structures. *Computational Linguistics* 43(3):465–520. https://doi.org/10.1162/COLI_a_00291.

Jonas Groschwitz, Alexander Koller, and Mark Johnson. 2016. Efficient techniques for parsing with tree automata. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 2042–2051. https://www.aclweb.org/anthology/P16-1192.

Johan Hall and Joakim Nivre. 2008. Parsing discontinuous phrase structure with grammatical functions. In Bengt Nordström and Aarne Ranta, editors, *Advances in Natural Language Processing*. Springer Berlin Heidelberg, Berlin, Heidelberg, pages 169–180. https://doi.org/10.1007/978-3-540-85287-2_17.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*. Vancouver, British Columbia, Canada, pages 53–64. https://www.aclweb.org/anthology/W05-1506.

Laura Kallmeyer and Wolfgang Maier. 2013. Data-driven parsing using probabilistic linear context-free rewriting systems. *Computational Linguistics* 39(1):87–119. https://doi.org/10.1162/COLI_a_00136.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*. Sapporo, Japan, pages 423–430. https://www.aclweb.org/anthology/P03-1054.

Donald E. Knuth. 1977. A generalization of Dijkstra's algorithm. *Information Processing Letters* 6(1):1 – 5. https://doi.org/10.1016/0020-0190(77)90002-3.

Alexander Koller and Marco Kuhlmann. 2011. A generalized view on parsing and translation. In *Proceedings of the 12th International Conference on Parsing Technologies*. Dublin, Ireland, pages 2–13. https://www.aclweb.org/anthology/W11-2902.

Karim Lari and Steve J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech & Language* 4(1):35 – 56. https://doi.org/10.1016/0885-2308(90)90022-X.

Wolfgang Maier. 2015. Discontinuous incremental shift-reduce parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, pages 1202–1212. https://www.aclweb.org/anthology/P15-1116.

Wolfgang Maier and Timm Lichte. 2016. Discontinuous parsing with continuous trees. In *Proceedings of the Workshop on Discontinuous Structures in Natural Language Processing*. San Diego, California, pages 47–57. https://www.aclweb.org/anthology/W16-0906.

Wolfgang Maier and Anders Søgaard. 2008. Treebanks and mild context-sensitivity. In *Proceedings of the 13th Conference on Formal Grammar (FG-2008)*. CSLI Publications, Hamburg, Germany, pages 61–76. https://cslipublications.stanford.edu/FG/2008/maier.pdf.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Ann Arbor, Michigan, pages 75–82. https://doi.org/10.3115/1219840.1219850.

Mark-Jan Nederhof. 2003. Weighted deductive parsing and Knuth's algorithm. *Computational Linguistics* 29(1):135–143. https://doi.org/10.1162/089120103321337467.

Mark-Jan Nederhof and Giorgio Satta. 2004. Kullback-Leibler distance between probabilistic context-free grammars and probabilistic finite automata. In *Proceedings of the 20th International Conference on Computational Linguistics*. Geneva, Switzerland. https://doi.org/10.3115/1220355.1220366.

Mark-Jan Nederhof and Heiko Vogler. 2014. Hybrid grammars for discontinuous parsing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland, pages 1370–1381. https://www.aclweb.org/anthology/C14-1130.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for AMR parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. Beijing, China, pages 32–41. https://www.aclweb.org/anthology/K15-1004.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. Sydney, Australia, pages 433–440. https://doi.org/10.3115/1220175.1220230.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. Rochester, New York, pages 404–411. https://www.aclweb.org/anthology/N07-1051.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the SPMRL 2014 shared task on parsing morphologically-rich languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*. Dublin, Ireland, pages 103–109. https://www.aclweb.org/anthology/W14-6111.

H. Seki, T. Matsumura, M. Fujii, and T. Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science* 88(2):191–229. https://doi.org/10.1016/0304-3975(91)90374-B.

Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. 2012. Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Jeju Island, Korea, pages 440–448. https://www.aclweb.org/anthology/P12-1046.

Miloš Stanojević and Raquel Garrido Alhama. 2017. Neural discontinuous constituency parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark, pages 1666–1676. https://www.aclweb.org/anthology/D17-1174.

Christoph Teichmann, Alexander Koller, and Jonas Groschwitz. 2017. Coarse-to-fine parsing for expressive grammar formalisms. In *Proceedings of the 15th International Conference on Parsing Technologies (IWPT)*. Pisa, Italy, pages 122–127. https://www.aclweb.org/anthology/W17-6317.

Christoph Teichmann, Kasimir Wansing, and Alexander Koller. 2016. Adaptive importance sampling from finite state automata. In *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata*. Berlin, Germany, pages 11–20. https://www.aclweb.org/anthology/W16-2402.

Andreas van Cranenburgh, Remko Scha, and Rens Bod. 2016. Data-oriented parsing with discontinuous constituents and function tags. *Journal of Language Modelling* 4(1):57–111. https://doi.org/10.15398/jlm.v4i1.100.

Tim Vieira and Jason Eisner. 2017. Learning to prune: Exploring the frontier of fast and accurate parsing. *Transactions of the Association for Computational Linguistics* 5:263–278. https://aclweb.org/anthology/Q17-1019.

Krishnamurti Vijay-Shanker, David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*. Stanford, California, USA, pages 104–111. https://doi.org/10.3115/981175.981190.

A Appendix

In this appendix we provide auxiliary calculations and definitions, an example in which the max-rule-product parsing objective is ill-defined, and choices of hyperparameters and preprocessing steps used during the experiments.

A.1 Computing the probability of an object using its chart

In Section 2.2 we state that $P(a \mid \mathbb{G}, p) = \sum_{\xi \in L(G_a)} W_{(G_a, p\circ\varphi_a)}(\xi)$. This follows from:

$$\begin{aligned}
\sum_{\xi \in \mathrm{parses}_{\mathbb{G}}(a)} W_{(G,p)}(\xi) &= \sum_{\xi \in \mathrm{parses}_{\mathbb{G}}(a)} \sum_{r \in \mathrm{runs}^v_G(\xi)} \prod_{\pi \in \mathrm{pos}(\xi)} p(\mathrm{rule}^\pi_r) \\
&= \sum_{\xi \in L(G_a)} \sum_{r' \in \mathrm{runs}^v_{G_a}(\xi)} \prod_{\pi \in \mathrm{pos}(\xi)} p(\mathrm{rule}^\pi_{\varphi_a \circ r'}) \\
&= \sum_{\xi \in L(G_a)} \sum_{r' \in \mathrm{runs}^v_{G_a}(\xi)} \prod_{\pi \in \mathrm{pos}(\xi)} (p \circ \varphi_a)(\mathrm{rule}^\pi_{r'}) \\
&= \sum_{\xi \in L(G_a)} W_{(G_a, p\circ\varphi_a)}(\xi)
\end{aligned}$$

A.2 Approximation of the likelihood loss.

Let G^c be the RTG at the beginning of a particular split/merge cycle and $G^f = \mu_{\mathrm{sp}}^{-1}(G^c)$ be the RTG after splitting. We describe how $\Delta(B_1, B_2)$ is computed for a pair B_1 and B_2 of nonterminals in G^f that are candidates for merging. Similar to Section 3.3, for each $a \in A$ we assume grammar morphisms $\mu'_{\mathrm{sp}} \colon G^f_a \to G^c_a$, $\varphi_a \colon G^c_a \to G^c$, and $\varphi'_a \colon G^f_a \to G^f$ which satisfy $\varphi_a \circ \mu'_{\mathrm{sp}} = \mu_{\mathrm{sp}} \circ \varphi'_a$.

Firstly, we compute *merge factors* p_1 and p_2 based on the relative frequency of B_1 and B_2 where

$$p_i = \frac{\mathrm{freq}(B_i)}{\mathrm{freq}(B_1) + \mathrm{freq}(B_2)} \ .$$

To compute the expected frequency of B_i ($i \in \{1, 2\}$), we sum over the expected frequency of each occurrence B'_i of B_i in some chart G^f_a:

$$\mathrm{freq}(B_i) = \sum_{a \in A} c_A(a) \cdot \sum_{B'_i \in {\varphi'_a}^{-1}(B_i)} \frac{\alpha(B'_i) \cdot \beta(B'_i)}{\beta(S_{G^f_a})} \ .$$

Secondly, we consider pairs (B'_1, B'_2) of occurrences of B_1 and B_2 in some chart G^f_a such that $\mu'_{\mathrm{sp}}(B'_1) = \mu'_{\mathrm{sp}}(B'_2)$. For each such pair, we introduce a hypothetical nonterminal B' which symbolizes merging just B'_1 and B'_2. Its inside and outside weight is obtained by $\alpha(B') = \alpha(B'_1) + \beta(B'_2)$ and $\beta(B') = p_1 \cdot \beta(B'_1) + p_2 \cdot \beta(B'_2)$, respectively.

Using these hypothetical nonterminals, we approximate the loss in likelihood due to merging B_1 and B_2 by accumulating likelihood losses due to merging pairs of occurrences:

$$\Delta(B_1, B_2) = \prod_{\substack{a \in A \\ (B'_1, B'_2) \text{ in } G^f_a}} \left(\frac{\beta(S_{G^f_a}) \ + \ \alpha(B') \cdot \beta(B') - \left(\sum_{i \in \{1,2\}} \alpha(B'_i) \cdot \beta(B'_i) \right)}{\beta(S_{G^f_a})} \right)^{c_A(a)} .$$

In the above formula, the numerator shall express the probability of the chart after merging and the denominator expresses the probability before merging. If a nonterminal A' occurs more than once in a run of the chart, then the sum of the probability of all runs that contain A' is not equal to its expected frequency $\alpha(A') \cdot \beta(A')$. Thus, the above formula is reasonable under the assumption that each nonterminal A' in N_{G_a} occurs at most once in a run on a derivation tree in $L(G^f_a)$. This assumption is violated if, e.g., the chart contains a chain rule $A' \to f(A')$.

A.3 The max-rule-product objective can be ill-defined

We give an example of a CFG with chain rules where max-rule-product is an ill-defined parsing objective. Consider the refined CFG G^f with nonterminals $\{S, A_1, A_2, A_3\}$, terminals $\{a\}$, and rules with probabilities:

$$\begin{aligned} S \to A_i \quad & \#1.0 \text{ if } i = 1 \quad \text{else } 0.0 \\ A_i \to A_j \quad & \#1.0 \text{ if } j = i + 1 \text{ else } 0.0 \\ A_i \to a \quad & \#1.0 \text{ if } i = 3 \quad \text{else } 0.0 \end{aligned}$$

Consider the chart G^f_a for the string a which is isomorphic to G^f. The inside and outside weight of each nonterminal of G^f_a is 1.0. If we merge A_1, A_2, and A_3 to A and project weights according to the max-rule-product principle, then we obtain the CFG G^c_a with weights

$$\begin{aligned} S \to A \ \ & \#1.0 \\ A \to A \ \ & \#2.0 \\ A \to a \ \ & \#1.0 \end{aligned}$$

Now the weight of some parse tree of G^c_a is exponential in the number of occurrences of the chain rules $A \to A$ it contains. Consequently, there cannot be a best tree.

A.4 Hyperparameters and preprocessing

The TiGer corpus is preprocessed before grammar induction. Specifically, punctuation tokens are attached to lower nodes in order to reduce the number of discontinuous brackets using *disco-dop*. Also, we replace words with less than 4 occurrences in the training corpus by a fresh "UNKNOWN" symbol. In addition, the training set and the validation set are enriched by copies of the constituent trees where every word is replaced by the "UNKNOWN" symbol. These copies get assigned frequency 0.1 in either set.

During the split/merge algorithm EM training is applied as follows. The probability assignment p_i of the i-th epoch ($m \leq i \leq 20$) with the best validation likelihood is selected as result. We set $m = 2$ after smoothing and $m = 5$ otherwise. Also, we skip all remaining EM epochs, if validation likelihood dropped in 6 consecutive epochs. When computing validation likelihood, we ignore trees with probability 0 except after smoothing.

Double Path Networks for Sequence to Sequence Learning

[1]Kaitao Song, [2]Xu Tan, [3]Di He, [1]Jianfeng Lu, [2]Tao Qin and [2]Tie-Yan Liu
[1]Nanjing University of Science and Technology
[2]Microsoft Research
[3]Key Laboratory of Machine Perception, MOE, School of EECS, Peking University
{kt.song, lujf}@njust.edu.cn
{xuta, taoqin, tyliu}@microsoft.com
{di_he}@pku.edu.cn

Abstract

Encoder-decoder based Sequence to Sequence learning (S2S) has made remarkable progress in recent years. Different network architectures have been used in the encoder/decoder. Among them, Convolutional Neural Networks (CNN) and Self Attention Networks (SAN) are the prominent ones. The two architectures achieve similar performances but use very different ways to encode and decode context: CNN use convolutional layers to focus on the local connectivity of the sequence, while SAN uses self-attention layers to focus on global semantics. In this work we propose Double Path Networks for Sequence to Sequence learning (DPN-S2S), which leverage the advantages of both models by using double path information fusion. During the encoding step, we develop a double path architecture to maintain the information coming from different paths with convolutional layers and self-attention layers separately. To effectively use the encoded context, we develop a cross attention module with gating and use it to automatically pick up the information needed during the decoding step. By deeply integrating the two paths with cross attention, both types of information are combined and well exploited. Experiments show that our proposed method can significantly improve the performance of sequence to sequence learning over state-of-the-art systems.

1 Introduction

Sequence to Sequence learning (S2S) (Cho et al., 2014; Sutskever et al., 2014) is one of the challenging tasks in artificial intelligence, which covers machine translation, document summarization and question answering, etc. It has attracted more and more attention in recent years due to the success of deep learning. Sequence to sequence learning is usually developed based on a general encoder-decoder framework: The encoder reads the source sequence and generates a set of representations. After that, the decoder estimates the conditional probability of each target token given the source representations and its preceding tokens.

Different network architectures for sequence to sequence modelling have been designed based on this framework. Long Short-Term Memory (LSTM) based model (Cho et al., 2014) is the most popular. In the LSTM based model, the tokens are encoded/decoded using LSTM units, which can effectively summarize the temporal information of the sentences in source/target domains. However, because of its recurrent nature, the LSTM based model is difficult to parallel, so the training efficiency becomes the major challenge. Recently, Convolutional Neural Network (CNN) based (Gehring et al., 2017a) and Self Attention Network (SAN) based (Vaswani et al., 2017) models have been proposed, which achieved significant improvements in accuracy. Both models replace the recurrent structure with networks that are easy to parallelize, leading to training time acceleration.

CNN based and SAN based sequence to sequence models encode and decode information in very different ways. CNN based model treats the sequences similar with images: The convolutional layer is used to summarize information for a local region of the sequence, which corresponds to a subsequence

This work was done when K. Song was an Intern at Microsoft Research

Proceedings of the 27th International Conference on Computational Linguistics, pages 3064–3074
Santa Fe, New Mexico, USA, August 20-26, 2018.

of tokens. Different from CNN based model, SAN based model uses self-attention layer to abstract the context based on semantic meanings: In each layer, for each token, the similarity between the token and all other tokens in the sequence are computed. Then the contexts are weighted combined based on the global semantic similarity. By stacking multiple convolutional layers or self-attention layers, the two models obtain the whole contextual information during the encoding and decoding step.

As we can see, the convolutional layer focuses more on abstracting information according to local connectivity, while self-attention layer focuses more on using global semantics. Both models achieve similar accuracy on several sequence to sequence learning tasks. Then a natural question raises: is there any way to integrate the two methods together to achieve a higher accuracy? This is exactly the purpose of the paper.

In this work, we investigate the possibility to combine the two models into a unique framework and develop efficient architecture to leverage both advantages. In particular, we propose Double Path Networks for Sequence to Sequence learning (DPN-S2S), which contain a convolutional path and a self-attention path with attention information fusion between the encoder and the decoder. In the encoding step, we use the convolutional path and self-attention path independently. Thus we can summarize the context of the source sequences from different aspects. In the decoding step, we develop a cross attention module with gating to automatically select appropriate context, and use the context in a similar double path structure. As there is no recurrent structure in convolutional layers and self-attention layers, the model is still easy to parallelize. Since we abstract the hidden representation from different ways, the contexts we obtain are more diverse, making it possible to achieve higher accuracy under the same parameter number.

Experimental results show that our proposed approach can achieve significant improvements than single path based S2S framework. The results are summarized as follows:

- With a similar number of parameters, our model outperforms CNN based, SAN based model. More specifically, on Nist Chinese-English translation task, we outperform the strong CNN based baseline by 0.46-2.96 BLEU, SAN based baseline by 0.74-0.99 BLEU respectively in different Nist test sets. Furthermore, our model also achieves state-of-art BLEU/ROUGE score on IWSLT14 German-English translation task and English Gigaword text summarization task.

- Through ablation studies and analyses we show that both the CNN and SAN path are important in the encoder and the decoder. Removing any part would cause performance drop, which demonstrates the effectiveness of our model.

2 Background

2.1 Encoder-Decoder Framework

Denote $x = (x_1, x_2, ..., x_m)$ and $y = (y_1, y_2, ..., y_n)$ as the sentence pair, in which x_i and y_t are the i-th and t-th tokens for sequences x and y, m and n are the lengths of the sequences. The sequence to sequence model learns to estimate the conditional probability $P(y|x)$. Denote the model parameter as θ and denote the training corpus as S. One of the most popularly used objective functions is the log likelihood function:

$$L(\theta; S) = \frac{1}{|S|} \sum_{(x,y)\in S} \log P(y|x; \theta). \tag{1}$$

According to the probability chain rule, we have:

$$P(y|x; \theta) = \prod_{t=1}^{n} P(y_t|y_{<t}, x; \theta), \tag{2}$$

where $y_{<t}$ is the sequence of proceeding tokens before position t. To model the conditional probability, an encoder-decoder framework is usually employed (Cho et al., 2014; Bahdanau et al., 2015). In the encoding phase, $(h_1, h_2, \cdots, h_m)$ is learnt as the representations of the source sequence:

$$(h_1, h_2, \cdots, h_m) = f(x_1, x_2, ..., x_m). \tag{3}$$

where $f(\cdot)$ can be implemented as RNN, CNN or SAN. During the decoding phase, the decoder generates the target token y_t at position t using the previous generated tokens $y_{<t}$ and the source representations $(h_1, h_2, \cdots, h_m)$.

An attention mechanism is usually adopted to choose the places in the source representations to focus on. There are several types of attention used in the literature, such as dot, $concat$ and $general$ (Luong et al., 2015; Bahdanau et al., 2015). In this paper, we mainly adopt the dot type which is formulated as the q-k-v form:

$$Attention(q, k, v) = softmax(qk^T)v, \tag{4}$$

where $softmax$ function computes the attention coefficient which evaluates how well the source representations k matches the target query q. q is usually the hidden state in the current decoder layer and k, v are usually the same as source representation h.

2.2 CNN based Encoder-Decoder

CNN based Encoder-Decoder framework (Gehring et al., 2017b) typically uses one-dimensional convolution in each layer to perform nonlinear transformation. Denote h_i^l with dimension d as the i-th hidden state in the l-th CNN layer and set $h_i^0 = e_i$, where e_i is the input embedding of the sequence. The hidden state h_i^l is computed by the convolution operation:

$$h_i^l = v([h_{i-r/2}^{l-1}, ..., h_{i+r/2}^{l-1}]W^l + b_w^l) + h_i^{l-1}, \tag{5}$$

where $W^l \in \mathcal{R}^{rd \times 2d}$ is the convolution filter with kernel size r, $b_w^l \in \mathcal{R}^{2d}$ is the bias and d is the hidden dimension. The convolution input is the concatenation of r elements in the lower layer with hidden dimension d. We use Gated Linear Units (GLU) (Dauphin et al., 2017) as the activation function v, with $2d$ input vector and d output vector.

2.3 SAN based Encoder-Decoder

SAN system is first proposed by Vaswani et al. (2017). Each single self-attention layer has two sublayers: a multi-head self-attention layer and a feed forward network. Both sublayers are stacked using residual connection (He et al., 2016) and layer normalization (Ba et al., 2016). The multi-head self-attention layer $MH(\cdot)$ is formulated as follows:

$$\begin{aligned} MH(q, k, v) &= Concat_{i=1}^{s} h_i(q, k, v), \\ h_i(q, k, v) &= Att(\frac{qW_i^q}{\sqrt{d_s}}, kW_i^k, vW_i^v). \end{aligned} \tag{6}$$

Each head uses weights W_i^q, W_i^k and W_i^v to linearly transform the input q, k, v and then performs $Att(\cdot)$ which is equal to Equation 4. W_i^q, W_i^k and $W_i^v \in \mathcal{R}^{d \times d_s}$, where d_s is a scale factor, which equal to d/s, d is the hidden size of q, and s is the number of heads.

For the feed forward network, we use a simple two-layer, fully connected network with ReLU activation function in the middle layer:

$$FeedForward(x) = f_2(Max(0, f_1(x))), \tag{7}$$

where $f_1(\cdot)$ and $f_2(\cdot)$ are both feed forward network. The feed forward network applies the non-linear transformation on each position identically and separately.

3 Double Path Networks

In our work, we introduce double path networks which incorporate CNN and SAN for sequence to sequence learning. A detailed model structure is shown in Figure 1.

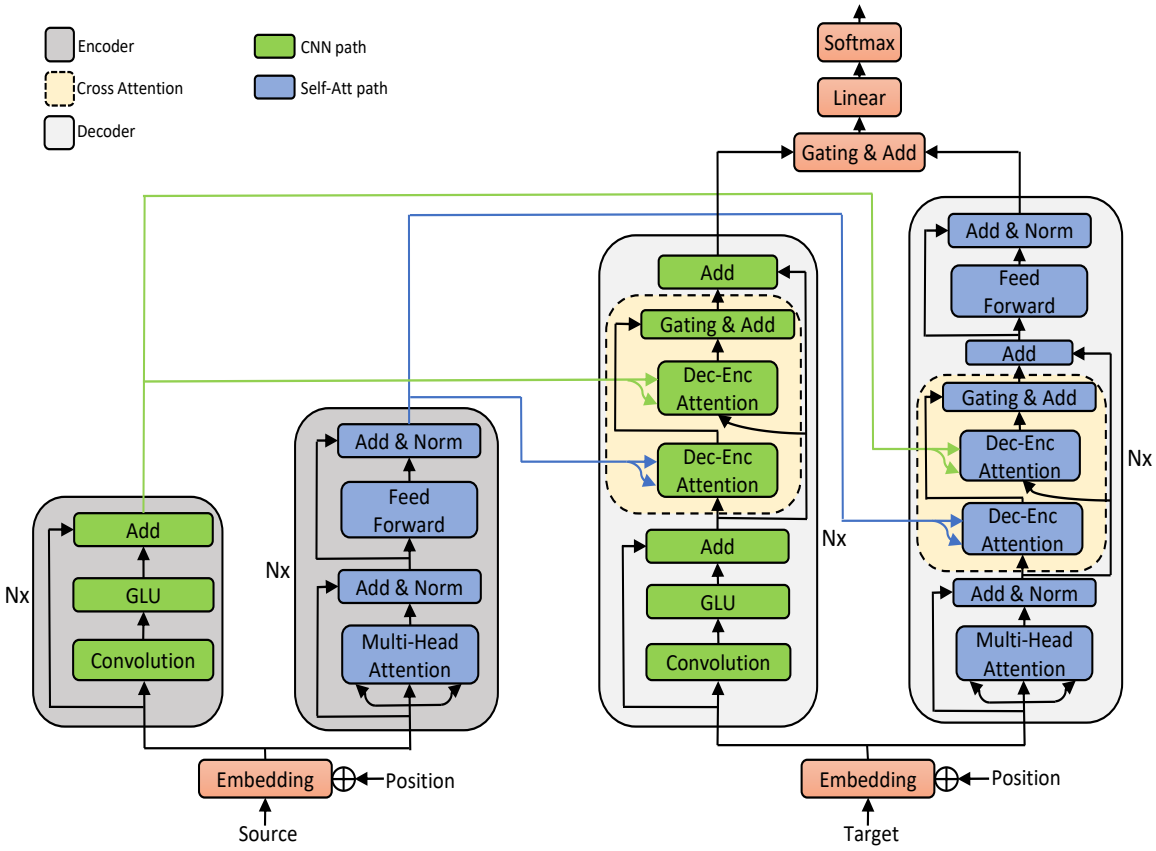


Figure 1: DPN-S2S architecture. The CNN and SAN path are depicted with green and blue module, respectively. The encoder part is colored with dark grey, attention fusion part with light yellow, and decoder part with light grey. The blue and green line show the information flow from encoder to decoder.

3.1 Double Path Encoder

To leverage the advantage of CNN and SAN layer, we design an encoder which includes a CNN and SAN path, introduced in Section 2.2 and 2.3 respectively. We sum up the position embedding $\mathbf{p} = (p_1, ..., p_m)$ and word embedding $\mathbf{w} = (w_1, ..., w_m)$ as the input of encoder, to give the model a sense of the token's position in the sentence. For each path, we employ residual connections to stack multiple layers for deeper network. Consequently, we can collect two various context representations from encoder.

3.2 Cross Attention with Gating

Between the encoder and the decoder, we develop a cross attention module with gating fusion, to automatically pick up the information needed to decode target word. As shown in Figure 1, the attention fusion module consists of two submodules: $Dec - Enc\ Attention$ and $Gating\ \&\ Add$. We now give detailed descriptions of the two submodules.

Each path in the decoder take encoder's double-path outputs as context to make attention. Thus, there are four types of information flows through encoder to decoder. The $Dec - Enc\ Attention$ module in the decoder takes Eq. (4) to compute attention results, which is denoted as:

$$\begin{aligned} ctx^{cc} &= Attention(q^c, k^c, v^c), \quad ctx^{ca} = Attention(q^c, k^a, v^a), \\ ctx^{ac} &= Attention(q^a, k^c, v^c), \quad ctx^{aa} = Attention(q^a, k^a, v^a). \end{aligned} \tag{8}$$

For example, $ctx^{ca} \in \mathcal{R}^d$ means the attention result with attention query q^c from CNN path in the decoder and attention key k^a and value v^a from SAN path in the encoder.

In order to fully exploit the information captured by different encoder paths, we design a gating mechanism to fuse them for balancing information usage between local connectivity and global semantics.

This fusion gate is shown as the $Gating\ \&\ Add$ module in the right side of Figure 1, formulated as follows:

$$\begin{aligned} g^c &= sigmoid([ctx^{cc}, ctx^{ca}]W^c + b^c), \\ g^a &= sigmoid([ctx^{aa}, ctx^{ac}]W^a + b^a), \\ ctx^c &= ctx^{cc} \cdot (1 - g^c) + ctx^{ca} \cdot g^c, \\ ctx^a &= ctx^{aa} \cdot (1 - g^a) + ctx^{ac} \cdot g^a, \end{aligned} \tag{9}$$

where g^c and g^a are scalar attention gates for CNN context and SAN context; $[\cdot, \cdot]$ is an element concatenate operation; W^c and $W^a \in \mathcal{R}^{2d}$ are the weight matrices; b^c and b^a are the scalar biases of the attention gates; and ctx^c and ctx^a are the fusion context vectors for the current CNN layer and SAN layer. For the gate g^c and g^a, both the two attention results are used as inputs to the gating computation.

3.3 Double Path Decoder

The decoder is illustrated in the right side of Figure 1. It is similar to the encoder except for the decoder-to-encoder attention. Each layer in CNN and SAN contains two decoder-to-encoder attention modules, which is denoted in last subsection. We also need remove the future information in the convolution and self-attention operations in order to be consistent with decoding phase, for target sentence is generated sequentially, without any prior knowledge about next word.

We also apply a gating mechanism to combine the outputs of the two decoder paths and feed the combined result into the $softmax$ layer to generate next word:

$$\begin{aligned} g^o &= sigmoid([z^c, z^a]W^o + b^o), \\ z^o &= z^c \cdot (1 - g^o) + z^a \cdot g^o, \\ P(y) &= softmax(z^o W^s + b^s), \end{aligned} \tag{10}$$

where z^c and z^a are the outputs of CNN and SAN path, respectively. z^o is the fusion output and W^s, b^s are linear transform weights and biases before $softmax$.

4 Experiment Settings

In this section, we introduce the datasets used in experiments, model settings, and training details.

4.1 Datasets

We evaluate our proposed method on two neural machine translation tasks and one abstractive text summarization task. The datasets are described as follows:

IWSLT2014 German-English We use the dataset of IWSLT14 German-English machine translation track (Cettolo et al., 2014) for training and evaluation. We tokenize the data which come from TED and TEDx talks. After preprocessing, the dataset remains 160K training sentences and 7K development sentences[1]. We concatenate dev2010, tst2010, tst2011 and tst2012 as the test set. We consider a joint source and target byte-pair encoding (BPE) with 10K types (Sennrich et al., 2016).

Nist Chinese-English We use the corpus consisting of 1.25M sentence pairs extracted from LDC corpora [2]. This dataset contains 27.9M Chinese words and 34.5M English words respectively. We learn a 25K subwords dictionary based on BPE for source and target languages separately. After tokenization, we get a source vocabulary with 37K words and a target vocabulary with 25K words. We choose Nist-2003 as the development set, and Nist-{2004, 2005, 2006, 2008, 2012} as the test sets.

[1] https://github.com/facebookresearch/fairseq-py/blob/master/data/prepare-iwslt14.sh

[2] The corpora includes LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06.

Text Summarization We use the Gigaword corpus (Graff et al., 2003) and follow the processing as Rush (2015). We obtain 3.8M training samples and 190K validation samples. We evaluate our approach on Gigaword dataset, which is a widely used test set with 2000 article-title pairs. Similar to Gehring (2017a), we use source and target vocabulary both with 30K words. Follow Gehring (2017a), we require the length of generated sequence is at least 14 words.

4.2 Model Settings

IWSLT14 German-English For CNN path, we adopt 4 idential convolutional layers with 256 neurons in both the encoder and the decoder. As one self-attention layer is comprised of 2 sublayers, we set 2 layers for SAN to guarantee the depth consistency of network. The hidden size and the filter size of SAN block are set as 256 and 1024 respectively. The word embedding dimension is 256. And the dropout rate is set as 0.1. In order to demonstrate the efficiency of our method, we compare our method with four following baselines: (i) *Deeper CNN Model*, a standard CNN based model with 8 layers of 256 neurons. (ii) *Wider CNN Model*, a standard CNN based model with 4 layers of 512 neurons. (iii) *Wider SAN Model*, a standard SAN based model with 2 layers of 512 neurons. (iv) *Deeper SAN Model*, a standard SAN based model with 4 layers of 256 neurons. Besides, we also list some related works for comparison.

Nist Chinese-English We perform a deep model on Nist Chinese-English translation task. This model adopts 12 stacked layers for the CNN path and 6 stacked layers for SAN path in the encoder and decoder respectively. The hidden size of both CNN/SAN layers are both set as 256 and the filter size of SAN layer is 1024. We set 256 neurons for the word embedding. The dropout rate is set as 0.2. For the purpose of comparison, We choose two model settings as baselines: (i) *CNN Model*, a CNN based model with 12 stacked layers of 512 neurons (ii) *SAN Model*, a SAN model with 6 stacked layers of 512 neurons and the filter size of layer is 2048. These model settings contain approximately similar number of parameters in contrast to our model. We also list some previous works for comparison.

Text Summarization We follow the same model setting as described in IWSLT14 German-English translation task. For this task, We adopt the models including RNN MRT (Ayana et al., 2016), WFE (Suzuki and Nagata, 2017) and ConvS2S (Gehring et al., 2017a) as the baselines.

4.3 Training and Evaluation

We adopt Nesterov Accelerated Gradient (NAG) (Bengio et al., 2012) as training optimizer. The initial learning rate is set as 0.25 for IWSLT14 German-English and text summarization, and 0.50 for Nist Chinese-English, with a decay schedule that shrinking by 10 when the validation loss stops decreasing. Each training batch contains approximately 4000 source and target tokens. During inference, we set beam size as 5 for IWSLT14 German-English and text summarization, and 10 for Nist Chinese-English. All models are implemented in PyTorch based on *fairseq-py*[3]. We use one Nvidia Titan x Pascal GPU for IWSLT14 German-English and text summarization, and 4 GPUs for Nist Chinese-English.

	Params	BLEU
AC+LL (Bahdanau et al., 2016)	-	28.53
NPMT+LM (Huang et al., 2017)	-	29.16
Wider CNN	20.37M	30.63
Wider SAN	27.06M	31.29
Deeper CNN	13.82M	30.70
Deeper SAN	13.54M	31.43
DPN-S2S	11.57M	**31.99**

Table 1: IWSLT14 German-English translation performance.

5 Results

In this section, we report our results on translation and summarization tasks. In addition, we also conduct a series of extensive analyses for a better understanding of our model.

[3]https://github.com/facebookresearch/fairseq-py

	Nist04	Nist05	Nist06	Nist08	Nist12
MC-NMT (Xiong et al., 2018)	40.79	38.49	-	31.51	26.90
NMT + Distortion (Zhang et al., 2017)	40.52	36.81	35.77	-	-
SD-NMT (Chen et al., 2017)	39.81	36.74	34.63	28.61	-
SAN	40.39	40.38	38.06	31.67	30.32
CNN	40.80	38.16	37.89	30.70	29.55
DPN-S2S	**41.26**	**41.12**	**38.87**	**32.66**	**31.17**

Table 2: Performance of different systems on Nist Chinese-English translation task.

5.1 IWSLT2014 German-English

We evaluate the translation accuracy by BLEU [4] (Papineni et al., 2002). Table 1 shows the experiment results on German-English translation task. We list the results of the wider network, deeper network and some related works for a comparison. The results can be summarized as follow:

When compared with wider network, DPN-S2S achieves an improvement of 1.36 BLEU and 0.70 BLEU over CNN and SAN model. When compared with deeper network, DPN-S2S achieves an improvement of 1.29 BLEU and 0.56 BLEU over CNN and SAN model. Generally, we note that DPN-S2S can produce better performance than wider or deeper networks with fewer parameters. In addition, we choose two methods which used an actor-critic algorithm (Bahdanau et al., 2016) and a phrased-based model with an auxiliary language model (Huang et al., 2017) separately. We find our method also outperforms these approaches. All of these comparsions indicate the effectiveness of our method.

5.2 Nist Chinese-English

Table 2 lists the results about our model on each Nist Chinese-English test set, together with several NMT baselines. These baselines are RNNSearchs with some well acknowledged techniques including: 1) Using multi-channel information into encoder (MC-NMT) (Xiong et al., 2018); 2) Incorporating word reorder information into NMT (NMT+Distortion) (Zhang et al., 2017); 3) An attention mechanism with a directed-syntax (SD-NMT) (Chen et al., 2017). We can observe that our method surpasses SAN and CNN baseline by 0.74-0.99 and 0.46-2.96 BLEU score in different test sets. In addition, we also compare our results with some related works, and our model outperforms MC-NMT by 0.47-4.27 BLEU, NMT+Distortion by 0.74- 4.31 BLEU and SD-NMT by 1.44-4.38 BLEU in Nist test sets. These results on Nist Chinese-English dataset prove that our model can also achieve improvement in large dataset.

	Gigaword		
	RG-1 (F)	RG-2 (F)	RG-L (F)
RNN MRT (Ayana et al., 2016)	36.54	16.59	33.44
WFE (Suzuki and Nagata, 2017)	36.30	17.31	33.88
ConvS2S (Gehring et al., 2017a)	35.88	17.48	33.29
DPN-S2S	**36.92**	**17.91**	**34.32**

Table 3: Accuracy on Gigaword text summarization task in terms of ROUGE-1 (RG-1), ROUGE-2 (RG-2), and ROUGE-L (RG-L). F stands for F1-score.

5.3 Text Summarization

We employ three variants of ROUGE (Lin, 2004) as evaluation metrics for text summarization: ROUGE-1 (unigrams), ROUGE-2 (bigrams), and ROUGE-L (longest-common substring).

[4] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

We compare our model with some previous work including RNN MRT (Ayana et al., 2016) which adopts RNNSearch on this task, WFE (Suzuki and Nagata, 2017) which addresses the problem of repeated sequence generation, and ConvS2S (Gehring et al., 2017a). Table 3 shows the text summarization results on the Gigaword dataset. DPN-S2S outperforms all the three models by 0.43-1.04 ROUGE-1, 0.43-1.32 ROUGE-2 and 0.64-1.23 ROUGE-L in terms of F1 score.

5.4 Ablation Study

To understand to what extent each path in the encoder/decoder can affect the model performance, we conduct some experiments for ablation study. For conciseness we just choose one task (IWSLT14 German-English) for this study. Table 4 shows the experiment settings and results. All the models are of the same dimensions for hidden states and word embedding. Note that model M9 is our proposed DPN-S2S. We have the following observations.

ID	Encoder CNN	Encoder SAN	Decoder CNN	Decoder SAN	BLEU
M1	Δ		Δ		30.38
M2	Δ			Δ	31.00
M3	Δ		Δ	Δ	31.26
M4		Δ	Δ		29.80
M5		Δ		Δ	30.63
M6		Δ	Δ	Δ	31.11
M7	Δ	Δ	Δ		31.03
M8	Δ	Δ		Δ	31.43
M9	Δ	Δ	Δ	Δ	**31.99**

Table 4: Ablation studies of DPN-S2S on IWSLT14 German-English. Δ means the model contains this component.

- Comparing model M4 with M7, M5 with M8, and M6 with M9, we can see that adding the CNN encoder leads to 0.80-1.23 BLEU gain.
- Comparing model M2 with M3, M5 with M6, and M8 with M9, we observe that adding the CNN decoder leads to 0.26-0.56 BLEU gain.
- Comparing model M1 with M7, M2 with M8, and M3 with M9, we find that adding the SAN encoder results in 0.43-0.73 BLEU improvement.
- Similarly comparing model M1 with M3, M4 with M6, and M7 with M9, we see 0.88-1.31 BLEU gain by adding the SAN decoder.

These results suggest that both the CNN path and the SAN path make positive contributions to the overall architecture, and they are needed in both the encoder and decoder.

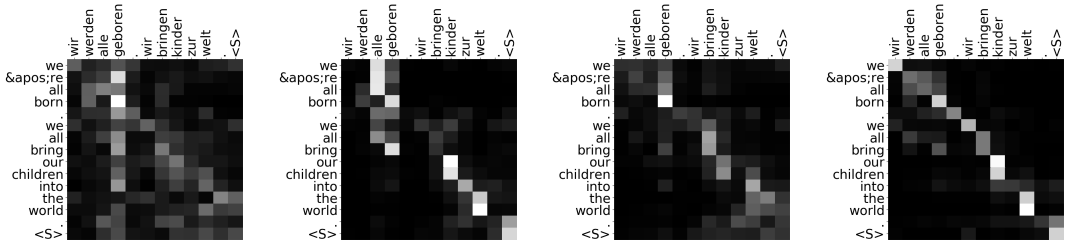
5.5 Attention Visualization

To demonstrate the different characteristics of the CNN and SAN path, we analyze the alignments (attention coefficients) from the four types of decoder-to-encoder attention. The alignments are computed by $Softmax(qk^T)$ from Eq. (4), and the alignments from each token in decoder form a distribution. We use the entropy of the distribution $-\sum_i x_i log x_i$ to depict to what extent the target token focuses on the source tokens. Small entropy means less uncertainty, i.e., the attention is more concentrated.

Encoder \ Decoder	CNN	SAN
CNN	2.208	1.406
SAN	1.886	1.377

Table 5: The entropy of the alignments of CNN/SAN decoder to CNN/SAN encoder on IWSLT14 German-English.

We choose the 6750 sentences in the German-English test set and calculate the entropy of each target token's alignments distribution, averaged by target token in the sentence and then averaged by sentence, as shown in Table 5. The entropy of alignments from both paths in decoder to CNN path in encoder (the first row of the table) is larger than that of SAN path in encoder. This is consistent with our intuitions: CNN focuses more on local information, and so the decoder needs to attend to multiple hidden states



(a) CNN to CNN (b) CNN to SAN (c) SAN to CNN (d) SAN to SAN

Figure 2: The alignments of CNN/SAN decoder to CNN/SAN encoder from DPN-S2S. The x-axis and y-axis of each plot correspond to the words in the source language and target language. Each pixel shows the alignment scores of the target word to source word.

	Translation
Source (De)	das ist die linke hlfte, die die logische seite ist und dann die rechte hälfte, die die intuitive seite ist.
Target (En)	there 's *the left half*, which is the logical side, and then *the right half*, which is the intuitive.
CNN	this is *half the left half*, which is the logical side, and then *half the rights* that is the intuitive side.
SAN	that's the left half, which is the logical side , and then *half*, that 's the intuitive side.
DPN-S2S	that's *the left half*, which is the logical side, and then *the right half*, which is the intuitive side.

Table 6: A German-English translation case to demonstrate DPN-S2S outperforms CNN and SAN in terms of translation accuracy.

of the CNN path in the decoder to extract necessary information for target word generation; in contrast, each hidden state in the SAN path has already included global semantics, and so the decoder only needs to focus on a small number of hidden states of the SAN path in the encoder, resulting in more focused alignments and smaller entropy (the second row of the table).

For an intuitive understanding, we visualize the alignments of a sentence pair in Figure 2. Comparing Figure 2b with 2a, we can see that the attention from the CNN path of the decoder to the SAN path of the encoder is more focused than that to the CNN path of the encoder. The same phenomenon can be found for attention from the SAN path of the decoder to the encoder, by comparing Figure 2d with 2c.

5.6 Case Study

To better understand the advantage of DPN-S2S over single path models, Table 6 shows a case from the German-English test set. We list the generated target sentence by CNN based model, SAN based model and DPN-S2S, respectively. As can be seen, CNN based model generates detailed words such as "half the left half" and "half the rights", without a global view to correctly organize these local meanings. SAN based model misses the local details about the "the right half" information. Our DPN-S2S well combine the local information from the CNN based path and the global information from the SAN based path, resulting in a better translation.

6 Conclusion

In this paper, we have proposed double path networks for sequence to sequence learning, named *DPN-S2S*, which uses a cross attention module to leverage the advantages of two different models, and achieves the state-of-art performance in several sequence to sequence tasks.

We plan to explore the following directions in the future. First, we will test the new model on more language pairs for neural machine translation and other sequence to sequence tasks. Second, it is interesting to investigate how to design better structures for information fusion. Third, we would like to extend the current double path model to multiple paths.

Acknowledgments

This work was supported by The National Key Research and Development Program of China under Grant 2018YFB1004904.

References

Ayana, Shiqi Shen, Zhiyuan Liu, and Maosong Sun. 2016. Neural headline generation with minimum risk training. *arXiv*.

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *arXiv*.

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

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv*.

Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. 2012. Advances in optimizing recurrent networks. *ICASSP*.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign. In *Proceedings of the 11th IWSLT*.

Kehai Chen, Rui Wang, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2017. Syntax-directed attention for neural machine translation. *AAAI*.

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

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In Doina Precup and Yee Whye Teh, editors, *ICML 2017*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017a. Convolutional sequence to sequence learning. In *ICML 2017*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017b. Convolutional sequence to sequence learning. *ICML 2017*.

Graff, David, Kong, Junbo, Chen, Ke, Maeda, and Kazuaki. 2003. English gigaword. In *Linguistic Data Consortium*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR 2016*.

Po-Sen Huang, Chong Wang, Dengyong Zhou, and Li Deng. 2017. Neural phrase-based machine translation. *arXiv*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP 2015*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *EMNLP 2015*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL 2016*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS 2014*.

Jun Suzuki and Masaaki Nagata. 2017. Rnn-based encoder-decoder approach with word frequency estimation. *arXiv*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NIPS*.

Hao Xiong, Zhongjun He, Xiaoguang Hu, and Hua Wu. 2018. Multi-channel encoder for neural machine translation. *AAAI*.

Jinchao Zhang, Mingxuan Wang, Qun Liu, and Jie Zhou. 2017. Incorporating word reordering knowledge into attention-based neural machine translation. In *ACL*.

An Empirical Investigation of Error Types in Vietnamese Parsing

Quy T. Nguyen[1,*] **Yusuke Miyao**[2] **Hiroshi Noji**[3] **Nhung T.H. Nguyen**[4]
[1]University of Information Technology, Ho Chi Minh City, Vietnam
[2]National Institute of Informatics, Tokyo, Japan
[3]Artificial Intelligence Research Center, AIST, Tokyo, Japan
[4]University of Science, Ho Chi Minh City, Vietnam
quynt@uit.edu.vn, yusuke@nii.ac.jp
hiroshi.noji@aist.go.jp, nthnhung@fit.hcmus.edu.vn

Abstract

Syntactic parsing plays a crucial role in improving the quality of natural language processing tasks. Although there have been several research projects on syntactic parsing in Vietnamese, the parsing quality has been far inferior than those reported in major languages, such as English and Chinese. In this work, we evaluated representative constituency parsing models on a Vietnamese Treebank to look for the most suitable parsing method for Vietnamese. We then combined the advantages of automatic and manual analysis to investigate errors produced by the experimented parsers and find the reasons for them. Our analysis focused on three possible sources of parsing errors, namely limited training data, part-of-speech (POS) tagging errors, and ambiguous constructions. As a result, we found that the last two sources, which frequently appear in Vietnamese text, significantly attributed to the poor performance of Vietnamese parsing.

Title and Abstract in Vietnamese

Khám phá dựa trên thực nghiệm các kiểu lỗi trong phân tích cú pháp tiếng Việt

Phân tích cú pháp đóng vai trò then chốt trong việc cải thiện chất lượng các hệ thống xử lý ngôn ngữ tự nhiên. Mặc dù đã có một vài dự án nghiên cứu về phân tích cú pháp tiếng Việt, chất lượng của những công trình này thấp hơn nhiều so với các nghiên cứu trên các ngôn ngữ phổ biến như tiếng Anh và tiếng Trung Quốc. Trong bài báo này, chúng tôi đánh giá các phương pháp phân tích cú pháp tiêu biểu trên ngân hàng cây cú pháp tiếng Việt để tìm ra phương pháp phân tích cú pháp phù hợp nhất cho tiếng Việt. Sau đó, chúng tôi kết hợp những ưu điểm của phân tích tự động và phân tích thủ công để khám phá các kiểu lỗi mà những phương pháp phân tích cú pháp được thực nghiệm mắc phải và tìm nguyên nhân cho những lỗi đó. Phân tích của chúng tôi tập trung vào ba nguồn chính gây ra lỗi, cụ thể là dữ liệu huấn luyện ít, lỗi trong việc gán nhãn từ loại, và các cấu trúc nhập nhằng. Kết quả cho thấy rằng lỗi trong việc gán nhãn từ loại và cấu trúc nhập nhằng là hai nguyên nhân chính gây ra hiệu suất thấp trong phân tích cú pháp tiếng Việt.

1 Introduction

Syntactic parsing plays a crucial role in improving the quality of natural language processing (NLP) applications and speech processing. Parsing has been broadly studied for languages such as English and Chinese, which leads to the development of many parsing methods. For example, Klein and Manning (2003), Matsuzaki et al. (2005), and Petrov and Klein (2007) improved probabilistic context-free grammar (PCFG) parsing by enriching contextual information, Finkel et al. (2008) and Hall et al. (2014) employed feature-based conditional random fields (CRFs), Zhu et al. (2013) and Coavoux and Crabbé (2017) employed shift-reduce algorithms, while Durrett and Klein (2015) and Dyer et al. (2016) used neural networks.

The parsing problem has not been studied thoroughly enough for Vietnamese. Since a Vietnamese Treebank was built by Nguyen et al. (2009), a few researchers have adapted available constituent parsers

[*]This work was carried out while the first author was a PhD student at The Graduate University for Advanced Studies (SOKENDAI), Japan.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3075–3089
Santa Fe, New Mexico, USA, August 20-26, 2018.

to construct a parser for Vietnamese; e.g., AC. Le et al. (2009) modified the Bikel's parser (Bikel, 2004) and Le-Hong et al. (2012) modified the LTAG parser developed by LORIA laboratory[1]. However, the parsing accuracies were far lower than the performances reported in English, Chinese, and French.

It is hard to parse Vietnamese text due to difficulties that result from characteristics of the Vietnamese language[2]. Vietnamese does not have word delimiters and inflectional morphemes in comparison to English. While similar problems also occur in Chinese (Xia et al., 2000), parsing Vietnamese may be more difficult because the modern Vietnamese writing system is based on Latin characters, which represent the pronunciation but not the meanings of words. As a result, there are many polysemous expressions in Vietnamese, i.e., expressions having the same surface form with different interpretations. Difficulties in Vietnamese parsing are also caused by word orders. Although Vietnamese is a subject-verb-object (SVO) language like English and Chinese, its word orders are different from these languages. For example, the word order in Vietnamese noun phrases (*NPs*) is exactly the same as those in simple sentences, which leads to ambiguities in labelling these two types of expressions. In addition, other problems, such as word omission, contribute to many complications in parsing Vietnamese texts.

Another possible reason for difficulties in Vietnamese parsing is treebank design[3]. For example, syntactic categories in the Vietnamese Treebank by Nguyen et al. (2016) are similar to those in the Penn Treebank, e.g., *NP* for noun phrases and *VP* for verb phrases. These categories were not fine-grained enough to distinguish various syntactic constructions in Vietnamese. In addition, there are ambiguous constructions that have the same tag sequences (including POS tags and phrase tags), but are bracketed in different ways. For instance, the sequence of POS tags *Nn Vv* can be bracketed as a noun phrase (*NP*) in which *Vv* is a modifier of the head noun *Nn*. The sequence can also be bracketed as a simple sentence where *Nn* is the subject and *Vv* is the predicate.

A worthwhile effort would be to analyze parsing errors to improve the quality of parsers. Based on this analysis, we can consider how to modify parsing models such that they can capture the necessary syntactic information. Several researchers (Levy and Manning, 2003; Hara et al., 2009) have manually investigated parsing errors. Manual investigations can help us find reasons for parsing errors that are specific to languages. However, this method cannot be applied to a large amount of parsing outputs and is also limited to a few syntactic phenomena. Kummerfeld et al. (2012) developed a tool to automatically classify English parsing errors within a set of predefined error types such as NP attachment and VP attachment. However, based on these analyzing results, we cannot specify any reasons to explain why the error types occurred, which is important clues to improve parsers.

In this paper, we firstly evaluated representative parsing models on the Vietnamese Treebank. We then investigated three possible sources of errors produced by parsers, viz., limited training data, confusing POS tags, and ambiguous constructions. Our analysis method combined an automatic tool and manual analysis. We used Kummerfeld et al. (2012)'s tool in the automatic phase to quantify how often the types of errors occurred in the experimented parsers. By comparing the results obtained from this analysis, we could identify which types of errors could be solved by different parsers as well as which errors were difficult for parsers to address. We manually investigated parsing errors in the second phase based on the results obtained from the analysis in the first phase to find the reasons for each error type.

2 Parsing evaluation

We applied six representative parsers in the literature in English and Chinese to the Vietnamese Treebank. For each parser, we used parameter settings that had the best accuracy in English.

Stanford-Unlex (Klein and Manning, 2003) is an unlexicalized probabilistic context free grammar (PCFG) parser where the coarse categories of PCFG, such as *NP* for noun phrases, are enriched by several simple structural annotations, e.g., markovization before splitting several symbols according to a manual grammar.

Stanford-RNN (Socher et al., 2013) is a combination of a PCFG with a recursive neural network,

[1] http://www.loria.fr

[2] An example illustrating Vietnamese charateristics is presented in Appendix C.

[3] Vietnamese Treebank's POS tags and constituent tags are described in Appendices A and B.

Parsers	Tagging accuracy	R	P	F
Stanf-Unlex	89.92	52.15	56.38	54.18
Stanf-RNN	91.83	63.66	66.15	64.88
Berkeley	93.94	70.38	73.48	71.90
Epic-CRF	93.15	72.20	72.96	72.58
Epic-NeuralCRF	93.85	71.68	72.42	72.05
RNNGs-D	94.65	71.94	72.45	72.19
RNNGs-G	**94.65**	**71.71**	**74.21**	**72.94**

Table 1: PARSEVAL scores on the test set for all parsers in our experiments.

which reranks the outputs of Stanford-Unlex. Phrase representations in this model are learned via a recursive neural network that is conditioned on the coarse PCFG.

Berkeley parser (Petrov et al., 2006) is an unlexicalized PCFG parser where the PCFG is automatically enriched and generalized by using informative latent annotations.

Epic-CRF (Hall et al., 2014) is a CRF parser in which the anchored rules are scored by using sparse features of the surface spans. The base grammar in this parser employs parent annotations, a subset of annotations in Stanford-Unlex.

Epic-NeuralCRF (Durrett and Klein, 2015) is also a CRF parser but the anchored rules are scored using dense features that are computed via a feedforward neural network.

RNNGs (Dyer et al., 2016) is a top-down transition-based neural model in which a recurrent neural network conditions on a full input sentence to parameterize decisions. The RNNGs parser supports two models, i.e., discriminative (RNNGs-D) and generative models (RNNGs-G).

We used the Vietnamese Treebank developed by Nguyen et al. (2016) which includes 10,377 sentences (containing 232,838 words and 280,505 syllables). We randomly selected 1,000 sentences for the development (dev) set, 1,000 sentences for the test set, and the rest, including 8,377 sentences, was used for training. We used the input data with gold word segmentation[4] and predicted POS tags (PredictedPOS)[5].

Table 1 summarizes the PARSEVAL results of each parser on the test set. We can see from this table that RNNGs-G, which is a top-down transition-based neural model, achieved the best accuracy on the Vietnamese Treebank, while the differences among the RNNGs-G, RNNGs-D, Epic-CRF, and Epic-NeuralCRF were not very significant. This indicates that parsing models based on CRF and neural networks are equally good for Vietnamese. However, these parsers still achieved far less accuracy in comparison to that in English. One possible reason is that although each of the parsing models had its own advantages, these were insufficient to address all Vietnamese constructions.

Although Stanford-Unlex and Stanford-RNN also enriched the coarse categories of PCFG with contextual information, they achieved far less accuracy in comparison with the RNNGs parser, Berkeley parser, Epic-CRF, and Epic-NeuralCRF. The main reason for this is that Stanford-Unlex and Stanford-RNN usually split tags into subcategories in response to linguistic trends in the English Penn Treebank. For example, determiners are distinguished from demonstratives, or the POS tag of *IN* is divided into six subcategories. It was difficult to apply these parsers to the Vietnamese Treebank because many linguistic trends occurring in the English Penn Treebank are not found in the Vietnamese Treebank.

In comparison with Stanford-Unlex and Stanford-RNNs, the Berkeley parser, Epic-CRF, and Epic-NeuralCRF obtained far higher accuracies. The reason is that the Berkeley parser, Epic-CRF, and Epic-NeuralCRF used automatic methods to enrich contextual information, which were easy to apply to other languages including Vietnamese. This also indicated that enriching contextual information is beneficial for Vietnamese parsing. Contextual information can be fine-grained categorizations or features extracted from the surface spans of anchor rules[6]. However, it seems that the contextual information extracted on the basis of CFG rules did not sufficiently capture Vietnamese constructions. While RNNGs modeled

[4]We used gold word segmentation for all experiments in this research.

[5]The POS tags were predicted by the parsers. In the case of RNNGs, since the parser cannot predict POS tags, we trained a Vietnamese POS tagger with the method of Nghiem et al. (2008), using the same training data as the parsers. This POS tagger achieved an accuracy of 94.65% on the test set.

[6]An anchor rule consists of the first and last words of spans, span lengths, span shape descriptions, and the start, stop, and split indexes where the rules are anchored.

Parser	F-score	1-Word Phrase	Unary	VP Attach	NP Attach	PP Attach	Clause Attach	NP Int.	Mod Attach	Diff Label	Co-ord	XoverX Unary	Other
Stanford-Unlex	53.49	1.33	0.86	**2.23**	**1.86**	1.80	1.06	0.68	0.78	0.92	0.06	0.03	4.06
Stanford-RNN	64.16	1.09	0.72	**1.87**	**1.33**	1.35	0.98	0.56	0.96	0.88	0.07	0.02	2.49
Berkeley	71.79	0.91	0.54	**1.60**	**1.10**	0.91	0.87	0.44	0.52	0.75	0.04	0.02	1.99
Epic-CRF	72.03	0.94	0.53	**1.48**	**1.16**	0.87	0.83	0.44	0.49	0.88	0.03	0.02	2.09
Epic-NeuralCRF	71.66	0.90	0.53	**1.53**	**1.27**	0.96	0.84	0.44	0.44	0.79	0.04	0.02	2.15
RNNGs-D	70.86	0.92	0.56	**1.54**	**1.25**	1.00	0.92	0.42	0.46	0.72	0.04	0.03	1.79
RNNGs-G	72.22	0.86	0.48	**1.36**	**1.10**	0.94	0.81	0.38	0.38	0.70	0.03	0.02	1.78

Table 2: Average number of bracket errors per sentence on the development set of Vietnamese Treebank.

sequences based on the full input sentences, which helps it find more useful contextual information.

Our evaluation of representative parsing models on the Vietnamese Treebank provided an overall sense of parsing accuracy in the Vietnamese language. While RNNGs-G achieved the highest accuracy, this was still far less than that reported in English or Chinese. In the following sections, we will explain how we investigated parsing errors to find reasons for the poor accuracy of Vietnamese parsing. This investigation was a valuable step toward improving the quality of Vietnamese parsing.

3 Investigating behaviors of parsers

This investigation was aimed at clarifying two main issues: (1) what are the most frequent errors in Vietnamese parsing?; and (2) which errors can be addressed by different parsing methods? To answer these two questions, we used a tool developed by Kummerfeld et al. (2012)[7] to classify the parsing errors into error types such as VP and NP attachments. We then computed the average number of bracket errors per sentence for each error type. The results on the development set of the Vietnamese treebank are presented in Table 2[8].

By comparing eleven types of errors produced by the parsers (columns from "1-Word Phrase" to "XoverX Unary" in Table 2), we can see that bracket errors caused by VP attachments are the most frequent, while the ones caused by NP and PP attachments are the second most. The table also indicates how the errors can be addressed by different parsing methods. For example, RNNGs-G is the best method that could address most error types in Vietnamese parsing.

In comparison to English parsing errors (Kummerfeld et al., 2012), errors in Vietnamese have the following attributes. (i) Most error types that appeared in English parsing also appeared in Vietnamese parsing. However, the average errors per sentence in Vietnamese parsing were much larger than those in English, except for co-ordination. (ii) PP-attachment was the most problematic constructions in English but it was not in Vietnamese (it was ranked at the third position). (iii) VP attachment was the most frequent error type in Vietnamese parsing while it did not occur in English parsing.

Kummerfeld et al. (2013) also analyzed Chinese parsing errors but we did not directly compare with them because our method of error classification is different from their method. Nevertheless, we drew three main conclusions from our analysis results: (i) the NP internal structure is the most problematic construction in Chinese parsing, while it is rare in Vietnamese parsing. (ii) Although VP, NP, and PP attachments are the top three difficult constructions in Vietnamese parsing, they are not frequent errors in Chinese parsing; no NP attachment errors were even detected in Chinese parsing. (iii) One-word phrases are the most problematic constructions in both languages.

Apart from containing the same complex constructions as English and Chinese such as PP attachments, Vietnamese parsing has its own constructions, e.g., VP attachments, that cannot be solved with the current parsing methods. A new parsing method to address such specific issues in the Vietnamese language is required. By quantifying error types produced by different parsing methods, our investigation provided an orientation for improving the quality of Vietnamese parsing. For example, solving the most frequent errors, such as VP attachment, NP attachment, and PP attachment errors can significantly improve the parsing performance. However, we could not find reasons of error types from this investi-

[7]This tool was designed to classify English parsing errors. We found about 20.1% of errors were identified as the "Other" type by directly applying this tool to Vietnamese parsing output (this ratio is about 11.3% for English). Several errors in the "Other" class could be classified into existing types such as clause and VP attachments.

[8]The types of errors reported in this paper are the same as those in Kummerfeld et al. (2012).

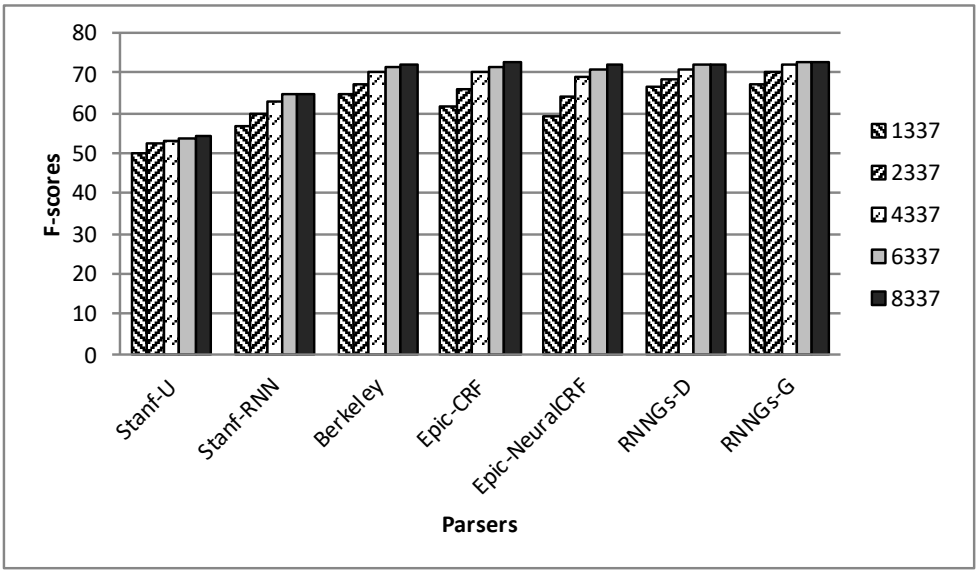


Figure 1: Parsing results from the parsers with different amounts of training data.

gation, which are very important clues to solving errors. We therefore conducted another investigation on three possible sources for the poor performance of Vietnamese parsing in the following sections, viz., the impact of training data size (Section 4), impact of POS tagging errors (Section 5), and impact of ambiguous constructions (Section 6).

4 Impact of training data size

Our training data consisted of 8,337 sentences, which were much fewer than the data used for English and Chinese parsing. To verify whether the small size of training data was a reason for the poor accuracy of the parsers, we evaluated them on different amounts of training data including 1,337, 2,337, 4,337, 6,337, and 8,337 sentences. The parsing results (F_1) on the test set of the Vietnamese Treebank are presented in Figure 1. The figure shows that the parsers' performance was improved when we eventually increased the training data from 1,337 to 6,337. However, the F-score is almost saturated when the number of sentences was increased from 6,337 to 8,337 sentences. These observations indicated that we could not expect a significant improvement in accuracy by simply increasing the amount of data. Consequently, the amount of training data was not the main reason for the poor performance of the parsers.

5 Impact of tagging errors

Figure 2 presents the PARSEVAL evaluations (F-scores) on two different versions of the test set, *PredictedPOS*, i.e., POS tags were automatically produced, and *GoldPOS*, i.e., gold POS tags were used. These results indicate that using gold POS tags could improve the accuracies of the parsers. The F-score of RNNGs-G especially achieved 78.2%, which was improved by 5.26% points in comparison with the use of an automatic POS tagger (*PredictedPOS*). However, this figure does not indicate the error types that could be solved by improving POS tagging.

To find how the gold POS tags affected parsing errors, we compared the error types produced by the best parser, RNNGs-G, on two different versions of the development set: (1) *Gold*: by using gold word segmentation and POS tags and (2) *Pred.*: by only using gold word segmentation. Results in Table 3 reveal that the occurrence of some error types was reduced when gold POS tags were used. However, it should be noted that the reductions were not significant, especially in the cases of VP and NP attachments. Therefore, enhancing POS tagger would somehow improve the performance of Vietnamese parsing but the improvement would not be radical since VP and NP attachments are the most frequent errors.

We used the same method as Kummerfeld et al. (2013) to find how an ambiguous POS pair impacted

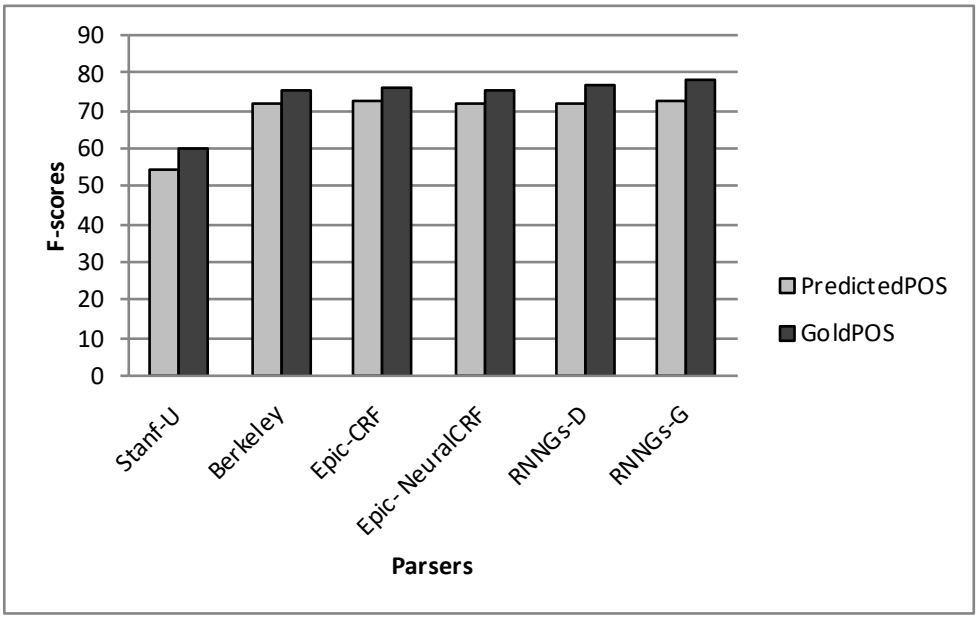


Figure 2: Results from the parsers on two different versions of test set, PredictedPOS and GoldPOS.

parsing errors. We began from *Gold*, and replaced the gold tags with tags predicted by the automatic tagger (semi-gold input). We then parsed the semi-gold input by using RNNGs-G and obtained the results from the PARSEVAL evaluation and automatic analysis. The impacts of the top four confusing POS pairs on overall bracketing errors and the F-scores are presented in Table 4. We can see that all of these four confusing POS pairs were potential contributors to bracketing errors that caused significant decreases in F-scores. However, this table also indicates that the frequency of confusing POS pairs was not directly proportional to the contributions they made to parsing errors. For example, although *Aa/Vv* was the third most confusing POS pair, it contributed most to parsing errors and caused the highest decrease in F-scores (2.58). While occurrences of *Nn/Vv* were highest (106 times), they caused the lowest decrease in F-scores (1.94).

Table 5 summarizes the impact that the top four confusing POS pairs had on each error type. The positive and negative numbers correspond to the number of bracketing errors that were created or reduced in parsing output when we replaced gold POS tags with predicted POS tags. Analysis based on four of the most frequent error types of VP, NP, PP, and clause attachments revealed the following.

VP and NP attachments: As previously explained, improving POS tagger did not assist in preventing VP and NP attachment errors. We can see from Table 5 that bracketing errors caused by VP attachment increased when we replaced the gold tag of *Vv* with predicted tags of *Nn* or *Aa*. Bracketing errors caused by NP attachment also increased when we revised the gold tag of *Aa* with the predicted tag of *Vv*. This indicated that gold POS tags were helpful in these cases. However, using predicted POS tags was better for some constructions, e.g., using the predicted tag of *Vv* instead of the gold *Nn* reduced 65 bracketing errors that were caused by NP attachment errors. These observations have revealed that the gold POS tags are important for solving VP and NP attachment errors. However, it is possible that the current 33 POS tags of the Vietnamese Treebank are not good enough for disambiguating constructions.

PP attachments: All four confusing POS pairs caused these types of parsing errors. However, *Aa/Vv* was the major contributor to parsing errors

Clause attachments: *Aa/Vv* was the major contributor to these types of parsing errors. In addition, ambiguous POS pairs, such as *Nn/Vv* and *Vv/Aa*, also created significant numbers of bracketing errors.

In summary, the quality of Vietnamese parsing could be improved by about 5% points through enhancing the Vietnamese POS tagging. We also found that all frequent ambiguous POS pairs contributed to parsing errors, in which *Aa/Vv* was the major contributor. Our analysis results indicated that improving POS tagging was beneficial for some constructions, such as PP and clause attachments. However,

Error type	Occurrences		Node errors		Node errors per sentence		Gain
	Pred.	Gold	Pred.	Gold	Pred.	Gold	
1-Word Phrase	765	517	857	544	0.86	0.54	0.32
Unary	475	386	475	386	0.48	0.39	0.09
VP Attach	392	373	1362	1368	1.36	1.37	-0.01
PP Attach	415	345	936	837	0.94	0.84	0.10
Diff label	350	157	700	314	0.70	0.31	0.39
NP Attach	333	339	1104	1101	1.10	1.10	0.00
Clause Attach	327	277	814	696	0.81	0.70	0.11
Mod Attach	216	182	381	362	0.38	0.36	0.02
NP Int.	245	211	379	324	0.38	0.32	0.06
Co-ord	34	44	34	44	0.03	0.04	-0.01
XoverX Unary	24	25	24	25	0.02	0.03	0.01
Other	969	729	1779	1435	1.78	1.44	0.34
Sum	4545	3585	8845	7436			

Table 3: Statistics on errors produced by RNNGs-G on two different versions of the development set, *Pred.* and *Gold*. "Gain" represents gain (positive number) and loss (negative number) of errors per sentence when replacing automatically predicted POS tags with gold POS tags (i.e., Errors per sentence (Pred.) - Errors per sentence (Gold)).

Confusing tags	Occurrences	Bracketing errors	F1	Δ F1
Nn/Vv	106	7694	76.26	-1.94
Vv/Nn	87	7764	76.15	-2.05
Aa/Vv	77	7897	75.62	-2.58
Vv/Aa	69	7749	76.07	-2.13
Gold		7436	78.20	

Table 4: The most frequently confusing POS tag pairs in Vietnamese POS tagging. Δ F1 indicates the decreases in F-scores when gold POS tags were replaced with predicted POS tags. Meanings of POS tags are: *Nn*: common nouns, *Vv*: common verbs, and *Aa*: adjectives. For each confusing POS pair *x/y*, *x* is the gold POS, and *y* is the predicted one.

not all gold tags are helpful for solving the parsing errors, which means that the current 33 POS tags of the Vietnamese Treebank are not good enough for disambiguating constructions. We therefore should improve the tag set (e.g., splitting tags) or find more features to address the parsing errors, especially VP and NP attachments—the two most frequent error types.

6 Ambiguous constructions in Vietnamese

This section aims to investigate parsing errors caused by ambiguous constructions in Vietnamese. Our investigation was carried out on the *Gold* data set parsed by the RNNGs-G model trained on 8,337 sentences. By doing that, we could eliminate the impacts of limited training data, word segmentation errors, and confusing POS tags from the parsing output.

We randomly selected 100 sentences from the parsing output for this investigation. We then classified

Error type	Nn/Vv	Vv/Nn	Aa/Vv	Vv/Aa
Single Word Phrase	80	70	66	68
Unary	16	22	52	38
VP Attachment	-26	44	-45	17
NP Attachment	-65	-45	37	-43
PP Attachment	27	51	83	48
Clause Attachment	75	26	103	70
NP Internal Structure	69	21	50	30
Modifier Attachment	11	25	-11	-14
Different label	42	44	32	22
Co-ordination	-21	-14	-14	-21
XoverX Unary	-1	-1	-2	-1
Other	51	85	110	99

Table 5: Gains and Losses in bracket errors when gold POS tags were replaced with predicted ones.

Ambiguous construction	Error type	Frequency
Nx|Vv|Aa|Cs Vv|Aa|Nx VP	VP attachment	19/34
Vv|Nx Vv|Aa|Nx NP	NP attachment	25/39
Nx|Vv Nx|Vv PP	PP attachment	17/29
NP VP NP VP	Clause attachment	10/26
NP|VP NP|VP VP	Clause attachment	6/26
Cs NP VP	Clause attachment	4/26

Table 6: Several frequent ambiguous constructions in Vietnamese parsing.

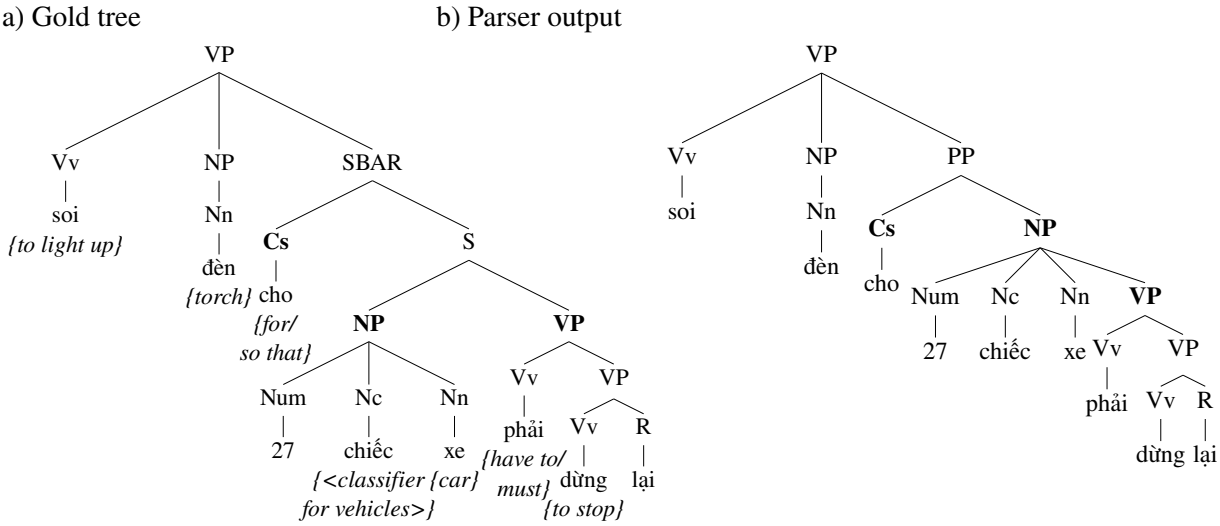


Figure 3: Examples illustrating an VP attachment error that is caused by ambiguous construction *Cs NP VP* in Vietnamese.

parsing errors into error types by applying the automatic analysis tool (Kummerfeld et al., 2012) to the selected sentences. Next, we manually investigated the reasons for each error type. Since individual error types were probably caused by particular constructions, finding reasons for parsing errors according to their error types would provide precise orientations for solving the errors.

Table 6[9] presents several frequent ambiguous constructions[10] in Vietnamese parsing that were found in this manual investigation. The column "Ambiguous construction" in this table presents frequent ambiguous tag sequences in Vietnamese. Error types caused by ambiguous tag sequences are indicated in the column "Error type". For example, tag sequences in the first row, such as *Cs Nx VP*, are candidates for VP attachment errors. For each pair of *x/y* in the column "Frequency", *x* represents the number of times that each ambiguous tag sequence caused errors in 100 investigated sentences; while *y* denotes the frequency of each error type in 100 investigated sentences. For instance, 19/34 in the first row means that we found 34 VP attachment errors in 100 investigated sentences, in which 19 errors were caused by the ambiguous constructions listed in the first column.

These ambiguous constructions are caused by the characteristics of the Vietnamese language, such as lack of inflectional morphemes, polysemous words, word order in which modifying lexical words follow the head word, and word omission.

Figure 3[11] illustrates an ambiguity attributed by most of the above-mentioned linguistic features. In

[9] *Nx* in Table 6 represents a noun that can be *Nn* (a common noun), *Nc* (a classifier noun), or *Ncs* (a special classifier noun). Each component of the tag sequence can be generalized as a phrase, e.g., instead of *Nn*, a noun phrase with head word *Nn* can be placed in the position of *Nn*.

[10] Refer to Appendix D for more details about the ambiguous constructions in Vietnamese parsing.

[11] Underscore "_" is used to link syllables of Vietnamese multi-syllabic words. English translations of Vietnamese words are provided as subscripts. If a Vietnamese word does not have a translatable meaning, the subscript is blank. The translation for the Vietnamese sentence is provided in braces below the original text.

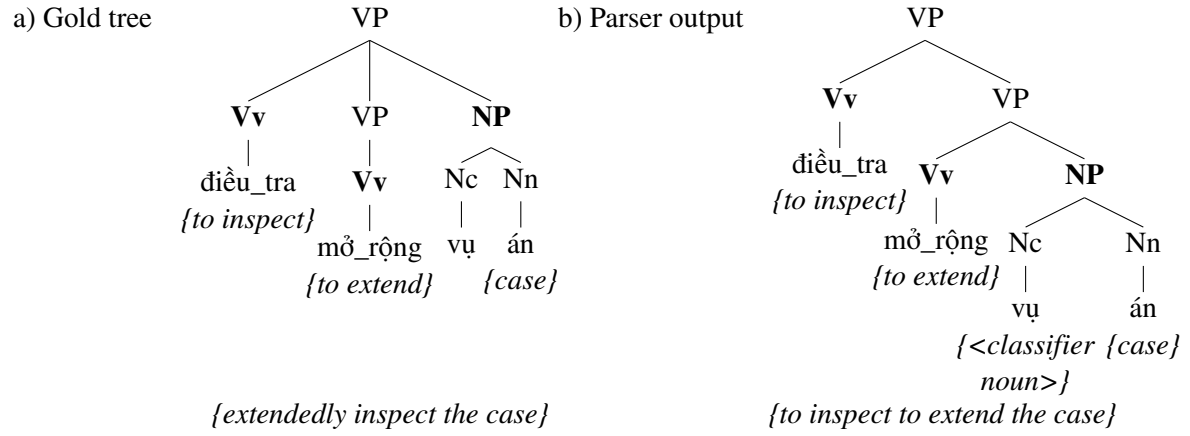


Figure 4: Examples illustrating an NP attachment error that is caused by ambiguous construction *Vv Vv NP* in Vietnamese.

the figure, the bold *NP VP* should be considered as a simple sentence (gold tree—Figure 3a) where *VP* is a predicate, but the parser labelled the whole construction as a noun phrase (*NP*) in which *VP* is an attachment (parser output—Figure 3b). A reason for that is the lack of inflectional morphemes in Vietnamese, in which a verb can (1) modify a noun, a verb, or an adjective, or (2) be the main verb of a sentence, without changing the word form. In addition, modifying verbs, adjectives, etc. follow the head words in Vietnamese noun phrases, which makes the word order in noun phrases identical to that in simple sentences. Polysemous words are also a possible reason for such parsing errors. In the figure, the word *"cho"* has two different interpretations. It means *"so that"* in the case it precedes a clause as shown in the gold tree. When it precedes a noun phrase, it means *"for"* as shown in the parser output. It is difficult for RNNGs-G to parse this case because both constructions *"cho S"* and *"cho NP"* occur in the Vietnamese Treebank.

Figure 4 presents the ambiguous construction of *Vv Vv NP* (row 2 in Table 6), a candidate for the NP attachment error in Vietnamese parsing. The *NP* in this construction can be annotated as the modifier of the first verb (gold tree) or the modifier of the second verb (parser output). This ambiguity is also caused by the common word order in Vietnamese, where modifying lexical words follow the head word. In addition, this also results from the lack of inflectional morphemes. We can see from the figure that this is not an ambiguous construction in English (or Chinese) parsing because cases like *"mở_rộng$_{\textit{to extend}}$"* are expressed by adverbs that appear before the head verb and cannot be combined with a noun. However, Vietnamese does not have adverbs; verbs and adjectives are used in such contexts. It is confusing to bracket these constructions because a noun phrase can be the modifier of both previous words.

Statistics on the top four frequent error types, i.e., VP, NP, PP, and clause attachments, revealed that about 63% of attachment errors were caused by ambiguous constructions in Vietnamese. Although RNNGs-G considered the full input sentence in its model, the contextual information was only found based on the coarse non-terminal symbols and surface forms of the words, which were not enough to address ambiguous constructions.

We found in Section 2 that non-neural Berkeley and Epic parsers perform competitively to RNNGs-G. This might suggest that these two parsers have their own advantages, and a possible direction for the future development of Vietnamese parsers is to integrate the idea of these parsers with the modern strong neural parsers. For example, although RNNGs represent each constituent in a continuous space, explicit, symbolic distinction among categories as done in the Berkeley and Epic parsers would still be useful for Vietnamese parsing. In the Vietnamese treebank, subordinate conjunctions that introduce a clause and a noun phrase are both annotated with the same POS tag of *Cs*. This caused ambiguous constructions in which a relative clause and a prepositional phrase have the same tag sequence of *Cs NP VP*. Symbol splitting might help for resolving this ambiguity. For example, we can refine the POS tag for *"cho"* mentioned in Figure 3 as *Cs-SBAR* and *Cs-PP*, depending on whether it precedes a sentence or a noun

phrase, with manual refinements as in Stanford and Epic parsers, or automatic refinements as in the Berkeley parser. We therefore expect that one possible approach for future research is a hybrid approach that combines the advantages of different parsers.

7 Conclusion

We have evaluated representative parsing models on the Vietnamese Treebank and found that parsing models based on CRF and neural networks are good for Vietnamese. Contextual information is very beneficial for improving the parsing performance. We have also investigated the errors produced by the parsers. This investigation has revealed the frequent errors in Vietnamese parsing: VP attachment, NP attachment, PP attachment, and clause attachment. Although POS tagging errors have significantly contributed to many parsing errors such as PP attachment and clause attachment errors, they are not an essential reason for the top two error types, VP attachment and NP attachment.

In addition, our investigation on parsing errors has found that ambiguous constructions are the major contributor to the errors in Vietnamese parsing, which are resulted from the characteristics of the Vietnamese language. Our goal in future work is to develop a parser that can overcome the challenges found by the analysis in this paper, and one possible direction is a hybrid approach that combines the advantages of different parsing models investigated in the paper.

References

Anh-Cuong AC. Le, Phuong-Thai Nguyen, Hoai-Thu Vuong, Minh-Thu Pham, and Tu-Bao Ho. 2009. An experimental study on lexicalized statistical parsing for vietnamese. In *Proceedings of Knowledge and Systems Engineering*, pages 162–167. IEEE.

Daniel M Bikel. 2004. *On the parameter space of generative lexicalized statistical parsing models*. Ph.D. thesis, Citeseer.

Maximin Coavoux and Benoit Crabbé. 2017. Incremental discontinuous phrase structure parsing with the gap transition. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1259–1270.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, page 302–312. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *Proceedings of NAACL-HLT 2016*, page 199–209. Association for Computational Linguistics.

Jenny Rose Finkel, Alex Kleeman, and Christopher D Manning. 2008. Efficient, feature-based, conditional random field parsing. In *ACL*, volume 46, pages 959–967.

David Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *Proceedings of the 52rd Annual Meeting of the Association for Computational Linguistics*, pages 228–237.

Tadayoshi Hara, Yusuke Miyao, and Jun'ichi Tsujii. 2009. Descriptive and empirical approaches to capturing underlying dependencies among parsing errors. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1162–1171. Association for Computational Linguistics.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Jonathan K Kummerfeld, David Hall, James R Curran, and Dan Klein. 2012. Parser showdown at the wall street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1048–1059. Association for Computational Linguistics.

Jonathan K Kummerfeld, Daniel Tse, James R Curran, and Dan Klein. 2013. An empirical examination of challenges in chinese parsing. In *ACL (2)*, pages 98–103.

Phuong Le-Hong, Thi-Minh-Huyen Nguyen, and Azim Roussanaly. 2012. Vietnamese parsing with an automatically extracted tree-adjoining grammar. In *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2012 IEEE RIVF International Conference on*, pages 1–6. IEEE.

Roger Levy and Christopher Manning. 2003. Is it harder to parse chinese, or the chinese treebank? In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 439–446. Association for Computational Linguistics.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic cfg with latent annotations. In *Proceedings of the 43rd annual meeting on Association for Computational Linguistics*, pages 75–82. Association for Computational Linguistics.

Minh Nghiem, Dien Dinh, and Mai Nguyen. 2008. Improving vietnamese pos tagging by integrating a rich feature set and support vector machines. In *Proceedings of Research, Innovation and Vision for the Future in Computing and Communication Technologies (RIVF)*, pages 128–133. IEEE.

Phuong-Thai Nguyen, Xuan-Luong Vu, Thi-Minh-Huyen Nguyen, Van-Hiep Nguyen, and Hong-Phuong Le. 2009. Building a large syntactically-annotated corpus of vietnamese. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 182–185. Association for Computational Linguistics.

Quy T. Nguyen, Yusuke Miyao, Ha T.T. Le, and Ngan L.T. Nguyen. 2016. Challenges and solutions for consistent annotation of vietnamese treebank. In *Proceedings of the Language Resources and Evaluation Conference*.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL*, volume 7, pages 404–411. Association for Computational Linguistics.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 455–465. Association for Computational Linguistics.

Fei Xia, Martha Palmer, Nianwen Xue, Mary Ellen Okurowski, John Kovarik, Fu-Dong Chiou, Shizhe Huang, Tony Kroch, and Mitchell P Marcus. 2000. Developing guidelines and ensuring consistency for chinese text annotation. In *Proceedings of the Second International Conference on Language Resources and Evaluation*.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 434–443.

Appendix A The POS tags designed for the Vietnamese Treebank

No.	POS tag	Meaning of tag	No.	POS tag	Meaning of tag
1	SV	Sino-Vietnamese syllable	17	NA	Noun-adjective
			18	Vcp	Comparative verb
2	Nc	Classifier noun	19	Vv	Other verb
3	Ncs	Special classifier noun	20	An	Ordinal number
4	Nu	Unit noun	21	Aa	Other adjective
5	Nun	Special unit noun	22	Pd	Demonstrative pronoun
6	Nw	Quantifier indicating the whole	23	Pp	Other pronoun
			24	R	Adjunct
7	Num	Number	25	Cs	Preposition or conjunction introducing a clause
8	Nq	Other quantifier			
9	Nr	Proper noun	26	Cp	Other conjunction
10	Nt	Noun of time	27	ON	Onomatopoeia
11	Nn	Other noun	28	ID	Idioms
12	Ve	Exitting verb	29	E	Exclamation word
13	Vc	Copula "là" verb	30	M	Modifier word
14	D	Directional verb	31	FW	Foreign word
15	VA	Verb-adjective	32	X	Unidentified word
16	VN	Verb-noun	33	PU	Punctuation

Appendix B The constituency tags designed for the Vietnamese Treebank

No.	Tag	Explanation
1	NP	Noun phrase
2	QP	Quantitative phrase
3	VP	Verb phrase
4	ADJP	Adjective phrase
5	PP	Prepositional phrase
6	RP	Adjunct phrase
7	CONJP	Conjunction phrase
8	UCP	Unlike coordinated phrase
9	QNP	Questioning noun phrase
10	QRP	Questioning adjunct phrase
11	QPP	Questioning prepositional phrase
12	QADJP	Questioning adjective phrase
13	MDP	Modal phrase
14	S	Simple/compound declarative sentence
15	SQ	Question
16	SPL	Special sentence
17	SBAR	Subordinate clause
18	XP	Unknown phrase

Appendix C Examples illustrating Vietnamese characteristics

Figure 5 represents an example illustrating two characteristics of Vietnamese, viz., lack of inflectional morphemes and the word orders. We can see from English translations in this figure that the noun *establishment* (Figure 5a) and the past tense verb *established* (Figure 5b) can be distinguished on the basis of inflectional morphemes *"-ment"* and *"-ed"*. However, Vietnamese does not have such clues because Vietnamese is a non-inflected language. This leads to the situation that the verbs $thiết\_lập_{to\ establish}$ have the same surface forms in both syntactic roles, i.e., the modifier of the noun $thủ\_tục_{procedure}$

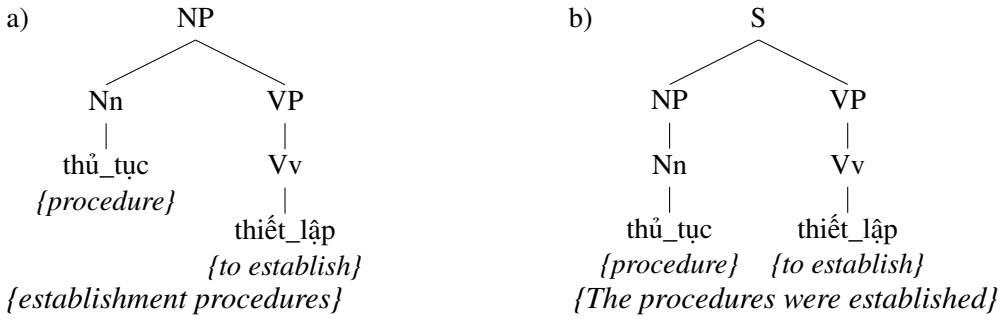


Figure 5: Examples illustrating Vietnamese characteristics.

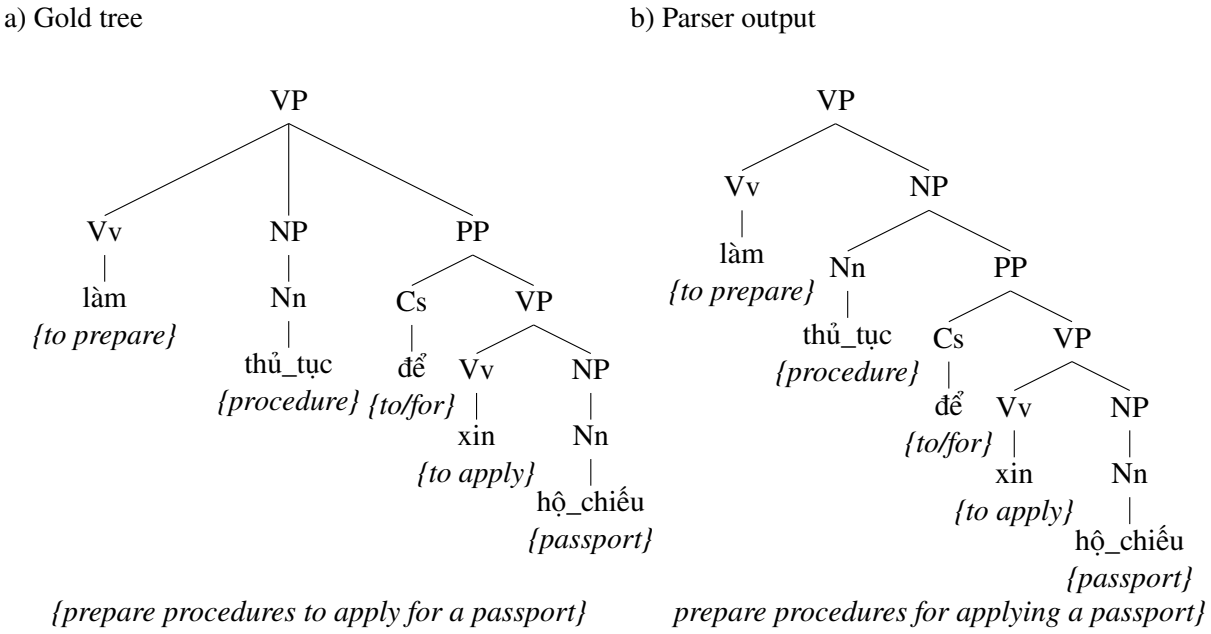


Figure 6: Examples illustrating the PP attachment error in Vietnamese parsing.

(Figure 5a) and the predicate of the simple sentence (Figure 5b). In addition, the modifying verb follows the head noun in a Vietnamese noun phrase (as shown in Figure 5a). This word order is the same as that in a simple sentence (Figure 5b). Parsing noun phrases and simple sentences is ambiguous because they have the same construction of *Nn Vv*.

Appendix D Examples illustrating several error types in Vietnamese parsing

As mentioned in the main content of the paper, we applied the error detection tool proposed by Kummerfeld et al. (2012) to Vietnamese. Therefore, our error types are the same as those in Kummerfeld et al. (2012). Follows are more details about the top four frequent error types in Vietnamese parsing.

D.1 VP attachment error

Constructions in which *VPs* are moved, viz., the incorrect bracket over an *VP* or incorrect attachment in noun phrases, verb phrases, adjective phrases, and prepositional phrases (an example was given in Section 6).

D.2 NP attachment error

Constructions in which *NPs* are moved (an example was given in Section 6).

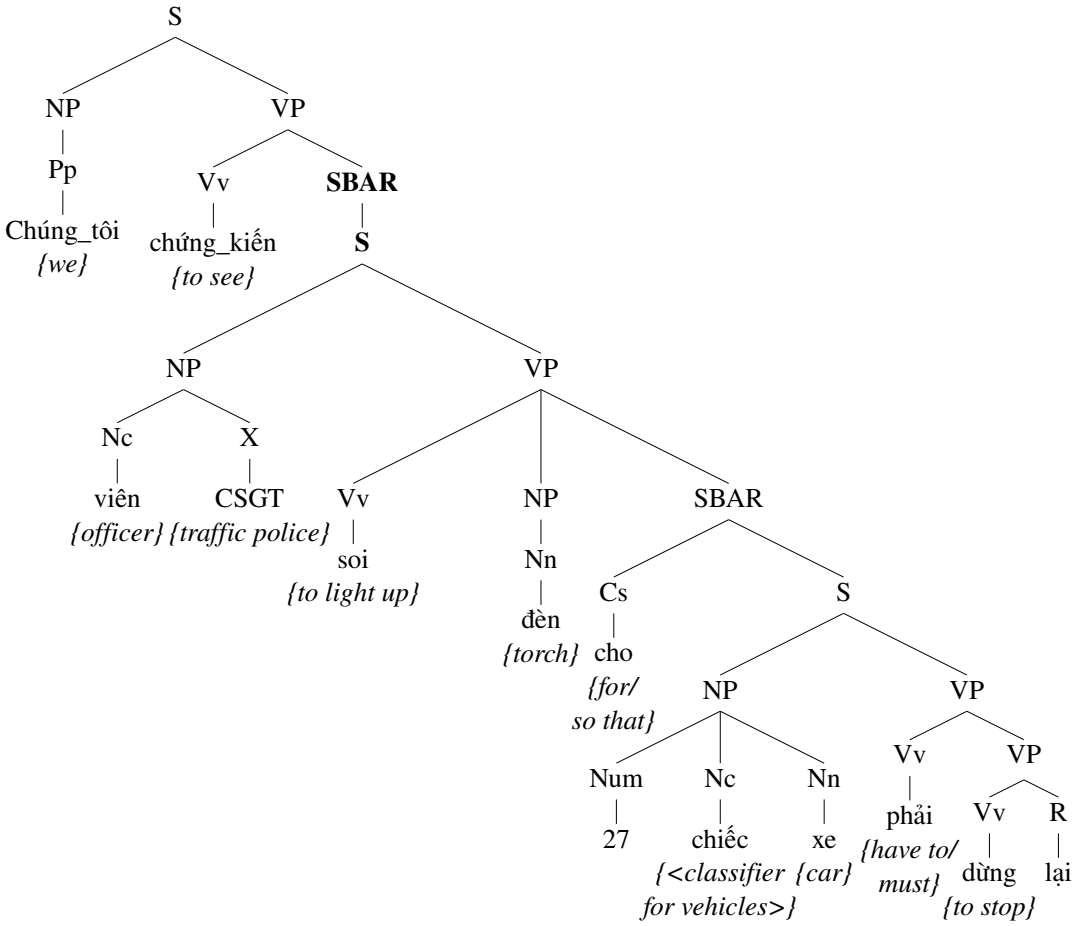
D.3 PP attachment error

Constructions in which *PPs* are moved. For example, Figure 6 represents an PP attachment error caused by the ambiguous construction *Vv Nn PP*. The *PP* in this example should be annotated as a modifier of the previous verb $làm_{to\ prepare}$ (gold tree), rather than attached to the previous noun $thủ\_tục_{procedure}$ (parser output).

D.4 Clause attachment error

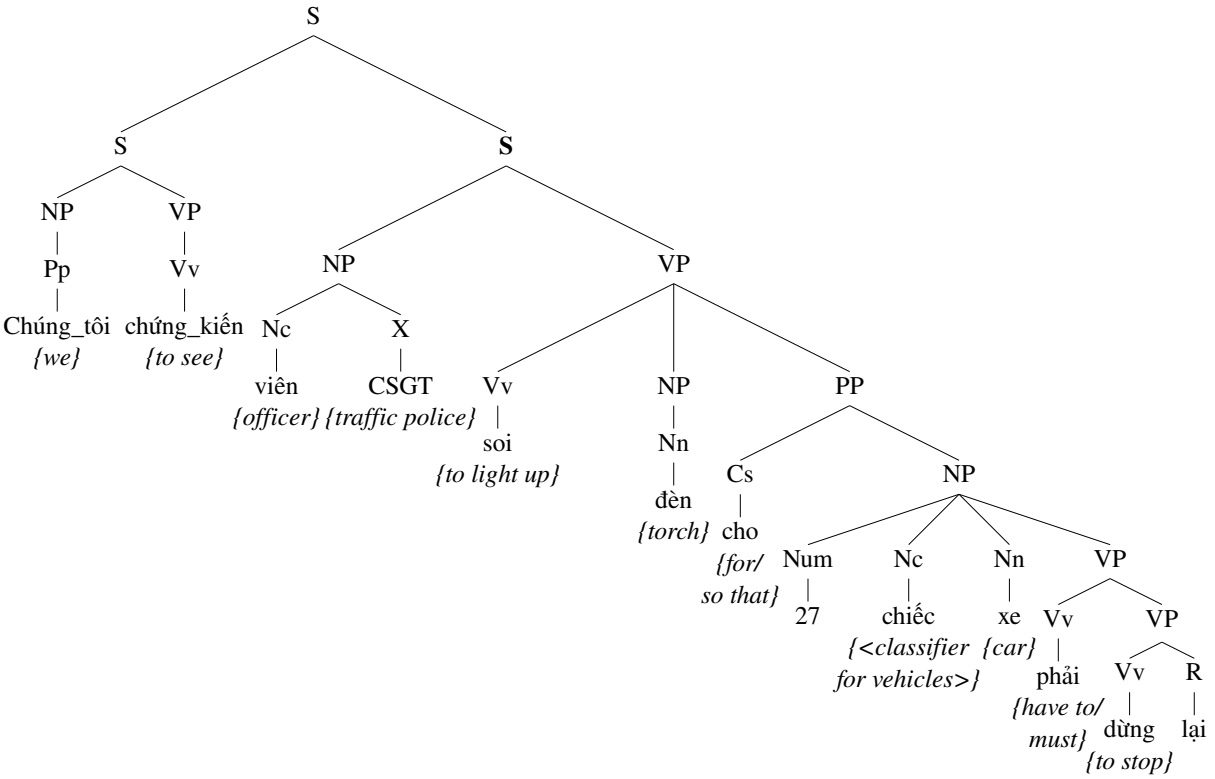
Constructions in which *Ss* or *SBARs* are moved. Figure 7 represents an example of clause attachment error caused by the ambiguous construction of *NP VP NP VP*. We can see that the later *NP VP* should be considered as an *SBAR* modifying the previous verb $chứng\_kiến_{to\ see}$ (gold tree). However, it was attached as the second clause of a compound sentence (parser output).

a) Gold tree



{We saw that the traffic police officer lighted up the torch so that 27 cars have to stop}

b) Parser output



{We saw the traffic police officer lighted up the torch for 27 cars to stop}

Figure 7: Examples illustrating the clause attachment error in Vietnamese parsing.

Learning with Noise-Contrastive Estimation: Easing training by learning to scale.

Matthieu Labeau and **Alexandre Allauzen**
LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, 91405 Orsay, France
{matthieu.labeau,alexandre.allauzen}@limsi.fr

Abstract

Noise-Contrastive Estimation (NCE) is a learning criterion that is regularly used to train neural language models in place of Maximum Likelihood Estimation, since it avoids the computational bottleneck caused by the output softmax. In this paper, we analyse and explain some of the weaknesses of this objective function, linked to the mechanism of *self-normalization*, by closely monitoring comparative experiments. We then explore several remedies and modifications to propose tractable and efficient NCE training strategies. In particular, we propose to make the scaling factor a trainable parameter of the model, and to use the noise distribution to initialize the output bias. These solutions, yet simple, yield stable and competitive performances in either small and large scale language modelling tasks.

1 Introduction

In many tasks, such as machine translation and speech recognition, statistical language models[1] play a key role. Neural models (Bengio et al., 2003; Mikolov et al., 2010; Józefowicz et al., 2016) have recently shown great improvement. However most of them share a common issue: a large output vocabulary implies a prohibitive computation time, due to the output normalization, along with a prediction challenge in a such high dimensional space. A workaround is to reduce the vocabulary size by considering sub-word units like morphemes or even characters. For several applications, this is an efficient solution, though the main issue is not directly addressed. Other trends consist in changing the output structure by using *shortlists* (Schwenk, 2007) or *hierarchical softmax* (Morin and Bengio, 2005; Mnih and Hinton, 2009; Le et al., 2011), while *self-normalisation* techniques (Devlin et al., 2014; Andreas et al., 2015; Chen et al., 2016) partially solve the issue at test time. Finally, sampling-based techniques like *Importance sampling* (Bengio and Sénécal, 2003; Jean et al., 2015), and *Noise-Contrastive Estimation* or NCE (Mnih and Teh, 2012) are also promising alternatives.

In this work, we focus on NCE, which uses a discriminative objective that approximates negative log-likelihood. Its main advantage is to consider the model as *un-normalized* instead of trying to approximate its normalization. With this objective function the model is explicitly learnt to estimate *un-normalized* probability distributions, while the partition function (or the scaling factor) is parametrized separately. While this method is very appealling in theory, empirical issues arise in large vocabulary applications: training divergence and instability, along with poor performance when compared to similar approaches, notably Importance Sampling. The contributions of this paper are twofold: first an empirical exploration of the NCE allows us to better explain the estimation process and why it sometimes fails; then we propose and explore several remedies yielding to tractable and efficient NCE training strategies.

While section 2 reviews two widely used training criteria for un-normalized model, Importance Sampling and NCE, section 3 provides an empirical analysis of the NCE. This algorithm is theoretically proven to converge when the partition function is parametrized separately. However, the scaling parameter

[1]Neural Machine Translation and Speech Recognition can be seen as a language model where the sequence probability is conditioned on a source sentence or a speech signal.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3090–3101
Santa Fe, New Mexico, USA, August 20-26, 2018.

is usually fixed and the *un-normalized* model therefore *self-normalizes*, as a side effect of the training procedure. We show experimentally that the training instability is tied to the difficulty of the model to self-normalize. After analysing the consequences of these issues, we experiment (section 4) with various solutions, including smoothing the sampling distribution or using the recent application of *Negative sampling* to language modelling (Melamud et al., 2017). Moreover, we propose to jointly learn the scaling parameter with the model and show that this approach yields a practical and efficient training strategy. We also propose to initialize the output bias to the logarithm of the noise distribution, which diminishes greatly the impact of the issues described in section 3.

2 Training objectives and partition function

For neural language models (NLMs), the main computational burden lies in the summations over the (very large) vocabulary $\mathcal{V}$. Indeed, a NLM parametrized by θ, outputs, for an input context H, a conditional distribution P_θ^H for the next word, over $\mathcal{V}$. This conditional distribution is defined using the *softmax* activation function:

$$P_\theta(w|H) = \frac{e^{s_\theta(w,H)}}{\sum\limits_{w' \in \mathcal{V}} e^{s_\theta(w',H)}} = \frac{e^{s_\theta(w,H)}}{Z_\theta(H)} \tag{1}$$

Here, $s_\theta(w, H)$ is a scoring function which depends on the network architecture. The denominator is the *partition function* $Z_\theta(H)$, which is used to ensure that for each input context H, output scores are normalized into a probability distribution. Then, each model can be written as an *un-normalized* model divided by the partition function. The natural objective is to minimize the negative log-likelihood of this conditional distribution for each tuple of input context and following word $(H, w) \in \mathcal{D}$, where $\mathcal{D}$ is the training set:

$$NLL(\theta) = -\sum_{(H,w)\in\mathcal{D}} \log P_\theta(w|H) \tag{2}$$

Using the Stochastic Gradient Descent (SGD) to train this objective implies taking the objectives gradient to make the parameter updates. For one training example (H, w), the gradient of the log-probability will be computed as follows:

$$\frac{\partial}{\partial\theta} \log P_\theta(w|H) = \frac{\partial}{\partial\theta} s_\theta(w, H) - \sum_{w' \in \mathcal{V}} P_\theta(w'|H) \frac{\partial}{\partial\theta} s_\theta(w', H). \tag{3}$$

The first term tends to increase the conditional log-likelihood of the word w, whereas the second term lowers probabilities for all the other words in the vocabulary. Unfortunately, the partition function $Z_\theta(H)$ is necessary to compute both $P_\theta(w'|H)$ at test time, and the parameter gradients are necessary to update the model during the training process. In (Gutmann and Hyvärinen, 2013), the authors detail why knowing the partition function, which represents the *'scale'* of the model, is essential to compute the likelihood[2]. Parametrizing separately the partition function (as a scaling parameter) would give irrelevant results, since we could minimize the negative log-likelihood independently of the data. Following this reasoning, we focus on objectives that allow for the estimation of un-normalized models. In the language modelling literature, methods based on Importance Sampling (IS) and NCE are the most widely used.

2.1 IS: approximating the partition function

As detailed in (Bengio and Sénécal, 2003), the idea is to rewrite the gradient described in equation 3 as an expectation that we estimate using importance sampling. We choose a distribution P_n from which it is easy to sample, and obtain the following gradient approximation:

[2]Beyond the fact that the concept of likelihood only applies to probability density functions, which are normalized.

$$\frac{\partial}{\partial\theta}\log P_\theta(w|H) \approx \frac{\partial}{\partial\theta}s_\theta(w,H) - \frac{1}{k}\frac{1}{Z_\theta(H)}\sum_{\substack{i=1\\ \hat{w}_i\sim P_n}}^{k}\frac{e^{s_\theta(\hat{w}_i,H)}}{P_n(\hat{w}_i)}\frac{\partial}{\partial\theta}s_\theta(\hat{w}_i,H) \tag{4}$$

While we replaced one summation over $\mathcal{V}$, the partition function remains. Rewriting it as an expectation, we can apply importance sampling a second time, using the same distribution P_n:

$$Z_\theta(H) = \mathbb{E}[e^{s_\theta(w,H)}] \approx \frac{1}{k}\sum_{\substack{i=1\\ \hat{w}_i\sim P_n}}^{k}\frac{e^{s_\theta(\hat{w}_i,H)}}{P_n(\hat{w}_i)} \tag{5}$$

And thus, by approximating the partition function, we obtain a biased estimator for the maximum likelihood gradient. Both its bias and variance can be reduced by increasing the sample size k. In order to limit this growth, (Bengio and Sénécal, 2008) investigates on adapting P_n during training.

2.2 NCE: avoiding normalization

NCE was first described in (Gutmann and Hyvärinen, 2010), as a way of estimating a parametric probabilistic model from data in the case where the probability function of the model is un-normalized. Considering the partition function as a separate parameter, the objective function mimics maximum-likelihood estimation by learning to discriminate between true examples from data or generated from a *noise distribution*. This method has been applied to language modelling in (Mnih and Teh, 2012): considering a mixture of the data and noise distributions, for each example $(H, w) \in \mathcal{D}$, we draw k noise samples from a noise distribution P_n. The posterior probability of the class C associated to the sample can be estimated ($C = 1$ if the sample comes for the data and $C = 0$ from the noise). Since the goal is to approximate the data distribution with a model parametrized by θ, the conditional class distribution is defined as $P(w|C = 1, H) = P_\theta(w|H)$ and $P(w|C = 0, H) = P_n(w)$, which gives the posterior class probabilities:

$$P(C=1|w,H) = \frac{P_\theta(w|H)}{P_\theta(w|H)+kP_n(w)} \quad \text{and} \quad P(C=0|w,H) = \frac{kP_n(w)}{P_\theta(w|H)+kP_n(w)} \tag{6}$$

If σ denotes the sigmoid function, these equations can be rewritten as a logistic regression problem:

$$P(C=1|w,H) = \sigma\left(\log\frac{1}{k}\frac{P_\theta(w|H)}{P_n(w)}\right) \tag{7}$$

The reformulation obtained in equation 7 shows that training a classifier based on a logistic regression will estimate the log-ratio of two distributions: this allows the learned distribution to be un-normalized, and the partition function to be parametrized separately[3]. However, the partition function is context-dependent. In (Mnih and Teh, 2012), the authors argue that these context-dependent scaling parameters can be put to one, and that given the number of free parameters, the output scores $e^{s_\theta(w,H)}$ will self-normalize for each context. In what follows, we adopt this trick and for clarity denote $p_\theta(w|H) = e^{s_\theta(w,H)}$ the un-normalized model score. Since the class labels are by assumption Bernoulli distributed and independent, the classification objective is given by maximizing the log-likelihood of the true examples to belong to class $C = 1$ and the noise samples $(\hat{w}_i)_{1\leq i\leq k}$ to $C = 0$, which is, for one example (H, w) from $\mathcal{D}$:

$$J_\theta^H(w) = \log\frac{p_\theta(w|H)}{p_\theta(w|H)+kP_n(w)} + \sum_{i=1}^{k}\log\frac{kP_n(\hat{w}_i)}{p_\theta(\hat{w}_i|H)+kP_n(\hat{w}_i)} \tag{8}$$

The gradient update is the following:

[3] (Pihlaja et al., 2012) and (Gutmann and Hirayama, 2011) show the NCE to be a particular case of larger classes of objective functions, with which the model learns to match a ratio of data and noise samples instead of directly matching the data samples, which allows un-normalized model estimation

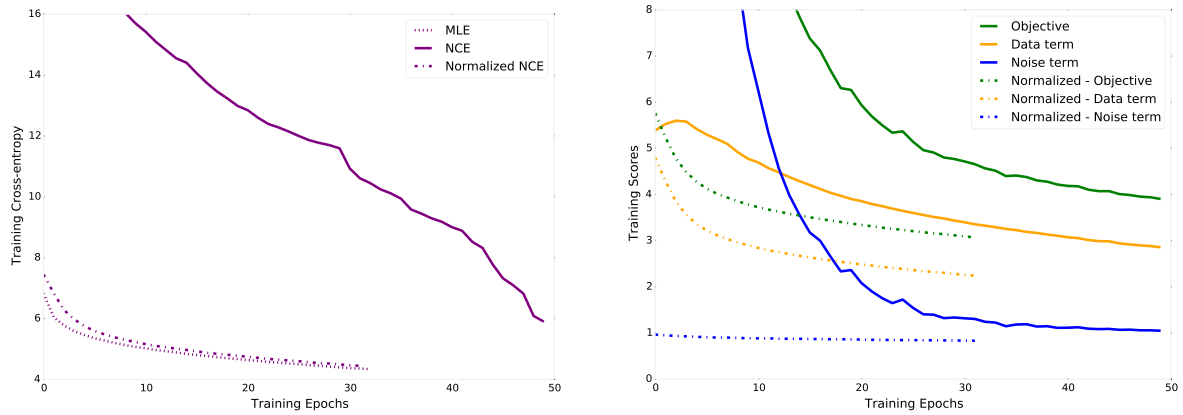


Figure 1: *Left*: Training cross-entropy curves on PTB with Maximum-Likelihood estimation, NCE on an un-normalized model and a model normalized before application of the NCE. *Right*: Training scores (see equation 10) of the same un-normalized and normalized models trained with NCE

$$\frac{\partial}{\partial\theta}J_\theta^H(w) = \frac{kP_n(w)}{p_\theta(w|H) + kP_n(w)}\frac{\partial}{\partial\theta}\log p_\theta(w|H) - \sum_{i=1}^{k}\left[\frac{p_\theta(\hat{w}_i|H)}{p_\theta(\hat{w}_i|H) + kP_n(\hat{w}_i)}\frac{\partial}{\partial\theta}\log p_\theta(\hat{w}_i|H)\right] \tag{9}$$

It is worth noting that this gradient converges to the Maximum-Likelihood Estimation (MLE) gradient as the number of samples k grows. Concerning the choice of the noise distribution, the estimation error of parameters θ is asymptotically independent of P_n when the ratio of noise samples by example k coming from the data is large enough. It is also shown that having both a noise distribution close to the data distribution and a high number of samples k lower the estimation error. (Mnih and Teh, 2012) compared using the uniform and unigram distribution in their experiments, finding the unigram distribution to give far more accurate results. In the literature, NCE was then mainly used in the context of machine translation: (Vaswani et al., 2013; Baltescu and Blunsom, 2015) report results with the unigram distribution, while (Zoph et al., 2016) used an uniform noise. However, despite strong theoretical guarantees, (Chen et al., 2016) highlighted the inconsistency of NCE training when dealing with very large vocabularies, showing very different perplexity results for close loss values. In another work (Józefowicz et al., 2016), NCE was shown far less data-efficient than IS.

3 Training behaviour of NCE

To understand these instabilities, the training process of the same language model is monitored, varying only the training criterion, *i.e* MLE and NCE. For a better understanding, we also consider an '*intermediate*' model denoted as *NCE normalized*. This model is trained using NCE; however, we normalize the scores $p_\theta(w|H) = e^{s_\theta(w,H)}$ into $P_\theta(w|H)$ right before computing the objective. While it is without any practical interest, this model will allow us to better assess the impact of the normalization process. We train the 3 models on the Penn Tree Bank (PTB) with a full vocabulary and $k = 100$ noise samples drawn from the unigram distribution for NCE. We use classic SGD[4]. To reduce the impact of the training criterion, models are learnt for a minimum of 30 epochs. Beyond that limit, we backtrack the epoch when no progress has been made on the validation set perplexity, stopping training after 10 consecutive backtrackings[5].

3.1 The objective and partition functions

The training cross-entropy for the 3 models are drawn in the top graph of figure 1. When trained with MLE or NCE, the normalized model exhibits similar learning curves. However, the un-normalized model

[4]Hyperparameters are the same than those used in (Melamud et al., 2017), except the vocabulary, i.e $\approx 44K$ words, instead of $10K$, to consider a more realistic learning situation for NCE.

[5]For the sake of clarity, backtrakings are discarded from the graphs, keeping only the epochs used to obtain the final model.

trained with NCE takes far longer to reach a comparable cross-entropy, ending with a higher value. In the bottom graph we can observe, for the normalized and un-normalized models trained with NCE, the values of minus the objective function[6], and both of its terms (the first, data-dependent, and the second, containing the noise samples):

$$-J_\theta = \sum_{H,w\in\mathcal{D}} -\log \frac{p_\theta(w|H)}{p_\theta(w|H)+kP_n(w)} - \sum_{H,w\in\mathcal{D}}\sum_{i=1}^{k} \log \frac{kP_n(\hat{w}_i)}{p_\theta(\hat{w}_i|H)+kP_n(\hat{w}_i)} \tag{10}$$

We can observe a small decrease of the second term for the normalized model, whereas for the un-normalized, it starts with high values and decreases to become closer to the normalized one. Moreover, the gap between the two data-dependent terms stays high at the end of training. For a deeper analysis, we study the values of the partition function during training. For both the normalized and the un-normalized models, the partition function $Z_\theta(H)$ associated to each training context H is computed and the results over an epoch are summarized with an histogram. Each bin represents a range of values, and its height represent the fraction of training contexts whose partition function belongs to this range. For a better readability, both range values and bin heights are in log-scale. These histograms are shown in figure 2 for different epochs. While the repartition of the partition function values for the normalized model does not change much - which is logical, since the model is always integrated to 1 - we observe that these values for the un-normalized model are chunked together and are decreasing during learning (epochs 5, 10 and 15). However, this trend seems to slow down towards the end of training, since, at epoch 30, the repartition resemble to the one of the normalized model, only shifted to smaller values (which are still quite high).

3.2 Self-normalisation is crucial for NCE

Considering these results, we dissect what happens during the training of the un-normalized model. Let us rewrite the objective J_θ as function of the following ratio:

$$r_\theta(w|H) = \frac{p_\theta(w|H)}{kP_n(w)}, \text{ then } -J_\theta = \sum_{H,w\in\mathcal{D}} -\log \frac{r_\theta(w|H)}{r_\theta(w|H)+1} - \sum_{H,w\in\mathcal{D}}\sum_{i=1}^{k} \log \frac{1}{r_\theta(\hat{w}_i|H)+1} \tag{11}$$

[6]Since the objective J_θ^H is negative, we minimize $-J_\theta^H$.

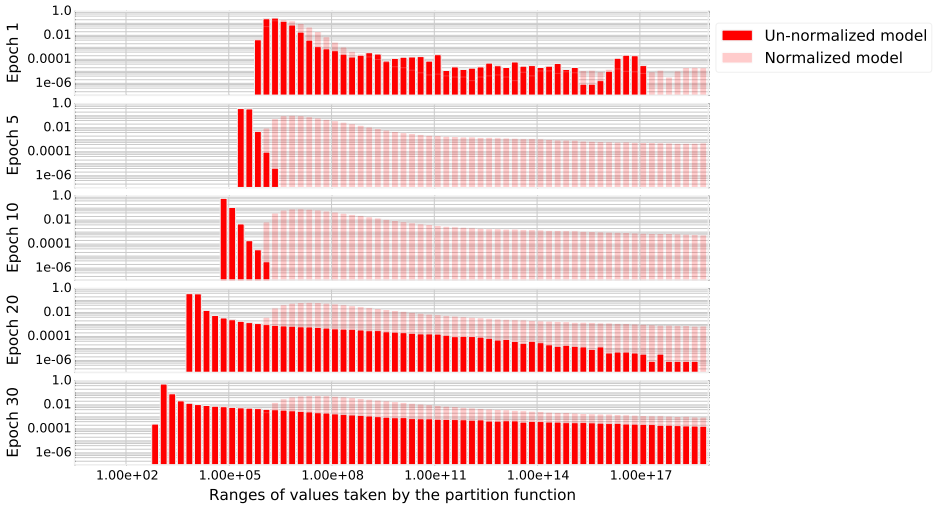


Figure 2: Repartition of the values of the partition function $Z_\theta(H)$ for all examples (H, w) during specific epochs of training on PTB with NCE. The fully coloured bars represent the repartition for the un-normalized model, while the faded bars represent the repartition for the normalized model. Both scales are logarithmic.

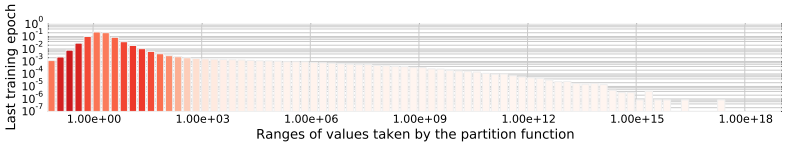


Figure 3: Repartition of the values of the partition function $Z_\theta(H)$ for all training examples (H, w), during the last epoch of training with NCE on PTB. For both this figure and figure 4, the colour of a bin indicates the proportion of training examples (H, w) for which the word w is one of the 10 most frequent according to the noise distribution P_n: the lighter the colour is, the higher is that proportion. Both scales are logarithmic.

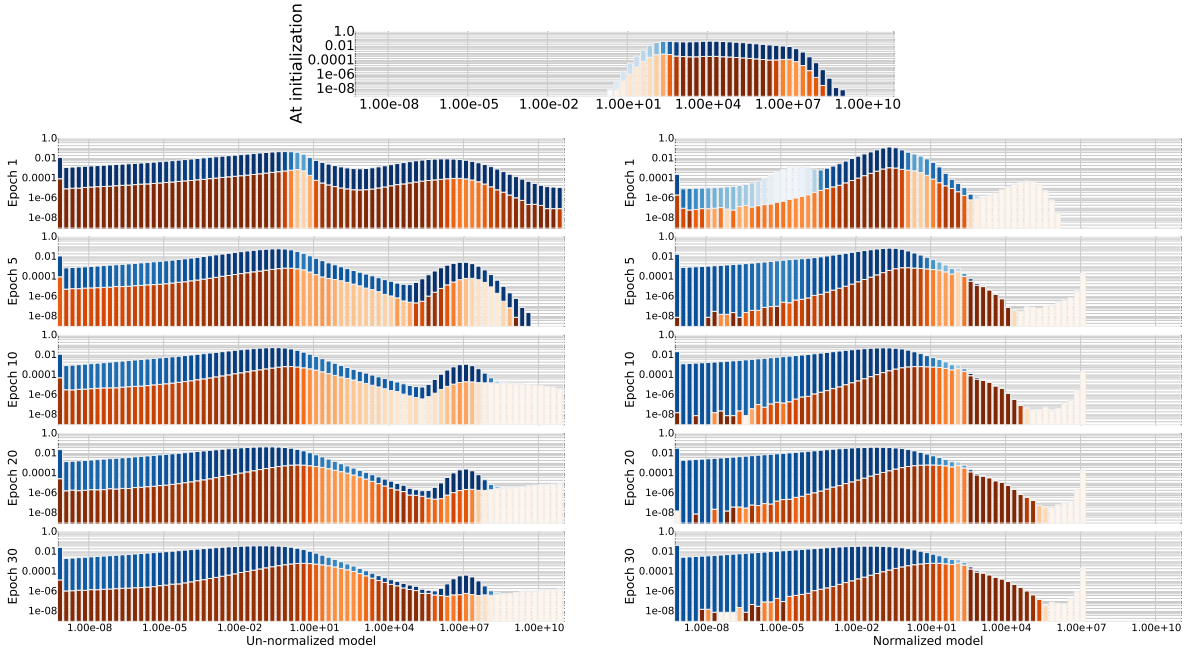


Figure 4: Repartition of the values of the ratio $r_\theta(w|H)$ for all positive training examples (H, w) in orange, and associated noise sample $(\hat{w}_i)_{i=1}^k$, in blue, at initialization (top) and during several training epochs with NCE on PTB. On the left, the model is un-normalized, while on the right it is normalized.

The second term of $-J_\theta$ is very high for large values of r_θ. Since values of P_n are always smaller than 1 and $k = 100$, and knowing the partition function at the beginning of training is large for all contexts, we can assume that $r_\theta(\hat{w}_i|H)$ will be large for a least part of the set of noise samples $(\hat{w}_i)_{i=1}^k$. Thus, this second term is the largest part of the objective, whose minimization leads to a decrease of the model scale. However, this is done at the expense of the data-dependent term, since its score increases a little during the first few epochs. If we look at its gradient, we see from equation 9 that it is always scaled by the weight $1/(r_\theta(w|H) + 1)$, which means that the objective will learn nearly nothing from data as long as the model scale has not decreased to a reasonable value. This shows that this 'self-normalization' mechanism is crucial for NCE, but we still need to understand what makes a scaling value 'reasonable'.

3.3 Impact of the noise distribution P_n

While the un-normalized model underperforms its normalized counterpart, its cross-entropy is not that far off. Let us explore the repartition of the partition function values during the last epoch of training. In particular, we expect an impact of the P_n values. The histogram is shown in figure 3: each bin is coloured according to the proportion of examples associated to the highest values of P_n it contains[7]. We clearly observe that while the majority of the partition functions have values under ≈ 100, functions associated

[7]Since we use the unigram distribution, it indicates here the examples who are among the 10 most frequent words.

with the words with the higher values of P_n - which are also the most sampled words - can take values that are far larger. To understand why, consider the weight of the gradient of the data-dependent term:

$$\frac{1}{r_\theta(w|H)+1} = \frac{kP_n(w)}{p_\theta(w|H)+kP_n(w)}.$$

Having higher values of P_n means that the model does not need to scale back the partition function as much to be able to learn from data. Besides, if the noise distribution is correlated with the data distribution, an example with a high value of P_n tends to be more frequent. Therefore the model will be able to learn first from words w with the higher values of $P_n(w)$. However, the lower $P_n(w)$ is, the more difficult it will be for the model to learn about w. And the associated partition function will have to be closer to 1 for the data-term gradient to be meaningful. To visualize how learning is affected, we study histograms showing the repartition of the values of r_θ during a training epoch. They are presented in figure 4. Again, we show histograms for both models, at epochs 1, 5, 10, 20, and 30, while on top is a histogram of the initial repartition (before learning). Similarly, they are in log-scale. On the right are histograms for the normalized model. For epoch 1, there is a peak of data examples with higher values of P_n - frequent words - that have higher values of r_θ, while for noise samples, the repartition still follows the initial one - samples with higher P_n have a smaller r_θ. As learning progresses, a clear trend appears: values of r_θ for data examples, seemingly ordered by values of P_n, become well-separated of the values of r_θ for noise samples. On the left are the repartition of values for the un-normalized model. While they are ordered by values of P_n, a clear separation between data examples values and noise samples values takes far more time to appear, and a cluster of high values of r_θ for noise samples still remains visible at epoch 30 - these high values correspond to small values of P_n: in our case, rare words.

These observations show us that an higher discrepancy between high and low P_n leads to a more difficult learning procedure with NCE. As a consequence, and because of the Zipfian distribution of word frequencies, using the unigram distribution will be harder as the size of the vocabulary grows.

4 Practical solutions and learning to scale

As discussed in section 3, two quantities strongly impact NCE training: first, the noise distribution (and the number k of noise samples); then, the partition function. To evaluate different training strategies, we provide training cross-entropy curves in figure 5. As a comparison point, results with importance sampling are also reported since it is nowadays the best choice to train large language models. To verify the validity of our analysis on the PTB, we also report results on a larger corpus, the 1 Billion Word Benchmark[8] (Chelba et al., 2014). In this case, the vocabulary contains $64K$ words, which renders normalized models not competitive[9]. We trained the models for at least one epoch, and up to a second if models made progress on the validation set perplexity. Final perplexity results are presented in table 1.

4.1 Acting on the noise distribution P_n

As the number of noise samples k increases, the estimation error reduces, along with the necessity for the model to downscale the partition function. With $k = 500$, we can observe indeed a faster convergence with a lower final perplexity. On the large corpus, the conventional NCE model doesn't even reach a perplexity under the size of the vocabulary, whereas augmenting k to 500 is not enough to reach a suitable perplexity. However, increasing k reduces the computational benefit of NCE, even with k far inferior to the vocabulary size. There is a trade-off in the choice of P_n: the closer P_n is to the data distribution, the lower is the estimation error; however the Zipfian distribution of word frequencies implies too few updates for rare words if P_n is the usual unigram distribution. Therefore P_n is often distorted into P_n^α, with a parameter $0 \leq \alpha \leq 1$. This smoothing helps reducing the range of values of P_n, and α is usually set to 0.75. With this choice, convergence is indeed faster, but the final perplexity is not that better[10]. However,

[8] We also used the same hyperparameter settings than in (Melamud et al., 2017)

[9] Besides the computation time constraints, the batch size is rapidly limited by the GPU memory, while a smaller batch size also increases the computation time.

[10] We should note that for both these experiments, convergence being faster would imply reducing the number of epochs without backtracking, which we didn't do here.

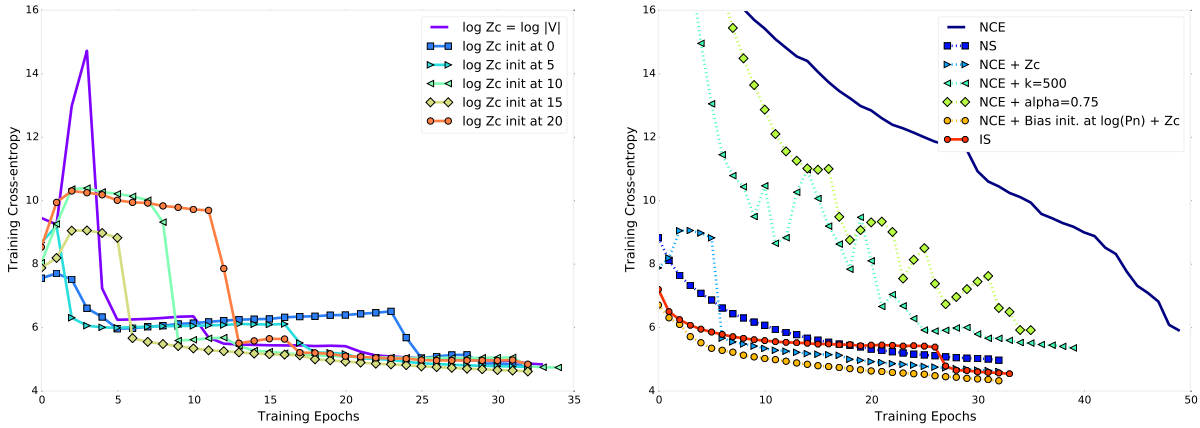


Figure 5: *Left*: Training cross-entropy curves on PTB for un-normalized models trained with NCE while fixing, then learning a parameter Z_c that shifts the partition function, depending on the initial value of Z_c, as presented in section 4.2. *Right*: Training cross-entropy curves on PTB for un-normalized models trained with Importance Sampling, NCE and alternative additions presented in section 4

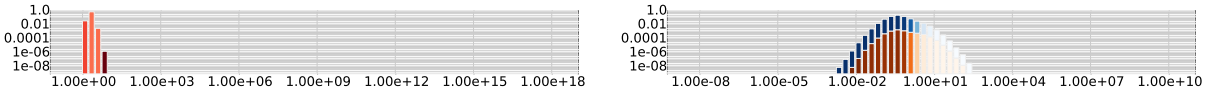


Figure 6: Repartition of the values of the partition function $Z_\theta(H)$ (*left*) and ratio $r_\theta(w|H)$ (*right*) for all training examples (H, w) at initialization, for a model whose output bias is initialized with $\log P_n$.

on the large corpus, this solution fails to stabilize training.

4.2 Shifting the partition function and learning to scale

Following a method introduced by (Chen et al., 2015), we could simply shift all values of the partition function by a fixed value Z_c. In practice, it means we set

$$p_\theta(w|H) = \frac{e^{s_\theta(w,H)}}{Z_c}.$$

Another similar workaround consists in initializing the output bias to $-\log|\mathcal{V}|$ (Vaswani et al., 2013; Melamud et al., 2017). It has the same effect as fixing $Z_c = |\mathcal{V}|$ and initializing the bias to 0. In (Chen et al., 2015) this value is set to $\log Z_c = 9$. In this paper we propose to learn the best shifting value by adding Z_c to the model parameters θ. In this case Z_c can be seen as an extra bias term that learns a normalizing constant. The left graph of figure 5 shows training cross-entropy curves depending on the initial value of Z_c, revealing its impact. On both corpora, learning to scale allows the model to achieve a better perplexity than fixing it to an arbitrary *e.g.* $|\mathcal{V}|$.

4.3 The case of Negative sampling

The *Negative Sampling* (NS) algorithm was popularized by the skip-gram embedding algorithm (Mikolov et al., 2013), and while closely linked to NCE, is not able to directly optimize the likelihood of a language model and learn conditional probabilities, as detailed in (Dyer, 2014). Recently, (Melamud et al., 2016; Melamud et al., 2017) showed Negative Sampling to be viable for language modelling. As previously, we use k samples $(\hat{w}_i)_{1\leq i\leq k}$ from the unigram distribution. The objective maximizes the likelihood of a logistic regression that discriminates true examples from noise samples. However, here, the model directly parametrizes the log-odds, since $P(C = 1|w, H) = \sigma\left(s_\theta^{NS}(w, H)\right)$, which gives the scoring function:

$$J_\theta^H(w) = \log\sigma(s_\theta^{NS}(w, H)) + \sum_{i=1}^{k} \log\sigma(-s_\theta^{NS}(\hat{w}_i, H)) \tag{12}$$

Training method	PTB	1BW Benchmark
MLE	150.2	-
Normalized NCE	159.3	-
NCE	306.0	X
NCE + $\alpha = 0.75$	277.0	X
NCE + $k = 500$	231.9	X
Importance Sampling	168.3	80.2
NS	228.3	99.0
NS + $\alpha = 0.75$	195.8	96.1
NCE + $Z_c = \|\mathcal{V}\|$	178.6	79.2
NCE + $Z_c \in \theta$	172.3	76.6
NCE + Bias init. at $\log(P_n)$	151.8	76.0
NCE + Bias init. at $\log(P_n)$ + $Z_c \in \theta$	148.4	74.7

Table 1: Best testing perplexities obtained on PTB with a full vocabulary, and on the 1 Billion Word Benchmark with a $64K$ words vocabulary. '-' indicates that the models were not trained due to technical limitations, and 'X' indicates that the model did not converge to a perplexity value under $|\mathcal{V}|$.

Therefore, this objective function corresponds to the log-ratio $\log r_\theta(w|H)$ for the NCE. Indeed, we can get back an estimation of the conditional probability using the model score and the noise distribution:

$$\hat{P}(w|H) \propto P_n(w) \exp\left(s_\theta^{NS}(w|H)\right) \tag{13}$$

The model directly learns what is, in the NCE, the ratio of the data and noise distribution, without knowing the values of P_n. The initial values of $r_\theta(w|H) = e^{s_\theta^{NS}(w,H)}$ simply depend on the model initialization, and the initial values of the partition function are close to 1. While the final perplexity values are not among the best[11], we can observe on the bottom graph of figure 5 a far smoother learning curve than that the ones for methods previously presented. More precisely, we avoid the initial increase in cross-entropy that is visible when we fix or learn Z_c (on the top graph). We believe this is due to having an initial model distribution very close to P_n, which is a consequence of multiplying final scores by P_n, as showed in equation 13. To verify this, we trained a NCE model for which we initialized the output bias with values of $\log P_n(w)$. It gave great results, especially on the smaller corpus. By looking at the repartitions of the values of the ratio and partition function at initialization, showed in figure 6, we can check that having $p_\theta(w|H) \approx P_n(w)$ mitigates the main issues described in section 3: first, our distribution is far closer to be normalized; then, the values of $r_\theta(w|H)$ are bundled together around $\frac{1}{k+1}$. This method clearly helps learning; furthermore, when used in complement to learning Z_c, it gives the best model on the PTB. It is however less impactful on the larger corpus. We believe it is due to the fact that the vocabulary is cut at $64K$ words and is thus very small for a corpus of this size: the frequency distribution does not have the Zipfian long tail of rare words, which mitigates the issue we try to solve with this initialization strategy.

4.4 Summary of training recommendations

To have a clearer idea of which solution to use depending on the size of the corpus and of the vocabulary, we experiment with the solutions presented earlier on smaller parts of the 1 Billion Word Benchmark, and with different vocabulary sizes. Testing cross-entropy curves are shown in figure 7. The top figure shows the experiments whose results we presented in table 1, with the full corpus and with $|\mathcal{V}| = 64K$. The two leftmost figures show experiments with 10 and 100 times less data, while the center and rightmost figures show experiments with smaller and larger vocabularies on the full corpus. The main noticeable result is that learning Z_c as a parameter is not working well when we have less data - in this case, initializing the output bias to $-\log|\mathcal{V}|$ is far more efficient, while initializing output bias with values of $\log P_n(w)$ gives even better results. However, this last method loses efficiency when the vocabulary becomes very

[11] Given the recommendation given in (Mikolov et al., 2013) on using $\alpha = 0.75$ with Negative Sampling, we suppose smoothing the noise distribution has an important impact, and added corresponding results, which are indeed better, to table 1.

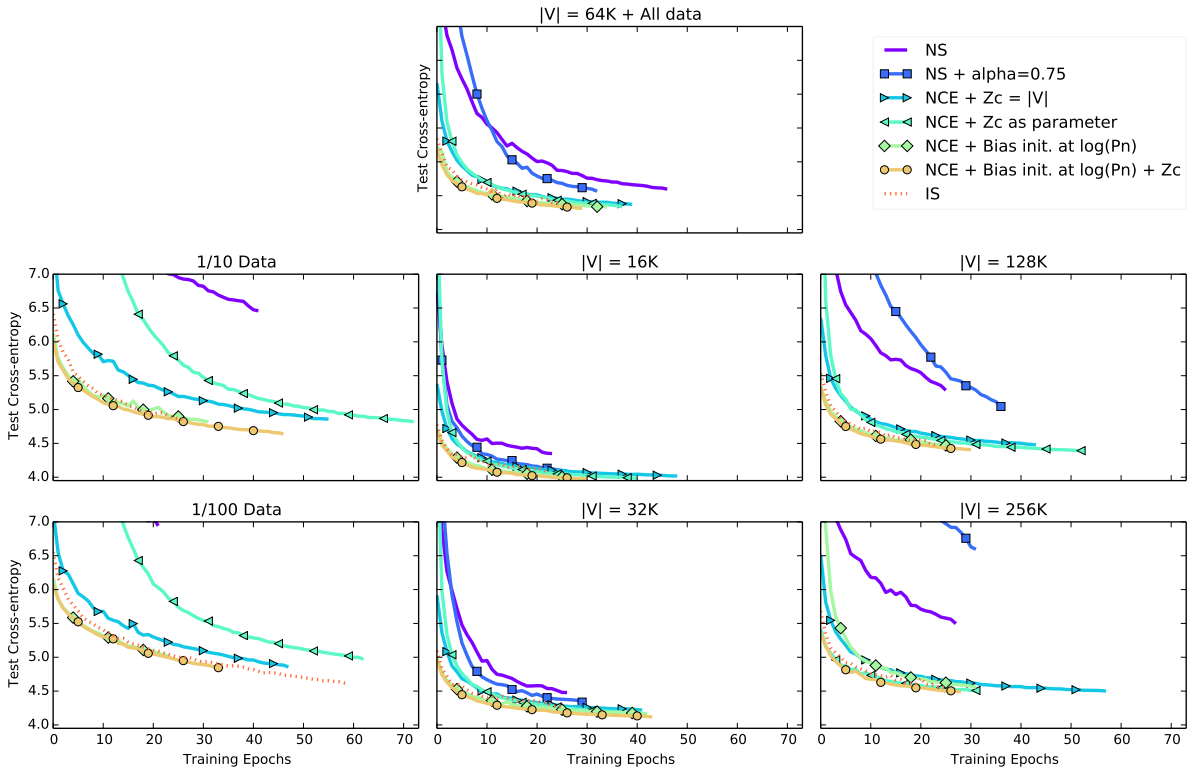


Figure 7: Testing cross-entropy curves on 1BW for un-normalized models trained with Importance Sampling, and NCE-based alternative additions presented in section 4 The seven figures show experiment with various vocabulary sizes $|\mathcal{V}|$ and quantities of Data.

large (here, for $|\mathcal{V}| = 256K$. Finally, combining this initialization scheme and the learning of Z_c as a parameter gives the best results accross all configurations.

5 Conclusion

Training neural language models with large output vocabularies is still challenging. Noise contrastive estimation is one of the most promising training criterion but was empirically shown to be unstable. In this paper, we carried out an extensive analysis of the training process of NCE. We showed how setting the scaling parameters to 1 yields an implicit self-normalization step which necessarily precedes actual learning from data examples. The difficulty of this process depends on the value range of the noise distribution P_n. Given the Zipfian distribution of word frequencies texts, learning with an unigram distribution is getting harder as the vocabulary size grows. This is quite inconsistent with the motivation of this learning criterion. However, we also explored several remedies and showed that when including the scaling parameter in the parameters of the model, the scaling process can be both stable and efficient. Additional improvements can be obtained by initializing the bias of the output layer according to the noise distribution. Combining this 'learning to scale' approach with the adapted initialization scheme yields a very competitive, yet simple, training strategy for neural language models with large vocabularies.

Acknowledgements

We wish to thank the anonymous reviewers for their helpful comments. This work has been funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 645452 (QT21).

References

[Andreas et al.2015] Jacob Andreas, Maxim Rabinovich, Michael I. Jordan, and Dan Klein. 2015. On the accuracy of self-normalized log-linear models. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1783–1791.

[Baltescu and Blunsom2015] Paul Baltescu and Phil Blunsom. 2015. Pragmatic neural language modelling in machine translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 820–829, Denver, Colorado, May–June. Association for Computational Linguistics.

[Bengio and Sénécal2003] Yoshua Bengio and Jean-Sébastien Sénécal. 2003. Quick training of probabilistic neural nets by importance sampling. In *Proceedings of the conference on Artificial Intelligence and Statistics (AISTATS)*.

[Bengio and Sénécal2008] Yoshua Bengio and Jean-Sébastien Sénécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Trans. Neural Networks*, 19(4):713–722.

[Bengio et al.2003] Yoshua Bengio, Rjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

[Chelba et al.2014] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 2635–2639.

[Chen et al.2015] Xie Chen, Xunying Liu, Mark J. F. Gales, and Philip C. Woodland. 2015. Recurrent neural network language model training with noise contrastive estimation for speech recognition. In *ICASSP*, pages 5411–5415. IEEE.

[Chen et al.2016] Wenlin Chen, David Grangier, and Michael Auli. 2016. Strategies for training large vocabulary neural language models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1975–1985, Berlin, Germany, August. Association for Computational Linguistics.

[Devlin et al.2014] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380, Baltimore, Maryland, June. Association for Computational Linguistics.

[Dyer2014] Chris Dyer. 2014. Notes on noise contrastive estimation and negative sampling. *CoRR*, abs/1410.8251.

[Gutmann and Hirayama2011] Michael Gutmann and Junichiro Hirayama. 2011. Bregman divergence as general framework to estimate unnormalized statistical models. In *UAI*.

[Gutmann and Hyvärinen2010] Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, pages 297–304.

[Gutmann and Hyvärinen2013] M.U. Gutmann and A. Hyvärinen. 2013. Estimation of unnormalized statistical models without numerical integration. In *Proceedings of the Workshop on Information Theoretic Methods in Science and Engineering*.

[Jean et al.2015] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China, July. Association for Computational Linguistics.

[Józefowicz et al.2016] Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

[Le et al.2011] Hai-Son Le, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and Francois Yvon. 2011. Structured output layer neural network language model. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP)*, pages 5524–5527, Prague, Czech Republic.

[Melamud et al.2016] Oren Melamud, Ido Dagan, and Jacob Goldberger. 2016. PMI matrix approximations with applications to neural language modeling. *CoRR*, abs/1609.01235.

[Melamud et al.2017] Oren Melamud, Ido Dagan, and Jacob Goldberger. 2017. A simple language model based on PMI matrix approximations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1861–1866, Copenhagen, Denmark, September. Association for Computational Linguistics.

[Mikolov et al.2010] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

[Mikolov et al.2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

[Mnih and Hinton2009] Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1081–1088. Curran Associates, Inc.

[Mnih and Teh2012] Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*.

[Morin and Bengio2005] Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 246–252. Society for Artificial Intelligence and Statistics.

[Pihlaja et al.2012] Miika Pihlaja, Michael Gutmann, and Aapo Hyvärinen. 2012. A family of computationally efficient and simple estimators for unnormalized statistical models. *CoRR*, abs/1203.3506.

[Schwenk2007] Holger Schwenk. 2007. Continuous space language models. *Comput. Speech Lang.*, 21(3):492–518, July.

[Vaswani et al.2013] Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392, Seattle, Washington, USA, October. Association for Computational Linguistics.

[Zoph et al.2016] Barret Zoph, Ashish Vaswani, Jonathan May, and Kevin Knight. 2016. Simple, fast noise-contrastive estimation for large RNN vocabularies. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1217–1222, San Diego, California, June. Association for Computational Linguistics.

Parallel Corpora for bi-lingual English-Ethiopian Languages Statistical Machine Translation

Solomon Teferra Abate[1], Michael Melese Woldeyohannis[1], Martha Yifiru Tachbelie[1], Million Meshesha[1], Solomon Atinafu[1], Wondwossen Mulugeta[1], Yaregal Assabie[1], Hafte Abera[1], Biniyam Ephrem[1], Tewodros Abebe[1], Wondimagegnhue Tsegaye[2], Amanuel Lemma[3], Tsegaye Andargie[4], Seifedin Shifaw[4]

[1]Addis Ababa University, Addis Ababa, Ethiopia, [2]Bahir Dar University, Bahir Dar, Ethiopia
[3]Aksum University, Axum, Ethiopia, [4]Wolkite University, Wolkite, Ethiopia
{solomon.teferra, michael.melese, martha.yifiru, million.meshesha, solomon.atnafu, wondwossen.mulugeta, yaregal.assabie, hafte.abera, binyam.ephrem, tewodros.abebe}@aau.edu.et, {wendeal, amanu.infosys, adtsegaye, seifedin28}@gmail.com

Abstract

In this paper, we describe an attempt towards the development of parallel corpora for English and Ethiopian Languages, such as Amharic, Tigrigna, Afan-Oromo, Wolaytta and Ge'ez. The corpora are used for conducting a bi-directional statistical machine translation experiments. The BLEU scores of the bi-directional Statistical Machine Translation (SMT) systems show a promising result. The morphological richness of the Ethiopian languages has a great impact on the performance of SMT specially when the targets are Ethiopian languages. Now we are working towards an optimal alignment for a bi-directional English-Ethiopian languages SMT.

1 Introduction

The advancement of technology and the rise of the internet as a means of communication led to an ever increasing demand for Natural Language Processing (NLP) applications. NLP applications are useful in facilitating human-machine and human-human communications. One of the NLP applications which facilitates human-human communication is Machine Translation (MT). MT refers to a process by which computer software is used to translate a text or speech from one language to another (Koehn, 2009). In the presence of high volume digital text, the ideal aim of machine translation systems is to produce the best possible translation with minimal human intervention (Hutchins, 2005).

The translation of natural language by machine becomes a reality, for technologically favored languages, in the late 20^{th} century although it is dreamt in seventieth century (Hutchins, 1995). Various approaches to MT have been and are being used in the research community, that can broadly classified as rule-based and corpus based (Koehn, 2009). The rule based machine translation demands various kinds of linguistic resources such as morphological analyzer and synthesizer, syntactic parsers, semantic analyzers and so on. On the other hand, corpus based approaches (as the name implies) require parallel and monolingual corpora. Since corpus based approaches do not require deep linguistic analysis of the source and target languages, it is the preferred approach for under-resourced languages of the world, including Ethiopian languages.

1.1 Machine Translation For Ethiopian Languages

Research in the development of MT has been conducted for technologically favored and economically as well as politically important languages of the world since the 17^{th} century. As a result, notable progress towards the development and use of MT systems has been made for these languages. However, research in the area of MT for Ethiopian languages, which are under-resourced as well as economically and technologically disadvantaged, has started very recently. Most of the

Proceedings of the 27th International Conference on Computational Linguistics, pages 3102–3111
Santa Fe, New Mexico, USA, August 20-26, 2018.

researches on MT for Ethiopian languages are conducted by graduate students (Tariku, 2004; Sisay, 2009; Eleni, 2013; Jabesa, 2013; Akubazgi, 2017), including two PhD works: one that tried to integrate Amharic into a unification based machine translation system (Sisay, 2004) and the other that investigated English-Amharic Statistical Machine Translation (Mulu, 2017). Beside these, Michael and Million (2017) attempted a bi-directional Amharic-Tigrigna SMT experiement using word and morpheme as translation units.

Due to unavailability of linguistic resources and since the most widely used MT approach is statistical, most of these researches have been conducted using SMT, which requires large bilingual and monolingual corpora. However, as there were no such corpora for SMT experiments, the researchers had to prepare small size corpora. This in turn, affects the results that they obtain.

In addition, since there is no standard corpora for conducting replicable and consistent experiment for performance evaluation, it is difficult to know the progress made in the area for local languages. Moreover, since the researchers spend their time on corpora preparation, they usually have limited time for experimentation, exploration and development of MT systems.

1.2 Motivation of this paper

African languages, which contribute around 30% (2139) of the world languages, highly suffer from lack of sufficient language resources (Simons and Fennig, 2017). This is true for Ethiopian languages as well. On the other hand, Ethiopia being multilingual and multiethnic country, its constitution decrees that each citizen has the right to speak, write and develop in his/her own language. However, a lot of written documents, brochures, text books, magazines, advertisements and other information in the web are being produced in technological favored and economically important languages such as English.

In order to enable Ethiopians to use the documents and information produced in technologically favored languages, the documents need to be translated. Since manual translation is expensive, a promising alternative is the use of machine translation, particularly SMT as Ethiopian languages suffer from lack of basic linguistic resources such as morphological analyzer, syntactic analyzer, morphological synthesizer, etc. The major and basic resource required for SMT is parallel corpora, which are not available for Ethiopian languages. The collection and preparation of parallel corpora for Ethiopian languages is, therefore, an important endeavor to facilitate future MT research and development. Corpus acquisition for SMT is actually one of the recommendations of Saba and Sisay (2006).

We have, therefore, collected and prepared parallel corpora for English and Ethiopian Languages that fall under the Semitic, Cushitic and Omotic language families. We have considered Amharic, Tigrigna and Ge'ez from the Semitic, Afan-Oromo from the Cushitic and Wolaytta from Omotic language families. This paper, therefore, describes an attempt that we have made to collect and prepare English-Ethiopian languages parallel corpora and the SMT experiments conducted using the corpora.

2 Nature of the language pairs

The language pairs in the corpora belong to Semitic (Ge'ez, Amharic and Tigrigna), Cushitic (Afan-Oromo) and Omotic (Wolaytta) language families. Except Ge'ez, these languages have native speakers. Presently, Ge'ez does not have native speakers. Ge'ez functions as a liturgical language of Ethiopian Orthodox Church. It is thought as a second language in traditional schools of churches and given as a course in different Universities. There is a rich body of literature in Ge'ez. It is not only literature but also philosophical, medical and astrological documents were written in Ge'ez. Because of this, there is a big initiative in translating those documents written in Ge'ez. On the other hand, Amharic is spoken by more than 27 million people which makes it the second most spoken Semitic language. Tigrigna is spoken by 9 million people. Afan-Oromo and Wolaytta are spoken by more than 34 million and 2 million speakers, respectively (Simons

and Fennig, 2017).

The writing systems of these language pairs are Ge'ez or Ethiopic script and Latin alphabet. Ge'ez, Amharic and Tigrigna are written in Ge'ez script whereas both Afan-Oromo and Wolaytta are written in Latin alphabet. It is believed that the earliest known writing in the Ge'ez script dated back to the 5^{th} century BC. The writing system is syllabry where each character represents a consonant and vowel. The basic features of the writing system is that each character gets its basic shape from the consonant of the syllable, and the vowel is represented through a systematic modifications of these basic shapes. The script is used to write other languages like Amharic, Tigrigna, Argoba, etc.

The Ethiopian languages considered in the project have got different functions in the country. Amharic for instance is the working language of the Federal Government of Ethiopia. It also serves as regional working language of some other regional states. It facilitates inter-regional communication as well. Tigrigna and Afan-Oromo are working language in Tigray and Oromiya regional administrations, respectively. Some of the governmental websites are available in Amharic, Tigrigna and Afan-Oromo. Apart from this, they serve as medium of instructions in primary and secondary schools. These languages are available in electronic media like news, blogs and social media except Ge'ez. Currently, Google offers a searching capability using Amharic, Tigrigna and Afan-Oromo. Further, Google also included Amharic in its translation service recently.

2.1 Morphological features

As in other Semitic language morphology, Ge'ez (Dillman, 1907), Amharic (Leslau, 2000; Teferra and Hudson, 2007) and Tigrigna (Mason, 1996; Yohannes, 2002), make use of the root and pattern system. In these languages, a root (which is called a radical) is set of consonants which bears the basic meaning of the lexical item whereas a pattern is composed of a set of vowels inserted between the consonants of the root. These vowel patters together with affixes results in derived words. Such derivational process makes these language to be morphologically complex languages.

In addition to the morphological information, some syntactic information are also expressed at word level. Furthermore, an orthographic word may attach some syntactic words like prepositions, conjunctions, negation, etc. which make word forms to be very varied (Gasser, 2010; Gasser, 2011). In these languages, nominals are inflected for number, gender, definiteness and case whereas verbs are inflected for person, number, gender, tense, aspect, and mood.

As we may observe in the Semitic languages, nominals are inflected for number, gender, case and definiteness and verbs are inflected for person, number, gender, tense, aspect and mood (Griefenow-Mewis, 1995). Essentially, unlike the Semitic languages which allow prefixing, Afan-Oromo allows suffixing. Most functional words like pospositions are also suffixed. However, there are some prepositions written as a separate word.

Wolaytta like Afan-Oromo is a suffixing language in which words can be generated from root words recursively by adding suffixes only. Wolaytta nouns are inflected for number, gender and case whereas verbs are inflected for person, number, gender, aspect and mood (Wakasa, 2008).

2.2 Syntactic Features

Ethiopian languages that are under our consideration follow Subject-Object-Verb (SOV) word-order except Ge'ez which allows the verb to come first. In Ge'ez, the basic word-order is Verb-Subject-Object (VSO). On the contrary, English language uses Subject-Verb-Object (SVO) word-order.

3 Challenges of SMT

Statistical machine translation is greatly affected by the linguistic features of the target languages. The challenges ranges from the writing system to that of word ordering and morphological complexity.

3.1 Writing System

Semitic language writing system are represented by a consonant vowel (CV) sequence and the basic shape of each character is determined by the consonant, which is modified for the vowel. These language script has inherited its writing system from Ge'ez (ግእዝ) /gə'əzə/ using a grapheme based writing system called fidel /fidälə/ which is written and read from left to right being the classical and ecclesiastical language of Ethiopia. Unlike the Semitic languages, The Cushitic (Afan-Oromo) and Omotic (Wolaytta) languages use a latin based writing system.

3.2 Word Ordering

Semitic languages like Amharic, Tigrigna and Ge'ez are morphologically rich where words are required to be further segmented or a single word from these languages would be aligned with as big as handful of words in languages like English. The languages under consideration have different word order. With this respect, Amharic, Afan-Oromo, Tigrigna and Wolaytta have SOV, Ge'ez has VSO and English has SVO topology. The different word orders used in these languages is major challenge for Multilingual machine translation system.

Another challenge is the existence of flexibility in word order. For instance, even though Afan-Oromo follows SOV word order format, nouns can be changed based on their role in a sentence which makes the word order to be flexible. Although the major word order of Ge'ez is VSO, it also follows free word order. Such flexibilities will pose another challenge for translation from source to Afan-Oromo and Ge'ez languages.

3.3 Morphological Complexity

While word alignment could be done automatically or with supervision, morphological agreement between words in the source and target are crucial. For instance, Amharic and Ge'ez have subject agreement, object agreement and genitive (possessive) agreement. Each of which are expressed as bound morphemes which should be aligned or translated as independent words in English. In Amharic, for the word **ገደልክ** /you killed/ the English subject "you" is represented by the suffix "+**ክ**" while the same subject is represented as "+**ከ**" in the Ge'ez (**ቀተልከ** /you killed/). Likewise, the definite marker for Amharic and English are quite different in their representation. While it is a bound morpheme in Amharic, it is a word (free morpheme) in English. Most of the local languages under consideration falls into this group.

4 Parallel Corpus preparation

The development of machine translation more often uses statistical approach because it requires very limited computational linguistic resources compared to the rule-based approach. Nevertheless, the statistical approach relies to a great extent on parallel corpora of the source and target languages.

The research team has applied different techniques to collect parallel corpora for the selected Ethiopian languages paired with English. The collected data fall under the religious, historical and legal domains.

The religious domain include Holy Bible and different documents written in spiritual theme and collected from Jehovah's Witnesses (JW[1]), Ethiopicbible[2], Ebible[3] and Ge'ez experience[4] which are freely available websites.

The historical domain is from one source which is the handbook of Africa ("African Almanac"). The source is griped from admase ethiopia github[5].

The legal domain includes documents collected from Ethiopian constitution, Proclamation and Regulation documents which are available for different period of time and languages (Amharic,

[1]available at `https://www.jw.org`
[2]available at `https://www.ethiopicbible.com`
[3]available at `http://ebible.org`
[4]available at `https://www.geezexperience.com`
[5]Corpus available at https://github.com/admasethiopia/parallel-text/

Tigrigna and Afan-Oromo aligned with English). The documents are taken from Ethiopian legal brief website.

Legal and historical domain data collected from sources specified above are available in text and pdf format. For the sources in pdf, a pdf miner tool is used for extracting texts.The contents in the pdf files are stored in multiple columns with a language per column. By using a Unicode range of characters, the contents in each column were extracted without distorting the sentence sequence. For the corpus in the religious domain, a simple web crawler was used to extract parallel text from targeted websites.

Python libraries such as requests and BeautifulSoup were used to analyze the structure of the website, extract texts and combine to a single text file. To collect the bible data, we have generated the structure of the URL so that it shows the book names, chapters and verse numbers of Bible in each language.

For the daily text which is published at Jehovah Witnesses (JW), we tried to use the date information to generate URL for each language. The page was requested to extract the data we are interested in. Finally, we organized and merged the data to a single UTF-8 text files for each language.

We could have all these domains only for a language pair Amharic-English. The Tigrigna-English and Afan Oromo-English corpora are in legal and religious (both bible and other religious collections) domains. The Wolaytta-English and Ge'ez-English language pairs are from the religious domain only. However, the Ge'ez-English corpius is only from Bible while the Wolaytta-English consists of Bible and other religious collections.

4.1 Preprocessing

Data preprocessing is an important and basic step in preparing bilingual and multilingual parallel corpora. Since the collected parallel data have different formats and characteristics, it is very difficult and time-consuming to prepare manually. To produce parallel corpus there is a need to analyze the structure of collected raw data by applying different techniques.

During preprocessing the following tasks have been performed: character normalization, sentence tokenization and alignment.

4.1.1 Character Normalization

There are characters in Amharic that have similar roles and are redundant. Characters ሀ, ሐ and ኀ; ሠ and ሰ; አ and ዐ as well as ጸ and ፀ are variants along with their orders. These characters are used interchangeably. To avoid words with the same meaning from being taken as distinct words due to these character variants, we have replaced a set of characters with similar function into a single most frequently used character.

As a result of normalizing character variants in Amharic text , reduction in the number of word types (vocabulary) has been obtained. Table 1 presents the vocabulary reduction of training, development and testing dataset. As can be seen from the table, the vocabulary size reduced by 15.78 % for training, 10.53% for development and 11.95% for testing from a total of 40,726 sentence (132,723 Token of 628,474 type).

	Word Normalization		**Percentage reduction**
	Before	**After**	
Training	98,784	83,196	15.78 %
development	23,701	21,207	10.53%
Testing	24,142	21,258	11.95%

Table 1: Amharic text corpus before and after character normalization.

4.1.2 Sentence Tokenization and Alignment

Lines that contain multiple sentences in both source and target languages are tokenized. The team have set two criteria to check whether the aligned sentences are correct or not. The first criterion is counting and matching the number of sentences in the source and target languages. The second criterion is mapping the sentence end marker in source and destination languages. For example, after sentence tokenization is applied the number of new sentences and sentence end marker of the source is compared with that of the target. If both criteria are fulfilled, the new sentence list is used as a parallel corpus.

For the English–Ge'ez[6] parallel corpus, the source language contain multiple verses in a single line while on the target side, each line contains a single verse. To align the two corresponding language pairs, we tried to merge verses to produce the desired parallel verse. In addition, removing unnecessary links, numbers, symbols and foreign texts in each language has been done.

4.2 Corpus Size and Distribution of tokens and vocabulary

The corpus has been analyzed to see the relationship between English and each one of the considered Ethiopian languages. As it can be seen from Table 1 to Table 5 and the corresponding Figures (Figure 1 to 5), there is a significant difference between the morphology of English and the Ethiopian languages. Due to this difference, the same number of sentences in these language pairs is tokenized into significantly different number of tokens and vocabularies in all the available domains.

The Figures clearly show that English vocabulary is much lower than vocabulary of all the considered Ethiopian languages. On the contrary, the English token is significantly higher than tokens of the Ethiopian languages. It is clear, therefore, that such differences between the languages in a language pair makes SMT difficult because it aggravates data scarcity and results into a weakly trained translation model. The morphological complexity of the Ethiopian languages also challenges the SMT towards them because it results into a poorly trained language model.

	History	Legal	Religion	
			Bible	Blog
English *	35,325	85,526	767,989	80,505
Amharic *	29,804	63,940	472,294	62,436
English **	8,420	9,029	39,113	9,838
Amharic **	10,560	12,779	93,001	16,383

Table 2: Distribution of Tigrigna and English text.

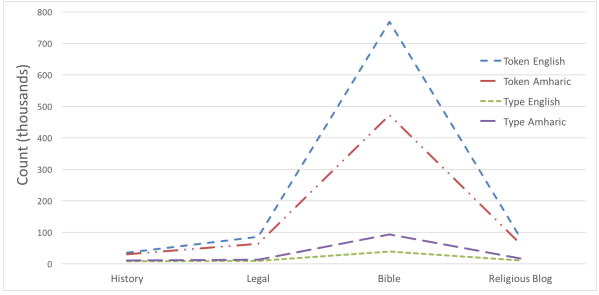


Figure 1: Comparison of Amharic-English SMT data

		Legal	Religion	
			Bible	Blog
Token	English	11,597	495,780	53,999
	Tigrigna	15,481	767,989	66,408
Type	English	2,989	81,674	13,494
	Tigrigna	2,286	39,113	8,818

Table 3: Distribution of Tigrigna and English text.

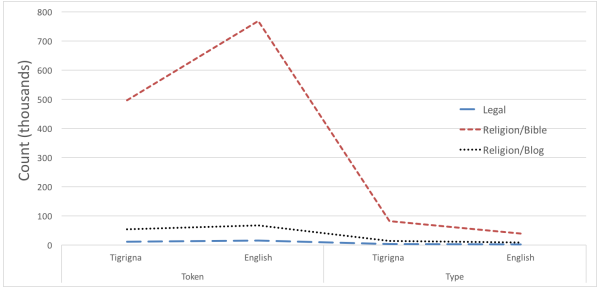


Figure 2: Comparison of Tigrigna-English SMT data

[6]English-Ge'ez parallel corpus available at https://www.ethiopicbible.com/amharic-bible-books

		Legal	Religion	
			Bible	Blog
Token	English	42,390	157,346	68,299
	Afan-Oromo	49,701	187,926	72,588
Type	English	7,819	19,844	10,110
	Afan-Oromo	7,819	13,659	9,320

Table 4: Distribution of Afan-Oromo and English text.

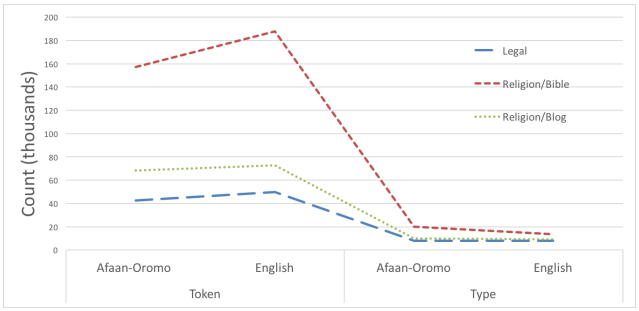


Figure 3: Comparison of Afan Oromo-English SMT data

		Religion	
		Bible	Blog
Token	English	468,122	41,041
	Wolaytta	700,321	59,754
Type	English	59,320	10,012
	Wolaytta	26,610	8,402

Table 5: Distribution of Wolaytta and English text.

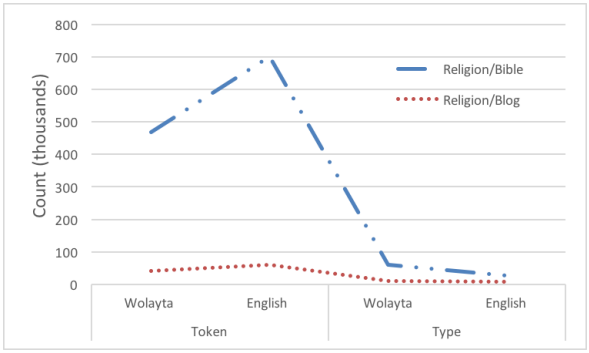


Figure 4: Comparison of Wolaytta-English SMT data

		Bible
Token	English	160,662
	Ge'ez	303,546
Type	English	33,894
	Ge'ez	15,260

Table 6: Distribution of Ge'ez and English text.

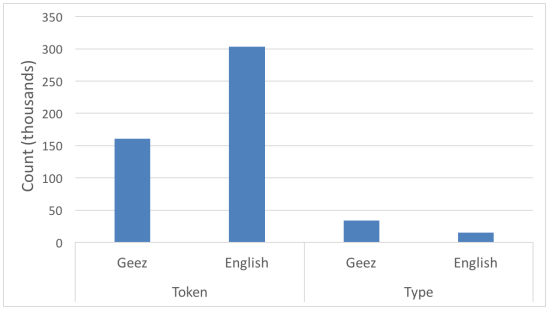


Figure 5: Comparison of Ge'ez-English SMT data

5 SMT Experiments and results

In this study, bi-directional SMT systems are developed to check the validity of the collected parallel corpora for English and the four Ethiopian languages.

5.1 Experimental setups

In the experimental setup, Moses is used along with GIZA++ alignment tool (Och and Ney, 2003) for aligning words and phrases. SRILM toolkit was used to develop language models using semi-automatically prepared corpora from the training and tuning corpora of target languages.

Table 7 shows the sentence length, the number of words, the total number of tokens and the average sentence length found in the corpora for the four Ethiopian languages with respect to English.

To carry out the experiments, each parallel corpus is divided into three partitions; 80% as a training set where a large subset of the whole corpus is used to train the language and translation models, 10% for tuning (useful to adjust the weights of the model combination) and 10% as a tes tset for evaluating the final bi-direction statistical machine translation system of each language pair.

Automatic metrics and subjective evaluation are the two most widely used techniques or methods for MT system evaluation. In this research, BiLingual Evaluation Under Study (BLEU)

		Sentence	Token	Type	Average word
Language Pairs	English	40,726	66,400	969,345	23
	Amharic		132,723	628,474	15
	English	35,378	50,217	849,878	19
	Tigrigna		98,157	561,376	14
	English	14,706	29,076	264,790	20
	Afan-Oromo		37,773	268,035	17
	English	30,232	35,012	760,075	21
	Wolaytta		69,332	509,163	14
	English	11,663	15,260	303,546	26
	Ge'ez		33,894	160,662	13

Table 7: Sentence and word distribution of Ethiopian languages and English text.

is used for automatic scoring. Table 8 presents the experimantal results of bi-directional English-Ethiopian languages SMT.

Language pair	BLEU	Language pair	BLEU
English-Amharic	13.31	Amharic-English	22.68
English-Tigrigna	17.89	Tigrigna-English	27.53
English-Afan Oromo	14.68	Afan Oromo-English	18.88
English-Wolaytta	10.49	Wolaytta-English	17.39
English-Ge'ez	6.76	Ge'ez-English	18.01

Table 8: Experimental results of bi-directional English-Ethiopian languages SMT

As shown in Table 8, the English-Amharic translation shows a BLEU score of 13.31 while the Amharic-English has a 22.68. Similarly, the English-Tigrigna and Tigrigna-English have BLEU scores of 17.89 and 27.53, respectively. Likewise, English-Afaan Oromo has a 14.68 BLEU and Afan Oromo-English has 18.88 BLEU score. In a similar way, the English-Wolaytta translation has BLEU of 10.49 while Wolaytta-English has 17.39. Finally, The English-Ge'ez and Ge'ez-English translation has BLEU score of 6.67 and 18.01, respectively.

Figure 6 presents summary of BLEU score registered for bi-directional English-Ethiopian languages using statistical approach.

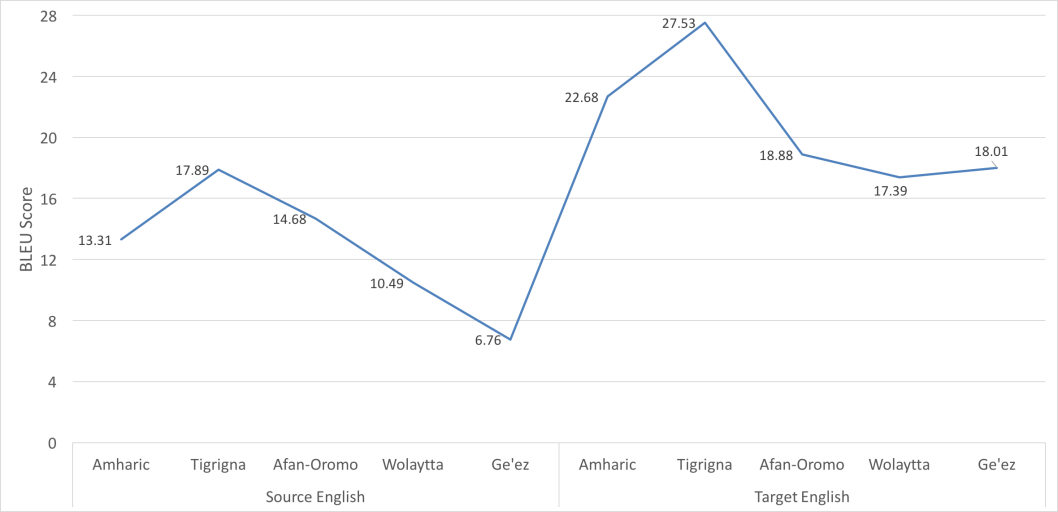


Figure 6: Comparison of Bi-directional English to Ethiopian language translation results

The BLEU score of Amharic-English translation system is lower than the Tigrigna-English translation system although the size of the Amharic-English parallel corpus is bigger than the Tigrigna-English one. This might be due to the number of domains considered in the corpora. The Amharic-English corpus covers all the three domains whereas the Tigrigna-English corpus is from only two domains.

Despite the size of the data, the English-Ethiopian languages SMT systems have less BLEU scores than that of Ethiopian languages-English ones. This is because of the fact that when the Ethiopian languages are used as a target language, the translation from English as a source language is challenged by many-to-one alignment. On the other hand, better performance is registered when the target language is English since the alignment is one-to-many taking each Ethiopian language as a source. In addition to this, the language model data favours the English language than that of Ethiopian languages due to the complexity of the morphology of these languages.

6 Conclusion and future work

This paper presents the attempt made in preparing standard parallel corpora for English and Ethiopian languages. The text data have been collected from the web in history, legal and religious domains. Then, the data are further pre-processed and normalized in preparing a bilingual parallel corpora for SMT task. Using the corpora, bi-directional statistical machine translation experiments have been conducted. The experimental results show that a translation from Ethiopian languages to English resulted in better BLEU score than that of the English to Ethiopian languages. The morphological richness of the Ethiopian languages greatly affect the performance of SMT specially when they are target languages.

To further see the impact, there is a need to conduct additional experiments with the objective of identifying an optimal one-to-many and many-to-one alignment when either of them used as a target language. Moreover, further research is needed to identify the exact reason behind the low performance of English to Ethiopian languages translation systems. Investigating the effect of domains on SMT performance is one of the future work we will work on.

References

Saba Amsalu and Sisay Fissaha Adafre. 2006. *Machine Translation for Amharic: Where we are.*, In proceedings of LREC 2006, pp. 47-50.

Philipp Koehn. 2009. *Statistical machine translation.*, volume 1. Cambridge University Press.

W.John Hutchins 1995. *Concise history of the language sciences: from the Sumerians to the cognitivists.*, volume 1. Edited by E.F.K.Koerner and R.E.Asher. Oxford: Pergamon Press, pp. 431-445

Tariku Tsegaye 2004. *English-Tigrigna Factored Statistical Machine Translation.*, MSc. Thesis, School of Information Science, Addis Ababa University, Addis Ababa, Ethiopia.

Sisay Adugna Chala 2009. *English-Afaan Oromo Machine Translation: An Experiment Using Statistical Approach.,*. MSc. Thesis, School of Information Science, Addis Ababa University, Addis Ababa, Ethiopia.

Eleni Teshome 2013. *Bidirectional English-Amharic Machine Translation: An Experiment Using Constrained Corpus.,*. MSc. Thesis, Department of Computer Science, Addis Ababa University, Addis Ababa, Ethiopia.

Jabesa Daba 2013. *Bi-directional English-Afaan Oromo Machine Translation Using Hybrid Approach,*. MSc. Thesis, Department of Computer Science, Addis Ababa University, Addis Ababa, Ethiopia.

Akubazgi Gebremariam 2013. *Amharic-Tigrigna Machine Translation Using Hybrid Approach,*. MSc. Thesis, Department of Computer Science, Addis Ababa University, Addis Ababa, Ethiopia.

Mulu Gebreegziabher Teshome 2017. *English-Amharic Statistical Machine Translation.*,. PhD Dissertation, IT Doctoral Program, Addis Ababa University, Addis Ababa, Ethiopia.

Sisay Fissaha Adafre. 2004. Adding Amharic to a Unification based Machine Translation System: An Experiment, ISBN: 9780820473314, Peter Lang GmbH.

Sisay Fissaha Adafre. 2004. *Adding Amharic to a Unification based Machine Translation System: An Experiment*, ISBN: 9780820473314, Peter Lang GmbH.

Michael Melese Woldeyohannis and Million Meshesha. 2017. *Experimenting Statistical Machine Translation for Ethiopic Semitic Languages : The case of Amharic-Tigrigna.*, International Conference on ICT for Development for Africa (ICT4DA) September 25–27, 2017 Bahir Dar, Ethiopia.

Gary F. Simons and Charles D. Fennig. . 2017. *Ethnologue: Languages of the World.* 20th Edition, SIL, Dallas, Texas.

John Hutchins. 2005. *The history of machine translation in a nutshell..* Retrieved March, 2018, pages 1–5, 2005. URL http://www.hutchinsweb.me.uk/Nutshell-2005.pdf

Leslau, W. 2000. Alternation. *Introductory Grammar of Amharic.* Otto Harrassowitz, Wiesbaden.

Teferra, A. and Hudson, G. 2007. *Essentials of Amharic.* Rudiger Koppe Verlag.

Wakasa, M. 2008. *A Descriptive Study of the Modern Wolaytta Language.* University of Tokyo.

Mason, J. S. 1996. *Tigrigna grammar.* Tipografia U. Detti.

Yohannes, T. 2002. *A Modern Grammar of Tigrigna.* Tipografia U. Detti.

Griefenow-Mewis, C. 01. *A grammatical sketch of written Oromo.*, volume 16. Rüdiger Köppe.

Gasser, M. 2010. *A Dependency Grammar for Amharic.*, In Workshop on Language Resources and Human Language Technologies for Semitic Languages.

Gasser, M. 2011. *HornMorpho: a system for morphological processing of Amharic, Oromo, and Tigrigna.*, In Conference on Human Language Technology for Development. Alexandria, Egypt.

Dillmann, A. 1907. *Ethiopiac Grammer*, 24(11):503–512. Improved and enlarged by Karl Bezold, Translated by J.A. Crichton. London: William and Norgate.

Och, F.J. and Ney, H. 2003. *A systematic comparison of various statistical alignment models.*, 29.1 (2003): 19-51. Computational linguistics.

Multilingual Neural Machine Translation with Task-Specific Attention

Graeme Blackwood **Miguel Ballesteros** **Todd Ward**
IBM Research AI
1101 Kitchawan Rd, Yorktown Heights, NY 10598, USA
{blackwood,toddward}@us.ibm.com, miguel.ballesteros@ibm.com

Abstract

Multilingual machine translation addresses the task of translating between multiple source and target languages. We propose task-specific attention models, a simple but effective technique for improving the quality of sequence-to-sequence neural multilingual translation. Our approach seeks to retain as much of the parameter sharing generalization of NMT models as possible, while still allowing for language-specific specialization of the attention model to a particular language-pair or task. Our experiments on four languages of the Europarl corpus show that using a target-specific model of attention provides consistent gains in translation quality for all possible translation directions, compared to a model in which all parameters are shared. We observe improved translation quality even in the (extreme) low-resource zero-shot translation directions for which the model never saw explicitly paired parallel data.

1 Introduction and Motivation

Multilingual machine translation is the task of building a system capable of translating between more than just a single source and target language. The approach is motivated by the idea that learning to translate between one pair of languages can help to improve the translation quality of related language pairs. Multilingual MT can be divided into three main types according to the support for source and target languages: (i) single source, multiple target (ii) multiple source, single target, and (iii) multiple source, multiple target. Our work focuses on improving the quality of fully n-way multilingual NMT models that can translate between multiple source and target languages.

Multinational organizations and companies increasingly publish content in a variety of languages. In addition to exploiting multiple sources of parallel data and leveraging similarities across different languages, multilingual translation can considerably simplify deployment. Directly supporting all possible translation directions for a set of n languages would require $n(n-1)$ sets of parallel data and trained models. This is an expensive proposal, from both a data and deployment perspective. The number of required systems can be reduced to n if translation by bridging is used. In contrast, a fully n-way multilingual system can be trained to translate between all known language pairs using a single model. It is even possible to translate between language pairs that were never explicitly paired in the parallel training data, so called zero-shot translation (Johnson et al., 2017).

Neural machine translation (NMT) (Bahdanau et al., 2014; Sutskever et al., 2014) has recently become the standard model of translation due to the high levels of fluency and accuracy that it can achieve (Koehn and Knowles, 2017). NMT is an excellent choice for multilingual translation since the neural architecture is language-agnostic and capable of capturing translation properties, such as long-distance re-ordering, between even highly dissimilar languages. One of the main advantages of NMT is that the model is able to learn useful linguistic representations for many languages, and to share parameters and leverage similarities in the abstractions learned by the embeddings and hidden layers of the model. Exploiting multilingual data and representations is of particular interest in improving the quality of translation for low-resource (or even zero-resource) language pairs.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3112–3122
Santa Fe, New Mexico, USA, August 20-26, 2018.

In this paper, we describe a simple but effective extension to the attentional model of neural machine translation that improves the quality of multilingual NMT. We seek to retain as much of the parameter sharing of NMT models as possible, while allowing for language-specific specialization of the attention model to a particular task. Our experiments on the Europarl corpus show that multilingual translation quality is improved for all tested language pairs, with the largest improvements in the (extreme low-resource) zero-shot directions.

The architecture of our sequence-to-sequence neural machine translation system is described in Section 2. An overview of related work in multilingual machine translation and multi-task learning is provided in Section 3. In Section 4, we introduce three task-specific attention model variants that can be used to improve the quality of multilingual NMT. Our experimental framework, implementation and results are described in Section 5, followed by an analysis and example translations with attention plots in Section 6. We conclude with a discussion of possible future work in Section 7.

2 Sequence to Sequence Translation with Attention

Our NMT system is based on the sequence-to-sequence model of translation described by Bahdanau et al. (2014), consisting of a recurrent neural network encoder and decoder with an attention mechanism.

The encoder uses bi-directional gated recurrent units (GRU) (Cho et al., 2014) to encode a source sequence $\mathbf{x} = (x_1, \ldots, x_l)$, where x_i is the embedding vector for the i-th word of the source sentence and l its length. The encoded form of the complete sentence is defined by the sequence of hidden states $\mathbf{h} = (h_1, \ldots, h_l)$, where each h_i is computed as

$$h_i = \begin{bmatrix} \overleftarrow{h}_i \\ \overrightarrow{h}_i \end{bmatrix} = \begin{bmatrix} \overleftarrow{f}(x_i, \overleftarrow{h}_{i+1}) \\ \overrightarrow{f}(x_i, \overrightarrow{h}_{i-1}) \end{bmatrix}, \tag{1}$$

and $\overleftarrow{f}$ and $\overrightarrow{f}$ denote the operations of the right-to-left and left-to-right GRU cells, respectively.

Given the encoded representation of the source sentence $\mathbf{h}$, the decoder produces the target translation $\mathbf{y}$ by computing the sequence $\mathbf{y} = (y_1, \ldots, y_m)$, where m is the target sentence length. At each time-step t, the probability of target word y_t is computed as

$$p(y_t|\mathbf{h}, y_{t-1}, \ldots, y_1) = g(s_t, y_{t-1}, H_t), \tag{2}$$

where g is a feed-forward network over the current decoder hidden state s_t, the word embedding of the previously predicted target word y_{t-1}, and a context vector H_t, followed by a softmax to predict the probability distribution over the output vocabulary. The decoder states are updated according to

$$s_t = q(s_{t-1}, y_{t-1}, H_t), \tag{3}$$

where q implements the conditional GRU with attention of Sennrich et al. (2017) and H_t is an attention-weighted representation of the encoded source sequence $\mathbf{h}$. The attention-weighted source representation at time-step t is defined as the weighted sum

$$H_t = \begin{bmatrix} \sum_{i=1}^{l}(\alpha_{t,i} \cdot \overleftarrow{h}_i) \\ \sum_{i=1}^{l}(\alpha_{t,i} \cdot \overrightarrow{h}_i) \end{bmatrix}, \tag{4}$$

where the normalized weights $\alpha_{t,i}$ indicate the relative importance of the vector representation of each source position h_i when producing the next target word at time-step t. The weights are computed using a two-layer feed-forward network r:

$$\alpha_{t,i} = \frac{\exp\{r(s_{t-1}, h_i, y_{t-1})\}}{\sum_k \exp\{r(s_{t-1}, h_k, y_{t-1})\}}. \tag{5}$$

3 Related Work

Early phrase-based approaches to multilingual MT focused on multi-source translation. Och and Ney (2002) used a simple product or max rule to select at the sentence-level the single best hypothesis with the highest translation score from multiple decoders. There is no sharing of parameters. Consensus network decoding (Matusov et al., 2006) can also be used to combine the word-level output of translations from multiple source languages. Such system combination techniques allow sharing of the words in the candidate translations, but still require training individual models for each language pair of interest.

Training multilingual MT systems capable of translating between multiple languages can be considered an instance of multi-task learning (Caruana, 1997), which is the idea of solving synergistic tasks while maximizing the number of shared parameters. Sharing parameters may be useful when attempting to solve different tasks, since we can minimize representation bias by learning a more regularized representation (Baxter, 2000). The flexibility nature of neural architectures allows for selection of the components of the model that are to be shared and those which are not.

Sequence to sequence models with attention are no exception. Each set of parameters provides different levels of generalization (Reimers and Gurevych, 2017), which is evidenced in the synergistic task of training multilingual translation models. For example, Dong et al. (2015) jointly train decoders while the rest of the parameters are task-specific; Zoph and Knight (2016) jointly train the encoders while the rest of the parameters are task-specific, and Johnson et al. (2017) train both encoders and decoders jointly with language-specific tokens to guide learning as in (Ammar et al., 2016). These latter approaches are the ones that we build on. We augment our decoder with a task-specific attention mechanism intended to better capture word order and language-specific nuances while continuing to share the rest of the model parameters (including token embeddings).

4 Task-Specific Attention Models

Fully n-way multilingual NMT systems need to support multiple source and target languages in the encoder and decoder GRUs. Our work focuses on improving the use of attention in the decoder. At each time-step t, the decoder computes the attention model weights $\alpha_{t,i}$ of Equation (5) in order to quantify the relative importance of the encoder states corresponding to each source position. To the extent that attention can be considered an analogue of word alignment, we would expect to see different patterns of attention for language-pairs with different word order and word-level alignments. Multilingual models restricted to a single shared attention will struggle when decoding multiple languages with different word orders, since the parts of the source sentence that should be attended to depend on the source and target languages of the translation task.

Our task-specific attention models can be used to address this issue. We train and empirically evaluate three task-specific attention model variants: *target-specific* attention, *source-specific* attention, and *paired* attention which is associated with a particular language pair (i.e. translation direction). Each of our models introduces conditioning on the weights and biases used to compute the attention coefficients. They differ only in the choice of the key used for conditioning attention:

- *Target-specific*: separate attention weights and bias for each target language
- *Source-specific*: separate attention weights and bias for each source language
- *Paired*: separate attention weights and bias for each source + target pair

Our multilingual NMT system follows Johnson et al. (2017) in using special tokens to indicate the desired target language. These tokens can be considered to define the 'task' from a multi-task learning perspective as shown in (Ammar et al., 2016; Kann et al., 2018). Since the encoder states are composed from bi-directional GRUs, we augment the source side of our parallel training data with both prefix and suffix task tokens. This ensures that the task token is not attenuated in the left-to-right encoder GRU. Such tokens are sufficient to ensure that the multilingual decoder produces words in the correct target language[1]. A French source sentence that is to be translated into English would be augmented as follows:

[1]We tried explicit softmax decoding constraints on the target vocabulary, but found them to be unnecessary.

target-specific	<ToEn>, <ToFr>, <ToEs>, <ToDe>
source-specific	<FromEn>, <FromFr>, <FromEs>, <FromDe>
paired	<FrEn>, <EnFr>, <EsEn>, <EnEs>, <DeEn>, <EnDe>

Table 1: Valid source-side task tokens for task-specific attention model training and decoding.

target-specific	<ToFr> w_1 w_2 ... w_l <ToFr>
source-specific	<FromFr> <ToEn> w_1 w_2 ... w_l <ToEn>
paired	<FrEn> w_1 w_2 ... w_l <FrEn>

Table 2: Source side training data augmented with task-specific attention model tokens.

<ToEn> Guide des industries canadiennes : <ToEn>

During training and decoding, we dynamically construct the computation graph using the parameters for each task. Table 1 defines the valid task tokens for target-specific, source-specific and paired attention models, given the four languages supported by our multilingual model. See Section 5 for full details of our experimental framework and supported language pairs.

The augmented source sides of the parallel training data for our three model variants take the abstract forms shown in Table 2. The target-specific and paired attention models simply add the desired target language or language pair prefix and suffix tokens. For the source-specific attention model, we want to continue to allow for run-time selection of the desired target language so we introduce a second prefix token to indicate the selection of the specific attention parameters. The first token for the source-specific models is used only to select the parameters and stripped from the source sentence before using it in training or decoding. This allows us to dynamically select the source-specific attention parameters associated with translation from French, while (i) still allowing run-time selection of the target language, (ii) supporting zero-shot directions, and (iii) ensuring that the multilingual model sees exactly the same sequence of tokens as the target-specific model.

Note that only the target-specific and source-specific model variants enable zero-shot translation. The paired form of attention model trains separate attention weights and biases for each of the translation directions observed in the training data. It would be possible to use a separate prefix token for the attention key (in a similar manner to that of source-specific attention), but there is no explicit set of attention parameters that should be used for the zero-shot directions. The prefix token could specify that the attention parameters associated with either the source or target language of the zero-shot direction should be used.

All three of our task-specific attention model variants still share most of their encoder and decoder parameters. The task-specific attention models require only a very small increase in the total number of parameters. For hidden state dimensionality d, we add one additional set of attention weights ($d \times d$ parameters) and bias (d parameters) for each supported task. For the neural network topology used in our experiments (see Section 5, below), a target-specific attention model with support for four distinct target languages (i.e. tasks) requires only a 1.2% increase in the total number of model parameters, compared to the shared-attention version of the model.

5 Experiments

We evaluate the quality of our multilingual translation models using training data from the Europarl Corpus[2], Release V7. Our experiments focus on three primary language pairs: French-English, Spanish-English and German-English. We include German with SOV-type word order to contrast the SVO-type word order of the other languages. The source and target sides of the parallel training data are processed with the standard tokenizer included in the Moses SMT Toolkit (Koehn et al., 2007). For

[2] http://www.statmt.org/europarl/

	Sentence Pairs	Source Tokens	Target Tokens
French-English	2.01m	62.6m	56.3m
Spanish-English	1.97m	57.1m	55.0m
German-English	1.92m	51.0m	53.6m
Merged	11.79m	335.6m	335.6m

Table 3: Tokenized corpus statistics for training single-data baselines and n-way multilingual models.

	Fr-En	Es-En	De-En	Merged
source (BPE)	35.3k	37.2k	43.1k	80.0k
target (BPE)	35.3k	35.2k	35.1k	80.0k

Table 4: Source and target vocabulary sizes for single-data baselines and n-way multilingual models.

training multilingual systems, we merge the parallel data for all available directions. The parallel data for each language pair is thus included twice, once in each direction.

In order to support experiments on many-to-many multilingual translation using a single model, we apply a jointly-learned set of 80k Byte-Pair Encoding (BPE) rules (Sennrich et al., 2015) obtained from the merged source and target sides of the training data for all three language pairs. This ensures that all experiments, including the single-language baselines, share exactly the same tokenization and sub-word vocabularies. Tokenized corpus statistics for the parallel training data are summarized in Table 3, while Table 4 compares the vocabulary sizes obtained from BPE processed data for the single-data baseline and fully n-way multilingual models.

The Europarl evaluation data set dev2006 is used as our validation set, while devtest2006 and test2007 are our blind test sets. We also evaluate the out-of-domain News Commentary test sets nc-dev2007 and nc-devtest2007 in order to demonstrate the robustness of our approach to different kinds of data. Case-sensitive single-reference BLEU scores (Papineni et al., 2002) are computed using the `multi-bleu.perl` script included with Moses. Reimers and Gurevych (2017) have shown that reporting a single metric score can sometimes be misleading for many neural architectures. For this reason, all BLEU scores reported in the tables are obtained by averaging the decoding results from five separate models initialized with distinct random seeds.

5.1 Implementation Details

Our sequence-to-sequence NMT model with task-specific attention is implemented in C++ using DyNet[3] (Neubig et al., 2017a). We use DyNet since the computation graph can be efficiently modified on a batch-by-batch basis during training and decoding, allowing for the runtime selection of attention weights and bias parameters according to the desired task.

We utilize the auto-batching feature (Neubig et al., 2017b) of DyNet for efficient matrix computations, but the parallel training data must still be separated into batches[4] such that each batch consists of sentences for a single task, e.g. `<ToEn>` for a batch of sentences that all use the same target-specific attention model for translating into English. The entire training data and batches are shuffled at the beginning of each epoch so that the order of tasks seen during training is random and that they occur in proportion to their distribution in the training data.

We use 256 dimensions for our source and target word embeddings, and 256 dimensions for the hidden states. A single recurrent layer is used for the encoder and decoder. The model parameters are optimized using an unbiased Adam[5] stochastic optimizer (Kingma and Ba, 2014) in order to minimize perplexity on a held-out validation set. The validation set for the single-data baseline systems is simply the dev2006 portion of the official Europarl evaluation data for that language pair, e.g. dev2006.fr and dev2006.en for

[3] `http://dynet.io/`

[4] Each batch contains a combined maximum of 5000 source and target tokens.

[5] We use a learning rate of 0.001.

	Fr-En			Es-En			De-En		
single-data	31.84	32.21	31.80	32.17	32.09	32.11	28.39	28.67	28.32
shared	29.99	30.34	30.12	30.99	30.87	30.78	26.30	26.52	26.06
target-specific	**30.50**	**31.00**	**30.68**	**31.58**	**31.62**	**31.63**	**26.96**	27.12	**26.85**
source-specific	30.47	30.87	30.36	31.47	31.54	31.51	26.75	**27.13**	26.70
paired	29.99	30.62	30.23	30.93	31.23	31.17	26.43	26.81	26.38

Table 5: BLEU scores for single-data baseline and task-specific attention model variants: xx-to-En

	En-Fr			En-Es			En-De		
single-data	32.24	32.44	32.87	32.60	32.60	33.18	22.15	22.55	22.36
shared	29.44	29.85	30.23	30.03	30.50	31.02	19.40	19.84	19.63
target-specific	**30.06**	**30.38**	**30.92**	**30.62**	**30.93**	**31.56**	**20.16**	**20.51**	**20.27**
source-specific	29.68	30.20	30.50	30.54	30.70	31.48	19.66	20.10	20.01
paired	29.19	29.68	30.07	30.17	30.40	31.07	19.62	20.13	19.97

Table 6: BLEU scores for single-data baseline and task-specific attention model variants: En-to-xx

French-to-English translation. For multilingual systems, we want the model to work well in all possible directions so we merge the dev2006 data for all directions of interest. Combining all six directions gives a validation set with a total of $6 \times 2000 = 12000$ sentences.

5.2 Task-Specific Attention Model Decoding Results

Tables 5 and 6 show decoding results for the Foreign-to-English (xx-to-En) and English-to-Foreign (En-to-xx) directions, respectively. In all six translation directions, the single-data baseline systems obtain the highest overall BLEU scores. We expect these baseline models to perform well on their respective test sets since the translation direction (i.e. task) of the test set exactly matches the parallel data and validation set used to train the models.

The fully n-way multilingual system with a shared attention model shows degradations of up to -2.0 BLEU score, exhibiting a similar pattern to previously reported multilingual NMT results (Johnson et al., 2017). When the encoder and decoder both support multiple languages, a single shared attention model is harmful. Our multilingual NMT model with target-specific attention mitigates much of this degradation. We observe gains of between +0.5 and +0.9 BLEU, with respect to the single attention model shared across all language pairs.

Source-specific attention is also better than the shared attention model, but not as good as target-specific attention. Paired attention (i.e. a separate set of attention weights and biases for each of the six translation directions with explicitly paired parallel data) shows little change compared to the standard shared attention model. The paired attention model has more tasks than the other models (6 for paired attention vs. 4 for both target-specific and source-specific attention). It has therefore seen less data for each task and benefits from no sharing of attention-related parameters. The paired model probably lacks sufficient training data to learn a good separate attention for all tasks.

Table 7 shows BLEU scores on the out-of-domain News Commentary nc-dev2007 and nc-devtest2007 test sets. Our task-specific attention model again provides gains of between +0.6 to +1.2 BLEU showing that the technique is robust even for test sets less closely matched to the parallel training data.

5.3 Zero-Shot Decoding Results

Table 8 shows decoding results for the six zero-shot translation pairs, i.e. those directions (such as French-to-Spanish) never explicitly paired in the parallel data. Although the absolute numbers are lower (as expected for zero-shot pairs), our target-specific attention model gives gains of between +1.0 and +1.5 BLEU over the model that shares attention parameters across all languages. Source-specific attention does not work well for zero-shot translation.

	Fr-En		Es-En		De-En	
shared	23.20	21.56	30.00	28.31	20.60	18.40
target-specific	**23.82**	**22.45**	**30.95**	**29.47**	**21.65**	**19.40**
	En-Fr		En-Es		En-De	
shared	26.05	24.95	31.74	30.36	16.20	14.44
target-specific	**26.96**	**25.51**	**32.64**	**30.91**	**16.99**	**15.12**

Table 7: BLEU scores for single-data baseline and task-specific attention model variants on out-of-domain News Commentary testsets nc-dev2007 and nc-devtest2007.

	Fr-Es			Fr-De			
shared	13.64	13.63	13.75	7.82	8.04	7.84	
target-specific	**15.10**	**15.29**	**15.31**	**8.76**	**8.95**	**8.82**	From Fr
source-specific	12.27	12.35	12.31	6.99	7.01	6.82	
	Es-Fr			Es-De			
shared	13.43	13.33	13.48	7.57	7.82	7.57	
target-specific	**14.55**	**14.40**	**14.40**	**8.70**	**8.63**	**8.65**	From Es
source-specific	12.53	12.35	12.26	7.07	6.88	6.86	
	De-Fr			De-Es			
shared	10.50	10.12	10.31	9.86	10.06	10.02	
target-specific	**11.53**	**11.24**	**11.38**	**11.28**	**11.31**	**11.29**	From De
source-specific	9.64	9.59	9.41	8.76	8.83	8.78	

Table 8: BLEU scores for multilingual NMT 'zero-shot' translations comparing the baseline shared attention model to target-specific and source-specific attention models.

No single-data baseline exists for zero-shot translation since we have no parallel data for those pairs. However, we can pivot by data (i.e. find new parallel sentence pairs $x : z$ from existing data $x : y$ and $y : z$, with common y) or translate by bridging (translate from x to y, and then from y to z) for the zero-shot pairs. Pivoting by data is possible since there is a high degree of overlap amongst the various languages of the Europarl Corpus.

Although our multilingual NMT model has not seen explicitly paired data in the zero-shot directions, the encoder and decoder have seen many of the source or target sentences paired with other languages. Our zero-shot experiments are designed to test the hypothesis that the target-specific attention model is better than a model which shares attention parameters for all languages. Any benefit due to source or target data similarity applies equally to each of these multilingual models.

6 Analysis and Examples

Learning curves for the shared attention and target-specific attention model variants are shown in Figure 1. The plot includes five curves for each model variant, corresponding to different random initializations. Our target-specific attention model achieves much higher BLEU scores on the validation set in earlier epochs. There is also considerably less variance. The first few times the validation set is decoded with the shared attention model, the gap with respect to the target-specific attention model can be as large as 10 points BLEU. Our validation set contains sentences from all six translation directions so it is not surprising that a model with a single set of shared attention weights and biases performs poorly when applied to the 'wrong' task. Even after a full epoch of training (11.8m sentence pairs), the shared attention model continues to lag behind the target-specific attention model by almost one BLEU point.

The following example shows German-to-English translations obtained using a multilingual NMT system with a shared or target-specific attention model. This is an interesting example since the model must attend to the distant German source words "angesprochen worden ist" when producing the English trans-

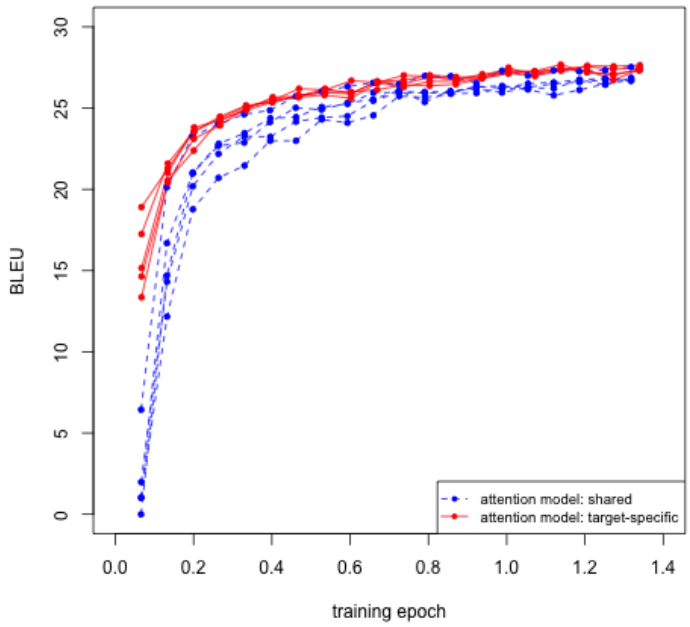


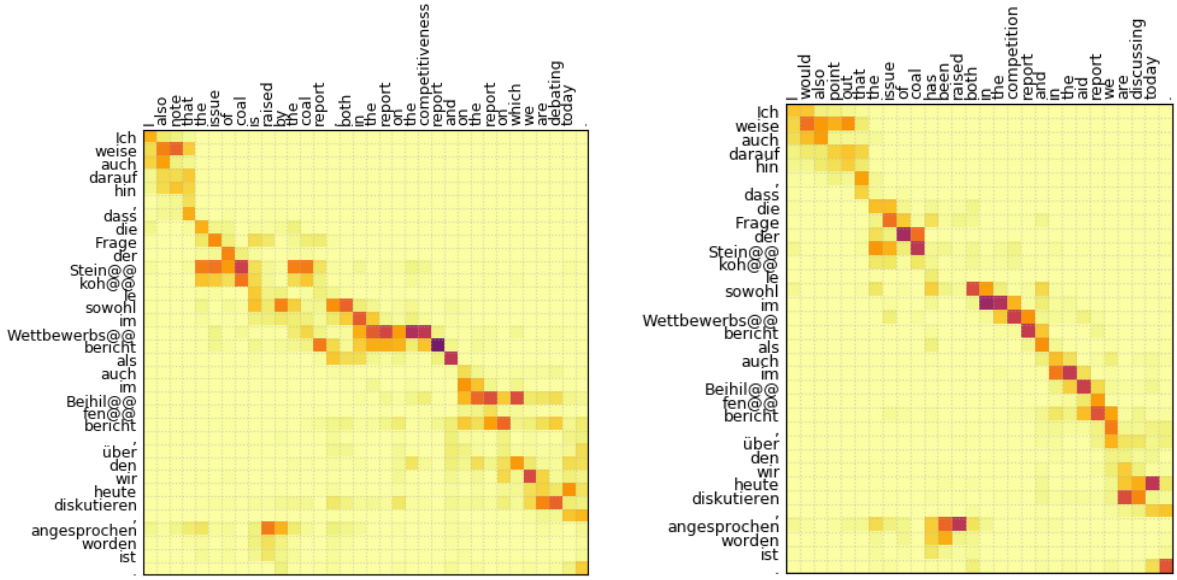
Figure 1: Multilingual validation set BLEU scores obtained during training for five different randomly initialized seeds using a shared or target-specific attention model.

lation. The multilingual NMT system with a shared model of attention produces a deficient translation, containing multiple repetitions of the word 'report' and omitting the important content word 'aid':

source	Ich weise auch darauf hin , dass die Frage der Stein@@ koh@@ le sowohl im Wettbewerbs@@ bericht als auch im Beihil@@ fen@@ bericht , ber den wir heute diskutieren , angesprochen worden ist .
reference	I would also draw your attention to the fact that the coal issue is raised both in the competition report and in the subsidy report that we are discussing today .
shared	I also note that the issue of coal is raised by the coal report , both in the report on the competitiveness report and on the report on which we are debating today .
target-specific	I would also point out that the issue of coal has been raised both in the competition report and in the aid report we are discussing today .

The attention matrices computed during decoding for the shared and target-specific model variants are shown in Figure 2. Rows correspond to the words of the German source sentence and columns to the words of the English target translation. The target-specific attention model results in a sharper and less diffuse attention over the words of the source sentence, especially at the beginning and end of the sentence, and for the translation of the passive German construction "angesprochen worden ist" which requires long-distance re-ordering.

For this example, the shared attention model leads to diffuse alignments since a single set of attention weights and biases must be used to align the words of many target languages with disparate word orders. Our target-specific attention model leads to more accurate alignments since the decoder is conditioned for a single target language (as indicated by the 'task' token prefix). Given the target-specific set of attention weights and biases, and the initialization of the decoder state using the RNN encoded source, the decoder is able to more accurately attend to the correct source words when producing the words of the target sentence at each time-step.



(a) Attention model: shared

(b) Attention model: target-specific

Figure 2: Attention plots obtained while decoding a German-to-English sentence pair in the test set. Rows correspond to source words and columns to each target position. Target-specific attention results in less diffuse alignments.

7 Conclusions and Future Work

We have described a simple but effective technique for improving the quality of multilingual NMT. Our technique mitigates much of the loss that occurs when multiple source and target languages are supported in a system with a single encoder and decoder. It is particularly effective in the zero-shot (i.e. extreme low-resource) translation directions.

Our approach extends the use of target language prefix tokens described by Johnson et al. (2017) to select a task-specific set of attention weights and biases for each task of interest. For NMT, where attention can be considered an analogue of word alignment, the use of separate attention weights and biases for language pairs with different word orders leads to improved BLEU scores in our multilingual decoder. Our results show that multilingual NMT works best with a target-specific attention model, i.e. a distinct set of attention weights and bias parameters for each supported target language. Our improved model handles all possible translation directions with only a small increase in the total number of parameters, compared to the single-data baseline or standard shared attention model systems.

In future work, we plan to apply our multilingual techniques to true low-resource language pairs, with the goal of augmenting low-resource poor quality translation systems with knowledge obtained on languages with richer resources. Given our results for zero-shot translation, we expect our approach to work well. Our attention model variants can also be applied to multi-task learning frameworks, e.g. to improve the quality of translation using knowledge learned from a variety of other natural language processing tasks such as POS tagging and dependency parsing (Kiperwasser and Ballesteros, 2018). Recent work on learning and unsupervised induction of multilingual word representations (Upadhyay et al., 2016; Zhang et al., 2017) could also be used to improve multilingual translation since the models share a single vocabulary for all source and target languages of interest.

References

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

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

Jonathan Baxter. 2000. A model of inductive bias learning. *J. Artif. Intell. Res.*, 12:149–198.

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

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732, Beijing, China, July. Association for Computational Linguistics.

Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernand a Vigas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Katharina Kann, Jesus Manuel Mager Hois, Ivan Vladimir Meza Ruiz, and Hinrich Schütze. 2018. Fortification of neural morphological segmentation models for polysynthetic minimal-resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 47–57. Association for Computational Linguistics.

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

Eliyahu Kiperwasser and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. *TACL*, 6:225–240.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *CoRR*, abs/1706.03872.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*.

E. Matusov, N. Ueffing, and H. Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *Proceedings of the European Association for Computational Linguistics*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017a. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Graham Neubig, Yoav Goldberg, and Chris Dyer. 2017b. On-the-fly operation batching in dynamic computation graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 3974–3984.

Franz Josef Och and Hermann Ney. 2002. Statistical multi-source translation. In *Machine Translation Summit 2001*, pages 253–258, 02.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 338–348.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain, April. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. *CoRR*, abs/1604.00425.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1970, Vancouver, Canada, July. Association for Computational Linguistics.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34, San Diego, California, June. Association for Computational Linguistics.

Combining Information-Weighted Sequence Alignment and Sound Correspondence Models for Improved Cognate Detection

Johannes Dellert
Seminar für Sprachwissenschaft
Universität Tübingen
johannes.dellert@uni-tuebingen.de

Abstract

Methods for automated cognate detection in historical linguistics invariably build on some measure of form similarity which is designed to capture the remaining systematic similarities between cognate word forms after thousands of years of divergence. A wide range of clustering and classification algorithms has been explored for the purpose, whereas possible improvements on the level of pairwise form similarity measures have not been the main focus of research. The approach presented in this paper improves on this core component of cognate detection systems by a novel combination of information weighting, a technique for putting less weight on reoccurring morphological material, with sound correspondence modeling by means of pointwise mutual information. In evaluations on expert cognacy judgments over a subset of the IPA-encoded NorthEuraLex database, the combination of both techniques is shown to lead to considerable improvements in average precision for binary cognate detection, and modest improvements for distance-based cognate clustering.

1 Introduction

The inference of cognate sets, i.e. sets of words which are related by being inherited from a common ancestor, is a central problem of historical linguistics. In the classical comparative method, the detection of cognates is an elementary preprocessing step for establishing sound laws, reconstructing the original forms in proto-languages, and thereby ultimately showing that groups of languages are related by common ancestry. In computational historical linguistics, cognacy-encoded datasets are the most common input format for modern methods of phylogenetic inference as pioneered by Gray and Jordan (2000). While in classical historical linguistics, a sharp distinction is made between cognates connected purely by inheritance, and words which are related due to borrowing, the common usage in the computational field refers to both types of relations as cognacy, with the understanding that the primary distinction is whether words are etymologically related, and the decision whether they are related by inheritance or borrowing will be made at a later refinement stage. Since this paper deals with automated cognate detection, the term will be used in the broader sense.

This paper builds on Dellert and Buch (to appear), where we introduced a segment-wise information content model as a technique for alignment and comparison of IPA strings. Information content is used to focus the string distance judgment on the parts of the word forms that are distinctive with respect to all the other words in the language. The main benefit for cognate detection is that it weakens distorting effects older approaches were faced with when dealing with lexical forms that were not reduced to stems.

In this paper, I take the next step of combining information weighting with the more established idea of inferring specialized segment similarity matrices for each language pair, with the goal of capturing some of the regularity in sound correspondences caused by sound laws, and therefore making cognates at a higher timedepth more similar.

In order to evaluate the resulting form distances independently of the choice of clustering algorithm, I continue to work primarily on the level of pairwise cognacy judgments. This allows performance to

Proceedings of the 27th International Conference on Computational Linguistics, pages 3123–3133
Santa Fe, New Mexico, USA, August 20-26, 2018.

be measured in terms of precision and recall for each threshold that could be used to separate cognates from non-cognates. The average precision across all possible threshold values provides a criterion for assessing the quality of form distances that is independent of the clustering algorithm. To assess whether improvements in average precision do indeed lead to improved cognate sets after inferring clusters from the distances, B-cubed precision and recall scores are compared in a second step.

As in previous work, our recently published NorthEuraLex database (Dellert and Jäger, 2017) is used for the experiments. NorthEuraLex is a data collection effort aiming to cover a large slice of the basic vocabulary (more than 1,000 concepts) across all sufficiently well-documented languages of Northern Eurasia. At the current stage, dozens of languages from the well-studied Uralic and Indo-European language make it a useful source of cross-family lexicostatistical data. The unique advantage of NorthEuraLex among deep-coverage databases is that it provides a full set of (mostly automatically generated) IPA transcriptions which adhere to a single unified set of conventions across more than 100 languages from a range of families. The disadvantage is that expert cognate annotations are currently only available for a small subset. What is available was produced by intersecting NorthEuraLex with the data present in the most recent online version of IELex (Dunn, 2015), a cognacy-annotated database covering about 200 concepts across more than 120 Indo-European languages.

The Java code implementing the entire toolchain described in this paper, from the NorthEuraLex dump file to the different form distances, as well as the evaluation code, is available under a GPL license in a public repository[1] for other researchers to inspect and build upon.

2 Cognate Detection

In its simplest form, cognate detection is the task of deciding for word forms a from a language L_a and b from a language L_b whether a and b can be traced back to a single proto-word from which both forms derive on different paths of regular sound shifts and borrowing.

On the algorithmic level, cognate detection can be approached directly as a binary classification problem. For instance, machine learning algorithms can be applied to training data in the form of pairwise cognate judgments in order to let a system learn to make correct binary cognacy decisions for pairs of words. Early work in this direction is summarized by Rama (2015), who trains a Support Vector Machine (SVM) over a feature representation that encodes shared subsequences, and shows that this feature set outperforms earlier SVM-based attempts. Moving to non-linear classifiers, Rama (2016) trains a convolutional neural network (CNN) on handcrafted representations of ASJP data, a large database by Wichmann et al. (2016) which covers 40 concepts across more than 5,000 languages in an approximate phonetic transcription. The trained network proves to be quite successful at deciding cognacy between pairs of words from families of limited timedepth.

2.1 Form Distances

A variety of string distance measures has been proposed and tested as a basis for cognate detection. Measures proposed can be as simple as the number of shared bigrams, the longest common subsequence, or even just a binary distinction where strings are judged as similar if there is a match of very coarse-grained equivalence classes assigned to the first two consonants, and as dissimilar otherwise (Turchin et al., 2010). Kondrak (2005) systematically evaluates the potential of some of these simple measures, and shows that they achieve acceptable results on the cognate detection task, albeit on small testsets covering only closely related languages.

However, as shown by List et al. (2017b), more elaborate measures are needed to arrive at state-of-the-art performance. The most widely used non-trivial string distance measure is the Levenshtein distance or edit distance (Levenshtein, 1965), which counts the minimal number of elementary editing operations (deletions, insertions, or replacements) needed to transform the one string into the other. The Levenshtein distance on either the orthography or some coarse-grained sound-class model tends to lead to a workable first approximation to phonetic form distance. In this paper, the **normalized edit distance (NED)**, the edit distance divided by the length of the longer string, will be used as a baseline for IPA-encoded data.

[1] `https://github.com/jdellert/iwsa`

Since assessing the usefulness of different sound class schemes would add an additional dimension to our evaluation, we are using NED on full IPA with the expectation that it will perform very badly, as in the absence of a notion of sound similarity, very small differences in pronunciation between e.g. [ɑ] and [ɒ], will be weighted just as much as e.g. the difference between a vowel and a consonant.

The major step improving on edit distance has been to estimate a similarity score for each pair of phonemes, and to count replacement of similar phonemes by only a fraction of a full replacement in distance computation. For instance, when assessing the similarity of English orthographic strings, changing an *o* to a *u* should be much better than changing an *l* to an *n*. This natural extension to the Levenshtein distance leads to the algorithm first presented by Needleman and Wunsch (1970), another dynamic programming approach which maximizes the similarity score between strings by introducing gaps. Variants of the Needleman-Wunsch algorithm are a very popular method for computing string distances in distance-based phylogenetics.

2.2 Clustering

Since the results of binary classification will typically not lead to a consistent result (as the cognacy relation is transitive, whereas binary decisions taken in isolation will typically not be), the output of a system trained for binary classification needs to be combined with an additional clustering stage in order to arrive at a partition into cognate sets. This approach is exemplified by Jäger and Sofroniev (2016), who train an SVM to predict pairwise probabilities of non-cognacy from phonetic distances, overall language distance, and average word length, and combine the result with UPGMA clustering (Sokal and Michener, 1958) to derive the cognate sets.

The **LingPy** system (List et al., 2017a) is an actively maintained and user-friendly suite of tools for computational historical linguistics. One of its core components is the **LexStat** module (List, 2012) which still provides state-of-the-art performance for cognate clustering, and has become the standard benchmark for the more recent cognate detection systems. For instance, Jäger and Sofroniev compare their system to LexStat in terms of the B-Cubed F-score measure of clustering quality, finding that it only slightly outperforms LexStat, an advantage which seems to hinge mostly on improved clustering methods. In LexStat, IPA input sequences are first converted into sound classes and annotated with sonority profiles. Segment similarity scores are then inferred for each language pair by comparing how often segments are mapped to each other when aligning candidate cognate pairs as opposed to semantically unrelated word pairs, and the Needleman-Wunsch algorithm is applied on these scores. The resulting scores are converted into pairwise form distances (**LexStat distances**), and these distances are used to infer cognate sets based on standard clustering algorithms like UPGMA and InfoMap.

Rama et al. (2017) combine distance scores based on pointwise mutual information (PMI) with InfoMap clustering, and evaluate the resulting system on a range of datasets against LexStat distances, as well as an alternative distance score derived from Pair Hidden Markov Models (PHMM). On a range of testsets, they find that PMI scores trained in an unsupervised fashion using online expectation-maximization, in combination with the Needleman-Wunsch algorithm and InfoMap clustering, beat LexStat by a small margin on a number of datasets.

List et al. (2017b) compare the performance of UPGMA clustering on various older form distance measures with UPGMA on LexStat distances as well as the more advanced InfoMap clustering on LexStat distances, finding that the two LexStat-based approaches clearly perform best, whereas InfoMap clustering only leads to a very small improvement over much simpler UPGMA. This means that the influence of the quality of form similarity scores on the quality of automatically inferred cognate sets seems to be much greater than that of the clustering method used on top of them, motivating the focus of this paper on improving and evaluating form similarity scores.

Whichever clustering method is used, all state-of-the-art cognate detection methods are ultimately based on form similarity or distance scores, whether they are computed from weighted edit distances, PHMMs, or stochastic models. The quality of these scores can (and should) be assessed independently of the clustering method on the level of pairwise cognacy judgments, since any improvement on this level is likely to increase the performance of all clustering methods explored in the literature.

There are quite a few interesting alternative approaches to cognate detection for which no implementations are available. Older approaches like Kondrak (2005) have tended to be evaluated only on very small wordlists from closely related languages. Some newer approaches would be interesting to compare because they provide very general models (Berg-Kirkpatrick and Klein, 2011), or even integrate cognate detection with phylogenetic inference and reconstruction in a comprehensive statistical model (Bouchard-Côté et al., 2013), but their source code remains unavailable, and the system descriptions do not contain all the necessary details for exact reimplementations.

3 Alignment-Based Form Distances

In bioinformatics, the Needleman-Wunsch algorithm is used on standardized and well-tested similarity matrices which encode current knowledge about the different probabilities for each nucleotide base to turn into a different one due to mutation. Unfortunately, no such standard matrix exists so far for phonemes, due to the absence of a global inventory of attested sound changes. One might try to derive such a matrix from phonological knowledge, but methods which estimate a distance matrix from large amounts of data consistently fare better in practice than such attempts, due to the impossibility for a human to assign an intuitive meaning to the distance weights. In practice, phoneme similarity distances are therefore always estimated from large datasets.

Estimation of phoneme similarity builds on observation of segment pairs which are likely to be equivalent given the context. Any method which uses dynamic programming to compute string similarity implicitly constructs an **alignment**, i.e. a separation of the two or more aligned strings into columns of equivalent segments. A binary alignment specifies which phonemes are cognate in a pair of cognate word, providing pairs of observations which can be counted and correlated to build models of phoneme distances. The procedure used in the toolchain for this paper to extract phoneme similarity scores from the NorthEuraLex database is detailed in Section 6.

4 Information Weighting for IPA Sequences

This section revisits the information weighting model which we proposed in Dellert and Buch (to appear), and adds some additional examples and considerations to motivate the changes that were necessary to handle pairwise correspondences. Assume we are faced with the task of assessing the closeness of the English word "to freeze" and its German equivalent "gefrieren". The NorthEuraLex IPA representations of the words are [friiz] and [gəfʁiiʁən], respectively. The NED between these forms is 0.667, which is clearly too high for a pair of cognates from closely related languages. Assuming that we use alignment weights, and that our global similarity matrix additionally tells us that [ɾ] and [ʁ] are a good fit, and (optimistically) that sound correspondence detection will have determined that English [z] clearly coincides with German [ʁ] in some contexts, we would still be left with distance of at least 0.444, i.e. only slightly better than, say, the distance between *sink* [sɪŋk] and *song* [sɔŋ].

The reason for the problems is, of course, that there is some additional material in the German form which would traditionally need to be stripped in order to only map the core portion, the stem *frier-*, to *freeze*. If we cannot extract the stems manually because it would require too much time, or because too little is known about the languages in question (which is frequently the case for languages where automated methods might yield new results), a mathematical model is needed to tell us which bits to ignore, and then a way to incorporate this information into the sequence distance computation.

4.1 Gappy Trigram Models

An important criterion for mathematical models of relevance is that the irrelevant material will be predictable. For instance, the infinitive ending *-ć* is present at virtually every Polish verb, so seeing it at the end of a verbal lexeme is completely unsurprising. Put differently, using the information-theoretic notion of surprise as high information, we will generally find the low-information segments to be more justified to ignore when comparing lexical material across languages. The most direct way to model predictability builds on the probability of seeing the item in question given the knowledge we already have. In phonetic strings, the knowledge we have are the surrounding symbols. If the probability of seeing a segment given

the neighboring segments is very high, this implies low information content. These considerations lead to the use of language-specific (gappy) n-gram models for modeling information content.

Depending on the context, I will use the terms **gappy trigram** and **extended bigram** interchangeably for trigrams where one of the three symbols (the "gap") is a wildcard, i.e. can be replaced by any symbol. For instance, the gappy trigram Xbc represents any of the trigrams abc, bbc, cbc, dbc, etc. A gappy trigram aXb with the gap in the middle could also be called a 1-skip-bigram in the terminology of e.g. Guthrie et al. (2006), but the other two types of extended bigrams are not skip bigrams.

Writing c_{abc}, c_{abX}, c_{Xbc}, c_{aXc} for the trigram and extended bigram counts extracted from all word forms of a language L, the **information content** of a segment c in its five-segment context $abcde$ is defined as

$$I_L(c, [ab\_de]) := -\log \left\{ \frac{c_{abc} + c_{bcd} + c_{cde}}{c_{abX} + c_{bXd} + c_{Xde}} \right\}$$

In words, one combines the number of times c was observed together with the two segments before it, the two segments after it, and its immediate neighbors, and compares this number of observations with the analogous count for the gappy bigrams. It is easy to show that these fractions define a probability distribution over segments in the context $[ab\_de]$, which implies that the negative logarithm does indeed define a measure of surprisal in the information-theoretic sense.

To also be able to define information content values for segments at the start and the end of an IPA representation, the word boundary symbol # is used for expanding a string a of length k to the positions a_{-1}, a_{-0}, a_{k+1}, and a_{k+2}. On these padded strings, no special definition is needed for the trigram and extended bigram counts involving peripheral segments. For instance, the counts for the affricate [t͡ɕ] in Polish *dać* [dat͡ɕ] "to give" are $c_{\text{dat͡ɕ}}$ = 13, c_{daX} = 30, $c_{\text{at͡ɕ\#}}$ = 132, $c_{\text{daX\#}}$ = 339, $c_{\text{t͡ɕ\#\#}}$ = 350, and $c_{\text{X\#\#}}$ = 1124. After smoothing, the information content of [t͡ɕ] in this word is I_{pol}(t͡ɕ,[da_##]) = 1.287. For comparison, an information content I_{pol}(d,[##_at͡ɕ]) = 3.306 is inferred for the first segment.

5 Information-Weighted Sequence Alignment

The next step is to define how the segment-wise information content values are used during alignment. Our solution, first presented in Dellert and Buch (to appear), is to use information content in a modified Needleman-Wunsch algorithm which we call **Information-Weighted Sequence Alignment (IWSA)**.

The modified dynamic programming procedure for computing the raw sequence similarity score $sc(a, b) := M(m, n)$ for two IPA strings $a \in L_a$ of length m and $b \in L_b$ of length n is defined by the following recursion:

$$\begin{array}{lcl} M(0,0) & := & 0 \\ M(i,0) & := & M(i-1,0) + w(a_i, \epsilon) \cdot I^2_{L_a,L_a}(a_i, a_i) \\ M(0,j) & := & M(0,j-1) + w(\epsilon, b_j) \cdot I^2_{L_b,L_b}(b_j, b_j) \\ M(i,j) & := & min \left(\begin{array}{c} M(i-1,j-1) + w(a_i, b_j) \cdot I^2_{L_a,L_b}(a_i, b_j), \\ M(i-1,j) + w(a_i, \epsilon) \cdot I^2_{L_a,L_a}(a_i, a_i), \\ M(i,j-1) + w(\epsilon, b_j) \cdot I^2_{L_b,L_b}(b_j, b_j), \end{array} \right) \end{array} \quad (1)$$

For the $w(a_i, b_j)$, any segment similarity score can be used. My choices for these scores, and their motivation, are described in Section 6. $I^2_{L_a,L_b}(a_i, b_j)$ is defined as the quadratic mean of information weights assigned to the segments:

$$I^2_{L_a,L_b}(a_i, b_j) := \sqrt{\frac{I_{L_a}(a_i, [a_{i-2} \ldots a_{i+2}])^2 + I_{L_b}(b_j, [b_{j-2} \ldots b_{j+2}])^2}{2}}$$

In the cases of insertion and deletion, the score combinations are equivalent to the information content of the segment that was matched to a gap. The quadratic mean is used because it remains high for similar segments with equally high information content, while not penalizing alignment of dissimilar low-information segments, but that of high-information with dissimilar low-information segments. These three properties combine into a focus on matching stems (high-information regions), while discounting differences in low-information parts such as frequently recurring affixes.

In Dellert and Buch (to appear), we only used a single globally inferred phoneme similarity matrix for IWSA. The Needleman-Wunsch scores $sc(a, b)$ were normalized through division by the average self-similarity of both word forms, leading to the following formula for deriving form distance values from similarity scores:

$$d(a, b) := 1 - \frac{2 \cdot sc(a, b)}{sc(a, a) + sc(b, b)}$$

When moving beyond the global similarity matrix and inferring pair-specific phoneme similarity scores in the style of LexStat, this definition turns out to be problematic. If we infer a separate model for each language pair, this leads to very high self-similarity scores because all words can be perfectly aligned with themselves. This causes distance values to be close to 1 for any pair of longer words, even if they are actually very similar. On the other hand, simply using the global similarity scores instead leads to negative distance values, because a specialized model will of course lead to higher $sc(a, b)$ values than the global one. I therefore normalize each similarity score by the length of the relevant sequence:

$$d(a, b) := 1 - \frac{2 \cdot \frac{sc(a,b)}{\max\{n,m\}}}{\frac{sc(a,a)}{m} + \frac{sc(b,b)}{n}}$$

The resulting scores will still often lie beyond the interval $[0, 1]$, but at least there are only very few negative distances, and distances can easily be scaled to the interval if necessary.

6 Inferring Phoneme Similarity Scores

We now turn to the problem of inferring good phoneme similarity matrices. Any model which attempts to estimate such matrices will quickly run into problems if too many parameters are to be estimated from too little data. For many language pairs where we would like to estimate correspondences (e.g. members from different branches of the same family), we often need to get by on little more than 100 cognate pairs. List (2012) reduces the number of similarity scores to estimate by internally reducing IPA (which LingPy accepts as input) to 28 equivalence classes, over which it is easy to estimate sound correspondences even based on only 100 or 200 cognate pairs. On the NorthEuraLex data, where about a thousand word pairs are available for each language pair, we can operate directly on the tokenized and simplified IPA representations in NorthEuraLex, which distinguish 105 IPA symbols.

Most work on phoneme similarity has been based on **pointwise mutual information (PMI)**, which is defined as the logarithm of the observed number of co-occurrences of two events divided by the number of co-occurrences we would expect if both events were independent. Pointwise mutual information has successfully been applied in many areas of computational linguistics. Treating the occurrences of segments in IPA strings as observations of a variable, the pointwise mutual information of two segments x and y would be defined as

$$i(x, y) := \log \frac{p(x, y)}{p(x)p(y)} \tag{2}$$

The problem of PMI scores for sound correspondences is that due to the non-random nature of phoneme sequences (e.g. a preference for the consonant-vowel pattern CVCVCV over CCCVVV), there will always be non-zero mutual information between any pair of consonants and vowels, even if the languages the correspondences are inferred for are completely unrelated. It is unclear how one could correct for this effect in an explicit parametrization of the expected probabilities.

The decisive idea which successfully addresses this problem goes back to Kessler (2001). The similarity scores $w(x, y)$ for IPA segments x and y are PMI scores based on the probability $p(x, y)$ of x being aligned with y in cognate pairs based on counts for a large set of likely cognate pairs, compared to an estimate $\hat{p}(x, y)$ of that probability on non-cognate words:

$$w_{glo}(x, y) := \log \frac{p(x, y)}{\hat{p}(x, y)} \tag{3}$$

6.1 Global Similarity Scores

The interesting decisions are now hidden behind the symbols p and $\hat{p}$. In our architecture, we stay within the framework of aligning randomly chosen wordpairs, and only modify the counting procedure in the case where information weights are used. Our implementation of this in the NWD case is very similar to LexStat, except that LingPy uses multiple-sequence alignments instead of only pairwise alignments to increase consistency. Due to the additional weighting factor, it is not obvious how IWSA could be generalized to multiple-sequence alignment in a consistent way, but as we shall see, this disadvantage does not appear to negatively impact performance.

The distribution $\hat{p}(x, y)$ to compare $p(x, y)$ against is derived by randomly sampling as many word pairs (of any meaning) from random language pairs as there are form pairs of identical meaning in the dataset, aligning each pair in the same way that the cognacy candidates are aligned, and then counting the number of times each pair of phonemes occurred in one column in the resulting alignments. 20% of the overall observation mass is redistributed for Laplace smoothing of the phoneme pair distributions.

$\hat{p}(x, y)$ is kept constant throughout each iteration of re-estimating $p(x, y)$ from a refined set of cognate candidates. Cognate candidates are selected based on an NED threshold (< 0.35) in the initial step, and on a threshold on the Needleman-Wunsch scores (< 1.2) for the current matrix in each subsequent iteration. Due to the large amounts of data in NorthEuraLex, three iterations were enough to derive very stable values for $\hat{c}(x, y)$. Also, no special treatment was needed for the gap symbol, which occurs in correspondences whenever insertions or deletions were inferred in the optimal alignment, and models the costs for inserting or deleting the respective phoneme in the inferred matrices.

In the information-weighted case, our implementation diverges from previous approaches in a crucial way. The probabilities are not directly based on counts of the number of times each symbol pair was aligned, but each instance in a candidate cognate pair only counts with its combined information content. Using the notation $al(a, b)$ for the optimal information-weighted alignment of a word pair (a, b) according to the current segment distances, $sc(a, b)$ for the corresponding Needleman-Wunsch score, and $al(a, b).a$ and $al(a, b).b$ to refer to the individual strings (with gap symbols) in which positions can be indexed by subscripts, this way of counting pairs of aligned segments can be written in one expression as follows:

$$c(x, y) := \sum_{L_1, L_2 \in \mathcal{L}} \sum_{\substack{(a,b) \in lex(L_a, L_b), \\ sc(a,b) < 1.2}} \sum_{\substack{1 \leq i \leq \max\{m,n\}, \\ al(a,b).a_i = x, \\ al(a,b).b_i = y}} I^2_{L_a, L_b}(a_i, b_i) \tag{4}$$

I will call the distance measure based on Needleman-Wunsch scores over the resulting global phoneme similarity matrix **Needleman-Wunsch Distance (NWD)**, whereas the variant using information weighting both for inferring the phoneme similarity scores and during alignment will be called **Information-Weighted Distance (IWD)**. NWD can thus be seen as a special case of IWD where all information weights $I_L(c, [ab\_de])$ are set to 1.

6.2 Pairwise Correspondences for NorthEuraLex

In the same way that global similarity scores were estimated, it is possible to infer specialized scores from the data for any pair of languages. These will tend to assign low costs to sound pairs which are equivalent across many alignments, and can therefore be interpreted as encoding some of the sound correspondences the comparative method operates with. For instance, given enough examples such as *water*/*Wasser*, *street*/*Straße*, and *foot*/*Fuß*, the alignment costs of English [t] and German [s] will be rather low, encoding the consequence of a part of the High German consonant shift.

To infer sound correspondence models for each pair of languages (leading to what I will call **local similarity scores**), we repeat the procedure that we used for inferring the global segment similarities on cognate pairs for that language pair. The values for $\hat{p}(x, y)$ are estimated based on 100,000 random word pairs sampled independently of the associated concepts. Like in LexStat, the inferred similarity scores will be based on a mixture of global and local PMI scores, because the local models for some pairs of unrelated languages will otherwise include some very strong correspondences that are only due to

random noise. To keep the mixture interpretable as an information measure, some weighted mean of both scores needed to be chosen. In order to avoid overfitting the method to the dataset, we based our decision for the proportions between local and global PMI scores on inspection of the resulting correspondences for a small subset of language pairs, and not on optimization. It turned out that one of the simplest option, the arithmetic mean of the local and global PMI scores, already resulted in convincing PMI scores for the inspected language pairs, leading the following definition of local similarity scores:

$$w_{L_1,L_2}(x,y) := \frac{w_{glo}(x,y) + \log \frac{p_{L_1,L_2}(x,y)}{\hat{p}_{L_1,L_2}(x,y)}}{2} \tag{5}$$

Again, using information weighting for the observation counts is crucial for getting good information-weighted alignments. If information weighting is not used for the counts, the local similarity scores will be influenced by reoccurring phonological material, leading to spurious low scores which are then applied across the board. For instance, the frequent mapping of the Persian infinitive ending [æn] to Ukrainian [tɪ] will result in an erroneous high local similarity score [æ] and [t], leading to unwanted small form distances between such pairs as Persian [ælæf] and Ukrainian [trɑwɑ] "grass". More generally, information content weighting of counts will diminish the effect of reoccurring morphological material in non-stemmed data.

I will call the extension of IWD by local similarity scores **Information-Weighted Distance with Sound Correspondences (IWDSC)**. Analogously, the combination of NWD with local similarity scores will be called **Needleman-Wunsch Distance with Sound Correspondences (NWDSC)**.

7 Evaluation

7.1 Test Data

To evaluate whether information weighting does indeed help to better separate cognate from non-cognate word pairs, all five methods were applied to a large IELex-based cognate pair testset which is available as part of the supplementary materials[2] to my dissertation (Dellert, 2017). For this testset, the 6,106 word forms which exist in both IELex and NorthEuraLex were manually mapped to each other, allowing the expert judgments contained in IELex to be added as annotations to the corresponding entries in NorthEuraLex. The resulting testset covers 185 concepts across 36 languages. At a size of 100,156 pairwise judgments, this is one of the largest expert-annotated testsets for cognacy detection currently available.

All the methods that are compared in this evaluation were run on the full version of NorthEuraLex 0.9 (121,615 forms covering 1,016 concepts across 107 languages), including the inference of global and pairwise sound similarity scores. The resulting distance values and clusterings were then reduced to the testset for evaluation. The numbers that will be reported are thus computed on the testset, whereas the dataset used for inference was about 20 times larger. Evaluation in previous literature on cognate detection has relied on testsets that are much smaller in terms of the number of concepts covered, and therefore do not include enough data for building high-quality language-specific information models as needed for information-weighted alignment.

7.2 Results

Using the form distances for pairwise cognate detection boils down to picking a single threshold value θ such that a word pair (a, b) is considered a pair of cognates if and only if $d(a, b) \leq \theta$. For any given threshold, each word pair annotated as cognates in the IELex-based gold standard is counted as a true positive if their distance is below the threshold, and as a false negative if it is not. Analogously, pairs which are considered non-cognates due to a distance above the threshold, are classified into true negatives and false positives depending on the gold standard. This makes it possible to compute precision and recall values for each value of θ.

[2] http://www.sfs.uni-tuebingen.de/~jdellert/pubs/jdellert-diss-supplements.tar.gz

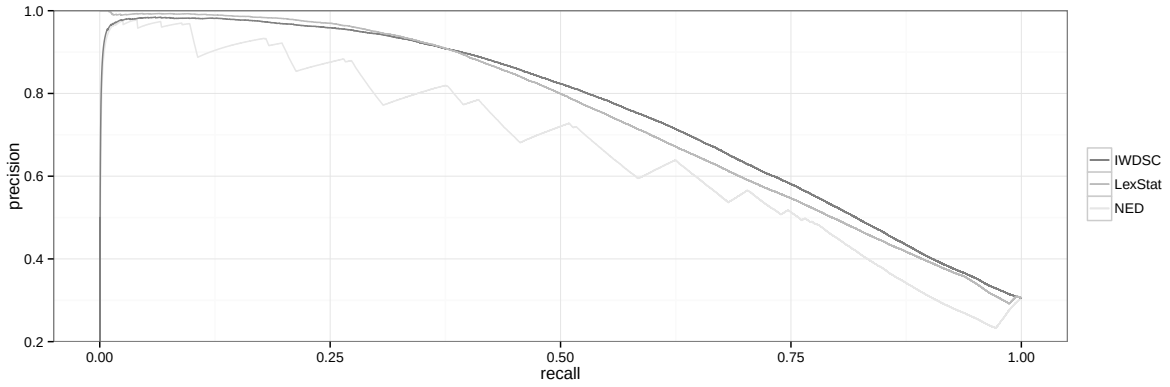


Figure 1: Precision-recall curves for form distance measures in cognacy detection.

As a threshold-independent performance measure, I use the **average precision**, defined as the precision integrated over all possible recall values. To approximate this integral, one simply orders the precision-recall pairs by recall, and then sums up the precision values weighted by the distance between the current recall value and the last, which approximates the area under the precision-recall curve as a sum of rectangles. For illustration, Figure 1 shows the curves for the three most important distance measures in my comparison. In this visualization of the tradeoff between precision and recall, the curve for the NED baseline is jagged and does not drop monotonously. The reason for this is that to ensure a fair comparison, all tied values (of which there are of course many in a distance based on counting edit operations) need to be ordered in the worst-case fashion, i.e. all non-cognate pairs need to be counted as if they had slightly lower distances, followed by the cognate pairs.

Between the smooth curves for LexStat and IWDSC, we observe that in the low-recall range (i.e. for the easy cases), LexStat is slightly better, but its precision decays more quickly with higher recall, showing that information weighting leads to improvement especially for the more difficult instances. The curve already suggests that IWDSC will have a global advantage at desirable recall values of more than 50%, and the average precision values will substantiate this impression.

Table 1 shows the average precision values for all the distance measures discussed in this paper. As already suggested by the curve, IWDSC distances clearly outperform LexStat distances at a gain in average precision of 4.3 percentage points, which in this case corresponds to an average reduction of the false discovery rate by 15.8%.

Method	**NED**	**LexStat**	**NWD**	**NWDSC**	**IWD**	**IWDSC**
Average Precision	0.604	0.727	0.741	0.747	0.764	**0.771**
Maximum F-score	0.599	0.631	0.652	0.654	0.673	**0.679**
Precision at max. F-score	0.639	0.652	0.666	0.660	0.696	**0.706**
Recall at max. F-score	0.564	0.611	0.639	0.648	0.652	**0.654**

Table 1: Quantitative comparison of form distance methods on pairwise cognacy testset.

To evaluate whether this advantage carries over to clustering results, and to maintain comparability with the previous literature, we now turn to the evaluation in terms of B-Cubed F-score for the full cognate detection task, i.e. in terms of the cognate clusters inferred over the testset. For this, the IWDSC values were squared to enhance the differences in the upper range, and then divided by 1.5 (with cutoff at the same value) to scale them to the interval $[0, 1]$ before clustering. To ensure a fair comparison, the main clustering parameter (a single threshold value) was varied with a step size of 0.05 in order to find a near-optimal value for the testset without overfitting. This emulates the usage of LingPy in practice, where on a new dataset, the user is encouraged try other threshold values instead of the default 0.6.

As the numbers in Figure 2 show, the advantage of IWDSC turns out to be far less prominent on the full task than it was for the pairwise cognacy judgments. In combination with UPGMA, IWDSC

does lead to better cognate clusters than LexStat, but only by about 1 percentage point in B-Cubed F-score. Regarding the comparison of clustering algorithms, the NorthEuraLex dataset does not confirm the superiority of InfoMap over much simpler UPGMA found by List et al. (2017b) on their testsets.

Method	Best Threshold	Precision	Recall	F-score
LexStat + InfoMap	0.70	0.757	0.612	0.677
LexStat + UPGMA	0.80	0.763	0.616	0.681
IWDSC + UPGMA	0.75	0.790	0.613	**0.690**

Table 2: B-Cubed measures for LexStat and IWDSC on the NorthEuraLex/IELex testset.

7.3 Discussion

To understand which of the differences between IWDSC and LexStat contributes most to the improvement in average precision, we consider the result figures for the other measures in Table 1. We see that about a fourth of the difference already appears in the comparison between LexStat and our implementation of NWD. This is likely due to the difference in the segment model (with full IPA vs. a reduction to equivalence classes in LexStat), since a test for the only other major difference (the normalization by length during the score-to-distance transformation) did not lead to a lower average precision score.

The performance gain from pairwise segment similarity matrices is of almost exactly the same size for NWD and IWD. This increases my confidence that the small improvement by little more than half a percentage point is a meaningful difference, establishing that there is a gain due to sound correspondences that is orthogonal to and can be combined with the improvements coming from information weighting.

Information weighting alone appears to account for about half of the difference between LexStat and IWDSC, a difference which persists whether we additionally infer sound correspondences (NWDSC vs. IWDSC) or not (NWD vs. IWD). This orthogonality to other improvements makes it a useful technique for inclusion in cognate detection systems.

The small improvement for the full cognacy task is of a similar size to the ones reported by Jäger and Sofroniev (2016) and List et al. (2017b), once more confirming the difficulty to achieve substantial gains over LexStat. To explain how the much more pronounced gain in average precision fits into this picture, the most obvious hypothesis is that while better for pairwise comparisons, the IWDSC distances are not more consistent within cognate sets than LexStat distances, making it difficult to transfer the clear advantage in pairwise judgments to the level of clusters.

8 Conclusion and Future Work

This paper has shown that information weighting and pair-specific phoneme similarity matrices can be used independently to increase the quality of pairwise form distances for automated cognate detection, and that a combination of both techniques improves the Needleman-Wunsch scores for this task.

A large part of the improvement appears due to the use of information weights already for counting the observations that the similarity matrices are based on, whereas the advantage of modeling sound correspondences in the form of local similarity scores turned out to lead to only small additional gains.

The reasons for the small size of the improvement on the full cognate detection task as opposed to the quality of pairwise form distances, and possible remedies, will be a focus of future work. Also, it will be interesting to apply IWSA to other language families as soon as cognacy-annotated databases for other subsets of the lexical data covered by NorthEuraLex (or possible future databases of similar coverage) become available.

Acknowledgements

This research has been supported by the ERC Advanced Grant 324246 EVOLAEMP, which is gratefully acknowledged. The author also wishes to thank three anonymous reviewers for their helpful comments and suggestions, especially the request to also evaluate IWDSC in combination with clustering.

References

Taylor Berg-Kirkpatrick and Dan Klein. 2011. Simple Effective Decipherment via Combinatorial Optimization. In *EMNLP 2011*, pages 313–321. ACL.

Alexandre Bouchard-Côté, David Hall, Thomas L. Griffiths, and Dan Klein. 2013. Automated reconstruction of ancient languages using probabilistic models of sound change. *Proceedings of the National Academy of Sciences*, 10.1073/pnas.1204678110.

Johannes Dellert and Armin Buch. to appear. A New Approach to Concept Basicness and Stability as a Window to the Robustness of Concept List Rankings. *Language Dynamics and Change*.

Johannes Dellert and Gerhard Jäger. 2017. NorthEuraLex (version 0.9). http://northeuralex.org.

Johannes Dellert. 2017. *Information-Theoretic Causal Inference of Lexical Flow*. Ph.D. thesis, University of Tübingen.

Michael Dunn. 2015. Indo-European Lexical Cognacy Database. http://ielex.mpi.nl/.

Russell D Gray and Fiona M Jordan. 2000. Language trees support the express-train sequence of Austronesian expansion. *Nature*, 405(6790):1052–1055.

David Guthrie, Ben Allison, W. Liu, Louise Guthrie, and Yorick Wilks. 2006. A Closer Look at Skip-gram Modelling. In *LREC-2006*, Genoa, Italy.

Gerhard Jäger and Pavel Sofroniev. 2016. Automatic cognate classification with a Support Vector Machine. Proceedings of the 13th Conference on Natural Language Processing (KONVENS).

Brett Kessler. 2001. *The significance of word lists. Statistical tests for investigating historical connections between languages.* CSLI Publications, Stanford.

Grzegorz Kondrak. 2005. N-gram similarity and distance. In *International Symposium on String Processing and Information Retrieval*, pages 115–126. Springer.

V. I. Levenshtein. 1965. Dvoičnye kody s ispravleniem vypadenij, vstavok i zameščenij simvolov. *Dokladi Akademij Nauk SSSR*, 163(4):845848.

Johann-Mattis List, Simon Greenhill, and Robert Forkel. 2017a. LingPy. A Python library for quantitative tasks in historical linguistics.

Johann-Mattis List, Simon J Greenhill, and Russell D Gray. 2017b. The Potential of Automatic Word Comparison for Historical Linguistics. *PloS one*, 12(1):e0170046.

Johann-Mattis List. 2012. LexStat: Automatic Detection of Cognates in Multilingual Wordlists. In *Proceedings of the EACL 2012 Joint Workshop of LINGVIS & UNCLH*, pages 117–125, Avignon, France. ACL.

Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453.

Taraka Rama, Johannes Wahle, Pavel Sofroniev, and Gerhard Jäger. 2017. Fast and unsupervised methods for multilingual cognate clustering. *CoRR*, abs/1702.04938.

Taraka Rama. 2015. Automatic cognate identification with gap-weighted string subsequences. In *Proceedings of NAACL HLT 2015, May 31 June 5, 2015 Denver, Colorado, USA*, pages 1227–1231.

Taraka Rama. 2016. Siamese convolutional networks based on phonetic features for cognate identification. *CoRR*, abs/1605.05172.

Robert R. Sokal and Charles D. Michener. 1958. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 38:1409–1438.

Peter Turchin, Ilja Peiros, and Murray Gell-Mann. 2010. Analyzing genetic connections between languages by matching consonant classes. *Journal of Language Relationship*, 3:117–126.

Søren Wichmann, Eric W. Holman, and Cecil H. Brown. 2016. The ASJP Database (version 17). http://asjp.clld.org/.

Tailoring Neural Architectures for Translating from Morphologically Rich Languages

Peyman Passban, Andy Way, Qun Liu
ADAPT Centre
School of Computing
Dublin City University, Ireland
firstname.lastname@adaptcentre.ie

Abstract

A morphologically complex word (MCW) is a hierarchical constituent with meaning-preserving subunits, so word-based models which rely on surface forms might not be powerful enough to translate such structures. When translating *from* morphologically rich languages (MRLs), a source word could be mapped to several words or even a full sentence on the target side, which means an MCW should not be treated as an atomic unit. In order to provide better translations for MRLs, we boost the existing neural machine translation (NMT) architecture with a double-channel encoder and a double-attentive decoder. The main goal targeted in this research is to provide richer information on the encoder side and redesign the decoder accordingly to benefit from such information. Our experimental results demonstrate that we could achieve our goal as the proposed model outperforms existing subword- and character-based architectures and showed significant improvements on translating from German, Russian, and Turkish into English.

Title and Abstract in Farsi (Persian)

بهبود معماری مدل‌های مبتنی بر شبکه‌های عصبی برای ترجمه از زبان‌های باساختار مورفولوژیکی پیچیده

کلمات با الگوهای پیچیده مورفولوژیکی همانند ساختارهای سلسله مراتبی با واحد (زیربخش) های معنادار هستند، لذا مدل‌های مبتنی بر کلمات که هر کلمه را یک واحد تجزیه‌ناپذیر در نظر می‌گیرند گزینه‌های مناسبی برای پردازش چنین ساختارهای سلسله مراتبی نمی‌باشند. به هنگام ترجمه از یک زبان با ساختارهای پیچیده مورفولوژیکی (ترکی) به یک زبان ساده‌تر از نظر ساختار (انگلیسی)، یک کلمه پیچیده می‌تواند به چندین کلمه و یا حتی یک جمله کامل در زبان مقصد نگاشت (ترجمه) شود، و این پدیده تایید کننده این ضرورت است که کلمات پیچیده نباید همانند واحدهای تجزیه‌ناپذیر درنظر گرفته شوند. در این پژوهش، به منظور تولید ترجمه‌های با کیفیت بهتر، ما سعی کردیم تا معماری متداول (Encoder-Decoder) در مدل‌های ترجمه ماشینی مبتنی بر شبکه‌های عصبی را تغییر دهیم. در معماری پیشنهادی بجای یک کانال در قسمت انکودر دو کانال به صورت همزمان فعال هستند تا اطلاعات غنی‌تری از سمت مبدا را فراهم کنند؛ در سمت دکودر نیز واحد تمرکز (Attention mechanism) موجود با واحد دیگری با ساختار پیچیده‌تر جابجا شده است. هدف اصلی از انجام این پروژه بهبود کیفیت ترجمه و بدست آوردن بازنمایی بهتری از سمت مبدا بود که نتایج آزمایشات انجام شده مؤ ید این دستاورد هستند. نتایج مطالعات نشان می‌دهد که مدل پیشنهادی قادر است عملکرد بهتری از دیگر مدل‌های موجود داشته باشد و در ترجمه از سه زبان مورد مطالعه ما یعنی آلمانی، روسی، و ترکی به انگلیسی موفق‌تر و با دقت بیشتری عمل می‌کند.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3134–3145
Santa Fe, New Mexico, USA, August 20-26, 2018.

1 Introduction

In this paper we propose a neural architecture which is designed to deal with morphological complexities on the source side. In such scenarios the neural model and specifically the encoder should be able to handle morphologically complex inputs. It is hard (or even impossible) to benefit from all subunit information within an MCW when it is treated as an atomic unit, e.g. the Turkish word '*ev.de.ki.ler.de.ki*' meaning '*one of those (things/people) at home*' clearly demonstrates that there are meaningful subunits within the word which should be processed separately. Expecting a neural model to learn such intra-word relations by itself complicates the whole process drastically as it requires powerful neural structures which may not lead to the desired result. Accordingly, we believe that there should be an architectural design (manipulation) inspired by the nature of this particular problem, which is able to decompose MCWs or process them in decomposed forms.

Subword-level NMT models are the most preferred alternatives to translate from MRLs, which can be discussed in two main categories. One group of models does not change the neural architecture but manipulates data by decomposing words into their subunits. In this approach either linguistically motivated morphological analyzers such as Morfessor (Smit et al., 2014) or purely statistical models such as the byte-pair encoding (*bpe*) model (Sennrich et al., 2016) are applied to process words. This approach, by its very nature, seems to be a promising solution as it changes the sparse surface form-based vocabulary set into a much smaller set of fundamental subunits. The size of a set including atomic subunits, especially for MRLs, is considerably smaller than that of the vocabulary set. Moreover, this solution partially solves the out-of-vocabulary word problem, as the chance of facing an unknown surface form is much higher than the chance of facing an unknown subunit.

The aforementioned approach could help generate better translations, but there is a potential problem with its single encoder shared among all different types of subunits. In the Morfessor-based segmentation (or any other linguistically motivated segmentation models) the NMT model is fed by stems and affixes instead of words, but all those subunits are processed with a single encoder. Clearly, this is not the best way of learning/representing source-side information, as stems and affixes are different constituents and should be processed separately. The same problem also applies to statistical models such as *bpe*, because although the difference between different subunits might not be obvious (as that of stems and affixes), it does not mean that they are the same types and should be treated equally. Furthermore, there could be another issue raised by the type of the neural architecture. The encoder-decoder architecture (Cho et al., 2014; Sutskever et al., 2014) employed in this approach was originally designed to work at the word level but is instead used for subwords, which might produce some problems, e.g. a sequence of subwords can be considerably longer than a sequence of words. Long sequences are usually hard to remember for neural models and this can affect NMT model's performance. Considering all positive and negative aspects of subword-based models, we still believe that such models are truly beneficial but our main concern is their architecture which we demonstrate can be improved to work with subwords.

As we previously mentioned, there are two main categories of models proposed to tackle the problem of translating from MRLs. We already explained the first group and its advantages and disadvantages. Unlike the first group, the second one focuses more on the neural architecture rather than data preprocessing (word segmentation), and boosts the network with character-based encoders. In such models words are consumed character by character. Usually, there is a convolutional module after the input layer to read characters, connect them to each other, and extract inter-character relations. This approach is helpful as it requires no preprocessing step, but there are two main problems with that. Similar to the previous group, the length of the input sequences could be an issue as by segmenting words into characters an input sequence with a limited number of words is changed to a long sequence of characters. Furthermore, the convolutional computation over characters could be quite costly, e.g. Lee et al. (2017) use 200 convolutional filters of width 1 along with 200 filters of width 2, 250 filters of width 3, and continue up to a filter size of 300 with width 8 to extract useful information on the source side. This amount of computation is carried out only in one layer (for one word), so in addition there are max-pooling

layers followed by 4 highway layers (Srivastava et al., 2015), and then recurrent layers on top. This is a complex architecture which is determined by the type of the input data (a sequence of characters).

Considering both groups of approaches, we assert that there is a need for a third one, for two main reasons: i) models from the first approach work at the subword level but the neural architecture they use is not designed for this purpose; and ii) systems from the second approach use complex architectures to be able to learn character-level information but (it seems) the complex mechanism is not completely effective as results reported from different studies demonstrate that character-based models usually fail to compete with other architectures.[1] Our experimental results also confirm this (see Table 2). The main issue we try to point out by studying these approaches is that existing models may provide state-of-the-art results but they are not designed based on the nature of the problem at hand. Rather, they are just powerful models with recurrent, convolutional, and highway layers which work with simplified (segmented) inputs. They might improve translation quality or partially address the problem, but do not resolve, in a meaningful way, the challenge of handling morphological (subword-level) complexities on the source side.

In this paper we propose a new architecture to address the shortcomings of these previous approaches which is our main contribution. We prefer to work with subwords not characters, because we might lose useful information when we segment structures into characters, or might not be able to extract desired information via the character-based architecture. Related characters are already grouped and connected to each other in subwords, so there is no need to decompose and then glue them back together. Therefore, our basic unit for translation is a morpheme (subword). As mentioned earlier, it seems that existing architectures can be improved further to work with subwords. Accordingly, we propose a novel neural architecture which relies on subwords and is supposed to outperform other character- and subword-based counterparts. In the next section, we first review similar models which tackle the same problem with different approaches, then we explain our own solution. The comparison between our model and others confirms that our hypothesis on designing the neural architecture according to the nature of the problem is true and directly affects performance.

2 Related Work

The problem of morphology and dealing with MCWs is an important issue in the field of sequence modeling as it directly affects the model's architecture and its performance. There is a fair amount of research carried out to address this problem, which can be discussed in two main categories. One group of models tries to cope with the problem on the encoder side where the goal is to understand the rich morphology of the source language. Kim et al. (2016) proposed a convolutional module to process complex inputs for the problem of language modeling. Costa-jussà and Fonollosa (2016) and Lee et al. (2017) adapted the same convolutional encoder to NMT. Luong and Manning (2016) designed a hybrid character- and word-based encoder to try to solve the out-of-vocabulary problem. Vylomova et al. (2016) tackled the problem by comparing the impact of different representation schemes on the encoder. Similarly, Burlot et al. (2017) investigated the impact of different word representation models in the context of factored NMT. Our work is also an example of models which try to provide richer information when the source side is an MRL.

Models reviewed so far address the problem of morphology on the source side. In contrast, there is a group of models which study the same problem for the target side. Huck et al. (2017) compared different word-segmentation models, including linguistically motivated as well as statistical techniques, to find the most appropriate segmentation scheme when translating into MRLs. Chung et al. (2016) tried to design a suitable architecture when the target language is an MRL. They benefit from using a character-based decoder which partially resolves the problem. Passban et al. (2018) proposed a similar approach in which they equipped the character-based decoder with an additional morphology table to inform the decoder with the target language's morphological structures.

[1] Comparing translation results generated by different word-, subword-, and character-based architectures on different language pairs shows that there is no character-based model which is able to outperform its word- and/or subword-based counterparts. Please see results at `http://matrix.statmt.org/?mode=all`

Apart from these models, there are others that do not directly address the problem of morphology but their solutions could be quite useful to translate MRLs. Jean et al. (2015) proposed a model to handle very large vocabularies. Luong et al. (2015) addressed the problem of rare and unseen words with the help of a post-translation phase to exchange unknown tokens with their potential translations. Dalvi et al. (2017) did not propose a new model but studied the impact of morphological information in NMT. They evaluated the behaviour of an encoder-decoder model to see what sort of morphological information is learned via the model and how the model deals with complex structures. Passban (2018) extensively discussed the problem of morphology at the word and sequence level and proposed solutions for modeling and translating sequences in monolingual and bilingual settings.

3 Proposed Approach

We propose an NMT architecture with a double-source encoder and double-attentive decoder for translating from MRLs. Our neural architecture is inspired by models proposed in Firat et al. (2016) and Zoph and Knight (2016). It takes inputs from two different channels: one channel which is referred to as the *main* channel sends stem information (main input), and the other one (the *auxiliary* channel) sends affix information. If the input is $w_0, w_1, ..., w_n$ for the (original) encoder-decoder model, our proposed architecture takes two sequences of $\varsigma_0, \varsigma_1, ..., \varsigma_n$ and $\tau_0, \tau_1, ..., \tau_n$ through the main and auxiliary channels, respectively, where w_i shows the surface form of a word whose stem is ς_i, and affix information associated with w_i is given by τ_i.

Our new neural architecture is based on a hypothesis which assumes the *core semantic unit* is the stem. The translation generated on the target side could appear in different surface forms but it should convey the core meaning dictated by source-side stems. Therefore, to generate a translation the minimum requirement is stem information. Defining the translation process based on stems considerably simplifies the problem, because we no longer need to work with complex surface forms. The sentence representation generated based on stems provides the decoder with high-level source-side information. While it could steer the decoder toward potentially correct target tokens, the decoder needs more than this to generate precise translations. Accordingly, stem information is accompanied with auxiliary information supplied by the affix channel.

There are two main motivations underpinning this type of modelling. We can have different words on the target side (morphologically naive) which could be translations of a single stem. The reason they differ from one another and appear in different forms is because the source-side stem collocates with different affixes. By having different input channels we can model these combinations to help the decoder with the generation of the translation. The double-channel encoder can understand such combinations better, as it separately processes the stem and its affixes. This is the first motivation for our design. Moreover, the neural model is not totally able to extract all information preserved in MCWs when it works with surface forms, but we can simplify the process by explicitly showing subword units to the network, which is a novel point in our architecture. This is the second reason why we process stems and affixes separately.

3.1 The Double-Channel Architecture

Our neural model is an extension to the encoder-decoder model of Cho et al. (2014) with some fundamental changes. In our architecture, given an input sequence $w_1, w_2, ..., w_n$, words are segmented into their stems (ς) and affix tokens (τ). Regarding the structure of the input data, our encoder should process two tokens (stem and affix) instead of one (word) at each time step, so the neural architecture is adapted to the new data structure. To this end, we have two encoders where the stem encoder reads stems one after another and the other one (affix encoder) reads affix tokens. The process can be formulated as in (1):

$$\begin{aligned} s_t &= f(s_{t-1}, \varsigma_t^e) \\ a_t &= f(a_{t-1}, \tau_t^e) \end{aligned} \tag{1}$$

where s_t and a_t are the hidden states of the stem and affix encoders, respectively, at time step t. ς_t^e indicates the embedding of the t-th word's stem, and τ_t^e is the embedding of the affix token for the same

word.

When the final time step is reached for an input sentence of length n, we have two vectors that represent the summary of stem and affix sequences. The decoder should be informed about source-side information to start sampling/generation, for which we place a fully connected layer with a transformation matrix between the encoder and decoder to map both source vectors to the first hidden state of the decoder, as in (2):

$$h_0 = tanh\big(W_{\varsigma\tau}[s_n \bullet a_n]\big) \tag{2}$$

where h_0 is the decoder's hidden state at the beginning, *tanh* applies non-linearity, $W_{\varsigma\tau} \in \mathbb{R}^{(|s_n|+|a_n|)\times|h_0|}$ is the transformation matrix, and $\bullet$ indicates the concatenation operation.

At each time step the decoder samples a token from the target vocabulary. In the simple encoder-decoder model, the decoder conditions the prediction on its hidden state h_t, the last predicted token y_{t-1}, and a dedicated context vector $\mathbf{c}_t$ generated by the attention mechanism (Bahdanau et al., 2014) over the encoder's hidden states. In our case, we have two sets of source-side hidden states and the decoder pays attention to both (instead of one). This process is simply formulated as in (3):

$$y_t = g(h_t, y_{t-1}, \mathbf{c}_t^{\varsigma}, \mathbf{c}_t^{\tau}) \tag{3}$$

Our decoder benefits from a double-attentive mechanism instead of the simple attention model. $\mathbf{c}_t^{\varsigma}$ is the stem-based context vector and provides information about source stems, and $\mathbf{c}_t^{\tau}$ is the affix context vector which informs the decoder about morphological properties of the input sequence.

In order to construct the stem-based context vector $\mathbf{c}_t^{\varsigma}$, we use exactly the same attention model proposed by Bahdanau et al. (2014), as in (4):

$$\begin{aligned} \mathbf{c}_t^{\varsigma} &= \sum_{t'=1}^{n} \alpha_{tt'} s_{t'} \\ \alpha_{tt'} &= \frac{\exp(e_{tt'}^{\varsigma})}{\sum_{k=1}^{n} \exp(e_{tk}^{\varsigma})} \\ e_{tt'}^{\varsigma} &= \mathbf{a}_{\varsigma}(s_{t'}, h_{t-1}) \end{aligned} \tag{4}$$

where $\mathbf{a}_{\varsigma}$ is a simple feed-forward connection. The equation shows that the decoder tries to select relevant stems at each time step which can help it sample better tokens. $\mathbf{c}_t^{\varsigma}$ summarizes all source stems and informs the decoder about the impact of each stem. For the affix context vector, the attention model has a slightly different mechanism whose detail is shown in (5):

$$\begin{aligned} \mathbf{c}_t^{\tau} &= \sum_{j=t'}^{n} \beta_{tt'} a_{t'} \\ \beta_{tt'} &= \frac{\exp(e_{tt'}^{\tau})}{\sum_{k=1}^{n} \exp(e_{tk}^{\tau})} \\ e_{tt'}^{\tau} &= \mathbf{a}_{\tau}(a_{t'}, [h_{t-1} \bullet \mathbf{c}_t^{\varsigma}]) \end{aligned} \tag{5}$$

Affix tokens are weighted given the hidden state of the decoder together with the stem context vector. We know that affixes associate with their stems, so to select the best affix we need to consider both the impact of stems and the decoder's information. Experimental results show that this combination obtains better performance (see Section 4). Figures 1 and 2 visualize our double-attentive attention module. They provide a comparison between simple and double-attentive attention models. It should be noted that for simplicity our figures only show connections associated with the attention module (and exclude others).

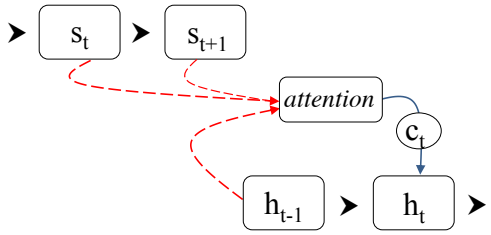


Figure 1: This figure shows the original attention mechanism where the source side-encoder consumes one word at each time step and updates its hidden state s_t. The decoder constructs the context vector $\mathbf{c}_t$ based on the decoder's previous hidden state and all source-side hidden states. The context vector is used along with the (current) hidden state h_t to generate the output.

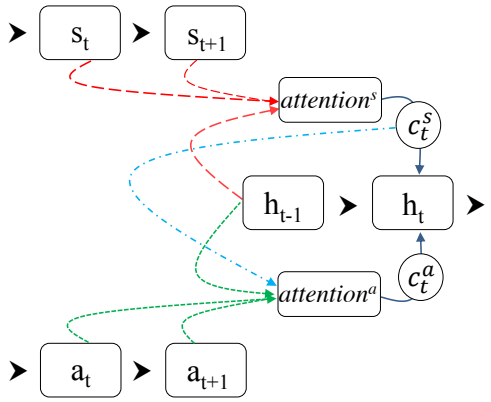


Figure 2: This figure illustrates the double-source encoder, where the stem encoder takes one stem at each time step and updates s_t, and the affix encoder updates a_t after consuming the t-th affix token. The stem-based context vector is depicted by $\boldsymbol{c}_t^s$. The affix-based context vector is constructed using information provided by all of the affix-encoder's hidden states, the decoder's previous hidden state, and the stem context vector.

4 Experimental Study

Our NMT model is an encoder-decoder model with gated recurrent units (GRUs) (Cho et al., 2014). On the encoder side we have two encoders, one for stems and the other for affix tokens. Both encoders include two GRU layers where the first layer is a bidirectional layer and the second layer is a unidirectional layer. Stems and affix tokens are represented with 512-dimensional embeddings. On the decoder side, we have the double-attentive attention module and two GRU layers. The GRU size for both the encoder and decoder is 1024. Similar to input tokens, target tokens are also represented with 512-dimensional embeddings. The network was trained using stochastic gradient descent with Adam (Kingma and Ba, 2015). The mini-batch size is 80, the beam search width is 20, and the norm of the gradient is clipped with the threshold 1.0.

4.1 Data Preparation

Our models are trained to translate from German (De), Russian (Ru), and Turkish (Tr) into English (En). For German and Russian, we used `WMT-15` datasets,[2] where the De–En corpus includes 4.5M parallel sentences and the size of the Ru–En corpus is 2.1M. The `newstest-2013` and `newstest-2015` datasets are used as the development and test sets, respectively. For Tr–En, we used the `OpenSubtitle2016` collection (Lison and Tiedemann, 2016).[3] We randomly selected 3K sentences for each of the development and test sets, and 4M for training.

[2] `http://www.statmt.org/wmt15/translation-task.html`
[3] `http://opus.nlpl.eu/OpenSubtitles2016.php`

	English–German		English–Russian		English–Turkish	
	En	**De**	**En**	**Ru**	**En**	**Tr**
Sentence	4.2M		2.1M		4.0M	
Token	103,692,553	96,235,845	43,944,989	39,694,475	24,875,286	17,915,076
Type	103,574	143,329	70,376	119,258	108,699	178,672
Stem	21,223	26,301	15,964	19,557	15,714	17,419
Affix	13,410	24,054	9,320	17,542	7,112	8,041
Prefix	3,104	5,208	2,959	3,324	1,211	753
Suffix	3,285	4,974	2,219	3,460	1,766	2,701
Char	355	302	360	323	189	231

Table 1: *Sentence* is the number of parallel sentences in the corpus. *Token* is the number of words. *Type* shows the number of unique words. *Stem*, *Prefix*, *Suffix*, and *Affix* show the number of unique stems, prefixes, suffixes, and affix tokens in the corpus. *Char* is the number of unique characters appeared in the corpus.

We segment input words with Morfessor. The longest subunit is selected as the stem. What appears before the stem is the prefix and what follows the stem is considered as the suffix. We do not postprocess Morfessor's output, e.g. Turkish is a suffix-based language with no prefixes, but Morfessor extracted some prefixes for this language. We use segmented units as they are and do not change them. Table 1 summarizes some statistics about our training corpora.

It is clear what the stem encoder takes as its input, but for a word we may have none (null), one or many prefixes and suffixes, whose combination constructs the affix token in our setting. For the affix encoder we have a look-up table which includes prefix and suffix embeddings. The encoder retrieves associated prefix and suffix embeddings at each time step, combines them (via vector summation) and sends the combined vector to the encoder. This architecture is referred to as *pre+suf* in our experiments. If a word has more than one prefix/suffix the encoder retrieves all of them, and if it has none the encoder uses a *null* embedding.

In the *pre+suf* setting, what the affix encoder receives as its input is a vector which has some information about morphological properties of the associated stem (word), but it is possible to provide the encoder with richer information, so we propose the *affix* extensions (*affix*-$\mathbf{c}^\tau$ and *affix*-$\mathbf{c}^\varsigma\mathbf{c}^\tau$).[4] In this new setting we combine the surface form of all prefixes and suffixes to generate a new affix token, e.g if $word_i = prefix_i^1.stem_i.suffix_i^1.suffix_i^2$, the affix token τ_i would be $prefix_i^1.suffix_i^1.suffix_i^2$. We assign a unique embedding for this new combination. Clearly, the embedding of the new affix token is different from the summation of the embeddings of the prefix(es) and suffix(es) and introduces a new token.

A real example can show the difference between the *pre+suf* and *affix* extensions. If the given word is *pre.process.ing*, in the first extension *pre+suf*, the stem encoder takes the embedding of *process* and the affix encoder takes the summation of the embeddings of *pre* and *ing*. In the second extension (either *affix*-$\mathbf{c}^\tau$ or *affix*-$\mathbf{c}^\varsigma\mathbf{c}^\tau$), the stem token is the same but the affix encoder takes an embedding which is exclusively defined for *pre.ing*.

The embedding of the affix token is a new unit which preserves information about: i) morphological properties of its associated word, ii) morphological properties of other words appearing in the sentence, and iii) the collocation and order of different prefixes and suffixes, at the word and sentence level. It seems that the simple combination of prefix and suffix embeddings as in the *pre+suf* extension cannot preserve this amount of information, because the role of each suffix/prefix is clear and their summation will provide us with specific information, while in the second extension we can expect the neural network to learn our desired information. Basically, by defining the affix token we try to introduce an additional cell of memory, in which the NMT model and the affix encoder store the useful information it has learned

[4]In these two extensions everything is the same but the attention mechanism. Equations (5) and (6) explain the attention mechanisms for *affix*-$\mathbf{c}^\varsigma\mathbf{c}^\tau$ and *affix*-$\mathbf{c}^\tau$, respectively.

(in addition to stem).

4.2 Experimental Results

The results obtained from our experiments are reported in Table 2. The numbers in the table are BLEU scores (Papineni et al., 2002) of different neural models. We compare our models to all existing models which translate from MRLs or reported experimental results on our datasets. The first row of the table shows an encoder-decoder model where the decoder works at the character level and uses the architecture proposed in Chung et al. (2016). The first row can be considered as a baseline for all other models reported in the table, as it does not use any complicated neural architecture on the source side. For each word it simply sums stem, prefix, and suffix embeddings together and sends the summed vector as the word-level representation to the GRU-based encoder. Although the model is quite simple, it is able to generate comparable results to other more complex architectures, which reinforces our claim that existing neural architectures are not suitable to work at the subword level. If we find a better way to provide the neural model with subword information, we will be able to improve translation quality still further. For all other models we use the same neural architecture where we keep the character-based decoder unchanged and only change the encoder to compare the capacity of different encoders in modeling morphological information.

Model	Source	Target	De→En	Ru→En	Tr→En
Baseline	ς+*pre*+*suf*	*char*	22.11	22.79	22.98
Costa-jussà and Fonollosa (2016)	*word*	*word*	18.83	-	-
	char	*word*	21.40	-	-
Firat et al. (2016)	*bpe*	*bpe*	24.00	22.44	-
Lee et al. (2017)	*bpe*	*char*	25.27	22.83	-
	char	*char*	25.83*	22.73	-
Sennrich and Haddow (2016)$\star$	$\varsigma \bullet \tau$	*char*	22.41	23.01*	23.14*
Our model					
pre+*suf*	$[\varsigma]^1$ [*pre*+*suf*]^2	*char*	26.11	22.95	23.62
affix-$\mathbf{c}^{\varsigma}\mathbf{c}^{\tau}$	$[\varsigma]^1$ $[\tau]^2$	*char*	**26.74**	**23.44**	**23.81**
affix-$\mathbf{c}^{\tau}$	$[\varsigma]^1$ $[\tau]^2$	*char*	26.29	23.40	23.74

Table 2: Source and Target indicate the data type for the encoder and decoder, respectively. ς is the stem, pre is the prefix and suf is the suffix. τ is the affix token. The bold-faced score is the best score for the direction and the score with * shows the best performance reported by other existing models. According to paired bootstrap re-sampling (Koehn, 2004) with $p = 0.05$, the bold-faced number is significantly better than the score with *. Brackets show different channels and the + mark indicates the summation, e.g. $[\varsigma]^1$ [*pre*+*suf*]^2 means the first channel takes a stem at each step and the second channel takes the summation of the prefix and suffix of the associated stem. $\bullet$ is the concatenation operation and $\star$ indicates that results were produced in our experimental setting using our re-implementation of the original model.

The second row shows the model proposed by Costa-jussà and Fonollosa (2016), in which a complicated convolutional module is used to model the relation between characters and build the word-level representation. It shows that the character-level modeling works better than the word-level model but the simple baseline engine still outperforms both settings. The third row belongs to Firat et al. (2016) which is a multi-way multilingual NMT model. The fourth row shows a fully character-level NMT model (Lee et al., 2017) where both the encoder and decoder are designed based on characters. As previously discussed, this model has quite a complicated architecture.

The fifth row reports results from the model of Sennrich and Haddow (2016) that is a suitable model to cope with the problem of complex structures on the source side. The idea behind the model is to enrich input words via additional annotations such as part-of-speech and/or morphological tags. The embedding of the additional annotation(s) is concatenated to the embedding of the word/stem. The idea of attaching

auxiliary information to words/stems was successful and our experimental results also confirm this. In their original model the training data is tagged with external tools which show the need for annotated training sets. We borrowed the same idea and tried to evaluate that architecture in our setting. In the original model, proper morphological tags are defined for input words/stems whereas in our case we consider the affix token as the morphological tag of each stem and we do not need a morphological tagger. The embedding of the affix token is concatenated to the stem embedding and sent to the encoder. Accordingly, the NMT engine is an extension to the baseline model in which the encoder is fed by concatenated structures and the decoder still has the same character-based architecture. This simple concatenation provides better results than the summation applied in the baseline model, which approves affix and subword level information is useful but we only need to find an optimal way to explore it.

The last block shows 3 variations of our model. In the *pre+suf* model the encoder has two encoding channels, one for stems and the other for prefixes and suffixes. At each time step, the stem encoder takes one stem embedding. On the other channel, the affix encoder takes an embedding which is the summation of the prefix and suffix embeddings of the word whose stem is processed by the stem encoder at the same time step. *pre+suf* employs our novel double-attentive attention mechanism. This new architecture enables the NMT engine to perform better than the complicated fully character-based model, which is an indication that the character-level representation does not necessarily provide a better representation. We think that our model outperforms the character-based model because its architecture is more compatible with the nature of MRLs. The impact of the compatible neural architecture becomes more important when we compare the baseline and *pre+suf* models. The input format is exactly the same for both, but the baseline model simply sums stem, prefix, and suffix embeddings, whereas *pre+suf* has two separate channels to process stem and affix information.

The *affix*-$\mathbf{c}^{\varsigma}\mathbf{c}^{\tau}$ variation is the best model among our double-channel models. As previously discussed, assigning a unique embedding to the combination of prefixes and suffixes instead of summing their embeddings generates better results and provides richer information.

The third and last variation, *affix*-$\mathbf{c}^{\tau}$, shows the impact of our architecture on the decoder side. As we modeled in Equation (5), attention weights assigned to the affix-encoder's hidden states are computed based on the decoder's hidden state and the stem context vector. As affix tokens provide complementary information to stem information, the affix context vector should be aware of the content of the stem context vector, so we proposed the model explained in Equation (5). If we only consider the decoder's information to compute affix weights, the equation will be revised as in (6):

$$\begin{aligned} \mathbf{c}_t^{\tau} &= \sum_{j=t'}^{n} \beta_{tt'} a_{t'} \\ \beta_{tt'} &= \frac{\exp(e_{tt'}^{\tau})}{\sum_{k=1}^{n} \exp(e_{tk}^{\tau})} \\ e_{tt'}^{\tau} &= \mathbf{a}_{\tau}(a_{t'}, h_{t-1}) \end{aligned} \tag{6}$$

In the *affix*-$\mathbf{c}^{\varsigma}\mathbf{c}^{\tau}$ version, the energy between the decoder's (t-1)-th state and the affix-encoder's t'-th state was computed by $e_{tt'}^{\tau} = \mathbf{a}_{\tau}(a_{t'}, [h_{t-1} \bullet \mathbf{c}_t^{\varsigma}])$, whereas this version simplifies the computation by estimating $e_{tt'}^{\tau}$ with $\mathbf{a}_{\tau}(a_{t'}, h_{t-1})$. The model described in Equation (6) is implemented in the *affix*-$\mathbf{c}^{\tau}$ extension. Results obtained from this extension show that, although the architecture is also successful compared to other exiting models, its performance is worse than the model which (additionally) involves the stem context vector to compute affix weights.

5 Conclusion and Future Work

In this paper we focused on morphological complexities on the source side in the NMT task and equipped the encoder to handle complex inputs. The proposed model includes two separate channels to consume both stem and affix tokens. This novel two-channel solution can improve performance in two ways, which can be discussed from linguistic and computational perspectives. In terms of linguistic issues,

when the first channel deals with stems instead of words, it is asked to process considerably fewer constituents as the number of stems are much fewer than the number of words. The network is trained to translate simple stem forms instead of complex words. This simplification is usually helpful when translating (from/into) MRLs (Sennrich and Haddow, 2016). Furthermore, there is another channel which preserves morphological information. Every time that the first channel is not able to provide precise translations or needs auxiliary information, the second channel completes it with morphological information.

The proposed architecture can also be studied from mathematical and computational points of view. Defining a secondary channel means adding extra neural modules and parameters (weights). Increasing the number of neural parameters increases the learning capacity of the network and gives a higher possibility to achieve better results. Experimental results confirm that this happens in practice and the extra channel is helpful.

Existing subword-based models segment MCWs using Morfessor, *bpe*, and other models, and feed the translation engine with simplified units. Character-based models also try to facilitate the process by working with characters. Both approaches are useful and improve performance but the improvement is somewhat random, as we do not exactly know what/where the origin of this improvement is. By proposing our double-channel architecture we address this problem and feed the network in a more structured way, in the expectation that we can represent the impact of morphological information. While we are still far from the goal of modeling morphological information for NMT, this research is an important step toward our goal.

Along with these useful properties of our approach, it also has some deficiencies which we plan to address in future work. Our affix information is extracted via Morfessor which could be noisy. Affix tokens are supposed to provide morphological information but we are not sure that our interpretation matches what actually happens in reality. According to experimental results, affix tokens are useful but we are not sure if they preserve real morphological information. In our future work we plan to linguistically enrich our pipeline with more precise and relevant information. As another plan, we also tend to focus on the decoder.

Acknowledgment

We thank our anonymous reviewers for their valuable feedback, as well as the Irish centre for high end computing (www.ichec.ie) for providing computational infrastructures. This work has been supported by the ADAPT Centre for Digital Content Technology which is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, Banff, Canada.

Franck Burlot, Mercedes García-Martínez, Loïc Barrault, Fethi Bougares, and François Yvon. 2017. Word representations in factored neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 20–31, Copenhagen, Denmark.

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

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1693–1703, Berlin, Germany.

Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 357–361, Berlin, Germany, August.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, and Stephan Vogel. 2017. Understanding and improving morphological learning in the neural machine translation decoder. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 142–151.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California, June.

Matthias Huck, Simon Riess, and Alexander Fraser. 2017. Target-side word segmentation strategies for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 56–67, Copenhagen, Denmark.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China, July.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 2741–2749, Phoenix, Arizona, USA.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, USA.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, vol. 5:365–378.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, pages 923–929, Portorož, Slovenia.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, Berlin, Germany, August.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China, July.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318, Pennsylvania, PA., USA.

Peyman Passban, Qun Liu, and Andy Way. 2018. Improving character-based decoding using target-side morphological information for neural machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL*, New Orleans, Louisiana, USA, Jun.

Peyman Passban. 2018. *Machine Translation of Morphologically Rich Languages Using Deep Neural Networks*. Ph.D. thesis, School of Computing, Dublin City University, Ireland.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91, Berlin, Germany, August.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August.

Peter Smit, Sami Virpioja, Stig-Arne Grönroos, and Mikko Kurimo. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 21–24, Gothenburg, Sweden.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. In *Proceedings of the ICML Deep Learning Workshop*, Lille, France.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Proceedings of the Confrence on Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112. Montreal, Canada.

Ekaterina Vylomova, Trevor Cohn, Xuanli He, and Gholamreza Haffari. 2016. Word representation models for morphologically rich languages in neural machine translation. *CoRR*, abs/1606.04217.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34, San Diego, California, June.

deepQuest: A Framework for Neural-based Quality Estimation

Julia Ive[1] **Frédéric Blain**[2] **Lucia Specia**[2]
King's College London, IoPPN, London, SE5 8AF, UK[1]
Department of Computer Science, University of Sheffield
Regent Court, 211 Portobello, Sheffield, S1 4DP, UK[2]
julia.ive@kcl.ac.uk, {f.blain, l.specia}@sheffield.ac.uk

Abstract

Predicting Machine Translation (MT) quality can help in many practical tasks such as MT post-editing. The performance of Quality Estimation (QE) methods has drastically improved recently with the introduction of neural approaches to the problem. However, thus far neural approaches have only been designed for word and sentence-level prediction. We present a neural framework that is able to accommodate neural QE approaches at these fine-grained levels and generalize them to the level of documents. We test the framework with two sentence-level neural QE approaches: a state of the art approach that requires extensive pre-training, and a new light-weight approach that we propose, which employs basic encoders. Our approach is significantly faster and yields performance improvements for a range of document-level quality estimation tasks. To our knowledge, this is the first neural architecture for document-level QE. In addition, for the first time we apply QE models to the output of both statistical and neural MT systems for a series of European languages and highlight the new challenges resulting from the use of neural MT.

1 Introduction

Quality Estimation (QE) (Blatz et al., 2004; Specia et al., 2009) aims at predicting the quality of machine translation (MT) without human intervention. Most recent work has focused on QE to predict sentence-level post-editing (PE) effort, i.e. the process of manually correcting MT output to achieve publishable quality (Bojar et al., 2014; Bojar et al., 2015; Bojar et al., 2016a; Bojar et al., 2017). In this case, QE indicates to what extent a MT sentence needs post-editing. Document-level QE, on the other hand, scores or ranks documents according to their quality for fully automated MT usage scenarios where no post-editing can be performed, e.g. MT for gisting of news articles online.

Recently, neural methods have been successfully exploited to improve QE performance. These methods mostly rely on either complex architectures, require extensive pre-training, or need some feature engineering (Patel and M, 2016; Kim et al., 2017a; Martins et al., 2017a; Jhaveri et al., 2018). In addition, these methods have only been developed for word, phrase and sentence-level QE. These cannot be directly used for document-level QE since this level requires to take into account the content of the document in its entirety. State-of-the-art document-level QE solutions still rely on non-neural methods, and extensive feature engineering (Scarton et al., 2016).

In this paper we propose a neural framework that is able to accommodate any QE approach at a fine-grained level (e.g. a sentence-level approach), and to generalize it to learn document-level QE models. We test the framework using a state of the art neural sentence-level QE approach (Kim et al., 2017b), which uses a complex architecture and requires resource-intensive pre-training, and a light-weight neural approach employing simple encoders and no pre-training. Our sentence-level prediction approach leads to comparable or better results than the state of the art at a much lower cost. Additionally, the document-level framework improves over previous work by a large margin. To our knowledge, this is the first attempt at document-level QE using purely neural methods.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3146–3157
Santa Fe, New Mexico, USA, August 20-26, 2018.

The majority of existing QE solutions for all prediction levels have been designed and tested for Statistical MT (SMT). Popular features extracted from SMT translation models are clearly no longer applicable to neural MT (NMT), while MT system-independent features, such as target language model probabilities, are likely to be less effective. For the first time, we experiment with sentence and document-level QE methods on the output of both SMT and NMT, for a series of European languages. We show that the main challenge for QE of high-quality NMT lies in detecting errors in otherwise generally fluent text. We focus on the estimation of MT quality for news texts, a type of text where gisting is seen as a popular use of MT.

We start by discussing related work in Section 2. We present our light-weight hierarchical neural QE architecture in Section 3. We then introduce our experimental settings in Section 4. We provide the results of state of the art sentence-level methods in Section 4.1 and of the proposed document-level framework in Section 4.2.

2 Related work

QE targets the prediction of MT quality in the absence of reference translations. Given a set of training examples labelled for quality, their features extracted from source units and their corresponding MT units (black-box, system-independent), optionally complemented with features related to the translation process itself (glass-box, system-dependent), a QE model can be trained to predict a score for unseen MT units. Various types of units are possible: documents, paragraphs, sentences, words and phrases have been studied in previous work to different extents. Most work has focused on sentence or word-level prediction, which have clear application in dissemination scenarios, such as MT followed by post-editing. Document-level QE, which is applicable in assimilation (i.e. gisting) scenarios, has received much less attention.

Recently, neural methods have been successfully exploited to improve QE performance. The best-performing system at the WMT 2017 shared task on QE (Bojar et al., 2017) for the three levels of prediction (word, phrase and sentence), namely `POSTECH`, is purely neural and does not rely on feature engineering (Kim et al., 2017b). `POSTECH` is a modular architecture that revolves around an encoder-decoder Recurrent Neural Network (RNN) (so-called predictor), stacked with an bidirectional RNN (so-called estimator) that produces quality estimates. It predicts quality using the weights assigned by the predictor to the words we seek to evaluate, which are concatenated with the representations of their left and right one-word contexts, and then used to feed the estimator. To perform multi-level predictions, `POSTECH` relies on a multi-task learning approach which makes the quality estimates, for different levels of prediction, interdependent. The highest level of quality labels reported by the `POSTECH` system at WMT 2017 was sentence-level. Note that, to be effective, this architecture has to be pre-trained using a significant amount of parallel data, which leads to high training requirements in terms of time, processing power and dataset size.

Jhaveri et al. (2018) propose a series of neural models for sentence-level QE by simplifying and extending the `POSTECH` architecture (e.g. they skip the Predictor step, use convolutional encoders or an additional attention mechanism). We propose an alternative RNN-based simplification of `POSTECH`.

Another well performing system in the WMT shared QE task, Unbabel (Martins et al., 2017a; Martins et al., 2017b), also uses an encoder-decoder architecture with bidirectional RNN layers as part of its stacked architecture. It follows a hybrid approach where the input to this encoder-decoder is a pre-extracted feature set: pre-trained word and part-of-speech embeddings, word alignments and contexts. The system was designed for word and sentence QE.

State-of-the-art QE solutions specifically designed for document-level prediction employ traditional machine learning algorithms with non-linear kernels, such as Support Vector Machines (Cortes and Vapnik, 1995) and Gaussian Process (Rasmussen and Williams, 2005). Standard sets of document-level features are largely inspired by sentence-level features (Bojar et al., 2016a). Additionally, various discourse and neural-based features have been explored (Scarton et al., 2016).

Document-level QE is traditionally framed as averaging over sentence-level QE (Scarton et al., 2016). Sentence-level architectures consider each sentence separately; at the document level the entirety of

sentences in the document and the importance of each of these sentences should be taken into account. While the first problem can be addressed by, for instance, merging all sentences in a document and reusing a sentence-level QE system, the second problem requires considering every sentence separately yet as a part of the document, which requires a different QE architecture.

In this work we take advantage of the ability of neural networks to capture hierarchical structures, and propose a neural framework able to generalize over any sentence-level QE approach to produce document-level QE models. We test the framework using the state of the art neural sentence-level QE approach of Kim et al. (2017b), and a low-cost neural approach employing simple encoders, which we propose.

To our knowledge, the only attempt to shed some light on QE for NMT output is that by Rikters and Fishel (2017). They use attention mechanism distributions as an indicator the confidence of the neural decoder on its output at word-level. The hypothesis is that "good" translations can be characterized by strongly focused attention connections. However, this internal information has not been proved to map directly into translation quality: a very weak correlation with human judgements in a small-scale assessment was reported. Therefore, this is the first time that experiments are performed with fully fledged, MT system-independent QE models for NMT.

3 A neural-based architecture for QE

Our framework performs multi-level translation quality prediction, which has been shown to be successful in both traditional feature-engineered QE frameworks, such as QuEst++ (Specia et al., 2015), and neural QE architectures (Kim et al., 2017a; Martins et al., 2017a). In such architectures, the representations at a given level rely on representations from more fine-grained levels (i.e. sentences for document, and words for sentence).

This is motivated by the nature of the task at hand: a document that is composed of high quality sentences is likely to have high quality as well. However, simply aggregating sentence-level predictions is not a good strategy, as a document needs to be cohesive and coherent as a whole, i.e. sentences cannot be considered completely in isolation, and thus the need of a multi-level architecture that is trained jointly arises. Another important feature of document-level prediction is that certain parts of a document may be more important than others, such sentences containing keywords in a news article, versus sentences containing background information. Therefore one should also attempt to assess whether those sentences in particular are translated accurately. We do so by using different sentence-level weighting schemes for labelling documents, and by relying on an attention mechanism.

In what follows, we will first present the two sentence-level architectures we employ to then introduce our document-level framework.

3.1 Sentence-level architectures

The encoder-decoder approach (Sutskever et al., 2014; Bahdanau et al., 2015) provides a general architecture for sequence-to-sequence prediction problems. This approach has become very popular in many applications where inputs and outputs are sequential, such as MT. In this approach, an input sequence is encoded into an internal representation, and then an output sequence is generated left to right from this representation. Current best practices implement encoder-decoder approaches using RNNs, which handle inputs as a sequence (here a sequence of words), while taking previous words into account. We consider two different architectures, both based on the RNN encoder-decoder approach:

POSTECH We reimplement[1] the neural-based architecture for multi-level prediction by Kim et al. (2017a; Kim et al. (2017b), the best performing system at the WMT 2017 shared task on QE. This architecture is a two-stage end-to-end stacked neural QE model that combines (a) a *predictor* step, an encoder-decoder RNN model to predict words based on their context representations; and (b) an *estimator* step, a bidirectional RNN model to produce quality estimates for words, phrases and sentences based

[1]No code was originally available; our implementation is based on the NMT-Keras framework (Peris, 2017), and the Keras tool (Chollet and others, 2015). The code is publicly available online: `https://github.com/sheffieldnlp/deepQuest`.

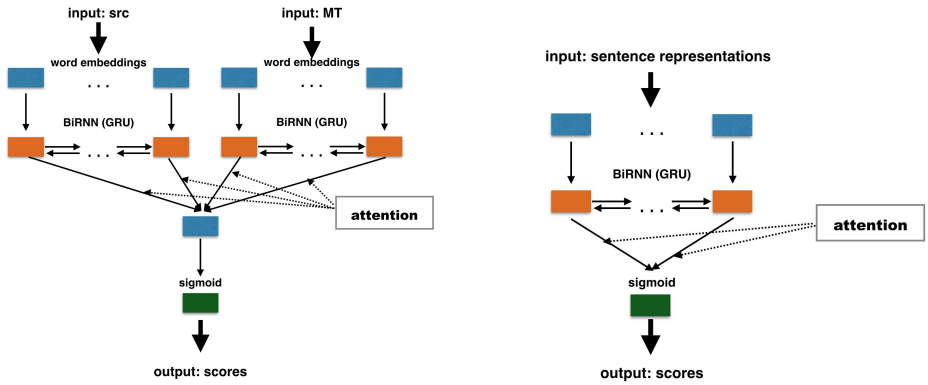


Figure 1: deepQuest framework. Left: the proposed sentence-level QE architecture: hidden states of source and MT encoders are concatenated and an attention mechanism over words is applied. Right: our document-level QE architecture: sentence representations are given to a bi-RNN and an attention mechanism over outputs of this RNN is applied to weight sentences according to their importance to the document.

on representations from the predictor. `POSTECH` requires extensive predictor pre-training to be effective, which means dependence on large parallel data and computational resources.

`BI-RNN` `BI-RNN` uses only two bi-directional RNNs (bi-RNN) as encoders to learn the representation of the (source, MT) sentence pair. A bi-RNN typically calculates a forward sequence of hidden states $(\overrightarrow{h}_1, ..., \overrightarrow{h}_J)$, and a backward sequence of hidden states $(\overleftarrow{h}_1, ..., \overleftarrow{h}_J)$. The hidden states $\overrightarrow{h}_j$ and $\overleftarrow{h}_j$ are concatenated to obtain the resulting representation h_j. In our approach, source and MT bi-RNNs are trained independently, as illustrated in Figure 1 (left side). The two representations are then combined via concatenation. However, sentence-level QE scores are not simple aggregations of word-level representations: they reflect some importance of words within a sentence. Thus, weights should be applied to those representations. Such weighting is provided by the attention mechanism. We apply the following attention function computing a normalized weight for each hidden state h_j:

$$\alpha_j = \frac{\exp(W_a h_j^\top)}{\sum_{k=1}^{J} \exp(W_a h_k^\top)}. \quad (1)$$

The resulting sentence vector is thus a weighted sum of word vectors: $v = \sum_{j=1}^{J} \alpha_j h_j$. A `sigmoid` output layer takes this vector as input and produces real-value quality scores.[2]

3.2 Document-level architecture

Our document-level framework uses a bi-RNN encoder. RNNs have been successfully used for document representation (Lin et al., 2015) and applied to a series of downstream tasks such as topic labeling, summarization, and question answering (Li et al., 2015; Yang et al., 2016).

The document-level quality predictor takes as input a set of sentence-level representations. The last hidden state of the decoder is the summary of an entire sequence. The sum, the maximum, or the average of hidden states for each sentence can then be provided to the output layer. Our assumption is that document-level QE scores are not a simple aggregations of sentence-level QE scores: they should reflect some notion of the importance of sentences within a document. To do so, we use an attention mechanism (Equation 1) to learn weights of different representations (different sentences). The weighted sum of the sentence representations is provided to the `sigmoid` output layer. This `sigmoid` output layer produces real-value predictions for a document, as illustrated in Figure 1 (right side).

[2]Note that Jhaveri et al. (2018) also propose an RNN-based QE architecture. However, following traditional NMT design, the attention mechanism is applied to the source encoder outputs, whereas we apply it to the outputs of both source and MT encoders.

4 Evaluation

In what follows, we first assess the performance of the sentence-level prediction approaches (Section 4.1), then move to the more coarse-grained level of predicting quality for documents (Section 4.2).

4.1 Sentence-level predictions

In this section we analyse the performance of our two sentence-level architectures against official baselines of the WMT 2017 QE shared task. We use the QE dataset in (Specia et al., 2017), which i) is a superset of the official dataset for the WMT 2017 QE task and for which the POSTECH architecture has been developed, allowing us to validate our reimplementation; ii) contains translations from neural MT systems. We also report results using the official evaluation metrics of the shared task: Pearson ρ correlation (primary) and Mean Average Error (MAE) (secondary).

Data The QE dataset contains 28,000 English-German (EN–DE) translations (IT domain) and 18,768 English-Latvian (EN–LV) translations (Life Science domain), produced with either a statistical MT (SMT), or an NMT system.

We randomly split the data for each language pair into training set (25K sentences for EN–DE, 16K for EN–LV), development set (1K sentences) and test (2K sentences) set.[3] In line with the WMT QE campaigns, data labelling was performed as described in (Bojar et al., 2017) using the TERCOM toolkit:[4] for sentence-level QE, with edit distance scores (HTER) used as labels.

To train POSTECH's predictor:[5] we used the Europarl corpus (Koehn, 2005) ($\approx$ 2M sentences; the version provided by Tiedemann (2012)) for EN–DE, and the parallel data of the WMT 2017 News translation task ($\approx$ 2M sentences) for EN–LV. Each experiment was run five times on the same split to estimate the stability of the model.

Baseline system: We reproduced the WMT 2017 baselines as described in (Bojar et al., 2017). For the extraction of the sentence-level features for EN–DE, we used the additional resources provided by the WMT 2017 QE shared task. For EN–LV, the corresponding resources were created using the data provided for the WMT 2017 News translation task, and the EMEA corpus (Tiedemann, 2009).

Implementation details: We implemented the sentence-level architectures using the Keras toolkit with Gated Recurrent Units (GRUs) (Cho et al., 2014) as RNNs, and the following hyperparameters: word embedding dimensionality = 300, vocabulary size = 30K, size of the hidden units of the encoder = 50. The model was trained to minimize the mean squared error loss using the Adadelta optimizer (Zeiler, 2012).

Results The results of our experiments are reported in Table 1. In general, different runs of the models do not lead to much variation in the performance. For EN–DE, a first observation is the major improvement in NMT quality compared to SMT (HTER = 0.09 vs. HTER = 0.24), which results in highly imbalanced NMT datasets (for EN–DE, $\approx$54% of all the sentences have 0 HTER vs. $\approx$13% for SMT). For EN–LV, the quality of NMT, limited by the amount of training data, is worse than SMT (HTER = 0.15 vs. HTER = 0.23).

These results show that the performance of baseline methods depends on the quality of translations rather than on the type of system that produced them. The baseline EN–DE methods achieve ρ scores that are 60% worse for NMT translations, as compared to SMT translation. For EN–LV, the behavior is the opposite: the performance is 45% better for NMT translations, as compared to SMT translations.

The performance of POSTECH is on average 40% higher for SMT than for NMT (e.g. for EN–LV, ρ=0.39 for SMT vs. ρ=0.24 for NMT). For EN–DE systems, this can be attributed to the data imbalance. For EN–LV systems, this could be because the neural QE system has difficulty to handle the many very

[3] As only part of the EN–DE SMT data was used for the WMT 2017 QE task, we could not use the task's official split.

[4] http://www.cs.umd.edu/~snover/tercom

[5] Hyperparameters size of hidden units of the word predictor = 500, word embedding dimensionality = 300, vocabulary size = 30K, QE vector size = 75.

long sentences produced by the EN–LV NMT ($\sigma^2 = 92$ of the distribution of the sentence length values for NMT vs. $\sigma^2 = 73$ for SMT).

For NMT the difference in variance of predicted HTER scores between language pairs is relatively high (as reflected in MAE differences between language pairs for neural methods, e.g., for POSTECH $\Delta = 0.084$), whereas for SMT this difference is lower (for POSTECH MAE $\Delta = 0.002$).

In general, without pre-training POSTECH does not perform better than the baseline methods (e.g. for EN–DE SMT, ρ=0.313 for the baseline vs. ρ=0.324 for POSTECH) and is systematically outperformed by BI-RNN (average $\Delta\rho$=0.06). This difference is statistically significant for both language pairs.[6] We believe that BI-RNN is able to better capture the fluency of NMT by encoding it directly as a sequence rather than assessing it word for word as POSTECH.

	EN–DE		**EN–LV**	
model	ρ	MAE	ρ	MAE
Baseline				
SMT	0.313 ±0.0	0.147 ±0.0	0.100 ±0.0	0.056 ±0.0
NMT	0.130 ±0.0	0.171 ±0.0	**0.318** ±0.0	0.070 ±0.0
POSTECH (no pre-training)				
SMT	0.324 ±0.015	0.146 ±0.002	0.294 ±0.015	0.136 ±0.002
NMT	0.153 ±0.023	0.103 ±0.001	0.240 ±0.010	0.205 ±0.008
POSTECH				
SMT	**0.481**±0.009	0.131 ±0.002	**0.390** ±0.016	0.129±0.005
NMT	**0.318** ±0.014	0.092 ±0.002	0.240 ±0.004	0.176 ±0.004
BI-RNN				
SMT	0.363 ±0.010	0.142 ±0.002	0.357 ±0.004	0.133 ±0.004
NMT	0.311±0.004	0.090 ±0.003	0.231 ±0.012	0.183 ±0.004

Table 1: Performance on sentence-level predictions for the baseline with QuEst, the POSTECH architecture and our BI-RNN architecture for EN–DE and EN–LV (average and error margins over five runs). We highlight the best performing systems for a dataset.

4.2 Document-level predictions

As introduced above, our framework relies on representations at sentence level to produce its predictions at document level. Therefore, we experiment with sentence-level representations from either the POSTECH or our BI-RNN predictor.

The document-level labels we predict are variants of BLEU (Papineni et al., 2002): (i) document-level BLEU,[7] (ii) the weighted average of sentence-level BLEU for all sentences in the document, where the weights correspond to the reference lengths:

$$\text{wBLEU}_d = \frac{\sum_{i=1}^{D} \text{len}(R_i)\text{BLEU}_i}{\sum_{i=1}^{D} \text{len}(R_i)},$$

where BLEU_i is the BLEU score of sentence i, and D is the size of the document in sentences,[8] and (iii), a variant of WBLEU by weighting each sentence by its TFIDF score computed with regard to its aligned reference (TBLEU). The numerator in the WBLEU equation is therefore replaced by: $\sum_{i=1}^{D} \text{TFIDF}_i\text{BLEU}_i$. Here, for each news document, we learn a TFIDF model on its reference in the target language, and compute the TFIDF score for each translated sentence, based on that model.

[6] We performed Kolmogorov–Smirnov test for not normally distributed data.

[7] We compute BLEU scores with the NLP toolkit NLTK (Bird and Loper, 2004). For scoring documents, we used the corpus_bleu() function. For sentence-level scores we used the sentence_bleu() function with smoothing method 7.

[8] According to Chen and Cherry (2014), WBLEU achieves a better correlation with human judgement than the original IBM corpus-level BLEU.

Following Turchi et al. (2012), our intuition is that the document-level score should reflect the overall translation quality at sentence level, weighted by how important each individual sentence (important sentences have important words) is in that document.

Data: We gathered all submissions at the WMT News shared tasks for various years. This is a task where each participating system is required to translate a set of news documents.[9] This results in a large set of language pairs, as well as a wide range of different translation quality levels. We collected system submissions from WMT 2008 to 2017, for four language pairs: German-English (DE–EN, 14,640 documents for 2008-2017, excluding 2010),[10] and English–Spanish (EN–ES, 6,733 documents for 2008-2013, excluding 2010[10]), English–French (EN–FR, 11,537 documents for 2008-2014) and English-Russian (EN–RU, 6,996 documents for 2013-2017). For each language pair, we consider either the full set of system submissions, or a filtered version of it (FILT), composed by only both the best and the worst performing systems for each year. The filtering was done based on the overall BLEU score achieved by each system, as reported on `matrix.statmt.org`. Our intuition is that by considering only the extreme quality levels we would make our data, while smaller, easier to discriminate.[11] We note that for all language pairs, MT systems include a variety of approaches, from rules-based MT, to SMT, hybrid approaches, and – from 2016 – NMT approaches. The filtered variants include at least one NMT approach for the language pairs in 2016/2017 (i.e. DE–EN and EN–RU). To support the reproduction of our work, as well as the development of new models for document-level QE, we release this dataset.[12]

Taking the heterogeneity of the data into account (composed of outputs of different systems and runs), to evaluate our models we perform 5-fold cross validation. For each language pair we shuffle the data, and for each fold we split it per year into train, development and test sets, as follows: for the FILT dataset, 10% of the documents per year were randomly selected for the development set, an another 10% for the test set; for ALL, the train data in the quantity equal to the FILT train data was selected randomly (development and test data were fixed, since ALL contains FILT). We present the averaged results. and use Kolmogorov–Smirnov test for not normally distributed data to test significance. As an example, statistics on one of the splits are shown in Table 2. Note that to avoid computation precision issues, we multiply TBLEU scores by 10.

As `POSTECH` training is very expensive, for the contrastive experiments we use only DE–EN and EN–ES. For the experiments with `BI-RNN`, we use all language pairs.

Baseline system: We reproduced the WMT 2016 document-level baseline as described in (Bojar et al., 2016b). We extract 17 black-box features using `QuEst++` (Specia et al., 2015) and train a document-level QE system using the Support Vector Regression (SVR) algorithm available in `scikit-learn` (Pedregosa et al., 2011). The language resources were created using the News Commentary and Europarl corpora as provided by WMT campaigns for the corresponding languages ($\approx$ 2M lines per language pair). These corpora were also used to train `POSTECH` predictors.[13]

For `BI-RNN`, we followed the implementation details as described in Section 4.1. To optimize the usage of computational resources, in each experiment we fixed the size of a document to the upper quartile of the distribution of document length values (in sentences). Shorter documents were extended with dummy sentences to fit to this length, which is a common practice in the field (Hewlett et al., 2017).

Results Results of our experiments are reported in Tables 3 (baseline) and 4 (neural approaches). For both `POSTECH` and `BI-RNN`, we use only the last hidden states of the document decoder (`Last`) – a configuration that has been chosen empirically as the best performing among the configurations without attention, or the vector sum weighted by the attention mechanism (`Att`) as input to the output layer.

[9] We considered using the document-level QE dataset in (Graham et al., 2017), however, the small number of documents (62) and language pairs (only one) made this resource less appealing for this work.

[10] Individual submissions are not available but system combinations only

[11] In (FILT) we have also discarded submissions for years 2008 and 2009, since official system-level scores are not available.

[12] `https://github.com/fredblain/docQE`

[13] `http://www.statmt.org/wmt18/translation-task.html;http://www.statmt.org/wmt13/translation-task.html`

set	# docs	av # sent	BLEU	wBLEU	tBLEU
DE–EN					
FILT	1147	26	0.529	0.316	0.457
ALL	1147	26	0.530	0.316	0.461
dev	140	26	0.527	0.319	0.454
test	140	26	0.521	0.314	0.459
EN–ES					
FILT	420	35	0.516	0.324	0.423
ALL	420	34	0.529	0.324	0.420
dev	51	34	0.504	0.323	0.424
test	51	34	0.533	0.324	0.426
EN–FR					
FILT	894	26	0.442	0.318	0.445
ALL	894	27	0.500	0.320	0.446
dev	109	27	0.464	0.320	0.454
test	109	24	0.475	0.320	0.440
EN–RU					
FILT	1052	22	0.591	0.329	0.561
ALL	1052	23	0.587	0.329	0.562
dev	130	22	0.585	0.330	0.561
test	130	24	0.581	0.330	0.551

Table 2: Statistics of the document-level dataset gathered from all submissions at the WMT News translation shared task. The first column presents the dataset considered, while second and third columns report the number of news documents in a dataset and the average number of sentences per document. The last three columns report respectively the average BLEU, average weighted BLEU (wBLEU) and the average variant of weighted BLEU (tBLEU) we propose in that set.

	DE–EN		**EN–ES**	
	ρ	MAE	ρ	MAE
	FILT			
BLEU	**0.065**	0.477	0.024	0.064
wBLEU	**0.177**	0.010	0.032	0.008
tBLEU	**0.043**	0.045	0.046	0.047
	ALL			
BLEU	0.044	0.973	**0.143**	0.063
wBLEU	0.033	0.010	**0.051**	0.007
tBLEU	0.019	0.046	**0.050**	0.050

Table 3: Baseline document-level score prediction results for DE–EN and EN-ES. We highlight the best performing systems for a dataset.

A first observation is that the performance of our neural approach varies significantly for different quality labels: the best performance is systematically observed for tBLEU, the worst – for BLEU (e.g. for DE–EN neural models, ρ=0.69 vs. ρ=0.39, on average respectively). This is not true for the baselines where the best performance is observed for other scores depending on the training data used. We attribute the high prediction performance for tBLEU to the fact that our architecture builds document-level representation from sentence-level representations, which in turn depend on word representations. The tBLEU reflects this hierarchy in the most consistent way as those document-level scores depend directly on semantic importance of words they contain. BLEU, on the other hand, depends on n-gram translation quality. MAE is in general the lowest for wBLEU with the lowest variance.

	DE–EN				EN–ES			
score	Last		Att		Last		Att	
	ρ	MAE	ρ	MAE	ρ	MAE	ρ	MAE
	FILT							
	POSTECH							
BLEU	0.122	0.064	0.170	0.060	0.472	0.078	0.548	0.070
WBLEU	0.330	0.010	0.511	0.007	0.317	0.015	0.300	0.032
TBLEU	0.632	0.035	0.744	0.088	0.739	0.030	0.854	0.020
	BI-RNN							
BLEU	0.213	0.056	0.157	0.050	0.487	0.084	**0.568**	0.070
WBLEU	0.413	0.008	0.590	0.007	0.512	0.072	0.407	0.025
TBLEU	0.770	0.031	0.814	0.029	0.898	0.020	0.903	0.020
	ALL							
	POSTECH							
BLEU	0.306	4.735	0.344	4.730	0.476	0.075	0.365	0.074
WBLEU	0.536	0.007	0.544	0.007	0.491	0.008	0.345	0.028
TBLEU	0.742	0.030	0.807	0.027	0.820	0.025	0.895	0.019
	BI-RNN							
BLEU	0.317	4.724	**0.363**	3.816	0.471	0.073	0.439	0.191
WBLEU	**0.660**	0.007	0.650	0.006	0.617	0.012	**0.655**	0.016
TBLEU	0.854	0.024	**0.889**	0.022	0.927	0.017	**0.941**	0.017

	EN–FR				EN–RU			
score	Last		Att		Last		Att	
	ρ	MAE	ρ	MAE	ρ	MAE	ρ	MAE
	FILT							
	BI-RNN							
BLEU	0.517	0.125	0.687	0.103	0.379	0.087	0.377	0.110
WBLEU	0.425	0.010	**0.504**	0.009	0.648	0.006	0.575	0.005
TBLEU	0.827	0.031	0.815	0.032	0.826	0.034	0.849	0.036
	ALL							
	BI-RNN							
BLEU	0.559	0.103	**0.723**	0.086	0.402	0.087	**0.418**	0.090
WBLEU	0.469	0.008	0.426	0.008	**0.726**	0.005	0.573	0.006
TBLEU	0.838	0.023	**0.844**	0.022	0.866	0.029	**0.876**	0.028

Table 4: Document-level score prediction results for neural approaches for DE–EN, EN–ES, EN–FR, and EN–RU. Last refers to the results after we take the last hidden state of the document-level encoder as input to the output layer; Att – with an attention mechanism. We highlight the best performing systems for a dataset.

The baseline yields poor performance (e.g., for BLEU ρ=0.07 on average across configurations), whereas BI-RNN systematically outperforms POSTECH for all three prediction tasks and across configurations (e.g., for DE–EN $\Delta\rho$=0.08). For DE–EN, this difference is statistically significant, while for EN–ES it is not. We believe this can be explained by lower stability of classifiers trained on the smaller EN–ES dataset.

The best performance improvement is observed for WBLEU, which we believe is because BI-RNN is less "focused" on word-level predictions (e.g., for DE–EN $\Delta\rho$=0.10). Additionally, BI-RNN is about 40 times faster to train than POSTECH.[14]

[14] BI-RNN takes around 20 minutes to train on a 12G GeForce TITAN X NVIDIA GPU with batch size = 10. Pre-training a POSTECH model in the same conditions with around 2M lines of parallel data takes around 12 hours plus the training of the

The contribution of the attention mechanism also depends on the type of quality label, but it is not statistically significant. It can be particularly beneficial for BLEU (for instance, for BI-RNN DE–EN across configurations, $\Delta\rho$=0.10 on average), but particularly harmful for WBLEU (for instance, for BI-RNN EN–RU, a decrease of ρ=0.12 is observed). This can be explained by the influence of variable reference lengths and hence the difficulty to find optimal weights.

The training data filtering procedure is not beneficial for the performance. This procedure is particularly harmful for DE–EN BLEU and WBLEU scores (for BI-RNN, average $\Delta\rho$=0.11), which is the most well represented language pair across WMT years. Thus, it may be the case that the random ALL selection contains more useful data.

Pre-training the predictor for POSTECH is essential for this architecture; for EN–ES BLEU Last, for example, a decrease of up to $\Delta\rho$=0.3 is observed without pre-training.

As for the difficulty of prediction for different language pairs: DE–EN and EN–RU BLEU prediction seems to be the most challenging (for BI-RNN, on average ρ=0.26 and ρ=0.40, respectively). This could be explained by the traditionally lower MT quality for those systems, involving significant word order differences for these language pairs.

5 Conclusions

We have proposed a new approach for neural-based document-level QE that is able to generalize any neural sentence-level architecture to the level of documents. This approach is part of our new framework for QE, named deepQuest, that reimplements the state of the art neural-based architecture to date for sentence-level quality prediction – the POSTECH approach – as well as our light-weight neural architecture relying on bi-directional RNNs. Our experiments have shown that latter outperforms POSTECH when used to predict document-level quality estimates for a range of quality scores and is 40 times faster to train. We also have reported a study of the performance of state of the art QE approaches on NMT output. Neural QE solutions are more efficient for imbalanced QE data, especially high-quality NMT. To our knowledge, this is the first time that results of QE on a large range of NMT data are reported.

In the future, we plan to reproduce our study for other document-level QE scores and text types. We also plan to adapt our light-weight neural architecture for other QE levels (e.g., phrase, paragraph levels). To support future work on these and other directions, we have made deepQuest open-source and freely available: https://github.com/sheffieldnlp/deepQuest.

Acknowledgements

The development of deepQuest received funding from the European Association for Machine Translation and the Amazon Academic Research Awards program. The first author worked on this paper during a research stay at the University of Sheffield.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Steven Bird and Edward Loper. 2004. NLTK: the natural language toolkit. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics: Interactive poster and demonstration sessions (ACL)*, page 31.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence estimation for machine translation. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58.

document-level system on average 2 additional hours. BI-RNN can also be trained on a CPU in about 3 hours.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016a. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation (WMT)*, pages 131–198.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016b. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation (WMT)*, pages 131–198.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation (WMT)*, pages 169–214.

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level BLEU. In *Proceedings of the Ninth Workshop on Statistical Machine Translation (WMT)*, pages 362–367.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

François Chollet et al. 2015. Keras. `https://github.com/fchollet/keras`.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

Yvette Graham, Qingsong Ma, Timothy Baldwin, Qun Liu, Carla Parra, and Carolina Scarton. 2017. Improving evaluation of document-level machine translation quality estimation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 356–361, Valencia, Spain, April. Association for Computational Linguistics.

Daniel Hewlett, Llion Jones, Alexandre Lacoste, and izzeddin gur. 2017. Accurate supervised and semi-supervised machine reading for long documents. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2011–2020.

Nisarg Jhaveri, Manish Gupta, and Vasudeva Varman. 2018. Translation quality estimation for indian languages. In *Proceedings of th 21st International Conference of the European Association for Machine Translation (EAMT)*.

Hyun Kim, Hun-Young Jung, Hongseok Kwon, Jong-Hyeok Lee, and Seung-Hoon Na. 2017a. Predictor-Estimator: Neural quality estimation based on target word prediction for machine translation. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 17(1):3:1–3:22, September.

Hyun Kim, Jong-Hyeok Lee, and Seung-Hoon Na. 2017b. Predictor-estimator using multilevel task learning with stack propagation for neural quality estimation. In *Proceedings of the Second Conference on Machine Translation (WMT)*, pages 562–568, September.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of the X Machine Translation Summit*, pages 79–86.

Jiwei Li, Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 1106–1115.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 899–907.

André Martins, Marcin Junczys-Dowmunt, Fabio Kepler, Ramón Astudillo, Chris Hokamp, and Roman Grundkiewicz. 2017a. Pushing the limits of translation quality estimation. *Transactions of the Association for Computational Linguistics*, 5:205–218.

André F. T. Martins, Fabio Kepler, and Jose Monteiro. 2017b. Unbabel's participation in the wmt17 translation quality estimation shared task. In *Proceedings of the Second Conference on Machine Translation (WMT)*, pages 569–574, September.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318.

Raj Nath Patel and Sasikumar M. 2016. Translation quality estimation using recurrent neural network. In *Proceedings of the First Conference on Machine Translation (WMT)*, pages 819–824, August.

Fabian Pedregosa, Gaël Varoquaux, Vincent Michel Alexandre Gramfort, Bertrand Thirion, Olivier Grisel, Peter Prettenhofer Mathieu Blondel, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Álvaro Peris. 2017. NMT-Keras. `https://github.com/lvapeab/nmt-keras`. GitHub repository.

Carl Edward Rasmussen and Christopher K. I. Williams. 2005. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. MIT Press.

Matīss Rikters and Mark Fishel. 2017. Confidence through attention. In *Proceedings of the 16th Machine Translation Summit (MT Summit XVI)*, pages 299–312.

Carolina Scarton, Daniel Beck, Kashif Shah, Karin Sim Smith, and Lucia Specia. 2016. Word embeddings and discourse information for quality estimation. In *Proceedings of the First Conference on Machine Translation (WMT)*, pages 831–837.

Lucia Specia, Marco Turchi, Nicola Cancedda, Marc Dymetman, and Nello Cristianini. 2009. Estimating the sentence-level quality of machine translation systems. In *Proceedings of 13th Annual Conference of the European Association for Machine Translation (EAMT)*, pages 28–37.

Lucia Specia, Gustavo Paetzold, and Carolina Scarton. 2015. Multi-level translation quality prediction with QuEst++. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*, pages 115–120.

Lucia Specia, Kim Harris, Frédéric Blain, Aljoscha Burchardt, Viviven Mackentanz, Inguna Skadiņa, Matteo Negri, and Marco Turchi. 2017. Translation quality and productivity: A study on rich morphology languages. In *Proceedings of the 16th Machine Translation Summit (MT Summit XVI)*, pages 282–299.

Ilya Sutskever, Oriol Vinyals, and Quoc V. V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Jörg Tiedemann. 2009. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In *Recent Advances in Natural Language Processing*, volume V, pages 237–248.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*.

Marco Turchi, Lucia Specia, and Josef Steinberger. 2012. Relevance ranking for translated texts. In *Proceedings of the 16th Annual Conference of the European Association for Machine Translation (EAMT)*, pages 153–160.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 1480–1489.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Butterfly Effects in Frame Semantic Parsing: impact of data processing on model ranking

Alexandre Kabbach
Department of Linguistics
University of Geneva

Corentin Ribeyre
Etermind

Aurélie Herbelot
Center for Mind/Brain Sciences &
Dept. of Information Engineering
and Computer Science
University of Trento

{firstname.lastname}@{unige.ch;etermind.com;unitn.it}

Abstract

Knowing the state-of-the-art for a particular task is an essential component of any computational linguistics investigation. But can we be truly confident that the current state-of-the-art is indeed the best performing model? In this paper, we study the case of *frame semantic parsing*, a well-established task with multiple shared datasets. We show that in spite of all the care taken to provide a standard evaluation resource, small variations in data processing can have dramatic consequences for ranking parser performance. This leads us to propose an open-source standardized processing pipeline, which can be shared and reused for robust model comparison.

Title and Abstract in French

Effets papillon en analyse automatique de cadres sémantiques:
impact du traitement des données sur la hiérarchie des modèles

Connaître l'état-de-l'art pour une tâche donnée est un élément essentiel de toute entreprise de linguistique computationnelle. Peut-on néanmoins s'assurer que l'état-de-l'art connu est bel et bien le meilleur des modèles ? Dans ces travaux nous nous intéressons au cas de l'analyse automatique de cadres sémantiques, une tâche bien établie disposant de multiples jeux de données partagés. Nous montrons qu'en dépit du soin porté à la standardisation du processus d'évaluation, de faibles variations dans le prétraitement des données peuvent conduire à une remise en question du classement final des analyseurs. Cette constatation nous conduit à proposer une plateforme libre de traitement standardisée permettant d'obtenir des données de comparaisons fiables entre les performances des différents analyseurs.

1 Introduction

A typical contribution to Computational Linguistics research is a new *model* for a specific task, which improves performance on a known dataset. The proposed model, whilst being the object of focus in the associated publication, normally relies on some potentially extensive pre- and post-processing of data which may only be briefly reported. As shown in the seminal work by (Fokkens et al., 2013), small alterations to such processing can result in dramatic changes in model performance. In this paper, we pick up on this point and make a case for harmonizing evaluations by sharing not just datasets but entire experimental pipelines. We illustrate our point by focusing on recent frame semantic parsing literature.

Frame semantic parsing is the task of automatically extracting semantic structures in text following the theory of Frame Semantics (Fillmore, 1982) and the framework of FrameNet (Baker et al., 1998, hereafter FN). Formally defined in the SemEval 2007 shared task 19 (Baker et al., 2007), it consists of three separate subtasks: (1) *target identification*: the task of identifying all frame evoking words in a given sentence; (2) *frame identification*: the task of identifying all frames of pre-identified targets in a given sentence; and (3) *argument identification*: the task of identifying all frame-specific frame

Proceedings of the 27th International Conference on Computational Linguistics, pages 3158–3169
Santa Fe, New Mexico, USA, August 20-26, 2018.

element spans and labels for pre-identified targets in a given sentence. The following sentence provides an example of FN annotation for the predicate *splash*.v evoking the `Cause_fluidic_motion` frame, and its corresponding FLUID, SOURCE and GOAL arguments:

(1) $[\text{Wine}]_{\text{Fluid}}$ was **splashed** $[\text{from jug}]_{\text{Source}}$ $[\text{to cup}]_{\text{Goal}}$ and often drained in one loud gulp.

Thanks to the shared task, the FN community has produced valuable datasets to evaluate models against. Still, as we will show in this paper, the actual evaluation settings used by various research groups exhibit differences which can in some cases have dramatic consequences for the reported results. In particular, all available research on frame semantic parsing relies on some form of pre- and post-processing of FN XML data. Post-processing usually involves converting FN train, dev and test XML splits to standard CONLL-based formats while removing erroneous or problematic annotation. Preprocessing involves part-of-speech tagging *ad minima*, and often lemmatization and dependency parsing as well. We will see that those pre- and post-processing pipelines turn out to be rather inconsistent across published investigations (see §2.2), preventing fair comparison between models. In fact, we demonstrate that when using a cleaned up and standardized pipeline, the ranking of existing parsers can change quite substantially.

The deviations we observe across experimental setups cover various aspects of pre- and post-processing. First, inconsistencies in the post-processing pipeline lead parsers to be actually evaluated on slightly different test sets across studies. Moreover, those test sets contain duplicate sentences and annotations – leading parsers to be scored multiple times on the same data – as well as overlapping data between train and test sets – thereby artificially improving overall scores. Second, inconsistencies in the pre-processing pipeline prevent fair comparison across models by conflating the contribution of the pre-processing toolkit with that of the statistical model itself. We further note that all recent work on frame semantic parsing rely on the FN 1.5 dataset, while the FN 1.7 dataset, available since 2016, contains nearly 20% more gold annotated data, allowing for larger train and test sets and hence more robust baselines.

The goal of this paper is to highlight the impact of small deviations in experimental settings on model performance. Our contributions are fourfold: we (1) replicate past results of frame semantic parsing on a single robust pre- and post-processing pipeline; (2) quantify the impact of the pipeline on model performance; (3) analyze the robustness of past results when tested on a normalized FN 1.7 setup; and (4) provide robust baselines on both FN 1.5 and FN 1.7 experimental setups for future research.

We focus specifically on replicating frame identification results of the SIMPLEFRAMEID system of Hartmann et al. (2017), and argument identification results of the SEMAFOR system of Das et al. (2014), as modified by Kshirsagar et al. (2015), and the OPEN-SESAME system of Swayamdipta et al. (2017). We exclude the FRAMAT system of Roth and Lapata (2015) for brevity, as it uses a different evaluation script on argument identification given gold frames, and we ignore all other contributions (Hermann et al., 2014; Täckström et al., 2015; FitzGerald et al., 2015; Roth, 2016; Yang and Mitchell, 2017) which do not provide open source systems.[1]

Finally, rather than releasing a fixed preprocessed FrameNet dataset like Bauer et al. (2012), we propose an application for testing various frame semantic parsers in custom experimental setups. Our application allows converting FrameNet XML data to various standard NLP formats (such as, e.g., BIO or CoNLL-X) relying on a common architecture which handles data processing side effects observed throughout this work. Preprocessing pipelines can thereby be easily modified while guaranteeing the consistency and robustness of the experimental setups across parsers.

The structure of the paper is as follows: in Section 2, we introduce all past research on frame semantic parsing replicated throughout this work, the statistical models used, as well as their experimental and evaluation setups. In Section 3 we introduce our own experimental setup designed to overcome the limitations observed in past research. In Section 4, we provide both frame and argument identification scores for past models trained on our own experimental setup, on both FN 1.5 and 1.7 datasets. In Section 5 we discuss the impact of pre-processing on argument identification, and in Section 6 and Section 7 we provide new robust baselines on both FN 1.5 and 1.7 datasets and conclude.

[1] For (Roth, 2016), although decoding and models are open source, training software is not

2 Existing frame semantic parsing pipelines

2.1 Models

SIMPLEFRAMEID The *frame identification* model of SIMPLEFRAMEID (Hartmann et al., 2017) is based on distributed representations of predicates and their (syntactic) context. The overall model relies on a disambiguation classifier which learns weights for all frames in the lexicon, given a vector space model providing dense vector representations for the predicate and its context. The context of the predicate can include all words in the sentence (SENTBOW model), or be restricted to direct dependents of the predicate (DEPBOW model). The overall classifier can rely on two distinct classification methods: a two-layer neural network (NN) or a WSABIE-based method following the approach of Hermann et al. (2014), consisting in mapping input representations and frame representations to a common latent space using the WSABIE algorithm (Weston et al., 2011). In the standard decoding setup, the best scoring frame is selected among all possible frames of a given predicate, as specified in the lexicon. If the predicate is not found in the lexicon, the best scoring frame is selected among all frames, while if the predicate is unambiguous and has a single possible frame, the frame is assigned directly. Hartmann et al. (2017) provide three evaluation metrics for all their models: *total*, which provides an accuracy score for all predicates in the test set; *ambig* which provides accuracy scores for ambiguous predicates only (predicates with more than one possible frames in the lexicon); and *no-lex*, accuracy scores while treating each predicate as *unknown* and scoring all frames in the lexicon.

SEMAFOR The *argument identification* model of SEMAFOR is a system originally proposed by Das et al. (2014) and modified by Kshirsagar et al. (2015). The model derives the set of argument spans via a rule-based algorithms over the syntactic context of a predicate word, and labels roles using a conditional log-linear model over spans for each role of each evoked frames, trained using maximum conditional log-likelihood. The original model makes extensive use of handcrafted features, ranging from lexical items combinations to syntactic dependency paths. To the original features are added the *exemplar* and *hierarchy* features (henceforth EX and H) proposed by Kshirsagar et al. (2015). EX incorporates FN exemplar data – lexicographic examples annotating a single predicate – to the original 'fulltext' train set, while H incorporates information regarding specific frame relations (Inheritance and SubFrame). Decoding is done using beam search which produces a set of k-best hypotheses of sets of span-role pairs. The approach enforces multiple constraints including the fact that a frame element label may be assigned to at most one span and that spans of overt arguments must not overlap.

OPEN-SESAME OPEN-SESAME (Swayamdipta et al., 2017) is another *argument identification* system based on the SegRNN model of Kong et al. (2015), which relies on a combination of bidirectional RNNs and semi-Markov CRF – with a slight modification to favor recall over precision – to learn embedded representations of targets, lexical units, frames, frame elements and spans. In its basic form, the OPEN-SESAME parser is syntax-free, although it may also incorporate syntactic cues such as dependency parses or phrase structures. For brevity, and to ease the replication process, we focus in this work on the syntax-free and dependency-based models only. Note that, throughout this work, we report results with single models and not ensemble models, and that, due to differences in initialization, variations of F_1 scores can be observed over our reported results, of up to 1 point of F_1 score on FN 1.5 setups and .5 points of F_1 score on FN 1.7 setups.

2.2 Variations in existing experimental settings

SIMPLEFRAMEID and SEMAFOR rely on the original DAS sets (Das and Smith, 2011; Das et al., 2014). OPEN-SESAME extracts its test set relying on the exact same list of FN fulltext as Das and Smith (2011), but via a different pre-processing pipeline. We found both test splits to contain duplicate sentences and annotationsets,[2] as well as at least one overlapping sentence and corresponding annotationsets between train and test splits.[3] The DAS test split contains 4457 annotationsets when the OPEN-SESAME test

[2] In FN, an annotationset is defined as one set of labels for a predicate and its corresponding arguments in a given sentence.

[3] The sentence in question being *Even if Iran possesses these biological agents, it faces a significant challenge in their weaponization and delivery* .

split contains 4428 annotationsets. Corresponding parsers are therefore not evaluated on the exact same set of gold data. The difference in the number of annotationsets is partially explained by the fact that OPEN-SESAME relies on the BIO tagging scheme, which does not support overlapping frame elements, to generate its gold data. Of the 4457 annotationsets of the DAS split, at least 162 were actually duplicates. Of the 2420 sentences listed in both test sets, only 982 contained annotation, and 107 of them were duplicates, making for an actual test set of 875 unique annotated sentences.

In (Das et al., 2014; Kshirsagar et al., 2015; Hartmann et al., 2017), data are tokenized with SED, part-of-speech tagged with MXPOST (Ratnaparkhi, 1996), lemmatized with WORDNET (Miller, 1995) and dependency-parsed with the MST parser (McDonald et al., 2006). In (Swayamdipta et al., 2017), data retain the original FN tokenization, are lemmatized with NLTK (Bird et al., 2009) and part-of-speech-tagged and dependency-parsed with SYNTAXNET (Andor et al., 2016). To incorporate exemplars into the training set, Kshirsagar et al. (2015) apply an extra layer of filtering by excluding annotationsets containing no overt frame element labels from the training data. As detailed in §5.3, this has a non-negligible impact on argument identification and we treat this filtering layer as an independent feature.

All the above systems generate their *lexicons* by scanning the *entirety* of FN data. This includes the predicate-to-frame map of SIMPLEFRAMEID, the frame-to-frame-element map of SEMAFOR and OPEN-SESAME, and the frame-to-frame and frame-element-to-frame-element relation maps of SEMAFOR when used with the *hierarchy* feature. Such an approach leads systems to have partial access to information which is only included in the test set, such as the number of frames available for an unknown predicate,[4] the list of frame elements of a given unknown frame, and the relation between an unknown frame and a known frame.

2.3 Evaluation software

Throughout this work we rely on two distinct evaluation scripts, motivated by considerations over the design of the original perl evaluation script of the SemEval shared task 19 on frame structure extraction. Our first script, called SEMEVAL, is the original SemEval evaluation script as-is, and is used to score argument identification given predicted frames. Our second script, called ARGSONLY, is a modified version of the evaluation script originally provided by Kshirsagar et al. (2015), which we use to score argument identification given gold frames. Both scripts provide global precision, recall and F_1 scores microaveraged across a concatenation of the test set.

Modifications to the original SEMEVAL script are motivated by its computing scores for both frames and arguments jointly, which dilutes contributions to argument identification in gold frames settings by systematically granting extra credits for gold frames. Unlike the evaluation script of Kshirsagar et al. (2015) however, our ARGSONLY evaluation does not skip the evaluation of frame-only annotation sets with no arguments, as it would not penalize parsers for over-predicting arguments.

3 Building a new harmonized experimental setup

3.1 Datasets and post-processing

We generate our train, dev and test splits following the list of fulltext documents provided by Swayamdipta et al. (2017). For both FN 1.5 and FN 1.7 datasets, dev and test splits are generated from the same list of FN fulltexts. To guarantee no duplicates and no overlap within and across datasets, we apply a strong filter based on an annotationset hash generated from the concatenated annotationset sentence text hash and the annotationset target hash. The sentence text hash is the sentence text lower-cased on stripped of all its whitespaces while the target hash is a concatenation between the target predicate string and the target indexes. To guarantee near-perfect matching between SEMEVAL-formatted test XML data and the original gold FN XML data, we directly convert FN XML data without relying on intermediate formats (e.g. CONLL or BIO formats), and retain all annotationsets except those with inconsistent labels.[5]

[4] I.e. a predicate in the test set not seen during training.

[5] We define inconsistent labels as those with inconsistent start/end indexes, e.g. when only one of the two indexes is missing. Note that when such annotationsets are found, we filter out the entire annotationset and not just the erroneous label, as we consider the entire annotationset as an indivisible piece of information

Table 1 details the number of sentences and annotationsets in both FN 1.5 and 1.7 setups. The 50% increase in test gold data between FN 1.5 and 1.7 guarantees more robust baselines when testing on the 1.7 setup.

	TRAIN 1.5 FT	TRAIN 1.5 FT+EX	TEST 1.5	TRAIN 1.7 FT	TRAIN 1.7 FT+EX	TEST 1.7
#sent	2,654	150,426	875	3,362	168,792	1,247
#anno	16,706	170,889	4,148	19,550	192,554	6,446

Table 1: Number of sentences and annotationsets in train and test splits for both FN 1.5 and FN 1.7 setups, with training data generated from fulltexts only (FT) or fulltexts and exemplars (FT+EX)

3.2 Pre-processing

In all our experiments we retain the original FN tokenization, while lemmatizing data with NLP4J (Choi, 2016). For part-of-speech tagging, we experiment with MXPOST (Ratnaparkhi, 1996) and NLP4J, and for dependency parsing with the MST (McDonald et al., 2006) and BIST (Kiperwasser and Goldberg, 2016) parsers. The BIST parser is used in both its graph-based (BMST) and transition-based (BARCH) variants. Both MST and BIST parsers are trained on sections 02-21 of the Penn TreeBank (Marcus et al., 1993). We choose MXPOST and MST for the purpose of replicating previous studies, and NLP4J and BIST for performance. At the time when this research started, state-of-the-art results on frame semantic parsing were reported by Roth (2016) who recommended the use of both NLP4J and BIST for pre-processing.

3.3 Frame semantic parsers

We fork the branch of SEMAFOR used in (Kshirsagar et al., 2015), as well as the SIMPLEFRAMEID and OPEN-SESAME master branches. For each parser, we re-implement lexicon creation methods in order to include only data seen during training (see §3.1). That is, we reproduce the predicate-frame map of SIMPLEFRAMEID, the frame-to-frame-element map of SEMAFOR and OPEN-SESAME, and the hierarchy feature-related relation maps of SEMAFOR. We keep all parsers and hyperparameters as detailed in (Hartmann et al., 2017), (Kshirsagar et al., 2015) and (Swayamdipta et al., 2017), and set the regularization parameter λ for SEMAFOR to $\lambda = 10^{-5}$.

3.4 Pipeline release

To ease replication and future work on frame semantic parsing, we release a single Python application called PYFN, which provides a set of Python models to process FN annotation. We thereby enforce data consistency across experiments, as data conversion[6] is processed via a single set of models. We create a larger standalone bundle including all forked parsers used as well as all datasets, resources and scripts necessary to replicate our experiments, and release it open source at `https://gitlab.com/akb89/pyfn`. Each experiment introduced in this work is labeled with a unique #id, to which corresponds a README file listing all necessary instructions for replicating the experiment, found under the *experiments* directory of the PYFN repository.

4 Replication of previous studies

The purpose of this section is to measure the contribution of each model introduced in §2.1 on our robust experimental setup (§3.1). Given that we use slightly different test sets and lexicons from the original studies, we focus on comparing the order of magnitude of each contribution and the resulting ranking of models, as well as analyzing the robustness of each contribution when tested on the FN 1.7 dataset.

4.1 Frame identification: the SIMPLEFRAMEID system

On frame identification, Tables 3 and 4 confirm one of the main results of Hartmann et al. (2017) presented in Table 2, namely, that the NN model outperforms the WSABIE model, and this in both FN 1.5

[6]E.g. to and from FN XML format, SEMEVAL XML format, CONLL-09 format, CONLL-X format, BIO tagging format, etc.

and FN 1.7 setups. Table 3 also confirms the hierarchy between models on the FN 1.5 setup, albeit not as clearly as originally reported, with differences between NN and WSABIE models closer to 2 points of accuracy rather than 3. However, Table 4 shows that the hierarchy between the NN + SENTBOW and NN + DEPBOW models are not robust across datasets, as the NN + DEPBOW model performs better than the NN + SENTBOW model on the FN 1.7 setup, by a margin of .6 accuracy point overall.[7] We conclude that the superiority of the NN + SENTBOW model cannot be generically ascertained, in line with original experiments reported by Hartmann et al. (2017).

Hartmann et al. (2017) FRAMENET 1.5 SETUP					
system	pos	dep	total	ambig	no-lex
WSB + SENTBOW	MXPOST	MST	84.5	67.6	72.0
WSB + DEPBOW	MXPOST	MST	85.7	69.9	71.2
NN + SENTBOW	MXPOST	MST	**87.6**	**73.8**	**77.5**
NN + DEPBOW	MXPOST	MST	87.5	73.6	76.5

Table 2: SIMPLEFRAMEID frame identification accuracy scores on DAS splits

(OUR) FRAMENET 1.5 SETUP				
xp	system	total	ambig	no-lex
#46	WSB + SENTBOW	81.2	66.8	72.9
#46	WSB + DEPBOW	81.4	69.1	74.3
#46	**NN + SENTBOW**	**83.1**	**73.0**	77.0
#46	NN + DEPBOW	82.9	72.7	**78.3**

Table 3: SIMPLEFRAMEID frame identification accuracy scores on our **FN 1.5** splits

(OUR) FRAMENET 1.7 SETUP				
xp	system	total	ambig	no-lex
#67	WSB + SENTBOW	80.9	64.9	73.2
#67	WSB + DEPBOW	81.7	67.1	72.5
#67	NN + SENTBOW	82.4	69.8	**77.0**
#67	**NN + DEPBOW**	**83.0**	**71.7**	76.1

Table 4: SIMPLEFRAMEID frame identification accuracy scores on our **FN 1.7** splits

4.2 Argument identification: OPEN-SESAME and SEMAFOR

On argument identification, we first try and replicate past results relying on our FN 1.5 and FN 1.7 splits, as well as a common pre-processing pipeline combining MXPOST and MST, in order to match as closely as possible the pipeline most widely used in past studies (see Table 5). We report results for the standard version of SEMAFOR and for all its adaptations introduced in (Kshirsagar et al., 2015), which include the hierarchy feature (H), the exemplar feature (EX) and the filtering feature (F).

Argument identification on FN 1.5				Gold Frames			Predicted Frames		
	POS	DEP		P	R	F_1	P	R	F_1
SEMAFOR	MXPOST	MST	(Das et al., 2014)	65.6	53.8	59.1	-	-	66.8
SEMAFOR H	MXPOST	MST	(Kshirsagar et al., 2015)	67.2	54.8	60.4	-	-	-
SEMAFOR EX + F	MXPOST	MST	(Kshirsagar et al., 2015)	66.0	58.2	61.9	-	-	-
SEMAFOR H + EX + F	MXPOST	MST	(Kshirsagar et al., 2015)	66.0	60.4	63.1	-	-	67.9
OPEN-SESAME	SYNTAXNET	-	(Swayamdipta et al., 2017)	64.7	**61.2**	62.9	68.0	**68.1**	68.0
OPEN-SESAME	SYNTAXNET	SYNTAXNET	(Swayamdipta et al., 2017)	**69.4**	60.5	**64.6**	**71.0**	67.8	**69.4**

Table 5: Argument identification scores given gold and predicted frames on DAS splits, as originally reported in (Kshirsagar et al., 2015; Swayamdipta et al., 2017). Frames are predicted with the proprietary system of Hermann et al. (2014)

[7] Note that decrease in absolute score was expected given that all models use a reduced lexicon compared to (Hartmann et al., 2017), although the unpredictability of performance on duplicates in the DAS test set could have compensated either way.

Table 6 shows that, contrary to what was previously reported, the OPEN-SESAME parser does not outperform the best version of SEMAFOR on the FN 1.5 splits. Although the dependency-based version of OPEN-SESAME does achieve comparable results than SEMAFOR, the syntax-free version is outperformed by a margin of .9 points of F_1 score. The contribution of each H, EX and F feature is confirmed across datasets, with similar order of magnitudes. Ranking of models changes however across datasets, and results on the FN 1.7 splits (Table 7) provide a ranking of models closer to the one reported in Table 5.

XP	FRAMENET 1.5 SETUP			P	R	F_1
#42	SEMAFOR	MXPOST	MST	59.1	54.3	56.6
#86	SEMAFOR H	MXPOST	MST	**60.3**	54.9	57.5
#75	SEMAFOR EX + F	MXPOST	MST	59.9	58.9	59.4
#89	**SEMAFOR H + EX + F**	MXPOST	MST	60.1	**60.6**	**60.4**
#43	OPEN-SESAME	MXPOST	-	59.7	59.4	59.5
#44	OPEN-SESAME	MXPOST	MST	59.8	59.8	59.8

Table 6: Argument identification scores given gold frames on our **FN 1.5** splits while pre-processing with MXPOST and MST

XP	FRAMENET 1.7 SETUP			P	R	F_1
#54	SEMAFOR	MXPOST	MST	61.2	53.5	57.1
#61	SEMAFOR H	MXPOST	MST	62.5	53.7	57.8
#97	SEMAFOR EX + F	MXPOST	MST	**64.8**	54.9	59.4
#63	SEMAFOR H + EX + F	MXPOST	MST	63.6	57.4	60.4
#53	OPEN-SESAME	MXPOST	-	62.9	58.4	60.6
#80	OPEN-SESAME	MXPOST	MST	63.2	**58.9**	**61.0**

Table 7: Argument identification scores given gold frames on our **FN 1.7** splits while pre-processing with MXPOST and MST

So while frame identification results proved rather consistent across setups in §4.1, argument identification results proved more erratic, with a clear mismatch of model hierarchy on the FN 1.5 setup, contradicting previous studies.

5 Impact of the preprocessing pipeline

In this section we report on the impact of different preprocessing pipelines on argument identification scores for both the SEMAFOR and OPEN-SESAME systems. Note that in §5.1 and §5.2, we compare OPEN-SESAME to the standard SEMAFOR model, without the H, EX or F features. However, similar trends can be observed for the feature-expanded versions of SEMAFOR (see §5.3).

5.1 Part-of-speech tagging

Table 8 shows a systematic improvement of F_1 score on the FN 1.5 splits when using NLP4J over MXPOST for part-of-speech tagging (compare with Table 6: #69 vs #42, #78 vs #43, and #79 vs #44). Results presented in Table 9 (compare with Table 7) tend to show that increase in performances with NLP4J is not robust across datasets (see notably #56 vs #54). However, further experiments, presented in §5.2 and §5.3 show that performance on argument identification is highly dependent on the *combination* of the part-of-speech tagger *and* the dependency parser used. In our case, the combination of NLP4J with the BIST dependency parser actually outperforms any MXPOST-based pre-processing pipeline.

XP	FRAMENET 1.5 SETUP			P	R	F_1
#69	SEMAFOR	NLP4J	MST	60.2	55.4	57.7
#78	OPEN-SESAME	NLP4J	-	59.8	60.2	60.0
#79	OPEN-SESAME	NLP4J	MST	61.2	60.2	60.7

Table 8: Argument identification scores given gold frames on our **FN 1.5** splits while pre-processing with NLP4J and MST

XP	FRAMENET 1.7 SETUP			P	R	F_1
#56	SEMAFOR	NLP4J	MST	59.1	53.4	56.1
#48	OPEN-SESAME	NLP4J	-	63.1	58.8	60.9
#55	OPEN-SESAME	NLP4J	MST	64.4	59.6	61.9

Table 9: Argument identification scores given gold frames on our **FN 1.7** splits while pre-processing with NLP4J and MST

5.2 Dependency-parsing

Table 10 and Table 11, show a near-systematic improvement of F_1 score for both SEMAFOR and OPEN-SESAME systems when using the BIST dependency parser over MST. On FN 1.5 , the contribution of the dependency parser itself to overall performance proves non-negligible, with an observed improvement ranging from .6 points of F_1 score for OPEN-SESAME between MST and BARCH (#79 and #83), to 4

points of F_1 score for SEMAFOR between MST and BARCH (#69 and #81). On certain FN 1.5 setups, the combination of pos tagger and dependency parser may even account, in order of magnitude, for more than half of the previously reported contribution of the OPEN-SESAME model over the SEMAFOR model (consider, e.g., #82 in Table 10 and #44 in Table 6 compared to the 1.5 F_1 score improvement reported in Table 5 between the best SEMAFOR model and the dependency-based OPEN-SESAME model).

XP	FRAMENET 1.5 SETUP			P	R	F_1
#69	SEMAFOR	NLP4J	MST	60.2	55.4	57.7
#70	SEMAFOR	NLP4J	BMST	67.5	56.4	61.4
#81	SEMAFOR	NLP4J	BARCH	**67.6**	**56.8**	**61.7**
#79	OPEN-SESAME	NLP4J	MST	61.2	60.2	60.7
#82	OPEN-SESAME	NLP4J	BMST	**64.1**	59.6	61.2
#83	OPEN-SESAME	NLP4J	BARCH	61.4	**61.3**	**61.3**

Table 10: Argument identification scores given gold frames on our **FN 1.5** splits while pre-processing with NLP4J and MST, BMST or BARCH

XP	FRAMENET 1.7 SETUP			P	R	F_1
#56	SEMAFOR	NLP4J	MST	59.1	53.4	56.1
#47	SEMAFOR	NLP4J	BMST	**64.8**	55.7	**59.9**
#84	SEMAFOR	NLP4J	BARCH	61.2	**55.8**	58.4
#55	OPEN-SESAME	NLP4J	MST	64.4	59.6	61.9
#57	OPEN-SESAME	NLP4J	BMST	**64.6**	59.3	61.8
#85	OPEN-SESAME	NLP4J	BARCH	63.6	**60.3**	**61.9**

Table 11: Argument identification scores given gold frames on our **FN 1.7** splits while pre-processing with NLP4J and MST, BMST or BARCH

5.3 Data filtering

As previously introduced in §2.2, Kshirsagar et al. (2015) apply a filtering layer on training data by removing all annotationsets with no overt role labels from the exemplar data. Such a filter may remove valid annotationsets where all arguments of the predicate are found to be *null instantiations* (Ruppenhofer et al., 2016) – which may correspond, e.g., to intransitive predicates, lexically licensed zero anaphora or imperative and passive constructions – but can also remove incompletely annotated annotationsets, where predicates have been disambiguated (frames have been assigned to specific targets), but frame element labels have yet to be specified. Such cases are much more frequent in exemplar than in fulltext data, justifying *a priori* the filtering layer of Kshirsagar et al. (2015) to prevent negatively biasing SEMAFOR.

Table 12 and Table 13 show that the filter feature has a systematic positive effect on argument identification performance, and that this result is robust across datasets and across pre-processing pipelines. Note that, in a configuration where EX and H features are combined, the addition of the F feature leads to a .5 F_1 (compare #89 and #87) to 1.4 F_1 (compare #63 and #62) increase in performance depending on the setup. This result suggests that the filtering layer applied by Kshirsagar et al. (2015) should be considered as a standalone feature, which contribution should be clearly separated from that of the exemplar feature, in order to better quantify the contribution of the exemplar feature itself.

XP	FRAMENET 1.5 SETUP			P	R	F_1
#42	SEMAFOR	MXPOST	MST	59.1	54.3	56.6
#45	SEMAFOR EX	MXPOST	MST	**63.0**	55.4	59.0
#75	SEMAFOR EX + F	MXPOST	MST	59.9	58.9	59.4
#87	SEMAFOR H + EX	MXPOST	MST	61.7	58.2	59.9
#89	SEMAFOR H + EX + F	MXPOST	MST	60.1	**60.6**	**60.4**
#70	SEMAFOR	NLP4J	BMST	**67.5**	56.4	61.4
#92	SEMAFOR EX	NLP4J	BMST	65.0	58.9	61.8
#93	SEMAFOR EX + F	NLP4J	BMST	65.0	60.0	62.4
#94	SEMAFOR H + EX	NLP4J	BMST	64.2	61.1	62.6
#95	SEMAFOR H + EX + F	NLP4J	BMST	62.7	**63.5**	**63.1**

Table 12: Argument identification scores given gold frames on our **FN 1.5** splits for various feature-augmented versions of SEMAFOR

XP	FRAMENET 1.7 SETUP			P	R	F_1
#54	SEMAFOR	MXPOST	MST	61.2	53.5	57.1
#88	SEMAFOR EX	MXPOST	MST	61.0	55.1	57.9
#97	SEMAFOR EX + F	MXPOST	MST	**64.8**	54.9	59.4
#62	SEMAFOR H + EX	MXPOST	MST	61.9	56.3	59.0
#63	SEMAFOR H + EX + F	MXPOST	MST	63.6	**57.4**	**60.4**
#47	SEMAFOR	NLP4J	BMST	64.8	55.7	59.9
#49	SEMAFOR EX	NLP4J	BMST	63.9	57.5	60.6
#73	SEMAFOR EX + F	NLP4J	BMST	64.3	59.4	61.8
#64	SEMAFOR H + EX	NLP4J	BMST	65.8	58.6	62.0
#65	SEMAFOR H + EX + F	NLP4J	BMST	**67.7**	**59.4**	**63.3**

Table 13: Argument identification scores given gold frames on our **FN 1.7** splits for various feature-augmented versions of SEMAFOR

Overall, we showed that divergence in the pre-processing pipeline can actually account for a significant part of the difference in performance previously observed across models, enforcing the need for standard pre-processing pipelines against which models can be fairly compared.

6 New baselines for FN 1.5 and FN 1.7

Building upon results of the previous sections, we introduce new baselines for both frame and argument identification on our FN 1.5 and FN 1.7 splits, relying on an updated preprocessing pipeline combining NLP4J for lemmatization and part-of-speech tagging, and the BMST variant of the BIST parser for dependency parsing. Although not necessarily the best pipeline observed in all setups, the NLP4J + BMST combination proved to be the best compromise between performance and speed, an important factor to take into account for facilitating future replication.

XP	FRAMENET 1.5 SETUP			P	R	F_1
#70	SEMAFOR	NLP4J	BMST	**67.5**	56.4	61.4
#95	SEMAFOR H + EX + F	NLP4J	BMST	62.7	**63.5**	**63.1**
#78	OPEN-SESAME	NLP4J	-	59.8	60.2	60.0
#82	OPEN-SESAME	NLP4J	BMST	64.1	59.6	61.2

Table 14: Argument identification baseline scores given gold frames on **FN 1.5** splits

XP	FRAMENET 1.7 SETUP			P	R	F_1
#47	SEMAFOR	NLP4J	BMST	64.8	55.7	59.9
#65	SEMAFOR H + EX + F	NLP4J	BMST	**67.7**	**59.4**	**63.3**
#48	OPEN-SESAME	NLP4J	-	63.1	58.8	60.9
#57	OPEN-SESAME	NLP4J	BMST	64.6	59.3	61.8

Table 15: Argument identification baseline scores given gold frames on **FN 1.7** splits

Table 14 and Table 15 confirm, in addition to Table 6 and Table 7, that the ranking of models reported in (Swayamdipta et al., 2017) is not robust across datasets and experimental setups. In the most robust setup (FN 1.7 with NLP4J and BMST reported in Table 15) SEMAFOR still outperforms both syntax-free and dependency-based versions of OPEN-SESAME, by a margin of at least 1.5 points of F_1 score.

Surprisingly however, the ranking of models is significantly modified when testing models with predicted frames output by SIMPLEFRAMEID, and a 10 points drop in recall is observed between the feature-expanded and the standard version of SEMAFOR (Tables 16 and 17). Those results suggest that the feature-expanded version of SEMAFOR acquires, most likely via the exemplar data, representations that deviate too much from that of those found in the fulltext test set, making it unable to compensate for the wrongly predicted frames of SIMPLEFRAMEID. Such considerations stress once again the possible domain-specific nature of both fulltext and exemplar data (Das et al., 2014), considerations which are reinforced by our results on frame identification trained on exemplar (see Table 18 and Table 19) which show a systematic decrease in performance on ambiguous predicates, suggesting that ambiguous predicates observed in the exemplar training data do not beneficially contribute to predictions on ambiguous predicates in the fulltext test set.

XP	FRAMENET 1.5 SETUP			P	R	F_1
#170	SEMAFOR	NLP4J	BMST	63.4	60.2	61.8
#295	SEMAFOR H + EX + F	NLP4J	BMST	**66.8**	50.3	57.4
#178	OPEN-SESAME	NLP4J	-	63.7	**64.4**	64.1
#182	OPEN-SESAME	NLP4J	BMST	66.1	64.0	**65.0**

Table 16: Argument identification baseline scores on **FN 1.5** splits, with frames predicted by SIMPLEFRAMEID trained on the **FN 1.5** train splits

XP	FRAMENET 1.7 SETUP			P	R	F_1
#147	SEMAFOR	NLP4J	BMST	63.1	60.6	61.8
#265	SEMAFOR H + EX + F	NLP4J	BMST	**68.7**	48.5	56.9
#148	OPEN-SESAME	NLP4J	-	64.0	55.2	59.3
#157	OPEN-SESAME	NLP4J	BMST	65.7	**62.9**	**64.3**

Table 17: Argument identification baseline scores on **FN 1.7** splits, with frames predicted by SIMPLEFRAMEID trained on the **FN 1.7** train splits

xp	FN 1.5	pos	dep	total	ambig	no-lex
#66	FT	NLP4J	BMST	83.2	**73.6**	**77.8**
#90	FT + EX	NLP4J	BMST	**84.6**	69.3	75.8

Table 18: Frame identification accuracy baseline on **FN 1.5** trained on fulltext (FT) and combined fulltext and exemplar (FT+EX) data

xp	FN 1.7	pos	dep	total	ambig	no-lex
#98	FT	NLP4J	BMST	82.3	**70.0**	**76.6**
#99	FT + EX	NLP4J	BMST	**83.6**	66.7	74.3

Table 19: Frame identification accuracy baseline on **FN 1.7** trained on fulltext (FT) and combined fulltext and exemplar (FT+EX) data

7 Conclusion

In this paper, we have discussed the impact of pre- and post-processing pipelines on frame semantic parsing performance. We have shown that the ranking of existing state-of-the-art systems can be severely impacted by arguably 'small' factors outside of the main models, to the point that respective model performance cannot be fully ascertained.

Our replication study drew the following conclusions. We found that on the whole, previous frame identification results by (Hartmann et al., 2017) could be replicated. Results on argument identification, however, gave a much patchier picture. Replication on a robust experimental setup showed a change in model ranking between OPEN-SESAME and SEMAFOR. Further experiments on pre-processing demonstrated that the *combination* of a particular parts-of-speech tagger and dependency parser (NLP4J and BIST) gives best performance on the task. The contribution of the dependency parser itself may be more than previously reported: we found that half of the advantage of OPEN-SESAME over SEMAFOR may be due to this pre-processing choice, closing the gap between the two models. We also highlighted that filtering the training data has a robust and very significant effect on performance.

Building upon these results, we ran two robust baselines on both FN1.5 and FN1.7. These experiments result in a different model ranking from what has previously been reported in the literature. However, we also found that the feature-expanded version of SEMAFOR suffers a huge drop in recall when given predicted rather than gold frames. Here again, it seems vital to be able to distinguish the contribution of different processing stages on the actual model.

On the back of the replication results presented here, we feel there is a need for a standardized experimental pipeline that would allow researchers to fully test the contribution of different pre- and post-processing factors independently from their model. Following on this recommendation, we release an open source toolkit[8] including all necessary scripts and datasets to replicate our experiments in a robust setting. We hope to thus foster more fine-grained analyses of past and future results in frame semantic parsing.

Acknowledgements

The authors are grateful to Meghana Kshirsagar for her help with the SEMAFOR parser.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally Normalized Transition-Based Neural Networks. *CoRR*, abs/1603.06042.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90, Montreal, Quebec, Canada, August. Association for Computational Linguistics.

Collin Baker, Michael Ellsworth, and Katrin Erk. 2007. SemEval-2007 Task 19: Frame Semantic Structure Extraction. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 99–104, Prague, Czech Republic, June. Association for Computational Linguistics.

Daniel Bauer, Hagen Fürstenau, and Owen Rambow. 2012. The Dependency-Parsed FrameNet Corpus. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 3861–3867, Istanbul, Turkey, May. European Language Resources Association (ELRA). ACL Anthology Identifier: L12-1619.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

Jinho D. Choi. 2016. Dynamic Feature Induction: The Last Gist to the State-of-the-Art. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 271–281. Association for Computational Linguistics.

[8] Available at `https://gitlab.com/akb89/pyfn`.

Dipanjan Das and Noah A. Smith. 2011. Semi-Supervised Frame-Semantic Parsing for Unknown Predicates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1435–1444, Portland, Oregon, USA, June. Association for Computational Linguistics.

Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame Semantic Parsing. *Computational Linguistics*, 40(1):9–56.

Charles J. Fillmore. 1982. Frame Semantics. *Linguistics in the morning calm*, pages 111–137.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic Role Labeling with Neural Network Factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 960–970, Lisbon, Portugal, September. Association for Computational Linguistics.

Antske Fokkens, Marieke van Erp, Marten Postma, Ted Pedersen, Piek Vossen, and Nuno Freire. 2013. Offspring from Reproduction Problems: What Replication Failure Teaches Us. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1691–1701, Sofia, Bulgaria, August. Association for Computational Linguistics.

Silvana Hartmann, Ilia Kuznetsov, Teresa Martin, and Iryna Gurevych. 2017. Out-of-domain FrameNet Semantic Role Labeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 471–482, Valencia, Spain, April. Association for Computational Linguistics.

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic Frame Identification with Distributed Word Representations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1448–1458, Baltimore, Maryland, June. Association for Computational Linguistics.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *TACL*, 4:313–327.

Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2015. Segmental Recurrent Neural Networks. *CoRR*, abs/1511.06018.

Meghana Kshirsagar, Sam Thomson, Nathan Schneider, Jaime Carbonell, Noah A. Smith, and Chris Dyer. 2015. Frame semantic Role Labeling with Heterogeneous Annotations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 218–224, Beijing, China, July. Association for Computational Linguistics.

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

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual Dependency Analysis with a Two-Stage Discriminative Parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 216–220, New York City, June. Association for Computational Linguistics.

George A. Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41.

Adwait Ratnaparkhi. 1996. A Maximum Entropy Model for Part-Of-Speech Tagging. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*. Association for Computational Linguistics, May.

Michael Roth and Mirella Lapata. 2015. Context-aware Frame-Semantic Role Labeling. *Transactions of the Association for Computational Linguistics*, 3:449–460.

Michael Roth. 2016. Improving Frame Semantic Parsing via Dependency Path Embeddings. In *Book of Abstracts of the 9th International Conference on Construction Grammar*, pages 165–167, Juiz de Fora, Brazil, October.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson, Collin F. Baker, and Jan Scheffczyk. 2016. FrameNet II: Extended Theory and Practice. Technical report, ICSI, Berkeley.

Swabha Swayamdipta, Sam Thomson, Chris Dyer, and Noah A. Smith. 2017. Frame-Semantic Parsing with Softmax-Margin Segmental RNNs and a Syntactic Scaffold. *CoRR*, abs/1706.09528.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient Inference and Structured Learning for Semantic Role Labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling Up To Large Vocabulary Image Annotation. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI*.

Bishan Yang and Tom Mitchell. 2017. A Joint Sequential and Relational Model for Frame Semantic Parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1258–1267, Copenhagen, Denmark, September. Association for Computational Linguistics.

Sensitivity to Input Order: Evaluation of an Incremental and Memory-Limited Bayesian Cross-Situational Word Learning Model

Sepideh Sadeghi
Tufts University
Human-Robot Interaction Laboratory
Medford, MA, USA
sepideh.sadeghi@tufts.edu

Matthias Scheutz
Tufts University
Human-Robot Interaction Laboratory
Medford, MA, USA
matthias.scheutz@tufts.edu

Abstract

We present a variation of the incremental and memory-limited algorithm in (Sadeghi et al., 2017) for Bayesian cross-situational word learning and evaluate the model in terms of its functional performance and its sensitivity to input order. We show that the functional performance of our sub-optimal model on corpus data is close to that of its optimal counterpart (Frank et al., 2009), while only the sub-optimal model is capable of predicting the input order effects reported in experimental studies.

1 Introduction

It has been suggested that early word learning is guided by observing the referent of words across situations (*cross-situational word learning*). Surely the co-occurrence statistics of word-object mappings is noisy (e.g., when the referent of a word is absent in the scene) and only provides ambiguous information (due to referential ambiguity) about word-object mappings, but it is the only source of information used by a true novice who does not have access to social and linguistic cues (Baldwin, 1993; Arunachalam and Waxman, 2010) to narrow down the hypothesis space of words' meanings. In fact, development of some initial meaning interpretations is the prerequisite for the development of useful social and syntactic knowledge (cues) for bootstrapping the process of word learning.

Recent experimental findings (Medina et al., 2011; Trueswell et al., 2013; Zhang and Yu, 2017) indicate that word learning is sensitive to the presentation order of learning trials. More specifically, their results suggest that early wrong interpretations of words' meanings can inhibit the acquisition of correct words' meanings and that performance is best when highly informative learning trials (e.g., with lower referential ambiguity) precede the low informative learning trials, so that the possibility of starting with wrong meaning interpretations is minimized. These results are incompatible with the prediction of the existing well-known word learning models which employ global learning and are equipped with perfect memory of past observations (Yu and Ballard, 2007; Kachergis et al., 2012; Xu and Tenenbaum, 2007; Frank et al., 2009; Alishahi and Fazly, 2010). These models are insensitive to input order due to two major reasons: (1) being purely driven by co-occurrence regularities and assuming full access to co-occurrence statistics which stays the same regardless of input order or (2) processing the data in batch. Word learning is one of a wide range of cognitive tasks which exhibit sensitivity to the presentation style of input. Inductive category learning (Carvalho and Goldstone, 2015), category generalization (Spencer et al., 2011), property projection (Lawson, 2017), etc. all demonstrate important differences under sequential vs. simultaneous presentation of input. Therefore, the root causes of the effects of presentation order on word learning results should be understood as it may generalize to other cognitive tasks and can make up an important aspect of word learning models.

The recent findings regarding the effect of presentation style and order provide evidence for local approaches to learning, while they fail to rule out the possibility of global learning. Exploring the intermix of local and global approaches to learning is important on three levels: (1) replicating human performance, (2) departing from computationally expensive ideal learners that become inefficient as the size of

Proceedings of the 27th International Conference on Computational Linguistics, pages 3170–3180
Santa Fe, New Mexico, USA, August 20-26, 2018.

input data grows, and (3) giving accounts for the real-world constraints faced by word learners (e.g., It is cognitively implausible to assume that infants can remember the details of all past observations). We use a variation of the incremental and memory-limited learning algorithm proposed in (Sadeghi et al., 2017) and provide the first corpus evaluation of this sub-optimal model in terms of its functional performance and the patterns of sensitivity to input order it exhibits which match those reported in the experimental studies (Medina et al., 2011).

2 Model Overview

Similar to previous work (Yu and Ballard, 2007; Alishahi and Fazly, 2010; Frank et al., 2009), we assume that the learner is capable of object and action categorization prior to word learning and the goal is to learn the basic level object category names. The *input* to the model (*observations*) are word learning *situations* (*trials*) each of which consists of an un-ordered set of words as utterance paired with a list of objects as the scene. The notion of *memory* refers to the number of observations (trials) that can be remembered at a time and the current model assumes memory=1. In each situation, the proposed incremental model (1) utilizes a Bayesian approach to infer the correct word-object mappings and (2) integrates the newly inferred word-object mappings (*mini-lexicon*) in the previously learned many-to-many mapping between words and objects (*lexicon*), based on their association strength in the presence of alternative mappings. The model's memory is limited to the word-object mappings stored in its full lexicon and the details of the current situation. The Bayesian approach is constrained on two distinct levels: (1) hypothesis generation (mini-lexicon generation) and (2) hypothesis evaluation (inferring the best mini-lexicon). Both hypothesis generation and hypothesis evaluation are limited to the knowledge stored in the lexicon and the details of the current situation. Therefore in our model, Bayesian approach neither performs global optimization, nor has access to all possible hypotheses (all possibilities if the model had full access to all observations). Instead, the Bayesian approach is a local approach as both hypothesis generation and hypothesis evaluation are limited to the knowledge stored in the lexicon and the details of the current situation.

3 Speaker's Model of Utterance Generation

The learner assumes that in each situation, the speaker uses the generative process illustrated in Fig. 1 to produce an utterance (W_s) corresponding to the current scene (O_s) and using a context appropriate portion (L) of the full lexicon. In each situation, the model has to infer which words are *referential* and to which object they refer to, where a *referential* word is a noun with a concrete object referent in the scene. *Referential intentions* of the speaker (I_s), which refer to the objects that are present in the scene and the speaker is talking about them, determine the space of possible referents for each referential word in the utterance (W_s).

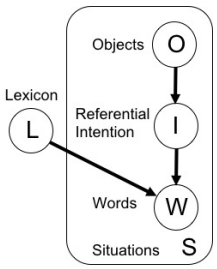


Figure 1: The graphical model describing the probabilistic dependencies between model random variables: utterance (W_s), intention (I_s), lexicon (L), and the objects present in the scene (O_s), where s indexes the situation. The plate indicates multiple copies of the model for different situations (utterance-scene pairs).

In each situation, the speaker uniformly samples a subset from the power-set of all the objects (O_s) present in the scene as the referential intention(s) of the speaker (I_s). For each object in I_s, one word

mapping is drawn from the lexicon and added to the utterance. The non-referential words of the utterance are generated randomly with probability $P_{NR} = 1$. In each situation the learner tries to reverse the generative process described in Fig. 1 to infer a context-appropriate portion of the full lexicon (*mini-lexicon*) used by the speaker, where *context* refers to the objects in the current situation. The inferred mini-lexicon then will be integrated in the full lexicon available to the learner (details in Section 4).

Inferring the best mini-lexicon in each situation, requires finding the MAP (maximum a posteriori) mini-lexicon by marginalizing over all possible referential intentions, since I_s is unobserved. The model finds the MAP mini-lexicon ($argmax_L P(L|C)$) according to the Bayes equation and the probability distribution that it defines over unobserved mini-lexica (L) and the corpus of situations (C). The current model only remembers one situation at a time (memory=1) due to which C includes one observed situation plus the context-appropriate situations extracted from the lexicon (see 4.2 for details).

$$P(L|C) \propto P(C|L)P(L) \tag{1}$$

We use $P(L) \propto e^{-\alpha \cdot |L|}$ serving as a soft mutual exclusivity constraint to produce a preference for one-to-one mappings in the mini-lexicon inferred in each situation. Marginalizing over all possible intentions in each situation we can rewrite the likelihood term $P(C|L)$ as:

$$P(C|L) = \prod_{s \in C} \sum_{I_s \subseteq O_s} P(W_s|I_s, L)P(I_s|O_s) \tag{2}$$

Assuming that $P(I_s|O_s) \propto 1$ and that the words of the utterance are generated independently, we can rewrite the term $P(W_s|I_s, L)$ as:

$$P(W_s|I_s, L) = \prod_{w \in W_s} [\gamma \cdot \sum_{o \in I_s} \frac{1}{|I_s|} P_R(w|o, L) + (1 - \gamma) P_{NR}(w|L)] \tag{3}$$

The words of utterance can be generated in two different ways: (1) referentially (with a concrete object referent in the scene) with probability γ and (2) non-referentially with probability $1 - \gamma$. The probability of non-referential use (P_{NR}) of a referential word (words in the model lexicon) is set to $\kappa < 1$ (to penalize the non-referential use of referential words), and is set to 1 for non-referential words. The probability of referential use of a referential word in reference to a particular object (P_R) is the probability of the word being chosen uniformly from the set of all words linked to that object in the lexicon.

4 Incremental Learning Algorithm

Our learning algorithm is a variation of the incremental and memory-limited algorithm proposed in (Sadeghi et al., 2017) which accounts for the following real-world constraints: (1) seeing each situation only once with no iteration over data, (2) using only the acquired knowledge and the current observation for hypothesis generation and evaluation. These constraints exclude the use of many proposed incremental algorithms in the literature (Liang et al., 2009; Pearl et al., 2010; Börschinger and Johnson, 2011; Börschinger and Johnson, 2012). In this section we first give a high-level overview of the incremental learning algorithm, while highlighting the differences between our version and the original version proposed in (Sadeghi et al., 2017). We refer the reader to (Sadeghi et al., 2017) for the details shared between the two algorithms. Furthermore, we refer curious readers to (Sadeghi and Scheutz, 2017; Sadeghi and Scheutz, 2018) which demonstrate the application of similar learning algorithms to the augmented versions of the graphical model presented here.

Our algorithm is composed of two major components: (1) inferring the MAP mini-lexicon in each situation, and (2) integrating the new mini-lexicon in the previous lexicon. Inferring the MAP mini-lexicon, subsequently has two components: (1) hypothesis generation (generating mini-lexicon proposals) and (2) hypothesis evaluation (evaluating the mini-lexicon proposals using Eq. 1).

Our algorithm departs from its batch counterpart (Frank et al., 2009) in the notion of C and L. L in (Frank et al., 2009) refers to the entire dictionary used by the speaker while in our model L (mini-lexicon)

refers to a part of the dictionary used by the speaker to produce the current utterance in correspondence to the current scene. In each situation, we try to infer a portion of the entire dictionary used by the speaker and we refer to the aggregation of all these mini-lexica as the lexicon. Similarly, C in (Frank et al., 2009) refers to all the observed situations (<*utterance*,*scene*> pairs), assuming full access to data, but in our model C refers to the current situation and a number of relevant situations extracted from the lexicon.

Our algorithm departs from the algorithm in (Sadeghi et al., 2017) in the definition of *context* and subsequently the *context-appropriate mappings* and the *context-appropriate situations* used for hypothesis generation and hypothesis evaluation accordingly in each situation. *Context* in our model refers to the list of objects present in the scene while *context* in (Sadeghi et al., 2017) refers to the objects and words present in the scene and utterance. Given *trial*=<*utterance*,*scene*>, we define *context*=*trial.objects*=$\{o_j | o_j \in$ *scene*$\}$, *trial.words*=$\{w_i | w_i \in$ *utterance*$\}$, and *trial.mappings*=$\{< w_i, o_j > | (o_j \in$ *scene*$) \wedge (w_i \in$ *utterance*$)\}$. Then the context appropriate information are defined as: *context-appropriate-mappings*=*trial.mappings* $\cup \{< w_i, o_j > | (< w_i, o_j > \in$ *lexicon*$) \wedge (o_j \in$ *context*$)\}$, and C=*context-appropriate-situations*=$\{<$ *utterance*=w_i,*scene*=$o_j > | (< w_i, o_j > \in$ *lexicon*$) \wedge (o_j \in$ *context*$)\} \cup$ *trial*.

We focus on objects as the context to extract appropriate information from the lexicon, since the goal of the model is to learn the label of objects while the mappings stored in the lexicon are noisy and can include object mappings for non-referential words. Therefore, we can benefit from filtering out the potential noise (co-occurrence of the non-referential words and objects) and retrieving only the stored mappings of the objects as a proxy for true object labels during hypothesis generation and hypothesis evaluation.

4.1 Generating Mini-Lexicon Proposals

Generating mini-lexicon proposals in each situation, is guided by semi-stochastic search techniques on the context-appropriate mappings based on their association strength (Point-wise mutual information accumulated incrementally and globally) analogous to (Sadeghi et al., 2017). Upon receiving a new input situation, first, we initialize 40 mini-lexica and take the one with the highest un-normalized posterior probability as the best initialized mini-lexicon (*init-lex*). We then explore the neighborhood around *init-lex* by mutating it *nStep* times (we used *nStep*=3). Mutation can be thought of as generating a tree of height=*nStep*+1, with branch factor of 3 (performing 3 operations on mappings of the lexicon at each node: add, remove, swap), the *init-lex* at the root, and its mutated versions on the leaves. We then compare the *init-lex* and its mutated versions against each other and report the one with the highest un-normalized posterior probability as the inferred mini-lexicon.

4.2 Evaluating Mini-Lexicon Proposals

The mini-lexica generated in the previous section are evaluated based on their relative posterior probability computed according to Eq. 1. C includes the context-appropriate situations and is used as evidence for Bayesian inference when evaluating the mini-lexicon proposals. Note that the model does not remember the details of past observations but is able to remember what it learned from them (word-object mappings stored in the lexicon). Hypothesis evaluation in our model departs from hypothesis evaluation in (Sadeghi et al., 2017) by virtue of using a different notion of context and context-appropriate situations used as evidence for Bayesian inference.

4.3 Integrating the MAP Mini-Lexicon in the Lexicon

Integration of the MAP mini-lexicon modifies (adds, deletes or keeps) the word mappings stored in the lexicon, for each item in the context (objects in the current scene). Our algorithm departs from the original version in how it handles conflict resolution when integrating the newly inferred mini-lexicon in the previous lexicon. In (Sadeghi et al., 2017), during conflict resolution, alternative mappings compete with each other and only mappings with the highest co-occurrence statistics are allowed in the output lexicon. This strict mutual exclusivity constraint not only inhibits learning of alternative word-object mappings (e.g., “cat”,“kitten”, and “kittycat” all refer to the same concept “CAT”), but also destabilizes the learning results. Our model on the other hand, modulates this strict mutual exclusivity constraint by

allowing the addition of alternative mappings for each object, if their co-occurrence statistics fall within a certain range of the co-occurrence statistics of the best existing mapping for that object. We introduce two parameters: (1) *keep-fraction* and (2) *add-fraction* for the inclusion of alternative mappings in the output lexicon, correspondingly from the previous lexicon and the MAP mini-lexicon. For each object in current context, we allow mappings from the previous lexicon and from the MAP mini-lexicon in the output lexicon if their association strength are correspondingly not less than *keep-fraction* and *add-fraction* of the word mapping with the highest association strength available in memory, for that object. Using small values for *keep-fraction* and *add-fraction* would enforce softer mutual exclusivity constraints and using large values would enforce harder mutual exclusivity constraints.

5 Evaluation Data

5.1 Input Properties

The input properties we study include: (1) the information inherent in learning trials and (2) the serial order of their presentation. We categorize the trials into two categories: *referential* and *non-referential* trials. A trial is *non-referential* if no word in its utterance has a concrete object referent in the scene. Otherwise, the trial is regarded as a *referential trial*. *non-referential* trials are misleading trials as the word-object co-occurrence statistics inherent in them do not guide the model towards learning the correct word-object mappings. Note that *non-referential* trials are generally misleading for any learner which uses object-word co-occurrence statistics to guide word learning. We use *misleading* and *non-referential* interchangeably to refer to the same kind of trials. The referential trials in turn, contain graded levels of useful co-occurrence statistics based on their overall *referential ambiguity*. We use the product of the number of objects in the scene and the number of words in the utterance as a simple measure of trial *ambiguity*.

5.2 Corpus Properties

Table. 1 and Table. 2 report the properties of the two annotated video files used for our corpus evaluations. Note that di06 and me03 differ from each other in the ratio of referential to non-referential trials (1:2.07 for di06 and 1:4.64 for me03) meaning that overall the data in di06 is more informative than the data in me03 for labeling objects. D-di-me and D-me-di differ from each other in the presentation order of the more informative video and D-diRef-meRef differs from D-di-me in that it contains no non-referential (misleading) trials. Our evaluation data is available at `https://tinyurl.com/y9omp52c`.

Table 1: Properties of the annotated child-directed video files used for corpus evaluations. di06 and me03 are taken from the Rollins section of CHILDES (MacWhinney, 1991). We used the annotations and gold lexicon used in (Frank et al., 2009).

Video	Referential trials	Non-referential trials	Average ambiguity	Average ambiguity (referential trials)
me03	56	260	7.3	10.73
di06	100	207	9.35	12.02

Table 2: Different datasets made up of the trials in me03 and di06 for testing the incremental model.

Data	Video files
D-di-me	trials in di06 followed by trials in me03
D-me-di	trials in me03 followed by trials in di06
D-diRef-meRef	referential trials in di06 followed by referential trials in me03

5.3 Target Effects of Input Properties

Recent experimental studies (Medina et al., 2011; Trueswell et al., 2013; Zhang and Yu, 2017) report a huge effect of input order in word learning results, where performance is reported to be sensitive to the

presentation order and informativity of learning trials. It has been demonstrated that highly informative trials (referential trials with low ambiguity) facilitate the acquisition of correct meaning interpretations while low informative trials (highly ambiguous referential trials or non-referential trials) facilitate the acquisition of incorrect meaning interpretations. Additionally, it has been reported that early correct interpretations of words' meanings can facilitate the acquisition of correct words' meaning in later encounters while early wrong interpretations of words' meanings can inhibit the acquisition of correct words' meanings in later encounters.

Taking these results together, we expect the model performance: (1) to be sensitive to the presentation order of learning trials, (2) to be best when there is no misleading learning trials in the input data, and (3) to be best when highly informative learning trials precede the low informative learning trials, so that the possibility of starting with wrong meaning interpretations is minimized. We assume that referential trials are more informative than non-referential (misleading) trials and that referential trials with low ambiguity are more informative than referential trials with high ambiguity.

6 Evaluations

In the following sections, for each model, we use precision, recall and fscore of its output lexicon (w.r.t the gold standard lexicon) to measure its functional performance at the end of the simulation and we use *mean word acquisition score* (Alishahi and Fazly, 2010) to evaluate its incremental performance during the simulation.

$$\textbf{mean word acquisition score} = \frac{\sum_{<w,o>\in goldLexicon} P(w|o, lexicon)}{size(goldLexicon)} \quad (4)$$

6.1 Functional Performance

Table. 3 contrasts the functional performance of the our proposed incremental model with that of its batch (optimal) counterpart (Frank et al., 2009) as well as a number of baseline incremental models (please refer to (Frank et al., 2009) for more details about the comparison models). The results reported for the comparison models are taken from (Frank et al., 2009). Analogous to (Frank et al., 2009), we report the precision, recall and fscore of the best lexicon found by the incremental model, where the lexicon with the highest fscore value is considered the best. We performed a grid search in the space of parameters to find the best lexicon for each model and data pair (top three scores are highlighted in each category).

The proposed incremental model is tested on three different datasets (see Table. 2) to evaluate the effect of input properties on model performance. We did not do the same for other models as they are not sensitive to input order.

Table 3: Precision, recall, and f-score of the best lexicon found by each model when run on two annotated video files (di06 and me03) from the Rollins section of (MacWhinney, 1991). The incremental model is the only model which is sensitive to input order and is tested on a number of datasets presented in Table. 2.

Model(data)	Precision	Recall	F-Score
batch model (Frank et al., 2009)	**0.67**	0.47	**0.55**
Incremental model(D-di-me)	**0.259**	**0.617**	**0.365**
Incremental model(D-me-di)	0.204	**0.558**	0.299
Incremental model(D-diRef-meRef)	**0.415**	**0.647**	**0.505**
Association frequency	0.06	0.26	0.1
$P(object\|word)$	0.07	0.21	0.1
$P(word\|object)$	0.07	0.32	0.11
Point-wise MI	0.06	0.47	0.11
Translation Model $P(object\|word)$	0.07	0.32	0.12
Translation Model $P(word\|object)$	0.15	0.38	0.22

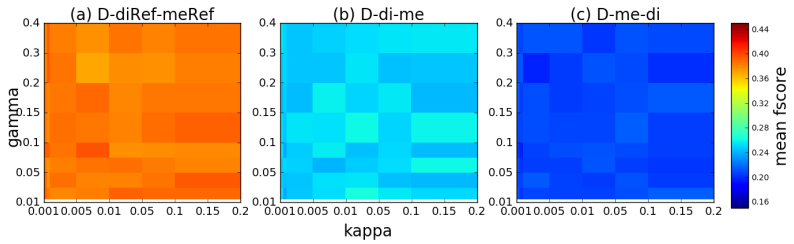


Figure 2: The mean fscore of the lexica found by running the model on (a) D-diRef-meRef, (b) D-di-me, and (c) D-me-di. D-di-me and D-me-di have 623 trials each, while D-diRef-meRef has only 156 trials. In each heatmap, the results are averaged over 50 runs for each set of parameter values using $\alpha = 3$, *keep-fraction* $= 0.5$, *add-fraction* $= 0.5$, and varying κ and γ.

Unsurprisingly, the batch model which has access to all datapoints has the highest precision and fscore on the dataset. Our incremental model has the second best fscore and the highest recall among all models as well as the best fscore among the incremental models. We used *keep-fraction* $=$ *add-fraction* $= 0.5$ (a soft mutual exclusivity constraint) which facilitates the addition of more mappings to the lexicon. This tends to improve the recall and hurt the precision of the model on small datasets such as the ones used here.

Note that the incremental model exhibits its best performance when misleading trials are removed from the corpus data (on D-diRef-meRef) which is inline with our second expectation discussed in Section 5.3 suggesting that misleading trials inhibit the process of learning the correct mappings by facilitating the acquisition of wrong meaning interpretations which in turn can interfere with the acquisition of correct meaning interpretations in later encounters.

As can be seen in Table. 1, the ratio of referential to non-referential trials in di06 is higher than that of me03. Therefore, based on what was discussed in Section 5.3, we expect the incremental model to exhibit better performance when: (1) di06 is presented first compared to when me03 is presented first, and (2) when non-referential trials are removed. Both (1) and (2) help minimize the possibility of early acquisition of wrong meaning interpretations. Our result is compatible with this expectation as the incremental model achieves better results on D-di-me dataset compared to D-me-di (reversed presentation order of video files), and better results on D-diRef-meRef compared to D-di-me (due to removal of misleading trials). Fig. 2 demonstrates that this outperformance pattern is robust against the choice of model parameters and is not an artifact of evaluating the best fscore values.

As can be seen in Fig. 2, regardless of parameter values, mean fscore values on D-diRef-meRef are better than mean fscore values on D-di-me which are in turn better than mean fscore values on D-me-di. Similar outperformance patterns were observed on the heatmaps of mean precision and mean recall scores. Furthermore, additional heatmaps generated using different α values provided further evidence for the robustness of the observed model outperformance on certain datasets regardless of the choice of model parameters and regardless of whether mean or max scores are used for comparison.

6.2 The Presentation Order of Ambiguous Trials

In this section, we seek to examine the effect of the presentation order of ambiguous trials. In order to filter out the effect of misleading trials, we use D-diRef-meRef with no misleading trials (see Table. 2) as the default ordered dataset. We reorded the trials in this dataset and created two new datasets: (1) with ascending trial ambiguity and (2) with descending trial ambiguity.

Fig. 3 demonstrates both the ideal rate of word acquisition (using a perfect model which only learns the correct mappings at the end of each trial) and the word acquisition rate achieved by the incremental model, on each dataset. As can be seen, high referential ambiguity (early trials of the data with descending order of ambiguity as well as the last trials of the data with ascending order of ambiguity) slows the rate of word acquisition as reflected on the slope of the green and red solid lines.

According to recent findings discussed in Section 5.3, performance is best when trials with low ref-

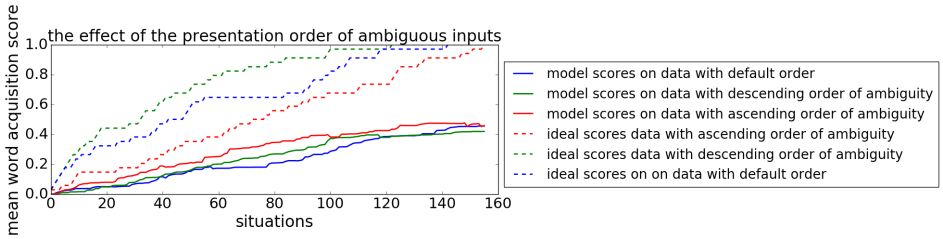


Figure 3: The mean word acquisition score over time as the incremental model (solid lines) is presented with three different datasets: (1) D-diRef-meRef (data with default order), (2) reordered D-diRef-meRef in ascending order of trial ambiguity, and (3) reordered D-diRef-meRef in decreasing order of trial ambiguity. The results are averaged over 50 runs, using the following parameter values: $\kappa = 0.005$, $\gamma = 0.3$, $\alpha = 5$, *keep-fraction* $= 0.5$ and *add-fraction* $= 0.5$. The dashed lines demonstrate the presentation rate of words in the gold standard lexicon in each dataset. The dashed lines also demonstrate the ideal mean word acquisition scores over time, if the model were to learn only the correct mappings at the end of each trial.

erential ambiguity precede the trials with high referential ambiguity. The results from our incremental model is in line with this expectation as the model achieves its best performance on the data with ascending order of ambiguity. Furthermore, the gap between the model performance and the ideal performance is the least when the data is presented in an increasing order of ambiguity. Finally, our results indicate that although input order affects the rate of acquisition, but (1) as more data becomes available, the extra cross-situational information inherent in data can make up for early input order effects resulting in similar word acquisition scores at the end of these experiments, and (2) that it is possible to replicate the input order effects using memory-limited continuous word learning mechanisms that accumulate information across all input trials.

6.3 The Presentation Order of Misleading Trials

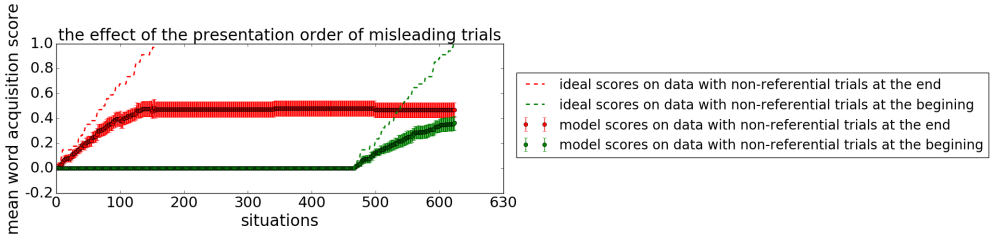


Figure 4: The mean word acquisition score over time as the incremental model (solid lines) is presented with two different datasets: (1) non-referential trials of di06 and me03 added at the end of D-diRef-meRef, (2) non-referential trials of di06 and me03 added at the beginning of D-diRef-meRef. The results are averaged over 50 runs, using the following parameter values: $\kappa = 0.005$, $\gamma = 0.3$, $\alpha = 5$, *keep-fraction* $= 0.5$ and *add-fraction* $= 0.5$. The dashed lines demonstrate the presentation rate of words in the gold standard lexicon in each dataset. The dashed lines also demonstrate the ideal mean word acquisition scores over time, if the model were to learn only the correct mappings at the end of each trial.

in Section 6.1, we demonstrated the inhibitory effect of misleading trials and their early presentation in the input data (as in D-me-di) on final word learning results (fscore, precision and recall). In order to establish what happens during the presentation of misleading trials and how the model is affected by these trials incrementally, we perform two more follow-up experiments using D-diRef-meRef dataset in increasing order of trial ambiguity: (1) putting back the non-referential trials of di06 and me03 at the beginning of D-diRef-meRef, (2) putting back the non-referential trials of di06 and me03 at the end of

D-diRef-meRef. These two follow-up experiments can be used to verify (1) if re-ordering the misleading trials can modulate their negative effect, and (2) how long lasting the effect of misleading trials is on word learning results.

Fig. 4 suggests that presenting the misleading trials after the referential trials seem to have no significant effect on previously learned correct mappings as no significant performance drop is observed on the red line. On the other hand, presenting the misleading trials before the referential trials, significantly affects the model performance as the solid green line remains significantly below the solid red line.

7 Discussion and Conclusion

We presented a variation of the incremental and memory limited word learning model in (Sadeghi et al., 2017) and provided the first corpus evaluations of this type of sub-optimal model. Our corpus evaluations demonstrated superior model performance compared to baseline incremental models. In addition to that the performance of the incremental model was close to the performance of the its batch counterpart (Frank et al., 2009), specially when misleading trials were removed from the corpus but only the incremental model was capable of accounting for input order effects.

We examined the replication of significant input order effects reported in (Medina et al., 2011; Trueswell et al., 2013; Zhang and Yu, 2017) in our model (1) incrementally, (2) on multiple datasets with different size and presentation order of ambiguous and misleading data, and (3) using different parameter values (Fig.2). The predictions of our model regarding the input order effects were in line with the recent empirical findings in (Medina et al., 2011; Trueswell et al., 2013; Zhang and Yu, 2017) as the model exhibited sensitivity to the information content of input trials as well as their presentation order. More specifically the model performance was best when highly informative (with low referential ambiguity) trials preceded the low informative (with high referential ambiguity) trials and when misleading trials were removed. Our simulations with different parameters showed that these outperformance patterns were robust against the choice of model parameters. Furthermore, our simulation results suggest that exposure to more data can neutralize early input order effects and that *memory-limited* continuous word learning mechanisms can replicate the input order effects.

The presented model is more cognitively plausible than its batch counterpart in that it is incremental and memory-limited. In addition to that, it relies on sub-optimal learning approaches (local optimization) as it utilizes a limited memory of past observations for incremental hypothesis generation and hypothesis evaluation performed by Bayesian inference. Applying the Bayesian inference locally and with limited data as evidence, allows the model to avoid becoming inefficient as the size of input data grows.

This work is the first corpus evaluation of a sub-optimal word learner which not only produces comparable functional performance to that of its optimal counterpart (Frank et al., 2009), but also successfully replicates the input order effects reported in experimental studies. Overall, the presented sub-optimal word learner is the first step towards exploring (1) local approaches to learning, (2) which depart from computationally expensive ideal learners, (3) without a huge loss in functional performance, (4) while replicating human performance, and (5) accounting for real-world constraints faced by learners (e.g., an infant or a robot). Furthermore, this work serves as an examination of the loss and gain of departing from optimal learners by performing direct comparisons between a sub-optimal model and its optimal counterpart. Taking (Frank et al., 2009) as a particular optimal model and the proposed model here as its sub-optimal version, we presented the loss (in functional performance) and gain (predicting input order effects) of departing from an optimal model (Frank et al., 2009).

In future work we plan to modulate model's memory of past observations (the number of observations that can be remembered with full details) to establish the effect of memory on replication of input order effects. Furthermore, we plan to compare our model with other sub-optimal models (Stevens et al., 2017) to gain a better understanding of the computational requirements for modeling the input order effects reported in experimental word learning literature.

Acknowledgements

This work was funded in part by Vienna Science and Technology Fund project ICT15-045.

References

Afra Alishahi and Afsaneh Fazly. 2010. Integrating syntactic knowledge into a model of cross-situational word learning. In *Proc. of CogSci*, volume 10.

Sudha Arunachalam and Sandra R Waxman. 2010. Meaning from syntax: Evidence from 2-year-olds. *Cognition*, 114(3):442–446.

Dare A Baldwin. 1993. Early referential understanding: Infants' ability to recognize referential acts for what they are. *Developmental psychology*, 29(5):832.

Benjamin Börschinger and Mark Johnson. 2011. A particle filter algorithm for bayesian word segmentation. In *Proceedings of the Australasian Language Technology Association Workshop*, pages 10–18. D. Mollá & D. Martinez.

Benjamin Börschinger and Mark Johnson. 2012. Using rejuvenation to improve particle filtering for bayesian word segmentation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 85–89. Association for Computational Linguistics.

Paulo F Carvalho and Robert L Goldstone. 2015. What you learn is more than what you see: what can sequencing effects tell us about inductive category learning? *Frontiers in psychology*, 6:505.

Michael C Frank, Noah D Goodman, and Joshua B Tenenbaum. 2009. Using speakers' referential intentions to model early cross-situational word learning. *Psychological Science*, 20:578–585.

George Kachergis, Chen Yu, and Richard M Shiffrin. 2012. An associative model of adaptive inference for learning word–referent mappings. *Psychonomic bulletin & review*, 19(2):317–324.

Chris A Lawson. 2017. The influence of task dynamics on inductive generalizations: How sequential and simultaneous presentation of evidence impacts the strength and scope of property projections. *Journal of Cognition and Development*, 18(4):493–513.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Probabilistic grammars and hierarchical dirichlet processes. *The handbook of applied Bayesian analysis*.

Brian MacWhinney. 1991. *The CHILDES project: Tools for analyzing talk.* Lawrence Erlbaum Associates, Inc.

Tamara Nicol Medina, Jesse Snedeker, John C Trueswell, and Lila R Gleitman. 2011. How words can and cannot be learned by observation. *Proceedings of the National Academy of Sciences*, 108(22):9014–9019.

Lisa Pearl, Sharon Goldwater, and Mark Steyvers. 2010. Online learning mechanisms for bayesian models of word segmentation. *Research on Language & Computation*, 8(2):107–132.

Sepideh Sadeghi and Matthias Scheutz. 2017. Joint acquisition of word order and word referent in a memory-limited and incremental learner. In *Proceedings of the 2017 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*.

Sepideh Sadeghi and Matthias Scheutz. 2018. Early syntactic bootstrapping in an incremental memory-limited word learner. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

Sepideh Sadeghi, Matthias Scheutz, and Evan Krause. 2017. An embodied incremental bayesian model of cross-situational word learning. In *Proceedings of the 2017 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (icdl-epirob)*.

John P Spencer, Sammy Perone, Linda B Smith, and Larissa K Samuelson. 2011. Learning words in space and time: Probing the mechanisms behind the suspicious-coincidence effect. *Psychological science*, 22(8):1049–1057.

Jon S Stevens, Lila R Gleitman, John C Trueswell, and Charles Yang. 2017. The pursuit of word meanings. *Cognitive science*, 41(S4):638–676.

John C Trueswell, Tamara Nicol Medina, Alon Hafri, and Lila R Gleitman. 2013. Propose but verify: Fast mapping meets cross-situational word learning. *Cognitive psychology*, 66(1):126–156.

Fei Xu and Joshua B Tenenbaum. 2007. Word learning as bayesian inference. *Psychological review*, 114:245–272.

Chen Yu and Dana H Ballard. 2007. A unified model of early word learning: Integrating statistical and social cues. *Neurocomputing*, 70:2149–2165.

Yayun Zhang and Chen Yu. 2017. How misleading cues influence referential uncertainty in statistical cross-situational learning. In *41st annual Boston University Conference on Language Development*, pages 820–833.

Sentence Weighting for Neural Machine Translation Domain Adaptation

Shiqi Zhang **Deyi Xiong***
School of Computer Science and Technology, Soochow University, Suzhou, China
zzssq77@163.com, dyxiong@suda.edu.cn

Abstract

In this paper, we propose a new sentence weighting method for the domain adaptation of neural machine translation. We introduce a domain similarity metric to evaluate the relevance between a sentence and an available entire domain dataset. The similarity of each sentence to the target domain is calculated with various methods. The computed similarity is then integrated into the training objective to weight sentences. The adaptation results on both IWSLT Chinese-English TED task and a task with only synthetic training parallel data show that our sentence weighting method is able to achieve an significant improvement over strong baselines.

1 Introduction

Neural machine translation (NMT) has achieved more satisfactory performance than statistical machine translation (SMT) on many language pairs with various advantages over SMT, such as no pipeline-style training, more fluent translations and so on. However, it is still confronted with a big challenge in domain adaptation as the translation quality of NMT is heavily dependent on the quantity of training data and the relevance between the training data and the in-domain testing data. The training corpus usually varies across domains, where out-of-domain instances that are relevant to the target domain are beneficial for training while those which are irrelevant to the in-domain may deteriorate translation quality. The widely-used domain adaptation method in NMT is to fine-tune an existing out-of-domain model with the in-domain data (Luong and Manning, 2015). However, the fine-tuning method is of no consideration for the harm caused by irrelevant out-of-domain data. It also tends to overfit rapidly due to the small size of the in-domain data.

In this paper, we propose a sentence weighting method that evaluates the weights of sentences with respect to the relevance of sentences to the target domain. We train NMT models by assigning each sentence with a corresponding weight computed according to a domain similarity metric. Domain similarity has been successfully used in some tasks such as parsing, knowledge adaptation, etc.(Plank and Van Noord, 2011; Ruder et al., 2017). We employ domain similarity to measure relevance between the sentences of out-of-domain and in-domain data. In this way, we take into consideration both the goodness and the badness brought by out-of-domain data, in a weighting fashion. Additionally, different from current sentence weighting methods used for NMT that score sentences by making use of other toolkits like the SRI Language Modeling Toolkit (Stolcke, 2002), or an RNN classifier (Chen et al., 2017), our method exploits the information from the NMT system itself. This means that we do not need to train extra toolkits. We also examine the effectiveness of the proposed sentence weighting method on NMT trained with only synthetic parallel data, which is beneficial for the low-resource domain translation. Our method can also be used in back translation and in conjunction with other training methods.

* Corresponding author

Proceedings of the 27th International Conference on Computational Linguistics, pages 3181–3190
Santa Fe, New Mexico, USA, August 20-26, 2018.

Experiments show clear gains on the IWSLT Chinese-English task. Comparing to the previous sentence weighting method (Wang et al., 2017b), we achieve the highest improvement of 1.71 BLEU among four test sets, and obtain an average gain of 1.42 BLEU over Wang et al. (2017b)'s method. The experiments on NMT trained with only synthetic parallel data further confirm the effectiveness of our sentence weighting method on domain adaptation.

The paper is organized as follows. Section 2 overviews related work. In section 3, we give a background introduction of NMT. We present our sentence weighting method in Section 4. In section 5, we introduce the experimental setting and the NMT model configurations. In section 6 we show and discuss the results of our two groups of experiments, followed by our conclusion and future works in section 7.

2 Related Work

A lot of investigations have already been conducted for domain adaptation in SMT while few in neural machine translation. These methods can be roughly categorized into two classes: the model-level and data-level method.

At the model level, combining multiple translation models in a weighted manner is used for SMT domain adaptation. For NMT, fine tuning, model stacking and muti-model ensemble have been explored (Sajjad et al., 2017). Luong and Manning (2015) propose a fine-tuning method, which continues to train the already trained out-of-domain system on the in-domain data. Model stacking is to build an NMT model in an online fashion, training the model from the most distant domain at the beginning, fine-tuning it on the closer domain and finalizing it by fine-tuning it on the in-domain data. Muti-model ensemble combines multiple models during decoding using a balanced or weighted averaging method.

At the data level, traditional domain adaptation approach can be done by data selection, data weighting or data joining. Data selection approaches select data similar to the in-domain data according to some criteria. Normally, the out-of-domain data can be scored by a model trained on the in-domain data and out-of-domain data. For example, a language model can be used for scoring sentences (Axelrod et al., 2011). Data weighting methods weight each item which can be a corpus, a sentence or a phrase, and then train SMT models on weighted items.

Although some existing SMT domain adaptation techniques can be directly applied to NMT, it is challenging for applying data weighting to NMT. For NMT, the data selection approach can also be used. Wang et al. (2017a) employ the data selection method for domain adaptation, which uses sentence embeddings to measure the similarity of a sentence pair to the in-domain data. A recent method to apply sentence weights to NMT is cost weighting (Wang et al., 2017b; Chen et al., 2017). The NMT objective function is updated by sentence weighting when computing the cost of each mini-batch during NMT training. Wang et al. (2017b) exploit an in-domain language model (Axelrod et al., 2011) to score sentences. Chen et al. (2017) use a classifier to assign weights for individual sentences pairs.

Domain control uses word-level domain features in the word embedding layer, aiming to allow a model to be built from a diverse set of training data to produce in-domain translations (Kobus et al., 2017).

3 NMT

In this paper, we use the vanilla attention-based NMT (Bahdanau et al., 2014), which we will briefly summarize here.It is built on a recurrent neural networks (RNN) in an encoder-decoder framework (Sutskever et al., 2014; Pascanu et al., 2013).

Given a source sentence $x = (x_1, x_2, ..., x_m)$ and its corresponding target sentence $y = (y_1, y_2, ..., y_n)$, NMT employs the encoder-decoder framework to jointly train to maximize the conditional probability $p(y|x)$. To solve the problem that a basic encoder-decoder framework deteriorates rapidly as the length of input sentence increases, the attention mechanism (Bahdanau et al., 2014; Luong et al., 2015) is proposed.

The encoder employes a bi-directional RNN to encode the source sentence x into a sequence of hidden states h. h is the concatenation of the forward hidden state $\overrightarrow{h_i}$ and backward hidden state $\overleftarrow{h_i}$. Each hidden state h_i is computed from the previous hidden state h_{i-1} and the current source word x_i as follows.

$$\overrightarrow{h_i} = f(\overrightarrow{h}_{i-1}, x_i)$$

The decoder reads the hidden states and predicts the target translation by maximizing the conditional log-probability of the correct translation y. Each word y_i is predicted based on a recurrent hidden state s_i and a context vector c_i that aims at capturing relevant source-side information. c_i is computed as a weighted sum of the annotation h_i,

$$c_j = \sum_{j=1}^{m} \alpha_{ji} h_i$$

The weight α_{ji} of each annotation h_i is a normolized output from a softmax operation with a two layer feed-forward neural network ϕ. The weight α_{ji} of each annotation h_i indicates the probability of the target y_j being aligned to the source x_i.

$$\alpha_{ji} = \frac{exp(\phi(s_{j-1}, h_i))}{\sum_{k=1}^{n} exp(\phi(s_{j-1}, h_k))}$$

The probability of the target sentence y given a source sentence x is modeled as

$$P(y|x) = \prod_{j=1}^{m} P(y_j|s_j, y_{j-1}, c_j)$$

with s_j being the decoder state, c_j being the context, y_{j-1} being the last generated word.

Given a parallel corpus D, the parameters θ are trained to maximize the conditional probabilities of all sentences:

$$J = \sum_{(x,y)\in D} log(y|x)$$

4 Sentence Weighting for NMT

We evaluate domain adaptation methods on two different kinds of scenarios. One is for the usual domain adaptation that adapts a model trained on a rich-resource out-of-domain corpus to the in-domain with limited data. The other is for low-resource domain translation task, where only synthetic parallel data are available. For the latter, all the data are from the same domain while there are some differences between the true data and pseudo data. There also must be some mismatch between pseudo data and test data. For this kind of translation task, we can consider the development set as in-domain data, and the whole training data as out-of-domain data, though actually all data are from the same domain.

4.1 Sentence Weighting for Standard Domain Adaptation

We evaluate the out-of-domain sentences according to their domain similarity to the in-domain data. We consider the distribution of sentences in one domain corpus as the representation of the domain. Our hypothesis is that the average of all sentence embeddings (Wang et al., 2017a) in the domain space represents the core of the domain. In this way, we define the similarity measure between a sentence and a domain into that of the sentence to the core of the domain. The sentence embedding (Wang et al., 2017a) s_i is the representation of the initial hidden state for the decoder (Bahdanau et al., 2014). We can measure the Euclidean distance between the sentences and the domain with a method used by Wang et al. (2017a). In addition, we propose a new sentence weighting method using Jensen-Shannon (JS) Divergence (Lin, 1991) to measure the domain similarity.

We use softmax function to transform sentence embeddings into a probability distribution. Given a domain corpus of size N where each sentence embedding s_i is the initial hidden state for the decoder which is transformed from the embedded source sentence vector (Wang et al., 2017a), the probability of a sentence is computed as follows.

$$\sigma_{s_i} = \frac{exp(s_i)}{\sum_{k=1}^{k=N} exp(s_k)}$$

We therefore choose the Jensen-Shannon (JS) Divergence as the similarity function to measure the similarity between a sentence and a domain based on the distribution..

The Jensen-Shannon Divergence is a probabilistically-motivated function. It is symmetric and computes the Kullback–Leibler (KL) divergence between q and r, and their average. We use the JS divergence defined in (Lee, 2001):

$$JS(q,r) = \frac{1}{2}\left[D(q||avg(q,r)) + D(r||avg(q,r))\right]$$

where $D(q||r)$ is the KL divergence, which is a classical measure of "distance" between two probability distributions, and is defined as:

$$D(q||r) = \sum_y q(y) log \frac{q(y)}{r(y)}$$

We use JS Divergence to rank the out-of-domain sentences as follows:

$$-(JS_I(\sigma_{s_i}) - JS_O(\sigma_{s_i}))$$

where $JS_I(\sigma_{s_i})$ is the JS divergence between sentence representation σ_{s_i} and the in-domain representation. $JS_O(\sigma_{s_i})$ is the JS divergence to the out-of-domain data.

In practice, we use bilingual JS Divergence to compute the similarity metric, partially inspired by (Axelrod et al., 2011; Wang et al., 2017a; Wang et al., 2017b). Bilingual JS Divergence indicates that we takes into account both similarities on the source and target side of the corpus, which is calculated as follows.

$$\begin{aligned}\alpha_i = JS_{Osrc}(\sigma_{s_i}) - JS_{Isrc}(\sigma_{s_i}) \\ + JS_{Otrg}(\sigma_{s_i}) - JS_{Itrg}(\sigma_{s_i})\end{aligned}$$

$w(s_i)$ is a normalized output from a Min-Max Normalization (Priddy and Keller, 2005) over α_i. $w(s_i)$ estimates the weight of sentence s_i, in range of [0,1].

$$w(s_i) = \frac{\alpha_i - min(\{\alpha_i\}_{i=1}^N)}{max(\{\alpha_i\}_{i=1}^N) - min(\{\alpha_i\}_{i=1}^N)}$$

To train NMT with the weighted sentences, we update the objective function as follows:

$$J^* = \sum_{(x,y)\in D} w(s) log(y|x)$$

4.2 Sentence Weighting for NMT Trained with Only Synthetic Parallel Data

In domain adaptation, we measure the similarity between sentences in out-of-domain and the in-domain. We also test our sentence weighting method on synthetic parallel data. The existing pseudo parallel corpora is constructed with true and synthetic sentence pairs. The pseudo

IWSLT ZH-EN	#sentences
TED training (in-domain)	210k
LDC training	1.25M
TED dev 2010	0.9k
TED tst 2010	1.5k
TED tst 2011	1.4k
TED tst 2012	1.7k
TED tst 2014	1.3k

Table 1: Statistcs of the data used for the IWSLT task

parallel corpora can be source-originated, target-originated or mixture of them. In the low-resource domain translation task , we evaluate the similarity between sentences in training data and the development set. We use the bilingual JS Divergence method as follows:

$$\alpha_i = -JS_{Isrc}(\sigma_{s_i}) - JS_{Itrg}(\sigma_{s_i})$$

We regard all training data in the low-resource domain as out-of-domain data. To improve the translation performance, we apply a different cost weighting method by adding 1 to the normalized probability (Chen et al., 2017). This is to give the original training data a bonus to some degree. We use the following objective function instead:

$$J^* = \sum_{(x,y)\in D} (1 + w(s))log(y|x)$$

5 Experiments

The proposed methods were evaluated on two scenarios: standard domain adaptation and translation with synthetic parallel data which can be cast as a domain adaptation task. We will detail the experiment settings and results in this section.

5.1 Domain Adaptation

5.1.1 Data

For the Chinese-to-English translation domain adaptation task,we used an LDC news corpus as the out-of-domain data for the IWSLT 2017 (mainly contains TED talks) target domain corpus. The LDC bilingual corpus contains 1.25M sentence pairs extracted from LDC corpora, with 27.9M Chinese words and 34.5M English words. The IWSLT 2017 in-domain corpus contains 210k and 0.9k sentence pairs, for training and development set, respectively. We chose the TED TST2010, TED TST2011, TED TST2012, TED TST2014 datasets as our test sets. Statistics on the data of this task are shown in Table 1.

5.1.2 Setting

We used the case-insensitive 4-gram NIST BLEU score as our evaluation metric (Papineni et al., 2002) and the script 'mteval-v11b.pl' to compute BLEU scores. For the efficient training of the neural networks, we limited the source (Chinese) and target (English) vocabularies to the most frequent 30k words, covering approximately 97.7% and 99.3% words of the two corpora respectively. All the out-of-vocabulary words were replaced with a special token UNK. The dimension of word embedding was 620 and the size of the hidden layer was 1000. All other settings were the same as in (Bahdanau et al., 2014). The maximum length of sentences that we used to train the NMT model in our experiments was set to 50, for both the Chinese and English sides. Additionally, during decoding, we used the beam-search algorithm and set the beam size to 10. The model parameters were selected according to the maximum BLEU points on the development set.

Methods	tst2010	tst2011	tst2012	tst2014	Avg
In+Out	17.26	18.73	17.43	16.1	17.38
Wang et al. (2017b)	17.1	19.35	17.26	16.69	17.6
JS	17.57	19.74	17.84	17.78	18.23(+0.63)
ED	18.81	20.45	18.81	18.02	19.02(+1.42)

Table 2: IWSLT Chinese-English results

5.1.3 Results

Experiment results are shown in Table 2. "In+Out" indicates that the NMT training corpus is the mixture of in-domain and out-of-domain data. We also compared Wang et al. (2017b)'s sentence weighting method (hereafter WM), which uses a language model to score sentences. "JS" and "ED" indicate the methods that we proposed in Section 4.1 with similarity measured by Jensen-Shannon Divergence and Euclidean distance functions, respectively.

Our "JS" method achieves improvements over NMT (In+Out) by 0.31 - 1.68 BLEU. It also outperforms Wang et al. (2017b)' s sentence weighting method by 0.39 - 1.09 BLEU, and achieves an average gain of 0.63 BLEU over WM. Our "ED" method is much better than NMT (In+Out) by 1.38 - 1.92 BLEU, outperforming Wang et al. (2017b)' s sentence weighting method by 1.1 - 1.71 BLEU, and achieves an average gain of 1.42 BLEU over WM.

5.2 NMT with Only Synthetic Parallel Data

5.2.1 Data and Setting

We crawled Chinese and English monolingual sentences from E-commerce websites, including JD.com, Suning.com, Ebay and Amazon.com. We translated the crawled Chinese sentences into English and English sentences into Chinese using Google Translate API. This process provided us with a pseudo parallel corpus where either the source or the target side is correct. The final corpus contains 720k sentence pairs after filtering. We used this crawled and machine-translated corpus as our bilingual training data for E-commerce experiments. we also used BPE to process the training data.

The test and development set are from Alibaba. The development set contains 500 sentences and the test set contains 1,500 sentences. All model parameters are the same as indicated in Section 5.1.2, except for using Adadelta optimizer.

5.2.2 Results

The baseline is a system trained with the entire pseudo parallel corpus. We also apply Wang et al. (2017b)' s method in our experiment. The "JS" method obtains 0.7 BLEU points over the baseline, and outperforms WM by 0.44 BLEU points. The "ED" method is also better than WM.

In addition, we introduce a weighting and stacking training method that is inspired by model stacking method as described in Section 2. We trained several models in an online fashion, in the order of the similarity between the training corpus and the development set. We employed our sentence weighting approach described in Section 4 to calculate weights for sentences, and order training sentences by their weights. We start training the model on the entire training data. Similar to the model stacking method, we continue to train the model with half data of the ordered training corpora in the next epoch. Then we continue the training process with the quarter data in the same way. This weighting and stacking method outperforms the baseline by 1.15 BLEU points on the test set.

Methods	dev	tst
Baseline	15.46	13.35
Wang et al. (2017b)	16.04	13.61
JS	15.17	14.05
ED	15.34	13.73
Weighting and Stacking	15.36	14.50

Table 3: E-commerce Chinese-English results

SRC	各位女同胞们，大家好！你们还好吗?
REF	Hello, TEDWomen, what's up.
In+Out	ladies and gentlemen , hello ! are you okay ?
Wang	ladies and gentlemen , everybody ! are you good ?
JS	you guys , everyone ! are you okay ?
SRC	如果有什么类似受压迫奥运会，我肯定能拿金牌。
REF	if there was an Oppression Olympics, I would win the gold medal.
In+Out	if anything like the <UNK> Olympics, I will have a gold medal.
Wang	if there's something like the <UNK> Olympics, I certainly have a gold medal.
JS	if there were any similar olympic games, I would certainly take the gold medal.
ED	if there was something like the <UNK> Olympics, I would have a gold medal.

Table 4: Examples from the test set. Header "SRC" denotes source sentences. Header "REF" denotes reference translations. "In+out" represents the translations generated by "in+out" method, "Wang" by the sentence weighting method proposed by Wang et al. (2017b).

6 Analysis

6.1 Analysis on Sentence Representations

Evidence of the efficacy of the proposed similarity measure method can be visualized in Figure 1. In this figure, each symbol indicates one sentence which is represented by the normalized sentence embedding output σ_{s_i} from a softmax operation. We sample the out-of-domain (the LDC corpus) source sentences into 3 groups. Each group contains 20 sentences whose weights are around 0, 0.5 and 0.9 respectively. We also randomly select 20 sentences from the in-domain data.

From this figure we can see that the in-domain sentences are around the center of the in-domain, so do the out-of domain sentences. The visualization confirms our hypothesis in that the average of sentence embeddings represents the core of the domain (Section 4.1). What's more, we can find that the weight generated by our domain similarity method is reasonable. Sentences assigned similar weights are grouped together and far apart from those with different weights. In addition, the more similar the representation of a sentence to the core of in-domain, the higher the weight assigned to it.

6.2 Analysis on Translation Examples

We show two translation examples in Table 4 to provide a deep look into how the proposed method help adapt the model to the in-domain data. In the first example in Table 4 , the reference sentence has an informally utterance, where the greeting "what's up" indicates that it is in an informal atmosphere. As depicted in the Table 4, the greeting "ladies and gentlemen" translated by "In+Out" and "Wang et al. (2017b)" is used on formal occasions. "you guys" in "JS" method is uesd in an informal occasion. The "JS" method is able to adapt the more official LDC style to the spoken language domain.

The source and reference sentences in the second example in Table 4 are expressions in sub-

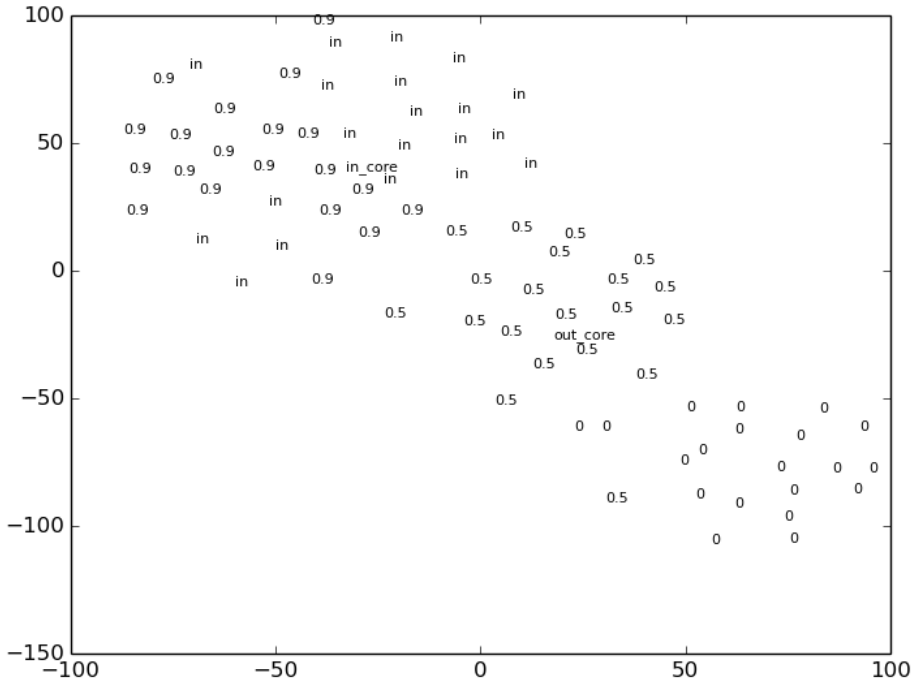


Figure 1: Visualization of sentence representations of both in-domain and out-of-domain examples by t-SNE. The symbols of "in", 0, 0.5, 0.9 in the figure represent the in-domain sentences, out-of-domain sentences with weights 0, 0.5 and 0.9, respectively. The symbol "in_core" denotes the in-domain center and "out_core" indicates the out-of-domain center.

junctive mood, which describes things that impossibly happen in real life. The translations of both "ED" and "JS" use the subjunctive mood and express the coincident meaning with the reference sentences. On the other hand, both "in+out" and "Wang et al. (2017b)" methods don't learn the right moods. As our LDC training data are in the news domain, sentences in this corpus seldom use the subjunctive mood. In the spoken language domain, the utterance is more free. Both the "ED" and "JS" methods have successfully adapted the model to the spoken language domain.

6.3 Analysis on the Terminology Adaptation

In domain adaptation, one of our goal is to extend generic NMT models to cover terminologies and styles in the target domain(Kobus et al., 2017). We analyse the distribution of terminologies on the IWSLT Chinese-English experiment. We calculate the proportion of in-domain and out-of-domain terminologies with respect to all words in the test set. We obtain terminology vocabularies according to differences of two sequences of words in both domains. The more in-domain terminologies and the fewer out-of-domain terminologies contained in the translations of the in-domain test sets, the better the domain adaptation is. The results are displayed in Table 5. The "ED" method with the best performance in terms of BLEU has successfully decreased the ratio of out-of-domain terminologies from 0.37% to 0.15% and increased the ratio of in-domain terminologies from 44.28% to 45.37%, indicating its ability to terminology adaptation.

7 Conclusions and Future Work

In this paper, we have proposed a sentence weighting method for neural machine translation on two different scenarios. We calculate the similarity of an out-of-domain sentence to the in-

Methods	ratio of O-D terminology(%)	ratio of I-D terminology(%)
In+Out	0.37	44.28
Wang et al. (2017b)	0.28	42.97
JS	0.20	43.80
ED	0.15	45.37

Table 5: The proportion of domain terminologies in the test sets of each method. "O-D" in header denotes out-of-domain. "I-D" represents in-domain.

domain with a variety of methods, including JS divergence and ED. The computed similarity scores are further incorporated into the training objective to weight sentences. The proposed method has obtained an achievement on both domain adaptation and NMT models trained with only synthetic parallel data where there is some mismatch between the training and test data. In the future, we would like to explore more accurate measures of domain similarity. We are also interested in the study of other weighting methods for NMT domain adaptation.

Acknowledgements

The present research was supported by the National Natural Science Foundation of China (Grant No. 61622209).We would like to thank three anonymous reviewers for their insightful comments and Alibaba for providing the test and development sets for our E-commerce experiments.

References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the conference on empirical methods in natural language processing*, pages 355–362. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Boxing Chen, Colin Cherry, George Foster, and Samuel Larkin. 2017. Cost weighting for neural machine translation domain adaptation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 40–46.

Catherine Kobus, Josep Crego, and Jean Senellart. 2017. Domain control for neural machine translation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 372–378. INCOMA Ltd.

Lillian Lee. 2001. On the effectiveness of the skew divergence for statistical language analysis. In *AISTATS*.

Jianhua Lin. 1991. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 76–79.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

Barbara Plank and Gertjan Van Noord. 2011. Effective measures of domain similarity for parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1566–1576. Association for Computational Linguistics.

Kevin L Priddy and Paul E Keller. 2005. *Artificial neural networks: an introduction*, volume 68. SPIE press.

Sebastian Ruder, Parsa Ghaffari, and John G Breslin. 2017. Knowledge adaptation: Teaching to adapt. *arXiv preprint arXiv:1702.02052*.

Hassan Sajjad, Nadir Durrani, Fahim Dalvi, Yonatan Belinkov, and Stephan Vogel. 2017. Neural machine translation training in a multi-domain scenario. *arXiv preprint arXiv:1708.08712*.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *In Proceedings of international conference on spoken language processing. Denver, Colorado, USA, September 2002*.

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

Rui Wang, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. 2017a. Sentence embedding for neural machine translation domain adaptation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 560–566.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. 2017b. Instance weighting for neural machine translation domain adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1482–1488.

Quantifying training challenges of dependency parsers

Lauriane Aufrant[1,2], **Guillaume Wisniewski**[1] and **François Yvon**[1]

[1]LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, 91 405 Orsay, France
[2]DGA, 60 boulevard du Général Martial Valin, 75 509 Paris, France
{lauriane.aufrant,guillaume.wisniewski,francois.yvon}@limsi.fr

Abstract

Not all dependencies are equal when training a dependency parser: some are straightforward enough to be learned with only a sample of data, others embed more complexity. This work introduces a series of metrics to quantify those differences, and thereby to expose the shortcomings of various parsing algorithms and strategies. Apart from a more thorough comparison of parsing systems, these new tools also prove useful for characterizing the information conveyed by cross-lingual parsers, in a quantitative but still interpretable way.

1 Introduction

10 annotated sentences suffice to train a dependency parser that can correctly predict around 70% of the dependencies in French and Italian, but also in Japanese, Greek, Urdu and other languages. This surprising fact raises questions on how to interpret such numbers: what do those 70% dependencies look like? Are they evenly distributed or do they correspond to specific dependency structures?

This work is based on the intuition that in any treebank, some dependencies are intuitively easy to learn (typically nearly deterministic attachments like the dependencies between a noun and its determiner), while others seem much harder to predict, corresponding to complex semantic attachments or long lists of exceptions in the underlying grammar of the language. We thus propose one way to investigate that distinction, by evaluating the amount of training material needed to learn each kind of dependencies: with that viewpoint, a dependency structure is qualified as 'easy to learn' when it can be learned from only a few examples and as 'difficult to learn' otherwise. In this paper, we formalize this idea and propose empirical ways to measure the difficulty to learn certain dependencies.

Our metric proposal can be useful as an analysis tool for comparing existing parsers, which we illustrate both in monolingual and cross-lingual settings, but also to gain some insight on the information reliably conveyed by parsers trained on a few sentences. Ulinski et al. (2016) have indeed shown that such parsers can be successfully leveraged to speed up the manual annotation process. This observation confirms that they provide *some* useful content to the annotators, which our formalism can help characterizing beforehand.

The rest of this paper is organized as follows. Section 2 presents the dependency parsing task and the parsers used in this study. Our method for estimating difficulty is introduced in Section 3. We then apply these newly designed tools to conduct two analyses: a detailed comparison of several state-of-the-art parsers (Section 4), as well as a fine-grained evaluation of cross-lingual transfer (Section 5).

2 Dependency parsing

The purpose of the dependency parsing task is to predict, for each token in a sentence (the child), the token it depends on (its head), and thereby to build a tree structure over the sentence. There are several approaches to build such parsers, mostly corresponding to two categories: transition-based parsers

Proceedings of the 27th International Conference on Computational Linguistics, pages 3191–3202
Santa Fe, New Mexico, USA, August 20-26, 2018.

(which score and apply locally a sequence of transitions affecting the parser's inner state) and graph-based parsers (which compute attachment scores for all pairs of tokens and then optimize the sentence score globally).

In this work, we consider three dependency parsers, based on diverse parsing algorithms and classifiers, to assess the generality of our findings: UDPIPE (transition-based parser, with a feedforward neural classifier, i.e. embedding-based), BEAM (transition-based parser, with an averaged perceptron classifier, i.e. feature-based), and MSTPARSER (graph-based parser). We use version 0.5.1 of MSTPARSER (McDonald et al., 2005) with default parameters. For UDPIPE, we use version 1.1 (Straka and Straková, 2017) with the same hyperparameters as Straka (2017), but without the word embeddings pre-trained on massive monolingual data (to ensure comparability). For BEAM, we rely on our own open source[1] implementation, PANPARSER (Aufrant and Wisniewski, 2016), using the ArcEager version (Nivre, 2004) with a dynamic oracle (Goldberg and Nivre, 2012) adapted for beam search (Aufrant et al., 2017) and the feature sets of Zhang and Nivre (2011) (coarse PoS, no labels).

3 Measuring difficulty

The purpose of this work is to investigate dependencies for the learning of which large training datasets are unnecessary, and which can therefore be qualified as 'easy to learn'. In this section, we formalize this intuition and introduce several empirical measures to quantify it. We then exploit the results of large-scale evaluations to design a new metric, COMPLEXITY, which estimates the challenges faced when learning a given 'type' of dependencies: by departing from individual dependencies, we aim at discovering higher-level properties related to language-independent syntactic phenomena (like the simplicity of annotating determiners, independently of the language and the parsing system).[2]

As our focus is on how parsers exploit the first examples of a training set, even just 10 sentences, we consider in this work only treebanks under 500 sentences.[3]

For all experiments, we use the Universal Dependencies 2.0 dataset (Nivre et al., 2017b; Nivre et al., 2017a), containing 73 treebanks covering a wide array of language families. We first downsize each treebank, when possible, to the 500 first training sentences,[4] and resample smaller trainsets of increasing sizes.[5] Since training is significantly unstable at that scale, all reported scores are averaged over five random samplings, thereby amounting to 5,880 parsers trained on 56 languages. UAS evaluation is performed using gold PoS tags,[6] excluding punctuation.

3.1 Class-level learning rate

Dependency *classes* refer, in the following, to any criterion describing a subset of dependencies (and by extension, a subset of child tokens) in a treebank. The definition of a class can, for instance, be based on child-head distance, in-tree depth, edge direction, dependency label, empirical values like frequency, PoS tags, etc. Even though our method applies to any such criterion, we focus in this section on classes defined by the child PoS and its attachment direction. Indeed, it fits particularly well our intuition regarding many parsing difficulties: due to its frequency and determinism, the $\overset{\curvearrowleft}{\text{ADJ}}$ class (that is, all adjectives whose head is on the right) appears for instance simple in the English UD as it mostly corresponds to the bigram 'ADJ NOUN' and, sometimes, to predicative adjectives. On the contrary, the

[1] Source code available at `https://perso.limsi.fr/aufrant`.

[2] Such properties can still depend on the annotation scheme though, as repeatedly pointed out in the literature (Schwartz et al., 2012; Wisniewski and Lacroix, 2017; Wisniewski et al., 2018).

[3] This size has been chosen to cover both the scale of 10 sentences and that of existing treebanks, around 300 sentences: there have been publications with 300 Irish sentences (Lynn et al., 2012) or 371 in Slavomír Čéplö's Maltese treebank (Tiedemann and van der Plas, 2016).

[4] We similarly downsize the validation sets, used only for early stopping, to their first 10 sentences. We do not alter the test sets. We only experiment on the 56 treebanks that are large enough to apply these sampling procedures. `ar_nyuad`, whose complete data is under license, is also excluded.

[5] Resulting trainsets contain 5, 10, 20, 50, 100, 200 and 500 sentences.

[6] While less representative of real-world processing capacities (Tiedemann, 2015), we believe this choice to be crucial in such studies focusing on syntactic properties, whose measurement would otherwise be biased by properties of the taggers and language-dependent vocabulary issues.

attachment decisions on $\overset{\curvearrowright}{\text{ADJ}}$ tokens seem more complex, first of all because the PoS of the head is uncertain (sometimes a VERB, a NOUN, another ADJ, etc.). What remains to ascertain is whether this simple/complex distinction can relate to measurable properties.

Our first experiment aims at studying the rate at which the different dependency classes are learned. In this experiment, we are not interested in the absolute scores over each class, but rather in how fast the *available* information (regarding a given class) can be extracted from a treebank: has everything been already learned in the 10 first sentences? Would 100 more sentences be really informative? Consequently, all measures are performed in terms of UAS normalized by its maximum, in order to set aside the differences due only to the inherent accuracy of a class. In other terms, for class C (e.g. $\overset{\curvearrowright}{\text{DET}}$) and treebank size s (e.g. 50 sentences), we measure $\text{UAS}_s[C]$, the number of tokens in class C whose head has been correctly predicted based on a treebank of size s, and then compute $\frac{\text{UAS}_s[C]}{\text{UAS}_{500}[C]}$. These values are computed separately for each language, and then averaged over all languages[7] to expose language-independent trends, i.e. searching for properties of e.g. determiners as a universal category, while ignoring their idiosyncratic uses in individual languages (e.g. their scarcity in Latin, the annotation scheme for French multi-word expressions containing determiners, etc.).

Figure 1 presents the resulting learning curves for the UAS broken down by class, as well as for the overall score in each language (using BEAM parsers). For legibility (notably around 10 sentences), but also to consider equally the learning speed at all size scales, it is displayed with a logarithmic scale. In this view, the slope of the curve can be interpreted as the marginal utility of doubling the treebank size.

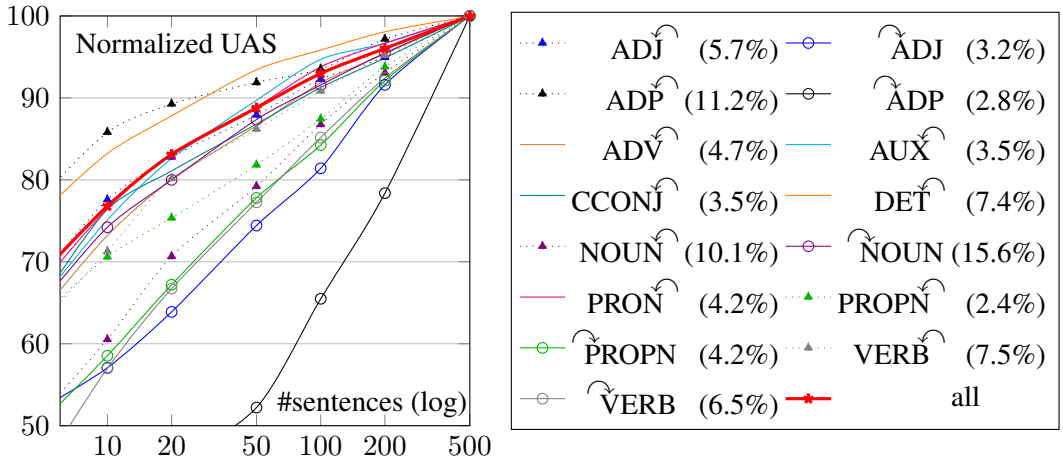


Figure 1: Learning curves of UAS by PoS/direction class. $\overset{\curvearrowright}{\text{P}}$ (resp. $\overset{\curvearrowleft}{\text{P}}$) means all dependencies whose child PoS is P and reference head is on the left (resp. right, including roots). Percentages indicate the size of each class; the rarest ones are not represented.

Unsurprisingly, all fine-grained curves and the overall one are roughly linear: even though they rise quickly to high accuracies, subsequent examples tend to contribute less and less to accuracy, so that the dataset must be doubled each time to achieve substantial UAS improvements. This supports the idea that, overall but also *for each class*, most knowledge is extracted from the first few examples, and additional data is poorly informative.

What is more intriguing in Figure 1 is that the relative curves can clearly be partitioned into two groups,[8] quickly-converging curves and slower-increasing ones, separated by a significant margin. This observation is consistent with our intuition whereby there are two kinds of dependency classes, the simple and the complex ones: as expected, $\overset{\curvearrowright}{\text{DET}}$ and $\overset{\curvearrowleft}{\text{AUX}}$ for instance belong to the former, i.e. the classes for which most of the available information can already be found in the first examples.

[7] To ensure reliable scores, we exclude the treebanks whose test set contains less than 30 occurrences of the considered class.

[8] Because Figure 1 has 3 outliers, it contains in fact 4 groups, but such detailed distinctions are out of the present scope, and the two retained groups are the curves which start at 70% of their final value, and those which start around 50%.

One explanation to these differences in learning rate could be the frequency of classes: those which occur more often are more likely to appear a lot in the first examples, which makes them mechanically easier to learn. However, the ranking of the curves does not seem fully consistent with the size of the corresponding classes: DET has a steeper curve than ADJ, which is itself steeper than NOUN, although ADJ is the least frequent of all three.

Evidence rather suggests that this ranking relates to linguistic properties (like word order entropy), complemented by phenomena of the 'rule versus exception' kind: among competing classes (with different directions for the same PoS, e.g. ADJ versus ADJ), there is systematically one simple and one complex class, in general according to their frequency – but not always, as shown by PROPN. A thorough investigation to explain learning speed differences is however out of the scope of this work, which focuses on measuring them.

3.2 Complexity measures

Although the aforementioned experiment has already revealed interesting trends, a systematic evaluation of complexity requires to go beyond visual reading of curve shapes. But their slope is not properly defined, considering that for several classes, learning speed can show significant variations between 5 and 500 sentences. We consequently aggregate the curve shape into a single metric by computing the area under the curve; yet we calibrate it on the average curve to enable cross-treebank comparison.[9] Formally, we define the COMPLEXITY of a class as the signed area[10] between the learning curve of the (normalized) overall score and that of the class. The logarithmic scale is kept to ensure that the slope at the smallest scale significantly contributes to the metric.[11] A negative COMPLEXITY (curve above average) means that training on this class is simple (most knowledge is embedded in the first few examples), a positive value denotes a complex class (compared to the typical complexity of the treebank). From now on, the terms *simple* and *complex* are respectively used to denote classes with negative and positive COMPLEXITY.

BEAM																	
COMPLEXITY	ADP	DET	PRON	AUX	ADJ	CCONJ	N	ADV	V	PN	SCONJ	N	PN	V	ADJ	AUX	ADP
	-18.8	-18.7	-0.6	0.2	1.9	6.3	7.6	9.6	12.6	23.4	35.0	42.0	49.5	52.5	57.7	68.0	131.2
UAS_{500}	DET	ADP	AUX	PRON	SCONJ	ADJ	CCONJ	ADV	V	PN	PN	N	N	ADJ	V	AUX	ADP
	91.3	89.0	83.9	82.4	80.2	80.0	77.1	76.1	75.1	69.0	68.4	68.2	67.9	60.6	56.4	52.8	48.0
UDPIPE																	
COMPLEXITY	ADP	AUX	CCONJ	DET	ADV	ADJ	PRON	PN	N	V	AUX	N	SCONJ	ADJ	ADP	PN	V
	-33.3	-24.5	-5.3	-3.7	-2.3	0.7	3.3	12.5	25.3	26.0	28.3	35.5	35.9	45.1	51.0	51.8	75.1
UAS_{500}	DET	ADP	AUX	SCONJ	PRON	ADJ	CCONJ	ADV	V	PN	N	N	PN	ADJ	V	AUX	ADP
	89.9	89.0	84.3	81.0	80.3	79.9	76.1	74.6	73.9	73.9	69.9	65.6	65.0	56.9	53.8	44.4	42.2
MSTPARSER																	
COMPLEXITY	ADP	DET	ADJ	V	CCONJ	PRON	PN	ADV	N	AUX	N	ADJ	SCONJ	PN	AUX	V	ADP
	-27.7	-24.1	-15.4	-3.5	6.2	8.7	13.1	13.8	23.4	28.9	34.2	47.4	53.8	55.5	56.7	83.8	89.1
UAS_{500}	DET	ADP	AUX	ADJ	PRON	V	CCONJ	ADV	SCONJ	N	PN	PN	N	ADJ	AUX	V	ADP
	90.6	89.3	80.8	79.8	79.6	76.0	74.9	74.3	72.7	67.8	67.2	67.1	64.7	57.0	56.4	50.6	50.5

Table 1: Comparison of the COMPLEXITY and UAS rankings for all 3 systems. For each system, the classes with the largest difference between both rankings are highlighted in blue. N, PN and V stand for NOUN, PROPN and VERB.

Beyond a sign-based grouping criterion, these quantitative measures also provide an interesting diffi-

[9] This aggregation step is inspired by Zubek and Plewczynski (2016)'s work on data complexity, which is close in spirit to ours (with the same intuition that a dataset is simple if most information can be found in the first few examples) but explores a different setting: in their work, they measure the similarity of subsets of increasing sizes to the dataset itself (without involving a prediction step) and aggregate these values to estimate the complexity of that dataset.

[10] This can be computed easily using any numerical integration method. In the following experiments, we use Simpson's rule.

[11] In these experiments the area is computed between 5 and 500 training sentences, but another low-resource range could be chosen, provided that it is the same for all treebanks.

culty ranking. Table 1 compares the rankings based on UAS (using 500 training sentences) and COMPLEXITY. Overall, the results remain consistent with intuition: the simplest classes are closed PoS classes with very deterministic attachments (DET$^{\curvearrowleft}$, ADP$^{\curvearrowleft}$), the most complex ones are rare ($^{\curvearrowright}$ADP) or semantics-driven attachments ($^{\curvearrowright}$VERB), while classes of average difficulty are mostly NOUNs and other open classes. But differences between both metrics still appear clearly, thereby confirming that they capture different properties. For instance, high accuracies are achieved on SCONJ$^{\curvearrowleft}$, but getting there is cumbersome as it requires many training examples. Conversely, UDPIPE does not learn much about the $^{\curvearrowright}$AUX class (whose UAS is particularly low), but it does so almost immediately (as shown by its low complexity).

Finally, we perform similar computations on each treebank separately:[12] they reveal that, even though some classes have emerged as consistently simple across languages, this property can still depend on the language (sometimes even on the corpus). For instance, ADJ$^{\curvearrowleft}$ is simple and $^{\curvearrowright}$ADJ is complex in the English treebank, but in French it is the opposite, which is consistent with the respective frequencies of those classes (and thus with the initial intuition). Similarly, $^{\curvearrowright}$ADP is complex for all but 6 treebanks (from which the competing class ADP$^{\curvearrowleft}$ is virtually absent), and DET$^{\curvearrowleft}$, whose attachments are highly deterministic, is a simple class for all but 7 treebanks (mostly Old Church Slavonic, Arabic, Latin-PROIEL, but also Basque, Estonian, Korean and Polish to a lesser extent).

4 Application 1: fine-grained comparison of parsers

The metrics we have proposed can now be used for large-scale computation of fine-grained evaluations, and therefore detailed comparison of parsers. This newly defined notion of complexity opens indeed new evaluation perspectives, as a complement to the more explicit properties (length, PoS tags, projectivity, etc.) used in prior work on comparative error analysis (McDonald and Nivre, 2007).

Complexity variations Coming back to Table 1, comparing the rankings between all 3 systems also emphasizes on their respective shortcomings. UDPIPE notably seems to have troubles with determiners: not only does it achieve a lower score on DET$^{\curvearrowleft}$ dependencies, but it is also much slower learning those, compared to the other systems; UDPIPE consequently appears to under-exploit the determinism of that class.[13] It is conversely particularly efficient on CCONJs. As for MSTPARSER, it handles VERB$^{\curvearrowleft}$ significantly faster than BEAM and UDPIPE, presumably because it does not rely on mostly local features, as transition-based parsers do. Regarding the BEAM system, its main particularity is its efficient handling of AUX$^{\curvearrowleft}$ attachments, even though their UAS is around the same. Overall, according to Figure 2, COMPLEXITY correlates well with UAS_{500}, at least for BEAM and MSTPARSER (Spearman's ρ = -.886 and -.882); but it is hardly the case for UDPIPE (ρ = -.576), which points out at least a difference, and maybe an issue, which remains to ascertain. All those remarks provide a valuable feedback on the inner workings of each parsing system, and thereby new insights on possible issues. The COMPLEXITY metric thus provides a promising way to analyze and improve parsing algorithms in general.

Composite scores A more systematic way to study fine-grained differences between parsers is to investigate composite scores. However, for large inventories of classes, class-level analysis can be tedious and hide the main trends. For this reason, we advocate aggregating that information into 2 scores only, as a trade-off between simplicity and detailed analysis. Contrasting differences among parsers along two categories, simple and complex dependencies in the present case, is indeed already much more informative than relying on a single metric.

Table 2 reports the average UAS of the 3 parsers, when computed separately on simple and complex classes (that is, depending on the sign of COMPLEXITY), and for various data sizes. It appears that all

[12] In other words, we compute the area under a specific learning curve, instead of the area under the average of all learning curves.

[13] As UDPIPE is the only neural parser in our experiments, this differential treatment of deterministic classes may be related to the choice of classifier; concluding on such properties is out of the scope of this paper, though.

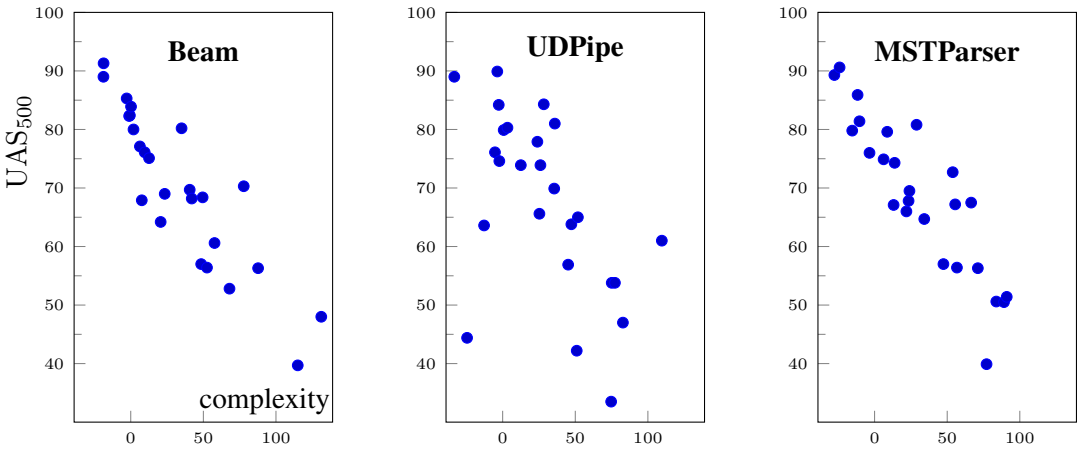


Figure 2: UAS of all 3 systems, with 500 training sentences, for classes of increasing complexity.

systems present a significant score difference (up to 30 points) between both categories, in favor of the simple one, in particular for tiny data (i.e. 10 sentences).

It is important to note that this result was not expected: it is true that UAS measures are involved when computing COMPLEXITY, but all these scores are first normalized. Therefore, the simple classes are not *by design* those with high accuracies on tiny data, despite what the results suggest.[14] In this regard, the score difference between both groups is an actual finding. Besides, the score gap is mostly maintained for full datasets, which means that the classes that have been singled out using tiny data have truly different properties (or at least are handled differently by the systems), whatever the data size.

	UAS_{10}			UAS_{500}			$UAS_{\text{full UD}}$		
	simple	overall	complex	simple	overall	complex	simple	overall	complex
UDPIPE	52.8	42.5	28.4	81.8	74.7	65.2	87.6	83.2	**77.3**
MSTPARSER	64.4	52.8	36.9	82.7	75.1	64.9	**88.2**	**83.4**	77.1
BEAM	**71.1**	**59.0**	**42.7**	**82.9**	**76.1**	**67.1**	87.3	82.6	76.4

Table 2: UAS of the 3 studied systems, broken down by simple and complex PoS/direction classes, and for various data sizes. The partitions and corresponding scores are computed separately for each language, before averaging. All partitions are computed using BEAM, so that the absolute scores remain comparable (best system in bold). 'Full UD' trainsets contain between 598 and 68,495 sentences.

When comparing each system's results, the dynamics between them become clear: while BEAM prevails across the categories around 10 sentences, MSTPARSER catches up first on simple dependencies (for 500 sentences), then also on the complex ones (for full UD). As for UDPIPE, it performs poorly with 10 training sentences, but then gradually reveals its worth for parsing specifically complex classes (which is consistent with its difficulties with determiners: this system is not so relevant for deterministic dependencies, its value is elsewhere).

5 Application 2: fine-grained evaluation of cross-lingual parsers

Apart from comparing systems, our proposal can also be used to compute reference values when evaluating other kinds of parsers, typically cross-lingual parsers. Cross-lingual transfer is an approach to process low-resourced languages, whereby resources available in other languages (the sources) are leveraged to

[14]There remains however an indirect impact of absolute scores on COMPLEXITY: the classes that reach already high scores (over 80 UAS) with only 10 sentences are doomed to be simple classes, because being complex would imply achieving more than 100 UAS with 500 sentences. So, there is indeed a score bias towards simple classes. But such classes with early high scores represent a minority of the dependencies (19% on average), and there are many other kinds of simple classes (for a total of 57% of dependencies), including poorly accurate ones.

build new systems in the language of interest (the target). The main methods involve typically projecting information through word-aligned parallel data (Yarowsky and Ngai, 2001), or training source models on delexicalized data (that is, using PoS information only) and directly applying them on the target side (Zeman and Resnik, 2008). The link with our work above is twofold: cross-lingual parsers also focus on low ranges of training sizes, and at the same time many of them yield UAS around the range covered by our 5 to 500-sentence long treebanks. We therefore propose to exploit our upper results in this new frame, with the goal of quantifying and characterizing the amount of knowledge that has been effectively transferred: what kind of information is learned by cross-lingual parsers – only simple classes or complex knowledge about non-trivial classes?

Multi-source weighted delexicalized transfer Our analysis first focuses on Rosa and Zabokrtsky (2015)'s state-of-the-art method for cross-lingual parsing: it consists in delexicalized transfer, where the hypotheses stemming from multiple sources are weighted and combined based on the KL_{cpos^3} metric (the Kullback-Leibler divergence of PoS trigram distributions between the source and the target). It is meant to favour the languages that are syntactically close to the target, while still benefiting from the diverse information conveyed by a large set of sources.

We reimplement the method of Rosa and Zabokrtsky (2015) on top of the BEAM system: we include as sources the delexicalized BEAM models based on all 56 (full) treebanks except those covering the same language. All scores are again averaged over 5 different trainings. To be fair regarding the available target resources, the KL_{cpos^3} metric is computed using the whole source treebank but only 10 target sentences. In the following, this strategy is referred to as KL-BEAM.

Composite scores Similarly to what has been done for monolingual parsers, we can express the performance of KL-BEAM in terms of simple-only UAS and complex-only UAS (using the partition computed monolingually with BEAM for instance). Those values alone are, however, hard to interpret, and notably to relate to the actual amount of knowledge transferred via KL-BEAM.

We therefore propose to position those scores along the learning curves of the monolingual parser, following Aufrant et al. (2016): if the cross-lingual parser achieves the same score as a parser trained on n sentences, we consider that the amount of transferred knowledge is the amount of knowledge contained in n sentences. Figure 3 consequently pictures the learning curves of each system on simple and complex classes (using the PoS/direction criterion), as well as the split UAS for KL-BEAM on the same categories.

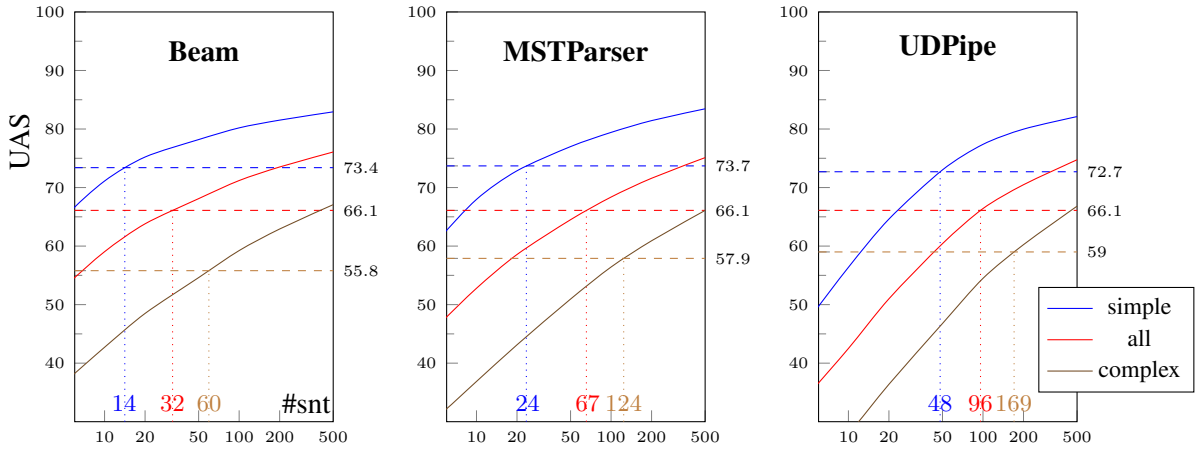


Figure 3: Overall and simple/complex UAS of all 3 parsers with increasing training sizes. The partitions and corresponding scores are computed separately for each language and system, before averaging. Dashed lines correspond to the overall, simple and complex UAS for KL-BEAM, using the partition computed for each system. Dotted lines represent the corresponding numbers of annotated training sentences.

Overall, KL-BEAM achieves 66.1 UAS on average, which corresponds to what could have been achieved by BEAM with 32 sentences. In other words, if more than 32 target sentences are available,

on average one should prefer monolingual parsing (at least when using BEAM) over KL-BEAM transfer:[15] the amount of knowledge transferred across languages seems very limited.

However, when considering the simple/complex scores, it appears that their difference is less pronounced for the cross-lingual parser than the monolingual ones, which results in distinct data-size equivalents. Delexicalized transfer thus appears much more useful for complex classes (score equivalent to 60 sentences) than for simple classes, for which few information was transferred (score equivalent to 14 sentences). This means that KL-BEAM is qualitatively better than a BEAM trained on 32 sentences: it is able to convey non-trivial information, which otherwise would have only been accessible by collecting more data.

As UDPIPE and MSTPARSER are less efficient on very small treebanks, the data-size equivalents are much higher, but the simple/complex difference remains similar in proportions. We additionally perform similar experiments with other class criteria and obtain similar results, although the simple/complex gap marginally fluctuates: for BEAM, we measure 20 versus 43 sentences (74.6 and 58.9 UAS) when considering only the PoS, 15 versus 60 sentences (74.6 and 56.4 UAS) when considering the relation label only, and 13 versus 67 sentences (74.8 and 53.6 UAS) when using the PoS, label and direction at the same time.

Other non-conventional parsers The observations made in this section shed a new light on the performance of state-of-the-art parsers trained using non-conventional data. The accuracies achieved with very small datasets are indeed comparable to other scores reported in the recent literature.

In the original experiments conducted with multi-source weighted delexicalized transfer (not based on BEAM), Rosa and Zabokrtsky (2015) achieve an UAS of 52.5 on HamleDT 2.0; in another line of cross-lingual parsing, the projection technique of Tiedemann and Agić (2016) yields 75.43 UAS on UDT 1.0 (COMBG model). Although neither work is conducted on UD 2.0, and thus the comparison is inexact, our metrics can still provide a rough estimation of each method's success: using UD 2.0, these scores would have been achieved by training on, respectively, 5 and 444[16] sentences. Empirical evidence hence raises serious doubts on the actual effectiveness of the main transfer techniques: cross-lingual parsers embed a limited amount of knowledge, and do not save much annotation effort. This is less true for methods based on projection, but these already require large data and significant human efforts to find, clean and sentence-align parallel data – a work which may turn out to be more demanding than annotating a small treebank.

Unsupervised parsing is another field in which parsers are trained in a non-conventional way, and would thus benefit from a comparison with monolingual supervision. For instance, Mareček and Straka (2013) achieve 48.7 UAS on average over the CoNLL 2006-2007 treebanks, while the average UAS of the rule-based approach of Martínez Alonso et al. (2017) is 57.5 on UD 1.2. These UAS correspond to 4 and 9 annotated sentences, which, again despite the dataset mismatch, still highlights how much work remains to be done in that field.

Size-based evaluation can consequently provide interesting reference points for parsers stemming from many fields: cross-lingual transfer and unsupervised models, but also many others like domain adaptation for instance.

Information conveyed by one source As a final experiment, we propose to evaluate the contribution of a given source to multi-source transfer, in terms of the amount of conveyed information.

[15] Because it is an average, this boundary should not be interpreted in a strict sense but as a trend: it is not applicable to each individual language in the considered set. The ability to transfer syntactic information depends indeed a lot on the linguistic setting (standard deviation of 15.3 UAS for KL-BEAM), so that the averages presented here do not replace any prior knowledge on transfer results for a given known language. However, when tackling a *new* language, they can provide a rough estimate to choose between the monolingual and cross-lingual approaches.

[16] While this amount can appear already substantial, it also denotes the fact that Tiedemann and Agić (2016)'s evaluation is based on a set of languages with marked relatedness, and thus good transfer properties. When averaging our measures only over that set of languages (but still not the same treebanks), the data-size equivalent drops to 92 sentences. Similarly, Lacroix et al. (2016)'s projection approach achieves 79.62 UAS, which is higher than with 500 sentences when considering all UD 2.0 treebanks, but corresponds to 295 sentences when retaining the same set of languages. Overall, the magnitude remains around a few hundred sentences for projection methods.

We build a cross-lingual parser for Romanian, using KL-BEAM as before but with a much smaller source set: French, Italian, Spanish and Bulgarian. The choice of the first three is motivated by their language family (Romance, like Romanian), and Bulgarian by geographic proximity (and hence various influences across history). This parser is then compared (overall and also along simple/complex classes) with those obtained when removing either one of the sources, in order to characterize precisely their contribution.

Table 3 reports the overall and simple/complex UAS measures, as well as data-size equivalents based on the Romanian BEAM. Comparing the full source set with the reduced ones, the respective benefits of each source appear clearly: Italian and French convey mostly complex information (respectively +0.3/+1.8 and +0.2/+1.2 on simple/complex scores), while Bulgarian improves mostly on simple classes (+1.5/+0.2 on simple/complex scores) and Spanish contributes equally in both cases (+1.5/+1.2).

Sources	UAS (ro)			#sentences (ro)		
	simple	overall	complex	simple	overall	complex
fr + it + es + bg	81.4	74.4	60.3	167	213	231
✗ it es bg	81.2	73.8	59.1	149	165	179
fr ✗ es bg	81.1	73.6	58.5	142	155	162
fr it ✗ bg	79.9	73.0	59.1	77	131	179
fr it es ✗	79.9	73.3	60.1	77	142	219

Table 3: Cross-lingual UAS and equivalent monolingual supervision in Romanian, for a KL-BEAM parser trained with various sets of sources. The simple/complex partition is that of BEAM in Romanian.

Regarding Spanish and Bulgarian, it is interesting to note that combining both sources more than doubles the amount of knowledge on simple classes (from 77 to 167). This suggests that multi-source combination itself enables valuable interactions between languages, thereby granting access to knowledge that would have been hard to obtain otherwise.

The results for Bulgarian are worth a closer look: the kind of information it conveys corresponds to syntactic structures that are easy to learn in Romanian (possibly deterministic), and yet not provided by other (Romance) sources. The Bulgarian influence on Romanian – in other words the structures that are out of the scope of Romance syntax – consequently seems to materialize as straightforward dependencies on unambiguous patterns.

On a final note, looking at composite scores instead of UAS only safeguards against overrating a given language: for instance, even though Spanish seems more valuable as a source than Italian (73.6 vs 73.0), it is only so because of the (numerous) simple dependencies, and it actually underperforms Italian on complex dependencies, where the true challenge is though.

6 Conclusion

In this work, we have introduced a new metric, called COMPLEXITY, to evaluate the difficulty to learn a given class of dependencies, in terms of data requirements: inspired by an in-depth analysis of fine-grained learning curves, we have aggregated their information into a single value. A series of systematic computations using that metric has unveiled interesting properties of the 3 considered parsing algorithms; it has notably revealed the kind of dependencies for which the parsers make an abnormally high amount of efforts during training, and hinted at some syntactic properties that they under-exploit.

Computing composite scores by separating simple and complex dependencies has been further enlightening for analyzing the aforementioned parsers, but above all it has enabled an extensive assessment of the benefits of parsers trained in a non-conventional way, typically cross-lingual parsers. While the typical UAS achieved by delexicalized transfer remain low and can easily be obtained with just a sample of target-side supervision, the information conveyed by cross-lingual parsers has in fact appeared of better quality: their actual added value is on complex dependencies. As a final case study, we have

leveraged the notion of complexity to characterize the contribution of a given source to a multi-source transfer procedure.

Regarding the notion of complexity, additional experiments with other class criteria can be found in the PhD thesis of the first author (Aufrant, 2016), as well as some insights on what makes a class complex, and proposals on how to leverage the COMPLEXITY information to improve cross-lingual transfer.

A natural continuation of this work would be to reproduce these measures with other cross-lingual transfer algorithms, and also with other kinds of non-conventional parsers, like unsupervised or out-of-domain parsers. Since the proposed procedure for estimating COMPLEXITY is in fact not specific to dependency parsing (only the class criterion is), it may also be interesting to apply it to other sequence labeling tasks, for instance PoS tagging.

Acknowledgments

This work has been partly funded by the French *Direction générale de l'armement* and by the *Agence Nationale de la Recherche* (ParSiTi project, ANR-16-CE33-0021).

References

Lauriane Aufrant and Guillaume Wisniewski. 2016. PanParser: a Modular Implementation for Efficient Transition-Based Dependency Parsing. Technical report, LIMSI-CNRS, March.

Lauriane Aufrant, Guillaume Wisniewski, and Franois Yvon. 2016. Cross-lingual and Supervised Models for Morphosyntactic Annotation: a Comparison on Romanian. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, 5. European Language Resources Association (ELRA).

Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2017. Don't Stop Me Now! Using Global Dynamic Oracles to Correct Training Biases of Transition-Based Dependency Parsers. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 318–323, Valencia, Spain, 4. Association for Computational Linguistics.

Lauriane Aufrant. 2016. *Training parsers for low-resourced languages: improving cross-lingual transfer with monolingual knowledge*. Ph.D. thesis, Université Paris-Saclay.

Yoav Goldberg and Joakim Nivre. 2012. A Dynamic Oracle for Arc-Eager Dependency Parsing. In *Proceedings of COLING 2012*, pages 959–976, Mumbai, India, 12. The COLING 2012 Organizing Committee.

Ophélie Lacroix, Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2016. Frustratingly Easy Cross-Lingual Transfer for Transition-Based Dependency Parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1058–1063, San Diego, California, 6. Association for Computational Linguistics.

Teresa Lynn, Ozlem Cetinoglu, Jennifer Foster, Elaine U Dhonnchadha, Mark Dras, and Josef van Genabith. 2012. Irish Treebanking and Parsing: A Preliminary Evaluation. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uur Doan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, 5. European Language Resources Association (ELRA).

David Mareček and Milan Straka. 2013. Stop-probability estimates computed on a large corpus improve Unsupervised Dependency Parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 281–290, Sofia, Bulgaria, 8. Association for Computational Linguistics.

Héctor Martínez Alonso, Željko Agić, Barbara Plank, and Anders Søgaard. 2017. Parsing Universal Dependencies without training. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 230–240, Valencia, Spain, 4. Association for Computational Linguistics.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the Errors of Data-Driven Dependency Parsing Models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131, Prague, Czech Republic, 6. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, 10. Association for Computational Linguistics.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Cristina Bosco, Gosse Bouma, Sam Bowman, Aljoscha Burchardt, Marie Candito, Gauthier Caron, Gülşen Cebirolu Eryiit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Silvie Cinková, Çar Çöltekin, Miriam Connor, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Marhaba Eli, Ali Elkahky, Tomaž Erjavec, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökrmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà M, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Petter Hohle, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşkara, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, Phng Lê Hng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Pinkey Nainwani, Anna Nedoluzhko, Lng Nguyn Th, Huyn Nguyn Th Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Livy Real, Siva Reddy, Georg Rehm, Larissa Rinaldi, Laura Rituma, Rudolf Rosa, Davide Rovati, Shadi Saleh, Manuela Sanguinetti, Baiba Saulīte, Yanin Sawanakunanon, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Antonio Stella, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jonathan North Washington, Zhuoran Yu, Zdeněk Žabokrtský, Daniel Zeman, and Hanzhi Zhu. 2017a. Universal Dependencies 2.0 – CoNLL 2017 Shared Task Development and Test Data. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

Joakim Nivre, Željko Agić, Lars Ahrenberg, et al. 2017b. Universal Dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University, Prague.

Joakim Nivre. 2004. Incrementality in Deterministic Dependency Parsing. In Frank Keller, Stephen Clark, Matthew Crocker, and Mark Steedman, editors, *Proceedings of the ACL Workshop Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57, Barcelona, Spain, 7. Association for Computational Linguistics.

Rudolf Rosa and Zdenek Zabokrtsky. 2015. KLcpos3 - a Language Similarity Measure for Delexicalized Parser Transfer. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 243–249, Beijing, China, 7. Association for Computational Linguistics.

Roy Schwartz, Omri Abend, and Ari Rappoport. 2012. Learnability-Based Syntactic Annotation Design. In *Proceedings of COLING 2012*, pages 2405–2422, Mumbai, India, 12. The COLING 2012 Organizing Committee.

Milan Straka and Jana Straková. 2017. Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada, 8. Association for Computational Linguistics.

Milan Straka. 2017. CoNLL 2017 Shared Task - UDPipe Baseline Models and Supplementary Materials. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

Jörg Tiedemann and Željko Agić. 2016. Synthetic Treebanking for Cross-Lingual Dependency Parsing. *Journal of Artificial Intelligence Research*, 55:209–248.

Jörg Tiedemann and Lonneke van der Plas. 2016. *Bootstrapping a dependency parser for Maltese – a real-world test case*.

Jörg Tiedemann. 2015. Cross-Lingual Dependency Parsing with Universal Dependencies and Predicted PoS Labels. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 340–349, Uppsala, Sweden, 8. Uppsala University, Uppsala, Sweden.

Morgan Ulinski, Julia Hirschberg, and Owen Rambow. 2016. Incrementally Learning a Dependency Parser to Support Language Documentation in Field Linguistics. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 440–449, Osaka, Japan, 12. The COLING 2016 Organizing Committee.

Guillaume Wisniewski and Ophélie Lacroix. 2017. A Systematic Comparison of Syntactic Representations of Dependency Parsing. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 146–152, Gothenburg, Sweden, 5. Association for Computational Linguistics.

Guillaume Wisniewski, Ophlie Lacroix, and Franois Yvon. 2018. Automatically Selecting the Best Dependency Annotation Design with Dynamic Oracles. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 401–406, New Orleans, Louisiana, 6. Association for Computational Linguistics.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS taggers and NP brackets via robust projection across aligned corpora. In *Proceedings of NAACL*.

Daniel Zeman and Philip Resnik. 2008. Cross-Language Parser Adaptation between Related Languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*, pages 35–42.

Yue Zhang and Joakim Nivre. 2011. Transition-based Dependency Parsing with Rich Non-local Features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA, 6. Association for Computational Linguistics.

Julian Zubek and Dariusz M. Plewczynski. 2016. Complexity curve: a graphical measure of data complexity and classifier performance. *PeerJ Computer Science*, 2:e76.

Seq2seq Dependency Parsing

Zuchao Li[1,2], **Jiaxun Cai**[1,2], **Shexia He**[1,2], **Hai Zhao**[1,2*]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, China
{charlee, caijiaxun, heshexia}@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn

Abstract

This paper presents a sequence to sequence (seq2seq) dependency parser by directly predicting the relative position of head for each given word, which therefore results in a truly end-to-end seq2seq dependency parser for the first time. Enjoying the advantage of seq2seq modeling, we enrich a series of embedding enhancement, including firstly introduced subword and node2vec augmentation. Meanwhile, we propose a beam search decoder with tree constraint and sub-root decomposition over the sequence to furthermore enhance our seq2seq parser. Our parser is evaluated on benchmark treebanks, being on par with the state-of-the-art parsers by achieving 94.11% UAS on PTB and 88.78% UAS on CTB, respectively.

1 Introduction

Dependency parsing for syntactic structure can be unstrictly put into two categories in terms of searching strategies over parsing trees, graph-based and transition-based. The previous work (Eisner, 1996; Mcdonald et al., 2005) searched the entire tree space but limited to local features with higher computational costs, while (Nivre, 2003) could adopt rich features but subjected to limited searching space. Besides, ensemble or hybrid methods on both the basic models have been well studied (Nivre and Mcdonald, 2008; Zhang and Clark, 2008). Most traditional dependency parsers rely heavily on feature engineering, especially for graph-based parser, which suffers from poor efficiency and generalization ability. Recent tendency for dependency parsing is adopting neural networks due to their significant success in a wide range of applications. Specially, leveraging sequence-to-sequence (seq2seq) model for dependency parsing started to appear (Wiseman and Rush, 2016; Zhang et al., 2017b).

The recently proposed seq2seq parsers focus on predicting a transition sequence to build a dependency tree, which makes them actually fall back into the transition-based model constrained by the adopted transition parsing algorithm. In this paper, we propose a seq2seq parser with a novel parsing structure encoding independent of transition parsing operation sequence. The proposed parser first directly generates the head position for each word in an input sentence using a seq2seq model, and then employs a BiLSTM-CRF model (Huang et al., 2015) to predict the relation label for each determined dependency arc. Since the greatest challenge of the applying seq2seq model in dependency parsing is to guarantee the tree structure of the outputs, a beam search with tree constraint method is proposed to enforce a well-formed dependency tree in the decoder side. Besides, dependency parsing has to well handle long-distance parsing with corresponding to too long sequence learning bias in seq2seq models, we accordingly introduce a sub-root based sequence decomposition to effectively alleviate this issue.

Our proposed parser is evaluated on benchmark treebanks on both English and Chinese, which demonstrates that our model achieves the state-of-the-art scores among seq2seq parsing, and yields competitive performance with traditional transition- and graph-based parsers. In summary, this paper provides the following contributions:

*Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100), National Natural Science Foundation of China (No. 61672343 and No. 61733011), Key Project of National Society Science Foundation of China (No. 15-ZDA041), The Art and Science Interdisciplinary Funds of Shanghai Jiao Tong University (No. 14JCRZ04).

Proceedings of the 27th International Conference on Computational Linguistics, pages 3203–3214
Santa Fe, New Mexico, USA, August 20-26, 2018.

• We propose a seq2seq model which does not rely on any transition sequence by directly predicting the head position for each word in the sentence, unlike all previous work that first predicted a transition sequence and then built dependency tree using a known transition parsing algorithm.

• We especially design a beam search algorithm in the decoder side, which always guarantees the output to be a well-formed dependency tree.

• To deal with the long-distance dependency, we introduce sub-root decomposition and it also alleviates the problem caused by too long seq2seq learning in the meantime.

2 Related Work

Early researches (Ma and Zhao, 2012; Martins et al., 2013) achieved comparable results with high-order feature. With the impressive success of deep neural networks in a wide range of NLP tasks (Cai and Zhao, 2017; Qin et al., 2017; He et al., 2018; Cai et al., 2018; Zhang et al., 2018; Bai and Zhao, 2018; Huang et al., 2018), neural models have been successfully applied to various dependency parsers (Li et al., 2018; Wang et al., 2017; Zhang et al., 2016). Dyer et al. (2015) introduced the stack LSTM to promote the transition-based parsing. Kiperwasser and Goldberg (2016) incorporated the bidirectional Long Short-Term Memory (BiLSTM) into both graph- and transition-based parsers. Andor et al. (2016) proposed globally normalized networks and achieved the best results of transition-based parsing, while the state-of-the-art result was reported in Dozat and Manning (2016), which proposed a deep biaffine model for graph-based parser.

Though previous neural parsers have achieved such inspiring progresses, the majority of them heavily rely on the primitive parsing framework. Usually, those neural parsers build a network for feature extracting and use the neural features in place of handcrafted ones to predict some discrete actions in a traditional parsing algorithm. Until recently, few work (Wiseman and Rush, 2016; Zhang et al., 2017b) considered the seq2seq model, resulting in an end-to-end parser. However, those seq2seq parsers focus on predicting the transition sequences and thus actually need some kinds of complicated post-processing to build well-formed dependency trees, which make them fall back to a transition-based parser with a seq2seq transition predictor.

In this work, we attempt to build a true seq2seq parser without transition parsing algorithm for post-processing. The proposed parser is different from previous proposed seq2seq parser in the way that it directly predicts the head position instead of the transition. Though both our parser and Zhang et al. (2017a) make direct head prediction during parsing, our parser essentially differs from their parser in two aspects. Firstly, their parser runs standard graph-based maximum spanning tree (MST) algorithm to guarantee the tree structure, and thus called head selection in their parser serves for the purpose of speeding up the graph-based parsing, while ours introduces a beam search with tree constraint only in the decoder side to enforce a well-formed tree and thus gets rid of relying on specific parsing algorithm. Secondly, each time the head selection parser selects a head, it relies on referring to each pair of words in the sentence. While our parser performs a fully generative decoding in one pass by using an attention layer to keep around the context of each word.

Another line of studies related to this work focus on beam search, which have attracted much interest and have been adopted to improve the performance of greedy parsers. Beam search commonly adopts approximate strategy for computational tractability, which aims at including more sophisticated features within reasonable cost. Zhang and Clark (2008) proposed a beam search based parser applied to integrate graph-based and transition-based parsers. Most transition-based neural models (Weiss et al., 2015; Zhou et al., 2015; Andor et al., 2016) incorporated the structured perceptron with beam search decoding, resulting in substantial improvement of accuracy. Additionally, beam search has been also incorporated in the seq2seq framework (Wiseman and Rush, 2016).

Following the previous works, we also adopt a beam search in our decoder. However, since our parser aims at predicting a structure instead of sequence, we especially design a beam search algorithm with several searching constraints, which enforces the tree structure of the outputs.

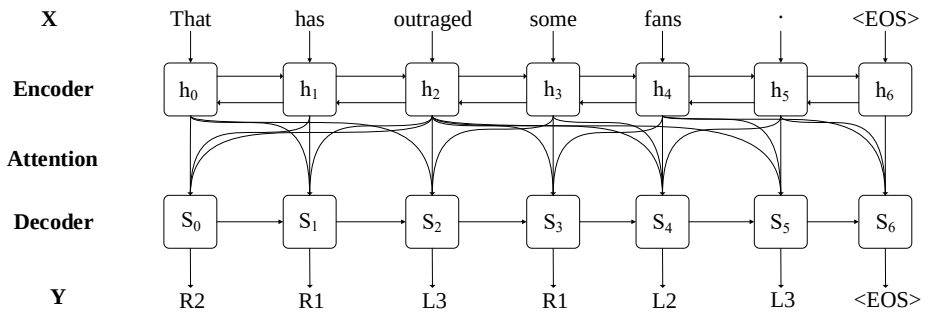


Figure 1: The framework of the proposed seq2seq model.

3 Seq2seq Parser

In this study, our parser is built upon a seq2seq model (Sutskever et al., 2014), which encodes the input sentence forward and backward, and predicts the head position for each word in the sentence. Our model contains three main components: (1) an encoder that processes the input sentence and maps it into some hidden states that lie in a low dimensional vector space $\mathbb{R}^h$, (2) a decoder that incorporates the hidden states and the previous prediction to generate head position of the current word, and (3) an attention layer that encodes the context for each focused word. Figure 1 illustrates our model.

Recurrent neural networks (RNNs) (Elman, 1990) is commonly used to build block for seq2seq model. Specifically, our model employs the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) variant of RNN, which uses numbers of gates to address the problem of vanishing or exploding gradient when trained with back-propagation through time. A standard LSTM unit consists of an input gate i_t, a forget gate f_t, an output gate o_t, a memory cell c_t and a hidden state h_t, where the subscript t denotes the time step.

3.1 Encoder

The encoder of our parser is a BiLSTM, which processes an input sequence in both directions by incorporating a stack of two distinct LSTMs.

Given an input sentence $X = \{w_1, \cdots, w_L\}$, the i-th element w_i is encoded by first feeding its embedding representation e_i in to two distinguish LSTMs: the forward LSTM and the backward LSTM, to obtain two hidden state vectors: h_i^f and h_i^b respectively, and then concatenating the two vectors as $h_i = [h_i^f; h_i^b]$.

3.1.1 Source Representation

Given a vocabulary W, each individual word $w_i \in W$ is mapped into a real-valued vector (word embedding) $w \in \mathbb{R}^m$ where m is the dimension. We employ the GloVe (Pennington et al., 2014) and Node2Vec (Grover and Leskovec, 2016) to generate the pre-trained word embedding, obtaining two distinct embedding for each word.

It is worth noting that Node2Vec is an embedding method that is capable of encoding the topological information of inside structure. To further incorporate the structural information of dependency tree, we perform a Node2Vec pre-training on the training set.

Besides, our model adopts subword which is obtained though BPE segmentation (Sennrich et al., 2015) broadly used in neural machine translation (NMT), and character augment embedding with AllenNLP toolkit according to (Gardner et al., 2017). A subword dictionary is built by running BPE on the Wikipedia, which segments each word into several subwords. Each subword in the dictionary is mapped into a real-valued vector (subword embedding). In our experiments, the subword embedding is randomly initialized and then trained jointly with other components of the network. To get the representation of the original word, we use an additional neural network component (Peters et al., 2018) to distill the character and subword embedding for representation. Our model also employs the part-of-speech (POS) tag embedding following previous works (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2016). The

POS embedding is randomly initialized and jointly trained with other model parameters too.

The adopted word representation of our model is the concatenation of all above-mentioned embeddings: $e = [e^g; e^n; e^s; e^c; e^p]$, where e^g is the GloVe embedding, e^n is the Node2Vec embedding, e^s is the subword embedding, e^c is the character embedding and e^p is the POS tag embedding.

3.2 Decoder

In our model, an LSTM decoder predicts the head position for each word in the input sentence. Given a word w_i, the probability of each word in the vocabulary to be its head is given by:

$$p(\hat{y}_i|\hat{y}_1, \cdots, \hat{y}_{i-1}) = LSTM^{\mathcal{D}}(\hat{y}_{i-1}, h_i), \tag{1}$$

where $\hat{y}_{i-1}$ is the last output of the LSTM decoder, h_i is the hidden state of current word. The joint probability of the final output is then a product of conditional probability of each predicted head:

$$P(\hat{y}) = \prod_{i=0}^{M} p(\hat{y}_i|\{\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_{i-1}\})$$

3.2.1 Target representation

In NMT context, the decoder output is sampled from the entire vocabulary using the probability computed by Eq.(1). However, in dependency parsing situation, the candidate set of the head words is subject to the words in the sentence. Thus, the predicted head would likely fall outside the input sentence when decoding with a full vocabulary upon corpus. To keep the generation of head within bounds, our model predicts the relative position instead of word form when assigning a head for each word. In the training process, we convert each head into a position representation, encoding relative distance between word and its head. When working, the parser picks a head for each word by predicting the relative position of the head. Our experiments show that the conversion is effective in bounding the output of our seq2seq parser.

Given a word w_i and its head word w_j, the relative position representation of w_j is obtained by:

$$R_{i,j} = \begin{cases} L_{j-i} & if \quad i < j, \\ R_{i-j} & if \quad i > j. \end{cases}$$

Figure 2 illustrates our relative position tag encoding for the output dependency structures.

3.3 Attention Mechanism

As usual, we employ an attention layer (Luong et al., 2015) to encode the context for each word. The context vector c_j at the j-step of the decoding process is calculated as the weighted sum of the hidden states of the input sequence:

$$c_j = \sum_{i=1}^{N} \alpha_j(i) h_i.$$

The attention weight $\alpha_j(i)$ between the i-th encoder hidden state h_i and the j-th decoder hidden state h_j is computed by a softmax function:

$$\alpha_j(i) = \frac{v^{\mathsf{T}} \tanh(W^{(a)}[h_i; h_j])}{\sum_{k=1}^{N} v^{\mathsf{T}} \tanh(W^{(a)}[h_k; h_j])},$$

where $[h_i; h_j]$ and $[h_k; h_j]$ denote the vector concatenation of the rows, $W^{(a)}$ and v are learnable parameters.

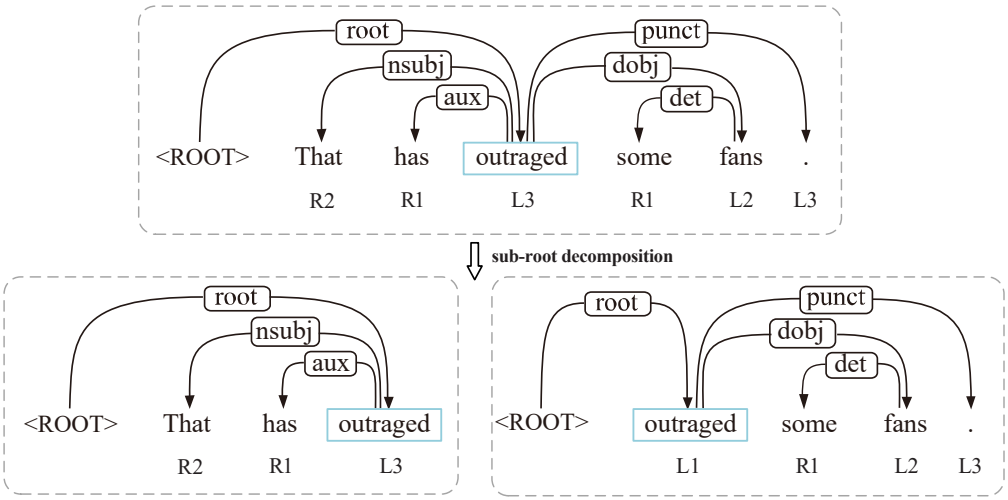


Figure 2: Illustration of target representation and sub-root segmentation for sentence ***That has outraged some fans .*** from PTB. The sequences below the word are the relative position tags which demonstrate the corresponding distance and direction (R: right, L: left) between the node and its syntactic head. The word **outraged** is defined as **sub-root**.

4 Sub-root Decomposition

It is well known that correctly recovering the long-distance dependency is the greatest challenge in dependency parsing. Even incorporating the BiLSTM encoder and the attention layer, the primitive seq2seq parser still performs poorly when encountering long sequences, as the long-distance dependencies heavily prevent the parser from building a correct tree.

To address the long-distance dependency problem and alleviate error propagation, we propose sub-root decomposition over the input parse tree. Such decomposition will cause the original dependency tree accordingly segmented into two or more subtrees, and the original input sentence segmented into shorter pieces. We define the notation **sub-root** as the direct syntactic children of the **ROOT** node. When working on 'nearly' projective dependency parsing, such decomposition would not break dependency arc in most situations.

Formally, given an input sentence sentence with length L: $X = \{w_1, w_2, \cdots, w_L\}$, for the i-th word w_i, we represent its head as $w_i.head$. Besides, we define a decomposition threshold β to determine whether an input sentence should be segmented into shorter ones. Only if the sentence length L is larger than the threshold β, we perform a decomposition on the sentence.

To perform the sub-root decomposition, we should first recognize the sub-root for a given sentence. We regard the sub-root recognition as a sequence tagging task, and employ a BiLSTM-CRF model which consists of 4-layer Bi-LSTM and a CRF layer to tag the sub-roots of a sentence. The output of this model is a binary indicator that distinguishes the sub-root nodes from others:

$$I(w_i) = \begin{cases} 0 & if \quad w_i.head \neq ROOT, \\ 1 & if \quad w_i.head = ROOT. \end{cases}$$

After recognizing the sub-root nodes, we decompose a sentence into several parts that come from collecting all nodes rooted by every sub-root, and perform the parsing on the sub-sentence. For example, given a sentence $X = \{w_1, w_2, \cdots, w_L\}$ with w_k as its sub-root, the sentence is decomposed into two sub-sentences: $X_1 = \{w_1, w_2, \cdots, w_k\}$ and $X_2 = \{w_k, w_{k+1}, \cdots, w_L\}$. Then we use the sentences X_1 and X_2 which are both rooted at w_k for training and parsing. It is worth noting that we use the gold sub-root to decompose a sentence in training, and predict the sub-roots for sentences in testing. Besides, the merge operation is necessary after decoding. The merge operation simply concatenates the sub-sentence together and set the head of sub-root node as ROOT.

Algorithm 1 Beam search with tree constraint

/* S decode scores; ts time step; L sentence length; $\mathcal{D}$ target vocabulary; Γ decode history;*/
procedure $step(S, ts, L, \mathcal{D}, \Gamma)$
Init empty valid target set $\mathcal{V}$ and visible history set Γ_v
if $ts > L$ **then**
  $S[\mathcal{D} \backslash \{\textbf{EOS}\}] \leftarrow -\infty$
else
  /* filter valid target by L and ts*/
  $\mathcal{V} \leftarrow valid\_target(\mathcal{D})$
  $\Gamma_v \leftarrow \Gamma[ts - \delta : ts - 1]$
  for $node \in \mathcal{D}$ **do**
    if $node \notin \mathcal{V}$ **then**
      $S[node] \leftarrow -\infty$
    else
      if $make\_circle(\Gamma_v, node)$ **then**
        $S[node] \leftarrow S[node] * \alpha_c$
      if $non\_projective(\Gamma_v, node)$ **then**
        $S[node] \leftarrow S[node] * \alpha_p$
$S \leftarrow top\mathbf{K}(S)$
$\Gamma \leftarrow \Gamma + \mathcal{D}[S]$
return S

4.1 Beam Search with Tree Constraint

Beam search is a popular approach to enlarge searching space in transition-based dependency parsing. The typical beam search algorithm keeps a fixed number (say K) of candidate states in the beam according to their scores (or probabilities). Each time a searching is performed, the algorithm explores all the possible next states for each states in the beam, and keeps the top-K states in beam again.

In inference stage, the beam B of time step ts is updated based on K states (hypotheses) of the last beam:

$$B_{ts} = topK[Y^k_{ts-1}, y^{k,n}_{ts}],$$

where $1 \leq k \leq K$ and $1 \leq n \leq N_k$. N_k is the number of branches of the k-th hypotheses in the last step. Y^k_{ts-1} represents the historical output and y_{ts} indicates the candidate target in the ts-th step. For each new exploring state, its score is computed by,

$$S(Y^k_{ts-1}, y^{k,n}_{ts}|x) = S(Y^k_{ts-1}|x) \ + \log p(y^{k,n}_{ts}|x, Y^k_{ts-1}).$$

Since our model directly predicts the head position for each word, its output might sometime violate the constraints of dependency tree structure. To enforce the model output to be a well-formed dependency tree, we introduce several constraints into the primitive beam search algorithm, resulting in the beam search with tree constraint.

The constraints that ensure a projective dependency tree are listed as follows:

• *Single headed*: A node should be assigned one and only one head. For our seq2seq parser, this also implies that the output sequence length should be exactly same as the input sequence length.

• *Acyclic*: In the decoding process, a newly generated head should not introduce a cycle in the dependency path.

• *Projective*: For the projective dependency parsing, the newly generated head should not cross the arc built by previous prediction (99.9% of PTB-SD, 100% of CTB on the training dataset are projective).

To enforce the above constraints, we introduce a weighted function $f_c(\cdot)$ that computes an additional weight for each candidate state to indicate whether the constraints are satisfied.

$$S(Y^k_{ts-1}, y^{k,n}_t|x) = S(Y^k_{ts-1}|x) \ + \log(p(y^{k,n}_t|x, Y^k_{ts-1}) \cdot f_c(y^{k,n}_{ts}|Y^k_{ts-1})),$$

$$f_c(y_{ts}^{k,n}|\cdot) = \begin{cases} 0, & ts > L, y_{ts} \neq \textbf{EOS}, \\ \alpha_c, & cycle(y_{ts-\delta}, \cdots, y_{ts}), \\ \alpha_p, & nproj(y_{ts-\delta}, \cdots, y_{ts}), \\ 1, & others. \end{cases}$$

where **EOS** is an additional symbol indicating the end of the predicted sequence, L is the length of the input sequence, $cycle(\cdot)$ is the cycle detected function which returns *True* if there exists a cycle on the input path, $nproj(\cdot)$ is an indicator of whether the resulting tree is projective. Note that for the sake of efficiency, we detect the violations of the *Acyclic* and *Projective* constraints by looking back a fixed number δ of historical predictions instead of enforcing them on the whole sequence. Thus, both $cycle(\cdot)$ and $nproj(\cdot)$ take a subsequence $y_{ts-\delta}, \cdots, y_{ts}$ instead of the whole sentence as input. By loosing the global constraints to local ones, the incorrect historical decisions that fall outside a window with size δ can never disturb a current prediction. In our experiments, δ is set to 8, both α_c and α_p are set to 0.8. Algorithm 1 shows the detail of the proposed beam search decoding with tree constraint.

5 Experiments

The proposed seq2seq parser is evaluated on the Penn Treebank (PTB) and the Chinese Treebank (CTB 5.1) with the unlabled attachment score (UAS) and the labeled attachment score (LAS) metrics (excluding punctuation). Follow the usual way processing the PTB and CTB:

For PTB, applying Stanford basic dependencies (SD) representation (De Marneffe et al., 2006), using sections 2-21 for training, section 22 for development and section 23 for testing as same as the standard splitting and tagging POS tag with the Stanford tagger (Toutanova et al., 2003).

For CTB, adopting Penn2Malt tool for conversion, splitting the dataset by sections 001-815, 1001-1136 for training, sections 886-931, 1148-1151 for development, and sections 816-885, 1137-1147 for testing as (Zhang and Clark, 2008) and using the golden segmentation and POS tags as (Chen and Manning, 2014).

5.1 Setup

In the experiments, our seq2seq parser implemented based on the OpenNMT-py project (Klein et al., 2017), in which employs a 4-layer Bi-LSTM as encoder and a 2-layer LSTM as decoder. The parameters are randomly initialized and optimized using the Adam algorithm with mini-batch size of 64. The initial learning rate is set to 0.001 (with $\beta_1 = 0.9, \beta_2 = 0.999$), and decays by 0.5 every epoch after running 20 epochs. In training phase, we adopt a dropout rate of 0.3. The size of the vocabulary in the target side is limited to 100 by setting the maximum relative position to 50, namely, the target side vocabulary is:

$$\mathcal{V}_t = \{L1, L2, \cdots, L50, R1, R2, \cdots, R50\}$$

In our experiments, the beam size is set to 5.

The representation of each input token is the concatenation of GloVe embedding, node2vec embedding and subword embedding. The GloVe embedding is pre-trained on Wikipedia 2014 and Gigaword 5 with 100 dimension (6B tokens). The node2vec embedding is pre-trained on the training set and the dimension is 128. It is trained with random walks in the syntactic graph (in order to make all the dependency tree form the graph, we add a dummy **ROOT** node). The walk length of node2vec is set to 30, per node visit(walk) times is 200 and the skip-gram window size is 10. To obtain subword segmentation for each word, we train the BPE model on the English and Chinese Wikipedia dataset respectively with number of merge operations set to 10K.

The sub-root tagging model consists of a 4-layer Bi-LSTM and a 1-layer CRF, taking the word and POS tag as inputs. Besides, after getting a dependency tree of the input sentence, we employ another BiLSTM-CRF model to predict the dependency relation of each arc. The predictor takes the word embedding, POS embeddings of head and dependent respectively as input, and performs a multiple labels tagging.

The code is available at `https://github.com/bcmi220/seq2seq_parser`.

5.2 Results

Table 1 shows comparison of our seq2seq parser with previously published parsers on PTB-SD and CTB. Unlike the transition-based parsers listed in ***Transition-NN*** block and the graph-based listed in the ***Graph-NN*** block, our model pursues a more simple way without any kind of transition- or graph-based algorithm. Our model outperforms most parsers that use either transition- or graph-based algorithm. Our model is competitive to the transition-based model of Andor et al. (2016) and slightly lower than the one of Zhu et al. (2015) which introduced a recursive convolutional neural network to encode the dependency tree and re-ranked the k-best trees. It is worth noting that Dozat and Manning (2016) used deep biaffine attention in the graph model (Kiperwasser and Goldberg, 2016), achieving the best results among traditional graph and transition parsing.

The parsers (Wiseman and Rush, 2016; Zhang et al., 2017b) are also seq2seq models, but they used transition sequence as target sequence encoding. The comparison shows our parser achieves state-of-the-art performance among seq2seq models. Kuncoro et al. (2016) also reported parsing result of 95.8% UAS on PTB. Since its result is converted from phrase-structure parsing and beyonds our focus, we excluded it from the table.

System	Method	PTB-YM LAS	PTB-YM (%) UAS	PTB-LTH LAS	PTB-LTH (%) UAS	PTB-SD LAS	PTB-SD (%) UAS	CTB LAS	CTB (%) UAS
Non-NN:									
Ma and Zhao (2012)	4th order	-	93.4	-	-	-	-	-	87.4
Zhang and McDonald (2012)	cube pruning	-	93.1	-	-	-	-	-	86.9
Transition-NN:									
Dyer et al. (2015)	greedy	-	-	-	-	90.9	93.1	85.5	87.1
Kiperwasser and Goldberg (2016)	greedy	-	-	-	-	91.9	93.9	**86.1**	**87.6**
Andor et al. (2016)	beam	-	-	-	-	**92.79**	**94.61**	-	-
Zhu et al. (2015)	re-ranking	-	-	-	-	-	94.16	-	87.43
Graph-NN:									
Zhang and McDonald (2014)	3rd order	92.48	93.57	-	-	90.64	93.01	86.34	87.96
Zhang et al. (2016)	3rd order	92.23	93.31	90.07	93.14	91.29	93.42	86.17	87.65
Wang and Chang (2016)	1st order	92.45	93.51	-	-	91.82	94.08	86.23	87.55
Kiperwasser and Goldberg (2016)	1st order	-	-	-	-	90.90	93.0	84.9	86.5
Dozat and Manning (2016)	1st order	-	-	-	-	**94.08**	**95.74**	**88.23**	**89.30**
Zhang et al. (2017a)	1st order+re-ranking	-	-	-	-	91.90	94.10	86.15	87.84
Seq2seq:									
Wiseman and Rush (2016)	beam	-	-	-	-	87.26	91.57	-	-
Zhang et al. (2017b)	attention,single	-	-	-	-	91.60	93.71	85.40	87.41
This work	beam+sub-root	-	-	-	-	**92.08**	**94.11**	**86.23**	**88.78**

Table 1: Comparison of results on PTB and CTB test datasets.

5.3 Analysis

5.3.1 Sub-root Decomposition

This subsection gives an analysis on the effectiveness of the proposed sub-root decomposition algorithm. Figure 3 shows sentence length distribution on PTB and our sub-root tagging $F1$ score. For the sub-root tagging model, we get 96.73% and 96.01% $F1$ scores on development and test datasets, respectively.

To determine an optimal decomposition length threshold, we conduct experiments on PTB and the results are shown in Figure 4, which indicates that the threshold 40 performs best, though sub-root tagging model got the similar performance below the length 40. Meanwhile, all thresholds larger than 40 are superior to the case without decomposition, especially for the sentences longer than 40. We are surprised that all the other threshold settings are not better than the original case without any decomposition, which can be attributed to the dissatisfactory sub-root tagging $F1$ score.

Our experiments reveal that the sub-root decomposition method does work in improving the performance of our seq2seq parser, for its help on alleviating the long-distance dependencies and error propagation in the course of decoding.

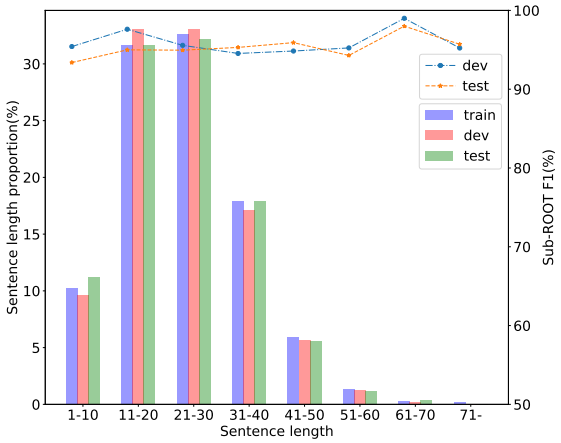


Figure 3: Sentence length distribution in PTB and the corresponding sub-root tagging accuracy.

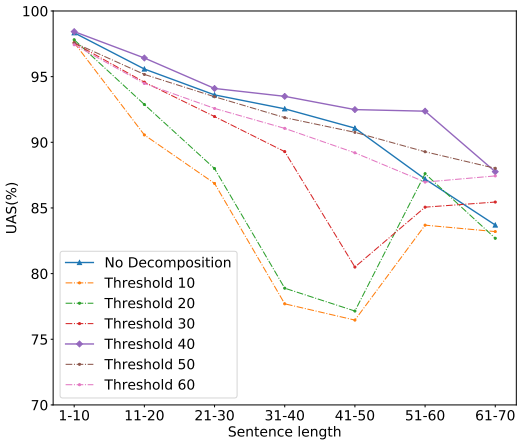


Figure 4: The UAS on PTB test dataset under different decomposition threshold.

5.3.2 Target Sequence

To make a direct comparison on the target encoding between our relative position sequence and previous transition sequence, we build a basic seq2seq model with the same source embedding, but without sub-root decomposition. The results are shown in Table 2, which indicates our relative position sequence is a better solution.

Since we limit the length of target sequence equal to the input sequence, the relative position sequence can also be predicted by sequence tagging model. We further compare the conventional sequence tagging model (Bi-LSTM+CRF with the same parameter settings as sub-root tagger) and the basic seq2seq model. The results demostrate that the seq2seq is necessarily better than the sequence tagging model for our concerned task.

Target encoding	**Model**	**Dev** (%) LAS	UAS	**Test** (%) LAS	UAS
Transition	seq2seq	83.56	87.34	82.77	87.49
Relative position	sequence tagging	83.81	87.58	81.37	87.60
Relative position	seq2seq	84.99	89.16	85.03	89.32

Table 2: Transition vs. relative position sequences in basic seq2seq model and sequence tagging vs. seq2seq model with relative position sequence (**without decomposition**).

5.3.3 Ablation Study

	Dev (%) LAS	UAS	**Test** (%) LAS	UAS
This work	91.86	93.84	92.08	94.11
-subword emb	91.35	93.76	91.77	93.95
-node2vec emb	91.60	93.71	91.85	94.01
-tree constraint	91.12	93.21	90.75	93.60

Table 3: Contribution of different components in our model.

In order to explore the contribution of the beam search with tree contraint and embeddings employed, we conduct a group of ablation experiments on PTB-SD dataset. The settings keep the same as before mentioned. Table 3 shows that the beam search with tree constraints substantially enhances the performance of our seq2seq parser. Meanwhile, both subword embedding and node2vec embedding contribute

to improving the parsing results.

6 Conclusion

In this paper, we propose a seq2seq parser with representation enhancement that directly predicts head positions without relying on transition sequence. To mitigate the long-distance dependency problem, we introduce the sub-root decomposition to shorten the sequence. To enforce a well-formed tree structure and alleviate error propagation, we employ beam search with constraints. Experiments on English and Chinese Penn Treebank verify the effectiveness of the proposed model. To the best of our knowledge, this is the first reported seq2seq parser with direct head prediction achieving promising performance.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany, August.

Hongxiao Bai and Hai Zhao. 2018. Deep enhanced representation for implicit discourse relation recognition. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*.

Deng Cai and Hai Zhao. 2017. *Pair-Aware Neural Sentence Modeling for Implicit Discourse Relation Classification*. IEA/AIE 2017, Part II, LNAI 10351.

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntactic-agnostic or syntactic-aware? In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454. Genoa Italy.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *Advances in Neural Information Processing Systems*.

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

Jason Eisner. 1996. Efficient normal-form parsing for combinatory categorial grammar. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 79–86.

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

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

Shexia He, Zuchao Li, Hai Zhao, Hongxiao Bai, and Gongshen Liu. 2018. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*.

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

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *Computer Science*.

Yafang Huang, Zuchao Li, Zhuosheng Zhang, and Hai Zhao. 2018. Moon IME: neural-based chinese pinyin aided input method with customizable association. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018), System Demonstration*.

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

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A Smith. 2016. What do recurrent neural network grammars learn about syntax? *arXiv preprint arXiv:1611.05774*.

Haonan Li, Zhisong Zhang, Yuqi Ju, and Hai Zhao. 2018. Neural character-level dependency parsing for Chinese. In *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. pages 1412–1421, September.

Xuezhe Ma and Hai Zhao. 2012. Fourth-order dependency parsing. *Proceedings of COLING 2012: posters*, pages 785–796.

André FT Martins, Miguel B Almeida, and Noah A Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers.

Ryan Mcdonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Meeting on Association for Computational Linguistics*, pages 91–98.

Joakim Nivre and Ryan T. Mcdonald. 2008. Integrating graph-based and transition-based dependency parsers. In *ACL 2008, Proceedings of the Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, Usa*, pages 950–958.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric P. Xing. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1006–1017.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *Computer Science*.

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

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2306–2315.

Hao Wang, Hai Zhao, and Zhisong Zhang. 2017. A transition-based system for universal dependency parsing. In *CONLL 2017 Shared Task: Multilingual Parsing From Raw Text To Universal Dependencies (CONLL 2017)*, pages 191–197.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 323–333, Beijing, China, July.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Austin, Texas, November.

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

Hao Zhang and Ryan McDonald. 2012. Generalized higher-order dependency parsing with cube pruning. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 320–331. Association for Computational Linguistics.

Hao Zhang and Ryan McDonald. 2014. Enforcing structural diversity in cube-pruned dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 656–661.

Zhisong Zhang, Hai Zhao, and Lianhui Qin. 2016. Probabilistic graph-based dependency parsing with convolutional neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1382–1392.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017a. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 665–676, Valencia, Spain, April.

Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou, and Enhong Chen. 2017b. Stack-based multi-layer attention for transition-based dependency parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1677–1682, Copenhagen, Denmark, September.

Zhuosheng Zhang, Jiangtong Li, Pengfei Zhu, and Hai Zhao. 2018. Modeling multi-turn conversation with deep utterance aggregation. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1213–1222, Beijing, China, July.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. 2015. A re-ranking model for dependency parser with recursive convolutional neural network.

Revisiting the Hierarchical Multiscale LSTM

Ákos Kádár
Tilburg University
a.kadar@uvt.nl

Marc-Alexandre Côté
Microsoft Research Montreal
macote@microsoft.com

Grzegorz Chrupała
Tilburg University
g.chrupala@uvt.nl

Afra Alishahi
Tilburg University
a.alishahi@uvt.nl

Abstract

Hierarchical Multiscale LSTM (Chung et al., 2016a) is a state-of-the-art language model that learns interpretable structure from character-level input. Such models can provide fertile ground for (cognitive) computational linguistics studies. However, the high complexity of the architecture, training procedure and implementations might hinder its applicability. We provide a detailed reproduction and ablation study of the architecture, shedding light on some of the potential caveats of re-purposing complex deep-learning architectures. We further show that simplifying certain aspects of the architecture can in fact improve its performance. We also investigate the linguistic units (segments) learned by various levels of the model, and argue that their quality does not correlate with the overall performance of the model on language modeling.

1 Introduction

Verifying and reproducing claims published in scientific articles is an essential part of building a solid foundation for future research. As such, reproduction has a long history in many scientific fields (Willett et al., 1985; Venables et al., 1993; Waltemath et al., 2011). Recent large-scale studies, however, raise concerns about the reproducibility in a variety of areas and the potential effect of this crisis (Baker, 2016). A reproduction study by the Open Science Collaboration (2015) estimates that only 40% of research in psychology is reproducible, while Begley and Ellis (2012) end up confirming only 11% of preclinical cancer studies. The latter work makes an important link between the low reproducibility rates and the notoriously low impact of preclinical cancer research on clinical practice (Hutchinson and Kirk, 2011).

Our work is motivated by a similar concern, specifically the applicability of complex deep learning architectures for computational (cognitive) linguistics research. State of the art systems often employ complex architectures which integrate various design features and use many optimization techniques. Because the focus is on boosting the final performance on a given task, often little effort is put into understanding where the power of the system comes from. This makes these models much harder to adapt for new tasks or domains. In our view, it is essential not only to be able to reproduce reported results, but also to understand the contribution of various design features through systematic ablation experiments.

The higher performance brought by modern neural network architectures often comes at the cost of our understanding of the representations and structural information the system learns. However, for models to be generalizable to new domains, it is important to move towards analyzing such structural representations, and investigating their impact on the final performance of the model.

In the current study, we examine the reproducibility of a language model with the ability to learn explicit linguistic structure: the Hierarchical Multiscale Recurrent Neural Network (HMLSTM) model. This architecture was introduced by Chung et al. (2016a) and set a new

Proceedings of the 27th International Conference on Computational Linguistics, pages 3215–3227
Santa Fe, New Mexico, USA, August 20-26, 2018.

state of the art on language-modeling benchmarks Text8, Hutter Prize and character-level Penn Treebank. Additionally, the paper features examples where the lowest layer of the model recovers word-level segmentation and in some cases performs interpretable morphological analysis. With this reproduction study our motivation is to provide insight into the dynamics of the model, which can facilitate re-using and further developing its architecture and repurposing it for tasks other than language modeling. Crucially, we investigate whether the performance of the model depends on the acquisition of high-quality linguistic structure.

1.1 The importance of reproducibility

There is no established tradition of reproduction studies in computational linguistics; however, scattered attempts at reproducing a number of studies have highlighted the importance of investigating the dataset, the model architecture and the evaluation scheme used in experimental designs. For example, Mieskes (2017) quantifies the availability of non-benchmark datasets underlying the experiments. Horsmann and Zesch (2017) re-run coarse-grained multilingual part-of-speech tagging experiments by Plank et al. (2016) and confirm the superior performance of LSTM-based architectures on fine-grained tagsets. Marrese-Taylor and Matsuo (2017) fail to reproduce the results of three articles in the domain of aspect-based opinion mining, with the conclusion that repeating experiments without the availability of source code is hindered due to lack of details on pre-processing, model architecture specification and exact parameter settings. Morey et al. (2017) replicate the results of 9 discourse parsers trained on the RST Discourse Treebank (RST-DT) (Carlson et al., 2003) and show that most of the recent gains in the domain are due to non-trivial differences in evaluation methodology.

We propose one further step in this direction: in addition to reproducing reported results, it is important to investigate the role the components of complex models play and examine their impact on the behavior of the model. The HMLSTM model that we choose as our case study has many desirable properties. However, it is fairly complex and allows for a considerable degree of freedom for implementation. We experimentally explore the function of architectural details of the HMLSTM model, as well as experiment with varying the specifics of the training procedure. We aim to shed light on the potential bottlenecks involved in re-purposing a complex deep learning sequence modeling architecture for computational linguistics studies.

1.2 The importance of interpretability

Language data, both in the form of speech and text, exhibits hierarchical structure: characters or phonemes form morphemes and words, which form phrases, which in turn form whole sentences. At each level, the units are composed to build up the meaning of the sentence at the highest level. Modern RNN architectures have been very successful in solving character-level NLP tasks (Chung et al., 2016b; Golub and He, 2016; Mikolov et al., 2012). However, they do not make the learned linguistic structure explicit: rather it can be presumed to be cryptically encoded in the states of the hidden layers. The higher performance brought by modern neural network architectures often comes at the cost of our understanding of the representations and structural information such systems learn. However, for a model to be generalizable to new domains, it is important to move towards analyzing such structural representations, and investigating their impact on the final performance of the model.

Understanding the structural information encoded in black-box neural network architectures has been a desired target since Elman (1990) and several studies have used indirect post-analysis techniques and auxiliary tasks to probe the acquired structure. For example, recent studies have assessed the ability of Long Short-Term Memory (LSTM) language models to encode subject-verb agreement (Linzen et al., 2016), analyzed the translation quality of character-level sequence-to-sequence models in terms of several morpho-syntactic grammaticality tests (Sennrich, 2016), Li et al. (2016) introduce a representation erasure technique to measure the amount of contribution of input words and specific phrases to the decision of RNN-based sentiment classifiers, Kádár et al. (2017) examine the linguistic representations of Gated Recurrent Unit-based architectures

for image-sentence ranking, and Chrupała et al. (2017) and Alishahi et al. (2017) analyze the linguistic structure learned by Recurrent Highway Network models of visually grounded speech understanding.

As an alternative approach, a number of *white-box* architectures have been proposed which learn explicit representations of linguistic structure. For example, Dyer et al. (2016) examine Recurrent Neural Network grammars and analyze its learned syntax (Kuncoro et al., 2016). They conclude that their architecture learns a similar notion of headedness as established head-rule sets and the model learns structure similar to traditional nonterminal categories. Williams et al. (2017) explore the learned grammars of two state-of-the-art natural language inference models: SPINN (Bowman et al., 2016) and Gumbel Tree-LSTM (Choi et al., 2017). They find that both SPINN and Gumbel Tree-LSTM have close to or worse than chance-level parsing performance on standard benchmarks. Furthermore, they note that in the case of the SPINN architecture the learned structure depends on the tuning of the model and that their findings are different from the SPINN implementation of Yogatama et al. (2016). They further note that consistency of the parses produced by Gumbel Tree-LSTM across multiple runs is not far off chance level.

The Hierarchical Multi-scale LSTM architecture (Chung et al., 2016a) is in the latter class of white-box models. It is designed specifically to allow for learning explicit structure in data: at each layer, it predicts a binary boundary variable at each time-step. These boundaries correspond to an explicit, interpretable segmentation of the input utterance at multiple levels, which can be thought of as different levels of granularity in linguistic structure. At the same time, it reportedly has state-of-the-art performance on several language modeling benchmarks. As such, we believe the model has the potential to impact future computational cognitive linguistics studies. However, it is not clear whether the high performance of this model demands the acquisition of high-quality linguistic structure from input text. Furthermore, due to the complex architectural design, implementation details might play a large role in the conclusion that can be drawn from the learned structure as in Williams et al. (2017). We examine the segmentation of the input characters by different versions of this model to see whether their quality impacts the overall performance.

2 Hierarchical Multiscale LSTM

Let us introduce the notations used throughout this section with the standard LSTM equations:

$$\begin{bmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{u}_t \\ \mathbf{o}_t \end{bmatrix} = W\mathbf{x}_t + U\mathbf{h}_{t-1} + \mathbf{b} \qquad \text{(gates and candidate)}$$

$$\mathbf{c}_t = \mathbf{c}_{t-1} \odot \sigma(\mathbf{f}_t) + \tanh(\mathbf{u}_t) \odot \sigma(\mathbf{i}_t) \qquad \text{(cell state)}$$

$$\mathbf{h}_t = \sigma(\mathbf{o}_t) \odot \tanh(\mathbf{c}_t) \qquad \text{(hidden state)}$$

The input to an LSTM are the current input $\mathbf{x}_t$, previous state $\mathbf{h}_{t-1}$ and previous cell state $\mathbf{c}_{t-1}$. Variables $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ are used to compute the input gate $\mathbf{i}_t$, forget gate $\mathbf{f}_t$, candidate activation $\mathbf{u}_t$ and output gate $\mathbf{o}_t$. In the second equation $\mathbf{f}_t$ and $\mathbf{i}_t$ are used to trade off the candidate activation $\mathbf{u}_t$ and the previous cell-state $\mathbf{c}_{t-1}$ using the element-wise multiplication $\odot$. This results in the current cell state $\mathbf{c}_t$. Finally, the output gate soft-selects the components of $\tanh(\mathbf{c}_t)$ to write to the hidden state $\mathbf{h}_t$. The function σ refers to the sigmoid function.

HMLSTM – To make the explanation of the HMLSTM architecture more straight-forward, let us first consider the bottom- and top-layer as they are special cases of any ℓ-th layer. In addition, we provide vectorized version equations of the HMLSTM architecture which allow for batch sizes larger than 1 (see Appendix A).

Bottom Layer – The bottom layer of the HMLSTM takes as input: 1) *bottom-up connection*: input at the current time-step $\mathbf{x}_t$; 2) *recurrent connection*: hidden and cell states $< \mathbf{h^1}_{t-1}, \mathbf{c^1}_{t-1} >$ and previous boundary variable z^1_{t-1} from $\ell = 1$; and 3) *top-down connection*: hidden state $\mathbf{h^2}_{t-1}$ from $\ell = 2$. It outputs the state tuple $< \mathbf{h}^1_t, \mathbf{c}^1_t >$ and boundary variable z^1_t.

$$\begin{bmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{u}_t \\ \mathbf{o}_t \\ z_t \end{bmatrix} = W\mathbf{x}_t + U\mathbf{h}^1_{t-1} + z_{t-1}V\mathbf{h}^2_{t-1} + \mathbf{b} \tag{1}$$

The activation function on z_t is the hard-sigmoid followed by the rounding operation to discretize it to $\{0, 1\}$: $z_t := \text{round}(\text{hard\_sigmoid}(z_t))$. The top-down connection is turned on and off by z_{t-1}. When the layer detects a boundary in the previous step (i.e. $z_{t-1} = 1$), takes context from the higher layer through the top-down connection. After computing the activations the layer either runs a regular LSTM UPDATE or does a FLUSH of its input:

$$\mathbf{c}_t = \begin{cases} \mathbf{c}_{t-1} \odot \sigma(\mathbf{f}_t) + \tanh(\mathbf{u}_t) \odot \sigma(\mathbf{i}_t) & \text{if } z_{t-1} = 0 \text{ (UPDATE)} \\ \tanh(\mathbf{u}_t) \odot \sigma(\mathbf{i}_t) & \text{if } z_{t-1} = 1 \text{ (FLUSH)} \end{cases} \tag{2}$$

The final hidden state is given by the regular LSTM update $\mathbf{h}_t = \sigma(\mathbf{o}_t) \odot \tanh(\mathbf{c}_t)$.

Top Layer – The top layer has both bottom-up $\mathbf{h}^{\ell-1}_t, z^{\ell-1}_t$ and recurrent connections $< \mathbf{h^\ell}_{t-1}, \mathbf{c^\ell}_{t-1} >$, but no top-down connection. At each step, it outputs the state tuple $< \mathbf{h}^\ell_t, \mathbf{c}^\ell_t >$:

$$\begin{bmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{u}_t \\ \mathbf{o}_t \end{bmatrix} = z^{\ell-1}_t W\mathbf{h}^{\ell-1}_t + U\mathbf{h}^\ell_{t-1} + b \tag{3}$$

The bottom-up input $\mathbf{h}^{\ell-1}_t$ is gated by $z^{\ell-1}_t$. When $z_t = 1$ the higher layer takes one step and performs UPDATE; otherwise it ignores the current time-step and performs COPY:

$$\mathbf{c}_t = \begin{cases} \mathbf{c}_{t-1} \odot \sigma(\mathbf{f}_t) + \tanh(\mathbf{u}_t) \odot \sigma(\mathbf{i}_t) & \text{if } z_t = 1 \text{ (UPDATE)} \\ \mathbf{c}_{t-1} & \text{if } z_t = 0 \text{ (COPY)} \end{cases} \tag{4}$$

Middle layers – The general HMLSTM layer runs all operations: UPDATE, COPY and FLUSH. It takes as input 1) *bottom-up connection*: lower layer state $\mathbf{h}^{\ell-1}_t$ and boundary variable $z^{\ell-1}_t$; 2) *recurrent connection*: hidden and cell states $< \mathbf{h^\ell}_{t-1}, \mathbf{c^\ell}_{t-1} >$ and boundary variable z^ℓ_{t-1}; and 3) *top-down connection*: hidden state $\mathbf{h}^{\ell+1}_{t-1}$. It outputs the tuple $< \mathbf{h}^\ell_t, \mathbf{c}^\ell_t >$ and boundary variable z^ℓ_t:

$$\begin{bmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{u}_t \\ \mathbf{o}_t \\ z_t \end{bmatrix} = z^{\ell-1}_t W\mathbf{h}^{\ell-1}_t + U\mathbf{h}^\ell_{t-1} + z^\ell_{t-1}V\mathbf{h}^{\ell+1}_{t-1} + b \tag{5}$$

The bottom-up connection is masked by $z^{\ell-1}_t$ as in the top-layer, while the top-down connection is masked by z^ℓ_{t-1} as in the bottom-layer. The variables $z^{\ell-1}_t$, z^ℓ_{t-1} determine which of the three operations the cell runs. FLUSH is executed if the layer detects a boundary in the previous step (i.e. $z^\ell_{t-1} = 1$). If this is not the case and the lower layer does not detect a boundary, the layer runs COPY. However, if the lower layer does detect a boundary $z^{\ell-1}_t$, it runs UPDATE:

$$\mathbf{c}_t = \begin{cases} \tanh(\mathbf{u}_t) \odot \sigma(\mathbf{i}_t) & \text{if } z_{t-1}^{l} = 1 \text{ (FLUSH)} \\ \mathbf{c}_{t-1} & \text{if } z_{t-1}^{\ell} = 0 \text{ and } z_{t}^{\ell-1} = 0 \text{ (COPY)} \\ \mathbf{c}_{t-1} \odot \sigma(\mathbf{f}_t) + \tanh(\mathbf{u}_t) \odot \sigma(\mathbf{i}_t) & \text{if } z_{t-1}^{\ell} = 0 \text{ and } z_{t}^{\ell-1} = 1 \text{ (UPDATE)} \end{cases} \tag{6}$$

Output gate – The output embedding $\mathbf{h}_t^e$ of the HMLSTM is computed by the following gating mechanism:

$$g_t^\ell = \sigma(\mathbf{w}_\ell^T [\mathbf{h}_1; \mathbf{h}_2; \mathbf{h}_3]) \tag{7}$$

$$\mathbf{h}_t^e = \text{ReLU}(g_t^\ell W_\ell^e \mathbf{h}_t^\ell) \tag{8}$$

It takes states from the three layers and computes a gating value for each g_t^1, g_t^2 and g_t^3 using the parameter vector $\mathbf{w}^\ell$. These scalars are used to weight the $W_l^e \mathbf{h}_t^l$ for each layer where W_ℓ^e is a learned parameter. It is important to add some mechanism to combine the activations of the three layers for language modeling as all three layers tick on a different time-scale, but the network has to generate an output at each time-step. This combined embedding $\mathbf{h}_t^e$ is then used in the final softmax layer to predict the probability distribution over the next character.

3 Experiments

3.1 Experimental setup

Upon our request, the authors of the original paper shared a code base with us implementing the HMLSTM architecture and functionality to run the experiments on Penn Treebank (henceforth PTB). The code was implemented in Tensorflow version 1.1 (Abadi et al., 2016), which we ported with minimal modifications to version 1.6 to be able to run it on our system. In the default settings, the code contains ℓ^2-norm weights penalty term with a coefficient of 0.0005, which we also kept fixed across experiments. Furthermore, the given source code has a special initialization strategy used for the weights matrices: until the last column vector it is initialized with orthogonal initialization with standard normal distribution. The last column vector (corresponding to the logit of z) is initialized with Glorot uniform initialization with $\sigma = \sqrt{\frac{6.0}{\text{fanin+fanout}}}$. Due to no initialization details in the original paper we used the initialization defaults provided in the code base for all PTB and Text8 experiments. Lastly, the authors did not refer to publicly available pre-processed versions of the datasets and we used our versions following the details of the paper. We made the following changes to the original code:

Baseline – The authors report LSTM baselines for both PTB and Text8 data sets. The exact configuration of the baseline is not discussed in the paper, however, the authors do share that they implement both layer-normalization and the output module. We train a three-layer stacked LSTM with the same hyperparamters as the HMLSTM.

Layer normalization – The original paper reports results using layer normalization (Ba et al., 2016), which was not implemented in the code base. There are multiple options where to apply it in the case of a LSTM cell. For the HMLSTM cell we implemented it based on equations 20-22 in the paper introducing the technique (Ba et al., 2016). Furthermore, we applied layer-norm to the input and output embedding layers since the performance of the model was found to be unstable otherwise.

Learning-rate schedule and early-stopping – The code base did not include an implementation of the learning-rate schedule that was reported to be used in the original experiments. The learning-rate schedule reported in the paper divides the learning rate by 50 when no improvement is observed in the validation loss. In our code we monitor the validation loss at each epoch. The original code base ran the models for 500 epochs with no early-stopping

and no learning-rate schedule, however, we use early stopping with patience of 4 as very small improvements are observed after reducing the learning rate four times by 50.

Evaluation – We report the bits-per-character (bpc) of the trained models by computing the entropy (in base 2) of the entire test set and dividing it by the number of characters. The code did not implement the evaluation procedure and we have not found the exact details in the paper. Our implementation follows Mikolov et al. (2012): we use batch size of 1 and carry the states of the recurrent model over to the following batch of 100 characters until the whole test sequence is processed.

The experimental results reported in Section 4 are all using this revised implementation, manipulating various ablation factors as described in Section 3.2. Following Chung et al. (2016a), we report results on the following two datasets.

Character-level Penn Treebank – The smaller-scale experiments apply variations of the model on the Penn Treebank dataset (Marcus et al., 1993) using the splits and preprocessing from Mikolov et al. (2012). All models are trained with sequence lengths of 100 and batch size of 64. Before each epoch the dataset is randomly cropped to be divisible by 100. The parameters are optimized with Adam (Kingma and Ba, 2014) with initial learning rate of 0.002, which is divided by 50 if no improvement on the validation data is observed after a full epoch. The norm of the gradient is clipped at 1.0. For all models we use 512 units for all layers and 128 dimension character embeddings as reported in the paper.

Text8 – The Text8 dataset (Mahoney, 2011) is extracted from Wikipedia and is a sequence of 100 million alphabetical characters and spaces. The splits used by Chung et al. (2016a) correspond to the (by now standard) splits introduced in Mikolov et al. (2012): first 90M characters for training, the next 5M for validation and the final 5M characters for testing. On this dataset models are trained on sequence length of 100 and batch size of 128. The initial learning rate of Adam was set to 0.001 and we applied the learning-rate schedule and early-stopping strategy described before. The norm of the gradient is clipped at 1.0. We use 1024 units for the HMLSTM layers and 2048 output embedding units.

Like in the original paper, models are trained to minimze the log-likelihood of the training set and the non-differentiability gap caused by the rounding operation when discretizing the boundary variables is solved by using the straight-through estimator (Bengio et al., 2013).

3.2 Ablation factors

In order to analyze the behavior of the model and the impact of its various architectural design features as well as evaluation settings, we manipulate the following factors in our experiments.

Layer normalization (LN). As mentioned in the previous section, there was a mismatch between the description of the HMLSTM model in Chung et al. (2016a) and the code base provided to us in terms of layer normalization. Therefore we report our experimental results both with and without layer normalization.

Learning-rate schedule (Schedule). Similarly, the paper and the code base do not match in applying learning-rate schedule, therefore we report results with and without scheduling.

Sensitivity to slope (α). Hard-sigmoid (Gulcehre et al., 2016) is a piecewise linear approximation of the sigmoid function, defined as the first-order Taylor expansion of the sigmoid around $x \approx 0$ and clipped between the limits of the original function (0 and 1): $\max(0, \min(1, 0.25x + 0.5))$; where 0.25 is the slope α. The authors describe the slope annealing trick for the PTB experiments and state that they start from slope 0.5 and slowly increase it at maximum until 2.5. We have not found details regarding the α values in other experiments, but used the default $\alpha = 0.5$ in the code-base. We test whether the

hard-sigmoid slope $\alpha = 0.25$ reaches the same performance as the specific choice of $\alpha = 0.5$. Furthermore, we gauge sensitivity to α using values 0.125 and 1.0.

COPY operation on the last layer (CopyLast). The code base by default, only applies the COPY operation at the last layer on the cell state $\mathbf{c}_t^l$, but not on the hidden state $\mathbf{h}_t^l$. In the latter case if the layer runs COPY it takes the previous cell state and applies the new output gate on it along with the tanh function $\mathbf{o}_t \odot \tanh(\mathbf{c}_{t-1})$. This is different from the original formulation of the HMLSTM. The main experiments were run with this default, but we also report the impact of changing this parameter.

Top-down connection (NoTopDown). When the HMLSTM cell runs the FLUSH operation, it computes the new cell-state by $\mathbf{i}_t \odot \mathbf{u}_t$ and the previous cell-state is dropped from the computation. However, both $\mathbf{i}_t$ and $\mathbf{u}_t$ depend on the previous state $\mathbf{h}_{t-1}$. As such, information from the past steps still "leaks" even at FLUSH and each layer can potentially retain enough information for future computations without the use of top-down connections. In our experiments we run the HMLSTM model with and without top-down input to evaluate the impact of this simplification.

Simpler output layer (SimplerOut). The hidden state of the last layer of the HMLSTM is not updated at every time-step and as such it computes a gated embedding combining the representation of all three layers to be able to provide a prediction for every time-step. Here we implement a simplification to the output layer by replacing it with a single affine transformation, embedding the concatenated hidden states: $\mathbf{h}_t^e = \text{ReLU}(W^e[\mathbf{h}_1; \mathbf{h}_2; \mathbf{h}_3])$.

Alternative architecture (HMRNN). One of the motivations for the HMLSTM architecture in the original paper (Chung et al., 2016a) is that the hierarchical multiscale structure might help with the vanishing gradient problem as gradients are back-propagated through fewer time-steps. Here we introduce an alternative architecture that we call HMRNN, which replaces the LSTM updates with simple Elman RNN recurrence. It is a much simpler model and it helps in understanding if the hierarchical multiscale structure itself is enough to alleviate the vanishing gradient problem, or if the LSTM-style updates are essential for the good performance.

$$\mathbf{h}_t = \begin{cases} \tanh(W\mathbf{h}_t^{l-1} + V\mathbf{h}_{t-1}^{l+1}) & \text{if } z_{t-1}^l = 1 \text{ (FLUSH)} \\ \mathbf{h}_{t-1} & \text{if } z_{t-1}^l = 0 \text{ and } z_t^{l-1} = 0 \text{ (COPY)} \\ \tanh(W\mathbf{h}_t^{l-1} + U\mathbf{h}_{t-1}^l) & \text{if } z_{t-1}^l = 0 \text{ and } z_t^{l-1} = 1 \text{ (UPDATE)} \end{cases} \tag{9}$$

The HMRNN architecture implements a simple formulation of the hierarchical multiscale intuition. In case of the FLUSH operation, it only takes as input the hidden state of the lower layer and the hidden state of the higher layer at the previous step. In case of COPY it just uses the previous hidden state. Finally, when running UPDATE it applies the simple Elman network update. See Appendix B for a vectorized implementation.

4 Experimental Results

4.1 Reproducing language modeling results

We attempt to replicate the language modeling experiments of Chung et al. (2016a) on the character level Penn Treebank and Text8 datasets. Table 1 shows the language modeling results reported by Chung et al. (2016a) on both datasets. It also reports results of our reproduction of the original experiments as described in the paper. This means using the HMLSTM architecture with layer normalization on all layers, and with learning-rate scheduling. Since the use of the COPY operation on the last layer is ambiguous and not consistent between the original paper and the code base, we report results both with COPY on and off.

	PTB	Text8
HMLSTM reported by (Chung et al., 2016a)	1.25	1.29
3-layer LSTM reported by (Chung et al., 2016a)	1.29	1.39
3-layer LSTM + Schedule + LN	1.32	1.37
HMLSTM + Schedule + LN	1.27	1.36
HMLSTM + Schedule + LN + CopyLast	1.29	1.36

Table 1: Reproducing language modeling results (in bpc) on PTB and Text8 datasets.

Our most faithful reproduction of the original experiments, which uses the HMLSTM architecture with learning-rate schedule, layer normalization and COPY on the last layer, results in 1.29 bpc on the PTB dataset. Dropping the COPY option improves the results to 1.27 bpc, which is still below the results reported in Chung et al. (2016a) on the same dataset (1.25). Our 3-layer LSTM baseline on PTB also under performs the baseline reported by Chung et al. (2016a)(1.32 compared to 1.29 bpc).

On the Text8 dataset, our results (1.36) are much further from the published ones (1.29), with the HMLSTM performing very close to the 3-layer LSTM baseline (1.37). We did not pursue further experiments on this dataset. The discrepancy could be due to the fact that the script that the authors shared with us was optimized for PTB, and some of the settings might need to be changed for Text8. However, since we have not found such differences reported in the paper, we used the same setting for both datasets. Interestingly, contrary to the HMLSTM results, our implementation of the 3-layer LSTM basline improves the reported baseline performance from 1.39 to 1.37 bpc.

4.2 Ablation results

		BPC	Iter.	z^1	z^2	z-ratio
1	HMLSTM + Schedule + LN + copylast	1.29	10K	0.41	0.25	1.64
2	HMLSTM	1.39	16K	0.23	0.15	1.53
3	HMLSTM + Schedule	1.32	12K	0.25	0.16	1.56
4	HMLSTM + Schedule + LN	1.27	12K	0.42	0.09	4.67
5	HMLSTM + Schedule + LN + α=0.125	1.28	14K	0.41	0.23	1.78
6	HMLSTM + Schedule + LN + α=0.25	1.27	12K	0.31	0.21	1.48
7	HMLSTM + Schedule + LN + α=1.0	4.32	4K	1.0	1.0	1.0
8	NoTopDown + Schedule + LN	1.28	12K	0.64	0.28	2.29
9	SimplerOutput + Schedule + LN	**1.25**	10K	0.64	0.31	2.06
10	3-layer LSTM + Schedule + LN	1.32	9K	N/A	N/A	N/A
11	3-layer LSTM + Schedule + LN + SimplerOut	1.26	12K	N/A	N/A	N/A
12	HMRNN + Schedule + LN	1.40	18K	0.0	1.0	0.0

Table 2: Ablation results on PTB

Since we did not succeed in reproducing the original results on Text8, we only perform the ablation experiments on the PTB dataset. Table 2 summarizes the results of these experiments. On the first row, we repeat our reproduction results using the most faithful replication of the original experiments (HMLSTM + Schedule + LN + CopyLast). We will compare the results from various modifications of the architecture or evaluation setup to these base results.

Layer normalization and learning-rate schedule – As is evident from the results on rows 2–4, adding learning-rate schedule and layer normalization improve the results over the base HMLSTM. Furthermore, the results on row 4 (HMLSTM + Schedule + LN) are better than the reproduction results which also implement the CopyLast feature. Therefore, in the rest

of the experiments, we consistently add layer normalization on all layers as well as learning-rate scheduling, and drop CopyLast.

Sensitivity to alpha – Rows 5–7 show results for different values of the slope α. Using the by definition hard-sigmoid slope $\alpha = 0.25$ instead of the default $\alpha = 0.5$ in the code does not degrade the performance. Lowering alpha to $\alpha = 0.125$ degrades the performance slightly. Since Chung et al. (2016a) mention that they increase the slope up to $\alpha = 2.5$, we also tested the results with a higher value of $\alpha = 1.0$ (row 7), which caused the model to diverge after 4,000 iterations.

Architectural modification: top-down connection and simpler output – Interestingly, as can be seen from row 8, removing the top-down connections (and keeping the model structure fixed otherwise) only degrades the performance by a small margin; from 1.27 bpc to 1.28. Even more surprising is the effect of using a simpler output layer: replacing the gated-output layer by our simple output module improves the results from 1.27 to 1.25. This, in fact, is the best results we have achieved. Similarly rows 10 and 11 show that the simpler output layer improves the baseline 3-layer LSTM performance from 1.32 bpc to a competitive 1.26 bpc.

Alternative architecture: HMRNN – The simplified architecture which replaces the LSTM updates with Elman RNN produces results that are much worse (1.40) than the HMLSTM counterpart (1.27). Our ablation experiments show that using layer normalization and learning-rate schedule is beneficial, high values for the slope of the sigmoid function hurt performance, but the model performance is robust otherwise to smaller α values. The LSTM updates are essential for good performance of the HMLSTM model. Finally, simplifying certain aspects of the model such as removing the top-down connection might not hurt much, and in other cases such as simplifying the output layer it can actually enhance the overall performance.

4.3 Segmentation results

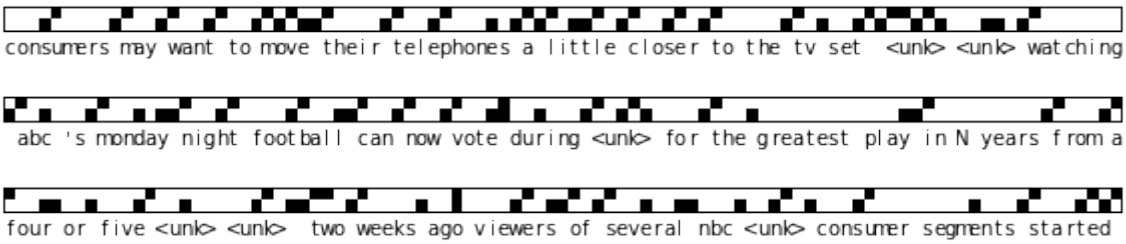


Figure 1: Segmentation examples. Black means $z_t = 1$, white $z_t = 0$.

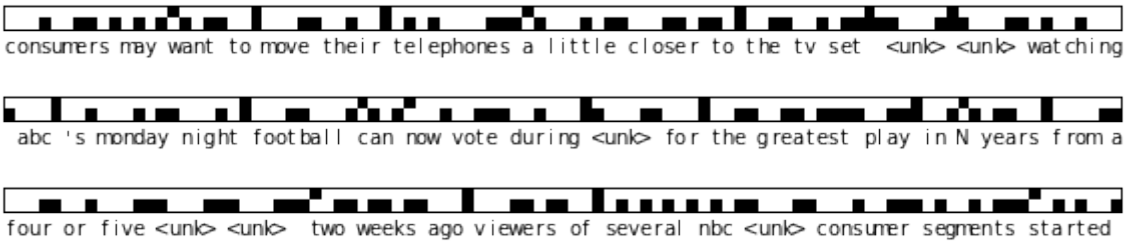


Figure 2: Segmentation examples with layer normalization. Black means $z_t = 1$, white $z_t = 0$.

The HMLSTM model predicts binary boundary variables z_t^1 and z_t^2 at each time-step, which correspond to a segment of the input utterance at each level. The frequencies of z_t^1 and z_t^2 provide a high-level description of segmentation properties of the network and are reported in Table 2. Based on the hierarchical multiscale intuition, the expected behavior is that the z_t^1

frequency is higher than z_t^2 as the former corresponds to words or morpheme-like units, whereas the latter corresponds to larger chunks. The ratio of z_t^1 to z_t^2 frequencies is reported in the last column of Table 2.

Running the HMLSTM default configuration with added layer-normalization and learning-rate schedule results in the largest separation in the time-scale of the layers as indicated by the z-ratio of 4.67. Interestingly, only changing the α value to 0.125 or 0.25 results in a much narrower gap (1.78 and 1.48 respectively), without a big impact on bpc. Furthermore, the HMLSTM with schedule, layer-norm and $\alpha = 0.125$ in row 5 performs on the same level as our NoTopDown ablation architecture, with a smaller z-ratio.

The segmentation examples reported in the original paper show the first layer of the HMLSTM segmenting the sequence at word boundaries (i.e. spaces), and the higher layer segments corresponding to multi-word chunks. Figure 1 shows segmentation results of the HMLSTM with learning-rate schedule and Figure 2 with the added layer normalization. Our runs could not reproduce the segmentation results visualized in the original paper and overall we do not find a relationship between the performance of the model and segmentation behavior.

5 Conclusion

In our attempt to reproduce the results in Chung et al. (2016a) we re-ran the HMLSTM model on two datasets: PTB and Text8. On PTB the final performance of the model almost matches the original results, but not quite. Our best result was achieved by a slight modification to the COPY operation of the last-layer provided in the source-code, but not detailed in the paper and with our added simplification to the output layer.

We could not reproduce the results on Text8 using the same code base received from the authors. This might be due to the non-availability of the pre-processed version of the dataset. Another potential source might be the lack of detail in the paper on the initialization scheme and weight penalty, which led us to keep these implementation details constant across datasets. Furthermore, the layer normalization might have been implemented on different layers across different datasets. We have made several attempts until reaching the conclusion that it needs to be implemented on every layer of the HMLSTM. This was found to perform best on PTB, but might not be the best setting for Text8. Similarly, the learning-rate schedule was applied based on the monitoring of the loss after every epoch, but this was an informed guess.

Two simplifications were applied to the architecture successfully: 1) removing top-down connections only slightly degraded performance, and 2) simplifying the output layer improved performance in our experiments. There is space for possible modifications that fell out of the scope of the current work like applying REINFORCE gradients in place of the straight-through estimator.

We could not reproduce the segmentation results provided in the paper. Furthermore, we did not observe a close relationship between the segmentation behavior and the final performance. Changing the slope variable α to 0.25 from 0.5, while keeping all other details constant, resulted in the same performance, but huge difference in segmentation behavior.

The observations made here are not specific to the work of Chung et al. (2016a) and the HMLSTM architecture. Similar results have been reported recently by Williams et al. (2017), whose re-implementation of the considered architecture did not produce the reported performances. One of their considered models learned a qualitatively different latent structure than another re-implementation by Yogatama et al. (2016) and the other architecture did not converge to the same structure across runs. The HMLSTM architecture was considered for reproduction due to its intriguing property of learning interpretable structure from character-level input. The detailed reproduction and ablation study was provided to surface some of the potential difficulties that can hinder the applicability of such models for future studies.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI*. volume 16, pages 265–283.

Afra Alishahi, Marie Barking, and Grzegorz Chrupała. 2017. Encoding of phonology in a recurrent neural model of grounded speech. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. pages 368–378.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* .

Monya Baker. 2016. 1,500 scientists lift the lid on reproducibility. *Nature News* 533(7604):452.

C Glenn Begley and Lee M Ellis. 2012. Drug development: Raise standards for preclinical cancer research. *Nature* 483(7391):531.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* .

Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. *arXiv preprint arXiv:1603.06021* .

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current and new directions in discourse and dialogue*, Springer, pages 85–112.

Jihun Choi, Kang Min Yoo, and Sang goo Lee. 2017. Learning to compose task-specific tree structures. AAAI.

Grzegorz Chrupała, Lieke Gelderloos, and Afra Alishahi. 2017. Representations of language in a model of visually grounded speech signal. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 613–622.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2016a. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704* .

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016b. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147* .

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. *arXiv preprint arXiv:1602.07776* .

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

David Golub and Xiaodong He. 2016. Character-level question answering with attention. *arXiv preprint arXiv:1604.00727* .

Caglar Gulcehre, Marcin Moczulski, Misha Denil, and Yoshua Bengio. 2016. Noisy activation functions. In *International Conference on Machine Learning*. pages 3059–3068.

Tobias Horsmann and Torsten Zesch. 2017. Do lstms really work so well for pos tagging?–a replication study. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 727–736.

Lisa Hutchinson and Rebecca Kirk. 2011. High drug attrition rates—where are we going wrong?

Akos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2017. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics* 43(4):761–780.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference for Learning Representations*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A Smith. 2016. What do recurrent neural network grammars learn about syntax? *arXiv preprint arXiv:1611.05774* .

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220* .

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *arXiv preprint arXiv:1611.01368* .

Matt Mahoney. 2011. Large text compression benchmark. *URL: http://www. mattmahoney. net/text/text. html* .

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

Edison Marrese-Taylor and Yutaka Matsuo. 2017. Replication issues in syntax-based aspect extraction for opinion mining. *arXiv preprint arXiv:1701.01565* .

Margot Mieskes. 2017. A quantitative study of data in the nlp community. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*. pages 23–29.

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. 2012. Subword language modeling with neural networks. *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)* .

Mathieu Morey, Philippe Muller, and Nicholas Asher. 2017. How much progress have we made on rst discourse parsing? a replication study of recent results on the rst-dt. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 1319–1324.

Open Science Collaboration. 2015. Estimating the reproducibility of psychological science. *Science* 349(6251):aac4716.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv preprint arXiv:1604.05529* .

Rico Sennrich. 2016. How grammatical is character-level neural machine translation? assessing mt quality with contrastive translation pairs. *arXiv preprint arXiv:1612.04629* .

KM Venables, N Farrer, L Sharp, BJ Graneek, and AJ Newman Taylor. 1993. Respiratory symptoms questionnaire for asthma epidemiology: validity and reproducibility. *Thorax* 48(3):214–219.

Dagmar Waltemath, Richard Adams, Frank T Bergmann, Michael Hucka, Fedor Kolpakov, Andrew K Miller, Ion I Moraru, David Nickerson, Sven Sahle, Jacky L Snoep, et al. 2011. Reproducible computational biology experiments with sed-ml-the simulation experiment description markup language. *BMC systems biology* 5(1):198.

Walter C Willett, Laura Sampson, Meir J Stampfer, Bernard Rosner, Christopher Bain, Jelia Witschi, Charles H Hennekens, and Frank E Speizer. 1985. Reproducibility and validity of a semiquantitative food frequency questionnaire. *American journal of epidemiology* 122(1):51–65.

Adina Williams, Andrew Drozdov, and Samuel R Bowman. 2017. Learning to parse from a semantic objective: It works. is it syntax? *arXiv preprint arXiv:1709.01121* .

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2016. Learning to compose words into sentences with reinforcement learning. *arXiv preprint arXiv:1611.09100* .

A Vectorized formulation of the HMLSTM

Here are the vectorized equations for the bottom, middle and top layers of the HMLSTM. Note, the computation process should be executed from top to bottom and left to right when there are multiple equations.

Bottom layer

$$\mathbf{c}_t = (1 - z_{t-1})\mathbf{c}_{t-1} \odot \sigma(\mathbf{f}_t) + \tanh(\mathbf{u}_t) \odot \sigma(\mathbf{i}_t) \tag{10}$$

Top layer

$$\begin{aligned} \hat{\mathbf{c}}_t &:= \mathbf{c}_{t-1} \odot \sigma(\mathbf{f}_t) + \tanh(\mathbf{u}_t) \odot \sigma(\mathbf{i}_t) & \hat{\mathbf{h}}_t &:= \sigma(\mathbf{o}_t) \odot \tanh(\mathbf{c}_t) \\ \mathbf{c}_t &:= z_t\hat{\mathbf{c}}_t + (1 - z_t)\mathbf{c}_{t-1} & \mathbf{h}_t &:= z_t\hat{\mathbf{h}}_t + (1 - z_t) \odot \mathbf{h}_{t-1} \end{aligned} \tag{11}$$

Middle layer

$$\begin{aligned} \mathrm{c}_m &= (1 - z_{t-1}^{\ell})(1 - z_t^{\ell-1}) & \mathbf{c}_t &= \mathbf{u}_g + \mathrm{c}_m(\mathbf{c}_{t-1} - \mathbf{u}_g) + \mathrm{u}_m(\sigma(\mathbf{f_t}) \odot \mathbf{c}_{t-1})) \\ \mathrm{u}_m &= (1 - z_{t-1}^{\ell})z_t^{\ell-1} & \mathbf{h}_t &:= \sigma(\mathbf{o}_t) \odot \tanh(\mathbf{c}_t) \\ \mathbf{u}_g &= \sigma(\mathbf{i_t}) \odot \tanh(\mathbf{u_t}) & \mathbf{h}_t &:= \mathrm{c}_m\mathbf{h}_t + \mathrm{c}_m \odot \mathbf{h}_{t-1} \end{aligned} \tag{12}$$

Variables $\mathrm{c}_m, \mathrm{u}_m$ and $\mathbf{u}_g$ stand for copy-mask, update-mask and gated-candidate activation respectively.

B Vectorized formulation of the HMRNN

Here are the vectorized equations for the bottom, middle and top layers of the HMRNN. Note, the computation process should be executed from top to bottom and left to right when there are multiple equations.

Bottom layer

$$\mathbf{h}_t = \tanh(W\mathbf{x}_t + (1 - z_{t-1}^{l})U\mathbf{h}_{t-1}^{l} + z_{t-1}^{l} \times V\mathbf{h}_{t-1}^{l+1}) \tag{13}$$

Top layer

$$\mathbf{h}_t = (1 - z_t^{l-1}) \times \mathbf{h}_{t-1} + z_t^{l-1} \times \tanh(W\mathbf{x}_t + U\mathbf{h}_{t-1}^{l}) \tag{14}$$

Middle layer

$$\begin{aligned} \mathrm{c}_m &= (1 - z_{t-1}^{\ell})(1 - z_t^{\ell-1}) & & \\ \mathrm{u}_m &= (1 - z_{t-1}^{\ell})z_t^{\ell-1} & \mathrm{h}_f &= W\mathbf{h}_t^{l-1} + V\mathbf{h}_{t-1}^{l+1} \\ \mathbf{h}_t &= \mathrm{c}_m\mathbf{h}_{t-1} + (1 - \mathrm{c}_m)\tanh((1 - \mathrm{u}_m)\mathrm{h}_f + \mathrm{u}_m\mathrm{h}_u)) & \mathrm{h}_u &= W\mathbf{h}_t^{l-1} + U\mathbf{h}_{t-1}^{l} \end{aligned} \tag{15}$$

Variables c_m and u_m stand for copy-mask and update-mask activation respectively.

Character-Level Feature Extraction with Densely Connected Networks

Chanhee Lee[1], Young-Bum Kim[2], Dongyub Lee[1], HeuiSeok Lim[1]*
[1]Korea University, Republic of Korea
{chanhee0222, judelee93, limhseok}@korea.ac.kr
[2]Amazon Alexa
youngbum@amazon.com

Abstract

Generating character-level features is an important step for achieving good results in various natural language processing tasks. To alleviate the need for human labor in generating hand-crafted features, methods that utilize neural architectures such as Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN) to automatically extract such features have been proposed and have shown great results. However, CNN generates position-independent features, and RNN is slow since it needs to process the characters sequentially. In this paper, we propose a novel method of using a densely connected network to automatically extract character-level features. The proposed method does not require any language or task specific assumptions, and shows robustness and effectiveness while being faster than CNN- or RNN-based methods. Evaluating this method on three sequence labeling tasks - slot tagging, Part-of-Speech (POS) tagging, and Named-Entity Recognition (NER) - we obtain state-of-the-art performance with a 96.62 F1-score and 97.73% accuracy on slot tagging and POS tagging, respectively, and comparable performance to the state-of-the-art 91.13 F1-score on NER.

1 Introduction

Effectively extracting character-level features from words is crucial in many Natural Language Processing (NLP) tasks, such as Named Entity Recognition (NER), Part-of-Speech (POS) tagging, and Slot tagging. Thus, most state-of-the-art methods for these tasks exploit some kind of character-level features (Huang et al., 2015; dos Santos and Zadrozny, 2014). Recently, generating character-level features with neural architectures such as Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN) has drawn much attention, mainly because it doesn't require human labor and shows superior performance (Ma and Hovy, 2016; dos Santos and Zadrozny, 2014). However, CNN struggles at distinguishing anagrams, and RNN is inherently slow due to its sequential nature.

In this paper, we propose an effective and efficient way of extracting character-level features using a densely connected network. The key benefits of the proposed method can be summarized as follows. First, it does not require any hand-crafted features or data preprocessing. Each word is processed based on n-gram statistics of the training data, and vectorized using bag-of-characters. Additional features are based on hexadecimal values of the character-set (e.g. UTF-16) and number of characters in the word. Second, it extracts effective character-level features while being efficient. State-of-the-art performance can be achieved using this method, and the feature extraction is done with a simple densely connected network with a single hidden layer. Third, it doesn't depend on features that are language or task specific, such as character type features or gazetteer (i.e. lists of known named entities such as cities or organization names). The only requirement for adopting this method is that the language should be processable as a sequence of words, which is made of sequence of characters. These benefits, combined with minimum requirements for application, make the proposed method an easy replacement for conventional methods such as CNN or RNN.

* corresponding author

Proceedings of the 27th International Conference on Computational Linguistics, pages 3228–3239
Santa Fe, New Mexico, USA, August 20-26, 2018.

Our contributions are three-fold: 1) We propose an effective yet efficient method for character-level feature extraction; 2) We quantitatively show that the proposed method is superior to CNN and RNN via extensive evaluation; 3) We achieve state-of-the-art or comparable to state-of-the-art performance on three of the most popular and well-studied sequence tagging tasks - Slot tagging, Part-of-Speech (POS) tagging, and Named Entity Recognition (NER).

2 Related Work

Prior to the introduction of neural architectures for character-level feature generation, manually engineered features were designed by experts based on language and/or domain knowledge. One example is word shape, in which each word is mapped to a simplified representation that encodes information such as capitalization, numerals, and length (e.g. CoNLL-2003 to AaAAA-0000). Finkel et al. (2005) combined this feature with other information such as n-grams and gazetteers to train a conditional Markov model for identification of gene and protein names in biomedical documents. Huang et al. (2015) introduced more hand-crafted features utilizing punctuation or non-letters and used these as an input to a Bi-LSTM-CRF tagger for POS tagging, CoNLL-2000 chunking, and CoNLL-2003 NER. Even though these kinds of hand-crafted features showed strong empirical results, they are more expensive than our approach in that they require expert knowledge of the target domain and language.

In recent years, methods that utilize neural networks to automatically extract character-level features have been proposed. The most widely adopted and successful method for this is CNN. dos Santos and Zadrozny (2014) combined this approach with a window-based fully-connected neural network tagger to perform English and Portuguese POS tagging. This work achieved state-of-the-art results in Portuguese and near state-of-the-art results in English. In Ma and Hovy (2016), a Bi-LSTM-CRF model incorporated with a character-level CNN is trained in an end-to-end fashion. They evaluated this approach on English POS tagging and NER, achieving state-of-the-art performance on both tasks. However, feature vectors generated by CNN are position-independent due to the max-over-time pooling layer, and are more sensitive to model weight initialization compared to the method proposed in this paper.

Another effective way of generating feature vectors from a variable length sequence of characters is to use RNN. For instance, Lample et al. (2016) extracted character-level features using a bi-directional LSTM and used them with pre-trained word embeddings as word representations for another Bi-LSTM-CRF model. Evaluating this model for NER, they obtained state-of-the-art results for Dutch, German, and Spanish, and close to state-of-the-art results for English. Intuitively, character-level feature generation via RNN should be more effective than CNN, since RNN processes each character sequentially and thus should form a better model of character ordering. However, Reimers and Gurevych (2017) empirically showed that these two methods have no statistically significant difference in terms of performance. Furthermore, RNN has a higher time-complexity caused by its sequential nature, which makes it less favorable.

3 Proposed Method

The proposed method is built on bag-of-characters (BOC) representation. However, BOC is prone to anagrams and thus is susceptible to word collisions, i.e. different words having the same vector representation. The main focus of the proposed method is to minimize word collision while maintaining the key benefits described above. To achieve this goal, we split the word into k pieces, and each piece is vectorized using BOC. Then, two non hand-crafted-features are extracted from the word - character order and word length. These sparse vectors are concatenated and normalized to form the sparse character-level feature vector. For a n-dimensional vector $\mathbf{x} = \langle x_1, x_2, \cdots, x_n \rangle$, normalizing is done as follows:

$$x'_i = \frac{x_i}{\sum_{j=1}^{n} x_j} \tag{1}$$

This sparse vector is then fed into a densely connected network with a single hidden layer to obtain the final dense character feature vector. Note that the sparse vector representation of each word is fixed, so it can be cached for efficiency. Figure 1 illustrates the overall process.

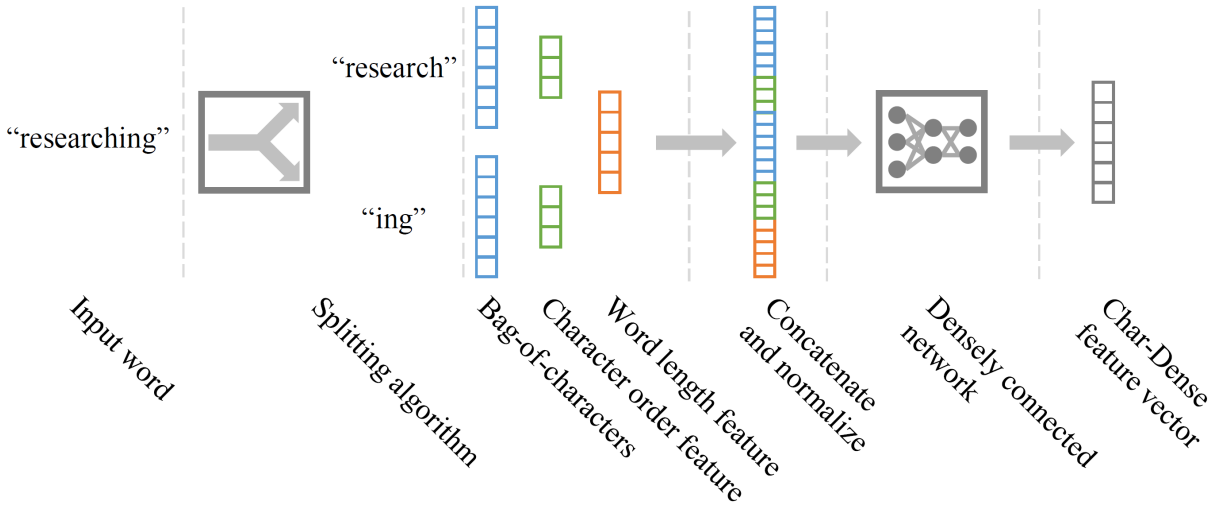


Figure 1: Process of generating the character-level feature vector of a word using the proposed method.

3.1 Splitting Words

Each word is split into k pieces to reduce the number of word collisions. To maintain the ordering of pieces, concatenation is used instead of summation or averaging to merge the vectors. Word splitting is done based on n-gram frequency. First, n-gram statistics C_{ng} is collected from the training corpus where $C_{ng}(s)$ is the number of times the n-gram s appears in the corpus. Then, the n-gram with the highest frequency gets merged into a single piece, and this merging is repeated until only k n-grams are left. The number of pieces k per word is a configurable hyperparameter. Finally, each piece is converted into a fixed length vector using BOC. The detailed algorithm is presented in Algorithm 1. This process is similar to the byte-pair encoding method in Sennrich et al. (2015), except that in the proposed method each word can only be split into k pieces whereas byte-pair encoding produces an arbitrary number of pieces. Producing a fixed number of pieces is important, since concatenation is used to merge the vectors.

Algorithm 1: Splitting word into k pieces

Input : word $w = (c_1, c_2, \cdots, c_n)$, n-gram statistics C_{ng}, number of pieces k
Output: $S = (s_1, \cdots, s_k)$ where $s_1 + s_2 + \cdots + s_k = w$

$S \leftarrow w$
while $|S| > k$ **do**
  $m = \underset{i}{\operatorname{argmax}}\, C_{ng}(c_i + c_{i+1})$
  $S \leftarrow (\cdots, s_{m-1}, s_m + s_{m+1}, s_{m+2}, \cdots)$
end
while $|S| < k$ **do**
  Append empty string to S
end
return S

3.2 Character Order Feature

Every character that has a digital representation can be converted into a numerical value via some character-set (e.g. UTF-16). Then, it is possible to numerically compare two characters. Let $T = \{c_1, c_2, \cdots, c_n\}$ be a character sequence of length n. Then $F_{asc}(T, k)$, $F_{des}(T, k)$, $C_{asc}(T)$, and $C_{des}(T)$ are defined as follows:

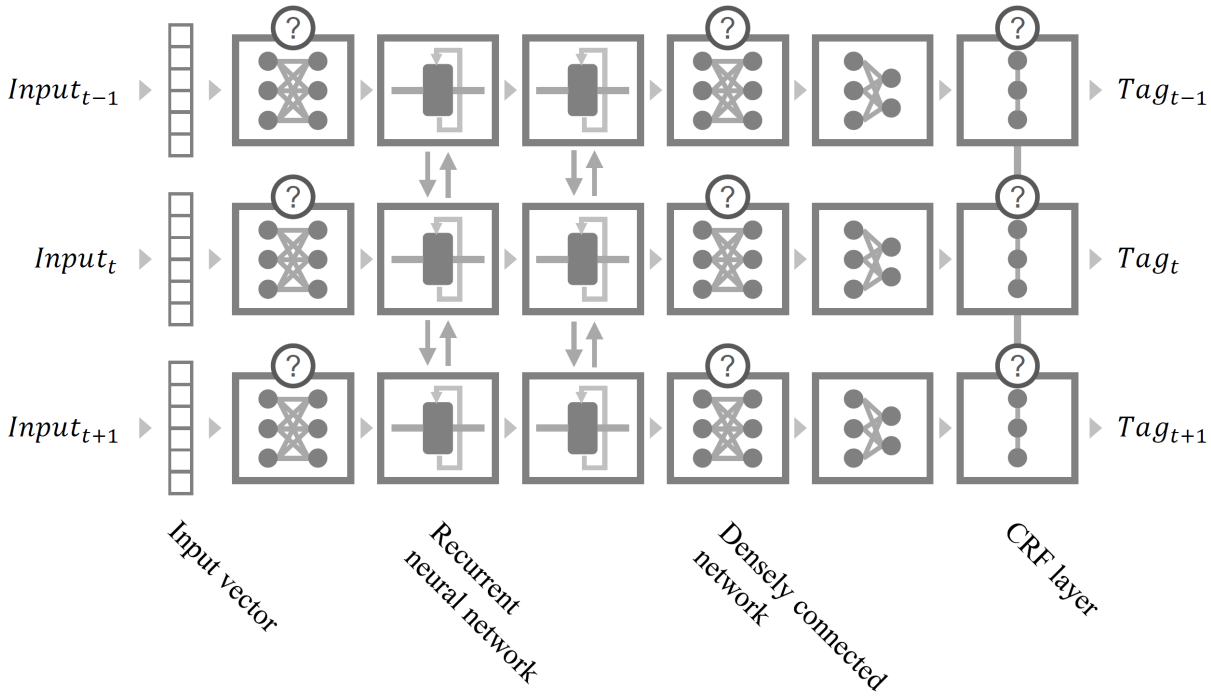


Figure 2: Overview of model architecture for sequence tagging experiments. Question mark indicates that the component is optional.

$$F_{asc}(T,k) = \begin{cases} 1, \text{if } c_k < c_{k+1} \\ 0, \text{otherwise} \end{cases}, \quad F_{des}(T,k) = \begin{cases} 1, \text{if } c_k > c_{k+1} \\ 0, \text{otherwise} \end{cases} \tag{2}$$

$$C_{asc}(T) = \sum_{k=1}^{n-1} F_{asc}(T,k), \quad C_{des}(T) = \sum_{k=1}^{n-1} F_{des}(T,k) \tag{3}$$

Bi-grams with the same character repeating are ignored. A sequence of characters can then be categorized into one of three classes: $C_{asc}(T) > C_{des}(T)$, $C_{asc}(T) = C_{des}(T)$, $C_{asc}(T) < C_{des}(T)$. This category info is calculated for each word piece, which is then converted into a 3-dimensional vector using one-hot encoding and concatenated to the sparse word piece vector.

3.3 Word Length Feature

To further reduce the number of word collisions, information about the word's length is added into the model. One-hot encoding is used to store an integer from 0 to 20, and any word exceeding 20 characters is treated as being 20 characters long.

4 Model

In this section, we describe the sequence tagging model's architecture in detail. Figure 2 illustrates the model architecture.

4.1 Sequence Tagging with Bidirectional RNN

In sequence tagging tasks, such as POS tagging or NER, both future and past input tokens are available to the model. Bidirectional RNNs (Graves and Schmidhuber, 2005) can efficiently make use of future and past features over a certain time frame. We use Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) for our RNN cell, which is better at capturing long-term dependencies than vanilla RNN. Output of the forward and backward RNN layers are summed to form the feature vector of each time-step. Each word is tagged based on this feature vector, using either a softmax layer or CRF layer.

To capture a more abstract and higher-level representation in different layers, a densely connected layer can be added before and after the Bi-LSTM layers. The input to this network at each time-step is the concatenation of the character-level feature vector and a pre-trained word vector (described in section 5).

4.2 Conditional Random Field

Even though a Bi-LSTM layer can efficiently extract features for each time-step utilizing past and future inputs, the prediction is made on each time-step, independent of past and future tag outputs. The Conditional Random Field (CRF) layer overcomes this limitation by considering state transition probability, thereby decoding the most probable output tag sequence. It has been shown that adding a CRF layer on top of a Bi-LSTM network can lead to statistically significant performance increases (Reimers and Gurevych, 2017). We also test a variant of our model using CRF as the final layer to perform tag sequence prediction.

4.3 Stacking RNNs with Residual Connection

Increasing the depth of the neural network architecture has proven to be an effective way of improving performance. However, naively stacking layers can lead to adversarial effects due to the degradation problem. Residual connection (He et al., 2016) has shown to be an effective way to tackle this issue by creating a shortcut between layers. The same strategy is adopted to our model when there are more than one Bi-LSTM layers, in which case the input is added to the Bi-LSTM layer's output.

4.4 Dropout

Dropout is a popular and effective way of regularizing neural network models, by randomly dropping nodes (Srivastava et al., 2014). In our model, Inverted dropout is applied to all densely connected layers for regularization. For the Bi-LSTM layers, variational recurrent dropout (Gal and Ghahramani, 2016) is used, since naive dropout can deteriorate performance. The word embedding matrix is regularized using the method proposed in Gal and Ghahramani (2016), i.e. dropping words at random.

5 Training Details

Pre-trained Word Embeddings Utilizing word embeddings pre-trained on large unlabeled text has shown to be one of the most effective ways to increase performance on various NLP tasks. Our model uses the GloVe (Pennington et al., 2014) 300-dimensional vectors trained on the Common Crawl corpus with 42B tokens as word level features, as this resulted in the best performance in preliminary experiments. Words that do not appear in the training data are replaced with a special Out-of-Vocabulary (OOV) token. To train the vector of this token, we randomly swap words with OOV tokens while training with a 0.01 probability, as in Lample et al. (2016). The word vector is then concatenated with the character-level feature vector and fed into the subsequent layer.

Freezing Embeddings It is common practice to fine-tune the pre-trained word vectors through the training process. However, preliminary experiments have revealed that fine-tuning the word vectors results in lower performance than freezing the vectors, especially in the early stages of training. We hypothesize that randomly initialized weights in the model act as noise and degrade the pre-trained word vectors. To circumvent this issue, the embeddings are frozen for the first T_{freeze} phase of training so that they are not affected by untrained weights. We use $T_{freeze} = 20\%$ for all experiments.

Dynamic Batch Size Keskar et al. (2016) showed that small batch sizes lead to more global and flat minimizers, while large batch sizes lead to more local and sharp minimizers. Therefore, starting from a small batch size and increasing it during training would result in a more global, but sharp minimizer. While having similar effect to learning rate decay, this strategy also has a benefit of accelerating the training as the batch size grows (Smith et al., 2017). Adopting this method, we start from a fixed initial batch size, and increase the batch size by a factor of two on each quarter of the course of training.

Tagging Scheme It is reported that more complicated tagging schemes such as IOBES does not have statistically significant advantage over BIO scheme (Reimers and Gurevych, 2017), thus we adopt the BIO scheme for all experiments.

Dataset	ATIS		PTB WSJ		CoNLL2003	
	Sentences	Tokens	Sentences	Tokens	Sentences	Tokens
Training	4978	56591	38219	912344	14987	204567
Develop.	-	-	5527	131768	3466	51578
Test	893	9198	5462	129654	3684	46666

Table 1: Corpus statistics of each task.

Parameter Optimization Our network is trained by minimizing the cross entropy loss over the tags for the softmax model, or maximizing the log-likelihood of the tag sequence for the CRF model. The objective function is optimized using the gradient-based optimization algorithm Adam (Kingma and Ba, 2014). For all experiments, we implement the model using the TensorFlow (Abadi et al., 2016) library.
Hyperparameter Tuning Most hyperparameters, with the following exceptions, are tuned on the development sets. Hyperparameters of the character-CNN and character-RNN models are adopted from Ma and Hovy (2016) and Lample et al. (2016), respectively. The chosen hyperparameters for all experiments are summarized in Appendix A.

6 Evaluation

We evaluate the effectiveness of the proposed method using three of the most well-studied and common English sequence tagging tasks - Slot tagging, POS tagging, and NER. Note that to test the generalizability of the proposed method, we do not perform any preprocessing for all experiments. Details on each task and baseline models are described in this section. Table 1 summarizes the statistics of each task.

6.1 Slot Tagging

For slot tagging, we use the Airline Travel Information System (ATIS) dataset. This dataset has 84 types of slot labels and 127 possible tags with BIO tagging scheme. Since this corpus lacks a development set, 20% of the training data is randomly sampled and used as the development set for tuning the hyperparameters. This task's performance is measured in F1-score, which is calculated using the publicly available *conlleval.pl* script.

6.2 Part-of-Speech Tagging

For POS tagging, we use the Wall Street Journal (WSJ) portion of the Penn TreeBank dataset (Marcus et al., 1993) and adopt the standard split for part-of-speech tagging experiments - section 0-18 as training data, section 19-21 as development data, and section 22-24 as test data. This dataset contains 45 different POS tags. Model performance is measured by token-level accuracy.

6.3 Named Entity Recognition

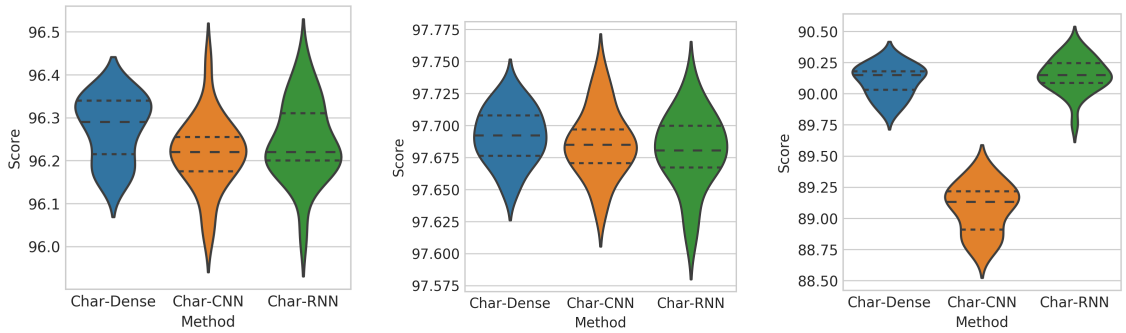
For NER, the English portion of the CoNLL-2003 shared task (Tjong Kim Sang and De Meulder, 2003) is used for evaluation. This dataset contains four different types of named entities, which results in nine possible tags with BIO tagging scheme and an 'O' tag. Like slot tagging, the final performance is measured in F1-score using the same *conlleval.pl* script.

6.4 Baseline Models

Character-level CNN and character-level RNN are the most effective and widely adopted methods for character-level feature extraction, and thus are suitable as strong baseline methods. We implement these two methods to use them as baselines for comparison. The CRF layer has the effect of making the model robust to architectural differences (Reimers and Gurevych, 2017). Since the goal of baseline experiments is to evaluate the effect of difference in character-level feature generation methods, we use the softmax layer instead of the CRF layer for these experiments. Every aspect of the sequence tagging model except the character-level feature generation method is identical for all baseline experiments.

Method	Task		
	Slot	POS	NER
Char-CNN	96.22 (SD 0.08)	97.68 (SD 0.03)	89.08 (SD 0.20)
Char-RNN	96.25 (SD 0.09)	97.68 (SD 0.03)	**90.15** (SD 0.14)
Char-Dense (Ours)	**96.28** (SD **0.07**)	**97.69** (SD **0.02**)	90.10 (SD **0.13**)

Table 2: Comparison with baseline models.



(a) Slot (b) POS (c) NER

Figure 3: Score distributions for all experiments. Quartiles marked with dashed lines.

7 Results and Discussion

7.1 Experimental Results

For a more in-depth analysis of the performance of the proposed method and two baselines, we train each model 20 times with different initial parameters, which are randomly initialized (Reimers and Gurevych, 2017). Table 2 summarizes the mean performance with standard deviation in parentheses. Performance distribution is also visualized using a violin plot in Figure 3.

7.1.1 Slot Tagging

On the task of tagging semantic slots using the ATIS dataset, the proposed method shows the best results in terms of both performance and variability. Our method has the highest mean F1-score of 96.28. Furthermore, it has the lowest standard deviation across all runs, which means it is robust to parameter initialization. On the contrary, both CNN and RNN models have lower performance and higher variability compared to the proposed method.

Analyzing the violin plot reveals that there are also differences in score distribution. While CNN models tend to have a low F1-score on average with occasional high peaks, RNN models have higher F1-score in general but suffer from a large performance drop with poor parameter initialization. This could be one of the reasons why models using CNN seem to have superior performance when only the best performance is reported. On the other hand, our model does not result in peaks or serious drops in performance with different seed values, which makes it more suitable for real-world applications.

7.1.2 Part-of-Speech Tagging

The proposed method also achieves the best results on the POS tagging task. Similar to the slot tagging task, our method shows the highest mean accuracy of 97.69 with the lowest standard deviation of 0.02. For the baseline models, CNN and RNN performed on par.

CNN-based models have higher variability with high peak performance on this task also, as shown in the violin plot. Similar to the slot tagging task, our method shows the lowest variability, which supports the robustness of this method.

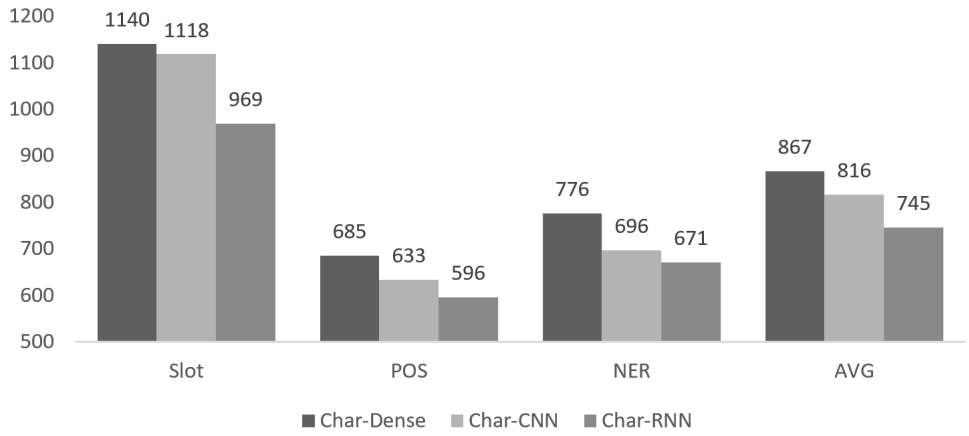


Figure 4: Sentence processing speed in terms of number of sentences per second.

Slot		**POS**		**NER**	
Approach	F1	Approach	Acc.	Approach	F1
Mesnil et al. (2015)	94.73	Toutanova et al. (2003)	97.24	Ando and Zhang (2005)	89.31
Yao et al. (2014)	94.85	Manning (2011)	97.32	Collobert et al. (2011)	89.59
Liu and Lane (2015)	94.89	Shen et al. (2007)	97.33	Huang et al. (2015)	90.10
Yao et al. (2014)	95.08	Sun (2014)	97.36	Chiu and Nichols (2015)	90.77
Peng and Yao (2015)	95.25	Moore (2015)	97.36	Ratinov and Roth (2009)	90.80
Vu et al. (2016)	95.56	Hajič et al. (2009)	97.44	Lin and Wu (2009)	90.90
Vu (2016)	95.61	Søgaard (2011)	97.50	Passos et al. (2014)	90.90
Kurata et al. (2016)	95.66	Tsuboi (2014)	97.51	Lample et al. (2016)	90.94
Zhu and Yu (2017)	95.79	Huang et al. (2015)	97.55	Luo et al. (2015)	91.20
Zhai et al. (2017)	95.86	Choi (2016)	97.64	Ma and Hovy (2016)	**91.21**
Char-Dense w/o CRF (Ours)	96.36	Char-Dense w/o CRF (Ours)	**97.73**	Char-Dense w/o CRF (Ours)	90.28
Char-Dense w/ CRF (Ours)	**96.62**	Char-Dense w/ CRF (Ours)	97.65	Char-Dense w/ CRF (Ours)	91.13

Table 3: Comparison with state-of-the-art approaches in the literature.

7.1.3 Named Entity Recognition

On the NER task, the RNN-based model has a slightly better F1-score (90.15) than the proposed method (90.10). However, our method consistently shows the lowest standard deviation, like as the other tasks, at 0.13. By analyzing the violin plot, we can see that the RNN again shows occasional performance drops for certain cases of poor weight initialization. Unlike the other two tasks, the model utilizing CNN has a relatively poor F1-score and does not show any peaks in performance.

7.2 Training Speed

To compare the efficiency of three models, average training speed (i.e. number of sentences processed per second) is presented in Figure 4. All trainings are performed utilizing a single GeForce GTX 1080 Ti GPU, and the RNN model is implemented using the highly efficient cuDNN LSTM API. It is clear that the proposed method has the highest training speed, followed by CNN and RNN. On average, our method was able to process around 867 sentences per second, which is 6.29% and 16.32% higher than CNN and RNN, respectively.

7.3 Comparison with Published Results

For comparison with published results, we summarize the performance of our best models along with state-of-the-art approaches in Table 3. The proposed method was able to surpass the previous state-of-the-art result on the ATIS dataset with a large margin, even without the CRF layer. With the help of CRF, our method obtains a new state-of-the-art result with a 96.62 F1-score.
For the POS tagging task with PTB WSJ dataset, we obtain a new state-of-the-art result with a 97.73% accuracy with the model without a CRF layer. Interestingly, utilizing a CRF layer on this model degraded the performance on this task whereas it helped with the other two tasks. We hypothesize that this is due to the fact that unlike the other two tasks where there are many hard constraints between labels (e.g. an O tag cannot be followed by I- tags), the label dependencies are more "soft" on POS tagging task. In the latter case, it is possible that naively taking label transition probability into account could have a negative impact on performance.
On the task of recognizing named entities, we obtain a result that is comparable to state-of-the-art with a 91.13 F1-score when a CRF layer is used. Like in slot tagging task, utilizing CRF lead to a significant increase in performance. It is notable that all results from our method are achieved without depending on any hand-crafted or language/task-specific features (e.g. capitalization, character type, gazetteer), whereas most previous approaches utilizes one or more type of such features. This fact supports the generalizability of the proposed method.

8 Conclusion and Future Work

In this paper, we proposed a fast and effective method of using a densely connected network to automatically generate character-level features. With extensive evaluation, it is shown that this method is robust to parameter initialization and has high processing speed compared to conventional methods such as CNN or RNN. This method has also high generalizability and this is supported by the fact that we were able to obtain superior performance without any task or language specific features.
We plan to explore the followings as future work: 1) In this work, we focused on clean text where there are minimal semantic or syntactic errors. We would like to test the robustness of this method against such errors to evaluate whether this method is suitable for real-world applications. 2) Adopting the proposed method and analyzing the effectiveness on other NLP tasks such as neural machine translation or automatic text summarization could also be worth investigating.

Acknowledgements

This research was supported by the MSIT (Ministry of Science and ICT), South Korea, under the ITRC (Information Technology Research Center) support program ("Research and Development of Human-Inspired Multiple Intelligence") supervised by the IITP (Institute for Information & Communications Technology Promotion). Additionally, this work was supported by the National Research Foundation of Korea (NRF) grant funded by the South Korean government (MSIP) (No. NRF-2016R1A2B2015912).

References

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

Jinho D Choi. 2016. Dynamic feature induction: The last gist to the state-of-the-art. In *Proceedings of NAACL-HLT*, pages 271–281.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *ICML*, pages 1818–1826.

Jenny Finkel, Shipra Dingare, Christopher D Manning, Malvina Nissim, Beatrice Alex, and Claire Grover. 2005. Exploring the boundaries: gene and protein identification in biomedical text. *BMC bioinformatics*, 6(1):S5.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Jan Hajič, Jan Raab, Miroslav Spousta, et al. 2009. Semi-supervised training for the averaged perceptron pos tagger. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 763–771. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. Leveraging sentence-level information with encoder lstm for semantic slot filling. *arXiv preprint arXiv:1601.01530*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Dekang Lin and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1030–1038. Association for Computational Linguistics.

Bing Liu and Ian Lane. 2015. Recurrent neural network structured output prediction for spoken language understanding. In *Proc. NIPS Workshop on Machine Learning for Spoken Language Understanding and Interactions*.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 879–888.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Christopher D Manning. 2011. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 171–189. Springer.

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

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Robert Moore. 2015. An improved tag dictionary for faster part-of-speech tagging. In *Proc. of EMNLP*. Citeseer.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *arXiv preprint arXiv:1404.5367*.

Baolin Peng and Kaisheng Yao. 2015. Recurrent neural networks with external memory for language understanding. *arXiv preprint arXiv:1506.00195*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. *arXiv preprint arXiv:1707.09861*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *ACL*, volume 7, pages 760–767. Citeseer.

Samuel L Smith, Pieter-Jan Kindermans, and Quoc V Le. 2017. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*.

Anders Søgaard. 2011. Semisupervised condensed nearest neighbor for part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 48–52. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Xu Sun. 2014. Structure regularization for structured prediction. In *Advances in Neural Information Processing Systems*, pages 2402–2410.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Yuta Tsuboi. 2014. Neural networks leverage corpus-wide information for part-of-speech tagging. In *EMNLP*, pages 938–950.

Ngoc Thang Vu, Pankaj Gupta, Heike Adel, and Hinrich Schütze. 2016. Bi-directional recurrent neural network with ranking loss for spoken language understanding. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 6060–6064. IEEE.

Ngoc Thang Vu. 2016. Sequential convolutional neural networks for slot filling in spoken language understanding. *arXiv preprint arXiv:1606.07783*.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. Spoken language understanding using long short-term memory neural networks. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 189–194. IEEE.

Feifei Zhai, Saloni Potdar, Bing Xiang, and Bowen Zhou. 2017. Neural models for sequence chunking.

Su Zhu and Kai Yu. 2017. Encoder-decoder with focus-mechanism for sequence labelling based spoken language understanding. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 5675–5679. IEEE.

Appendix A Hyperparameters

Group	Hyperparameter	Slot	POS	NER
Char-CNN	Window size	3	3	3
	Number of filters	30	30	30
	Character dimension	30	30	30
Char-RNN	Layer size	50	50	50
	Character dimension	50	50	50
Char-Dense	Layer size	50	50	50
	Number of pieces per word	2	2	2
Word-level	Pre-trained word embeddings	GloVe 300d	GloVe 300d	GloVe 300d
	RNN layer size	350	350	350
	RNN layer depth	2	3	3
	Pre-RNN layer size	350	350	None
	Post-RNN layer size	350	350	None
Dropout keep probability	Char-Dense	0.7	0.7	0.7
	Word feature	0.9	0.9	0.9
	Word-level RNN layer	0.5	0.5	0.5
	Pre/post-RNN layers	0.5	0.5	0.5
Training	Initial batch size	8	16	16
	Number of epochs	100	100	100

Table 4: Chosen hyperparameters for all experiments.

Neural Machine Translation Incorporating Named Entity

[1]Arata Ugawa [2]Akihiro Tamura [2]Takashi Ninomiya
[1,3]Hiroya Takamura [1]Manabu Okumura

[1]Department of Information and Communications Engineering, Tokyo Institute of Technology
[2]Graduate School of Science and Engineering, Ehime University
[3]National Institute of Advanced Industrial Science and Technology

ugawa.a.aa@m.titech.ac.jp
{tamura, ninomiya}@cs.ehime-u.ac.jp
{takamura, oku}@pi.titech.ac.jp

Abstract

This study proposes a new neural machine translation (NMT) model based on the encoder-decoder model that incorporates named entity (NE) tags of source-language sentences. Conventional NMT models have two problems enumerated as follows: (i) they tend to have difficulty in translating words with multiple meanings because of the high ambiguity, and (ii) these models' ability to translate compound words seems challenging because the encoder receives a word, a part of the compound word, at each time step. To alleviate these problems, the encoder of the proposed model encodes the input word on the basis of its NE tag at each time step, which could reduce the ambiguity of the input word. Furthermore, the encoder introduces a chunk-level LSTM layer over a word-level LSTM layer and hierarchically encodes a source-language sentence to capture a compound NE as a chunk on the basis of the NE tags. We evaluate the proposed model on an English-to-Japanese translation task with the ASPEC, and English-to-Bulgarian and English-to-Romanian translation tasks with the Europarl corpus. The evaluation results show that the proposed model achieves up to 3.11 point improvement in BLEU.

1 Introduction

Neural machine translation (NMT) models based on the encoder-decoder model, also known as the sequence-to-sequence model (Sutskever et al., 2014), have successfully shown their quality translation. Consequently, various NMT models are studied in the field of machine translation. To date, the most successful model is the bi-directional multi-layered encoder-decoder model with long short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and an attention mechanism (Luong et al., 2015; Bahdanau et al., 2015), also known as attention-based NMT. LSTM and the attention mechanism are introduced to mitigate the difficulty in handling long sentences in the encoder-decoder model. The conventional attention-based NMT model is known to achieve high translation accuracy in bilingual evaluation understudy (BLEU). However, this model encounters two general problems: (i) it tends to have difficulty in translating words with multiple meanings because their translations have high ambiguity, and (ii) translation of compound words seems difficult because the encoder receives only a word, a part of the compound word, at each time step.

This study proposes a new NMT model based on the encoder-decoder model, incorporating named entity (NE) information in source-language sentences. The proposed model alleviates the problems of the conventional attention-based NMT by incorporating information of NE tags to the encoder and modeling chunk information of NE tags as the encoder's network structures. The encoder of the proposed model

Proceedings of the 27th International Conference on Computational Linguistics, pages 3240–3250
Santa Fe, New Mexico, USA, August 20-26, 2018.

encodes the input word based on its NE tag at each time step to reduce the ambiguity of the input word. Furthermore, the encoder introduces a chunk-level LSTM layer over a word-level LSTM layer and hierarchically encodes a source-language sentence to capture a compound NE as a chunk based on its NE tags. We evaluate the proposed model on an English-to-Japanese translation task with the Asian Scientific Paper Excerpt Corpus (ASPEC) (Nakazawa et al., 2016) and English-to-Bulgarian and English-to-Romanian translation tasks with the Europarl corpus. The evaluation results show that the proposed model achieves up to 3.11 point improvement in BLEU.

The main contributions of this study are as follows:

1. Semantic class information of NE tags is incorporated to the encoder of the attention-based NMT model.

2. Chunk information of NE tags is modeled as a part of network structures in the attention-based NMT model.

2 Related Work

This section describes the attention-based NMT model, a previous NMT model incorporating linguistic features of source-language sentences, and previous NMT models based on chunks/phrases.

2.1 Attention-based NMT Model

The attention-based NMT model is an NMT model, wherein an attention mechanism is introduced in the encoder-decoder model. The encoder-decoder model consists of two recurrent neural networks (RNNs), namely, an encoder and a decoder. Gated Recurrent Unit (Cho et al., 2014) or LSTM is typically used as units of RNNs. In this work, LSTM is employed as units of RNNs.

Given a word sequence of the source language $x = (x_1, x_2, \ldots, x_j, \ldots, x_{T_x})$, the encoder generates a hidden vector $(h_1, h_2, \ldots, h_j, \ldots, h_{T_x})$. Each hidden state of the encoder $h_j \in \mathbb{R}^{d\times 1}$ is calculated using LSTM, given the previous state $h_{j-1} \in \mathbb{R}^{d\times 1}$ and the input word x_j, as follows:

$$h_j = f_{enc}(h_{j-1}, E_x(x_j)), \tag{1}$$

where f_{enc} is the encoder's LSTM, E_x is an embedding layer, and $E_x(x_j)$ is an embedding vector of the word x_j.

Then, the decoder sequentially generates a word sequence of the target language $y = (y_1, y_2, \ldots, y_i, \ldots, y_{T_y})$, given the final state of the encoder, h_{T_x}. The decoder outputs a word sequence having the maximum logarithmic likelihood with respect to the input word sequence x as the output word sequence y:

$$\log p(y|x) = \sum_{i=1}^{T_y} \log p(y_i|y_{1:i-1}, h_{T_x}). \tag{2}$$

The probability distribution of the output word y_i is calculated from the hidden state $\bar{h_i}$ of the decoder:

$$p(y_i|y_{1:i-1}, h_{T_x}) = softmax(proj(\bar{h_i})), \tag{3}$$

where $proj$ is a projection function to resize a decoder's hidden state to a vector, wherein the vector's dimension is the target vocabulary size. $softmax$ is a softmax function to convert the projected vector o into a probability distribution by normalizing each element of o as follows:

$$o = proj(\bar{h_i}), \tag{4}$$

$$softmax(o) = \frac{\exp(o)}{\sum_{k=1}^{K} \exp(o_k)}, \tag{5}$$

where K is the vocabulary size of the target language.

To prevent deterioration of the translation performance for long source-language sentences, the attention-based NMT tries to capture the relation between the states in the encoder and the decoder, and the decoder generates the translations by referring to the history of the hidden states in the encoder. In particular, the alignment score a_i is calculated on the basis of the similarity between the decoder's hidden state $\bar{h_i}$ and each of the encoder's hidden states $c = (h_1, h_2, \ldots, h_j, \ldots, h_{T_x})$. The context vector s_i is calculated as the weighted average, where a weight is a_i, over all the encoder's hidden states:

$$a_i(j) = \frac{\exp(\bar{h_i} \cdot h_j)}{\sum_{j'=1}^{T_x} \exp(\bar{h_i} \cdot h_{j'})}, \tag{6}$$

$$s_i = \sum_{j=1}^{T_x} a_i(j) h_j, \tag{7}$$

where T_x is the length of the source-language sentence x, and $\cdot$ denotes the inner product. In the decoder, the probability distribution of the output word is calculated on the basis of the context vector s_i in addition to the decoder's hidden state $\bar{h_i}$:

$$p(y_i|y_{1:i-1}, c) = softmax(proj([\bar{h_i}; s_i])), \tag{8}$$

where [;] denotes the concatenation of the two vectors.

2.2 Linguistic Input Features for NMT

Sennrich and Haddow (2016) have improved the attention-based NMT model by using the linguistic features of source-language sentences. In particular, the encoder receives the lemma, subword tags, POS tags, and dependency labels of source-language sentences in addition to source-language words. A lemma is the original form of the word. Subword tags express prefix, stem, and suffix. A POS tag is part-of-speech information of a word, such as a noun or a verb. Moreover, a dependency label indicates a syntactic relation between words, such as a head and dependents. The encoder converts each of the linguistic features into its embedding vector, and then generates hidden states $(h_1, h_2, \ldots, h_j, \ldots, h_{T_x})$ from the word embedding vectors of source-language words $x = (x_1, x_2, \ldots, x_j, \ldots, x_{T_x})$ and the linguistic feature's embedding vectors as follows:

$$h_j = f_{enc}(h_{j-1}, (E_{word}(x_j)[\,;\, {}^{|F|}_{k=1} E_k(x_{jk})])), \tag{9}$$

where E_{word} is an embedding layer for source-language words, E_k is an embedding layer for the k-th type of linguistic features, and $|F|$ is the number of types of linguistic features (i.e., $|F|$ = 4). Note that the model has not used NE tags as a linguistic feature, and the incorporation of NE tags into NMT is still under exploration.

2.3 NMT Based on Chunk/Phrase Units

Ishiwatari et al. (2017) have improved the attention-based NMT by designing chunk-based decoders, each of which models global dependencies by a chunk-level decoder and local word dependencies by a word-level decoder. In their decoders, the chunk-level decoder first generates a chunk representation. Then, the word-level decoder predicts each target word from the chunk representation.

Wang et al. (2017) have improved attention-based NMT by integrating a phrase memory, which stores target phrases provided by a statistical machine translation (SMT) (Pal et al., 2010) model, to perform a phrase-by-phrase translation rather than a word-by-word translation. Their model dynamically selects a word or phrase to be output at each decoding step.

Meanwhile Ishiwatari et al. (2017) and Wang et al. (2017) have incorporated chunks identified by a chunker and a SMT model, respectively, our work focuses on chunk information of NE tags. Note that our proposed model can incorporate general chunk information, such as chunks found by a chunker. We will leave this aspect for future work.

3 NMT Incorporating NE Tags

In this section, we propose a new NMT model, incorporating NE tags of source-language sentences. By considering NE types of source-language words, we aim to improve translation performance for words with multiple meanings. Furthermore, by using chunk information of NE tags (e.g., IO or BIO information), we aim to improve translation performance for compound words.

The proposed model assumes that an NE tag is attached to each word in source-language sentences by using an NE tagger. The proposed model receives the sequence of NE tags along with the sequence of source-language words and generates each encoder's hidden state on the basis of not only source-language words but also NE tags. Furthermore, the proposed model introduces a chunk-level LSTM layer over a word-level LSTM layer into the encoder and hierarchically encodes a source-language sentence.

In Section 3.1, we overview NE tags, and in Section 3.2, we describe the model architecture of the proposed model.

3.1 Named Entity (NE)

NEs are words/phrases for specific entities, such as the name of persons, organizations, and locations. Moreover, NEs are sometimes extended to include time expressions and numerical representations. In CoNLL2003 shared task[1], the four types of NEs, namely 'person', 'organization', 'location', and 'names of miscellaneous', have been used. In Message Understanding Conference (MUC)[2], the seven kinds of NEs, namely 'person', 'location', 'organization', 'time', 'date', 'money', and 'percent', have been used. Information Retrieval and Extraction Exercise (IREX)[3] have used the eight kinds of NEs, where 'artifact' is added to the NEs used in MUC. Moreover, on Ontonotes 5.0[4], the 18 kinds of NEs have been defined. NEs could be considered as a kind of semantic category of words/phrases. Therefore, NEs have contributed to many NLP tasks, including SMT. Note that NEs have not been used in NMT.

NE Recognition is a classification task to identify NE words/phrases and their NE categories in given input sentences. In the tasks, NEs are usually expressed by chunk tags, such as BIO and IO tags, for words because an NE might be a phrase. For example, the sentence 'I arrived at Tokyo Station at 10:20.' is tagged by BIO2 tags as follows: 'I:O arrived:O at:O Tokyo:B-location Station:I-location at:O 10:20:B-time .:O', where B, I, and O indicate the beginning, inside, and outside of NEs, respectively. In this work, we used IO tags. By IO tags, NE words/phrases are represented as I (inside) and the others are expressed as O (outside). An example of the tagged sentence is as follows: 'I:O arrived:O at:O Tokyo:I-location Station:I-location at:O 10:20:I-time .:O'.

As mentioned above, NE tags include two features: (i) semantic class of words (e.g., location and time) and (ii) chunk information (e.g., I and O). By using the first feature, we aim to decrease the ambiguity of source-language words and improve the performance for words with high ambiguity. By using the second feature, we aim to capture the chunks of compound words and improve the performance for compound words.

3.2 Model Architecture

Figure 1 shows the encoder of the proposed model. The encoder of the proposed model is composed of three components, namely, the embedding layer, word-level LSTM layer, and chunk-level LSTM layer.

The encoder receives the sequence of NE tags, $tag = (tag_1, tag_2, \ldots, tag_i, \ldots, tag_{T_x})$, along with the sequence of source-language words, $x = (x_1, x_2, \ldots, x_i, \ldots, x_{T_x})$, where tag_i denotes the NE tag of x_i. At each time step i, the encoder generates the word embedding vector for the source-language word x_i through a word embedding layer E_x. Additionally, the embedding vector for the NE tag tag_i is generated through a tag embedding layer E_{tag}. Then, these vectors are added:

$$\hat{x}_i \quad = \quad E_x(x_i) + a * E_{tag}(tag_i), \tag{10}$$

[1] https://www.clips.uantwerpen.be/conll2003/

[2] http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/muc_7_toc.html

[3] http://nlp.cs.nyu.edu/irex/

[4] https://catalog.ldc.upenn.edu/ldc2013t19

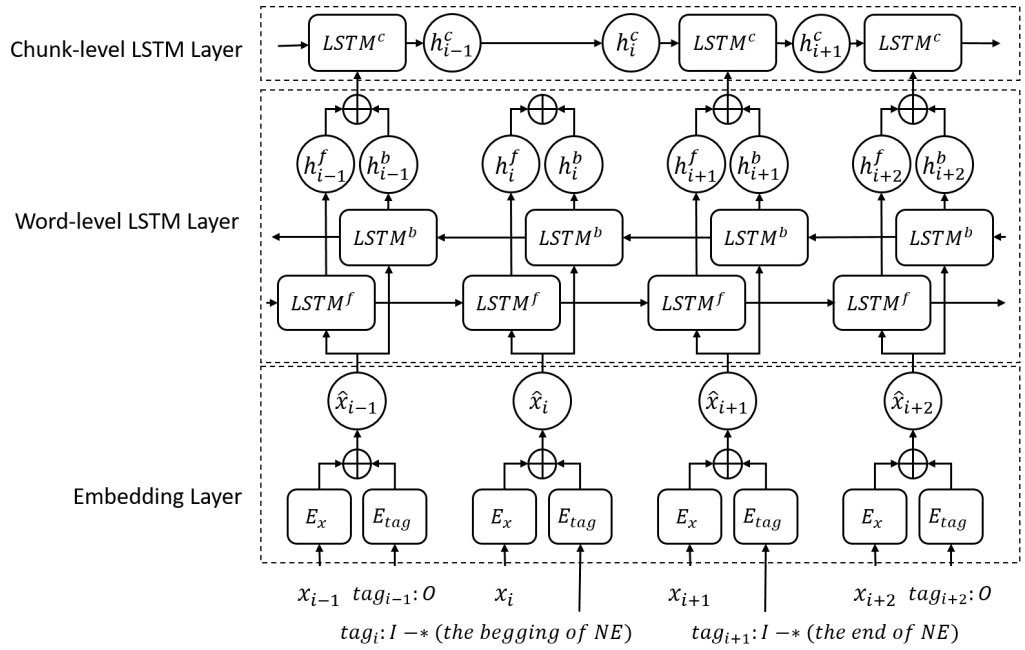


Figure 1: Encoder of the Proposed Model

where a is a parameter to control the influence of NE tags and is optimized in model training. The added vector, $\hat{x}_i$, is fed to a word-level LSTM component.

The word-level LSTM layer is a bi-directional LSTM, which encodes an input sentence by word. The bi-directional LSTM consists of a forward LSTM, $LSTM^f$, and a backward LSTM, $LSTM^b$. The forward LSTM obtains hidden states $(h^f_1, \ldots, h^f_{T_x})$ from the beginning to the end of the sentence:

$$h^f_i = LSTM^f(h^f_{i-1}, \hat{x}_i). \tag{11}$$

The backward LSTM obtains hidden states $(h^b_{T_x}, \ldots, h^b_1)$ from the end to the beginning of the sentence:

$$h^b_i = LSTM^b(h^b_{i+1}, \hat{x}_i). \tag{12}$$

Then, the bi-directional LSTM computes the average of h^f_i and h^b_i as the i-th word-level encoder's hidden state h_i:

$$h_i = (h^f_i + h^b_i)/2. \tag{13}$$

The hidden state h_i is fed to the chunk-level LSTM layer.

The chunk-level LSTM layer is a uni-directional forward LSTM, $LSTM^c$, which receives the hidden states of the word-level layer and encodes a source sentence by chunk identified by NE tags. Based on the chunk information of NE tags, the hidden states of the chunk-level LSTM layer, $(h^c_1, \ldots, h^c_{T_x})$, is calculated as follows:

$$h^c_i = \begin{cases} LSTM^c(h^c_{i-1}, h_i) & \text{if } tag_i \text{ is 'O' or the last part of the NE tags} \\ h^c_{i-1} & \text{otherwise} \end{cases} \tag{14}$$

The decoder of the proposed model is the same as in the conventional attention-based NMT model. The initial state of the decoder is set to the sum of the final state of the word-level LSTM layer and that of the chunk-level LSTM layer.

4 Experiment

This section compares the proposed model, which incorporates NE tags of source sentences, with a conventional NMT model, which does not use NE tags, under English-to-Japanese (En-Jp), English-to-Bulgarian (En-Bg), and English-to-Romanian (En-Ro) translation tasks to confirm the effectiveness of the proposed model.

Table 1: Statistics on Experimental Data (# of parallel sentences)

	Training Data	Development Data	Test Data
En-Jp	1,320,591	1,768	1,802
En-Bg	363,112	3,000	3,000
En-Ro	357,247	1,972	3,000

Table 2: Vocabulary Size

	Source Language	Target Language
En-Jp	78,591	57,771
En-Bg	27,872	30,000
En-Ro	27,651	30,000

4.1 Experimental Data

We evaluated the En-Jp translation performance on the ASPEC, which is used in WAT 2017[5], and the En-Bg and En-Ro translation performance on the Europarl corpus (Koehn, 2005).

The English and Japanese sentences are tokenized by spaCy[6] and KyTea (Neubig et al., 2011), respectively. The Bulgarian and Romanian sentences are tokenized by byte-pair encoding (Sennrich et al., 2016) implemented in sentencepiece[7], where we set the vocabulary size to 30,000. The words that appeared less than five times in the En-Jp training data and those less than twice in the English side of the En-Bg and En-Ro training data were replaced with the special symbol $\langle \text{UNK} \rangle$. In the training, all words were lowercased by lowercase.perl[8], and long sentences with over 50 words were filtered out.

In the En-Jp task, we used the development and test data employed in WAT 2017. In the En-Ro task, we used the newsdev-2016 as the development data and randomly sampled 3,000 parallel sentences from the training data as the test data. In the En-Bg task, we randomly sampled 3,000 parallel sentences from the training data for the development data and test data. The statistics on the experimental data are summarized in Table 1 and Table 2.

We used spaCy NE tagger[9], which was trained from the OntoNotes5.0[10] corpus attached with NE tags, to identify NE tags of English sentences in the proposed model. Table 3 shows the types of NE tags, and Table 4 gives the Top 3 NE types on the training corpus.

4.2 Competing Models

We compared the proposed model (*Proposed*) with the standard attention-based encoder-decoder NMT model, wherein the encoder is two stack bi-directional LSTM and the decoder is two stack forward LSTM (*Baseline*). Note that the proposed model introduces the embedding layer for NE tags and chunk-level forward LSTM layer into *Baseline*. We also compared the proposed model with the simulated model of Sennrich and Haddow (2016) (*Baseline_Concat*). In *Baseline_Concat*, the encoder generates a hidden state from the concatenated vector of the embedding vector for an input word and that for its NE tag at each time step (refer to Equation (9)). The network structures other than the embedding layer are the same as in *Baseline*.

All of the embedding size and hidden size of the encoder and decoder both in the baselines and proposed model are set to 256. The parameter a, which controls the effect of NE tags, is initialized to 0.5.

We used Adam (Kingma and Ba, 2015) with a mini-batch size of 64, 128, and 64 for learning each parameter in the En-Jp, En-BG, and En-Ro tasks, respectively. We also employed a gradient clipping

[5] http://orchid.kuee.kyoto-u.ac.jp/WAT/WAT2017/index.html
[6] https://spacy.io/
[7] https://github.com/google/sentencepiece
[8] http://www.statmt.org/moses/
[9] https://spacy.io/usage/linguistic-features#101
[10] https://catalog.ldc.upenn.edu/ldc2013t19

Table 3: Types of NE Tags

NE Type	Descriptions
PERSON	people, including fictional
NORP	nationalities or religious or political groups
FACILITY	buildings, airports, highways, bridges, etc.
ORG	companies, agencies, institutions, etc.
GPE	countries, cities, states
LOC	non-GPE locations, mountain ranges, bodies of water
PRODUCT	objects, vehicles, foods, etc. (not services.)
EVENT	named hurricanes, battles, wars, sports events, etc.
WORK_OF_ART	titles of books, songs, etc.
LAW	named documents made into laws
LANGUAGE	any named language
DATE	absolute or relative dates or periods
TIME	times smaller than a day
PERCENT	percentage, including '%'
MONEY	monetary values, including unit
QUANTITY	measurements, as of weight or distance
ORDINAL	'first', 'second', etc.
CARDINAL	numerals that do not fall under another type
OTHER	others

Table 4: Top 3 NE Types

Corpus	Top 3 NE Types
En-Jp	O: 94.1%, ORG: 1.65%, CARDINAL: 1.12%
En-Bg	O: 91.3%, ORG: 3.56%, DATE: 1.16%
En-Ro	O: 91.2%, ORG: 3.58%, DATE: 1.16%

Table 5: Experimental Results (BLEU score (%))

	Baseline	*Baseline_Concat*	*Proposed*
En-Jp	29.38	28.17	29.62
En-Bg	37.99	37.48	38.38
En-Ro	34.31	36.77	37.44

technique with the clipping value of 5.0, dropout with the dropout ratio of 0.2, and the weight decay with the coefficient of 1.0×10^{-6}. The training stopped after 30 epochs, and the model that achieved the best BLEU score on the development data, was used for testing.

4.3 Results

We evaluated the translation performances via the case-insensitive BLEU4 metric (Papineni et al., 2002), which is computed by multi-bleu.perl[11]. Table 5 shows the experimental results.

From Table 5, the proposed model is better than *Baseline* for all the translation tasks, which shows that the effectiveness of the incorporation of NE tags of source sentences into NMT. In addition, the proposed model outperforms *Baseline_Concat* for all the translation tasks. This observation indicates that the proposed model makes use of NE tags more effectively than the previous model (Sennrich and Haddow, 2016).

5 Discussion

This work aims to improve the translation performance for compound words and words with multiple meanings by using NE tags of source sentences. In this section, we discussed the effectiveness of the proposed model through actual examples. Table 6 shows the output examples of the baseline model, *Baseline*, and the proposed model for three English sentences.

The word 'first' has multiple meanings in Japanese (e.g., 最初の (beginning), 一つの (one of), 一位の (top)). (a) of Table 6 shows that the proposed model correctly translates the word 'first', whereas the baseline model does not. We guess the reason is that the proposed model could disambiguate the meanings of the word 'first' on the basis of the NE tag 'ORDINAL'.

From (b) in Table 6, the proposed model correctly translates the English phrase 'between 0.2 to 0.5' into the Japanese phrase '０．２～０．５', whereas the baseline model fails and results in 'over- translation' (i.e., the Japanese words '～０．０５' are generated twice). Probably, the reason is that the

Table 6: Output Examples of Baseline and Proposed Models

(a) Example1

Input	about	60	%	of	the	first	peak	showed	ca2	+	dependence	.
NE tags	I-PERCENT	I-PERCENT	I-PERCENT	O	O	I-ORDINAL	O	O	O	O	O	O

Reference: 最初のピークの約６割がｃａ２＋依存性を示した．
Baseline: １つのピークの約６０％はｃａ＋依存性を示した．
Proposed: 最初のピークの約６０％はｃａ２＋依存性を示した．

(b) Example 2

Input	quantum	yields	of	their	decomposition	are	between	0.2	to	0.5	,
	depending	on	their	substituents	.						
NE tags	I-ORG	O	O	O	O	O	I-CARDINAL	I-CARDINAL	I-CARDINAL	I-CARDINAL	O
	O	O	O	O	O						

Reference: それらの分解の量子収量は，それらの置換基に依存して，０．２～０．５である。
Baseline: 分解の量子収率は０．２～０．０５～０．０５になり，それらの⟨UNK⟩に依存して０．０５５．０５μｍである。
Proposed : この分解の量子収量は，それらの⟨UNK⟩に依存して０．２～０．５である。

(c) Example 3

Input	a	new	electromagnetic	coupling	structure	has	been	proposed	for	a	millimeter	wave	dr	-	vco	.
NE tags	O	O	O	O	O	O	O	O	O	O	I-CARDINAL	O	I-GPE	I-GPE	I-GPE	O

Reference: ミリ波ｄｒ-ｖｃｏ用の新しい電磁結合構造を提案した。
Baseline: ミリ波ｄｒ板のための新しい電磁結合構造を提案した。
Proposed: ミリ波ｄｒ-ｖｃｏの新しい電磁結合構造を提案した。

[11] http://www.statmt.org/moses/

Table 7: Experimental Results to Confirm the Effectiveness of Chunk Information

	Baseline	*Baseline_Chunk*	*Proposed_IO*	*Proposed*
En-Jp	29.38	28.61	29.51	29.62
En-Bg	37.99	37.55	37.93	38.38
En-Ro	34.31	35.48	35.78	37.44

proposed model could consider the four words as one phrase on the basis of the chunk information (I/O) of the NE tags.

Furthermore, as can be seen from (c) in Table 6, the compound word 'dr - vco' is correctly translated to 'd r - v c o ' by the proposed model, whereas it is wrongly translated by the baseline model. This finding also indicates that the proposed model is better for the translation of compound words. An interesting observation from (c) is that although the NE type of 'dr - vco' is wrong (i.e., the correct type is 'OTHER' although the attached type is 'GPE'), the proposed model translates the compound word correctly. This finding indicates that the chunk information of NE tags could be useful to NMT even if the NE type is not correctly identified.

The above examples indicate that both semantic class information of words (i.e., NE types) and chunk information (i.e., I/O) of NE tags could be helpful to NMT. To quantitatively confirm the pure effectiveness of chunk information of NE tags, we evaluated the performance of the proposed model that uses only chunk information of NE tags (*Proposed_IO*). In particular, *Proposed_IO* receives either of 'I' tag or 'O' tag as the NE tag. In addition, we evaluate the *Baseline* model that naively uses chunk information of NE tags (*Baseline_Chunk*). In *Baseline_Chunk*, each source sentence is preliminarily chunked on the basis of NE tags and each chunk is treated as a word. For example, the two words 'donald:I-PERSON trump:I-PERSON' are concatenated into one chunk 'donald_trump' and the chunk is treated as one word.

Table 7 summarizes their performance. As can be seen from Table 7, *Baseline_Chunk* is worse than *Baseline* in the En-Jp and En-Bg tasks although *Baseline_Chunk* outperforms *Baseline* in the En-Ro task. This finding indicates that the naive incorporation of chunk information of NE tags could have negative effect to NMT. Table 7 also shows that *Proposed_IO* is at least comparable to, or better than, *Baseline*. This observation indicates that only chunk information of NE tags could improve NMT performance in the proposed model. We conjecture that chunking increases vocabulary sizes. Thus, the naive incorporation (i.e., *Baseline_Chunk*) might suffer from the data sparseness problem, whereas *Proposed_IO* might alleviate the sparseness problem by the hierarchical structure of the encoder (i.e., word-level LSTM layer). In addition, Table 7 shows that *Proposed* outperforms *Proposed_IO* for all the tasks. This finding indicates that NE types enable further improvement in NMT performance.

6 Conclusion

We have proposed a new encoder-decoder NMT model, which alleviates the problems of word ambiguities and compound words in the conventional attention-based NMT by incorporating NE tags of source-language sentences. The encoder of the proposed model encodes both input word and NE tag at each time step and has a chunk-level LSTM layer over a word-level LSTM layer to hierarchically encode a source-language sentence.

The experiments on the English-to-Japanese translation task with the ASPEC show that the proposed model achieved 0.24 point improvement in BLEU. In addition, the experiments on the English-to-Bulgarian and English-to-Romanian translation tasks with the Europarl corpus show that the proposed model achieved 0.39 and 3.11 point improvements in BLEU, respectively.

We qualitatively analyzed translation results to investigate the effectiveness of the NE information in the encoder-decoder model and found several examples that show the effectiveness of either semantic class information or chunk information of NE tags.

Finally, we also conducted experiments to evaluate how chunk information (i.e., I/O) of NE tags contributed to improve the translation accuracy without semantic class information. From the experiments, we observed that chunk information could improve the translation accuracy, and the translation accuracy

was further improved by adding semantic information of NE tags.

Future work includes finding good models to incorporate NE tags and verify the effectiveness of the proposed models for other datasets. In addition, we would like to incorporate an NE tagging model to the proposed model to learn both the NE model and machine translation model.

Acknowledgments

This work was partially supported by JSPS Grants-in-Aid for Scientific Research Grant Number 25280084. We appreciate Hidetaka Kamigaito for his helpful suggestions on this work. We also thank the anonymous reviewers for their careful reading of our study and their insightful comments.

References

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations, arXiv:1409.0473*.

Kyunghyun Cho, Bart van Merrienboer, aglar Gulehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. In *Proceedings of The Neural computation*, volume 9, pages 1735–1780. MIT Press.

Shonosuke Ishiwatari, Jingtao Yao, Shujie Liu, Mu Li, Ming Zhou, Naoki Yoshinaga, Masaru Kitsuregawa, and Weijia Jia. 2017. Chunk-based decoder for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1901–1912.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations, arXiv:1412.6980*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the MT summit*, pages 79–86.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. Aspec: Asian scientific paper excerpt corpus. In *Proceedings of the 10th Conference on International Language Resources and Evaluation*, pages 2204–2208.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 529–533.

Santanu Pal, Sudip Kumar Naskar, Pavel Pecina, Sivaji Bandyopadhyay, and Andy Way. 2010. Handling named entities and compound verbs in phrase-based statistical machine translation. In *Proceedings of the 2010 Workshop on Multiword Expressions: from Theory to Applications*, pages 46–54.

Kishore Papineni, Salam Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Trunslation*, volume 1, pages 83–91.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725.

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

Xing Wang, Zhaopeng Tu, Deyi Xiong, and Min Zhang. 2017. Translating phrases in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1421–1431.

Semantic Parsing of Technical Support Questions

Abhirut Gupta, Anupama Ray, Gargi Dasgupta,
Gautam Singh, **Pooja Aggarwal** and **Prateeti Mohapatra**
IBM Research AI
{abhirutgupta, anupamar, gaargidasgupta,
gautamsi, aggarwal.pooja, pramoh01}@in.ibm.com

Abstract

Technical support problems are very complex. In contrast to regular web queries (that contain few keywords) or factoid questions (which are a few sentences), these problems usually include attributes like a detailed description of what is failing (symptom), steps taken in an effort to remediate the failure (activity), and sometimes a specific request or ask (intent). Automating support is the task of automatically providing answers to these problems given a corpus of solution documents. Traditional approaches to this task rely on information retrieval and are keyword based; looking for keyword overlap between the question and solution documents and ignoring these attributes. We present an approach for semantic parsing of technical questions that uses grammatical structure to extract these attributes as a baseline, and a CRF based model that can improve performance considerably in the presence of annotated data for training. We also demonstrate that combined with reasoning, these attributes help outperform retrieval baselines.

1 Introduction

Technical support services involve enterprises providing after-sales support to users of technology products. Traditionally, this entails employing a large number of human agents to manually diagnose, troubleshoot, and resolve customer problems. However, with ever increasing variety of products and the exponential growth of users, this model doesn't scale. Costs of hiring and training additional talent to keep up with this growth is a huge overhead. There is a clear incentive to automate this process, reducing manual effort and resolution times. Automating technical support refers to the task of automatically understanding user questions, performing diagnosis and troubleshooting, either remotely or in conversation with the user, and providing resolutions. Given the massive potential impact, it is no surprise that the problem draws attention and investment from many large corporations as well as startups (Flinders, 2015), and (Dhoolia et al., 2017).

Automating technical support has a very broad goal, and involves solving various sub-problems like document representation (Yang et al., 2017), identifying and extracting procedures, log analysis, and goal based dialoging (Bordes and Weston, 2016). Each of these problems has a plethora of daunting research challenges that remain largely unsolved. In this paper, we restrict our focus to the problem of understanding questions in technical support and retrieving relevant resolution documents from a corpus of support content. While a complete automation system would also tackle the challenging problem of identifying the troubleshooting/diagnostic procedure from these documents, we assume that a returned "relevant" document contains procedures required for resolving the problem.

As identified in (Gupta et al., 2017) technical support problems are complex, and involve various attributes like failing entities, a detailed description of the problem (symptom), steps taken in an effort to remediate the failure (activity), and sometimes a specific request or ask (intent). Consider two actual support problems in Figure 1 from the software and the hardware domains. The hardware query comprises a problem description with the entity "tape drive", mentioning the symptom "stuck" and an unsuccessful

Proceedings of the 27th International Conference on Computational Linguistics, pages 3251–3259
Santa Fe, New Mexico, USA, August 20-26, 2018.

Table 1: Average length of questions (in words) from QA datasets.

Dataset	Average length
Wiki QA	7.32
TREC 2007 QA	8.05
DB2 Prop	150.97
DB2 Open	105.87

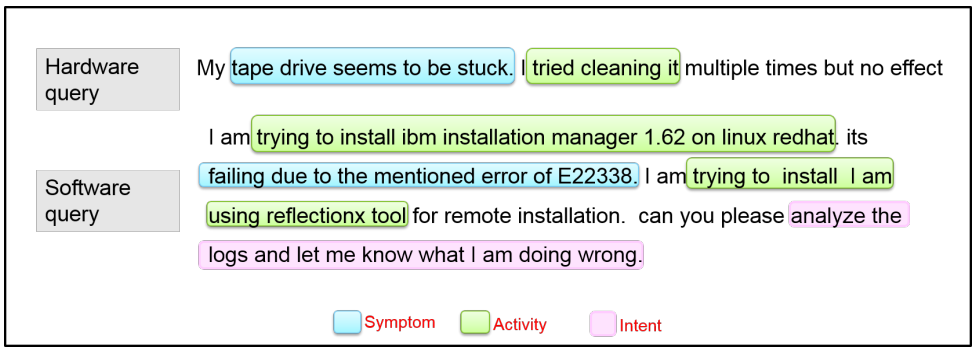


Figure 1: Illustration of Annotation of attributes on a Support ticket

attempt to mitigate the problem by "cleaning". The software question describes relevant details regarding the "failing installation". It also contains an explicit ask from the technician for "analysis of log files". State-of-the-art information retrieval based systems which depend on keywords do not perform well on such queries. For example, without an understanding of the hardware problem description and its attributes, the system might return pages which suggest cleaning the tape drive. For the software query, understanding the explicit ask of log analysis is key to a relevant response. This inherent complexity of questions in this domain differentiates it from traditional Question Answering or Information Retrieval problems. Table 1 compares the average length of questions in popular open QA datasets - WikiQA (Yang et al., 2015) and TREC 2007 QA (Dang et al., 2007), to two datasets in the domain of technical support that we curated. For effective retrieval of documents, it is essential that we extract these attributes (removing noise), and implement reasoning on top.

In this paper, we present a method for semantic parsing of support questions, to extract these attributes, which uses rules based on the grammatical structure of sentences. To the best of our knowledge, we are first to address the complexities of problems in technical support by identifying pertinent semantic attributes, extracting them, and proving their usefulness in retrieving relevant results.

In section 2 we present existing related work. In section 3, we present the details of our model for semantic parsing. The goal of extracting these attributes, however, is to improve quality of returned documents. To emphasize the importance of extracting these attributes, we show marked improvements in retrieval. The datasets used for experiments are discussed in 4 and evaluations are presented in section 5. We conclude the paper with a discussion of the learnings from evaluation and analysis, and directions for future research in section 6.

2 Related Work

The problem of answering natural language questions has received significant attention from the natural language processing research community. Dependency and simple phrase structure grammar models have been around for a while for understanding natural language text. The authors in (de Marneffe and Manning, 2008) parse grammatical relations from text for automated understanding. In Watson Jeopardy (McCord et al., 2012), authors use slot grammar parsing (McCord, 1990b) that decomposes the natural language text to slots, which is widely used for applications like question analysis (McCord et al., 2012), question decomposition, knowledge extraction for QA (Fan et al., 2012), relation extraction,

(Wang et al., 2012), candidate generation (Lally et al., 2012), and analysis and gathering of textual evidence (Murdock et al., 2012). Works focusing on understanding user intent use classifier models or dependency parsing schemes (Li and Roth, 2006; Metzler and Croft, 2005). Other recent works build neural models that represent a question as a continuous-valued vector (Chen et al., 2016). The key difference in these approaches and our problem setting is that they usually work on simpler, entity seeking questions. However questions in support are usually very complex, contain multiple attributes describing the situation that leads to a failure, attempts made to resolve the problem, and explicit asks. The answers to these questions are usually diagnostic or troubleshooting procedures contained in documents which need to be searched based on an understanding of the problem.

(Vtyurina and Clarke, 2016) tackle open domain complex questions by learning keywords from a large supervised dataset of questions and answers. Extraction of such keywords from questions certainly helps retrieval by removing noise. However, in technical support, the multiple types of attributes necessitate their separate extraction and reasoning. "Activity" words extracted from the question as keywords would hurt retrieval unless they are correctly identified, and their matching score is reduced. But, the same word if present as "Intent" would certainly help improve retrieval and their matching score would need to be boosted. Finally, (Yang et al., 2017) presents a system which improves the state of the art for technical support question answering. Their work is complementary to ours, as they address the problem of knowledge representation by constructing a knowledge graph from content and do not delve into the complexities of technical support problems.

To the best of our knowledge, ours is the first attempt that defines important finer attributes in technical support questions, proposes a model to extract them, and performs reasoning to retrieve relevant results.

3 Semantic Parsing And Reasoning

As explained earlier, we identify the following three attributes for technical support questions:

1. **Symptom** - A description of what's failing
2. **Activity** - Steps that have been tried to remediate the problem
3. **Intent** - A specific request or ask

Questions can contain multiple of each attributes. Because of this inherent complexity, it is imperative to parse these attributes for retrieving relevant results. Table 2 shows the frequency distribution of the different attributes in the training set.

Formally, the problem of semantic parsing is formulated as follows. Given a support question Q, which is comprised of a sequence of tokens - $Q = [q_1, q_2, .., q_n]$, we want to extract three sequences S, A and I which denote the symptoms, activities and intents (these sequences are not necessarily contiguous, allowing for multiple symptoms, activities, and intents in the same question).

We present a Semantic Parser to extract these attributes from questions using rules built on the grammatical structure of sentences. We also present a Conditional Random Field model, which, in the presence of large amount of supervision, outperforms the rule based Semantic Parser. However, such annotations are expensive to curate. Details of both the models are presented below.

3.1 Semantic Parser

There are several natural language parsers that allow tokenization and POS tagging. Some also provide a simple description of the grammatical relations in a sentence, but this is not sufficient for several applications. Slot Grammar system has a lexicalist character and treats different grammatical structures in a language independent manner through different rule types. IBM Watson uses two deep parsing components: English Slot Grammar (ESG) followed by Predicate Argument Structure (PAS) for linguistic analysis of text (McCord, 1990a).

ESG works on a sentence to produce a parse tree which has a deep structure (logical analysis) and a surface structure (grammatical analysis). PAS simplifies the ESG parse tree structure to the core semantic meaning. PAS forms a labeled directed graph which is more flexible and requires less knowledge than

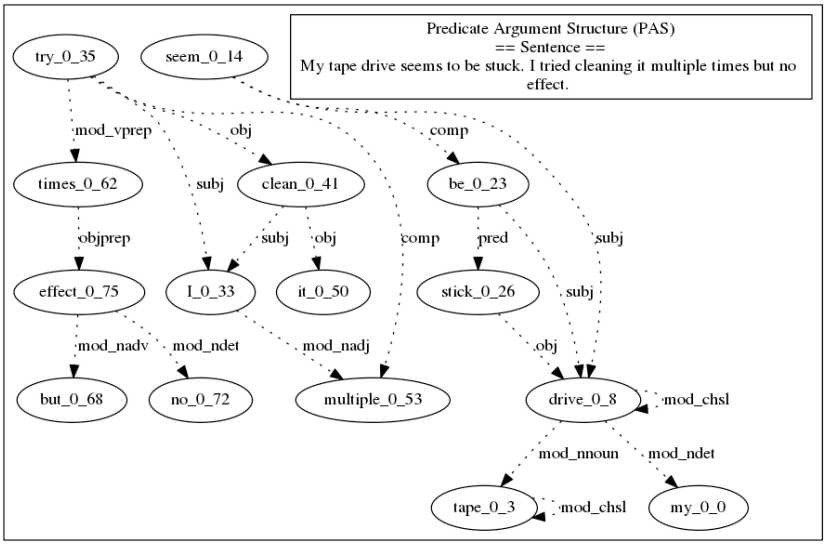


Figure 2: Sample PAS tree

```
pattern = there_is_np(@1) -> vp0[hasLemmaForm("be"),!hasParseFeature("q")]
{
        subj -> var0[hasLemmaForm("there")]
}
{
        pred -> vp1[hasLemmaForm("@1")]
}
```

Figure 3: Example of template that captures cases like “there is error”, “there was error”. “@1” would be replaced by a dictionary word such as *error message, symptom, blocker, expiration, or limitation*

ESG. PAS combines the two dimensions of ESG by omitting nodes with auxiliary verbs, closed class nodes introducing verb phrases, determiners (except for high semantic determiners), forms of *be* with no predicate and with adjective predicate. The negation annotator modifies the standard PAS by retaining some of these nodes, and we use the negation annotator to identify symptoms/problems like *does not work, cannot boot* from a question. Figure 2 shows an PAS tree structure for a sample sentence. While ESG retains syntactic information, PAS only represents semantics. Sentences with a similar meaning but different structures (like the active and passive form of the same sentence), would have varying ESG parses but the same PAS parse. Therefore, it is much more efficient to create patterns on the PAS tree than the ESG tree.

To obtain the attributes from user utterances, we create patterns to be matched to the input query. These patterns are essentially rules written on top of the slot grammar. Since manually specifying all possible rules can get cumbersome, templates are used to represent these patterns. Figure 3 presents a sample template for extracting symptoms like “there is/was an error”. These templates are combined with the dictionary words identified for each attribute (symptom, activity and intent) to form rules that are matched to the PAS tree for extraction. Text spanned by the ESG node, corresponding to the matched PAS node, is extracted and marked as the annotation output for the corresponding attribute. The base lexicon in ESG is augmented with support specific terminologies and Word-Net for wider coverage. Additionally, multi-word entries were added into chunk lexicons to obtain multi-word phrases as a single node in the PAS tree.

3.2 CRF based Parser

The task of extracting attributes from technical support problems can be formulated as a sequence labeling problem. This follows from the observation that these attributes manifest in a rough order -

symptoms, followed by attempts to mitigate, and finally explicit requests. Context words are also important in identifying spans of text for these attributes. CRFs are graphical models that can capture such dependencies among input observations (Lafferty et al., 2001). A CRF defines a conditional distribution $p(y|x)$ where *y* is the corresponding label sequence and *x* is the observed data sequence.The current label at time step t, y_t can be dependent on any of the previous *n* labels, and all the observations in an *n*-order CRF. In our case, the input sequence *x* corresponds to a series of words, while the label sequence *y* corresponds to the symptom, activity, intent or other assigned to the input sequence. The probabilistic model of a label sequence given some sequence of words is mediated in this model through a set of weighted functions f_i:

$$p(y|x) = \frac{exp(\sum_i \sum_t w_i f_i(y_{t-1}, y_t, x, t))}{Z(x)}$$

where the w_i are the weights assigned by the learning algorithm, and $Z(x)$ is a normalization factor over all label sequences.

For training, a manually annotated set of support questions is used. Spans are labeled with the B-I-O encoding, where each word in the query is annotated with one of the following classes (labels): the beginning of a symptom (B-S), inside of a symptom (I-S), the beginning of an activity (B-A), inside of an activity (I-A), the beginning of an intent (B-I), inside of an intent (I-I) or other (O).

To conduct our experiments, we used the linear-chain Stanford CRF implementation (Finkel et al., 2005) . We used current word, previous word, next word, current word character n-gram ($n \leq 6$), presence of word in left window (size = 4), presence of word in right window (size = 4), position of word in the sentence, and current POS tag as features computed using the Stanford NER toolkit.

3.3 Reasoning

We need to use the extracted attributes to perform reasoning on the retrieved results. For example, for the problem in Figure 1, we would not want to return back to the user, pages which detail a procedure for cleaning the tape drive. Similarly, when a user specifies a particular intent, we would ideally like to prioritize matches with this intent over matches with the symptom which might include other diagnostic or resolution procedures. For specifying these priorities, we create a module that weighs query fields in the retrieval engine based on the attribute type of the field. Query fields containing intents are given the highest weights, symptom containing query fields are given slightly lower weight, and fields with activity are given negative weights to suppress results matching this field. The actual values for weights are determined empirically.

4 Dataset and Experimental Setup

Training and evaluation of our parser models is done on a proprietary dataset of 1972 actual customer problems. These utterances have been manually annotated with the identified attributes by SMEs. 80% of this data is set aside for training and validating the CRF model. The rest of the data is used for evaluating both models.

Evaluation of retrieval results is done on two datasets which we call DB2 Prop, and DB2 Open, both of which contain questions and relevant answer links for DB2. The DB2 Prop dataset consists of proprietary user problems for which the correct answer url is provided by SMEs. The DB2 Open dataset is crawled from the developer works website [1]. It consists of user questions and urls extracted from "accepted" answers for those questions. We get manual annotations of attributes on 96 questions from DB2 Prop, and 48 questions from DB2 Open dataset from SMEs, and use these to evaluate retrieval performance.

For evaluation of retrieval results, we use elastic search as the retrieval index. The corpus is a crawl of publicly available technote pages [2] for the DB2 product, which are majority of the answers for both the DB2 Prop and DB2 Open datasets. From the downloaded pages, we extract and index the title and content of the body as document fields. The whole text of the problem is sent to this index (on both document

[1] https://developer.ibm.com/answers/topics/db2/

[2] https://www.ibm.com/mysupport/s/topic/0TO500000001fUNGAY/db2-linux-unix-and-windows, https://www.ibm.com/support/knowledgecenter/

Table 2: Distribution of questions with S, I, and A

Label	Count
Symptom	1773
Activity	121
Intent	510
Multiple Symptom	467
Multiple Activity	13
Multiple Intent	12

Table 3: Performance (%) Comparison for Symptoms: Strict Scoring

Method	Precision	Recall	F1
Semantic Parser	52.33	42.42	46.46
CRF	78.3	71.20	74.33

fields) as a baseline. The query containing the Semantic Parser extractions are sent with weights on query fields as described in section 3.3.

5 Evaluation

In this section we evaluate our system along two dimensions -

1. Precision, recall and F1 score for the extractions from the Semantic Parser and CRF
2. Relevance of the retrieved results between a whole query baseline and Semantic Parse based query

5.1 Evaluation of Attribute Extraction Models

Since annotations of these attributes were not readily available to us, we had Subject Matter Experts (SMEs) manually annotate 1972 questions from a proprietary ticketing system. The SMEs were asked to identify spans from questions which indicate the problem, efforts in resolving the problem, and explicit requests. 80% of these questions were used for training the CRF model, and the rest were used for evaluating both models. The distribution of symptom, activity and intent in our training dataset is shown in the Table 2. Longer more complex queries often had multiple symptoms, intent and activity (distributions shown in table 2).

As we can see, a large number of support questions have multiple symptoms. We designed two scoring schemes: strict scoring, where an extraction is marked correct only when all the values from ground truth are identified; and partial scoring, where the extraction is marked correct if any one of the values from ground truth is identified. The partial scoring scheme would not harshly evaluate models when most, but not all values of an attribute are extracted. Tables 3 and 4 summarize the performance of all the models. We present results on the strict scoring scheme only for symptoms, since there are very few questions with multiple activities and intents. For an attribute which has just one value in the ground truth, strict and partial scoring are the same. In Table 3, we note that CRF does the best in extracting multiple symptoms. The main reason is that CRF uses feature as well as contextual modeling of the words in the sentence. It inherently being a sequential labeling algorithm, accounts for its 78% precision in identifying all symptoms.

Note that CRF is a supervised learning model and needs manually annotated training data. The rule based Semantic Parser does not have this limitation and instead uses linguistic pattern matching on dependency trees. This renders it more generic and can be applied to problems from other products with minimal intervention. With a typical big service provider supporting close to 7-8k products, this is a useful property. We observe that the CRF outperforms the rule based Semantic Parser on Symptoms for which we have an abundance of training examples. For the other two attributes, the Semantic Parser is slightly better than the CRF.

Table 4: Performance (%) Comparison (Symptom, Activity and Intent): Partial Scoring

Model	Symptom			Activity			Intent		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Semantic Parser	67.81	58.84	62.17	71.33	37.2	48.64	88.44	59.29	70.63
CRF	88.64	81.60	84.35	70.0	33.33	44.85	93.74	48.67	64.07

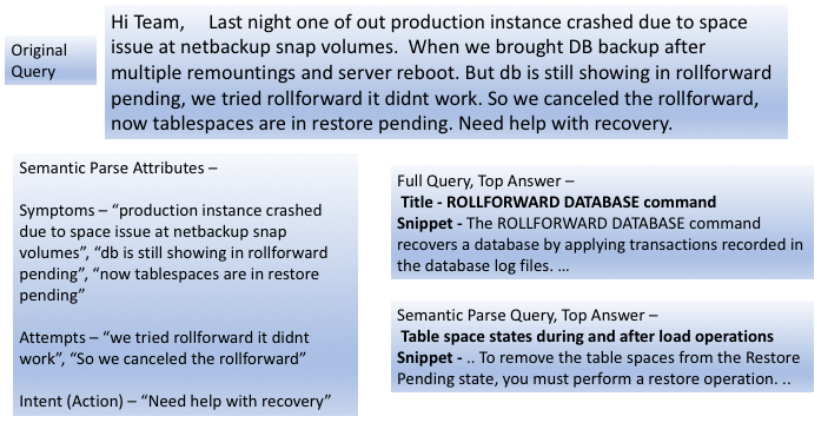


Figure 4: Question with extractions and retrieved results

5.2 Evaluation of retrieved results

As explained in the setup, evaluation of retrieved results is performed using elastic search. Figure 4 gives an example of the top retrieved results using the original query and the Semantic Parse attributes. The baseline query consists of the whole question, and extractions from the Semantic Parser are weighed according to the attribute-type in the reasoner. The highest weight is given to intent extractions, and a negative weight is assigned to attempt extractions. We see that because of repetitions of the token "rollforward" in the original query, the top result returned is about the "ROLLFORWARD DATABASE command". When we give a negative weight to the attempt Semantic Parse extraction, and increase the weight of the symptom extraction (although less than that of the intent extraction), the result about "Table space states" bubbles to the top.

The retrieval results (Precision@k) on the two datasets are presented in Table 5. Precision@k measures the fraction of questions with the correct answer in the top k results.

$$Precision@k = \frac{n_k * 100}{N} \tag{1}$$

where n_k is the number of questions with the correct answer in top k results, and N is the total number of questions.

We see a slight improvement in P@1 on the DB2 Prop dataset, and a 42.9% improvement in P@5. However, on the DB2 Open dataset, we see very slight improvement on the P@1 and no improvement in P@5. On further analysis we find that there are two reasons for this. Firstly, we find that for a large number of the queries, the answer marked as the ground truth is only partially related to the solution, and sometimes it is completely different. Since the accepted answer was assumed to be correct and there is no manual verification of the correctness in this dataset (unlike in DB2 Prop), we find that the confidence on ground truth in DB2 Open is low. Another reason we find is that in DB2 Open, people often attach code snippets and stack traces in the question, which can easily be understood by humans but are difficult for the models to annotate. Identifying and extracting out (and in the future, understanding) these code snippets and stack traces, would improve the accuracy on this dataset further.

Table 5: Retrieval Performance (P@k)

Model	DB2 Prop			DB2 Open		
	P@1	P@3	P@5	P@1	P@3	P@5
Whole Query	7.29	14.58	28.13	25.0	33.33	37.5
Semantic Parse Query	9.38	19.79	40.63	27.08	31.25	37.5

6 Discussion and Future Work

Through our experiments, evaluation and error analysis, we stumble across a number of problems that can be addressed to improve understanding of technical support questions, and increase retrieval accuracy. Some of the interesting problems are detailed below -

- **Sub-division of Activity** Through careful analysis of activity extractions, we find that it can further be split into two classes that help better understanding of the question. The first class are actions or steps that were taken before the problem occurred. For example, in the snippet - *We tried to upgrade the DB2 AESE ON RHEL from 10.5.5 to 11.1 and in upgrade check got orphan rows*, "trying to upgrade DB2" is not a step taken to remediate the problem, but the action that resulted in the error. The second class are actions that are taken to remediate the problem (the activity extraction in figure 4). It is clear that we would not want to exclude documents/procedures containing tokens from the first class from appearing in the results. The challenge with creating such a distinction, we find, is that examples from both these classes manifest in very similar linguistic patterns. Correct distinction between these two classes would probably require us to use position features from the symptom extraction.

- **Identify Stack Traces, Core Dumps, and Code Snippets** As identified in section 5.2, stack traces, core dumps, and code snippets, confuse the Semantic Parser. Returning these poor extractions hurts retrieval performance. Further, if we could truly understand these sections in the way humans do, it would certainly increase efficiency of retrieval.

- **Noisy Intent Extractions** Extraction of patterns like "please help" or "please suggest resolution" as intent, creates noise for retrieval. Such extractions could be filtered if they do not contain entity or action keywords from the domain.

- **Semantic Parsing performance** - We see that the CRF models outperforms the Semantic Parser in the presence of large amounts of training data by leveraging contextual features. However, for cases where training data is not available, it fails to reach comparable performance with the rule based parser. The Semantic Parser also has the advantage of being easily generalizable across products as it works with the semantic parse of sentences. An ideal model would be able to combine these two strengths, being generalizable and being able to leverage contextual features.

References

[Bordes and Weston2016] Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *CoRR*, abs/1605.07683.

[Chen et al.2016] Long Chen, Joemon M. Jose, Haitao Yu, Fajie Yuan, and Dell Zhang. 2016. A semantic graph based topic model for question retrieval in community question answering. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 287–296.

[Dang et al.2007] Hoa Trang Dang, Diane Kelly, and Jimmy J Lin. 2007. Overview of the trec 2007 question answering track. In *Trec*, volume 7, page 63.

[de Marneffe and Manning2008] Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8.

[Dhoolia et al.2017] Pankaj Dhoolia, P Chugh, P Costa, Neelamadhav Gantayat, M Gupta, N Kambhatla, R Kumar, S Mani, P Mitra, C Rogerson, et al. 2017. A cognitive system for business and technical support: A case study. *IBM Journal of Research and Development*, 61(1):7–74.

[Fan et al.2012] J. Fan, A. Kalyanpur, D. C. Gondek, and D. A. Ferrucci. 2012. Automatic knowledge extraction from documents. *IBM Journal of Research and Development*, 56(3.4):5:1–5:10.

[Finkel et al.2005] J. R. Finkel, T. Grenager, and C. D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Association for Computational Linguistics*.

[Flinders2015] Karl Flinders. 2015. Amelia, the ipsoft robot. *Computer Weekly*.

[Gupta et al.2017] Abhirut Gupta, Arjun Akula, Gargi Dasgupta, Pooja Aggarwal, and Prateeti Mohapatra. 2017. Desire: Deep semantic understanding and retrieval for technical support services. In Khalil Drira, Hongbing Wang, Qi Yu, Yan Wang, Yuhong Yan, François Charoy, Jan Mendling, Mohamed Mohamed, Zhongjie Wang, and Sami Bhiri, editors, *Service-Oriented Computing – ICSOC 2016 Workshops*, pages 207–210, Cham. Springer International Publishing.

[Lafferty et al.2001] J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *International Conference on Maching Learning*.

[Lally et al.2012] Adam Lally, John M. Prager, Michael C. McCord, Branimir Boguraev, Siddharth Patwardhan, James Fan, Paul Fodor, and Jennifer Chu-Carroll. 2012. Question analysis: How watson reads a clue. *IBM Journal of Research and Development*, 56(3):2.

[Li and Roth2006] Xin Li and Dan Roth. 2006. Learning question classifiers: the role of semantic information. *Natural Language Engineering*, 12(3):229–249.

[McCord et al.2012] Michael C. McCord, J. William Murdock, and Branimir Boguraev. 2012. Deep parsing in watson. *IBM Journal of Research and Development*, 56(3):3.

[McCord1990a] Michael C. McCord. 1990a. Slot grammar. In Rudi Studer, editor, *Natural Language and Logic*, pages 118–145, Berlin, Heidelberg. Springer Berlin Heidelberg.

[McCord1990b] Michael C. McCord. 1990b. Slot grammar: A system for simpler construction of practical natural language grammars. In *Proceedings of the International Symposium on Natural Language and Logic*, pages 118–145.

[Metzler and Croft2005] D. Metzler and B. W. Croft. 2005. Analysis of statistical question classification for fact-based questions. *Information Retrieval*, 8(3):481–504.

[Murdock et al.2012] J. William Murdock, James Fan, Adam Lally, Hideki Shima, and Branimir Boguraev. 2012. Textual evidence gathering and analysis. 56:8.

[Vtyurina and Clarke2016] Alexandra Vtyurina and Charles LA Clarke. 2016. Complex questions:let me google it for you. In *Proceedings of the second Web QA Workshop*.

[Wang et al.2012] Chang Wang, Aditya Kalyanpur, James Fan, Branimir Boguraev, and David Gondek. 2012. Relation extraction and scoring in deepqa. *IBM Journal of Research and Development*, 56(3):9.

[Yang et al.2015] Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018.

[Yang et al.2017] Shuo Yang, Lei Zou, Zhongyuan Wang, Jun Yan, and Ji-Rong Wen. 2017. Efficiently answering technical questions a knowledge graph approach.

Deconvolution-Based Global Decoding for Neural Machine Translation

Junyang Lin[1,2], Xu Sun[2], Xuancheng Ren[2], Shuming Ma[2], Jinsong Su[3], Qi Su[1]
[1]School of Foreign Languages, Peking University
[2]MOE Key Lab of Computational Linguistics, School of EECS, Peking University
[3]School of Software, Xiamen University
{linjunyang, xusun, renxc, shumingma, sukia}@pku.edu.cn
jssu@xmu.edu.cn

Abstract

A great proportion of sequence-to-sequence (Seq2Seq) models for Neural Machine Translation (NMT) adopt Recurrent Neural Network (RNN) to generate translation word by word following a sequential order. As the studies of linguistics have proved that language is not linear word sequence but sequence of complex structure, translation at each step should be conditioned on the whole target-side context. To tackle the problem, we propose a new NMT model that decodes the sequence with the guidance of its structural prediction of the context of the target sequence. Our model generates translation based on the structural prediction of the target-side context so that the translation can be freed from the bind of sequential order. Experimental results demonstrate that our model is more competitive compared with the state-of-the-art methods, and the analysis reflects that our model is also robust to translating sentences of different lengths and it also reduces repetition with the instruction from the target-side context for decoding.

1 Introduction

Deep learning has achieved tremendous success in machine translation, outperforming the traditional linguistic-rule-based and statistical methods. In recent studies of Neural Machine Translation (NMT), most models are based on the sequence-to-sequence (Seq2Seq) model based on the encoder-decoder framework (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014) with the attention mechanism (Bahdanau et al., 2014; Luong et al., 2015). While traditional linguistic-rule-based and statistical methods of machine translation require much work of feature engineering, NMT can be trained in the end-to-end fashion. Besides, the attention mechanism can model the alignment relationship between the source text and translation (Bahdanau et al., 2014; Luong et al., 2015), and some recent improved versions of attention have proved successful in this task (Tu et al., 2016; Mi et al., 2016; Meng et al., 2016; Xiong et al., 2017; Vaswani et al., 2017).

However, the decoding pattern of the recent Seq2Seq models is inconsistent with the linguistic analysis. As the conventional decoder translates words in a sequential order, the current generation is highly dependent on the previous generation and it is short of the knowledge about future generation. Nida (1969) pointed out that translation goes through a process of analysis, transfer and reconstruction, involving the deep syntactic and semantic structure of the source and target languages. Language generation involves complex syntactic analysis and semantic integration, instead of a step-by step word generation (Frazier, 1987). Moreover, from the perspective of semantics and pragmatics, the syntactic analysis of utterance can be guided by the global lexical-semantic and discourse information (Altmann and Steedman, 1988; Trueswell et al., 1994, 1993; Tyler and Marslen-Wilson, 1977). In brief, the process of translation is in need of the global information from the target-side context, but the decoding pattern of the conventional Seq2Seq model in NMT does not meet the requirement.

Recent researches in NMT have taken this issue into consideration by the implementation of bidirectional decoding. Some methods of bidirectional decoding (Liu et al., 2016; Cong et al., 2017) rerank the candidate translations with the scores from the bidirectional decoding. However, these bidirectional

Proceedings of the 27th International Conference on Computational Linguistics, pages 3260–3271
Santa Fe, New Mexico, USA, August 20-26, 2018.

decoding methods cannot provide effective complementary information due to the limited search space of beam search.

In this article, we extend the conventional attention-based Seq2Seq model by introducing the deconvolution-based decoder, which is a Convolutional Neural Network (CNN) to perform deconvolution. Recently, deconvolution has been applied to the studies of natural language (Zhang et al., 2017; Shen et al., 2017), which can be regarded as the transposition of the convolution (Long et al., 2015; Noh et al., 2015). Zhang et al. (2017) applied this method to natural language by modeling sentences with a convolution-deconvolution autoencoder. The study of Zhang et al. (2017) showed that the deconvolution solves the problems by reconstructing a representation of high quality irrespective to the order or length. It can be found that deconvolution owns the potential to provide global information for guidance of decoding. Therefore, we follow this idea and propose a new model with deconvolution for NMT.

To be specific, the conventional RNN encoder encodes the source sentences to new representations and sends the final state to the decoder, and the conventional RNN decoder decodes it to the target sentences with the attention to the encoder outputs. In our model, our designed deconvolution-based decoder decodes the final state of the encoder to a matrix representing the global information of the target-side contexts. Each column of the matrix is learned to be close to the word embedding of the target words. The conventional RNN decoder can attend to the columns for the information of the target-side context to perform global decoding in the translation.

Our contributions in this study are illustrated in the following:

- We propose a new model for NMT, which contains a deconvolution-based decoder to provide global information of the target-side contexts to the RNN decoder, so that the model is able to perform global decoding[1].
- Experimental results demonstrate that our model outperforms the baseline models in both the Chinese-to-English translation and the English-to-Vietnamese translation, outperforming the Seq2Seq model in the BLEU score evaluation with the advantages of BLEU score 2.82 and 1.54 respectively.
- The analysis shows that our model that performs global decoding is more capable of reducing repetition and more robust to the translation of sentences of different lengths, and the case study reflects that it is able to capture the syntactic structure for the translation and has a better reflection of the semantic meaning of the source text.

2 Model

In the following, we introduce the details of our model, including the encoder, the deconvolution-based decoder and the conventional RNN-based decoder. The functions of each decoder are illustrated below to show how they collaborate to improve the quality of the translation.

2.1 Encoder

In our model, the encoder reads the embeddings of the input text sequence $x = \{x_1, ..., x_n\}$ and encodes a sequence of encoder outputs $h = \{h_1, ..., h_n\}$. The final hidden state h_n is sent to the decoder as the initial state for it to decode a sequence of output text. The encoder outputs provide the information of the source-side contexts to our RNN-Based decoder through the attention mechanism.

The encoder in our model is a bidirectional LSTM (Hochreiter and Schmidhuber, 1997), which reads the input in two directions to generate two sequences of hidden states $\overrightarrow{h} = \{\overrightarrow{h_1}, \overrightarrow{h_2}, \overrightarrow{h_3}, ..., \overrightarrow{h_n}\}$ and $\overleftarrow{h} = \{\overleftarrow{h_1}, \overleftarrow{h_2}, \overleftarrow{h_3}, ..., \overleftarrow{h_n}\}$, where:

$$\overrightarrow{h_i} = LSTM(x_i, \overrightarrow{h_{i-1}}, C_{i-1}) \quad (1)$$

$$\overleftarrow{h_i} = LSTM(x_i, \overleftarrow{h_{i-1}}, C_{i-1}) \quad (2)$$

The encoder outputs corresponding to each time step are concatenated as $h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]$.

[1]The code is released in `https://github.com/lancopku/DeconvDec`

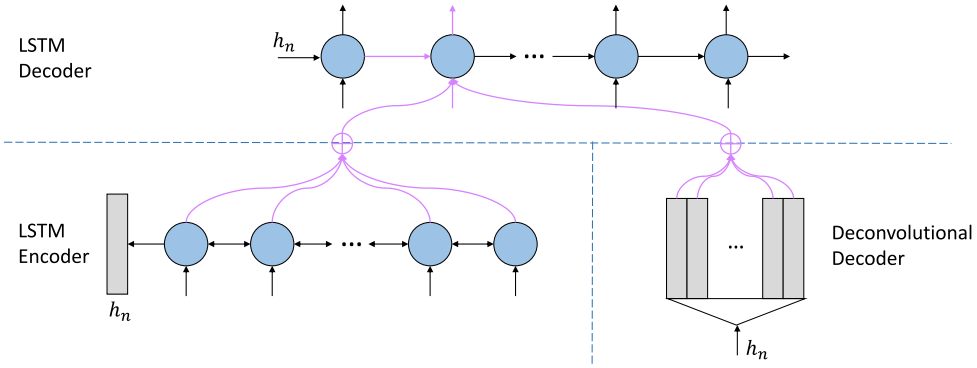


Figure 1: Model architecture. There are three components in the proposed model, i.e., the LSTM encoder, the deconvolution-based decoder, and the conventional LSTM decoder. The encoder distills the input sentence into a state h_n, which is then used in the deconvolution-based decoder to obtain the global information of the target-side contexts. Based on the target-side contexts and the input-side contexts, the conventional LSTM decoder generates the output from the state h_n.

2.2 Deconvolution-Based Decoder

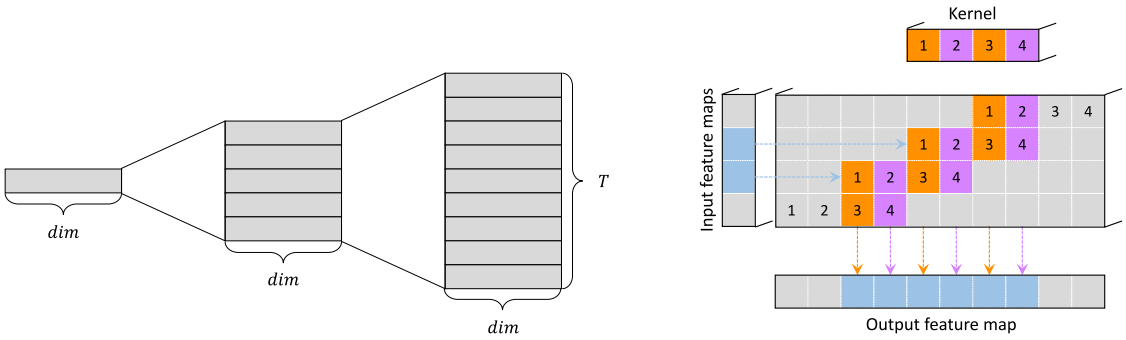
In our model, there are two decoders, which perform different tasks for the whole decoding process. While the RNN-based decoder is similar to the conventional decoder, which decodes the output text sequence in a sequential order and attends to the annotations of the encoder via the attention mechanism, our proposed deconvolution-based decoder does not decode the text but provide global information of the target-side contexts to the RNN-based decoder so that it can decode structurally instead of sequentially. To be specific, the deconvolution-based decoder learns to generate the word embedding matrix of the target text sequence.

In order to provide global information of the target-side contexts to the RNN decoder, we implement a multilayer CNN as the deconvolution-based decoder to perform deconvolution. With deconvolution, it is available for a vector or a small matrix to be transformed to a large matrix. In our model, the deconvolution is implemented on the final states in both directions from the encoder. As words in our model are represented with word embedding vectors, sentences can be formed as word embedding matrices. In our model, the deconvolution-based decoder is designed to learn word embedding matrices of the target sequences with the representation matrix from the encoder. As the conjugate operation of convolution, deconvolution expands the dimension of the input representation to a matrix of our designed size. There are L layers in the deconvolution, each of which has f_l filters of kernel size k_l. The i^{th} filter $W_l^i \in R^{k \times dim}$ (dim refers to the size of the input representation vector) with stride s_l^i and padding p_l^i performs deconvolution on the input representation matrix $I \in R^{m \times dim}$ ($m \times dim$ refers to the size of the input representation matrix), the final hidden state of the encoder. The computation of convolution is illustrated as below:

$$c_l^i = g(W_l^i * X + b) \tag{3}$$

where X refers to the convolved matrix and g refers to non-linear activation function, which is ReLU (Nair and Hinton, 2010) following Zhang et al. (2017), and deconvolution is its transposed operation. With the input I, our objective of the deconvolution operation is to generate a word embedding matrix $E \in R^{T \times dim}$ where T refers to the sentence length designed for the output text sequence, which is a hyper-parameter. At the l^{th} layer, deconvolution generates a matrix $E_l \in R^{T_l \times dim}$ where $T_l = T_{l-1} \times s_l + k_l - 2 \times p_l$. With the control of stride and kernel size, the height of the matrix can be assigned, and with the control of the number of filters, the width of the matrix can also be assigned. In our model, they are the length of the output sequence and the dimension of the word embedding respectively.

The deconvolution-based decoder can generate meaningful representation with information different from that in the conventional RNN decoder. The conventional RNN decoder generates sequence in a way similar to Markov Decision Process, which is highly dependent on the previous generation and follows



(a) Deconvolution-based decoder. (b) Deconvolution.

Figure 2: Deconvolution-based decoder. On the right shows an example of a 1d deconvolution on a input of size 2 with a kernel of size 4, a padding of 1 and, a stride of 2. The depth means the dimension of the channel, which is dim in our case.

a strict sequential order, so it contains high sequential dependency. On the contrary, the deconvolution-based decoder generates a word embedding matrix depending on the representation from the encoder without considering the order, which does not have the problem of long-term dependency. Moreover, although it is not capable as the RNN encoder to generate coherent text, it reveals the information of the text from a global perspective, including syntactic and semantic features.

2.3 RNN-Based Decoder

Different from the deconvolution-based decoder, the RNN-based decoder is responsible for decoding the representation h_n to generate the translation $y = \{y_1, ..., y_m\}$. With the final encoder state as the initial state, the decoder is initialized to decode in sequential order, until it generates the token representing the end of sentence. During decoding, the attention mechanism is applied for the decoder to extract the information from the source-side contexts, which are the annotations of the encoder, as well as the information from the target-side contexts, which are the outputs of the deconvolution-based decoder.

For the RNN-based decoder, we implement a unidirectional LSTM. The output of the RNN-based decoder at each time step is sent into a feed-forward neural network to be projected into the space of the target vocabulary $Y \in R^{|Y| \times dim}$ for the prediction of the translated word. At each time step, the decoder generates a word y_t by sampling from a conditional probability distribution of the target vocabulary P_{vocab}, where:

$$P_{vocab} = softmax(W_o v_t) \tag{4}$$

$$v_t = g(s_t, c_t, \tilde{c}_t) \tag{5}$$

$$s_t = LSTM(y_{t-1}, s_{t-1}, C_{t-1}) \tag{6}$$

where $g(\cdot)$ refers to non-linear activation function, and c_t and $\tilde{c}_t$ are the outputs of the attention mechanism, which are illustrated in the following.

The attention mechanism in our model is the global attention mechanism (Luong et al., 2015). Different from the conventional attention mechanism, which only computes the attention scores on the source-side contexts, the attention mechanism in our model consists of two parts. The first one is similar to the conventional one, attending to the source-side contexts from the encoder, but the second one is original, which attends to the target-side contexts, which is the word embedding matrix generated by the deconvolution-based decoder. By attending to the encoder annotations, the model computes the attention $\alpha_{t,i}$ of the RNN-based decoder output s_t on the annotations of the encoder h_i and generates the context vector c_t. Similarly, by attending to the outputs of the deconvolution-based decoder, the RNN-based decoder computes the attentions of s_t on each column E_i of its matrix E and generates the context vector

$\tilde{c}_t$:

$$c_t = \sum_{i=1}^{n} \alpha_{t,i} h_i \tag{7}$$

$$\tilde{c}_t = \sum_{i=1}^{n} \tilde{\alpha}_{t,i} E_i \tag{8}$$

where $\alpha_{t,i}$ and $\tilde{\alpha}_{t,i}$ are defined as below (as they are computed in the same way, they are both represented by $\alpha_{t,i}$ and the annotations are represented by x_i):

$$\alpha_{t,i} = \frac{exp(e_{t,i})}{\sum_{j=1}^{n} exp(e_{t,j})} \tag{9}$$

$$e_{t,i} = s_{t-1}^{\top} W_a x_i \tag{10}$$

2.4 Training

The training for the Seq2Seq model is usually based on maximum likelihood estimation. Given the parameters θ and source text x, the model generates a sequence $\tilde{y}$. The learning process is to minimize the negative log-likelihood between the generated text $\tilde{y}$ and reference y, which in our context is the sequence in target language for machine translation:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \log P(y_t^{(i)} | \tilde{y}_{<t}^{(i)}, x^{(i)}, \theta) \tag{11}$$

where the loss function is equivalent to maximizing the conditional probability of sequence y given parameters θ and source sequence x.

However, as there are two decoders in our model, the loss function should also be designed for the deconvolution-based decoder. In our model, we compute the smooth L1 loss between the generated matrix of the deconvolution-based decoder E and the word embedding matrix $\tilde{E}$ (which both contain M elements), which is more robust to outliers (Girshick, 2015), as well as the cross-entropy loss between the prediction of the deconvolution-based decoder $\hat{y}$ and reference y given the parameters of the encoder and the deconvolution-based decoder $\theta^{'}$. Therefore, the generated matrix E can be closer to the word embedding matrix $\tilde{E}$, and it contains information beneficial to the prediction of the target words. Moreover, for the cross entropy loss of the deconvolution-based decoder, we apply the method of Ma et al. (2018a) as it increases no parameter for the prediction by computing the cosine similarity between the output and the word embeddings. To sum up, the loss function is defined as below:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} (\sum_{t=1}^{T} \log P(y_t^{(i)} | \tilde{y}_{<t}^{(i)}, x^{(i)}, \theta) + \sum_{m=1}^{M} smooth_{L1}(E_m - \tilde{E}_m) + \sum_{t=1}^{T} \log P(y_t^{(i)} | x^{(i)}, \theta^{'})) \tag{12}$$

where smooth L1 loss is defined below:

$$smooth_{L1}(x, y) = \begin{cases} 0.5 \mid\mid x - y \mid\mid_2^2 & \text{if } \mid\mid x - y \mid\mid < 1 \\ \mid\mid x - y \mid\mid_1 - 0.5 & \text{if } \mid\mid x - y \mid\mid \geq 1 \end{cases} \tag{13}$$

We have tested L1 loss, L2 loss as well as smooth L1 loss in our experiments and found that smooth L1 loss encourages the model to reach the best performance.

3 Experiment

3.1 Datasets

We evaluate our proposed model on the NIST translation task for the Chinese-to-English translation and provide the analysis on the same task. Moreover, in order to evaluate the performance of our model on

the low-resource translation, we also evaluate our model on the IWLST 2015 (Cettolo et al., 2015) for the English-to-Vietnamese translation task.

Chinese-to-English Translation For the NIST translation task, we train our model on 1.25M sentence pairs extracted from LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06, with 27.9M Chinese words and 34.5M English words. Following Wang et al. (2016), we validate our model on the dataset for the NIST 2002 translation task and tested our model on that for the NIST 2003, 2004, 2005, 2006 translation tasks. We use the most frequent 30K words for both the Chinese vocabulary and the English vocabulary, which includes around 97.4% and 99.5% of the Chinese and English words in the training data. The sentence pairs longer than 50 words are filtered. The evaluation metric is BLEU (Papineni et al., 2002).

English-to-Vietnamese Translation The data is from the translated TED talks, containing around 133K training sentence pairs provided by the IWSLT 2015 Evaluation Campaign (Cettolo et al., 2015). We follow the studies of Huang et al. (2017), and use the same preprocessing methods as well as the same validation and the test set. The validation set is the TED tst2012 with 1553 sentences and the test set is the TED tst2013 with 1268 sentences. The English vocabulary is 17.7K words and the Vietnamese vocabulary is 7K words. The evaluation metric is also BLEU score.

3.2 Setting

We implement the models on PyTorch[2], and the experiments are conducted on an NVIDIA 1080Ti GPU. Both the size of word embedding and the number of units in the hidden layers are 512, and the batch size is 64. We use Adam optimizer (Kingma and Ba, 2014) to train the model with the setting $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 1 \times 10^{-9}$ following Vaswani et al. (2017), and we initialize the learning rate to 0.0003.

Gradient clipping is applied so that the norm of the gradients cannot be larger than a constant, which is 10 in our experiments. Dropout is used with the dropout rate set to 0.3 for the Chinese-to-English translation and 0.4 for the English-to-Vietnamese translation, based on the performance on the validation set.

Based on the performance on the validation set, we use beam search with a beam width of 10 to generate translation for the evaluation and test, and we normalize the log-likelihood scores by sentence length.

3.3 Baselines

For the Chinese-to-English translation, we compare our model with the state-of-the-art NMT systems for the task.

- **Moses** An open source phrase-based translation system with default configurations and a 4-gram language model trained on the training data for the target language;
- **RNNsearch** An attention-based Seq2Seq with fine-tuned hyperparameters (Bahdanau et al., 2014);
- **Coverage** The method extends RNNSearch with a coverage model for the attention mechanism that tackles the problem of over-translation and under-translation (Tu et al., 2016);
- **Lattice** The Seq2Seq model with a word-lattice-based RNN encoder that tackles the problem of tokenization in NMT (Su et al., 2016);
- **InterAtten** The Seq2Seq model that records the interactive history of decoding (Meng et al., 2016);
- **MemDec** Based on the RNNSearch, it is equipped with external memory that the model reads and writes during decoding (Wang et al., 2016).

For the English-to-Vietnamese translation, we compare our model with the recent NMT models for this task, and we present the results of the baselines reported in their articles.

[2] `http://pytorch.org`

Model	MT-03	MT-04	MT-05	MT-06	Ave.
Moses	32.43	34.14	31.47	30.81	32.21
RNNSearch	33.08	35.32	31.42	31.61	32.86
Lattice	34.32	36.50	32.40	32.77	34.00
Coverage	34.49	38.34	34.91	34.25	35.49
InterAtten	35.09	37.73	35.53	34.32	35.67
MemDec	36.16	**39.81**	35.91	35.98	36.97
Seq2Seq+Attention	35.32	37.25	33.52	33.54	34.91
+DeconvDec	**38.04**	39.75	**36.77**	**36.32**	**37.73**

Table 1: Results of our model and the baselines (the results are those reported in the referred articles, and the models are trained on the identical training data or larger training data) on the Chinese-to-English translation, tested on the NIST Machine Translation tasks in 2003, 2004, 2005, 2006 by BLEU score evaluation.

Model	BLEU
RNNSearch-1	23.30
RNNSearch-2	26.10
LabelEmb	26.80
NPMT	27.69
Seq2Seq+Attention	26.93
+DeconvDec	**28.47**

Table 2: Results of our model and the baselines (directly reported in the referred articles) on the English-to-Vietnamese translation, tested on the TED tst2013 with the BLEU score evaluation.

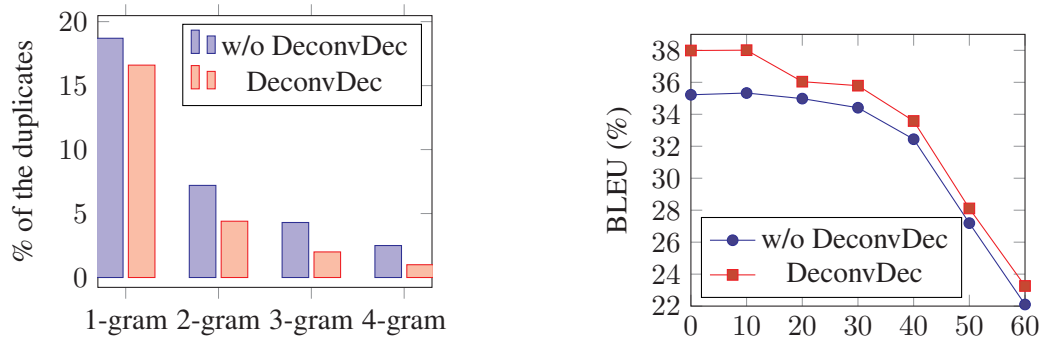
- **RNNsearch-1** The attention-based Seq2Seq model by Luong and Manning (2015);
- **RNNsearch-2** The implementation of the attention-based Seq2Seq by Huang et al. (2017);
- **LabelEmb** Extending RNNSearch with soft target representation (Sun et al., 2017);
- **NPMT** The Neural Phrased-based Machine Translation model by Huang et al. (2017);

4 Results and Analysis

4.1 Results

Table 1 shows the overall results of the models on the Chinese-to-English translation task. Beside our reimplementation of the attention-based Seq2Seq model, we report the results of the recent NMT models, which are results in their original articles or improved results of the reimplementation. To facilitate fair comparison, we compare with the baselines that are trained on the same training data. The results have shown that for the NIST 2003, 2004, 2005 and 2006 translation tasks, our model with the deconvolution-based decoder outperforms the baselines, and the advantage of BLEU score over the attention-based Seq2Seq model is 2.82 on average compared with our reimplementation of the attention-based Seq2Seq model. From the results mentioned above, it can be inferred that the global information of the target-side contexts retrieved from the deconvolution-based decoder is contributive to the translation. Our analysis and case study in the following can further demonstrate how the deconvolution-based decoder improves the attention-based Seq2Seq model.

Table 2 presents the results of the models on the English-to-Vietnamese translation. Compared with the attention-based Seq2Seq model, including the implementation with the strongest performance, our model with the deconvolution-based decoder can outperform it with the advantage of BLEU score 1.54. We also display the most recent model NPMT (Huang et al., 2017) trained and tested on the dataset.



(a) Percentage of the duplicates at the sentence level.

(b) BLEU scores on sentences of different lengths

Figure 3: **Percentage of the duplicates at sentence level and the BLEU scores on sentences of different lengths** Tested on the NIST 2003 dataset. The red bar and line indicate the performance of our model, and the blue bar and line indicate that of the attention-based SeqSeq model.

Compared with NPMT, our model has an advantage of BLEU score 0.78. It can be indicated that for the low-resource translation, the information from the deconvolution-based decoder is important, which brings significant improvement to the conventional attention-based Seq2Seq model.

4.2 Analysis

As our model generates translation with global information from the deconvolution-based decoder, it should learn to reduce repetition as it can learn to avoid generating same contents according to the conjecture by the deconvolution-based decoder about the target-side contexts. In order to test whether our model can mitigate the problem of repetition in translation, we test the repetition on the NIST 2003 dataset, following See et al. (2017). The proportions of the duplicates of 1-gram, 2-gram, 3-gram and 4-gram in each sentence are calculated. Results on Figure 3(a) show that our model generates less repetitive translation. In particular, the proportion of duplicates of our model is less than half of that of the conventional Seq2Seq model.

Moreover, to validate its robustness on different sentence-length levels, we test the BLEU scores on sentences of length no shorter than 10 to 60 of the NIST 2003 dataset. According to the results on Figure 3(b), though with the increase in length, the performance of our model is always stronger than the Seq2Seq model. However, with the increase of length, the advantage of our model becomes smaller. This is consistent with our hypothesis. Since the length of generation of the deconvolution-based decoder is assigned a particular value (30 words in Chinese-to-English translation) due to the limited computation resources, there is not enough global target-side information for translating long sentences (say, longer than 30 words). In our future work, we will delve into this problem and conduct further research to reduce computation cost.

Figure 4 presents the attention heatmaps of the RNN-based decoder on the generated matrix of the deconvolution-based decoder in the English-to-Vietnamese translation. They reflect that the RNN-based decoder has diverse local focuses on the self-contained target-side contexts at different time steps. Contrary to the conventional attention on the source-side contexts which captures the corresponding annotations, it focuses on groups of the columns of the generated matrix from the deconvolution-based decoder. With the guidance of the information of global decoding, the model generates translation of higher accuracy and higher coherence. However, as the deconvolution-based decoder is not responsible for generating translation, it is hard to interpret what each column of the generated matrix represents. Moreover, as it does not capture alignment relationship as the conventional attention mechanism does, it is our future work to improve the attention on the outputs of the deconvolution-based decoder and explain the group focuses as shown in the heatmaps.

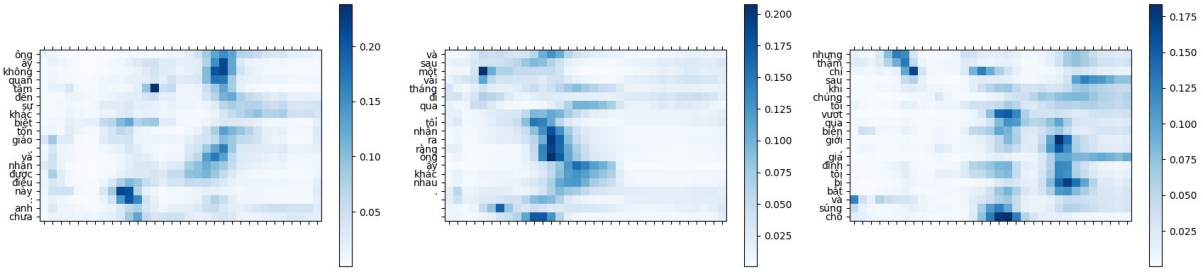


Figure 4: **Attention heatmaps of the RNN-based decoder on the deconvolution-based decoder** Words on the left refer to the translation of the RNN-based decoder. The heatmaps show that the RNN-based decoder can focus on certain parts of the outputs from the deconvolution-based decoder.

4.3 Case Study

Table 3 demonstrates three examples of the translation of our model in comparison with the Seq2Seq model and the golden translation. In Table 3(a), while the Seq2Seq model cannot recognize the objects of the main clause and the infinitive, causing inaccuracy and repetition, our model better captures the syntactic structure of the sentence and translates the main idea of the source text, though leaving the information "that causes disease". In Table 3(b), the source sentence is more complicated. With a temporal adverbial clause, its syntactic structure is more complex than simple sentence. The translation of the conventional Seq2Seq model cannot capture the syntactic information in the source text and regards the "parliament members" as the argument of the predicate "talk". Moreover, it is confused by the word "中期", meaning "middle", and translates "mid - autumn festival". In comparison, our model can recognize the adverbial clause and the main clause as well as their syntactic structures. In Table 3(c), it can be shown that when translating the long and relatively complex text, the baseline model makes a series of mistake of repetition. In contrast, the translation generated by our model though repeats the word "disaster", it is much more coherent and more semantically consistent with the source text as it successfully presents "sent 250,000 yuan" corresponding to the source text "调拨25万元人民币", while the baseline cannot translate the content.

5 Related Work

In the following, we review the studies in NMT and the application of deconvolution in NLP.

Kalchbrenner and Blunsom (2013); Cho et al. (2014); Sutskever et al. (2014) studied the application of the encoder-decoder framework on the machine translation task, which launched the development of NMT. Another significant innovation in this field is the attention mechanism, which builds connection between the translated contents and the source text (Bahdanau et al., 2014; Luong et al., 2015). To improve the quality of NMT, researchers have focused on improving the attention mechanism. Tu et al. (2016) and Mi et al. (2016) modeled coverage in the NMT, Meng et al. (2016) and Xiong et al. (2017) incorporated the external memory to the attention, and Xia et al. (2017) as well as Lin et al. (2018a) utilized the information from the previous generation by target-side attention and memory for attention history respectively. For more target-side information, Ma et al. (2018b) incorporated bag of words as target. A breakthrough of NMT in recent years is that Vaswani et al. (2017) invented a model only with the attention mechanism that reached the state-of-the-art performance.

Although many researches in NLP focused on the application of RNN, CNN is also an important type of network for the study of language (Kim, 2014; Kalchbrenner et al., 2014; Zhang et al., 2015; Lin et al., 2018b). Also, its application in NMT has been successful (Gehring et al., 2017). Recently, deconvolution was applied to modeling text (Zhang et al., 2017; Shen et al., 2017), which is able to construct a representation of high quality with the self-contained information.

Text: 基因 科学家 的 目标 是 , 提供 诊断 工具 以 发现 致病 的 缺陷 基因
Gold: the goal of geneticists is to provide diagnostic tools to identify defective genes that cause diseases
Seq2Seq: the objective of genetic scientists is to provide genes to detect genetic genetic genes
DeconvDec: the objective of the gene scientist is to provide diagnostic tools to detect the defective genes

(a)

Text: 叛军 暗杀 两位 菲 国 国会 议员 后, 菲律宾 总统 雅罗育 在 二零零一 年 中期 停止 与 共党 谈判 。
Gold: after the rebels assassinated two philippine legislators , philippine president arroyo ceased negotiations with the communist party in mid 2001 .
Seq2Seq: philippine president gloria arroyo stopped the two philippine parliament members in the mid - autumn festival .
DeconvDec: philippine president gloria arroyo stopped holding talks with the communist party after the rebels assassinated two philippine parliament members .

(b)

Text: 中国 红十字会 已 在 24日 地震 发生 后 紧急 向 新疆 灾区 调拨 25万 元 人民币 , 又 于 25日 向 灾区 派出 救灾 工作组 。
Gold: china red cross has released 250 thousand renminbi for the xinjiang disaster area immediately after the earthquake on the 24th . a disaster relief team was dispatched to the area on the 25th .
Seq2Seq: the red cross society of china (red cross) , the china red cross society (red cross) , emergency relief team sent an emergency team to xinjiang for disaster relief in the disaster areas after the earthquake on 24 june .
DeconvDec: the china red cross society has sent 250,000 yuan to the disaster areas in xinjiang after the earthquake occurred on the 24 th, and sent a relief team to disaster disaster areas on the 25 th.

(c)

Table 3: Two examples of the translation of our model in comparison with the conventional attention-based Seq2Seq model on the NIST 2003 Chinese-to-English translation task. The errors in the translation are colored in red and the successful translation of some particular contents are colored in yellow (e.g., the contents that the model successfully translates but the other does not).

6 Conclusion and Future Work

In this paper, we propose a new model with the global decoding mechanism. With our deconvolution-based decoder, which provides global information of the target-side contexts, the model can effectively exploit the information for the inference of syntactic structure and semantic meaning in the translation. Experimental results on the Chinese-to-English translation and English-to-Vietnamese translation demonstrate that our model outperforms the baseline models, and the analysis shows that our model generates less repetitive translation and demonstrates higher robustness to the sentences of different lengths. Furthermore, the case study shows that the translation of our model better observes the syntactic and semantic requirements for the translation and generates coherent and accurate translation with fewer irrelevant contents.

In the future, we will further develop analysis of the mechanism of deconvolution in NMT and try to figure out its generalized patterns for the construction of the target-side contexts.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No. 61673028) and the National Thousand Young Talents Program. Xu Sun is the corresponding author of this paper.

References

G Altmann and M Steedman. 1988. Interaction with context during human sentence processing. *Cognition*, 30(3):191.

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

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. 2015. The iwslt 2015 evaluation campaign. *Proc. of IWSLT, Da Nang, Vietnam*.

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

Duy Vu Hoang Cong, Gholamreza Haffari, and Trevor Cohn. 2017. Towards decoding as continuous optimisation in neural machine translation. In *EMNLP 2017*, pages 146–156.

Lyn Frazier. 1987. *Sentence Processing: A Tutorial Review*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML 2017*, pages 1243–1252.

Ross Girshick. 2015. Fast r-cnn. *Computer Science*.

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

Po-Sen Huang, Chong Wang, Dengyong Zhou, and Li Deng. 2017. Neural phrase-based machine translation. *CoRR*, abs/1706.05565.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP 2013*, pages 1700–1709.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL 2014*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP 2014*, pages 1746–1751.

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

Junyang Lin, Shuming Ma, Qi Su, and Xu Sun. 2018a. Decoding-history-based adaptive control of attention for neural machine translation. *CoRR*, abs/1802.01812.

Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. 2018b. Global encoding for abstractive summarization. *CoRR*, abs/1805.03989.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Agreement on target-bidirectional neural machine translation. In *NAACL HLT 2016*, pages 411–416.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *CVPR 2015*, pages 3431–3440.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP 2015*, pages 1412–1421.

Shuming Ma, Xu Sun, Wei Li, Sujian Li, Wenjie Li, and Xuancheng Ren. 2018a. Query and output: Generating words by querying distributed word representations for paraphrase generation. In *NAACL-HLT 2018*, pages 196–206.

Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. 2018b. Bag-of-words as target for neural machine translation. *CoRR*, abs/1805.04871.

Fandong Meng, Zhengdong Lu, Hang Li, and Qun Liu. 2016. Interactive attention for neural machine translation. In *COLING 2016*, pages 2174–2185.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation. In *EMNLP 2016*, pages 955–960.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML 2010*, pages 807–814.

Eugene A Nida. 1969. Science of translation. *Language*, pages 483–498.

Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. 2015. Learning deconvolution network for semantic segmentation. In *ICCV 2015*, pages 1520–1528.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL, 2002*, pages 311–318.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL 2017*, pages 1073–1083.

Dinghan Shen, Yizhe Zhang, Ricardo Henao, Qinliang Su, and Lawrence Carin. 2017. Deconvolutional latent-variable model for text sequence matching. *CoRR*, abs/1709.07109.

Jinsong Su, Zhixing Tan, Deyi Xiong, and Yang Liu. 2016. Lattice-based recurrent neural network encoders for neural machine translation. *CoRR*, abs/1609.07730.

Xu Sun, Bingzhen Wei, Xuancheng Ren, and Shuming Ma. 2017. Label embedding network: Learning label representation for soft training of deep networks. *CoRR*, abs/1710.10393.

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

J. C. Trueswell, M. K. Tanenhaus, and S. M. Garnsey. 1994. Semantic influences on parsing: Use of thematic role information in syntactic ambiguity resolution. *Journal of Memory and Language*, 33(3):285–318.

J. C. Trueswell, M. K. Tanenhaus, and C Kello. 1993. Verb-specific constraints in sentence processing: separating effects of lexical preference from garden-paths. *Journal of Experimental Psychology Learning Memory and Cognition*, 19(3):528–553.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *ACL 2016*.

Lorraine K. Tyler and William D. Marslen-Wilson. 1977. The on-line effects of semantic context on syntactic processing. *Journal of Verbal Learning and Verbal Behavior*, 16(6):683–692.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2016. Memory-enhanced decoder for neural machine translation. In *EMNLP 2016*, pages 278–286.

Yingce Xia, Fei Tian, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Sequence generation with target attention. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18-22, 2017, Proceedings, Part I*, pages 816–831.

Hao Xiong, Zhongjun He, Xiaoguang Hu, and Hua Wu. 2017. Multi-channel encoder for neural machine translation. *CoRR*, abs/1712.02109.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS, 2015*, pages 649–657.

Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. 2017. Deconvolutional paragraph representation learning. In *NIPS 2017*, pages 4172–4182.

Pattern-revising Enhanced Simple Question Answering over Knowledge Bases

Yanchao Hao[1,2], **Hao Liu**[1], **Shizhu He**[1], **Kang Liu**[1,2], **Jun Zhao**[1,2]
[1] National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
{yanchao.hao, hao.liu2017, shizhu.he, kliu, jzhao}@nlpr.ia.ac.cn

Abstract

Question Answering over Knowledge Bases (KB-QA), which automatically answer natural language questions based on the facts contained by a knowledge base, is one of the most important natural language processing (NLP) tasks. Simple questions constitute a large part of questions queried on the web, still being a challenge to QA systems. In this work, we propose to conduct pattern extraction and entity linking first, and put forward pattern revising procedure to mitigate the error propagation problem. In order to learn to rank candidate subject-predicate pairs to enable the relevant facts retrieval given a question, we propose to do joint fact selection enhanced by relation detection. Multi-level encodings and multi-dimension information are leveraged to strengthen the whole procedure. The experimental results demonstrate that our approach sets a new record in this task, outperforming the current state-of-the-art by an absolute large margin.

1 Introduction

As the amount of the knowledge bases (KBs) grows, such as DBpedia[1], Freebase[2], and WikiData[3], people are paying more attention to seeking effective methods for accessing these precious intellectual resources. While knowledge bases are usually very large and not easily accessible for users. KB-QA (Unger et al., 2014), which takes natural language as query language, is a more user-friendly solution, and has become a research focus in recent years. At the same time the growing amount of data has led to a heterogeneous data landscape where QA systems struggle to keep up with the volume, variety and veracity of the underlying knowledge.

Simple question answering over knowledge bases, which answers a question using a single fact in knowledge bases, is not simple at all and far from being solved. This is the setup of the SimpleQuestions benchmark recently presented by Bordes et al. (2015). The task of KB-QA for simple questions could be put as follows. Let $\mathcal{G} = (s_i, p_i, o_i)$ be a background knowledge base represented as a set of triples, where s_i represents a subject entity, p_i a predicate (also denoted as relation), and o_i an object entity. Given a natural language question represented as an utterance $q = w_1, ..., w_n$, the task of simple question answering is to find a triple $(\hat{s}, \hat{p}, \hat{o}) \in \mathcal{G}$ such that $\hat{o}$ is the intended answer for question q. This task can be formulated to finding the best matches of subject $\hat{s}$ and predicate $\hat{p}$.

$$\hat{s}, \hat{p} = \arg\max_{s,p \in \mathcal{G}} p(s, p|q) \tag{1}$$

The setup of using a single fact to answer a question seems simple, while there are many existing challenges faced by QA systems. One challenge is the extremely complex language used in question utterance. On one hand, in many different ways natural language is used to express the same information need. On the other hand, the same word used in different sentences may express different meaning. For example, two questions "*who created Apple Inc.*" and "*who started Apple*" have the same meaning and

[1] http://dbpedia.org
[2] https://developers.google.com/freebase/
[3] https://www.wikidata.org

Proceedings of the 27th International Conference on Computational Linguistics, pages 3272–3282
Santa Fe, New Mexico, USA, August 20-26, 2018.

in order to retrieve the correct answer we need to map the meaning conveyed by them to the "*founder*" relation in KB. The word "*Apple*" in the questions refers to the "*Apple Corporation*" instead of the fruit "*apple*". Natural language is complex while KB is much more universal. Another challenge is the vast amount of facts in large scale KB.

KB is quite large to some extent. KB has a huge number of fact triples and entities. A common and effective way to remain only a small subset of facts and entities is to conduct entity linking of a question over KB firstly. There are some work conducting entity linking by searching n-gram words of a question among all entity names (Bordes et al., 2015; Golub and He, 2016). While some other work propose a special-purpose sequence labeling network to focus on more probable candidates in question utterance, then linking them to entities (Dai et al., 2016; Yin et al., 2016b). Previous approaches assume that their preceding steps are correct to produce a good result for next procedure. While it should be pointed out that both the labeling and the entity linking process may lead to error propagation problems. For example, if the previous steps provide a wrong labeling result or don't recall the gold subject entity in the generated candidates, the following procedure can't make a good choice to retrieve the gold subject and predicate pair.

Faced with these problems and following previous work, we propose to do pattern extraction and entity linking first to extract possible entity mention and question pattern[4], and reduce the large number of candidate entities in KB. Due to the performance of the pattern extraction step, we observe that there is a certain proportion of bad cases in the extracted mention-pattern pairs. So we propose a novel pattern revising procedure to improve the quality of the extracted patterns. Then the extracted mentions are used to identify candidate entities in KB that the question refers to. This is the entity linking procedure. The candidate fact pool is formed by incorporating all the predicates connected to the corresponding subjects. The next step is to select the gold subject and predicate from the candidate fact pool.

For one question, there may be a large set of candidate entities. It is unrealistic to use all of them to generate candidate facts in practice. In this case, we need to let the candidate entities which are more likely to generate the gold fact get ahead of the rest. Previous methods rarely considered the information contained in the question pattern or candidate entities when doing subject selection. So we conduct relation detection, which identifies the KB predicate that a question utterance refers to, to reorder the entity linking results. Enhanced entity linking provides a strong support to generate high-quality candidate subjects.

In SimpleQuestions benchmark, a question, such as "which release was desperado the release track off of?", asks a direct relation of an entity called "desperado". While there are dozens of entities named "desperado" in Freebase which linked to different types and predicates[5]. Previous work either conduct relation inference firstly (Dai et al., 2016), or conduct entity linking firstly (Yih et al., 2015) may lead to no recall problem. In our framework, we propose to do joint fact selection to alleviate the problem. We leverage entities' name information and type information to represent entities' different aspects. As for predicates, we use the unique relation name and dispersive words information. In order to represent the sentence utterance properly, char-level and word-level encodings both are incorporated. The experimental results demonstrate the effectiveness of the proposed approach.

To sum up, our main contributions are: (1) We propose to conduct pattern extraction and entity linking, and put forward pattern revising procedure to mitigate the error propagation problem. (2) In order to learn to rank candidate subject-predicate pairs to enable the relevant facts retrieval given a question, we propose to do joint fact selection enhanced by relation detection. Multi-level encodings and multi-dimension information are leveraged to strengthen the whole procedure. (3) Our approach sets a new record in Simple KB-QA task, outperforming the current state-of-the-art by an absolute large margin.

[4]The question pattern of a question is produced by substituting the possible entity mention with a special token "#head_entity#" in original question. For example, for the question "*what position does carlos gomez play*", the correct entity mention is "*carlos gomez*", and the pattern should be "*what position does #head_entity# play*".

[5]Some predicate sets linked to these entities which has the same name "desperado", contains the gold predicate "music/release_track/recording" in common.

2 Related Work

The research of KB-QA has evolved from earlier domain-specific QA (Zelle and Mooney, 1996; Tang et al., 2001) to open-domain QA based on large-scale KB. There are two mainstream research directions for the KB-QA task. One of the promising approaches is semantic parsing (Cai and Yates, 2013; Yih et al., 2015; Yih et al., 2016; Reddy et al., 2016), which uses logic language CCG (zettlemoyer and Collins, 2009; zettlemoyer and Collins, 2012; Kwiatkowski et al., 2013; Reddy et al., 2014; Choi et al., 2015) or DCS (Berant et al., 2013) to map a question to its formal logical form to query on a KB. The other category exploit vector space embedding approaches (Bordes et al., 2014a; Bordes et al., 2014b; Bordes et al., 2015; Dong et al., 2015; Xu et al., 2016a; Xu et al., 2016b; Hao et al., 2017; Lukovnikov et al., 2017) to measure the semantic similarity between question utterance and candidate resources in the background KB, such that the correct supporting evidence will be the nearest neighbor of the question utterance in the learned vector space.

Instead of measuring the similarity between a question and an evidence triple with a single model, Yih et al. (2015) adopt a multi-stage approach. In each stage, one element of the triple is compared with the question utterance to produce a partial similarity score by a dedicated model. Then these partial scores are combined to generate the overall measurement condition. Dong et al. (2015) use three columns of CNNs to represent questions respectively when dealing with different answer aspects. Xu et al. (2016a; 2016b) incorporate Wikipedia free text to address KB-QA problems, in which they use multi-channel CNNs to extract relations. Dai et al. (2016) employ a conditional factoid factorization by inferring the target relation first and then the target subject associated with the candidate relations. Yin et al. (2016b) stack an attentive maxpooling above convolution layer to model the match of candidate predicates and questions. Lukovnikov et al. (2017) train a neural network, which contains a nested word/character-level question encoder, for answering simple questions in an end-to-end manner. In this work, we propose a two-stage framework, which exploit multi-dimension information leveraging multi-level encodings to compute semantic similarity, to tackle the problem of simple KB-QA.

Relation detection for KB-QA starts with feature-rich approaches (Yao and Van Durme, 2014) towards usages of vector space embedding models (Yih et al., 2016; Xu et al., 2016a; Golub and He, 2016; Yu et al., 2017). Actually many of the mentioned researches could support large relation dictionary, fitting the goal of open-domain question answering. Some work (Yin et al., 2016b; Yu et al., 2017) split relations into word sequences for single-relation detection. Golub and He (2016) propose a generative model for relation detection which predicts relation in a character-level sequence-to-sequence manner. Many researches have proved that relation detection benefit for KB-QA task. In our approach, we use relation detection enhance entity linking results, generating a more focused candidate subject set.

Conducting fact selection in KB-QA is inspired by work on answer selection (Yu et al., 2014; Yin et al., 2016b) which looks for correct answers from some candidates given a question. In machine comprehension (Yin et al., 2016a) scenario, the answer candidates are raw text, not structured information as facts in KB are. In our joint fact selection procedure, we leverage multi-level encodings of question utterance and subject-predicate pairs to measure their similarity in order to get the retrieved answer.

3 Overview of Simple Question Answering

The goal of simple KB-QA task could be formulated as follows. Given a natural language question q, the system returns the answer $(\hat{s}, \hat{p})$. The architecture of our proposed framework is shown in Figure 1, which illustrates the basic flow of our approach. In our approach, we conduct pattern-revising enhanced pattern extraction and entity linking to identify each question's corresponding pattern and mention, and generate candidate subjects together with their corresponding candidate facts from Freebase. Then a multi-level encoding based deep neural network is employed to select fact from the generated candidate facts, i.e. the joint fact selection procedure, in which relation detection is leveraged to reorder the entity linking results. The similarity score between the question utterance and each corresponding candidate fact is calculated, and the candidate fact with the highest score will be selected as the final predicted answer to the question.

Freebase (Bollacker et al., 2008) is utilized as our background KB, which has more than 3 billion facts,

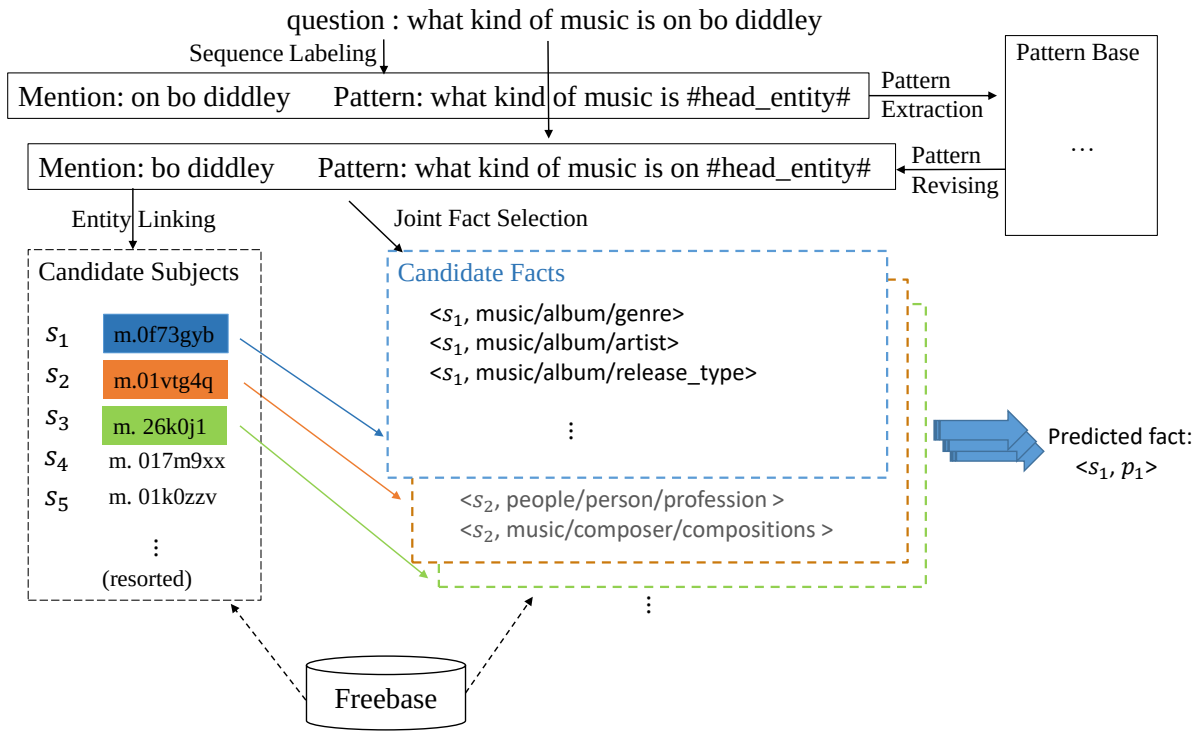


Figure 1: The overview of our proposed simple question answering approach, which is tailored via first stage of *pattern extraction and entity linking* enhanced by pattern revising, and second stage of *joint fact selection* which learns to rank candidate subject-predicate pairs to enable the relevant facts retrieval given a question.

and is used as the supporting KB for many QA tasks. We use SimpleQuestions dataset as our benchmark. The SimpleQuestions dataset consists of more than one hundred thousand simple questions which can be answered with a single fact in Freebase. We will give more detail in Section 6.1.

4 Pattern Extraction and Entity Linking

Given a question, two problems should be carefully tackled: (1) identifying the mention span in the question that refers to an entity in KB, (2) and generating candidate entities in KB that the question should refers to. So we conduct pattern extraction and entity linking procedure, which is enhanced by pattern revising. This is the first stage of our approach.

Given a question, the first stage should provide a set of top-N subject candidates. Generally, we should use all N-grams to retrieve the candidate subjects in KB in order to guarantee the recall rate. In this case, the candidate space is restricted to entities whose name or alias matches an n-gram of the question, as in (Bordes et al., 2015). However, the candidate pool is too noisy due to the large number of non-subject-mention n-grams. So we try to find a method to split the question utterance into two parts: mention span and question pattern. Mention span should be relatively rare words that refers to an entity in KB, while question pattern usually consists of common words that reflect the relation the question utterance refers to.

Inspired by previous work (Dai et al., 2016; Yin et al., 2016b), the problem could be treated as a sequential labeling task in which we aim to label the relatively rare continuous words of mention span to one kind, and label the relatively common words which belong to question pattern to another kind. This is the question pattern extraction step. The key idea is to train a model to predict the continuous word of text mention span which may refers to the topic entity, similar as Name Entity Recognition (NER). In the training set of SimpleQuestions, the topic entity of each question is labeled. We map the gold entity back to the text to label the text mention span for each question, then train a BiLSTM-CRF model to detect

the subject entity mention span in question utterance. After substituting each subject entity mention span by "#head_entity#", we get the question pattern for each question. Each question is split into (mention, pattern) pair. The extracted patterns are taken in the pattern base.

4.1 Pattern Revising

Due to the performance of sequential labeling model, there are a certain portion of bad cases in our (mention, pattern) pairs result. The more common kind of bad patterns are the ones that have obvious mistakes together with the wrong labeled mention span. The more extreme error type is that, for instance, mention is the question utterance and pattern is null. Based on the principle that question patterns which consist of common words should occur in dataset more than one time and the more times a pattern appears the more likely it is to be correct, we propose a novel pattern revising procedure to correct wrong patterns in order to increase the proportion of "good patterns", following the steps illustrated in Algorithm 1[6]. We keep all the extracted pattern which appears more than one time as the pattern base $P_{n'}$, and all the original questions as the sentence utterance base Q_n. For each question in the sentence utterance base, we reappraise its extracted pattern and try to find whether we can find a more suitable pattern or not.

Algorithm 1 Pattern Revising

Input:
The set of extracted pattern base, $P_{n'}$;
The set of original question utterance base, Q_n;
Output:
The set of the revised Question-Pattern base, Q_P;

for each sentence utterance $q \in Q_n$ **do**
  for each candidate pattern $p \in P_{n'}$ **do**
    split p according to "#head_entity#";
    if all split results are contained in q and p, q satisfy consistency policy **then**
      add p to candidate pattern set q_{cp}
    end if
  end for
  if $|q_{cp}| == 1$ **then**
    set $p \in q_{cp}$ as pattern q_p
  else
    calculate word match count for each $p \in q_{cp}$
    select p with max word match count as pattern q_p
  end if
  add (q, q_p) to Q_P
end for

After getting all the revised pattern, we extract their corresponding mention span for each question utterance. The mention span is used to link entities in KB. Based on the mention span, we use each word of it to retrieve subject entities whose names contain this word, deriving the longest consecutive common subsequence for each candidate subject entity. Then we rank the candidate subject entities according to each entity's longest consecutive common subsequence, similarly as Yin et al. (2016b) did. Top-M ranked entities are kept for each question, denoted as C'_e, and their scores for ranking are denoted as $s_{linker}(e, q)$.

[6]The consistency policy means that the split results combine the words which are in the position of "#head_entity#" should equal to the original question utterance. It is not allowed to have redundant words.

5 Joint Fact Selection

5.1 Relation Detection enhanced Subject Candidates

For one question, there may be a large set of candidate entities. It's unrealistic to use all of them to generate candidate facts in practice. We need to let the candidate entities which are more likely to generate the gold fact get ahead of the rest. So we conduct relation detection to resort the subject candidates. We train a relation detection network which maps each question utterance to their most possible predicate. Predicates are represented from different granularity: predicate-level and word-level. Predicate-level is represented by the predicate name as a single token, while word-level is represented by splitting predicate name into word sequence. Word-level encoding serves as a supplement to predicate-level encoding, mitigating data sparsity[7].

We use question utterance as input for a relation detector to score all predicates connected to entities in C'_e, for each question. The similarity score $s_{rel}(r, q)$ is computed using cosine distance between the LSTM-based encoding of question utterance and the predicate embedding r_{pred}. We extract candidate subjects' notable type on behalf of candidate subjects' type information. Then word-level LSTM-based encoder is leveraged to get the type information embedding:

$$r_{type} = ENC_{type}(w_1, w_2, ...) \tag{2}$$

We represent predicate from different granularity: predicate-level and word-level, jointly considering the corresponding candidate subject's type information:

$$r_{pred} = [e_{pred-lev}; ENC_{word-lev}(w_1, w_2, ...); r_{type}] \tag{3}$$

For each entity $e \in C'_e$ and its associated relation R_e, we generate the final rank score:

$$s_{rank}(e, q) = s_{linker}(e, q) + \alpha(\max_{r \in R_e} s_{rel}(r, q) + \beta) \tag{4}$$

where α and β are constant hyperparameters. Finally top-N ($N < M$) ranked entities are kept for each question to generate fact pool, denoted as $\mathcal{C}_s$.

For each candidate subject entity s in $\mathcal{C}_s$ and its associated predicate set P_s, we can generate candidate fact pairs (s, p) ($p \in P_s$). Then for each question, we have all candidate fact pair pool $\mathcal{C}_{(s,p)}$. From a global perspective, the last is fact selection problem, also a matching task. Given a question q $(mention, pattern)$ and sets of candidate subject entities and predicates, $\mathcal{C}_s = \{s_1, ..., s_n\}$ and $\mathcal{C}_p = \{p_1, ..., p_m\}$ respectively, our fact selection model should jointly returns the subject and predicate that matches the question best. The calculation process is shown in the Figure 2.

5.2 Subject Matching Network

Entity names are used to calculate similarity score with the mention span detected in question utterance. Since the coverage of word embeddings is limited, many words in entity names are out-of-vocabulary (OOV) words, so we leverage character-level RNN-based encoding (the string is seen as a sequence of characters) for mention span and candidate subjects' names, mitigating the high prevalence of OOV problem.

As illustrated in Figure 2, we first look up a character embedding matrix E_c, which is randomly initialized and updated during the training process, to get the character embeddings for the entity name and the mention span. Then the embeddings are fed into an unidirectional LSTM networks, where the last hidden state is taken as the encoding of current sequences. Then we compute cosine similarity $t_{subj}(s, q)$ between the encodings of the entity name and the mention span in string surface-form.

[7]There are a certain portion of predicates' names not appearing in training data.

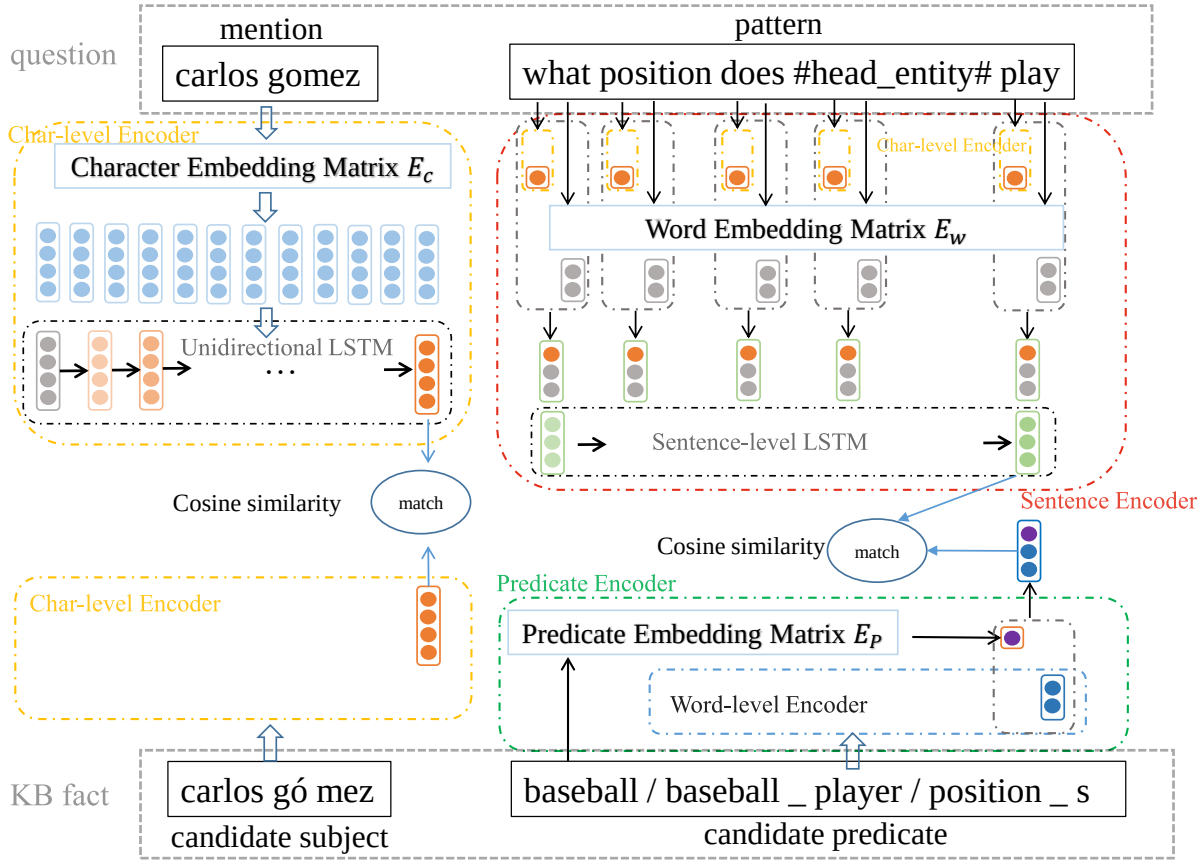


Figure 2: The overview of the proposed fact selection procedure. The mention span in the question utterance and the name of candidate subject are encoded using char-level encoder, then cosine distance is calculated to represent their similarity score. The sentence encoder for question pattern is in a nested structure, both leverage each word's char-level encoding and word embedding. The predicate encoder is similar as in 5.1, utilizing predicate-level encoding and word-level encoding.

5.3 Predicate Matching Network

Predicate matching network tries to match the patterns extracted from question utterance with the corresponding candidate predicates.

[**Question pattern encoder**] The question pattern encoder maps the question pattern to its corresponding predicate in the vector space. The process is done by using a LSTM based encoder network.

As for word representation in pattern utterance, we exploit a nested hierarchical encoding manner, leveraging both character-level and word-level information of each word in question pattern. Char-level encoder is first employed to get the representation for each word, which is a microcosmic operation. Then the microcosmic encoding on character level is concatenated with the macroscopical word embedding, which is randomly initialized and updated during the training process, to get the representation of each word:

$$m_i = [e_{w_i}; ENC_{char-lev}(c_1, ..., c_t)] \tag{5}$$

This is the way how we get the nested word embeddings. For sentence level encoding, we use word-level LSTM encoder network to get question pattern representation:

$$r_{pattern} = ENC_{pattern}(m_1, ..., m_n) \tag{6}$$

[**Predicate encoder**] In the matching procedure, we incorporate the candidate predicate's hierarchical encoding information to measure similarity with question pattern. This is similar with what we did in Section 5.1.

We represent predicate from different granularity: predicate-level and word-level. In detail, the micro-cosmic encoding on word-level is done by using a word-level LSTM based encoder network. Then the encoding is concatenated with its upper macro predicate-level encoding:

$$r_{pred} = [e_{pred-lev}; ENC_{word-lev}(w_1, w_2, ...)] \tag{7}$$

Both predicate-level and word-level encodings are useful to represent the predicate properly. We compute cosine similarity between r_{pred} and $r_{pattern}$ to get the matching score $t_{pred}((s, p), q)$.

5.4 Training

The overall ranking score of a fact triple t is $S(t, q) = t_{subj}(s, q) + t_{pred}((s, p), q)$. Our objective is to minimize the ranking loss:

$$L_{q,t,t'} = [\gamma + S(t', q) - S(t, q)]_+ \tag{8}$$

where γ is a hyper-parameter, t' is a negative triple. Our fact pool consists of all facts whose subject entity is in the top-N entity candidates. For train, we sample 200 negative facts for each ground truth fact; and keep all candidate facts for valid and test.

6 Experiments

6.1 Dataset and Evaluation

SimpleQuestions benchmark is a typical Simple QA task, which provides a set of simple questions that can be answered by a single Freebase triple[8]. This dataset is split into three parts: train (75,910 instances), valid (10,845 instances) and test (21,687 instances) sets. The benchmark also provides two subsets of Freebase: FB2M (2,150,604 entities, 6,701 predicates, 14,180,937 atomic fact triples), FB5M (4,904,397 entities, 7,523 predicates, 22,441,880 atomic fact triples).

The evaluation metric is accuracy. Only a fact that matches the ground truth answer in both subject and predicate is counted as correct.

$$accuracy = \frac{\sum_{\mathrm{i}=1}^{N} 1_{[(\hat{s}_i, \hat{p}_i)=(s_i, p_i)]}}{N} \tag{9}$$

6.2 Experimental Settings

In our approach, three types of embedding matrices are utilized: character embedding matrix, word embedding matrix, and predicate-level KB recource embedding matrix (i.e., the predicate embedding matrix). All of them are randomly initialized and updated during training process. The dimension of character embedding and predicate embedding is set to 100, and 200 for word embedding matrix. The hidden layer size of LSTMs is consistent to their corresponding dimension of the sequences to be processed. The candidate subjects number N is 20. We employ Adagrad (Duchi et al., 2011) to minimize the hinge training loss, where the margin γ is set to 1.0, and negative sample number is 200. All hyperparameters are determined according to the performance on the validation set.

6.3 Results

To demonstrate the effectiveness of the proposed approach, we compare our method with state-of-the-art systems.

Bordes et al. (2015) proposed an implementation of memory network for SimpleQuestions task. Golub and He (2016) tackled Simple QA using a character-level attention-based encoder-decoder LSTM model. A RNN-based approach was proposed by Dai et al. (2016), which utilized a focused pruning method called CFO. Yin et al. (2016b) leveraged an attentive CNN model to deal with the problem of Simple QA, and achieved best experimental result in previous state-of-the-art systems. Lukovnikov et al. (2017) adopted an end-to-end neural network, achieving a relatively good result.

[8]The object entity in the fact triple is the answer to the question utterance.

Approach	Setting	*Accuracy*
Dai et al. (2016)	N-Gram+	62.6*
Bordes et al. (2015)	end-to-end	62.7
Yin et al. (2016b)	passive linker	68.3
Golub and He (2016)	end-to-end	70.9
Lukovnikov et al. (2017)	end-to-end	71.2
Dai et al. (2016)	focused pruning	75.7*
Yin et al. (2016b)	attentive pooling	76.4
Our approach	**pattern-revising + joint fact selection**	**80.2**

Table 1: Comparison with state-of-the-art systems on SimpleQuestions benchmark. When marked with (*), accuracy in FB5M setting is given, otherwise FB2M setting results are shown.

The experimental results demonstrate that our framework sets new record in this task, outperforming the the state-of-the-art by an absolute large margin, achieving a competitive result, as shown in Table 1. From the results, we observe that our framework outperforms the latest end-to-end system record of (Lukovnikov et al., 2017) by 9.0 points, higher than previous best reported system result in (Yin et al., 2016b) by 3.8 points. While our framework exploits a quite simple and universal model which can be easily applied to other simple KB-QA tasks, not using attention model or other complex settings.

6.4 Framework Analysis

In this part, we further discuss the impacts of the augmented components which can be removed in our approach. Table 2 indicates the effectiveness of different parts in the framework.

Approach	Setting	*Accuracy*
our approach	fully integrated	80.2
our approach	w/o pattern revising	77.8
our approach	w/o relation detection	77.5
our approach	w/o both	74.8

Table 2: The ablation results of our approach.

The accuracy of original system is 74.8%. When augmented with pattern revising procedure, the accuracy of our framework increase 2.7 points, indicating that our pattern revising algorithm does lift the quality of generated patterns. The accuracy increase to 77.8 when we use relation detection to resort the results of subject candidates, showing that incorporating predicate and entity type information for reranking candidate entities can promote the quality of the recalled candidates. Either component is important for our framework. The performance decreases obviously when we remove any of them.

6.5 Error Analysis

We perform error analysis on our approach by randomly sampling 100 wrongly answered questions. The errors can be categorized into 4 types.

The most common kind of error (56%) is the wrongly picked subject entity or predicate. Due to the performance of pattern extraction and entity linking procedure, the gold entity may not appear in our candidate subjects. In this case our joint fact selection process can not select the gold subject entity. The probability of selecting wrong subjects is much higher than selecting a wrong predicate. Another kind of error case (22%) could be concluded into the "long drug name" error[9]. Some drug entities' names are quite long (more than 15 words), while mention span in question utterance may be much shorter (less than 4 words). In this case our mention-subject matching network in the joint fact selection process is failed. The third common error (13%) can be classified as "KB error". There are some entities with the

[9]For example, in the question "what is an active ingredient in childrens earache relief", the labeled gold subject "m.0jxn044" has a name longer than 15 words.

same name and the same type, indeed. The KB or the question can't support more evidence to distinguish them. The last type of error (9%) is label error. In the dataset there is a small portion of questions which are repeated, but the labeled subject or predicate is different.

7 Conclusion

In this work, we propose a pattern revising procedure to mitigate the error propagation problem in pattern extraction and entity linking process. In order to learn to rank candidate subject-predicate pairs to enable the relevant facts retrieval given a question, we propose to do joint fact selection enhanced by relation detection. Multi-level encodings and multi-dimension information are leveraged to strengthen the whole procedure. The experimental results demonstrate the effectiveness of the proposed approach.

Acknowledgements

The research work is supported by the National Key Research and Development Program of China under Grant No.2017YFB1002101, the Natural Science Foundation of China (No.61533018 and No.61702512), and the independent research project of National Laboratory of Pattern Recognition. This work is also supported in part by Beijing Unisound Information Technology Co., Ltd.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. *Semantic Parsing on Freebase from Question-Answer Pairs*. EMNLP 2013, volume 2.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. *Freebase: a collaboratively created graph database for structuring human knowledge*. Proceedings of the 2008 ACM SIGMOD international conference on Management of data:1247–1250.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. *Semantic Parsing on Freebase from Question-Answer Pairs*. arXiv preprint arXiv:1406.3676.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. *Large-scale simple question answering with memory networks*. arXiv preprint arXiv:1506.02075.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. *Open question answering with weakly supervised embedding models*. Joint European Conference on Machine Learning and Knowledge Discovery in Databases:165–180. Springer.

Qingqing Cai, Alexander Yates. 2013. *Large-scale Semantic Parsing via Schema Matching and Lexicon Extension*. ACL 2013:423–433.

Eunsol Choi, Tom Kwiatkowski, and Luke Zettlemoyer. 2015. *Scalable Semantic Parsing with Partial Ontologies*. Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers):1311–1320.

Zihang Dai, Lei Li, and Wei Xu. 2016. *CFO: Conditional Focused Neural Question Answering with Large-scale Knowledge Bases*. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers):800–810.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. *Question Answering over Freebase with Multi-Column Convolutional Neural Networks*. Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics:260–269.

Join Duchi, Elad Hazan, and Yoram Singer. 2011. *Adaptive subgradient methods for online learning and stochastic optimization*. JMLR, 12:2121–2159.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. *An End-to-End Model for Question Answering over Knowledge Base with Cross-Attention Combining Global Knowledge*. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers):221–231.

David Golub, and Xiaodong He. 2016. *Character-Level Question Answering with Attention*. Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing:1598–1607.

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

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. *Scaling semantic parsers with on-the-fly ontology matching*. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing.

Denis Lukovnikov, Asja Fischer, and Jens Lehmann. 2017. *Neural Network-based Question Answering over Knowledge Graphs on Word and Character Level*. International Conference on World Wide Web 2017:1211–1220.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. *Large-scale Semantic Parsing without Question-Answer Pairs*. Transactions of the Association of Computational Linguistics, volume 2:377–392.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. *Transforming dependency structures to logical forms for semantic parsing*. Transactions of the Association for Computational Linguistics, volume 4:127–140.

Lappoon R. Tang, and Raymond J. Mooney. 2001. *Using Multiple Clause Constructors in Inductive Logic Programming for Semantic Parsing*. European Conference on Machine Learning 2001:466-477.

Christina Unger, André Freitas, and Philipp Cimiano. 2014. *An introduction to question answering over linked data*. Reasoning Web International Summer School:100-140. Springer.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. *Hybrid Question Answering over Knowledge Base and Free Text*. Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers:2397–2407.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. *Question Answering on Freebase via Relation Extraction and Textual Evidence*. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics:2326–2336.

Xuchen Yao, and Benjamin Van Durme. 2014. *Information Extraction over Structured Data: Question Answering with Freebase*. Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics:956–966.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. *Semantic Parsing for Single-Relation Question Answering*. Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics:643–648.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. *Semantic parsing via staged query graph generation: Question answering with knowledge base*. Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. *The value of semantic parse labeling for knowledge base question answering*. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics.

Wenpeng Yin, Sebastian Ebert, and Hinrich Schütze. 2016. *Attention-based convolutional neural network for machine comprehension*. Proceedings of NAACL Human-Computer QA Workshop.

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. *Simple Question Answering by Attentive Convolutional Neural Network*. Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers:1746–1756.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. *Improved Neural Relation Detection for Knowledge Base Question Answering*. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers):571–581.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. *Deep learning for answer sentence selection*. Proceedings of 2014 ICLR Workshop.

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

Luke S Zettlemoyer, and Michael Collins. 2009. *Learning context-dependent mappings from sentences to logical form*. Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP:976–984.

Luke S Zettlemoyer, and Michael Collins. 2012. *Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars*. arXiv preprint arXiv:1207.1420.

Integrating Question Classification and Deep Learning for improved Answer Selection

Harish Tayyar Madabushi
School of Computer Science,
University of Birmingham,
United Kingdom.
H.T.Madabushi@gmail.com

Mark Lee
School of Computer Science,
University of Birmingham,
United Kingdom.
M.G.Lee@cs.bham.ac.uk

John Barnden
School of Computer Science,
University of Birmingham,
United Kingdom.
J.A.Barnden@cs.bham.ac.uk

Abstract

We present a system for Answer Selection that integrates fine-grained Question Classification with a Deep Learning model designed for Answer Selection. We detail the necessary changes to the Question Classification taxonomy and system, the creation of a new Entity Identification system and methods of *highlighting* entities to achieve this objective. Our experiments show that Question Classes are a strong signal to Deep Learning models for Answer Selection, and enable us to outperform the current state of the art in all variations of our experiments except one. In the best configuration, our MRR and MAP scores outperform the current state of the art by between 3 and 5 points on both versions of the TREC Answer Selection test set, a standard dataset for this task.

1 Introduction and Motivation

Question Answering (QA) is the task of automatically generating answers to questions posed in natural language. The task has received significant attention from researchers over several decades with a renewed interest in recent times. Current interest has been partly due to significant improvements in Natural Language Processing, and partly due to the increase in demand for such systems, amongst both the general populace, and corporate entities.

An important subtask of QA is Question Classification (QC), which deals with the classification of questions based on the expected class of the answer. For example, the question "How much does the Capitol Dome weigh?" could be classified into the class "Numeric, Weight", while the question "Name the actress in the movie Titanic" could be classified into the class "Human, Individual". While there exist QA systems that do not make use of QC, the addition of QC to a QA system has been shown to improve its accuracy (Hovy et al., 2001). The specific classification that a QC system uses is called its taxonomy, and taxonomies vary quite widely in both specificity and form.

Intuitively, QC improves QA by reducing the search space of potential answers, thus making the discovery of answers more efficient and accurate. While research into QC is fairly mature, fine grained QC is not always used for QA. For example, Tsai et al. (2015) use very simple rules based partly on question wh-words (i.e. "what", "where", "who", etc.), resulting in 13 question classes. Not only is this number low compared to the 50 classes defined by (Li and Roth, 2002) or the over 120 used by (Hermjakob, 2001), the rules themselves are fairly weak. As an example, Tsai et al. (2015) classify all questions containing "name" under the class "Name" ("Name the country most famous for cheese.")

Most QA systems consist primarily of three components: **a)** a question analysis component, **b)** an Information Extraction (IE) component that extracts a set of candidate sentences, and **c)** an answer extraction component that prunes this set of sentences in order to extract the answer. QC is performed in the first component. Its results are sometimes useful in the IE component, but generally most useful in answer extraction. The other important aspect of the answer extraction component is the analysis of linguistic features. Together, these two elements can be used to prune a set of sentences, some of which might contain the answer to a given question.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3283–3294
Santa Fe, New Mexico, USA, August 20-26, 2018.

This task of selecting, from a list of sentences produced by an IE component, a subset A (where $|A| \geqslant 0$) which contains the answer to a given question is called Answer Selection (AS). For example, given the question "Where is the group Wiggles from?", and two possible sentences (called *candidate sentences*): "the Wiggles are four effervescent performers from the Sydney area: Anthony Field, Murray Cook, Jeff Fatt and Greg Page", and "six of the Wiggles' videos have reached multiplatinum status in Australia", the task would require one to return the first (*positive*) candidate and not the second (*negative*) candidate. We note that AS leaves the task of extracting the Answer from a positive candidate to a downstream task.

Methods of AS rely on establishing some form of relation between the question and each of the answer candidates, such as bag-of-words, tree edit models (Heilman and Smith, 2010), semantic distances based on word embeddings (Wang and Ittycheriah, 2015), or deep learning methods such as Convolutional Neural Networks (Rao et al., 2016). To the best of our knowledge however, this task has not been attempted with extensive use of fine grained QC.

2 Related Work

A lot of the work into using QC for QA took place before the resurgence of Machine Learning. For example, Kwok et al. (2001) introduce a QA system "MULDER", that makes use of wh-phrases, which they define as the interrogative word followed by the words associated with it. Hermjakob (2001) used an extensive QC system consisting of 115 elementary question classes in their work on QA.

2.1 Question Taxonomy and Classification

The specific system of classes used by a QC system is known as a taxonomy, and while several taxonomies are available, we pick that proposed by Li and Roth (2002), for two reasons: **a)** This is one of the most widely used taxonomies, possibly because of the large training set that Li and Roth (2002) provide, **b)** while it has been pointed out that this taxonomy might not have the widest coverage (Mishra and Jain, 2016), we show below that it is most suited for domain independent QA.

This taxonomy originally consisted of fifty fine classes divided amongst six coarse classes. Table 1 provides a complete list of these classes along with the changes we make (described in Section 4.1).

While our work on QC is an extension of the work by Tayyar Madabushi and Lee (2016), a rule-based system that achieved an accuracy of 97.2%, other work on the same taxonomy has involved the use of Linear SVMs by Van-Tu and Anh-Cuong (2016) and Pota et al. (2016) which achieved accuracies of 91.6% and 89.6% respectively. Work using Convolutional Neural Networks (Kim, 2014) and Skip-Thought Vectors (Kiros et al., 2015) have not focused on fine-grained classification.

2.2 Answer Selection

Answer Selection became popular as a task after being proposed by Punyakanok et al. (2004), who modified the TREC QA task to that of AS. This modification does not simplify the task, as extracting relevant sentences is not only as hard as extracting the actual answer, but users often find it more useful to see answers to their questions in their original context (Wang et al., 2007).

Wang et al. (2007) follow a similar approach to Punyakanok et al. (2004), while providing a training set extracted from TREC 8-12 datasets and setting aside the TREC 13 dataset for development (84 questions) and testing (100 questions). AS has since become the standard in measuring the accuracy of Question Answering systems[1] and the dataset has since diverged into two versions: The "Clean Version" (Wang and Ittycheriah, 2015), which has been cleaned to remove questions with no candidate answer sentences and those that have no negative candidate sentences from the development and the test sets; the non-cleaned version called the "Raw Version". Rao et al. (2016) have shown that results from the two datasets are not comparable. Results for this task are reported using two measures, common in Information Retrieval and Question Answering Research: Mean Average Precision (MAP) and Mean

[1] http://www.aclweb.org/aclwiki/index.php?title=Question_Answering_(State_of_the_art)

Reciprocal Rank (MRR). These results are attained using the standard program *trec_eval*, provided by TREC.

Recent work on Answer Selection has depended heavily on word embeddings with the state of the art work on the Raw version by Rao et al. (2016) who rank candidate sentences using a Multi-Perspective Convolutional Neural Network and that on the Clean version by Shen et al. (2017) who use a novel method of sentence pair modelling. Previous work on the same dataset used similar models including that by dos Santos et al. (2016) that made use of Attentive Pooling CNNs and that by Bian et al. (2017) who use a Compare-Aggregate framework. A complete list of work on the dataset is available on the ACL wiki on Question Answering.

3 System Overview and Contribution

In working towards a method of integrating QC into AS, we first redefine the taxonomy provided by Li and Roth (2002) to better suit entity identification, before then modifying the Question Classification system developed by Tayyar Madabushi and Lee (2016) to match this modified taxonomy. We then create an entity identification method to extract entities belonging to those classes in our taxonomy. Finally, we use different methods of "highlighting" entities, so this information can be passed on to *any* model that uses word embeddings. We use the model developed by Rao et al. (2016), which performs AS, to test our method.

In addition to showing the significant impact that Question Classification has on Answer Selection, we make several datasets available so others might exploit QC in Question Answering tasks including a Question Classification API that reflects the modified taxonomy[2].

4 Question Classification

Our experiments with using the taxonomy proposed by Li and Roth (2002) showed the need for changes to allow the classification system to lend itself more easily to Entity Identification and AS. For one, we found that some categorisations would make entity identification harder. For example, the question "What's the world's longest suspension bridge?" is categorised under "Location" while we believe that it is more appropriate to consider a bridge an entity. We base this on the hierarchical classification provided by WordNet (Miller, 1995), an extensive human crafted hierarchical dictionary that has been used as a standard for several different tasks.

Similarly, we disagree with the prioritisation of the classes provided. Prioritisation is important as this particular taxonomy does not allow a question to be a part of two classes. As an example, the question "What country did the ancient Romans refer to as Hibernia?" can be classified as either belonging to the class "Location:Country" or "Entity:termeq" (Equivalent Term). While Li and Roth (2002) categorise this question under the first, we categorise it under the latter, because the question is not about where something is or happens. We also believe that this choice makes it easier for a downstream QA system.

We also found the need for a new class that constitutes either "Human:Individuals" *or* "Human:Groups" (such as companies, teams and universities). This specific requirement is a direct result of the restriction that a question must be classified without prior knowledge of the answer. For example, the question "Who won the Nobel Peace Price in 2012?" is impossible to classify without knowing if the answer was an organisation (as it was in 2012) or an individual (as in 2016), even if we were to ignore the possibility of multiple individuals (as in 2014). To get around this we introduce the class "Human:IndividualOrGroup". We retain the classes "Human:Individuals" and "Human:Groups" for instances where the distinction is clear.

Finally, we find that certain types of entities within certain classes are much more frequent than others in that class. While this could be because of the specific method we use for Entity Identification (Section 5), we create separate classes for these types of entities so as to avoid noise in our AS feature generation. We also expand the class "Location:State" to include the provinces of Canada and the counties of the U.K. We list the taxonomy thus modified in Table 1.

[2] www.harishmadabushi.com/research/questionclassification/

Coarse	Fine
ABBR	abbreviation*, expansion*
DESC	definition*, description*, manner*, reason*
ENTY	animal, body, colour, creation, currency, disease, event, food, instrument, language, letter*, other*, plant, product, religion, sport, substance*, symbol*, technique, term*, vehicle*, word*, **movie***, **book***, **extraterrestrial**
HUM	description*, group, individual, title, **individualOrGroup***
LOC	city, country, mountain, other, **state**
NUM	code, count, date, distance, money, order, other*, percent, percent, period, speed, temperature, size, weight, **year**, **volume (Size)**, **volume (Liquid)**, **time**, **numeric range***

Table 1: Question Taxonomy introduced by Li and Roth (2002), with our modifications in bold (Section 4.1) and those classes not used in AS starred* (Section 8)

4.1 Modifications to Question Classification

The QC system used by us is an extension of that by Tayyar Madabushi and Lee (2016). It primarily involves: **a)** extracting a Question's Syntactic Map (a structure they define for holding certain types of syntactic information), **b)** identifying the headword of the noun phrase in the question, while handling Entity Identification and phrase detection, and **c)** using rules to map words at different positions in the Syntactic Map to question classes using a hyrarchical structure.

Their system classifies questions as follows: Consider the question "What is the name of the actress from England in the movie 'The Titanic'?". The system identifies its Question Class by analysing the question's parse tree to generate the Question's Syntactic Map, which enables the identification of the headword "actress" using, what they call, "prepositional rolling". This process provides us with the question's wh-word ("What"), the auxiliary verb ("is"), and the headword ("actress"). This information is used by the system to check for the existence of a rule that classifies this question. Such a rule is found by matching the noun "actress" to the rule: 'occupation.n.01' and its hyponyms in this section of the question when the wh-word is 'what' indicate that the question class is hum:ind.

These rules are manually defined using sets of WordNet synsets they call Types. Types are defined by manually picking specific synsets within WordNet and associating them and all their hyponyms to a particular Question Class based on where in a question they appear. In the previous example, the relevant Type is the word occupation and all hyponyms of the synset 'occupation.n.01'. Similarly, the synsets 'people.n.01', 'organization.n.01', 'university.n.01', 'company.n.04', 'socialgroup.n.01', and all of their hyponyms are assigned to the Question Class "Human Group".

For a detailed description, we direct the reader to the original work. However, we describe herein some elements which we have modified, a necessity given our changes to the taxonomy.

4.2 Word Sense Disambiguation and Rule Extensions

A primary difficulty in identifying the specific rule to use once the correct head of the question has been identified arises due to the polysemous nature of some words. For example, the question "What rank did you achieve in the test?" and the question "What rank did she achieve in the military?" both have the same headword "rank" but differ in the meaning of that word (position in ordering versus military status such as captain). The question class assigned to each of these question must also change based on these meanings (Number:order versus Human:title).

As described by Tayyar Madabushi and Lee (2016), words useful in identifying the question class are often nouns, as in the case of the question "What is the name of the *actress* in the Titanic?". However, such words need not always be a noun. In the case of the question "How much does the President get paid?", for example, it is the adverb "much" which allows us to infer that the expected answer is a number and additionally, the word "paid" allows us to infer that the number represents money hence resulting in the question class "number:money" as opposed to the question class "number:weight" as in the case of "How much does the Big Ben weigh?"

Rules defined by the QC system map sub-trees in WordNet to specific question classes. We make changes to the rules to align the classification of questions with the modifications we make to the taxonomy (Table 1) and add further rules where possible to cover a larger section of WordNet. Additionally, there are instances wherein the system makes use of certain heuristics to find the appropriate rule to use, as in the case of questions starting with "How much ..." which sometimes leads to classification errors. To mitigate this problem, we modify the system to return a possible second class when there is ambiguity.

5 Named Entity Recognition

Given that our objective is to "highlight" all entities in candidate answers that belong to the class assigned to a particular question, we require a method of Named Entity Recognition (NER) at the same granularity as our taxonomy. Unsurprisingly, there is no off the shelf NER system that identifies entities with the exact granularity and classes that we classify questions into. To get around this problem we start by relating entities in text to Wikipedia titles and subsequently mapping those titles to our classes. This process of mapping entities in text to Wikipedia titles is called Wikification. The hierarchical tree-structure provided by Wikipedia helps in mapping a large number of titles to a given class by allowing us to map sub-trees to classes.

5.1 Wikification

Wikification was introduced by Mihalcea and Csomai (2007), as a means of automatic keyword extraction and Word Sense Disambiguation. It has since been used for a variety of tasks especially the semantic enrichment of text. A significant advantage of using Wikification is that entities, once identified, are in a normalised format, namely the title of the linked Wikipedia article, thus making entity matching (Section 6) easier.

While simple entity identification involves the direct matching of phrases to Wikipedia titles, more advanced versions of Wikification additionally involve mapping phrases to *related* titles based on the contents of the Wikipedia article. For example, one might choose to map the phrase "the first Briton in space" to the Wikipedia article on "Helen Sharman", who was the first Briton in space. We however, limit ourselves to the simpler version as we are only interested in finding entities and not concepts.

Typically, Wikification involves the identification of potential entities and the subsequent matching of those entities with Wikipedia titles. For example, given a sentence, one could potentially use a Parts of Speech (PoS) tagger to tag the sentences before then extracting sequences of PoS tags that match a predefined set (such as NNP+ or DT*NNP+ and so on). Entities thus extracted could then be matched with Wikipedia titles.

After experimenting with several off the shelf Wikification tools, we found them lacking in the ability to work with Wikipedia Disambiguation pages and topic specific pages. For example, when looking for entities of type "movie" in the sentence "He went to watch the movie 'New York"', we want to be able to match this to the Wikipedia article "New York (film)" and *not* "New York (state)" or "New York City".

5.2 Wikification without PoS Tagging

The obvious way to tag entities in text with Wikipedia titles would be to match every possible phrase in a sentence with every title on Wikipedia. This, however, is impractical as there are over 13.04 million titles in the English Wikipedia. To get around this we run through the titles, and for each title, we split it into its constituent words and save the rest of the title in a file whose name is the first word. Thus, all titles that begin with a particular word are clubbed into a single file and for those titles that are of length one, we add an empty line into the corresponding file. This results in just over 2.1 million files each of which are relatively short and easy to process.

As we sweep through each word in a sentence, we process the file containing Wikipedia titles starting with the same word, and check to see if it contains entities that match the current sentence. This greatly speeds up the process of matching titles to the words in a sentence and provides us with a list of titles that are contained in a given sentence.

We note that this method of Wikification can be used in languages where capitalisation is dissimilar to English or even those languages wherein there is no capitalisation.

5.3 Wikification to Question Classes

Once candidate answers are Wikified, we are then left with the task of mapping these titles to the question classes. We do this by first linking each Wikipedia title to the corresponding DBPedia entry. DBPedia is an attempt to extract structured information from Wikipedia and provides a list of labels and classes associated with each entry. We use these labels and classes to map Wikipedia (and so DBPedia) titles to question classes associated with our taxonomy.

5.4 NER without Wikification

We use the Stanford Named Entity Recogniser (Finkel et al., 2005) to identify entities belonging to the classes "Human Individual", "Human Group" (such as institutions, universities, etc.), and "Location Other", the three classes with compatible granularity. All numeric entities, such as Number:Money, Number:count, and Number:date are identified using an extensive list of regular expressions.

6 Entity Matching

When grading papers, a good maxim to identify plagiarism is "While there is only one way to get it right, there are several ways to get it wrong". We observe that this maxim works because the probability of two students answering a question incorrectly in *the same way* is extremely small, unless of course it's a trick question. Similarly, the chance of candidate answers having the same incorrect entity that also match the class of the question is exceedingly small and machine learning models can make use of this information. To this end, we count the number of occurrences of each entity across all answer candidates of a given question. This requires us to be able to match entities that have been written differently, but are in fact the same.

Entities that have been extracted through Wikification are often normalised "for free." However, there is no simple way to get around this problem in the case of entities extracted through regular expressions, as in the case of numbers and dates where it is common for sentences to contain approximations. For example, consider the question "How many lives were lost in the recent air-crash?", the answer is contained in all of the following sentences: "253 lives were lost in the recent air-crash", "241 passengers and 12 crew died in the recent air-crash", and "around 250 lives were lost in the recent air-crash". This problem is further expanded when the numbers we are dealing with become larger as it is more common for non-technical literature to approximate large numbers. To get around this we round down all numbers to the nearest billion, million, hundred thousand, thousand, hundred or ten.

7 Model Details

We use the Answer Selection model developed by Rao et al. (2016) who rank candidate sentences using a Multi-Perspective Convolutional Neural Network (He et al., 2015a) and a triplet ranking loss function which uses triplets of the question paired with a positive and a negative candidate answer. While other methods model this problem as a pointwise classification problem (He and Lin, 2016; Severyn and Moschitti, 2015), this method models the problem of Answer Selection as a pairwise ranking problem. This involves developing representations for positive and negative answer candidates paired with the question, the primary reason for us choosing this model.

Yet another advantage of this method is that it can make use of existing pointwise models to generate representations which can then be fed into the triplet ranking function. The authors make use of two such pointwise models, one that uses a sentence-level model (He et al., 2015b) and the other that uses a word-level model (He and Lin, 2016). We refrain from elaborating on these methods due to the limitations of space and refer to the reader to the original works.

We make use of the word-level model[3] as we introduce new representations for entities (Section 7.1) which requires us to modify them with answer candidates. The model is additionally initialised with the

[3] `https://github.com/castorini/Castor`

GloVe word embeddings (Pennington et al., 2014) which are also updated during training.

7.1 Highlighting Entities

Having extracted and normalised entities that are contained in each of the answer candidates, we are faced with the task of highlighting these entities within the answer candidates. Before we do this however, we perform some prepossessing steps. We discard any entities that also appear in the question as such entities are unlikely to be the answer. For example, the question "Who is the author of the book, 'The Iron Lady: a biography of Margaret Thatcher"' has, as an answer candidate, the sentence "The iron lady; a biography of Margaret Thatcher by Hugo Young" in which both "Margaret Thatcher" and "Hugo Young" are entities that match the question class, namely "Human Individual". The entity "Margaret Thatcher", however, is discarded as it is also contained in the question.

For each question we count the number of occurrences of each entity across all candidate answers and if the most frequently occurring entity occurs more than twice the number of times the second most frequently occurring one, we pick the first as the maximal entity. For those questions where this is not the case, we pick no maximal entity.

We also create four new "words", *max_entity_left*, *max_entity_right*, *entity_left*, and *entity_right*, which are strings that are not contained in the vocabulary, along with associated word vectors which are randomly initialised with entries between -0.05 and 0.05 and are of the same length as the GloVe word embeddings (300). We then add these embeddings to our embedding dictionary and the words to the vocabulary.

Entities are highlighted in the answer candidates by inserting the words *max_entity_left* and *max_entity_right* on either side of maximal entities, and *entity_left* and *entity_right* around other entities. We also include *entity_left* and *entity_right* at the end of the question and the "words" *max_entity_left* and *max_entity_right* at the end of questions that contain maximal entities. We call this method of highlighting *bracketing*.

A second method of highlighting entities in candidate answers is to replace the entity with a word, a method we call *replacing*. To avoid creating two new words for this method, we reuse two of the four words used above: *max_entity_left* and *entity_left*. Table 2 details the modifications made to an example question and candidate answer using each of the above methods.

Method	Question	Answer Candidate
Original	Who is the author of the book, 'The Iron Lady: a biography of Margaret Thatcher'	in 'The Iron Lady,' Young traces the winding staircase of fortune that transformed the younger daughter of a provincial English grocer into the greatest woman political leader since Catherine the Great.
Bracketing	Who is the author of the book, 'The Iron Lady: a biography of Margaret Thatcher' *max_entity_left max_entity_right entity_left entity_right*	in 'The Iron Lady,' *max_entity_left* Young *max_entity_right* traces the winding staircase of fortune that transformed the younger daughter of a provincial English grocer into the greatest woman political leader since *entity_left* Catherine the Great *entity_right*.
Replacing	Who is the author of the book, 'The Iron Lady: a biography of Margaret Thatcher' *max_entity_left entity_left*	in 'The Iron Lady,' *max_entity_left* traces the winding staircase of fortune that transformed the younger daughter of a provincial English grocer into the greatest woman political leader since *entity_left*.

Table 2: Entities *highlighted* in answer candidates using two different methods. The example assumes that the entity "Young" is a maximal entity.

8 Empirical Evaluation

Having described our method of Question Classification, Entity Identification and Entity Highlighting, we next evaluate our method on the task of AS using different Highlighting methods and training data.

The training data commonly used for this task consists of two sets: The first consists of one hundred manually examined questions and corresponding answers candidates and the second, an automatically

generated set consisting of just over 1200 questions. We found the manually inspected, and hence higher quality test set to be too small for use in this task. However, we also found that the automatically generated training set contained several inconsistencies. To prevent noise, we discarded any questions with answer candidates that contained more than one false positive. We similarly discarded questions from the development set. Thus cleaned, we were left with 1164 questions in the training set and 76 and 60 questions in the Raw and Clean versions of the development set respectively. The development data is what the learning model is optimised on before the best performing model is used to evaluate the test data. To ensure that our results are comparable to those published by others, we make no changes to the test data.

Some question classes cannot have entities highlighted, as in the case of "Description" and "Definition". Some other question classes do not have Entity Identification implemented as we found it impossible to identify all possible elements of the class, as in the case of "Vehicles". We call such questions unhighlighted questions and the rest highlighted questions.

For each of the Clean and Raw versions of the data, we run the model on **a)** unhighlighted data, **b)** data highlighted using *bracketing*, **c)** data highlighted using *replacement*, **d)** unhighlighted data and highlighted data using *bracketing* combined, and **e)** unhighlighted data and highlighted data using *replacement* combined. In cases where we split the data into highlighted and unhighlighted sections, the results are combined to find the MRR and MAP scores of the complete data. The model run on data without entities highlighted (as presented in (Rao et al., 2016)) is the baseline. We also calculate the MRR and MAP scores for the baseline for *each* of these variations as we change the training data in each case. We use the same hyper-parameters as those provided in the implementation of the work by Rao et al. (2016). We include the highlighted versions of the training, development and test data for each of the variations above along with details of the hyper-parameters used, the trained models and the output as part of the supplemental material. We present our results in Table 3.

8.1 Result Analysis

The use of question classes embedded in candidate answers *outperforms the current state of the art in literature in every case except one*. This result (Number 10 in Table 3) is an anomaly that we attribute to over-fitting as we perform no hyper-parameter tuning. The strong performance of the baseline on the unhighlighted data (Sr. No. 1) is expected as answer candidates that are descriptive in nature (as is the case for questions belonging to the classes "Description", "Definition", etc., which also do not have entities identified) must necessarily have a larger overlap with the question. Once again, we ascribe the low performance of the corresponding unhighlighted baseline on the Clean Version (Sr. No. 8) to over-fitting.

The highlighting method of *replacement* performs better than *bracketing* except in the case of the anomaly. We believe this is because Named Entities, with their limited frequency, carry little information. Additionally, the replacement of entities that are phrases with a single frequently occurring word could improve sentence representation.

We expected the combination of the model independently trained on unhighlighted and highlighted data to perform better than that trained on the combination. This is not the case for either version of the test data. As in the case of the anomaly (Sr. No. 10), it is impossible to say if this is a result of over-fitting without performing hyper-parameter tuning on all ten of the models we present.

We also experimented with training different models for each of the coarse classes. We did this by extracting subsets of the training, development and test sets belonging to each of the coarse classes ("HUM", "LOC", ...), training the model on the training subset, optimised on the development subset, and testing it on the test subset. We found these results to be surprisingly poor, and believe this to be a result of deep learning models gaining more from increased data rather than homogeneous data. Homogeneity in data might, in fact, lead to overfitting and hence be detrimental. These results are consistent with results in work by Kyashif (2018) who used the QC system described in this work in task-based and common sense QA, where a similar subdivision led to poorer performance.

Data version		Sr. No.	Question Class	Highlight Method	Train #	Test #	Baseline MRR	Baseline MAP	This Work MRR	This Work MAP
RAW	This Work	1	unhighlighted	N.A.	515	13	(0.8065)	(0.8462)	N.A.	N.A.
		2	highlighted	bracketing	649	82	(0.7583)	(0.8179)	(0.8124)	(0.8304)
		3	highlighted	replacement	649	82	(0.7583)	(0.8179)	(0.8174)	(0.8293)
		4	Combining results from 1 & 2						0.8116	0.8326
		5	Combining results from 1 & 3						0.8262	0.8422
		6	combined	bracketing	1164	95	0.7783	0.8386	0.806	0.8316
		7	combined	replacement	1164	95	0.7783	0.8386	**0.8362**	**0.8625**
	Reported Baseline Performance (Rao et al., 2016)					95	0.764	0.827		
	Implementation Best					95	0.7904	0.8223		
	Prior state of the art (Rao et al., 2016)					95	*0.78*	*0.834*		
CLEAN	This Work	8	unhighlighted	N.A.	515	6	(0.6621)	(0.7222)	N.A.	N.A.
		9	highlighted	bracketing	649	62	(0.7449)	(0.7926)	(0.8354)	(0.8679)
		10	highlighted	replacement	649	62	(0.7449)	(0.7926)	(0.6991)	(0.8211)
		11	Combining results from 8 & 9						0.8201	0.855
		12	Combining results from 8 & 10						0.6958	0.8123
		13	combined	bracketing	1164	68	0.7713	0.8368	0.8324	0.862
		14	combined	replacement	1164	68	0.7713	0.8368	**0.8647**	**0.9039**
	Reported Baseline Performance (Rao et al., 2016)					68	0.762	0.854		
	Prior state of the art (Shen et al., 2017)					68	0.822	0.899		

Table 3: Results for each of the different highlighting methods on the Raw and Clean data versions. "Reported Perfomance" indicates the performance reported by Rao et al. (2016) when using the word-level model (Section 7) which is our baseline and the model that we adapt. "Implementation Best" represents the performance of the same model on the implementation that we use, and "Prior state of the art" represents the prior state of the art reported in literature for this dataset. Results in parentheses represent results on a subset of the test data.

As part of this work we release the following datasets[4]:

1. We release 3,500 of the total 5,500 training questions along with the 500 test questions originally released by Li and Roth (2002) manually updated with our classification.
2. We release the manually verified question classes for all 1500 questions in the AS task.
3. From the questions contained in the AS task, we release the list of entities identified for each answer candidate for the complete test set and for a highlighted training set of 649 questions.
4. We also make available an Application Programming Interface (API) to the modified Question Classification system we describe in this work[5].

8.2 Additional Advantages

Unlike other systems, a significant advantage of the system presented in this work is the fact that, not only does the system succeed in Answer Selection but in most cases *can also extract the specific factoid answer*, when one is present. When highlighting is possible and a maximal entity exists (as is the case in 60 of the 68 questions in the clean test set), such a maximal entity is nearly always the answer.

[4]Download from: `www.harishmadabushi.com/research/answer-selection/`
[5]API available at: `www.harishmadabushi.com/research/questionclassification/`

9 Error Analysis

One of the biggest source of errors tends to be incorrect entity tagging. This is especially so in the case of dates where we tag the date and the year independently thus allowing for the tagging of entities in candidate answer that only mention the year. Unfortunately, this often leads to incorrectly identified maximal entities when both the date and the year are present. In hindsight, we believe a better approach to dates would have been to combine sequential entities before entity matching, so as to capture exact dates, such as "13 October 1997" as a single entity rather than two different entities, one consisting of the day and month and the other of the year.

A similar source of errors occurs when processing the names of individuals, where common ways of shortening names cause errors in both the entity identification and the entity matching steps. For example, the ex-CEO of GE, Jack Welch, is referred to as GE-Welch and John Welch in some answer candidates.

When a candidate answer has too many entities of the required type, it is often misclassified. For example, the candidate answer "on its billions of kilometres flight toward Saturn, Cassini is scheduled to loop its path around Venus, the Earth and Jupiter to get the gravity boost needed for closing in onto its destination" for the question "What is Cassini's destination?", contains a large number of planetary objects diluting the signal generated by the entity. We believe the solution to this lies in increasing the granularity of the QC taxonomy.

There are some completely unexpected sources of errors as in the case of the question "Who established the Nobel Prize awards?", where all entities that contain the answer ("Alfred Nobel") are discarded as they are contained in the question. A more sophisticated method of establishing which entities are discarded could reduce errors of this kind.

Finally, there are some questions and answer candidates that are incorrectly tagged in the Answer Selection test set. For example, the question "What years did Sacajawea accompany Lewis and Clark on their expedition?" has, amongst its candidate sentences, the following two sentences: "the coin honors the young woman and teen-age mother who accompanied explorers meriwether lewis and william clark to the pacific ocean in 1805", and "in 1804 , toussaint was hired by lewis and clark , not for his own skills but for those of sacagawea." While the first candidate answer is marked as one containing the answer the second is not marked as such. While this is the only error of this kind, we found several other sentence candidates similarly marked as containing the answer when they, in fact, do not. Despite this, *we make no modifications to the test set*.

10 Conclusions and Future Work

We presented in this paper a method of Answer Selection that first required us to redefine a QC taxonomy provided by Li and Roth (2002), modify the Question Classification system developed by Tayyar Madabushi and Lee (2016) to match this modified taxonomy, create an entity identification method to extract entities belonging to those classes in our taxonomy, and finally, to use different methods of highlighting entities, so this information can be passed on to the Answer Selection model developed by Rao et al. (2016).

Our experiments show that Question Classes provide such a strong signal to the Deep Learning Model that we outperform current state of the art in all variations of our experiments except one. In the best configuration, our MRR and MAP scores outperform the current state of the art by between 3 and 5 points on both the Raw and Clean versions of the TrecQA Answer Selection test set.

We plan to test this method on different datasets including WikiQA and the Stanford Question Answering dataset, while also increasing the number of classes in our taxonomy and the number and kinds of entities identified by the our Entity Identification system.

The relatively small size of this dataset combined with the rather high accuracy achieved by current state of the art systems highlights the need for research in the field of AS through QC to move to other datasets. One challenge in doing so could be the lack of QA datasets with QC annotations. We hope that our release of the QC API as part of this work will help in this regard.

References

Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A compare-aggregate model with dynamic-clip attention for answer selection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 1987–1990, New York, NY, USA. ACM.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR*, abs/1602.03609.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 937–948, San Diego, California, June. Association for Computational Linguistics.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015a. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586. Association for Computational Linguistics.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015b. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586, Lisbon, Portugal, September. Association for Computational Linguistics.

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 1011–1019, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ulf Hermjakob. 2001. Parsing and question classification for question answering. In *Proceedings of the Workshop on Open-domain Question Answering - Volume 12*, ODQA '01, pages 1–6, Stroudsburg, PA, USA. Association for Computational Linguistics.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*, HLT '01, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *CoRR*, abs/1506.06726.

Cody Kwok, Oren Etzioni, and Daniel S. Weld. 2001. Scaling question answering to the web. *ACM Trans. Inf. Syst.*, 19(3):242–262, July.

Denis Kyashif. 2018. Machine comprehension using commonsense knowledge - a dynamic memory network approach. `https://github.com/deniskyashif/sweet-reason`.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 233–242, New York, NY, USA. ACM.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November.

Amit Mishra and Sanjay Kumar Jain. 2016. A survey on question answering systems with classification. *Journal of King Saud University - Computer and Information Sciences*, 28(3):345 – 361.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Marco Pota, Massimo Esposito, and Giuseppe De Pietro, 2016. *A Forward-Selection Algorithm for SVM-Based Question Classification in Cognitive Systems*, pages 587–598. Springer International Publishing, Cham.

Vasin Punyakanok, Dan Roth, and Wen tau Yih. 2004. Natural language inference via dependency tree mapping: An application to question answering. In *IN SUBMISSION*.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1913–1916, New York, NY, USA. ACM.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 373–382, New York, NY, USA. ACM.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017. Inter-weighted alignment network for sentence pair modeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1179–1189. Association for Computational Linguistics.

Harish Tayyar Madabushi and Mark Lee. 2016. High accuracy rule-based question classification using question syntax and semantics. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1220–1230, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Chen-Tse Tsai, Wen-tau Yih, Chris J.C. Burges, and Scott Wen-tau Yih. 2015. Web-based question answering: Revisiting askmsr. Technical report, Microsoft Research, Redmond, Washington, April.

Nguyen Van-Tu and Le Anh-Cuong. 2016. Improving question classification by feature extraction and selection. *Indian Journal of Science and Technology*, 9(17).

Zhiguo Wang and Abraham Ittycheriah. 2015. Faq-based question answering via word alignment. *CoRR*, abs/1507.02628.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA. In *EMNLP-CoNLL*.

Knowledge as A Bridge: Improving Cross-domain Answer Selection with External Knowledge

Yang Deng[1], **Ying Shen**[1,*], **Min Yang**[2], **Yaliang Li**[3], **Nan Du**[3], **Wei Fan**[3], **Kai Lei**[1]
[1]School of Electronics and Computer Engineering, Peking University Shenzhen Graduate School
[2]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[3]Tencent Medical AI Lab
ydeng@pku.edu.cn, shenying@pkusz.edu.cn, min.yang@siat.ac.cn,
{yaliangli,ndu,davidwfan}@tencent.com, leik@pkusz.edu.cn

Abstract

Answer selection is an important but challenging task. Significant progress has been made in domains where a large amount of labeled training data is available. However, obtaining rich annotated data is a time-consuming and expensive process, creating a substantial barrier for applying answer selection models to a new domain which has limited labeled data. In this paper, we propose Knowledge-aware Attentive Network (KAN), a transfer learning framework for cross-domain answer selection, which uses the knowledge base as a bridge to enable knowledge transfer from the source domain to the target domains. Specifically, we design a knowledge module to integrate the knowledge-based representational learning into answer selection models. The learned knowledge-based representations are shared by source and target domains, which not only leverages large amounts of cross-domain data, but also benefits from a regularization effect that leads to more general representations to help tasks in new domains. To verify the effectiveness of our model, we use SQuAD-T dataset as the source domain and three other datasets (i.e., Yahoo QA, TREC QA and InsuranceQA) as the target domains. The experimental results demonstrate that KAN has remarkable applicability and generality, and consistently outperforms the strong competitors by a noticeable margin for cross-domain answer selection.

1 Introduction

Answer selection, which is a key component of question answering (QA), has attracted increasing attention recently due to its broad applications in natural language processing and information retrieval, such as factoid question answering (Wang et al., 2007) and community-based question answering (Tay et al., 2017). Given a question, answer selection aims to pick out the most relevant answer from a set of candidates. In the literature, answer selection has been extensively studied in the last decade (Severyn and Moschitti, 2015; Wang and Nyberg, 2015; Tan et al., 2016; dos Santos et al., 2016).

Despite the effectiveness of previous studies, answer selection remains a challenge in real-world applications for two reasons. (1) The background information and knowledge beyond the context, which play crucial roles in human text comprehension, have received little attention in recent work for answer selection. (2) Impressive answer selection performances were achieved in domains where a large amount of labeled data is available. However, such fruitful results are subject to an assumption that the test data should be drawn from the same distribution as the training data. Previous studies struggle to cope with answer selection across different data domains. For example, the model trained on SQuAD-T dataset that consists of open-domain factoid questions, is difficult to generalize to the InsuranceQA dataset that consists of non-factoid questions in the insurance domain. In a new domain where its own labeled data is in short supply, obtaining more labels is usually labor-intensive and time-consuming.

Knowledge base (KB), such as YAGO (Weikum et al., 2007), Freebase (Bollacker et al., 2008), provides rich information of relations between entities. It has been widely studied and applied in many tasks (Yang and Mitchell, 2017; Liu et al., 2017). However, its applicability to answer selection has yet to be

* Corresponding Author

Proceedings of the 27th International Conference on Computational Linguistics, pages 3295–3305
Santa Fe, New Mexico, USA, August 20-26, 2018.

Question	Who played ***Dumbledore*** in ***Harry Potter***?
Positive Answer	After Harris' death , ***Michael Gambon*** portrayed ***Dumbledore*** for all of the remaining films.
Negative Answer	Professor Albus Percival Wulfric Brian ***Dumbledore*** is a major character and protagonist of ***J. K. Rowling***'s ***Harry Potter*** series.

Table 1: Example of QA Candidate Pairs.

well-studied. Considering the example in Table 1, existing context-based models may assign a higher score to the negative answer than the positive answer, since the negative answer is more similar to the given question at word level. However, with the background knowledge, we can correctly identify the positive answer based on the relative facts contained in the knowledge base (KB) such as (*Dumbledore*, *played_by*, *Michael Gambon*), (*Michael Gambon*, *cast_in*, *Harry Potter*). Furthermore, QA datasets in different domains or types might differ from syntactic and lexical features, but relations of knowledge in sentences are coherent in the same knowledge base.

Inspired by recent work on transfer learning (TL) and domain adaptation, in this paper, we study how we can leverage labeled data of source domain and external knowledge in knowledge base to help the answer selection in the new target domain which has only limited labeled data. Although transfer learning was employed in many applications (Mou et al., 2016; Li et al., 2017), its application in answer selection is still a relatively new territory and under-explored.

Our idea of leveraging external knowledge as a bridge between source and target domains is motivated by the observation that the data in different domains shares certain common background knowledge which can possibly be transferred from the source domain to the target domain. Thus, we proposed Knowledge-aware Attentive Network (KAN) for transfer learning on answer selection task. In particular, we design a context-guided attentive convolutional neural network, which incorporates knowledge embeddings into sentence representations, to strengthen the representation learning of documents. The training of the transfer learning is performed in two steps: First, the proposed model is trained with the labeled data from source domain and the external knowledge from knowledge base, called pre-training procedure. We expect that the pre-training learns the knowledge-based representation, which enables domain-independent knowledge to be transferred across domains. In addition, pre-training also gives a good initialization of the model parameters, and therefore training at the latter stage gives a good generalization performance even if the size of the target domain dataset is limited. Second, we fine-tune the model on a target domain dataset which has limited labeled data, with the hope that one can safeguard the performance of answer selection in the target domain by leveraging the shared knowledge learned from knowledge base.

To verify the effectiveness of our model for cross-domain answer selection, we use SQuAD-T data as the source domain and three other datasets (i.e., Yahoo QA, TREC QA and InsuranceQA) as the target domains. The experimental results show that our model consistently outperforms previous methods.

2 Related Work

Answer Selection Recent years have witnessed great successes of applying different neural networks, e.g., convolutional neural network (CNN) (Severyn and Moschitti, 2015) and recurrent models like the long short-term memory (LSTM) (Wang and Nyberg, 2015), into Answer Selection. The key idea behind deep neural networks is to encode the input sentences as vector representations. Based on the representations, an output layer is utilized to provide the matching score of two texts. Instead of learning the representations of the question and the answer separately, some recent studies exploit attention mechanisms to learn the interaction information between questions and answers, which can better focus on relevant parts of the input (Tan et al., 2016; dos Santos et al., 2016; Chen et al., 2017). However, these methods are subject to the amount of labeled data and the limited information provided by contexts. Thus, we attempt to apply transfer learning and knowledge base to address these issues.

Transfer Learning Transfer learning aims to transfer knowledge from the source data to the target data in different domains, tasks, or distributions (Pan and Yang, 2010). Most recent studies in transfer learning for natural language processing employ deep neural networks to learn the shared feature representation between two different datasets (Mou et al., 2016; Li et al., 2017). However, it was not until recent years that the application of transfer learning on QA received extensive attention. Min et al. (2017) and Wiese et al. (2017) both employ supervised transfer learning techniques to pre-train a model from a large-scale dataset. Yuan et al. (2018) and Yu et al. (2018) study unsupervised transfer learning under the circumstance that there is only a little labeled target data or some unlabeled target data. In this paper, to make the shared representation more knowledge-aware, we go deeper into the first kind of researches and incorporate external knowledge into transfer learning framework in our method.

Knowledge Base Application As seen in many other tasks, it is a trend to leverage external knowledge from KBs to enrich the representational learning of deep learning models. Several efforts have been made on integrating knowledge embeddings trained by knowledge embedding methods (Bordes et al., 2013) to learn a knowledge-aware sentence representation on machine reading (Yang and Mitchell, 2017), entity typing (Xin et al., 2018) and relation extraction (Han et al., 2018). In this paper, we aim to develop a more general framework for current models to learn knowledge-aware sentence representations on answer selection task.

Inspired by these researches, we propose a transfer learning method for answer selection task, which leverages external knowledge from knowledge base to enrich the representational learning of QA sentences as well as bridge cross-domain QA datasets.

3 Model

In this section, we present the general framework of our transfer learning method on answer selection, which leverages external knowledge from knowledge base as a bridge connecting cross-domain QA datasets. Given a question q, our model aims to rank a set of candidate answers $A = \{a_1, \ldots, a_n\}$.

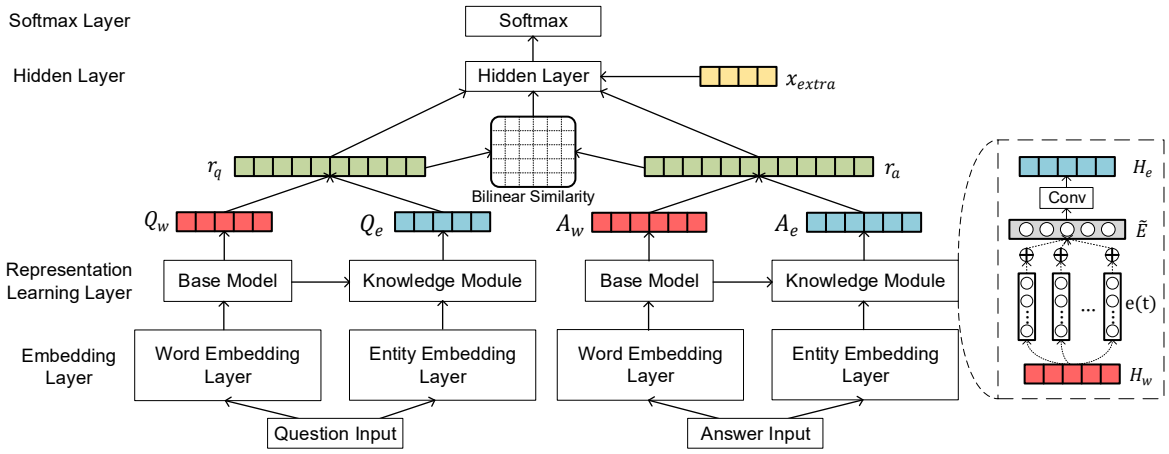


Figure 1: Knowledge-aware Attentive Network for Cross-domain Answer Selection. Blue, red and green matrices denote knowledge-based representations, context-based representations and final knowledge-aware sentence representations, respectively.

As is illustrated in Figure 1, the overall architecture of KAN contains two main components: *Base Model* and *Knowledge Module*. In base model, we employ a pair of deep neural networks to learn the initial context-based representations of questions and answers, separately (Section 3.1). In knowledge module, a context-guided attentive CNN is designed to learn the knowledge-based sentence representation from entities in the sentence (Section 3.2). Afterwards, for both question and answer sentences, there are two different sentence-level representation vectors. Q_w and A_w are learned from base model, while

Q_e and A_e are derived from knowledge module. We obtain the final knowledge-aware attentive sentence representations of question q (i.e., $r_q = [Q_w : Q_e]$) and answer a (i.e., $r_a = [A_w : A_e]$), where [:] is the concatenation operation. There is a fully connected hidden layer before the final binary classification to join all the features (Section 3.3). Finally, a transfer learning method is proposed to transfer the shared knowledge to bridge two different datasets (Section 3.4).

3.1 Base Models

Given a question q and a set of candidate answers $A = \{a_1, \ldots, a_n\}$, we first transform them into vector representations with an embedding layer, and then input these embedding vectors into the base model.

Among numerous deep learning models proposed for answer selection task, we adopt several popular and typical models as our base model to demonstrate the strong applicability and generality of our transfer learning method. The selected base models include: (1) Bi-LSTM, a bidirectional Long-Short Term Memory (Bi-LSTM) network to generate sentence representation (Wang and Nyberg, 2015); (2) Att-LSTM, an Attentive LSTM model with a simple but effective attention mechanism for the purpose of improving the semantic representations for the answers based on the questions (Tan et al., 2016); (3) AP-LSTM, an AP-LSTM model with attentive pooling, a two-way attention mechanism that information from the question and the answer can directly influence the computation of each others representations (dos Santos et al., 2016); and (4) Conv-RNN, a hybrid framework of attention-based Convolutional Recurrent Neural Network (Conv-RNN) (dos Santos et al., 2016) with a similar attention mechanism as Att-LSTM but in the input of the RNN. Note that, unlike the original paper of Conv-RNN, we employ Bi-LSTM as the RNN model instead of the Bi-GRU. For the details of the models, please refer to the original papers.

As for a question vector q and an answer vector a, we obtain the initial context-based sentence representation H_w for the question and the answer from the base model, namely Q_w and A_w.

3.2 Knowledge Module

Knowledge module is responsible for transferring external knowledge from KB into the knowledge-based sentence representations. We first employ n-gram matching to detect all the entity mentions in the sentence, and then retrieve a set of top-K entity candidates from KB for each entity mention. However, the ambiguity issue of the entity is still remained to be tackled, e.g., Santiago can refer to a city or a person. Thus, a context-guided attention mechanism is designed to perform a soft entity linking and learn the knowledge representation for each entity mention simultaneously. We present candidate entities that related to the t-th word in the sentence as $e(t) = \{e_{t_1}, e_{t_2}, \ldots, e_{t_K}\} \in \mathbb{R}^{K \times d_e}$, where d_e is the dimension of the entity embedding in KB. Then, the context-guided knowledge embedding for t-th word is given by

$$m(t) = \tanh\left(W_{em} e(t) + W_{hm} H_w\right), \tag{1}$$

$$\alpha_{t_i} = \frac{exp\left(w_m^T m_{t_i}\right)}{\sum_{m_{t_j} \in m(t)} exp\left(w_m^T m_{t_j}\right)}, \tag{2}$$

$$\widetilde{e_t} = \sum_{e_{t_i} \in e(t)} \alpha_{t_i} e_{t_i}, \tag{3}$$

where W_{em}, W_{hm} and w_m are parameter matrices to be learned. $m(t)$ is a context-guided knowledge vectors, and α_{t_i} denotes the context-guided attention weight that is applied over each candidate entity embedding e_{t_i}. The contextual sentence representations H_w of questions and answers are learned by the base model, while the embeddings of entities in KB are learned by TransE (Bordes et al., 2013).

This procedure produces a context-guided representation for each entity mention in the sentence. Then we apply an CNN layer to learn a higher level knowledge-based sentence representation. The input of the CNN layer are the attentive knowledge embeddings $\widetilde{E} = \{\widetilde{e_1}, \widetilde{e_2}, \ldots, \widetilde{e_L}\} \in \mathbb{R}^{L \times d_e}$.

In the convolution layer, a filter of size n slides over the input embedding matrix to capture the local n-gram information, which is useful to extract the entity features since an entity is likely to be a phrase. Each move computes a hidden layer vector as

$$x_i = [\widetilde{e}_{i-\frac{n-1}{2}}, \ldots, \widetilde{e}_i, \ldots, \widetilde{e}_{i+\frac{n-1}{2}}], \tag{4}$$

$$h_i = \tanh\left(W x_i + b\right), \tag{5}$$

where W and b are the convolution kernel and the bias vector to be learned. Then we employ max-pooling over the hidden layer vectors $h_1, \ldots, h_n$ to generate the final output vector y:

$$y_j = max\left\{h_{1j}, \ldots, h_{nj}\right\}, \tag{6}$$

where y_j and h_{ij} are the j-th value of the output vector y and the hidden vector h_i.

Due to the uncertainty of the length of entities, we exploit several filters of various sizes to obtain n-gram features $\left\{y^{(1)}, y^{(2)}, \ldots, y^{(n)}\right\}$, where $y^{(i)}$ denotes the output vector obtained by the i-th filter. We pass these output vectors through a fully-connected layer to get the final knowledge-based sentence embedding $H_e \in \mathbb{R}^{L \times d_f}$, where d_f is the total filter sizes of CNN and L is the length of the sentence. As for the question q and the answer a, we generate their knowledge-based sentence representations Q_e and A_e as:

$$Q_e = [y_q^{(1)}, y_q^{(2)}, \ldots, y_q^{(n)}]; \quad A_e = [y_a^{(1)}, y_a^{(2)}, \ldots, y_a^{(n)}]. \tag{7}$$

3.3 Training

The initial context-based representations and the knowledge-based representations are concatenated to form the final sentence representations of question q and answer a:

$$r_q = [Q_w : Q_e]; \quad r_a = [A_w : A_e]. \tag{8}$$

Following the ideas in Severyn and Moschitti (2015) and Tay et al. (2017), we incorporate the same additional features into our overall architecture. First, we compute the bilinear similarity score between final question and answer vectors, $s\left(r_q, r_a\right) = r_q^T W r_a$, where $W \in \mathbb{R}^{L \times L}$ is a similarity matrix to be learned. Besides, the same word overlap features $x_{extra} \in \mathbb{R}^4$ as Severyn and Moschitti (2015) and Tay et al. (2017) are also integrated into our model. Thus, the inputs of the hidden layer is a vector $x = [r_q, s(r_q, r_a), r_a, x_{extra}]$, and its output then go through a softmax layer for binary classification:

$$y(q, a) = \text{softmax}\left(W_s x + b_s\right). \tag{9}$$

where $W_s \in \mathbb{R}^{d_x \times 2}$ and $b_s \in \mathbb{R}^2$ are the parameters in the hidden layer. The overall end-to-end model is trained to minimize the cross-entropy loss function:

$$L = -\sum_{i=1}^{N} \left[y_i \log p_i + (1 - y_i) \log\left(1 - p_i\right)\right] + \lambda \|\theta\|_2^2, \tag{10}$$

where p is the output of the softmax layer. θ contains all the parameters of the network and $\lambda\|\theta\|_2^2$ is the L2 regularization.

3.4 Transfer Learning Method

Our goal is to leverage external knowledge from an open-domain knowledge base to bridge cross-domain answer selection data. The transfer learning method between different datasets is divided into two steps: we first initialize the the parameters of model pre-trained on the source dataset, then we further fine-tune on the target dataset. A straightforward way is to fine-tune all the parameters pre-trained by the source data on the target training dataset. Another fashion is to fine-tune a certain part of parameters and keep the rest of parameters fixed during fine-tuning. In our overall model, we present three ways to

fine-tune: (1) Find-tune the entire model; (2) Only find-tune all the weights in the knowledge module. This way aims at sharing the context-based representational learning module learned from the source data and training a better knowledge-based representational learning module with the target training set; (3) Only find-tune all the weights in the base model. On the contrary, this way considers knowledge-based representational learning module a coherent part for transfer, but the context-based representational learning module need to be fine-tuned. Note that for training the model, we followed the same procedure as in Yuan et al. (2018), where pre-trained word embeddings are not updated during training. In this work, so are pre-trained knowledge embeddings.

4 Experiments

4.1 Datasets and Metrics

In the paper, we adopt four widely-used QA datasets for evaluation, including two factoid QA datasets, SQuAD-T and TREC QA, and two community-based QA datasets, InsuranceQA and Yahoo QA. We adopt SQuAD-T as our source dataset for transfer learning due to its high quality and large quantity, while the other datasets are used as target datasets.

- **TREC QA**, collected from TREC QA track 8-13 data (Wang et al., 2007), is a benchmark for open-domain factoid question answering. Most questions are short and factoid-based, and answers are usually consisting trivia information. Following previous works (Tay et al., 2018; Rao et al., 2016), we use Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) as evaluation metrics.

- **InsuranceQA**, proposed by (Feng et al., 2015), contains community-based question and answer pairs from the insurance domain and is split into a training set, a validation set, and two test sets. For the development and test sets, the InsuranceQA also includes an answer pool of 500 candidate answers for each question. This answer pool was produced by including the correct answer and randomly selected candidates from the complete set of unique answers. The top-1 accuracy of the answer selection is reported.

- **Yahoo QA**, introduced in Tay et al. (2017), is a cleaned version of original Yahoo QA dataset [1], which is an open-domain CQA dataset collected from Yahoo Answers. In their setting, questions and answers that are not in the range of 5 - 50 tokens are filtered. Additionally, 4 negative samples are generated for each question by sampling from the top 1000 hits. Note that almost all the cases in this dataset are non-factoid questions. For this dataset, we use the same metrics as (Tay et al., 2017; Tay et al., 2018), including Precision@1 and MRR.

- **SQuAD-T**, introduced in Min et al. (2017), is a modification of SQuAD dataset (Rajpurkar et al., 2016) for answer selection task. SQuAD is a recent open-domain machine reading QA dataset, in which each case is a pair of context paragraph from Wikipedia and a question created manually, and the answer is a span in the context. The task of SQuAD-T is to classify whether each sentence contains the answer, since its contained context paragraphs are split into sentences.

The statistics of all datasets, i.e., training sets, development sets and testing sets, are given in Table 2.

Dataset	#Question (train/dev/test)	#QA Pairs (train/dev/test)	Question Example
TREC QA	1229/82/100	53417/1148/1517	What was the first Gilbert and Sullivan opera?
InsuranceQA	12887/1000/1800x2	37.1K/500K/900Kx2	Why have renter insurance?
Yahoo QA	50.1K/6.2K/6.2K	253K/31.7K/31.7K	How to maximize returns on your investments?
SQuAD-T	87.1K/10.5K/-	708K/53.6K/-	What sits on top of the main building at Notre Dame?

Table 2: Summary statistics of datasets.

[1] http://webscope.sandbox.yahoo.com/catalog.php?datatype=l&did=10

4.2 Experimental Settings

Pre-trained GloVE embeddings[2] of 300 dimensions are adopted as word embeddings. We use a subset of Freebase (FB5M[3]) as our KB, which includes 4,904,397 entities, 7,523 relations, and 22,441,880 facts.

For the base models, we followed exactly the same parameter settings as those in their original papers. For the knowledge module, the width of the convolution filters is set to be 2 and 3, and the number of convolutional feature maps and the attention sizes are set to be 200. For the rest of our models, the final hidden layer size is set to 200 and all other parameters are randomly initialized from [-0.1, 0.1]. The model parameters are regularized with an L2 regularization strength of 0.0001. The learning rate and the dropout rate are set to 0.0005 and 0.5 respectively. The maximum length of sentence is set to be 40. We train all the models in batches with size of 64.

4.3 Results and Analysis

4.3.1 Comparisons Between TL Methods

To evaluate the proposed transfer learning method, we conduct the following comparison experiment with the same base model AP-LSTM (dos Santos et al., 2016), i.e., KAN(AP-LSTM). We first pre-train the models on the source dataset, and then fine-tune the models on target datasets. We also report the ablation tests in terms of discarding the knowledge module (i.e., AP-LSTM) and exploiting several different TL methods as follows:

- **Tgt-Only** is the baseline trained in the training set of target data.
- **Src-Only** is another baseline trained in the training set of source data.
- **Mixed** is to simply mix the training data from both target and source dataset to train the model.
- **Fine-Tune** is a widely used TL method, where we first train a model on the source data, and then use the learned parameters to initialize the model parameters for training another model on the target data. As is presented in Section 3.4, we employ three kinds of fine-tuning methods. Fine-Tune (all) means that we fine-tune all the parameters on the target data. Find-Tune (1) indicates that we fix all the weights of the base model and fine-tune parameters of the rest part of the overall model, while we fix all the weights of the knowledge module in Find-Tune (2).

Model	TL Method	Yahoo QA		TREC QA		InsuranceQA		
		P@1	MRR	MAP	MRR	DEV	TEST1	TEST2
AP-LSTM	(a) Tgt-Only	60.8	76.1	75.9	81.0	69.2	69.8	67.0
	(b) Src-Only	14.5	36.2	74.9	78.1	61.1	62.1	58.8
	(c) Mixed	56.8	73.1	75.1	79.5	69.1	70.2	67.3
	(d) Fine-Tune (all)	72.4	81.9	76.9	82.4	72.2	73.3	70.4
KAN(AP-LSTM)	(e) Tgt-Only	67.2(+6.4)	80.3(+4.2)	78.1(+2.2)	83.1(+2.1)	71.3(+2.1)	71.5(+1.7)	68.8(+1.8)
	(f) Src-Only	15.9(+1.4)	37.4(+1.2)	76.7(+1.8)	81.8(+3.7)	58.1(-3.0)	59.4(-2.7)	56.1(-2.7)
	(g) Mixed	62.4(+5.6)	76.8(+3.7)	76.7(+1.6)	81.6(+2.1)	70.2(+1.1)	71.0(+0.8)	68.5(+1.2)
	(h) Fine-Tune (all)	74.2(+1.8)	83.4(+1.5)	78.8(+1.9)	84.1(+1.7)	**74.6**(+2.4)	75.3(+2.0)	72.1(+1.7)
	(i) Fine-Tune (1)	73.0	82.6	78.7	84.1	74.4	**75.6**	**72.6**
	(j) Fine-Tune (2)	**74.4**	**84.0**	**79.7**	**85.0**	74.4	75.2	72.5
Rank 1		60.1	75.5	77.1	83.8	71.7	71.4	68.3
Rank 2		57.3	73.6	78.0	83.4	68.4	71.7	66.4
Rank 3		55.7	73.5	75.0	81.5	70.0	70.1	62.8

Table 3: Comparisons Between TL Methods. The number in the parenthesis indicates the accuracy change over the same TL method in the base model.

Table 3 reports the experimental results of our transfer learning method on Yahoo QA, InsuranceQA and TREC QA and the performance of previous models that achieve the state of the art. To compare the previous competing models, we adopt the best three results on each dataset reported in the literature, which are presented as Rank 1,2,3 in Table 3. Both Yahoo QA and TREC QA results are reported from

[2]http://nlp.stanford.edu/data/glove.6B.zip

[3]https://research.facebook.com/researchers/1543934539189348

Tay et al. (2018), including Tay et al. (2018), Bradbury et al. (2016), Rao et al. (2016) and Tay et al. (2017). For InsuranceQA, they are from Wang et al. (2017), dos Santos et al. (2016), and Wang et al. (2016). There are multiple interesting observations from Table 3 as follows:

(1) Our transfer learning method (row (h,i,j)) outperforms the state-of-the-art results on Yahoo QA, TREC QA and InsuranceQA datasets by about 10%, 2% and 4% respectively.

(2) The results of KAN(AP-LSTM) (row (e-h)) significantly outperform AP-LSTM (row (a-d)), which demonstrates the effectiveness of incorporating external knowledge into the base model. Among these results, Src-Only on InsuranceQA is an exception since the target dataset is completely different from the source dataset in domain, which interferes the training of the knowledge module with the source dataset and brings a negative effect on the performance.

(3) For row (e), (f) and (g), we observe that Src-Only perform much worse than Tgt-Only, which indicates that the source QA dataset is quite different from all the target QA datasets. The greater the difference between the experimental results, the greater the difference between source and target datasets. Thus, we notice that Yahoo QA is the most different from SQuAD-T, and InsuranceQA is also very different from SQuAD-T, while TREC QA is the most similar one. Besides, the performance of Mixed is also worse than Tgt-Only, which implies that simply mixing the training data from two different datasets may lead to the overfitting of the source data.

(4) For row (h), (i) and (j), it can be observed that the Fine-Tune methods outperform the Tgt-Only method (row (e)), demonstrating that pre-training the model parameters on a source dataset is better than randomly initializing them. In specific, fine-tuning all the parameters of the model (row (h)) cannot always achieve the best result, because it may cause overfitting to fine-tune the entire model. For Yahoo QA and TREC QA, only fine-tuning the parameters of the base model (row (j)) achieves the best result. We infer that it is due to the shared knowledge from knowledge base is coherent between the source and target datasets. Conversely, InsuranceQA achieves its best performance with fine-tuning only the weights of the knowledge module (row (i)). Results on InsuranceQA indicate that one should pay attention to fine-tune the knowledge module rather than the base model when domain knowledge of the target dataset is quite different from the source dataset as InsuranceQA and SQuAD-T.

4.3.2 Applicability and Generality of our TL Method

To demonstrate the applicability and generality of our transfer learning method, besides AP-LSTM, we implement the overall transfer learning framework with other three base models, including Conv-RNN (Wang et al., 2017), Bi-LSTM (Wang and Nyberg, 2015), and Att-LSTM (Tan et al., 2016). The experimental results on TREC QA are summarized in Table 4.

From Table 4, we observe a similar result as Section 4.3.1. For these three base models, it makes a significant performance boost to incorporate external knowledge into the overall architecture. Besides, compared with Tgt-Only, Fine-Tune methods perform much better, which demonstrates that our transfer learning framework works on all the given models.

TL	Knowledge	Conv-RNN		Bi-LSTM		Att-LSTM	
Method	Module	MAP	MRR	MAP	MRR	MAP	MRR
Tgt-Only	w/o	77.1	82.4	75.0	80.4	73.5	79.2
	w/	78.0	83.1	77.4	82.5	75.9	80.1
Src-Only	w/o	74.9	78.7	74.1	79.5	72.1	76.9
	w/	76.3	80.2	75.6	81.3	74.8	78.6
Mixed	w/o	76.4	81.0	75.2	80.6	73.3	79.3
	w/	77.6	82.7	77.0	82.4	75.5	80.0
Fine-Tune (all)	w/o	78.3	83.2	76.8	82.3	75.3	79.8
	w/	79.4	**84.8**	78.2	83.6	77.1	81.3
Fine-Tune (1)	w/	78.8	84.0	78.4	83.1	77.7	81.6
Fine-Tune (2)	w/	**79.7**	**84.8**	**78.5**	**83.7**	**77.9**	**81.9**

Table 4: Experiment with Different Base Models.

4.3.3 Size of Target Dataset for Fine-tuning

We conduct experiments to study the relationship between the performance and the amount of training data from the target dataset for fine-tuning the model. We first pre-train the models on SQuAD-T, then vary the training data size of the target dataset, i.e., TREC QA and InsuranceQA, for fine-tuning. Note that we employ the AP-LSTM as the base model (i.e., KAN(AP-LSTM)) and fix all the parameters of the knowledge module during fine-tuning (i.e., Fine-Tune (2)).

The experimental result is summarized in Table 5. In general, fine-tuning with more target data actually improves the overall results. We observe that the performance increase from using 0% to 20% of target training data is substantially larger than latter increases. This result shows that we can achieve a competitive result with a small amount of target labeled data, by using our transfer learning method. Besides, it is obvious to notice that when increasing the number of target training data, the improvement on InsuranceQA is much more significant than that on TREC QA. As is presented in the analysis in Section 4.3.1, the InsuranceQA dataset is much different from the source dataset than the TREC QA dataset, not only in types but also in domains. Thus, the experiment result demonstrates the strong applicability in transferring shared knowledge between diverse datasets, even with limited labeled target data.

Percentage of target	TREC QA		InsuranceQA		
data for fine-tuning	MAP	MRR	DEV	TEST1	TEST2
0%	76.7	81.8	58.1	59.4	56.1
20%	78.1 (1.4)	83.4 (1.7)	72.0 (13.9)	72.8 (13.4)	69.5 (13.4)
40%	78.9 (0.8)	83.7 (0.3)	73.2 (1.2)	74.0 (1.2)	71.1 (1.6)
60%	79.1 (0.2)	84.2 (0.5)	73.6 (0.4)	74.5 (0.5)	71.6 (0.5)
80%	79.3 (0.2)	84.7 (0.5)	74.1 (0.5)	74.9 (0.4)	72.1 (0.6)
100%	79.7 (0.4)	85.0 (0.3)	74.4 (0.3)	75.2 (0.3)	72.5 (0.4)

Table 5: Results of varying sizes of the target datasets used for fine-tuning. The number in the parenthesis indicates the accuracy increases over the previous row.

4.3.4 Completeness of Knowledge Base

To observe the effect of the completeness of KB on the performance, we report the results on TREC QA and InsuranceQA with the incomplete knowledge base that randomly drops 20%-80% knowledge. Note that as the experimental settings in Section 4.3.3, we also employ the AP-LSTM as the base model (i.e., KAN(AP-LSTM)) and fix all the parameters of the knowledge module during fine-tuning (i.e., Fine-Tune (2)). Figure 2 shows that our model is robust and achieves excellent performance on the KB with different completeness. Besides, the more complete the knowledge base we leverage, the better the overall performance, which demonstrates the effectiveness of integrating external knowledge into the proposed method.

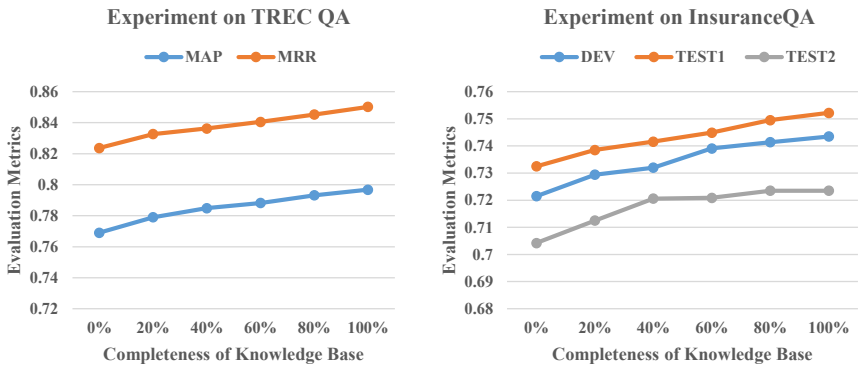


Figure 2: Effect of KB Completeness

5 Conclusion

In this paper, we propose knowledge-aware attentive network (KAN), a novel and general transfer learning framework, for cross-domain answer selection. We incorporate external knowledge from knowledge base into deep learning models to enrich the sentence representational learning and aid in transferring more valuable information between cross-domain datasets. Experimental results on three benchmark datasets demonstrate the superiority of our proposed method on answer selection task. We also conduct experiments to show the applicability and generality of our method and show that a resource-poor dataset can benefit from not only the scale of a resource-rich dataset but also the shared knowledge learned from knowledge base.

Acknowledgements

This work was financially supported by the National Natural Science Foundation of China (No.61602013), the Shenzhen Science and Technology Innovation Committee (Grant No. JCYJ20170412150946024 and JCYJ20170818091546869), and Huawei Innovation Research Program (YBN2017125201).

References

[Bollacker et al.2008] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250. ACM.

[Bordes et al.2013] Antoine Bordes, Nicolas Usunier, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795.

[Bradbury et al.2016] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2016. Quasi-recurrent neural networks.

[Chen et al.2017] Qin Chen, Qinmin Hu, Jimmy Xiangji Huang, Liang He, and Weijie An. 2017. Enhancing recurrent neural networks with positional attention for question answering. In *SIGIR*, pages 993–996. ACM.

[dos Santos et al.2016] Cıcero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR, abs/1602.03609*.

[Feng et al.2015] Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. pages 813–820.

[Han et al.2018] Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *AAAI*.

[Li et al.2017] Zheng Li, Yu Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017. End-to-end adversarial memory network for cross-domain sentiment classification. In *Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 2237–2243.

[Liu et al.2017] Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Meeting of the Association for Computational Linguistics*, pages 1789–1798.

[Min et al.2017] Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. 2017. Question answering through transfer learning from large fine-grained supervision data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 510–517.

[Mou et al.2016] Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in nlp applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 479–489.

[Pan and Yang2010] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge & Data Engineering*, 22(10):1345–1359.

[Rajpurkar et al.2016] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

[Rao et al.2016] Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *CIKM*, pages 1913–1916. ACM.

[Severyn and Moschitti2015] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*, pages 373–382.

[Tan et al.2016] Ming Tan, Cicero Dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Meeting of the Association for Computational Linguistics*, pages 464–473.

[Tay et al.2017] Yi Tay, Minh C Phan, Luu Anh Tuan, and Siu Cheung Hui. 2017. Learning to rank question answer pairs with holographic dual lstm architecture. In *SIGIR*.

[Tay et al.2018] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Cross temporal recurrent networks for ranking question answer pairs. In *AAAI*.

[Wang and Nyberg2015] Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *Meeting of the Association for Computational Linguistics*, pages 707–712.

[Wang et al.2007] Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP*, pages 22–32.

[Wang et al.2016] Bingning Wang, Kang Liu, and Jun Zhao. 2016. Inner attention based recurrent neural networks for answer selection. In *Meeting of the Association for Computational Linguistics*, pages 1288–1297.

[Wang et al.2017] Chenglong Wang, Feijun Jiang, and Hongxia Yang. 2017. A hybrid framework for text modeling with convolutional rnn. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2061–2069. ACM.

[Weikum et al.2007] Gerhard Weikum, Gerhard Weikum, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *International Conference on World Wide Web*, pages 697–706.

[Wiese et al.2017] Georg Wiese, Dirk Weissenborn, and Mariana Neves. 2017. Neural domain adaptation for biomedical question answering. In *Conference on Computational Natural Language Learning*, pages 281–289.

[Xin et al.2018] Ji Xin, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2018. Improving neural fine-grained entity typing with knowledge attention. In *AAAI*.

[Yang and Mitchell2017] Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *Meeting of the Association for Computational Linguistics*, volume 1, pages 1436–1446.

[Yu et al.2018] Jianfei Yu, Minghui Qiu, Jing Jiang, Jun Huang, Shuangyong Song, Wei Chu, and Haiqing Chen. 2018. Modelling domain relationships for transfer learning on retrieval-based question answering systems in e-commerce.

[Yuan et al.2018] Chung Yuan, Hung Yi Lee, and James Glass. 2018. Supervised and unsupervised transfer learning for question answering. In *NAACL-HLT*.

Modeling Semantics with Gated Graph Neural Networks for Knowledge Base Question Answering

Daniil Sorokin and **Iryna Gurevych**
Ubiquitous Knowledge Processing Lab (UKP) and Research Training Group AIPHES
Department of Computer Science, Technische Universität Darmstadt
www.informatik.tu-darmstadt.de/ukp/

Abstract

The most approaches to Knowledge Base Question Answering are based on semantic parsing. In this paper, we address the problem of learning vector representations for complex semantic parses that consist of multiple entities and relations. Previous work largely focused on selecting the correct semantic relations for a question and disregarded the structure of the semantic parse: the connections between entities and the directions of the relations. We propose to use Gated Graph Neural Networks to encode the graph structure of the semantic parse. We show on two data sets that the graph networks outperform all baseline models that do not explicitly model the structure. The error analysis confirms that our approach can successfully process complex semantic parses.

1 Introduction

Knowledge base question answering (QA) is an important natural language processing problem. Given a natural language question, the task is to find a set of entities in a knowledge base (KB) that constitutes the answer. For example, for a question "What is Princess Leia's home planet?" the answer, "Alderaan", could be retrieved from a general-purpose KB, such as Wikidata[1]. A successful KB QA system would ultimately provide a universally accessible natural language interface to factual knowledge (Liang, 2016).

QA requires precise modeling of the question semantics through the entities and relations available in the KB in order to retrieve the correct answer. Figure 1 shows how the above example question could be modeled using Wikidata. The depicted graph structure consists of entities and relations from the KB and the special q-node. Any entity in the KB that can take the place of the q-node will be a part of the answer.

In this paper, we describe a semantic parsing approach to the problem of KB QA. That is, for each input question, we construct an explicit structural semantic parse (*semantic graph*), as in Figure 1. Semantic parses can be deterministically converted to a query to extract the answers from the KB. Similar graphical representations were used in previous work (Yih et al., 2015; Reddy et al., 2016; Bao et al., 2016).

However, the modern semantic parsing approaches usually focus either on the syntactic analysis of the input question (Reddy et al., 2016) or on detecting individual KB relations (Yu et al., 2017), whereas *the structure of the semantic parse* is ignored or only approximately modeled. Reddy et al. (2016) use the syntactic structure of the question to build all possible semantic parses and then apply a linear model with manually defined features to choose the correct parse. A subset of their features encodes basic information about the graph structure of the semantic parse (e.g. number of nodes). The state-of-the-art approach of Yih et al. (2015) and Bao et al. (2016) restricts all semantic parses to a single core relation and a small set of constraints that can be added to it. Their system uses manual features for the constraints and a similarity score between the core relation and the question to model the semantic parses.

The abovementioned systems were evaluated on the WebQuestions data set (Berant et al., 2013). Figure 2 plots results for the state-of-the-art systems by the number of relations that needs to be identified to get the correct answer to a question.[2] For example, the question in Figure 1 requires two relations

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

[2] We include results for the most recent systems that have published the output on the WebQuestions data set. We compute the number of needed relations from the manual semantic parses provided by Yih et al. (2016).

Proceedings of the 27th International Conference on Computational Linguistics, pages 3306–3317
Santa Fe, New Mexico, USA, August 20-26, 2018.

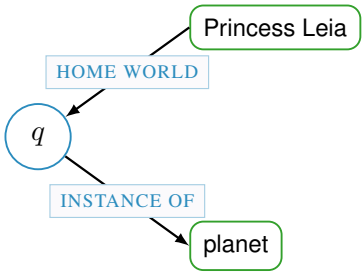


Figure 1: Semantic graph for an example question "What is Princess Leia's home planet?"

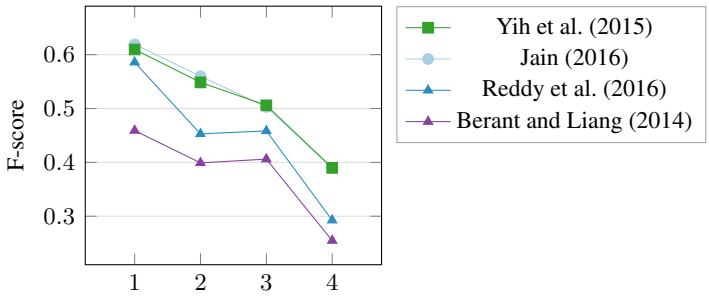


Figure 2: F-score QA results from previous work by the number of relations needed to find the correct answer

to find the answer. It can be clearly seen in Figure 2 that the lack of structure modeling in the modern approaches results in a worse performance on more complex questions that require more than one relation.

We claim that one needs to *explicitly model the semantic structure* to be able to find the correct semantic parse for complex questions. In this paper, we address this gap and investigate ways to encode the structure of a semantic parse and to improve the performance for more complex questions. In particular, we adapt Gated Graph Neural Networks (GGNNs), described in Li et al. (2016), to process and score semantic parses. To verify that GGNNs indeed offer an improvement, we construct a set of baselines based on the previous work that we train and evaluate in the same controlled QA environment. Throughout the experiments, we use the Wikidata open-domain KB (Vrandečić and Krötzsch, 2014) to construct semantic parses and retrieve the answers.[3]

Contributions To summarize, the main contributions of our work are:

(i) Our analysis shows that the current solutions for KB QA do not perform well on complex questions;

(ii) We apply Gated Graph Neural Networks on directed graphs with labeled edges and adapt them to handle a large set of possible entities and relations types from the KB. To the best of our knowledge, we are the first to use GGNNs for semantic parsing and KB QA;

(iii) Our Gated Graph Neural Network implementation for semantic parsing improves performance on complex questions in comparison to strong baselines. The results show a 27.4% improvement of the F-score against the best non-graph model.

Code and data sets Our system can be used with a pre-trained model to answer factual questions with Wikidata or trained anew on any data set that has question-answer pairs. The complete code, the scripts that produce the evaluation data and the installation instructions can be found here: `https://github.com/UKPLab/coling2018-graph-neural-networks-question-answering`.

2 Semantic parsing

2.1 Semantic graphs

We employ structural semantic representations in the form of graphs to encode the meaning of a question. Our *semantic graphs* consist of a question variable node (q), Wikidata entities (Taylor Swift), relation types from Wikidata (PERFORMER) and constraints (see Figure 3 for an example graph with a constraint). The q-node is always present and denotes the answer to the question. That is, all entities from the KB that can take its place so that all relations and constraints hold, constitute the answer to the question.

We write down a graph as a set of edges $\mathcal{E}$. Each edge links the q-node to one or two Wikidata entities (binary or ternary edge). The edge set $\mathcal{E}$ is defined via Wikidata relations and entities. Formally, Wikidata can be described as a very large graph $W = (E, R, I)$, where E is a set of entities, R is a set of binary

[3] At the moment, Wikidata contains more than 40 million entities and 350 million relation instances: `https://www.wikidata.org/wiki/Special:Statistics`

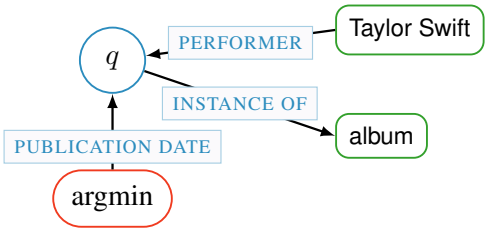


Figure 3: A semantic graph for an example question "What was the first Taylor Swift album?"

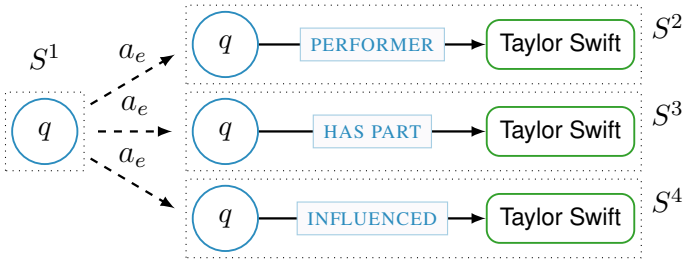


Figure 4: A single graph generation step: applying the add entity action a_e on an empty graph

relation types and I is a collection of relation instances encoded as $r(e_1, e_2),\ \ r \in R,\ \ e_1, e_2 \in E$ (e. g. CAPITAL(Hawaii, Honolulu)). Ternary relation instances in Wikidata are stored as a main relation triple and an attached modifer which itself is also a binary relation: $r_2\big(r_1(e_1, e_2), e_3\big),\ \ r_1, r_2 \in R,\ \ e_1, e_2, e_3 \in E$ (e.g. CHARACTER ROLE$\big($CAST MEMBER(Star Wars, Carrie Fisher), Princess Leia$\big)$. Then, the edge set $\mathcal{E}$ of a semantic graph consists of binary and ternary edges that correspond to Wikidata relation instances with q in place of one of the relation arguments. For temporal relations, the second argument can be either an argmax or an argmin constraint. This means that only entities that have the maximum or the minimum value of that relation are considered for the answer. This definition represents a subset of first-order logic semantic parses restricted to conjunctions of predicates. The graphs are also isomorphic to SPARQL queries that can be used to evaluate a graph against the KB.

Our semantic graphs were inspired by the query graphs of Yih et al. (2015) and their extension in Bao et al. (2016), the key difference being that we do not differentiate between the core relation and modifiers, but rather allow graphs that have multiple independent relations. We also allow relations to attach to other nodes rather than the q-node, enabling longer relation paths between a known entity and the q-node. Thus, the semantic graphs defined here are a superset of the query graphs in Yih et al. (2015) and allow us to model more complex meanings. Consequently, they also correspond to the formalized representations used by Reddy et al. (2014) and the simple λ-DCS (Berant et al., 2013), since those were the foundation for the query graphs (Yih et al., 2015).

2.2 Semantic graph construction

Yih et al. (2015) defined a step-by-step staged graph generation that does not need full syntactic parses and was also adopted in subsequent publications (Bao et al., 2016; Peng et al., 2017). We use the same procedure as Yih et al. (2015) to construct semantic graphs with a set of modifications that allow us to build more expressive and complex graphs.

We define a set of states and a set of actions that can be applied at each state to extend the current semantic graph. A state is a tuple of a graph and a set of free entities: $\mathcal{S} = (\mathcal{E}, \mathcal{F})$, where the graph is $\mathcal{E}$, as defined above, and $\mathcal{F} = \{e | e \in E\}$. The set of free entities $\mathcal{F}$ is the entities that were identified in the question but were not yet added to the graph.[4] The first state is a tuple of an empty graph with no edges and a set of all entities identified in the question $\mathcal{S}^1 = (\{\}, \mathcal{F})$.

Let $\mathcal{A} = \{a_e, a_c, a_m\}$ be the set of actions that can be taken to extend the graph at the current state. The action a_e (*add entity*) pops a free entity e from the set $\mathcal{F}$ at the current state $\mathcal{S}^t$. Then it queries the KB and retrieves the set of available relation types R_e for the entity e. For each relation type $r \in R_e$, it creates a new copy of the graph and adds a new directed edge between the q-node and e with the relation type r: $a_e(\mathcal{E}, \mathcal{F}) = \{\mathcal{E} \cup r(q, e), \mathcal{F} - e \mid e \in \mathcal{F}, r \in R_e\}$. Contrary to Yih et al. (2015) and Bao et al. (2016), we allow two-step paths between the q-node and the entity e: $r(q, d) \wedge r(d, e)$.[5]

The action a_c (*add constraint*) pops a free entity e and follows the same procedure as a_e, but instead of adding a new edge, it adds a modifier to the last added edge of the graph, thus creating a ternary edge: $a_c(\mathcal{E}, \mathcal{F}) = \{\mathcal{E} \cup r_2(r_1(q, e_1), e_2), \mathcal{F} - e_2 \mid e_2 \in \mathcal{F}, r_2 \in R_{e_2}, r_1(q, e_1) \in \mathcal{E}\}$. Finally, the action a_m

[4] We use an off-the-shelf entity linker to identify and disambiguate entity mentions in the question (see Section 4.4).

[5] Freebase relation instances always have an intermediate node and a two-step path corresponds to a single step in Wikidata.

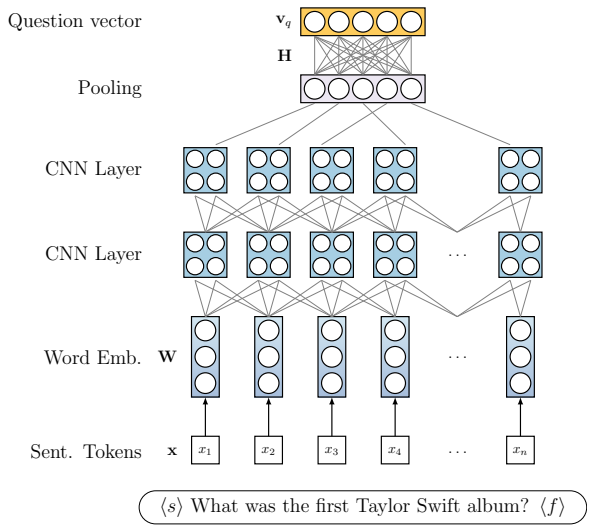


Figure 5: Deep Convolutional Neural Networks (DCNN) architecture, here used to process an example question

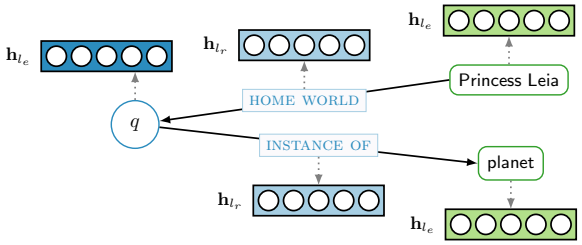


Figure 6: Encoding a graph into initial hidden states

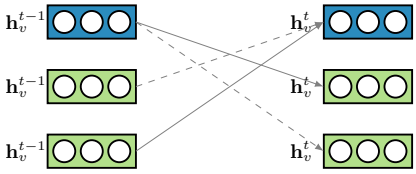


Figure 7: Unrolled recurrence for one timestep, solid lines show updates along the direction of the relation in the graph and the dashed lines in the opposite direction

(*add argmax/argmin*) adds a new edge with either argmax or argmin sorting constraint to the semantic graph: $a_m(\mathcal{E}, \mathcal{F}) = \{\mathcal{E} \cup r(q, \text{argmax}), \mathcal{F}; \mathcal{E} \cup r(q, \text{argmin}), \mathcal{F} \mid r \in R_d\}$, where R_d is a set of KB relation types that allow dates as values. Our semantic graph construction process allows to effectively search the space of possible graphs for a given question through an iterative application of the defined actions $\mathcal{A}$ on the last state $\mathcal{S}^t$ (see, for example, Figure 4).

3 Representation learning

We follow the state-of-the-art approaches for QA (Yih et al., 2015; Dong et al., 2015; Bao et al., 2016) and learn representations for the question and every possible semantic graph. Then we use a simple reward function γ to judge if a semantic graph is a correct semantic parse of the input question. Below we describe the architectures that we use to learn the representations for questions and graphs.

3.1 Deep Convolutional Networks

We use Deep Convolutional Neural Networks (DCNN) to learn a representation for a question. DCNNs have been proven effective for constructing sentence-level representations on a variety of NLP tasks, including named entity recognition (Strubell et al., 2017) and KB QA (Yih et al., 2015; Dong et al., 2015).

The DCNN architecture is depicted in Figure 5, where it is used to map an input question to a fixed-size vector representation. The input question is first tokenized and the special start and end tokens are added to the sequence: $\mathbf{x} = \{\langle s \rangle, x_1, x_2 \ldots x_n, \langle f \rangle\}$. Next, we map the tokens in $\mathbf{x}$ to d_w-dimensional pre-trained word embeddings, using a matrix $\mathbf{W}_{glove} \in \mathbb{R}^{|V_w| \times d_w}$, where $|V_w|$ is the size of the vocabulary. We use 50-dimensional GloVe embeddings that were trained on a 6 billion corpus (Pennington et al., 2014). The sequence of word embeddings is further processed by an array of CNN layers. We apply a pooling operation after the last CNN layer and transform the output with a fully connected layer $\mathbf{H}$ and a ReLU non-linearity. We take the resulting vector $\mathbf{v}_q$ as the representation of the question.

3.2 Gated Graph Neural Networks

Gate Graph Neural Networks (GGNN) process graphs by iteratively updating representations of graph nodes based on the neighboring nodes and relations (Li et al., 2016). We adopt GGNNs for semantic parsing to learn a vector representation of a semantic graph. Li et al. (2016) give a formulation of GGNNs for graphs with labeled nodes and typed directed edges. We extend their formulation to include labeled edges. To the best of our knowledge, we are the first to apply GGNN to semantic parsing and KB QA.

For a graph $\mathcal{E}$, we extract a set of all entities $\mathcal{V}$ in the graph and a set of all relation types $\mathcal{R}$ of its edges. We use labels from Wikidata to compute vectors for entities and relation types (Figure 6). This enables us to directly incorporate the information on millions of entities and hundreds of relation types from the KB. For an entity $e \in \mathcal{V}$ or a relation type $r \in \mathcal{R}$ we retrieve the label and tokenize it: $\mathbf{l} = \{l_1, l_2 \dots l_n\}$. Then we map each token in $\mathbf{l}_e$, $\mathbf{l}_r$ to a word embedding using the matrix $\mathbf{W}_{glove}$, sum them and process with a fully connected layer to get a single label vector: $\mathbf{h}_l = \tanh\big(\mathbf{W}_\mathbf{l}\big[\sum_{n=1}^{|\mathbf{l}|} w_n\big] + \mathbf{b}_\mathbf{l}\big)$. We initialize the hidden states for the graph nodes with the label vectors of the entities: $\mathbf{h}_v^{(1)} = \mathbf{h}_{l_e}$. We further transform the relation label vectors to get directional embeddings for relations types: $\mathbf{h}'_r = \mathbf{W}_{\rightarrow}\mathbf{h}_{l_r}$, $\mathbf{h}''_r = \mathbf{W}_{\leftarrow}\mathbf{h}_{l_r}$. Using the same word embeddings as an input to construct the question and relation representations has been shown successful in previous work (Jain, 2016).

Propagation Model GGNNs are a type of recurrent neural networks. The recurrence is unrolled for a fixed number of steps T and the gating mechanism works akin to Gated Recurrent Units (Cho et al., 2014). The propagation model for GGNN is defined as follows:

$$\begin{aligned}\mathbf{a}_v^{(t)} &= \mathbf{A}_{v:}^\top\big[\mathbf{h}_1^{(t-1)\top} \dots \mathbf{h}_{|\mathcal{V}|}^{(t-1)\top}\big] \\ &\quad + \mathbf{A'}_{r:}^\top\big[\mathbf{h}_1'^\top \dots \mathbf{h}_{|\mathcal{R}|}'^\top, \mathbf{h}_1''^\top \dots \mathbf{h}_{|\mathcal{R}|}''^\top\big]\end{aligned} \quad (1)$$

$$\mathbf{z}_v^t = \sigma\big(\mathbf{W}^z\mathbf{a}_v^{(t)} + \mathbf{U}^z\mathbf{h}_v^{(t-1)} + \mathbf{b}^z\big) \quad (2)$$

$$\mathbf{r}_v^t = \sigma\big(\mathbf{W}^r\mathbf{a}_v^{(t)} + \mathbf{U}^r\mathbf{h}_v^{(t-1)} + \mathbf{b}^r\big) \quad (3)$$

$$\widetilde{\mathbf{h}_v^{(t)}} = \tanh\big(\mathbf{W}\mathbf{a}_v^{(t)} + \mathbf{U}\big(\mathbf{r}_v^t \odot \mathbf{h}_v^{(t-1)}\big) + \mathbf{b}\big) \quad (4)$$

$$\mathbf{h}_v^{(t)} = (1 - \mathbf{z}_v^t) \odot \mathbf{h}_v^{t-1} + \mathbf{z}_v^t \odot \widetilde{\mathbf{h}_v^{(t)}}, \quad (5)$$

where σ is the logistic sigmoid function and $\odot$ is the element-wise multiplication. The matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times 2|\mathcal{V}|}$ stores the structure of the graph: a row of the matrix $\mathbf{A}_{v:}$ records the edges between the node v and the other nodes in the graph (we differentiate between incoming and outgoing edges). The second matrix $\mathbf{A'} \in \mathbb{R}^{|\mathcal{V}|\times 2|\mathcal{R}|}$ stores the relation types of the incoming and outgoing edges.

The main difference from the model defined in Li et al. (2016) is that we compute activations $\mathbf{a}_v^{(t)}$ based on both the node hidden vectors $\mathbf{h}_v^{(t-1)}$ and relation hidden vectors $\mathbf{h}'_r$ (Eq 1). The $\mathbf{z}_v^t$ in Eq 2 and $\mathbf{r}_v^t$ in Eq 3 are update and reset gates, that incorporate information from nodes, relation types and from the previous step to update nodes' hidden states at each iteration. We do not make updates to the hidden vectors of the relations and use them only to pass the information to the nodes' hidden states.

Graph-level Output Vector We unroll the recurrence for $T = 5$ steps in the experiments (Figure 7). To produce a graph-level output vector, we take the hidden vector for the q-node at the last time step $t = T$ and transform it with a fully-connected layer and the ReLU non-linearity: $\mathbf{v}_g = \mathrm{ReLU}\big(\mathbf{W}\mathbf{h}_q^{(t)} + \mathbf{b}\big)$.

4 Experiments

4.1 Data

We use Wikidata to show experimentally that GGNNs are better at learning representations of semantic graphs than previous approaches. Hence, we choose two data sets that can be processed with Wikidata to compare the GGNNs to other models: WebQSP-WD and QALD-7.

WebQSP-WD We derive WebQSP-WD from WebQSP (Yih et al., 2016), which is a corrected version of the popular WebQuestions data set (Berant et al., 2013). WebQSP contains natural language questions, Freebase IDs of the correct answers and SPARQL queries to retrieve them that also use Freebase. Freebase was a common choice of a KB among the previous work, but was discontinued and is no longer up-to-date, including unavailability of APIs and new dumps. The questions in the data set were collected with the Google Suggest API and are thus more 'natural' than manually constructed questions. The answers were retrieved from Freebase with the help of crowd-sourcing. The data set contains both simple questions that can be answered with a single relation as well as complex questions that require multiple relations and constraints. It is a common benchmark for semantic parsers and information retrieval systems and was used in the most recent studies on KB QA. We automatically map Freebase IDs in the WebQSP train and test sets to Wikidata IDs and filter out questions which answers do not have the mapping. We designate

this version of the dataset WebQSP-WD. It is important to note that this does not ensure that a question is answerable with Wikidata as there still might be no relation paths in the KB that connect the entities in the question with the answer.

QALD-7 As the second data set, we use QALD-7 that was developed for Task 4 of the QALD-7 Shared Task, "English question answering over Wikidata" (Usbeck et al., 2017). The QALD-7 data set contains a small number of manually constructed complex questions that were specifically created to test system's ability to process questions with multiple entities and constraints. The data set uses Wikidata IDs for all annotations. We do not train on QALD-7 data set and use it solely for an out-of-domain evaluation. The data set statistics can be found in Table 1.

4.2 Models

We define five models for our experiments, including three baselines and two graph models. We use cosine similarity between the question representation and the semantic graph representation as a reward function that judges whether the semantic graph is the correct parse of the question: $\gamma = \cos(\mathbf{v_s}, \mathbf{v_g})$.

1. STAGG (re-implementation of Yih et al. (2015)) — We implement a model that uses a combination of a neural network and manual features suggested in Yih et al. (2015) and in the follow up work of Bao et al. (2016). The approach of Yih et al. (2015) performed best among the previous work on complex questions (see Figure 2). First, DCNN is used to produce a representation for the input question, as described in Section 3.1. Then, we replace the entity tokens in the input question with a special entity symbol $\langle e \rangle$ and apply DCNN on it again to get a second representation for the question. For each semantic graph, we take the label of the relation in the first edge (*core edge*), tokenize it and likewise apply DCNN on it to get a representation. We produce two representations for the core relation: one that includes the label of the attached entity and one that includes the entity symbol. The manual features include binary indicators for modifiers and constraints in the semantic graphs, as well as features for certain keywords in the question (see Yih et al. (2015) for a detailed description).

   The original system and the models of Yih et al. (2015) and of Bao et al. (2016) are not available and therefore we use our own implementation of their approach. There is a number of small difference with the original model: we use a deep CNN instead of a single CNN layer and train the DCNN together with the manual features, whereas Yih et al. (2015) pre-trained the CNN model on a separate corpus and used its output in the feature model.

2. Single Edge model — We use the DCNN to encode the question and the label of the first edge of a semantic graph. The rest of the information is ignored.

3. Pooled Edges model — We use the DCNN to encode the question and the label of each edge in the semantic graph. To get a fixed-vector representation of the graph, we apply a pooling operation over the representation of the individual edges. This model encodes all information about the semantic graphs, but disregards their structure.

4. Graph Neural Network (GNN) — To judge the effect of the gated graph neural architecture, we also include a model variant that does not use the gating mechanism and directly computes the hidden state as a combination of the activations (Eq 1) and the previous state.

5. Gated Graph Neural Network (GGNN) — We use the GGNN to process semantic parses, as described in Section 3.2. This model encodes all information from semantic graphs, including their structure, into a vector representation. To encode the question we use the same DCNN model (see Section 3.1).

The defined baselines use either manual features to capture the structure of the semantic graph (STAGG), a simple pooling mechanism (Pooled Edges) or disregard the structure completely (Single Edge). The two graph models (GNN and GGNN) make the full use of the graph structure of a semantic parse and encode it with graph neural networks. With this set-up, we are able to demonstrate what effect different levels of inclusion of the graph structure into the model have on the final performance for KB QA. We do

not include the published models of Berant et al. (2013) and Reddy et al. (2014) in the comparison since they were trained on Freebase and are not compatible with Wikidata. We use the more recent STAGG approach to position the graph models against the previous work and the feature-based models.

4.3 Training the model

To train the model, we need positive pairs of questions and semantic graphs. Since WebQSP does not contain semantic parses for Wikidata, we use weak supervision as suggested in Berant et al. (2013). Specifically, we follow Yih et al. (2015) and run our semantic graph construction procedure to create training instances (see Section 2.2). We use the state-of-the-art S-MART entity linking system for noisy data (Yang and Chang, 2015) to extract a set of entities $\mathcal{F}$ from each question.[6] Instead of scoring the semantic graphs with the model, we evaluate each graph against Wikidata and compare the extracted answers to the manual answers in the data set. The semantic graphs that result in a correct answer are stored as positive training instances and the rest of the graphs generated during the same process are used as negative instances. Due to the differences between Freebase and Wikidata, some question can not be answered exactly using Wikidata. We generate positive semantic graphs for 1945 questions out of 2880 (see Table 1) and put 628 as a development set aside.

Practical considerations At each training epoch, we take all positive semantic graphs and up to 100 negative graphs per question. We optimize the maximum margin loss function: $\mathcal{L} = \sum_{g \in C} \max\big(0, (m - \gamma(\mathbf{v}_q, \mathbf{v}_g^+) + \gamma(\mathbf{v}_q, \mathbf{v}_g^-))\big)$, where C is a set of semantic parses for the given question. From the loss function, we compute updates for the GGNN and the DCNN parts of the model.

All models are trained using the Adam optimizer (Kingma and Ba, 2014) with a batch size of 64. We use an early stopping criterion on the development data to determine the number of training epochs. The learning rate is fixed to 0.001 and the other optimization parameters are set as recommended in Kingma and Ba (2014): $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-08$. We apply Dropout (Srivastava et al., 2014) at every fully-connected layer as well as on the embeddings layers. We determine the hyper-parameters, such as the size of the hidden layer, with the random search on the development set (see Table 2). On the 1945 training questions, the GGNN model has usually finished training in under two hours on a single GPU.

	WebQSP-WD (train)	WebQSP-WD (test)	QALD-7
questions	2880	1033	80
complex questions	419	293	42
avg. # of relations per question	1.35	1.39	2.13

Table 1: Dataset statistics

Parameter	Tested values	Selected
Hidden layer size	$[32, 64, 128, 256, 512]$	256
CNN filter size	$[32, 64, 128, 256, 512]$	256
Dropout ratio	$\mathcal{U}(0.0, 0.5)$	0.2
# of CNN layers	$[1, 2, 3]$	2
Pooling operation	$[\text{avg}, \text{sum}, \max]$	max

Table 2: Optimized hyper-parameter values

4.4 Inference

We take the steps described in Section 2.2 to construct possible semantic graphs for a question at inference time. For WebQSP-WD, we use the entities produced by S-MART (Yang and Chang, 2015) to start the graph construction. On QALD-7, we use the annotated entities provided by the Shared Task organizers.

Each question and semantic graph are encoded into fixed-size vector representations and the reward function γ is used to score the graphs. The highest scoring graph is used to retrieve the answers from Wikidata. Given the iterative nature of our semantic graph construction procedure, we adopt beam search to speed up the computation. We score the graphs after each step and select the top 10 to proceed.

[6] S-MART is not openly available, but the output on the WebQuestions dataset was made available by Yih et al. (2015): https://github.com/scottyih/STAGG

	P	R	F
STAGG	0.1911	0.2267	0.1828
Single Edge	0.2240	0.2713	0.2148
Pooled Edges	0.2094	0.2553	0.2032
GNN	0.2419	0.2890	0.2326
GGNN	**0.2686**	**0.3179**	**0.2588**

Table 3: Results on the WebQSP-WD test set

	P	R	F
STAGG	0.1934	0.2463	0.1861
Single Edge	0.2173	0.1963	0.1951
Pooled Edges	0.1904	0.1800	0.1605
GNN	0.1652	0.2072	0.1703
GGNN	**0.2176**	**0.2751**	**0.2131**

Table 4: Results on the QALD-7 data set

5 Results

We follow the previous work (Berant et al., 2013) and use precision, recall and F-score to evaluate the models. The measures are computed for each individual question and then macro-averaged. This ensures a fair evaluation, since a system might provide a partially correct answer that is nevertheless better than a complete miss. We compare the graph models to the baselines including the previous state-of-the-art STAGG architecture.[7]

5.1 WebQSP-WD

We compare the results on the WebQSP-WD data set in Table 3. As can be seen, the graph models outperform all other models across precision, recall and F-score, with GGNN showing the best overall result. We confirm thereby that the architecture that encodes the structure of a semantic parse has an advantage over other approaches. To validate the results, we have re-trained the model with different random seeds and observed little variance in the results (F-score, $\sigma = 0.0205$).

The STAGG architecture delivers the worst results in our experiments, the main reason being supposedly that the model had to rely on manually defined features that are less flexible. The Single Edge model outperforms the more complex Pooled Edges model by a noticeable margin. The Single Edge baseline prefers simple graphs that consist of a single edge which is a good strategy to achieve higher recall values.

Since our main goal is to produce better encoding of semantic graphs, we break down the performance of the evaluated models by the number of relations that are needed to find the correct answer.[8] In Figure 8, we see that for the STAGG and Single Edge baselines the performance on more complex questions drops compared to the results on simpler questions. The Pooled Edges model maintains a better performance across questions of different complexity, which shows the benefits of encoding all graph edges. Looking at Figure 8 we also get a further insight into the difference between the Single Edge and the Pooled Edges models. These two models achieve almost identical results on the simple questions, which is to be expected since these models are equivalent when the number of edges is 1. On other questions, the Single Edge baseline performs mostly under the Pooled Edges model, but significantly outperforms it on the questions that need 4 edges to get the correct answer.

We see that the GGNN model offers the best results both on simple and complex questions, as it effectively encodes the structure of semantic graphs. The performance of the model drops for the most complex questions (# of edges ≥ 4). That does not happen for the GNN model variant without the gating mechanism. We conjecture that this happens because GGNN has more parameters than GNN and therefore needs more data to learn. By looking at the model errors on the most complex questions, we could see that GGNN tends to incorrectly predict one of the relations in the graph, which results in a wrong answer. GNN, on the other hand, more often predicts less relations that are needed and therefore gets a non-zero score for a partially correct answer. For example for the question "Who is the prime minister of Spain 2011?", GNN predicts a graph of two relations INSTANCE OF(human) and HEAD OF GOVERNMENT which returns a list of all Spanish prime ministers. The complete correct graph would also include temporal constraints.

[7] Since we use Wikidata and WebQSP-WD, the values reported in this work are not directly comparable to those for Freebase.

[8] We use again the manually constructed queries provided by Yih et al. (2016) to estimate it.

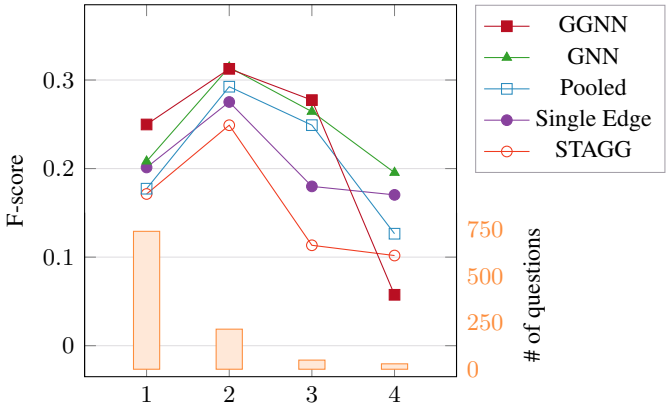


Figure 8: Evaluation results by # of relations needed to find the correct answer (bars show # of questions)

	Pooled	GGNN
F-score > 0.5	22.27%	28.37%
F-score > 0.0	29.82%	37.08%
Entity linker errors	6%	8%
No path to answer	14%	18%
Data set inconsistency	8%	14%
Wrong prediction	72%	60%
hit@10	35.62%	44.05%

Table 5: Manual error analysis results on WebQSP-WD

Notably, GGNN also has the best performance on the most simple semantic parses that only have one edge. In these cases, two nodes in the graph interact with each other through the single edge in both directions. The GGNN is better at capturing this interaction than other models.

5.2 QALD-7

Next, we examine the out-of-domain results on the QALD-7 data set for the five models (see Table 4). The difference in performance is less prominent on this data set, but we can observe the same trends. The Single Edge model outperforms both the STAGG and Pooled Edges baselines. The GGNN delivers the best performance, although overall the best result is worse than on the WebQSP-WD data set. QALD-7 is much smaller, but also more complex on average (cf. the average number of edges needed to find the correct answer in Table 1). Overall, we can conclude that the explicit modeling of the structure of semantic graphs with GGNN results in a better performance both in-domain on WebQSP-WD and out-of-domain on QALD-7 that was only used for evaluation.

5.3 Error analysis

To better understand the difference between the approaches that encode graph structure and the approaches that disregard it, we closely look at the output of GGNN compared to the Pooled Edges model. The first two rows in Table 5 show how often the respective model has returned an answer with F-score higher than 0.5 which is a mostly correct answer and how often it returned an answer that was not just completely wrong (F-score > 0.0). We see that GGNN delivers an acceptable answer almost 25% more often than Pooled Edges model, but there is still a lot of questions that are not answered correctly.

We have manually analyzed 100 sample answers from the two models where the resulting F-score was lower than 0.5 (see rows 3 through 6 in Table 5). Since GGNN makes less mistakes in general, the error propagation from the entity linking takes a slightly larger portion of the final error count. In 18% of the cases, it was impossible to find the answer in Wikidata since there were no path between entities in the question and the answer. For example, for the question "Where did Harper Lee attend high school?", the correct answer "Monroe County High School" is a valid entity in Wikidata, but it is not connected to "Harper Lee" via the EDUCATED AT relation. 14% of the time, the data set contained an inconsistent answer and even though the model has predicted the correct semantic graph, the answers did not match. For example, a correct answer for a question about someone's place of birth is usually a city or a town, yet for a smaller set of questions a city borough (e.g. Manhattan) or a country are listed in the data set.

Overall, in 32% of the cases the error was caused by the gap between the KB and the data set. This lets us put the current results into a perspective with the previously reported numbers for the Freebase KB. If we approximately adjust our results for this kind of errors, we achieve between 0.469 and 0.51 F-score. Reddy et al. (2016) report results for various approaches ranging from 0.404 to 0.503 F-score on the original WebQuestions data set that is a superset of WebQSP-WD (see Section 4.1).

The majority of errors for both models are caused by wrong predictions (row 6 in Table 5). GGNN selects significantly less wrong semantic graphs and more often successfully handles graphs with multiple edges. For example, for a question "What language do people speak in Brazil?", the GGNN model correctly predicts the graph with two edges HOME COUNTRY and NATIVE LANGUAGE to get a list of all languages that are spoken in Brazil. Whereas the other models either select the relation OFFICIAL LANGUAGE that returns only the official language of the country or choose a wrong interpretation altogether. We also look at the hit@10 measure that shows how often the correct semantic graph was in the top 10 scored graphs by the model (row 7 in Table 5). Notably, in 44% of the cases the correct semantic graph was still among the top scored graphs for the GGNN model.

6 Related work

We have focused on the problem of the increasing error rates for complex questions and the encoding of the semantic graph structure. In this paper, we describe a semantic parsing system for KB QA and follow the approach of Yih et al. (2015) who do not rely on syntactic parsing to construct semantic parses. Our semantic graphs do not cover some aspects of the first-order logic, such as negation. Reddy et al. (2016) define a semantic parser that builds first-order logic representations from syntactic dependencies. They further specify how it can be extended with negation in Fancellu et al. (2017).

We only train on the WebQSP-WD data set and we note that more data might be necessary to effectively train the gated graph architecture. Reddy et al. (2014) suggest an unsupervised learning method to learn a model from a large web corpus, while Su et al. (2016) use patterns and crowdsourcing to create new data with specific properties. These techniques can be used to further improve the performance of our model.

An alternative solution to semantic parsing is to build an information extraction pipeline that views the question as a query and the KB as a source of relevant information (Yao et al., 2014). Dong et al. (2015) and Jain (2016) construct a vector representation for the question and use it to directly score candidate answers from the KB. However, such approaches are hard to analyze for errors or to modify with explicit constraints. For example, it is not directly possible to implement the temporal sorting constraint (argmax).

We apply GGNNs to the problem of semantic parsing. Li et al. (2016) have developed the gated architecture based on the graph neural network formulation of Scarselli et al. (2009). Recently, a slightly different design of Graph Convolutional Networks was proven effective on a KB completion task (Schlichtkrull et al., 2018). Kipf and Welling (2017) introduced Graph Convolutional Networks, while Marcheggiani and Titov (2017) employed them for natural language processing for the first time and compared them to other formulations. Graph Convolutional Networks have a similar gated architecture and share most of the same properties with the Gated Graph Neural Networks used.

7 Conclusions

In this work, we have used Gated Graph Neural Networks to encode the structure of the target semantic parse for KB QA. We have shown that disregarding the semantic structure leads to a falling performance on questions that require complex semantic parses to get the correct answers. Our GGNN architecture was able to successfully model the structure of semantic parses. We have compared the performance of GGNNs against the previous work and non-graph models on two data sets and have broken down the results by question complexity. The analysis has shown that the suggested graph architectures do not have the same drop in performance on complex questions and produce better overall results.

Recently, Peng et al. (2017) and Yu et al. (2017) have attempted to incorporate entity linking into a feature based QA model. In the future, we plan to follow their work and integrate GGNNs with entity linking into a single model. We also see possible applications for GGNNs on other semantic parsing tasks, such as text comprehension.

Acknowledgments

This work has been supported by the German Research Foundation as part of the Research Training Group AIPHES (grant No. GRK 1994/1), and via the QA-EduInf project (grant GU 798/18-1 and RI 803/12-1). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X GPU.

References

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-Based Question Answering with Knowledge Graph. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 2503–2514, Osaka, Japan.

Jonathan Berant and Percy Liang. 2014. Semantic Parsing via Paraphrasing. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1415–1425, Baltimore, MD, USA.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1533–1544, Seattle, WA, USA.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation Kyunghyun. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question Answering over Freebase with Multi-Column Convolutional Neural Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 260–269, Beijing, China. Association for Computational Linguistics.

Federico Fancellu, Siva Reddy, Adam Lopez, and Bonnie Webber. 2017. Universal Dependencies to Logical Forms with Negation Scope. In *Proceedings of the Workshop "Computational Semantics Beyond Events and Roles"*, pages 22–32, Valencia, Spain. Association for Computational Linguistics.

Sarthak Jain. 2016. Question Answering over Knowledge Base using Factual Memory Networks. In *Proceedings of the NAACL Student Research Workshop*, pages 109–115, San Diego, CA, USA.

Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization.

Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *In Proceedings of the 5th International Conference on Learning Representations (ICLR)*, pages 1–14, Toulon, France.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2016. Gated Graph Sequence Neural Networks. In *In Proceedings of the 4th International Conference on Learning Representations (ICLR)*, pages 1–19, San Juan, Puerto Rico.

Percy Liang. 2016. Learning Executable Semantic Parsers for Natural Language Understanding. *Communications of the ACM*, 59(9):68–76.

Diego Marcheggiani and Ivan Titov. 2017. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1506–1515, Copenhagen, Denmark. Association for Computational Linguistics.

Haoruo Peng, Ming-Wei Chang, and Wen-Tau Yih. 2017. Maximum Margin Reward Networks for Learning from Explicit and Implicit Supervision. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2358–2368, Copenhagen, Denmark. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale Semantic Parsing without Question-Answer Pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, M. Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne vanden Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Tordai Anna, and Mehwish Alam, editors, *The Semantic Web*, pages 593–607, Cham. Springer International Publishing.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and Accurate Entity Recognition with Iterated Dilated Convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2670–2680, Copenhagen, Denmark.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gur, Zenghui Yan, and Xifeng Yan. 2016. On Generating Characteristic-rich Question Sets for QA Evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 562–572, Austin, Texas.

Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Bastian Haarmann, Anastasia Krithara, Michael Röder, and Giulio Napolitano. 2017. 7th Open Challenge on Question Answering over Linked Data (QALD-7). In Mauro Dragoni, Monika Solanki, and Eva Blomqvist, editors, *Semantic Web Challenges*, pages 59–69, Cham. Springer International Publishing.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A Free Collaborative Knowledgebase. *Communications of the ACM*, 57(10):78–85.

Yi Yang and Ming-Wei Chang. 2015. S-MART: Novel Tree-based Structured Learning Algorithms Applied to Tweet Entity Linking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 504–513, Beijing, China.

Xuchen Yao, Jonathan Berant, and Benjamin Van Durme. 2014. Freebase QA: Information Extraction or Semantic Parsing? In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 82–86, Baltimore, MD, USA.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 1321–1331, Beijing, China.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The Value of Semantic Parse Labeling for Knowledge Base Question Answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 201–206, Berlin, Germany.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved Neural Relation Detection for Knowledge Base Question Answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 571–581, Vancouver, Canada.

Rethinking the Agreement in Human Evaluation Tasks (Position Paper)

Jacopo Amidei and **Paul Piwek** and **Alistair Willis**
School of Computing and Communications
The Open University
Milton Keynes, UK

Abstract

Human evaluations are broadly thought to be more valuable the higher the inter-annotator agreement. In this paper we examine this idea. We will describe our experiments and analysis within the area of Automatic Question Generation. Our experiments show how annotators diverge in language annotation tasks due to a range of ineliminable factors. For this reason, we believe that annotation schemes for natural language generation tasks that are aimed at evaluating language quality need to be treated with great care. In particular, an unchecked focus on reduction of disagreement among annotators runs the danger of creating generation goals that reward output that is more distant from, rather than closer to, natural human-like language. We conclude the paper by suggesting a new approach to the use of the agreement metrics in natural language generation evaluation tasks.

1 Introduction

Human evaluations are broadly thought to be more valuable the higher the inter-annotator agreement (IAA). In natural language processing (NLP), IAA is often viewed as a means of assessing the quality of data on a task, in particular, the reliability. Also, it helps to identify poor annotations, provide information on where annotation guidelines can be improved, and establish an upper bound on system performance. Nevertheless some studies, for example (Sampson and Babarczy, 2008), (Kovář 2016) and (Kovář et al., 2016)[1], propose IAA limitations when used in the annotation of natural language generation (NLG)[2], where annotation usually concerns the quality of a generated sentence – including a range of features such as syntax, semantic and pragmatics. Indeed, natural language brings in itself a variability which cannot be reduced, except by weakening its expressive power. For instance, the perception of sentence idiomaticity can change from person to person because of styles, educational and regional difference (for example differences between British and American English for English sentences). Far from being a problem this shows the richness, variability and beauty of human languages.

In this paper we are going to study IAA in the area of Automatic Question Generation (AQG)[3]. We performed experiments which show that there are factors inherent to question annotation that place an upper-bound on IAA. We found subjective bias that cannot be removed by improving the annotation

[1] It is also worth mentioning Martinez Alonso et al. (2016), where the authors use the systematic disagreement in order to train their model. Martinez Alonso et al. raise the discussion about how to distinguish and use potentially relevant from irrelevant disagreement. Although this problem is very interesting and deserves a deep analysis, it goes beyond the scope of the present paper.

[2] In what follows, when we refer to NLG tasks, we consider tasks that require generation of human language and whose possible output space is, as a result, very wide (given that a natural language often allows the same information to be expressed in many different ways). Input may vary, and range from text and data to images. Examples of these are: dialogue generation, question generation, image caption generation, narrative generation, and so on.

[3] Although we focused on the AQG task, we believe the causes we discovered behind annotators' disagreement are more general, and they are likely to apply to different NLG tasks.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3318–3329
Santa Fe, New Mexico, USA, August 20-26, 2018.

guidelines, unless we completely distort the goal of producing human language variability[4]. Our empirical study provides new evidence in strong support of the claim that high IAA is not always desirable. This work shows that for AQG tasks, a low IAA in the evaluation phase is part of the nature of language. More generally, the experimental results suggest that in NLG the ability to automatically generate sentences as similar as possible to human generated ones seems to go in the opposite direction from the achievement of a high IAA evaluation score. Although the Kappa coefficients agreement, normally used for measuring IAA, takes into account some level of randomness introduced by annotators, it might be that the Kappa scales of interpretation used to check IAA are not generally adequate for NLG purposes. Accordingly, we propose to rethink the agreement in human evaluation tasks.

2 Related work and preliminary concepts

Since Carletta's (1996) paper, which introduced the use of the kappa statistic in NLP, several papers have studied the IAA in NLP tasks. Among those, we now describe the most relevant to our work.

Bayerl and Paul (2011) investigate different factors which impact the IAA score, performing a meta-analytic investigation that involves 96 annotation studies. From their analysis, the authors conclude that at least seven factors are determinant in affecting the IAA values. These factors are: "annotation domain, number of categories in a coding scheme, number of annotators in a project, whether annotators received training, the intensity of annotator training, the annotation purpose, and the method used for the calculation of percentage agreements". They conclude their study giving some suggestions to decrease annotator disagreement. Ng et al. (1999) study IAA in the field of word sense disambiguation. The authors found that we can increase the IAA by collapsing sense classes. Indeed, they report that, for the 53 nouns they examined, reducing the nouns' senses from an average of 7.6 per noun to an average of 4, raises the average Kappa score (computed as an average of the Kappa value of the individual nouns) from 0.463 to 0.862. In order to improve the evaluation IAA of the task B defined in the Shared Evaluation Task Challenge (QG -STEC) (Rus et al., 2012), Godwin and Piwek (2016) define an interactive process where the annotators can discuss their opinions about the criteria used in the evaluation. At the end of the evaluation process, repeated three times with three annotators, they achieved high IAA with a pick of 0.94 for one of the five criteria used in the evaluation. In recent work, Novikova et al. (2018), use a rank-based magnitude estimation methodology in order to improve the agreement of human judgements.

The papers we have just presented share the assumption that a high IAA is a desirable goal and they study some ways to increase it. In this paper we reject the assumption that arbitrarily high IAA is always desirable or obtainable. Our paper is more in line with the direction of Sampson and Babarczy (2008), who investigate the upper bounds that can be achieved on IAA. More precisely "the limits to the potential precision of English grammar annotation" are studied. Sampson and Babarczy perform their experiment using the SUSANNE scheme (Sampson, 1995) for a parse-tree structure task. From their study they conclude that discrepancy in IAA emerges for three main reasons: A) violation of an explicit feature of the annotation scheme; B) the lack of a single, unambiguous annotation decision yielded by the scheme, even though the meaning of the text is clear; C) structural ambiguity in the text.

A really interesting survey for understanding of the IAA in NLP is presented by Artstein and Poesio (2008), who give an excellent analysis of the K agreement measures. The authors discuss the mathematics and interpretation of these coefficients and their use in several computational linguistic tasks.

Automatic Question Generation task definition: Automatic question generation is the task "*of automatically generating questions from various inputs such as raw text, database, or semantic representation*" (Rus et al., 2008). This definition, adopted by the AQG community, leaves room for researchers to decide which kind of questions and which kind of input to work with. Following Piwek and Boyer (2012), we can characterize a specific AQG task as relying on three features: what the input is, what the output is and finally the relation between the input and the output.

[4]Since human evaluations are valuable tools for generative system developers — indeed the evaluation results can be used for error analysis — artificial restrictions on annotation schemes can bias system developers towards ignoring important aspect of human language. This problem can be more accentuated if the annotation guidelines are used in order to create a corpus for training purposes.

Given the wide variety of AQG tasks we must specify the task we are interested in to perform our experiments. Regarding the output, we are looking for interrogative English sentences. Having defined the output, we can now move to specify the input and the input/output relationship. As an input we consider English text, where the relation can be described by the following situation: the output question is answered by the input text, that is, the questions are about the input information. In this task the AQG systems, given a text T as an input, generate a question which can be used, for instance, to verify the respondent's knowledge about T. We will give some examples in section 4. For the sake of simplicity, in what follows we will refer to this task as text2text. To our knowledge, at the moment the text2text task is the most studied task in the AQG community.

Gold standard evaluation approach: As a generation task, whose possible output space is very large, text2text faces the problem of the absence of a comprehensive gold standard. A gold standard, usually obtained through human annotations, is thought to be the correct set of solutions for a particular task T, and it is used either for training or evaluated models for T's purpose. Given a task, the gold standard evaluation approach is based on the assumption that we can always define a precise and exhaustive (or near exhaustive) set of outputs for the task.

In NLG the gold standard is a set of natural language words, sentences or more generally texts. Given the huge human ability to use many different expressions to reach the same communicative goal, language generation tasks cannot safely rest on the gold standard approach for evaluation purposes.

In NLP the gold standard approach has been used for evaluation through automatic metrics, which compare the automatic generated sentences against the gold standards. In the last few years many studies in NLP have shed light on the correlation between human judgement and metrics such as BLEU (Papineni et al., 2002)[5]. The results, which have shown this correspondence to be weak[6], cast doubt on the feasibility of use of these metrics to evaluate the overall model quality. Indeed, a model may generate very high quality sentences that are different from the ones in the gold standard set used for evaluation — this is especially possible if we train and evaluate a model with different corpus text.

For the text2text task, this problem can be expressed in the following way: Given a text T it is possible to generate many different correct questions about T. More specifically, given a text T with an answer A relative to T, it is possible to generate different correct questions answered by A. In both cases the questions generated, although good, can be very different from the gold standards previously chosen, and so get a low ranking from automatic metrics. For NLG tasks, the idea of some platonic gold standard which, given a task t, can provide an ideal solution for t, is unrealistic.

As we will see soon, the difficulty of having a platonic gold standard is not just a problem for automatic evaluation metrics but it is also reflected in the difficulty of reaching a high human evaluation agreement.

Annotation guidelines: The absence of a comprehensive gold standard evaluation approach is reflected in the difficulty of formulating precise annotation guidelines. These are either defined in order to annotate corpus data or to give instructions for human evaluation tasks.

The annotation guidelines accompany a particular annotation scheme[7] , and should strictly define the features that the humans have to annotate, as well as how they should be annotated. The goal is that of reducing the annotators' subjective interpretation so as to have a consistent corpus. Indeed, human annotations are prone to subjective bias, so the guidelines have the aim of reducing this subjectivity to make the annotation process more reliable. Although we do not have a standard way to create annotation guidelines, some common ground rules have been defined. Usually a sound annotation guideline introduces at least some criteria which define the annotation task, alongside a description and some examples whose aim is to help the annotators to understand the criteria (see for example, Palmer and Xue (2010) and Pustejovsky and Stubbs (2013)). In a language generation task, where the possible space of outputs cannot be strictly defined, as it is subject to individual human differences, the guideline definition be-

[5]For an exhaustive list of automatic metrics used in NLP we refer to (Gatt and Krahmer, 2017) page 126.

[6]For an in depth discussion about this point we refer again to (Gatt and Krahmer, 2017), especially the section 7.4.1 and the reference there presented.

[7]An annotation scheme determines the conceptual content of the annotation task, for example by identifying the set of legitimate alternatives the annotator can choose from.

comes particularly difficult and there is a high risk of ending up with a set of poorly defined criteria. Far from there being a lack of precision in writing the guideline, here the problem seems to be intrinsic to the nature of human language. A fascinating analogy that aims to explain this point, presented by Geoffrey Sampson in the online page https://www.grsampson.net/RSue.html, is the following:

> Suppose we wanted to be able to say how large particular clouds are – what volume of space they occupy. Clouds are fuzzy things, so one problem would be what we mean by the volume of a cloud – what exactly should we count as its edge? But even if we adopted some precise definition of cloud boundaries, so that it became meaningful to say that this cloud is exactly N cubic yards in size, not N + 1 or N - 1, it might still be beyond mankind's abilities actually to measure clouds so exactly.

As a result of the fuzziness of human language, the criteria defined in an annotation guideline can be quite vague and be left to annotators' interpretation, which can result in a low IAA.

Kappa coefficients for IAA: The IAA measures the agreement between annotators. Thanks to this measurement we can estimate the annotation reliability. Generally, it is believed by the NLP community that the annotations are reliable if annotators agree, to a certain extent, on the category assigned. The kind of extent is determined by the method chosen to measure the agreement. Usually the IAA is performed by Kappa coefficients K[8].

The common theme to a variety of formulations is that K corrects annotators' agreement by the expected chance agreement:

$$K = \frac{P(A) - P(E)}{1 - P(E)}$$

where $P(A)$ is the proportion of times the annotators agree, whereas $P(E)$ is the proportion of times the annotators would be expected to agree by chance.

Kappa coefficients are used with Kappa scales of interpretation. For example, Krippendorff (1980) considers good any data annotation with agreement in the interval [0.8, 1], tentative any data annotation where agreement is in the interval [0.67, 0.8] and to discard any data annotation where agreement is below 0.67 [9]. Given the arbitrary nature of these choices of number, we can find different scales for K interpretation (see for example Landis and Koch (1977) , Green (1997) and for a general discussion Artstein and Poesio (2008)).

Since Carletta (1996), the K coefficients and its scales interpretation have been introduced in the area of NLP. Carletta took these metrics from content analysis in order to "compare results in a standard way across different coding schemes and experiments and to evaluate current developments" specifying that "whether we have reached (or will be able to reach) reasonable level of agreements in our work as a field remains to be seen".

3 The case of Automatic Question Generation

Problem statement: Human language generation tasks, given the richness of human languages, can usually not be evaluated with respect to some platonic gold standard. Given a sentence, different persons can have conflicting opinions about the sentence's ability to fulfill the task for which it was generated. More generally, given a sentence S, different persons can have conflicting opinions about the fluency, the grammaticality, the naturalness, the elegance etc. of S. These discrepancies are reflected, firstly, in the difficulty of generating strong annotation guidelines, and secondly, in the difficulty of reaching

[8] Here we are following the notation used in Carletta (1996). It is worth notice that the K formulation presented catch the most used agreement metrics from the NLP community, such as the π coefficient of Scott (1955) its Fleiss' generalization (Fleiss, 1971), and the k coefficient of Cohen (1960). The α coefficient of Krippendorff (1980) is expressed in a similar way but in term of disagreement. This difference is not relevant to the aim of this paper, so in what follows when we use the term$K$ coefficient we are referring at some instance of the above mentioned metrics. We refer to (Artstein and Poesio, 2008) for an excellent survey of the several agreement coefficients present in the literature.

[9] Arstein and Poesio (2008) note that: *the description of the 0.67 boundary in Krippendorff (1980) was actually "highly tentative and cautious," and in later work Krippendorff clearly consider 0.8 the absolute minimum value of α to accept for any serious purpose: "Even a cutoff point of α = .800 ... is a pretty low standard"* (page 576).

a satisfactory IAA in the evaluation phase — where what is a satisfactory IAA is defined by some K interpretation scales. Given the nature of the tasks in play, we believe it is time to rethink the IAA in the NLG community. Far from being a problem, a low IAA score reflects the richness, variability and beauty of human languages.

IAA in text2text task: A careful examination of the text2text literature shows that the human evaluation IAA, where reported, is usually below 0.6[10]. This means, following the Krippendorff Kappa interpretation scale, that the evaluations are to be considered inadequate for a satisfactory analysis.

It is also worth noticing the absence of shared annotation guidelines in the area. Indeed, we can find a large variety of criteria used in the human evaluations alongside more or less detailed evaluation guidelines. Although this absence is a deficiency in the area, which makes it hard to have a uniform way to check the quality across generation systems, we cannot completely ascribe to it the low IAA reached in the evaluation phases.

About this point we can find an analysis by Sampson and Babarcz (2008). As we said before, the authors perform their experiment using the SUSANNE scheme (Sampson, 1995), which is an annotation guideline of 449 pages. Of the three conclusions they reach in the study, the most interesting to the aims of this paper is the following: "while it is not easy to define watertight boundaries between the categories of a comprehensive structural annotation scheme, limits on inter-annotator agreement are in practice set more by the difficulty of conforming to a well-defined scheme than by the difficulty of making a scheme well defined".

It is worth noting that Sampson and Babarczy's experiment was done with two experts. Despite the fact that the annotation guidelines were very precise and the annotations were done by experts, divergent rank opinions still emerged from the experiment.

Experiments description: The current paper was motivated by our attempt to define annotation guidelines for the text2text task. The methodology we used was that of refining the criteria chosen through several iterations of discussions and pilot evaluations. During these iterations we noticed that regardless of how many changes we made, there remained a divergence in the judgements that we could not reduce by modifying the guidelines.

We started by choosing the criteria to use. We decided to study the evaluation along two dimensions, a linguistic one, which aimed to evaluate the question quality from a grammatical and idiomatic point of view, and a task-oriented one, which aimed to evaluate how well the questions fulfill the task for which they were generated[11]. In the text2text case the task is that of generating a question in relation to a given paragraph. So in the task-oriented dimension we were interested in the degree to which the question was answered by the input text. Although these two dimensions are tacitly shared between researchers, making them explicit has the merit of outlining a common space to check quality across generation systems. Within this dimension we defined four criteria, which we will present in the next section. We aimed for a sound description of each criterion, accompanied by examples. Once we defined the first version of the annotation guideline we tested it by performing a pilot assessment phase.

We tested our evaluation guidelines taking random input paragraphs and questions from the SQuAD dataset (Rajpurkar et al., 2016). We repeated this process three times. In each iteration we took five paragraphs and for each paragraph six questions. All the questions were human generated. Between these, three were taken from the reference questions of the input paragraph (for one of these we automatically swapped some words in order to change the grammaticality), and three were about the topic of the input paragraph, although generated with a different (but with similar topic) paragraph as an input. We asked two volunteers to join us in the process of annotating the questions. Both volunteers were

[10]To our knowledge the only significant exception is (Godwin and Piwek, 2016), where, as we said before, the authors define an interactive process in which the annotators can discuss their opinions about the criteria used in the evaluation. Such a process presupposes that the annotators talk with each other in order to profitably improve the criteria in play. In this we can see at least two difficulties. On the one hand, this process is time consuming and is not practicable in Crowdsourcing evaluations. On the other hand, although the process allows a small set of people to agree among themselves, we could have different results with different groups of people.

[11]Gatt and Krahmer (2017) note that traditionally, in NLP, the linguistic dimension is defined by the *fluency* or *readability* criteria, whereas the task-oriented one is defined in term of *accuracy*, *adequacy*, *relevance* or *correctness* criteria.

Spanish, who had lived several years in the UK or US, but had not formally studied linguistics. Each pilot was performed with four people: two of the authors of this paper and the two volunteers. At the end of each iteration, with the pilot example support, the authors of the present paper discussed the adequacy of the criteria. Based on that discussion we decided how to improve description of and examples for the criteria.

After several iterations we concluded with a final iteration. This time seven annotators, including the two of the authors of this paper, engaged in the annotation task. Between the seven annotators three were native speakers of English. The other five were proficient in English. Three of the annotators had a background in linguistics. Once again we used the input paragraphs from the SQuAD dataset. This time we took two paragraphs and five question for each paragraph. One of the questions was automatically generated by the question generation algorithm freely available at the page https://kjmazidi.pythonanywhere.com/. Three were human generated questions from the SQuAD dataset: two were taken from the reference questions of the input paragraph (for one of these we automatically swapped some words in order to change the grammaticality), and one was about the topic of the input paragraph, although generated with a different input paragraph. The last question was generated by the first author of this paper, who did not personally take part in the evaluation. At the end of the evaluation, the annotators discussed their responses for each criterion and for each question, one by one. During this discussion, we identified a series of differences. As we will see in the next sections, they were based on annotators' subjective taste and experiences. Again, for each criterion, we got a IAA lower than 0.67, with the lowest score being 0.11.

4 A new taxonomy of divergences

We classify the main sources of disagreement we found through our experiments in five categories: i) *Style and taste*; ii) *background knowledge*; iii) *personal assumptions*; iv) *use of common sense inferences*; v) *attention to detail*.

We are going to present some examples for each category in order to explain them. As we said before, we conducted the study along two dimensions, a linguistic one and a task-oriented one. The annotation criteria attempted to characterise these two dimensions. Following the process we described in the previous section, we attempted to improve the criteria by clarifying their descriptions, adding clarifying examples and splitting them into sub-criteria.

In the last version of our evaluation guidelines, we had the following four criteria: i) Pertinence, ii) Grammaticality, iii) Comprehensibility, iv) Fluency. The Pertinence criterion is for the task-oriented dimension. We used this name to underline the need for the question to be directly and strictly related to the paragraph. The linguistic dimension was split into three sub-dimensions or criteria: Grammaticality, Comprehensibility and Fluency. The grammaticality is intended to check possible grammatical errors in the questions. Comprehensibility aims to better frame the grammaticality judgement. Indeed, it can happen that a question, although ungrammatical, is perfectly understandable[12]. That is, it can be possible to work out what the question is asking, even if it is ungrammatical. Finally, the fluency criterion is intended to judge whether the question is idiomatic/natural. The Pertinence criterion is ranked on a scale from 0 to 3, whereas the other criteria use a binary scale. We decided to evaluate the linguistic dimension without providing the paragraph while the task dimension after reading the paragraph. Indeed, we believe that the grammaticality, comprehensibility and idiomaticity of a question can be judged independently of the paragraph from which it was generated. So each annotation was carried out in two stages. First, the annotator was presented with the question in isolation, and asked to provide a judgement on the questions of grammaticality, comprehensibility and fluency. Next, the annotator was presented with both the question and the input text, and asked to provide a judgement on the pertinence.

In our experiments, of the 100 questions we analysed, we found a total of 55 divergences in the linguistic dimensions and a total of 78 divergences in the task-oriented dimension. We present some examples of these divergences in order to explain how the above categories were delineated.

[12]We discuss some examples of this in the next section.

Style and taste: We found some divergences were related to the annotator's taste and their writing style. This kind of divergence emerged in the question on idiomaticity judgements. For example, we notice that American and British English differences played against question idiomaticity. Given the question:

Jean De Rely's illustrated French-language scriptures were first published in what city?

we found divergent judgements because the question sounded awkward to some British annotators, who preferred the use of "which" rather than "what". Similarly, we had divergent judgements with the question:

The adaptive immune system must distinguish between what types of molecules?

where one of the British annotators not only preferred the use of "which" instead of "what", but also marked the question as not idiomatically natural because he preferred the question written in the reverse order: Which types of molecules must the adaptive immune system distinguish between? In a case like this, we can see how the personal writing style influences the evaluation. This eventuality is a result of the fact that, in a generation task, we can use very different sentences, in this case questions, in order to reach the same communicative goal.

Background knowledge: Another issue we found concerned the amount of background knowledge that is needed in order to understand a question. Where annotators did not share relevant background knowledge, they often gave different judgements for a question's comprehensibility and pertinence. For example the question:

How many Time incarnations can a Lord have?

even if not grammatical (the original question is: How many incarnations can a Time Lord have?), was recognised as comprehensible by the annotators who were aware of the 'Doctor Who' television programme, whereas it was marked as not comprehensible by the ones who did not know the show. In the same way, given the paragraph:

Microorganisms or toxins that successfully enter an organism encounter the cells and mechanisms of the innate immune system. The innate response is usually triggered when microbes are identified by pattern recognition receptors, which recognize components that are conserved among broad groups of microorganisms...

the question:

What part of the innate immune system identifies microbes and triggers immune response?

raised divergent opinions for two reasons: on the one hand, it was necessary to know that microorganisms are the same as microbes. On the other, it was necessary to know that the pattern recognition receptors are part of the innate immune system. The lack of this knowledge from some annotators, resulted in a divergence in ranking the pertinence criterion. In these examples, we can see how the annotators' knowledge determines the evaluation. Indeed, personal knowledge and experiences are reflected in the way we generate and understand sentences. So, divergences in knowledge can be reflected in evaluation divergences.

Personal assumptions: Other divergences arose from different annotator assumptions. Also in these cases the divergences emerged predominantly in the question's comprehensibility and pertinence judgements. For example the question:

Which team finished the regular season?

was considered not comprehensible by the annotators who assumed that all teams would finish the season. These annotators considered the question meaningless. In contrast, other annotators considered a scenario in which some teams could not finish the season. In this case the question was marked as comprehensible. A similar problem was found with the question:

What was the win/loss ratio in 2015 for the Carolina Panthers during their regular season?

Given the paragraph:

The Panthers finished the regular season with a 15-1 record, and quarterback Cam Newton was named the NFL Most Valuable Player (MVP)...

one annotator noticed that the question can be marked pertinent under the assumption that the Carolina Panthers in the question and the Panthers in the paragraph are referring to the same team. He did not make this assumption. Other annotators did make this assumption and ranked the question as pertinent. In a similar way to background knowledge and experiences, personal assumptions are reflected in the way we generate and understand sentences. Again, divergences in assumptions can be reflected in divergences in evaluation.

Use of common sense inferences: This kind of divergence emerged predominantly in the question pertinence judgements. For example, given the paragraph:

The availability of the Bible in vernacular languages was important to the spread of the Protestant movement and development of the Reformed church in France. The country had a long history of struggles with the papacy by the time the Protestant Reformation finally arrived. Around 1294, a French version of the Scriptures was prepared by the Roman Catholic priest, Guyard de Moulin. A two-volume illustrated folio paraphrase version based on his manuscript, by Jean de Rély, was printed in Paris in 1487.

the question:

Jean De Rely's illustrated French-language scriptures were first published in what city?

was marked as not pertinent by one annotator who noted that the paragraph provides information about the place where Jean De Rely's scriptures were printed and not where they were published. To consider this question as unambiguously answered by the paragraph it is necessary to assume that the place where the Jean De Rely's illustrated scriptures were printed is the same where they were published. Although this inference goes much further than the information presented in the text, other annotators made it and marked the question as pertinent. Likewise, the question:

When did the first French language bible appear?

was marked as pertinent by one annotator, who inferred the answer 1294 from the paragraph. Other annotators considered the question as not pertinent because the answer cannot be correctly inferred from the paragraph. As a matter of fact, in the text there is no mention of the fact that the Guyard de Moulin version of the Scriptures was the first French language bible to appear and at the same time it is not possible to rigorously reach this conclusion by the information provided by the text. Here we faced another problem: people reason in different ways. Obviously this divergence also emerges in the way people generate, understand, and in this case evaluate, sentences.

Attention to detail: We noticed that in some cases, annotators overlooked some question details. Also in these cases, the divergences emerged predominantly in the question pertinence judgements. For example, given the paragraph:

The most impressive examples of rococo architecture are Czapski Palace (1712-1721), Palace of the Four Winds (1730s) and Visitationist Church (façade 1728-1761).

the question:

What type of architecture is the Palace of Four Windows an impressive example of?

was ranked as pertinent by some annotators who did not notice that the question uses the word "Windows" instead of "Winds", but it was ranked as not pertinent by the annotators who did notice this detail. Another example of this kind of rank divergences is the following. Given the paragraph:

Victorian farms produce nearly 90% of Australian pears and third of apples. It is also a leader in stone fruit production. The main vegetable crops include asparagus, broccoli, carrots, potatoes and tomatoes. Last year, 121,200 tonnes of pears and 270,000 tonnes of tomatoes were produced.

the question:

How many tonnes of tomatoes does Victoria produce?

was ranked as pertinent by some annotators who did not notice that the paragraph was speaking about "last year" production, whereas the question is more general (maybe it is asking for an annual average, which is not mentioned in the paragraph). Other annotators did notice this detail and ranked the question as not pertinent. This problem is linked to the fact the people can generate and interpret sentences to different levels of generality and detail. This is reflected in the way people understand sentences, in this case the questions, in the context from which they are generated.

We conclude this section by observing that together with the five sources of disagreement we have just presented, we found other sources of discrepancy between annotators — which we suppose are present in each annotation task — related to distractions, misunderstanding the guidelines, or forgetting how to apply the guidelines.

IAA, a new research direction: Many studies have shown IAA appearing to depend on several factors, for example: data typology, data ambiguity, the number of categories used in the annotation, the number of annotators, the use of expert or non-expert annotators, and annotators' attention, skills, memory and training. In the present study we found that, in a language generation task such as text2text, we also need to add the following factors: the annotators' background knowledge, their personal taste and personal assumptions, their attention to detail and their inferential skills.

The obvious conclusion we can draw is that human annotators' disagreements are part of the nature of language. Nevertheless, much effort has been spent on trying to understand the source of this disagreement in order that the disagreement can be reduced, for example by modifying the annotation scheme. According to our experiments, we believe that in NLG tasks, attempts to create annotation guidelines that strongly reduce disagreement among annotators, are in danger of missing the goal of producing human language variability. Indeed, such methods can artificially restrict the annotators' language interpretation and their opinion about sentences' goodness. Consequently, the methods do not catch the model's ability to generate language variability.

Rather than regarding the evaluation disagreement as a problem to be fixed, we suggest that it should be thought of as an ineliminable feature of generation tasks, which reflects the variety of human languages and its users. To this end we propose that the scales of K interpretation should be reconsidered,

for example by introducing confidence intervals. These would delineate boundaries within which we expect to find the IAA. An evaluation agreement that falls below the low boundary can suggest that the evaluation is too unreliable to be considered further. At the opposite end, an evaluation agreement that exceeds the upper boundary may suggest that the task was overconstrained and therefore possibly of limited interest. Such an interval, recognising that a greater level of disagreement between annotators—so that the level of disagreement falls inside the boundaries — is not problematic, would take into account the phenomena we have introduced in the previous section. Given a corpus to evaluate, different groups of annotators can end up with different levels of disagreement regarding the quality of that corpus. These differences, mainly due to how language is produced and understood, as long as they lie inside the interval boundaries, can be considered as inherent to the task. Although there is not a general way to determine these intervals — because of the range of tasks in NLG, such confidence intervals should be defined on a task by task basis — we believe a first step could be to analyse the correlation between differences in IAA and growing linguistic difficulty. Indeed, consider a situation where a generative system is evaluated on its capability to produce human-like language (e.g. as in a Turing test). In such a scenario, it is likely to get the best agreement in those cases were the generated language is somewhat simplistic (for example very shallow sentences or sentences clearly ungrammatical or completely out of context). Conversely, in the cases where the generated language is more complex from a semantic point of view, as in our experiments, it is likely achieve lower agreement. If, using high levels of linguistic sophistication, a similar level of agreement can be reached, then it is possible to consider that level as a bottom threshold. Similarly, it can be done with low levels of linguistic sophistication in order to choose a top threshold. We are undertaking some experiments in this direction for text2text tasks.

In the same vein, following the Shared Task and Evaluation Campaigns (STECs) proposal (see for example (Gatt and Belz, 2010)), we believe that for each NLG task, a common and shared evaluation guideline should be defined, maybe together with a corpus data used only for evaluation purposes. Furthermore, inside such a framework, researchers could also share, alongside their results, their methodologies and ideas to attempt to understand and classify any divergences between annotators. A better understanding of these divergences can indeed help us to understand and improve the tasks at hand. For each NLG task, a new IAA interpretation, alongside a shared evaluation guidelines, could bring in each NLG community a uniform way to check the quality across generation systems.

In order to better interpret IAA, instead of focusing exclusively on interannotor agreement, more attention may need to be paid to the internal consistency of the responses of each annotator. We have argued that while there are irreducible differences between annotators' judgements, the judgements of an individual annotator should be consistent, if they are to represent that annotator's particular linguistic abilities and preferences.

Conclusion: In this paper we investigated the IAA problem for a text2text task. We performed some experiments that show how a low IAA is inherent to the task of language annotation. Our experiments have outlined that factors such as: annotators' background knowledge, personal taste, personal assumptions, attention to detail and inferential skills are presents in evaluation judgments. We believe that these kind of factors are intimately connected to the people experiences and their language understanding and use, and cannot be effectively removed by the choice of annotation guidelines. We believe that, in generation tasks, the efforts of reaching a high IAA run the danger of leading research and system building away from the goal of generating text that is natural and human-like (and possibly towards artificial non-natural language). Indeed, it could not take into account the differences that make human language so broad in its meaning and use. For this reason, we suggest new research directions, for example the introduction of confidence intervals for the IAA interpretation and attention on internal annotator consistency. In our opinion such a shift in interpretation would take better account of the richness and variability in human language.

Acknowledgements

We warmly thank Erika Renedo Illarregi, Luisa Ruge, German Ruiz Marcos, Suraj Pandey, Simon Cutajar, Neil Smith and Robin Laney for taking part in the experiments and sharing with us opinions and

feedback. We would also thanks Karen Mazidi to give us the login access to her online Question Generator. We finally thanks the anonymous reviewers for their helpful suggestions.

References

Ron Artstein and Massimo Poesio. 2008. Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics*, 34(4): 555-596.

Petra S. Bayerl and Karsten I. Paul. 2011. What Determines Inter-Coder Agreement in Manual Annotations? A Meta-Analytic Investigation. *Computational Linguistics*, 37(4): 699-725.

Jean Carletta. 1996. Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22(2): 249-254.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1): 37-46.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5): 378-382.

Albert Gatt and Anja Belz. 2010. Introducing Shared Tasks to NLG: The TUNA Shared Task Evaluation: In Emiel Krahmer and Mariët Theune. (eds.) *Empirical Methods in Natural Language Generation*, pages 264-293.

Albert Gatt and Emiel Krahmer. 2017. Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61: 65-170.

Annette M. Green. 1997. Kappa statistics for multiple raters using categorical classifications. *In Proceedings of the Twenty-Second Annual Conference of SAS Users Group*, San Diego, USA.

Vojtěch Kovář 2016. Evaluating Natural Language Processing Tasks with Low Inter-Annotator Agreement: The Case of Corpus Applications. *In: Recent Advances in Slavonic Natural Language Processing, RASLAN,* pages 127-134.

Vojtěch Kovář, Miloš Jakubíček and Aleš Horák 2016. On Evaluation of Natural Language Processing Tasks Is Gold Standard Evaluation Methodology a Good Solution? *In: Proceedings of the 8th International Conference on Agents and Artificial Intelligence,* Rome, pages 540-545.

Klaus Krippendorff. 1980. *Content Analysis: An Introduction to Its Methodology,* Sage Publications, Beverly Hills, CA.

Keith Godwin and Paul Piwek. 2016. Collecting Reliable Human Judgements on Machine-Generated Language: The Case of the QG-STEC Data. *In Proc. INLG16,* pages 212-216.

Art Graesser, Vasile Rus and Zhiqiang Cai. 2008. Question classification schemes. *Proceedings of the 1st Workshop on Question Generation.*

J. Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics:* 33(1): 159-174.

Hector Martinez Alonso, Anders Johannsen and Barbara Plank. 2016. Supersense tagging with inter-annotator disagreement. *Linguistic Annotation Workshop,* Aug 2016, Berlin, Germany, pages 43-48.

Hwee Tou Ng, Chung Yong Lim and Shou King Foo. 1999. A Case Study on Inter-Annotator Agreement for Word Sense Disambiguation. *Proceedings of the ACL SIGLEX Workshop: Standardizing Lexical Resources,* Aug 2016, College Park, MD, USA, pages 9-13.

Jekaterina Novikova, Ondřej Dušek and Verena Rieser. 2018. RankMe: Reliable Human Ratings for Natural Language Generation. *arXiv:1803.05928*.

Martha Palmer and Nianwen Xue. 2010. Linguistic Annotation. In: A. Clark, C. Fox, and S. Lappin. (eds.) *The Handbook of Computational Linguistics and Natural Language Processing,* Wiley-Blackwell, Chichester, pages 238-270.

Kishore Papineni, Salim Roukos, Todd Ward and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation *ACL '02 Proceedings of the 40th Annual Meeting on Association for Computational Linguistics,* Philadelphia, Pennsylvania – July 07 - 12, 2002, pages 311-318.

Paul Piwek and Kristy Elizabeth Boyer. 2012. Varieties of Question Generation: Introduction to this Special Issue. *Dialogue and Discourse,* 3(2): 1-9.

James Pustejovsky and Amber Stubbs. 2013. Natural Language Annotation for Machine Learning. *OReilly Media, Inc.*

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *Empirical Methods in Natural Language Processing (EMNLP).*

Ehud Reiter and Somayajulu Sripada. 2002. Should Corpora Texts Be Gold Standards fo NLG? *in: Proceedings of the Second International Conference on Natural Language Generation,* pages 97-104.

Vasile Rus, Zhiqiang Cai and Art Graesser. 2008. Question Generation: Example of A Multi-year Evaluation Campaign. In: V. Rus, and A. Graesser. (eds.) *Online Proceedings of 1st Question Generation Workshop, NSF, Arlington, VA.*

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev and Cristian Moldovan. 2012. A detailed account of the First Question Generation Shared Task Evaluation challenge *Dialogue & Discourse,* 3(2):177-204.

Geoffrey R. Sampson. 1995. English for the Computer: The SUSANNE Corpus and Annotation Scheme. *Oxford: Clarendon Press (Oxford University Press).*

Geoffrey Sampson and Anna Babarczy. 2008. Definitional and human constraints on structural annotation of English. *Natural Language Engineering.* 14(4): 471-494.

William A. Scott. 1955. Reliability of content analysis: The case of nominal scale coding. *Public Opinion Quarterly,* 19(3): 321-325.

Dependent Gated Reading for Cloze-Style Question Answering

Reza Ghaeini, Xiaoli Z. Fern, Hamed Shahbazi, and Prasad Tadepalli
Oregon State University, Corvallis, OR, USA
{ghaeinim, xfern, shahbazh, tadepall}@eecs.oregonstate.edu

Abstract

We present a novel deep learning architecture to address the cloze-style question answering task. Existing approaches employ reading mechanisms that do not fully exploit the interdependency between the document and the query. In this paper, we propose a novel *dependent gated reading* bidirectional GRU network (DGR) to efficiently model the relationship between the document and the query during encoding and decision making. Our evaluation shows that DGR obtains highly competitive performance on well-known machine comprehension benchmarks such as the Children's Book Test (CBT-NE and CBT-CN) and Who DiD What (WDW, Strict and Relaxed). Finally, we extensively analyze and validate our model by ablation and attention studies.

1 Introduction

Human language comprehension is an important and challenging task for machines that requires semantic understanding and reasoning over clues. The goal of this general task is to read and comprehend the given document and answer queries.

Recently, the cloze-style reading comprehension problem has received increasing attention from the NLP community. A cloze-style query (Taylor, 1953) is a short passage of text containing a blank part, which we must fill with an appropriate token based on the reading and understanding of a related document. The recent introduction of several large-scale datasets of cloze-style question answering made it feasible to train deep learning systems for such task (Onishi et al., 2016; Hill et al., 2015; Hermann et al., 2015). Various deep learning models have been proposed and achieved reasonable results for this task (Yang et al., 2017; Dhingra et al., 2017; Munkhdalai and Yu, 2017; Cui et al., 2017; Trischler et al., 2016; Kadlec et al., 2016; Chen et al., 2016; Cui et al., 2016; Sordoni et al., 2016). The success of recent models are mostly due to two factors: 1) Attention mechanisms (Bahdanau et al., 2014), which allow the model to sharpen its understanding and focus on important and appropriate subparts of the given context; 2) Multi-hop architectures, which read the document and/or the query in multiple passes, allowing the model to re-consider and refocus its understanding in later iterations. Intuitively, both attention mechanisms and multi-hop reading fulfill the necessity of considering the dependency aspects of the given document and the query. Such a consideration enables the model to pay attention to the relevant information and ignore the irrelevant details. Human language comprehension is often performed by jointly reading the document and query to leverage their dependencies and stay focused in reading and avoid losing relevant contextual information. Current state-of-the-art models also attempt to capture this by using the reading of the query to guide the reading of the document (Yang et al., 2017; Dhingra et al., 2017), or using the memory of the document to help interpret the query (Munkhdalai and Yu, 2017). However, these systems only consider uni-directional dependencies. Our primary hypothesis is that we can gain further improvements by considering bidirectional dependencies.

In this paper, we present a novel multi-hop neural network architecture, called Dependent Gated Reading (DGR), which addresses the aforementioned gap and performs dependent reading in both directions. Our model begins with an initial reading step that encodes the given query and document, followed by an

Proceedings of the 27th International Conference on Computational Linguistics, pages 3330–3345
Santa Fe, New Mexico, USA, August 20-26, 2018.

iterative reading module (multi-hop) that employs soft attention to extract the most relevant information from the document and query encodings to augment each other's representation, which are then passed onto the next iteration of reading. Finally, the model performance a final round of attention allocate and aggregate to rank all possible candidates and make prediction.

We evaluate our model on well-known machine comprehension benchmarks such as the Children's Book Test (CBT-NE & CBT-CN), and Who DiD What (WDW, Strict & Relaxed). Our experimental results indicate the effectiveness of DGR by achieving state-of-the-art results on CBT-NE, WDW-Strict, and WDW-Relaxed. In summary, our contributions are as follows: 1) we propose a new deep learning architecture to address the existing gap of reading dependencies between the document and the query. The proposed model outperforms the state-of-the-art for CBT-NE, WDW-Strict, and WDW-Relaxed by $0.5\%, 0.8\%$, and 0.3% respectively; 2) we perform an ablation study and analysis to clarify the strengths and weaknesses of our model while enriching our understanding of the language comprehension task.

2 Related Work

The availability of large-scale datasets (Onishi et al., 2016; Hill et al., 2015; Hermann et al., 2015) has enabled researchers to develop various deep learning-based architectures for language comprehension tasks such as cloze-style question answering.

Sordoni et al. (2016) propose an *Iterative Alternative Attention* (IAA) reader. IAA is a multi-hop comprehension model which uses a GRU network to search for correct answers from the given document. IAA is the first model that does not collapse the query into a single vector. It deploys an iterative alternating attention mechanism that collects evidence from both the document and the query.

Kadlec et al. (2016) Introduce a single-hop model called *Attention Sum Reader* (AS Reader) that uses two bi-directional GRUs (BiGRU) to independently encode the query and the document. It then computes a probability distribution over all document tokens by taking the softmax of the dot product between the query and the token representations. Finally, it introduces a *pointer-sum attention* aggregation mechanism to aggregate the probability of multiple appearances of the same candidate. The candidate with the highest probability will be considered as the answer. Cui et al. (2017) introduce a similar single-hop model called *attention-over-attention* (AOA) reader which uses a two-way attention mechanism to allow the query and document to mutually attend to one another.

Trischler et al. introduce EpiReader (2016), which uses AS Reader to first narrow down the candidates, then replaces the query placeholder with each candidate to yield a different query statement, and estimate the entailment between the document and the different query statements to predict the answer.

Munkhdalai and Yu (2017) (NSE) propose a computational hypothesis testing framework based on memory augmented neural networks. They encode the document and query independently at the beginning and then re-encode the query (but not the document) over multiple iterations (hops). At the end of each iteration, they predict an answer. The final answer is the candidate that obtains the highest probability over all iterations.

Dhingra et al. (2017) extend the AS Reader by proposing *Gated Attention Reader* (GA Reader). GA Reader uses a multi-hop architecture to compute the representation of the documents and query. In each iteration the query is encoded independent of the document and previous iterations, but the document is encoded iterative considering the previous iteration as well as an attention mechanism with multiplicative gating to generate query-specific document representations. GA reader uses the same mechanism for making the final predictions as the AS reader. Yang et al. (2017) further extend the GA Reader with a fine grained gating approach that uses external semantic and syntactic features (i.e. NER, POS, etc) of the tokens to combine the word and character level embeddings and produce a final representation of the words.

Among the aforementioned models, the GA Reader is the closest to our model in that we use a similar architecture that is multi-hop and performs iterative reading. The main distinct between our model and the GA Reader is the reading and encoding of the query. Instead of performing independent reading of query in each iteration, our reading and encoding of the query not only depends on the document but also the reading of previous iterations.

Although cloze-style question answering task is well studied in the literature, the potential of dependent reading and interaction between the document and the query is not rigorously explored. In this paper, we address this gap by proposing a novel deep learning model (DGR). Experimental results demonstrate the effectiveness of our model.

3 Dependent Gated Reading

Figure 1 depicts a high-level view of our proposed Dependent Gate Reading (DGR) model, which follows a fairly standard multi-hop architecture, simulating the multi-step reading and comprehension process of humans.

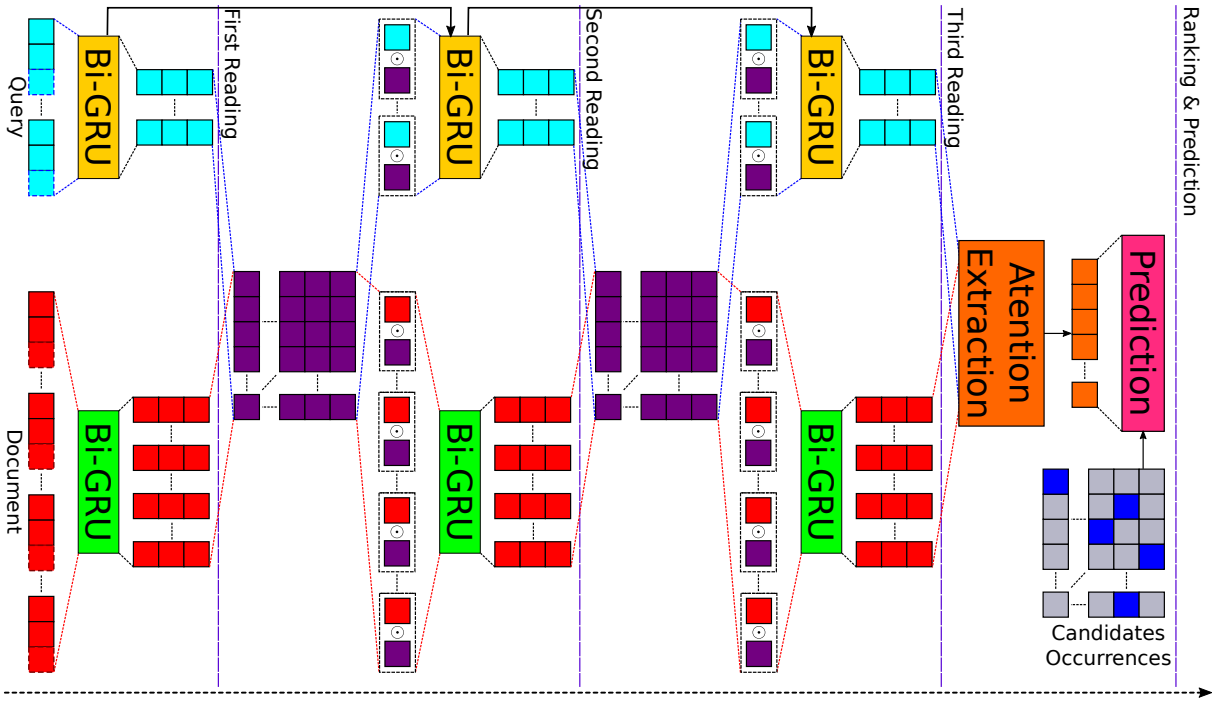


Figure 1: A high-level view of dependent gated reading model (DGR). The data (document d and query q, depicted with red and cyan tensors respectively) flows from left to right. At the first (input) layer, the word representations are shown with black solid borders while the character representations are shown with colored dashed borders. The figure is color coded; relevant tensors and elements are shown with the same color. Note that none of the elements share parameters. The purple matrices extract relevant information between document and query representations. The black arrows between the query BiGRUs (yellow ones) pass the final hidden state of a BiGRU to another one as initialization value for its hidden state.

The input to our model at the training stage can be represented as a tuple (D, Q, C, a), where $D = [d_1, \cdots, d_n]$ is the document of length n, $Q = [q_1, \cdots, q_m]$ is the query of length m with a placeholder, $C = [c_1, \cdots, c_g]$ is a set of g candidates and $a \in C$ is the ground truth answer. Here we assume d_i, q_j are some form of embedding of the individual tokens of document and query. At the testing stage, given the input document D, query Q and candidate set C, the goal is to choose the correct candidate a among C for the placeholder in Q.

DGR can be divided to two major parts: Multi-hop Reading, and Ranking & Prediction.

3.1 Multi-hop Reading of Document and Query

Recurrent networks provide a natural solution for modeling variable length sequences. Consequently, we use bi-directional Gated Recurrent Units (BiGRUs) (Cho et al., 2014) as the main building blocks for encoding the given document and query. For the initial step of our multi-hop reading, the document D and the query q are read with two separate BiGRUs (Equations 1 and 2) where $\hat{d}^0 \in \mathbb{R}^{n \times r}$ and

$\hat{q}^0 \in \mathbb{R}^{m \times r}$ are the first BiGRU reading sequences of D and Q respectively. h^0 consists of two parts, h_f^0 and h_b^0, which record the final output of forward and backward GRU reading of Q respectively. Note that "$-$" in equations means that we do not care about the associated variable and its value.

$$\hat{d}^0, - = BiGRU_{d0}(D, 0) \tag{1}$$

$$\hat{q}^0, h^0 = BiGRU_{q0}(Q, 0) \tag{2}$$

We use $s \in [0, S]$ to denote the reading iteration, with $S + 1$ total iterations. For the initial iteration ($s = 0$), both BiGRUs are fed with a zero vector for the initial hidden state as shown in Equations 1 and 2. Once the document and query encodings ($\hat{d}^s$ and $\hat{q}^s$ respectively) are computed, we employ a soft alignment method to associate the relevant sub-components between the given document and query. In deep learning models, this is often achieved with a soft attention mechanism. We follow the same soft attention mechanism as used in the GA reader (Dhingra et al., 2017), which is described below for completeness.

Given $\hat{d}^s$ and $\hat{q}^s$, we first compute the unnormalized attention weights between the i-th token of the document and the j-th token of the query as the similarity between the corresponding hidden states with Equation 3 (energy function).

$$e_{ij}^s = (\hat{d}_i^s)^T \hat{q}_j^s, \quad \forall i \in [1, n], \forall j \in [1, m], \forall s \in [0, S-1] \tag{3}$$

For each document token and query token, the most relevant semantics from the other context are extracted and composed based on $e_{ij}^s \in \mathbb{R}^{n \times m}$. Equations 4 and 5 provide the specific details of this procedure where $\tilde{d}_i^s \in \mathbb{R}^r$ represents the extracted information from the current reading of the query, $\hat{q}^s$, that is most relevant to the i-th document token by attending to $\hat{d}_i^s$. Similarly $\tilde{q}_j^s \in \mathbb{R}^r$ represents, for the j-th query token, the extracted relevant document information from $\hat{d}^s$ by attending to $\hat{q}_j^s$.

$$\tilde{d}_i^s = \sum_{j=1}^{m} \frac{\exp(e_{ij}^s)}{\sum_{k=1}^{m} \exp(e_{ik}^s)} \hat{q}_j^s, \quad \forall i \in [1, n], \forall s \in [0, S-1] \tag{4}$$

$$\tilde{q}_j^s = \sum_{i=1}^{n} \frac{\exp(e_{ij}^s)}{\sum_{k=1}^{n} \exp(e_{kj}^s)} \hat{d}_i^s, \quad \forall j \in [1, m], \forall s \in [0, S-1] \tag{5}$$

To incorporate the context information, we use element-wise product of the tuples $(\hat{d}_i^s, \tilde{d}_i^s)$ or $(\hat{q}_j^s, \tilde{q}_j^s)$ to produce a new representation of the hidden states for the document and the query as described in Equations 6 and 7.

$$u_i^s = \hat{d}_i^s \odot \tilde{d}_i^s, \quad \forall s \in [0, S-1] \tag{6}$$

$$v_j^s = \hat{q}_j^s \odot \tilde{q}_j^s, \quad \forall s \in [0, S-1] \tag{7}$$

Here $\odot$ stands for element-wise product, and $u^s \in \mathbb{R}^r$ and $v^s \in \mathbb{R}^r$ are the new encodings of the document and query respectively.Note that GA-reader uses the same mechanism to update the document encoding but does not change the query representation according to the document.

We then pass the new document (u^s) and query (v^s) embeddings to the BiGRUs for the next iteration $s + 1$. Note that for query reading, we feed, h^s, the final hidden state of the previous reading (without document based updates) to the BiGRU of the next iteration as the initial hidden state. Intuitively, h^s provides a summary understanding of the query from the previous iteration, without the document modulated updates. By considering both h^s and v^s, this encoding mechanism provides a richer representation of the query. This is formally described by Equations 8 and 9.

$$\hat{d}^{s+1}, - = BiGRU_{ds}(u^s, 0), \forall s \in [0, S-1] \tag{8}$$

$$\hat{q}^{s+1}, h^{s+1} = BiGRU_{qs}(v^s, h^s), \forall s \in [0, S-1] \tag{9}$$

We should note that using the following configuration variations did not yield any improvement to our model: 1) Other choices for gating aggregation strategy (Equations 6 and 7) like addition, concatenation, or applying a transformation function on different sub-members of {element-wise product, concatenation and difference}. 2) Residual connection.

3.2 Ranking & Prediction

Given the final document and query encodings, $\hat{d}^S$ and $\hat{q}^S$, the final stage of our model computes a score for each candidate $c \in C$. This part of our model use the same *point sum attention* aggregation operation as introduced by the Attention Sum (AS) reader (Kadlec et al., 2016), which is also used by the GA reader (Dhingra et al., 2017).

Let idx be the position of the the placeholder in Q, and $\hat{q}^S_{idx}$ be the associated hidden embedding of the placeholder in the given query. We first compute the probability of each token in the document to be the desired answer by computing the dot product between $\hat{q}^S_{idx}$ and $\hat{d}^S_j$ for $j = 1, ..., n$ and then normalize with the softmax function:

$$y = \text{softmax}((\hat{q}^S_{idx})^T \hat{d}^S) \tag{10}$$

where $y \in \mathbb{R}^n$ gives us a normalized attention/probability over all tokens of the document. Next, the probability of each particular candidate $c \in C$ for being the answer is computed by aggregating the document-level attentions of all positions in which c appears:

$$p(c|D, Q) \propto \sum_{i \in I(c,D)} y_i, \quad \forall c \in C \tag{11}$$

where $I(c, D)$ indicates the positions that candidate c appears in the document D (Candidate Occurrences in Figure 1). Finally the prediction is given by $a^* = \textit{argmax}_{c \in C}\ p(c|D, Q)$.

Key differences from the GA reader. Given the strong similarity between our model and the GA reader, it is worth highlighting the three key differences between the two models: (a) Document gated query reading: we compute a document-specific query representations to pass to the next query reading step; (b) Dependent query reading: in each iteration, the input to the query BiGRU comes from the document gated encoding of the query from the last iteration whereas the GA Reader reads the queries independently in all iterations; (c) Dependent query BiGRU initialization: the query BiGRU is initialized with the final hidden states of the query BiGRU from the previous iteration. These key differences in query encoding are designed to better capture the interdependences between query and document and produce richer and more relevant representations of the query and enhance the comprehension and query answering performance.

3.3 Further Enhancements

Following the practice of GA reader, we included several enhancements which have been shown to be helpful in previous work.

Question Evidence Common Word Feature. To generate the final document encoding $\hat{d}^S$, an additional modification of u^{S-1} is introduced before applying Equation 8. Specifically, an additional *Question Evidence Common Word Feature* (qe-comm) (Li et al., 2016) is introduced for each document token, indicating whether the token is present in the query. Assume f_i stands for the qe-comm feature of the i-th document token, therefore, $u^{S-1}_i = [u^{S-1}_i, f_i]$.

Character-level embeddings. Word-level embeddings are good at representing the semantics of the tokens but suffers from out-of-vocabulary (OOV) words and is incapable of representing sub-word morphologies. Character-level embeddings effectively address such limitations (Ling et al., 2015; Dhingra et al., 2016). In this work, we represent a token by concatenating its word embedding and character embedding. To compute the character embedding of a token $w = [x_1, \cdots, x_l]$, we pass w to two GRUs in forward and backward directions respectively. Their outputs are then concatenated and passed through a linear transformation to form the character embedding of the token.

4 Experiments and Evaluation

4.1 Datasets

We evaluate the DGR model on three large-scale language comprehension datasets, Children's Book Test Named Entity (CBT-NE), Common Noun (CBT-CN), and Who Did What (WDW) Strict and Relaxed.

The first two datasets are formed from two subsets of the Children's Book Test (CBT) (Hill et al., 2015). Documents in CBT consist of 20 contiguous sentences from the body of a popular children's book, and queries are formed by replacing a token from the 21^{st} sentence with a placeholder. We experiment on subsets where the replaced token is either a named entity (CBT-NE) or common noun (CBT-CN). Other subsets of CBT have also been studied previously but because simple language models have been able to achieve human-level performance on them, we ignore such subsets (Hill et al., 2015).

The Who Did What (WDW) dataset (Onishi et al., 2016) is constructed from the LDC English Gigaword newswire corpus. Each sample in WDW is formed from two independent articles. One article is considered as the passage to be read and the other article on the same subject is used to form the query. Missing tokens are always person named entities. For this dataset, samples that are easily answered by simple systems are filtered out, which makes the task more challenging. There are two versions for the training set (Strict and Relaxed) while using the same development and test sets. Strict is a small but focused/clean training set while Relaxed is a larger but more noisy training set. We experiment on both of these training sets and report corresponding results on both settings. Statistics of all the aforementioned datasets are summarized in Table 3 of the appendix.

Other datasets for this task include CNN and Daily Mail News (Hermann et al., 2015). Because previous models already achieved human-level performance on these datasets, following Munkhdalai and Yu (2017), we do not include them in our study.

4.2 Training Details & Experimental Setup

We use pre-trained 100-D Glove $6B$ vectors (Pennington et al., 2014) to initialize our word embeddings while randomly initializing the character embedding. All hidden states of BiGRUs have 128 dimensions ($o = 100$ and $r = 128$). The weights are learned by minimizing the negative log-loss (Equation 12) on the training data via the Adam optimizer (Kingma and Ba, 2014). The learning rate is 0.0005. To avoid overfitting, we use dropout (Srivastava et al., 2014) with rate of 0.4 and 0.3 for CBT and WDW respectively as regularization, which is applied to all feedforward connections. While we fix the word embedding, character embeddings are updated during the training to learn effective representations for this task. We use a fairly small batch size of 32 to provide more exploration power to the model.

$$L = \sum_i -\log(p(a|D, Q)) \tag{12}$$

4.3 Results

Table 1 shows the test accuracy of the models on CBT-NE, CBT-CN, WDW-Strict, and WDW-Relaxed. We divide the previous models into four categories: 1) Single models (rows 1-5), 2) Ensemble models (rows 6-9), 3) NSE models (rows 10-14), and 4) the FG model (row 15). Table 1 primarily focuses on comparing models that do not rely on any NLP toolkit features (i.e. POS, NER, etc), with the exception of the FG model which uses additional information about document tokens including POS, NER and word frequency information to produce the embedding of the token.

From Table 1, we can see that DGR achieves the state-of-the-art results on all aforementioned datasets expect for CBT-CN. The targets of CBT-NE, WDW-Strict, and WDW-Relaxed are all Named Entities while the CBT-CN focuses on Common Noun. We believe that our architecture is more suitable for Named Entity targeted comprehension tasks. This phenomenon warrants a closer look in future work. Comparing GA Reader, FG, and DGR (the three models with similar architectures), we see that FG outperform the GA Reader on CBT-CN and WDW-Strict datasets while DGR outperforms both FG and GA Reader results on CBT-NE, WDW-Strict, WDW-Relaxed datasets with noticeable margins. This suggests

Method	Test Accuracy(%)			
	CBT-NE	CBT-CN	WDW-Strict	WDW-Relaxed
AS Reader (Kadlec et al., 2016)	68.6%	63.4%	57.0%	59.0%
EpiReader (Trischler et al., 2016)	69.7%	67.4%	-	-
IAA Reader (Sordoni et al., 2016)	68.6%	69.2%	-	-
AOA Reader (Cui et al., 2017)	72.0%	69.4%	-	-
GA Reader (Dhingra et al., 2017)	74.9%	70.7%	71.2%	72.6%
AS Reader (Ensemble) (Kadlec et al., 2016)	70.6%	68.9%	-	-
EpiReader (Ensemble) (Trischler et al., 2016)	71.8%	70.6%	-	-
IAA Reader (Ensemble) (Sordoni et al., 2016)	72.0%	71.0%	-	-
AOA Reader (Ensemble) (Cui et al., 2017)	74.5%	70.8%	-	-
NSE (T=1) (Munkhdalai and Yu, 2017)	71.1%	69.7%	65.5%	65.3%
NSE Query Gating (T=2) (Munkhdalai and Yu, 2017)	71.5%	70.7%	65.1%	65.5%
NSE Query Gating (T=6) (Munkhdalai and Yu, 2017)	71.4%	72.0%	65.7%	65.8%
NSE Adaptive Computation (T=2) (Munkhdalai and Yu, 2017)	72.1%	71.2%	65.4%	66.0%
NSE Adaptive Computation (T=12) (Munkhdalai and Yu, 2017)	73.2%	71.4%	66.2%	66.7%
FG (Yang et al., 2017)	74.9%	72.0%	71.7%	72.6%
DGR	**75.4%**	70.7%	**72.0%**	**72.9%**

Table 1: Performance of proposed model (DGR) on the test set of CBT-NE, CBT-CN, WDW-Strict, and WDW-Relaxed datasets.

Method	Development Accuracy(%)			
	CBT-NE	CBT-CN	WDW-Strict	WDW-Relaxed
1) DGR	**77.90**	**73.80**	**71.78**	**72.26**
2) DGR - (a)	75.60	72.25	71.04	71.82
3) DGR - (c)	77.50	72.45	71.29	71.93
4) DGR - (a) & (b)	77.85	73.05	71.67	72.20
5) DGR - (a) & (c)	76.00	72.85	71.37	72.13
6) DGR - (a) & (b) & (c)	77.65	73.00	71.61	72.16

Table 2: Ablation study results. Performance of different configurations of the proposed model on the development set of the CBT-NE, CBT-CN, WDW-Strict, and WDW-Relaxed datasets

that while the NLP toolkit features such as POS and NER could help the performance of the comprehension models (specially in CBT-CN), capturing richer dependency interaction between document and query appears to play a more important role for comprehension tasks focusing on Named Entities.

Finally, For each of the three datasets on which our model achieves the state-of-the-art performance, we conducted the one-sided McNemars test to verify the statistical significance of the performance improvement over the main competitor (GA reader). The obtained p-values are 0.03, 0.003, and 0.011 for CBT-NE, WDW-Strict, and WDW-Relaxed respectively, indicating that the performance gain by DGR is statistically significant.

4.4 Ablation Study

We conducted an ablation study on our model to examine the importance and the effect of proposed strategies. We investigate all settings on the development set of the CTB-NE, CBT-CN, WDW-Strict, and WDW-Relaxed datasets. Consider the three key differences of our method from the GA Reader: (a) Document gated query reading — here we compute a document-specific query representations to pass to the next reading layer; (b) Dependent query reading — the query readings are dependent from one layer to the next as the input to the next reading layer comes from the output of previous layer; (c) Dependent BiGRU initialization — query BiGRUs of a later layer are initialized with the final hidden states of previous layer's query BiGRU.

Table 2 shows the ablation study results on the development set of CBT-NE, CBT-CN, WDW-Strict, and WDW-Relaxed for a variety of DGR configurations by removing one or more of the key differences with GA reader. Note that by all removing all three difference elements, configuration 6 reduces to the GA reader.

According to Table 2, DGR achieves the best development accuracy on all datasets which indicates that collectively, the three elements lead to improved effectiveness.

Effect of document dependent reading. Configuration 2 removes the document dependent reading, and retains the other two elements. Interestingly, this configuration achieved the worst performance among all variations. Without proper guiding from the document side, iteratively reading the query actually leads to worse performance than independent query reading. This suggests that document dependent reading is a critical element that helps achieve better reading of query.

Effect of Dependent Query BiGRU initialization. In Configuration 3, we remove the dependent query BiGRU initialization, which results in a performance loss ranging from 0.33% (WDW-relaxed) to 1.35% (CBT-CN), suggesting that this connection provides important information that helps the reading of the query. Note that simply adding dependent query BiGRU initialization to GA reader (configuration 4) leads to a slight improvement over GA reader, which again confirms the usefulness of this channel of information.

Effect of dependent query reading. Unfortunately, we cannot only remove (b) from our model because it will cause dimension mismatch between the document and query representation preventing the gating operation for computing the document gated query representation. Instead, we compare the GA reader (configuration 6) with configure 5, which adds dependent query reading to the GA reader. We can see that adding the dependent query reading to the GA reader actually leads to a slight performance loss. Note that further including document gated reading (configuration 3) improves the performance on CBT-NE, but still fails to outperform GA reader. This points to a potential direction to further improve our model by designing a new mechanism that is capable of document dependent gating without the dependent query reading.

4.5 Analysis

In this section, We first investigate the performance of DGR and its variations on two attributes: the *document length*, and *query length*. Then we show a layer-wise visualization of the energy function (Equation 3) for an instance from the CBT-NE dataset.

4.5.1 Length Study

Among the four datasets that we use in this paper, WDW-Relaxed is the biggest and the most noisy one which makes it as a good candidate for analyzing the trend and behavior of our models.

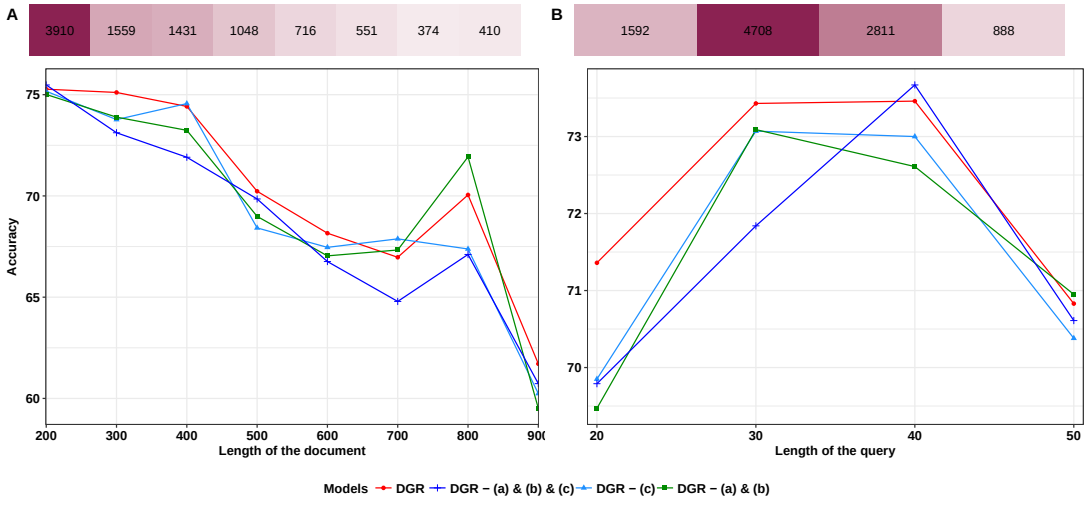


Figure 2: Test accuracy of DGR and its variations against the length of the document (A), and length of the query (B) on the WDW-Relaxed dataset. The bar on top of each figure indicates the number of samples in each interval. Darker color in the bars illustrates more samples.

Figure 2 depicts the performance of DGR and its variations against the length of document (left), and the length of query (right). A bar on top of each diagram indicates the frequency of samples in each

intervals. Each data sample is added to the closet interval.

Overall Figure 2 suggests that DGR achieves highly competitive performance across different document and query lengths in comparison to the other variations including the GA reader. In particular, DGR perform better or similarly to the GA reader ("DGR - (a) & (b) & (c)") in all categories except when query length is between 30 and 40 where GA reader wins with a small margin. Furthermore, we see that "DGR - (a) & (b)" wins over "DGR - (a) & (b) & (c)" in most document length categories. This suggests the positive effect of the connection offered by (c), especially for longer documents.

4.5.2 Attention Study

To gain insights into the influence of the proposed strategies on the internal behavior of the model, we analyze the attention distribution at intermediate layers. We show a visualization of layer-wise normalized aggregated attention weights (energy function, Equation 3) for candidate set over the query (for more examples look at Section C of the appendix). In each figure, the top plots show the layer-wise attention of DGR and the bottom plots show the layer-wise attention of the GA reader, i.e., "DGR - (a) & (b) & (c)". Moreover, the left and middle plot show the aggregated attention of candidates over the whole query while the right plot depicts the aggregated attention of the candidates for the placeholder in the query in the final layer.

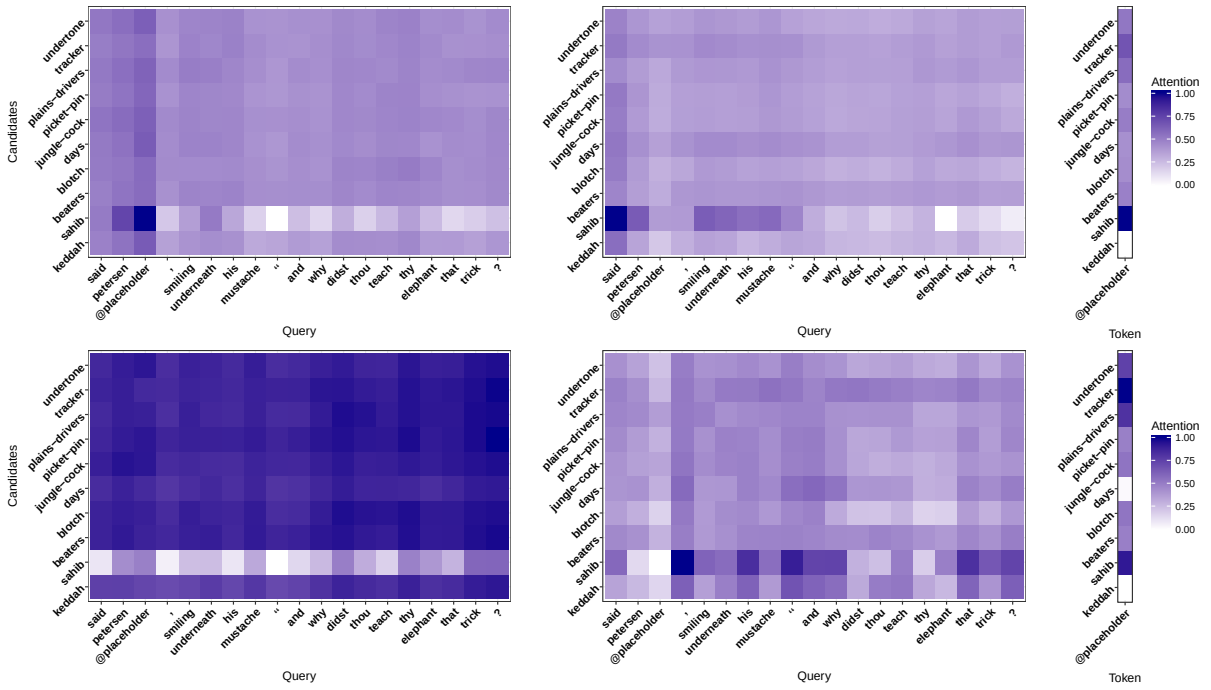


Figure 3: Layer-wise normalized attention visualization of "DGR" (top) and "DGR - (a) & (b) & (c)" (bottom) for a sample from the CBT-NE test set. Darker color illustrates higher attention. Figures only show the aggregated attention of candidates. The gold answer is "sahib".

A generic pattern observed in our study is that GA reader tends to generate more uniform attention distributions while DGR produces more focused attention. In other words, each layer of DGR tends to focus on different sub-parts and examine different hypotheses, illustrating the significant impact of the proposed strategies on the attention mechanism.

5 Conclusion

We proposed a novel cloze-style question answering model (DGR) that efficiently model the relationship between the document and the query. Our model achieves the the state-of-the-art results on several large-scale benchmark datasets such as CBT-NE, WDW-Strict, and WDW-Relaxed. Our extensive analysis

and ablation studies confirm our hypothesis that using a more sophisticated method for modeling the interaction between document and query could yield further improvements.

References

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

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Empirical Methods in Natural Language Processing*, pages 1724–1734.

Yiming Cui, Ting Liu, Zhipeng Chen, Shijin Wang, and Guoping Hu. 2016. Consensus attention-based neural networks for chinese reading comprehension. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 1777–1786.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-attention neural networks for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 593–602.

Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William W. Cohen. 2016. Tweet2vec: Character-based distributed representations for social media. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1832–1846.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *CoRR*, abs/1511.02301.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

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

Peng Li, Wei Li, Zhengyan He, Xuguang Wang, Ying Cao, Jie Zhou, and Wei Xu. 2016. Dataset and neural recurrent sequence labeling model for open-domain factoid question answering. *arXiv preprint arXiv:1607.06275*.

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1520–1530.

Tsendsuren Munkhdalai and Hong Yu. 2017. Reasoning with memory augmented neural networks for language comprehension. *ICLR*, abs/1610.06454.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David A. McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2230–2235.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Alessandro Sordoni, Phillip Bachman, and Yoshua Bengio. 2016. Iterative alternating neural attention for machine reading. *CoRR*, abs/1606.02245.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Wilson L Taylor. 1953. cloze procedure: a new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

Adam Trischler, Zheng Ye, Xingdi Yuan, Philip Bachman, Alessandro Sordoni, and Kaheer Suleman. 2016. Natural language comprehension with the epireader. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 128–137.

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W. Cohen, and Ruslan Salakhutdinov. 2017. Words or characters? fine-grained gating for reading comprehension. volume abs/1611.01724.

A Dataset Statistics

	CBT-NE	CBT-CN	WDW-Strict	WDW-Relaxed
# training set	108,719	120,769	127,786	185,978
# development set	2,000	2,000	10,000	10,000
# test set	2,500	2,500	10,000	10,000
# vocabulary	53,063	53,185	347,406	308,02
max document length	1,338	1,338	3,085	3,085

Table 3: Dataset statistics

B Rule-based Disambiguation Study

In this section, we present a simple rule-based detection strategy for CBT-NE dataset which disambiguates about 30% and 18% of the samples in CBT-NE development and test sets. For each query q, assume w is previous/next next word in the placeholder which start with upper case character. If such a w exists, we look for w in the document d and collect all words that could appears next/before w. After removing all collected words that are not in the candidate list C, the samples is disambiguated and solved if we end up with a single word (answer). We refer to the set of such samples as disambiguated set. Table 4 shows the statistics of this rule-based strategy on the rule-based disambiguated test set of CBT-NE. Furthermore, Table 5 shows a data sample in CBT-NE that is correctly disambiguated with our rule-based approach.

Set	Correct Disambiguation(%)	Wrong Disambiguation(%)	Total Disambiguation
Development	29.65%	0.1%	595
Test	18.36%	0.12%	462

Table 4: Statistics and performance of the proposed rule-based strategy on CBT-NE dataset.

Figure 4 shows the performance of DGR and its variations on the set of data samples in CBT-NE test set that could be disambiguated with the proposed rule-based strategy. Although we use the lower case words in the training process, all models perform substantially well on disambiguating such samples. This observation could demonstrate the effectiveness of the general architecture.

C Attention Study

In this section, we show visualizations of 8 samples of layer-wise normalized attention (energy function, see Equation 3 in the main paper). Each column in Figures 5-12, represents the same layer in "DGR" and "DGR - (a) & (b) & (c)". Also, each row is allocated to a specific model (Top: DGR, and Bottom: DGR - (a) & (b) & (c)).

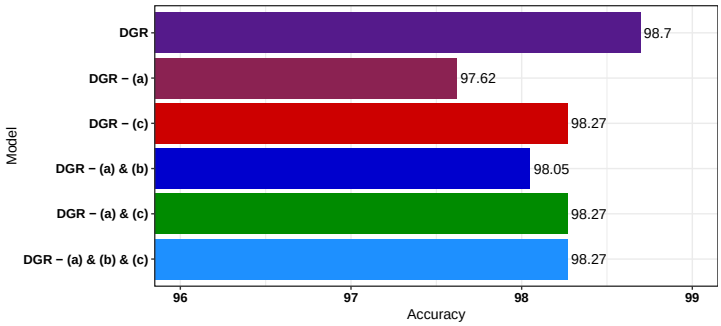


Figure 4: Performance of DGR and its variations on the rule-based disambiguated test set of CBT-NE.

doc[a]	1 Instead of answering , **Jimmy** Skunk began to laugh . 2 “ Who ’s a bug ? ” 3 demanded Old Mr. Toad , more crossly than before . 4 “ There is n’t any bug , Mr. Toad , and I beg your pardon , ” replied **Jimmy** , remembering his politeness . 5 “ I just thought there was . 6 You see , I did n’t know you were under that piece of bark . 7 I hope you will excuse me , Mr. Toad . 8 Have you seen any fat beetles this morning ? ” 9 “ No , ” said Old Mr. Toad grumpily , and yawned and rubbed his eyes . 10 “ Why , ” exclaimed **Jimmy** Skunk , “ I believe you have just waked up ! ” 11 “ What if I have ? ” 12 demanded Old Mr. Toad . 13 “ Oh , nothing , nothing at all , Mr. Toad , ” replied **Jimmy** Skunk , “ only you are the second one I ’ve met this morning who had just waked up . ” 14 “ Who was the other ? ” 15 asked Old Mr. Toad . 16 “ Mr. Blacksnake , ” replied **Jimmy** . 17 “ He inquired for you . ” 18 Old Mr. Toad turned quite pale . 19 “ I – I think I ’ll be moving along , ” said he . 20 XVII OLD MR. TOAD ’S MISTAKE If is a very little word to look at , but the biggest word you have ever seen does n’t begin to have so much meaning as little “ if . ”
query	21 If **Jimmy** @placeholder had n’t ambled down the Crooked Little Path just when he did ; if he had n’t been looking for fat beetles ; if he had n’t seen that big piece of bark at one side and decided to pull it over ; if it had n’t been for all these “ ifs , ” why Old Mr. Toad would n’t have made the mistake he did , and you would n’t have had this story .
cands[b]	Blacksnake, Jimmy, Mr., Skunk, Toad, XVII, bug, morning, pardon, second
ans[c]	Skunk
pred[d]	Skunk

[a] doc, Document
[b] cands, Candidates
[c] ans, Answer
[d] pred, Prediction

Table 5: Example of a disambiguated sample in CBT-NE dataset with the proposed rule-based approach.

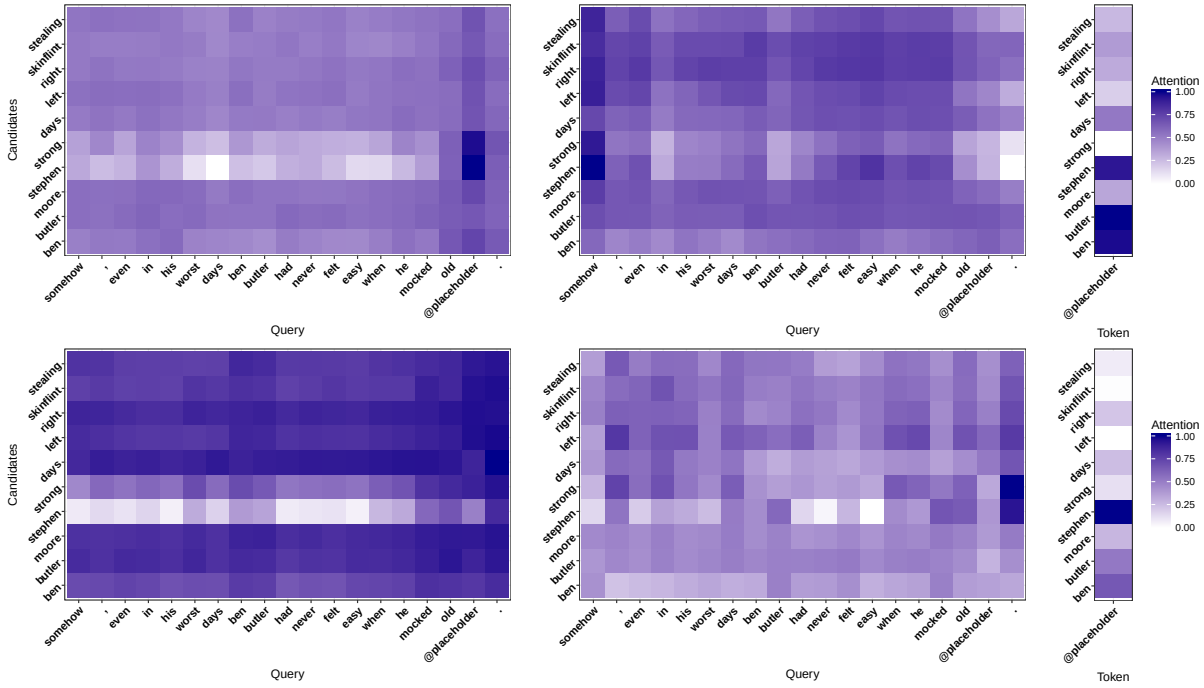


Figure 5: Layer-wise normalized attention visualization of "DGR" (top) and "DGR - (a) & (b) & (c)" (bottom) for a sample from the CBT-NE test set. Darker color illustrates higher attention. Figures only show the aggregated attention of candidates. The gold answer is "butler".

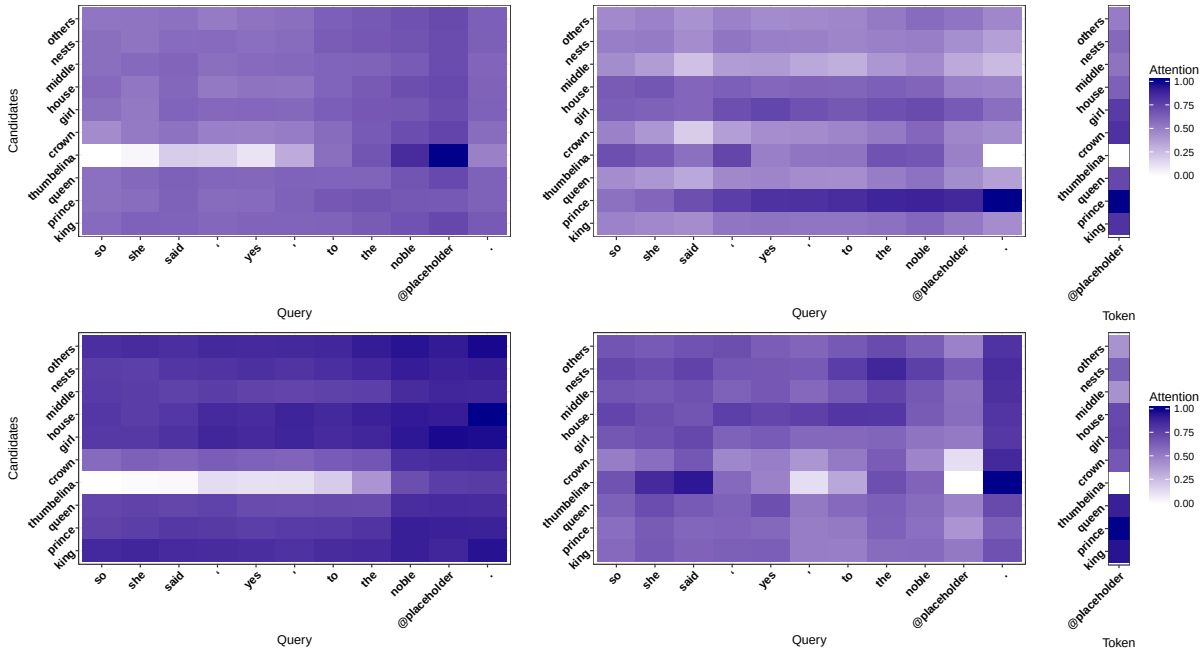


Figure 6: Layer-wise normalized attention visualization of "DGR" (top) and "DGR - (a) & (b) & (c)" (bottom) for a sample from the CBT-NE test set. Darker color illustrates higher attention. Figures only show the aggregated attention of candidates. The gold answer is "prince".

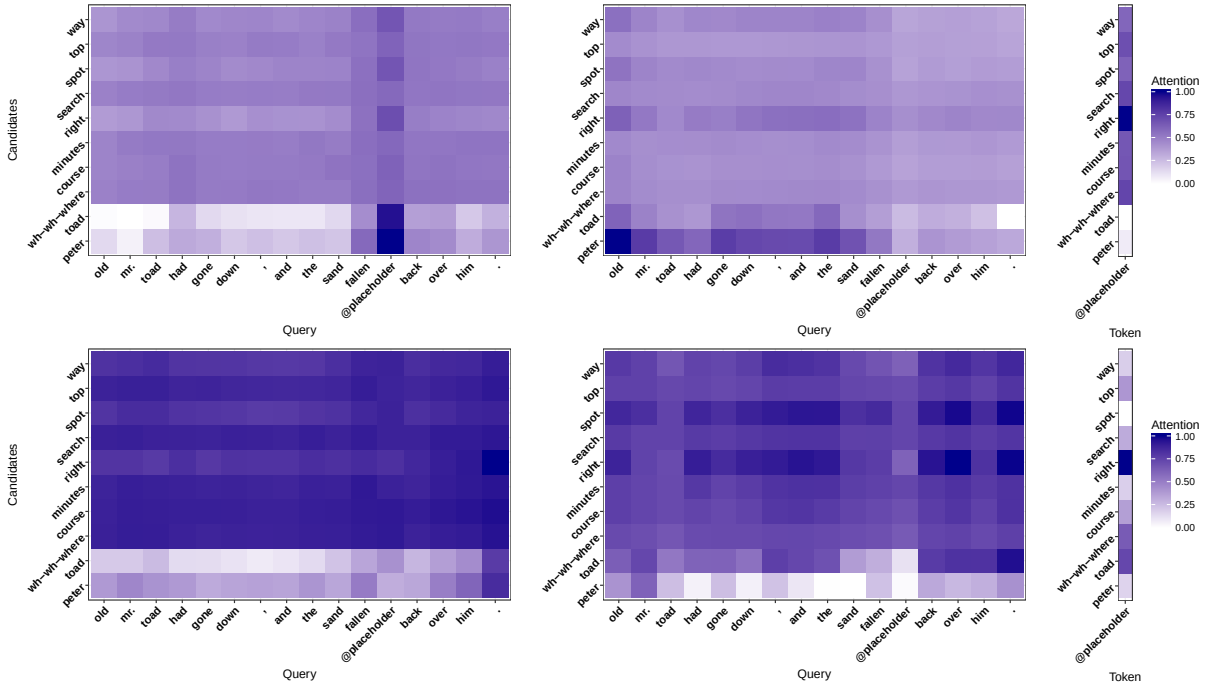


Figure 7: Layer-wise normalized attention visualization of "DGR" (top) and "DGR - (a) & (b) & (c)" (bottom) for a sample from the CBT-NE test set. Darker color illustrates higher attention. Figures only show the aggregated attention of candidates. The gold answer is "right".

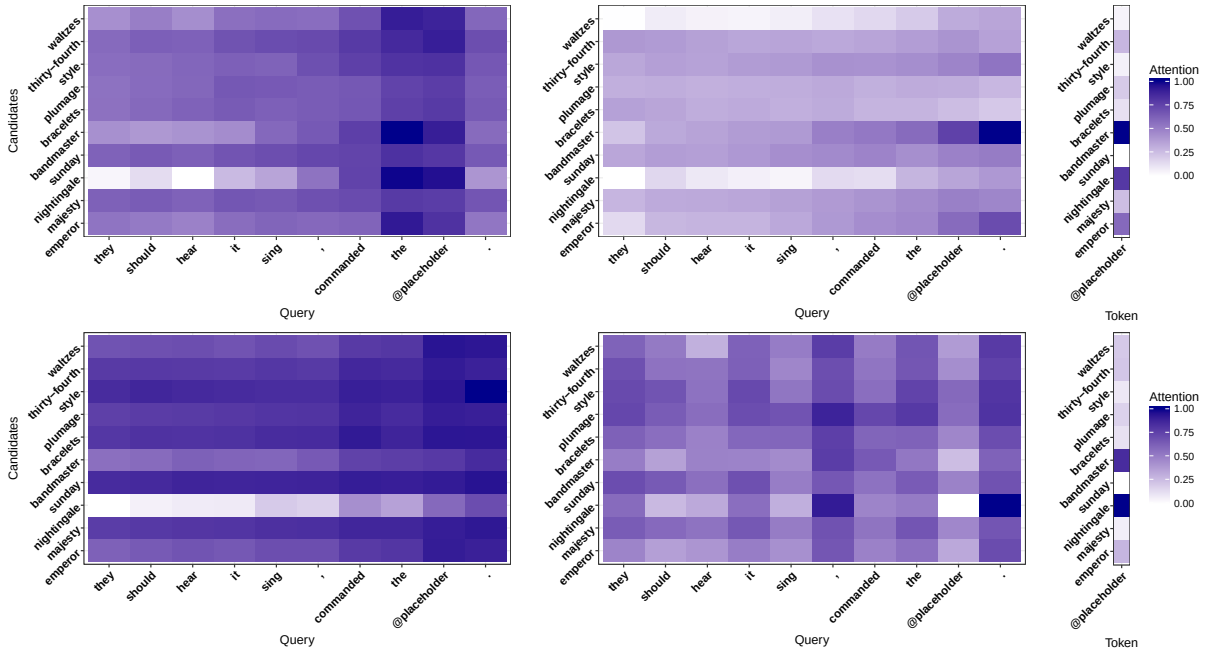


Figure 8: Layer-wise normalized attention visualization of "DGR" (top) and "DGR - (a) & (b) & (c)" (bottom) for a sample from the CBT-NE test set. Darker color illustrates higher attention. Figures only show the aggregated attention of candidates. The gold answer is "bandmaster".

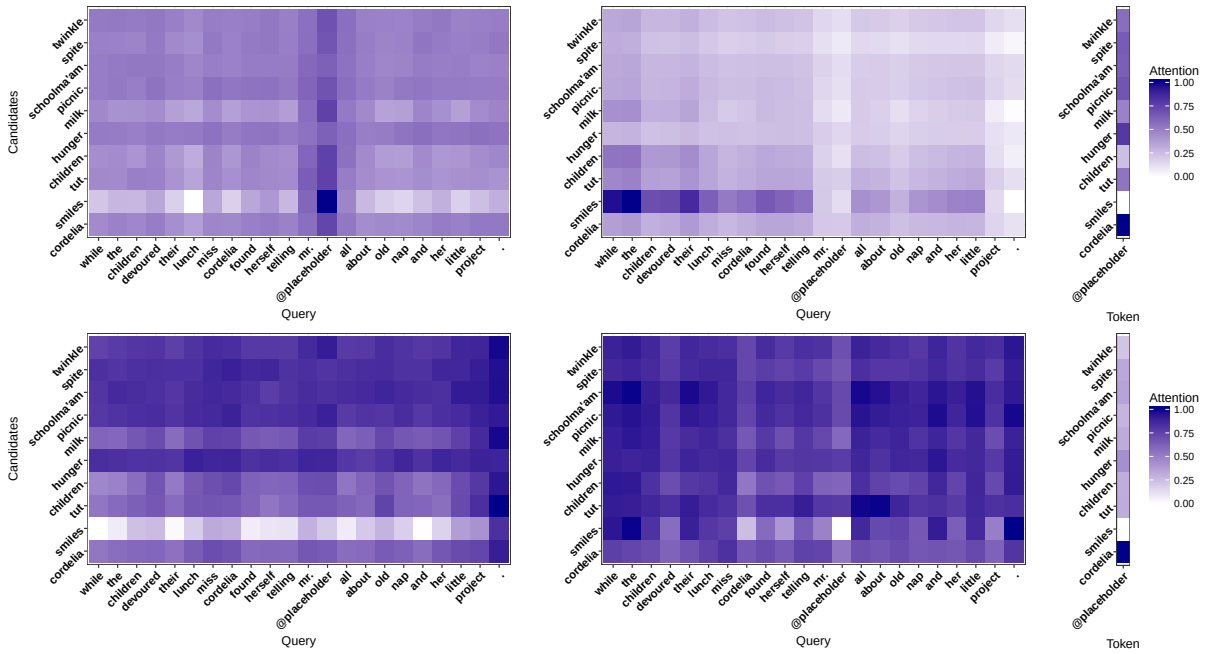


Figure 9: Layer-wise normalized attention visualization of “DGR” (top) and “DGR - (a) & (b) & (c)” (bottom) for a sample from the CBT-NE test set. Darker color illustrates higher attention. Figures only show the aggregated attention of candidates. The gold answer is “cordelia”.

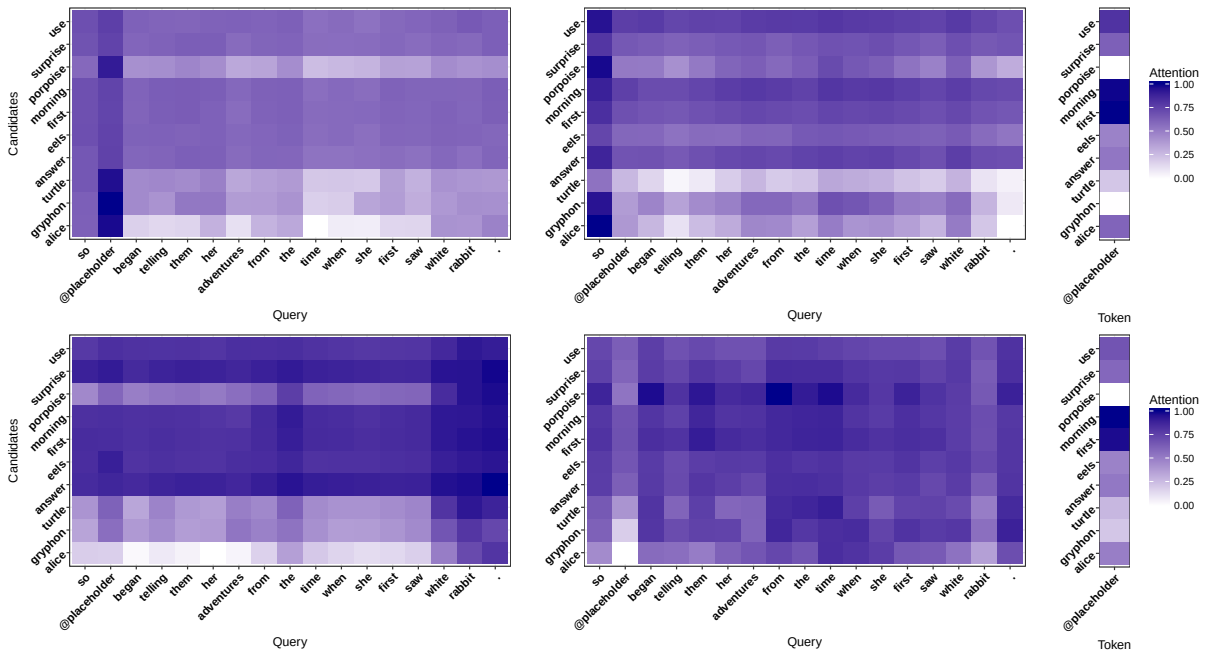


Figure 10: Layer-wise normalized attention visualization of “DGR” (top) and “DGR - (a) & (b) & (c)” (bottom) for a sample from the CBT-NE test set. Darker color illustrates higher attention. Figures only show the aggregated attention of candidates. The gold answer is “first”.

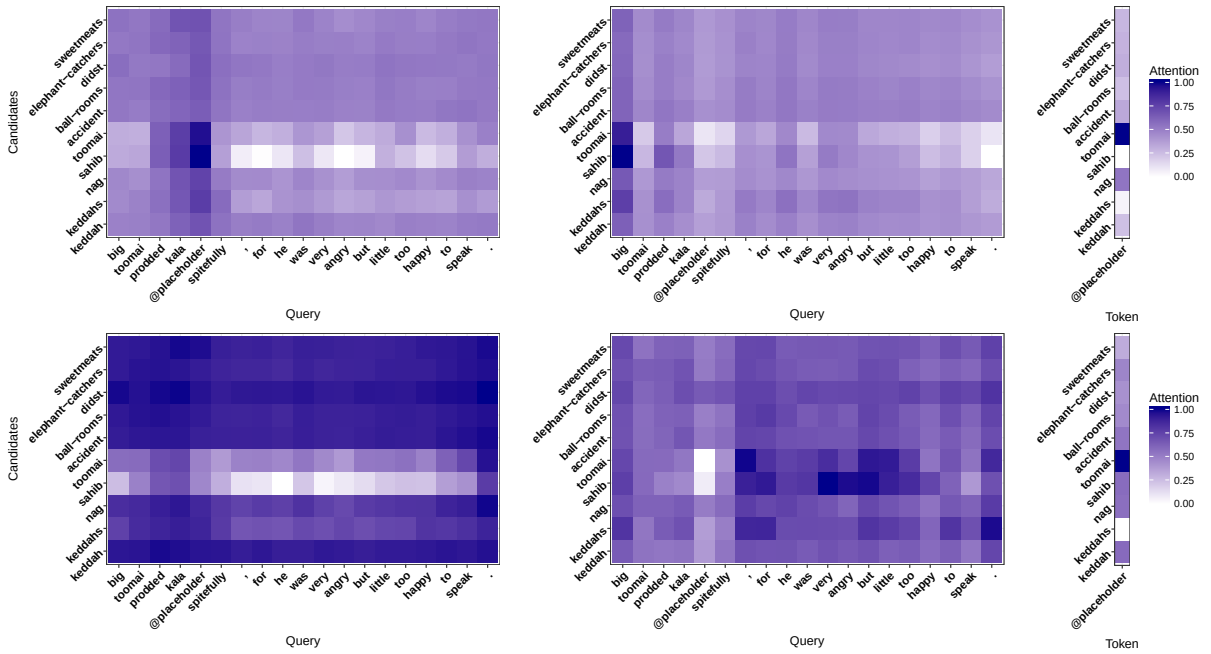


Figure 11: Layer-wise normalized attention visualization of "DGR" (top) and "DGR - (a) & (b) & (c)" (bottom) for a sample from the CBT-NE test set. Darker color illustrates higher attention. Figures only show the aggregated attention of candidates. The gold answer is "toomai".

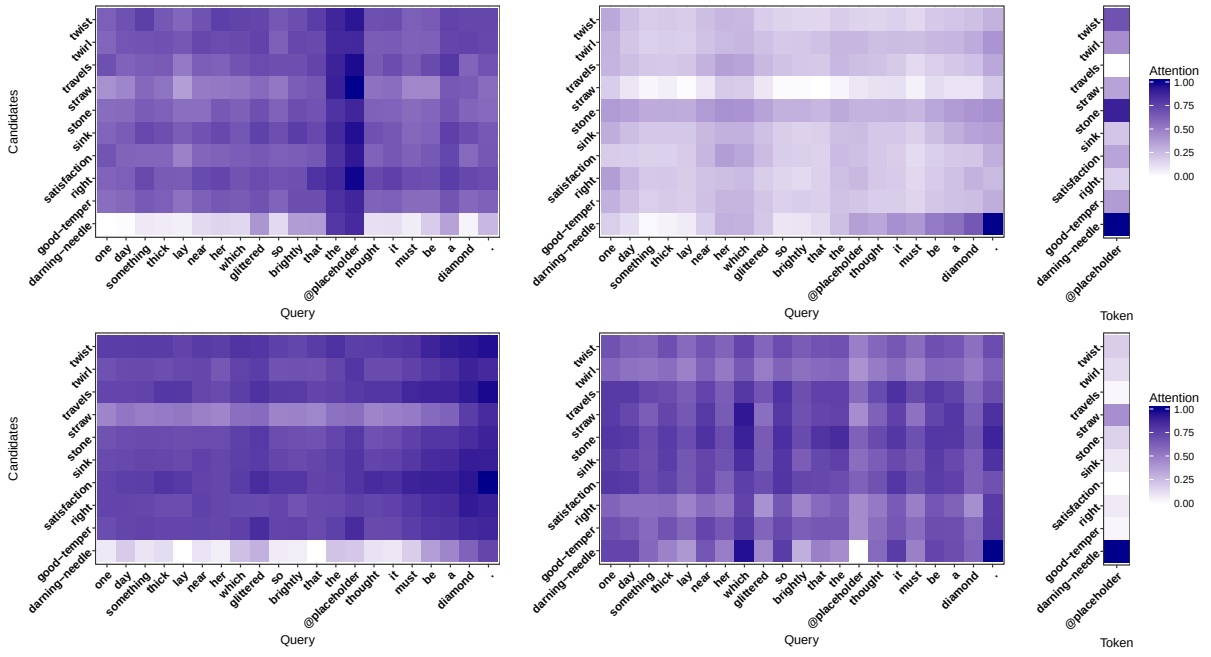


Figure 12: Layer-wise normalized attention visualization of "DGR" (top) and "DGR - (a) & (b) & (c)" (bottom) for a sample from the CBT-NE test set. Darker color illustrates higher attention. Figures only show the aggregated attention of candidates. The gold answer is "darning-needle".

Automated Fact Checking: Task formulations, methods and future directions

James Thorne
Department of Computer Science
University of Sheffield, UK
j.thorne@sheffield.ac.uk

Andreas Vlachos
Department of Computer Science
University of Sheffield, UK
a.vlachos@sheffield.ac.uk

Abstract

The recently increased focus on misinformation has stimulated research in fact checking, the task of assessing the truthfulness of a claim. Research in automating this task has been conducted in a variety of disciplines including natural language processing, machine learning, knowledge representation, databases, and journalism. While there has been substantial progress, relevant papers and articles have been published in research communities that are often unaware of each other and use inconsistent terminology, thus impeding understanding and further progress. In this paper we survey automated fact checking research stemming from natural language processing and related disciplines, unifying the task formulations and methodologies across papers and authors. Furthermore, we highlight the use of evidence as an important distinguishing factor among them cutting across task formulations and methods. We conclude with proposing avenues for future NLP research on automated fact checking.

Title and Abstract in Greek

Αναγνώριση Ψευδών Ισχυρισμών:
Ορισμοί, μέθοδοι και κατευθύνσης για μελλοντική έρευνα

Το πρόσφατα αυξημένο ενδιαφέρον για το φαινόμενο της παραπληροφόρησης έχει κινητοποιήσει ερευνητικές προσπάθειες για μεθόδους αναγνώρισης ψευδών ισχυρισμών. Η έρευνα σε αυτοματοποιημένες μεθόδους για το πρόβλημα αυτό διεξάγεται σε διάφορους επιστημονικούς κλάδους, όπως επεξεργασία φυσικής γλώσσας, μηχανική μάθηση, αναπαράσταση γνώσης, βάσεις δεδομένων και δημοσιογραφία. Αν και έχει γίνει σημαντική πρόοδος, τα άρθρα σχετικά με το αντικείμενο δημοσιεύονται στα συνέδρια και τα περιοδικά του κάθε κλάδου και χρησιμοποιούν διαφορετική ορολογία, με συνέπεια να δυσχεραίνεται η περαιτέρω πρόοδος. Στο παρόν άρθρο παρουσιάζουμε μια ανασκόπηση της έρευνας στον αυτόματη αναγνώριση ψευδών ισχυρισμών με κεντρικό άξονα την επεξεργασίας φυσικής γλώσσας, ενοποιώντας ορισμούς και μεθόδους που έχουν προταθεί στη βιβλιογραφία. Μια βασική διάκριση που διαχωρίζει τις ορισμούς και μεθόδους που έχουν προταθεί σε διαφορετικούς κλάδους είναι η χρήση ή μη αποδεικτικών στοιχείων. Η ανασκόπηση στο άρθρο αυτό καταλήγει με κατευθύνσεις για μελλοντική έρευνα στην επεξεργασία φυσικής γλώσσας σχετική με το πρόβλημα αυτό.

1 Introduction

The ability to rapidly distribute information on the internet presents a great opportunity for individuals and organizations to disseminate and consume large amounts of data on almost any subject matter. As many of the barriers required to publish information are effectively removed through the advent of social media, it is possible for content written by any individual or organization to reach an audience of hundreds of millions of readers (Allcott and Gentzkow, 2017). This enhanced ability of reaching audiences

Proceedings of the 27th International Conference on Computational Linguistics, pages 3346–3359
Santa Fe, New Mexico, USA, August 20-26, 2018.

can be used to spread both true and false information, and has become an important concern as it is speculated that this has affected decisions such as public votes, especially since recent studies have shown that false information reached greater audiences (Vosoughi et al., 2018). Consequently, there has been an increased demand for fact checking of online content.

In the domain of journalism, fact checking is the task of assessing whether claims made in written or spoken language are true. This is a task that is normally performed by trained professionals: the fact checker must evaluate previous speeches, debates, legislation and published figures or known facts, and use these, combined with reasoning to reach a verdict. Depending on the complexity of the claim, this process may take from less than one hour to a few days (Hassan et al., 2015a). Due to the large volume of content to be checked, this process has been the subject of calls from the journalism community to develop tooling to automate parts of this task (Cohen et al., 2011; Babakar and Moy, 2016; Graves, 2018). The process of fact checking requires researching and identifying evidence, understanding the context of information and reasoning about what can be inferred from this evidence. The goal of automated fact checking is to reduce the human burden in assessing the veracity of a claim.

In this paper we survey automated fact checking work from the viewpoint of natural language processing (NLP) and related fields such as machine learning, knowledge representation, databases and social media analysis. While it is important to consider social aspects, including how fact checking is communicated and incorporated into social media platforms, these are considered out of scope for this survey. There is a vast body of related works with different motivations and approaches that we can learn from to develop natural language technologies for fact checking. However, because these are conducted in a siloed manner and often without wider awareness of the issue, there is inconsistency in the definitions and lack of communication between disciplines, impeding understanding and further progress. For example, the classification of whether an article title can be supported by content in the body has been interchangeably referred to as both fake news detection (Pomerleau and Rao, 2017), stance classification (Ferreira and Vlachos, 2016) and incongruent headline detection (Chesney et al., 2017).

In this survey we aim to unify the definitions presented in related works and identify common concepts, datasets and modelling approaches. In this process, we summarize key results, highlight limitations and propose some open research challenges. In particular, we detail the different types of evidence used by different approaches and how they affect the NLP requirements on the systems developed. For example, many approaches rely on textual entailment/natural language inference (Dagan et al., 2009), while others rely on knowledge base construction techniques (Ji and Grishman, 2011).

The structure of this survey is as follows: we first discuss fact checking in the context of journalism as this provides definitions and distinctions on key terminology that will be used throughout in the remainder. We then proceed to discussing previous research in automated fact checking in terms of what inputs they expect, what outputs they return and the evidence used in this process. Following this, we provide an overview of the most commonly used datasets and the models developed and evaluated on them. Subsequently, we discuss work related to automated fact checking and conclude by proposing avenues for future NLP research.

2 Fact checking in journalism

Fact checking is an essential component in the process of news reporting. Journalism has been defined by scholars as a "discipline of verification" to separate it from "entertainment, propaganda, fiction, or art" (Shapiro et al., 2013). While fact checking and verification are often used interchangeably, recent work has sought to define them as two distinct but complementary processes (Silverman, 2016). In particular, verification is defined as "scientific-like approach of getting the fact and also the right facts" (Kovach and Rosenstiel, 2014), which often involves verifying the source, date and the location of materials. Fact-checking on the other hand "addresses the claim's logic, coherence and context" (Mantzarlis, 2015). Consequently, verification is a necessary and crucial first step in the process of fact checking as it assesses the trustworthiness of the contexts considered. We adopt this distinction for the remainder of this survey

as it helps highlight key differences between various automated approaches.[1]

A term that has become strongly associated with fact checking is "fake news", especially since its use in the context of the 2016 US presidential elections. However its popularity has resulted in its meaning becoming diluted, as it is used to label claims on a number of aspects not necessarily related to veracity (Vosoughi et al., 2018). The most prominent example of such usage of the term of fake news is its application to media organizations of opposing political sides. Furthermore, it is often grouped together with the term "hate speech" as in the case of recent legislation (Deutsche Welle, 2017), despite the latter being more related to the use of emotive language instead of truth assessment (Rahman, 2012). Therefore we avoid using this term in this survey.

A further consideration is the relation of fact checking with the terms misinformation and disinformation. While the former is the distribution of information that may not be accurate or complete, the latter additionally assumes malicious motives to mislead the reader (Jowett and O' Donnell, 2006). Due to this additional consideration, disinformation is considered a subset of misinformation. Fact checking can help detect misinformation, but not distinguish it from disinformation.

3 Towards Automated Fact Checking

The increased demand for fact checking has stimulated a rapid progress in developing tools and systems to automate the task, or parts thereof (Babakar and Moy, 2016; Graves, 2018). Thus, there has been a diverse array of approaches tailored to specific datasets, domains or subtasks. While they all share the same goal, these approaches utilize different definitions of what the task being automated is. In the following discussion, we highlight the differences between the task definitions used in previous research in the following axes: input, i.e. what is being fact checked, output, i.e. what kinds of verdicts are expected, and the evidence used in the fact checking process.

3.1 Inputs

We first consider the inputs to automated fact checking approaches as their format and content influences the types of evidence used in this process. A frequently considered input to fact checking approaches is subject-predicate-object triples, e.g. `(London, capital_of, UK)`, and is popular across different research communities, including NLP (Nakashole and Mitchell, 2014), data mining (Ciampaglia et al., 2015) and Web search (Bast et al., 2015). The popularity of triples as input stems from the fact that they facilitate fact checking against (semi-)structured knowledge bases such Freebase (Bollacker et al., 2008). However, it is important to acknowledge that approaches using triples as input implicitly assume a non-trivial level of processing in order to convert text, speech or other forms of claims into triples, a task falling under the broad definition of natural language understanding (Woods, 1973).

A second type of input often considered in automated fact checking is textual claims. These tend to be short sentences constructed from longer passages, which is a practice common among human fact checkers on dedicated websites such as PolitiFact[2] and Full Fact[3] with the purpose of including only the context relevant to the claim from the original passage. The availability of fact checked claims on such websites has rendered this format very popular among researchers in NLP.

A useful taxonomy of textual claims was proposed in the context of the HeroX fact checking challenge (Francis and Full Fact, 2016), in which four types of claims were considered:

- **numerical claims** involving numerical properties about entities and comparisons among them
- **entity and event properties** such as professional qualifications and event participants
- **position statements** such as whether a political entity supported a certain policy
- **quote verification** assessing whether the claim states precisely the source of a quote, its content, and the event at which it supposedly occurred.

[1] Note that Mantzarlis and Silverman disagree on whether fact-checking should be considered a subset of verification, or just overlapping with it. While this is in an important question, we do not address it in this survey as it does not affect our analysis.

[2] `http://www.politifact.com/`

[3] `http://www.fullfact.org/`

While some of these claims could be represented as triples, they typically require more complex representations; for example, events typically need to be represented with multiple slots (Doddington et al., 2004) to denote their various participants. Regardless of whether textual claims are verified via a subject-predicate-object triple representation or as text, it is often necessary to disambiguate the entities and their properties. For example, the claim 'Beckham played for Manchester United' is true for the soccer player 'David Beckham', but not true (at the time of writing) for the American football player 'Odel Beckham Jr'. Correctly identifying, disambiguating and grounding entities is the task of Named Entity Linking (McNamee and Dang, 2009). While this must be performed explicitly if converting a claim to a subject-predicate-object triple with reference to a knowledge base, it may also be performed implicitly through the retrieval of appropriate textual evidence if fact checking against textual sources.

Finally, there have been approaches that consider an entire document as their input. These approaches must first identify the claims and then fact check them. This increases the complexity of the task, as approaches are required to extract the claims, either in the form of triples by performing relation extraction (Vlachos and Riedel, 2015) or through a supervised sentence-level classification (Hassan et al., 2015b).

3.2 Sources of evidence

The type of evidence that is used for fact checking influences the model and the types of outputs that the fact checking system can produce. For example, whether the output is a label or whether a justification can be produced depends largely on the information available to the fact checking system.

We first consider task formulations that do not use any evidence beyond the claim itself when predicting its veracity such as the one by Rashkin et al. (2017). In these instances, surface-level linguistic features in the claims are associated with the predicted veracity. We contrast this to how journalists work when fact checking, where they must find knowledge relating to the fact and evaluate the claim given the evidence and context when making the decision as to whether a claim is true or false. The predictions made in task formulations that do not consider evidence beyond the claim are based on surface patterns of how the claim is written rather than considering the current state of the world.

Wang (2017) incorporate additional metadata in fact checking such as the originator of the claim, speaker profile and the media source in which the claim is presented. While these do not provide evidence grounding the claim, the additional context can act as a prior to improve the classification accuracy, and can be used as part of the justification of a verdict.

Knowledge graphs provide a rich collection of structured canonical information about the world stored in a machine readable format that could support the task of fact checking. We observe two types of formulations using this type of evidence. The first approach is to identify/retrieve the element in the knowledge graph that provides the information supporting or refuting the claim at question. For example, Vlachos and Riedel (2015) and Thorne and Vlachos (2017) identify the subject-predicate-object triples from small knowledge graphs to fact check numerical claims. Once the relevant triple had been found, a truth label is computed through a rule based approach that considers the error between the claimed values and the retrieved values from the graph. The key limitation in using knowledge graphs as evidence in this fashion is that it assumes that the true facts relevant to the claim are present in them. However, it is not feasible to capture and store every conceivable fact in the graph in advance of knowing the claim.

The alternative use of a knowledge graph as evidence is to consider its topology in order to predict how likely a claim (expressed as a node in the graph) is to be true (Ciampaglia et al., 2015). While graph topology can be indicative of the plausibility of a fact, nevertheless if a fact is unlikely to occur that does not negate its truthfulness. Furthermore, improbable but believable claims are more likely to become viral and thus in greater need of verification by fact checkers.

Text, such as encyclopedia articles, policy documents, verified news and scientific journals contain information that can be used to fact check claims. Ferreira and Vlachos (2016) use article headlines (single sentences) as evidence to predict whether an article is for, against or observing a claim. The Fake News Challenge (Pomerleau and Rao, 2017) also formulated this part of the fact checking process in the same way, but in contrast to (Ferreira and Vlachos, 2016), entire documents are used as evidence, thus allowing for evidence from multiple sentences to be combined.

The Fact Extraction and VERification (FEVER) task (Thorne et al., 2018) requires combining information from multiple documents and sentences for fact checking. Unlike the aforementioned works which use text as evidence, the evidence is not given but must be retrieved from Wikipedia, a large corpus of encyclopedic articles. Scaling up even further, the triple scoring task of the WSDM cup (Bast et al., 2017) required participants to assess knowledge graph triples considering both Wikipedia and a portion of the web-scale ClueWeb dataset (Gabrilovich et al., 2013).

A different source of text-based evidence is repositories of previously fact checked claims (Hassan et al., 2017b). Systems using such evidence typically match a new claim to claim(s) in such a repository and return the label of the match if one is found. This type of evidence enables the prediction of veracity labels instead of only whether a claim is supported or refuted. However, this evidence is limiting fact checking only to claims similar to the ones already existing in the repository.

As an alternative to using knowledge stored in texts or databases as evidence, aggregate information on the distribution of posts on social networks can provide insights into the veracity of the content. Rumor veracity prediction is the assessment of the macro level behaviors of users' interactions with and distribution of content to predict whether the claims in the content are true or false (Derczynski and Bontcheva, 2015; Derczynski et al., 2017). This crowd behavior provide useful insights, especially in cases where textual sources or structured knowledge bases may be unavailable.

The trustworthiness of the sources used as evidence, i.e. the verification aspect of fact checking in journalistic terms, is rarely considered by automated approaches, despite its obvious importance. Often the sources of evidence are considered given, e.g. Wikipedia or Freebase, as this facilitates development and evaluation, especially when multiple models are considered. Nevertheless, approaches relying on Twitter metadata often consider credibility indicators for the tweets used as evidence in their fact checking process (Liu et al., 2015). Similar to journalism, trustworthiness assessment is often considered as a separate task and we discuss it further in the Related Work section.

3.3 Output

The simplest model for fact checking is to label a claim as true or false as a binary classification task (Nakashole and Mitchell, 2014). However, we must also consider that it is possible in natural language to be purposefully flexible with the degree of truthfulness of the information expressed or express a particular bias using true facts. Journalistic fact checking agencies such as Politifact model the degree of truthfulness on a multi-point scale (ranging from true, mostly-true, half-true, etc). Rather than modeling fact checking as a binary classification, Vlachos and Riedel (2014) suggested modeling this degree of truthfulness as an ordinal classification task. However, the reasoning behind why the manual fact checking agencies have applied these more fine-grained labels is complex, sometimes inconsistent[4] and likely to be difficult to capture and express in our models. Wang et al. (2017) and Rashkin et al. (2017) expect as output multiclass labels following the definitions by the journalists over a multi-point scale but ignoring the ordering among them.

The triple scoring task of the WSDM cup (Bast et al., 2017) expected as output triples scored within a numerical range indicating how likely they are to be true. The evaluation consisted of calculating both the differences between the predicted and the manually annotated scores, as well as the correlation between the rankings produced by the systems and the annotators.

Ferreira and Vlachos (2016) expect as output whether a claim is supported, refuted or just reported by a news article headline. Pomerleau and Rao (2017) added an extra label for the article being irrelevant to the claim, and consider the full article instead of the headline. While this output can be used in the process of fact checking, it is not a complete fact check.

In FEVER (Thorne et al., 2018) the output expected consists of two components: a 3-way classification label about whether a claim is supported/refuted by Wikipedia, or there is not enough information in the latter to Wikipedia it, and in the case of the first two labels, the sentences forming the evidence to reach the verdict. In effect this combines the multiclass labeling output with a ranking task over Wikipedia

[4] http://www.poynter.org/news/can-fact-checkers-agree-what-true-new-study-doesnt-point-answer

sentences. If the label predicted is correct but the evidence retrieved is incorrect, then the answer is considered incorrect, highlighting the importance of evidence in this task formulation. Nevertheless, this output cannot be considered a complete fact check though, unless we restrict world knowledge to Wikipedia. The use of full world knowledge in the justifications was considered in the HeroX shared task (Francis and Full Fact, 2016); while this allowed for complete fact checks, it also necessitated manual verification of the outputs.

4 Fact Checking Datasets

There are currently a limited number of published datasets resources for fact checking. Vlachos and Riedel (2014) collected and released 221 labeled claims in the political domain that had been fact checked by Politifact and Channel4.[5] For each labeled claim, the speaker or originator is provided alongside hyperlinks to the sources used by the fact checker as evidence. The sources range from statistical tables and Excel spreadsheets to PDF reports and documents from The National Archives. While these sources can be readily assessed by human fact checkers, the variety and lack of structure in the evidence makes it difficult to apply machine learning-based approaches to select evidence as separate techniques may be required for the each different document format. Where documents are provided as evidence, we do not know which portions of the document pertain to the claim, which compounds the difficulty of assessing and evaluating a fact checking system, given the need for evidence and accountability. Furthermore, with such a limited number of samples, the scale of this dataset precludes its use for developing machine learning-based fact checking systems.

Wang (2017) released a dataset similar to but an order of magnitude larger than that of Vlachos and Riedel (2014) containing 12.8K labeled claims from Politifact. In addition to the claims, meta-data such as the speaker affiliation and the context in which the claim appears in (e.g. speech, tweet, op-ed piece) is provided. At this scale, this dataset can support training and evaluating a machine learning-based fact checking system. However, the usefulness of the dataset may be limited due the claims being provided without machine-readable evidence beyond originator metadata, meaning that systems can only resort to approaches to fact checking such as text classification or speaker profiling.

Rashkin et al. (2017) collated claims from Politifact without meta-data. In addition, the authors published a dataset of 74K news articles collected from websites deemed as Hoax, Satire, Propaganda and Trusted News. The prediction of whether a news article is true or false is modeled as the task of predicting whether an article originates from one of the websites deemed to be "fake news" according to a US News & World report.[6] This allows systems to fact check using linguistic features, but does not consider evidence, or which aspects of a story are true or false.

Ferreira and Vlachos (2016) released a dataset for rumor debunking using data collected from the Emergent project (Silverman, 2015). For 300 claims, 2,595 corresponding news articles were collected and their stances were labeled as for, against or observing a claim. This dataset was extended for the 2017 Fake News Challenge (Pomerleau and Rao, 2017) dataset which consisted of approximately 50K headline and body pairs derived from the original 300 claims and 2,595 articles.

The HeroX Fast and Furious Fact Checking Challenge (Francis and Full Fact, 2016) released 90 (41 practice and 49 test) labeled claims as part of the competition. Because part of the challenge required identification of appropriate source material, evidence for these claims was not provided and therefore manual evaluation was needed on case-by-case basis. As with the claims collected by Vlachos and Riedel (2014), the size of the dataset prevents its use for training a machine learning-based fact checking system. However, the broad range of types of claims in this dataset highlights a number of forms of misinformation to help identify the requirements for fact checking systems.

Thorne et al. (2018) introduced a fact checking dataset containing 185K claims about properties of entities and concepts which must be verified using articles from Wikipedia. While this restricts the pool of evidence substantially compared to HeroX, it allowed for a challenging evidence selection subtask

[5] http://channel4.com/news/factcheck

[6] www.usnews.com/news/national-news/articles/2016-11-14/avoid-these-fake-news-sites-at-all-costs

that could be feasibly annotated manually at a large scale in the form of sentences selected as evidence. Combined with the labels on the claims, these machine readable fact checks allow training machine learning-based fact checking models.

5 Methods

The majority of the methods used for automated fact checking are supervised: learning a text classifier from some form of labeled data that is provided at training. This trait that is independent of the task input or what sources of evidence are considered. Vlachos and Riedel (2014) suggested fact checking using supervised models, making use of existing statements that had been annotated with verdicts by fact checking agencies and journalists. Wang (2017) and Rashkin et al. (2017) applied this approach to classify the veracity of fake news stories. The main limitation of text classification approaches is that fact checking a claim requires additional world knowledge, typically not provided with the claim itself. While language can indicate whether a sentence is factual (Nakashole and Mitchell, 2014), credible sounding sentences may also be inherently false. Text classification on claims alone has been used for the related task of detecting fact-check worthy claims (Hassan et al., 2017a).

Ciampaglia et al. (2015) use network analysis to predict whether an unobserved triple is likely to appear in a graph by modeling the task as a path ranking problem (Lao et al., 2011). The truth verdict is derived from the cost of traversing a path between the two entities under transitive closure, weighted by the degree of connectedness of the path. Nakashole and Mitchell (2014) combine linguistic features on the subjectivity of the language used with the co-occurrence of a triple with other triples from the same topic, a form of collective classification.

Ferreira and Vlachos (2016) modeled fact checking as a form of Recognizing Textual Entailment (RTE) (Dagan et al., 2009; Bowman et al., 2015), predicting whether a premise, typically (part of) a news article, is *for*, *against*, or *observing* a given claim. The same type of model was used by most of the 50 participating teams in the Fake News Challenge (Pomerleau and Rao, 2017), including most of the top entries (Riedel et al., 2017; Hanselowski et al., 2017).

The RTE-based models assume that the textual evidence to fact check a claim is given. Thus they are inapplicable in cases where this is not provided (as in HeroX), or it is a rather large textual resource such as Wikipedia (as in FEVER). For the latter, Thorne et al. (2018) developed a pipelined approach in which the RTE component is preceded by a document retrieval and a sentence selection component. There is also work focusing exclusively on retrieving sentence-level evidence from related documents for a given claim (Hua and Wang, 2017).

Vlachos and Riedel (2015) and Thorne and Vlachos (2017) use distantly supervised relation extraction (Mintz et al., 2009) to identify surface patterns in text which describe relations between two entities in a knowledge graph. Because these fact checking approaches only focus on statistical properties of entities, identification of positive training examples is simplified to searching for sentences containing numbers that are approximately equal to the values stored in the graph. Extending this approach to entity-entity relations would pose different challenges, as a there may be many relations between the same pair entities that would need to be accurately distinguished for this approach to be used.

A popular type of model often employed by fact checking organizations in their process is that of matching a claim with existing, previously fact checked ones. This reduces the task to sentence-level textual similarity as suggested by Vlachos and Riedel (2014) and implemented in ClaimBuster (Hassan et al., 2017b), Truthteller by The Washington Post[7] and one of the two modes of Full Fact's Live platform.[8] However, it can only be used to fact check repeated or paraphrased claims.

Long et al. (2017) extend the models produced by Wang (2017) and improve accuracy of a simple fact checking system through more extended profiling of the originators of the claims. The most influential feature in this model is the *credit history* of the originator, a metric describing how often the originator's claims are classified as false. This feature introduces a bias in the model that fits with the adage "never

[7] https://www.knightfoundation.org/articles/debuting-truth-teller-washington-post-real-time-lie-detection-service-your-service-not-quite-yet

[8] https://fullfact.org/blog/2017/jun/automated-fact-checking-full-fact/

trust a liar". In the case of previously unannotated sources which may haven no recorded history, this feature would be unavailable. Furthermore, the strong reliance on credit history has some important ethical implications that need to be carefully considered. Despite these limitations, speaker profiling has been shown to be effective in other related studies (Gottipati et al., 2013; Long et al., 2016) and is discussed further in Section 6.

6 Related Tasks

Verification In Section 2, we highlighted the difference between verification and fact checking. While the models considered in Section 4 operate under a closed-world paradigm where we assume that all evidence provided is true, for these fact checking technologies to scale-to and operate on the web, we must also consider information that is of unknown veracity as evidence.

Methods of predicting the authoritativeness of web pages such as PageRank (Brin and Page, 1998) only consider the hyperlink topology. TrustRank (Gyöngyi et al., 2004) provides a framework which incorporates annotated information and predicts the trustworthiness of pages based on graph-connectedness to known-bad nodes rather than the information content. An alternative is Knowledge-based Trustworthiness scoring (Dong et al., 2015), which allows predicting whether the facts extracted from a given document page are likely to be accurate given the method used to extract the facts and the website in which the document is published.

Common Sense Reasoning Fact checking requires the ability to reason about arguments with common sense knowledge. This requires developing systems that go beyond recognizing semantic phenomena more complex than those typically considered in textual entailment tasks. Habernal et al. (2018) introduced a new task and dataset for predicting which implicit *warrant* (the rationale of an argument) is required to support a claim from a given premise. Angeli and Manning (2014) proposed a method of extracting common sense knowledge from WordNet for reasoning about common sense knowledge using Natural Logic and evaluated their approach on a subset of textual entailment problems in the FraCaS test suite (Cooper et al., 1996). It is important to build systems that can reason about both explicit world knowledge and implicit common sense knowledge is an essential step towards automating fact checking.

Subjectivity and Emotive Language Rashkin et al. (2017) assess the reliability of entire news articles by predicting whether the document originates from a website classified as Hoax, Satire or Propaganda. This work is an instance of subjective language detection and does not represent evidence-based fact checking. The authors used supervised classifiers augmented with lexicons including the Linguistic Inquiry and Word Count (LIWC) (Pennebaker et al., 2015), a sentiment lexicon (Wilson et al., 2005), a hedging lexicon (Hyland, 2015), and a novel 'dramatic' language lexicon to identify emotive, subjective and sensational language in the article bodies. Furthermore, analysis of the lexical features using a logistic regression classifier shows that the highest weighted (most distinguishing) features for the unreliable sources included the use of hedge words (such as 'reportedly') or words pertaining to divisive topics (such as 'liberals' or 'Trump'). The authors apply a similar model to the prediction of claims made by politicians from claims collected from Politifact. The addition of the LIWC lexicon which provides an indication of emotive tone and authenticity marginally improved the classification accuracy for simple lexical models.

Deceptive Language Detection There are linguistic cues and features in written text that are useful in identifying deceptive language (Zhou et al., 2004). In the context of detecting deceptive user-generated content - a specific form of disinformation, Mihalcea and Strapparava (2009) use a simple lexical classification model without further feature engineering. Analysis of the model identifies a number of word classes of the LIWC lexicon which pertain only to the deceptive texts. Ott et al. (2011) incorporate the use of psycholinguistic cues to improve classification accuracy. Mihalcea and Strapparava (2009) found that truthful texts were more likely to contain words belonging to the 'optimistic' LIWC class such as 'best', 'hope', and 'determined'. This is corroborated by the study of sentiment in deceptive texts (Ott et al., 2013) which also identified that texts with negative sentiment were more likely to be deceptive. These

feature classes however may be an artifact of the data generation process as crowd-sourced volunteers were first asked to write an honest text and rewrite it to make it deceptive.

Feng et al. (2012) detect deceptive texts and customer-generated reviews through the use of syntactic style rather word-based content. Non-terminal nodes of the constituency parse trees are used as features in conjunction with a lexical model to increase the accuracy over using words alone. Hai et al. (2016) identify deceptive reviews also using lexical features. However, rather than relying on labeled data, the authors induce labels over an unlabeled dataset through a semi-supervised learning approach that exploits a minimal amount labeled data from related tasks in a multi-task learning set up.

Even though linguistic content, emotive language and syntax are useful indicators for detecting deceit, the truthfulness of a statement depends also on the context. Without considering these factors these approaches cannot be used to fact check information alone.

Rumor Detection Rumor detection (Qazvinian et al., 2011) is the task of identifying unverified reports circulating on social media. A prediction is typically based on language subjectivity and growth of readership through a social network. While these are important factors to consider, a sentence can be true or false regardless of whether it is a rumor (Zubiaga et al., 2018).

Speaker Profiling Identifying claims that do not fit with the profile of the originator may provide insights as to whether the information is truthful or not. Furthermore, determining which topics the originator is truthful about may allow for generation of a risk-based approach to fact checking. As mentioned earlier, Long et al. (2017) introduced a notion of credit history which increases the classification accuracy for fake news detection. However, this notion doesn't account for which topics the originator lies about. Furthermore, the assumption that each source has an overall trustworthiness score to be attached to every claim from there is not a valid one, since inaccurate information may be found even on the most reputable of sources.

An alternative is to consider the compatibility of a claim with respect to the originator's profile. This remains an open research area. Feng and Hirst (2013) perform the inverse of this task for deceptive language detection in product reviews by creating an average profile for the targets (products) and using the distance between a review and the target as a feature. The shortcomings of this method are that number of reviews are required for each target. Considering the tasks of verifying automatically extracted information or fact checking for politics, for a new topic the challenge is that there may be insufficient data to create a profile for it.

Pérez-Rosas and Mihalcea (2015) identify author characteristics (such as age and gender) that influence the linguistic choices made by the authors when fabricating information in product reviews. As the affiliation, age and gender of most politicians is public-domain knowledge, it is conceivable that these features may assist fact checking political claims. While the use of meta-data does improve classification accuracy (Wang et al., 2017; Long et al., 2017), for fact checking we must consider the meaning of the claim with respect to ground truth rather than based on the linguistic style of the originator.

Click Bait Detection Intentionally misleading headlines or titles that are designed specifically to encourage a user to click through and visit a website are called click bait. Studies into the detection of click bait have have yielded positive results from relatively simple linguistic features (Chen et al., 2015; Potthast et al., 2016; Chakraborty et al., 2016). These approaches only consider the article headline and do not make use of evidence. We contrast this to the task of detecting headlines which are incongruent to the document body (Chesney et al., 2017), where existing methods for recognizing textual entailment such as those used for the Fake News Challenge (Pomerleau and Rao, 2017) can be applied.

7 Open Research Challenges

In this paper, we provided an NLP-motivated overview of fact checking, considering the case for evidence - similar to how the task is performed by journalists. We highlighted existing works that consider fact checking of claims expressed in text or via knowledge base triples and pointed out their shortcomings, given the need to justify their decisions using evidence.

We did not identify approaches that make use of open-world knowledge. Assessing the ability of a system in an open-world setting will be difficult; the HeroX challenge resorted to fully manual evaluation, which is difficult to scale up, especially for the purposes of developing machine learning-based approaches. Thus we need to consider how to address the information retrieval challenge of the task, including its evaluation. We must also consider the verification of the evidence used, which is ignored under the closed world assumption. Ideally we should consider verification jointly with fact checking, which is in fact how it is conducted in journalism.

The relative scarcity of resources designed explicitly for fact checking, highlights the difficulties in capturing and formalizing the task, especially considering the strong interest in it. While recently published large scale resources such as FEVER can stimulate progress, they only consider simple short sentences (8 words long on average). Thus, there is scope to fact check compound information or complex sentences, and scale up to fact checking at the document level.

Furthermore, in FEVER the justification for the labels is restricted to sentences selected from Wikipedia. This is much unlike the rationales produced by human fact checkers, who synthesize information. The generation of such rationales has attracted attention only recently in NLP (Ling et al., 2017), and automated fact checking could provide an ideal testbed to develop it further.

While text is often used to make a claim, often the evidence need for fact checking appears in other modalities, such as images and videos. Given the recent interest in multi-modal NLP, we argue that this would be an important direction for future research, especially when considering that a lot of the verification efforts in journalism are focused on identifying forged images and footage.

Finally, it is important to acknowledge that the complexity of the fact checking conducted by journalists is for the moment beyond the abilities of the systems we can develop due to the rather complex reasoning needed. Even a simple short statement such as that the "UK, by leaving the EU, will take back control of roughly £350 million per week" takes a substantial amount of work to be checked.[9] In this fact check for example, the first step is to adjust the claim to render it more accurate in terms of its meaning so that the actual fact check can proceed. While this complexity can help stimulate progress in NLP and related fields, it should also calibrate our expectations and promises to society.

Acknowledgements

Andreas Vlachos is supported by the EU H2020 SUMMA project (grant agreement number 688139). The authors would like to thank Dhruv Ghulati and Alexios Mantzarlis for their feedback.

References

Hunt Allcott and Matthew Gentzkow. 2017. Social media and fake news in the 2016 election. Technical report, National Bureau of Economic Research.

Gabor Angeli and Christopher D. Manning. 2014. NaturalLI : Natural Logic Inference for Common Sense Reasoning. *Proceedings of EMNLP*, pages 534–545.

Mevan Babakar and Will Moy. 2016. The State of Automated Factchecking. Technical report.

Hannah Bast, Björn Buchhold, and Elmar Haussmann. 2015. Relevance scores for triples from type-like relations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 243–252. ACM.

Hannah Bast, Björn Buchhold, and Elmar Haussmann. 2017. Overview of the Triple Scoring Task at the WSDM Cup 2017. *WSDM Cup*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. *SIGMOD 08 Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

[9] https://fullfact.org/europe/350-million-week-boris-johnson-statistics-authority-misuse/

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference.

S Brin and L Page. 1998. The anatomy of a large scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1/7):107–17.

Abhijnan Chakraborty, Bhargavi Paranjape, Sourya Kakarla, and Niloy Ganguly. 2016. Stop Clickbait: Detecting and preventing clickbaits in online news media. *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2016*, pages 9–16.

Yimin Chen, Niall J. Conroy, and Victoria L. Rubin. 2015. Misleading Online Content. *Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection - WMDD '15*, (November):15–19.

Sophie Chesney, Maria Liakata, Massimo Poesio, and Matthew Purver. 2017. Incongruent Headlines: Yet Another Way to Mislead Your Readers. In *Natural Language Processing meets Journalism workshop at EMNLP 2017*, number 1, pages 56–61.

Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M. Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. 2015. Computational fact checking from knowledge networks. *PLoS ONE*, 10(6):1–13.

Sarah Cohen, Chengkai Li, Jun Yang, and Cong Yu. 2011. Computational Journalism: a call to arms to database researchers. *Proceedings of the 5th Biennial Conference on Innovative Data Systems Research (CIDR 2011) Asilomar, California, USA.*, (January):148–151.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, et al. 1996. Using the framework. Technical report, Technical Report LRE 62-051 D-16, The FraCaS Consortium.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, 15(4):i–xvii.

Leon Derczynski and Kalina Bontcheva. 2015. Veracy in Digital Social Networks. *Pheme.eu*.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. SemEval-2017 Task 8: RumourEval: Determining rumour veracity and support for rumours.

Deutsche Welle. 2017. Germany to force facebook, twitter to delete hate speech. `http://www.dw.com/en/germany-to-force-facebook-twitter-to-delete-hate-speech/a-37927085`. Accessed: 2018-09-30.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The Automatic Content Extraction (ACE) Program – Tasks, Data, and Evaluation. In *Proceedings of the Fourth Conference on Language Resources and Evaluation*.

Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. 2015. Knowledge-based trust: Estimating the trustworthiness of web sources. *Proc. VLDB Endow.*, 8(9):938–949, May.

Vanessa Wei Feng and Graeme Hirst. 2013. Detecting Deceptive Opinions with Profile Compatibility. *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, (October):338–346.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Syntactic stylometry for deception detection. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2. Association for Computational Linguistics*, (July):171–175.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1163–1168, San Diego, California, June. Association for Computational Linguistics.

Diane Francis and Full Fact. 2016. Fast & furious fact check challenge. `https://www.herox.com/factcheck/`. Accessed: 2018-09-30.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase annotation of ClueWeb corpora, Version 1 (Release date 2013-06-26, Format version 1, Correction level 0), June.

Swapna Gottipati, Minghui Qiu, Liu Yang, Feida Zhu, and Jing Jiang. 2013. Predicting User ' s Political Party. pages 177–191.

Lucas Graves. 2018. Understanding the Promise and Limits of Automated Fact-Checking. Technical report, Reuters Institute, University of Oxford.

Z Gyöngyi, H Garcia-Molina, and J Pedersen. 2004. Combating web spam with TrustRank. *Proceedings of the Thirtieth international conference on Very large data bases*, 30:576–587.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. 2018. The argument reasoning comprehension task: Identification and reconstruction of implicit warrants. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1930–1940.

Zhen Hai, Peilin Zhao, Peng Cheng, Peng Yang, Xiao-Li Li, and Guangxia Li. 2016. Deceptive Review Spam Detection via Exploiting Task Relatedness and Unlabeled Data. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP-16)*, pages 1817–1826.

Andreas Hanselowski, Avinesh P.V.S, Benjamin Schiller, and Felix Caspelherr. 2017. Description of the system developed by team athene in the fnc-1. Technical report.

Naeemul Hassan, Bill Adair, James Hamilton, Chengkai Li, Mark Tremayne, Jun Yang, and Cong Yu. 2015a. The quest to automate fact-checking. In *Proceedings of the 2015 Computation+Journalism Symposium*.

Naeemul Hassan, Chengkai Li, and Mark Tremayne. 2015b. Detecting Check-worthy Factual Claims in Presidential Debates. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. ACM*.

Naeemul Hassan, Fatma Arslan, Chengkai Li, and Mark Tremayne. 2017a. Toward automated fact-checking: Detecting check-worthy factual claims by claimbuster. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1803–1812. ACM.

Naeemul Hassan, Anil Kumar Nayak, Vikas Sable, Chengkai Li, Mark Tremayne, Gensheng Zhang, Fatma Arslan, Josue Caraballo, Damian Jimenez, Siddhant Gawsane, Shohedul Hasan, Minumol Joseph, and Aaditya Kulkarni. 2017b. ClaimBuster: the first-ever end-to-end fact-checking system. *Proceedings of the VLDB Endowment*, 10(12):1945–1948.

Xinyu Hua and Lu Wang. 2017. Understanding and detecting supporting arguments of diverse types. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 203–208. Association for Computational Linguistics.

Ken Hyland. 2015. Metadiscourse. *The International Encyclopedia of Language and Social Interaction*, (April 2015):1–11.

Heng Ji and Ralph Grishman. 2011. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1148–1158, Stroudsburg, PA, USA. Association for Computational Linguistics.

Garth S. Jowett and Victoria O' Donnell. 2006. What is propaganda, and how does it differ from persuasion? In *Propaganda and Misinformation*, chapter 1. Sage Publishers.

Bill Kovach and Tom Rosenstiel. 2014. *The elements of journalism: What newspeople should know and the public should expect*. Three Rivers Press (CA).

N Lao, T Mitchell, and WW Cohen. 2011. Random walk inference and learning in a large scale knowledge base. *Proceedings of the Conference . . .*, (March):529–539.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 158–167.

Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, and Sameena Shah. 2015. Real-time rumor debunking on twitter. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1867–1870, New York, NY, USA. ACM.

Yunfei Long, Qin Lu, Yue Xiao, Minglei Li, and Chu Ren Huang. 2016. Domain-specific user preference prediction based on multiple user activities. *Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016*, pages 3913–3921.

Yunfei Long, Qin Lu, Rong Xiang, Minglei Li, and Chu-Ren Huang. 2017. Fake News Detection Through Multi-Perspective Speaker Profiles. *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 2:252–256.

Alexios Mantzarlis. 2015. Will verification kill fact-checking? `https://www.poynter.org/news/will-verification-kill-fact-checking`. Accessed: 2018-09-30.

Paul McNamee and Hoa Trang Dang. 2009. Overview of the tac 2009 knowledge base population track.

Rada Mihalcea and Carlo Strapparava. 2009. The Lie Detector: Explorations in the Automatic Recognition of Deceptive Language. *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, (August):309–312.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, (August):1003–1011.

Ndapandula Nakashole and Tom M Mitchell. 2014. Language-Aware Truth Assessment of Fact Candidates. *Acl*, pages 1009–1019.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 309–319, Stroudsburg, PA, USA. Association for Computational Linguistics.

Myle Ott, Claire Cardie, and Jeffrey T. Hancock. 2013. Negative deceptive opinion spam. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 497–501. Association for Computational Linguistics.

James W Pennebaker, Ryan L Boyd, Kayla Jordan, and Kate Blackburn. 2015. The Development and Psychometric Properties of LIWC2015. *Environment and Planning D: Society and Space*.

Verónica Pérez-Rosas and Rada Mihalcea. 2015. Experiments in Open Domain Deception Detection. (September):1120–1125.

Dean Pomerleau and Delip Rao. 2017. Fake News Challenge. \url{http://fakenewschallenge.org/}.

Martin Potthast, Sebastian Köpsel, Benno Stein, and Matthias Hagen. 2016. Clickbait Detection. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9626(1):810–817.

Vahed Qazvinian, Emily Rosengren, Dragomir R Radev, and Qiaozhu Mei. 2011. Rumor has it : Identifying Misinformation in Microblogs. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1589–1599.

Jacquelyn Rahman. 2012. The n word: Its history and use in the african american community. *Journal of English Linguistics*, 40(2):137–171.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2931–2937. Association for Computational Linguistics.

Benjamin Riedel, Isabelle Augenstein, George Spithourakis, and Sebastian Riedel. 2017. A simple but tough-to-beat baseline for the Fake News Challenge stance detection task. *CoRR*, abs/1707.03264.

Ivor Shapiro, Colette Brin, Isabelle Bedard-Brule, and Kasia Mychajlowycz. 2013. Verification as a strategic ritual. *Journalism Practice*, 7(6):657–673.

Craig Silverman. 2015. Lies , Damn Lies , and Viral Content: How News Websites Spread (and Debunk) Online Rumors, Unverified Claims, and Misinformation. *Columbia Journalism School*.

Craig Silverman. 2016. Verification handbook: Additional materials. `http://verificationhandbook.com/additionalmaterial/`. Accessed: 2018-09-30.

James Thorne and Andreas Vlachos. 2017. An Extensible Framework for Verification of Numerical Claims. In *European Chapter of the Association for Computational Linguistics (EACL)*, pages 37–40.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and verification. In *NAACL-HLT*.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 18–22, Baltimore, MD, USA, June. Association for Computational Linguistics.

Andreas Vlachos and Sebastian Riedel. 2015. Identification and Verification of Simple Claims about Statistical Properties. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, (September):2596–2601.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science*, 359(6380):1146–1151.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2017. Evidence Aggregation for Answer Re-Ranking in Open-Domain Question Answering. 1:1–13.

William Yang Wang. 2017. "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 422–426.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing - HLT '05*, (October):347–354.

W. A. Woods. 1973. Progress in natural language understanding: An application to lunar geology. In *Proceedings of the June 4-8, 1973, National Computer Conference and Exposition*, AFIPS '73, pages 441–450, New York, NY, USA. ACM.

Lina Zhou, Judee K. Burgoon, Jay F. Nunamaker, and Doug Twitchell. 2004. Automating Linguistics-Based Cues for detecting deception in text-based asynchronous computer-mediated communication. *Group Decision and Negotiation*, 13(1):81–106.

Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. 2018. Detection and resolution of rumours in social media: A survey. *ACM Comput. Surv.*, 51(2):32:1–32:36, February.

Can Rumour Stance Alone Predict Veracity?

Sebastian Dungs[♪], **Ahmet Aker**[♪♩], **Norbert Fuhr**[♪] **and Kalina Bontcheva**[♩]
[♪] University of Duisburg-Essen, Lotharstraße 65, 47057 Duisburg, Germany
[♩] University of Sheffield, 211 Portobello, Sheffield S1 4DP, UK
firstname.lastname@uni-due.de
k.bontcheva@dcs.shef.ac.uk

Abstract

Prior manual studies of rumours suggested that crowd stance can give insights into the actual rumour veracity. Even though numerous studies of automatic veracity classification of social media rumours have been carried out, none explored the effectiveness of leveraging crowd stance to determine veracity. We use stance as an additional feature to those commonly used in earlier studies. We also model the veracity of a rumour using variants of Hidden Markov Models (HMM) and the collective stance information. This paper demonstrates that HMMs that use stance and tweets' times as the only features for modelling true and false rumours achieve F_1 scores in the range of 80%, outperforming those approaches where stance is used jointly with content and user based features.

1 Introduction

Social media are rife with rumours, which are fast-spreading, unverified pieces of information (Zubiaga et al., 2018). With fake news and misinformation now widely recognised as a major problem for journalists, media, online platforms, and citizens, automatic rumour detection and analysis has become a hot research topic too. Rumour analysis research has focused on Twitter in particular, as it has established itself as the go-to social platform for real-time news (Hu et al., 2012). Twitter's unmoderated nature is also the perfect ground for spreading rumours (Qazvinian et al., 2011). A key focus of prior work on rumour analysis has been *rumour stance classification*, where the stance of each tweet on a given rumour is classified as supporting, denying, questioning or commenting on the rumour (Procter et al., 2013).

Our work builds on the hypothesis that, as rumours evolve over time, so does the stance expressed by the public towards those rumours. In the early stages of a rumour, its actual veracity tends to be unknown. However, as new evidence emerges over time, Twitter users take more pronounced and continuously evolving stance towards the information asserted in the rumour. For instance, in the early stages of a rumour supporting tweets might prevail, simply due to the lack of information to the contrary. However, when authoritative sources or reliable evidence emerge either for or against the rumour, a similar trend tends to be observed in the collective rumour stances. This was first noted in a manual analysis by Mendoza et al. (2010), who found that true rumours tended to be affirmed more than 90% of the time, whereas false rumours were challenged (questioned or denied) 50% of the time. This encourages the use of crowd (or collective) stance as a feature in an automatic rumour veracity classifier.

A rumour consists of a source tweet and several other tweets that responded to the source one—the source tweet contains the rumour. Each of those responding tweets is associated with a particular stance (supporting, denying, questioning or commenting). In this work we make use of the stances of those responding tweets to judge the veracity of the rumour. We refer to the stances of those tweets as crowd or collective stance.

Hidden Markov Models (HMM) are well known for their application in temporal pattern recognition. By regarding the individual stances over a rumour's lifetime as an ordered sequence of observations, we

Proceedings of the 27th International Conference on Computational Linguistics, pages 3360–3370
Santa Fe, New Mexico, USA, August 20-26, 2018.

can then model the actual veracity of a rumour by the hidden part of the HMM. The general assumption is that true and false rumours would have different patterns in the stance distribution over time. Therefore, after training models for *true* and *false* rumours, we can build a binary veracity classifier by comparing sequence occurrence probabilities for the two cases.

This paper investigates whether and to what extent rumour veracity classification can be predicted on the basis of crowd stance. While there is a body of work on automatic veracity classification such as (Castillo et al., 2011; Kwon et al., 2013; Vosoughi, 2015; Wu et al., 2015; Ma et al., 2015; Lukasik et al., 2016), only few have applied stance information as a feature (Liu et al., 2015; Enayet and El-Beltagy, 2017), and none of these studies investigated the collective power of the crowd as a source of stance information, which can then be used as a feature in rumour veracity classification. Here we argue that the content of the posts is not necessarily helpful towards determining the veracity of a story, as the content can be either misleading or inaccurate. We believe it is the aggregation of stances which can provide useful information to determine the veracity. Despite the fact that some of the users will be inevitably mistaken, and sharing the wrong stance, we argue that the aggregation of stances will correct itself towards being a useful feature for veracity classification. The novel contribution of this paper is in demonstrating that collective stance together with the tweets' time as features can indeed boost performance results. We demonstrate that HMM using only stance and time features achieve an F_1 score of around 80%, rendering significantly better results than traditional rich feature engineered approaches that entails stance as additional feature. We also show that our HMM are indeed applicable in case of early rumour detection.

2 Related Work

The veracity classification task aims to determine whether a given rumour can be confirmed as true, debunked as false, or in some cases its truth value is yet to be resolved (i.e. unknown). Related work has tackled the problem in a supervised fashion by applying state-of-the-art machine learning algorithms on features extracted from rumour datasets. One of the key differences between the approaches is the set of features proposed to tackle the problem. The pioneering paper of Castillo et al. (2011) proposed message, user, topic and propagation-based features. In most subsequent studies these features have been used as baselines. Following these feature sets, Kwon et al. (2013) and Kwon et al. (2017) proposed a new set of feature categories: temporal, structural and linguistic and showed their importance in fighting with rumours. Other than Twitter, the Chinese microblogging platform Sina Weibo has been also analysed for rumours. For this Yang et al. (2012) proposed client and location-based features and showed that these help to increase prediction accuracy.

Liu et al. (2015) used the approaches reported by Yang et al. (2012) and Castillo et al. (2011) as baseline systems and compared them against a new approach based on verification features, which include stance of individual tweets, rather than collective stance. Ma et al. (2015) adapted features from earlier studies and proposed to model them over time. Wu et al. (2015) extracted features from message propagation trees. Three categories of features were considered: message, user and report-based. The idea of message propagation was also investigated by Wang and Terano (2015). Vosoughi (2015) tackled veracity classification using three categories of features (linguistic, user oriented and temporal propagation related) and speech recognition inspired machine learning approaches, such as Dynamic Time Wrapping and Hidden Markov Models.

Chen et al. (2016) treated rumour veracity classification as an anomaly detection problem where false rumours are regarded as anomalies. Several features related to the content, crowd opinion and post propagation were used. Chang et al. (2016) put the emphasis on the characteristics of users who post the rumours to determine the veracity. Unlike previous studies, Tong et al. (2017) aimed at blocking rumours rather than detecting them or marking tweets as true or false. Motivated by the fact that later corrections are not as effective, the authors argued that the first post seen by a user is influential for their future opinion and thus it is important to show users rumours only once they are confirmed to be true. Based on this they proposed a reverse-tuple based randomised algorithm to block rumours. The algorithm aimed at producing positive seeds to be shown to users first.

Rumour veracity classification has also been studied in the RumourEval shared task at SemEval 2017 (Derczynski et al., 2017a). Subtask B consisted in determining if each of the rumours in the dataset were true, false or remained unverified. It considered two different settings, one *closed* where participants could not make use of external knowledge bases and another *open* where use of external resources was allowed. Participants viewed the task either as a three-way (Enayet and El-Beltagy, 2017; Wang et al., 2017; Singh et al., 2017) or two-way (Chen et al., 2017; Srivastava et al., 2017), single tweet classification task. The winning system (Enayet and El-Beltagy, 2017) added features more specific to the distribution of stance labels in the tweets replying to the source tweet (percentage of reply tweets classified as either support, deny or query). The survey paper of Zubiaga et al. (2018) provides an extensive summary of current work on rumour verification but also related task such as detection of rumours as well as stance classification of messages involved in rumours.

In this work we propose to use stance only to tackle the rumour verification task. As highlighted earlier, we aim to capture collective stance information for veracity classification. First we investigate the impact of using sequences of individual stances only. In our second approach, we integrate stance and time information into a unified model. Both approaches are based on modelling the state-changes of stances using HMM. Thus the most related studies to ours are Ma et al. (2015) because they model features over time, Vosoughi (2015) due to the use of HMM as well as Chang et al. (2016), Liu et al. (2015) and Enayet and El-Beltagy (2017) because they add the reactions (stances) of the users as additional feature. However, we differ from those because we rely only on the power of collective stance, model this using HMM and apply this model to predict the veracity of rumours.

3 Dataset

The rumour dataset used in this paper (Zubiaga et al., 2016) is the only rumour stance and veracity classified tweet dataset which is publicly available. The authors identified rumours associated with major news events as well as tweets associated with those rumours and then annotated each of the tweets for stance and the rumours for their overall veracity. The dataset consists of rumours that emerged during eight different events. However, three of these events evoked less than five rumours consisting of five or more tweets. Therefore, we limit our experiments to the events detailed in Table 1.

Event	Rumours	True / False
Charlie Hebdo	46	24 / 22
Ferguson Riots	34	2 / 32
Germanwings Crash	12	2 / 10
Ottawa Shooting	31	20 / 11
Sydney Siege	50	33 / 17
Total	173	81 / 92

Table 1: Rumours in five events. Each rumour has at least 10 tweets.

Each of the remaining events consists of several rumours and each rumour of several tweet threads. In the data set each tweet is identified by its unique ID, is associated with exactly one event and a unique rumour ID, has a time stamp and is assigned a stance label. Possible stances σ are *supporting, denying, questioning* and *commenting*. Rumours are marked with their veracity values (*true* or *false*).

4 Method: HMM for Veracity Classification

A Hidden Markov Model is a stochastic model in which the system is presumed to be a Markov process including unobservable hidden states. The hidden system undergoes discrete state changes which trigger the emission of observable signals. Based on these signals the hidden states' properties can be learnt.

4.1 Model Generation

More formally, a HMM describes two random processes

$$\{X_t\}_{t\in\mathbb{N}} \text{ and } \{Y_t\}_{t\in\mathbb{N}} \tag{1}$$

of which only the latter is directly observable. A HMM is defined as a 5-tupel

$$\lambda = \{S, E, A, B, \pi\} \tag{2}$$

where $S = \{s_1; \ldots; s_n\}$ is the set of N possible hidden states, $E = \{e_1; \ldots; e_m\}$ is the alphabet of possible observations—i.e. the system's emissions, $A \in \mathbb{R}^{n\times n}$ is the hidden state transition matrix where a_{ij} denotes the probability of the system changing from state i to j, $B \in \mathbb{R}^{n\times m}$ is the emission probability matrix where $b_i(e_j)$ denotes the probability of observing emission $e_j \in \{Y\}$ when the system is in state s_i and finally the starting state probability vector $\pi \in \mathbb{R}^n$, where $p_i = P(X_1 = s_i)$.

For rumour classification we consider tweets related to five events (see Section 3). Consequently, for classification the system λ is defined as follows:

State Space S

Since there is no a-priori solution to determine the optimal size of S (Rabiner, 1990) we consider various hidden state counts $N = \{n \in \mathbb{N} \mid 1 \leq n \leq 15\}$ and repeat calculations accordingly.

Observation Alphabet E

Unlike prior veracity classification approaches, we do not consider the tweets' textual content. Instead, classification is based on stance information alone. Therefore, we set the observation alphabet to

$$E = \{support, deny, question, comment\} \tag{3}$$

Transition Probabilities

Transition matrix A, emission matrix B and start state probability vector π are chosen at random. Baum-Welch parameter optimization algorithm is used with 10 iterators to learn models' final properties. To avoid particularly suboptimal start value configurations and local optima 100 configurations are tested for every $n \in N$. The best performing model is kept while the others are discarded.

4.2 Class Assignment ε

Training data (see Section 5 for details) is used to learn model λ_{false} for false and model λ_{true} for true rumours. For all remaining rumours ε_i (i.e. the testing data) their respective class $C(\varepsilon_i)$ is assigned as follows:

$$C(\varepsilon_i) = \underset{c\in\{false,true\}}{\operatorname{argmax}} P(\varepsilon_i|\lambda_c) \tag{4}$$

In (4) the expression $P(\varepsilon_i|\lambda_c)$ is calculated by using the Forward-Algorithm for Hidden Markov Models.

4.3 Multi Spaced HMM Including Tweet Time

As described above, the data set contains tweets' time stamps which are used to build ordered sequences of tweets beginning with the first reply to a rumourous tweet and ending with the last, while the actual distance in time between tweets is disregarded.

Therefore, a straightforward extension of the classification framework is the inclusion of actual posting times. Since basic HMM implementations do not support combining discrete (stance) and continuous (time) emissions into a unified model, we applied Multi Spaced Hidden Markov Models (MSHMM) for this purpose.

MSHMM were introduced by Tokuda et al. (2002) and originally used in speech synthesis tasks. While HMM use probability matrices in the discrete case or one dimensional distribution functions in

the continuous case, MSHMM use multi spaced observation probability distributions where the sample space Ω contains G spaces:

$$\Omega = \bigcup_{g=1}^{G} \Omega_g \tag{5}$$

Each Ω_g is a n-dimensional real space R^{n_g} with a space index g. Furthermore, each space has a probability w_g, where

$$\sum_{g=1}^{G} w_g = 1 \tag{6}$$

and an observation probability density function

$$N_g(x), x \in \mathbb{R}^1, \text{where } \int N_g(x)dx = 1 \tag{7}$$

Established algorithms for HMM can be adapted to multi spaced observation probability functions including Baum-Welch-, Viterbi- and Forward-Backward-Algorithm.

For classification of tweets including their time stamps a MSHMM λ' is defined. Here each stance σ is assigned its own 1-dimensional real space $\Omega_\sigma = R^{1_\sigma}$, while space weights w_σ are determined by stances' occurrence counts. Space probability density functions are learned based on the timestamps which have been pre-processed to represent only the time elapsed since the start of the rumour until a replying tweet occurred.

Accordingly, observation alphabet of MSHMM λ' is defined as a random vector $o = (X, x)$, where X is a space index (i.e. a stance) and $x \in \mathbb{R}^1$ the processed timestamp.

Multi spaced observation probability of o is defined as

$$b(o) = \sum_{g \in X} w_g N_g(x) \tag{8}$$

Additionally, state set S, state transition matrix A and starting state probability vector π of system λ' are defined analogue to system λ.

5 Evaluation Settings

To evaluate the performance of the classification framework leave-one-event-out cross validation was performed, following the methodology of Lukasik et al. (2015). This means that we train on $n-1$ events (on all rumours within these events) and test it on an unseen n^{th} event (on all rumours within this event). We use F_1 score to measure classifier performance and compare our results against two baselines. The first one does not make use of stance, whereas the second one integrates stance as one of a set of features. These two baselines have been selected to simulate the impact of collective stance i.e., that collective stance have positive impact on rich traditional feature engineered approaches. However, our results (see Section 6) show that it is important how this collective stance information is used to judge the veracity of rumours. We describe these baselines in more details next.

5.1 Stance Unaware Baseline: B1

As described in Section 2, prior work on tweet veracity classification investigated a wide range of features and classification methods. Most influential is the feature set proposed by Castillo et al. (2011), which we have adopted for training a classification model. We experimented with various classifiers such as simple decision trees, k-nn, etc. but achieved best performance with Random Forests, more precisely, we use a range of 33 different features varying from syntactical, semantic, indicator, user-specific and message-specific categories. Details of our features can be obtained from (Aker et al., 2017b).

5.2 Stance Aware Baseline: B2

Following the approach of Liu et al. (2015), we integrate stance as several features, additional to those used in the first baseline. Since there are four different stance classes, the following additional feature(s) are used:

Relative-Stance-Score

Percentage of supporting, denying, questioning, and commenting stances extracted from tweets within a rumour. To obtain this feature we count e.g. how many individual tweets express support for the rumour and divide it by the total number of tweets. We have in total four features, one of each class. Similar to above, we use Random Forest as the machine learning algorithm.

6 Results

Table 2 details a performance comparison for both our systems, as well as B1 and B2. For completeness we also include a simplistic majority-vote baseline on event level. Using system λ we achieved an F_1 score of 0.756 across the 5 events including 173 rumours. The multi spaced system λ' shows superior performance with an F_1 score of 0.804. Both our systems significantly outperform both baselines' F_1 scores of 0.553 and 0.557 respectively.

System	Precision	Recall	F_1
B1	0.650	0.481	0.553
B2	0.661	0.481	0.557
λ	**0.747**	0.765	0.756^*
λ'	0.690	**0.963**	$\mathbf{0.804}^*$
majority-vote	0.059	0.025	0.035

Table 2: Overall classification scores.

* indicates significant difference to B1 and B2 (Tukey's HSD $p < 0.05$)

Breaking down the results into precision and recall, we can observe that system λ achieves the highest precision score of 0.747, outperforming both baselines (0.650 and 0.661) as well as system λ'. The latter also gives results slightly more precise than the baselines (0.690). Regarding recall system λ' outperforms all other systems by a great margin achieving a score of 0.963. System λ has also much higher recall of 0.765 comparing to the baseline score of 0.481 for B1 and B2.

Additionally, we look at the individual results for each event (Table 3). It can be seen that the performance varies substantially across events and classifiers. This becomes most obvious in case of the Ferguson event, where B1 and B2 fail to deliver a single true positive result—hence the F_1 score of 0. In other events such as Ottawa Shooting and Sydney Siege all systems have acceptable precision in their results. However, only our proposed (MS-)HMM also have a high to very high recall leading to a vastly superior F_1 score in these events. Another interesting observation can be made regarding the Germanwings Crash event where system λ yields perfect classification while B2 achieves exactly the same results as the overall best performing system λ'. However, it also has to be noted that this particular event contains only 12 rumours and these results therefore have limited expressiveness.

Finally, we also examined system λ''s noticeable low F_1 score of 0.4 in the Ferguson event. Looking at the classification outcome on rumour level we observe that this score is largely caused by the unbalanced nature of the event with only 2 of the 34 rumours being actually true. Out of these true rumours the system managed to capture one (λ: 2; B1: 0; B2: 0) while overall misclassifying only three rumours (λ: 1; B1: 4; B2: 5). In fact, the performance of all classifiers concerning this event is much more similar than the particular F_1 scores suggest.

6.1 Early detection of rumours

For an eventual practical application of our rumour classification system it is also important to consider its performance as a function of time, i.e. how many tweets are necessary to achieve a reasonable classification performance. Therefore, we also explore early detection of rumours by confining our systems to the first ten (first five) tweets only during classification. Classification results using shortened

Charlie Hebdo			
System	Precision	Recall	F_1
B1	0.634	0.792	0.704
B2	**0.667**	0.667	0.667
λ	0.643	0.750	0.692
λ'	0.605	**0.958**	**0.742**
Ferguson			
System	Precision	Recall	F_1
B1	0	0	0
B2	0	0	0
λ	**0.667**	**1**	**0.800**
λ'	0.333	0.500	0.400
Germanwings			
System	Precision	Recall	F_1
B1	0.333	0.500	0.400
B2	0.500	**1**	0.667
λ	**1**	**1**	**1**
λ'	0.500	**1**	0.667
Ottawa			
System	Precision	Recall	F_1
B1	0.818	0.450	0.581
B2	**0.909**	0.500	0.645
λ	0.882	0.750	0.811
λ'	0.792	**0.950**	**0.864**
Sydney Siege			
System	Precision	Recall	F_1
B1	0.714	0.303	0.426
B2	0.647	0.333	0.440
λ	**0.758**	0.758	0.758
λ'	0.750	**1**	**0.857**

Table 3: Performance across events

sequences are summarized in Table 4. As expected the best scores can be achieved by using complete sequences, which feature a median tweet count of 18. For sequences shortened to the first 10 tweets performance drops down to an F_1 score of 0.658 for system λ and 0.642 for system λ' respectively. Further reducing sequences' length to five tweets leads to worse classification performance. However, performance decrease is considerably lower for system λ' with an F_1 score of 0.618 (λ: 0.524).

Additionally, we also retrained our best performing model λ' on shortened sequences using first 10 and first 5 tweets only. Comparing F_1 scores between training conditions we only find marginal decrease of roughly 1% when using the first 10 tweets. However, narrowing down to the first 5 tweets during model training gives unsatisfactory classification results with an F_1 score 0.529.

Overall it can be seen that given only the first 10 tweets at classification time our models still perform better than the baselines. Using even shorter sequences is only reasonable when utilizing the full potential of MSHMM which were still able to beat the baselines while using only the first 5 tweets.

System	All tweets	First 10	First 5
λ	0.756	**0.658**	0.524
λ'	**0.804**	0.642	**0.618**

Table 4: Early detection F_1 scores

6.2 Using Automatic Stance Labels

Results in Table 2 are obtained using gold stance labels. However, this restricts the idea of stance-based rumour verification to only data where human stance labels are available. To overcome this limitation, we have adopted the state-of-the-art stance classifier of Aker et al. (2017a). It has been evaluated on the RumourEval'2017 shared task A on rumour stance classification (Derczynski et al., 2017b) and achieved the best results, namely 79.02% accuracy. The classifier's performance measure was obtained by the RumourEval organizers. Note that for evaluating our models a subset of the data from shared task A has been used. The system performs standard feature engineering such as the extraction of bag-of-words, sentiments, indicator features, etc, but also adds features that are specific to modelling the stance classification problem such as whether the tweet entails some surprise, doubt, certainty and support terms. Apart from feature extraction the system does not require any further parameters to set nor any domain knowledge.

We label automatically all tweets in our dataset with tweet-level stance information using this classifier. Once these stance labels are obtained, we repeat the evaluation of (MS-)HMMs λ and λ'. Results for systems λ_a and λ'_a with automatic stance labels are shown in Table 5.

Both variants show only slight changes in F_1 score compared to their gold data counterparts. In terms of precision and recall λ_a shows a shift towards recall compared to λ—combined with a corresponding loss in precision. However, system λ'_a remains stable in precision and recall while using automatically generated labels. This demonstrates that our veracity classification approach based on the features stance and time has viable practical applications.

System	Precision	Recall	F_1
λ_a	0.632	0.888	0.738
λ'_a	**0.669**	**0.975**	**0.794**

Table 5: Overall scores using automatic labels

7 Discussion

In our results in Table 2 we showed that overall collective stance indeed is an important feature to consider for the purpose of veracity prediction. However, this depends on how this collective feature is used. When stance is added as additional feature to those features reported by related work (setting B2) we could only gain a neglectable improvement. On the other side, when collective stance was used within HMM we observe superior results indicating that using HMM is a better way for capturing the crowd stance wisdom and applying this effectively on the veracity classification task. In case of baseline B2 we used rather a crude way of capturing the stance wisdom by counting different stance types. However, collective stance might obey some specific patterns of development, as indicated by Mendoza et al. (2010), and capturing these patterns is an important factor. This is where our systems λ and λ' shine and the power of capturing this is reflected in the ample classification performance increase. However, this increase is not distributed equally across all events. In the following paragraphs we are going to raise a few points that likely have contributed to these outcomes:

Our MSHMM λ' overall produces results which are especially distinct from the other models' results by being able to correctly classify 12 rumours where all other models fail. Note that this occurs in only two and one cases for baselines B1 and B2 respectively. Further investigation of these 12 correctly

classified rumours shows that they tend to be shorter than average with a median length of 11 tweets. Interestingly, λ' correctly classifies true rumours in 9 out of the 12 cases, although they contain only 1 to 3 *supporting* stances each.

On the other hand, there are also 15 rumours that are solely misclassified by our HMM λ'. Again, this is less often the case for the baselines with 8 occurrences each for model B1 and B2. As it is also indicated by the comparably lower precision scores, model λ's produces more false positive classifications than the other models, especially concerning the events *Sydney siege* and *Ferguson* (B1: 15; B2: 14; λ: 18; λ': 26). Consequently, all 15 misclassified rumours are false positives. Further investigation of the individual rumours' properties remained inconclusive.

Overall the baselines have a negative bias, i.e. they tend to classify rumours as false. While the actual true/false ratio of all rumours is 46.8% true, model B1 classifies 35% of the rumours as true (B2 34%). On the contrary, λ shows no bias with a ratio of 48%, while λ' has a positive bias with a ratio of 64.3%. This observation is in line with the baselines F_1 score of 0 for rumours related to the *Ferguson riots* event where only 2 out of all 34 rumours are true. Out of these two true rumours one was found by both our models while the other was correctly classified solely by model λ.

We investigated our models' performance on short sequences of tweets to accommodate for the fact that a timely detection of rumours would be of great practical benefit. Naturally, we experience a performance drop when reducing sequences' length. We believe that this is due to the collective stance being expected to stabilize only over time (see Section 1). However, we indeed could show that especially our MSHMM is capable of capturing a rumours' veracity very fast with a high level of recall and sufficient precision, still outperforming the baselines even when using only the first five tweets of a rumour.

Especially the high recall affirms our intend to adapt our framework for the task of rumour detection, which is a necessary step before rumour veracity classification can be performed. It will be interesting to observe in future work whether the model can perform equally well at this task.

As described in Section 6 our model λ' including tweets' times achieved best performance values on this particular data set. This result—as well as the strong result of model λ—shows our HMM-based method's general applicability to the problem at hand. Furthermore, we have demonstrated the general suitability and classification stability of our automated stance annotation framework. Since we seem to have overcome the need for manual annotation when using MSHMM in future work the data sets can be extended to more recent events featuring potentially large amounts of tweets.

However, it is also reasonable to assume that events are heterogeneous in their stance distribution patterns, which might have an impact on classification performance and generalizability across events. Therefore, in subsequent analysis different event types should be considered when training MSHMM. A basic event distinction might be *sharp*, sudden events that also feature a definite ending vs. *soft* events where multiple sub-events occur that trigger new developments in the corresponding discussion threads.

Finally, note that the discussed HMM approach achieves results in the 80% margin in terms of F_1 when it uses gold standard stance. This performance drops insignificantly when the stance information is obtained by automatic methods. With that the stance information is the only dependent variable for the HMM and any performance improvement on that site will, as the results on gold standard stance show, also increase the performance of the HMM-based classifier. This is also valid for new rumours where it can be expected that the current automatic stance detection approach does not perform as well as in the SemEval2017 data and thus have negative impact on the verification results. This is a known problem that there is a performance drop when the system moves to new unseen data. However, the performance gap can be closed with extending training data (obtained either manually or using some distance learning) and focusing more on domain independent features.

8 Conclusions

This paper investigated whether rumour stance alone and combined with tweets' times can be used to predict rumour veracity automatically. We followed two different strategies in applying stance for the verification task. In the first strategy we added stance as an additional feature to those commonly used in earlier studies. We demonstrated that including collective stances observed in sequences of tweets

leads to only neglectable performance increase when using approaches adapted from the literature. The second strategy was to use stance as the only feature to model true and false rumours with discrete HMM. Finally, we employed multi-spaced HMM to jointly model the temporal changes in stance information. We demonstrated that already using stance-based HMM without time information leads to substantially better classification results over the first strategy. Though, the extension to multi-spaced HMM with time incorporated has led to even superior results with an F_1 score of 80.4%.

Next, we will investigate further the integration of temporal information for the rumour detection task. Furthermore, we will explore if veracity classification can be improved further by combining our HMM-based classifier with other state-of-the-art approaches. In addition, we plan to adapt the crowd stance along with HMM for the fake news detection task.

Acknowledgements

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under grant No. GRK 2167, Research Training Group "User-Centred Social Media", the European Union funded COMRADES project (grant agreement No. 687847), and an EPSRC career acceleration fellowship (EP/I004327/1). Last but not least we thank all anonymous reviewers for their valuable feedback.

References

Ahmet Aker, Leon Derczynski, and Kalina Bontcheva. 2017a. Simple open stance classification for rumour analysis. *CoRR*, abs/1708.05286.

Ahmet Aker, Arkaitz Zubiaga, Kalina Bontcheva, Anna Kolliakou, Rob Procter, and Maria Liakata. 2017b. Stance classification in out-of-domain rumours: A case study around mental health disorders. In *International Conference on Social Informatics*, pages 53–64. Springer.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684. ACM.

Cheng Chang, Yihong Zhang, Claudia Szabo, and Quan Z Sheng. 2016. Extreme user and political rumor detection on twitter. In *Advanced Data Mining and Applications: 12th International Conference, ADMA 2016, Gold Coast, QLD, Australia, December 12-15, 2016, Proceedings*, pages 751–763. Springer.

Weiling Chen, Chai Kiat Yeo, Chiew Tong Lau, and Bu Sung Lee. 2016. Behavior deviation: An anomaly detection view of rumor preemption. In *Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2016 IEEE 7th Annual*, pages 1–7. IEEE.

Yi-Chin Chen, Zhao-Yand Liu, and Hung-Yu Kao. 2017. IKM at SemEval-2017 Task 8: Convolutional Neural Networks for Stance Detection and Rumor Verification. In *Proceedings of SemEval*. ACL.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017a. SemEval-2017 Task 8: RumourEval: Determining rumour veracity and support for rumours. In *Proceedings of SemEval*. ACL.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017b. SemEval-2017 Task 8: RumourEval: Determining rumour veracity and support for rumours. In *Proceedings of SemEval*. ACL.

Omar Enayet and Samhaa R. El-Beltagy. 2017. NileTMRG at SemEval-2017 Task 8: Determining Rumour and Veracity Support for Rumours on Twitter. In *Proceedings of SemEval*. ACL.

Mengdie Hu, Shixia Liu, Furu Wei, Yingcai Wu, John Stasko, and Kwan-Liu Ma. 2012. Breaking news on twitter. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2751–2754. ACM.

Sejeong Kwon, Meeyoung Cha, Kyomin Jung, Wei Chen, and Yajun Wang. 2013. Prominent features of rumor propagation in online social media. In *2013 IEEE 13th International Conference on Data Mining*, pages 1103–1108. IEEE.

Sejeong Kwon, Meeyoung Cha, and Kyomin Jung. 2017. Rumor detection over varying time windows. *PLOS ONE*, 12(1):e0168344.

Xiaomo Liu, Armineh Nourbakhsh, Quanzhi Li, Rui Fang, and Sameena Shah. 2015. Real-time rumor debunking on twitter. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1867–1870. ACM.

Michal Lukasik, Trevor Cohn, and Kalina Bontcheva. 2015. Classifying tweet level judgements of rumours in social media. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pages 2590–2595.

Michal Lukasik, PK Srijith, Duy Vu, Kalina Bontcheva, Arkaitz Zubiaga, and Trevor Cohn. 2016. Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter. In *Proceedings of 54th Annual Meeting of the Association for Computational Linguistics*, pages 393–398. Association for Computational Linguistics.

Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. 2015. Detect rumors using time series of social context information on microblogging websites. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1751–1754. ACM.

Marcelo Mendoza, Barbara Poblete, and Carlos Castillo. 2010. Twitter under crisis: can we trust what we rt? In *Proceedings of the first workshop on social media analytics*, pages 71–79. ACM.

Rob Procter, Jeremy Crump, Susanne Karstedt, Alex Voss, and Marta Cantijoch. 2013. Reading the riots: What were the police doing on twitter? *Policing and society*, 23(4):413–436.

Vahed Qazvinian, Emily Rosengren, Dragomir R. Radev, and Qiaozhu Mei. 2011. Rumor has it: Identifying misinformation in microblogs. In *Proceedings of EMNLP*, pages 1589–1599.

Lawrence R. Rabiner. 1990. Readings in speech recognition. chapter A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Vikram Singh, Sunny Narayan, Md Shad Akhtar, Asif Ekbal, and Pushpak Bhattacharya. 2017. IITP at SemEval-2017 Task 8: A Supervised Approach for Rumour Evaluation. In *Proceedings of SemEval*.

Ankit Srivastava, Rehm Rehm, and Julian Moreno Schneider. 2017. DFKI-DKT at SemEval-2017 Task 8: Rumour Detection and Classification using Cascading Heuristics. In *Proceedings of SemEval*, pages 486–490. ACL.

Keiichi Tokuda, Takashi Masuko, Noboru Miyazaki, and Takao Kobayashi. 2002. Multi-space probability distribution HMM. *IEICE TRANSACTIONS on Information and Systems*, 85(3):455–464.

Guangmo Tong, Weili Wu, Ling Guo, Deying Li, Cong Liu, Bin Liu, and Ding-Zhu Du. 2017. An efficient randomized algorithm for rumor blocking in online social networks. *arXiv:1701.02368*.

Soroush Vosoughi. 2015. *Automatic detection and verification of rumors on Twitter*. Ph.D. thesis.

Shihan Wang and Takao Terano. 2015. Detecting rumor patterns in streaming social media. In *Big Data (Big Data), 2015 IEEE International Conference on*, pages 2709–2715. IEEE.

Feixiang Wang, Man Lan, and Yuanbin Wu. 2017. ECNU at SemEval-2017 Task 8: Rumour Evaluation Using Effective Features and Supervised Ensemble Models. In *Proceedings of SemEval*, pages 491–496. ACL.

Ke Wu, Song Yang, and Kenny Q Zhu. 2015. False rumors detection on sina weibo by propagation structures. In *2015 IEEE 31st International Conference on Data Engineering*, pages 651–662. IEEE.

Fan Yang, Yang Liu, Xiaohui Yu, and Min Yang. 2012. Automatic detection of rumor on sina weibo. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, page 13. ACM.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PLoS ONE*, 11(3):1–29, 03.

Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. 2018. Detection and resolution of rumours in social media: A survey. *ACM Computing Surveys (CSUR)*, 51(2):32.

Attending Sentences to detect Satirical Fake News

Sohan De Sarkar
Dept. of Computer Science
Indian Institute of Technology
Kharagpur, West Bengal, India
sohandesarkar@gmail.com

Fan Yang
Dept. of Computer Science
University of Houston
3551 Cullen Blvd., Houston
fyang11@uh.edu

Arjun Mukherjee
Dept. of Computer Science
University of Houston
3551 Cullen Blvd., Houston
arjun4787@gmail.com

Abstract

Satirical news detection is important in order to prevent the spread of misinformation over the Internet. Existing approaches to capture news satire use machine learning models such as SVM and hierarchical neural networks along with hand-engineered features, but do not explore sentence and document difference. This paper proposes a robust, hierarchical deep neural network approach for satire detection, which is capable of capturing satire both at the sentence level and at the document level. The architecture incorporates pluggable generic neural networks like CNN, GRU, and LSTM. Experimental results on real world news satire dataset show substantial performance gains demonstrating the effectiveness of our proposed approach. An inspection of the learned models reveals the existence of key sentences that control the presence of satire in news.

1 Introduction

In the era of the Internet, online journalism is now a common practice. Online news articles have a major contribution in keeping people informed about what is happening in the world. The usage of Internet to spread news comes with the disadvantage of deception. The presence of deceptive and misleading news articles has been around for a while. Although some news articles often have a disclaimer about it being fake, many other don't and thus readers could be led to believe them to be true. This leads to spread of misinformation, which may also start off a rumour. The importance of the detection of deceptive news is increasing rapidly, as more and more people start relying on online news as their major source of news.

News satire is a genre of deceptive news that is found on the web, with the intent of dispensing satire in the form of legitimate news articles. These articles differ from "fake" news, in the sense that fake news intend to mislead people by providing untrue facts, while satirical news intends to ridicule and criticize something by providing satirical comments or through fictionalized stories. Satire is the intention of the author to be discovered as "fake", unlike fake news, in which the intention is to make make the readers believe in the news as true. Detection of news satire is thus important to control the spread of false stories.

We propose a hierarchical deep neural network model for satirical news detection, that is able to capture satire both at the sentence level and at the document level. The architecture is very extensible and caters to a variety of plug-and-play neural network models such as CNN, LSTM and GRU. This pipelined architecture allows for optimal learning of parameters required to capture satire. We show that our model is able to capture satire more efficiently than existing models, by using only pretrained word embeddings as input, without the aid of any syntactic information or any hand-crafted features. We show that word level semantic information is sufficient for effective detection of satire, with word level syntax information only marginally improving the performance. An analysis of the learned models reveals that news satire is decided by a few key sentences of the news article, the last sentence being one of them.

We use the dataset introduced in (Yang et al., 2017) as the dataset for satire news detection. We trained our proposed plug-and-play hierarchical model end-to-end on the ground truth data. Our model works at the sentence level as opposed to paragraph level attention in (Yang et al., 2017). We perform extensive

Proceedings of the 27th International Conference on Computational Linguistics, pages 3371–3380
Santa Fe, New Mexico, USA, August 20-26, 2018.

experiments on the dataset, fine-tuning the model by plugging different neural network models into the architecture. Experimental results on the dataset shows superior performance of our model compared to existing state-of-the-art approaches.

2 Related Work

Previous approaches for generic deception detection include the use of traditional machine learning model such as SVM (Zhang et al., 2012) and Naive Bayesian models (Oraby et al., 2015). These approaches focus on using linguistic cues and the social network behavior (Conroy et al., 2015) to detect deception. Much work has been done for deception detection on social media platforms (Davidov et al., 2010; Reyes et al., 2012) and opinion spam (Ott et al., 2011; Mukherjee et al., 2012; Mukherjee et al., 2013) In the context of deceptions in news, the field of "fake" news detection has been explored before (Jin et al., 2016; Rubin et al., 2016). These also include the use of machine learning, some of them also leveraging neural networks (Wang, 2017; Ruchansky et al., 2017) for the task.

Existing works towards satirical news detection focus on engineering features to denote satire. (Burfoot and Baldwin, 2009) filter satirical news from true news with headline features, profanity, and slang. (Rubin et al., 2016) propose additional features to classify satirical news, including absurdity, humour, grammar, negative affect, and punctuation. (Yang et al., 2017) further show linguistic features could be incorporated at paragraph level and reveal the different behaviour of each feature at paragraph level and document level. These models heavily rely on linguistic/word features as opposed to our representation learning approach. From these works, we observe that word level features contribute to the detection most while linguistic features only improve the result by a little, so we focus on our model to detect satire without further hand-crafted features.

While features generated with careful hand analysis might contribute a robust classifier, neural network based models, from convolutional neural network (Kim, 2014; Kalchbrenner et al., 2014) to recurrent neural network (Tang et al., 2015), or a hybrid of the two (Lai et al., 2015), have pushed classification task to a new level. Also, the recent advances in learning distributed representations for word semantics in the form of word embeddings (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2016) allow for better modeling of semantics both at the sentence and document level. In this work, we utilize the power of neural networks and aim to advance the result of satirical news detection. We pack two separate composition models to further enhance the performance of the learned representation.

3 Model

We propose an approach for building a robust hierarchical neural network architecture for detecting satire news, as shown in Figure 1. We abstract the whole network into two major components, the S and D module. The compositional module S creates a sentence embedding, taking a sequence of word embeddings as inputs. The compositional module D creates a document embedding, which acts as a summarization of the document, taking sentence embeddings as input. We use the learned document embeddings to classify the news as satire or true. This kind of abstraction helped us to fine-tune the architecture by applying different choices of compositional models for the S and D module.

3.1 Word embeddings and Syntax

We use different pretrained word embeddings such as Glove[1] (Pennington et al., 2014) and fastText[2] (Bojanowski et al., 2016) as the initial word embeddings. These pretrained embeddings are (optionally) concatenated with one-hot embeddings that contain the syntax information[3] of the word (Baccianella et al., 2010; Miller, 1995). The various syntactic features used and their corresponding one-hot vector lengths is shown in Table 1. The named entities used are: FACILITY, GPE, GSP, LOCATION, ORGANIZATION, PERSON, NULL(representing no named entity). The SentiWordnet scores are 16 discrete values ranging between 0 and 1, thus requiring a one-hot vector of size 16 to represent each score. These word

[1] https://nlp.stanford.edu/projects/glove

[2] https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

[3] http://www.nltk.org

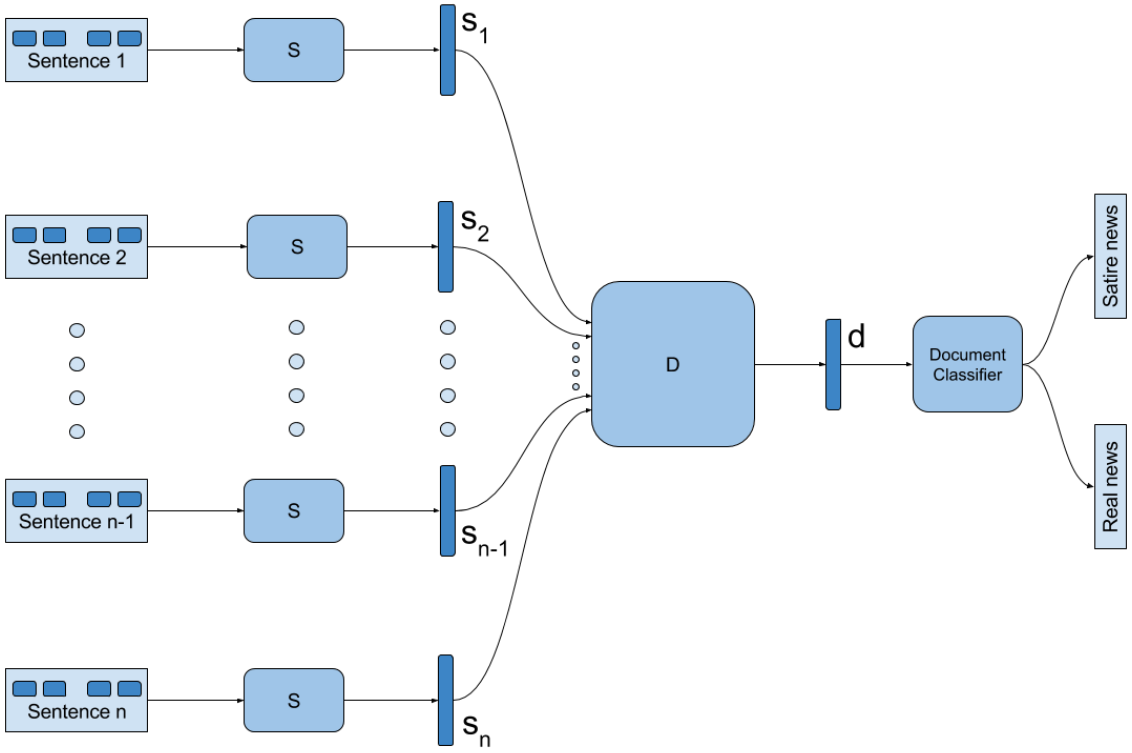


Figure 1: Model Architecture

embeddings (concatenated with syntax information) are multiplied with a weight matrix W_{emb} (learned) to produce a final word embedding, that summarizes the required semantics of the word for capturing satire.

3.2 Sentence (S) Module

The S module takes a sequence of word embeddings as input, and produces a sentence embedding. This module tries to capture the essential information for capturing satire in a news at the sentence level. The various model choices for the S module include Temporal Convolutional neural networks(Kim, 2014) (CNN) and sequential models like Long Short-Term Memory(Hochreiter and Schmidhuber, 1997) (LSTM) and Gated Recurrent Unit(Cho et al., 2014) (GRU).

Let $v_i \in \mathbb{R}^d$ be the d-dimensional word embedding of the i^{th} word of a sentence of length n. We show 3 different models to produce a sentence embedding from the word embeddings. Then, the S module can be represented mathematically as a composition function f that takes a sequences of n word embeddings as input to produce a sentence embedding s. Thus,

$$s = f([v_1, v_2, \dots v_n]) \tag{1}$$

where the choice of the composition function are standard generic neural networks like $LSTM, GRU, CNN$. In the case of LSTM/GRU, we use their deep bidirectional versions, where we stack multiple bidirectional LSTM/GRU on top of each other.

3.3 Document (D) Module

Similar to the S module, D module takes a sequence of sentence embeddings as input and produces a document embedding, capturing information at the document level. The embedding produced by this

Syntax feature	Length
Part-Of-Speech tag	44
SentiWordnet positive score	16
SentiWordnet negative score	16
SentiWordnet objective score	16
Named Entity IOB tag	3
Named Entity tag	7
Topmost Wordnet synset same as word synset	2
Starts with uppercase letter	2
All uppercase letters	2
Is number	2

Table 1: Word level syntax features and their one-hot vector lengths

module is directly used for classification. The choice of compositional models for the D module are the same as those for the S module, i.e. Temporal CNN and sequential neural networks like GRU and LSTM, with the difference that it takes sentence embeddings as input instead of word embeddings.

3.3.1 Attention Layer

This layer performs a weighted average of its inputs, where the weights are learned by a neural network, and serve as attention weights (Bahdanau et al., 2014) for the sentences. Let the outputs of the D module be $\{o_1, o_2, \ldots o_n\}$ for a document containing n sentences. We use a two hidden layer neural network for obtaining the attention weights. We obtain the attention weights, $\hat{a}$, as follows

$$a_i = W_a^{(2)} \cdot \tanh(W_a^{(1)} \cdot o_i) \tag{2}$$

$$a = [a_1, a_2, \ldots a_n] \tag{3}$$

$$\hat{a} = softmax(a) \tag{4}$$

Here, $W_a^{(1)} \in \mathbb{R}^{mk}$ and $W_a^{(2)} \in \mathbb{R}^k$ are weight matrices (trained), with m being the size of the output embeddings and k being the size of the hidden layer of the neural network. The final document embedding d is taken as a weighted sum of the outputs, i.e. $d = \sum_{i=1}^{n} \hat{a}_i \cdot o_i$.

3.3.2 Attentive Concatenate Layer

This layer is applied before the application of the compositional model of the D module, taking input the sentence embeddings obtained from the S module. The layers applies attention over the sentences in context of a particular sentence. Let the sentence embeddings of the sentences be $\{s_1, s_2, \ldots s_n\}$ in a document of n sentences. Let A be the attention module presented above. This layer applies A over the sentence embeddings after concatenating the context sentence embedding, to obtain a weighted average embedding, which is concatenated with context sentence embedding to obtain the output, $o_i = [s_i, t_i]$.

$$t_i = A(\{[s_i, s_1], [s_i, s_2], \ldots [s_i, s_n]\}) \tag{5}$$

3.4 Document Classifier

The final document classifier is a neural network with two hidden layers followed by a softmax layer. This takes a m dimensional document vector, d as a input, and produces a label, l, as Real(0) or Satire(1).

$$l = \arg\max(softmax(W_d^{(2)} \cdot f(W_d^{(1)} \cdot d))) \tag{6}$$

Here, f is the ReLU activation function. $W_d^(1) \in \mathbb{R}^{mk}$ and $W_d^{(2)} \in \mathbb{R}^{2k}$ are the weight matrices.

4 Experimental Evaluation

We now detail the evaluation. We first detail the experiment settings, followed by baselines and finally the results.

4.1 Dataset and Preprocessing

	#Train	#Vali	#Test	#Sent	#Word	#Digit
True	101,268	33,756	33,756	32	734	93
Satire	9538	3,608	3,103	25	587	49

Table 2: The split of the dataset and the average count of sentences, words, and digits per document.

4.1.1 Dataset

We utilize the dataset from (Yang et al., 2017). The satirical news were originally collected from the 14 satirical websites, including Onion, theSpoof, SatireWorld, Beaverton, Ossurworld, DailyCurrent, DailyReport, EnduringVision, Gomerblog, NationalReport, SatireTribune, SatireWire, Syruptrap, and UnconfirmedSources. While the true news is collected from Google News and major news outlets, including CNN, Dailymail, WahingtonPost, NYTimes, TheGuardian, and Fox. For true news, we did not split sources. For satirical news, we used two most popular satirical websites, Onion and theSpoof, which have the largest number of satirical news, for training, while the rest of the sources were chosen randomly for test and validation to yield a richer evaluation set. The split and the description of the dataset can be found in Table 2.

4.1.2 Preprocessing

Sentences occurring twice or more are removed from the dataset. Also, each sentence in the dataset is padded to a length of 100, with a special <PAD> word. For sentences having more than 100 words, we consider the sentence as only the first 100 words. Similarly, each news article (or document) is padded to a length of 100 sentences by adding extra sentences containing the <PAD> word. For documents having more than 100 sentences, we consider only the first 100 sentences.

4.2 Experiment Settings

The whole model is trained end-to-end, i.e. all the parameters (weights) of the model are conditioned on the response variable (True news or Satire news). We experimented with different models by changing the choice of word embeddings, the S module, and the D module. For each model, we trained it for 20 epochs with Mini-batch Stochastic Gradient Descent algorithm. Training loss function used was cross entropy. For learning the optimal parameters for each model, we optimized on either the validation accuracy or the validation F1-score based on whichever performed better (these are listed in Table 3). Sentence embedding size and document embedding size were fixed at 300, for all the models. Word embedding (summarized) size was 300 for all models.

The word embedding for the <PAD> was set to all-zero embedding. We used the word embedding for "unk" as the word embedding for all words whose word embeddings were not available. We used the word embedding for "#" to represent all numbers.

4.3 Baselines

We compare our model with 4 baselines:

- **SVM word+char ngrams**: 1,2-word grams plus bigrams and trigrams of the characters
- **Rubin et al**(Rubin et al., 2016): Unigram and bigrams tf-idf with satirical features proposed in their works. It reports a better result than (Burfoot and Baldwin, 2009) so we omit the comparison with the latter.
- **Le et al**(Le and Mikolov, 2014): Unsupervised method learning distributed representation for document
- **Yang et al**(Yang et al., 2017): A 4 level hierarchical network considering charactors, words, paragraphs, and documents. The also incorporate 4 families of linguistic features at both paragraph level and document level.

4.4 Results and Discussion

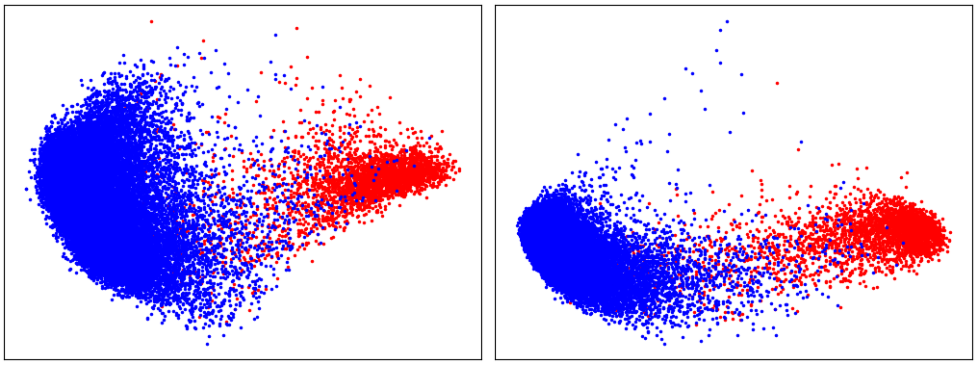
S module	D module	Max	Validation				Test			
			Acc	P	R	F	Acc	P	R	F
Baselines										
SVM word + char ngrams		Val F1	97.43	87.10	81.57	84.24	97.64	90.76	84.12	87.31
Rubin et al (Rubin et al., 2016)		Val F1	97.73	90.21	81.92	85.86	97.79	93.47	82.95	87.90
Le et al (Le and Mikolov, 2014)		Val F1	92.48	58.48	71.66	64.40	90.48	50.52	67.88	57.92
Yang et al (Yang et al., 2017)		Val F1	**98.54**	**93.31**	**89.01**	**91.11**	**98.39**	**93.51**	**89.50**	**91.46**
Word embeddings used: One-hot										
D-B-GRU	A-D-B-GRU	Val F1	97.77	91.78	84.56	88.02	97.96	89.49	85.94	87.68
Word embeddings used: Glove										
CNN	Average	Val acc	97.43	90.86	81.62	85.99	97.77	88.09	85.11	86.57
CNN	CNN	Val acc	97.45	87.45	86.00	86.72	97.76	84.92	**89.30**	87.05
CNN	B-LSTM	Val acc	97.81	95.32	81.37	87.79	98.02	94.22	81.50	87.40
D-B-GRU	A-D-B-GRU	Val acc	98.02	**95.93**	83.09	89.05	98.28	**94.97**	84.04	89.17
D-B-GRU	D-B-GRU	Val acc	98.00	93.05	85.78	89.27	98.23	91.03	87.65	89.31
CNN	A-D-B-GRU	Val acc	97.99	93.67	84.97	89.11	98.26	92.15	86.72	89.35
D-B-LSTM	A-D-B-LSTM	Val F1	**98.18**	94.15	**86.55**	**90.19**	98.31	93.45	86.01	89.57
B-GRU	B-GRU	Val acc	97.90	94.83	82.87	88.44	98.42	94.43	86.43	90.25
CNN	A-B-LSTM	Val F1	97.99	95.42	83.23	88.91	**98.57**	94.20	88.49	**91.25**
Word embeddings used: Glove + fastText										
D-B-LSTM	A-D-B-LSTM	Val F1	97.67	93.49	81.65	87.17	98.30	93.04	86.27	89.53
CNN	A-D-B-GRU	Val F1	**98.06**	92.31	**87.19**	**89.68**	98.39	90.98	**89.81**	90.39
B-GRU	A-D-B-GRU	Val F1	98.01	**95.63**	83.23	89.00	**98.63**	**95.20**	88.20	**91.56**
Word embeddings used: Glove + Syntactic Information										
CNN	A-B-GRU	Val F1	97.81	90.79	**86.08**	**88.37**	97.91	87.40	87.88	87.64
D-B-LSTM	A-D-B-LSTM	Val F1	97.48	89.17	84.22	86.63	98.10	87.77	90.04	88.89
CNN	A-D-B-GRU	Val F1	**97.84**	**93.02**	83.95	88.25	**98.59**	**92.02**	**91.16**	**91.59**

Table 3: Results. Prefix notation; **A: Attentive, D: Deep (3 layers), B: Bidirectional**

The results of the various experiments performed by us, using different choices of word embeddings, S module and D module is summarized in Table 3. Our best model outperforms the baseline models on the dataset. We observe that adding word level syntax information improves the performance only by a small margin. Thus, we can conclude that at the word level, semantic information is more relevant to capture satire than syntax information. For further analysis, we shall refer to the (CNN, A-B-LSTM) model using only Glove embeddings as Model A, and the (B-GRU, A-D-B-GRU) model using Glove and fastText embeddings as Model B.

4.4.1 Analysis of D-module

Figure 2 shows a PCA decomposition of the document embeddings learned by the models, on the test data. We see that satirical news (in red) and real news (in blue) form two separate clusters in both the models. This clearly shows that the embeddings learned by the D module successfully captures satire.

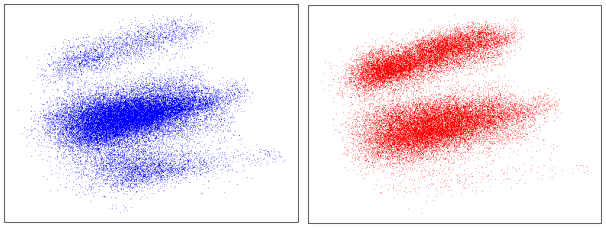


(a) Model A (b) Model B

Figure 2: Visualization of document embeddings of true(blue) and satire(red) news using PCA

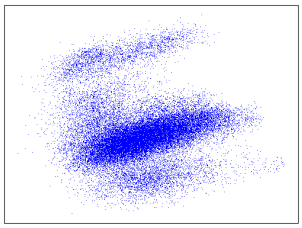
4.4.2 Analysis of S-Module

Applying PCA decomposition on the sentence embeddings learned by Model A reveals a few interesting properties that may be relevant in capturing satire. Figure 3 shows the PCA decomposition of sentence embeddings of news articles from the train and the test data. A closer look at the plots of the train data (Figures 3a) reveal that the sentence embeddings form three clusters. Most of the sentences lie in the central cluster, while some lie the cluster above the central one. A third cluster is observed below the central cluster, and consists mainly of sentences from real news articles. We also note that the density of the upper cluster is higher for satirical news sentences (visibly evident only in the training data), as compared to that for real news. We find that the values along the first principal component (horizontal axis) hold a correlation of **-0.76** with the length of the sentences. This means that having a sentences to the right of the PCA diagram tend of have smaller lengths. It is also worthwhile ot note that the clusters have similar shapes across both classes in train and test explaining that the learned embeddings have decent generalization performances.

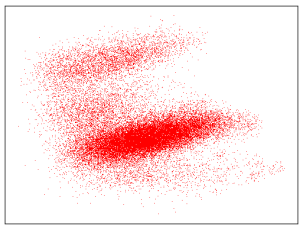


(a) Train: Real news

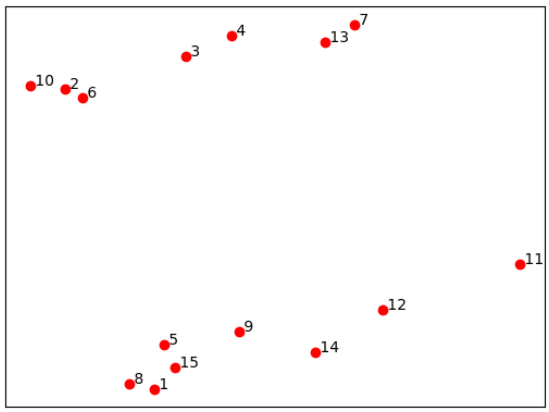
(b) Train: Satire



(c) Test: Real news



(d) Test: Satire



(e) Visualization of sentence embeddings of satire news using PCA

Figure 3: Visualization of sentence embeddings using PCA

We show the PCA decomposition for a satire news article containing 15 sentences (Table 4) in Figure 3e. We clearly see that sentences 2, 3, 4, 6, 7, 10, and 13 (lying in the upper cluster) are separated from the rest of the sentences (lying in the central cluster). The high correlation between the horizontal axis and the length of the sentence is also visible in Figure 3e. It is also interesting to note that sentences 8, 5, 1, 15 in the lower cluster are more verbose as opposed to the upper cluster samples 10, 2, 6 indicating the learned embeddings of the S module could capture verbosity as a fine-grained property of satire.

4.4.3 Analysis of Attention Layer

We normalized the attention weights (using "Min-Max" normalization) of the Attention layer of Model B, such that maximum weight in a news article is 1, and the minimum is 0. Firstly, we observe that the attention weights of the <PAD> sentences are less than 0.01 times the mean attention weight of the other sentences of the news article. This shows the effectiveness of the Attention layer in ignoring <PAD> sentences, thus not adding their contribution in the final document embedding. Next, we observe that the weight (normalized) of the last sentence of a news article is high for satirical news, while is almost 0 for real news. Figure 4a shows the distribution of the attention weights (normalized) over a news article, for a 3 satirical news and 3 real news from the test data. The mean normalized attention weight of the last sentence of satirical news in **0.728**, while the same for real news is **0.07**. Since these weights directly contribute to the document embedding (which we have shown to be able to distinguish satire effectively), they must reflect the amount of satire present in the sentence. Thus, having a higher weight would mean the sentence is more relevant to capture satire. This means that the last sentence of a new article contains

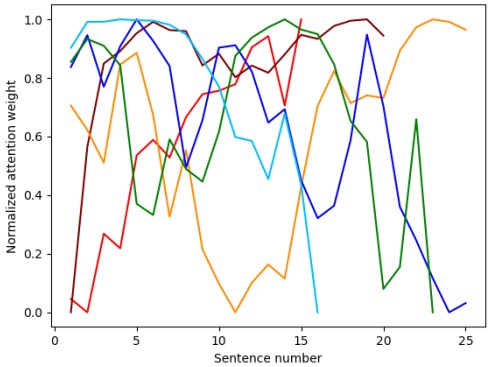
No.	Sentence	Att.
1	OTTAWA With the holiday season drawing near , Prime Minister Harper issued a statement today urging all Canadians to enjoy the time of year that brings us closer to our most beloved industries	0.045
2	" I think that Canadians understand what this season is really about , " the Prime Minister said . "	0.0
3	We ' ve all seen enough Holiday movies to know that what really matters is unfeelingly capitalizing on the emotions of others in order to make millions upon millions of dollars . "	0.268
4	" Actually , now that I mention it , that is what every single other season of the year is also about . "	0.218
5	The statement , released along with a Christmas card featuring the members of the Prime Minister ' s estranged family standing with blank faces in front of a Sears , has already roused the nation into a frenzy of holiday cheer .	0.535
6	" Giving is fine , " said local man Derek Williams , backhand slapping an elderly woman away from a pile of One Direction themed Furbies .	0.588
7	" But what really matters is the opportunity to reward multinationals for doing business in my country . "	0.528
8	Since the statement was released , retail outlets and superstores have made record amounts of money : money which will allow them to make more money , which in turn will be put toward the noble cause of making yet more money.	0.666
9	The Prime Minister has expressed pleasure at the fact that even the innocuous act of giving a gift to a loved one has been subjected to the economics of reckless consumerism .	0.744
10	" Look at all of these Legos , " said the Prime Minister , gesturing at the pile of new boxes of Lego covering Laureen ' s otherwise empty half of the bed .	0.756
11	" I was given these Legos .	0.778
12	Now , of all the people in my family , I have the most Legos .	0.904
13	Economically speaking , I am the best . "	0.942
14	This is not the first time the Prime Minister has used holidays to spread cheer to everyday Canadians .	0.705
15	Last Valentine ' s day , Mr Harper urged Canadians to spice up their romantic lives by switching their personal lubricant from boring old petroleum jelly to titillating new bitumen sand jelly .	1.0

Table 4: Normalized attention weights (from the Attention Layer of Model B) of a satire news article

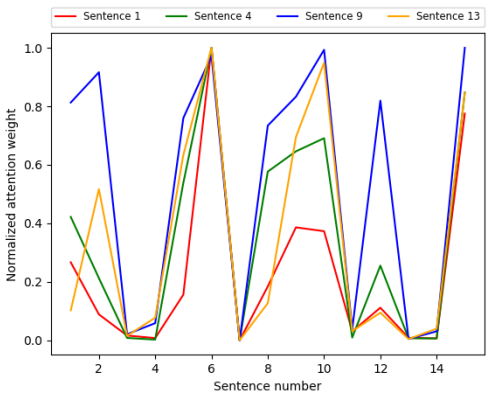
more satirical features if the news is satirical. Therefore, we conclude that the last sentence of a news article is a key feature for detecting satire.

4.4.4 Analysis of Attentive-Concatenate Layer

The normalized attention weights of the Attentive-Concatenate layer reveal that, in the context of each of the sentences of a news article, there are a few key sentences that contain relevant satire information. Figure 4b show the distribution of attention weights (normalized) of the Attentive-Concatenate layer for a few sentences of the news article shown in Table 4. We find that for each sentence, the attention distribution peaks at the same set of sentences (more or less), across all news articles. From this, we can draw the conclusion that these sentences must be more important to capture satire than the other sentences present in the news. For the news article in Table 4, these key sentences are sentence 6, 10, 12 and 15. We also note that the final sentence is one of those key sentences that are important to detect satire. Thus, we conclude that news satire is effectively decided by a few key sentences of the news article.



(a) Attention layer (Real news: [Cyan, Blue, Green], Satire news: [Red, Maroon, Orange])



(b) Attentive-Concatenate Layer

Figure 4: Normalized attention weights

5 Conclusion

We proposed a novel, robust, plug-and-play hierarchical architecture for detecting subtle linguistic nuances like satire in news. Our approach achieves comparable results with the existing state-of-the-art models, without the use of any hand-crafted linguistic feature reflecting satire. This hierarchical approach enables to learn satire features both at the sentence level (learned by S module) and at the document level (learned by D module). An extensive comparison with several state-of-the-art methods for satire news detection was also explored on a real world satire news dataset. We experimented with different choices of initial word embeddings and different S and D modules that include CNN, LSTM and GRU. Experimental results showed slightly superior performance of our proposed architecture with the combination CNN as the S module and a Attentive Deep Bidirectional GRU (3 layers deep) as the D module, using Glove vectors concatenated with syntactic information as the input embeddings to be performing the best. The architecture apart from state-of-the-art detection performances, allowed us to perform fine-grained sentence level analyses giving us a deeper insight into the phenomena of satire. An analysis of the learned models revealed the existence of a few key sentences (including the last sentence) that are important to detect satire. For our future work, we wish to explore Recursive neural networks in order to incorporate the structure of the language for better modeling of sentences.

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments. This work was supported in part by the NSF grant 1527364.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Clint Burfoot and Timothy Baldwin. 2009. Automatic satire detection: Are you having a laugh? In *Proceedings of the ACL-IJCNLP 2009 conference short papers*, pages 161–164. Association for Computational Linguistics.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Niall J Conroy, Victoria L Rubin, and Yimin Chen. 2015. Automatic deception detection: Methods for finding fake news. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the fourteenth conference on computational natural language learning*, pages 107–116. Association for Computational Linguistics.

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

Zhiwei Jin, Juan Cao, Yongdong Zhang, and Jiebo Luo. 2016. News verification by exploiting conflicting social viewpoints in microblogs. In *AAAI*, pages 2972–2978.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, pages 2267–2273.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Arjun Mukherjee, Bing Liu, and Natalie Glance. 2012. Spotting fake reviewer groups in consumer reviews. In *Proceedings of the 21st international conference on World Wide Web*, pages 191–200. ACM.

Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013. Spotting opinion spammers using behavioral footprints. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 632–640. ACM.

Shereen Oraby, Lena Reed, Ryan Compton, Ellen Riloff, Marilyn A Walker, and Steve Whittaker. 2015. And that's a fact: Distinguishing factual and emotional argumentation in online dialogue. In *ArgMining@ HLT-NAACL*, pages 116–126.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 309–319. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Antonio Reyes, Paolo Rosso, and Davide Buscaldi. 2012. From humor recognition to irony detection: The figurative language of social media. *Data & Knowledge Engineering*, 74:1–12.

Victoria L Rubin, Niall J Conroy, Yimin Chen, and Sarah Cornwell. 2016. Fake news or truth? using satirical cues to detect potentially misleading news. In *Proceedings of NAACL-HLT*, pages 7–17.

Natali Ruchansky, Sungyong Seo, and Yan Liu. 2017. Csi: A hybrid deep model for fake news. *arXiv preprint arXiv:1703.06959*.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432.

William Yang Wang. 2017. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*.

Fan Yang, Arjun Mukherjee, and Eduard Dragut. 2017. Satirical news detection and analysis using attention mechanism and linguistic features. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Hu Zhang, Zhuohua Fan, Jia-heng Zheng, and Quanming Liu. 2012. An improving deception detection method in computer-mediated communication. *JNW*, 7(11):1811–1816.

Predicting Stances from Social Media Posts using Factorization Machines

Akira Sasaki[1*], **Kazuaki Hanawa**[2], **Naoaki Okazaki**[3,4], **Kentaro Inui**[2,5]
[1]Recruit Technologies Co., Ltd., Tokyo, Japan
[2]Graduate School of Information Sciences, Tohoku University, Miyagi, Japan
[3]School of Computing, Tokyo Institute of Technology, Tokyo, Japan
[4]National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan
[5]RIKEN Center for Advanced Intelligence Project (AIP), Tokyo, Japan
[1]sakira@r.recruit.co.jp [2]{hanawa,inui}@ecei.tohoku.ac.jp
[3]okazaki@c.titech.ac.jp

Abstract

Social media provide platforms to express, discuss, and shape opinions about events and issues in the real world. An important step to analyze the discussions on social media and to assist in healthy decision-making is stance detection. This paper presents an approach to detect the stance of a user toward a topic based on their stances toward other topics and the social media posts of the user. We apply factorization machines, a widely used method in item recommendation, to model user preferences toward topics from the social media data. The experimental results demonstrate that users' posts are useful to model topic preferences and therefore predict stances of silent users.

1 Introduction

Web and social media provide platforms to express, discuss, and shape opinions about events and issues in the real world. However, these platforms suffer from new emerging problems such as filter bubble (Pariser, 2011), echo chamber (Jamieson and Cappella, 2008), and fake news (Lazer et al., 2018). An important step to analyze the argumentation structure of social media and to assist in healthy decision-making is *stance detection* (Mohammad et al., 2016): predicting whether a given text/user is in favor (agree), against (disagree), or neutral toward a target topic (e.g., Donald Trump).

Unfortunately, users rarely make an explicit statement about a topic. For example, computers may easily detect a stance for the topic of the Trans-Pacific Partnership (TPP) from the sentence, "I totally disagree with TPP," containing the topic word 'TPP' with the explicit linguistic pattern "I totally disagree with ..." However, many social media posts may not refer to a topic but only to its related topics, for example, "We should protect Japanese agriculture." We need commonsense knowledge about the topic (e.g., "TPP suppresses Japanese agriculture") to predict the stance (objection to TPP) for such a sentence (see Figure 1). Furthermore, social media users are often *silent* on a topic (Gong et al., 2016): users who are interested in a topic without posting anything related to it are called the *silent majority* (Lassiter, 2007) and *lurkers* (Gong et al., 2015).

This paper presents an approach to detect the stance of a user toward a topic based on: (1) their stances toward other topics, and (2) the social media posts of the user. We apply factorization machines (FMs), a widely used method in item recommendation, to model user preferences toward topics from data consisting of the stance statements and social media posts of users. Contributions of this work are twofold.

1. We present a method to obtain stance statements from tweets and to model topic preferences from the stance statements jointly with the users' tweets.

2. The experimental results demonstrate that users' posts are useful to model topic preferences and therefore predict stances of silent users.

[*]Work done while at Tohoku University.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3381–3390
Santa Fe, New Mexico, USA, August 20-26, 2018.

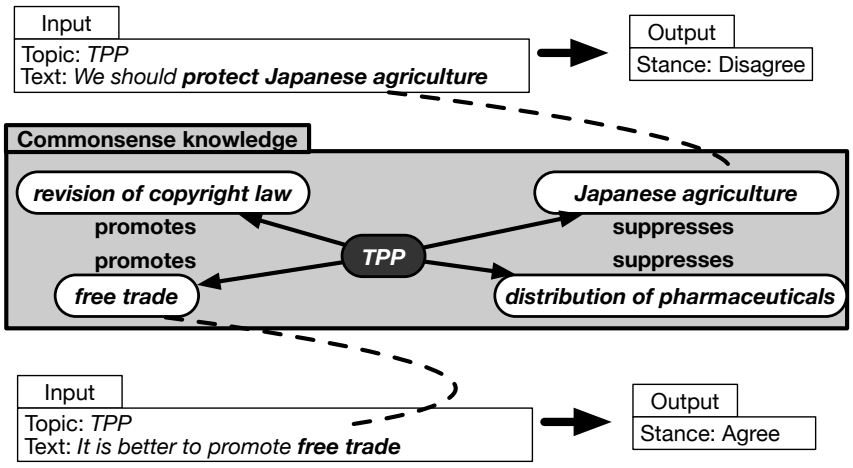


Figure 1: An example of commonsense knowledge.

2 Related Work

In many previous studies, social media such as Twitter has been used to make various analyses and predictions with respect to politics. Of these, several studies have focused on the analysis of political leaning such as liberal or conservative (Bakshy et al., 2015; Wong et al., 2016) and the prediction of people's support for parties or candidates during the election period (Tumasjan et al., 2010; Burnap et al., 2016). These studies were realized because social media has become a means for people to publish messages and to acquire information. Furthermore, because social media has become a part of our daily lives, many candidates in the United States presidential election made use of social media to win the election (Hong and Nadler, 2012).

However, few studies have addressed the silent majority (who hardly expresses a stance), limiting the target to users who explicitly expressed their opinions (Gayo-Avello, 2012). In our work, we shed light on the silent majority as well as vocal users (who express stances explicitly). We aim to predict stances of the silent majority by modeling the topic preferences of users with explicit stance statements and ordinary posts.

Stance detection has been extensively studied in recent years as a task to predict a stance of a user or text toward a specific topic (Murakami and Raymond, 2010; Mohammad et al., 2016; Persing and Ng, 2016). However, most studies have depended heavily on labeled training data (Tutek et al., 2016; Liu et al., 2016). Therefore, these methods had difficulty in predicting a stance toward an unseen topic. Accordingly, the SemEval-2016 task 6B released unlabeled data on a topic to be predicted and labeled data on other topics (Mohammad et al., 2016). However, the accuracy on the setting dropped drastically compared with the setting when labeled data for the topic are given. One reason for the relatively low accuracy is that, if no labeled data are available for a new topic, it is impossible to learn expressions that may imply stances for that topic.

One way to overcome this problem is to incorporate external knowledge about the topics. The most relevant work is our previous work (Sasaki et al., 2017). Main goal of this work was to acquire knowledge such as *"those who agree with the TPP also agree with free trade."* We called such knowledge "inter-topic preferences." We used texts on Twitter and modeled inter-topic preferences using matrix factorization. As a side effect of this modeling, we were able to predict the missing stances of users by considering users' explicit stances toward other topics. However, the method of this previous work cannot be applied to the silent majority, who does not explicitly express any stances.

Several studies have focused on the silent majority and lurkers. Gong et al. (2015) defined users who spend most of their time in social media without posting as lurkers, and analyzed the characteristics and behaviors of such users. Gong et al. (2016) named users who did not explicitly mention specific topical issues "i-silent" users, and analyzed their behavior on social media such as Twitter and Facebook.

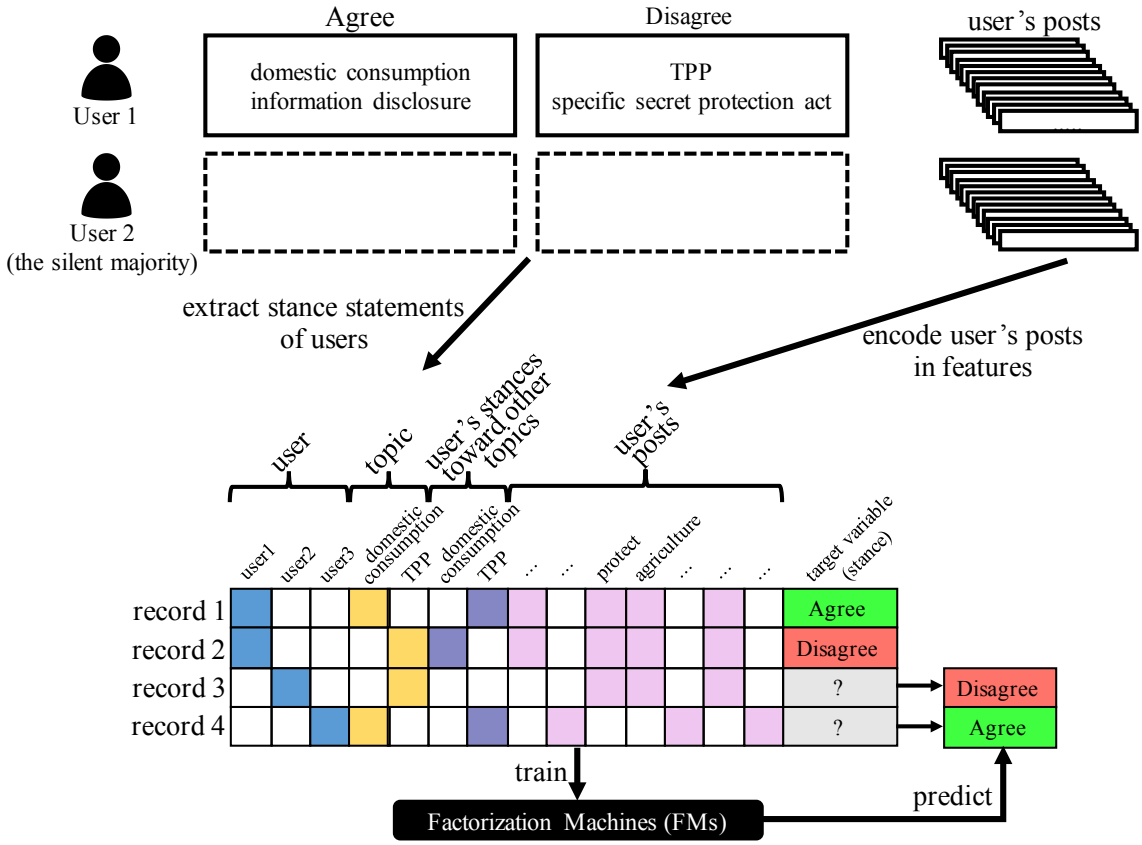


Figure 2: The overview of the proposed method.

They asked the following questions to the users of social media: whether they are interested in specific issues (e.g., healthcare cost and retirement); and whether they have posted on those issues on Twitter or Facebook. In contrast to their work covering only seven issues, we deal with more than 1000 topics. In addition, our method automatically learns the relationships between stances and users' posts. Our method is a versatile method applicable to multiple topics and users, including the silent majority.

Predicting stances of the silent majority is similar to the *cold-start problem* in item recommendation: recommending items to users who have not purchased any items in the past (Schein et al., 2002; Lam et al., 2008). In order to overcome this problem, a recommendation incorporating not only the purchase history of items but also other contexts has been extensively studied as a *context-aware recommendation* (Adomavicius and Tuzhilin, 2015). One method for realizing context-aware recommendation is factorization machines (FMs) (Rendle, 2010; Rendle et al., 2011). In our work, we consider the users' posts as *context* in item recommendation.

3 Proposed Method

The final goal of this work is to construct a user-level stance detection model that can be useful even for the silent users, i.e., users who do not explicitly express any stance. Figure 2 shows the overview of the proposed method. We collect users' stance statements using linguistic patterns that explicitly agree/disagree with topics. For example, we can extract a positive stance toward Donald Trump from the sentence, "*Vote for* Donald Trump", where *vote for* is a linguistic pattern for a positive stance. In addition, we utilize users' posts, for example, "Make America Great Again," as an additional information source to predict a stance of a user. Here, we may think that "Make America Great Again" explicitly agrees with Donald Trump. However, this expression is only applicable to Donald Trump and cannot be used for other general topics. Since it is not practical to manually collect such topic-dependent expressions for a large number of topics, we decided to use topic-independent linguistic patterns to collect users'

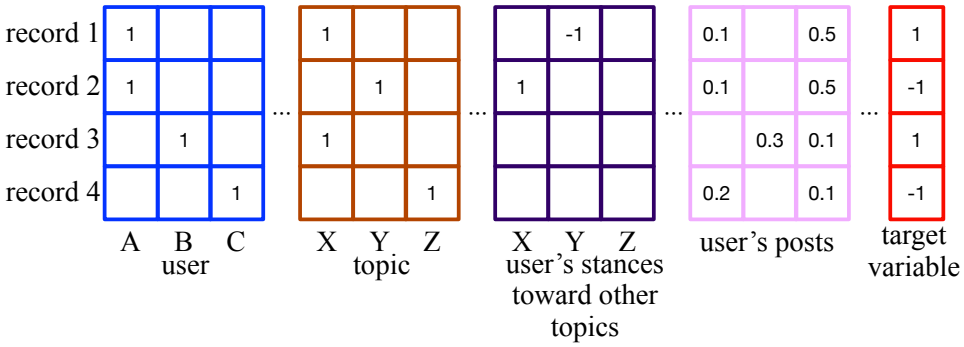


Figure 3: An example of an input of factorization machines.

stances.

In order to model stance statements and users' posts jointly to predict user-level stances, we apply factorization machines (FMs), a widely used and scalable method for recommendation and click-through rate (CTR) predictions, etc. More specifically, we consider that a stance for a topic to analogously correspond to a like/dislike of an item and that a user's post provides additional information for item recommendation. In this section, we first review the formalization of FMs and present definitions of records as an input for FMs. In addition, we explain the details of the linguistic patterns to detect stance statements and the feature extractions used to encode users' posts into FMs.

3.1 Background: factorization machines

FMs predicts a target value (e.g., a rating in item recommendation or a rate in CTR prediction) from n variables (e.g., features of the item) $x_1, ..., x_n$. FMs use second-order feature combinations $x_i x_j$ $(i, j \in \{1, ..., n\})$ as well as first-order features x_i $(i \in \{1, ..., n\})$. Formally, FMs compute a target value using Equation 1,

$$\hat{y}(x) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j. \quad (1)$$

Here, $x_i \in \mathbb{R}$ is the value of the i-th variable, $w_0 \in \mathbb{R}$ is the global bias, $w_i \in \mathbb{R}$ presents the weight of the i-th variable, $v_i \in \mathbb{R}^d$ presents the weight vector used for second-order feature combinations, d is the number of dimensions of the weight vectors, and $\langle v_i, v_j \rangle$ denotes a dot product of two vectors v_i and v_j. We use `tffm`[1] to train a model (i.e., to find the parameters $w_0, ..., w_n$ and $v_1, ..., v_n$ for the given supervision data).

3.2 Applying factorization machines to our task

In order to apply FMs to user-level stance detection, we assume a target variable to present the stance of a user toward a topic: $+1$ (a positive stance to the topic) and -1 (a negative stance to the topic). The user and topic are represented as features for the input to FMs. In this work, we define four types of variables to build a *record* for FMs: user, topic, user's stances toward other topics, and user's posts. Figure 3 shows an example of input records. We describe each type of variable in detail.

user

We define a variable to identify each user in the data. We set 1 to the variable corresponding to the user of a record and 0 to the other variables in this category.

topic

We also define a variable for each topic in the data. We set 1 to the variable for which the target variable presents a stance. Given that a record is associated with a topic, only one variable in this category is set to 1 and one of the variables in this category must be 1.

[1] `https://github.com/geffy/tffm`

user's stances toward other topics

This category of variables indicates stances of the user toward other topics. The number of variables in this category is the same as that in the topic category. A value of this category ranges from -1 (negative to the topic) to $+1$ (positive to the topic). We will explain the values of these features in detail in Section 3.3.

user's posts

Variables in this category encode textual features of posts sent by the user. We describe these variables in detail in Section 3.4.

Let us consider an example where the user A is favor to the topic X, but against to the topic Y. Figure 3 illustrates records for this example. Record 1 presents the stance toward topic X as the target variable and the stance toward Y as other topics. Similarly, record 2 presents the stance toward topic Y as the target variable and the stance toward X as other topics. Given that records 1 and 2 are for the same user, the values for the user's posts are essentially the same. When we ignore the features of the user's posts for simplicity, Equation 3 for the first record is,

$$\begin{aligned} 1 = w_0 + w_{\mathrm{user:A}} + w_{\mathrm{topic:X}} - w_{\mathrm{other:Y}} \\ + \langle v_{\mathrm{user:A}}, v_{\mathrm{topic:X}} \rangle - \langle v_{\mathrm{topic:X}}, v_{\mathrm{other:Y}} \rangle - \langle v_{\mathrm{other:Y}}, v_{\mathrm{user:A}} \rangle. \end{aligned} \tag{2}$$

Here, $\mathrm{user:A}$, $\mathrm{topic:X}$, and $\mathrm{other:Y}$ present the index numbers of features indicating that the user is A, the topic of the target variable is X, and the stance for the other topic is Y, respectively. Similarly, Equation 3 for the second record is,

$$\begin{aligned} -1 = w_0 + w_{\mathrm{user:A}} + w_{\mathrm{topic:Y}} + w_{\mathrm{other:X}} \\ + \langle v_{\mathrm{user:A}}, v_{\mathrm{topic:Y}} \rangle + \langle v_{\mathrm{topic:Y}}, v_{\mathrm{other:X}} \rangle + \langle v_{\mathrm{other:X}}, v_{\mathrm{user:A}} \rangle, \end{aligned} \tag{3}$$

where $\mathrm{topic:Y}$ and $\mathrm{other:X}$ denote the index numbers of features indicating that the topic of the target variable is Y, and the stance for the other topic is X, respectively.

3.3 Extracting stance statements of users

In order to obtain data for target variables and stances on other topics, we extract stance statements of users from SNS data. Throughout this work, we use SNS data consisting of 1,763,164,770 Japanese tweets (444,321 users) crawled from February 6, 2013, to September 30, 2016[2].

Given that Twitter does not have an option to store an explicit indicator about a stance, we use a set of 200 linguistic patterns to automatically extract stance statements from tweets (Sasaki et al., 2017)[3]. The patterns were manually designed to extract stance statements with high-precision and low-recall, for example, "support X" (positive to X) and "X is terrible" (negative to X). We compute a stance value for a user toward a topic as,

$$\frac{(\#\mathrm{pos} - \#\mathrm{neg})}{(\#\mathrm{pos} + \#\mathrm{neg})}. \tag{4}$$

Here, $\#\mathrm{pos}$ and $\#\mathrm{neg}$ present the number of tweets from the user that included positive patterns and negative patterns, respectively, accompanying the target topic.

3.4 Encoding users' posts in features

In order to acquire features based on posts of each user, we focus on posts of users extracted in Section 3.3 not including texts containing positive or negative patterns. Here, tweets including positive or negative patterns are eliminated because, when evaluating the performance of the stance detection for

[2]We used the Twitter API to crawl these tweets. We collected 43B Japanese tweets since February 6, 2013 without a specific purpose. Because the volume of the entire crawled tweets is extremely large, we randomly selected one-sixteenth users based on hash values computed from screen names. We removed retweets from the corpus.

[3]http://www.cl.ecei.tohoku.ac.jp/ja_stance/

Feature type	Examples
1-gram	war
2-gram	(war, bill)
adnominal	(terrible, bill)
adjective → noun phrase	(long, working hours)
noun phrase → adjective	(train, plentiful)
noun phrase → verb	(salary level, return)

Table 1: Examples of features based on users' posts.

Used information				Numbers of stances stated					Numbers of stances stated				
Topic	User	Other	Posts	≥ 0	≥ 5	≥ 10	≥ 30	≥ 50	≤ 0	≤ 5	≤ 10	≤ 30	≤ 50
✓	✓	✓	✓	62.80	62.30	63.35	72.55	85.46	65.35	62.99	62.67	62.66	62.71
✓	✓	✓		62.62	62.69	63.45	69.78	87.22	64.97	62.53	62.44	62.50	62.52
✓	✓		✓	63.34	63.22	63.76	73.70	88.11	65.24	63.40	63.21	63.18	63.24
✓	✓			62.97	62.39	63.64	70.59	88.11	65.11	63.14	62.80	62.86	62.87
✓		✓	✓	65.99	66.40	66.83	74.39	**89.43**	**66.99**	65.78	65.81	65.86	65.90
✓		✓		63.95	63.82	63.39	66.44	74.45	65.10	64.10	64.04	63.90	63.91
✓			✓	**66.45**	**66.57**	**67.23**	**75.09**	88.55	66.91	**66.37**	**66.25**	**66.31**	**66.36**
Majority baseline				63.67	62.25	60.99	55.82	55.51	65.23	64.47	64.18	63.78	63.70
Matrix factorization (topic&user)				61.12	64.17	64.56	72.55	80.18	54.31	59.63	60.48	60.95	61.05

Table 2: Evaluation result of completing missing stances. Each cell presents the accuracy. "Numbers of stances stated" indicates the condition for evaluation. For example, as to ≥ 5, we evaluate the performance for completing missing stances for users who declared their stances toward no less than five topics. Note that, this treatment is applied only for the test portion of the cross validation.

each topic, these patterns are strong clues and it is thought that a proper evaluation could not be performed if they were included. We perform preprocessing on target tweets using the Japanese dependency parsing tool `CaboCha` (Kudo et al., 2004)[4]. We then extracted features about uni-grams, bi-grams and features based on dependencies from posts of each user. Table 1 shows examples of extracted features. Here, we choose only adnominal, adjective → noun phrase, noun phrase → adjective, and noun phrase → verb as features based on dependencies. Verb → noun phrase and triplet such as (subject, predicate, object) are not suitable as features because the number of users using each feature is very small and they become sparse.

4 Evaluation

4.1 Completing missing stances

The first experiment examines how well FMs complete missing stances of users with the additional information from users' posts. In order to measure the performance of the completion, we hide some stances in the matrix and evaluate the accuracy of the prediction. Given that it is impossible to evaluate users who do not express agreement/disagreement, we only select users who declare stances more than once. As a result, a matrix consisting of 326,202 stances was obtained. There are 130,635 users and 1,142 topics in the matrix. In the evaluation, we use 5% of the rows in this matrix as a validation set for parameter tuning and an early stopping of training. We evaluate the accuracy of a 10-fold cross validation with respect to the remainder of the matrix (95% of the rows).

For comparison, we implemented the majority baseline and matrix factorization. The majority baseline predicts a stance of a user based on the stance of the majority (other users) in the training set. For example, if the number of users who expressed disagreement toward "Article 9" exceeded the number of users who expressed agreement, the majority baseline always predicts "disagreement" toward "Article 9". For matrix factorization, we use the same parameters as in Sasaki et al. (2017) ($k = 100$, $\lambda_P = 0.1$,

[4]We used `mecab-ipadic-NEologd` (`https://github.com/neologd/mecab-ipadic-neologd`) as a dictionary.

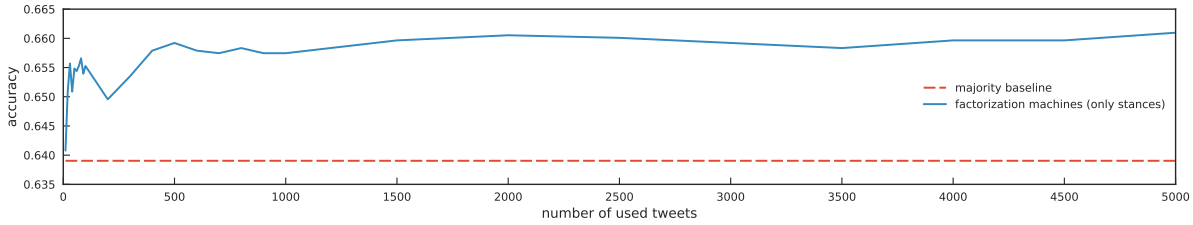


Figure 4: Evaluation result for pseudo-silent majority.

$\lambda_Q = 0.1$). Note that, matrix factorization used in Sasaki et al. (2017) cannot incorporate text information (users' posts). Therefore, we only use stances of users in the baseline method of matrix factorization. For FMs, we set the number of dimensions of the weight vector v_i to $d = 8$.

Table 2 shows the evaluation result. Here, topics that are extremely biased toward an agreement or a disagreement are thought to be inappropriate for the evaluation because stances on such topics can easily be predicted with even a simple model. Therefore, we performed experiments with limited targets considering t_{avg} , which is an average stance on a topic in the data. More specifically, we select targets of $-0.5 \leq t_{avg} \leq 0.5$. Table 2 indicates that, even though the matrix factorization exceeds the majority baseline for instances with numbers of stances stated ≥ 5, the result was lower than the majority baseline for instances with numbers of stances stated ≤ 5. Conversely, all methods of FMs outperformed the majority baseline in most cases. In addition, for the FMs, users' posts contributed to increasing the accuracy compared to a method using only users' stances. From this result, we confirmed that users' posts are useful clues for predicting their stances.

For instances, without stances stated (numbers of stances stated ≤ 0), methods that do not consider users' posts underperform the majority baseline. Conversely, methods utilizing users' posts were better than the majority baseline. This suggests that, even for users without stances stated, it is possible to predict their stances for each topic with a certain degree of accuracy if posts of such users are available. Meanwhile, how many posts for each user are sufficient to make a prediction? In Section 4.2, we focus on these users (pseudo-silent majority). Hereafter, we conducted experiments using the model with topic variables and users' posts variables, which showed the highest performance in numbers of stances stated ≥ 0.

4.2 Evaluation for pseudo-silent majority

In this subsection, we treat users with only one stance as the pseudo silent majority, and perform evaluations and analyses. Note that, it would be preferable to evaluate users who did not declare stances at all as the silent majority. However, since it is virtually impossible for a third person to annotate stances of such users, we defined the pseudo silent majority for this experiment.

For the training data, we used all the users who expressed two or more stances. For the test data, we used users who expressed only one stances and posted more than 5,000 tweets. Then, we changed the number of tweets derived from the pseudo-silent majority, and measured the change in accuracy. Figure 4 shows the evaluation result. This result shows that 500 tweets are enough for predicting stances of a silent user.

4.3 The size of the silent majority

In Sections 4.1 and 4.2, by considering users' posts with our proposed method, we could also predict stances of the pseudo silent majority, which should have characteristics close to the 'real' silent majority. Then, how many users can the proposed method target at? In this work, as described in Section 3.3, we treated 444,321 users as the population. We calculate the number of topics for which users explicitly declared a stance. From this result (Figure 5), we can see that 313,686 (70.60%) of 444,321 users did not explicitly declare stances. The model of Sasaki et al. (2017) cannot be applied to these users.

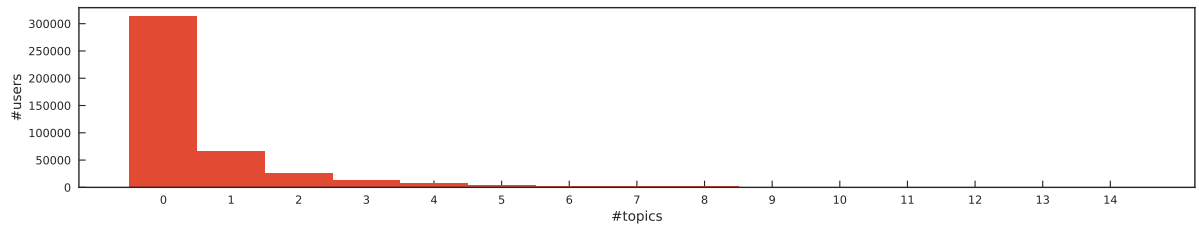


Figure 5: Number of topics for which users explicitly declared a stance.

variable usage				ρ
topic	user	other	users' posts	—
✓	✓	✓	✓	0.0650
✓	✓	✓		0.0380
✓	✓		✓	0.0047
✓	✓			0.0402
✓		✓	✓	0.0935
✓		✓		0.0384
✓			✓	0.2049
matrix factorization (topic&user)				**0.2652**
(Sasaki et al., 2017)				0.2210

Table 3: Spearman's rank correlation coefficient (ρ) between human judgments and similarity of feature vectors obtained via our method.

4.4 Inter-topic preference

In FMs, each variable (user, topic, user's stances to other topics, and users' posts) is expressed as a feature vector. We examine whether inter-topic preferences (e.g., "those who agree with A also agree with B") can be derived from the feature vectors. In this subsection, we perform a quantitative evaluation to verify whether the model could acquire inter-topic preferences.

Here, we use the dataset created in Sasaki et al. (2017). This dataset includes inter-topic preferences given by six to ten crowd workers for 450 topic pairs. An inter-topic preference is defined as one of the following: -1 (*those who agree/disagree with topic A may conversely disagree/agree with topic B*), +1 (*those who agree/disagree with topic A may also agree/disagree with topic B*), and 0 (*no association between A and B*). Then, we obtained the average of the values of the multiple workers. We exclude topics that do not exist in the data used in our study. Thus, we used 221 topic pairs as evaluation targets.

We calculated Spearman's rank correlation coefficient (ρ) between manually created scores and similarities of feature vectors for topic pairs. Table 3 shows the result. The result of the proposed method was are lower than that of matrix factorization. This is probably because the proposed method encourage the model to utilize feature vectors corresponding to users' posts. In other words, stances are represented not only by vectors of topics but also by vectors of other variables. We would like to examine this phenomenon in more detail in the future.

5 Conclusion

In this work, we focused on posts of users on social media. In the evaluation, we confirmed that posts of users contribute to predicting stances of the users. This work dealt with 444,321 users on Twitter, approximately 70% of which do not express any stances for any topics. Using our proposed method, it is possible to target these users by considering their posts. Therefore, our method is useful for analyzing opinions, including those of the silent majority.

However, the performance of the stance detection in this work is not yet sufficient. To improve this performance, there are several points that need to be considered. First, even though features based on

n-gram and dependencies are currently used as features of users' posts, it may be possible to replace these with distributed representations. In recent years, multiple researchers have tried to build distributed representations of sentences (Logeswaran and Lee, 2018; Hill et al., 2016). These methods are known to demonstrate high performance in many tasks. By incorporating these methods into our proposed method, we will be able to express users' posts more compactly, and this is expected to improve the accuracy of predicting users' stances. In addition, FMs used in this work are advantageous because various types of features can be used simultaneously in an input matrix. Therefore, it is expected that social media specific features such as users' follow-follower relationships, retweet relationships, or profile could be useful information for predicting stances.

Moreover, it is possible to overlook the public opinions including the silent majority for each topic with our proposed method. In the future, we hope to apply our proposed method to newly appearing topics. We can then construct a framework that can monitor public opinions on various topics simultaneously. The above system is expected to be one application of our proposed method.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 15H05318, JST CREST Grant Number JPMJCR1301 and AIP Challenge Program, Japan.

References

Gediminas Adomavicius and Alexander Tuzhilin. 2015. Context-aware recommender systems. In *Recommender systems handbook*, pages 191–226. Springer.

Eytan Bakshy, Solomon Messing, and Lada A. Adamic. 2015. Exposure to ideologically diverse news and opinion on Facebook. *Science*, 348(6239):1130–1132.

Pete Burnap, Rachel Gibson, Luke Sloan, Rosalynd Southern, and Matthew Williams. 2016. 140 characters to victory?: Using Twitter to predict the UK 2015 general election. *Electoral Studies*, 41:230–233.

Daniel Gayo-Avello. 2012. "I wanted to predict elections with Twitter and all I got was this lousy paper" - A balanced survey on election prediction using Twitter data. *CoRR*, abs/1204.6441.

Wei Gong, Ee-Peng Lim, and Feida Zhu. 2015. Characterizing silent users in social media communities. In *Proceedings of the 9th International AAAI Conference on Web and Social Media*, pages 140–149.

Wei Gong, Ee-Peng Lim, Feida Zhu, and Pei Hua Cher. 2016. On unravelling opinions of issue specific-silent users in social media. In *Proceedings of the 10th International AAAI Conference on Web and Social Media*, pages 141–150.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377.

Sounman Hong and Daniel Nadler. 2012. Which candidates do the public discuss online in an election campaign?: The use of social media by 2012 presidential candidates and its impact on candidate salience. *Government Information Quarterly*, 29(4):455 – 461.

Kathleen Hall Jamieson and Joseph N. Cappella. 2008. *Echo Chamber: Rush Limbaugh and the Conservative Media Establishment*. Oxford University Press.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 230–237.

Xuan Nhat Lam, Thuc Vu, Trong Duc Le, and Anh Duc Duong. 2008. Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication*, pages 208–211.

Matthew D. Lassiter. 2007. *The Silent Majority: Suburban Politics in the Sunbelt South*. Princeton University Press.

David M. J. Lazer, Matthew A. Baum, Yochai Benkler, Adam J. Berinsky, Kelly M. Greenhill, Filippo Menczer, Miriam J. Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, Michael Schudson, Steven A. Sloman, Cass R. Sunstein, Emily A. Thorson, Duncan J. Watts, and Jonathan L. Zittrain. 2018. The science of fake news. *Science*, 359(6380):1094–1096.

Can Liu, Wen Li, Bradford Demarest, Yue Chen, Sara Couture, Daniel Dakota, Nikita Haduong, Noah Kaufman, Andrew Lamont, Manan Pancholi, Kenneth Steimel, and Sandra Kübler. 2016. IUCL at SemEval-2016 task 6: An ensemble model for stance detection in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, pages 394–400.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *Proceedings of the 6th International Conference on Learning Representations*.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, pages 31–41.

Akiko Murakami and Rudy Raymond. 2010. Support or oppose?: classifying positions in online debates from reply activities and opinion expressions. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 869–875.

Eli Pariser. 2011. *The Filter Bubble: How the New Personalized Web Is Changing What We Read and How We Think*. Penguin Books.

Isaac Persing and Vincent Ng. 2016. Modeling stance in student essays. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2174–2184.

Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 635–644.

Steffen Rendle. 2010. Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining series*, pages 995–1000.

Akira Sasaki, Kazuaki Hanawa, Naoaki Okazaki, and Kentaro Inui. 2017. Other topics you may also agree or disagree: Modeling inter-topic preferences using tweets and matrix factorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 398–408.

Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260.

Andranik Tumasjan, Timm Oliver Sprenger, Philipp G Sandner, and Isabell M Welpe. 2010. Predicting elections with Twitter: What 140 characters reveal about political sentiment. In *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, pages 178–185.

Martin Tutek, Ivan Sekulic, Paula Gombar, Ivan Paljak, Filip Culinovic, Filip Boltuzic, Mladen Karan, Domagoj Alagić, and Jan Šnajder. 2016. TakeLab at SemEval-2016 task 6: Stance classification in tweets using a genetic algorithm based ensemble. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 464–468.

Felix Ming Fai Wong, Chee Wei Tan, Soumya Sen, and Mung Chiang. 2016. Quantifying political leaning from tweets, retweets, and retweeters. *IEEE Transactions on Knowledge and Data Engineering*, 28(8):2158–2172.

Automatic Detection of Fake News

Verónica Pérez-Rosas[1], **Bennett Kleinberg**[2], **Alexandra Lefevre**[1]
Rada Mihalcea[1]
[1]Computer Science and Engineering, University of Michigan
[2]Department of Psychology, University of Amsterdam
vrncapr@umich.edu,b.a.r.kleinberg@uva.nl,mihalcea@umich.edu

Abstract

The proliferation of misleading information in everyday access media outlets such as social media feeds, news blogs, and online newspapers have made it challenging to identify trustworthy news sources, thus increasing the need for computational tools able to provide insights into the reliability of online content. In this paper, we focus on the automatic identification of fake content in online news. Our contribution is twofold. First, we introduce two novel datasets for the task of fake news detection, covering seven different news domains. We describe the collection, annotation, and validation process in detail and present several exploratory analyses on the identification of linguistic differences in fake and legitimate news content. Second, we conduct a set of learning experiments to build accurate fake news detectors, and show that we can achieve accuracies of up to 76%. In addition, we provide comparative analyses of the automatic and manual identification of fake news.

1 Introduction

Fake news detection has recently attracted a growing interest from the general public and researchers as the circulation of misinformation online increases, particularly in media outlets such as social media feeds, news blogs, and online newspapers. A recent report by the Jumpshot Tech Blog showed that Facebook referrals accounted for 50% of the total traffic to fake news sites and 20% total traffic to reputable websites.[1] Since as many as 62% of U.S. adults consume news on social media (Jeffrey and Elisa, 2016), being able to identify fake content in online sources is a pressing need.

Until now, computational approaches for fake news detection have relied on satirical news sources such as "The Onion" (Rubin et al., 2016), viral news tracking websites such as BuzzFeed (Potthast et al., 2017) and fact-checking websites such as "politiFact" (Wang, 2017) and "Snopes" (Popat et al., 2016). However, the use of these sources poses several challenges and potential drawbacks. For instance, using satirical content as a source for fake content can bring underlying confounding factors into the analysis, such as humor and absurdity. This is particularly the case for satirical news from "The Onion", which has been used in the past to explore other text properties such as humor (Mihalcea and Strapparava, 2005) and irony (Wallace, 2015). Moreover, fact-checking websites are usually constrained to a particular domain of interest, such as politics, and require human expertise to verify the news claims making it difficult to obtain datasets that provide some degree of generalization over other domains (Chen et al., 2015).

In this paper, we develop computational resources and models for the task of fake news detection. We introduce two novel datasets covering seven different domains. One of the datasets is collected by combining manual and crowdsourced annotation approaches, while the second is collected directly from the web. Using these datasets, we conduct several exploratory analyses to identify linguistic properties that are predominantly present in fake news content, and we build fake news detectors relying on linguistic features that achieve accuracies of up to 76%. To place our results in perspective, we compare the performance of the developed classifiers with an empirical human baseline.

[1]https://www.jumpshot.com/data-facebooks-fake-news-problem/

Proceedings of the 27th International Conference on Computational Linguistics, pages 3391–3401
Santa Fe, New Mexico, USA, August 20-26, 2018.

2 Related Work

To date, there are two important lines of research into the automated classification of genuine and fake news items. First, on a conceptual level, a distinction has been made between "three types of fake news" (Rubin et al., 2015): serious fabrications (i.e., news items about false and non-existing events or information such as celebrity gossip), hoaxes (i.e., providing false information via, for example, social media with the intention to be picked up by traditional news websites), and satire (i.e., humorous news items that mimic genuine news but contain irony and absurdity). Second, on an operational level, linguistic and fact-checking based approaches have been proposed to discriminate between real and fake news content (Conroy et al., 2015).

The linguistic approach attempts to identify text properties, such as writing style and content, that can help to discriminate real from fake news articles. The underlying assumption for this approach is that linguistic behaviors such as punctuation usage, word type choices, part-of-speech tags, and emotional valence of a text are rather involuntary and therefore outside of the author's control, thus revealing important insights into the nature of the text. The linguistic approach has yielded promising results in differentiating satire from real news (Rubin et al., 2016). Relying in a corpus of satire news (from *The Onion* and *The Beaverton*) and real news (*The Toronto Star* and *The New York Times*) in four domains (civics, science, business, soft news), the authors explored the use of several linguistic features to discriminate between real and satirical news content. The best classification performances were achieved with feature sets representing absurdity, punctuation, and grammar.

On the other hand, fact-checking approaches rely on automated verification of propositions made in the news articles (e.g., "Barack Obama assumed office on a Tuesday") to assess the truthfulness of their claims (Conroy et al., 2015). Knowledge databases such as DBpedia [2] have been used to query the Web in a structured manner. The results of such queries can then be used to test whether different sources also contain information confirming the news claim (e.g., that Barack Obama assumed office on a Tuesday). Other works have used social network activity (e.g., tweets) on a specific news item to assess its credibility, for instance by identifying tweets voicing skepticism about the truthfulness of a claim made in a news article (Hannak et al., 2014; Jin et al., 2014). Although fact-checking approaches are becoming increasingly powerful, a major drawback is that they are built on the premise that the information can be verified using external sources, for instance FakeCheck.org and Snopes.com. However, this is not a straightforward task, as external sources might not be available, particularly for just-published news items. Therefore, the fact-checking approach is predominantly useful for the detection of deception in texts for which external, verifiable information is available.

Furthermore, also related to the current paper is work on the automatic identification of deceptive content, which has explored domains such as forums, consumer reviews websites, online advertising, online dating, and crowdfunding platforms (Warkentin et al., 2010; Ott et al., 2011a; Zhang and Guan, 2008; Toma and Hancock, 2010; Shafqat et al., 2016). While fake news detection is closely related to deception detection (i.e. determining whether or not someone is lying), there are important differences between the two tasks. First, fake news producers usually seek political or financial gain as well as self-promotion while deceivers have motivations that are more socially driven such as self protection, conflict or harm avoidance, impression management or identity concealment. Second, they differ significantly in their target and in the form they propagate: fake news items are usually disseminated at larger scale through the Internet and social media whereas deception is more specifically targeted at individuals. However, since both tasks deal with deceptive content, we hypothesize that there are linguistic aspects that might be shared between these tasks. Thus, we focus on the linguistic approach and build upon an emerging body of research on computer-automated verbal deception detection (Fitzpatrick et al., 2015).

3 Fake News Datasets

As highlighted earlier, the datasets used in previous work have either relied on satirical news (e.g., "The Onion"), which also have confounds such as humor or irony; or used fact-checking websites (e.g., "poli-

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

Dataset	Class	Entries	Average Words/Sent	Words
FakeNewsAMT	Fake	240	132/5	31,990
	Legitimate	240	139/5	33,378
Celebrity	Fake	250	399/17	39,440
	Legitimate	250	700/33	70,975

Table 1: Class distribution and word statistics for fake news datasets

tiFact" or "Snopes"), which are typically focused on only one domain (generally politics). To address these shortcomings, we decided to construct two novel datasets containing fake news covering several news domains and specifically model the deceptive property of fake news. One dataset is collected via crowdsourcing covering six news domains (e.g., business, education). The second dataset is obtained directly from the web and covers celebrity news.

3.1 Crowdsourced Fake News Dataset

Collecting Legitimate News. We started by collecting a dataset of legitimate news belonging to six different domains (sports, business, entertainment, politics, technology, and education). The news were obtained from a variety of mainstream news websites predominantly in the US such as the ABCNews, CNN, USAToday, NewYorkTimes, FoxNews, Bloomberg, and CNET among others.

To ensure the veracity of the news, we conducted manual fact-checking on the news content, which included verifying the news source and cross-referencing information among several sources. Using this approach, we collected 40 news in each of the six domains, for a total of 240 legitimate news.

LEGITIMATE	FAKE
Nintendo Switch game console to launch in March for $299 The Nintendo Switch video game console will sell for about $260 in Japan, starting March 3, the same date as its global rollout in the U.S. and Europe. The Japanese company promises the device will be packed with fun features of all its past machines and more. Nintendo is promising a more immersive, interactive experience with the Switch, including online playing and using the remote controller in games that don't require players to be constantly staring at a display.	**New Nintendo Switch game console to launch in March for $99** Nintendo plans a promotional roll out of it's new Nintendo switch game console. For a limited time, the console will roll out for an introductory price of $99. Nintendo promises to pack the new console with fun features not present in past machines. The new console contains new features such as motion detectors and immerse and interactive gaming. The new introductory price will be available for two months to show the public the new advances in gaming.

Table 2: Sample legitimate and crowdsourced fake news in the Technology domain

LEGITIMATE	FAKE
Kim And Kanye Silence Divorce Rumors With Family Photo. Kanye took to Twitter on Tuesday to share a photo of his family, simply writing, "Happy Holidays." In the picture, seemingly taken at Kris Jenner's annual Christmas Eve party, Kim and a newly blond Kanye pose with their children, North, 3, and Saint, 1. After Kanyes hospitalization, reports that there was trouble in paradise with Kim started brewing. But E! News shut down the speculation with a family source denying the rumors and telling the site, "It's been a very hard couple of months."	**Kim Kardashian Reportedly Cheating With Marquette King as She Gears up for Divorce From Kanye West.** Kim Kardashian is ready to file for divorce from Kanye West but has she REALLY been cheating on him with Oakland Raiders punter Marquette King? The NFL star seemingly took to Twitter to address rumors that they've been getting close amid Kanye's mental breakdown, which were originally started by sports blogger Terez Owens. While he doesn't appear to confirm or deny an affair, her reps said there is "no truth whatsoever" to the reports and labeled the situation "fabricated."

Table 3: Sample legitimate and web fake news in the Celebrity domain

Crowdsourcing Fake News. To generate fake versions of the legitimate news items, we made use of crowdsourcing via Amazon Mechanical Turk (AMT). Despite having been successfully used in the past to collect deceptive data on several domains, including opinion reviews (Ott et al., 2011b), and controversial topics such as abortion and death penalty (Pérez-Rosas and Mihalcea, 2015), the use of

AMT on the news domain poses additional challenges. First, the reporting language used by journalists might differ from AMT workers' language (e.g., journalistic vs. informal style). Second, journalistic articles are usually lengthier than consumer reviews and opinions, thus increasing the difficulty of the task for AMT workers as they would be required to read a full news article and create a fake version from it.

To address the former, we asked the workers to the extent possible to emulate a journalistic style in their writing so we could obtain news with homogeneous writing style. To simplify the fake news production task (and also address the latter challenge), we also opted for working with a shorter version of the original news article. Thus, we manually select a news excerpt – about two or three paragraphs – that summarizes the news article. This process resulted in 240 news excerpts derived from the legitimate news dataset collected earlier.

Next, we set up an AMT task that asked workers to generate a fake version of a given news. Each hit included the legitimate news headline and its corresponding body. We instructed workers to produce both a fake headline and a fake news body within the same topic and length as the original news. Workers were also requested to avoid unrealistic content and to keep the names mentioned in the news. The fake news were produced by unique authors, as we allowed only a single submission per worker. We restricted the submission to workers located in the US as they might be more familiar with news published in the US media. In addition, to ensure crowdsourcing quality, we restricted participation to workers who had an AMT approval rate of at least 95% for previous tasks.

It took approximately five days to collect 240 fake news. Each hit was manually checked for spam and to make sure workers followed the provided guidelines. In general, we received few spam responses and most of the workers followed instructions satisfactorily; the only exceptions were a few cases where they provided only the headline or included unrealistic content. The corpus statistics, including class distribution and word/sentence statistics, are shown in Table 1. Table 2 shows an excerpt of a fake news article in our dataset, along with its legitimate version, in the technology domain.

Interestingly, we observed that AMT workers succeeded in mimicking the reporting style from the original news, which may be partly explained by typical verbal mirroring behaviors that drive individuals to produce utterances that match the grammatical structure of sentences they have recently read (Ireland and Pennebaker, 2010).

Importantly, note that the AMT process of generating news mirrors the fake news production process quite well: similar to the AMT workers, the producers of fake news write for the purpose of generating quick money, and do not undergo the same professional writing training that the writers of legitimate news do.

Throughout the rest of the paper, we refer to this crowdsourced dataset as FakeNewsAMT.

3.2 Web Dataset Celebrity

For our second dataset, we sought to collect news from web sources to identify fake content that naturally occurs on the web. We opted for collecting news from public figures as they are frequently targeted by rumors, hoaxes, and fake reports. We focused mainly on celebrities (actors, singers, socialites, and politicians) and our sources include online magazines such as Entertainment Weekly, People Magazine, RadarOnline, among other tabloid and entertainment-oriented publications. The data was collected in pairs, with one article being legitimate and the other fake. In order to determine if a given celebrity news was legitimate or not, the claims made in the article were evaluated using gossip-checking sites such as "GossipCop.com", and also cross-referenced with information from other entertainment news sources on the web.

During the initial stages of the data collection, we noticed that celebrity news tend to center on sensational topics that sources believe readers want to read about, such as divorces, pregnancies, and fights. Consequently, celebrity news tends to follow certain celebrities more than others further limiting topic diversity in celebrity news. To address this issue, we evaluated several sources to make sure we obtain a diversified pool of celebrities and topics.

Using this approach, we collected a total of 500 news articles, with an even distribution for fake

and legitimate news. The corpus statistics, including class distribution and word/sentence statistics, are shown in Table 1. Table 3 shows a example excerpt of a celebrity fake/legitimate news pairing in the dataset. Throughout the rest of the paper, we refer to this web dataset as Celebrity.

4 Linguistic Features

To build the fake news detection models, we start by extracting several sets of linguistic features:

Ngrams. We extract unigrams and bigrams derived from the bag of words representation of each news article. To account for occasional differences in content length, these features are encoded as tf-idf values.

Punctuation. Previous work on fake news detection (Rubin et al., 2016) as well as on opinion spam (Ott et al., 2011b) suggests that the use of punctuation might be useful to differentiate deceptive from truthful texts. We construct a punctuation feature set consisting of twelve types of punctuation derived from the Linguistic Inquiry and Word Count software (LIWC, Version 1.3.1 2015) (Pennebaker et al., 2015). This includes punctuation characters such as periods, commas, dashes, question marks and exclamation marks.

Psycholinguistic features. We use the LIWC lexicon to extract the proportions of words that fall into psycholinguistic categories. LIWC is based on large lexicons of word categories that represent psycholinguistic processes (e.g., positive emotions, perceptual processes), summary categories (e.g., words per sentence), as well as part-of-speech categories (e.g., articles, verbs). Previous work on verbal deception detection showed that LIWC is a valuable tool for the deception detection in various contexts (e.g., genuine and fake hotel reviews, (Ott et al., 2011b; Ott et al., 2013); prisoners' lies (Bond and Lee, 2005)). In our work, we cluster the single LIWC categories into the following feature sets: summary categories (e.g., analytical thinking, emotional tone), linguistic processes (e.g., function words, pronouns), and psychological processes (e.g., affective processes, social processes). We also test a combined feature set of all the LIWC categories (including punctuation).[3]

Readability. We also extract features that indicate text understandability. These include content features such as the number of characters, complex words, long words, number of syllables, word types, and number of paragraphs, among others content features. We also calculate several readability metrics, including the Flesch-Kincaid, Flesch Reading Ease, Gunning Fog, and the Automatic Readability Index (ARI).

Syntax. Finally, we extract a set of features derived from production rules based on context free grammars (CFG) trees using the Stanford Parser (Klein and Manning, 2003). The CFG derived features consist of all the lexicalized production rules (rules including child nodes) combined with their parent and grandparent node, e.g., *NNˆNP→commission (in this example NN –a noun– is the grandparent node, NP –noun phrase– the parent node, and "commissions" the child node). CFG-based features have been previously shown to be useful for linguistic deception detection (Feng et al., 2012). Features in this set are also encoded as tf-idf values.

5 Automatic Fake News Detection

We conduct several experiments with different combinations of feature sets to explore their predictive separately and jointly. We use a linear SVM classifier and conduct our evaluations using five-fold cross-validation, with accuracy, precision, recall, and F-score as performance metrics. We use the machine learning algorithms implementation available in the caret (Kuhn et al., 2016) and e1071 packages (Meyer et al., 2015) with their default parameters.

Tables 4 and 5 show the results obtained for the different feature sets and the two datasets. Since our datasets contain an even distribution between fake and real news items, we use a random baseline of 50% as reference value. As seen in the tables most of the classifiers obtain performances well above the baseline, which indicates that the task of fake news detection can be effectively addressed using linguistic

[3]The feature sets linguistic processes and punctuation correspond to the 'grammar' and punctuation feature set, respectively, in (Rubin et al., 2016)

features. For the FakeNewsAMT dataset, the best performing classifiers are the ones that rely on stylistic features (i.e., Punctuation and Readability), followed by the ones build using psycholinguistic features drawn from the LIWC lexicon. The classifiers build with the Celebrity dataset show the best performance when using the LIWC features, followed by the ngrams and syntactic features (CFG). Overall, our results suggest that fake news differ from real news mainly in aspects such as writing style (punctuation, readability, syntactic structure) and aspects related to writer's internal processes (LIWC features). Regarding the differences in performance between the two datasets, we believe that they can be attributed to the domain in which the news are generated. For instance, to spot fake news in more serious topics such as technology or education we might need to pay more attention to linguistic aspects of writing whereas to spot fake news in the celebrity domain we might need to focus on writing differences related to people's feelings and perceptions.

Finally, our results show that when using all the features on the two datasets we achieve the best accuracies, with 0.74 and 0.76 respectively. These results suggest that an integrated use of linguistic, syntactic and semantic features is useful to discriminate between real and fake news content.

Features (# features)	Acc.	$F1_{Legit.}$	$F1_{Fake}$
Punctuation (12)	0.71	0.69	0.72
LIWC-Summ (7)	0.61	0.58	0.64
LIWC-LingProc. (21)	0.67	0.66	0.66
LIWC-PsyProc. (40)	0.56	0.56	0.55
LIWC (80)	0.70	0.70	0.70
Readability (26)	0.78	0.77	0.79
Ngrams (634)	0.62	0.62	0.62
CFG (1377)	0.65	0.64	0.65
All Features (2140)	0.74	0.74	0.74

Table 4: Classification results for the FakeNewsAMT dataset collected via crowdsourcing.

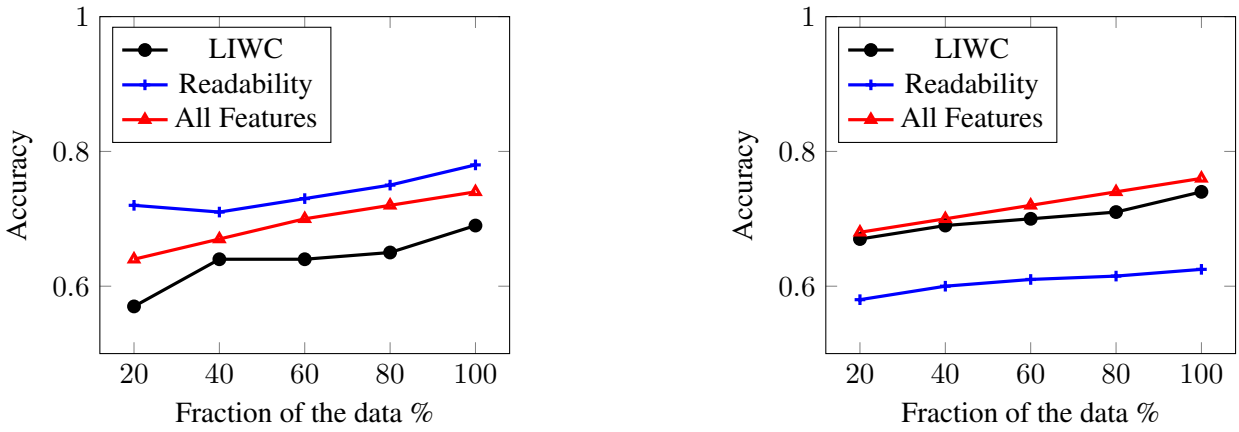
Features (# features)	Acc.	$F1_{Legit.}$	$F1_{Fake}$
Punctuation (12)	0.69	0.69	0.69
LIWC-Summ. (7)	0.67	0.66	0.69
LIWC-LingProc (21)	0.72	0.72	0.71
LIWC-PsyProc (40)	0.67	0.68	0.66
LIWC (80)	0.74	0.74	0.74
Readability (28)	0.62	0.61	0.63
Ngrams (1317)	0.71	0.72	0.71
CFG (2599)	0.72	0.72	0.72
All Features (4048)	0.76	0.77	0.76

Table 5: Classification results for the Celebrity news dataset.

Learning Curves. Next, seeking to investigate whether larger amounts of training data can improve the identification of fake content, we analyzed the learning trend of our best classifiers. Thus, we plot the learning curves of the LIWC features, readability features, and the combination of all features sets using incremental amounts of data as shown in Figures 1a and 1b. Overall, the learning trend in both datasets shows steady improvement, thus suggesting that larger quantities of training data could improve the classification performance.

Cross-domain Analyses. We also explore the applicability of our methods across domains, using the two best feature sets identified during our previous experiments, (*Readability* and *LIWC*), as well as the classifier relying on all the features (*All Features*).

Table 6 shows the results obtained in cross-domain experiments, where we train our models using the



(a) FakeNewsAMT dataset

(b) Celebrity dataset

Figure 1: Learning curves using incremental fraction of the data and three feature sets

Training	Testing	Feature set	Accuracy
FakeNewsAMT	Celebrity	LIWC	0.48
		Readability	0.52
		All Features	0.50
Celebrity	FakeNewsAMT	Complete LIWC	0.60
		Readability	0.65
		All Features	0.64

Table 6: Cross-domain analysis for best performing feature sets.

FakeNewsAMT dataset and test on the Celebrity dataset. Perhaps not surprisingly, there is a significant loss in accuracy as compared to the within-domain results shown in Tables 4 and 5.

Possible explanations for the drop in performance might be (1) that the linguistic properties of deception in one domain are structurally different from those of deception in a second domain, and (2) that the feature sets applied for the cross-domain evaluation, in particular the readability feature set (accuracy = 0.62), were not performing well in the respective domain in the first place. To test this idea, we also applied cross-domain evaluation where we trained the classifiers using the celebrity domain (Celebrity) and tested in the other domain (FakeNewsAMT).

This time, the readability feature set classifier of the Celebrity data yielded an accuracy of 0.65 on the FakeNewsAMT data (compared to the original 0.78) and similarly, the *LIWC* classifier resulted in an accuracy of 0.60 (compared to 0.70). Likewise, the performance using all features dropped from 0.74 and 0.76 to 0.50 and 0.64 for the FakeNewsAMT and celebrity datasets, respectively. Overall, these findings hint at the important role of domain in the fake news detection.

As an additional experiment, we assess the cross-domain classification performance for the six news domains in the FakeNewsAMT dataset. We do this by training on five of the six domains in the dataset, and testing on the remaining one. Table 7 shows the results obtained in these experiments. The politics, education, and technology domains appear to be rather robust against classifiers trained on other domains. The technology and politics domains, moreover, are both classified with a high accuracy of 0.90 and 0.91 with the *Readability* feature set, which may suggest that fake and legitimate news in each of these three domains might be structurally similar to the fake and legitimate content in the other five domains. By contrast, domains such as sports, business and entertainment are less generalizable and might therefore be more domain-dependent. Although further research is needed to consolidate these findings, a possible explanation could be the rather unique content and style of these domains.

Test Domain	Readability	LIWC	All features
Technology	0.90	0.62	0.80
Education	0.84	0.68	0.84
Business	0.53	0.76	0.85
Sports	0.51	0.73	0.81
Politics	0.91	0.73	0.75
Entertainment	0.61	0.70	0.75

Table 7: Cross-domain classification accuracy for the complete LIWC and readability feature sets. Training data consists of all but the test domains in the FakeNewsAMT dataset.

	Agreement	Kappa
FakeNewsAMT	70%	0.38
Celebrity	73%	0.45

Table 8: Agreement among two human annotators on the FakeNewsAMT and the Celebrity datasets.

6 Human Performance

Fake news detection is a challenging task for humans as readers frequently find themselves sharing fake news content or being lured by clickbait headlines. Seeking to identify a human baseline for the fake news detection task, we conducted a study to evaluate the human ability to spot fake news on the two developed datasets. We created an annotation interface that shows an annotator either a fake or a legitimate news article, and asks them to judge its credibility. We asked annotators to select a label of "Fake" or "Legitimate" according to their own perceptions upon reading the news item. We also asked them to indicate whether or not they have read or heard about the presented news item in the past; overall, the annotators read less than 5% of the news before, which we considered a negligible fraction.

Two annotators labeled the news in each dataset. In both cases, the news articles were presented in a random order to avoid annotation bias. Annotators evaluated 480 and 200 news for the FakeNewsAMT and Celebrity datasets respectively. Annotators were not offered a monetary reward and we consider their judgments to be honest as they participated voluntarily in this experiment.

Table 8 shows the observed agreement and Kappa statistics for each dataset. Resulting Kappa values show moderate agreement values with slightly lower Kappa for the FakeNewAMT dataset.

In addition, we evaluate the performance of the automatic fake news classifiers against the human capability to spot fake news. Thus, we compare the accuracy of our system to that of human annotators. Table 9 summarizes the accuracies obtained by the human annotators and our system on the two fake news datasets. The findings indicate that humans are better at detecting fake content in the Celebrity domain than in the other fake news domain. Notably, our system outperforms humans while detecting fake news in more serious and diverse news sources.

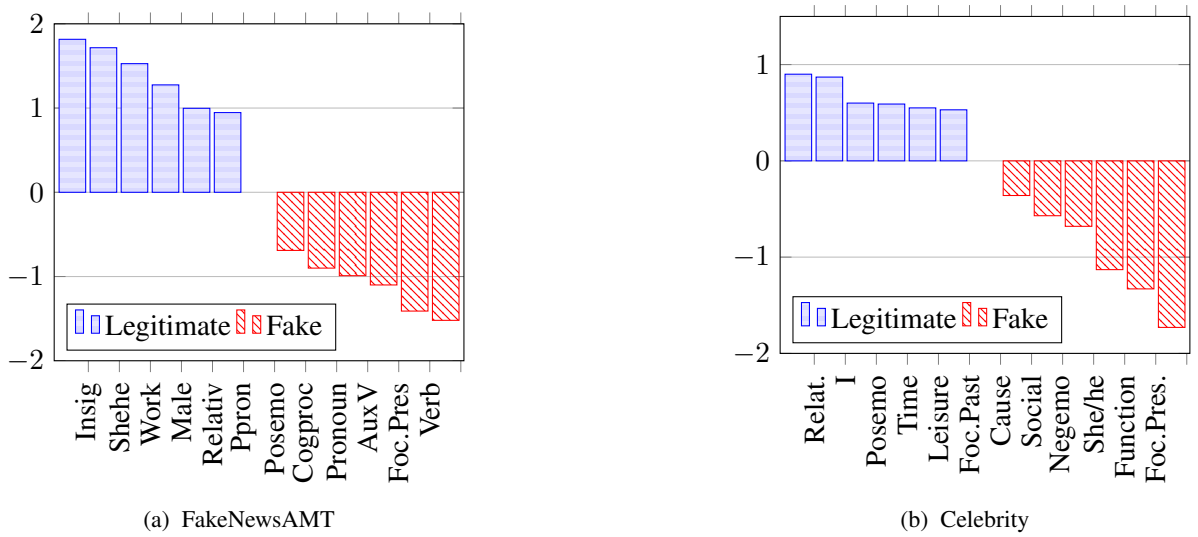
7 Further Insights

Our experiments suggest important differences in fake news content as compared to legitimate news content. Particularly, we observe that classifiers relying on the semantic information encoded in the LIWC lexicon show consistently good performance across domains. To gain further insights into the semantic classes that are associated with fake and legitimate content, we evaluate which classes show significant differences between the two groups of news. To compare both types of content, we subtract the average percentage of words in each LIWC category in the fake news from its corresponding values in the legitimate news set. Therefore, a positive result indicates an association between a LIWC class and legitimate content, and a negative result indicates an association between a LIWC class and fake content. Results for the FakeNewsAMT and Celebrity datasets are shown in Figures 2a and 2b respectively. All the differences shown in the graphs are statically significant (one-tailed t-test, $p < 0.05$).

Figure 2a indicates that the language used to report legitimate content in the FakeNewsAMT dataset

	FakeNewsAMT	Celebrity
A1	0.71	0.80
A2	0.70	0.77
Sys	0.74	0.76

Table 9: Performance of two annotators (A1, A2) and the developed automatic system (Sys) on the fake news datasets



(a) FakeNewsAMT

(b) Celebrity

Figure 2: Language differences in each dataset using the LIWC word categories

often includes words associated with cognitive processes such as insight and differentiation. In addition, legitimate content includes more function words (e.g., pronouns such as he, she), negations, and expressions of relativity. On the other hand, language used when reporting fake content uses more social and positive words, expresses more certainty and focuses on present and future actions. Moreover, the authors of fake news use more adverbs, verbs, and punctuation characters than the authors of legitimate news.

Likewise, the results in Figure 2b show noticeable differences among legitimate and fake content on the celebrity domain. Specifically, legitimate news in tabloid and entertainment magazines seem to use more first person pronouns, talk about time (Relativity,Time, FocusPast), and use positive emotion words (posemo), which interestingly were also found as markers of truth-tellers in previous work on deception detection (Pérez-Rosas and Mihalcea, 2014). On the other hand, fake content in this domain has a predominant use of second person pronouns (he, she), negative emotion words (negemo) and focus on the present (Foc.Pres).

8 Conclusions

With an increasing focus of academic researchers and practitioners alike on the detection of online misinformation, the current investigation allows for two key conclusions.

First, computational linguistics can aide in the process of identifying fake news in an automated manner well above the chance level. The proposed linguistics-driven approach suggests that to differentiate between fake and genuine content it is worthwhile to look at the lexical, syntactic and semantic level of a news item in question. The developed system's performance is comparable to that of humans in this task, with an accuracy up to 76%. Nevertheless, while linguistics features seem promising, we argue that future efforts on misinformation detection should not be limited to these and should also include *meta* features (e.g., number of links to and from an article, comments on the article), features from different modalities (e.g., the visual makeup of a website using computer vision approaches), and embrace the increasing potential of computational approaches to fact verification (Thorne et al., 2018). Thus,

future work might want to explore how hybrid decision models consisting of both fact verification and data-driven machine learning judgments can be integrated.

Second, we showed that it is possible to build resources for the fake news detection task by combining manual and crowsourced annotation approaches. Our paper presented the development of two datasets using these strategies and showed that they exhibit linguistic properties related to deceptive content. Furthermore, different from other available fake news datasets, our dataset consists of actual news excerpts, instead of short statements containing fake news information.

Finally, with the current investigation and dataset, we encourage the research community and practitioners to take on the challenge of tackling misinformation. The datasets introduced in this paper are publicly available at `http://lit.eecs.umich.edu/downloads.html`.

Acknowledgments

This material is based in part upon work supported by the Michigan Institute for Data Science and by the National Science Foundation (grant #1344257). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the Michigan Institute for Data Science or the National Science Foundation.

References

Gary D Bond and Adrienne Y Lee. 2005. Language of lies in prison: Linguistic classification of prisoners' truthful and deceptive natural language. *Applied Cognitive Psychology*, 19(3):313–329.

Yimin Chen, Niall J Conroy, and Victoria L Rubin. 2015. News in an online world: The need for an "automatic crap detector". *Proceedings of the Association for Information Science and Technology*, 52(1):1–4.

Niall J Conroy, Victoria L Rubin, and Yimin Chen. 2015. Automatic deception detection: Methods for finding fake news. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 171–175. Association for Computational Linguistics.

Eileen Fitzpatrick, Joan Bachenko, and Tommaso Fornaciari. 2015. Automatic detection of verbal deception. *Synthesis Lectures on Human Language Technologies*, 8(3):1–119.

Aniko Hannak, Drew Margolin, Brian Keegan, and Ingmar Weber. 2014. Get Back! You Don't Know Me Like That: The Social Mediation of Fact Checking Interventions in Twitter Conversations. In *Proceedings of the 8th International AAAI Conference on Weblogs and Social Media (ICWSM'14)*, Ann Arbor, MI, June.

Molly E Ireland and James W Pennebaker. 2010. Language style matching in writing: synchrony in essays, correspondence, and poetry. *Journal of personality and social psychology*, 99(3):549.

Gottfried Jeffrey and Shearer Elisa. 2016. News use across social media platforms 2016. In *Pew Research Center Reports*.

Zhiwei Jin, Juan Cao, Yu-Gang Jiang, and Yongdong Zhang. 2014. News credibility evaluation on microblog with a hierarchical propagation model. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 230–239. IEEE.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA. Association for Computational Linguistics.

Max Kuhn, Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang, and Can Candan., 2016. *caret: Classification and Regression Training*. R package version 6.0-70.

David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch, 2015. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. R package version 1.6-7.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: Investigations in automatic humor recognition. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 531–538, Stroudsburg, PA, USA. Association for Computational Linguistics.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey Hancock. 2011a. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 309–319, Stroudsburg, PA, USA. Association for Computational Linguistics.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. 2011b. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 309–319, Portland, Oregon, USA, June. Association for Computational Linguistics.

Myle Ott, Claire Cardie, and Jeffrey T Hancock. 2013. Negative deceptive opinion spam. In *HLT-NAACL*, pages 497–501.

James W Pennebaker, Ryan L Boyd, Kayla Jordan, and Kate Blackburn. 2015. The development and psychometric properties of liwc2015. Technical report.

Verónica Pérez-Rosas and Rada Mihalcea. 2014. Cross-cultural deception detection. In *ACL (2)*, pages 440–445.

Verónica Pérez-Rosas and Rada Mihalcea. 2015. Experiments in open domain deception detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1120–1125, Lisbon, Portugal, September. Association for Computational Linguistics.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2016. Credibility assessment of textual claims on the web. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 2173–2178, New York, NY, USA. ACM.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2017. A stylometric inquiry into hyperpartisan and fake news. *CoRR*, abs/1702.05638.

Victoria L. Rubin, Yimin Chen, and Niall J. Conroy. 2015. Deception detection for news: Three types of fakes. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4.

Victoria L Rubin, Niall J Conroy, Yimin Chen, and Sarah Cornwell. 2016. Fake news or truth? using satirical cues to detect potentially misleading news. In *Proceedings of NAACL-HLT*, pages 7–17.

Wafa Shafqat, Seunghun Lee, Sehrish Malik, and Hyun-chul Kim. 2016. The language of deceivers: Linguistic features of crowdfunding scams. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 99–100. International World Wide Web Conferences Steering Committee.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*.

C. Toma and J. Hancock. 2010. Reading between the lines: linguistic cues to deception in online dating profiles. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, CSCW '10, pages 5–8, New York, NY, USA. ACM.

Byron C Wallace. 2015. Computational irony: A survey and new perspectives. *Artificial Intelligence Review*, 43(4):467–483.

William Yang Wang. 2017. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 422–426. Association for Computational Linguistics.

D. Warkentin, M. Woodworth, J. Hancock, and N. Cormier. 2010. Warrants and deception in computer mediated communication. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pages 9–12. ACM.

Linfeng Zhang and Yong Guan. 2008. Detecting click fraud in pay-per-click streams of online advertising networks. In *Distributed Computing Systems, 2008. ICDCS'08. The 28th International Conference on*, pages 77–84. IEEE.

All-in-one: Multi-task Learning for Rumour Verification

Elena Kochkina[1,2], **Maria Liakata**[1,2], **Arkaitz Zubiaga**[1]
[1] University of Warwick, Coventry, United Kingdom
[2] Alan Turing Institute, London, United Kingdom
{E.Kochkina, M.Liakata, A.Zubiaga}@warwick.ac.uk

Abstract

Automatic resolution of rumours is a challenging task that can be broken down into smaller components that make up a pipeline, including rumour detection, rumour tracking and stance classification, leading to the final outcome of determining the veracity of a rumour. In previous work, these steps in the process of rumour verification have been developed as separate components where the output of one feeds into the next. We propose a multi-task learning approach that allows joint training of the main and auxiliary tasks, improving the performance of rumour verification. We examine the connection between the dataset properties and the outcomes of the multi-task learning models used.

Title and Abstract in Russian

Три в одном: Многозадачное Обучение для Верификации Слухов

Автоматическая верификация слухов в Интернете - это сложная задача, которая может быть разбита на подзадачи, включающие в себя обнаружение слухов, отслеживание слухов, классификацию отношения пользователей к правдивости этих слухов и, в итоге, разрешение основной задачи - определение достоверности слуха. В предыдущих исследованиях модели для решения каждой из подзадач разрабатывались как отдельные компоненты, где вывод модели для задачи предыдущего этапа являлся вводдом для модели задачи следующего этапа. Мы предлагаем использование многозадачного обучения для совместного обучения основной и вспомогательных задач в одной модели, что позволяет улучшить качество модели верификации слухов. Также мы рассматриваем связь между свойствами набора данных и результатами использования моделей с многозадачным обучением.

1 Introduction

Social media have gained popularity as platforms that enable users to follow events and breaking news (Hu et al., 2012). However, not all information that spreads on social media during such events is accurate. The serious harm that inaccurate information can cause to society in critical situations (Lewandowsky et al., 2012) has led to an increased interest within the scientific community to develop tools to verify information from social media (Shu et al., 2017). Likewise, social media platforms themselves, such as Facebook,[1] are investing significant effort in mitigating the problems caused by misinformation. One of the features that characterises social media is the rapid emergence and spread of new information. This leads to the circulation of rumours, claims that are unverified at the time of posting, and for which eventually evidence proving their true or false nature may come out (DiFonzo and Bordia, 2007). Therefore, a rumour resolution system needs to go through a set of steps, from detecting that a new circulating claim is a rumour, to the ultimate step of determining its veracity value.

[1] https://www.recode.net/2017/2/16/14632726/mark-zuckerberg-facebook-manifesto-fake-news-terrorism

Proceedings of the 27th International Conference on Computational Linguistics, pages 3402–3413
Santa Fe, New Mexico, USA, August 20-26, 2018.

The rumour resolution process has been defined as a pipeline involving four sub-tasks (Zubiaga et al., 2018a) (see Figure 1): (1) rumour detection, determining whether a claim is worth verifying rather than the expression of an opinion; (2) rumour tracking, collecting sources and opinions on a rumour as it unfolds; (3) stance classification, determining the attitude of the sources or users towards the truthfulness of the rumour, and (4) rumour verification, as the ultimate step where the veracity value of the rumour is predicted. These steps can be performed at different times in the life-cycle of a rumour, making this a time-sensitive process. Ideally, rumours can be resolved as either true or false. However, they can also remain unverified when there is no sufficient evidence to determine their veracity (Caplow, 1947).

A recent body of work studies each of these four tasks separately (Lukasik et al., 2016; Liu et al., 2016; Enayet and El-Beltagy, 2017). However, the way these subtasks interact and their integration into a complete rumour resolution system is yet to be explored. In this work we assume that veracity classification is the crucial component of a rumour resolution system, as the final output that determines, given information collected about a rumour at a certain point in time, if the rumour is true, false, or remains unverified. We express the rumour resolution process as a multi-task problem that needs to address a number of challenges, where the veracity classification task is the main task and the rest of the components are auxiliary tasks that can be leveraged to boost the performance of the veracity classifier.

We propose to achieve this setting by using a multi-task learning approach. Multi-task learning (Caruana, 1998) refers to the joint training of multiple tasks, which has gained popularity recently for a range of tasks in Machine Learning and Natural Language Processing (Collobert and Weston, 2008) and has been applied in a number of different tasks and machine learning architectures. Its effectiveness is mainly attributed to learning shared representations of closely related tasks, such that two complementary tasks can give each other 'hints'.

We propose joint learning of the tasks in the verification pipeline, that will allow us to leverage relations between the tasks. We assess the effectiveness of using a multi-task learning approach for the rumour resolution process in four different scenarios: (1) performing single task learning to perform veracity classification, (2) performing multi-task learning that combines stance and veracity classification with the aim of boosting performance of the latter, (3) performing multi-task learning that combines rumour detection and verification, comparing the improvement brought by each of the two auxiliary tasks, and (4) performing multi-task learning that combines rumour detection, stance classification and veracity prediction to improve the performance for veracity. We employ a deep learning architecture that views each subtask as having a sequential nature. We compare our results with a state-of-the-art veracity classification approach introduced by Enayet and El-Beltagy (2017).

While a lot of work reports positive outcomes of the application of multi-task learning to various NLP tasks (Collobert and Weston, 2008; Aguilar et al., 2017; Lan et al., 2017), there are also studies showing that this is not always the case (Alonso and Plank, 2017; Bingel and Søgaard, 2017). Alonso and Plank (2017) were the first to demonstrate that multi-task learning brings benefits only for some combinations of main and auxiliary tasks. They also investigate the relationship between the multi-task learning outcome and the properties of the dataset. We perform a similar analysis to examine the link between the properties of our rumour datasets and the results of the multi-task learning approach used.

Our results show that a multi-task learning scenario that leverages all three subtasks, where veracity classification is the main task and stance classification and rumour detection are auxiliary tasks, leads to substantial improvements over a standard, single task veracity classification system, as well as a majority baseline and a state-of-the-art system. The combination of all three subtasks also outperforms the multi-task learning scenario where only two of the subtasks are combined.

2 Related work

2.1 Rumour classification system

Rumour classification is a complex task that can be represented as a pipeline of sub-tasks. While there are different possibilities for structuring this pipeline, here we adopt the architecture of a rumour classification system proposed in Zubiaga et al. (2018a) (see Figure 1). Thus, a rumour classification system is expressed as a sequence of subtasks, namely rumour detection, rumour tracking, rumour stance classi-

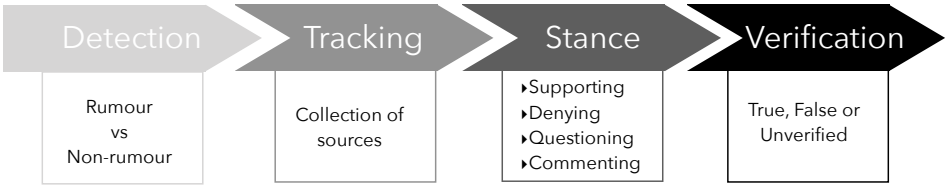


Figure 1: Rumour resolution pipeline.

fication leading to rumour verification. In this work we consider three of these tasks: rumour detection, stance and verification, focusing on the evaluation of the latter. Rumour tracking does not involve classification but rather consists of collecting tweets following up on a rumour in the form of replies.

A recent survey (Zubiaga et al., 2018a) defines rumour detection as the task which distinguishes rumours (unverified pieces of information) from non-rumours (all other circulating information). Here we adhere to this definition. Therefore, once rumour detection is performed, the information classified as a rumour is then fed into the the stance and veracity classification components of a system to ultimately determine the veracity of the rumour.

While we are not aware of previous work tackling the entire rumour classification pipeline, there has been a body of work tackling each of the tasks individually.

Stance classification. An increasing body of work has focused on the stance classification component, i.e. determining whether different posts associated with a rumour support it, deny it, query it, or just comment on it. The stance expressed by users towards a particular rumour can be indicative of the veracity of a rumour, furthermore, it has been shown that rumours attracting higher levels of skepticism in the form of denials and questioning responses are more likely to be proven false later (Mendoza et al., 2010; Procter et al., 2013; Derczynski et al., 2014). Previous work on stance classification (Lukasik et al., 2016; Kochkina et al., 2017; Zubiaga et al., 2017; Zubiaga et al., 2018b) has explored the use of sequential classifiers, treating the task as one that evolves over time; it has been shown that these sequential classifiers substantially outperform standard, non-sequential classifiers.

Rumour detection. Work on rumour detection is more scarce. One of the first approaches was introduced by Zhao et al. (2015), who built a rule-based approach to identify skepticism (e.g. is this true?) and therefore determine that the associated information is a rumour. The limitations of this approach consist in having to wait for responses to arrive, as well as lack of generalisability due to manually-defined rules. Zubiaga et al (2017) proposed a sequential approach to leverage context from earlier posts during an event. The sequential approach achieved significant improvements, especially in terms of recall, where the rule-based approach proved limited.

Rumour verification. Common approaches to rumour verification involve first the collection of corpora of resolved rumours from rumour debunking websites such as snopes.com, emergent.com, politifact.com. Wang (2017) created a dataset based on claims from politifact.com that are annotated for degrees of truthfulness. To resolve these they propose a hybrid convolutional neural network that integrates metadata with text. Twitter is a popular platform to study rumours, while often seeds and annotations for rumours are still taken from rumour debunking websites. Giasemidis et al. (2016) collected a dataset of 72 rumours from Twitter. This work measured trustworthiness of a claim at varying time windows. Boididou et al. (2014; 2017) have focused on tweets that are related to fake images, bringing in the multimedia component into the verification process.

Sequence classification. Works that consider tasks of rumour detection and verification (within their own definition of the tasks, not necessarily aligning with the one used here) also highlighted the importance of a sequential time-sensitive approach when dealing with rumours (Ma et al., 2016; Kwon et al., 2017; Chen et al., 2017). As previous work in the literature has highlighted the importance of sequential classifiers in rumour detection we have decided to use an LSTM based architecture for our experiments.

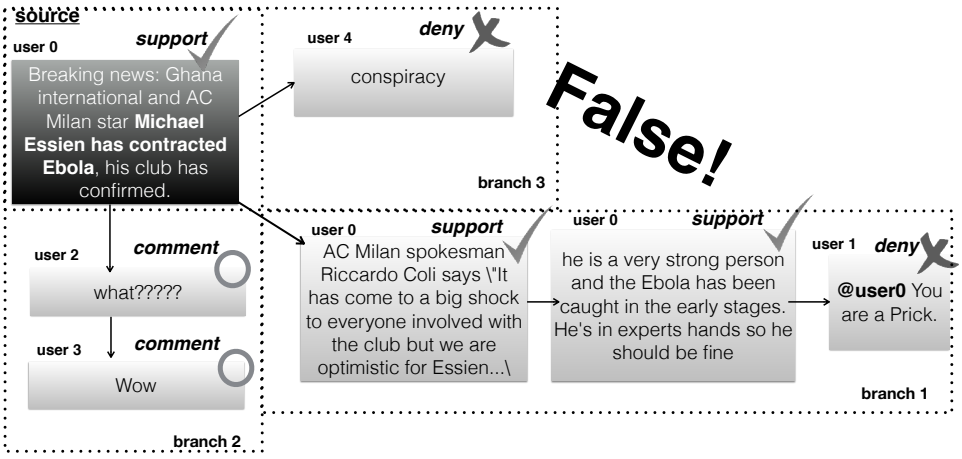


Figure 2: Example of a conversation with 3 branches from the RumourEval dataset.

2.2 Multi-task learning

Multi-task learning refers to the joint learning of several related tasks with a shared representation. In recent years, a multi-task learning approach has been successfully applied in combination with neural networks for a variety of NLP tasks (Collobert and Weston, 2008). In this work we use the most common approach to multi-task learning, namely hard parameter sharing, meaning that different tasks are using the same hidden layer(s). The effectiveness of a multi-task learning approach is attributed to: effectively increasing the size of the training set by using additional datasets for related tasks and regularisation, as the model has to learn a shared representation for multiple tasks there is less risk of overfitting on one of them. In multi-task learning auxiliary tasks can be used to direct the main task to use/learn the features that it otherwise would have ignored or failed to identify due to complex relations between the tasks and features. For example, multi-task learning is particularly useful when potentially helpful features are not used as such for the main task, but become labels in an auxiliary task. This use case is very relevant to this work; stance classification could be used as a feature in a system for veracity classification, as has indeed been the case in previous studies, which have shown a relationship between the two tasks. Zhao et al. (2015) and Enayet and El-Beltagy (2017) have created successful models using this premise. However these studies assume access to stance and veracity labels for the same data, which does not apply in our case as we do not have stance labels for all of the threads in the dataset (see section 3).

3 Data

We use two publicly available datasets of rumours: PHEME (Zubiaga et al., 2016; Zubiaga et al., 2017) and RumourEval (Derczynski et al., 2017) that contain different levels of annotation for the tasks of rumour detection, rumour stance and veracity classification. Both datasets contain Twitter conversation threads associated with different newsworthy events including the Ferguson unrest, the shooting at Charlie Hebdo, the shooting in Ottawa, the hostage situation in Sydney and the crash of a Germanwings plane. Figure 2 shows an example of a conversation discussing a rumour about Michael Essien having contracted Ebola. The conversation consists of a source tweet conveying a rumour and a tree of responses, expressing their opinion towards the claim contained in the source tweet. The veracity label of the rumour is false, while each of the responses could be tagged as either supporting, denying, questioning or commenting on the rumour. Conversations can be decomposed into branches, such that a branch is a linear sequence of tweets starting from a leaf node of the conversation tree and going through its parent nodes up to the source tweet. The conversation on the Figure 2 can be decomposed into three branches.

	Threads	Branches	Tweets	True	False	Unverified	S	D	Q	C
Development	25	215	281	10	12	3	69	11	28	173
Testing	28	772	1049	8	12	8	94	71	106	778
Training	272	3030	4238	127	50	95	841	333	330	2734
Total	325	4017	5568	145	74	106	1004	415	464	3685

Table 1: Number of threads, tweets and class distribution in RumourEval dataset.

3.1 RumourEval

RumourEval is a dataset that was released as part of the SemEval-2017 Task 8 competition (Derczynski et al., 2017). It contains 325 Twitter threads discussing rumours. All conversations in the RumourEval dataset are rumours, therefore this dataset only covers the tasks of rumour stance and veracity classification. It is split into training, testing and development sets. The testing set contains a mix of rumours related to the same events as in the training and development sets, with the addition of two rumours: about Marina Joyce and the health condition of Hillary Clinton. Table 1 shows the number of conversation threads, branches and tweets in each of the sets of the RumourEval dataset, as well as the class distribution for both tasks. In the stance classification task there is a strong class imbalance towards the commenting class, while questioning and denying are minority classes, which holds true for all subsets. For the verification task, the training set contains more true instances than false or unverified, whereas the development and testing sets are more balanced.

3.2 PHEME

We are making available the extended version of the PHEME dataset of rumours and non-rumours related to nine events, each updated with veracity information for each of the rumours[2].

This dataset contains 3 levels of annotation. First, each thread is annotated as either rumour or non-rumour; second, rumours are labeled as either true, false or unverified. And third, a subset (threads used in RumourEval) is annotated for stance classification at the tweet level through crowd-sourcing.

Rumours in this dataset were labeled as true, false and unverified by professional journalists (Zubiaga et al., 2016). While stance can be labeled by non-expert workers as labels can be inferred directly from the text, the rumour verification task is more challenging as it requires analysis of the context, and further understanding of the rumours in order to determine if the underlying story is true, false, or remains unverified. To assess the difficulty of performing the verification task by a non-expert, an author of this paper went through the rumours and annotated them for veracity. The overlap between the non-expert annotator and the journalist was within the range of $60 - 65\%$ on the rumour stories from five largest events.

Table 2 shows the size of each event in the PHEME dataset as well as the label distribution for the tasks of rumour detection and verification. The information about the stance classification task is in table 1. Events differ in size drastically and have different class label proportions. Overall, the PHEME dataset contains fewer rumours than non-rumours, while the majority class for rumours is true.

We perform two types of experiments using different subsets of the PHEME dataset: (1) using the five largest events (see Table 2) and (2) using all nine events. The five largest events create a more balanced dataset as those are major crisis events during which true updates and false information on different aspects of the event were shared and discussed, whereas the 4 smallest events only contain a single rumour story at the core of the event. In both cases, cross-validation experiments are performed by relying on a leave-one-event-out setting, i.e. using all the events except the one left for testing in each case. This set up is more challenging than the one presented in RumourEval, however it is more representative of the real world. We micro-average the performance across events.

[2] https://figshare.com/articles/PHEME_dataset_for_Rumour_Detection_and_Veracity_Classification/6392078

Events	Threads	Tweets	Rumours	Non-rumours	True	False	Unverified
Charlie Hebdo	2,079	38,268	458	1,621	193	116	149
Sydney siege	1,221	23,996	522	699	382	86	54
Ferguson	1,143	24,175	284	859	10	8	266
Ottawa shooting	890	12,284	470	420	329	72	69
Germanwings-crash	469	4,489	238	231	94	111	33
Putin missing	238	835	126	112	0	9	117
Prince Toronto	233	902	229	4	0	222	7
Gurlitt	138	179	61	77	59	0	2
Ebola Essien	14	226	14	0	0	14	0
Total	6,425	105,354	2,402	4,023	1,067	638	697

Table 2: Number of threads, tweets and class distribution in the PHEME dataset.

4 Models

4.1 Sequential approach

As the benefits of using a sequential approach were suggested by previous studies of rumour stance classification and rumour detection tasks (discussed in Section 2), we are following the branchLSTM approach described in Zubiaga et al. (Zubiaga et al., 2018b). We split the conversations into linear branches and use them as training instances that become an input to a model consisting of an LSTM layer followed by several dense ReLU layers and a softmax layer that predicts class probabilities. As the stance classification task is annotated at the tweet level, we use the output from each time step of LSTM, whereas for the tasks of rumour detection and verification, annotated at the thread level, we are only using outputs from the final time steps (this idea is illustrated in Figure 3). To get per-thread predictions, we use majority voting for each of the branches from the thread. The model is trained using categorical cross entropy loss.

4.2 Multi-task learning approach

We leverage the relationship between the tasks from the rumour classification pipeline in a joint multi-task learning setup. Figure 3 illustrates our approach. At the base of it is a sequential approach, as discussed above, represented by a shared LSTM layer (hard parameter sharing), which is followed by a number of task-specific layers. The possible task combinations are shown as dotted lines on Figure 3 that can be present or absent depending on the combination. We perform experiments in three set ups: joint training of (1) stance or (2) rumour detection together with veracity classification, and (3) learning all three tasks together. The cost function in the multi-task models is a sum of losses from each of the tasks. Datasets for each of the three tasks are not equal in size, therefore when the training instance is lacking a label for one of the tasks, its prediction does not add anything to the loss function, as if it had been predicted correctly.

4.3 Baselines

We compare proposed sequential and multi-task learning approaches with several baselines. First of all, majority vote, a strong baseline which results in high accuracy due to the class imbalance in the veracity classification task. Another strong benchmark is NileTMRG (Enayet and El-Beltagy, 2017), the best veracity classification system from SemEval-2017 Task 8. The NileTMRG model is based on a linear SVM that uses a bag-of-words representation of the tweet concatenated with selected features: presence of URL, presence of hashtag and proportion of supporting, denying and querying tweets in the thread. We have made our own implementation NileTMRG* of the NileTMRG model based on their SemEval paper. This model requires stance labels for each of the tweets in the dataset, however these are not available for the PHEME dataset. To obtain the proportion of different stance labels among responses in the thread, we use our own implementation of the stance classification model from Kochkina et al. (2017). On the RumourEval dataset our implementation NileTMRG* achieves better results than the original NileTMRG score (Enayet and El-Beltagy, 2017) (accuracy 0.570 vs reported 0.536) as it utilises

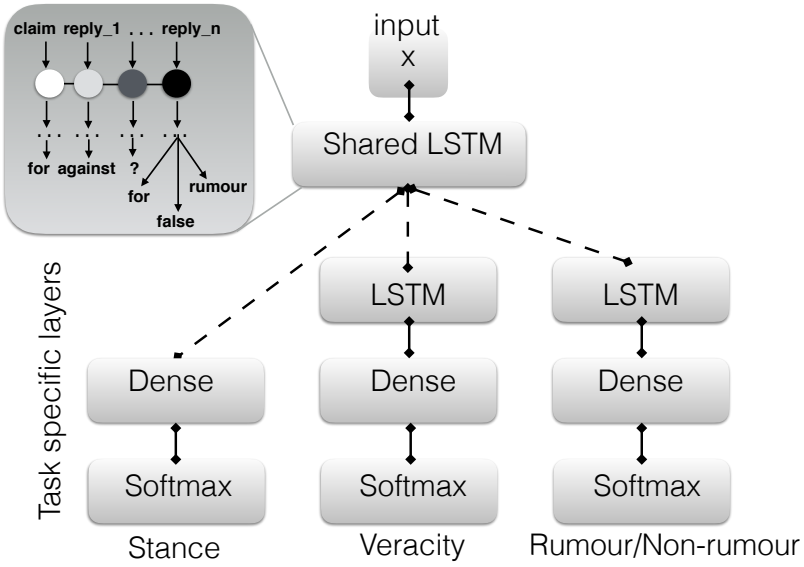


Figure 3: Multi-task learning models. Dotted lines represent that same set up is used for learning one, two or three tasks. Input format is a branch of tweets. Stance classification implies predictions per-tweet, whereas detection and verification tasks only require per-branch output.

a better model for stance classification. The NileTMRG model is a baseline showing the scenario where pipeline tasks are performed sequentially and the outcome of the previous step (stance classification) is an input to the next one (veracity classification). The NileTMRG model has set a precedent in demonstrating that patterns of support are useful indicators of rumour veracity.

4.4 Features

We perform the following pre-processing of the tweets in the datasets: remove nonalphabetic characters, convert all words to lower case and tokenise texts[3]. Once tweet texts are pre-processed, we extract word2vec word embeddings pre-trained on the Google News dataset (300d)[4] (Mikolov et al., 2013) for each word in a tweet and take the average, thus obtaining a tweet representation. This representation was shown to work well for tweets due to their short length (Kochkina et al., 2017).

5 Experiment setup

5.1 Hyperparameters

We determined the optimal set of hyperparameters by testing the performance of our models on the development set for different parameter combinations. We used the Tree of Parzen Estimators (TPE) algorithm[5] to search the parameter space and minimise the loss function expressed as $(1 - macroF)$ for single task, and $(1 - macroF_a)(1 - macroF_b)$ or $(1 - macroF_a)(1 - macroF_b)(1 - macroF_c)$ for multi-task learning of two and three tasks respectively. This loss function gives equal weight to all tasks. The parameter space is defined as follows: the number of dense ReLU layers varies from one to four; the number of LSTM layers is $\{1, 2\}$; the mini-batch size is 32; the number of units in the ReLU layer is $\{300, 400, 500, 600\}$, and in the LSTM layer is $\{100, 200, 300\}$; the strength of the L2 regularisation is $\{10^{-4}, 10^{-3}\}$ and the number of epochs is 50. We performed 30 trials of different parameter combinations optimising for accuracy on the development set in order to choose the best combination. We also use 50% dropout before the output layer. Zero-padding and masks that account

[3]For implementation of all pre-processing routines we use Python 2.7 with the NLTK package

[4]Embeddings were retrieved from `https://code.google.com/archive/p/word2vec/`. Processing was performed using gensim package (Řehůřek and Sojka, 2010).

[5]We used the implementation of the TPE algorithm in the hyperopt package

RumourEval						
	Majority (True)	**NileTMRG***	**branchLSTM**	**MTL2 Veracity+Stance**	**MTL2 Veracity+Detection**	**MTL3**
Macro F	0.148	0.539	0.491	0.558	-	-
Accuracy	0.286	0.570	0.500	0.571	-	-
PHEME 5 events						
	Majority (True)	**NileTMRG***	**branchLSTM**	**MTL2 Veracity+Stance**	**MTL2 Veracity+Detection**	**MTL3**
Macro F	0.226	0.339	0.336	0.376	0.373	0.396
Accuracy	0.511	0.438	0.454	0.441	0.410	0.492
PHEME 9 events						
	Majority (True)	**NileTMRG***	**branchLSTM**	**MTL2 Veracity+Stance**	**MTL2 Veracity+Detection**	**MTL3**
Macro F	0.205	0.297	0.259	0.318	0.345	0.405
Accuracy	0.444	0.360	0.314	0.357	0.397	0.405

Table 3: Comparison of performance of sequential single task approach, multi-task learning approaches with two and three tasks with majority and NileTMRG baselines on veracity classification task.

for the varying lengths of the input branches were used in all our models. Models were implemented[6] using Python 3 and the Keras package.

5.2 Evaluation

The RumourEval dataset was provided with a training/development/testing split. We tune parameters on the development set and then retrain the model on the combined training and development sets before evaluation on the testing set. On the PHEME dataset we perform leave-one-event-out (LOEO) cross-validation, which makes this task set up harder than for RumourEval but closer to the realistic scenario where we want to verify unseen rumours. When choosing parameters, the Charlie Hebdo event was used as the development set as it has balanced labels. We evaluate models using accuracy and macro-averaged F-score as the tasks in the PHEME dataset suffer from a class imbalance.

6 Results and Discussion

The main results of our experiments are presented in Table 3. It shows the results of the multi-task learning models MTL2 with two tasks: Veracity+Stance and Veracity+Detection; MTL3 with three tasks Stance+Veracity+Detection, Majority, NileTMRG* and single task branchLSTM baselines on the main task, veracity classification.

As the datasets contain a significant class imbalance, the majority baseline achieves fairly high accuracy scores. However due to the nature of this task, it is more important for a model to recognize all of the classes, especially false rumours, therefore the macro-averaged F-score is more important for performance evaluation. All models demonstrate improvement over the majority baseline in terms of macro F-score. We observe improvements of multi-task approaches over single task learning in both accuracy and macro F-score, and adding the third task brings further improvement.

MTL2 Veracity+Detection and MTL3 experiments were performed only on the PHEME dataset as the RumourEval dataset consists only of rumours (no rumour detection). On the RumourEval dataset the single task model branchLSTM outperforms the majority baseline, although it does not perform as well as NileTMRG*, while MTL2 shows improvement over both NileTMRG* and branchLSTM. Experiments on the PHEME dataset also show a pattern of increasing scores: MTL2 outperforms single task models and MTL3 outperforms MTL2. Comparing the performance of the MTL2 model using stance as an auxiliary task with the model using rumour detection as an auxiliary task on the PHEME 5 events dataset we observe that both models bring an improvement over the single task branchLSTM baseline.

When using 9 events of the PHEME dataset we observe worse performance than on 5 events. Even though we are adding more training data, the 4 additional events are qualitatively different to the 5 large news-breaking events. Each of these 5 large events contained rumours labeled with all classes as well

[6]https://github.com/kochkinaelena/Multitask4Veracity

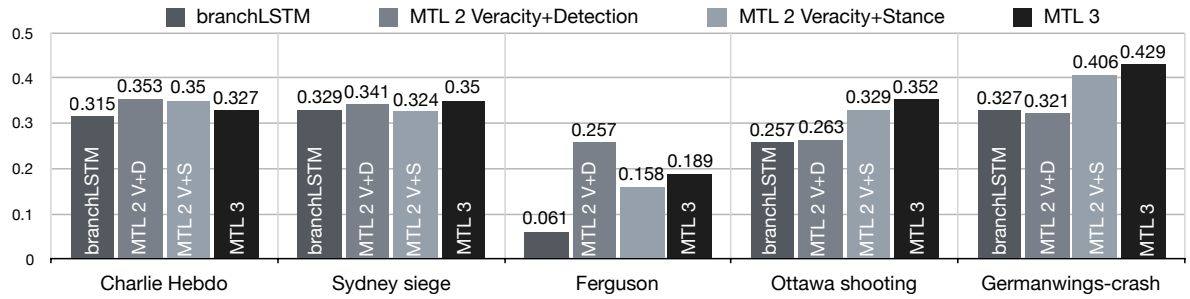


Figure 4: Comparison of macro F-score of sequential single task approach (branchLSTM), multi-task learning approaches with two and three tasks on different events from the PHEME dataset.

MTL3 5 events	**MacroF**	**Accuracy**	**TRUE**	**FALSE**	**Unverified**
Charlie Hebdo	0.327	0.369	0.502	0.227	0.251
Sydney siege	0.350	0.575	0.731	0.153	0.168
Ferguson	0.189	0.338	0.058	0	0.508
Ottawa shooting	0.352	0.645	0.789	0.168	0.100
Germanwings-crash	0.429	0.420	0.538	0.358	0.364

Table 4: Per event and per-class results for multi-task learning approach with 3 tasks on PHEME 5 events.

as non-rumours, whereas the 4 additional events are small and the event itself is a false or unverified rumour. This highlights the difficulty of the rumour verification task in a leave-one-event-out setup and the importance of high quality data. NileTMRG* is a very strong baseline, and while the single task branchLSTM model is competitive when we are using 5 largest events, NileTMRG* is only outperformed by the multi-task learning approach.

6.1 Per-event and per-class results analysis

Here we analyse the performance of proposed models on each of the 5 largest events. Figure 4 illustrates the comparison of macro-averaged F-scores of the proposed models for each of the events. Multi-task learning models outperform single task learning approaches for each event. The Ferguson event is the hardest one for all of the models as it has a different class distribution to all other events (see Table 2).

Table 4 shows per event and per class performance of the multi-task learning model that incorporates all three tasks (MTL3). We have analysed similar performance breakdown tables for other models but omit them here because of space constraints. All models tend to predict the majority class (true) the best. As the Ferguson event is strongly dominated by unverified rumours, it is the only event with high performance on the unverified class. Single task models (NileTMRG*, branchLSTM) are better at identifying false than unverified rumours, whereas multitask models (MTL2, MTL3) are better at identifying unverified than false rumours.

6.2 Analysis of data properties

Alonso and Plank (2017) link the gains/losses in performance from multi-task learning with information-theoretical metrics, properties of the label distributions: entropy (indicating the amount of uncertainty in the distribution) and kurtosis (indicating the skewness of the distribution). They have shown that the multi-task learning setup works best for tasks which have label distributions with lower kurtosis and relatively high entropy. Table 5 shows the kurtosis and entropy properties of the label distribution for each of the events in the dataset, as well as token-type ratio (TTR). Each of the events has very different properties; there is a strong difference in properties between larger events (top) and smaller ones (bottom). Smaller events tend to have more extreme values, which may explain that their addition to the evaluation adds further complexity to the task of rumour verification. In line with findings of Alonso and Plank (2017), in our case both auxiliary tasks have on average lower kurtosis than the main

	Kurtosis			Entropy			TTR		
Events	**S**	**V**	**D**	**S**	**V**	**D**	**S**	**V**	**D**
charliehebdo	-0.73	-1.25	-0.18	0.89	1.08	0.53	0.2	0.11	0.07
ottawashooting	-0.83	0.33	-1.99	1.04	0.82	0.69	0.19	0.11	0.09
germanwings-crash	-1.11	-0.86	-1.99	0.99	0.99	0.69	0.26	0.16	0.13
sydneysiege	-0.79	0.71	-1.91	1.01	0.76	0.68	0.19	0.11	0.07
ferguson	-0.5	17.44	-0.64	0.99	0.28	0.56	0.17	0.09	0.06
ebola-essien	-0.22	-3	-3	1.01	0	0	0.38	0.27	0.27
putinmissing	-1.55	9.08	-1.98	1.12	0.26	0.69	0.49	0.26	0.24
prince-toronto	-1.05	27.75	53.26	1.04	0.14	0.09	0.36	0.18	0.18
gurlitt	-	25.5	-1.95	-	0.14	0.68	-	0.31	0.25

Table 5: Properties of the datasets for each of the events and each of the tasks, where S - Stance, V - Veracity, and D - Detection.

task. The stance classification dataset has on average higher entropy than the rumour detection dataset. Whereas Alonso and Plank (2017) were considering low-level linguistically related tasks as auxiliary tasks, such as part-of-speech tagging, we are working with higher level theme-related tasks. Therefore, it is interesting that we still observe similar trends when looking at the same data properties.

7 Conclusions and Future Work

We have proposed a rumour verification model that achieves improved performance for veracity classification by leveraging task relatedness with auxiliary tasks, specifically rumour detection and stance classification, through a multi-task learning approach. We have compared single task learning approaches with the proposed multi-task learning approaches that combine the verification classifier with the stance and rumour detection classifiers individually, as well as with both the rumour detection system and the stance classifier. Our results show that the joint learning of two tasks from the verification pipeline outperforms a single-learning approach to rumour verification. The combination of all three tasks leads to further performance improvements. We have also investigated the link between the properties of the label distribution in the dataset and the outcomes of our multi-task learning models. Our results support findings from previous research (Alonso and Plank, 2017).

In future work we plan to investigate whether further improvements on main and auxiliary tasks are possible with multi-task learning by: adapting the training schedule to account for different dataset sizes, such that none of the tasks dominates the model; by incorporating the hierarchy between tasks into the model. We would like to investigate the effect of adding extra features (such as user features and interactions) on different tasks and at which stage to incorporate them, in private or shared layers.

Acknowledgements

This work was supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1. Cloud computing resources were kindly provided through a Microsoft Azure for Research Award. Work by Elena Kochkina was partially supported by the Leverhulme Trust through the Bridges Programme and Warwick CDT for Urban Science & Progress under the EPSRC Grant Number EP/L016400/1.

References

Gustavo Aguilar, Suraj Maharjan, Adrian Pastor López Monroy, and Thamar Solorio. 2017. A multi-task approach for named entity recognition in social media data. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 148–153.

Héctor Martínez Alonso and Barbara Plank. 2017. When is multitask learning effective? semantic sequence prediction under varying data conditions. In *15th Conference of the European Chapter of the Association for Computational Linguistics*.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. *arXiv preprint arXiv:1702.08303*.

Christina Boididou, Symeon Papadopoulos, Yiannis Kompatsiaris, Steve Schifferes, and Nic Newman. 2014. Challenges of computational verification in social multimedia. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 743–748. ACM.

Christina Boididou, Symeon Papadopoulos, Lazaros Apostolidis, and Yiannis Kompatsiaris. 2017. Learning to detect misleading content on twitter. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pages 278–286. ACM.

Theodore Caplow. 1947. Rumors in war. *Social Forces*, pages 298–302.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, pages 95–133. Springer.

Tong Chen, Lin Wu, Xue Li, Jun Zhang, Hongzhi Yin, and Yang Wang. 2017. Call attention to rumors: Deep attention based recurrent neural networks for early rumor detection. *arXiv:1704.05973*.

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

Leon Derczynski, Kalina Bontcheva, Michal Lukasik, Thierry Declerck, Arno Scharl, Georgi Georgiev, Petya Osenova, Toms Pariente Lobo, Anna Kolliakou, Robert Stewart, et al. 2014. Pheme: computing veracity: the fourth challenge of big social data. In *Proceedings of ESWC EU Project Networking*.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. Semeval-2017 task 8: Rumoureval: Determining rumour veracity and support for rumours. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 69–76.

Nicholas DiFonzo and Prashant Bordia. 2007. Rumor, gossip and urban legends. *Diogenes*, 54(1):19–35.

Omar Enayet and Samhaa R El-Beltagy. 2017. Niletmrg at semeval-2017 task 8: Determining rumour and veracity support for rumours on twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 470–474.

Georgios Giasemidis, Colin Singleton, Ioannis Agrafiotis, Jason RC Nurse, Alan Pilgrim, Chris Willis, and Danica Vukadinovic Greetham. 2016. Determining the veracity of rumours on twitter. In *International Conference on Social Informatics*, pages 185–205. Springer.

Mengdie Hu, Shixia Liu, Furu Wei, Yingcai Wu, John Stasko, and Kwan-Liu Ma. 2012. Breaking news on twitter. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2751–2754. ACM.

Elena Kochkina, Maria Liakata, and Isabelle Augenstein. 2017. Turing at semeval-2017 task 8: Sequential approach to rumour stance classification with branch-lstm. *In Proceedings of SemEval.ACL.*

Sejeong Kwon, Meeyoung Cha, and Kyomin Jung. 2017. Rumor detection over varying time windows. *PloS one*, 12(1):e0168344.

Man Lan, Jianxiang Wang, Yuanbin Wu, Zheng-Yu Niu, and Haifeng Wang. 2017. Multi-task attention-based neural networks for implicit discourse relationship representation and identification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1299–1308.

Stephan Lewandowsky, Ullrich KH Ecker, Colleen M Seifert, Norbert Schwarz, and John Cook. 2012. Misinformation and its correction: Continued influence and successful debiasing. *Psychological Science in the Public Interest*, 13(3):106–131.

Xiaomo Liu, Quanzhi Li, Armineh Nourbakhsh, Rui Fang, Merine Thomas, Kajsa Anderson, Russ Kociuba, Mark Vedder, Steven Pomerville, Ramdev Wudali, et al. 2016. Reuters tracer: A large scale system of detecting & verifying real-time news events from twitter. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 207–216. ACM.

Michal Lukasik, PK Srijith, Duy Vu, Kalina Bontcheva, Arkaitz Zubiaga, and Trevor Cohn. 2016. Hawkes processes for continuous time sequence classification: an application to rumour stance classification in twitter. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 393–398.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In *IJCAI*, pages 3818–3824.

Marcelo Mendoza, Barbara Poblete, and Carlos Castillo. 2010. Twitter under crisis: Can we trust what we RT? In *1st Workshop on Social Media Analytics*, SOMA'10, pages 71–79.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Rob Procter, Farida Vis, and Alex Voss. 2013. Reading the riots on twitter: methodological innovation for the analysis of big data. *International journal of social research methodology*, 16(3):197–214.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. `http://is.muni.cz/publication/884893/en`.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1):22–36.

William Yang Wang. 2017. Liar, liar pants on fire: A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 422–426.

Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1395–1405. International World Wide Web Conferences Steering Committee.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PloS one*, 11(3):e0150989.

Arkaitz Zubiaga, Maria Liakata, and Rob Procter. 2017. Exploiting context for rumour detection in social media. In *International Conference on Social Informatics*, pages 109–123. Springer.

Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. 2018a. Detection and resolution of rumours in social media: A survey. *ACM Comput. Surv.*, 51(2):32:1–32:36, February.

Arkaitz Zubiaga, Elena Kochkina, Maria Liakata, Rob Procter, Michal Lukasik, Kalina Bontcheva, Trevor Cohn, and Isabelle Augenstein. 2018b. Discourse-aware rumour stance classification in social media using sequential classifiers. *Information Processing & Management*, 54(2):273–290.

Open Information Extraction on Scientific Text: An Evaluation

Paul Groth
Elsevier Labs
1600
John F. Kennedy Blvd.
Suite 1800
Philadelphia, PA
p.groth@elsevier.com

Mike Lauruhn
Elsevier Labs
1600
John F. Kennedy Blvd.
Suite 1800
Philadelphia, PA
m.lauruhn@elsevier.com

Antony Scerri
Elsevier Labs
1600
John F. Kennedy Blvd.
Suite 1800
Philadelphia, PA
a.scerri@elsevier.com

Ron Daniel, Jr.
Elsevier Labs
1600
John F. Kennedy Blvd.
Suite 1800
Philadelphia, PA
r.daniel@elsevier.com

Abstract

Open Information Extraction (OIE) is the task of the unsupervised creation of structured information from text. OIE is often used as a starting point for a number of downstream tasks including knowledge base construction, relation extraction, and question answering. While OIE methods are targeted at being domain independent, they have been evaluated primarily on newspaper, encyclopedic or general web text. In this article, we evaluate the performance of OIE on scientific texts originating from 10 different disciplines. To do so, we use two state-of-the-art OIE systems using a crowd-sourcing approach. We find that OIE systems perform significantly worse on scientific text than encyclopedic text. We also provide an error analysis and suggest areas of work to reduce errors. Our corpus of sentences and judgments are made available.

1 Introduction

The scientific literature is growing at a rapid rate (Bornmann and Mutz, 2015). To make sense of this flood of literature, for example, to extract cancer pathways (Poon et al., 2014) or find geological features (Leveling, 2015), increasingly requires the application of natural language processing. Given the diversity of information and its constant flux, the use of unsupervised or distantly supervised techniques are of interest (Quirk and Poon, 2017). In this paper, we investigate one such unsupervised method, namely, Open Information Extraction (OIE) (Banko et al., 2007). OIE is the task of the unsupervised creation of structured information from text. OIE is often used as a starting point for a number of downstream tasks including knowledge base construction, relation extraction, and question answering (Mausam, 2016).

While OIE has been applied to the scientific literature before (Groth et al., 2016), we have not found a systematic evaluation of OIE as applied to scientific publications. The most recent evaluations of OIE extraction tools (Gashteovski et al., 2017; Schneider et al., 2017) have instead looked at the performance of these tools on traditional NLP information sources (i.e. encyclopedic and news-wire text). Indeed, as (Schneider et al., 2017) noted, there is little work on the evaluation of OIE systems. Thus, the goal of this paper is to evaluate the performance of the state of the art in OIE systems on scientific text.

Specifically, we aim to test two hypotheses:

1. H1: There is no significant difference in the accuracy of OIE systems on scientific vs. general audience content.

2. H2: There is no significant difference in performance of current state-of-the-art OIE systems on scientific and medical content.

Additionally, we seek to gain insight into the value of unsupervised approaches to information extraction and also provide information useful to implementors of these systems. We note that our evaluation differs from existing OIE evaluations in that we use crowd-sourcing annotations instead of expert annotators. This allows for a larger number of annotators to be used. All of our data, annotations and analyses are made openly available.[1]

[1] http://dx.doi.org/10.17632/6m5dyx4b58.2

Proceedings of the 27th International Conference on Computational Linguistics, pages 3414–3423
Santa Fe, New Mexico, USA, August 20-26, 2018.

The rest of the paper is organized as follows. We begin with a discussion of existing evaluation approaches and then describe the OIE systems that we evaluated. We then proceed to describe the datasets used in the evaluation and the annotation process that was employed. This is followed by the results of the evaluation including an error analysis. Finally, we conclude.

2 Existing Evaluation Approaches

OIE systems analyze sentences and emit relations between one predicate and two or more arguments (e.g. Washington :: was :: president). The arguments and predicates are not fixed to a given domain. (Note, that throughout this paper we use the word 'triple" to refer interchangeably to binary relations.) Existing evaluation approaches for OIE systems have primarily taken a ground truth-based approach. Human annotators analyze sentences and determine correct relations to be extracted. Systems are then evaluated with respect to the overlap or similarity of their extractions to the ground truth annotations, allowing the standard metrics of precision and recall to be reported.

This seems sensible but is actually problematic because of different but equivalent representations of the information in an article. For example, consider the sentence "The patient was treated with Emtricitabine, Etravirine, and Darunavir". One possible extraction is:

(The patient :: was treated with :: Emtricitabine, Etravirine, and Darunavir)

Another possible extraction is:

(The patient :: was treated with :: Emtricitabine)
(The patient :: was treated with :: Etravirine)
(The patient :: was treated with :: Darunavir)

Neither of these is wrong, but by choosing one approach or the other a pre-constructed gold set will falsely penalize a system that uses the other approach.

From such evaluations and their own cross dataset evaluation, (Schneider et al., 2017) list the following common errors committed by OIE systems:

- wrong boundaries around the arguments or predicate of a relation;
- generation of redundant relations from the same sentence;
- wrong extractions (e.g. omitting large parts of a sentence as an argument);
- missing extractions - extractions that should have been extracted from a sentence;
- uninformative extractions that omit critical information from a sentence;

In our evaluation, we take a different approach. We do not define ground truth relation extractions from the sentences in advance. Instead, we manually judge the correctness of each extraction after the fact. We feel that this is the crux of the information extraction challenge. Is what is being extracted correct or not? This approach enables us to consider many more relations through the use of a crowd-sourced annotation process. Our evaluation approach is similar to the qualitative analysis performed in (Schneider et al., 2017) and the evaluation performed in (Gashteovski et al., 2017). However, our evaluation is able to use more judges (5 instead of 2) because we apply crowd sourcing.[2] For our labelling instructions, we adapted those used by (Gashteovski et al., 2017) to the crowd sourcing setting.[3]

As previously noted existing evaluations have also only looked at encyclopedic or newspaper corpora. Several systems (e.g. (Banko et al., 2007; Del Corro and Gemulla, 2013)) have looked at text from the web as well, however, as far as we know, none have specifically looked at evaluation for scientific and medical text.

[2](Schneider et al., 2017) also perform a ground truth based evaluation. This evaluation had many more sentences than ours, but used predefined gold extractions from multiple corpora annotated according to different criteria; illustrating the kinds of issues we mention with such an evaluation method.

[3]Labelling instructions are included in the associated dataset. - `http://dx.doi.org/10.17632/6m5dyx4b58.2#file-7edb4b86-c0e6-4169-aea0-4862d39461d3`

3 Systems

We evaluate two OIE systems (i.e. extractors). The first, OpenIE 4 (Mausam, 2016), descends from two popular OIE systems OLLIE (Fader et al., 2011) and Reverb (Fader et al., 2011). We view this as a baseline system. The second was MinIE (Gashteovski et al., 2017), which is reported as performing better than OLLIE, ClauseIE (Del Corro and Gemulla, 2013) and Stanford OIE (Del Corro and Gemulla, 2013). MinIE focuses on the notion of minimization - producing compact extractions from sentences. In our experience using OIE on scientific text, we have found that these systems often produce overly specific extractions that do not provide the redundancy useful for downstream tasks. Hence, we thought this was a useful package to explore.

We note that both OpenIE 4 and MiniIE support relation extractions that go beyond binary tuples, supporting the extraction of n-ary relations. We note that the most recent version of Open IE (version 5) is focused on n-ary relations. For ease of judgement, we focused on binary relations. Additionally, both systems support the detection of negative relations.

In terms of settings, we used the off the shelf settings for OpenIE 4. For MinIE, we used their "safe mode" option, which uses slightly more aggressive minimization than the standard setting. In the recent evaluation of MiniIE, this setting performed roughly on par with the default options (Gashteovski et al., 2017). Driver code showing how we ran each system is available.[4]

4 Datasets

We used two different data sources in our evaluation. The first dataset (WIKI) was the same set of 200 sentences from Wikipedia used in (Gashteovski et al., 2017). These sentences were randomly selected by the creators of the dataset. This choice allows for a rough comparison between our results and theirs.

The second dataset (SCI) was a set of 220 sentences from the scientific literature. We sourced the sentences from the OA-STM corpus.[5] This corpus is derived from the 10 most published in disciplines. It includes 11 articles each from the following domains: agriculture, astronomy, biology, chemistry, computer science, earth science, engineering, materials science, math, and medicine. The article text is made freely available and the corpus provides both an XML and a simple text version of each article.

We randomly selected 2 sentences with more than two words from each paper using the simple text version of the paper. We maintained the id of the source article and the line number for each sentence.

5 Annotation Process

We employed the following annotation process. Each OIE extractor was applied to both datasets with the settings described above. This resulted in the generation of triples for 199 of the 200 WIKI sentences and 206 of the 220 SCI sentences. That is there were some sentences in which no triples were extracted. We discuss later the sentences in which no triples were extracted. In total 2247 triples were extracted.

The sentences and their corresponding triples were then divided. Each task contained 10 sentences and all of their unique corresponding triples from a particular OIE systems. Half of the ten sentences were randomly selected from SCI and the other half were randomly selected from WIKI. Crowd workers were asked to mark whether a triple was correct, namely, did the triple reflect the consequence of the sentence. Examples of correct and incorrect triples were provided. Complete labelling instructions and the presentation of the HITS can be found with the dataset. All triples were labelled by at least 5 workers.

Note, to ensure the every HIT had 10 sentences, some sentences were duplicated. Furthermore, we did not mandate that all workers complete all HITS.

We followed recommended practices for the use of crowd sourcing in linguistics (Erlewine and Kotek, 2016). We used Amazon Mechanical Turk as a means to present the sentences and their corresponding triples to a crowd for annotation. Within Mechanical Turk tasks are called Human Intelligence Tasks (HITs). To begin, we collected a small set of sentences and triples with known correct answers. We did

[4]See directory "Code for applying information extraction tools" in the associated data - `http://dx.doi.org/10.17632/6m5dyx4b58.2#folder-39ce9705-ccf3-4f95-890a-508ce155ece4`

[5]`http://elsevierlabs.github.io/OA-STM-Corpus/`

this by creating a series of internal HITs and loaded them the Mechanical Turk development environment called the Mechanical Turk Sandbox. The HITs were visible to a trusted group of colleagues who were asked to complete the HITs.

Having an internal team of workers attempt HITs provides us with two valuable aspects of the eventual production HITs. First, internal users are able to provide feedback related to usability and clarity of the task. They were asked to read the instructions and let us know if there was anything that was unclear. After taking the HITs, they are able to ask questions about anomalies or confusing situations they encounter and allow us to determine if specific types of HITs are either not appropriate for the task or might need further explanation in the instructions. In addition to the internal users direct feedback, we were also able to use the Mechanical Turk Requester functionality to monitor how long (in minutes and seconds) it took each worker to complete each HIT. This would come into factor how we decided on how much to pay each Worker per HIT after they were made available to the public.

The second significant outcome from the internal annotations was the generation of a set of 'expected' correct triples. Having a this set of annotations is an integral part of two aspects of our crowdsourcing process. First, it allows us to create a qualification HIT. A qualification HIT is a HIT that is made available to the public with the understanding the Workers will be evaluated based on how closely they matched the annotations of the internal annotators. Based upon this, the Workers with the most matches would be invited to work on additional tasks. Second, we are able to add the internal set of triples randomly amongst the other relations we were seeking to have annotated. This allows us to monitor quality of the individual Workers over the course of the project. Note, none of this data was used in the actual evaluation. It was only for the purposes of qualifying Workers.

We are sensitive to issues that other researchers have in regards to Mechanical Turk Workers earning fair payment in exchange for their contributions to the HITs (Fort et al., 2011) . We used the time estimates from our internal annotation to price the task in order to be above US minimum wage. All workers were qualified before being issued tasks. Overall, we employed 10 crowd workers. On average it took 30 minutes for a worker to complete a HIT. In line with (Crump et al., 2013), we monitored for potential non-performance or spam by looking for long response times and consecutive submitted results. We saw no indicators of low quality responses.

6 Judgement Data and Inter-Annotator Agreement

In total, 11262 judgements were obtained after running the annotation process. Every triple had at least 5 judgements from different annotators. All judgement data is made available.[6] The proportion of overall agreement between annotators is 0.76 with a standard deviation of 0.25 on whether a triple is consequence of the given sentence. We also calculated inter-annotator agreement statistics. Using Krippendorff's alpha inter-annotator agreement was 0.44. This calculation was performed over all data and annotators as Krippendorff's alpha is designed to account for missing data and work across more than two annotators. Additionally, Fleiss' Kappa and Scott's pi were calculated pairwise between all annotators where there were overlapping ratings (i.e. raters had rated at least one triple in common). The average Fleiss's Kappa was 0.41 and the average of Scott's pi was 0.37. Using (Artstein and Poesio, 2008) as a guide, we interpret these statistics as suggesting there is moderate agreement between annotators and that agreement is above random chance. This moderate level of agreement is to be expected as the task itself can be difficult and requires judgement from the annotators at the margin.

Table 1 shows examples of triples that were associated with higher disagreement between annotators. One can see for example, in the third example, that annotators might be confused by the use of a pronoun (him). Another example is in the last sentence of the table, where one can see that there might be disagreement on whether the subsequent prepositional phrase behind light microscope analysis should be included as part of the extracted triple.

We take the variability of judgements into account when using this data to compute the performance of the two extraction tools. Hence, to make assessments as to whether a triple correctly reflects the content

[6] 'aggregated_results_anon.csv' in the associated dataset. - http://dx.doi.org/10.17632/6m5dyx4b58.2#file-03de3c93-8a01-4cd2-bfe5-c5bfeaa4f492

SYSTEM	SOURCE	TRIPLE	PAIRWISE AGREEMENT	SENTENCE
MinIE	SCI	additional QUANT_S_1 on scalp face :: were digitized :: QUANT_O_1 anatomical landmarks	40% False	To coregister MEG and sMRI data, three anatomical landmarks (nasion and right and left preauriculars) as well as an additional 150+ points on the scalp and face were digitized for each subject using the Probe Position Identification (PPI) System (Polhemus, Colchester, VT).
OpenIE 4	WIKI	Dawlish :: to 2nd :: place L:in the Western League	40% True	The previous season had seen him lead Dawlish to 2nd place in the Western League , their highest ever league finishing position .
MiniIE	WIKI	him :: lead Dawlish to 2nd place in :: Western League'	40% False	
MinIE	SCI	$> 250\,\mu\mathrm{m}$ fractions :: be found through :: light microscope analysis to be dominated by amorphous organic matter	40% True	'The $> 250\,\mu\mathrm{m}$ fractions, found through light microscope analysis to be dominated ($> 90\%$) by amorphous organic matter (AOM), were also analysed for $\delta 13C$.

Table 1: Examples of difficult to judge triples and their associated sentences.

from which it is extracted, we rely on the unanimous positive agreement between crowd workers. That is to say that if we have 100% inter-annotator agreement that a triple was correctly extracted we label it as correct.

7 Experimental Results

Table 2 show the results for the combinations of systems and data sources. The CORRECT TRIPLES column contains the number of triples that are labelled as being correct by all annotators. TOTAL TRIPLES are the total number of triples extracted by the given systems over the specified data. Precision is calculated as typical where Correct Triples are treated as true positives. On average, 3.1 triples were extracted per sentence.

Figure 1 shows the performance of extractors in terms of precision as inter-annotator agreement de-

SYSTEM	SOURCE	TOTAL TRIPLES	CORRECT TRIPLES	PRECISION
OpenIE 4 + MinIE	SCI + WIKI	2247	985	0.44
OpenIE 4 + MinIE	WIKI	1101	590	0.54
OpenIE 4 + MinIE	SCI	1146	395	0.34
MinIE	SCI + WIKI	1591	617	0.39
MinIE	WIKI	785	382	0.49
MinIE	SCI	806	235	0.29
OpenIE 4	SCI + WIKI	656	368	0.56
OpenIE 4	WIKI	316	208	0.66
OpenIE 4	SCI	340	160	0.47

Table 2: Results of triples extracted from the SCI and WIKI corpora using the Open IE and MinIE tools.

creases. In this figure, we look only at agreement on triples where the majority agree that the triple is correct. Furthermore, to ease comparison, we only consider triples with 5 judgements this excludes 9 triples. We indicate not only the pair-wise inter-annotator agreement but also the number of annotators who have judged a triple to be correct. For example, at the 40% agreement level at least 3 annotators have agreed that a triple is true. The figure separates the results by extractor and by data source.

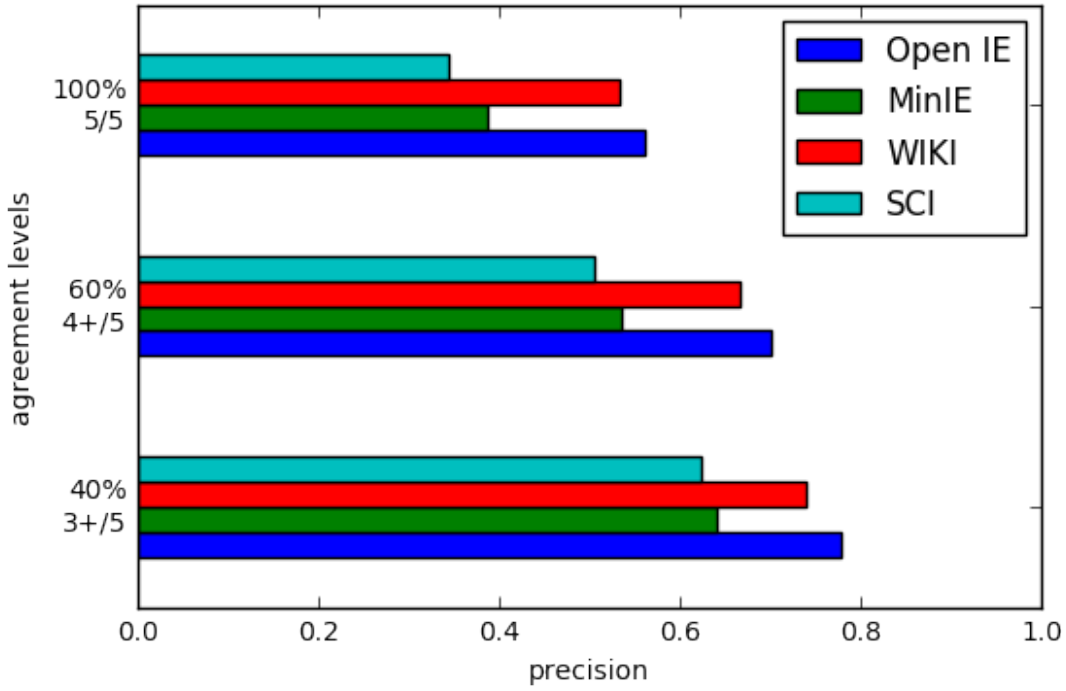


Figure 1: Precision at various agreement levels. Agreement levels are shown as the proportion of overall agreement. In addition, we indicate the the minimum number of annotators who considered relations correct out of the total number of annotators.

We see that as expected the amount of triples agreed to as correct grows larger as we relax the requirement for agreement. For example, analyzing Open IE's results, at the 100% agreement level we see a precision of 0.56 whereas at the 40% agreement level we see a precision of 0.78. Table 3 shows the total number of correct extractions at the three agreement levels.

AGREEMENT	OPEN IE 4 SCI	MINIE SCI	OPEN IE 4 WIKI	MINIE WIKI	OPEN IE 4 TOTAL	MINIE TOTAL
40% (3+/5)	253	462	256	553	509	1015
60% (4+/5)	218	361	241	488	459	849
100% (5/5)	160	235	207	377	367	612

Table 3: Correct triples at different levels of agreement subsetted by system and data source . Agreement levels follow from Figure 1

7.1 Testing H1: Comparing the Performance of OIE on Scientific vs. Encyclopedic Text

From the data, we see that extractors perform better on sentences from Wikipedia (0.54 P) than scientific text (0.34 P). Additionally, we see that there is higher annotator agreement on whether triples extracted from Wikipedia and scientific text are correct or incorrect: 0.80 - SD 0.24 (WIKI) vs. 0.72 - SD 0.25 (SCI). A similar difference in agreement is observed when only looking at triples that are considered to be correct by the majority of annotators: 0.87 - SD 0.21 (WIKI) vs. 0.78 - SD 0.25 (SCI) . In both cases, the difference is significant with p-values < 0.01 using Welch's t-test. The differences between

data sources are also seen when looking at the individual extraction tools. For instance, for Open IE 4 the precision is 0.19 higher for wikipedia extractions over those from scientific text. *With this evidence, we reject our first hypothesis that the performance of these extractors are similar across data sources.*

7.2 Testing H2: Comparing the Performance of Systems

We also compare the output of the two extractors. In terms precision, Open IE 4 performs much better across the two datasets (0.56P vs 0.39P). Looking at triples considered to be correct by the majority of annotators, we see that Open IE 4 has higher inter-annotator agreement 0.87 - SD 0.22 (Open IE) vs 0.81 - SD 0.24 (MinIE). Focusing on scientific and medical text (SCI), again where the triples are majority annotated as being correct, Open IE has higher inter-annotator agreement (Open IE: 0.83 - SD 0.24 vs MiniIE: 0.76 - SD 0.25). In both cases, the difference is significant with p-values < 0.01 using Welch's t-test. This leads us to conclude that Open IE produces triples that annotators are more likely to agree as being correct.

MinIE provides many more correct extractions than OpenIE 4 (935 more across both datasets). The true recall numbers of the two systems can not be calculated with the data available, but the 40% difference in the numbers of correct extractions is strong evidence that the two systems do not have equivalent behavior.

A third indication of differences in their outputs comes from examining the complexity of the extracted relations. Open IE 4 generates longer triples on average (11.5 words) vs. 8.5 words for MinIE across all argument positions. However, Open IE 4 generates shorter relation types than MinIE (Open IE - 3.7 words; MiniIE 6.27 words) and the standard deviation in terms of word length is much more compact for Open IE 4 - 1 word vs 3 words for MinIE. *Overall, our conclusion is that Open IE 4 performs better than MinIE both in terms of precision and compactness of relation types, while not matching MinIE's recall, and thus we reject our second hypothesis.*

7.3 Other Observations

The amount of triples extracted from the scientific text is slightly larger than that extracted from the Wikipedia text. This follows from the fact that the scientific sentences are on average roughly 7 words longer than encyclopedic text.

The results of our experiment also confirm the notion that an unsupervised approach to extracting relations is important. We have identified 698 unique relation types that are part of triples agreed to be correct by all annotators. This number of relation types is derived from only 400 sentences. While not every relation type is essential for downstream tasks, it is clear that building specific extractors for each relation type in a supervised setting would be difficult.

8 Error Analysis

We now look more closely at the various errors that were generated by the two extractors.

Table 4 shows the sentences in which neither extractor produced triples. We see 3 distinct groups. The first are phrases that are incomplete sentences usually originating from headings (e.g. Materials and methods). The next group are descriptive headings potentially coming from paper titles or figure captions. We also see a group with more complex prepositional phrases. In general, these errors could be avoided by being more selective of the sentences used for random selection. Additionally, these systems could look at potentially just extracting noun phrases with variable relation types, hence, expressing a cooccurrence relation.

We also looked at where there was complete agreement by all annotators that a triple extraction was incorrect. In total there were 138 of these triples originating from 76 unique sentences. There were several patterns that appeared in these sentences.

- Long complex sentences led to incorrect extractions. For example, ``Most strikingly, there was a mutant gene-dose-dependent increase in caspase-cleaved TAU fragments, as determined by the caspase-cleaved TAU-specific antibody C3 (Gamblin et al., 2003; Guillozet-Bongaarts et al., 2005), in neurons derived from the isogenic

SOURCE	SENTENCE
SCI	Note that Eq.
	Materials and methods
	Site and experimental carbonate chemistry
	Soil aggregate characteristics
	An equation analogous to Eq.
	An Experimental Platform for the Efficient Generation of Human Cranial Placodes In Vitro
	Analysis 1: Manifest Vascular and Nonvascular Disease as a Predictor of Depressive Symptoms
	Autografts Elicit Only a Minimal Immune Response in the Primate Brain
	Production of wild-type TTR and ATTR Val30Met by differentiated hepatocyte-like cells
	Cryopreservation and recovery of hiPSCs in suspension culture
	For a detailed description of analysis methods and precision see Lee et al. (1997).
	Thus there are, up to associativity, only finitely many such q.
	In the absence of heavy elements, H3+ forms near the base of the model and subsequent infrared cooling balances the EUV heating rates.
	Assume that L is a line bundle.
	The set of the representative points is called the non-dominated front (or Pareto front).
WIKI	Simultaneously won the " Hope of the World Ballet " Prize .

Table 4: Sentences in which no triples were extracted

`TAU-A152T-iPSCs (Figures 4J and 4K).''`) led to the following triples that were deemed incorrect: `isogenic TAU-A152T-iPSCs :: is :: Figures` and `Gamblin et :: is :: caspase-cleaved TAU-specific antibody C3`

- The use of pronouns as subjects led to triples that were deemed to be incorrect. (e.g. `''So he was forced to requisition not only the public treasury of Gades but also the wealth from its temples.` led to the following incorrect triples: `he :: was forced :: to requisition not only the public treasury of Gades but also the wealth from its temples` and `he :: to requisition :: not only the public treasury of Gades but also the wealth from its temples.`

- Complex mathematical formula often generated incorrect triples

- MinIE as part of its extraction process substitutes quantities with variables (e.g. QUANT_1). We included this potential in our labelling instructions but this often led to triples that were labelled as incorrect by the annotators. We believe this source of error could stem from the given instructions.

- Reuse of abbreviations as adjectives also led to incorrect triples. For example, "BMP antagonists" and "BMP pathway" in the following sentence: `We also observed significant transcriptional changes in WNT and BMP pathway components such as an increase in the WNT pathway inhibitor DKK-1 and BMP antagonists, such as GREMLIN-1 and BAMBI (Figures 2B-2D), which are known transcriptional targets of BMP signaling (Grotewold et al., 2001).` led to the following triples that were labelled incorrect: `BMP antagonists :: are known ::` and `increase :: is :: bmp pathway component.`

We also see similar errors to those pointed out by (Schneider et al., 2017), namely, uninformative extractions, the difficulty in handling n-ary relations that are latent in the text, difficulties handling negations, and very large argument lengths. In general, these errors together point to several areas for further improvement including:

- deeper co-reference resolution either for variables in mathematical formula or for pronouns;
- improved handling of prepositional phrases;
- relaxing requirements for correct grammar within sentences;
- better handling of abbreviations.

9 Conclusion

The pace of change in the scientific literature means that interconnections and facts in the form of relations between entities are constantly being created. Open information extraction provides an important tool to keep up with that pace of change. We have provided evidence that unsupervised techniques are needed to be able to deal with the variety of relations present in text. The work presented here provides an independent evaluation of these tools in their use on scientific text. Past evaluations have focused on encyclopedic or news corpora which often have simpler structures. We have shown that existing OIE systems perform worse on scientific and medical content than on general audience content.

There are a range of avenues for future work. First, the application of Crowd Truth framework (Aroyo and Welty, 2013) in the analysis of these results might prove to be useful as we believe that the use of unanimous agreement tends to negatively impact the perceived performance of the OIE tools. Second, we think the application to n-ary relations and a deeper analysis of negative relations would be of interest. To do this kind of evaluation, an important area of future work is the development of guidelines and tasks for more complex analysis of sentences in a crowd sourcing environment. The ability, for example, to indicate argument boundaries or correct sentences can be expected of expert annotators but needs to implemented in a manner that is efficient and easy for the general crowd worker. Third, we would like to expand the evaluation dataset to an even larger numbers of sentences. Lastly, there are a number of core natural language processing components that might be useful for OIE in this setting, for example, the use of syntactic features as suggested by (Christensen et al., 2011). Furthermore, we think that coreference is a crucial missing component and we are actively investigating improved coreference resolution for scientific texts.

To conclude, we hope that this evaluation provides further insights for implementors of these extraction tools to deal with the complexity of scientific and medical text.

References

Lora Aroyo and Chris Welty. 2013. Crowd truth: Harnessing disagreement in crowdsourcing a relation extraction gold standard. *WebSci2013. ACM*, 2013.

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2670–2676, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Lutz Bornmann and Rüdiger Mutz. 2015. Growth rates of modern science: A bibliometric analysis based on the number of publications and cited references. *Journal of the Association for Information Science and Technology*, 66(11):2215–2222.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2011. An analysis of open information extraction based on semantic role labeling. In *Proceedings of the Sixth International Conference on Knowledge Capture*, K-CAP '11, pages 113–120, New York, NY, USA. ACM.

Matthew J. C. Crump, John V. McDonnell, and Todd M. Gureckis. 2013. Evaluating amazon's mechanical turk as a tool for experimental behavioral research. *PLOS ONE*, 8(3):1–18, 03.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: Clause-based open information extraction. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 355–366, New York, NY, USA. ACM.

Michael Yoshitaka Erlewine and Hadas Kotek. 2016. A streamlined approach to online linguistic surveys. *Natural Language & Linguistic Theory*, 34(2):481–495, may.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545, Stroudsburg, PA, USA. Association for Computational Linguistics.

Karën Fort, Gilles Adda, and K. Bretonnel Cohen. 2011. Amazon Mechanical Turk: Gold Mine or Coal Mine? *Computational Linguistics*, 37(2):413–420, jun.

Kiril Gashteovski, Rainer Gemulla, and Luciano Del Corro. 2017. MinIE: Minimizing Facts in Open Information Extraction. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2620–2630.

Paul T. Groth, Sujit Pal, Darin McBeath, Brad Allen, and Ron Daniel. 2016. Applying universal schemas for domain specific ontology expansion. In Jay Pujara, Tim Rocktäschel, Danqi Chen, and Sameer Singh, editors, *Proceedings of the 5th Workshop on Automated Knowledge Base Construction, AKBC@NAACL-HLT 2016, San Diego, CA, USA, June 17, 2016*, pages 81–85. The Association for Computer Linguistics.

Johannes Leveling. 2015. Tagging of temporal expressions and geological features in scientific articles. In *Proceedings of the 9th Workshop on Geographic Information Retrieval*, GIR '15, pages 6:1–6:10, New York, NY, USA. ACM.

Mausam Mausam. 2016. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 4074–4077. AAAI Press.

Hoifung Poon, Kristina Toutanova, and Chris Quirk. 2014. Distant supervision for cancer pathway extraction from text. In *Pacific Symposium on Biocomputing Co-Chairs*, pages 120–131. World Scientific.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In Mirella Lapata, Phil Blunsom, and Alexander Koller, editors, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 1171–1182. Association for Computational Linguistics.

Rudolf Schneider, Tom Oberhauser, Tobias Klatt, Felix A. Gers, and Alexander Löser. 2017. Analysing errors of open information extraction systems. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 11–18. Association for Computational Linguistics.

Simple Algorithms For Sentiment Analysis On Sentiment Rich, Data Poor Domains.

Prathusha K Sarma
University of Wisconsin-Madison
kameswarasar@wisc.edu

William A Sethares
University of Wisconsin-Madison
sethares@wisc.edu

Abstract

Standard word embedding algorithms learn vector representations from large corpora of text documents in an unsupervised fashion. However, the quality of word embeddings learned from these algorithms is affected by the size of training data sets. Thus, applications of these algorithms in domains with only moderate amounts of available data is limited. In this paper we introduce an algorithm that learns word embeddings jointly with a classifier. Our algorithm is called SWESA (Supervised Word Embeddings for Sentiment Analysis). SWESA leverages document label information to learn vector representations of words from a modest corpus of text documents by solving an optimization problem that minimizes a cost function with respect to both word embeddings and the weight vector used for classification. Experiments on several real world data sets show that SWESA has superior performance on domains with limited data, when compared to previously suggested approaches to word embeddings and sentiment analysis tasks.

1 Introduction

Word embedding algorithms learn vector representations for words that are useful to quantify semantic relationships between words in a given text. Additionally, word embeddings are used to initialize several algorithms for sentiment analysis, sentence encoding etc. Currently popular embedding algorithms such as word2vec (Mikolov et al., 2013b; Le and Mikolov, 2014), GloVe (Pennington et al., 2014), are based off neural network methods and have achieved tremendous success in various word, sentence/document level evaluation tasks. These algorithms thrive on the large volumes of training data sets when learning high quality word embeddings. However, there is an increase in application domains where getting large amounts of data is not always possible. Furthermore, in some of these domains, representing words from off-the-shelf word embeddings such as ones obtained from training word2vec, GloVe on Wikipedia or common-crawl may not be efficient. This is because the sentiment expressed by a word in such datasets could be somewhat different from the sentiment expressed by the same word when it appears in Wikipedia, common-crawl. A concrete example of such domains is given in the next paragraph. *In such cases off-the-shelf word embeddings are not very useful and better techniques are required to learn word embeddings and classifiers for sentiment analysis.*

The goal of this paper is to build classification algorithms for sentiment analysis on small, domain specific data sets that are sentiment rich. Such an algorithm is not limited to sentiment classification and easily generalizes to other text classification problems as well. An example of a small sized and sentiment rich data set is the Substance Use Disorder (SUD) data set (Mohr et al., 2013), (Moore et al., 2011) obtained from digital health intervention treatments. These treatments aim to predict relapse risk by analyzing the content of participants' text messages. Though forum moderators can monitor and provide support when participants are struggling, considerable labor is involved in reviewing and deciding the risk level of each text message. Text data obtained from these discussion forums is rich in sentiments such as 'determination,' 'pleasure,' 'anger,' 'fear' and by analyzing discussion messages for

Proceedings of the 27th International Conference on Computational Linguistics, pages 3424–3435
Santa Fe, New Mexico, USA, August 20-26, 2018.

sentiments such as 'anger', 'fear' it is possible to develop efficient algorithms that can automatically tag if a message is positive (i.e. benign) or negative (indicative of relapse).

One approach to this problem would be to use off-the-shelf word embeddings to learn document embeddings and then run a classification algorithm on the obtained embeddings. While this approach does decently well it fails to capture and exploit the semantics of within domain words. For example, consider words such as 'alcohol,' 'holiday,' 'party.' Such words are typically neutral or positive sentiment in the Wikipedia corpus. However, with data sets such as SUD, these words are indicative of a moderate/strong negative sentiment. As a result using off-the-shelf word embeddings from GloVe, word2vec embeddings trained on Wikipedia corpus for building a sentiment analyzer will result in a sentiment analyzer that performs poorly. An alternative is to obtain word embeddings by training word2vec or GloVe on the target dataset, and then using the resultant embeddings for sentiment analysis. Unsurprisingly this approach succeeds only when one has the luxury of large datasets. However, in the SUD application, as mentioned above, the amount of data that is available is limited and hence this approach is not a successful alternative. In fact we argue that in the presence of limited, but sentiment rich data, it is better to learn word embeddings in a supervised manner. This way the resulting embeddings tend to be polarity-aware and are better suitable for downstream tasks such as sentiment analysis. Our contributions are as follows,

1. We introduce an algorithm (Section 3) that jointly learns word embeddings as well as a sentiment analyzer (classifier) by solving a bi-convex optimization problem. Our algorithm is called Supervised Word Embedding for Sentiment Analysis (SWESA) . This is an iterative algorithm that minimizes a cost function for both a classifier and word embeddings under unit norm constraint on the word vectors.

2. SWESA uses document labels for learning word embeddings. Using document labels within SWESA helps us overcomes the problem of small-size training data and allows learning of polarity-aware word embeddings. Via a thorough empirical evaluation (Section (4)) we show that our algorithm outperforms classifiers built by re-training off-the-shelf word embeddings such as word2vec, GloVe. We also compare SWESA against an algorithm that uses convolutional neural networks to represent sentences (Kim, 2014) for sentiment analysis, and a sentiment analysis algorithm based on recurrent neural networks (Socher et al., 2013). On the A-CHESS dataset where data is limited but rich in sentiment, SWESA outperforms all algorithms by at least 12% on the precision.

3. To demonstrate the fact that the word embeddings learned by SWESA are better than embeddings learned from unsupervised learning algorithms we investigate the polarity of various word embeddings. We show that the embeddings learned from SWESA perform well on antonym task. For example, 'Awful/Good' is the antonym pair returned via SWESA as opposed to 'Awful/Could' obtained via word2vec. SWESA learns such antonym pairs, using document polarities, and as a byproduct of our optimization based formulations, independent of an antonym pairs training data set.

2 Related Work

Modern embedding algorithms such as word2vec (Mikolov et al., 2013a) and GloVe (Pennington et al., 2014) are neural network based algorithms that learn word embeddings in an unsupervised fashion. These algorithms exploit word co-occurrence statistics in order to learn word embeddings. Due to this, typically these algorithms thrive on large training data sets in order to learn generic word embeddings. Also these models learn word embeddings in an unsupervised fashion. However, using labeled data can often help with learning sentiment-aware word embeddings more appropriate to the corpus at hand In their work (Maas et al., 2011) propose a probabilistic model that captures semantic similarities among words across documents. This model leverages document label information to improve word vectors to better capture sentiment of the contexts in which these words occur. While it may seem that this model is similar to SWESA, it is not so because in their model (Maas et al., 2011) first learn word embeddings and then fit a classifier in two different objective functions. On the other hand SWESA is a single bi-convex objective. The probabilistic model used by (Maas et al., 2011) is similar to that in Latent Dirichlet

Allocation (LDA) (Blei et al., 2003) in which each document is modeled as a mixture of latent topics. In (Maas et al., 2011), word probabilities in a document are modeled directly assuming a given topic. Yet another model that makes use of word co-occurence features in a supervised learning task is that of (Aga et al., 2016). In this model word co-occurences and context are used to learn word embeddings through matrix factorization algorithms. However, the classification task is independent of the process used to learn word embeddings.

(Tang et al., 2014) propose a supervised neural network based model to classify Twitter data. The proposed algorithm learns sentiment specific word vectors, from tweets making use of emoticons in text to guide sentiment of words used in the text instead of annotated sentiment labels. The Recursive Neural Tensor Network (RNTN) proposed by (Socher et al., 2013) classifies sentiment of text of varying length. To learn sentiment from long text, this model exploits compositionality in text by converting input text into the Sentiment Treebank format with annotated sentiment labels. The Sentiment Treebank is based on a data set introduced by Pang and Lee (Pang and Lee, 2005). This model performs particularly well on longer texts by exploiting compositionality as opposed to a regular bag of features approach. A popular algorithm for sentiment analysis is the CNN based approach proposed by (Kim, 2014). This algorithm takes as input word embedding and learns a CNN based sentence composition on which sentiment analysis is performed. Finally we would like to mention that in some domains, the vocabulary in the domain is manually labeled. For example, SentiStrength (Thelwall et al., 2010), LIWC (Pennebaker et al., 2001) projects provide manually labeled data. However, such efforts are not scalable and need considerable human expertise.

Notation: Throughout this paper we shall denote word vectors as $\mathbf{w}_j \in \mathbb{R}^k$, for $j = 1, \ldots, V$, where V indicates the size of the vocabulary. The matrix of word vectors is $\mathbf{W}$ where $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_V] \in \mathbb{R}^{k \times V}$. The classifier to be learned is represented by $\boldsymbol{\theta} \in \mathbb{R}^k$, weights of word vectors $\mathbf{w}_j$ in document i are contained in the vector $\boldsymbol{\phi}_i \in \mathbb{R}^V$, and the document label of the i^{th} document is indicated by y_i, document i is represented as $d_i = \mathbf{W}\,\boldsymbol{\phi}_i$. Let $\Phi = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_N] \in \mathbb{R}^{V \times N}$ be the matrix containing weight vectors $\boldsymbol{\phi}_i$ and vector $\mathbf{y} = [y_1, y_2, ...y_N]$ be the vector containing document labels.

3 Supervised Word Vectors for Sentiment Analysis

Given a collection of documents $d_1, d_2, \ldots d_N$ with binary sentiments $y_1, y_2, \ldots, y_N$ respectively, the aim is to learn a classifier that when given a new, previously unseen document d can accurately estimate the sentiment of the document. There could be class imbalance in the training data and so the algorithm should explicitly account for such a class imbalance. We approach this problem by introducing a new algorithm called SWESA. SWESA simultaneously learns word vector embeddings and a classifier, by making use of document polarity/sentiment labels. Representation of documents within SWESA is motivated by the fact that in short texts like "I am sad", "I am happy", polarity of the sentence hinges on the words "sad" and "happy". As a result, by learning polarity aware word embeddings, good vector representations for documents can be achieved. For instance, in the above example, the distance between the vectors $(\mathbf{w}_I + \mathbf{w}_{am} + \mathbf{w}_{sad})$ and $(\mathbf{w}_I + \mathbf{w}_{am} + \mathbf{w}_{happy})$ would capture dissimilarities in sentiment of these two documents while at the same time reflecting similarities in sentence structure.

Text documents in this framework are represented as a weighted linear combination of words in a given vocabulary. Weights can be either the term frequencies (tf) of words within each document or term frequency-inverse document frequency (tf-idf). Weights provided as input to SWESA for experiments described in Section (4) are term frequencies. We choose this weighting scheme to mimic the concept of local context used in the word2vec family of algorithms. Global co-occurrence information can be leveraged by using tf-idf for weighting words in documents. Such an approach in not entirely unheard off in sentiment analysis tasks, where word embeddings are considered as features for a classification algorithm (Labutov and Lipson, 2013).

SWESA aims to find vector representations for words, which when combined using weights, in a bag-of-words framework provide us with embeddings for text documents. These embeddings should be such that applying a nonlinear transformation f to the product $(\theta^\top\, \mathbf{W}\, \boldsymbol{\phi})$ results in a binary label y indicating

the polarity of document. Mathematically we assume that,

$$\mathbb{P}[Y = 1|d = \mathbf{W}\,\boldsymbol{\phi}, \boldsymbol{\theta}] = f(\boldsymbol{\theta}^\top \mathbf{W}\,\boldsymbol{\phi}) \tag{1}$$

for some function f. In order to solve for $\boldsymbol{\theta}$ and $\mathbf{W}$, a regularized negative likelihood minimization problem is solved. This optimization problem is as (1) and can be solved as a minimization problem with objective function,

$$J(\boldsymbol{\theta}, \mathbf{W}) \stackrel{\text{def}}{=} \frac{-1}{N}\Big[\mathrm{C}_+ \sum_{y_i=+1} \log \mathbb{P}(Y = y_i|\,\mathbf{W}\,\boldsymbol{\phi}_i, \boldsymbol{\theta}) + \mathrm{C}_- \sum_{y_i=-1} \log \mathbb{P}(Y = y_i|\,\mathbf{W}\,\boldsymbol{\phi}_i, \boldsymbol{\theta})\Big] + \lambda_{\boldsymbol{\theta}}||\,\boldsymbol{\theta}\,||_2^2. \tag{2}$$

This optimization problem can now be written as

$$\begin{aligned} \min_{\substack{\boldsymbol{\theta}\in\mathbb{R}^k,\\ \mathbf{W}\in\mathbb{R}^{k\times V}}} \quad & J(\boldsymbol{\theta}, \mathbf{W}) \\ \text{s.t. } & ||\,\mathbf{w}_j\,||_2 = 1\ \forall j = 1, \ldots V. \end{aligned} \tag{3}$$

The vector $\boldsymbol{\phi}_i$ is a vector of weights, corresponding to the different words, for document d_i. As mentioned previously, for testing SWESA term frequencies of different words in a certain document i are used in $\boldsymbol{\phi}_i$. $\lambda_\theta > 0$ is the regularization parameter for the classifier θ, C_+ is the cost associated with misclassifying a document from the positive class and C_- is the cost associated with misclassifying a document from the negative class. Following the heuristic suggested by (Lin et al., 2002), $\mathrm{C}_+ = \frac{\mathrm{N}_-}{N}$ and $\mathrm{C}_- = \frac{\mathrm{N}_+}{N}$, where N_+ is the number of positive documents in the corpus and N_- is the number of negative documents in the corpus. This scheme is particularly useful when dealing with data sets with imbalanced classes. When using a balanced data set $\mathrm{C}_+ = \mathrm{C}_-$. Sentiment in a given document is captured by the document label y_i, which in this framework is a binary label that capture sentiments such as 'positive/negative' or 'threatening/benign' depending on the data set.

The unit norm constraint in the optimization problem shown in (3) is enforced on word embeddings to discourage degenerate solutions of $\mathbf{w}_j$. For example in the absence of this constraint, the optimal $\mathbf{w}_j^*$ is typically a vector of zeros. Note that this optimization problem is bi-convex, but it is not jointly convex in the optimization variables. Algorithm 1 shows the algorithm that we use to solve the optimization problem in (3). This algorithm is an alternating minimization procedure that initializes the word embedding matrix $\mathbf{W}$ with $\mathbf{W}_0$ and then alternates between minimizing the objective function w.r.t. the weight vector $\boldsymbol{\theta}$ and the word embeddings $\mathbf{W}$.

Algorithm 1 Supervised Word Embeddings for Sentiment Analysis (SWESA)

Require: $\mathbf{W}_0$, Φ, C_+, C_-, λ_θ, $0 < k < V$, Labels: $\mathbf{y} = [y_1, \ldots, y_N]$, Iterations: $T > 0$,

1: Initialize $\mathbf{W} = \mathbf{W}_0$.
2: **for** $t = 1, \ldots, T$ **do**
3: Solve $\boldsymbol{\theta}_t \leftarrow \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \mathbf{W}_{t-1})$.
4: Solve $\mathbf{W}_t \leftarrow \arg\min_{\mathbf{W}} J(\boldsymbol{\theta}_t, \mathbf{W})$.
5: **end for**
6: Return $\boldsymbol{\theta}_T, \mathbf{W}_T$

3.1 Logistic regression model

The optimization problem in (2) assumes a certain probability model and minimizes the negative log-likelihood under norm constraints. While, the specific goal of the user might dictate an appropriate choice of probabilistic model, for a large class of classification tasks such as sentiment analysis, the logistic regression model is widely used. In this section we assume that the probability model of interest is the logistic model. Under this assumption the minimization problem in Step 3 of Algorithm 1 is a

standard logistic regression problem [1]. Many specialized solvers have been devised for this problem and in this implementation of SWESA, a standard off-the-shelf solver available in the scikit-learn package in Python is used. In order to solve the optimization problem in line 4 of Algorithm 1 a projected stochastic gradient descent (SGD) with suffix averaging (Rakhlin et al., 2011) is used. In suffix averaging the last few iterates obtained during stochastic gradient descent are averaged. Suffix averaging guarantees that the noise in the iterates is reduced and has been shown to achieve almost optimal rates of convergence for minimization of strongly convex functions. For experiments in Section 4 we set $\tau = 50$.

Gradient updates for $\mathbf{W}$ given $\boldsymbol{\theta}$ are of the form

$$\nabla J(\boldsymbol{\theta}, \mathbf{W}) = \frac{1}{N}\Big[\sum_{y_i=+1} \frac{-\,\mathrm{C}_+\, y_i\, \boldsymbol{\theta}\, \boldsymbol{\phi}_i^\top}{1 + e^{y_i(\boldsymbol{\theta}^\top \mathbf{W}\, \boldsymbol{\phi}_i)}} + \sum_{y_i=-1} \frac{-\,\mathrm{C}_-\, y_i\, \boldsymbol{\theta}\, \boldsymbol{\phi}_i^\top}{1 + e^{y_i(\boldsymbol{\theta}^\top \mathbf{W}\, \boldsymbol{\phi}_i)}}\Big]. \tag{4}$$

Algorithm 2 implements the SGD algorithm (with stochastic gradients instead of full gradients) for solving the optimization problem in step 4 of Algorithm 1.

Algorithm 2 Stochastic Gradient Descent for $\mathbf{W}$

Require: $\boldsymbol{\theta}, \gamma, \mathbf{W}_0$, Labels: $\mathbf{y} = [y_1, \ldots, y_N]$, Iterations: N, step size: $\eta > 0$, and suffix parameter: $0 < \tau \leq N$.

1: Randomly shuffle the dataset.
2: **for** $t = 1, \ldots, N$ **do**
3: Set $C_t = C_+$ if $y_t = +1$, $C_t = C_-$ if $y_t = -1$.
4: $\widetilde{\mathbf{W}}_{t+1} = \mathbf{W}_t - \frac{\eta C_t}{1+e^{y_i(\boldsymbol{\theta}^\top \mathbf{W}\, \boldsymbol{\phi}_i)}} \times (-y_i\, \boldsymbol{\theta}\, \boldsymbol{\phi}_i^\top)$
5: $\mathbf{W}_{t+1,j} = \mathbf{W}_{t+1,j} \,/ || \,\mathbf{W}_{t+1,j}\, ||_2 \; \forall j = 1, 2, \ldots, V$
6: $\eta \leftarrow \frac{\eta}{t}$
7: **end for**
8: Return $\mathbf{W} = \frac{1}{\tau} \sum_{t=N-\tau}^{N} \mathbf{W}_t$

Convergence of SWESA: At a high level, SWESA can be seen as a variation of the supervised dictionary learning problem (SDL). Within SDL (Mairal et al., 2009) given labeled data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, and unlabeled part of the data that lies in a d dimensional space, the goal is to learn a dictionary D of size $d \times k$, $(k >> d)$ such that each $x_i = Dz_i$ where z_i is a sparse encoding of x_i w.r.t. dictionary D. Further, the label is generated by a linear classifier w.r.t z_i, i.e. $y_i = \mathbf{W}^\top z_i$. The learning problem is to estimate the dictionary, the codes of each data point and the classifier. SWESA can be roughly mapped to the SDL by considering dictionary D of size $k \times V$, where each column corresponds to a word embedding. However, there are significant differences between SWESA and SDL. Despite these differences, with suitable modifications convergence analysis for SWESA can be performed. Extensive literature on alternative minimization and their convergence guarantees for SDL and related problems such as matrix completion (Jain et al., 2013) exist. A full theoretical analysis is out of scope of this paper and is left for future work. A brief description of differences between SWESA and SDL can be found in Appendix A.

3.2 Initialization of W

Two different initialization procedures are used to obtain $\mathbf{W}_0$. The first method uses the Latent Semantic Analysis (Dumais, 2004) (LSA) procedure to form the matrix of word vectors $\mathbf{W}_0$ from the given corpus of text documents. The second method uses the word2vec (w2v) algorithm to form word vector matrix $\mathbf{W}_0$ from the corpus. Since word2vec is ideally trained on a dataset that is much larger in size than the test datasets in Section 4, it is expected that the initialization vectors obtained by retraining word2vec on the test data sets will not be as suitable as the LSA intialization. However, as expected SWESA improves the performance of any initial guess parameters.

[1] A bias term, γ can be trivially introduced in the logistic regression model.

4 Experimental Evaluation and Results

We now describe in brief the data sets and the algorithms that we use to evaluate SWESA.

4.1 Data Sets

We conduct our experiments on four small sized data sets out of which 3 are balanced and 1 is imbalanced. All four data sets are drawn from different domains and hence have different vocabularies and contexts. The four data sets are:

- **Yelp:** This data set consists of 1000 restaurant reviews labeled 'positive' or 'negative'. There are a total of 2049 distinct word tokens in this data set.
- **IMDB:** This data set consists of 1000 reviews of movies with labels 'positive' or 'negative'. There are a total of 3075 distinct word tokens.
- **Amazon:** This data set contains 1000 product reviews with labels 'positive' or 'negative'. This data set has 1865 distinct word tokens.
- **A-CHESS:** This is a proprietary data set[2] and is obtained from an intervention treatment aimed at alcohol disorder. Text is obtained via the discussion forums part of the A-CHESS mobile app. There are a total of 2500 messages with 8% of the messages indicating relapse risk or 'threat'. Typical messages from this data set are of the form, *"I've been clean for 7 months now, but I still feel I may not make it"*. Such a message is marked as 'threat'. Positive or 'benign' messages are of the form, *"30 days and sober! I feel great!!"*. This labeling is human moderated. The goal is to automate this process. After removing special characters, this data set consists of 3400 distinct word tokens.

The first three data sets are obtained from (Kotzias et al., 2015).

4.2 Baselines

We choose two types of baselines that cover both neural and non-neural network style algorithms.
Standard baselines:

1. **Two-Step (TS):** The TS algorithm is a family of algorithms where we perform two steps (i) First, word embeddings are trained in an unsupervised manner on the target dataset. These word embeddings are combined together with appropriate weights (as is done in SWESA) to obtain document embeddings. (ii) In the second step the document embeddings obtained from the first step are used in a logistic regression classifier to obtain a classification model. To implement our first step we use two types of word embeddings, namely LSA word embeddings, and word2vec (re-w2v) obtained by re-training LSA, and word2vec algorithms on our target datasets. We use an open source implementation in gensim[3] to get word2vec embeddings.
2. **Naive Bayes classifier:** The classic Naive Bayes classifier for sentiment classification based on the Bag-of-words features, optimized in NLTK toolkit in Python is used.

Neural network based baselines:

1. **Recursive Neural Tensor Network (RNTN):** RNTN is an RNN proposed by (Socher et al., 2013) that learns compositionality from text of varying length and performs classification in a supervised fashion with fine grained sentiment labels. Since SWESA is aimed at binary classification, RNTN is also used in a binary classification framework. RNTN has been shown to perform better than the previously proposed Recursive Auto Encoder (RAE) by (Socher et al., 2011) and hence we limit our comparisons to RNTN only.

[2]Center for Health Enhancement System Services at UW-Madison.
[3]`https://radimrehurek.com/gensim/models/word2vec.html`

2. **CNN based sentence classification:** (Kim, 2014) propose a Convolutional Neural Network (CNN) based architecture that takes as input word embeddings and uses a CNN based architecture to learn sentence embeddings. These sentence embeddings are then used for classification. This algorithm belongs to a large class of algorithms that take as input word embedding and learn sentence embeddings for other tasks. While it seems that SWESA is similar to these algorithms, the significant difference between the two lies in the application domain of SWESA. SWESA is geared towards small sized datasets, whereas neural network based algorithms are data intensive and can extract useful relationships from very large amounts of data.

Note that neural network based baselines are used in two modes,

- **Pre-trained (Pr-tr):** In this framework both RNTN and CNN-static (CNN-S), CNN-non static (CNN-NS) are initialized with word2vec embeddings (w2v) and are trained on the Pang and Lee data set (Pang and Lee, 2005). This data set consists of roughly 10k text documents and is roughly 10 times the size of the data sets considered in experiments in Section 4.4. Note that in these experiments, word embeddings used are obtained from word2vec trained on the Google news data set and do not make use of any labels from the test data sets considered.

- **Re-trained (Re-tr):** In this framework, RNTN is retrained on the data sets described in subsection 4.1. Further, CNN-static and non static are used in the following ways,

    1. CNN-static and CNN-non static are initialized with re-trained w2v (re-w2v) embeddings. The training set is still the Pang and Lee data set used by the authors (Kim, 2014). We expect the performance of this baseline to be poorer when compared to CNN-static and non static initialized with pre-trained word embeddings. This baseline is used to investigate the sensitivity of CNNs to initial input word embeddings.
    2. CNN-static and CNN-non static are initialized with pre-trained word2vec embeddings but trained on train/dev sets obtained from the data sets described in subsection 4.1. We expect this data set to be the least well performing data set due to the size of the training data. This baseline is used to illustrate the limitations of small sized training data sets on neural network algorithms.

4.3 Dimensionality of word embeddings and hyperparameters

Dimensions of word embeddings and other hyperparameters such as regularization on the logistic regression classifier are determined via 10 fold cross validation. Pre-trained RNTN[4] and CNN-static/non static[5] are used with the training sets and parameters as described by the authors of both works. Procedure for retraining RNTN as well as for fine tuning word embeddings for use in CNN-non static follow the methods described in (Socher et al., 2013) and (Kim, 2014) respectively.

4.4 Results from classification tasks

Performance metrics reported are average Precision and average AUC. AUC is only reported for model that provide prediction probabilities in addition to predicted label. For the A-CHESS data set the minor class, i.e 'threat' owing to the large imbalance is of more importance. This is because, the system can accept few false positives, but the risk of misclassifying a 'threat' message as 'benign' has a larger impact. Owing to this, on this data set, the best performing classifier will be one that achieves maximum precision. Note that the objective of these results is to show how well SWESA performs in applications with small sized data sets as compared to i) completely re-training a neural network with a smaller training set, ii) or initializing a neural network with word embeddings learned on small data sets and iii) against a neural network that uses pre-trained word embeddings and larger training sets.

SWESA against pre-trained neural nets initialized with pre-trained word embeddings: On the balanced Yelp, Amazon and IMDB data sets, pre-trained RNTN, CNN-static and non static perform the

[4] https://nlp.stanford.edu/sentiment/code.html
[5] https://github.com/yoonkim/CNN_sentence

best. From table 1, observe that the best performing baseline CNN-NS achieves an average precision of 87.98, 87.15 and 87.77 respectively on the three balanced data sets. This result is *not surprising*, given that the pre-trained RNTN and CNNS are i)initialized with pre-trained word embeddings and ii)trained on a data set with roughly 10 times more training data points than within SWESA. However, note that on the imbalanced A-CHESS data set RNTN fails to perform any classification. This can be attributed to the fact that RNTN is a dependency parser and on varying length text like within SWESA, it is very hard to capture dependencies. CNN-static and CNN-non static achieve poor precision owing to the class imbalance which is not accounted for when training them.

SWESA against re-trained neural nets: CNN-static (re-w2v) and CNN-non static (re-w2v) perform comparably well to SWESA on the balanced data sets. From table 1 notice that on the balanced Yelp, Amazon and IMDB data sets, SWESA achieves an average precision of 78.35, 80.36 and 77.27 respectively while CNN-non static (re-w2v) achieves an average precision of 78.10, 80.40 and 75.85 respectively. These results suggest that SWESA is a suitable alternative to neural network architectures that are sensitive to initializing word embeddings. This result is particularly useful for data sets that are not large enough to enable learning of high quality word embeddings. Note that on the A-CHESS data set CNN-static and non static perform poorly.

SWESA against neural nets initialized with re-trained word embeddings: re-trained RNTN performs very poorly on the three balanced data sets and fails to classify the A-CHESS test set. On the other hand while CNN-static (re-w2v) and CNN-non static (re-w2v) perform comparably well to SWESA on the balanced data sets, both algorithms perform poorly on the A-CHESS data set. CNN-static and non static achiever average precision of 15.62 and 18.98 respectively on the A-CHESS data set while SWESA achieves an average precision of 35.80. This is not a surprising result because the training data to RNTN and CNNs is much smaller than the Pang and Lee Data set.

Polarity of word embeddings. The objective of SWESA is to perform effective sentiment analysis by learning embeddings from text documents with sentiment labels. Since, word embeddings are learned via a joint optimization framework one can expect that the resulting word embeddings are polarity aware. To test our hypothesis, we investigate if one can predict the antonym of a word. That is, given a word v whose word embedding is $\mathbf{w}_v$ we determine the antonym to be that word a with word embedding $\mathbf{w}_a$ such that the cosine similarity between $\mathbf{w}_a, \mathbf{w}_v$ is minimized over all possible choices of a. Figure 1 shows a small sample of word embeddings learned on the Amazon data set by SWESA and word2vec. The cosine similarity (angle) between the most dissimilar words is calculated and owing to the assumptions on word embeddings, words are depicted as points on the unit circle.

From figure 1 it is evident that a supervised algorithm like SWESA projects document level polarity onto word level embeddings while an unsupervised algorithm like word2vec that learns embeddings of words via virtue of word co-occurrences will fail to embed polarity. It is important to notice that SWESA learns word polarities by using document polarities, and these word polarities are useful for antonym tasks. Unlike classical antonym tasks where examples of known antonym pairs are provided, in our setup no such pairs are provided, and yet SWESA was able to do a good job discovering antonym pairs. For example the most dissimilar word to given word 'Excellent' is 'Poor' when learned via SWESA as opposed to 'Work' when learned via word2vec. From these examples, the advantage of SWESA is particularly towards exploiting strong polarity differences in words such as 'Excellent' and 'Poor', however not much success is noted towards less polarized synonyms such as 'Wet' and 'Dry'. However, with finer grained sentiment labels we would expect subtler antonym pairs to be distinguished by SWESA. This will be actively pursed in future extensions of SWESA, particularly when used to perform multi-class classification tasks.

Thus, word antonym pairs (w_a, w) can be obtained by calculating cosine similarities. Since the aim of experiments in section 4 was to perform document level sentiment analysis and observations on word dissimilarities is a consequence of this task, extensive experiments validating word antonym pairs have not been performed. This is left for future work and only a qualitative analysis of learned word embeddings is discussed here.

Data Set	Method	Avg Precision	Avg AUC
Yelp	SWESA (LSA)	78.09±2.8	86.06±2.4
	SWESA (re-w2v)	**78.35±4.6**	**86.93±3.5**
	TS (LSA)	76.27±3.0	83.05±5.0
	TS (re-w2v)	65.22±4.4	69.08±3.5
	Naive Bayes	57.07±3.3	61.16±4.5
	RNTN (Pre-tr)	83.31±1.1	-
	RNTN (retrained)	51.15±4.3	-
	CNN-S (Pre-tr)	86.45±0.8	-
	CNN-NS (Pre-tr)	87.98±0.5	-
	CNN-S (re-w2v)	76.89±2.5	-
	CNN-NS (re-w2v)	78.10±1.5	-
	CNN-S (Re-tr)	68.55±1.2	-
	CNN-NS (Re-tr)	72.80±0.5	-
Amazon	SWESA (LSA)	80.31±3.3	87.54±4.2
	SWESA (re-w2v)	**80.36±2.8**	**87.19±3.3**
	TS (LSA)	77.32±4.6	85.00±6.2
	TS (re-w2v)	71.09±6.2	77.09±5.3
	Naive Bayes	72.54±6.4	61.16±4.5
	RNTN (Pre-tr)	82.84±0.6	-
	RNTN (retrained)	49.15±2.1	-
	CNN-S (Pre-tr)	87.08±1.0	-
	CNN-NS (Pre-tr)	87.15±0.8	-
	CNN-S (re-w2v)	78.85±1.2	-
	CNN-NS (re-w2v)	80.40±1.0	-
	CNN-S (Re-tr)	70.15±1.8	-
	CNN-NS (Re-tr)	72.86±2.5	-
IMDB	SWESA (LSA)	76.40±5.2	81.08±7.6
	SWESA (re-w2v)	**77.27±5.4**	**81.04±6.8**
	TS (LSA)	70.36±5.5	77.54±6.8
	TS (re-w2v)	56.87±7.6	59.34±8.9
	Naive Bayes	73.31±5.6	48.40±2.9
	RNTN (Pre-tr)	80.88±0.7	-
	RNTN (retrained)	53.95±1.9	-
	CNN-S (Pre-tr)	85.54±0.6	-
	CNN-NS (Pre-tr)	87.77±0.5	-
	CNN-S (re-w2v)	72.46±2.0	-
	CNN-NS (re-w2v)	75.85±2.1	-
	CNN-S (Re-tr)	62.07±1.5	-
	CNN-NS (Re-tr)	70.16±1.2	-
A-CHESS	**SWESA (LSA)**	**35.80±2.5**	**83.80±3.1**
	SWESA (re-w2v)	35.40±2.0	83.40±2.6
	TS (LSA)	32.20±3.2	**83.80±3.1**
	TS (re-w2v)	23.60±2.4	68.00±1.2
	Naive Bayes	30.30±3.8	45.23±3.3
	RNTN (Pre-tr)	-	-
	RNTN (retrained)	-	-
	CNN-S (Pre-tr)	18.75±3.2	-
	CNN-NS (Pre-tr)	23.02±2.8	-
	CNN-S (re-w2v)	20.12±2.4	-
	CNN-NS (re-w2v)	25.45±1.8	-
	CNN-S (Re-tr)	15.62±2.7	-
	CNN-NS (Re-tr)	18.98±2.2	-

Table 1: This table shows results from a standard sentiment classification task on all four data sets. Results from SWESA are in boldface and results from pre-trained models are in blue. No AUC values are reported for models where classification probabilities of classifiers is not available.

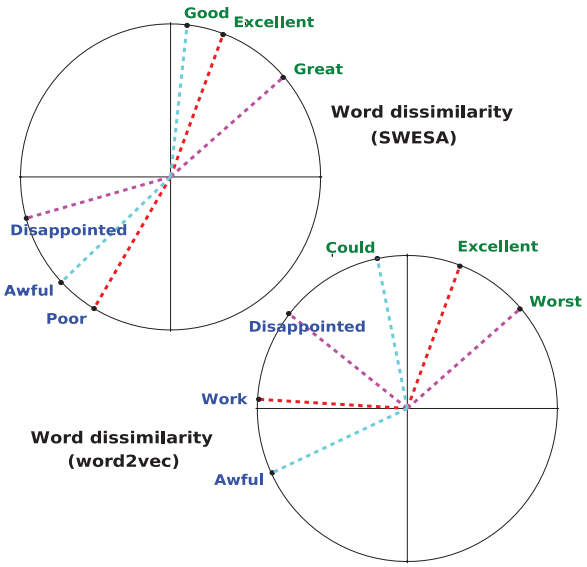


Figure 1: *This figure depicts word embeddings on a unit circle. Most dissimilar word pairs are plotted based on the cosine angle between the respective word embeddings learned via SWESA and word2vec.*

5 Discussions and Conclusions

In this paper we provide a simple optimization based framework, called SWESA that jointly learns word embeddings and a classifier for the task of sentiment analysis. Through extensive experimentation we show that SWESA outperforms both non-neural network algorithms (naive Bayes, LSA) as well as state-of-the-art neural network algorithms based on CNNs and RNNs. As a byproduct of our optimization formulation we show that the word embeddings learned by SWESA are polarity aware and perform very well on antonym tasks, even without being explicitly trained for such tasks. Particularly, strongly polarized word embeddings are easily distinguished. This results indicates that when using finer grained sentiment labels, word polarized along other scales such as dimension, etc can be determined. Our contributions strongly emphasize the point that on domains where data is limited, i.e few thousands of points, but sentiment rich, data size is not a handicap and simple algorithms can do a better task than more involved neural network based algorithm. In the future, we shall investigate modifications of our framework for tasks other than sentiment analysis.

References

[Aga et al.2016] Rosa Tsegaye Aga, Lucas Drumond, Christian Wartena, and Lars Schmidt-Thieme. 2016. Integrating distributional and lexical information for semantic classification of words using mrmf. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2708–2717.

[Blei et al.2003] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

[Dumais2004] Susan T Dumais. 2004. Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230.

[Jain et al.2013] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. 2013. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 665–674. ACM.

[Kim2014] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

[Kotzias et al.2015] Dimitrios Kotzias, Misha Denil, Nando De Freitas, and Padhraic Smyth. 2015. From group to individual labels using deep features. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 597–606. ACM.

[Labutov and Lipson2013] Igor Labutov and Hod Lipson. 2013. Re-embedding words. In *ACL (2)*, pages 489–493.

[Le and Mikolov2014] Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.

[Lin et al.2002] Yi Lin, Yoonkyung Lee, and Grace Wahba. 2002. Support vector machines for classification in nonstandard situations. *Machine learning*, 46(1-3):191–202.

[Maas et al.2011] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics.

[Mairal et al.2009] Julien Mairal, Jean Ponce, Guillermo Sapiro, Andrew Zisserman, and Francis R Bach. 2009. Supervised dictionary learning. In *Advances in neural information processing systems*, pages 1033–1040.

[Mikolov et al.2013a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mikolov et al.2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Mohr et al.2013] David C Mohr, Michelle Nicole Burns, Stephen M Schueller, Gregory Clarke, and Michael Klinkman. 2013. Behavioral intervention technologies: evidence review and recommendations for future research in mental health. *General hospital psychiatry*, 35(4):332–338.

[Moore et al.2011] Brent A Moore, Tera Fazzino, Brian Garnet, Christopher J Cutter, and Declan T Barry. 2011. Computer-based interventions for drug use disorders: a systematic review. *Journal of substance abuse treatment*, 40(3):215–223.

[Pang and Lee2005] Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

[Pennebaker et al.2001] James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001):2001.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.

[Rakhlin et al.2011] Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. 2011. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv preprint arXiv:1109.5647*.

[Socher et al.2011] Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*, pages 151–161. Association for Computational Linguistics.

[Socher et al.2013] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642.

[Tang et al.2014] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL (1)*, pages 1555–1565.

[Thelwall et al.2010] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment strength detection in short informal text. *Journal of the Association for Information Science and Technology*, 61(12):2544–2558.

Appendix A: Differences between SDL and SWESA.

The significant differences between SDL and SWESA are i) Input to SDL is a labeled dataset, with each data point already represented as a vector. This allows for a definition of reconstruction error within algorithms designed for SDL. In contrast, SWESA has labeled unstructured data without a direct vector

representation, and the aim is to learn vector representations for such data. As a result the notion of reconstruction error used in SDL does not apply to SWESA and hence the optimization formulation used is significantly different from the one used in SDL. ii) In SDL sparse encoding of each data point is to be learned, whereas in SWESA this sparse encoding is known and is proportional to the number of times a word appears in the document. (iii) Finally, in SDL the classifier is a high-dimensional vector that acts on the latent codes.

Word-Level Loss Extensions for Neural Temporal Relation Classification

Artuur Leeuwenberg and **Marie-Francine Moens**
Department of Computer Science
KU Leuven, Belgium
{*tuur.leeuwenberg, sien.moens*}*@cs.kuleuven.be*

Abstract

Unsupervised pre-trained word embeddings are used effectively for many tasks in natural language processing to leverage unlabeled textual data. Often these embeddings are either used as initializations or as fixed word representations for task-specific classification models. In this work, we extend our classification model's task loss with an unsupervised auxiliary loss on the word-embedding level of the model. This is to ensure that the learned word representations contain both task-specific features, learned from the supervised loss component, and more general features learned from the unsupervised loss component. We evaluate our approach on the task of temporal relation extraction, in particular, narrative containment relation extraction from clinical records, and show that continued training of the embeddings on the unsupervised objective together with the task objective gives better task-specific embeddings, and results in an improvement over the state of the art on the THYME dataset, using only a general-domain part-of-speech tagger as linguistic resource.

1 Introduction

Word representations in the form of continuous vectors are often pre-trained on large amounts of raw text to learn general word features, using unsupervised objectives. These representations are then used in supervised models for various classification tasks. However, such tasks sometimes require very specific features that may not have been captured by the unsupervised objective. In other domains such as computer vision, representations are often learned jointly from multiple resources for classification. In this work, we explore the possibility to exploit learning signals from both settings to construct better task-oriented word representations, and obtain a better relation classification model.

The main task in this work is the extraction of narrative containment relations (CR) from English clinical texts, as annotation of clinical data is costly and it is therefor crucial to fully exploit both the labeled as well as the unlabeled data that is available. The aim of CR extraction is to find if, given events A and B, event A is temporally contained in event B (i.e. if event A happens within the time span of event B). An example of such relation is given in Fig. 1, where the model should predict all containment edges given the entities (events and temporal expressions), by classifying each pair of entities as containment or no containment. Temporal relation classification in clinical text is a very important task in the secondary use of clinical data from electronic health records. The patient time-line is crucial for making a good patient prognosis and clinical decision support (Onisko et al., 2015). This task has already been addressed

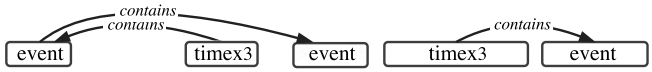


Figure 1: A sentence annotated with events, temporal expressions (Timex3), and containment relations.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3436–3447
Santa Fe, New Mexico, USA, August 20-26, 2018.

in three iterations of the Clinical TempEval Shared Task (Bethard et al., 2016). Still there is a gap of more than 0.20 in F-measure between the state-of-the-art CR extraction systems and the inter-adjunctator agreement (indicating an upper bound for performance). This shows that this task is very challenging.

Following the current trend in NLP, the recent state-of-the-art models for extraction of CR are neural network models. These models all use pre-trained word embeddings as word representations (Tourille et al., 2017; Dligach et al., 2017; Lin et al., 2017).

Pre-training of the embeddings is done with an auxiliary task (a task where one is not interested in the final predictions, but in the trained model components), like the skip-gram task (Mikolov et al., 2013). When used for classification tasks in NLP, these pre-trained word representations are often either used as fixed inputs for the classification model, or as initialization for the word representations of the classification model (sometimes called fine-tuned embeddings).

A problem with pre-trained representations in classification models is that solving the main task often requires different information than the auxiliary task. Training word representations only on the auxiliary objective can result in loss of crucial information for the task, and afterwards fine-tuning on the task loss does not influence words that are not in the task's training data, losing generalization to those words.

In the current work, we propose a neural relation classification (RC) model that learns its word representations jointly on the main task (supervised, on labeled data) and on the auxiliary task (unsupervised, on unlabeled data) in a multi-task setting to overcome this problem, and ensure that the embeddings contain valuable information for our main task, while still leveraging the unlabeled data for more general feature learning. As auxiliary task we implement a skip-gram (SG) architecture, similar to Mikolov et al. (2013). Our proposed models use only unlabeled data and a general (news, out of domain) part-of-speech (POS) tagger as external resources, in contrast to the current state-of-the-art models, to ease extension to other languages for which specialized NLP tools for clinical texts might not be available. The main contributions of this work are that it:

- Shows that training the word-level representations jointly on its main task and an auxiliary objective results in better representations for classification, compared to using pre-trained variants.
- Shows that the method's increased performance and hyper-parameters are robust across different training set sizes, and that single-loss training settings act as lower bounds on performance.
- Constitutes a new state of the art for temporal relation extraction on the THYME dataset even without dedicated clinical preprocessing.

2 Related Work

The model we present draws inspiration from prior research on (temporal) relation classification and neural multi-task learning.

2.1 Clinical Temporal Relation Extraction

Temporal relation extraction from clinical texts is a widely studied area in NLP and has been explored through various shared tasks, such as the i2b2 shared task on clinical temporal information (Sun et al., 2013), and three iterations of Clinical TempEval (Bethard et al., 2015; Bethard et al., 2016; Bethard et al., 2017). Until recently, most of the top performing systems employed manually constructed linguistic feature sets (Lin et al., 2015; Lee et al., 2016; Leeuwenberg and Moens, 2017). In the last few years, there has been a shift towards using neural models, using LSTM (Tourille et al., 2017) and CNN models (Dligach et al., 2017; Lin et al., 2017) inspired by the work on relation classification in other domains (Zeng et al., 2014; Zhang and Wang, 2015; Zhou et al., 2016; Nguyen and Grishman, 2015). The top results in clinical temporal relation extraction are still achieved when enhancing the neural models with dedicated clinical NLP tools for preprocessing the clinical texts, often using the English cTAKES system (Savova et al., 2010), which contains tools for clinical POS tagging, named entity recognition, and a dependency parser all trained on clinical data. The main reason for using these dedicated clinical tools is that parsers trained on non-clinical texts perform significantly worse on clinical data (Jiang et al., 2015).

Dedicated clinical NLP tools are not available for most languages though, and retraining NLP tools on clinical data is quite resource intensive, because it requires extra annotation effort. Additionally, clinical data is often difficult to obtain or share publicly for patient privacy reasons. Hence, we keep resource intensive preprocessing to a minimum and employ only a general news domain POS tagger (Toutanova et al., 2003), providing important temporal relation extraction cues, such as tense shifts (Derczynski, 2017), and for which training data are available for many languages (Petrov et al., 2012).

2.2 Multi-task Learning

Our proposed model training can be seen as multi-task learning (MTL), where the aim is to improve model generalization by leveraging the information from training signals of different related tasks (Caruana, 1998). In earlier work, MTL has shown to be quite effective for different NLP tasks such as machine translation (Dong et al., 2015), sentiment analysis (Peng and Dredze, 2015; Yu and Jiang, 2016), sentence level name prediction (Cheng et al., 2015), semantic role labeling (Collobert and Weston, 2008), and many more. For example, Collobert and Weston (2008) used an auxiliary unsupervised objective for semantic role labeling (SRL). They alternately trained embeddings in a language model and a SRL model. In contrast to their work, we learn both tasks truly jointly, and optimize a single semi-supervised objective. Typically in neural MTL, one or more layers of the network are shared among different models. Two issues in MTL are (1) how to determine if the tasks are related enough to benefit from each other, and (2) what layers to share among the models. Baxter and others (2000) theoretically argue that tasks are related when they share an inductive bias. In our model, we expect that the skip-gram task (Mikolov et al., 2013) can act as a reasonable word-level inductive bias for our task, as it has already shown its effectiveness in SRL (Collobert and Weston, 2008) and sentiment analysis (Peng and Dredze, 2015) in MTL, and for many NLP classification tasks when using them as pre-trained embeddings. Hashimoto et al. (2017) showed that even when combining many tasks, considering the task hierarchy (simpler tasks lower in the network) allows them to benefit from each other. In most work on MTL the auxiliary tasks are supervised and specifically chosen for their relatedness to the main task (Ruder, 2017), whereas in our model we chose the unsupervised auxiliary skip-gram task, and share weights of the word embedding layer. This results in a new joint relation classification objective that is semi-supervised on the word-level and provides better generalization for the final classification model.

3 The Model

Our model consist of two components: (1) a relation classification component (RC), and (2) a skip-gram component (SG). A high-level schematic overview of our model's training setting is shown in Fig. 2.

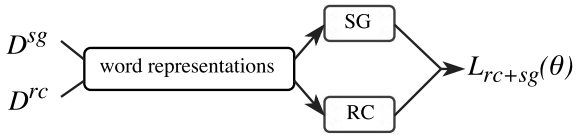


Figure 2: High-level overview of our model training setting. D^{rc} and D^{rc} indicate the dataset for the relation classification and skip-gram components respectively, and $L_{rc+sg}(\theta)$ the model's combined loss.

3.1 Relation Classification (RC)

To classify relations we employ a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) relation classification model (RC) (Zhang and Wang, 2015). We frame the task as a sequence classification problem, taking as an input: the textual candidate relation description, i.e. the arguments of the candidate relation (the entity pair), and the context words surrounding the arguments, all read as a sequence from left to right. A schematic overview of the RC model component is shown in Fig. 3.

Generation of candidate entity pairs is described later on in section 4.4. The locations of the arguments of each candidate relation are indicated by two types of features taken from the literature: (1) position indicators, which are XML tags added to the original input sequence indicating the start and end of the

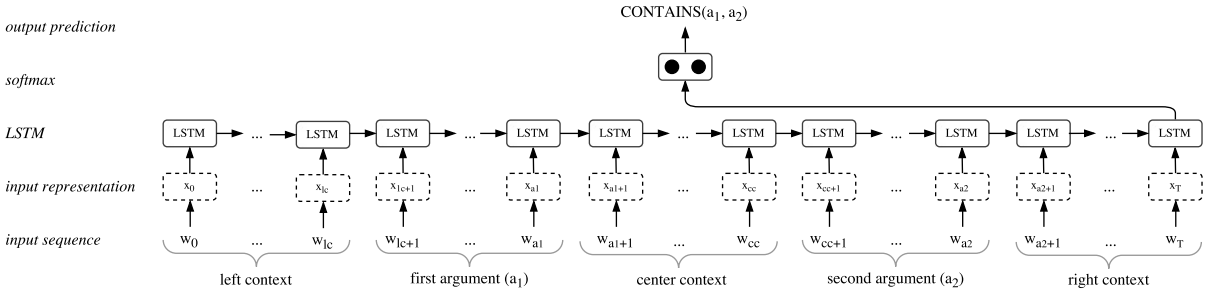


Figure 3: Schematic representation of the relation classification (RC) model component. Arrows represent sets of fully connected weights. The dashed box indicates a word input as shown in Fig. 4.

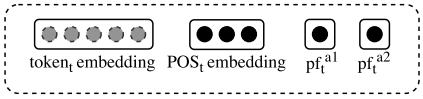


Figure 4: Each word input x_t of the RC model (at time step t) is a concatenation of a token embedding, a POS embedding, and two positional features (one for each argument).

arguments (Zhang and Wang, 2015), and (2) by position features, indicating the relative token-distance of each word to each argument (Zeng et al., 2014). Each word's total input x_t at time step $t \in \langle 0, 1, ..., T \rangle$ consists of the two argument locating features, pf_t^{a1} and pf_t^{a2}, together with a word embedding $x_t^{token} \cdot W_{em}^{token}$, and a POS embedding $x_t^{pos} \cdot W_{em}^{pos}$, where W_{em}^{token} and W_{em}^{pos} are the embedding matrices for tokens and POS respectively, and x_t^{token} and x_t^{pos} their one-hot representations. A schematic overview of the concatenated input x_t for each word to the LSTM unit is shown in Fig. 4.

The predicted class probabilities $\hat{p}_{rc}(x)$ are given by a softmax classifier placed on top of the LSTM output h at the last time step T (in Eq. 1).[1]

$$\hat{p}_{rc}(x) = \textit{softmax}(W_p h_T + b_p) \tag{1}$$

The RC model's loss function is cross-entropy loss, as shown in Eq. 2. D^{rc} indicates the supervised relation classification dataset, and θ^{rc} is the collection of all trainable parameters of the model.

$$L_{rc}(\theta^{rc}) = -\sum_{i=1}^{|D^{rc}|} y_i \log \hat{p}_{rc}(x_i) \tag{2}$$

3.2 Context Prediction (SG)

As the unsupervised auxiliary task, we implemented a feed-forward neural network for a word context prediction task, known as the continuous skip-gram (SG), following Mikolov et al. (2013). As input, the model takes a one-hot encoded input word w_j, which is projected to a word embedding, from which the probability distribution y over its surrounding context words $w_{j-c}, ..., w_{j-1}, w_{j+1}, ..., w_{j+c}$ is predicted, given a context window size c. The full model is given by Eq. 3.

$$\hat{p}_{sg}(w_j) = softmax(W_{p_{sg}}(w_j \cdot W_{em}^{token}) + b_{p_{sg}}) \tag{3}$$

Like the RC model, we use cross-entropy loss for our SG model, as shown in Eq. 4. D^{sg} indicates the unsupervised dataset, consisting of words and their contexts. θ^{sg} is the collection of all trainable parameters of our model.

$$L_{sg}(\theta^{sg}) = -\sum_{i=1}^{|D^{sg}|} y_i \log \hat{p}_{sg}(w_i) \tag{4}$$

[1]We also experimented with bidirectional LSTMs (Zhang et al., 2015) and adding attention (Zhou et al., 2016). In our experiments, this did not result in significant improvements.

3.2.1 Separate Left & Right Context (SGLR)

The skip-gram model is quite rough in its context description and does not take into account word order very well. However, for temporal relations we expect word order to be relevant. For this reason, we also experimented with a variation on the skip-gram model, separating the left and right context, following the intuition of Ling et al. (2015). The context separation is achieved by extending the context words by a 'left' or 'right' prefix depending on their location relative to the sampled word.

3.3 Combination (RC + SG)

We train our proposed model on a combination of both loss functions, each with their own dataset D^{rc}, and D^{sg} respectively. The combined loss, shown in Eq. 5, is a weighted sum of their cross-entropy losses, where λ_{sg} determines the importance of the SG loss.

$$L_{rc+sg}(\theta) = L_{rc}(\theta^{rc}) + \lambda_{sg} L_{sg}(\theta^{sg}) \tag{5}$$

A crucial part of our model is that although both models sample different types of inputs (the RC: sequences, the SG: single words) from different datasets, and have different classification weights, the word embeddings are shared, i.e. W_{em}^{token} ($\theta^{rc} \cap \theta^{sg} = W_{em}^{token}$), also illustrated in Fig. 2. So only the word embeddings are directly influenced by both losses. All other weights (from RC or SG) are only influenced indirectly, through the word embedding weights, as both models are trained simultaneously. Søgaard and Goldberg (2016) showed that, for NLP, sharing representations at the lower levels of the network is most effective: when lower level features are shared, there is room for the model to learn task specific abstractions in higher layers. For this reason we choose our model to share only the word embedding layer, as schematically illustrated in Fig. 5.

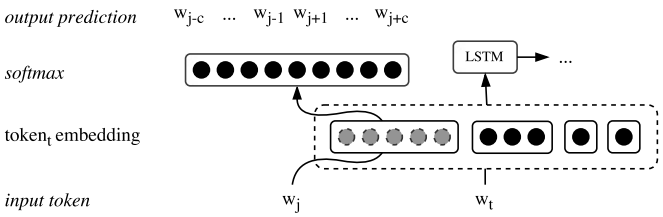


Figure 5: Schematic representation of how the SG model component extends the RC model when using the combined loss on input word $w_j \in D^{sg}$, and word w_t at time step t from input sequence $x_i \in D^{rc}$. The gray layer indicates the shared word embedding parameters. The dashed box represents the total word input x_t for RC, as in Fig. 3 and 4.

3.4 Training

We train all our models for at least 10 epochs, using Adam (Kingma and Ba, 2014) stochastic gradient descent, with the default parameters from the original paper (lr=0.001), and a batch size of 1024 on a Titan X GPU. As stopping criteria we employ early stopping (Morgan and Bourlard, 1990) with a patience of 20 epochs, based on F-measure on a small validation set of 3 documents (from Train). After each epoch on the main task, we shuffle the data and start the next one. During training we employ a dropout of 0.5 on the input, and on the second last layer (Srivastava et al., 2014).

For our semi-supervised setting, with the combined loss, we sample 1024 samples in our batch for each task from the corresponding dataset, and do a single weight update on the combined loss. A high-level schematic overview of our semi-supervised training setting is shown in Fig. 2.

4 Experimental Setup

4.1 Datasets

We conduct our experiments on the THYME corpus (Styler IV et al., 2014), a temporally annotated corpus of clinical notes in the colon cancer domain, also used in the Clinical TempEval Shared Tasks

(Bethard et al., 2015; Bethard et al., 2016). We use the provided Train, Dev and Test split so we can directly compare to other approaches from the literature. The Train section consists of 195 documents, with 11,2k annotated candidate relations, the Dev section of 98 documents and 6,2 annotated candidates, and the Test section contains 100 documents and 5,9k candidates. We now refer to this dataset as D^{rc}.

As data for the SG(LR) loss, we use the raw THYME train texts, extended with a MIMIC III (Johnson et al., 2016) section of 500 discharge summaries that contain the terms 'colon' and 'cancer' at least twice. We refer to this dataset as D^{sg} and it is used in all pre-training and joint training settings.

4.2 Training Settings

We compare five training settings for the model of which the first three settings act as baselines to compare with our proposed models (settings 4 and 5):

1. **RC (random initialization):** Uses random word embedding initializations (picked from [0.05, 0.05]) and trains on loss L_{rc}.
2. **RC (SG initialization):** Initializes the model with pre-trained SG embeddings, and trains on L_{rc}.
3. **RC (SG fixed):** Initializes the model with pre-trained SG embeddings, and trains the model on L_{rc}, while not updating the word embedding weights, keeping them as fixed features.
4. **RC + SG:** Initializes the model with pre-trained SG embeddings, and trains on L_{rc+sg}.
5. **RC + SGLR:** Initializes the model with pre-trained SG embeddings, and trains on $L_{rc+sglr}$.

4.3 Evaluation

As evaluation metrics we use precision, recall, and F-measure, calculated using the evaluation script provided by the Clinical TempEval organizers[2], which evaluates the CR under the temporal closure (UzZaman et al., 2012), taking into account transitivity properties of the temporal relations.

4.4 Preprocessing & Hyper-parameters

As preprocessing of the corpus, we employ very simple tokenization: splitting the text on spaces and considering punctuation[3] and newlines as individual tokens. Additionally, we lowercase the corpus, and conflate digits ($1992 \rightarrow 5555$). To extract POS we use the Stanford POS Tagger v3.7 (Toutanova et al., 2003), using the pre-trained (on WSJ) caseless left-3-words model. Finally, all 1-time occurring tokens in the training dataset are replaced by a <UNK>-token, to represent out-of-vocabulary words at test-time.

We employ the same candidate generation as Leeuwenberg and Moens (2017), considering all pairs of events (Event×Event, or EE) and events and temporal expressions (Timex3×Event, or TE) with a maximum token distance of 30 as candidate relations to be classified (ignoring sentence boundaries, as relations also occur across them). This candidate generation has a maximum recall of 0.87%, and gives a ratio between the positive and negative class of 1:36, also indicating the task's difficulty.

In our experiments, we tuned each model type within the same hyper-parameter search space on the Dev set. The number of LSTM units was chosen from $\{25, 50, 100\}$, and the word embedding dimension from $\{25, 50, 100\}$. This resulted in 100 LSTM units, and a word embedding size of 25. The loss weights λ_{sg} and λ_{sglr} were chosen from $\{0.01, 0.1, 1.0, 10, 100\}$, resulting in λ_{sg}=0.1 and λ_{sglr}=0.1. The context window size of the skip-gram was set to 2, chosen from $\{2, 4, 8\}$. The context size for the RC, and POS embedding dimension size were not tuned and set to 10 (left and right), and 40 respectively.

5 Results

5.1 Influence of Word-Level Loss

We looked at model performance when increasing the importance of the auxiliary word-level loss (λ_{sg} and λ_{sglr}). The results when changing these hyper-parameters are shown in Fig. 6.

[2] https://github.com/bethard/anaforatools

[3] , . / \ " ' = + − ; : ( ) ! ? < > % & $ * | [ ] { }

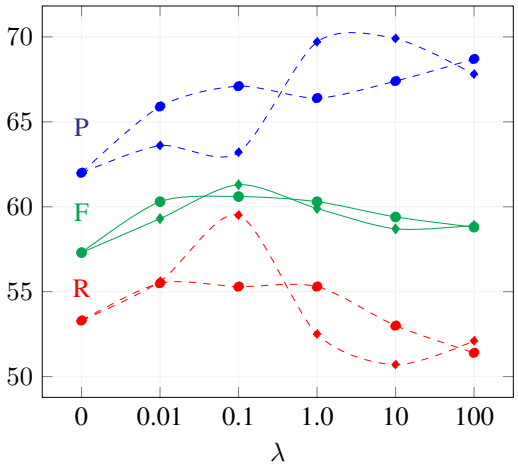


Figure 6: Precision (P), Recall (R) and F-measure (F) on the THYME Dev set for different values of λ_{sg} (○) and λ_{sglr} (◇).

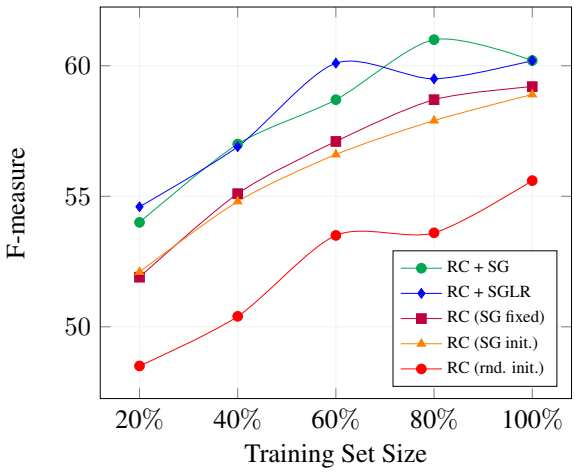


Figure 7: F-measure on the THYME Dev set for different training settings, over different training set sizes (in % of the full train set).

When choosing $\lambda = 0$ (λ_{sg} or λ_{sglr}) for our model, we obtain the same model as RC (SG init.), as the auxiliary objective has no influence. For very high values of λ, we hypothesize that the models converge towards RC (SG fixed), because when taking $\lambda \to \infty$, the word embeddings are solely optimized for the auxiliary loss, as the influence of the task loss is proportionally zero, i.e. $\lim_{\lambda \to \infty} \frac{1}{\lambda} = 0$. This property is interesting, as it shows these baseline models can act as lower bounds for our model performance when choosing a bad λ value. This can be observed in Fig. 6, where the F-measure is highest for a λ that balance both objectives, whereas for extremes F-measure decreases.

5.2 Comparison Across Training Set Size

We evaluated all model settings for different training set sizes. From Fig.7 we can see that the worst model is the one with random word embedding initializations. One improvement is to initialize the model with pre-trained SG embeddings. Fixing the pre-trained embeddings or continued training on the main task objective does not result in very different F-measure scores. However, continued training on the combined objective does seem to give a significant increase in F-measure, consistent over different training sizes for both the RC + SG as well as the RC + SGLR variant. Additionally, it should be noticed that parameters are not returned on each dataset size, but obtained from tuning on the full Dev set. Still the model ranking is consistent.

5.3 Evaluation on Subsets of Relations

To get a more detailed insight in what each model learns relative to the others, we evaluated our models on different subsets of the data. First, we split the containment relations based on their argument types and separately evaluated the 3.3k EE relations and the 2.7k TE relations. EE relations are generally found more difficult than TE relations (Lin et al., 2015; Lin et al., 2016; Dligach et al., 2017; Lin et al., 2017). In Table 1 we can see that also for our model, EE relations are harder to recognize than the TE relations, as all models achieve higher scores for TE compared to EE relations. What is interesting to see is that when training with the combined loss (SG or SGLR) we obtain a clear improvement on the more difficult EE relations, and perform slightly worse on TE relations compared to using pre-trained embeddings (the three upper settings). The reason could be that EE relations are more diverse in vocabulary, and are consequently more influenced by the quality of the embeddings.

We also analyzed the models w.r.t. total frequency in the training data ($D^{rc} + D^{sg}$) and made three subsets based on the average word frequency of the argument tokens in each relation. The three buckets of relations, 0-100, 100-500, and 500+, are of sizes 2.2k, 2.2k, and 1.8k respectively. What can be observed is that the RC+SG model performs best for low-frequency words, and RC+SGLR performs

Table 1: Evaluation on subsets of THYME Dev (in F-measure). The subsets of Event×Event (EE) and Timex3×Event (TE) relation pairs are of sizes 3.3k and 2.7k respectively. The intervals 0-100, 100-500 and 500+ are subsets reflecting average argument token frequency in the training data (of sizes 2.2k, 2.2k and 1.8k respectively).

Model	EE	TE	0-100	100-500	500+	All
RC (random initializations)	44.5	64.4	40.5	57.8	63.4	53.4
RC (SG initializations)	49.5	68.6	44.1	62.5	67.0	57.3
RC (SG fixed)	48.9	**68.7**	44.1	62.7	67.3	57.6
RC + SG	51.6	67.4	**46.4**	62.5	66.8	58.2
RC + SGLR	**51.7**	68.5	45.3	**63.0**	**68.1**	**58.4**

best for the higher frequency ranges. This can be explained by the fact that the SGLR separates left and right context words, creating sparser and more precise contexts compared to SG. Sparse context descriptions can hurt representations of low frequency words as there may not be enough words that share contexts. But, for more frequent words, more precise context descriptions as in SGLR help to prevents incorrect generalizations (such as cases where word order matters). When evaluating on the full Dev set, both combined loss settings outperform the baselines consistently.

5.4 Comparison to the State of the Art

We also compared our proposed models to various state-of-the-art systems from the literature:

The THYME system, by Lin et al. (2016), consist of separate models for EE relations and TE relations. They employ two feature rich support vector machines (SVM), using POS and dependency parse features from the cTAKES clinical pipeline (Savova et al., 2010) together with augmented training through extended UMLS entities. They later replaced the TE component by a token-based CNN model which improved their model (Lin et al., 2017; Dligach et al., 2017). Also replacing the EE component by a CNN model decreased model performance, showing that the CNN was not able to replace the feature rich SVM. Leeuwenberg and Moens (2017) used a feature rich structured perceptron, also using cTAKES POS and dependency parse features, jointly learning different relation types on the document level. Tourille et al. (2017) used two bidirectional LSTM models, one for inter-sentence and one for intra-sentence relations. They used fixed word embeddings pre-trained on the MIMIC III corpus, and also incorporated character level information. To obtain their top results they added ground truth event attribute features enhanced with entity information also obtained from cTAKES.

As can be noticed, all state-of-the-art baselines used dedicated clinical NLP tools to enhance their features in order to obtain their top results, in contrast to our model, which uses only the Stanford POS Tagger (trained on news texts).

Table 2 shows that initializing the model with the pre-trained embeddings gives a significant [4] 1.1 point increase in F-measure compared to random initialization, due to an increase in precision. Fixing the embeddings gives slightly better performance than using them as initialization, an increase of 0.9 point in F-measure, mostly due to higher recall. When extending the loss with the SGLR loss, we gain[6] 1.6 in F-measure compared to fixing the word embeddings, and also surpass the state of the art by 0.4 even without specialized resources. If we train our model using the SG loss extension we obtain the best results, and gain[6] 1.9 points in F-measure compared to using pre-trained fixed word embeddings. This setting also exceeds the state of the art (Lin et al., 2017) by 0.7 points in F-measure, due to a gain of 1.2 points in recall, again without using any specialized clinical NLP tools for feature engineering, in contrast to all state-of-the-art baselines.

5.5 Manual Error Analysis

Finally, we manually analyzed 50 false positives and 50 false negatives picked randomly from the test set predictions for different settings.

[4] $P < 0.0001$ for a document-level pairwise t-test

Table 2: THYME test set results, reporting precision (P), recall (R) and F-measure (F), macro-averaged over three runs. The standard deviation for F is also given.

Model	P	R	F
With specialized resources:			
Best Clinical TempEval (2016)	58.8	55.9	57.3
Lin et al. (2016)	66.9	53.4	59.4
Leeuwenberg et al. (2017)	-	-	60.8
Tourille et al. (2017)	65.7	57.5	61.3
Lin et al. (2017)	66.2	58.5	62.1
No specialized resources:			
RC (random initialization)	67.9	52.1	$58.9_{\pm 0.2}$
RC (SG initialization)	**71.2**	52.0	$60.0_{\pm 1.2}$
RC (SG fixed)	68.9	54.6	$60.9_{\pm 0.8}$
RC + SG	66.2	**59.7**	$\mathbf{62.8}_{\pm 0.2}$
RC + SGLR	68.7	57.5	$62.5_{\pm 0.3}$

From Table 3 we can see that all models have difficulties with distant relations that cross sentence or clause boundaries (CCR). This could be because class imbalance correlates with distance between the arguments of the temporal relations. Furthermore, arguments that are frequent in the supervised data (> 250) are a dominant error category. We suspect this is because frequent events often function both as container and as contained, whereas infrequent events are less ambiguous in their argument position. This hurts RC (SG fixed) most as its embeddings are not influenced by D^{rc}. Furthermore it can be noticed that RC+SG has less errors for infrequent arguments (< 10) in the supervised data. This could be because it leverages the few available instances from both the D^{rc} and D^{sg} data better than the single-loss models.

Table 3: Error analysis on 50 FP and 50 FN (random from test) for different settings. Clause boundaries are: newlines and sub-clause or sentence boundaries. Error categories are not mutually exclusive.

Error Type	RC + SG	RC (SG fixed)	RC (SG init.)
Cross-Clause Relations (CCR)	42	39	36
Infrequent Arguments (< 10)	11	15	26
Frequent Arguments (> 250)	37	50	40
Mistake in Ground-Truth	10	8	5
Other	21	15	28

6 Conclusions

In this work, we proposed a neural relation classification model for the extraction of narrative containment relations from clinical texts.[5]

The model trains word representations jointly on the supervised relation classification task and an unsupervised auxiliary skip-gram objective (with separate datasets) through weight sharing to more effectively exploit both the unlabeled and labeled data, as annotated clinical data is costly to create. We show that this word-level joint training results in significantly better generalizing classification models compared to using pre-trained word embeddings (either as initialization or fixed embeddings). Furthermore, we show that performance trends and good values for λ (balance between tasks) are robust over different training set sizes, and that even for (badly tuned) extreme values of λ the quality of the model's

[5]Code is available at: `http://liir.cs.kuleuven.be/software.html`

embeddings is naturally lower-bounded by their pre-trained variants. Additionally, our model sets a new state of the art for temporal relation extraction on the THYME dataset, without using extra dedicated clinical resources, in contrast to current state-of-the-art models.

As future work, it would be interesting to see how well the improvements caused by the word-level joint training generalize to other NLP tasks that typically use pre-trained word embeddings.

Acknowledgment

The authors would like to thank the reviewers for their constructive comments which helped us to improve the paper. Also, we would like to thank the Mayo Clinic for permission to use the THYME corpus. This work was funded by the KU Leuven C22/15/16 project "MAchine Reading of patient recordS (MARS)", and by the IWT-SBO 150056 project "ACquiring CrUcial Medical information Using LAnguage TEchnology" (ACCUMULATE).

References

Jonathan Baxter et al. 2000. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12(149-198):3.

Steven Bethard, Leon Derczynski, Guergana Savova, James Pustejovsky, and Marc Verhagen. 2015. SemEval-2015 task 6: Clinical TempEval. In *Proc. of SemEval*, pages 806–814. ACL.

Steven Bethard, Guergana Savova, Wei-Te Chen, Leon Derczynski, James Pustejovsky, and Marc Verhagen. 2016. SemEval-2016 task 12: Clinical TempEval. *Proc. of SemEval*, pages 1052–1062.

Steven Bethard, Guergana Savova, Martha Palmer, and James Pustejovsky. 2017. SemEval-2017 task 12: Clinical TempEval. In *Proc. of SemEval*, pages 565–572, Vancouver, Canada, August. ACL.

Rich Caruana. 1998. Multitask learning. In *Learning to Learn*, pages 95–133. Springer.

Hao Cheng, Hao Fang, and Mari Ostendorf. 2015. Open-domain name error detection using a multi-task RNN. In *Proc. of EMNLP*, pages 737–746.

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

Leon RA Derczynski. 2017. Automatically ordering events and times in text. In *Studies in Computational Intelligence*, volume 677. Springer.

Dmitriy Dligach, Timothy Miller, Chen Lin, Steven Bethard, and Guergana Savova. 2017. Neural temporal relation extraction. *Proc. of EACL*, page 746.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proc. of ACL*, pages 1723–1732. ACL.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. *Proc. of EMNLP*.

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

Min Jiang, Yang Huang, Jung-wei Fan, Buzhou Tang, Josh Denny, and Hua Xu. 2015. Parsing clinical text: how good are the state-of-the-art parsers? *BMC Medical Informatics and Decision Making*, 15(1):S2.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Hee-Jin Lee, Hua Xu, Jingqi Wang, Yaoyun Zhang, Sungrim Moon, Jun Xu, and Yonghui Wu. 2016. Uthealth at SemEval-2016 task 12: an end-to-end system for temporal information extraction from clinical notes. In *Proc. of SemEval*, pages 1292–1297. ACL.

Artuur Leeuwenberg and Marie-Francine Moens. 2017. Structured learning for temporal relation extraction from clinical records. In *Proc. of EACL*, pages 1150–1158. ACL.

Chen Lin, Dmitriy Dligach, Timothy A Miller, Steven Bethard, and Guergana K Savova. 2015. Multilayered temporal modeling for the clinical domain. *Journal of the American Medical Informatics Association*, 23(2):387–395.

Chen Lin, Timothy Miller, Dmitriy Dligach, Steven Bethard, and Guergana Savova. 2016. Improving temporal relation extraction with training instance augmentation. In *Proc. of ACL*, page 108. ACL.

Chen Lin, Timothy Miller, Dmitriy Dligach, Steven Bethard, and Guergana Savova. 2017. Representations of time expressions for temporal relation extraction with convolutional neural networks. In *Proc. of BioNLP*, page 322. ACL.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proc. of HLT-NAACL*, pages 1299–1304. ACL.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Nelson Morgan and Hervé Bourlard. 1990. Generalization and parameter estimation in feedforward nets: Some experiments. In *Advances in Neural Information Processing Systems*.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proc. of NAACL-HLT*, pages 39–48.

Agnieszka Onisko, Allan Tucker, and Marek J. Druzdzel. 2015. Prediction and prognosis of health and disease. In *Foundations of Biomedical Knowledge Representation: Methods and Applications*, pages 181–188. Springer.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 548–554.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proc. of LREC*, pages 2089–2096. European Language Resources Association (ELRA), May.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Guergana K Savova, James J Masanz, Philip V Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C Kipper-Schuler, and Christopher G Chute. 2010. Mayo clinical text analysis and knowledge extraction system (cTAKES): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association*, 17(5):507–513.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proc. of ACL*, volume 2, pages 231–235. ACL.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

William F Styler IV, Steven Bethard, Sean Finan, Martha Palmer, Sameer Pradhan, Piet C de Groen, Brad Erickson, Timothy Miller, Chen Lin, Guergana Savova, et al. 2014. Temporal annotation in the clinical domain. *Transactions of the Association for Computational Linguistics*, 2:143–154.

Weiyi Sun, Anna Rumshisky, and Ozlem Uzuner. 2013. Evaluating temporal relations in clinical text: 2012 i2b2 challenge. *Journal of the American Medical Informatics Association*, 20(5):806–813.

Julien Tourille, Olivier Ferret, Aurelie Neveol, and Xavier Tannier. 2017. Neural architecture for temporal relation extraction: A Bi-LSTM approach for detecting narrative containers. In *Proc. of ACL*, pages 224–230, Vancouver, Canada, July. ACL, ACL.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of NAACL-HLT*, pages 173–180. ACL.

Naushad UzZaman, Hector Llorens, James Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. 2012. Tempeval-3: Evaluating events, time expressions, and temporal relations. *arXiv preprint arXiv:1206.5333*.

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *Proc. of EMNLP*, pages 236–246. ACL, November.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *Proc. of COLING*, pages 2335–2344. ACL.

Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network. *arXiv preprint arXiv:1508.01006*.

Shu Zhang, Dequan Zheng, Xinchen Hu, and Ming Yang. 2015. Bidirectional long short-term memory networks for relation classification. In *Proc. of PACLIC*.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proc. of ACL*.

Personalized Text Retrieval for Learners of Chinese as a Foreign Language

Chak Yan Yeung and **John Lee**
Department of Linguistics and Translation
City University of Hong Kong
chak.yeung@my.cityu.edu.hk, jsylee@cityu.edu.hk

Abstract

This paper describes a personalized text retrieval algorithm that helps language learners select the most suitable reading material in terms of vocabulary complexity. The user first rates their knowledge of a small set of words, chosen by a graph-based active learning model. The system trains a complex word identification model on this set, and then applies the model to find texts that contain the desired proportion of new, challenging, and familiar vocabulary. In an evaluation on learners of Chinese as a foreign language, we show that this algorithm is effective in identifying simpler texts for low-proficiency learners, and more challenging ones for high-proficiency learners.

1 Introduction

Since extensive reading is beneficial for learning a foreign language, students are encouraged to read a wide range of texts, beyond their textbooks and graded readers. Reading materials for language learning need to be carefully selected. On the one hand, if the text is too difficult, it would hinder comprehension and leave the student overwhelmed and discouraged. On the other hand, if the text is too easy, it would not serve to stretch the student's language skills. The best material should be challenging yet highly readable, containing just the right proportion of new, challenging, and familiar vocabulary. Graded readers created with specific language proficiency levels in mind alleviate this problem to some extent, but they provide only a limited amount of materials, and the grade levels are still coarse-grained. It often falls to language teachers to identify appropriate material, but they cannot do so for every individual student.

This paper describes a personalized text retrieval algorithm that helps language learners select the most suitable reading material in terms of vocabulary complexity. The user first rates their knowledge of a small set of words, chosen by a graph-based active learning model. The system trains a complex word identification model (CWI) on this set. Most current CWI approaches assume vocabulary knowledge to be a dichotomy, labeling each word either as "non-complex", if the user is familiar with the word, or as "complex", if the user is not familiar with it (Paetzold and Specia, 2016; Ehara et al., 2014). There are, however, many dimensions of vocabulary knowledge (Richards, 1976). As our first contribution, we take one step in this direction by attempting to identify not only known and unknown words, but also challenging words.

Further, we apply CWI to the text retrieval task, and measure its effect on the vocabulary complexity of retrieved texts from the perspective of a language learner. Specifically, the algorithm attempts to find texts with a minimum proportion of non-complex words and a maximum number of challenging words for the learner. As our second contribution, we show that the algorithm is effective in retrieving simpler texts for low-proficiency learners, and more difficult ones for high-proficiency learners.

The rest of this paper is organized as follows. The next section summarizes previous research in automatic CWI and text retrieval for language learners. Section 3 outlines our approach. Section 4 describes our dataset. Section 5 presents our experimental results. Section 6 concludes the paper.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3448–3455
Santa Fe, New Mexico, USA, August 20-26, 2018.

2 Previous work

2.1 Complex word identification

The complex word identification (CWI) task aims to tag a word as "complex" if it is unfamiliar for a user, and "non-complex" otherwise. In the lexical simplification task, CWI is commonly used for identifying words that should be simplified to improve a user's understanding of a text while avoiding unnecessary simplification (Shardlow, 2013). CWI is also an important part in systems that aim to provide suitable reading material for users. Elhadad (2006), for example, built a system to predict terms in medical literature that were unlikely to be understood by layman readers and provide definitions. This work uses CWI to find texts suitable for learners of Chinese as a foreign language by estimating the proportion of complex, challenging, and non-complex words in a text.

Most previous research has focused on CWI for English as a foreign language. One approach for predicting vocabulary knowledge is by "word sampling" (Laufer and Nation, 1999). Using a ten-level proficiency scale, with 1000 words at each level, this approach samples a fixed number of words from each level and estimates the size of the learner's vocabulary from his/her knowledge of the sampled words.

Word frequencies were found to be the most reliable predictor of word complexity in the 2016 SemEval shared task for CWI in English (Paetzold and Specia, 2016). The best team, which combined various lexicon-based, threshold-based and machine learning voter subsystems, achieved a precision of 0.147 and recall of 0.769. This approach has yet to be evaluated on language learners at different levels of proficiency, since the entire test set was annotated by one learner.

Personalized models for CWI adjust their predictions based on user characteristics. Zeng et al. (2005) showed that demographic features can help improve CWI performance for individual users in the medical domain. Ehara et al. (2012; 2014) performed CWI for individual learners with a two-step approach. In the first step, an active learning approach selects the k most informative nodes, or words, to be annotated by the user (see Section 3.1 for details). The user rates their knowledge of these k words on a five-point scale (Table 1). In the second step, the system uses Local and Global Consistency (Zhou et al., 2004), a label propagation algorithm, to train an independent classifier for each user. The algorithm performs binary classification on the nodes to predict whether a user knows a word or not. The assumption behind this algorithm is that two nodes connected by a heavily weighted edge should have similar labels, and more heavily weighted edges should propagate more labels. The best model, where k=50, achieved 76.44% accuracy on a dataset of Japanese learners of English.

Score	Description	Label (Ehara)	Label (this study)
1	Never seen the word before	complex	complex
2	Probably seen the word before	complex	complex
3	Absolutely seen the word before but do not know its meaning, or tried to learn the word before but forgot its meaning	complex	complex
4	Probably know, or able to guess, the word's meaning	complex	challenging
5	Absolutely know the word's meaning	non-complex	non-complex

Table 1: Five-point scale for rating vocabulary knowledge, and the mapping to CWI labels used by Ehara et al. (2012; 2014) and in this study.

Recently, CWI for other languages has begun to receive more attention. The latest CWI shared task, for example, featured multilingual datasets, covering English, German, Spanish, and French (Yimam et al., 2018). To the best of our knowledge, there has so far been only one reported study on CWI for Chinese as a foreign language (CFL) (Lee and Yeung, 2018). They followed Ehara et al. (2012; 2014) in selecting 50 most informative words to be rated by CFL learners. They then used a support vector machine (SVM) classifier to train a CWI classifier for each individual learner based on his/her 50-word

training set. The classifier used features based on word difficulty levels in two assessment scales for CFL, namely the Test of Chinese as a Foreign Language (TOCFL) (Zeng, 2014) and the Hanyu Shuiping Kaoshi (HSK) (Hanban, 2014). Details on the features are provided in Section 3.2. On a dataset with seven CFL learners, this approach achieved 78% CWI accuracy, outperforming the label propagation method. In this study, we adopt this classification approach and extend it to three-way, so that we may use not just the proportion of complex and non-complex words, but also challenging words, as criteria for text retrieval.

2.2 Text retrieval for language learners

Past research has shown that the optimal reading material should fit the learner in terms of vocabulary and grammatical complexity, as well as personal interests (Heilman et al., 2007). In this study, we focus on the proportion of non-complex words in a text.

Several studies have been conducted to examine the proportion of words in a text that a learner needs to know to facilitate reading comprehension. There is no consensus on the "optimal" proportion of known words in a text. Laufer and Nation (1999), Liu and Nation (1985) and Hirsh et al. (1992) showed that a student could read and understand a text with ease if 95% to 98% of the words were known. If the goal is to stretch the student's vocabulary, the threshold can be made lower. Indeed, in the first 5 books of the *New Practical Chinese Reader*, a popular textbook series for Chinese as a foreign language (CFL), the proportion of difficult words ranges from 9% to 31% (Liang and Song, 2009).

3 Approach

We propose a text retrieval algorithm that aims to help language learners select the most suitable reading material in terms of vocabulary complexity. The algorithm has three components. First, it employs a graph-based active learning model to select a small set of words as training set (Section 3.1). After the user has rated his/her knowledge of these words, the system trains a complex word identification model on this set (Section 3.2). It then applies the model to find texts that contain the desired proportion of complex, challenging, and non-complex words (Section 3.3). We now provide details on each of these components.

3.1 Training set creation

The system performs graph-based active learning to select a small number of words to be included in the training set. The entire vocabulary is first organized as a multiple complete graph. Nodes correspond to words, and edge weights reflect the similarity between the frequency ranks of the words. The assumption is that words with similar frequency ranks are known to learners whose familiar words are similar to each other. Using Error Bound Minimization (Gu and Han, 2012), a non-interactive graph-based active learning algorithm, the system selects the k most informative nodes from the vocabulary graph in a non-interactive way. This algorithm selects nodes that are globally important, based on the number of edges. Further, the nodes must not be heavily connected to previously sampled nodes. Following previous work (Lee and Yeung, 2018; Ehara et al., 2014), we set k as 50. While a larger k can yield better performance, burdening users with a large amount of vocabulary annotation would be undesirable in practice.

3.2 Complex word identification

The CFL learners scored their knowledge on the 50 words in the training set with the five-point scale in Table 1. Following Ehara et al. (2014) and Lee and Yeung (2018), we asked the learners to score their knowledge of words in isolation, rather than in context such as in the CWI shared task.

The context of a word provides clues for learners to guess its meaning, and thus affects how they score their knowledge of a word. Even if the learner is able to guess a word in one context, the same is not guaranteed in another context since the content and the density of new words in each text is different. Since the CWI model in our system is intended for text selection, we did not wish to assume any one particular context when determining the learner's knowledge of a word. To more accurately judge the learners' ability to understand different reading materials, we asked them to consider each word in isolation.

For each individual user, we trained a classifier based on his/her annotation on the 50-word training set to perform CWI on all Chinese words. In a departure from previous CWI studies, we attempted three-way classification, labelling each word as "non-complex", "challenging" or "complex" (see Table 1). We experimented with a feature set similar to that of Lee and Yeung (2018):

- **WordList**: The difficulty level of the word (1, 2, 3, 4, 5, or 6), according to the HSK and TOCFL guidelines. HSK was used as the basis for this feature. For words not covered by HSK, we mapped their TOCFL level to HSK.
- **HSK-char-max**: A Chinese word may contain multiple characters. This feature takes the maximum level among the characters in the word (1, 2, 3, 4, 5, or 6), according to the HSK guidelines.
- **Freq**: The frequency of the word in Chinese Wikipedia.
- **LearnerFreq**: The frequency of the word in the *Jinan Corpus of Learner Chinese* (JCLC) (Wang et al., 2015).
- **LearnerFreq-char-min**: For each character in the word, we compute its frequency count in JCLC. This feature takes the minimum frequency.

3.3 Text retrieval

Our system allows the user to specify the minimum (estimated) proportion of words in the retrieved text that he/she should be able to understand. This parameter can be set to an arbitrary percentage between 0 and 100, but in consideration of the manual effort needed for evaluation, we set this parameter to 80% in our experiment.

Ehara et al. (2012) assumed that learners could only understand the words that received a score of 5 on the five-point scale in Table 1, and that they did not know any word with a score of 4 or below. The aim of our system, however, is to stretch the student's language skills. Taking lexical guessing in context into account, we consider both non-complex and challenging words, scored five and four on the five-point scale respectively, as words that the user should be able to understand.

To stretch the learner's language skills while maintaining a high level of comprehension, the system retrieves all documents that satisfy the minimum threshold of 80% non-complex words, and then ranks them from the highest proportion of challenging words to the lowest.

4 Datasets

4.1 Complex word identification

We used the same training set of 50 words and test set of 550 words as Lee and Yeung (2018). We retained the original annotations of the seven CFL learners, and included the annotations of an additional learner. The learners labelled each word in the datasets on a five-point scale (see Table 1). We considered a word to be "non-complex" if it was scored five, "challenging" if it was scored four, and "complex" otherwise.

The numbers of "non-complex", "challenging", and "complex" words of the test set and training set for each learner are shown in Table 2. We divided the eight learners into two groups. The four who knew less than 150 of the 550 words were designated as "Low-Proficiency", and the other four as "High-Proficiency".

4.2 Text retrieval

We compiled a collection of texts from Chinese Wikipedia, Chinese Internet Corpus (Sharoff, 2006), and *Duan Mei Wen*[1]. For each text, we extracted the first n sentences such that the total character count was between 200 to 225. A total of 38,881 texts were included as candidates for the text retrieval algorithms.

We evaluated three text retrieval methods. Our baseline method randomly selects texts. The other two methods use the two CWI models that achieved the best performance on low-proficiency learners and

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

User	Training Set			Test Set		
	non-complex	challenging	complex	non-complex	challenging	complex
1	11	2	37	68	25	457
2	7	9	34	78	91	381
3	22	10	18	296	38	216
4	14	8	28	113	114	323
5	17	8	25	217	116	217
6	13	15	22	188	109	253
7	14	8	28	146	75	329
8	20	13	17	268	135	147

Table 2: The numbers of non-complex, challenging, and complex words in the training sets and test sets.

high-proficiency learners (Table 3), respectively, coupled with the retrieval criteria described in Section 3.3. We retrieved the top five texts for each of these three methods.

We asked four of the eight learners — two low-proficiency learners and two high-proficiency learners — to annotate the fifteen retrieved texts with the same five-point scale used in the CWI experiment. They were asked to highlight the complex words in red and the challenging words in blue.

5 Experimental results

5.1 Complex word identification

Following Ehara et al. (2014) and Lee and Yeung (2018), we trained SVM classifiers and logistic regression (LR) classifiers. We used the implementation in scikit-learn (Pedregosa et al., 2011), and we tried all combinations of the features listed in Section 3.1. The results are shown in Table 3.

Features	Model	All	Low-proficiency	High-proficiency
WordList	SVM	64.18%	**70.96%**	57.41%
WordList	LR	63.66%	70.41%	56.91%
WordList + HSK-char-max	SVM	63.57%	69.68%	57.46%
WordList + HSK-char-max	LR	58.82%	65.46%	52.18%
Freq	SVM	40.52%	36.82%	44.23%
Freq	LR	56.23%	62.82%	49.64%
LearnerFreq	SVM	57.02%	64.50%	49.55%
LearnerFreq	LR	62.93%	70.09%	55.78%
LearnerFreq + Freq	SVM	51.11%	60.64%	41.59%
LearnerFreq + Freq	LR	60.64%	66.73%	54.55%
LearnerFreq + LearnerFreq-char-min	SVM	43.14%	50.46%	35.82%
LearnerFreq + LearnerFreq-char-min	LR	64.27%	69.87%	58.68%
LearnerFreq + LearnerFreq-char-min + WordList	SVM	54.61%	56.36%	52.86%
LearnerFreq + LearnerFreq-char-min + WordList	LR	**64.68%**	70.59%	**58.77%**

Table 3: The average accuracies of 3-way complex word identification for all eight learners, low-proficiency learners, and high-proficiency learners.

The LR model with LearnerFreq, LearnerFreq-char-min, and WordList features achieved the highest accuracy, at 64.68%. It slightly outperformed the LR model with LearnerFreq and LearnerFreq-char-min features, at 64.27%, and the SVM model with WordList features, at 64.18%. In general, CWI performance is better on low-proficiency learners than on high-proficiency learners.

The WordList features seem particularly useful for capturing the limited vocabulary of beginners, with the word lists covering most of the words that they know. This hypothesis is consistent with the fact that the SVM model with WordList features achieved the best performance for the low-proficiency learners, at 70.96%. For high-proficiency learners, however, the word lists are insufficient in modelling their larger vocabularies, and have to be augmented with frequency features in order to cover a wider range of words. The LR model with LearnerFreq, LearnerFreq-char-min, and WordList features achieved the highest accuracy for high-proficiency learners, at 58.77%.

5.2 Text retrieval

We evaluated three text retrieval methods:

- **WordList**: The SVM model with WordList features, which achieved the best CWI results for low-proficiency learners.
- **Learner + WordList**: The LR model with LearnerFreq, LearnerFreq-char-min, and WordList features, which achieved the best CWI results for high-proficiency learners,
- **Baseline**: Random selection.

Experimental results for high- and low-proficiency learners are shown in Tables 4 and 5, respectively. To prevent outliers from skewing the results, we removed the texts with the highest and the lowest proportion of non-complex words for each user.

Model	Non-complex words	Challenging words
WordList	**90.18%** (estimated: 85.06%)	7.84% (estimated: 18.13%)
Learner + WordList	**87.87%** (estimated: 84.62%)	6.13% (estimated: 23.39%)
Random selection	95.36%	4.82%

Table 4: The average proportion of non-complex and challenging words in text retrieved for high-proficiency learners.

Model	Non-complex words	Challenging words
WordList	**80.60%** (estimated: 82.66%)	5.91% (estimated: 38.24%)
Learner + WordList	**86.26%** (estimated: 83.91%)	3.76% (estimated: 20.45%)
Random selection	72.91%	6.09%

Table 5: The average proportion of non-complex and challenging words in text retrieved for low-proficiency learners.

For high-proficiency learners, randomly selected texts were relatively easy to read. This can be seen in the high proportion of non-complex words (95.36%) in texts returned by the baseline method. In comparison, the WordList model and Learner+WordList model both retrieved more difficult texts, with 90.18% and 87.87% of non-complex words, respectively. Thus, these models were able to find texts that presented more new vocabulary to advanced learners, while still exceeding the minimum comprehension rate of 80

For low-proficiency learners, experimental results showed the opposite effect. The baseline method retrieved rather difficult texts, in which only 72.91% of the words were non-complex, below the desired minimum rate of 80%. In contrast, the WordList model and Learner+WordList model retrieved texts with 80.60% and 86.26% of non-complex words, respectively. Thus, they succeeded in finding more readable texts, with above 80% comprehension rate, for the beginners.

For both the WordList model and the Learner+WordList model, the proportion of challenging words in the retrieved texts was lower than estimated. As shown in Table 6, among the words estimated to be challenging by the two best CWI models, most (63.11% and 75.15%, respectively) were scored as

non-complex by the learners. The discrepancy may be attributable to the fact that learners, with the benefit of context while reading a text, can better guess the meaning of a word that would otherwise be challenging. This phenomenon can be even more clearly seen by comparing the scores on the same words by the same learner in the CWI dataset (Section 4.1) and the text retrieval dataset (Section 4.2). Indeed, among the words that were annotated as challenging in isolation in the CWI dataset, 68% were annotated as non-complex in the text retrieval dataset by the same learners. Similarly, among the words that were annotated as complex in the CWI dataset, 36.67% were annotated as non-complex in the text retrieval dataset.

We conducted further analysis on the kinds of words that learners are more likely to find non-complex when reading a text. We found that "challenging" or "complex" single-character words (e.g. 以 *yǐ* (with), 称 *chēng* (named)) were more likely to be considered "non-complex" in context. Words related to the theme of the texts (e.g. 口音 *kǒu yīn* (accent) in a text about dialects and 政治 *zhèng zhì* (politic) in a text about democracy) were also more likely to be changed from "challenging" or "complex" to "non-complex".

Gold label	All	Low-proficiency	High-proficiency
WordList			
Complex	26.99%	32.16%	21.83%
Challenging	9.90%	6.04%	13.76%
Non-complex	63.11%	61.79%	64.42%
Learner + WordList			
Complex	17.44%	14.83%	20.06%
Challenging	7.41%	2.51%	12.30%
Non-complex	75.15%	82.67%	67.64%

Table 6: Gold label for words estimated to be challenging in the text retrieval dataset (Section 4.2).

6 Conclusion

Reading materials for language learning need to be carefully selected. The best material should be challenging yet highly readable, containing just the right proportion of new and challenging vocabulary. This paper describes a personalized text retrieval algorithm that aims to help language learners select the most suitable reading material in terms of vocabulary complexity.

The system first employs a graph-based active learning model to select a small set of informative words. Each user rates his/her knowledge of these words to create a training set for complex word identification (CWI). The system then trains a personalized CWI model, and applies it to find texts that contain the desired proportion of complex, challenging, and non-complex words. We evaluated this algorithm on learners of Chinese as a foreign language, and showed its effectiveness in identifying easier texts for low-proficiency learners, and more challenging ones for high-proficiency learners.

Acknowledgements

This work is funded by the Language Fund under Research and Development Projects 2015-2016 of the Standing Committee on Language Education and Research (SCOLAR), Hong Kong SAR. We thank Dr. Lis Pereira for assisting with the experiments.

References

Yo Ehara, Issei Sato, Hidekazu Oiwa, and Hiroshi Nakagawa. 2012. Mining words in the minds of second language learners: learner-specific word difficulty. *Proceedings of COLING 2012*, pages 799–814.

Yo Ehara, Yusuke Miyao, Hidekazu Oiwa, Issei Sato, and Hiroshi Nakagawa. 2014. Formalizing word sampling for vocabulary prediction as graph-based active learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1374–1384.

Noemie Elhadad. 2006. Comprehending technical texts: Predicting and defining unfamiliar terms. In *AMIA annual symposium proceedings*, volume 2006, page 239. American Medical Informatics Association.

Quanquan Gu and Jiawei Han. 2012. Towards active learning on graphs: An error bound minimization approach. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 882–887. IEEE.

Hanban. 2014. *International curriculum for Chinese language education*. Beijing Language and Culture University Press, Beijing, China.

Michael Heilman, Kevyn Collins-Thompson, Jamie Callan, and Maxine Eskenazi. 2007. Combining lexical and grammatical features to improve readability measures for first and second language texts. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 460–467.

David Hirsh, Paul Nation, et al. 1992. What vocabulary size is needed to read unsimplified texts for pleasure? *Reading in a foreign language*, 8:689–689.

Batia Laufer and Paul Nation. 1999. A vocabulary-size test of controlled productive ability. *Language testing*, 16(1):33–51.

John Lee and Chak Yan Yeung. 2018. Automatic prediction of vocabulary knowledge for learners of chinese as a foreign language. In *Proceedings of the 2nd International Conference on Natural Language and Speech Processing (ICNLSP)*.

Ji-hua Liang and Shao-li Song. 2009. Construction of an approach for counting chinese graded words and characters——a tool for assessing difficulty level of word in chinese language teaching materials writing system. *Modern Educational Technology*, 7:024.

Na Liu and ISP Nation. 1985. Factors affecting guessing vocabulary in context. *RELC journal*, 16(1):33–42.

Gustavo Paetzold and Lucia Specia. 2016. Semeval 2016 task 11: Complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Jack C Richards. 1976. The role of vocabulary teaching. *TESOL quarterly*, pages 77–89.

Matthew Shardlow. 2013. A comparison of techniques to automatically identify complex words. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 103–109.

Serge Sharoff. 2006. Creating general-purpose corpora using automated search engine queries. *WaCky*, pages 63–98.

Maolin Wang, Shervin Malmasi, and Mingxuan Huang. 2015. The jinan chinese learner corpus. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 118–123.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo H Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A report on the complex word identification shared task 2018. *arXiv preprint arXiv:1804.09132*.

Qing Zeng, Eunjung Kim, Jon Crowell, and Tony Tse. 2005. A text corpora-based estimation of the familiarity of health terminology. In *International Symposium on Biological and Medical Data Analysis*, pages 184–192. Springer.

W. Zeng. 2014. Huayu baqianci ciliang fenji yanjiu (classification on chinese 8000 vocabulary). *Huayu Xuekan*, pages 6: 22–33.

Denny Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. In *Advances in neural information processing systems*, pages 321–328.

Punctuation as Native Language Interference

Ilia Markov
INRIA
Paris, France
ilia.markov@inria.fr

Vivi Nastase
University of Heidelberg
Heidelberg, Germany
nastase@
cl.uni-heidelberg.de

Carlo Strapparava
Fondazione Bruno Kessler
Trento, Italy
strappa@fbk.eu

Abstract

In this paper, we describe experiments designed to explore and evaluate the impact of punctuation marks on the task of native language identification. Punctuation is specific to each language, and is part of the indicators that overtly represent the manner in which each language organizes and conveys information. Our experiments are organized in various set-ups: the usual multi-class classification for individual languages, also considering classification by language groups, across different proficiency levels, topics and even cross-corpus. The results support our hypothesis that punctuation marks are persistent and robust indicators of the native language of the author, which do not diminish in influence even when a high proficiency level in a non-native language is achieved.

1 Introduction

Native Language Identification (NLI) – identifying the native language (L1) of a person based on his/her writing in the second language (L2) – is useful for a variety of purposes, including security, marketing, and educational applications. The effect of native language phenomena seeping into texts produced in a different language is known as language transfer (Odlin, 1989). Numerous aspects of the language have been explored for NLI – character-level language models (Ionescu et al., 2014), lexical choice (Brooke and Hirst, 2012; Lahiri and Mihalcea, 2013), grammar (Nagata and Whittaker, 2013), spelling errors (Chen et al., 2017; Koppel et al., 2005), cognates (Nicolai et al., 2013), and general etymology (Nastase and Strapparava, 2017).

While punctuation has been included in some of these studies (e.g., in character-level models), its impact has not been studied. It is however an important, and often revealing, aspect of written language. For example, punctuation is a strong indicator of authorship, and has been used successfully in stylometric analysis for authorship attribution (Markov et al., 2017b; Grieve, 2007). More generally, from a linguistic point of view, punctuation has been disputed as following prosodic principles or as a clarifier of grammatical structure (Baron, 2001; Bruthiaux, 1993). Moore (2016) finds a common ground for these two views by observing that prosody and punctuation realize the same function – revealing/emphasizing the information structure of an utterance – in the spoken and respectively written modes of language. Since grammar and prosodic structure are language specific, indicators that reveal them would be language specific as well. As with other aspects of language, grammatical/prosodic influences from the native language may surface in the new language as particular punctuation choices. As an example, consider the following English sentence, written by a native German speaker[1]:

I think the biggest question is , how to defin an " enjoyed life " .

A native English speaker would not insert a comma between *is* and *how*, but it reflects correct punctuation usage in German:

[1]Extracted from one of the training essays in the data we work with (NLI: 10086.txt), with the author's misspelling of 'define'.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3456–3466
Santa Fe, New Mexico, USA, August 20-26, 2018.

Ich denke, die größte Frage ist, wie man ein "glückliches Leben" definiert.

We propose the hypothesis that punctuation usage from the native language appears in a speaker's new language, and is distinctive enough to contribute to the native language identification task. To investigate this hypothesis we perform a series of experiments that measure the impact of punctuation on the NLI task, which we approach from a machine-learning perspective, as a multi-class classification problem of English (as L2) written documents, using two representations – word n-grams and part-of-speech (POS) tags – and analyze the performance of these feature sets when they include or not punctuation marks (PMs). We perform one-step classification – classify the L1 of the author directly – and also a two-step classification where in the first step we classify the language family/geographical group, and then the actual L1. This experiment shows that there are commonalities across language families, and also particularities that distinguish punctuation usage for specific languages within a family/group. We evaluate the use of punctuation within different proficiency levels, to test whether with better L2 skills, the speakers also adopt punctuation usage closer to a native speaker's. Surprisingly, the results indicate that punctuation usage remains influenced by L1 even for learners with high proficiency in L2.

In the previously mentioned experiments we focused on more abstract features (POS tags), which we combine with punctuation marks to capture a higher-level representation of punctuation usage. However, BoW/word n-gram features are the ones that lead to the best results on the task, even though they have been disputed as being useful (Brooke and Hirst, 2012), but potentially overfitting (Brooke and Hirst, 2011). To test whether punctuation marks are robust and can compensate some shortcomings of lexical features, we perform a final set of experiments in which we produce word and punctuation n-grams and test them in cross-dataset classification (training and testing on different datasets). The high drop in performance when using word n-gram features – consistent with the hypothesis that this representation leads to models that overfit the training data – is partly countered when punctuation marks are added to the mix.

The consistent and substantial improvement in native language identification brought by including punctuation marks in the different set-ups explored indicates that punctuation usage is a robust and persistent indicator of the native language of the author.

In a nutshell, the research questions addressed in this work are the following: (i) evaluate the contribution of PMs to NLI, (ii) examine how robust and persistent their contribution is with respect to levels of proficiency and topic/corpus variation.

2 Related Work

Numerous aspects of language, possibly all, can insinuate themselves from a language learner's native language to the targeted L2.

Numerous studies (Brooke and Hirst, 2012; Lahiri and Mihalcea, 2013; Tsur and Rappoport, 2007) show that the *lexical choices* of non-native speakers are strong indicators of their native language, and so are the spelling errors (Chen et al., 2017; Koppel et al., 2005). Lexical features and character n-grams (Ionescu et al., 2014) are considered the most indicative feature types for the NLI task. Nicolai et al. (2013) and Nastase and Strapparava (2017) find that lexical choice is partly influenced by cognates or more generally, by etymologically related ancestor languages.

Both Wong and Dras (2009) and Nagata and Whittaker (2013) investigate the influence of *grammar* as grammatical structures transfer from L1 to L2, whether they are legitimate patterns in L2 as well (Nagata and Whittaker, 2013), or are erroneous with respect to L2 (Wong and Dras, 2009).

The interest in the NLI task led to the organization of several NLI shared tasks (Tetreault et al., 2013; Malmasi et al., 2017), where participating teams used a variety of features: character n-grams of different types, n-grams of lexical features (words, lemmas, POS tags), grammatical information (parse tree rules, syntactic dependency-based n-grams), spelling errors, function words, among others. The top two teams in the recent edition of the NLI shared task (Cimino and Dell'Orletta, 2017; Markov et al., 2017a) used a combination of these features, achieving accuracy close to 90% for this task.

As linguistic studies show (Moore, 2016), punctuation is an important aspect of language. For the written mode of language, it serves to reveal/emphasize the information structure of a sentence, partly

	English proficiency					
L1	**Low**		**Medium**		**High**	
Arabic	296	26.9%	605	55.0%	*199*	18.1%
Chinese	*98*	8.9%	727	66.1%	275	25.0%
French	*63*	5.7%	577	52.5%	460	41.8%
German	*15*	1.4%	412	37.5%	673	61.2%
Hindi	*29*	2.6%	429	39.0%	642	58.4%
Italian	*164*	14.9%	623	56.6%	313	28.5%
Japanese	233	21.2%	679	61.7%	*188*	17.1%
Korean	*169*	15.4%	678	61.6%	253	23.0%
Spanish	*79*	7.2%	563	51.2%	458	41.6%
Telugu	*94*	8.5%	659	59.9%	347	31.5%
Turkish	*90*	8.2%	616	56.0%	394	35.8%
Total	1,330	11.0%	6,568	54.3%	4,202	34.7%

Table 1: Data statistics for the three English proficiency levels in TOEFL11.

	Number of essays per prompt							
L1	**P0**	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**
Arabic	139	133	141	138	138	136	138	137
Chinese	140	141	139	139	140	134	126	141
French	156	68	160	151	158	160	87	160
German	151	28	153	152	155	150	157	154
Hindi	86	53	161	158	161	156	163	162
Italian	187	12	141	173	173	187	138	89
Japanese	138	142	143	141	116	138	140	142
Korean	128	142	143	141	140	137	136	133
Spanish	159	157	162	160	141	134	54	133
Telugu	55	41	171	166	165	169	167	166
Turkish	170	43	169	167	169	147	90	145
Total	1,509	960	1,683	1,686	1,656	1,648	1,396	1,562

Table 2: Distribution of topics in TOEFL11.

sharing the sentence structuring function of grammar. In this paper, we propose to investigate punctuation usage as a source of native language information, and understand to what degree it is reflected in writing in a new language. We do this through a suite of experiments described in Sections 4 and 5.

3 Data

3.1 Datasets

To investigate the impact of the punctuation usage on native language identification (NLI), we conducted experiments on two datasets commonly used in NLI research:

TOEFL11 (Blanchard et al., 2013): the ETS Corpus of Non-Native Written English (TOEFL11) contains 1,100 essays in English (with an average of 348 tokens per essay) for each of the following 11 native languages: Arabic, Chinese, French, German, Hindi, Italian, Japanese, Korean, Spanish, Telugu, and Turkish. The essays were written in response to eight different writing prompts/topics (P0–P7), all of which appear in all 11 L1 groups. The dataset also contains information regarding the proficiency level (low, medium, high) of the essay authors. Dataset statistics in terms of proficiency levels and writing prompts are presented in Tables 1 and 2, respectively.

ICLEv2 (Granger et al., 2009): the ICLEv2 dataset (henceforth, ICLE) consists of essays written by highly-proficient non-native college-level students of English. We used a 7-language subset of the corpus normalized for topic and character encoding (Tetreault et al., 2012; Ionescu et al., 2014)[2]. This subset contains 110 essays (with an average of 747 tokens per essay after tokenization and removal of metadata) for each of the 7 languages: Bulgarian, Chinese, Czech, French, Japanese, Russian, and Spanish.

[2]The authors express their gratitude to A. Cahill for providing the list of documents used in their paper.

3.2 Features

The suite of experiments we report were designed to investigate the impact of punctuation-based features on native language identification. The hypothesis we are testing is whether patterns of punctuation usage – possibly motivated by prosody or grammatical structure in the native language – are indicative of the native language of the author of written essays in English. Lexicalized representation of documents (through word/character n-grams of different sizes) have been criticized as overfitting the data (Brooke and Hirst, 2012). Because of this we use more abstract POS n-gram features to represent the documents (essays) in our data, to which we add punctuation marks to capture punctuation usage in the analyzed texts.

Part-of-speech (POS) tags POS features capture the morpho-syntactic patterns in a text. They are considered indicative features for NLI, especially when used in combination with other features, such as word and character n-grams (Cimino and Dell'Orletta, 2017; Markov et al., 2017a). POS tags were obtained with the TreeTagger software package (Schmid, 1999), which uses the Penn Treebank tagset (36 POS tags). We used POS n-grams with n = 1–3.

Punctuation marks usage (PM) Because linguistic theories propose that punctuation emphasizes the information structure of a sentence, and different languages structure information differently, the usage of punctuation for an author's L1 is distinct from norms of L2. To encode punctuation usage we incorporate them in the POS and word n-gram features. We use a set of 36 punctuation marks (only 31 of which appear in the ICLE dataset).[3]

3.3 Experiment setup

We used the tokenized version of the TOEFL11 dataset and performed tokenization of the ICLE dataset using the Natural Language Toolkit (NLTK)[4] tokenizer. We performed lowercasing and removed the text surrounded by $<\ldots>$ in the ICLE dataset, since it corresponds to the dataset metadata. Each essay was represented through the sets of features described above. We used term frequency (tf) weighting scheme and the liblinear scikit-learn (Pedregosa et al., 2011) implementation of Support Vector Machines (SVM) with OvR (one vs. the rest) multi-class strategy. The effectiveness of SVM has been proven by numerous experiments on text classification tasks. The results were measured in terms of classification accuracy. Where not otherwise specified, the experiments were carried out under 10-fold cross-validation.

4 Experiments

We have designed a suite of experiments to help clarify the role/impact of punctuation usage as indicators of the author's native language. We start with the usual multi-class classification setting, then investigate the punctuation usage within the represented language families/geographical groups, with respect to the levels of proficiency, and finally with respect to the topics represented in the data.

Multi-class classification This setting is the usual NLI task, where the L1 of the author of a document is predicted based on a specific representation of the document.

2-step classification We postulate that punctuation usage from the native language is reflected in the text produced in L2. Since native languages belong to specific families, it is natural to ask whether there are strong influences within the language families, as well as at individual language level. This experimental setting will investigate this issue, through a 2-step classification: (i) a coarse classification into language families/geographical grouping of languages, (ii) fine-grained classification within each language group.

[3] Punctuation marks used as separate features would not capture their usage and role in the sentence, which is what we aim to represent. For the curious reader, however, we can report that used in isolation, punctuation marks achieve 18.74% accuracy on the TOEFL11 dataset under 10-fold cross-validation and 33.25% on ICLE, which are twice as high as the baselines (which are included in Table 4).

[4] http://www.nltk.org

Based on the languages represented in each dataset, we group the languages either by language family or by geographical location[5]. The language grouping used is the following:

TOEFL11: Arabic; Asian = {Chinese, Korean, Japanese}; Romance = {French, Italian, Spanish}; German; Indian = {Hindi, Telugu}; Turkish.

ICLE: Slavic = {Bulgarian, Czech, Russian}; Asian = {Chinese, Japanese}; Romance = {French, Spanish}.

Proficiency-level classification As students master a language better and better, it would be expected that their usage of punctuation will get closer to a native's, and the influence of their native language to get weaker. To test whether this is indeed the case, we have built a balanced dataset (from the point of view of proficiency levels) as a subset of the TOEFL11 dataset. The distribution of English proficiency levels in the TOEFL11 dataset is quite imbalanced, as shown in Table 1. To produce a balanced subset, we extract the same number of essays within each proficiency level (equal to the minimum number of essays for each level for each L1, in italics in Table 1).

We use both the imbalanced and balanced subsets to perform multi-class classification based on the proficiency level using POS n-grams with and without PM features, to determine the impact the punctuation has within each proficiency level.

Cross-topic and cross-corpus classification Brooke and Hirst (2012) have criticized the datasets used for NLI because they represent different topics, and thus the performance of the n-gram-based classifiers is questionable as capturing topics rather than native language phenomena. To investigate the impact of punctuation features which are rather abstract, we perform cross-topic and cross-corpus classification:

cross-topic: the essays in the TOEFL11 dataset were written in response to eight different topics or prompts (P0–P7), and all eight prompts are represented in all 11 L1 groups. We split the dataset in two ways:

(1) We split the TOEFL dataset into folds based on the topics – a topic will be present in only one fold (8 topics $\rightarrow$ 8-fold cross-validation).

(2) We used 5,838 essays written on the first four prompts (P0–P3) for training and 6,262 essays written on the P4–P7 prompts for testing. To compare the result of this experiment with a mixed-topic scenario with approximately same number of essays for training and testing, we split the TOEFL11 dataset using half of the essays on each prompt for training (6,050 essays) and testing (6,050 essays). For example, there are 140 essays of Chinese learners on P0, so we used 70 for training and 70 for testing, etc.

cross-corpus: we extract subsets of our two datasets that represent the same languages. The TOEFL11 and the ICLE datasets have 7 common languages: Chinese, French, German, Italian, Japanese, Spanish, and Turkish. We extract the subsets corresponding to these languages from the two corpora. We use each in turn for training and testing, respectively. For this experiment, we did not balance the ICLE dataset and used all the essays for each of the selected languages. The number of essays per class in the ICLE dataset is shown in Table 3.

5 Results and Discussion

5.1 Multi-class classification

Table 4 shows the multi-class classification results (column *1-step*) in terms of accuracy (%) for POS n-grams (n = 1, 2, 3, and 1–3) with and without PMs. As a reference point we provide the random baseline (also used in the NLI shared tasks 2013 (Tetreault et al., 2013) and 2017 (Malmasi et al., 2017)): 9.09% for 11 classes in the TOEFL11 dataset and 14.29% for 7 classes in the ICLE dataset. We include

[5]While a grouping based on language family is more theoretically justifiable, the close results (and for some settings better) in terms of accuracy for the 2-step classification seem to support the geographical grouping of languages as well, which can be explained by shared prosody – and in the written mode, shared information organizational patterns (also evidenced by the results presented in Section 5).

Language	No. of essays
Chinese	982
French	347
German	437
Italian	392
Japanese	366
Spanish	251
Turkish	280
Total	3,055

Table 3: Number (No.) of essays per class in the ICLE dataset used for the cross-corpus experiment.

the improvement (as absolute percentage points) when using PM features over the setting when PMs are omitted. In this and further experiments, the number of features (No.) is provided; the improvements are shown in bold typeface.

Features	TOEFL11 dataset			ICLE dataset		
	1-step acc.	2-step acc.	No.	1-step acc.	2-step acc.	No.
Random baseline	9.09	9.09		14.29	14.29	
POS w/o PMs	12.72	15.67	35	32.34	34.68	35
POS w/ PMs	17.50	19.12	71	48.05	46.49	66
Improvement:	**4.78**	**3.45**		**15.71**	**11.81**	
POS 2-grams w/o PMs	32.40	32.43	923	52.34	51.04	826
POS 2-grams w/ PMs	43.11	41.93	2,262	67.14	65.32	1,678
Improvement:	**10.71**	**9.50**		**14.80**	**14.28**	
POS 3-grams w/o PMs	37.99	38.19	14,036	55.19	52.73	9,455
POS 3-grams w/ PMs	46.88	47.31	27,431	65.45	61.69	16,850
Improvement:	**8.89**	**9.12**		**10.26**	**8.96**	
POS 1–3-grams w/o PMs	38.88	39.08	14,993	59.87	51.04	10,316
POS 1–3-grams w/ PMs	48.83	48.43	29,763	69.48	65.19	18,594
Improvement:	**9.95**	**9.35**		**9.61**	**14.15**	

Table 4: 10-fold cross-validation results (accuracy, %); POS n-grams with and without PMs; 1- and 2-step approaches.

As the results from Table 4 show, the inclusion of PMs improves the results for all the considered settings. The improvement in results brought by including the punctuation marks in the representation shown in Table 4 and throughout this section are statistically significant (unless explicitly mentioned otherwise) according to McNemar's statistical significance test (McNemar, 1947) with an α value of 0.05.

5.2 2-step classification

As explained in Section 4, the 2-step classification set-up would be useful to determine whether there are commonalities in punctuation usage across languages within the same family/geographical group. This would be reflective of grammatical/prosody/information structuring in different language families or groups.

The improvement for the 2-step approach demonstrates that there are shared patterns of punctuation usage across the grouped languages and across the individual languages.

The analysis of the 10 top features according to their weights for each dataset revealed that PMs are present among the 10 top features for all of the classes. The most frequent punctuation marks in these highly ranked features (bigrams and trigrams) were commas and full stops. An ablation study conducted to reveal the most indicative PM-enriched features showed that the performance does not come from one pattern, but L1-specific combinations.

5.3 Proficiency-level classification

We investigate whether higher proficiency levels lead to punctuation usage closer to an L2 native speaker. Should that be the case, we should note lower performance in native language identification with higher

proficiency levels, and in particular lower improvement in performance when adding punctuation marks to the document representations.

The results for each proficiency level on the imbalanced and balanced subsets of the TOEFL11 dataset are shown in Tables 5 and 6, respectively. The results are provided for 1- and 2-step approaches. Here, and in further experiments, the impact of PMs is evaluated using POS 1–3-gram features with and without PMs.

Features	**1-step acc.**	**2-step acc.**	**No.**
Low proficiency			
POS 1–3-grams w/o PMs	41.47	39.32	8,681
POS 1–3-grams w/ PMs	46.71	45.49	13,311
Improvement:	**5.24**	**6.17**	
Medium proficiency			
POS 1–3-grams w/o PMs	42.60	42.48	13,259
POS 1–3-grams w/ PMs	51.34	51.51	24,800
Improvement:	**8.74**	**9.03**	
High proficiency			
POS 1–3-grams w/o PMs	32.39	33.58	12,480
POS 1–3-grams w/ PMs	42.05	43.93	23,006
Improvement:	**9.66**	**10.35**	

Table 5: *Imbalanced setting*: 10-fold cross-validation results (accuracy, %) for each proficiency level.

Features	**1-step acc.**	**2-step acc.**	**No.**
Low proficiency			
POS 1–3-grams w/o PMs	37.87	38.22	8,471
POS 1–3-grams w/ PMs	44.08	42.76	12,900
Improvement:	**6.21**	**4.54**	
Medium proficiency			
POS 1–3-grams w/o PMs	36.00	35.52	8,953
POS 1–3-grams w/ PMs	43.36	44.11	13,996
Improvement:	**7.36**	**8.59**	
High proficiency			
POS 1–3-grams w/o PMs	31.93	31.31	9,367
POS 1–3-grams w/ PMs	37.25	39.39	14,992
Improvement:	**5.32**	**8.08**	

Table 6: *Balanced setting*: 10-fold cross-validation results (accuracy, %) for each proficiency level.

It is interesting to note that while the L1 classification results based on POS n-grams go down for high proficiency levels, the impact of adding the punctuation marks is higher for each proficiency level compared to the lower ones. According to the study conduced by Hirvela et al. (2012), L2 English learners are confident about their use of punctuation. However, the high improvement for high-proficiency learners in both imbalanced and balanced settings suggests that learners keep their L1 punctuation style even when achieving high English proficiency.

5.4 Cross-topic experiments

The cross-topic and cross-corpus experiments were performed to show that the influence of punctuation from the native language transcends topics and corpora, through features that capture PM usage, can partly compensate for the loss in performance under cross-topic or cross-corpus conditions.

	TOEFL11 (10FCV)		**TOEFL11 (topic = fold)**	
Features	**Acc.**	**No.**	**Acc.**	**No.**
POS 1–3-grams w/o PMs	38.88	14,993	33.88	14,636
POS 1–3-grams w/ PMs	48.83	29,763	43.21	28,635
Improvement:	**9.95**		**9.33**	

Table 7: 10-fold cross-validation and one fold/topic setting results.

Features	TOEFL11 (mixed-topic) Acc.	No.	TOEFL11 (cross-topic) Acc.	No.
POS 1–3-grams w/o PMs	36.63	13,174	32.27	13,042
POS 1–3-grams w/ PMs	45.95	24,215	40.74	23,950
Improvement:	**9.32**		**8.47**	

Table 8: Mixed- and cross-topic settings results.

The results for cross-topic classification are presented in Tables 7–8. Separating the training and test data based on topics leads to a drop in performance of approx. 5 percentage points in both the cross-validation and train/test split conditions. But for both settings, adding the punctuation-based features leads to very similar increases in performance whether the topics are separated or mixed. This indicates that the punctuation-based features are robust and portable across topics.

5.5 Cross-corpus experiments

The cross-corpus experiments explore further the robustness of the punctuation-based features. An overfitting model would lead to lower scores when tested on a corpus different to the training corpus. We include here experiments done using word n-grams (n = 1–3), tf weighted just like our other features (as described in Section 3).

Training on TOEFL, testing on ICLE					
	10FCV		Test Set		
Features	**Acc.**	**F1**	**Acc.**	**F1**	**No.**
POS 1–3-grams w/o PMs	48.62	48.51	43.27	40.70	13,587
POS 1–3-grams w/ PMs	60.29	60.22	54.50	53.05	25,870
Improvement:	**11.67**	**11.71**	**11.23**	**12.35**	
Word 1–3-grams w/o PMs	80.91	80.87	73.81	71.27	1,904,839
Word 1–3-grams w/ PMs	83.52	83.47	74.47	72.05	1,806,102
Improvement:	**2.61**	**2.60**	**0.66**	**0.78**[6]	
Training on ICLE, testing on TOEFL					
	10FCV		Test Set		
Features	**Acc.**	**F1**	**Acc.**	**F1**	**No.**
POS 1–3-grams w/o PMs	79.47	75.21	34.22	32.26	13,730
POS 1–3-grams w/ PMs	86.67	83.82	41.64	39.72	26,890
Improvement:	**7.20**	**8.61**	**7.42**	**7.46**	
Word 1–3-grams w/o PMs	92.47	90.85	43.66	42.41	1,706,554
Word 1–3-grams w/ PMs	94.11	93.04	47.19	45.26	1,644,978
Improvement:	**1.64**	**2.19**	**3.53**	**2.85**	

Table 9: Cross-corpus classification results for POS and word n-grams with and without PMs.

The results for cross-corpus experiments (training on TOEFL11 and testing on ICLE, and vice versa) are shown in Table 9. 10FCV stands for 10-fold cross-validation on the training data (accuracy, % and F1 macro, %). We note that despite the loss in performance suffered by the model based on POS and word n-gram features, the PM features are robust and lead to the same increase in performance on testing as they did on training. While the loss in performance when training on TOEFL, testing on ICLE is relatively small (5–7 percentage points for accuracy), training on ICLE and testing on TOEFL leads to much more dramatic drops (45 percentage points for accuracy). For the models based on word n-grams, their high results are harder to improve by the addition of PM features, but they contribute nonetheless, and when added to the model trained on ICLE their impact on the TOEFL data is higher than on ICLE.

In a detailed, per-language view, presented through the confusion matrices (Figure 1), we can note that the highest improvement when including PM features in this cross-corpus study was achieved for German. There is also a high improvement for Turkish and Italian. They indicate that these language have stronger, or maybe more consistent, punctuation styles that interfere in the production of L2. Personal

[6]This improvement is not statistically significant.

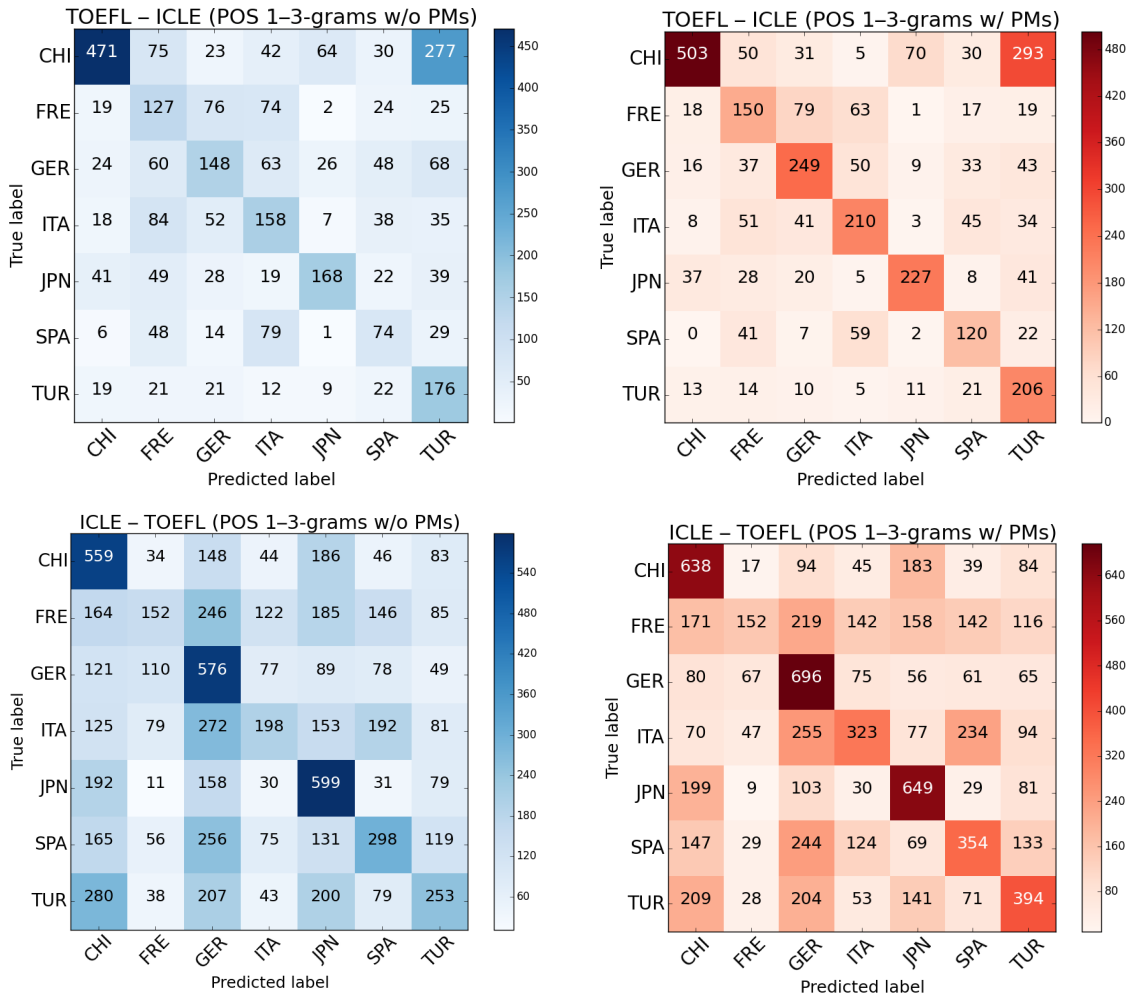


Figure 1: Training on TOEFL, testing on ICLE: POS 1–3-grams without (top left)/with PMs (top right). Training on ICLE, testing on TOEFL: POS 1–3-grams without (bottom left)/with PMs (bottom right).

experience with German and Italian punctuation confirms these findings, but our models could support deeper linguistic exploration into these phenomena.

6 Conclusions

While the role of punctuation is still disputed in linguistic theory – as a written indicator of prosody, or as grammatical features – punctuation is however linked to each language and the manner in which languages organize and convey information. We proposed the hypothesis that punctuation usage in L2 is indicative of an author's native language. We have conducted a series of experiments to investigate the impact of punctuation on native language identification. The experiments show that punctuation marks provide useful information, and when combined with POS and even word n-gram features – thus capturing their usage – lead to significant and substantial improvements. Their impact is positive for both coarse (family-language level) and fine-grained classification, indicating that there are patterns of punctuation usage that are common across language families, but also patterns specific to individual languages. Punctuation interference does not seem to decrease with the level of proficiency: while we would expect that as the proficiency level increases an author's usage of punctuation will be closer to English and thus the native language will be harder to detect, this is not the case. Finally, contrary to word n-grams which necessarily capture also topic-specific information and thus tend to overfit the training data, punctuation is more abstract and as such a more robust feature, as shown by the results of cross-topic and cross-corpus experiments.

References

Naomi Baron. 2001. Commas and canaries: The role of punctuation in speech and writing. *Language Sciences*, 23(1):15–67.

Daniel Blanchard, Joel Tetreault, Derrick Higgins, Aoife Cahill, and Martin Chodorow. 2013. TOEFL11: A corpus of non-native english. *ETS Research Report Series*, 2013(2):i–15.

Julian Brooke and Graeme Hirst. 2011. Native language detection with 'cheap' learner corpora. In *Proceedings of the Conference of Learner Corpus Research*, pages 37–47. Presses universitaires de Louvain.

Julian Brooke and Graeme Hirst. 2012. Robust, lexicalized native language identification. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 391–408. The COLING 2012 Organizing Committee.

Paul Bruthiaux. 1993. Knowing when to stop: Investigating the nature of punctuation. *Language and Communication*, 13(1):27–43.

Lingzhen Chen, Carlo Strapparava, and Vivi Nastase. 2017. Improving native language identification by using spelling errors. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 542–546. Association for Computational Linguistics.

Andrea Cimino and Felice Dell'Orletta. 2017. Stacked sentence-document classifier approach for improving native language identification. In *Proceedings of the 12th Workshop on Building Educational Applications Using NLP*, pages 430–437. Association for Computational Linguistics.

Sylviane Granger, Estelle Dagneaux, Fanny Meunier, and Magali Paquot. 2009. *International Corpus of Learner English v2 (ICLE)*. Presses Universitaires de Louvain, Louvain-la-Neuve, Belgium.

Jack Grieve. 2007. Quantitative authorship attribution: An evaluation of techniques. *Literary and Linguistic Computing*, 22(3):251–270.

Alan Hirvela, Alexander Nussbaum, and Herbert Pierson. 2012. ESL students' attitudes toward punctuation. *System*, 40(1):11–23.

Radu Tudor Ionescu, Marius Popescu, and Aoife Cahill. 2014. Can characters reveal your native language? A language-independent approach to native language identification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1363–1373. Association for Computational Linguistics.

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. 2005. Determining an author's native language by mining a text for errors. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 624–628. Association for Computing Machinery.

Shibamouli Lahiri and Rada Mihalcea. 2013. Using n-gram and word network features for native language identification. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 251–259. Association for Computational Linguistics.

Shervin Malmasi, Keelan Evanini, Aoife Cahill, Joel Tetreault, Robert Pugh, Christopher Hamill, Diane Napolitano, and Yao Qian. 2017. A report on the 2017 native language identification shared task. In *Proceedings of the 12th Workshop on Building Educational Applications Using NLP*, pages 62–75. Association for Computational Linguistics.

Ilia Markov, Lingzhen Chen, Carlo Strapparava, and Grigori Sidorov. 2017a. CIC-FBK approach to native language identification. In *Proceedings of the 12th Workshop on Building Educational Applications Using NLP*, pages 374–381. Association for Computational Linguistics.

Ilia Markov, Efstathios Stamatatos, and Grigori Sidorov. 2017b. Improving cross-topic authorship attribution: The role of pre-processing. In *Proceedings of the 18th International Conference on Computational Linguistics and Intelligent Text Processing*. Springer, in press.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Nick Moore. 2016. What's the point? the role of punctuation in realising information structure in written english. *Functional Linguistics*, 3(1):6.

Ryo Nagata and Edward Whittaker. 2013. Reconstructing an indo-european family tree from non-native english texts. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1137–1147. Association for Computational Linguistics.

Vivi Nastase and Carlo Strapparava. 2017. Word etymology as native language interference. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2692–2697. Association for Computational Linguistics.

Garrett Nicolai, Bradley Hauer, Mohammad Salameh, Lei Yao, and Grzegorz Kondrak. 2013. Cognate and misspelling features for natural language identification. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 140–145. Association for Computational Linguistics.

Terence Odlin. 1989. *Language Transfer: cross-linguistic influence in language learning*. Cambridge University Press, Cambridge, UK.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Helmut Schmid, 1999. *Improvements In Part-of-Speech Tagging With an Application to German*, pages 13–25. Springer Netherlands.

Joel Tetreault, Daniel Blanchard, Aoife Cahill, and Martin Chodorow. 2012. Native tongues, lost and found: Resources and empirical evaluations in native language identification. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 2585–2602. The COLING 2012 Organizing Committee.

Joel Tetreault, Daniel Blanchard, and Aoife Cahill. 2013. A report on the first native language identification shared task. In *Proceedings of the Eighth Workshop on Building Educational Applications Using NLP*, pages 48–57. Association for Computational Linguistics.

Oren Tsur and Ari Rappoport. 2007. Using classifier features for studying the effect of native language on the choice of written second language words. In *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition*, pages 9–16. Association for Computational Linguistics.

Sze-meng Jojo Wong and Mark Dras. 2009. Contrastive analysis and native language identification. In *Proceedings of the Australasian Language Technology Association Workshop 2009*, pages 53–61. Association for Computational Linguistics.

Investigating Productive and Receptive Knowledge: A Profile for Second Language Learning

Leonardo Zilio **Rodrigo Wilkens** **Cédrick Fairon**
Centre de Traitement Automatique du Langage (Cental)
Université Catholique de Louvain (UCL), Belgium
{leonardo.zilio,rodrigo.wilkens,cedrick.fairon}@uclouvain.be

Abstract

The literature frequently addresses the differences in receptive and productive vocabulary, but grammar is often left unacknowledged in second language acquisition studies. In this paper, we used two corpora to investigate the divergences in the behavior of pedagogically relevant grammatical structures in reception and production texts. We further improved the divergence scores observed in this investigation by setting a polarity to them that indicates whether there is overuse or underuse of a grammatical structure by language learners. This led to the compilation of a language profile that was later combined with vocabulary and readability features for classifying reception and production texts in three classes: beginner, intermediate, and advanced. The results of the automatic classification task in both production (0.872 of F-measure) and reception (0.942 of F-measure) were comparable to the current state of the art. We also attempted to automatically attribute a score to texts produced by learners, and the correlation results were encouraging, but there is still a good amount of room for improvement in this task. The developed language profile will serve as input for a system that helps language learners to activate more of their passive knowledge in writing texts.

Title and Abstract in Portuguese

Investigação de Conhecimento Produtivo e Receptivo: Um perfil de aprendizado de segunda língua

A literatura de aquisição de segunda língua frequentemente aborda diferenças no vocabulário produtivo (que um aprendiz consegue utilizar em textos) e receptivo (que um aprendiz consegue entender em textos) de um aprendiz de língua, mas as estruturas gramaticais normalmente não são investigadas nesse nível. Neste trabalho, usamos dois corpora para investigar as divergências nas ocorrências de estruturas gramaticais pedagogicamente relevantes em textos de produção e recepção. Os valores de divergência observados foram incrementados com a atribuição de uma polaridade que indica um sobreuso ou subuso de uma estrutura gramatical por aprendizes de língua. Essa investigação levou à compilação de um perfil de linguagem voltado ao aprendizado de segunda língua que posteriormente foi combinado com informações lexicais e de legibilidade para uma classificação de textos de produção e recepção em três categorias: iniciante, intermediário e avançado. Os resultados da classificação automática tanto de textos de produção (medida F de 0,872) quanto de recepção (medida F de 0,942) foram comparáveis ao atual estado da arte. Também realizamos um experimento para atribuir automaticamente uma nota aos textos produzidos por aprendizes, e os resultados da correlação foram encorajadores, mas mostram que ainda há muitas lacunas a serem supridas e questões em aberto para a realização da tarefa, especialmente no que diz respeito à subjetividade envolvida na atribuição de notas. O perfil de linguagem apresentado servirá como base para um sistema de auxílio à ativação de conhecimento receptivo na escritura de textos.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3467–3478
Santa Fe, New Mexico, USA, August 20-26, 2018.

1 Introduction

Second Language Acquisition (SLA) involves a series of different linguistic knowledge that a second language learner has to deal with, such as, but obviously not limited to, vocabulary and syntax of the second language. Besides the linguistic levels, another important topic for SLA is the knowledge activation capacity, i.e., how much from the learner's knowledge can be activated during a productive task. These are all aspects that help conform the SLA process, and are factors that get included in the evaluation of second language learners in terms of their capabilities of performance in a second language environment.

In the literature, it is possible to find many studies discussing how the passive/receptive vocabulary knowledge relates to the active/productive vocabulary knowledge, be it in a native or second language setting (e.g. Morgan and Oberdeck (1930), Meara (1990), Laufer (1998), Fan (2000), Schmitt (2008), Pignot-Shahov (2012)). In those studies, it is agreed that the passive vocabulary knowledge is larger than the active one. However, none of those studies devote some attention to grammar, so that it is not debated how grammar works in terms of productive and receptive knowledge.

In the case of second language learning, there is also the adherence to a commonly used framework, namely the Common European Framework of Reference for Languages (CEFR) (Verhelst et al., 2009), which divides and describes the communicative goals that should be achieved by an idealized second language learner, no matter in which language or from which native language, in each of the levels. The levels scale from A to C, and there are two sublevels each: A1, A2, B1, B2, C1, and C2. The CEFR also conceives the existence of a second subdivision, such as A1-, A1, and A1+, so that, if needed, the sublevels can be further specified. As such, if one intends to deal with SLA, there is a lot of ground to cover in terms of knowledge that is spread along all the different levels and subclassifications.

One of the ways of dealing with this spread knowledge is by organizing language profiles. The methods for characterizing a language profile use as basis the observation of different groups of people that use language in distinct ways, but that have some fundamental common traces when compared to other groups of speakers of the same language. The whole of these common traces can be called a language profile (Argamon et al., 2009). Two examples of profiles that deal with a specific linguistic level for language learning are the English Vocabulary Profile and the English Grammar Profile[1].

In this study, we are interested in the observation of receptive and productive knowledge present in second-language-related corpora to draw information about the differences between expected receptive and actual productive knowledge. We compare texts designed as input for SLA and the actual output of learners of English to see where the divergence between both of these types of tasks lies. Since vocabulary has already been studied in various works, here we will be using vocabulary as well, but we will deal especially with grammar, and we aim at automatically generating a language profile that can model the relationship between production and reception in SLA. We further hypothesize that, by adding grammatical information, we can improve the modeling of SLA in terms of productive and receptive knowledge.

The language profile developed in this study will also be used to help learners in the task of producing written texts that use the most from the learner's passive knowledge. For achieving this goal, we describe the relation between reception texts and production texts in terms of their divergence, and later we extrinsically evaluate our model, by applying the information drawn for our language profile to a text classification task. As an associated task, we also delve into the automatic evaluation of texts produced by language learners, by correlating scores generated by a classification algorithm to the actual scores given by professional evaluators.

This paper is divided as follows: in Section 2, we briefly contextualize the task addressed in this study and how we are going to deal with it; in Section 3, we describe the corpora that we used as representation of reception and production texts and their automatic annotation with pedagogically relevant grammatical information; we then explain the grammatical profiling that was carried out in Section 4; Sections 5 and 6 are dedicated to the experiments that we conducted using our language profile, these two sections present

[1] Both the EVP and the EGP are available at http://www.englishprofile.org/.

the methodology used for each experiment and also discuss their results; finally, we sum up our findings and report some future work in Section 7.

2 The Task at Hand

In this paper, we are dealing with the contrast between texts produced by learners and texts that are used as input for learners. Our main goal is to draw from this type of data a language profile that can be used to help language learners to produce texts that use the most of their passive knowledge. In this paper, we deal with texts written in English, but the methodology could be applied to other languages, provided there are similar resources.

The contrastive study of distributions was successfully used to identify characteristics that are relevant to text profiling (Biber, 1991). In a complementary manner, entropy-based approaches have been employed to identify similarities (and dissimilarities) among text sets (Dagan et al., 1997; Oakes, 2008) and to study vocabulary variation (Oakes and Farrow, 2007). In this respect, there are different divergence equations that can be employed for evaluating entropy, and one that stands out is the Kullback–Leibler divergence (Kullback and Leibler, 1951; Kullback, 1997), which is also called relative entropy. This divergence is defined in Equation 1, where P and Q are the probability distribution of two text sets. Despite its usefulness, it has a known problem: a lack of symmetry, which makes it hard to use in terms of a distance measure. In that sense, the Jensen-Shannon divergence (Endres and Schindelin, 2003; Österreicher and Vajda, 2003; Fuglede and Topsoe, 2004), presented in Equation 2, averages the divergence of the interchange of the distributions taking into account the average of the distribution P and Q, as shown in Equation 3. The Jensen-Shannon divergence solves the lack of symmetry, so that the result can be understood as a measure of distance, which allowed us to use it to observe the difference between reception and production.

$$D_{KL}\left(P||Q\right) = -\sum_{i} P(i) log\left(\frac{Q(i)}{P(i)}\right) \tag{1}$$

$$D_{JS}\left(P||Q\right) = \frac{1}{2} D_{KL}\left(P||M\right) + \frac{1}{2} D_{KL}\left(Q||M\right) \tag{2}$$

$$M = \frac{1}{2}\left(P + M\right) \tag{3}$$

For achieving the objectives described in this paper, we used two corpora: one made up of texts that are used as input in language learning tasks, and the other one composed of texts that were produced by language learners. Different parts of these corpora were used for deriving our language profile, which was based on a Jensen-Shannon Divergence model, and for extrinsically testing our profile in different tasks related to the evaluation of texts used in language learning.

By observing the divergence between production and reception and generating a language profile, we can also create a tool that guides language learners to potentially use more of their passive knowledge in written production tasks. The idea here is to match a learner's written text to the patterns found in reception texts, so that we can provide cues for the learner on how the text could be improved.

3 Corpora

In this study, we conducted several separate experiments that were related to the goal of developing a language profile that takes into account both sides of written language learning and applying this profile to classify and evaluate written texts in accordance to a given language level. The experiments had as basis two corpora: one corpus containing texts used as input for language learning, and another one containing texts that were produced by language learners. In the next two subsections, we describe the sources we used for both corpora. Since we used them differently in each of the experiments[2], here we provide only a general description, leaving the more fine details to be described together with the

[2]It is important to make it clear that we did not mix the texts used for creating the profile with those that were posteriorly used for extrinsically testing it.

experiments. At the end, we describe the automatic annotation that was applied to both corpora for the experiments.

3.1 Reception Corpus

For observing the type of linguistic information, in terms of vocabulary and grammar, that is present in the input for a language learning task, we combined texts from two different sources: the Breaking News English Lessons (BNE) (Banville, 2009), and Altissia's animation database for the English course[3].

The BNE corpus consists of texts written for learners of English that are distributed along the Common European Framework of Reference for Languages (CEFR) levels, albeit using a level classification from 0 to 7 (that covers the levels A2 to C2). It contains documents that range from 120 words on the easiest level to 250 words on the higher levels. Since the CEFR levels are mixed in some of the texts, what we did was to consider a division of the documents into a beginner class (for those addressing the A2 level), an intermediate class (for those addressing B1 and B2 levels), and an advanced class (for those addressing C1 and C2 levels). As such, levels 1 and 2 of the BNE were allocated to the beginner class; levels 4 and 5 composed the intermediate class; and levels 6 and 7 formed the advanced class. Level 3 was discarded for being a hybrid between the levels A2 and B1, and level 0 was discarded for balancing reasons. The final BNE corpus that we used contains 930 documents and 194,207 tokens.

Altissia's animation database is composed of transcribed texts from short animation sequences, comprising dialogs and narratives that are used as input for learners on an online language learning platform. The transcriptions are all classified according to the CEFR, so we put together texts from levels A1 and A2 (for the beginner class), B1 and B2 (for the intermediate class), and C1 and C2 (for the advanced class), in the same way as we did for the BNE corpus. This corpus amounts to 154 documents and 23,672 tokens, which conforms a rather small corpus, but we decided to add it to the BNE corpus for achieving more variation in terms of text genre.

3.2 Production Corpus

As our representative of the productive knowledge, we used the EF-Cambridge Open Language Database (EFCAMDAT) (Geertzen et al., 2013)[4]. This corpus is divided according to the CEFR (Verhelst et al., 2009), and contains a huge amount of documents written by learners, totaling more than 500 thousand documents with more than 33 million tokens. They were written by 83,385 learners from 137 nationalities, and each document has an evaluation score ranging from 0 to 100 and an associated topic (e.g. *introducing yourself by email*). The data from the corpus is organized following the CEFR classification, so that we just transposed the original division to our own three categories: beginner, intermediate, and advanced.

3.3 Corpus Annotation

Both the reception and the production corpora were automatically annotated with pedagogically relevant grammatical information. For this type of annotation, we used the features described in the SMILLE system (Zilio and Fairon, 2017; Zilio et al., 2017a; Zilio et al., 2017b), which can recognize 107 different grammatical (both morphosyntactic and syntactic) structures in English texts that are relevant for the learning of English as second language[5]. These grammatical structures are ranked according to the CEFR levels, from level A1 to C1, following the pedagogical organization of Altissia's English course. Large part of the information annotated by the SMILLE system is different from a simple parsing annotation, since it represents a combination of parser information with manually created rules that focus on a pedagogical approach to language learning. The annotation was done automatically, and previous work has shown that the overall precision of the system lies around 90.1% for syntactic structures in

[3] www.altissia.com.

[4] We used the EF-CAMDAT version 1 (https://corpus.mml.cam.ac.uk/efcamdat1/) in this paper, because version 2 (https://corpus.mml.cam.ac.uk/efcamdat2/public_html/) was not yet available at the beginning of the research.

[5] For a more complete description of these grammatical structures, please refer to Zilio et al. (2018). The annotated vectors of the EF-CAMDAT corpus can be downloaded from http://cental.uclouvain.be/resources/smalla_smille/sgate/.

the EF-CAMDAT corpus (Zilio et al., 2018) (and precision for morphologic structures is based on the Stanford Parser (Manning et al., 2014) performance)[6].

After the annotation process, each document in both corpora was represented as a vector with the normalized frequency of each of the annotated structures. The normalization of each of the syntactic features was done by dividing their absolute frequency in the document by the number of sentences in it. For the morphosyntactic features, the normalization was carried out by dividing the absolute frequency of the feature in the document by the number of tokens in it.

4 Grammar Profile: Production vs. Reception

The methodology that is generally applied to profiling consists in the use of a training corpus with annotated documents, which are then converted to vectors of features, and methods of classification based on machine learning. The creation of vectors normally takes into account the stylometry of the document. As an example of profiling, Argamon et al. (2009) employ a taxonomy of a specific grammatical class, combining it with other metrics for training a classification algorithm.

By using a corpus of input texts (texts that are used as a receptive means for language learning) and a corpus of output texts (texts that are produced by language learners), we could contrast the occurrences of pedagogically relevant grammatical structures in both of them. This enabled us to come up with a language profile oriented to language learning that considers both sides of the writing spectrum.

In this profiling task, while we used a random selection of 250 documents from each of the three classes (beginner, intermediate, and advanced) of the input corpus (totaling 750 documents), in terms of output, we randomly selected the same amount, but considering only those texts for which the evaluation score given by an expert was between 90 and 95. This ensured, at the same time, that our sample was not made up of texts that were above their level, which is a possible explanation for a maximum score, nor did it contained texts that were not so well formed from a linguistic point of view, because this could pose problems for the automatic annotation and, thus, for our profile.

The annotated corpora were analyzed in terms of the Jensen-Shannon divergence that exists in each of the three classes (beginner, intermediate and advanced) for each of the 107 annotated grammatical structures. As explained in Section 3.3, for a more balanced analysis, the structures were normalized in relation to the number of sentences (in the case of syntactic structures) and tokens (for morphosyntactic structures), and then, for applying the divergence score between both corpora, they were averaged in relation to the number of documents in each corpus. In this process, any structure with a relative frequency equal to zero, whether in the reception or the production corpus, was discarded. The divergence was calculated using ten different random samples for ensuring a better confidence in the data, and the results from the ten samples were afterwards averaged.

After the calculation of the divergence of each of the non-discarded grammatical structures, we had to select a cut-off point for making up our language profile. Since the divergence provides scores, but not a clear cut-off point, we deliberated that the structures classified at the upper quartile regarding the divergence distribution were a good choice to form our language profile. By the end of this process, we identified structures that could be considered problematic in terms of writing for a language learner in each of the three levels. However, the divergence score gave us only the magnitude of the divergence, but it did not account for where the divergence exactly lies, and, for our purposes, we deemed important to know the direction of the divergence, i.e., to know whether a structure is more prominent in the reception or the production. So we went further on the task and used a voting system based on normalized frequency to see which of the corpora presented more of a given structure. This voting system was used to attribute a polarity to the divergence score, representing cases of overproduction (positive scores), which are probably a sign of easier structures for the learners, but that are not that important in a more naturally written text, or cases of underproduction (negative scores), which are an indicator of structures that are more frequent in the input, but the learners may have some difficulties in producing them. For attributing a polarity to the grammatical structure, at least seven out of the ten random samples of the

[6]Further evaluation of more specific structures in different corpora can be found in Zilio et al. (2017a) and Zilio et al. (2017b).

output corpus had to point to overuse or underuse, if there was no side with seven votes or more, we filtered the grammatical structure out of the results, because it represented a case in which the use of the grammatical structures fluctuated too much in the samples.

Many of the highly divergent structures (upper quartile) were similar among the classes, but some of them showed an overuse in some levels and an underuse in others. Table 1 presents the structures that were divergent from input to output along all the levels, and the over- or underuse is expressed by plus or minus symbols, respectively. In the table, lines 1 to 9 represent structures that are divergent in all three classes, from beginner to advanced; lines 10 to 12 show structures that were divergent both in the beginner and intermediate classes; then lines 13 to 22 present structures that were divergent both in the beginner and advanced classes; and, finally, the remaining lines display structures that were divergent in only one of the classes (lines 23 to 26 for beginner, lines 27 and 28 for intermediate, and lines 29 to 34 for advanced). As we can see, the grammatical structures that present divergence go from more coarse-grained syntactic phenomena, such as passive voice and imperative, to more fine-grained ones, like the verb "to be" in different tenses and use of "would" to express hypothesis.

Before going to the other experiments, there are some interesting information presented by our profile that we would like to discuss. For instance: it seems that learners of English tend to overuse the present tense, while neglecting the past tense in relation to the texts that are considered fit for reception. There is also a constant trend to neglect the use of genitive, the present perfect tense and the use of direct complements. These structures that tend to be neglected throughout the levels may be clues for grammatical structures that pose a general problem for language learners. The divergence on the use of direct complements may be also a clue for the existence of a language use with emphasis on intransitive verbs. All these indications present a good basis for further linguistic investigation that would need to be directly assessed in the texts.

4.1 Applying the Grammar Profile

The grammar profile that we generated is intended to describe the behavior of grammatical structures in texts that are either input or output of a language learning task. So, we used it as a basis for other experiments that are important in helping a language learner to improve their writing skills but also that serve as an extrinsic evaluation of the profile.

For these experiments, that we describe and discuss in Sections 5 and 6, we used the grammatical structures that were present in the profile, which amounted to 26 for the beginner class, 14 for the intermediate class and 25 for the advanced class, in terms of their normalized frequency and of their significant difference in relation to the relative frequencies that were learnt in the samples used for generating the profile.

Following this principle, each of the documents in all of the following experiments were rendered as a vector with two scores per grammatical structure in the profile, according to the class to which it belongs: the first score is the normalized frequency observed in the document divided by the normalized frequency from the profile for the reception corpus; and the second score is the normalized frequency observed in the document divided by the normalized frequency from the profile for the production corpus. For both those scores, if there is no significant difference between the normalized frequencies, the score was set to 1.

Since a well-formed text is composed not only of grammar, but also of a series of linguistic information, including some that are not yet computationally verifiable (e.g., certain pragmatical and discursive aspects), we designed the experiments in the next sections in a way that they include not only our profile based on grammar, but also vocabulary and readability measures. By doing so, we are adding to our profile features that are well described in the literature and that may compensate for some of the extra linguistic information that is lacking in the profile. Even so, in each of the experiments, we also tested all of the feature sets separately. This allowed us to observe the possible contribution of others measures to the language profile that we propose here.

For the vocabulary features, we used as basis a frequency list that was extracted from the British National Corpus (BNC) (Aston and Burnard, 1998) and divided in five pedagogically distributed ranges

Table 1: Grammatical Structures: cases of overuse and underuse contrasting the productive with the receptive corpus

		Beginner	**Intermediate**	**Advanced**
1	*genitive*	-	-	-
2	*advanced modal verbs*	-	+	+
3	*past simple*	-	-	-
4	*present perfect*	-	-	-
5	*present simple*	+	+	+
6	*verb "to be" in the present simple*	+	+	+
7	*infinitive with "to" after a verb*	-	+	+
8	*number of direct complements*	-	-	-
9	*present participle*	+	+	-
10	*imperative*	+	+	
11	*number of main verbs*	-	-	
12	*number of subjects*	-	-	
13	*simple modal verbs*	-		+
14	*modal verbs with ellipsed main verb*	-		+
15	*passive voice*	-		-
16	*relative clauses*	-		-
17	*future with "will"*	-		+
18	*verb "to be" in the past simple*	-		-
19	*infinitive with "to" after an adjective*	-		+
20	*connectives of time*	-		+
21	*connectives of reason and result*	+		+
22	*connectives of purpose*	-		-
23	*use of articles*	-		
24	*use of plurals*	-		
25	*use of numbers*	+		
26	*present continuous*	+		
27	*use of present and present perfect after time connectives*		+	
28	*connectives of condition*		+	
29	*use of would to express hypothesis*			+
30	*prepositional verbs*			+
31	*infinitive with "to" after a noun*			+
32	*gerund after prepositions*			-
33	*connectives of alternative*			-
34	*connectives of example and explanation*			+

(Martinez, 2011; Cobb, 2013). Using this list we calculated the frequency of words from each range in each of the documents from our corpora. As a means of normalizing the frequency, we divided the sum of the frequency of words from each BNC range in the document by the total number of words from all ranges in the same document. Finally, for the readability measures, we used the Dale-Chall Score (Dale and Chall, 1948), the Flesch-Kincaid measure (Kincaid et al., 1975), and the Gunning Fog Index (Gunning, 1952), which are commonly used in the literature.

5 Text Classification: Production Corpus

Having a language profile at hand that emphasizes the divergence between reception and production in language learning, we used it to assess how good it is for classifying texts produced by language learners in the respective levels. This is an important task in the goal of helping learners to write texts that match their level, since the first step in evaluating a text is to attribute it to the correct level, so that it is possible

to further analyze its weaknesses and strengths.

For this test, we randomly selected documents from the production corpus that were not used to generate the divergences. For these random selection of documents, we used certain constraints, so that the documents must have been evaluated above 90%, so as to match the criteria employed by Wilkens et al. (2018), who use the same corpus, so that we would have a comparable basis. The corpus was divided in beginner, intermediate, and advanced classes, as explained in Section 3.2. The full sample contained 15,068 documents (6 thousand for beginner and intermediate, and 3,068 for advanced[7]). For the classification task, we used the Random Forest algorithm (Breiman, 2001) with a ten-fold cross-validation.

The results of this experiment are shown in Table 2 in terms of F-measure and standard deviation. For the beginner level, our best F-measure was 0.939, decreasing to 0.858 in the intermediate level and further to 0.822 for the advanced level. By averaging the three levels, we got a non-weighted result of 0.872 (0.882 if weighted). The best average result was achieved by mixing all three measures, and, with exception of the advanced level, which fared a bit better with only vocabulary features, the other levels also had better results using the combined approach. We can also see that the worst results were produced by readability measures, which scored consistently below the other features. Our best results in this experiment are consistent with the state of the art for document classification in levels, as can be seen in the description by Wilkens et al. (2018).

Table 2: F-measure and Standard Deviation for the Level Classification in the Production Corpus

	Profile	**Vocabulary**	**Readability**	**All**
Beginner	0.922 (0.024)	0.893 (0.020)	0.775 (0.030)	**0.939** (0.022)
Intermediate	0.819 (0.000)	0.811 (0.000)	0.589 (0.000)	**0.858** (0.000)
Advanced	0.747 (0.000)	**0.822** (0.000)	0.464 (0.000)	0.819 (0.000)
Average	0.830 (0.009)	0.842 (0.010)	0.609 (0.014)	**0.872** (0.009)

Values in bold reflect the best scores in each level and the average best score. All scores were significantly different from the best scores with a confidence of 99%.

5.1 Text Scoring

This experiment was conducted to observe if our language profile was good for evaluating texts produced by language learners in terms of score (the actual score attributed to a text by an evaluator regarding its well-formedness in relation to the given task). To do so, we randomly selected 70 texts from the production corpus per each ten score points[8] as input for the Random Forest algorithm in a three-fold cross-validation. As such, for each document received as input, the algorithm produced a score from zero to a hundred, in the same fashion as an expert would, for each of the EFCAMDAT documents in the sample. After that, we calculated the correlation of the scores produced by the algorithm with the actual scores that were given by the experts that evaluated the Cambridge Exams.

Using a total of 1,982 documents (734 documents for beginner, 700 for intermediate and 548 for advanced), we ran a three-fold cross-validation. The results in terms of Pearson's correlation and standard deviation are presented in Table 3. Again, as we saw for the classification experiment, the combined features yielded the best results for the beginner and intermediate levels, while the vocabulary alone was better to predict the scores of the advanced level, although this result was not significantly different from the one achieved by the combined features. Looking at the table and disconsidering the readability alone, which was a catastrophe, the correlations, especially for the advanced level, were not so bad, ranging from 0.509 to 0.685. However, by looking at the root mean squared error (RMSE), we see values ranging from 30.42 (advanced level) to 32.14 (beginner level), with a stop at 31.24 (intermediate

[7]This is the whole of EFCAMDAT for C1 and C2 combined for scores above 90% after excluding those that were used for calculating the divergence scores.

[8]We divided the whole corpus in eleven truncated ranges: from 0 to 9, 10 to 19 etc., up to 90-99, and 100 as a separate range, for each of the three classes (beginner, intermediate, and advanced). The EFCAMDAT contains lots of documents evaluated with higher notes, but not so many on the lower side. For some of the lower score ranges, there was not 70 documents, so we used all those available. None of these documents were present in the production corpus that was used for developing the language profile.

level). These values, which were not significantly different between each other in the levels for all setups, are far from good.

These results indicate that other factors need to be taken into consideration for the evaluation of a learner's text in terms of score, not only grammatical and/or vocabulary features, let alone readability measures. It is also important to consider that, for the scores, there is always a strong subjective impact based on the expert's opinion, which is beyond our actual capacity of modeling.

Table 3: Correlation and Standard Deviation for the Scoring Experiment

	Profile	**Vocabulary**	**Readability**	**All**
Beginner	0.457 (0.044)	0.445 (0.042)	0.253 (0.054)	**0.509** (0.034)
Intermediate	0.459 (0.044)	*0.512 (0.034)	0.242 (0.056)	**0.524** (0.037)
Advanced	0.624 (0.038)	**0.685** (0.047)	0.364 (0.055)	*0.674 (0.033)

Values in bold reflect the best scores in each level and the average best score. The values with a star (*) were not significantly different from the best scores with a confidence of 99%.

6 Text Classification: Reception Corpus

This last experiment was designed to test the capacity of our profile to classify texts that are used as input for language learning activities. Although this task does not directly relate to helping a learner in a writing activity, it does serve as an extrinsic test for evaluating the quality of our language profile, while also being an important task in language learning in general.

This experiment followed the exact methodology described in Section 5 for the production corpus, with the obvious difference that here we used the reception corpus as basis and, as such, we had a quite drastic reduction in the number of documents used for the classification, especially because we used only those documents that were not used for generating our language profile. So, to improve the corpus size, we sorted 10 random document samples from those that were not used in the profile. This process yielded a corpus of 1,892 documents[9]. We then used a ten-fold cross-validation with the Random Forest algorithm. Table 4 shows the results for this experiment.

Table 4: F-measure and Standard Deviation for the Level Classification in the Reception Corpus

	Profile	**Vocabulary**	**Readability**	**All**
Beginner	0.891 (0.053)	*0.903 (0.042)	0.878 (0.049)	**0.922** (0.039)
Intermediate	0.861 (0.000)	0.916 (0.000)	0.853 (0.000)	**0.921** (0.000)
Advanced	0.914 (0.000)	**0.984** (0.000)	0.932 (0.000)	0.982 (0.000)
Average	0.889 (0.033)	*0.934 (0.027)	0.888 (0.031)	**0.942** (0.026)

Values in bold reflect the best scores in each level and the average best score. The values with a star (*) were not significantly different from the best scores with a confidence of 99%.

The results of this experiment were not much different from the one that we carried out with the production corpus. We saw a general improvement of the scores, achieving a non-weighted average F-measure of 0.942 (0.940, if weighted). But here the vocabulary features excelled, presenting an average result similar to the combined model, and significantly exceeding the combined model for the advanced class. The language profile alone fared less well for the classification of reception texts, but still beat the readability features for the beginner and intermediate classes.

7 Conclusion

This study aimed at developing a language profile for second language acquisition that considers both productive and receptive knowledge, so that the information from the profile could be used as a helping tool for the learners to write a more naturally sounding text. We also hypothesized that, by adding

[9]Many of the documents in the corpus are duplicated due to the ten random samples that were selected.

grammatical information to already tested and proven vocabulary and readability features, we would achieve better results in terms of SLA-related tasks.

For achieving these goals, we used two corpora, one representing the productive knowledge and the other the receptive knowledge. We annotated both corpora with pedagogically relevant grammatical information and calculated a divergence score for each of the grammatical structures to find out which ones were different in the comparison of both corpora.

The results of the divergence score were further improved by an analysis of overuse or underuse of structures in the production corpus in relation to the production corpus. This improvement to the divergence score resulted in a grammatical profile that shows where and how the written texts from language learners differ from reception texts. The structures shown in Table 1 also present clues for grammatical structures that are possibly too easy or for those that remain a difficulty throughout the learning process.

By conceiving experiments for the classification of production and reception texts, we could observe that our profile can very well model the written texts, achieving results that are comparable to those of true and tested vocabulary and readability features. Regarding our hypothesis, the addition of grammar to lexical information did yield the best results in the production corpus, but the lexical information alone was enough to achieve the same result as the combined approach in the reception corpus.

It is important to emphasize that our results for the classification experiment in terms of the reception corpus are competitive with the state of the art. For instance, Xia et al. (2016), by means of a profound analysis of parameters, achieved an F-measure of 0.845 using readability features and a lexical distribution similar to our vocabulary features. The same is true for the classification experiment regarding the production corpus, since we achieved similar results to those presented by Wilkens et al. (2018).

In terms of the scoring experiment, we observed that scoring is a very challenging task and that there is still need for further study before we can get good results. Even so, we did get some encouraging results, reaching almost 0.7 correlation for the advanced class in a task that is known to present high discordance among evaluators.

As future work, we intend to increase the size of the reception corpus by including texts from didactic material that is used in second language classes. We are also going to develop a system for supporting the writing of texts that takes as basis our grammar profile, by detecting deviating behavior of grammatical structures in texts produced by the user.

Acknowledgements

The authors would like to thank the Walloon Region (Projects BEWARE n. 1510637 and 1610378) for support, and Altissia International for research collaboration.

References

Shlomo Argamon, Moshe Koppel, James W Pennebaker, and Jonathan Schler. 2009. Automatically profiling the author of an anonymous text. *Communications of the ACM*, 52(2):119–123.

Guy Aston and Lou Burnard. 1998. *The BNC handbook: exploring the British National Corpus with SARA*. Capstone.

Sean Banville. 2009. Breaking news english. *Teaching english as Second or Foreign Language*, 13(1).

Douglas Biber. 1991. *Variation across speech and writing*. Cambridge University Press.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Thomas Cobb. 2013. Frequency 2.0: Incorporating homoforms and multiword units in pedagogical frequency lists. *L2 vocabulary acquisition, knowledge and use: New perspectives on assessment and corpus analysis*, pages 79–108.

Ido Dagan, Lillian Lee, and Fernando Pereira. 1997. Similarity-based methods for word sense disambiguation. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 56–63. Association for Computational Linguistics.

Edgar Dale and Jeanne S Chall. 1948. A formula for predicting readability: Instructions. *Educational research bulletin*, pages 37–54.

Dominik Maria Endres and Johannes E Schindelin. 2003. A new metric for probability distributions. *IEEE Transactions on Information theory*, 49(7):1858–1860.

May Fan. 2000. How big is the gap and how to narrow it: an investigation into the active and passive vocabulary knowledge of l2 learners. *An investigation into the active*.

Bent Fuglede and Flemming Topsoe. 2004. Jensen-shannon divergence and hilbert space embedding. In *Information Theory, 2004. ISIT 2004. Proceedings. International Symposium on*, page 31. IEEE.

Jeroen Geertzen, Theodora Alexopoulou, and Anna Korhonen. 2013. Automatic linguistic annotation of large scale l2 databases: The ef-cambridge open language database (efcamdat). In *Proceedings of the 31st Second Language Research Forum. Somerville, MA: Cascadilla Proceedings Project*.

Robert Gunning. 1952. The technique of clear writing.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Technical Training Command Millington TN Research Branch.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Solomon Kullback. 1997. *Information theory and statistics*. Courier Corporation.

Batia Laufer. 1998. The development of passive and active vocabulary in a second language: same or different? *Applied linguistics*, 19(2):255–271.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Ron Martinez. 2011. *The development of a corpus-informed list of formulaic sequences for language pedagogy*. Ph.D. thesis.

Paul Meara. 1990. A note on passive vocabulary. *Interlanguage studies bulletin (Utrecht)*, 6(2):150–154.

BQ Morgan and Lydia M Oberdeck. 1930. Active and passive vocabulary. *Studies in modern language teaching*, pages 213–221.

Michael P Oakes and Malcolm Farrow. 2007. Use of the chi-squared test to examine vocabulary differences in english language corpora representing seven different countries. *Literary and linguistic computing*, 22(1):85–99.

Michael P Oakes. 2008. Statistical measures for corpus profiling. In *Proceedings of the Open University Workshop on Corpus Profiling, London, UK (October 2008)*.

Ferdinand Österreicher and Igor Vajda. 2003. A new class of metric divergences on probability spaces and its applicability in statistics. *Annals of the Institute of Statistical Mathematics*, 55(3):639–653.

Virginie Pignot-Shahov. 2012. Measuring l2 receptive and productive vocabulary knowledge. *Language Studies Working Papers*, 4(1):37–45.

Norbert Schmitt. 2008. Instructed second language vocabulary learning. *Language teaching research*, 12(3):329–363.

Norman Verhelst, Piet Van Avermaet, Sauli Takala, Neus Figueras, and Brian North. 2009. *Common European Framework of Reference for Languages: learning, teaching, assessment*. Cambridge University Press.

Rodrigo Wilkens, Leonardo Zilio, and Cédrick Fairon. 2018. Sw4all: a cefr classified and aligned corpus for language learning. In *Proceedings of the 11th Language Resources and Evaluation Conference*.

Menglin Xia, Ekaterina Kochmar, and E Briscoe. 2016. Text readability assessment for second language learners.

Leonardo Zilio and Cédrick Fairon. 2017. Adaptive system for language learning. In *Advanced Learning Technologies (ICALT), 2017 IEEE 17th International Conference on*, pages 47–49. IEEE.

Leonardo Zilio, Rodrigo Wilkens, and Cédrick Fairon. 2017a. Enhancing grammatical structures in web-based texts. In *Proceedings of the 25th EUROCALL*, pages 839–846. Accepted.

Leonardo Zilio, Rodrigo Wilkens, and Cédrick Fairon. 2017b. Using nlp for enhancing second language acquisition. In *Proceedings of Recent Advances in Natural Language Processing*, pages 839–846.

Leonardo Zilio, Rodrigo Wilkens, and Cédrick Fairon. 2018. An sla corpus annotated with pedagogically relevant grammatical structures. In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France, may. European Language Resources Association (ELRA).

iParaphrasing: Extracting Visually Grounded Paraphrases via an Image

Chenhui Chu,[1] **Mayu Otani**[2] **and Yuta Nakashima**[1]
[1]Institute for Datability Science, Osaka University
[2]CyberAgent, Inc.
chu,n-yuta@ids.osaka-u.ac.jp, otani_mayu@cyberagent.co.jp

Abstract

A paraphrase is a restatement of the meaning of a text in other words. Paraphrases have been studied to enhance the performance of many natural language processing tasks. In this paper, we propose a novel task *iParaphrasing* to extract *visually grounded paraphrases (VGPs)*, which are different phrasal expressions describing the same visual concept in an image. These extracted VGPs have the potential to improve language and image multimodal tasks such as visual question answering and image captioning. How to model the similarity between VGPs is the key of iParaphrasing. We apply various existing methods as well as propose a novel neural network-based method with image attention, and report the results of the first attempt toward iParaphrasing.

1 Introduction

A paraphrase is a restatement of the meaning of a word, phrase, or sentence within the context of a specific language (e.g., "a red jersey" and "a red uniform shirt" in Figure 1 are paraphrases) (Bhagat and Hovy, 2013). Paraphrases have been exploited for natural language understanding, and shown to be very effective for various natural language processing (NLP) tasks, including question answering (Riezler et al., 2007), summarization (Zhou et al., 2006), machine translation (Chu and Kurohashi, 2016), text normalization (Ling et al., 2013), textual entailment recognition (Androutsopoulos and Malakasiotis, 2010), and semantic parsing (Berant and Liang, 2014).

In this paper, we propose a novel task named, *iParaphrasing*, to extract *visually grounded paraphrases (VGPs)*. We define VGPs as different phrasal expressions that describe the same visual concept in an image. Nowadays, with the spread of the web and social media, it is easy to collect large amounts of images with their describing text. For example, different news sites release news with the same topic using the same image; photos with many comments are posted to social networking sites and blogs. As these describing texts are written by different people but about the same image, there are potentially large amounts of VGPs in the describing text (Figure 1). We aim to accurately extract these paraphrases using the image as a pivot to associate different phrases.

The extracted VGPs can be applied to various computer vision (CV) and NLP tasks, such as image captioning (Vinyals et al., 2015) and visual question answering (VQA) (Wu et al., 2017), for the better understanding of both images and languages. For example, a VQA system must understand queries of different expressions about the same visual concept (e.g., "a male" and "the pitcher" in Figure 1) in order to answer a question properly. VGPs can also be applied to the evaluation of image captioning systems in the similar way as paraphrases have been applied for machine translation evaluation (Snover et al., 2009).

As a pioneering study, we work on iParaphrasing on the Flickr30k entities dataset (Plummer et al., 2015). This dataset contains 30k images with 5 captions per image annotated via crowdsourcing, which can be seen as a very small subset of the data available in the web and social media. Figure 1 shows an example image together with its five captions taken from this dataset. In the Flickr30k entities dataset, entities (i.e., noun phrases) in the captions have been manually aligned to their corresponding image

Proceedings of the 27th International Conference on Computational Linguistics, pages 3479–3492
Santa Fe, New Mexico, USA, August 20-26, 2018.

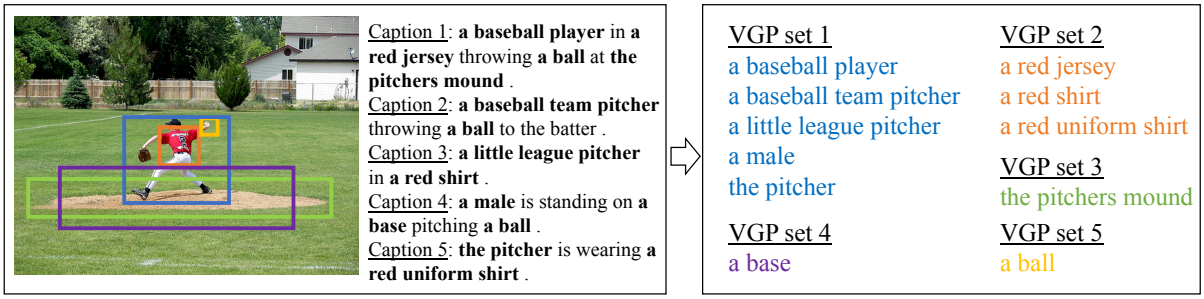


Figure 1: An example from the Flickr30k entities dataset, in which an image is described by five captions (entities in the captions are marked in bold). Our task is to extract the entities that describe the same visual concept (represented as an image region) in the image as VGPs. Note that the image regions are not given as input but are drawn here for comprehensibility.

regions (Plummer et al., 2015). Therefore, we can obtain a set of phrases annotated with the same image region. This set of phrases are used as the ground truth VGPs in our study. The goal of this work is to extract these VGPs.

We formulate our task as a clustering task (Section 3), where the similarity between each entity pair is crucial for the performance. We apply many different unsupervised similarity computation methods (Section 4) including phrase localization-based similarity (Plummer et al., 2017) (Section 4.1), translation probability-based similarity (Koehn et al., 2007) (Section 4.2), and embedding-based similarity (Mikolov et al., 2013; Klein et al., 2014; Plummer et al., 2015) (Section 4.3). In addition, we propose a supervised neural network (NN)-based method using both textual and visual features to explicitly model the similarity of an entity pair as VGPs (Section 5). Experiments show that our proposed NN-based method outperforms the other methods.[1]

2 Related Work

2.1 Paraphrase Extraction

Previous studies extract paraphrases from either monolingual corpora or bilingual parallel corpora. One major approach is to use the distributional similarity (Harris, 1954) with regular monolingual corpora (a large collection of text in a single language) (Lin and Pantel, 2001; Bhagat and Ravichandran, 2008; Marton et al., 2009), or monolingual comparable corpora (a set of monolingual corpora that describe roughly the same topic in the same language) (Barzilay and Lee, 2003; Chen and Dolan, 2011). Distributional similarity stems from the distributional hypothesis (Harris, 1954), stating that words/phrases that share similar meanings should appear in similar distributions. This approach sometimes suffers from noisy results, because the distributed similarity often maps antonyms to closer points. Some methods try to extract paraphrases from monolingual parallel corpora (a collection of sentence level paraphrases) (Arase and Tsujii, 2017; MacCartney et al., 2008), but such monolingual parallel corpora are rarely available.

Bilingual parallel corpora (a collection of sentence-aligned bilingual text) enjoys more availability than monolingual parallel corpora as they are mandatory for training machine translation systems. Bilingual parallel corpora can be used for paraphrase extraction, with bilingual pivoting (Bannard and Callison-Burch, 2005). This method assumes that two source phrases are a paraphrase pair if they are translated to the same target phrase. Bilingual pivoting has been further refined by using syntax information (Callison-Burch, 2008) or mutual information (Kajiwara et al., 2017). These methods have led to the construction of a multilingual paraphrase database (Ganitkevitch and Callison-Burch, 2014).

Note that our definition of paraphrases may look different from the studies mentioned above, as our paraphrases are a set of noun phrases that represent the same visual concept. Our idea to extract paraphrases under this definition is to use image captioning datasets (Young et al., 2014; Chen et al., 2015),

[1]Codes and data for reproducing the results reported in this paper are available at `https://github.com/ids-cv/coling_iparaphrasing`

which usually contain several captions for each image, and currently scale to sub-million images, instead of a bilingual parallel corpus with limited availability. To the best of our knowledge, this is the first study that aims to extract paraphrases from such multimodal datasets consisting of images and their captions.[2]

2.2 Coreference Resolution

Coreference resolution is a task to find the expressions that refer to the same entity in a text (Soon et al., 2001; Lee et al., 2017). Our task in this paper focuses on extracting entities that describe the same visual concept, making the formulation similar to coreference resolution. Our task differs from conventional coreference resolution that it requires visual grounding. In addition, the targets of coreference resolution are the entities in a sentence or a document, while our targets are the entities in the captions of an image that are quasi-paraphrases but are not related to each other in discourse level like sentences in a document. For coreference resolution, the context in a sentence or discourse information in a document are crucial, but discourse information does not exist in our task.[3] In the context of vision and language tasks, Konget al. (2014) used noun/pronoun coreference resolution in sentential descriptions of RGB-D scenes for improving 3D semantic parsing. The texts being handled in their work are either sentences or documents, and their targets are limited to noun words and pronouns but we extract noun phrases. Because our goal is not limited to entities but arbitrary phrases, we believe that comparing to coreference resolution, iParaphrasing is a more forethoughtful name to define our task for future research.

2.3 Phrase Localization

Phrase localization is a task to find an image region that corresponds to a given phrase in a caption, which is closely related to our VGP extraction task. Plummer et al. (2015) pioneered this work, in which they annotated phrase-region alignment in the Flickr30k image-caption dataset (Young et al., 2014) and released it as the Flickr30k entities dataset. They also proposed a method based on canonical correlation analysis (CCA) (Hardoon et al., 2004) that learns joint embeddings of phrases and image regions for associating them. Wang et al. (2016a) proposed joint embeddings using a two-branch NN. Fukui et al. (2016) used a multimodal compact bilinear pooling method to combine textual and visual embeddings. Rohrbach et al. (2016) proposed a convolutional NN (CNN)-recurrent NN (RNN)-based method for this task. They learn to detect a region for a given phrase and then reconstruct the phrase using the detected region. Wang et al. (2016b) noticed that the relationships between phrases should agree with their corresponding regions, and proposed a joint matching method, but their method only considers the "has-a" relationship that is explicitly indicated by possessive pronouns. Previous studies rely on region proposal to produce a number of region candidates for phrase localization, Yeh et al. (2017) proposed a unified framework that can search over all possible regions. Plummer et al. (2017) used spatial relationships between pairs of entities connected by verbs or prepositions, which achieved the state-of-the-art performance. In this paper, we use the current state-of-the-art phrase localization method of (Plummer et al., 2017) as a baseline for VGP extraction.

2.4 Other Vision and Language Tasks

Vision and language tasks have been a hot research area recently in both the CV and NLP communities. Various efforts have been made for many multimodal tasks such as image object/region referring expression grounding (Kazemzadeh et al., 2014; Mao et al., 2016; Hu et al., 2017; Cirik et al., 2018) visual captioning (Vinyals et al., 2015; Xu et al., 2015; Bernardi et al., 2016; Laokulrat et al., 2016), text-image retrieval (Otani et al., 2016), visual question answering (Wu et al., 2017), visual dialog (Das et al., 2017a; Das et al., 2017b) and video event detection (Phan et al., 2016). Some researchers also have employed images for improving NLP tasks, such as multimodal machine translation (Specia et al., 2016), cross-lingual document retrieval (Funaki and Nakayama, 2015), and textual entailment recognition (Han

[2] Although (Plummer et al., 2015) annotated the VGPs in the Filickr30k entities dataset, they did not propose any methods to extract them. (Regneri et al., 2013) collected sentence level paraphrases by aligning video scripts with the same time frame; these sentence level paraphrases are essentially similar to captions of an image.

[3] If we treat multiple captions as a document forcibly, we could apply a coreference resolution approach for our task.

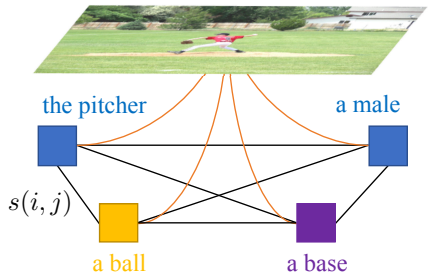


Figure 2: An overview of our VGP extraction formulation. We extract VGP via clustering, where the entity-entity similarity $s(i, j)$ is the key. We compare both unsupervised and supervised methods using entity-image and entity-entity associations for computing this similarity.

et al., 2017). iParaphrasing is a novel CV+NLP task, which to the best of our knowledge has not been studied before and can boost the performance of various multimodal and NLP tasks.

3 Paraphrase Extraction via Clustering

We formulate the paraphrase extraction from the Flickr30k entities dataset as a clustering task. Given an image and all the entities in the corresponding captions, the task is to cluster the entities[4] to its corresponding visual concepts represented as image regions. The number of clusters (i.e., the number of paraphrase sets in a set of an image and captions) is not explicitly given in our task. Therefore, we apply the affinity propagation algorithm (Frey and Dueck, 2007) to cluster entities, which can estimate the number of clusters as well.

Affinity propagation creates clusters by iteratively sending two types of messages between pairs of entities until convergence. The first type is the responsibility $r(i, j)$ sent from entity i to candidate representative entity j, indicating the strength that entity j should be the representative entity for entity i, which is defined as:

$$r(i,j) \leftarrow s(i,j) - \max_{\forall j' \neq j} \{a(i,j') + s(i,j')\} \tag{1}$$

where $s(i, j)$ is the similarity between entities i and j. The second type is the availability $a(i, j)$ sent from candidate representative entity j to entity i, indicating to what degree that candidate representative entity j is the cluster center for entity i, which is defined as:

$$a(i,j) \leftarrow \min\Big\{0, r(j,j) + \sum_{\forall i' \notin \{i,j\}} \max\{0, r(i',j)\}\Big\} \tag{2}$$

At the beginning, the values of $r(i, j)$ and $a(i, j)$ are set to zero, and they are updated in every iteration until convergence. We optimize the number of clusters on a validation split by adjusting the preference (i.e., self similarity $s(i, i)$) of affinity propagation.

Figure 2 shows an overview of our formulation, where the similarity between the entities is the key. We apply various unsupervised methods for computing this similarity, and propose a supervised NN-based model.

4 Unsupervised Similarity Methods

We apply phrase localization for modeling the entity-entity similarity based on entity-image association (Section 4.1). In addition, we apply various methods for modeling the entity-entity similarity directly (Sections 4.2, and 4.3).

[4]In this paper, we assume that entities are given. In the case that entities are not given, we can easily extract them by chunking the noun phrases.

4.1 Phrase Localization-Based Similarity

The similarity between entities i and j is defined as:

$$s(i,j) = \sum_{r_m \in R} p(i|r_m)p(j|r_m) \tag{3}$$

where R is a set of image regions that are aligned to both entities i and j obtained with the phrase localization method of (Plummer et al., 2017); $p(i|r_m)$ is the localization probability of r_m for i, defined as:

$$p(i|r_m) = \frac{l(i,r_m)}{\sum_{r_m \in R} l(i,r_m)} \tag{4}$$

where $l(i,r_m)$ is the localization score of region r_m for entity i obtained using the method of (Plummer et al., 2017).

4.2 Translation Probability-Based Similarity

The similarity between entities i and j is defined as:

$$s(i,j) = p(i|j)p(j|i) \tag{5}$$

where $p(i|j)$ and $p(j|i)$ are the direct and inverse translation probabilities of an entity pair i and j, which are calculated using a conventional statistical machine translation (SMT) (Koehn et al., 2007) method:

1. Generate a pseudo parallel corpus using the captions in the dataset, which treats the 5 captions for each image as monolingual parallel sentences and pair each of the sentences that leads to $\binom{5}{2} = 10$ sentence pairs per image.

2. Apply word alignment to the parallel corpus using IBM alignment models (Brown et al., 1993) in two directions with the grow-diag-final-and heuristic (Koehn et al., 2007) to align the words in each caption pair.

3. From the word-aligned parallel corpus, extract entity pairs such that the words inside an entity pair are aligned. Then $p(i|j)$ and $p(j|i)$ are calculated as follows:

$$p(i|j) = \frac{c(i,j)}{\sum_k c(i,k)}, \quad p(j|i) = \frac{c(i,j)}{\sum_k c(j,k)} \tag{6}$$

   where $c(i,j)$ is the number of co-occurrence of i and j in the word-aligned corpus.

4.3 Embedding-Based Similarity

In this method, the similarity between entities i and j is defined as:

$$s(i,j) = \frac{\mathbf{t}_i^\top \mathbf{t}_j}{\|\mathbf{t}_i\|\|\mathbf{t}_j\|} \tag{7}$$

where $\mathbf{t}_i$ and $\mathbf{t}_j$ are the phrase embeddings of i and j. We compare three different methods for phrase embeddings.

4.3.1 Word Embedding Average

We represent each word with a 300 dimensional word2vec (Mikolov et al., 2013) vector pre-trained on the Google News corpus.[5] We remove stop words in each entity, and calculate the representation of each entity using the average of all word embeddings.

[5] https://github.com/mmihaltz/word2vec-GoogleNews-vectors

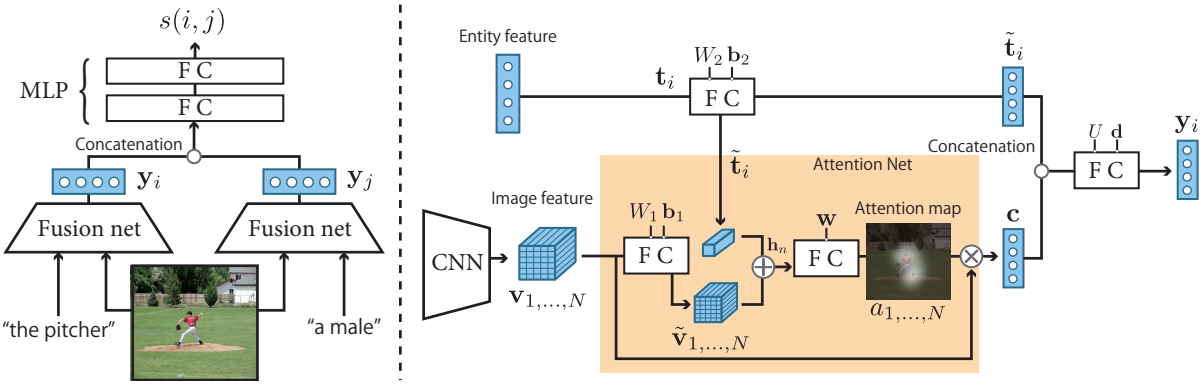


Figure 3: Our supervised NN with image attention-based similarity model (left) and its fusion sub-network (right).

4.3.2 Fisher Vector

Fisher vector is a pooling over word2vec vectors of individual words (Klein et al., 2014), which has been used in the phrase localization task for representing the entities (Plummer et al., 2015). To compute the Fisher vector for an entity, we represent the entity by the HGLMM Fisher vector encoding (Klein et al., 2014) of the word vectors, following (Plummer et al., 2015).[6]

4.3.3 Fisher Vector with CCA

Projecting the feature vectors of image regions and entities to a shared semantic space can provide strong associations between the image regions and entities, which has the potential to improve the performance of VGP extraction. Therefore, we learn a CCA projection on the Flickr30k entities dataset for the image region feature vectors and entity feature vectors with (Plummer et al., 2015), in which the normalized CCA formulation of (Gong et al., 2014) is used. The columns of the CCA projection matrices are scaled by the eigenvalues, and the feature vectors are projected by these matrices and normalized to the dimensionality of 4,096. The image region feature vectors are extracted using Faster R-CNN (Ren et al., 2015).[7] We use Fisher vectors for entity feature vectors.

5 Supervised Similarity Model Based on Neural Network with Image Attention

We propose a NN-based supervised model. This model computes the similarities of entity pairs as VGPs by explicitly modeling the associations between them and an image. Figure 3 illustrates our proposed NN model.[8] Given an entity pair and its corresponding image, we construct two separated *fusion nets* for each entity (Figure 3 (right)). Note that parameters of these two fusion net are shared. A fusion net represents an entity with a concatenation of its entity feature vector and visual context vector. The visual context vector is computed with an attention mechanism, indicating to which part of the image should be paid attention, in order to judge whether the entity pair is VGP or not. The outputs of the two fusion nets are then fed into a multilayer perceptron (MLP) to compute the similarity of the two entities.

Formally, let V be a 196×512 feature map[9] extracted from the `conv5_3` layer in the VGG-16 network (Simonyan and Zisserman, 2015) for an input image; $\mathbf{v}_n$ is a 512 dimensional vector at position n of V. Given an entity feature vector $\mathbf{t}_i$ and $\mathbf{v}_n$, we first transform them with fully connected (FC) layers whose unit sizes are 512:

$$\tilde{\mathbf{v}}_n = \mathrm{norm}_{\mathrm{L2}}(W_1\mathbf{v}_n + \mathbf{b}_1), \quad \tilde{\mathbf{t}}_i = \mathrm{norm}_{\mathrm{L2}}(W_2\mathbf{t}_i + \mathbf{b}_2) \tag{8}$$

[6]The Fisher vector is constructed with 30 centers of both first and second order information, which results in a very sparse vector whose dimensionality is $300 \times 30 \times 2 = 18000$. Therefore, we apply principal component analysis (PCA) to convert it to a lower dimensionality of 4,096.

[7]https://github.com/ShaoqingRen/faster_rcnn

[8](Yin et al., 2016) proposed a CNN network with attention for sentence level paraphrase identification; our model differs from theirs that we fuse both textual and visual information while theirs is a text only model.

[9]An image is split into $14 \times 14 = 196$ sub-images, and represented as a 196×512 feature map.

where $\mathrm{norm}_{\mathrm{L2}}(\cdot)$ indicates L2 normalization to an input vector. We then compute an attention value a_n for $\mathbf{v}_n$ as:

$$\mathbf{h}_n = \mathrm{relu}(\tilde{\mathbf{v}}_n + \tilde{\mathbf{t}}_i), \quad e_n = \mathbf{w}^\top \mathbf{h}_n, \quad a_n = \frac{\exp(e_n)}{\sum_{n=1}^{N} \exp(e_n)} \tag{9}$$

where $N = 196$. After obtaining a_n, we fuse a visual and an entity feature vector to $\mathbf{y}_i$ as:

$$\mathbf{c} = \sum_{n=1}^{N} a_n \mathbf{v}_n, \quad \mathbf{y}_i = U[\mathrm{norm}_{\mathrm{L2}}(\mathbf{c}), \tilde{\mathbf{t}}_i] + \mathbf{d} \tag{10}$$

where $[\cdot, \cdot]$ indicates the concatenation of two vectors, $\mathbf{c}$ is a visual context vector. We compute fusion feature vectors $\mathbf{y}_i$ and $\mathbf{y}_j$ with the corresponding image. Finally, we feed them to a two-layer MLP network with ReLU non-linearity, whose unit sizes are 128 and 1, respectively, to produce the similarity of the entity pair.

6 Experiments

6.1 Settings

We conducted experiments on the Flickr30k entities dataset (Plummer et al., 2015). This dataset contains 31,837 images, which is described with 5 captions annotated via crowdsourcing. We followed the 29,873 training, 1,000 validation, and 1,000 test image splits used in the phrase localization task (Plummer et al., 2015). Our task is to automatically cluster the entities in the captions that describe the same visual concept (i.e., region in the dataset) in the image as VGPs. Entities that share the same ID and group type (e.g., "a red jersey," "a red shirt" and "a red uniform shirt" in Figure 1 share the same entity ID and group type "/EN#19026/clothing") are treated as the ground truth VGP clusters in our evaluation.[10] As stop words should not be considered for computing the entity similarities, we preprocessed the entities in the dataset by removing stop words for all the methods.

We evaluated both clustering and pairwise performance.[11] The entity clustering performance for each image was measured with adjusted Rand index (ARI) (Hubert and Arabie, 1985). We used the implementation in the Scikit-learn machine learning toolkit (Thirion et al., 2011)[12] for computing ARI. We report the mean of ARI scores for all the images in the test split. To evaluate the performance for clustering, we optimized the number of clusters by adjusting the preference for affinity propagation on the validation split to maximize the ARI using the Bayesian optimization algorithm (Mockus, 1989) implemented in GPyOpt.[13] The pairwise performance was evaluated with precision, recall, and F-score, defined as:

$$\mathrm{precision} = \frac{\#\mathrm{predicted\_positive}}{\#\mathrm{predicted}}, \mathrm{recall} = \frac{\#\mathrm{predicted\_positive}}{\#\mathrm{true\_positive}}, \mathrm{F\text{-}score} = 2 \times \frac{\mathrm{precision} \times \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} \tag{11}$$

where an entity pair with a similarity higher than a threshold is treated as $predicted$, which is compared against the ground truth to judge whether it is $predicted\_positive$ or not. We report the performance using the similarity threshold tuned on the validation split that maximizes the F-score.

We used the affinity propagation implementation[14] in Scikit-learn for clustering. We compared the performance of the different similarity methods described in Sections 4 and 5, where the detailed settings for the methods were as follows:

[10] There is an entity type named "notvisual" in the dataset (e.g., "the batter" in Figure 1), which means this entity has no corresponding visual regions in the image. In our evaluation, we excluded this "notvisual" type, because all entities that are not visual are annotated with the same entity ID and thus ground truth VGPs for these "notvisual" entities are unavailable in the dataset. There are entity pairs in the dataset that are the same after removing the stop words (e.g., "a man" and "the man"), we treated them as one entity for evaluation. In addition, entities that do not have corresponding regions in the image were excluded from evaluation.

[11] We did not report phrase localization accuracies for our proposed NN-based supervised model, because attention is different from phrase localization. Instead of determining the best region corresponding to the given phrases, attention provides attention probabilities to the 196 sub-images, which cannot be used for evaluation directly. A soft-accuracy metric could be reported for discussion, but this metric is not directly comparable to previous phrase localization studies.

[12] http://scikit-learn.org/stable/modules/clustering.html#adjusted-rand-index

[13] https://github.com/SheffieldML/GPyOpt

[14] http://scikit-learn.org/stable/modules/generated/sklearn.cluster.AffinityPropagation.html

- Phrase localization (PL): we used the pl-clc toolkit,[15] which is an implementation of the localization method of (Plummer et al., 2017). For R in equation 3, we used the top 30 localization candidates for each entity. The localization scores for each entity and region pair obtained with (Plummer et al., 2017) were used to compute the similarity.

- Translation probability (TP): to get the entity translation probabilities, we first applied the GIZA++ toolkit[16] that is an implementation of the IBM alignment models (Brown et al., 1993) on the pseudo parallel corpus, and then a phrase table was extracted and the phrasal translation probabilities were calculated using state-of-the-art SMT toolkit Moses (Koehn et al., 2007).

- Word embedding average (WEA): see the detailed setting in Section 4.3.1.

- Fisher vector (FV): entity feature vectors were computed using the Fisher vector toolkit released by the authors,[17] following the settings described in Section 4.3.2.

- Fisher vector w/ CCA (FV+CCA): image region feature vectors and entity feature vectors were projected into a 4,096 dimensional space CCA trained on the training split of the Flickr30k entity dataset (Section 4.3.3).

- Supervised NN (SNN): to show the effectiveness of the fusion net (Section 5), we compared a supervised NN-based setting that only feeding the entity feature vectors to the MLP (Figure 3 (left)) for paraphrase similarity prediction. This setting only uses entity feature vectors as input for the NN. It was trained on the training split of the Flickr30k entity dataset. We used all the ground truth VGP pairs in the training split as positive instances. During training, we constructed mini-batches with 15% of positive instances and 85% of randomly sampled negative instances. We used Adam for optimization with a mini-batch size of 300 and weight decay of 0.0001. The learning rate was initialized to 0.01, which was halved at every epoch. We used sigmoid cross entropy loss. We terminated training after 5 epochs, where we observed the loss converged on the validation split. For the entity feature vectors, we compared three different settings described above namely: WEA, FV, and FV+CCA.

- SNN+image: this setting is for our proposed supervised NN-based method described in Section 5. We again compared the three different entity feature vectors. We used VGG-16 (Simonyan and Zisserman, 2015) for the image features. The model was trained with the same configuration as the SNN setting.

- Ensemble: the ensemble of the SNN and SNN+image models that takes the average similarity given by both models. The motivation of this setting is to complement these two models to each other.

6.2 Results

Table 1 shows the results of all the different methods. We report the performance based on the entity types to better understand the performance difference of each method, i.e., "all" evaluates on all entities, whereas "single" and "multi" only evaluate on entities with one single token and multiple tokens, respectively, after removing stop words. For the unsupervised methods, we can see that PL does not show good performance. This is due to the low performance of phrase localization.[18] TP shows a fairly high F-score, but a very low ARI score. The reason for this is that the translation probabilities are computed based on word alignment, leading to a similarity score of 0 to the unrelated entity pairs, which is not suitable for affinity propagation. WEA shows relatively good performance that is better than FV. This is because 45.84% of the entities in our task are single word type after removing the stop words, and converting the low dimensional word embedding to high dimensional and sparse Fisher vectors is harmful

[15] https://github.com/BryanPlummer/pl-clc

[16] http://code.google.com/p/giza-pp

[17] https://owncloud.cs.tau.ac.il/index.php/s/vb7ys8Xe8J8s8vo

[18] Although (Plummer et al., 2017) is the current state-of-the-art for phrase localization, the accuracy is only 55.85%.

Method	ARI all / single / multi	Precision all / single / multi	Recall all / single / multi	F-score all / single / multi
PL	43.23 / 45.92 / 46.35	59.32 / 51.53 / 62.86	63.12 / 47.99 / 74.14	61.16 / 49.70 / 68.04
TP	37.61 / 50.32 / 36.79	66.23 / 63.20 / **82.17**	64.20 / 66.10 / 56.31	65.20 / 64.62 / 66.83
WEA	49.82 / 47.16 / 49.58	61.51 / 46.11 / 62.84	71.29 / 67.93 / 78.47	66.04 / 54.93 / 69.79
FV	39.85 / 43.55 / 41.26	63.79 / 40.84 / 67.51	60.87 / 35.41 / 77.03	62.30 / 37.94 / 71.96
FV+CCA	54.91 / 51.01 / 49.30	64.79 / 55.79 / 68.24	82.20 / 75.83 / 84.98	72.46 / 64.28 / 75.69
SNN (WEA)	60.23 / 55.06 / 53.26	77.86 / 83.66 / 74.50	84.58 / 75.16 / **88.96**	81.08 / 79.18 / 81.09
SNN+image (WEA)	60.55 / 55.42 / **55.82**	79.47 / 81.01 / 77.26	84.56 / 79.35 / 87.06	81.94 / 80.17 / 81.86
Ensemble (WEA)	60.65 / 54.92 / 54.56	80.65 / 78.68 / 77.38	84.79 / 83.14 / 88.85	82.67 / 80.85 / 82.72
SNN (FV)	48.13 / 45.97 / 47.22	64.21 / 45.92 / 66.40	65.93 / 50.89 / 76.51	65.06 / 48.28 / 71.10
SNN+image (FV)	47.90 / 47.39 / 48.31	63.49 / 52.62 / 66.86	68.20 / 55.62 / 78.01	65.76 / 54.08 / 72.01
Ensemble (FV)	49.82 / 48.16 / 48.34	65.48 / 54.87 / 70.51	71.43 / 56.24 / 76.54	68.33 / 55.55 / 73.40
SNN (FV+CCA)	60.56 / **56.35** / 54.06	**83.11** / **85.19** / 77.44	82.13 / 79.30 / 87.69	82.62 / 82.14 / 82.25
SNN+image (FV+CCA)	61.17 / 54.86 / 54.14	82.51 / 84.52 / 80.28	84.19 / 81.85 / 86.82	83.34 / 83.16 / 83.43
Ensemble (FV+CCA)	**62.35** / 54.98 / 54.84	82.71 / 84.10 / 80.91	**85.67** / **83.50** / 87.06	**84.16** / **83.80** / **83.87**

Table 1: VGP extraction results ("all" evaluates on all entities, "single" and "multi" only evaluate on entities consist of one single token and multiple tokens after removing stop words, respectively; the methods above and below the double line are unsupervised and supervised, respectively).

for these single word entity pairs. However, for the performance of entities containing multiple words, the Fisher vector is better than word embedding average in the perspective of F-score. FV+CCA significantly outperforms FV. This is because it uses visual information in the training split that transforms the entity vectors and visual vectors into the semantic space that is helpful for detecting VGPs.

Regarding the supervised methods, NN-based methods using any entity feature vectors outperforms the methods that uses them in an unsupervised way. The reason for this is that it directly uses the paraphrase supervision in the training split, while the unsupervised methods do not. Using entity representation with better ARI and F-score for the SNN method can achieve better results. Our proposed method (SNN+image) that uses both textual and visual features shows better performance compared to SNN that uses textual features only, indicating that the usage of visual features is helpful for our VGP extraction task.[19] However, the performance improvements are not very large. We discuss the reason for this in detail in Section 6.3.1. The ensemble of SNN and SNN+image further improves the performance, which means that these two models complement each other.

6.3 Discussion

6.3.1 Neural Network w/ and w/o Images

We compared the SNN and SNN+image results, and found that image attention is helpful for identifying people-related paraphrases in about 50% cases, which are difficult to be determined based on the textual information only. In addition, the attention for these people-related paraphrases are well learned. We believe the reason for this is that many entities in the training split are people-related and thus they are well modeled. Figure 4a shows such an example, where the SNN (FV+CCA) model fails to identify these two entities "a group of order men" and "a group of people" as VGPs due to the diverse textual descriptions of the the same visual concept. Our proposed NN+image (FV+CCA) model correctly identifies these VGPs by paying attention to the image region of people in the image. In about 30% cases, the visual information is also helpful for the identification of other types of paraphrases, although the attention is not accurate. Figure 4b shows an example, where the SNN (FV+CCA) model could not identify two entities "a large display of artifacts" and "an art exhibit" as VGPs.

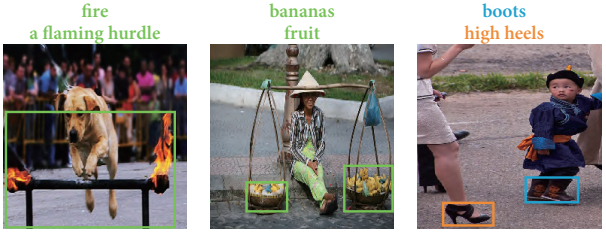
In about 20% cases, visual information could bring negative effects for paraphrase identification. Figure 4c shows an example that "the street window shops" and "a clothing store window" are mistakenly judged as a paraphrase after using the image information while using textual information judges correctly. Although, the attention for these entities refer to the same visual concept in the image, the entities actually refer to different concepts (i.e., "shop" and "window").

[19]The difference between SNN (FV+CCA) and SNN+image (FV+CCA) is whether using visual features for iParaphrasing explicitly or not. SNN (FV+CCA) uses image region features for learning entity features, but it does not use visual features explicitly for iParaphrasing.

(a) An improved example of people-related paraphrases.

(b) An improved example of scene-related paraphrases.

(c) A worsened example of scene-related paraphrases.



(d) Failed examples.

Figure 4: Examples comparing SNN (FV+CCA) with SNN+image (FV+CCA) ((a), (b), and (c); the leftmost images are the original ones, the images in the middle and on the right show attention of the entity pairs on the images, the degree of whiteness indicates the strength of attention, the identification results are shown under the images), and failed examples (d).

6.3.2 Failed Examples

Even the best method, namely Ensemble (FV+CCA), only achieves a ARI of 62.35 and a F-score of 84.16. We found that most false negative examples are sparse entity pairs that describe a image region in an image in a very diverse way, for example "fire" and "a flaming hurdle" (Figure 4d (left)), "bananas" and "fruit" (Figure 4d (middle)). These pairs are difficult not only for using textual features, but also for using image attentions. Most false positive examples are produced by the noisy phrase embedding method. For example, "boots" and "high heels" referring to the shoes on a boy and a lady, respectively, are identified as a paraphrase pair because of their closeness in the embedding space (Figure 4d (right)). Some of the false positive examples are caused by the noise introduced by the wrong attention in an image. For example, "a green snowman" and "his new toy" are attended to the similar image regions.

7 Conclusion

In this paper, we proposed iParaphrasing: a novel task to extract VGPs describing the same visual concept in an image. We not only applied various existing techniques for this task, but also proposed a NN-based method that uses both the textual and visual information to model the similarity between the VGPs. Experiments on the Flickr30k entities dataset showed that we achieved good performance.

For future work, we plan to study a multi task method for both VGP extraction and phrase localization to further improve the performance. We worked on the Flickr30k entities dataset, where noun phrases are given and VGP supervision is available, extracting VGP in an end-to-end manner without supervision in other datasets such as the Microsoft COCO caption dataset (Chen et al., 2015) is a more realistic scenario and could be more interesting. Extracting other types of paraphrases (e.g., prepositional and verb paraphrases) is another possible extension, which requires a much deeper understanding of the relation between phrases and image regions. We also plan to apply the VGPs for CV and NLP multimodal tasks, such as VQA.

Acknowledgement

This work was supported by ACT-I, JST and JSPS KAKENHI No. 18H03264. We are very appreciated to Prof. Kumiyo Nakakoji, Prof. Yuki Arase and Prof. Sadao Kurohashi for the helpful discussion of this paper. We also thank the anonymous reviewers for their insightful comments.

References

Ion Androutsopoulos and Prodromos Malakasiotis. 2010. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research*, 38(1):135–187, May.

Yuki Arase and Jun'ichi Tsujii. 2017. Monolingual phrase alignment on parse forests. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Copenhagen, Denmark, September. Association for Computational Linguistics.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 597–604, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 16–23, Edmonton, May. Association for Computational Linguistics.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1415–1425, Baltimore, Maryland, June. Association for Computational Linguistics.

Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. 2016. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *Journal of Artificial Intelligence Research*, 55(1):409–442, January.

Rahul Bhagat and Eduard Hovy. 2013. What is a paraphrase? *Computational Linguistics*, 39(3):463–472, September.

Rahul Bhagat and Deepak Ravichandran. 2008. Large scale acquisition of paraphrases for learning surface patterns. In *Proceedings of the 46rd Annual Meeting of the Association for Computational Linguistics: the Human Language Technology Conference*, pages 674–682, Columbus, Ohio, June. Association for Computational Linguistics.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312.

Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 196–205, Honolulu, Hawaii, October. Association for Computational Linguistics.

David Chen and William Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 190–200, Portland, Oregon, USA, June. Association for Computational Linguistics.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *CoRR*, abs/1504.00325.

Chenhui Chu and Sadao Kurohashi. 2016. Paraphrasing out-of-vocabulary words with word embeddings and semantic lexicons for low resource statistical machine translation. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 644–648, Portoro, Slovenia, may. European Language Resources Association (ELRA).

Volkan Cirik, Taylor Berg-Kirkpatrick, and Louis-Philippe Morency. 2018. Using syntax to ground referring expressions in natural images. In *The Thirty-Second AAAI Conference on Artificial Intelligence*.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M.F. Moura, Devi Parikh, and Dhruv Batra. 2017a. Visual Dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Abhishek Das, Satwik Kottur, José M.F. Moura, Stefan Lee, and Dhruv Batra. 2017b. Learning cooperative visual dialog agents with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Brendan J Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *Science*, 315(5814):972–976.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 457–468, Austin, Texas, November. Association for Computational Linguistics.

Ruka Funaki and Hideki Nakayama. 2015. Image-mediated learning for zero-shot cross-lingual document retrieval. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 585–590, Lisbon, Portugal, September. Association for Computational Linguistics.

Juri Ganitkevitch and Chris Callison-Burch. 2014. The multilingual paraphrase database. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4276–4283, Reykjavik, Iceland, May. European Language Resources Association (ELRA).

Yunchao Gong, Qifa Ke, Michael Isard, and Svetlana Lazebnik. 2014. A multi-view embedding space for modeling internet images, tags, and their semantics. *International Journal of Computer Vision*, 106(2):210–233, Jan.

Dan Han, Pascual Martínez-Gómez, and Koji Mineshima. 2017. Visual denotations for recognizing textual entailment. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2843–2849, Copenhagen, Denmark, September. Association for Computational Linguistics.

David R. Hardoon, Sandor R. Szedmak, and John R. Shawe-taylor. 2004. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation.*, 16(12):2639–2664, December.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Ronghang Hu, Marcus Rohrbach, Jacob Andreas, Trevor Darrell, and Kate Saenko. 2017. Modeling relationships in referential expressions with compositional modular networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of Classification*, 2(1):193–218, Dec.

Tomoyuki Kajiwara, Mamoru Komachi, and Daichi Mochihashi. 2017. Mipa: Mutual information based paraphrase acquisition via bilingual pivoting. In *Proceedings of the 8th International Joint Conference on Natural Language Processing*, pages 80–89, Taipei, Taiwan, November. Asian Federation of Natural Language Processing.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. 2014. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 787–798. Association for Computational Linguistics.

Benjamin Klein, Guy Lev, Gil Sadeh, and Lior Wolf. 2014. Fisher vectors derived from hybrid gaussian-laplacian mixture models for image annotation. *CoRR*, abs/1411.7399.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

Chen Kong, Dahua Lin, Mohit Bansal, Raquel Urtasun, and Sanja Fidler. 2014. What are you talking about? text-to-image coreference. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June.

Natsuda Laokulrat, Sang Phan, Noriki Nishida, Raphael Shu, Yo Ehara, Naoaki Okazaki, Yusuke Miyao, and Hideki Nakayama. 2016. Generating video description using sequence-to-sequence model with temporal attention. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 44–52, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark, September. Association for Computational Linguistics.

Dekang Lin and Patrick Pantel. 2001. Dirt – discovery of inference rules from text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 323–328, New York, NY, USA. ACM.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2013. Paraphrasing 4 microblog normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 73–84, Seattle, Washington, USA, October. Association for Computational Linguistics.

Bill MacCartney, Michel Galley, and Christopher D. Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 802–811, Honolulu, Hawaii, October. Association for Computational Linguistics.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L. Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June.

Yuval Marton, Chris Callison-Burch, and Philip Resnik. 2009. Improved statistical machine translation using monolingually-derived paraphrases. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 381–390, Singapore, August. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

J. Mockus. 1989. *Bayesian approach to global optimization: theory and applications*. Mathematics and its applications: Soviet series. Kluwer Academic.

Mayu Otani, Yuta Nakashima, Esa Rahtu, Heikkila Janne, and Naokazu Yokoya. 2016. Learning joint representations of videos and sentences with web image search. In *European Conference on Computer Vision (ECCV)*, pages 651–667, October.

Sang Phan, Yusuke Miyao, Duy-Dinh Le, and Shin'ichi Satoh. 2016. Video event detection by exploiting word dependencies from image captions. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3318–3327, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 2641–2649, December.

Bryan A. Plummer, Arun Mallya, Christopher M. Cervantes, Julia Hockenmaier, and Svetlana Lazebnik. 2017. Phrase localization and visual relationship detection with comprehensive image-language cues. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 1928–1937, Oct.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *Transactions of the Association for Computational Linguistics*, 1:25–36.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 91–99. Curran Associates, Inc.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 464–471, Prague, Czech Republic, June. Association for Computational Linguistics.

Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. 2016. Grounding of textual phrases in images by reconstruction. In *European Conference on Computer Vision (ECCV)*, pages 817–834, October.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recoginition. In *International Conference on Learning Representations (ICLR)*, pages 1–14.

Matthew G. Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. 2009. Ter-plus: Paraphrase, semantic, and alignment enhancements to translation edit rate. *Machine Translation*, 23(2-3):117–127, September.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Comput. Linguist.*, 27(4):521–544, December.

Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. 2016. A shared task on multimodal machine translation and crosslingual image description. In *Proceedings of the First Conference on Machine Translation*, pages 543–553, Berlin, Germany, August. Association for Computational Linguistics.

Bertrand Thirion, Edouard Duschenay, Vincent Michel, Gael Varoquaux, Olivier Grisel, Jacob VanderPlas, alexandre granfort, fabian pedregosa, Andreas Mueller, and Gilles Louppe. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164, June.

Liwei Wang, Yin Li, and Svetlana Lazebnik. 2016a. Learning deep structure-preserving image-text embeddings. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5005–5013, June.

Mingzhe Wang, Mahmoud Azab, Noriyuki Kojima, Rada Mihalcea, and Jia Deng. 2016b. Structured matching for phrase localization. In *European Conference on Computer Vision (ECCV)*, pages 696–711, October.

Qi Wu, Damien Teney, Peng Wang, Chunhua Shen, Anthony Dick, and Anton van den Hengel. 2017. Visual question answering: A survey of methods and datasets. *Computer Vision and Image Understanding*, pages 1–20.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France, 07–09 Jul. PMLR.

Raymond Yeh, Jinjun Xiong, Wen-Mei W. Hwu, Minh Do, and Alexander G. Schwing. 2017. Interpretable and globally optimal prediction for textual grounding using image concepts. In *Advances in Neural Information Processing Systems*, pages 1909–1919.

Wenpeng Yin, Hinrich Schutze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

Peter Young, Alice Lai, Micah Hodosh, and Hockenmaier Julia. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association of Computational Linguistics*, 2(1):67–78.

Liang Zhou, Chin-Yew Lin, Dragos Stefan Munteanu, and Eduard Hovy. 2006. Paraeval: Using paraphrases to evaluate summaries automatically. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, Main Conference*, pages 447–454, New York City, USA, June. Association for Computational Linguistics.

MCDTB: A Macro-Level Chinese Discourse TreeBank

Feng Jiang[1], Sheng Xu[1], Xiaomin Chu[1], Peifeng Li[1], Qiaoming Zhu[12], Guodong Zhou[1]
[1]School of Computer Science and Technology, Soochow University, China
[2]Institute of Artificial Intelligence, Soochow University, China
{fjiang, sxu, xmchu}@stu.suda.edu.cn
{pfli, qmzhu, gdzhou}@suda.edu.cn

Abstract

In view of the differences between the annotations of micro and macro discourse relationships, this paper describes the relevant experiments on the construction of the Macro Chinese Discourse Treebank (MCDTB), a higher-level Chinese discourse corpus. Following RST (Rhetorical Structure Theory), we annotate the macro discourse information, including discourse structure, nuclearity and relationship, and the additional discourse information, including topic sentences, lead and abstract, to make the macro discourse annotation more objective and accurate. Finally, we annotated 720 articles with a Kappa value greater than 0.6. Preliminary experiments on this corpus verify the computability of MCDTB.

1 Introduction

In the field of natural language processing, discourse analysis is becoming increasingly important as the object of research gradually shifts from the word level to sentence, event and other semantic aspects. Discourse analysis primarily examines the text coherence and cohesion, including the analysis on structure, nuclearity and relationship. In discourse analysis, discourse refers to a series of continuous clauses, sentences or paragraphs of language as a whole; it not only includes text sequences but also the text structure and the logical relationship between text sequences.

Discourse analysis aims at studying the internal structure of texts and understanding the semantic relation between different text units. The granularity of this research can be clause, sentence, sentence group, paragraph and whole article. Commonly, discourse analysis is divided into microstructure and macrostructure analysis. The former is the study of intra-sentence or inter-sentence discourse relations, while the latter is discourse relation between sentence clusters, paragraphs and chapters (Van Dijk, 1976), which highlights a higher semantic level for text comprehension.

Example 1 is the content of the article chtb0155, which has a title (T) and five paragraphs (P1-P5), i.e., five discourse units. Figure 1 shows the example's macro discourse structure tree. The leaf nodes of the macro discourse structure tree refer to discourse units, and the internal nodes refer to relational nodes, which represent the larger discourse units of the relevant children. When connecting the parent and child nodes, the directed edge indicates that the child is an important discourse unit in the relationship, and the undirected edge indicates that the child is secondary in the relationship. Therefore, in Figure 1, there is an *Evaluation* relation between the discourse units P1 and P2, and they form a higher level unit (relational node) DU1-2. The discourse units P4 and P5 exhibit the *Purpose-Behaviour* relation and form a higher level unit DU4-5. Consequently, DU1-2 and P3 have the *Elaboration* relation and form a higher level unit DU1-3. Finally, DU4-5 is supplementary to DU1-3.

Based on the discourse structure tree shown in Figure 1, the article of Example 1 is easily understood. In the task of automatic summarization, for example, according to this discourse structure tree and the directed edges from the root node to the leaf nodes, the topic sentences of the leaf nodes can be used as the text summary. The summary of Example 1 is a combination of the first paragraph and the second paragraph's topic sentence, which is more appropriate than the simple use of the first paragraph as a summary of the full text.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3493–3504
Santa Fe, New Mexico, USA, August 20-26, 2018.

(T) 中保财险公司为上海提供迄今最大一笔出口信用保险 (The Property Insurance Co. of PICC provides Shanghai with the largest amount of export credit insurance to date)
(P1) 中保财产保险有限公司今天为上海…有限公司向巴西出口…，提供了…出口信用保险。(Today, the Property Insurance Co. of PICC, Ltd., provided an export credit insurance… to the Shanghai … Company, Ltd. for them to export … to Brazil.)
(P2) 这是上海地区迄今提供的最大一笔出口信用保险。 (This is the largest export credit insurance in the Shanghai region to date.)
(P3) 这个出口项目包括…起重机。巴西方面先预付…定金，余下…款项…。中保财险公司…提供. . 服务就是为保证…安全收汇，以达到…目的。 (This export project includes …cranes. The Brazilian side will pay… in advance, with the remaining … payments... PICC …is to ensure that …exchange safely … so as to …)
(P4) 出口信用保险制度是…惯例。中国政府责成中保财产保险有限公司代政府开办…业务，为…. 保险等。(The system of export credit insurance is … internationally. The Chinese government is obliged to set up a business by the government of PICC, for... insurance, etc.)
(P5) 为此，中国中央财政于一九八八年拨…专款作为风险准备金，用于…保险业务。中保财险公司…承保了…出口收汇风险，其中…项目四十多个。 (For this, in 1998 China's Central Treasurer allocated ... as risk preparation funds, to be used … insurance services. The company has underwritten about … in the risk of foreign exchange receipts, including … more than 40 … projects.)

Example 1: Content of chtb0155 (simplified version).

This paper uses the RST style to annotate macro discourse structure, nuclearity and relationship and constructs a Macro Chinese Discourse Treebank (MCDTB) including 720 articles. Compared with micro discourse annotation, macro discourse annotation has different characteristics. Macro discourse units are longer and fuzzier, and the relationship between discourse units is looser and less logical. To this end, we have formulated detailed annotation guidelines and quality assurance strategies. When marking macro discourse structure, nuclearity and relationship, we also annotate the annotation of the macro discourse information, such as topic sentences, lead and abstract, to make the annotation of macro discourse relations more objective and accurate.

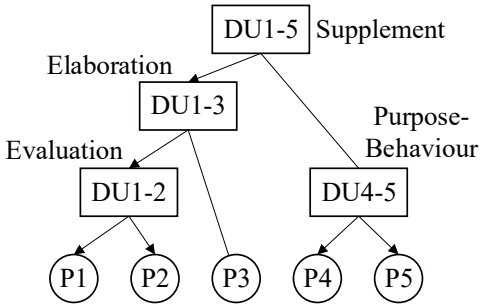


Figure 1: Macro discourse structure tree of chtb0155.

The rest of this paper is organized as follows. Section 2 overviews the related work. Section 3 describes the annotation system of MCDTB. Section 4 introduces our annotation process. Section 5 shows the corpus statistics on MCDTB. Section 6 provides a preliminary experiment. Finally, we present the study's conclusions and outline future work in Section 7.

2 Related Work

Discourse analysis is mainly divided into micro and macro discourse analysis. The theory of micro discourse primarily includes Hobbs model (Hobbs, 1985), rhetorical structure theory (RST) (Mann and Thompson, 1987; Mann et al., 1992), sentence group theory (Wu and Tian, 2000) and complex sentence theory (Fu, 2001), and the macro discourse theory has macro hyper-theme theory (Martin and Rose,

2003) and macro structure theory (Van Dijk, 1980). Under the guidance of these theories, various corpora have been developed accordingly.

In micro discourse analysis, the most popular corpora are RST-DT and PDTB. Based on RST, RST-DT (Marcu et al., 1999; Carlson et al., 2003) contained 385 articles from the Wall Street Journal annotated by 16 categories and 78 classes' rhetorical relations, to represent the relationship between two or more discourse units. Most of the elements of discourse units in RST-DT are phrases including partial clauses. To ensure the universality of the corpus, PDTB (Parsad et al., 2008) uses the LTAG (Lexicalized Tree-Adjoining Grammars) theory, and the annotation is entirely based on lexicalization. This approach primarily annotated the argument structure, including the conjunction and the semantic distinction information. The basic unit of argument is the event or state, and the specific form is the clause or sentence. The corpus is relatively large with 2,159 articles and approximately 1 million words.

Regarding Chinese micro discourse corpus, Zhou et al. (2015) used the PDTB annotation style to annotate 164 Chinese documents from CTB (Xue, 2005; Xue et al., 2005). The experiment demonstrated the transferability of the English discourse analysis theory in Chinese language. Lv et al. (2015) constructed a Chinese corpus of 496 news articles from People's Daily under the framework of the Chinese Framing Network (CFN) (Hao et al., 2007). Each news article annotated the discourse frame, structure and relationship. However, the discourse units are also smaller with a minimum of one sentence and a maximum of five sentences, and their research object is primarily at the word level. Li et al. (2014) integrated RST-DT and PDTB and used the representation method based on the connection dependence tree to annotate the Chinese Dependency Treebank (CDTB) containing 500 Chinese Xinhua news articles. However, this method only considered the discourse relationship annotation inside the paragraph and did not annotate the discourse relationship between paragraphs and the macro discourse information of the whole text.

Compared with the well-studied micro discourse analysis, macro discourse analysis is in the trial stage both in English and Chinese. In English, Sporleder and Lascarides (2004) attempted to perform an experiment on macro discourse analysis on RST-DT. However, RST-DT focuses on the micro discourse level, which is only tagging the relations on the sentence level and ignoring the relations on the paragraph level. Thus, it leads to some tailoring and correction when Sporleder carries out macro discourse analysis on the paragraph level. In Chinese, Chu et al. (2017) carried out relevant research on the nuclearity of macro discourse and made corresponding attempts to construct a macro discourse corpus.

In the field of Chinese discourse annotation, the number of relevant corpora is still relatively small, and its scale is also small. In addition, the existing micro discourse corpora still lack macro discourse information. For the task of analysis of macro discourse relationships, it is necessary to construct a macro discourse corpus, especially in Chinese.

3 Annotation System

3.1 Discourse Structure, Nuclearity and Relationship

Macro discourse analysis includes the analysis on discourse structure, nuclearity and relationship, similar to micro discourse analysis. However, macro discourse relationships are different from those of micro discourse. First, since the element of a macro discourse unit is a paragraph, macro discourse mainly considers the structure of paragraphs or its upper levels (e.g., DU1-2 in Figure 1). As shown in Figure 1, the annotation goal is to construct a discourse structure tree, where the leaf nodes are the element of the discourse units, i.e., paragraphs. Second, different from micro discourse relations, there are no connectives between the macro discourse units with lengths that are longer than those of micro discourse units. Consequently, it is difficult to grasp the theme of macro discourse units and the relationship between them. To ensure the correctness of macro discourse structure annotation, we first determine the theme of discourse units.

Category	Nuclearity
Mononuclear	Nucleus Ahead, Nucleus Behind
Multi-Nuclear	Multi-Nucleus

Table 1: Classification of discourse nuclearity.

In the annotation of discourse nuclearity, we are consistent with RST, which is divided into hypotactic ("Mononuclear") and paratactic ("Multi-Nuclear"). Specifically, it is divided into three categories: Nucleus Ahead, Nucleus Behind and Multi-Nucleus. Nucleus Ahead and Nucleus Behind belong to the Mononuclear, and Multi-Nucleus belongs to the Multi-Nuclear as Table 1.

Different from the discriminant method of micro discourse nuclearity, we not only compare importance among candidate macro discourse units but also consider which unit is more important with the full-text writing intention. In example 1, discourse units P1 and P2 have the *Evaluation* relation, where P1 tells the main story, and P2 is its evaluation. If the global information is not considered, P1 is more important than P2, and it should be annotated as *Nucleus Ahead*. However, given the full text, the title of which is "The Property Insurance Co. of PICC provides Shanghai with the largest amount of export credit insurance to date", P1 and P2 jointly form the subject of the full text such that P1 and P2 are equally salient; therefore, we annotated it by the relation *Multi-Nucleus*.

Following previous corpora on discourse analysis, we also construct a macro discourse corpus on news articles. In the annotation of macro discourse relationships, the theme of the paragraph as the element of the discourse unit is vaguer; the logic between discourse units is more insignificant. Considering the overall intentions of a news article, it is rare to have a *Transition* relation on the macro-level, and most of them have a *Contrast* relation. Therefore, based on the micro discourse relation representation that has 4 categories and 17 relations defined by CDTB, we delete the relations, such as *Transition*, *Concession* and other relations in the *Transition* category, and finally form the macro discourse relationship representation with 3 categories and 15 relations, as shown in Table 2.

Category	Relations
Coordination	Joint, Sequence, Progression, Contrast, Supplement
Causality	Cause-Result, Result-Cause, Background, Behaviour-Purpose, Purpose-Behaviour
Elaboration	Elaboration, Summary, Evaluation, Statement-Illustration, Illustration-Statement

Table 2: Classification of macro discourse relation.

3.2 Additional Macro Discourse Information

Van Djik (1980) notes that theme, gist, keystone and main points all belong to the overall macro discourse structure. The topic sentence of a paragraph, the lead and the abstract of a full text are related to the theme and key points of the text and are the overall structure of semantics, which also belong to the macro discourse structure. Therefore, in addition to the traditional annotation of micro discourse corpus, we also annotated the additional macro discourse information, including the topic sentence of a paragraph, the lead and the abstract of a full text. In addition, we automatically annotate the pragmatic functions of discourse elements.

The discourse structure tree can play a crucial role in building a natural language generation system (Carlson et al., 2003). A good macro discourse structure tree can preserve the semantic integrity of the text better (Liu and Zou, 2017). In discourse-level automated summarization tasks, a more natural and complete chapter summary can be generated in conjunction with annotated macro discourse information.

4 Annotation Process

Our annotation team consists of a doctoral candidate, two senior master degree candidates and three junior master degree candidates. All annotators are engaged in research on Natural Language Processing or Computational Linguistics and have a certain theoretical foundation of linguistics. Such an annotation team has increased professionalism and improved the efficiency of labelling. To ensure annotation quality, the entire annotation process has four phases:

1) Initialization phase. We annotated 10 articles per cycle. In each cycle, the doctoral candidate and two senior master degree candidates were annotating the same article at the same time, that is, triple tagging. At this stage, we had annotated 97 documents.
2) Revision phase. Different from the previous stage, the doctoral candidate double-tagged with two senior master degree candidates in each cycle, respectively. Finally, we annotated 147 documents.

3) Trial tagging. We annotated 20 articles per cycle. The tagging method is in line with the second stage, and we finally cumulatively tagged 305 documents. At the same time, three junior master degree candidates participated in the annotation.
4) Formal annotation phase. We annotated 30 articles per cycle. During each cycle, three senior annotators (one doctoral candidate and two senior master degree candidates) were matched with three junior master degree candidates, respectively, and formed three groups. After each cycle, we exchanged the partner between different groups. Finally, 720 articles have been tagged.

4.1 Annotation criteria

To ensure consistency and reliability of annotations in the macro discourse relationship labelling, the following annotation criteria are used in our annotations:

1) The annotation of nuclearity and relationship is independent of each other, and we do not consider the correlation between them. In Example 1, when judging the nuclearity of the two discourse units P1 and P2, we pay more attention to their relevance to the topic, not considering their relation Evaluation. Therefore, P1 and P2 are equally salient.
2) We annotate only one type of discourse relations to two discourse units. For example, both the *Sequence* and *Cause-Result* relations have the chronological order. When two discourse units have the chronological order, and there is a clear causal relation between them, they are annotated as *Cause-Result*; otherwise, it will be annotated as *Sequence*.
3) We adopt the way of majority voting. An article may have different discourse structure trees from the different perspectives. When these structures are all reasonable, we use a majority vote to choose a more widely accepted understanding to ensure the objectivity of annotation.
4) We adopt incremental annotation. At the time of macro discourse relationship tagging, our annotation strategy is to determine the topic sentence of each paragraph first. After grasping the intentions of each paragraph, we first divide the whole article into several independent regions from top to bottom and build sub-trees from bottom to top in each area. Next, we consider the structure of each region and eventually form a complete discourse structure tree. According to the complete discourse structure tree, we also annotate the lead and the abstract of an article. In example 1, when we read this article, if we can first find that P4 and P5 have the relationship of *Purpose-Behaviour*, we could directly connect them without considering the specific structure of P1 to P3.

4.2 Annotation sample

To facilitate labelling and accelerating the annotation speed, we have developed a platform for macro discourse annotation. In this platform, an article to be annotated should go through the following steps: data preprocessing, annotating paragraph topic sentence, annotating discourse structure, nuclearity and relationships, annotating lead and abstract, and automatically annotating pragmatic functions. The final generated annotation results of example 1 are shown in Table 3. An annotated document includes three parts: DISCOURSE, RELATION and TEXT. DISCOURSE refers to the abstract information of a document, including date (Dateline), topic (DiscourseTopic), lead (LEAD) and abstract (ABSTRACT). RELATION provides all discourse relationships in this document and its attributes include nuclearity and relation. TEXT shows all elementary discourse units (EDUs). The labels used in MCTDB are as follows.

- ID: the ID of the relation between two discourse units.
- Centre: the nucleairity relation (1-*Nucleus Ahead* / 2-*Nucleus Behind* / 3-*Multi-Nucleus*).
- ChildList: the ID list of its children relations (nodes). In this example, this relation has two children relations whose IDs are 2 and 3.
- Function: the pragmatic functions are derived from Van Dijk (1990)'s Hypothetical structure of the news schema and we modified them to suit the characteristics of Chinese macro discourse. Details of the labels are shown in Table 10.
- Layer: the layer in a macro discourse structure tree.
- ParagraphPosition: the positions of two discourse units which are split by "|". In this example, "1...3" indicates that the beginning paragraph and ending paragraph of the first discourse unit

are 1 and 3, respectively, while “4...5" indicates those of the second paragraph are 4 and 5, respectively.

```
<DOC>
<DISCOURSE Dateline="新华社上海三月六日电" DiscourseTopic="中保财险公司为上海提供迄今最大一笔出口信用保险">
<LEAD>中保财产保险有限公司……的最大一笔出口信用保险。</LEAD>
<ABSTRACT>中保财产保险有限公司……的目的。</ABSTRACT>
</DISCOURSE>
<RELATION>
<R ID="1" Center="1" ChildList="3|2" Function="NewsReport" Layer="1" ParagraphPosition="1...3|4...5" ParentId="-1" RelationType="Supplement" StructureType=" Hierarchical segmentation" />……
 <TEXT>
<P Function="Lead" ID="1" ParagraphTopic="中保财产保险有限公司……的出口信用保险。">中保财产保险有限公司…. . . 的出口信用保险。</P>……
<P Function="Behaviour" ID="5" ParagraphTopic="中国中央财政于一九八八年拨一亿美元专款作为风险准备金，用于中保集团公司开办出口信用保险业务。">为此，中国中央财政……四十多个。（完）</P>
</TEXT>
</DOC>
```

Table 3: Annotation format of chtb0155.

- ParentId: the ID of the relation’s parent.
- RelationType: the type of this relation, defined as RST, and details of the labels are shown in Table 2.
- StructureType: the annotation is divided into 2 categories: *Hierarchical segmentation* and *Parallel segmentation*. When the relationship is Mononuclear and has only two children, it is marked as *Hierarchical segmentation*. Otherwise, it will be marked as *Parallel segmentation* (usually the relation is *Joint*).
- ParagraphTopic: the topic sentence of the paragraph.

4.3 Quality Assurance

We guarantee the annotation quality of the corpus from two aspects. The first one is the tree verification, which is to ensure the correctness of corpus annotation. The second one is consistent assessment, which is designed to ensure the objectivity of corpus annotation.

In tree validation, we take two steps to verify the annotated corpus. First, the annotated corpus is imported to the macro discourse annotation platform to verify its correctness. Second, an automated pragmatic annotation tool is used to annotate the pragmatic functions of each discourse unit. Through pragmatic annotation rules, it can be used to calibrate the annotation of discourse relations and structures. When a problem occurs in the validation process, we manually verify the error type and then correct it. Compared with the traditional manual calibration, we use a double automatic verification method to improve the speed and ensure quality.

The evaluation of consistency can be used to measure the objectivity of corpus annotation. For evaluating the consistency of the discourse structure tree, tree consistency can more objectively reflect the quality of annotation. CDTB and PDTB only evaluated the consistency of certain indicators, such as discourse relation, and did not evaluate the tree consistency. RST-DT proposed a method of evaluating the complete consistency of a discourse tree, but it also had several shortcomings. The evaluation objects of RST-DT are the discourse structure, nuclearity and relationship annotated on the children nodes, while the evaluation object of the standard syntax tree (Black et al., 1991) is the annotations on the parent node. In this way, RST-DT will be at least double the standard syntax tree evaluation of the sample, and

because of the mutual constraint relationship between children nodes, the consistency of corpus annotation of RST-DT is objectively higher.

In the annotation of a macro discourse relationship, the leaf nodes are natural paragraphs and do not need to be manually divided. Therefore, unlike the assessment method of the RST-DT agreement, we do not use the leaf nodes as an example of a consistent assessment. In addition, we adopted the standard syntax tree method using the annotation on the parent node as the evaluation object. Although this may lead to a reduction in evaluation examples and a decline in the indicators of consistency, we believe that this evaluation method is more objective.

Indicators	Structure	Nuclearity	Relation
Agreement	86.24%	83.88%	80.46%
Kappa	0.68	0.66	0.61

Table 4: Consistency indicators of MCDTB.

To eliminate coincidental annotations, we also used the Kappa value (Siegel and Castellan, 1988) as an assessment of the consistency of annotations while using agreement rates. The average value of the agreement assessments of the 200 articles sampled served as the ultimate evaluation indicators of the corpus. As shown in Table 4, the corpus has an agreement rate of greater than 80% and a Kappa value greater than 0.6 in discourse structure, nuclearity and relationships. Krippendorff (1980) noted that the Kappa value of the annotation data is above 0.6, indicating that it has good annotation quality.

Confusing Relationships	Proportion
Elaboration and Supplement	21.90%
Elaboration and Joint	9.52%
Sequence and Joint	8.57%
Supplement and Joint	7.62%
Elaboration and Evaluation	5.71%
Elaboration and Sequence	5.71%

Table 5: Confusing Relationships in double-tagged.

To find the most confusion annotations among the annotators, we sampled 100 articles and obtained the double-tagged information (720 articles are double-tagged). The top six confusion relationships are shown in Table 5. Apart from the difference in the structure annotation, the primary confusing relationships are *Elaboration* and *Supplement*, accounting for 21.90%. Both of these relationships belong to the *Elaboration* category; therefore, it is easier to confuse them. The second confusing relationship is *Elaboration* and *Joint*, accounting for 9.52%. The confusion results from significant differences in the understanding of the article.

Confusing Nucleus	Proportion
Nucleus Ahead and Multi-Nucleus	89.19%
Nucleus Ahead and Nucleus Behind	8.11%
Nucleus Behind and Multi-Nucleus	2.70%

Table 6: Confusing Nucleus in double-tagged.

As shown in Table 6, the main confusion nucleus are primarily Nucleus Ahead and Multi-Nucleus, accounting 89.19%; this finding may be due to the confusion between *Elaboration* and *Joint* in the relationship. This result is also consistent with previous experimental results (Jiang et al., 2018), which show that humans and models are confused in the same place.

5 Statistics on MCDTB

Our macro discourse corpus MCDTB[1] annotated 720 news articles from CTB 8.0. As shown in Table 7, the entire annotated corpus has 3,981 paragraphs, and the average number of paragraph is 5.53 per document. Each article contains at most 22 paragraphs and at least 2 paragraphs.

#documents	720	#paragraphs	3,981
# paragraphs of the longest document	22	Average length (paragraphs /document)	5.53
# paragraphs of the shortest document	2	Average length (sentences/paragraph)	2.09

Table 7: MCDTB corpus details.

The distribution of document numbers on paragraph numbers in MCDTB is shown in Figure 2. 54.72% of the documents have 4 to 6 paragraphs, while this number is 92.08% when a document has 2 to 9 paragraphs, which indicates that news articles primarily focus on the middle passage.

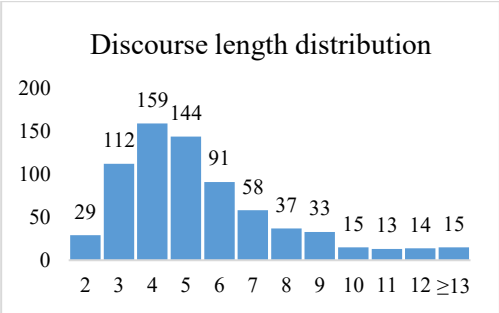


Figure 2: Discourse length distribution in MCDTB.

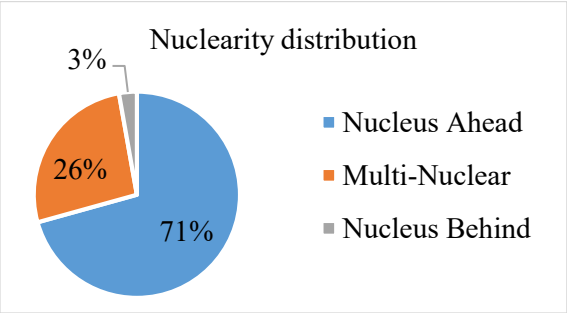


Figure 3: Nuclearity distribution in MCDTB.

The nuclearity distribution in MCDTB is shown in Figure 3. Among these numbers, the number of the *Nucleus Ahead* is 2,026, accounting for 70.69%. There are 760 *Multi-Nuclear* relations, accounting for 26.52%, while there are only 80 *Nucleus Behind*, accounting for only 2.79%. This finding shows that news articles tend to explain important events first, which is in line with the conclusion of linguistic statistics by Li and Liao (2001).

Relation	**Number**	**Proportion**	**Relation**	**Number**	**Proportion**
Elaboration	990	34.54%	Joint	634	22.12%
Supplement	493	17.20%	Background	237	8.27%
Evaluation	127	4.43%	Result-Cause	103	3.59%
Sequence	99	3.45%	Cause-Result	49	1.71%
Statement-Illustration	39	1.36%	Summary	23	0.80%
Illustration-Statement	20	0.70%	Contrast	17	0.59%
Behaviour-Purpose	14	0.49%	Purpose-Behaviour	11	0.38%
Progression	10	0.35%			

Table 8: Statistics on discourse relations.

As shown in Table 8, MCDTB contains 2,866 discourse relations, of which the top six relations are *Elaboration*, *Joint*, *Supplement*, *Background*, *Evaluation* and *Result-Cause*. These six types of relations accounted for 90.16% of the total number of relations.

[1] The Macro Chinese Discourse TreeBank is available at https://figshare.com/s/250474dba44e4161b040.

Furthermore, we analysed the correlation between the nuclearity and the relationship. The nuclearity distributions on different discourse relationships are shown in Table 9. This finding shows that most of the *Elaboration* and *Causality* relations are *Nucleus Ahead*, while those of the *Coordination* relations are more inclined to be equally salient in each discourse unit. The three relationships are about the same in *Nucleus Behind*.

Category	Nucleus Ahead	Nucleus Behind	Multi-Nucleus
Elaboration	1208	36	9
Causality	365	29	20
Coordination	453	15	731

Table 9: Nuclearity distribution of discourse relationships.

In particular, we analysed the distribution of pragmatic functions. MCDTB has a total of 6,847 pragmatic functions, including 3,981 annotations on the paragraph level, and 2,866 on the larger discourse unit level, as shown in Table 10. Apart from annotating the necessary pragmatic functions, such as *Situation* and *Story*, the annotations which are helpful to the automatic summarization and information extraction tasks, such as *Lead*, *Summary*, *Background*, *Comment*, *Cause* and *Result*, make up the majority.

Pragmatic function (in paragraph)	Number	Proportion	Pragmatic function (in discourse units)	Number	Proportion
Situation	1902	27.78%	Story	1437	20.99%
Supplement	439	6.41%	News Report	720	10.52%
Summary-Lead	354	5.17%	Summary	300	4.38%
Lead	284	4.15%	Sub-Summary	95	1.39%
Background	196	2.86%	Result	73	1.07%
Sub-Summary	193	2.82%	Cause	69	1.01%
Story-Situation	183	2.67%	Supplement	54	0.79%
Comment	117	1.71%	Background	41	0.60%
Cause	83	1.21%	Statement	22	0.32%
Result	79	1.15%	Behaviour	17	0.25%
Others	151	2.21%	Others	38	0.55%

Table 10: MCDTB pragmatic function statistics.

6 Preliminary Experiment

To demonstrate the computability of MCDTB, we conducted a preliminary experiment on the identification of macro discourse structures. Discourse structure identification is the first and most critical step in the related tasks of discourse analysis. We use a Conditional Random Field (CRF) model to experiment with the parameter C of 4, the feature window of 3, and the rest of the parameters as a default value. There were 8,863 samples in total, including 3,261 positive samples and 5,602 negative samples.

Because of the language differences between English and Chinese, as well as the microscopic and macroscopic differences in features, such as the absence of syntax trees and the absence of tense features, we select only several of the structural features of the previous study (Feng and Hirst, 2012) as our Feature Set 1 as follows:

- The position of the beginning and end of a discourse unit;
- The number of sentences and the number of paragraphs contained in a discourse unit;
- The comparison of the number of sentences in the discourse unit to the previous unit;
- The comparison of the number of paragraphs in the discourse unit to the previous unit.

Considering the process of constructing discourse structure trees, we may need some procedural characteristics. We regard whether the discourse unit is a leaf node or whether the discourse unit is merged in the previous round as organizational characteristics. In terms of semantic features, we use semantic similarity, the connectives, and their part-of-speech in the first sentence of the discourse unit. On the calculation of semantic similarity, we used the word2vec model to train on CTB8.0 and calculate the semantic similarity between the two discourse units according to the semantic similarity algorithm proposed by Xu (2009). In particular, we discretized the final semantic similarity into 10 levels. Finally, we use the semantic similarity, the conjunctions of the first sentence of the discourse unit with their part-of-speech, whether they are the leaf nodes and whether they were merged in the previous round as the characteristics of Feature Set 2 as follows:

- The semantic similarity between the discourse unit and the previous unit;
- The conjunction of the first sentence and its part of speech in the discourse unit;
- Whether the discourse unit is a leaf node;
- Whether the discourse unit was merged in the previous round.

Feature selection	Feature Set 1	Feature Set 2	Feature combination
Acc.	76.09%	77.01%	**77.56%**

Table 11: Comparison of the accuracy of the various models.

In addition, we use the most probabilistic way to fuse two feature sets, that is, we use two feature sets to model and finally select the predictive labels with the larger probability value as the final annotation result. In the five-fold cross-validation experiment of the MCDTB corpus, the performance of each model is shown in Table 11.

The experimental results show that the performance of the model using the structural features and semantic features is similar, and the best performance is obtained by using the joint model based on the maximum probability of the feature combination, which improves the accuracy by 1.47% and 0.55% over Feature Set 1 and Feature Set 2, respectively.

7 Conclusions

In this paper, we describe relevant experiments concerning the construction of a Macro Chinese Discourse Treebank (MCDTB). In view of the differences between the annotations of micro discourse relationships and macro discourse relationships, we annotate the corpus following RST and provide the higher level macro discourse information, such as topic sentences of paragraphs, lead and abstract of discourse, to make the macro discourse annotation more objective and accurate. To speed the annotation process and ensure the annotation quality, we formulated detailed annotation criteria and quality assurance strategies and developed a platform for macro discourse annotation. Finally, we annotated 720 Chinese news articles, and we achieved a better consistency of labelling with an improved Kappa value of greater than 0.6 based on the improved consistency assessment standard. Preliminary experiments on this corpus verify the computability of MCDTB. In future work, we will conduct research on identifying macro discourse structure, recognizing nuclearity and classifying relationship and eventually form a complete end-to-end macro discourse analyser.

Acknowledgements

The authors would like to thank three anonymous reviewers for their comments on this paper. This research was supported by the National Natural Science Foundation of China under Grant Nos. 61751206, 61772354 and 61773276, and was also supported by the Strategic Pioneer Research Projects of Defense Science and Technology under Grant No. 17-ZLXDXX-02-06-02-04.

Reference

Cohan Arman, and Nazli Goharian. 2015. Scientific article summarization using citation-context and article's discourse structure. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015). Pages 390-400, Lisbon, Portugal, September. Association for Computational Linguistics.

Ezra Black, Steven Abney, Dan Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini and T. Strzalkowski. 1991. Procedure for quantitatively comparing the syntactic coverage of English grammars. In Proceedings of the workshop on Speech and Natural Language. Pages 306-311. Pacific Grove, California, February .Association for Computational Linguistics.

Xiaomin Chu, Qiaoming Zhu and Guodong Zhou. 2017. Discourse primary-secondary relationships in natural language processing. Chinese Journal of Computers, 40(04): 842-860.

Teun A. Van Dijk. 1976. Narrative macrostructures. PTL: A journal for descriptive poetics and theory of literature, 1: 547-568.

Teun A. Van Dijk. 1980. Macrostructure: An interdisciplinary study of global structures in discourse, interaction, and cognition. Hillsdale, New Jersey: Lawrence Erlbaum Associates, Inc., Publishers.

Teun A. Van Dijk. 1990. News as discourse. Hillsdale, New Jersey: Lawrence Erlbaum Associates, Inc., Publishers.

Vanessa Wei Feng and Graeme Hirst. 2012. Text-level discourse parsing with rich linguistic features. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1(ACL 2012). Pages 60-68. Jeju, Korea, July. Association for Computational Linguistics.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2014). Pages 511-521, Baltimore, USA, June. Association for Computational Linguistics.

Fuyi Fu. Study of complex sentences in Chinese. 2001. Beijing: Commercial Press.

Xiaoyan Hao, Wei Liu, Ru Li and Kaiying Liu. 2007. Description systems of the Chinese framenet database and software tools. Journal of Chinese Information Processing, 21(5): 96-100.

Jerry R. Hobbs 1985. On the coherence and structure of discourse. Center for the Study of Language and Information. Pages 1-36.

Klaus Krippendorff. 1980. Content analysis: An introduction to its methodology. Seikeigeka Orthopedic Surgery, 79(385): 204.

Feng Jiang, Xiaomin Chu, Sheng Xu, Peifeng Li and Qiaoming Zhu. 2018. A Macro Discourse Primary and Secondary Relation Recognition Method. Journal of Chinese Information Processing, 32 (01): 43-50.

Jin Li and Kaihong Liao. Comparison of theme patterns and paragraph structures in Chinese and English. 2001. Jinan Journal (Philosophy & Social Science Edition), 23(05): 89-93.

Yancui Li, Wenhe Feng, Jing Sun, Kong Fang and Guodong Zhou. 2014. Building Chinese discourse corpus with connective-driven dependency tree structure. In Conference on Empirical Methods in Natural Language Processing (EMNLP 2014). Pages 2105-2114. Doha, Qatar, October. Association for Computational Linguistics.

Yancui Li. 2015. Research on the structure of Chinese text structure and the construction of resources. Soochow University.

Yancui Li, Jing Sun and Guodong Zhou. 2015. Automatic recognition and classification on Chinese discourse connective. Acta Scientiarum Naturalium Universitatis Pekinensis, (02): 307-314.

Jiayi Liu and Yimin Zou. 2017. A review of automatic text summarization research in recent 70 years. Information Science. 35 (07), pages 154-161.

Guoying Lv, Na Su, Ru Li, Zhiqiang Wang and Qinghua Chai. 2015. Frame-based discourse structure modeling and relation recognition for Chinese sentence. Journal of Chinese Information Processing, 29 (06): 98-109.

Carlson Lynn, Daniel Marcu and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In Current and new directions in discourse and dialogue. Pages 85-112. Springer, Dordrecht.

Maria Liakata, Simon Dobnik, Shyamasree Saha, Colin Batchelor and Dietrich Rebholz-Schuhmann. 2013. A discourse-driven content model for summarising scientific articles evaluated in a complex question answering task.

In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013). Pages 747-757, Seattle, USA, October. Association for Computational Linguistics.

William C. Mann and Sandra A. Thompson. 1987. Rhetorical structure theory: a theory of text organization. University of Southern California, Information Sciences Institute.

William C. Mann, Christian M. I. M. Matthiessen and Sandra A. Thompson. 1992. Rhetorical structure theory and text analysis. Discourse description: Diverse linguistic analysis of a fund-raising text. Pages 39-78.

Daniel Marcu, Estibaliz Amorrortu and Magdalena Romera. 1999. Experiments in constructing a corpus of discourse trees. In Proceedings of the ACL'99 Workshop on Standards and Tools for Discourse Tagging. Pages 48-57.

James Robert Martin and David Rose. 2003. Working with discourse: Meaning beyond the clause. London: Continuum.

Rashmi Prasad, Nikhil Dinesh, Alan Lee and Eleni Miltsakaki. The Penn Discourse Treebank 2.0. 2008. In International Conference on Language Resources and Evaluation, LREC 2008. Marrakech, Morocco, May. Pages 2961-2968.

Sidney Siegel and N. John Castellan. 1988. Nonparametric statistics for the behavioral sciences. 2. New York: McGraw-Hill Book Company.

Caroline Sporleder and Alex Lascarides. 2004. Combining hierarchical clustering and machine learning to predict high-level discourse structure. In Proceedings of the 20th international conference on Computational Linguistics (Coling 2004). Pages 43. Association for Computing Machinery.

Yizhong Wang, Sujian Li and Houfeng Wang. 2017. A two-stage parsing method for text-level discourse analysis. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (ACL 2017), pages 184-188, Vancouver, Canada, July. Association for Computational Linguistics.

Weizhang Wu and Xiaolin Tian. 2000. Chinese sentence group. Beijing: Commercial Press.

Shuai Xu. 2009. Research and implementation of paraphrase recognition for question answering. Harbin Institute of Technology.

Nianwen Xue. 2005. Annotating discourse connectives in the Chinese Treebank. In Proceedings of the Workshop on Frontiers in Corpus Annotations II: Pie in the Sky. Association for Computational Linguistics. Pages 84-91.

Nianwen Xue, Fei Xia, Fudong Chiou and Marta Palmer. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. Natural language engineering, 11(2): 207-238.

Yuping Zhou and Nianwen Xue. 2015. The Chinese Discourse TreeBank: a Chinese corpus annotated with discourse relations. Language Resources & Evaluation, 49(2): 397-431.

Bowei Zou, Guodong Zhou and Qiaoming Zhu. 2014. Negation focus identification with contextual discourse information. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014). Pages 522-530, Baltimore, USA, June. Association for Computational Linguistics.

Corpus-based Content Construction

Balaji Vasan Srinivasan, Pranav Ravindra Maneriker, Kundan Krishna, Natwar Modani
Adobe Research Big data Experience Lab
Bangalore, India
[balsrini,pmanerik,kkrishna,nmodani]@adobe.com

Abstract

Enterprise content writers are engaged in writing textual content for various purposes. Often, the text being written may already be present in the enterprise corpus in the form of past articles and can be re-purposed for the current needs. In the absence of suitable tools, authors manually curate/create such content (sometimes from scratch) which reduces their productivity. To address this, we propose an automatic approach to generate an initial version of the author's intended text based on an input content snippet. Starting with a set of extracted textual fragments related to the snippet based on the query words in it, the proposed approach builds the desired text from these fragment by simultaneously optimizing the information coverage, relevance, diversity and coherence in the generated content. Evaluations on standard datasets shows improved performance against existing baselines on several metrics.

1 Introduction

Due to the increase in the number of channels and the need for more customized content for different audience segments, content authors need to produce content at a very rapid pace. On the other hand, a lot of such content has already been produced and is available in enterprise content repositories. Ideally, the author looks for such content and re-purposes them for her current need. Due to the tedium of this task, the author may resort to creating the content from scratch. Automating this task can greatly aid the author in improving their overall productivity.

Given an enterprise author's requirement in the form of a snippet describing the content she is trying to build, standard querying mechanisms can be utilized to fetch the right set of articles that can cater to her needs. However, constructing an initial piece of 'article' that can cater to her needs would still be a non-trivial task. Existing work in this direction focuses on generating content for standard platforms like Wikipedia (Banerjee and Mitra, 2016), where the target content is divided into different 'categories/section' and text is generated for each section. While this could be useful, the absence of a predefined structure in an enterprise setting makes the extension of such an approach challenging. This calls for an approach that can automatically build an author's content **covering various aspects** of the target article.

The content style in an enterprise corpus is fairly consistent. However, there could be multiple representations of the same information across the repository. This necessitates the content generation to **reduce the redundancy** in the constructed content and adding topical diversity to the article. Finally, the content generated should be coherent with a **cohesive flow** of information.

To address all these requirements, we propose an approach that automatically generates a *coherent* and *non-redundant* article based on an author's input snippet simultaneously optimizing the *relevance* of the generated text to the snippet, *coverage* of different aspects of the content and *diversity* in the information content. The acceptance criteria for such systems can be a reasonable draft instead of publishable content, given that the objective of such a system is not to replace the authors, but to provide them with a draft that can be polished quickly. Table 1 shows a sample article generated by the proposed algorithm based on an input snippet from DUC2007 data (Dang, 2007).

Proceedings of the 27th International Conference on Computational Linguistics, pages 3505–3515
Santa Fe, New Mexico, USA, August 20-26, 2018.

Table 1: A sample snippet and the corresponding article generated by the proposed algorithm based on the DUC2007 data (Dang, 2007).

Input Snippet: Microsoft's antitrust problems and its alleged illegal behavior and antitrust proceedings against the company.
Generated article by CCC: a justice department spokesman confirmed thursday that we have an ongoing investigation and its continuing. microsoft based in redmond of washington has been the focus of antitrust investigations in the past years. microsoft corp is making a last minute offer to settle the charges. nbc news reported wednesday that the justice department and at least 19 states plan to file historic antitrust cases thursday against microsoft. if they proceed the suits could be among the biggest corporate challenges in history according to analysts. on monday in washington dc the federal government and 20 states kicked off the trial that accused microsoft of illegal activities including bullying competitors. back in seattle washington state where microsoft is based the company fired off five upbeat productoriented press releases and other announcements. its web site on monday carried two articles in favor of microsoft. it helps the atmospherics of the governments case but its not clear beyond that what the effect will be. meanwhile the antitrust trial is already well under way and judge thomas penfield jackson is expected to rule early next year. but sherlund noted the ruling does rein microsoft in and it clearly comes at an awkward time for the company given the antitrust case. antitrust experts said that most countries still investigating microsofts practices were taking a waitandsee stance toward the company pending the outcome of the major us antitrust case

Our algorithm takes author intent in the form of a short snippet (say a sentence) and finds relevant content from a repository to create a pool of content. It then identifies the most representative part of the content (factoring in relevance level) and generates multiple candidate sentences by compressing information from multiple related sentences to enrich the information content (and reduce redundancy). The compression algorithm leverages usage patterns in the repository to form sentences with good readability. Subsequent candidate sentence generations account for the already generated candidates to ensure diversity. We finally assemble the content in an appropriate order by solving an integer linear program that simultaneously optimizes the coherence with various linguistic quality measures to determine the best ordering. Thus, our system strives to generate a draft of content that is relevant to the author intent, diverse, information rich, cohesive and readable.

We present the results of performance evaluation of our system with DUC 2007 (Dang, 2007) and Australian Legal (Galgani et al., 2012) datasets. Our experiments show that our system produces drafts which are as good as state-of-art baselines for content generation in Rouge scores, diversity, readability and cohesiveness, and beat each of the baselines in at least one of these metrics.

2 Related Work

There is a significant body of work in the domain of **text summarization** (Nenkova and McKeown, 2011). The problem of corpus based content construction could be perceived as a summarization task once we have identified the required candidate content fragments from the corpus. In particular, the problem could be cast as a multi-document summarization or query-focused summarization. However, in our problem, there is no requirement to maintain the distribution of concepts/information between input document and output content, thus making it different from standard summarization problems.

There have been some efforts towards extending multi-document summarization for expanding a piece of content. Banerjee et al. (2015) use a sentence ranking and clustering formulation along with ILP based constraint optimization to create expanded documents. Bing et al. (2015) explore a more fine-grained setting, constructing sentences using noun/verb phrase level breakdown of input text. However, neither of these two approaches handle the document coherence problem at a granular level and rely on using the original sentence order as their notion of coherence.

Query-focused multi-document summarization approaches such as those of Li and Li (2014) and Wang et al. (2013) do not have an explicit model of coherence, in spite of using coherence as one of the evaluation criteria. In the proposed algorithm, we use an Integer Linear Program formulation to achieve a coherent generation despite not explicitly selecting contiguous content fragments.

Srinivasan et al. (2017) extend standard extractive summarization techniques to build article by combining different paragraphs in a corpus. However, they do not account for coherence or information redundancies in the generated content. Often, retaining an entire paragraph might not yield the best

information coverage. We explicitly account for these aspects in our proposed algorithm.

There are some recent efforts towards accounting for coherence in summarization by extending the transition probability based approach by Lapata and Barzilay (2005) and including coherence in their optimization frameworks. Parveen et al. (2016) use 'patterns' from a standard corpus to define the transition probability, while Nayeem and Chali (2017) and Wang et al. (2016) use information overlap based metric to define the transitions. Our framework also uses an approach similar to the latter to account for coherence in our text generation.

Knowledge enhancement: There have also been some efforts towards content elaboration in the context of knowledge enhancement. Schlaefer et al. (2011) aim to enhance question-answering by expanding a 'seed document' using web resources to construct the answer based on 'paragraph nuggets'. However, both lexical and semantic redundancies are undesirable for a content author. In our approach, we minimize the lexical and semantic redundancies based on a graph based sentence compression which jointly optimizes for relevance and diversity of the expanded content.

The work by Bairi et al. (2016) uses expansion of short documents to enhance the document representation for improved document classification. Their framework aims to improve performance for a task without using task-specific heuristics. However, since the objective is to maximize the classification performance, the algorithm proposed by Bairi et al. (2016) does not care about linguistic aspects of the elaborated content.

Taneva and Weikum (2013) identify relevant text snippets ('gems') by using an integer linear program to maximize relevance of selected words, and prefer the selection of contiguous content pieces since this maximizes the coherence of the content resulting in a lot of content being chosen from the same source. However, the information for a specific article can come from multiple sources and hence this will not achieve information coverage. We use this as one of the baselines for our comparisons and show improvements in the coverage of our proposed approach.

Wikipedia Generation: Mihalcea and Csomai (2007) identify key concepts in a document and link them to corresponding Wikipedia pages. While this helps to identify relevant Wikipedia areas for the document, it will not help in expanding the seed content from these sources. Sauper and Barzilay (2009) came up with the first complete version of such a system. Their work leverages the structure of Wikipedia articles of a category to produce a new article in the same category. Banerjee and Mitra (2016) extend these ideas to construct a Wikipedia article by learning the article structure from the structure of related articles, rather than relying on a single category. While this is closer to what we would like to achieve, the lack of structure in an enterprise article poses a unique challenge in our case. Further, if the corpus is huge, this can result in a lesser information coverage. We address this in our proposed algorithm via an iterative reward framework to maximize the information coverage of the generated article.

3 CCC: Corpus based Content Construction

We propose a **C**orpus-based **C**ontent **C**onstructor (CCC) that simultaneously optimizes for relevance to the input snippet, covers different information about the target content, avoids redundant sentences and improves the coherence of the final content. The algorithm involves 3 stages: extracting the intent of author from the snippet, identifying and **generating candidate content** towards the author's need and **assembling the generated content** in a coherent fashion.

We begin with extracting the top-k keywords in the snippet using the inverse document frequency (IDF) of the words in the corpus, thus capturing the most significant keywords in the snippet with respect to the corpus. A query q is then formulated by concatenating these 'k' keywords (Aula, 2003), where the choice of k determines the relevance and the amount of content that is available and fetched from the repository. A lower value of k results in the query under-representing the content and fetching articles that may not be very relevant. On the other hand, a higher value of k will result in a very specific query that might not fetch many results from the repository.

The retrieved fragments are broken into sentences. To select the appropriate content for construction, we create a graph and represent the sentences as nodes in the graph. The information content of the sentences is assigned as a "reward" to the corresponding nodes and their information overlap as the edge

weight. As explained later, the most "rewarding" part of the graph is identified and the sentences in that sub-graph are compressed to prepare candidates to be included in the final content. The sentence compression reduces the syntactic and lexical redundancies arising due to multiple manifestations of the same information in different parts of the corpus. Once the candidates are generated, the rewards of the nodes in the graph are adjusted to account for the information in the generated candidates. This allows for increased information coverage in the generated sentences by discounting the information covered by the previously extracted candidates.

The candidates are generated iteratively till the overall information relevant to the input snippet is exhausted in the graph. To select the final content, an Integer Linear Program (ILP) is formulated that maximizes the relevance to the snippet along with the information cohesion in the generated content. The optimization problem is also constrained to keep the constructed content to a certain budget (length) and avoid simultaneous selection of semantically very-similar sentences. The ILP outputs a set of compressed sentences along with their sequence to yield a cohesive content.

The candidate generation and coherent candidate organization are the key contributions of our algorithm and we describe the details below.

3.1 Candidate Generation via Sentence Compression:

The candidate set of fragments is broken into sentences and a graph $G(v, e)$ is constructed with every sentence as a node $v_i \in V$ in the graph. An initial "reward" r_i^0 for v_i is assigned based on its similarity to the query q. The edges $e \in E$ are constructed between every pair of fragment that contain a significant information overlap and the similarity between the sentences is assigned as its edge weight w_{ij}.

We iteratively select a sub-graph in G whose nodes can be used as candidates for compression. Each sub-graph consists of a node v_i along with its $1-$hop neighbors. The gain G_{v_i} of processing a node v_i (along with its $1-$hop neighbors) for sentence compression at round l is given by,

$$G^l_{v_i} = r_i^{l-1} + \sum_{v_j \in N_i} r_j^{l-1} \times w_{ij} \tag{1}$$

where N_i refers to the neighbors of node v_i. At step l, we select the v_i^* that has the maximum gain G_{v_i}. The sentence corresponding to the node selected v_i along with the sentences in its $1-$hop neighborhood is put into a set s_{v_i}, which is used for multi-sentence compression (Filippova, 2010) to remove the redundancy across the multiple sentences and generate candidates.

Multi-sentence compression constructs a (different) graph ($\hat{G}_c$) out of the candidates such that each word is a node, and each sentence represents a directed path in the graph. Specialized nodes serve as start-of-sentence and end-of-sentence nodes. A single node is used to represent all occurrences of a word with the same POS tag. The edges between nodes have weights representing the number of times the ordered combination of those node-words occurs across all sentences in s_{v_i}. The shortest paths (normalized by path length) are identified and the top-K generated sentences are used for further processing. For our purposes, we restrict the minimum number of words per generated sentence to 10 words and pick 1 sentence in every iteration.

A reward $r_{\hat{k}}$ is assigned to every compressed sentence based on its similarity to the query q. In order to factor the information captured by the compressed sentences for subsequent selections, the rewards of vertices v_i in the original graph ($G(v, e)$) whose information overlap ($w_{i\hat{k}}$, computed in the same way as the edge weights) with the compressed sentences is significant, is updated as as $r_i^{l+1} = r_i^l \times (1 - w_{i\hat{k}})$. This reduces the rewards for sentences which are covered by current set of compressed sentences thus reducing the chance of their inclusion in subsequent compression. This ensures that the information covered by the subsequent compression is different from already generated sentences while still relevant to the input query.

The candidate generation is stopped when there are no nodes left with significant reward (greater than a threshold) resulting in the same sub-graph being selected for successive compression.

3.2 Coherent Candidate Organization via ILP

The sentence compression yields candidates that cover the information space relevant to the input snippet. However, the compressed sentences might be grammatically incorrect and not ordered appropriately to be coherent. Therefore we select the right set of compressed sentences along with their order via an Interger Linear Program (ILP) formulation by extending the framework used by Deshpande et al. (2007) and Banerjee and Mitra (2016). The ILP objective is given by,

$$J = \sum_{i=1}^{K} w_i x_i + \lambda \sum_{i,j|coh_{ij}>\sigma} coh_{i,j} y_{i,j} \tag{2}$$

$$\text{such that: } \sum_{i=1}^{K} c_i x_i \leq B \tag{3}$$

$$y_{i,j} = 0 \text{ if } coh_{ij} < \sigma \text{ or } i = j \tag{4}$$

$$x_i + x_j \leq 1, \forall i, j | sim_{i,j} > 0.7 \tag{5}$$

$$\sum_i y_{s,i} = 1, \sum_i y_{i,e} = 1 \tag{6}$$

$$\sum_i y_{i,j} = \sum_i y_{j,i}, \sum_i y_{i,j} + \sum_i y_{j,i} = 2x_j \tag{7}$$

The binary variable x_i indicates the selection/non-selection of a compressed sentence i and the binary variable $y_{i,j}$ indicates a transition from a sentence x_i and x_j. Traversing the path of the sentences via $y_{i,j}$ would yield the final generated sentence. The w_i for each compressed sentence indicates a combination of the sentence's relevance to the snippet along with its overall linguistic quality. This ensures that the compressed sentence selected is not noisy and is relevant to what the author is looking to build. The second term in Eq. 2 maximizes the coherence of the selected flow of sentences. This follows from the work by Lapata and Barzilay (2005) which defines a model of coherence based on the topic overlap between successive sentences.

The constraint in Eq. 3 accounts for the length requirement of the author and limits the generated content to a target budget B. Eq. 4 prohibits a flow of content between lesser coherent sentences and avoids cycles in the arcs. Eq. 5 limits simultaneous selection two sentences with higher information overlap to produce less redundant content.

Similar to Banerjee and Mitra (2016), we constrain the optimization problem to also account for the overall quality of the generated content. The formulations to constrain the arcs is extended from WikiWrite (Banerjee and Mitra, 2016) by introducing dummy start s and end e nodes to the graph of sentences. An arc between two nodes exists if both the sentences are selected for the final content and are contiguous. The dummy nodes s and e are used to connect to the start and end sentences. Eq. 6 limits the number of starting and ending sentences to 1. Eq. 7 limit the number of incoming and outgoing arcs from a selected node to 1 each forcing a path via the arcs that indicate the flow in the selected content. Algorithm 1 summarizes the complete algorithm.

4 Evaluating different stages in CCC

In CCC, sentence compression (Filippova, 2010) is used to avoid redundant information in the constructed content by compressing information from multiple sentences into a single sentence. Further, the graph based candidate selection for compression enables covering diverse aspects around the target content. Finally, the ILP formulation facilitates coherence in the generated content.

To test if the different stages achieve the targeted goals, we test different parts of our algorithm on the DUC2007 Multi-document Summarization data (Dang, 2007). This data consists of a topic along with a set of documents related to the topic and reference summaries for the set of documents. For the sake of this experiment, we bypass the query construction stage. We use the topic associated with the document

Algorithm 1: Corpus-based Content Construction

Input: q: Snippet input by author, $\mathcal{F}$: Set of all fragments
Output: C: Generated Expanded Content
K $\leftarrow$ Top-k keywords from q based on IDF from $\mathcal{F}$;
F $\leftarrow$ query $\mathcal{F}$ with K;
$\mathcal{S} \leftarrow$ split F into sentences;
begin
  $\mathcal{G} \leftarrow$ graph($\mathcal{V}$, $\mathcal{E}$) where,
  $\mathcal{V} \leftarrow \mathcal{S}$, with reward $r_{v_i} \leftarrow$ info content of v_i
  $\mathcal{E} \leftarrow \{e_{ij} \mid$ weight $w_{ij} \leftarrow$ info overlap$(v_i, v_j)\}$
end
$\mathcal{S} \leftarrow \phi$;
repeat
  forall $v_i \in \mathcal{V}$ **do**
    $N_i \leftarrow$ neighbours of v_i;
    $G_{v_i} \leftarrow r_i + \sum_{v_j \in N_i} r_j \times w_{ij}$;
  end
  $v_{\mathcal{I}} \leftarrow$ node with maximum G_{v_i} ;
  $\Gamma \leftarrow$ subgraph of $\mathcal{G}$ with $v_{\mathcal{I}}$ and its 2-hop neighbourhood ;
  $S_{\mathcal{K}} \leftarrow$ top-$\mathcal{K}$ sentences after sentence compression on $v_i \in \Gamma$;
  $\mathcal{S} \leftarrow \mathcal{S} \cup S_{\mathcal{K}}$;
  forall $v_k \in \mathcal{V}$ **do**
    $r_k \leftarrow r_i \times (1 - w_{\mathcal{S}_{\mathcal{K}} k})$;
  end
until $\forall v_i$, $r_{v_i} \leq$ *threshold*;
C $\leftarrow$ ILP($\mathcal{S}$) ; // Integer Linear Program (Eqns. 2-7)
return *C*

as the 'author' content snippet and the set of documents as the input. Further, we considered the reference summaries as the articles that the author has constructed and is used as a basis for our quality evaluations. The average length of the reference articles is used as the desired length of the generated content.

In order to evaluate the generated content, we use the following **metrics** to measure different aspects of the generated content.

Rouge-N (Lin, 2004) scores measure the precision, recall and f-measure for the occurrence of n-grams in the generated summary with respect to the reference human generated summary. We compute the Rouge scores of the generated content against the reference article.

To compute the **coherence** of the generated content, we follow the definition of local coherence by (Lapata and Barzilay, 2005). Measuring local coherence is quantified by the the degree of semantic relatedness between consecutive sentences and is obtained by taking the mean of all individual transitions. Coherence is given by,

$$coherence(c_{\text{gen}}) = \frac{\sum_{i=1}^{n-1} sim(c_{\text{gen}}^{i}, c_{\text{gen}}^{i+1})}{n-1} \tag{8}$$

where $sim(c_{\text{gen}}^{i}, c_{\text{gen}}^{i+1})$ is a measure of semantic similarity between sentences c_{gen}^{i} and c_{gen}^{i+1}.

To measure the **relevance** of the generated content to the input snippet, we follow the formulation by (Srinivasan et al., 2017). Relevance is computed as a weighted average given by,

$$rel(c_{\text{ref}}, c_{\text{gen}}) = \sum_{i=1}^{M} \frac{\sum_{k=1}^{TopK(c_{\text{ref}}, c_{\text{gen}}^{i})} \gamma^{k} sim(c_{\text{gen}}^{i}, c_{\text{ref}}^{k})}{M \sum_{k=1}^{K} \gamma^{k}}, \tag{9}$$

where c_{ref} are the sentences in the reference and $c_{\text{gen}}^{1 \ldots M}$ are the sentences in the generated content. $TopK(c_{\text{ref}}, c_{\text{gen}}^{i})$ returns the top-K sentences in the reference content similar to c_{gen}^{i} (in the decreasing order of their similarly as computed by $sim(c_{\text{exp}}^{i}, c_{\text{inp}}^{k})$). The parameter γ, $(0 \leq \gamma \leq 1)$, penalizes the addition of a sentence that is similar to only a small set of the input sentence via a decayed-weighted-average.

Finally, to calculate the **information diversity** in the generated content, we again extend the formula-

tion by (Srinivasan et al., 2017) and compute a weighted average given by,

$$div(c_{\text{gen}}) = \sum_{i=1}^{N} \frac{\sum_{k=1}^{TopK(c_{\text{gen}}, c^i_{\text{gen}})} \gamma^k sim(c^i_{\text{gen}}, c^k_{\text{gen}})}{N \sum_{k=1}^{K} \gamma^k}. \quad (10)$$

4.1 Information Coverage and Diversity:

Picking content from a corpus to construct a text can yield redundant content because of multiple representations of the same information in the corpus. If the final content is budgeted, this can result not only in redundant content, but also in a lost opportunity to cover other aspects of the content. In order to address this, we extend the sentence compression (Filippova, 2010) to combine multiple such sentences to provide a compressed content.

However, performing the sentence compression on a larger set of sentences may result in the compression of totally different pieces of information and can often be detrimental to information quality. Our proposed graph formulation identifies similar sentences to be compressed. The iterative subgraph selection in our formulation also enables us to cover different aspects of the content that are required to be generated.

Fig. 1(a) shows the performance of the sentence-compressed content against the uncompressed content selected based on relevance maximization (Carbonell and Goldstein, 1998). We have used box-and-whiskers plots for our results in this paper. The box gives the boundaries of the first and third quartile, and the middle band shows the median value. The whiskers extending from the box boundaries on both sides show the minimum and maximum values, except for the outliers which are shown as points by themselves.

The Rouge scores are comparable across the 3 methods indicating similar quality of the different settings. Similarly, the relevance of the generated content is also comparable across the 3 methods. Since the graph based compression iteratively selects candidates to compress based on those previously generated, it produces content that is more diverse as indicated in Fig. 1(a). Since the plain sentence compression works on a large set of candidates, the diversity improvement is lower than that for the graph based compression. This validates our hypothesis that graph based compression yields diverse yet relevant content.

Finally, the graph based sentence compression improves the overall readability of the text based on the Flesch reading ease (Kincaid et al., 1975) suggesting that the improvements observed with relevance and diversity are not coming at the cost of the linguistic quality of the generated content.

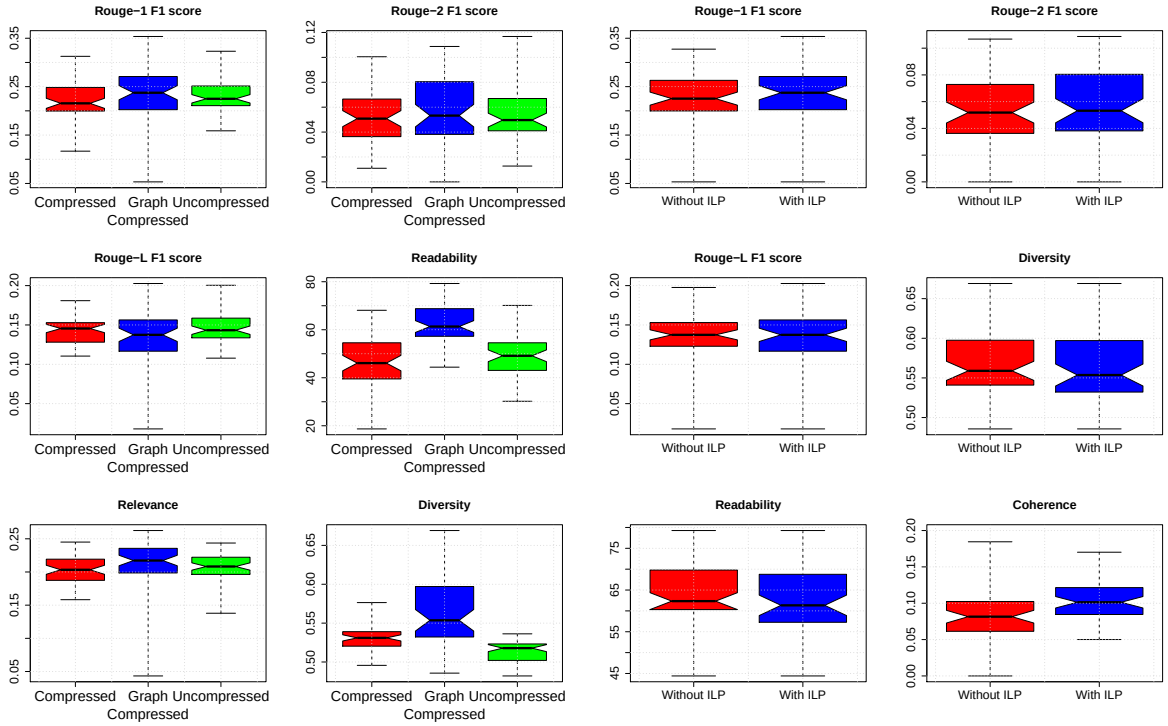
4.2 Coherence with ILP formulation:

The ILP formulation reorders the generated content in a way that improves the overall coherence of the content. Fig. 1(b) shows the performance of the generation with and without ILP. For the non-ILP version, candidates were selected from the compressed sentences via relevance maximization (Carbonell and Goldstein, 1998). It can be seen that the ILP formulation improves the overall coherence of the generated content while maintaining the other quality aspects.

5 Performance evaluation of CCC against baselines

To position our work against existing work, we use two state-of-the-art baseline systems. Our first baseline is GEMS, proposed by (Taneva and Weikum, 2013). GEMS is an extractive generation framework that uses an ILP formulation to extract a set of sentences towards the required content optimizing for the coherence by selecting contiguous sentence fragments. Our second baseline, WikiWrite (Banerjee and Mitra, 2016), has been used to generate Wikipedia articles in a coherent fashion. Similar to our approach, WikiWrite also uses a combination of sentence compression with ILP, however, the sentence compression does not account for the information covered by previously generated candidates, leading to lesser diversity.

Our goal is to obtain the best generation given a set of retrieved text. Alternative methods to modify *tf*idf* criteria (Lecorvé et al., 2008) could be used to increase the specificity of retrieved documents to



(a) Information Coverage & Diversity in the generated content (b) Coherence of the generated content with and without ILP with and without Sentence Compression

Figure 1: Impact of different stages of CCC on the various quality aspects of the generated content

the topics in the query. However, to prevent any bias in the quality of query search results, we ensure that the seed set of documents provided as input to each of the approaches is the same (specified in Section 3) without any such modifications.

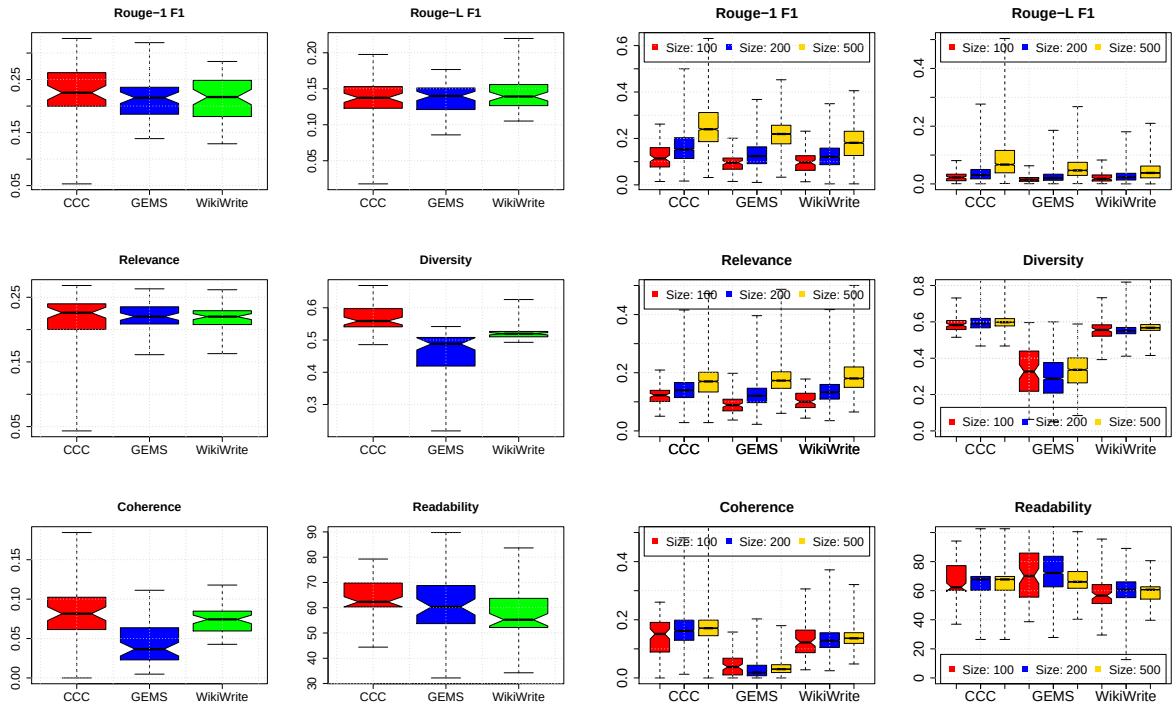
In the rest of this section, we present a comparison of CCC against the baselines. To check for statistical significance, we employ the Wilcoxon signed-ranked sum test (Wilcoxon, 1945), comparing the paired difference in values between the baseline and CCC. Test statistic and p-value indicated are from one sided tests, as indicated. The significance level is taken at $p = 0.05$. We also report the sum of ranks assigned to differences with positive sign (V) for each test.

Fig. 2(a) shows the Rouge scores for the generated content against the human generated content. At the entity level, the proposed generation has a significantly better performance against both the extractive (GEMS) ($V = 977$, $p = 0.042$) and abstractive (WikiWrite) ($V = 979, p = 0.040$) approaches. When compared beyond unigrams, the generation quality is comparable as indicated by the Rouge-L scores.

Since all the methods optimize for the relevance, the proposed approach has comparable relevance to the baselines. However, our proposed approach yields a higher information diversity and coverage as compared to the other approaches. As observed previously, sentence compression results in more information compressed into the content and hence both WikiWrite and our approach generate more diverse content. The diversity of our approach is more than that of WikiWrite due to the optimized sentence selection that precedes compression in our approach. This indicates that our approach yields an article much **more informative** than those generated by the baselines - GEMS ($V = 1540$, $p < 0.001$) and WikiWrite ($V = 1440$, $p < 0.001$).

GEMS is outperformed by both our approach ($V = 1440, p < 0.001$) and Wikiwrite ($V = 1398$, $p < 0.001$) at coherence, since both of these algorithms account for coherence explicitly. The coherence of our algorithm and Wikiwrite was found to be comparable.

Finally, we compute the Flesch reading ease (Kincaid et al., 1975) to quantify the readability of the generated content as its based on the Flesch reading ease. Since GEMS uses human generated sentences as-is from the input, it yields readable sentences. Unlike WikiWrite, our approach compresses only



(a) DUC 2007 Data

(b) Australian Legal Data

Figure 2: Generated content quality for CCC against GEMS ((Taneva and Weikum, 2013)) and WikiWrite ((Banerjee and Mitra, 2016)) on the Australian Legal Dataset (Galgani et al., 2012)

related sentences and thus does not result in unusually larger and less readable sentences, thus, our readability is higher ($V = 1380$, $p < 0.001$). The reading ease is comparable to GEMS despite not using the sentences as-is.

To conclude, we are able to achieve a performance which parallels that of the better of the two baselines in most aspects, while significantly outperforming them in other aspects.

Australian Legal Dataset: Although the performance of our proposed algorithm on the DUC2007 dataset is promising, the dataset is relatively small as compared to an enterprise content repository. In order to test the performance on a larger enterprise-like corpus, we evaluate the proposed approach on the Australian Legal Case Reports dataset[1] (Galgani et al., 2012), a collection of 3890 legal cases from the Federal Court of Australia (FCA) from 2006 to 2009. We used the gold standard summary for every case as the input snippet and the actual article as the reference. The legal articles were 6406-word long on an average, while the summaries were 65-word long on an average. Here, we also try to generate articles of different length - 100, 200 and 500 - to test the robustness of extraction. Fig. 2(b) shows the generation quality for our approach against the baselines. The trends observed in Fig. 2(a) for the DUC dataset carry over to Australian Legal Case Reports one as well. This validates the performance of our approach across corpora.

As the target size of generated content is increased, the algorithms' output becomes increasingly more relevant to the original article. The diversity of the proposed algorithm also increases with a minor dip in coherence. This can be attributed to the fact that the coherence measures inter-sentence topical overlap and with increased diversity, there is a minor dip in local coherence since the topic overlap will reduce.

[1] `http://archive.ics.uci.edu/ml/datasets/Legal+Case+Reports`, Accessed 16 March 2017.

6 Conclusion

We studied the problem of constructing a piece of text by picking content from an existing corpus and combining them by optimizing relevance, information coverage and coherence with reduced redundancy. Evaluations based on several metrics against state-of-the-art baselines shows superior performance of the proposed solution and hence demonstrates the feasibility of the approach. We believe that automated text generation will play a key role in smart authoring workflows for several domains in the near future.

References

Anne Aula. 2003. Query formulation in web information search. In *ICWI*.

Ramakrishna B. Bairi, Raghavendra Udupa, and Ganesh Ramakrishnan. 2016. A framework for task-specific short document expansion. In *ACM International Conference on Information and Knowledge Management (CIKM)*.

Siddhartha Banerjee and Prasenjit Mitra. 2016. Wikiwrite: Generating wikipedia articles automatically. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document abstractive summarization using ilp based multi-sentence compression. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. In *53rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *21st ACM SIGIR Conference on Research and Development in Information Retrieval*.

Hoa T. Dang. 2007. Overview of DUC 2007. In *Document Understanding Conference*.

Pawan Deshpande, Regina Barzilay, and David R Karger. 2007. Randomized decoding for selection-and-ordering problems. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Katja Filippova. 2010. Multi-sentence compression: Finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*.

Filippo Galgani, Paul Compton, and Achim Hoffmann. 2012. Combining different summarization techniques for legal text. In *Workshop on Innovative Hybrid Approaches to the Processing of Textual Data*.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Technical Training Command Millington TN Research Branch.

Mirella Lapata and Regina Barzilay. 2005. Automatic evaluation of text coherence: Models and representations. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Gwénolé Lecorvé, Guillaume Gravier, and Pascale Sébillot. 2008. An unsupervised web-based topic language model adaptation method. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Yanran Li and Sujian Li. 2014. Query-focused multi-document summarization: Combining a topic model with graph-based semi-supervised learning. In *23rd International Conference on Computational Linguistics (COLING)*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: ACL 2004 workshop*.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking documents to encyclopedic knowledge. In *ACM Conference on Information & Knowledge Management (CIKM)*.

Mir Tafseer Nayeem and Yllias Chali. 2017. Extract with order for coherent multi-document summarization. In *TextGraphs-11: Workshop on Graph-based Methods for Natural Language Processing*.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Springer Information Retrieval*.

Daraksha Parveen, Mohsen Mesgar, and Michael Strube. 2016. Generating coherent summaries of scientific articles using coherence patterns. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Christina Sauper and Regina Barzilay. 2009. Automatically generating wikipedia articles: A structure-aware approach. In *47th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nico Schlaefer, Jennifer Chu-Carroll, Eric Nyberg, James Fan, Wlodek Zadrozny, and David Ferrucci. 2011. Statistical source expansion for question answering. In *ACM International Conference on Information & Knowledge Management (CIKM)*.

Balaji Vasan Srinivasan, Rishiraj Saha Roy, Harsh Jhamtani, Natwar Modani, and Niyati Chhaya. 2017. Corpus-based automatic text expansion. *18th International Conference on Computational Linguistics and Intelligent Text Processing*.

Bilyana Taneva and Gerhard Weikum. 2013. Gem-based entity-knowledge maintenance. In *22nd ACM International Conference on Information & Knowledge Management (CIKM)*.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multi-document summarization. In *51st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Xun Wang, Masaaki Nishino, Tsutomu Hirao, Katsuhito Sudoh, and Masaaki Nagata. 2016. Exploring text links for coherent multi-document summarization. In *The 26th International Conference on Computational Linguistics (COLING)*.

Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics bulletin*.

Bridging resolution: Task definition, corpus resources and rule-based experiments

Ina Rösiger, Arndt Riester and Jonas Kuhn
Institute for Natural Language Processing
University of Stuttgart, Germany
{roesigia,arndt,jonas}@ims.uni-stuttgart.de

Abstract

Recent work on bridging resolution has so far been based on the corpus ISNotes (Markert et al., 2012), as this was the only corpus available with unrestricted bridging annotation. Hou et al.'s (2014) rule-based system currently achieves state-of-the-art performance on this corpus, as learning-based approaches suffer from the lack of available training data. Recently, a number of new corpora with bridging annotations have become available. To test the generalisability of the approach by Hou et al. (2014), we apply a slightly extended rule-based system to these corpora. Besides the expected out-of-domain effects, we also observe low performance on some of the in-domain corpora. Our analysis shows that this is the result of two very different phenomena being defined as bridging, which we call referential and lexical bridging. We also report that filtering out gold or predicted coreferent anaphors before applying the bridging resolution system helps improve bridging resolution.

1 Introduction

Bridging is an anaphoric phenomenon where the interpretation of a bridging anaphor, sometimes also called associative anaphor (Hawkins, 1978), is based on the non-identical associated antecedent. The corresponding NLP task of bridging resolution is about linking these anaphoric noun phrases and their antecedents, which do not refer to the same referent but are related in a way that is not explicitly stated. Bridging anaphors are thus discourse-new but dependent on previous context.

(1) Our correspondent in Egypt is reporting that **the opposition** is holding a rally against **the constitutional referendum**.[1]

One can think of bridging anaphors as expressions with an implicit argument, e.g. *the opposition (in Egypt)*. Compared to coreference resolution, which has become one of the standard NLP tasks, the progress in bridging resolution is much slower. The main issue for most researchers aiming to apply statistical algorithms to this task is the lack of training data, as well as the rather diverse bridging definitions and annotations. The resolution of bridging links is important because it can prove beneficial in tasks which use the concept of textual coherence, for example Barzilay and Lapata's (2008) entity grid or Hearst's (1994) text segmentation.

Note that while a benchmark dataset for bridging has not yet been established, most recent work is based on the ISNotes corpus (Markert et al., 2012), which contains Wall Street Journal articles. Full bridging resolution on this corpus has been investigated in Hou et al. (2014), following earlier experiments on the subtasks of bridging anaphor detection (Hou et al., 2013a) and antecedent selection (Hou et al., 2013b). Apart from this, there is some work on bridging detection as a subclass of information status classification, where bridging is typically a category with low annotator agreement and low detection accuracy (Markert et al., 2012; Rahman and Ng, 2012; Hou, 2016a). In the meantime, a few other corpora

[1] Anaphors are marked in bold face, their antecedents are underlined.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3516–3528
Santa Fe, New Mexico, USA, August 20-26, 2018.

have been made available (some of the same domain as ISNotes, newspaper, some of other domains) which make it possible to assess how generalisable the rule-based approach is.

The focus of this paper lies on full bridging resolution but we also report numbers for anaphor detection. We start with a re-implementation of Hou et al. (2014),[2] which we extend with one new rule to find more general bridging cases, as we think that this contributes to the generalisability of the approach, before we apply the extended system to other corpora. We also include a couple of additional experiments, where we compare predicted and gold markables and investigate the effect of coreference information. We report that filtering out gold or even just predicted coreferent anaphors before bridging resolution significantly helps improve bridging resolution.

Besides the expected out-of-domain effects, we also observe low performance on some of the in-domain corpora. Our analysis shows that this is the result of two very different – though often co-occurring – phenomena being defined as bridging, namely *referential* and *lexical bridging*, which is why we have included a rather extensive review of bridging definitions in the next section.

2 Defining bridging

Bridging has been examined in many theoretical studies (Clark, 1975; Hawkins, 1978; Hobbs et al., 1993; Asher and Lascarides, 1998; Baumann and Riester, 2012) as well as in corpus and computational studies (Fraurud, 1990; Poesio et al., 1997; Vieira and Teufel, 1997; Poesio and Vieira, 1998; Poesio et al., 2004; Nissim et al., 2004; Nedoluzhko et al., 2009; Lassalle and Denis, 2011; Cahill and Riester, 2012; Markert et al., 2012; Hou et al., 2013b; Hou et al., 2013a; Hou, 2016b; Zikánová et al., 2015; Grishina, 2016; Roitberg and Nedoluzhko, 2016; Riester and Baumann, 2017).

Unlike in work on coreference resolution, these studies do not follow an agreed upon definition of bridging. On the contrary, many different phenomena have been described as bridging. While some of the issues have been controversial for a long time, e.g. the question of definiteness, the importance of the distinction between *referential* and *lexical* bridging, inspired by the two-level *RefLex* annotation scheme by Baumann and Riester (2012), became evident in our experiments. The two terms describe two different phenomena which are currently both defined and annotated as bridging.

2.1 Referential bridging

Referential bridging describes bridging at the level of referring expressions, i.e. we are considering noun phrases that are truly anaphoric, in the sense that they need an antecedent in order to be interpretable, like in (2). As such, (referential) bridging anaphors are non-coreferent, context-dependent expressions.

(2) The city is planning a new townhall and **the construction** will start next week.

Referential bridging is often a subclass of (referential) information status annotation. We claim that referential bridging anaphors can be seen as expressions which require for their interpretation the antecedent as an implicit argument, e.g. *the construction of the new townhall* in (2). When uttered out of context, their referent is unidentifiable. The two above-mentioned examples are captured by this linguistically motivated definition.

Referential bridging anaphors are typically short, definite expressions (*the construction, the door*), and several accounts explicitly restrict bridging to definites, e.g. Poesio and Vieira (1998), Nedoluzhko et al. (2009), Grishina (2016), Rösiger (2016) or Riester and Baumann (2017), while others also allow for indefinite bridging, e.g. Löbner (1998) or Markert et al. (2012), with the consequence that some studies have linked indefinites as bridging anaphors (e.g. in ISNotes and others). Although having held different views on this issue, we nowadays think that indefinite expressions can indeed – in some cases – be referential bridging anaphors, for example in (3) or (4), where the (partitive) expressions *one employee (of Starbucks)* or *leaves (of the old oak tree)* are introduced.

(3) Starbucks has a new take on the unicorn frappuccino. **One employee** accidentally leaked a picture of the secret new drink.

[2]The system will be made available here: `https://github.com/InaRoesiger/BridgingSystem`

(4) Standing under the old oak tree, she felt **leaves** tumbling down her shoulders.

However, while short, definite expressions signal identifiability and are thus either anaphoric expressions or familiar items, it is much harder to decide which indefinite expressions are bridging anaphors, since indefinite expressions are prototypically used to introduce new discourse referents and principally do not need an antecedent/argument in order to be interpretable. This is, for example, also reflected in the higher inter-annotator-agreement for definite than for indefinite bridging anaphors (Rösiger, 2018a).

Thus, despite the interpretational uncertainty surrounding indefinites, we take linguistic anaphoricity/context-dependence to be the defining criterion for referential bridging. Semantic relations like meronymy will be addressed in the next section under the notion *lexical bridging*. In this connection, it is important to concede, however, that the reason why certain definite or indefinite expressions function as bridging anaphors (while others do not) is typically due to some kind of semantic proximity between antecedent and anaphor. However, the specific relation we are dealing with may be rather abstract, vague and difficult to define, as the examples in (1)-(3) show.

2.2 Lexical bridging

Baumann and Riester (2012) use the term *lexical accessibility* to describe lexical semantic relations, such as meronymy or hyponymy, at the word or concept level (e.g. *house – door*). It is important to bring to mind that lexical relations are defined as part of the intrinsic meaning of a pair of concepts, thus, abstracting away from specific discourse referents: it is the words *house* and *door* which stand in a meronymic relation, not two actual physical objects or their mental images, although typically the referents of a holonym-meronym combination will, at the same time, stand in a physical whole-part relation. Since this physical relation has often been taken as one of the defining criteria for bridging, e.g. by Gardent et al. (2003), Nissim et al. (2004), Nedoluzhko et al. (2009) or Grishina (2016), we suggest to use the term *lexical* (or *lexically induced*) *bridging* for this phenomenon.

The referents of the proper nouns *Europe* and *Spain* are in a whole-part relation,[3] and the referring expressions can thus be considered a case of lexical bridging. However, the expression *Spain* is not anaphoric, since its interpretation does not depend on the "antecedent" *Europe*. Whole-part is probably the prototypical pre-defined relation, and it is a straightforward concept to annotate in the case of nouns denoting physical objects. However, it is less applicable in connection with abstract nouns, which is why many additional relations have been suggested, including, for instance *thematic role in an event, attribute of an object* (like *price*), *professional function in an organisation* (like *president*), *kinship* (like *mother*), *possessed entity* and so on. And yet, few schemes get by without an "other" category for the many examples which cannot be naturally classified into one of the assumed classes.

It should be noted that lexical and referential bridging are two different concepts with completely different properties: one deals with the question of pragmatic anaphoricity (or grammatical saturation) of an expression, the other with lexical proximity between two words and the relation between entities in the real world, although the two types of bridging often co-occur within one and the same pair of expressions, such as in (5), where we have a relation of meronymy between the content words *sea urchin(s)* and *spine(s)*, but also an anaphoric relation between the referring expressions *most sea urchins* and *the spines*, i.e. a case of referential bridging.

(5) In most sea urchins, touch elicits a prompt reaction from **the spines**.

The second release of the ARRAU corpus (Uryupina et al., to appear, first released in Poesio and Artstein, 2008), as used in the first shared task on bridging resolution, for example, contains instances of both referential and lexical bridging, with the majority of the bridging links being purely lexical bridging pairs, i.e. most expressions labeled as bridging are actually not context-dependent.

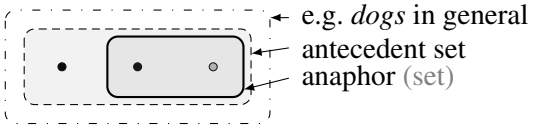
[3]Note that for proper nouns (names), like *Spain*, there is a one-to-one mapping between the word and its referent in the real world, which is not the case for common nouns, cf. Kripke (1972).

2.3 Subset relations and lexical givenness

Another relation often brought up in connection with (lexical) bridging is the *subset* or *element-of* relation, which is the most common relation in ARRAU. In principle, an expression referring to an element or a subset of a previously introduced group can be of the referential type of bridging, like in (6), where the anaphor is interpreted as *the small pug (from the prementioned group of dogs)*, but this is not always the case, as (7) shows.

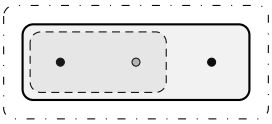
(6) I saw some dogs yesterday. **The small pug** was the cutest.

(7) Newsweek said it will introduce the Circulation Credit Plan, which awards space credits to advertisers on renewal advertising. The magazine will reward with page bonuses **advertisers who in 1990 meet or exceed their 1989 spending**, [. . .]



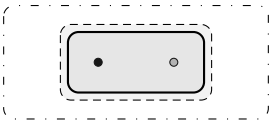
The subset relation can sometimes be reversed, as shown in (8).

(8) I saw a small pug yesterday. I like **many dogs**.



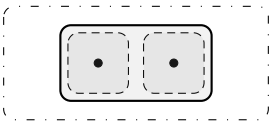
It should be noted, however, that subset/element-of pairs also have a lot in common with coreference pairs, since the lexical relation between their head nouns tends to be hypernymy, synonymy or plain word repetition (lexical relations which are summarised as *lexical givenness* in Baumann and Riester, 2012) or hyponymy (i.e. *lexical accessibility*). Note that, although the antecedent and anaphor expressions in (9) stand in a hypernym-hyponym relation (or reverse), their respective referent is the same. Hence, these cases do not exemplify bridging but coreference.

(9) a. I saw a dog yesterday. **The small pug** was very cute.
b. I saw small pugs yesterday. **The dogs** were very cute.



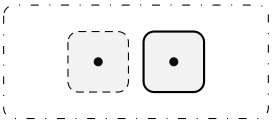
Note that *element-of* bridging is also conceptually very close to the phenomenon of aggregation/summation, in which the group entity follows a list of elements, and which also counts as a case of coreference.

(10) I saw a pug and a Yorkshire terrier. **The dogs** were very cute.



A final case, which is treated as a special class of information status in Markert et al. (2012) and annotated as a subclass of bridging in ARRAU, are so-called *comparative* or *other*-anaphors. The head noun of the anaphor must be lexically given (Riester and Piontek, 2015, 242f.) and the two expressions are marked as two contrastive elements from the same alternative set (Rooth, 1992). Comparative anaphors can be considered cases of referential bridging where the implicit argument is the implicit or explicit alternative set, i.e. *another dog (from a specific or unspecific set dogs)*.

(11) I saw a small pug two days ago and **another dog** yesterday.



2.4 Near-identity

While many approaches distinguish only between coreferent anaphors, which refer to the same referent as their antecedent, and bridging anaphors, which refer to a different referent, Recasens and Hovy (2010) and Recasens et al. (2012) have introduced a third concept, the concept of *near-identity* which has been picked up by others (e.g. Grishina, 2016). Near-identity is defined to hold between an anaphor and an antecedent whose referents are almost identical, but differ in one of four respects: name metonomy, meronymy, class or spatio-temporal functions.

(12) On homecoming night Postville feels like Hometown, USA, but a look around this town of 2,000 shows its become a miniature Ellis Island ... For those who prefer **the old Postville**, Mayor John Hyman has a simple answer.

We believe that the introduction of this additional category in between coreference and bridging introduces more uncertainty and, therefore, potentially makes the annotation process more difficult. Ex. (12), for instance, is structurally analogous to comparative anaphors.

3 Available corpus resources

Table 1 presents English corpora containing bridging annotations as well as their number of bridging pairs, limitations on the bridging anaphor, the type of bridging and the respective domain. Each corpus is quickly summarised below.

Corpus	# of pairs	Anaphor	Type	Domain
ISNotes	663	all	referential	news
ARRAU	5,512	all	mostly lexical, some referential	news, (narrative, dialogue)
BASHI	459	all	referential	news
SCiCorp	1366	only definite	referential	scientific text
CorefPro	188	only definite	referential	news, narrative, medicine
GUM	(growing)	all	referential, some lexical	dialogue, narrative, informative, instructional, ...

Table 1: Overview of available English corpora containing bridging annotations

There exist a few older corpora with bridging annotations which are not listed here because their annotations deviate from the ones presented here in one or several aspects. For example, in earlier experiments, "different-head-coreference", compare Ex. (9), was considered bridging (e.g. in Poesio and Vieira, 1998). As the bridging corpora are already rather diverse, we limit ourselves to the ones presented in Table 1.

ISNotes The ISNotes corpus (Markert et al., 2012), a corpus of newspaper text, contains bridging as a subclass of information status annotation, with 633 annotated bridging pairs. It contains definite and indefinite bridging anaphors, but no comparative anaphors, as these cases were considered a different information status category. To the best of our knowledge, the ISNotes corpus has been the only corpus on which recent results have been published, cf. Hou et al. (2013b; 2014).

ARRAU Recently, the first shared task on bridging resolution was announced. As a data basis, the second release of the ARRAU corpus was used, which contains 5,512 bridging pairs in three different domains: news text, dialogue and narrative text. This is, as far as we know, currently the largest corpus resource containing bridging pairs. However, only a small subset of the annotated pairs contains truly anaphoric bridging anaphors (cf. Section 2 for the distinction between referential and lexical bridging).

BASHI The BASHI corpus (Rösiger, 2018a) is a corpus of news text, which contains 459 bridging pairs, of which 114 are labeled as indefinite bridging anaphors and 70 as comparative anaphors.

SciCorp SCiCorp (Rösiger, 2016) is a corpus annotated with information status and bridging as a subclass. It contains scientific text of two disciplines, computational linguistics and genetics. 1366 bridging pairs were annotated. However, the bridging pairs contain pairs where possessive pre-modification was involved, as in (13).

(13) them ... **their interest**.

This is sometimes called *containing inferrable* (Prince, 1981) or *bridging-contained* (Baumann and Riester, 2012), as the antecedent is a syntactic argument within the markable. We filter out these cases, as we think that they should not be included in the category bridging proper, since anaphoricity is achieved by linking the pronoun *their* to its coreferential antecedent.

GUM The GUM corpus (Zeldes, 2017) is a corpus of (currently) 85,350 tokens The corpus is expanded every year by students as part of a curriculum at Georgetown University. As GUM contains bridging annotation as part of their information status classification, most bridging pairs are expected to be referential bridging. However, after a quick scan of the data, we also found some non-anaphoric, lexical bridging (or lexically given) pairs, e.g. in (14) or (15).

(14) However, there are four hotels in Hadibo – **Taj Socotra Hotel**, **Hafiji Hotel**, **Socotra Hotel** and **Summer land Hotel**.

(15) Name your language. This is the most fundamental property in **all languages**. You have many names to choose from [...] **most languages** [...]

GUM also contains a number of bridging-contained cases as well as some cases of aggregation, which we see as a special case of coreference, e.g. in (16).

(16) Mix .2 grams of luminon, 4 grams of sodium carbonate, .4 grams of copper sulfate [...] in a second bowl. Unfortunately, **these hazardous chemicals** will not float freely in mid-air like this graphic suggests.

CorefPro Grishina (2016) recently described a parallel corpus of German, English and Russian texts with 432 German bridging pairs that have been transferred to their English and Russian counterparts, resulting in 188 transferred English bridging pairs. The corpus has recently been made available.[4] In contrast to the other corpora, they apply a three-way classification: anaphors can be coreferent, bridging or of the category near-identity. Only definite anaphors were annotated.

4 Experimental setup

4.1 Corpora

ISNotes Hou et al. (2014) split the corpus into a development (10 documents) and test set (40 documents). The rules were optimised on the development set and the performance of the system reported on the test set. Unfortunately, the concrete development/test split is not specified. We report numbers for our own test-development-split[5] as well as for the whole corpus.

ARRAU The data was obtained from the LDC and consists of training, development and test sets for the three domains newspaper, narrative text and dialogue, with most of the text being news text. As the number of bridging anaphors in the narrative and dialogue part is quite small, we report numbers only on the test set of the news part (RST). This is also done in order to be compatible to the scores of the shared task associated with this data, which also focused on the RST domain.

BASHI The data was downloaded from the webpage.[6] As we simply apply our systems to this data, we report performance on the whole corpus.

SCiCorp The data was downloaded from the webpage.[7] Again, we report numbers on the whole corpus.

[4] https://github.com/yuliagrishina/corefpro

[5] The 10 dev docs are: wsj1101, wsj1123, wsj1094, wsj1100, wsj1121, wsj1367, wsj1428, wsj1200, wsj1423, wsj1353.

[6] http://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/bashi.html

[7] http://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/scicorp.html

Rule	Example	Anaphor	Antecedent search	Window
1	A white woman's house ← The basement	building part	semantic connectivity	2
2	She ← Husband David Miller	relative	closest person NP	2
3	The UK ← The prime minister	GPE job title	most frequent GEO entity	-
4	IBM ← Chairman Baker	professional role	most frequent ORG NP	4
5	The firms ← Seventeen percent	percentage expression	modifying expression	2
6	Several problems ← One	number/indefinite pronoun	closest plural, subject/object NP	2
7	Damaged buildings ← Residents	head of modification	modifying expression	-
8	A conference ← Participants	arg-taking noun, subj pos.	semantic connectivity	2

Table 2: Overview of rules in Hou et al. (2014).

4.2 Evaluation metrics

The evaluation of bridging resolution is computed using the widely known precision and recall measures (and the harmonic mean between them, F1). The bridging anaphor is considered a mention, while the antecedent is considered an entity. This is taken into account by including gold coreference chains during the evaluation. If the predicted antecedent is one of the mentions in the coreference chain of the gold antecedent, the bridging pair is considered correct. The evaluation is rather strict, as overlapping markables are considered wrong. Furthermore, bridging anaphors with more than one link, e.g. comparative anaphora in the sense of (17), cannot be predicted with our system, as it is based on predicting just one antecedent.

(17) Canada, the US and **other countries**

It is also unclear how these multiple antecedents should be evaluated in case of partial correctness. In the current study, when the pair *the US* and *other countries* is suggested by the system, it is considered wrong. The same holds for antecedents with discontinuous or empty antecedents, which are contained in the ARRAU corpus.

5 Re-implementation of Hou et al. (2014)

As a basis for our experiments, we re-implement the rule-based system proposed by Hou et al. (2014).[8] For more details on the re-implementation, please refer to the supplementary material of this paper.[9]

The re-implementation comprises all three components of the original paper: pre-processing, rule adaptation and post-processing. During pre-processing, markables are extracted, which are then passed on to eight rules which predict bridging anaphor and antecedent pairs. These eight hand-crafted rules are based on linguistic intuitions about (referential) bridging. Most of the rules are rather specific, aiming at high precision, while some of the rules are designed to capture more general bridging cases, thus increasing the recall. Finally, the rules are applied in order of their precision. We extract NPs as our predicted markables. We also extract the markables of the information status annotation as our set of gold markables, where available. These form the initial set of anaphors and antecedents. We follow Hou et al.'s (2014) suggestion to exclude NPs whose head appeared before in the document, as these cases are typically involved in coreference chains. We also experiment with filtering out predicted and gold coreferent anaphors before applying the rules, described in Section 5.4.

5.1 Rules

Each rule is applied separately to the list of extracted markables and proposes pairs of bridging anaphors and antecedents. Table 2 gives an overview of the rules implemented. Two measures are computed independently of the actual bridging resolver and are needed as input for several rules, the semantic connectivity and the argument taking ratio. In order to find more general bridging cases, we have also implemented one additional rule, which is explained in Section 5.3.

[8]The re-implementation was necessary because the source code has not been made publicly available.

[9]Code and supplementary material: `https://github.com/InaRoesiger/BridgingSystem`

Computing the semantic connectivity The semantic connectivity between two words can be approximated by the number of times two words occur in a noun preposition noun pattern in a big corpus. This means that two nouns like *window* and *room* have a high semantic connectivity because they often occur as *windows in the room*, whereas other nouns do not appear often in such a construction and are therefore not highly semantically connected. Following Hou et al. (2014), we take the GigaWord corpus as the basis for the computation of the scores.

Computing the argument-taking ratio The argument taking ratio of a mention's head reflects how likely an NP is to take arguments (Hou et al., 2014), i.e. how *relational* an NP is. This can be used for bridging resolution as we assume the bridging anaphor to be lacking an implicit modifier in the form of the antecedent. If it has a low argument taking ratio, then the likeliness of an expression to be a bridging anaphor is also low. For example, the lemma *child* is often used without arguments, when we are generically speaking about *children*. *Brainchild*, however, seems to be an expression that is exclusively used with modification, e.g. in *the brainchild of* ... For details on the computation of the scores, please refer to the supplementary material.

5.2 Results

Setting	Corpus	Anaphor recognition			Full bridging		
		Precision	Recall	F1	Precision	Recall	F1
Hou (2014), gold markables	test set	61.7	18.3	28.2	42.9	11.9	18.6
Re-implementation with gold markables							
	test set	73.4	12.6	21.6	60.6	10.4	17.8
	whole corpus	65.9	14.1	23.2	57.7	10.1	17.2
Re-implementation with predicted markables							
	test set	69.3	12.2	20.7	57.7	10.1	17.2
	whole corpus	65.2	13.6	22.5	49.2	10.3	17.0
Extended re-implemenation (+new rule) using gold markables							
	test set	51.7	17.1	25.7	36.8	12.2	18.3
	whole corpus	45.9	18.3	26.2	32.0	12.8	18.3
Filtering out coreferent anaphors							
No coreference	whole corpus	45.9	18.3	26.2	32.0	12.8	18.3
Predicted coreference	whole corpus	68.6	18.3	28.9	47.9	12.8	20.2
Gold coreference	whole corpus	71.6	18.3	29.2	50.0	12.8	20.4

Table 3: Performance of the re-implementation of Hou et al. (2014), with different settings

Hou et al. (2014) state a precision of 42.9, a recall of 11.9 and an F1 score of 18.6 for full bridging resolution and a precision of 61.7, a recall of 18.3 and F1 score of 28.2 for anaphor detection. In both settings, they use gold markables but no coreference information. Table 3 contains the scores of the re-implementation for the test and the whole corpus when using gold or predicted markables. As mentioned above, we have defined a different test-development-split, which is why the results are not directly comparable. In general, however, we think that our re-implementation achieves comparable results, as our rules also achieve similar precision values and firing rates than in Hou (2016b), which are not shown here due to lack of space. As we have simply re-implemented the system from the original paper without any hand-tuning on the development set, we also report the numbers on the whole ISNotes corpus. Here, our re-implementation yields a precision of 57.7, a recall of 10.1 and an F1 score of 17.2 for full bridging resolution. Compared to the original numbers in Hou et al. (2014), we achieve higher precision, but lower recall, resulting in an overall lower F1 measure.

5.3 New rule and final performance

In order to include more general information and to increase recall, we apply the distributional (DS) classifier described in Shwartz and Dagan (2016) to distinguish certain semantic relations, e.g. hyponyms and meronyms. The classifiers input are word embeddings, taken from ConceptNet (Speer et al., 2017). As we expect the prototypical bridging relation to be the relation of meronymy (part-whole), we include this information in our bridging resolver, in the form of the following rule: the anaphor has to be a definite, unmodified expression in the form of *the N*. We search for an antecedent within the last three sentences

Corpus	Domain	Bridging type	Anaphor recognition			Full bridging		
			Prec.	Recall	F1	Prec.	Recall	F1
ISNotes (gold markables)	news	ref.	71.6	18.3	29.2	50.0	12.8	20.4
ISNotes (pred markables)	news	ref.	64.1	18.3	28.5	41.4	11.9	18.4
BASHI (pred)	news	ref.	49.4	20.2	28.7	24.3	10.0	14.1
ARRAU (original, gold mark.)	news	ref./lex.	13.3	0.9	1.7	2.2	0.2	0.3
ARRAU (adapted, gold mark.)	news	ref./lex.	29.2	32.3	30.8	18.5	20.6	19.5
SciCorp (pred)	scientific	ref.	17.7	0.9	8.1	3.2	0.9	1.5

Table 4: Performance of the rule-based method on other corpora. We use gold markables for ARRAU in order to be compatible with the shared task results, and predicted mentions for BASHI and SciCorp as they do not contain gold markables.

for which the classifier has predicted the pair of antecedent-anaphor to be an instance of meronymy. Additionally, the constraint that the two pairs need to have a cosine similarity score over the threshold of 0.2 has increased results (this threshold has been optimised on the development set). The new rule decreases precision, but significantly increases recall[10], which is why we get a significant increase in F1 score. The final system performance is shown in Table 3. More details on our experiment on integrating predictions from neural-net relation classifiers can be found in Rösiger et al. (2018).

5.4 Filtering out coreferent anaphors: Gold vs. predicted

Bridging anaphors are difficult to distinguish from coreferent anaphors, as they both often are short, unmodified expressions which are either interpretable because they are coreferent with a previously mentioned entity or are bridging anaphors which require an antecedent for their interpretation. Thus, we think it may be beneficial for the precision of our system to filter out coreferent anaphors before applying the bridging system. We experiment with three settings: (i) no coreference information, (ii) predicted coreference information and (iii) gold annotated coreference information. For predicted coreference, we applied the IMSHotCoref system (Björkelund and Kuhn, 2014) with its default settings on the ISNotes corpus.[11] We report the change in performance on the whole corpus, as there was no optimisation involved in the filtering of the coreferent anaphors. In Table 3, it can be seen that both predicted and gold coreference significantly improve the precision of the system, as well as the final F1 score. Surprisingly, the difference between gold and predicted coreference is small, compared to having no coreference information at all. The same effect can be observed with predicted mentions. We use gold coreference for the remaining experiments of the paper, as it is available in all corpora on which we want to apply our system.

6 Application to other corpora

6.1 BASHI (in-domain)

We first apply our re-implementation to a corpus of the exact same domain, BASHI. As can be seen in Table 4, the F1 score for anaphor recognition is 28.7, which is comparable to the score on ISNotes, although we observe a much lower precision on BASHI. Lower precision is also the reason for the overall lower score on BASHI for full bridging resolution, which means that the performance for anaphor detection is about the same, while we are worse in finding the correct antecedent. Still, the system performs relatively well on this data.

6.2 ARRAU (in-domain)

We apply our system to the test set of the RST (news) part, without making any changes. As can be seen in Table 4, the performance is extremely poor. We carefully analysed the reasons for the huge difference in performance between ISNotes/BASHI and ARRAU, which both contain Wall Street Journal articles and can thus not be explained with domain effects. We soon realised that the annotations differ quite a lot with respect to the understanding of the category bridging. We noticed that besides predicting wrong

[10] We compute significance using the Wilcoxon signed rank test (Siegel and Castellan, 1988) at the 0.05 level.

[11] We made sure to exclude the ISNotes part of OntoNotes from the training data for the coreference system, of course.

pairs, the original system would suggest bridging pairs which are fine from a referential point of view on bridging, but are not annotated in the corpus, such as (18).

(18) As competition heats up in Spain's bank market, [...] **The government** directly owns...

Additionally, it would miss a lot of lexical bridging and subset/lexical givenness pairs, as these often involve mentions with matching heads, which are filtered out in the pre-processing step of the system, such as in (19).

(19) Her husband and older son [...] run a software company. Certainly life for her has changed considerably since the days in Kiev, when she lived with her parents, her husband and **her two sons** in a 2 1/2-room apartment. (*relation: element-inverse*).

This is why the performance is so poor: a lot of referential bridging pairs which are not annotated were predicted, while the system missed almost all cases of (pure) lexical bridging. With the modular approach of the rule-based system, however, one can define new rules to also capture lexical bridging. The rules have been developed on the training and development set of the RST domain of the corpus and include rather specific rules, to find e.g. locations *Hollywood – LA* or *Los Angeles – California*, as well as more general rules to find the predominant relations in the corpus, namely subset and element-of. For more details on the adapted rule-based system, please refer to Rösiger (2018b). The final performance of the adapted system (F-score of 19.5) is also given in Table 4.

6.3 SciCorp (out-of-domain)

In contrast to ARRAU, SciCorp is an out-of-domain corpus annotated with referential bridging. When applying our system, we observe that it really does not generalise well to completely different domains, as the F1 score for full bridging resolution drops to 1.46. SciCorp also differs from BASHI and ISNotes with respect to the definiteness criterion: all bridging anaphors are definite. Of course, rules designed for indefinite anaphors cannot work. While we expected some of the rules designed for news text to perform poorly (e.g. building parts, relatives, job titles etc.), the rules designed to find more general cases of bridging also do not seem to predict a lot of pairs in this domain. The reason for this might also lie in the coverage of the semantic connectivity and argument-taking ratio, which are applied in these general rules: only 32 percent of the nouns in SciCorp are represented in the argument-taking-ratio lists, and only 3.9 percent of the noun pairs are contained in the semantic connectivity scores. Adding some in-domain text (e.g. large PubMed/ACL corpora) to the general corpora used to create these resources would be necessary to improve performance for the general rules of the system to work. We are positive that doing some form of domain adaptation, i.e. designing specific rules for scientific text and combining them with the improved general rules would lead to better results.

7 Conclusion and future work

We have presented a re-implementation of Hou et al. (2014), the state-of-the-art system for bridging resolution (which will be made publicly available), and applied it to other corpora, to test the generalisability of the approach. Besides the expected out-of-domain effects, we also observe low performance on some of the in-domain corpora. Our analysis shows that the system generalises well to in-domain corpora, if they are of the same type of bridging, namely referential bridging. We think that the distinction between referential and lexical bridging is a valuable contribution towards the understanding of the phenomenon of bridging and that it can also help design computational approaches. We also report that filtering out gold or predicted coreferent anaphors before bridging resolution helps improve bridging resolution.

As the GUM and CorefPro corpus have in the meantime been made available, we are planning to analyse the bridging annotations in these corpora according to our categorisation as well as to test the rule-based system on it.

References

Nicholas Asher and Alex Lascarides. 1998. Bridging. *Journal of Semantics*, 15(1):83–113.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Stefan Baumann and Arndt Riester. 2012. Referential and Lexical Givenness: semantic, prosodic and cognitive aspects. In Gorka Elordieta and Pilar Prieto, editors, *Prosody and Meaning*, number 25 in Interface Explorations. Mouton de Gruyter, Berlin.

Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 47–57, Baltimore, Maryland, June. Association for Computational Linguistics.

Aoife Cahill and Arndt Riester. 2012. Automatically acquiring fine-grained information status distinctions in German. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 232–236. Association for Computational Linguistics.

Herbert H. Clark. 1975. Bridging. In *Proceedings of the 1975 workshop on Theoretical issues in natural language processing*, pages 169–174. Association for Computational Linguistics.

Kari Fraurud. 1990. Definiteness and the processing of noun phrases in natural discourse. *Journal of Semantics*, 7(4):395–433.

Claire Gardent, Hélène Manuélian, and Eric Kow. 2003. Which bridges for bridging definite descriptions? In *Proceedings of EACL: Fourth International Workshop on Linguistically Interpreted Corpora*, pages 69–76, Budapest.

Yulia Grishina. 2016. Experiments on bridging across languages and genres. In *CORBON@ HLT-NAACL*, pages 7–15.

John A Hawkins. 1978. Definiteness and indefiniteness: A study in reference and grammaticality prediction. atlantic highlands.

Marti A. Hearst. 1994. Multi-paragraph segmentation of expository text. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 9–16.

Jerry R Hobbs, Mark E Stickel, Douglas E Appelt, and Paul Martin. 1993. Interpretation as abduction. *Artificial Intelligence*, 63:69–142.

Yufang Hou, Katja Markert, and Michael Strube. 2013a. Cascading collective classification for bridging anaphora recognition using a rich linguistic feature set. In *EMNLP*, pages 814–820.

Yufang Hou, Katja Markert, and Michael Strube. 2013b. Global inference for bridging anaphora resolution. In *Proceedings of NAACL-HLT*, pages 907–917.

Yufang Hou, Katja Markert, and Michael Strube. 2014. A rule-based system for unrestricted bridging resolution: Recognizing bridging anaphora and finding links to antecedents. In *EMNLP*, pages 2082–2093.

Yufang Hou. 2016a. Incremental fine-grained information status classification using attention-based lstms. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1880–1890.

Yufang Hou. 2016b. *Unrestricted Bridging Resolution*. Ph.D. thesis.

Saul Kripke. 1972. Naming and necessity. In Donald Davidson and Gilbert Harman, editors, *Semantics of Natural Language*, pages 253–355. Springer, Dordrecht.

Emmanuel Lassalle and Pascal Denis. 2011. Leveraging different meronym discovery methods for bridging resolution in french. *Anaphora Processing and Applications*, pages 35–46.

Sebastian Löbner. 1998. Definite associative anaphora. *manuscript) http://user. phil-fak. uniduesseldorf. de/˜ loebner/publ/DAA-03. pdf.*

Katja Markert, Yufang Hou, and Michael Strube. 2012. Collective classification for fine-grained information status. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 795–804. Association for Computational Linguistics.

Anna Nedoluzhko, Jiří Mírovskỳ, Radek Ocelák, and Jiří Pergler. 2009. Extended coreferential relations and bridging anaphora in the prague dependency treebank. In *Proceedings of the 7th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC 2009), Goa, India*, pages 1–16.

Malvina Nissim, Shipra Dingare, Jean Carletta, and Mark Steedman. 2004. An annotation scheme for information status in dialogue. *LREC 2004*.

Massimo Poesio and Ron Artstein. 2008. Anaphoric Annotation in the ARRAU Corpus. In *International Conference on Language Resources and Evaluation (LREC)*, Marrakech, Morocco, May.

Massimo Poesio and Renata Vieira. 1998. A corpus-based investigation of definite description use. *Computational linguistics*, 24(2):183–216.

Massimo Poesio, Renata Vieira, and Simone Teufel. 1997. Resolving bridging references in unrestricted text. In *Proceedings of a Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, pages 1–6. Association for Computational Linguistics.

Massimo Poesio, Rahul Mehta, Axel Maroudas, and Janet Hitzeman. 2004. Learning to resolve bridging references. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 143. Association for Computational Linguistics.

Ellen F. Prince. 1981. Toward a taxonomy of given-new information. In *Radical Pragmatics*, pages 223–55. Academic Press.

Altaf Rahman and Vincent Ng. 2012. Learning the fine-grained information status of discourse entities. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 798–807, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marta Recasens and Eduard Hovy. 2010. A typology of near-identity relations for coreference (nident). In *LREC*.

Marta Recasens, Maria Antonia Marti, and Constantin Orasan. 2012. Annotating near-identity from coreference disagreements. In *LREC*, pages 165–172.

Arndt Riester and Stefan Baumann. 2017. The RefLex Scheme – Annotation guidelines. SinSpeC. Working papers of the SFB 732 Vol. 14, University of Stuttgart.

Arndt Riester and Jörn Piontek. 2015. Anarchy in the NP. When new nouns get deaccented and given nouns dont. *Lingua*, 165(B):230–253.

Anna Roitberg and Anna Nedoluzhko. 2016. Bridging corpus for russian in comparison with czech. In *CORBON@ HLT-NAACL*, pages 59–66.

Mats Rooth. 1992. A theory of focus interpretation. *Natural Language Semantics*, 1(1):75–116.

Ina Rösiger, Maximilian Köper, Kim Anh Nguyen, and Sabine Schulte im Walde. 2018. Integrating predictions from neural-network relation classifiers into coreference and bridging resolution. In *Proceedings of NAACL-HLT: Workshop on Computational Models of Reference, Anaphora and Coreference*, New Orleans, USA, June.

Ina Rösiger. 2016. Scicorp: A corpus of english scientific articles annotated for information status analysis. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Ina Rösiger. 2018a. Bashi: A corpus of wall street journal articles annotated with bridging links. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Ina Rösiger. 2018b. Rule- and learning-based methods for bridging resolution in the arrau corpus. In *Proceedings of NAACL-HLT: Workshop on Computational Models of Reference, Anaphora and Coreference*, New Orleans, USA, June.

Vered Shwartz and Ido Dagan. 2016. Path-based vs. distributional information in recognizing lexical semantic relations. *arXiv preprint arXiv:1608.05014*.

Sidney Siegel and N. John Jr. Castellan. 1988. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, Berkeley, CA, 2nd edition.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*.

Olga Uryupina, Ron Artstein, Antonella Bristot, Frederica Cavicchio, Francesca Delogu, Kepa Rodriguez, and Massimo Poesio. to appear. Annotating a broad range of anaphoric phenomena, in a variety of genres: the arrau corpus. *Journal of Natural Language Engineering*.

Renata Vieira and Simone Teufel. 1997. Towards resolution of bridging descriptions. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 522–524. Association for Computational Linguistics.

Amir Zeldes. 2017. The gum corpus: creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612, Sep.

Šárka Zikánová, Eva Hajicová, Barbora Hladká, Pavlína Jínová, Jirí Mírovskỳ, Anja Nedoluzhko, Lucie Poláková, Katerina Rysová, Magdaléna Rysová, and Jan Václ. 2015. Discourse and coherence. *From the Sentence Structure to Relations in Text. Institute of Formal and Applied Linguistics*.

Semi-Supervised Disfluency Detection

Feng Wang[1,2], **Wei Chen**[1*], **Zhen Yang**[1,2] , **Qianqian Dong**[1,2], **Shuang Xu**[1], **and Bo Xu**[1]
[1]Institute of Automation, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences, Beijing, China
{feng.wang, wei.chen.media,yangzhen2014
dongqianqian2016 ,shuang.xu, xubo}@ia.ac.cn

Abstract

While the disfluency detection has achieved notable success in the past years, it still severely suffers from the data scarcity. To tackle this problem, we propose a novel semi-supervised approach which can utilize large amounts of unlabelled data. In this work, a light-weight neural net is proposed to extract the hidden features based solely on self-attention without any Recurrent Neural Network (RNN) or Convolutional Neural Network (CNN). In addition, we use the unlabelled corpus to enhance the performance. Besides, the Generative Adversarial Network (GAN) training is applied to enforce the similar distribution between the labelled and unlabelled data. The experimental results show that our approach achieves significant improvements over strong baselines.

1 Introduction

A characteristic of spontaneous speech is different from written text, since it's usually accompanied by disfluencies. Identifying and removing these non-fluent factors would help to improve the spontaneous speech quality. It often plays a significant role in understanding the semantics of these sentences and it's vital for the downstream NLP tasks, such as question answering, machine translation, and information extraction.

$$\text{I want to flight [} \underbrace{\text{to Boston}}_{\text{RM}} + \{ \underbrace{\text{um}}_{\text{IM}} \} + \underbrace{\text{to Denver}}_{\text{RP}} \text{]}$$

Figure 1: Example of disfluency annotation style in Switchboard corpus.

Figure 1 shows a standard annotation of disfluency structure (Shriberg, 1994), which includes three types of annotations: RM (*reparandum*, words that are discarded, or corrected by the following words), RP (*repair*, the correct words), and IM (*interregnum*, such as filled pauses, discourse cue words, etc.) The interruption point (+) marks the end of the reparandum and an optional interregnum. Ignoring the interregnum, disfluencies can be further categorized into three types: repetition (when RP is same as RM), repair, and restart (when RP is empty). Table 1 gives a few examples. However, it is usually difficult to identify the reparandum non-fluent factors. The main challenge behind is that this type of structure is flexible, variable in length, able to occur anywhere in a sentence, and in many cases will be nested. Detecting the disfluency can work in arbitrary form.

Recently, a number of approaches based on deep neural networks have been proposed to address the problem of disfluency detection under the framework of sequence labeling or sequence to sequence(Hough and Schlangen, 2015; Zayats et al., 2016; Wang et al., 2016; Wang et al., 2017). For most of the approaches mentioned above, the researchers switch between the RNN and CNN. But the RNN/CNN shows less flexibility than the self-attention layer which has achieved great success in neural machine translation (Shen et al., 2015; Wu et al., 2016; Gehring et al., 2017) and language understanding

*Wei Chen is the corresponding author of this paper

Proceedings of the 27th International Conference on Computational Linguistics, pages 3529–3538
Santa Fe, New Mexico, USA, August 20-26, 2018.

Type	Annotation
repair	[I just + I] enjoy work
repair	[we want + well in our area we want] to
repetition	[it's + {um} + it's] almost like
repetition	[the + {um} + the] decision was
restart	[by +]it was attached to
restart	[we would like +] let's go to the

Table 1: Different types of disfluencies.

(Shen et al., 2017). In the framework of sequence to sequence mapping, the model of NMT has achieved great success. Therefore, it is natural for us to investigate how to apply the NMT system to the problem of disfluency detection. In this paper, we take the disfluency detection as the translation task and build our model following the architecture of newly emerged NMT system, i.e., Transformer(Vaswani et al., 2017). The Transformer has achieved state-of-the-art results on both WMT2014 English-German and English-French translation tasks. Inspired by the success of self-attention, we propose a RNN/CNN-free network for disfluency detection through multi-task learning, which is built only on the self-attention layers. In the proposed model, we apply the self-attention layers to extract the hidden features of the input sentence. And two softmax layers, namely the label softmax and word softmax, are applied to perform classifications. The label softmax is used to calculate the probability of the disfluency label, and the word softmax is applied to compute the probability of the word.

In addition, we leverage unlabelled data to improve the performance of the disfluency detection. Due to lack of tagging information, using the unlabelled corpora is very promising, since the unlabelled corpora is usually easy to be collected. To utilize the unlabelled data, we extend the traditional encoder-decoder model by leveraging two independent encoders but with some layers shared. Specifically, we share the weights of the last few layers of two encoders that are responsible for extracting high-level representations of input sentences. In the proposed model, one encoder is utilized to encode the unlabelled data and the other is applied for the labelled data. For the unlabelled data, the corresponding encoder and decoder perform an Auto-Encoder (AE), where the encoder generates the latent representations from the perturbed input sentences and the decoder reconstructs the sentences from the latent representations. We utilize the GAN to constrain the latent representations from the labelled and unlabelled corpus to subject to a similar distribution, whereby the encoders try to fool a discriminator which is simultaneously trained to distinguish whether the latent representation is from labelled or unlabelled data.

In summary, we mainly make the following contributions:

- We propose a novel semi-supervised approach for the problem of disfluency detection, which can utilize large amounts of unlabelled data.
- We firstly introduce the self-attention mechanism into the disfluency detection through multi-task learning . Without relying on the RNN/CNN, the proposed model has more flexibility in sequence length and allows for more parallelization, which makes great significance.
- We conduct extensive experiments to test the proposed model. Experimental results show that the proposed approach consistently achieves great success.

2 Related Work

One of the most common approaches to disfluency detection is taking the task as a sequence labeling problem, where each sentential word is assigned with a label. Many approaches had achieved good performance by using the Condition-based Random Field (CRF) as a classifier (Ostendorf and Hahn, 2013; Liu et al., 2006; Cho et al., 2013; Zayats et al., 2014). (Qian and Liu, 2013) proposed a detection model based on maximum interval Markov random field (M^3N), which outperformed CRF by using an f-score matching objective function. (Ferguson et al., 2015) proposed the Semi-Markov CRF model for

disfluency detection and achieved the best performance of linear statistical sequence labeling methods by integrating prosodic features. Sequence labeling methods mentioned above are not powerful enough to model long-range dependencies of complicated disfluencies. RNN has been widely used to disfluency detection (Zayats et al., 2016; Hough and Schlangen, 2015), since it can capture dependencies at any length in theory. However, these RNN methods still suffer from the defect of unable to model the linguistic structural integrity, since it does not model the transition between tags which is important in recognizing the repair phrases of multi-words. Recently, sequence-to-sequence methods based on deep neural networks are applied to disfluency detection. (Wang et al., 2016) proposed an attention-based detection model and achieved good performance. It can capture a global representation of the input sentence by RNN when encoding and model tag transition when decoding.

Another branch of researches adopt transition-based parsing model, which jointly perform dependency parsing and disfluency detection (Rasooli and Tetreault, 2013; Honnibal and Johnson, 2014; Wu et al., 2015). These syntax-based models are capable of capturing long-range dependencies by modeling the repair phrases on a syntax tree. However, their high requirements for training data reduce their practicality, since it is expensive to obtain training data containing both syntax trees and disfluency annotations. Furthermore, the performance of syntax parsing also restricts the performance of disfluency detection. (Wang et al., 2017) proposed a transition-based model for disfluency detection, which learned representation of both chunks and global contexts without using any syntax information. It achieved the state-of-the-art f-score on the commonly used English Switchboard test set.

Lack of large scale of labelled data is always a bottleneck for improving the performance of disfluency detection (Wang et al., 2016; Wang et al., 2017). Recently, the unsupervised and semi-supervised NMT have arose many interests in the research area. To improve the translation performance of NMT, they use the auto-encoder and back-translation to train the model(Saha et al., 2016; Cheng et al.,). Following the same idea, we firstly propose the semi-supervised training for disfluency detection.

3 The Approach

In this section, we will give more details about the design of our proposed semi-supervised disfluency detection model.

3.1 The Input and Output

Since we consider the problem of disfluency detection as a translation task, translating the non-fluent sequence into the fluent sequence, we prepare our input and output as the training examples of translation. In traditional translation, the training example consists of one source-side sequence and one target-side sequence. However, for the proposed model, the source-side input is the non-fluent sequence, the target side includes two output sequences, one is the fluent sequence, which is the real needed output of disfluency detection task, and the other is the label sequence, which is an auxiliary information sequence. Considering a sentence from the training data, the non-fluent input of source side is:

”I want to flight to Boston um to Denver”

The target-side fluent sequence is represented as:

”I want to flight E E E to Denver ”

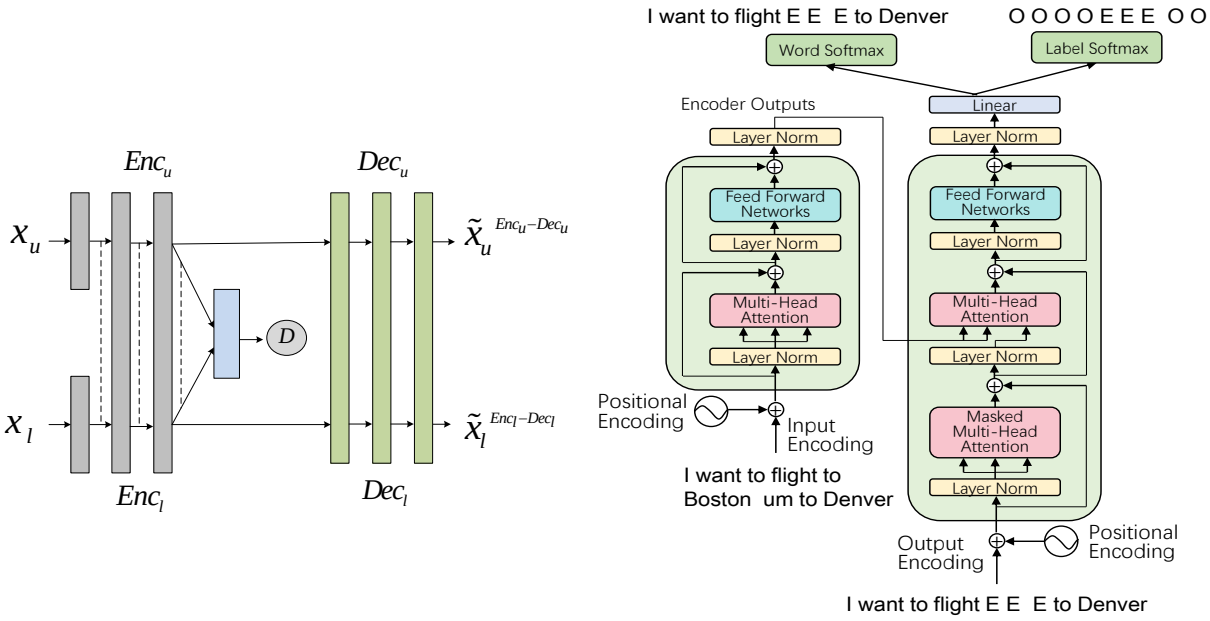
And the target-side label sequence can be represented as:

”O O O O E E E O O”

Where ’O’ indicates that the word at this time step is fluent, which should be copied from the source side to the fluent sequence, and ’E’ means the word at this time step is non-fluent, i.e., RM and IM words, which should be discarded. The length of the label sequence is the same as the input sequence. The motivation of the design of two output sequences will be elaborated in section 3.2.

3.2 The Model Architecture

Our proposed semi-supervised model is extended on traditional encoder-decoder framework. As illustrated in Figure 2(a), the model architecture is composed of four subnetworks: including two partially



(a) Weight sharing model with GAN

(b) Multi-task Transformer model

Figure 2: The framework of the proposed model. (a) is the whole architecture of our model, which contains two independent encoders with some weight sharing and the fully-shared decoder. (b) is the specific architecture of the proposed model which extends the Transformer into multi-task learning setting.

shared encoders Enc_l and Enc_u, one totally shared decoder Dec_l (Dec_u is the same as Dec_l), and a discriminator D, where subscript 'l' and 'u' represent labelled and unlabelled data, respectively. Each encoder consists of three identical layers, where each layer consists of a multi-head self-attention and a simple position-wise fully connected feed-forward network, which follows the powerful self-attention based model Transformer (Vaswani et al., 2017). Specifically, the encoder is composed of a stack of three identical layers. Symmetrically, the decoder is also a stack of three identical layers, and each layer also follows the Transformer. The key difference is the last output layer of the decoder, where we use two softmax layers to predict label sequence and word sequence separately (see Figure 2(b)). The decoding is constrained by the label prediction and word prediction at the same time, which is critical important for improving the disfluency detection performance.

To utilize unlabelled data x_u for training, the output of Enc_u should have similar distribution with the output of Enc_l. Therefore, we apply the weight sharing constraint to associate the two encoders by sharing the weights of the last few layers of two encoders that are responsible for extracting high-level representations of input sequences. Afterwards, a discriminator, implemented as a multilayer perceptron, is used to further constrain the latent representations from the labelled and unlabelled data to have similar distribution, whereby the encoder tries to fool the discriminator which is simultaneously trained to distinguish whether the latent representation is from the labelled or unlabelled data. As a matter of fact, the encoders and the discriminator constitute a GAN tactfully, where the encoders act as the cheater and the discriminator plays the role of police. The weight sharing and GAN training are pivotal strategies to make the semi-supervised training manner feasible.

To give more detailed interpretation of the weight sharing model, we piece together several roles by combining some subnetworks of the model, which are summarized in table 2. If only consider using labelled data x_l for training, it is enough to compose a complete model only with the encoder Enc_l and the decoder Dec_l, which will be back to a self-attention based Transformer model with multi-task learning. As for the unlabelled data, the encoder Enc_u generates the latent representation of the perturbed input sentence (such as adding random noises). And the corresponding decoder Dec_u reconstructs the

normal input sentence from the latent representation. Actually, Enc_u and Dec_u constitute an AE. The key information of these roles will be described in the following subsections successively.

Networks	Roles
$\{Enc_l, Dec_l\}$	Transformer for labelled data
$\{Enc_u, Dec_u\}$	AE for unlabelled data
$\{Enc_l, Enc_u\}$	Weight sharing of labelled and unlabelled data
$\{Enc_l, Enc_u, D\}$	GAN for discriminating data reality

Table 2: Interpretation of the roles for the subnetworks in the weight sharing model.

3.2.1 Multi-task Learning and Constrained Decoding

Our improvement on the Transformer model focuses on the last output layer of the decoder. Traditional output layer of Transformer is only one softmax layer. However, either the word sequence or the label sequence is incapable of handling the problem of disfluency detection independently. For the word sequence, it guides the model to treat each fluent word and disfluency word equally. However, in disfluency detection task, the model just needs to distinguish the non-fluent words from the fluent words in sentence. Too careful classification will detriment the classification performance. For the label sequence, it guides the model to output the fluent label or the non-fluent label. Nevertheless, it may cause overfitting during training since the fluent label 'O' and non-fluent label 'E' dominate in the training example. In other words, the word sequence needs a simple classification as the label sequence does, and the label sequence can benefit from the semantic information of the word sequence. Therefore, we fuse the disfluency detection sequence and the label sequence through multi-task learning to improve the performance.

Multi-task Learning

Given N such non-fluent source word sequences, target word sequences and label sequences. Our goal is to maximize the probability of observing the target word sequence y_n and minimize the label classification error rate of the label softmax, with regard to the model parameter θ. Different from the traditional end-to-end model in disfluency detection which only maximizes the probability of observing the target label sequence, the word error rate of the word softmax is also leveraged in the proposed model. From the multi-goal learning perspective, two different goals are used to dictate the parameters of the proposed model to converge to a better region. Formally, the objective function used in the joint punctuation model can be represented as:

$$\underset{\theta}{\operatorname{argmax}} \frac{1}{N} \sum_{n=1}^{N} (log_{p_\theta}(y_n|x_n) + \gamma * log_{p'_\theta}(z_n|x_n)) \tag{1}$$

Apart from the cross entropy loss used in the label neural network model, the error rate of the word neural network is also considered to train the proposed model.

$$J = J_w + \gamma * J_l \tag{2}$$

here J_w refers to the loss of the word, J_l denotes the loss of the label and γ is the hyper-parameter. We use the γ to balance the loss between the word softmax and label softmax.

Constrained Decoding

In the test decoding, we design a constrained decoding algorithm to adapt the proposed self-attention based network. In this algorithm, we need to judge the class of the label softmax to determine the decode sequence of the model. Specifically, if the label softmax outputs 'O', we copy the word from the source side and feed it into the model for the next time step. And if the output of the label softmax is 'E', we feed the generated 'E' into the model for the decoding in the next time step. The decoding process repeats until the "eos", i.e., the end of sentence token, is emerged and all the source words are copied. The constrained decoding algorithm could effectively avoid the inconsistency between the source word and the target word in a transduction model.

3.2.2 Denoising Auto-Encoding

In order to learn some knowledge from the unlabelled corpus to make up the scarcity of labelled data, an AE is utilized to train the unlabelled data via reconstructing their inputs. Specifically, the encoder is responsible for composing the high-level latent representation of the input and the decoder endeavors to decompose this representation into its corresponding input sequence. However, the AE usually fails to capture any internal structure of the involved sentences under unconstrained condition. Instead it just learns to copy every word one by one. To solve this problem, we employ the similar strategy of denoising AE (Vincent et al., 2008) and randomly add some noises to the input sentences (Hill et al., 2016; Artetxe et al., 2017).

To this end, for each sentence, we add two types of noises separately, one is repeating, the other is inserting. In repeating operation, we randomly choose one to three positions in sentence. And for each position, one to five words starting from selected position are repeated randomly, which is abide by the constraint that the repeat fragment does not overlap with the next selected position. In inserting operation, we still randomly choose one to three positions in sentence to add noises. Different from repeating operation which simply repeats words in the sentence, we insert words randomly selected from a top frequent dictionary, which is extracted from the corpus by keeping the top 10,000 most frequent 1-grams to 5-grams respectively beforehand. Thus, inserting operation can introduce more complicated noises. In this way, the system needs to learn some useful structure of the involved languages to be able to recover the correct word order.

3.2.3 Weight Sharing

To decrease even eliminate the distribution differences between the labelled and unlabelled corpora in some ways, a weight sharing constraint is applied to the two encoders according to the shared-latent space assumption. Different from (Cheng et al., 2016; Saha et al., 2016) which adopt the fully shared encoder, we share only partial weights for the encoders, considering that 1) The independent weights are devoted to encode the hidden features about the internal characteristics of each corpus, such as the terminology, style, and sentence structure; 2) The shared weights are concentrated on projecting those hidden features into the shared-latent space. Therefore, we share the weights of the last few layers of encoders Enc_l and Enc_u, which are committed to capture the high-level representations of the input sentences. A totally shared decoder is applied to decode those high-level representations that are vital for reconstructing the fluent sentences, since both for the labelled and unlabelled input sentences, they share the same decoding space and target.

3.2.4 Discriminator

For the weight sharing constraint based on the shared-latent space assumption, we would expect that the corresponding sentences in the labelled or unlabelled corpus will have the similar latent representations. However, the weight sharing constraint alone does not necessarily guarantee the same latent representations. To further enforce the shared-latent space, we train a discriminative neural network to classify whether the encoding representation is from the auto-encoded unlabelled or the labelled sentences.

The discriminator is implemented as a multi-layer perception with two hidden layers of size 256. The discriminator performs as a binary classifier and predicts whether the input is the labelled or unlabelled data.Given the encoding vector Enc(x), i.e., the output of the encoders either from the Enc_u or Enc_l, we build the perception layer as:

$$x_c = \rho(BN(W \times Enc(x) + b)) \tag{3}$$

where W and b is the learning parameters. ρ is a non-linear activation function which is implemented as ReLU. At last, we pass x_c into a softmax layer to generate the probability $p(f|Enc(x))$ as:

$$p(f|Enc(x)) = softmax(V * x_c) \tag{4}$$

where V is the transformation matrix and f is the data class, satisfying $f \in \{l, u\}$, where l means labelled data and u means unlabelled data. The discriminator is trained to predict the label probability

by minimizing the following cross-entropy loss:

$$L_D(\theta_D) = -\mathbb{E}_{x \in x_u}[\log p(f = u|Enc_u(x))] - \mathbb{E}_{x \in x_l}[\log p(f = l|Enc_l(x))] \quad (5)$$

here θ_D represents the discriminator parameters. The encoders are trained to fool the discriminator:

$$L_{Enc_u}(\theta_{Enc_u}) = -\mathbb{E}_{x \in x_u}[\log p(f = l|Enc_u(x))] \quad (6)$$

$$L_{Enc_l}(\theta_{Enc_l}) = -\mathbb{E}_{x \in x_l}[\log p(f = u|Enc_l(x))] \quad (7)$$

where θ_{Enc_u} and θ_{Enc_l} are the parameters of the two encoders.

4 Experiments

4.1 Datasets

To evaluate the effectiveness of utilizing unlabelled corpus on disfluency detection task, we firstly conduct an experiment solely on a labelled corpus named Switchboard. Then we expand the training data by introducing large-scale unlabelled corpus.

Switchboard is the largest available corpus for disfluency detection task, which is portion of the English Penn Treebank. Two annotation layers are provided in Switchboard corpus: one for syntactic bracketing (MRG files), the other for disfluencies (DPS files). Our proposed method only needs the DPS files. The disfluency annotation style of DPS files has been described in introduction. In our experiments, the word labelled with RM and IM would be tagged with 'E' and others would be tagged with 'O'. Some previous work has been done on this corpus. To keep the direct comparison possible, we follow the experiment settings in (Johnson and Charniak, 2004), taking directory 2 and directory 3 in subcorpus of PARSED/MRG/SWBD as training set, and splitting directory 4 into test and development sets. Table 3 shows the detailed scale of each dataset, where "total" means the total number of sentences and "non-fluent" means the number of non-fluent sentences among total.

Unlabelled corpus is randomly extracted from the source side of WMT2014 English-German corpus, consisting of English news. Noises, which make the unlabelled fluent sentence non-fluent, are added to the unlabelled datasets (named as WMT2014 English corpus) according to the description in subsection of denoising auto-encoding. The scale of training, development and test sets of WMT2014 English corpus is shown in Table 4.

Dataset	Total	Non-fluent
Training set	261,882	87,701
Dev set	2,000	1,455
Test set	2,000	1,425

Table 3: Scale of Switchboard corpus.

Dataset	Total	Non-fluent
Training set	1,000,000	1,000,000
Dev set	2,000	1,500
Test set	2,000	1,500

Table 4: Scale of WMT2014 English corpus.

4.2 Training Details

BPE Byte Pair Encoding (Sennrich et al., 2015) is a simple data compression technique, which is highly efficient to reduce the OOV quantity by iteratively replacing the most frequent pair of bytes in a sequence with a single unused byte. In our experiments, we set source number operations to 20,000 and the vocabulary size to 30,000.

Hyper-parameter The configuration of the hyper-parameter is critical to the performance of the proposed self-attention based model, since it directly affects the generalization and regression of the model. To balance the whole information from the word sequence and the label sequence, we introduce γ parameter. The hyper-parameter γ is designed to balance the loss from the label sequence and the word sequence. In order to highlight the word sequence loss, we set the γ value to 0.15 to weaken the influence of label sequence .

Metric Following previous works (Wang et al., 2016), token-based precision (P), recall (R), and F-score (F1) are used as the evaluation metrics.

Sharing layer	Switchboard			WMT2014		
	P	R	F1	P	R	F1
0	89.2	87.9	88.5	68.5	59.5	63.7
1	84.2	78.9	81.4	94.5	96.4	95.4
2	92.1	90.2	91.1	93.2	95.9	94.5
3	90.7	87.8	89.2	95.3	96.9	96.1

Table 5: Performance on Switchboard and unlabelled WMT2014 test set with different sharing layers.

4.3 Number of Weight-Sharing Layers

To investigate how the number of weight-sharing layers affects the translation performance, we vary the number of weight-sharing layers of the two encoders from 0 to 3. Table 5 shows the experimental results. It is observed that the number of weight-sharing layers has a significant impact on the performance of disfluency detection. When we set the number of weight-sharing layers to 0, resulting in two independent encoders, we get pool F-score on both test sets, especially on the WMT2014 test set. This is mainly because in testing phase, the unlabelled data will be encoded by encoder Enc_l and decoded by the totally shared decoder, while Enc_l could not encode the unlabelled data well under this condition since the encoder Enc_l could not learn any knowledge from the unlabeled data in training phase. Furthermore, the decoder needs to balance the two different kinds of input data, which leads to the small decline of performance on Switchboard corpus as well. From another perspective, it also indicates that without the weight sharing constraint, the GAN alone is insufficient to enforce the labelled data and unlabelled data to learn the similar latent representations, which results in a negative role. This confirms our intuition that the shared layers are crucial to project the labelled and unlabelled latent representations into a shared-latent space. At the opposite extreme, when all three layers of the two encoders are shared, resulting in a totally shared encoder, it is equal to feeding labelled and unlabelled data into a multi-task Transformer directly. In spite of a larger amount of training corpus, the F-score is better than the one of two independent encoders but still 1.8% lower than the one of two weight-sharing layers on Switchboard corpus. This verifies our conjecture that unlabelled data can enhance the generalization ability of model, but the shared encoder may also detrimental to the performance since it will weaken the unique characteristic of different corpora. Choosing proper number of weight-sharing layers makes much difference. We achieve best F-score of 91.1% on the commonly used Switchboard corpus test set when the two encoders share the last two layers. Under this condition, model can learn both the general characteristic and the internal characteristics such as the terminology and sentence structure of each corpus well.

Method	P	R	F1
Multi-task Transformer	91.5	87.1	89.2
Weight sharing	92.1	90.2	91.1
Transition-based (Wang et al., 2017)	91.1	84.1	87.5
Attention-based (Wang et al., 2016)	91.6	82.3	86.7
Bi-LSTM (Zayats et al., 2016)	91.6	80.3	85.9
semi-CRF (Ferguson et al., 2015)	90.0	81.2	85.4
UBT (Wu et al., 2015)	90.3	80.5	85.1
M^3N (Qian and Liu, 2013)	-	-	84.1

Table 6: Comparison with previous approaches on the test set of English Switchboard.

4.4 Results and Analysis

We compare our self-attention based models with six previous top performing systems. Table 6 shows the comparable results on English Switchboard test set. Our first model, named *multi-task Transformer*,

only utilizes the Switchboard corpus for training as other baselines. It is designed to assess the performance of self-attention based model directly. As it shown in Table 6, our multi-task Transformer achieves 89.2% F-score, outperforming the Transition-based method (Wang et al., 2017), which is the state-of-the-art, by 1.7 point improvements. Compared with the semi-CRF method (Ferguson et al., 2015), which gains the best performance of linear statistical sequence labeling methods, our multi-task Transformer achieves 3.8 point improvements without leveraging any prosodic features. The performance gab between our multi-task Transformer and UBT (Wu et al., 2015), the best syntax-based method for disfluency detection, is even larger. The F-score of multi-task Transformer increases by 5.1% than UBT. Furthermore, our second model focuses on how to take the advantage of unlabelled data, since labelled data is usually scarce. A weight sharing model (named *weight sharing* in Table 6) is introduced to tackle the training problem of mixed data. It improves the best F-score of our self-attention based models from 89.2% to 91.1%, which indicates that our novel proposed weight sharing model is capable of learning useful language knowledge from unlabelled noisy dataset automatically. As a result, our weight sharing semi-supervised model achieves 3.6% improvements than the previous state-of-the-art.

5 Conclusion

In this paper, we propose a novel semi-supervised model based on weight sharing mechanism and self-attention based multi-task learning scheme, which views the disfluency detection as a translation task. In the multi-task self-attention based network, the word sequence information and labelling information are incorporated into the model at the same time, and a constrained decoding method in testing phase is applied. Experimental results show that the proposed model achieves significant improvements than the strong baseline models. We are among the first endeavors to use the translation model to handle the disfluency detection task, which can be applied to any other sequence labelling task easily. In addition, we utilize the unlabelled corpus to enhance the performance by introducing the weight sharing strategy and the generative adversarial training to enforce the similar distribution between the labelled and unlabelled data. Experimental results show that the semi-supervised model can further improve the performance significantly.

Acknowledgements

The research work is supported by the National Key Research and Development Program of China under Grant No. 2017YFB1002102 and the NSFC project 61702514.

References

[Artetxe et al.2017] Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

[Cheng et al.] Yong Cheng, Qian Yang, Yang Liu, Maosong Sun, Wei Xu, Yong Cheng, Qian Yang, Yang Liu, Maosong Sun, and Wei Xu. Joint training for pivot-based neural machine translation. In *Twenty-Sixth International Joint Conference on Artificial Intelligence*.

[Cheng et al.2016] Yong Cheng, Yang Liu, Qian Yang, Maosong Sun, and Wei Xu. 2016. Neural machine translation with pivot languages. *arXiv preprint arXiv:1611.04928*.

[Cho et al.2013] Eunah Cho, Thanh-Le Ha, and Alex Waibel. 2013. Crf-based disfluency detection using semantic features for german to english spoken language translation. *IWSLT, Heidelberg, Germany*.

[Ferguson et al.2015] James Ferguson, Greg Durrett, and Klein Dan. 2015. Disfluency detection with a semi-markov model and prosodic features. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 257–262.

[Gehring et al.2017] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

[Hill et al.2016] Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483*.

[Honnibal and Johnson2014] Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association of Computational Linguistics*, 2(1):131–142.

[Hough and Schlangen2015] Julian Hough and David Schlangen. 2015. Recurrent neural networks for incremental disfluency detection. *Interspeech 2015*.

[Johnson and Charniak2004] Mark Johnson and Eugene Charniak. 2004. A tag-based noisy channel model of speech repairs. In *Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain*, pages 33–39.

[Liu et al.2006] Yang Liu, Elizabeth Shriberg, et al. 2006. Enriching speech recognition with automatic detection of sentence boundaries and disfluencies. *IEEE Transactions on audio, speech, and language processing*.

[Ostendorf and Hahn2013] Mari Ostendorf and Sangyun Hahn. 2013. A sequential repetition model for improved disfluency detection. In *INTERSPEECH*, pages 2624–2628.

[Qian and Liu2013] Xian Qian and Yang Liu. 2013. Disfluency detection using multi-step stacked learning. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 820–825.

[Rasooli and Tetreault2013] Mohammad Sadegh Rasooli and Joel Tetreault. 2013. Joint parsing and disfluency detection in linear time. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 124–129.

[Saha et al.2016] Amrita Saha, Mitesh M Khapra, Sarath Chandar, Janarthanan Rajendran, and Kyunghyun Cho. 2016. A correlational encoder decoder architecture for pivot based sequence generation. *arXiv preprint arXiv:1606.04754*.

[Sennrich et al.2015] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

[Shen et al.2015] Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2015. Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433*.

[Shen et al.2017] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *arXiv preprint arXiv:1709.04696*.

[Shriberg1994] Elizabeth Ellen Shriberg. 1994. *Preliminaries to a theory of speech disfluencies*. Ph.D. thesis, Citeseer.

[Vaswani et al.2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

[Vincent et al.2008] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

[Wang et al.2016] Shaolei Wang, Wanxiang Che, and Ting Liu. 2016. A neural attention model for disfluency detection. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 278–287.

[Wang et al.2017] Shaolei Wang, Wanxiang Che, Yue Zhang, Meishan Zhang, and Ting Liu. 2017. Transition-based disfluency detection using lstms. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2785–2794.

[Wu et al.2015] Shuangzhi Wu, Dongdong Zhang, Ming Zhou, and Tiejun Zhao. 2015. Efficient disfluency detection with transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 495–503.

[Wu et al.2016] Yonghui Wu, Mike Schuster, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

[Zayats et al.2014] Victoria Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2014. Multi-domain disfluency and repair detection. In *Fifteenth Annual Conference of the International Speech Communication Association*.

[Zayats et al.2016] Vicky Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2016. Disfluency detection using a bidirectional lstm. *arXiv preprint arXiv:1604.03209*.

ISO-Standard Domain-Independent Dialogue Act Tagging for Conversational Agents

Stefano Mezza, Alessandra Cervone, Giuliano Tortoreto,
Evgeny A. Stepanov, Giuseppe Riccardi
Signals and Interactive Systems Lab, University of Trento, Italy
name.surname@unitn.it

Abstract

Dialogue Act (DA) tagging is crucial for spoken language understanding systems, as it provides a general representation of speakers' intents, not bound to a particular dialogue system. Unfortunately, publicly available data sets with DA annotation are all based on different annotation schemes and thus incompatible with each other. Moreover, their schemes often do not cover all aspects necessary for open-domain human-machine interaction. In this paper, we propose a methodology to map several publicly available corpora to a subset of the ISO standard, in order to create a large task-independent training corpus for DA classification. We show the feasibility of using this corpus to train a domain-independent DA tagger testing it on out-of-domain conversational data, and argue the importance of training on multiple corpora to achieve robustness across different DA categories.

1 Introduction

The correct interpretation of the intents behind a speaker's utterances plays an essential role in determining the success of a dialogue. Hence, the module responsible for intents classification lies at the very core of many dialogue systems, both in research and industry (e.g. Alexa, Siri). Moreover, although the task of intent recognition is traditionally linked to task-based systems, recently it has also proved crucial for non task-based conversational systems. According to the results of the Amazon Alexa Prize challenge (Ram et al., 2017), the most successful conversational systems in the competition relied on a strong spoken language understanding module, while more than 60% of the approaches explicitly used intents.

Nevertheless, automatic intent recognition is hard, since participants' intents in a dialogue are implicit. Intent classification has therefore been mostly modeled as a supervised machine learning problem (Gupta et al., 2006; Xu and Sarikaya, 2013; Yang et al., 2017), with the consequent definition of intents taxonomies. Over time this led to the creation of expensive annotated resources (Price, 1990; Henderson et al., 2014) with the related time-consuming design of multiple intent schemes. In most cases, however, intents taxonomies are defined specifically for a given application or a dataset and are not generalizable to other systems or tasks, making these resources difficult to reuse (e.g. the popular Air Travel Information Services (ATIS) corpus include heavily domain-dependent intents such as *Airfare* or *Ground Service*).

Dialogue Acts (DA), also known as speech or communicative acts, represent an attempt to create a formalized and generalized version of intents. As such, DAs have been investigated by the research community for many years (Stolcke et al., 2000) and have been applied successfully to many tasks. In particular, their aspiration to generality makes them an appealing option for non task-based application (e.g. more than 20% of the teams in Amazon Alexa Prize Challenge explicitly used DAs (Cervone et al., 2017; Bowden et al., 2017), including the winning team (Fang et al., 2017)). Also, in the case of DAs, over the years there have been several efforts to produce publicly available annotated resources (Godfrey et al., 1992; Carletta, 2006; Alexandersson et al., 1998) to train DA taggers. The DA taxonomies created

Proceedings of the 27th International Conference on Computational Linguistics, pages 3539–3551
Santa Fe, New Mexico, USA, August 20-26, 2018.

for these resources, albeit arguably more general compared to corpora like ATIS (for example utilizing categories such as *wh-questions*), are still dataset specific; since many of these schemes lack coverage of some crucial aspects of dialogic interaction. Furthermore, given that all these datasets utilize different schemes, these resources are hardly compatible.

The ISO 24617-2 (Bunt et al., 2010; Bunt et al., 2012), the international ISO standard for DA annotation, represents the first attempt to create a truly domain and task independent scheme. Given its holistic approach compared to previous schemes, ISO 24617-2 can be used as a lingua franca for cross-corpora DA mapping, as confirmed by successful attempts to remap single corpora to the standard (Chowdhury et al., 2016; Fang et al., 2012).

However, there is no reference training set for the standard, since the only public resource currently available with ISO 24617-2 annotation (DialogBank, (Bunt et al., 2016)) is too small to be used to train classifiers. Therefore, most DA tagging research still focuses on in-domain studies on large datasets with incompatible DA annotations (Stolcke et al., 2000; Ji et al., 2016). Moreover, most publicly available corpora are imbalanced with respect to the distribution of various DA dimensions such as *Information Transfer* (e.g. "What's your favourite book?") or *Action Discussion* (e.g. "Tell me the news."), which are required for successful open-domain conversational systems.

In this work, we show how to reuse and combine publicly available annotated resources to create a large training corpus for domain-independent DA tagging experiments. We map different corpora using an ISO standard compliant DA taxonomy, following the previous research on the topic (Fang et al., 2012; Petukhova et al., 2014b) and we share this resource with the research community.[1]

In order to investigate the soundness of our approach compared to in-domain models we further experiment with domain-independent DA tagging. As previously done in the literature we cast the Dialogue Act tagging task as a supervised multi-class classification problem using Support Vector Machines. The correctness of the approach is first tested on the de facto DA tagging standard – the Switchboard (SWDA) corpus (Godfrey et al., 1992), using the reference training and test sets and achieving performance comparable to the state-of-the-art approaches. Secondly, we experiment with domain-independent DA tagging following the same approach and using our combined resource as a training corpus. The DialogBank corpus, that represents a reference manual DA annotation for the ISO standard, is used for the evaluation of the tagger. To the best of our knowledge this is the first attempt to test automatic DA annotation on this corpus.

The domain-independence and suitability of the tagger for conversational systems trained on multiple resources is additionally evaluated on two other corpora annotated following our optimized taxonomy (human-machine conversations from the Amazon Alexa Prize Challenge). The performances achieved on these three datasets suggest that the training on multiple corpora represents a step forward for DA tagging of open-domain non task-based human-machine conversations. Finally, we present experiments to investigate the contribution of the different corpora to the performance of the classifiers. The results of our experiments show the importance of utilizing multiple resources to achieve a sound performance across different types of DA categories. The multi-domain DA tagger presented here was successfully employed in Roving Mind, our open-domain conversational system for the Alexa Prize (Cervone et al., 2017).

2 State of the Art

2.1 Dialogue Act Annotation Schemes

The notion of Dialogue Acts can be traced back to the one of illocutionary acts introduced by (Austin and Urmson, 1962). The illocutionary act represents a level of description of an utterance's meaning that goes beyond the purely semantic level ("Is the window open?") to encompass the intent of the speaker in producing that utterance ("Please, close the window.").

One of the first DA taxonomies was the one created for the task-based corpus MapTask (Anderson et al., 1991) in the early nineties. The MapTask scheme distinguishes between *initiating moves* – such

[1]The suite of scripts we wrote to map and combine publicly available corpora can be found at `https://github.com/ColingPaper2018/DialogueAct-Tagger`

as giving instructions, explaining, checking information or asking questions – and *response moves* – for example acknowledging instructions, answering questions and clarifying information. The corpus also makes a distinction regarding the grammatical and semantic structure of the interactions, classifying, for example *wh-questions*, *yn-questions* and *positive/negative answers*. Although pioneering at the time, the MapTask annotation scheme is very specific to the described scenario, and some of its DAs (e.g. *instruct*, *clarify*, *check*) do not scale well to generic, non task-based conversations. Moreover, its taxonomy was not designed to capture all human behaviours during conversations, and, as a consequence, its coverage for labelling a non task-based interaction is inadequate.

The first attempt to define a unified, non task-based standard for DA tagging was the Discourse Annotation and Markup System of Labeling (DAMSL) (Core and Allen, 1997) tag-set for the SWDA (Godfrey et al., 1992) corpus. This annotation scheme proposes a taxonomy of 42 tags, describing both semantic aspects of conversation (*opinion*, *non-opinion*, *preference*, etc.), syntactic aspects (*yn-questions*, *wh-questions*, *declarative questions*, etc.) and behaviours related to the dialogue (*conventional closing*, *hedge*, *backchanneling*, etc.). Nevertheless, the taxonomy still has some issues: tags are mutually exclusive (making it impossible to annotate, for example, a no answer which was also signaling non-understanding) and are organised in a flat taxonomy, which does not take into account similarities and differences between the tags.

Bunt (1999) introduced the Dynamic Interpretation Theory (DIT) for dialogues, setting the theoretical foundation for a domain-independent and task-independent DA taxonomy. The paper introduced some very important concepts like the idea of *multidimensionality* of DAs and the distinction between *Action-Discussion*, a macro-category of DAs encompassing cases in which interlocutors negotiate actions to be performed (e.g. requests like "Let's switch topic."), and *Information-Transfer* interactions, capturing the DAs through which speakers exchange information (e.g. sharing personal information like "My name is John."). The DIT++ taxonomy (Bunt, 2009) was then defined in 2009 with the aim of providing a unique and universally recognized standard for DA annotation based on the theoretical ideas introduced in the DIT scheme. Its fifth version was accepted as ISO 24617-2.

The core aspects of the ISO standard are its multidimensionality and its domain and task independence. The ISO scheme is multidimensional in the sense that it makes a clear distinction between *semantic dimensions* (i.e. the aspect of the communication which the DA describes) and *communicative functions* (i.e. the illocutionary act performed within that dimension). In this way, ambiguities between various aspects of the communication and overlapping between DAs are removed. Furthermore, the scheme contains a generic dimension and communicative functions, which is suitable for mapping virtually any kind of conversation, both task-based and non task-based. Moreover, its multidimensional aspect and hierarchical taxonomy make it extensible and potentially adaptable to specific conversational sets.

2.2 Dialogue Act Tagging

The automatic recognition of Dialogue Acts has been addressed by the literature using various machine learning techniques. In particular DA classification has been modeled both as a sequence labeling problem, using techniques such as HMM (Stolcke et al., 2000), neural networks (Ji et al., 2016; Kumar et al., 2017) or CRF (Quarteroni et al., 2011), and as a multi-class classification problem, using for example SVM (Quarteroni and Riccardi, 2010). Mentioning and comparing all DA classification approaches is difficult because of the differences in annotation schemes and datasets used. All approaches, however, are usually tested on in-domain data.

One of the most popular datasets for benchmarking is SWDA (Godfrey et al., 1992), a dataset of human-human open-domain telephone conversations. The state-of-the-art on SWDA (77.0% accuracy on 42 DA tags) was achieved by (Ji et al., 2016) using deep neural networks.

The ISO standard (Bunt et al., 2010) can be seen as a generalization of all these annotation schemes. However, there is no available training data for the ISO standard.

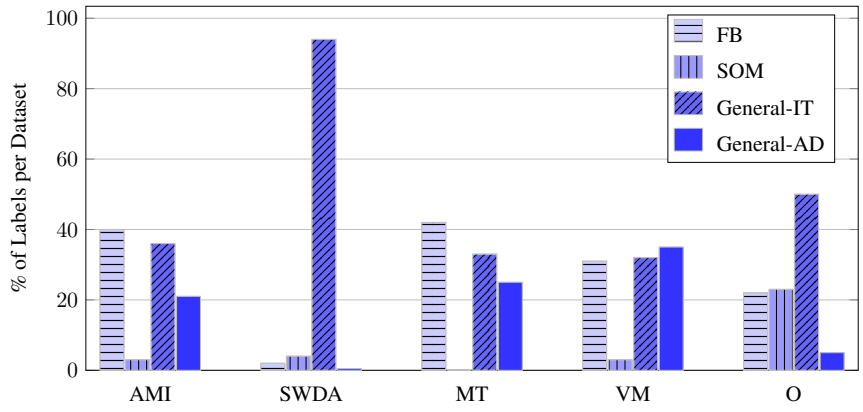


Figure 1: The distribution of dialogue act categories (after mapping to ISO standard) in various corpora – AMI, SWDA, MapTask (MT), VerbMobil (VM) and BT Oasis (O). The represented DA categories are Social Obligations Management (SOM) and Feedback dimension DAs, as well as *Action-Discussion* (AD) and *Information-Transfer* (IT) DAs from general dimension.

3 Data Sets

3.1 Training sets

The scarcity of resources of adequate size annotated with the ISO standard makes it difficult to train a DA tagger for this taxonomy. To the best of our knowledge, the DBOX corpus (Petukhova et al., 2014a) – the only resource manually annotated using the ISO standard – is not yet publicly available . The best possible approach given the current availability of data is to map existing corpora's DA schemes to the ISO scheme. Given the limited – and often domain-dependent – annotation scheme of these resources, it is impossible to map enough data to train a DA tagger for the full ISO taxonomy, since some of the ISO dialogue acts have no correspondence in any of the considered corpora. Therefore, we opted for a reduced version of the taxonomy, limiting our research to subsets of the General (Task), Social Obligation Management and Feedback dimensions. So far, we mapped the following five different corpora to our scheme:

SWDA: The Switchboard corpus (Godfrey et al., 1992) is a dataset of transcribed open-domain telephone conversations. The Switchboard Dialogue Act Corpus (SWDA) is a subset of the Switchboard corpus annotated with DAs. SWDA represents a logical choice when building a training set for a domain-independent DA tagger, as it is a large collection of open-domain, non task-based conversations, and therefore provides a natural similarity to the conversational domain of social bots. Moreover, there are already examples in literature of mappings from the Switchboard corpus to the ISO standard (Fang et al., 2012). As visible from Figure 3.1, drawbacks of the corpus with respect to the task include its unbalancedness (60% of utterances are *Information-providing*) and lack of *Action-Discussion* interactions (less than 1% of overall corpus).

AMI: This corpus contains transcriptions from 100 hours of meeting recordings of the European-funded AMI project (FP6-506811), a consortium dedicated to the research and development of technology (Carletta, 2006). This dataset presents a reasonably balanced collection of utterances and a taxonomy which shares some similarities with the ISO standard (e.g. distinction between *Action-discussion* and *Information-transfer*). Drawbacks of the corpus include the fact that there are multiple speakers (it is therefore more difficult to capture contextual information) and sometimes its scheme does not map to the leaves of the ISO tree.

MapTask: This is a task-based dialogue corpus collected by the HCRC at the University of Edinburgh (Anderson et al., 1991). Dialogues involve two participants, one with an empty map and one with a route-marked map which must instruct the other speaker to draw the same route. The corpus was chosen due to its abundance of *Action-discussion* interactions (more than 30% of the overall corpus), which are often lacking in other corpora.

VerbMobil: This is a collection of task-based dialogues released in 1997 (Burger et al., 2000). A subset of these dialogues is annotated with DAs (Alexandersson et al., 1998). The scenario involves two speakers, which play respectively the roles of a travel agent and of a client. The client usually provides a set of constraints and requests to be satisfied, while the traveling agent has to ask questions and provide information in order to satisfy the client's requests. Interactions happening within the VerbMobil 2 corpus closely resemble those usually seen with personal assistants, with a user looking for the fulfillment of a task and a serving agent interacting with the user to solve his/her issues making it an appealing addition to our training set.

BT Oasis: The BT Oasis corpus is a collection of task-based conversations involving personal assistance for clients of the British Telecom services (Leech and Weisser, 2003). The conversations are human-to-human, and usually involve a user who has a problem to solve and an assistant who helps the user solving his issues. The BT Oasis corpus was chosen as part of the training set for its interesting scheme, called SPAAC (Speech Act Annotation scheme for Corpora), which is easily mappable to the ISO standard due to its clear separation of grammatical and illocutionary act.

3.2 Test sets

DialogBank (DB): the DialogBank (Bunt et al., 2016) is a corpus[2] annotated with ISO 24617-2 which currently contains 15 English dialogues: 3 from MapTask and 3 from TRAINS (Traum, 1996) (both task-based), 5 from DBOX (games collected in a Wizard-of-Oz fashion) and 4 from Switchboard (open-domain human-human conversation). Overall there are 1,596 DAs. The corpus currently represents the only publicly available resource manually annotated using the ISO standard.

Common Alexa Prize Conversations (CAPC): The CAPC corpus (Ram et al., 2017) is a dataset of 3,764 anonymised individual user turns pooled from different users interacting with all socialbots participating in the Alexa Prize. We have extracted a balanced subset of 458 turns and have annotated it with DAs from our adapted version of the ISO standard by 3 annotators, with an inter-annotator agreement of $\kappa = 0.82$. CAPC exemplifies frequent user interaction data not biased by the interaction with one socialbot in particular. Another advantage of this dataset is that it is balanced across different DA categories. One drawback is that no interaction context (previous DA) is available for the individual turns.

Socialbot Logs (S-Logs): S-Logs is a dataset of 13 open-domain conversations that different native American English speakers had with one of the socialbots of the Alexa Prize Challenge 2017. Overall this dataset contains 310 machine DAs and 165 user DAs. Two annotators tagged this dataset with DAs from our adapted version of ISO 24617-2, with an inter-annotator agreement of $\kappa = 0.81$. While we have annotated both machine and user turns, we test only on the latter and exploit machine turns as features for our classification experiments.

4 Methodology

4.1 Preprocessing

Before mapping the DA schemes of the corpora to the ISO subset scheme, a series of preprocessing steps have been performed to obtain a uniform training resource with the same surface text features as the testing corpora, since in S-Logs data, user input is lowercased and the punctuation is limited to apostrophes. More specifically, the text has been lowercased (including any named entity appearing in the original transcription and excluding the 'I' pronoun), punctuation has been removed (except for the apostrophe character in contracted expressions like "let's" and "can't") and any special characters have been deleted from the utterances. Moreover, any information regarding prosody has been removed, since this feature is not available in our test sets.

For experiments on the SWDA DAMSL corpus we recreated the same setting described in (Stolcke et al., 2000), using the same train and test set and preprocessing the corpus in the same way following the WS97 manual annotator guidelines (Jurafsky, 1997).

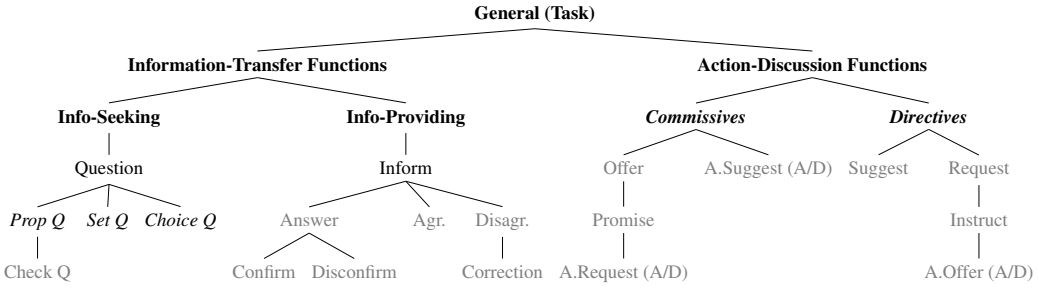


Figure 2: Communicative functions from ISO 24617-2 General purpose (Task) dimension. The nodes of the taxonomy that are not considered are grayed out.

S-scheme	**ISO 24617-2**
Social Obligation Management	
Salutation	Greeting, Goodbye, Self-Intro
Apology	Apology, Accept Apology
Thanking	Thanking, Accept Thanking
Feedback	
Feedback	Auto-Feedback (all), Allo-Feedback (all)

Table 1: Our scheme (S-scheme) compared to the corresponding ISO 24617-2 scheme for the SOM and Feedback dimensions

4.2 Dialogue Act scheme and mapping

The Socialbot scheme (S-scheme), the DA scheme used during the classification experiments, is a subset of the official ISO standard. Only three dimensions out of the official eight defined in the standard are considered (*Task*, *Social Obligation Management* and *Feedback*), and some of the communicative functions are generalized with an higher level of the tree.

Figure 4.2 shows the labeled subset of the ISO standard taxonomy for the General-purpose functions (i.e. functions independent from any given dimensions), while table 1 shows the correspondence for dimension-specific functions. The main difference between the DA scheme labeled in this work and the complete ISO taxonomy is the lack of further specification for the *Inform*, *Commissive* and *Directive* tags. This is due to the fact that most of the DA schemes used when building the training set do not provide contextual information detailed enough to label these tags accurately. Moreover, there is confusion and discrepancies about when these contextual DA should be used, even in the official ISO guidelines. Indeed, in (Fang et al., 2012), which provides the official mapping from Switchboard to the ISO standard, it is reported that some contextual DA tags (for example *other_answer*) do not have a direct mapping to the standard. This becomes even more problematic considering that among the training resources there are corpora like AMI, MapTask or VerbMobil, which label answers as Informs, which would make training data for this class extremely noisy. A similar argument can be raised on the lower leaves of the *Directive* and *Commissive* nodes, some of which are not labeled even in the very detailed SWDA taxonomy. Mapping of the available corpora to this scheme was done according to the available documentation in literature.

For the Switchboard corpus, a detailed mapping is provided in (Fang et al., 2012), which was followed exactly for the supported dimensions/communicative functions. For MapTask and AMI, there is already research highlighting similarities and differences between their schemes and the ISO standard one (Petukhova et al., 2014b). These results do not provide an exact mapping between the two schemes, which in some cases is impossible: for example the AMI *Elicit-inform* tag is the equivalent of ISO 24617-

[2] https://dialogbank.uvt.nl/

DA	SWDA	MapTask	VerbMobil	Oasis BT	AMI	DB	CAPC	S-Logs
Semantic Dimensions								
General (Task)	83,652	15,054	5,330	2587	1,523	1035	442	142
Social OM	2,866	0	384	588	10,039	21	16	7
Feedback	39,866	5,070	2,768	1,172	31,985	407	0	16
Total*	126,384	20,508	8,482	2,381	43,547	855	329	109
% of Corpus	79%	100%	72%	58%	74%	100%	100%	100%
General (Task) Dimension								
Commissives	63	-	7	25	1,523	57	20	1
Directives	7	4,075	2,911	181	10,039	131	93	32
Inform	75,667	4,860	-	1,648	33,403	652	105	91
Prop. Question	1,986	583	-	492	-	61	68	12
Set Question	5,506	1,692	-	241	-	134	149	6
Choice Question	423	-	-	-	-	8	7	-
Total*	83,652	11,210	2,918	2,587	44,965	1,035	442	142
% of Corpus	57%	30%	32%	35%	34%	N/A	N/A	N/A
Social Obligations Management								
Salutation	2,711	-	340	231	-	13	6	2
Apology	75	-	-	44	-	6	3	4
Thanking	80	-	44	193	-	2	7	1
Total*	2,866	0	384	468	2,201	21	16	7
% of Corpus	2%	0%	2%	8%	0%	N/A	N/A	N/A
Feedback								
Total	39,886	5,070	2,768	1,172	31,985	407	-	16
% of Corpus	79%	100%	72%	58%	74%	N/A	N/A	N/A

Table 2: Dialogue Act category counts across the considered corpora for different levels of the taxonomy. *Percentages of corpora* indicate the percentage of data available for the particular level in the corpus.
$*$ It is frequently the case that DA tags do not map to any leaf-node, e.g. VerbMobil for Task dimension and AMI for Social Obligations Management.

2's *Question*, but will not map to any specific question type (*SetQ*, *PropQ*, *ChoiceQ*, etc.). Utterances whose tags cannot be directly mapped to the ISO scheme were dropped and do not appear in the training set.

Since there is no available literature on mapping the VerbMobil 2 and BT Oasis corpora to the ISO standard, a specific mapping was designed from scratch by drawing inspiration from the approaches available on other corpora.

Table 2 presents counts of the DAs after mapping to our scheme, across all training and testing corpora. As mentioned in the paper, the corpora present quite imbalanced distributions of DA categories.

5 Experiments and Results

Since, to the best of our knowledge, there is no established state of the art on DialogBank – the only corpus manually annotated following the ISO 24617-2 scheme – we first establish the tagging methodology on the SWDA corpus using the DAMSL 42 tag set and compare it to the state of the art (Ji et al., 2016). Then, the feature set and the parameters of the best performing models are used for the training of the DA tagger on the aggregate dataset, considering some of the semantic dimensions and the communicative functions of the ISO 24617-2 . The models are then evaluated on the DialogBank and open-domain human-machine data from Amazon Alexa Prize Challenge. McNemar's test (McNemar, 1947) for statistical significance has been used to analyze whether introduced features give a significant contribution to the overall performance.

5.1 Experiments on SWDA

Prior to training the classification models, the SWDA (Jurafsky, 1997) utterances are preprocessed following (Stolcke et al., 2000). The dataset is split into training (1,115 dialogues) and test set (19 dialogues) following the same paper, and the remaining 21 dialogues are used as development set to tune the C parameter of Support Vector Machines (SVM) (Vapnik, 1995). For the experiments, we used the SVM

Features	Acc.	Features	Acc.
BL: Majority	31.5		
HMM (Stolcke et al., 2000)	71.0	1-2-grams + PREV	74.6 *
SVM (Quarteroni and Riccardi, 2010)	72.4	1-2-grams + PREV + POS	74.6
DrLM (LSTM) (Ji et al., 2016)	77.0	1-2-grams + PREV + I-POS	76.2 *
1-grams	71.2	1-2-grams + PREV + I-POS + DEP	76.0
1-2-grams	71.7	1-2-grams + PREV + I-POS + I-DEP	76.1
1-2-3-grams	71.4	1-2-grams + PREV + I-POS + WE	**76.7** *

Table 3: Classification accuracy of the different feature combinations on the SWDA test set. The best results are highlighted in bold. The results that are significantly better are marked with *.

implementation of scikit-learn (Pedregosa et al., 2011) with linear kernel (i.e. its *liblinear* (Fan et al., 2008) wrapper).

The results of the experiments on SWDA are presented in Table 3. The performances are on the SWDA test set with the SVM C parameter set to 0.1, with respect to the best results on the development set. It is worth mentioning that tuning the C parameter boosts the performance on the development set by 2 points.[3] For comparison, the table also includes majority baseline, the results from (Stolcke et al., 2000), the SVM results from (Quarteroni and Riccardi, 2010), and the state-of-the-art results from (Ji et al., 2016) that were achieved using deep learning methods.

Following the previous studies on SWDA (Stolcke et al., 2000; Quarteroni and Riccardi, 2010), we experiment with n-grams (unigrams, bigrams, and trigrams) and previous DA tag features. We do not consider the unit length feature from (Quarteroni and Riccardi, 2010), since classification instances in the SWDA scheme and ISO 24617-2 are different (slash unit vs. functional unit). The results are reported in Table 3; since the results reported were obtained with SVM C parameter set to 0.1, they are higher than the ones reported in (Quarteroni and Riccardi, 2010): e.g. for 1-2-grams 70.0 vs. 71.7.

The first observation is that the addition of the previous DA significantly improves the performance. Addition of part-of-speech tags does not yield any improvement; however, when POS-tags are indexed with their positions in an utterance, accuracy is significantly improved and rises to 76.2. Addition of dependency relations (both with and without indexing with their position) does not improve the performance. Addition of the averaged pre-trained word-embedding vectors (from Google News) to the model with indexed POS-tags, however, rises the accuracy to 76.7. The model with word embeddings comes 0.3 short of the state-of-the-art results reported in (Ji et al., 2016).

5.2 Experiments on Aggregate ISO-standard Data

The methodology established on SWDA is applied to training the ISO 24617-2 subset models using the aggregate data set. Since in ISO 24617-2 annotation scheme DAs consist of semantic dimensions and communicative functions, the utterances are first classified into the considered semantic dimensions – general, social obligations management (SOM), and feedback. Then, we experiment with the Task dimension, reporting the results without error propagation from the previous step, in order to give the reader a clearer understanding of the current classification capabilities when restricting interactions with the system to general communicative functions.

5.2.1 Semantic Dimension Classification

The results of the binary dimension classification models on the test sets – DialogBank (DB), CAPC, and S-Logs – are reported in Table 4. The CAPC corpus consists of isolated utterances; consequently, the *Feedback* dimension is not present. On DB and S-Logs, on the other hand, the *Feedback* dimension yields the lowest accuracy in comparison to General and *SOM* dimension communicative functions. Low performances on the *Feedback* dimension could be explained by the fact that the training data mostly contains *Allo-feedback* and lacks *Auto-feedback* and *Feedback elicitations*, which are present in DB.

[3]From 73 ($C = 1.0$) to 75 ($C = 0.1$) for the model trained on unigrams, bigrams and previous DA-tag.

Dimension	DB	CAPC	S-Logs
General	73.3	83.0	80.2
SOM	78.1	90.7	86.6
Feedback	56.3	–	71.3
Overall	68.4	83.3	79.4

Table 4: Classification accuracies of the binary semantic dimension models: General, SOM and Feedback. The CAPC corpus does not contain Feedbacks, therefore results for this dimension are not reported.

Features	DB	CAPC	S-Logs	Features	DB	CAPC	S-Logs
BL: Majority	53.4	22.9	63.4				
1-2-grams	64.2	71.2	78.7	+ I-DEP	67.1*	74.3	82.3
+ PREV	64.3	70.7	81.6*	+ WE	65.2	74.8*	82.0
+ I-POS	65.8*	73.8*	82.2	+ I-DEP + WE	66.6	75.1	81.8

Table 5: Accuracies of the feature combinations on the general-purpose communicative functions on the test sets. The best results are marked in bold, and statistically significant differences with *.

5.2.2 Communicative Function Classification

The utterances are further classified into communicative functions of the General (Task) dimension, using the methodology established on SWDA, i.e. the same hyper-parameter settings ($C = 0.1$) and features. However, since models with dependency relations do not yield statistically significant differences, they are also considered. The results of the models on the test sets are reported in Table 5. The behavior of the models trained with various feature combinations is in-line with the SWDA experiments: the addition of the previous DA tags and part-of-speech tags indexed with their positions in a sentence improves the performance. Different from the SWDA, the addition of the indexed dependency relations improves the performance on the test sets. In the case of DialogBank and CAPC, their contribution is statistically significant. Additionally, unlike for SWDA, the addition of word embeddings with and without index dependency relation (I-DEP) does not produce significant improvements for all but CAPC. Consequently, the model trained on 1-2-grams, previous DA tags, indexed POS-tags and dependency relations is chosen for the ablation study.

5.2.3 Corpora Combinations

The aggregation of all the corpora mapped to our subset of ISO 24617-2 is not necessarily the best one, as the distributions of DA categories varies from corpus to corpus. Consequently, we also present results on the test sets for the models trained solely on SWDA and AMI; as well as perform an ablation experiment removing one corpus at a time. The best performing model from the previous subsection (1-2-grams, previous DA-tag, indexed POS-tags, and indexed dependency relations) is used for the study. The results of these experiments are reported in Table 6.

While the best results for Dialog Bank are achieved considering all the corpora, for CAPC the best results are achieved by removing MapTask. For S-Logs, on the other hand, the best performing corpora combination is all except VerbMobil. However, the performance differences from the models trained on all corpora are not statistically significant. Training DA taggers solely on SWDA and AMI – the largest and the most diverse corpora – yields performances inferior to the combination of all the corpora. From the table, we can also observe that these two corpora – SWDA and AMI – contribute most to the performance, as removing them affects the performance the most. On the other hand, removing the smaller datasets – BT Oasis, MapTask, and VerbMobil – affects the performance less.

6 Conclusions

We have presented an effective methodology for corpora aggregation for domain-independent Dialogue Act Tagging on a subset of the ISO 24617-2 annotation. We have also reported an accurate evaluation

Dataset	DB	CAPC	S-Logs	Dataset	DB	CAPC	S-Logs
ALL	**67.1**	74.3	82.3	- AMI	59.7	73.7	71.3
				- SWDA	60.2	68.3	77.5
SWDA only	57.9	71.3	53.5	- Oasis BT	66.1	74.2	81.8
AMI only	53.2	39.8	61.6	- MapTask	66.8	**74.6**	80.5
				- VerbMobil	66.5	74.0	**82.6**

Table 6: Accuracies of the corpora combinations on the test sets – Dialog Bank (DB), CAPC, and S-Logs.

of our approach on both in-domain and out-of-domain datasets, proving that the described DA tagging technique is indeed independent from the underlying scheme and task of the annotated corpora. Finally, the machine learning technique used for DA tagging was tested on a popular DA tagging task (the Switchboard corpus), obtaining very close to state-of-the-art results.

This work represents one of the first attempts to use an ISO compliant DA scheme for a real-life application, as well as one of the first structured approaches for evaluation of dialogue resources annotated with this taxonomy.

Research on available training resources is one of the first things to look forward to, since the current data proved to be effective, but also presented numerous drawbacks (lack of adequate coverage for the Feedback dimension, imbalanced DAs, lack of context-aware communicative functions). We plan to make our resource continue to grow in the future by adding and mapping additional corpora, such as the MRDA (Shriberg et al., 2004) corpus.

7 Acknowledgments

This project was partially funded by Amazon Alexa Prize 2017 research grant.

References

Jan Alexandersson, Bianka Buschbeck-Wolf, Tsutomu Fujinami, Michael Kipp, Stephan Koch, Elisabeth Maier, Norbert Reithinger, Birte Schmitz, and Melanie Siegel. 1998. *Dialogue acts in Verbmobil 2*. DFKI Saarbrücken.

Anne H Anderson, Miles Bader, Ellen Gurman Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, Stephen Isard, Jacqueline Kowtko, Jan McAllister, Jim Miller, et al. 1991. The hcrc map task corpus. *Language and speech*, 34(4):351–366.

John Langshaw Austin and JO Urmson. 1962. *How to Do Things with Words. The William James Lectures Delivered at Harvard University in 1955.[Edited by James O. Urmson.].* Clarendon Press.

Kevin K Bowden, Jiaqi Wu, Shereen Oraby, Amita Misra, and Marilyn Walker. 2017. Slugbot: An application of a novel and scalable open domain socialbot framework. *Alexa Prize Proceedings*.

Harry Bunt, Jan Alexandersson, Jean Carletta, Jae-Woong Choe, Alex Chengyu Fang, Koiti Hasida, Kiyong Lee, Volha Petukhova, Andrei Popescu-Belis, Laurent Romary, Claudia Soria, and David Traum. 2010. Towards an iso standard for dialogue act annotation. *Seventh conference on International Language Resources and Evaluation (LREC'10)*.

Harry Bunt, Jan Alexandersson, Jae-Woong Choe, Alex Chengyu Fang, Koiti Hasida, Volha Petukhova, Andrei Popescu-Belis, and David R Traum. 2012. Iso 24617-2: A semantically-based standard for dialogue annotation. In *LREC*, pages 430–437.

Harry Bunt, Volha Petukhova, Andrei Malchanau, Kars Wijnhoven, and Alex Chengyu Fang. 2016. The dialogbank. In *LREC*.

Harry Bunt. 1999. Dynamic interpretation and dialogue theory. *The structure of multimodal dialogue*, 2:1–8.

Harry Bunt. 2009. The dit++ taxonomy for functional dialogue markup. In *AAMAS 2009 Workshop, Towards a Standard Markup Language for Embodied Dialogue Acts*, pages 13–24.

Susanne Burger, Karl Weilhammer, Florian Schiel, and Hans G Tillmann. 2000. Verbmobil data collection and annotation. In *Verbmobil: Foundations of speech-to-speech translation*, pages 537–549. Springer.

Jean Carletta. 2006. Announcing the ami meeting corpus. *The ELRA Newsletter 11(1), January-March, p. 3-5.*

Alessandra Cervone, Giuliano Tortoreto, Stefano Mezza, Enrico Gambi, and Giuseppe Riccardi. 2017. Roving mind: a balancing act between open–domain and engaging dialogue systems. In *Alexa Prize Proceedings*.

Shammur Absar Chowdhury, Evgeny A Stepanov, and Giuseppe Riccardi. 2016. Transfer of corpus-specific dialogue act annotation to iso standard: Is it worth it? In *LREC*.

Mark G. Core and James F. Allen. 1997. Coding dialogs with the damsl annotation scheme. In *Proceedings of AAAI Fall Symposium on Communicative Action in Humans and Machines*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June.

Alex C. Fang, Jing Cao, Harry Bunt, and Xiaoyue Liu. 2012. The annotation of the Switchboard Corpus with the new ISO standard for dialogue act analysis. In *Workshop on Interoperable Semantic Annotation*.

Hao Fang, Hao Cheng, Elizabeth Clark, Ariel Holtzman, Maarten Sap, Mary Ostendorf, Yejin Choi, and Noah A. Smith. 2017. Sounding board – university of washington's alexa prize submission. In *Alexa Prize Proceedings*.

John J Godfrey, Edward C Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 517–520. IEEE.

Narendra Gupta, Gokhan Tur, Dilek Hakkani-Tur, Srinivas Bangalore, Giuseppe Riccardi, and Mazin Gilbert. 2006. The at&t spoken language understanding system. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):213–222.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *SIGDIAL Conference*, pages 263–272.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse relation language models. In *Proceedings of NAACL-HLT*, pages 332–342.

Dan Jurafsky. 1997. Switchboard swbd-damsl shallow-discourse-function. *Annotation, Technical Report, 97-02, University of Colorado, CO, USA.*

Harshit Kumar, Arvind Agarwal, Riddhiman Dasgupta, Sachindra Joshi, and Arun Kumar. 2017. Dialogue act sequence labeling using hierarchical encoder with crf. *arXiv preprint arXiv:1709.04250*.

Geoffrey Leech and Martin Weisser. 2003. Generic speech act annotation for task-oriented dialogues. In *Procs. of the 2003 Corpus Linguistics Conference, pp. 441Y446. Centre for Computer Corpus Research on Language Technical Papers, Lancaster University*.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, Jun.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*.

Volha Petukhova, Martin Gropp, Dietrich Klakow, Anna Schmidt, Gregor Eigner, Mario Topf, Stefan Srb, Petr Motlicek, Blaise Potard, John Dines, et al. 2014a. The dbox corpus collection of spoken human-human and human-machine dialogues. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC" 14)*, number EPFL-CONF-201766. European Language Resources Association (ELRA).

Volha Petukhova, Andrei Malchanau, and Harry Bunt. 2014b. Interoperability of dialogue corpora through ISO 24617-2-based querying. In *LREC*.

Patti J Price. 1990. Evaluation of spoken language systems: The atis domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Silvia Quarteroni and Giuseppe Riccardi. 2010. Classifying dialog acts in human-human and human-machine spoken conversations. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association*, pages 2514–2517.

Silvia Quarteroni, Alexei V Ivanov, and Giuseppe Riccardi. 2011. Simultaneous dialog act segmentation and classification from human-human spoken conversations. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5596–5599. IEEE.

Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, Eric King, Kate Bland, Amanda Wartick, Yi Pan, Han Song, Sk Jayadevan, Gene Hwang, and Art Pettigrue. 2017. Conversational ai: The science behind the alexa prize. In *Alexa Prize Proceedings*.

Elizabeth Shriberg, Raj Dhillon, Sonali Bhagat, Jeremy Ang, and Hannah Carvey. 2004. The icsi meeting recorder dialog act (mrda) corpus. Technical report, INTERNATIONAL COMPUTER SCIENCE INST BERKELEY CA.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3).

David Traum. 1996. Conversational agency: The trains-93 dialogue manager. In *In Susann LuperFoy, Anton Nijholt, and Gert Veldhuijzen van Zanten, editors, Proceedings of Twente Workshop on Language Technology, TWLT-II*. Citeseer.

V.N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 78–83. IEEE.

Xuesong Yang, Yun-Nung Chen, Dilek Hakkani-Tür, Paul Crook, Xiujun Li, Jianfeng Gao, and Li Deng. 2017. End-to-end joint learning of natural language understanding and dialogue manager. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 5690–5694. IEEE.

A Appendix A. Dialogue Acts mappings

ISO	SWDA	MapTask	VerbMobil	Oasis BT	AMI
Task					
Inform	Statement-non-opinion, Statement-opinion, Rhetorical-question, Statement expanding y/n answer, Hedge	explain, clarify	–	Inform	Inform
ChoiceQ	Or-question Or-clause	–	–	–	–
SetQ	Wh-question Declarative wh-question	query_w	–	q_wh	–
PropQ	Yes-no-question, Backchannel in question form, Tag-question, Declarative Yes-no-question	query_yn	–	q_yn	–
Commissive	Offer, Commit	-	Offer, Commit	Offer	Offer
Directive	Open-Option	Instruct	Request (all), Suggest	Suggest, imp	Suggest, Elicit-offer
Social Obligation Management					
Thanking	Thanking, You're-welcome	–	–	thank	–
Apology	Apology Downplayer	–	–	pardon regret	–
Salutation	Conventional-closing	–	–	bye, greet	–
Feedback					
Feedback	Signal-not-understanding, Acknowledge (backchannel), Acknowledge answer, Appreciation, Sympathy, Summarize/ reformulate, Repeat-phrase	Acknowledge	Feedback (all)	ackn	Backchannel

Table 7: Mapping for Dialogue Acts from each individual corpus to ISO DA-tags.

Arrows are the Verbs of Diagrams

Malihe Alikhani and Matthew Stone
Computer Science, Rutgers University
malihe.alikhani, matthew.stone@rutgers.edu

Abstract

Arrows are a key ingredient of schematic pictorial communication. This paper investigates the interpretation of arrows through linguistic, crowdsourcing and machine-learning methodology. Our work establishes a novel analogy between arrows and verbs: we advocate representing arrows in terms of qualitatively different structural and semantic frames, and resolving frames to specific interpretations using shallow world knowledge.

1 Introduction

Natural communication is multimodal—people get their ideas across not just through words but through gestures, diagrams, illustrations, and even practical activity. Research in discourse has found suggestive evidence for interpretative parallels across modalities; the challenge now is to substantiate these parallels at large scale, by developing broad-coverage cognitive models of multimodal communication.

This paper illustrates the promise of new data sets to further such a general understanding of visual communication. In particular, Kembhavi et al. (2016) have curated, annotated and released a sizeable corpus of scientific diagrams in diverse domains. We draw on formal analysis, crowdsourcing, and machine-learning experiments to carry out a systematic study of the interpretation of arrows in these diagrams. The central insight in our work is to give a novel account of arrows using existing *linguistic* concepts and methodology—specifically, ideas developed for explaining verb meaning in context.

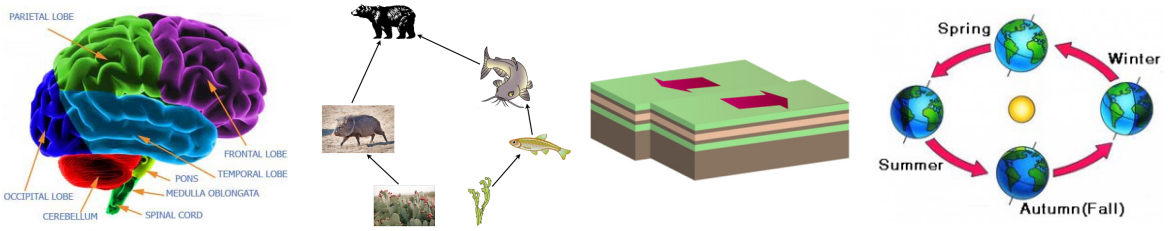
We begin in Section 2 with a tour of work on multimodal communication, which has made a strong case for parsing diagrams to organize basic conventional elements into recursive structural relationships, and for interpreting such parses by combining compositional meaning with discourse-based inference. We use this framework in Section 3 to motivate a linguistically-inspired approach to the interpretation of arrows: we characterize arrows into four qualitatively different structures, analogous to verb subcategorization frames; we associate each structure with a meaning, analogous to verb frame semantics (Baker et al., 1998); and we describe the contextual supplementation some structures require, by analogy to the co-compositionality of generative lexicon theory (Pustejovsky, 1998, GLT).

The remainder of the paper offers empirical evidence in support of our approach. Our first experiments, presented in Section 4, assess our semantic frames for arrows in light of crowd workers' judgments about the Kembhavi et al. (2016) data set. People label our categories with high agreement, and the categories account for the overwhelming majority of items in the corpus. Moreover, we find that arrows are normally used in one sense per diagram, much as lexical items exhibit one sense per discourse (Gale et al., 1992).

Our next experiments, presented in Section 5, assess our approach to contextual interpretation: certain frames must be supplemented with a salient relationship. Assuming one sense per diagram, we train a machine-learning model that predicts this relationship from the textual content of the diagram. The performance of the method corroborates our hypothesis (analogous to GLT) that these relationships can and should be resolved based on relatively shallow encyclopedic knowledge.

Our contributions are primarily formal—to characterize the knowledge needed to model diagrams. Fundamental challenges remain to apply such ideas in practice. We conclude in Section 6 with an

Proceedings of the 27th International Conference on Computational Linguistics, pages 3552–3563
Santa Fe, New Mexico, USA, August 20-26, 2018.



(a) Indication (b) Action (c) Direction (d) Motion

Figure 1: Examples of diagrams with arrows from our four different semantic frames. Diagram (a) has indication arrows, diagram (b) has action arrows, diagram (c) has direction arrows, and diagram (d) has motion arrows. See Section 3.2.

appraisal of our contributions: we suspect many other abstract pictorial elements can be described using linguistic concepts, enabling more expressive systems for diagram understanding and generation.

2 Related Work

Schematic pictorial content abounds in everyday communication, from informal interactions around a whiteboard to figures in formal scientific and engineering documents. Consequently, diagrams have been studied from the standpoint of design, as in Horn (1998); psychology, as in Larkin and Simon (1987); computer graphics, as in Agrawala et al. (2003); computer vision, as in Alvarado and Davis (2004); and common-sense reasoning, as in Forbus et al. (2011)—as well as natural language (NL) technology.

Our work is most directly inspired by prototypes in mulitmodal communication that use NL techniques either to synthesize communicative presentations, as in André et al. (1993) and Feiner and McKeown (1993), or to interpret multimodal user input, as in Johnston (1998), Johnston and Bangalore (2005) and Bangalore and Johnston (2009). For example, Johnston (1998) argues for processing diagrams with a syntactic parser, so that systems can recognize their essential hierarchical relationships, and for using a constraint-based grammar formalism, so specific productions can inherit constraints from general rule schemas, as in HPSG (Pollard and Sag, 1994). The success of such efforts shows that NL architectures can effectively capture communicative acts across a diverse range of modalities and contexts. This leads us to ask how we might develop domain-general tools and methodologies that could support such multimodal architectures—perhaps following the successful application to text of wide-coverage grammars (Copestake and Flickinger, 2000, among others) and parsers (de Marneffe et al., 2006, among others).

To pursue this direction, we build on an emerging trend of applying the theory of NL semantics and discourse to multimodal communication, which includes work on the grammar of coreference, such as Schlenker and Chemla (2017) for gesture, and Abusch (2013) and Cohn (2013) for comics, and work on the grammar of coherence, such as Lascarides and Stone (2009) for gesture, and Bateman and Schmidt (2013) and Cumming et al. (2017) for film. Looking at arrows is a novel contribution to this literature.

Of course, we can find many loose parallels to our ideas in work on diagrams across many disciplines. For example, researchers from Alvarado and Davis (2004) to Tversky et al. (2000) have observed that diagrams are composed from a small taxonomy of abstract elements, including arrows, that are grouped together in hierarchical relationships. Researchers such as Alvarado and Davis (2004), Forbus et al. (2011), and Horn (1998) have argued that these elements take on diverse interpretations, which must be recovered in context using conceptual content from a specific domain. Researchers such as Agrawala et al. (2003) and Forbus et al. (2011) have emphasized that effective diagrams abstract cognitively essential information in consistent and recognizable ways.

Our findings are entirely compatible with such observations, but our contribution is to approach them using the tools and methods of NL research. Other ways of implementing the ideas include custom architectures based on cognitive design principles, as in Agrawala et al. (2003), or structured user interaction, as in Forbus et al. (2011), as well as general methods for visual recognition and grouping, as in Alvarado and Davis (2004) and Kembhavi et al. (2016). Such approaches may be effective, especially for purely

visual processing, but they make it difficult to compare and integrate the reasoning involved in interpreting text and diagrams in combination, as well as to develop models that can be used simultaneously for understanding, generation and mixed-initiative interaction—a key virtue of NL techniques.

Our work is made possible by Kembhavi et al. (2016). They have collected, annotated and distributed a data set of more than 5000 diagrams scraped from the web using science education key phrases as seeds for Google searches. We refer to this as the AI2D data set. Kembhavi et al. (2016) characterize diagram content by associating diagrams with high-level questions and answers. Meanwhile, their work formalizes the structure of diagrams using a graph-based representation that features a regimented set of elements and relationships. Their annotation protocol broke this process of building these representations down into a series of lightweight decisions that could be accomplished reliably by crowd workers on Amazon Mechanical Turk. This annotation scheme, while useful in describing the shallow organization and overall content of diagrams, says little about the semantic and pragmatic knowledge that connects form and content. Our work addresses one case of this open problem.

3 Towards a Formal Analysis of Arrows

We start by exploring how linguistically-inspired representations might capture the intuitive structure and interpretation of arrows. We draw on the AI2D examples in Figure 1, which are representative and diverse: Figure 1a maps areas of the human brain; Figure 1b shows part of the food chain of a northern forest; Figure 1c illustrates plate tectonics; and Figure 1d explains the seasons. Our observations about the form, meaning and interpretation of arrows in these diagrams lays the groundwork for the experiments we report in Sections 4 and 5 (and anticipates further formal investigations).

3.1 Formal Structure

An arrow consists of a head, a tail and a body.[1] The head and tail can each be structurally linked to another constituent in the diagram, to create a larger unit containing the arrow and its affiliates. Thus the arrows of Figure 1a link the text at the tail to the pictorial element at the head; the arrows of Figure 1b link two pictorial elements; and the arrows of Figure 1c are anchored into the scene rather than connected to elements at the head or tail. We assume this structure in giving examples in running text: ♘ → ♙. This schematizes an arrow with ♘ at the tail and ♙ at the head. Where arrows are not understood to depict spatial information, we will use these schemas to abstract away from details of layout and rendering (so ♙ ← ♘ would be equivalent).

Three classes of form are particularly important to our further investigations:

1. arrows with pictorial elements at both the head and tail, as in ♘ → ♙;
2. arrows with pictorial elements at the head and text at the tail, as in bishop → ♗; and
3. arrows with an empty head or tail area (and perhaps pictorial elements elsewhere), as in ♖ → or → .[2]

We see this structural variation as analogous to the syntactic subcategorization of verbs: the idiosyncratic character of verbal complementation motivates a fine-grained account of syntactic combination. For example, the verb *get* can take arguments of a range of different syntactic categories, involving noun phrases, predicates and full sentences: “I got flowers for my mother”, “Get well soon”, “I get that this is difficult”. In a similar way, arrows have optional argument positions (head and tail, comparable to subject and object) which may be filled by elements of formally different categories (for example, with textual or pictorial elements).

[1] We focus on simple arrows here of the form →, excluding double-headed arrows ↔ and pure line segments (—). Arrows can also differ in rendering attributes such as width, perspective and color, which sometimes provide further cues to the role of the arrow in the diagram.

[2] Arrows with pictorial elements at the tail and text at the head also occur: ♠ → oxygen. We hypothesize that these have analogous properties to purely pictorial arrows, but defer investigation to future work.

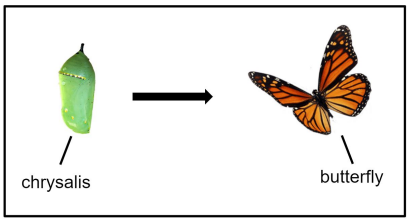


Figure 2: Part of a butterfly cycle of life diagram.

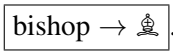
Just as with verbs, some structural possibilities for arrows seem not well-formed: it's odd, for example, to have text at the tail of an arrow and nothing at its head, as in # problem →.[3]

As noted by many researchers (Alvarado and Davis, 2004; Johnston, 1998; Tversky et al., 2000), diagrams are hierarchically organized. We hypothesize that arrows can contribute to the overall structure of a diagram in diverse ways, just as verbs contribute in diverse ways to the structure of sentences and discourse. Of course, an arrow can establish the overall structure of a diagram, as in Figure 2—just as the main verb projects the root of a sentence. Arrows in diagrams can also be modified, for example with labels and other annotations; we show a few such examples in Figures 3 and 4. In addition, many arrows seem to attach to and modify existing elements, just as verbs can anchor relative clauses, which adjoin in as modifiers to existing structure. For example, we might analyze the indication arrows of Figure 1a as modifiers that expand on the presentation of individual brain regions, with the head of the arrow playing a role analogous to an extracted argument in a subject or object relative clause—compare *the region we call the frontal lobe*. Similarly, the direction arrows of Figure 1c are similar to locative relative clauses—compare *a place things are pushed eastward*. Finally, the arrows of Figure 1b or 1d seem broadly analogous to the use of sentence-final appositives to carry the discourse forward (Koev, 2012)—compare *the prickly pear is eaten by the boar, which is eaten by the bear*. Note, however, that the two-dimensional structure of diagrams allows for new structural possibilities, such as the circular dependencies exhibited in Figure 1d. Formalisms for encoding diagram structure involve generalized notions of syntactic combination (Johnston, 1998). Integrating our account of arrow structure into such a generalized syntactic framework remains a topic for future work.

3.2 Meaning Frames

Arrows have qualitatively different meanings. One difference is whether the path has spatial meaning. For example, the arrows in Figure 1b indicate temporal or causal relationships with no spatial content. By contrast, the arrows of Figure 1c are rendered in perspective with the diagram's tectonic plates to depict the direction of action in an earthquake.

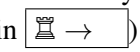
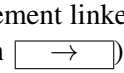
These qualitatively different interpretations are correlated with the structural description of an arrow. We see these constraints as analogous to the semantic frames assigned to verbs in databases like FrameNet (Baker et al., 1998). A semantic frame is a conceptual structure describing an event or a relation and the participants in it. For example, "Mary sold her car to John" and "John bought Mary's car" describe the same semantic frame. We interpret each structural option with an appropriate underlying conceptual representation, or frame, and assign each argument an appropriate role in the frame. Our frames for arrows are INDICATION, ACTION, DIRECTION, and MOTION.

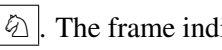
- The INDICATION frame, for arrows like those in Figure 1a, involves a SUBJECT (prototypically, a pictorial element at the head), and a LABEL (text at the tail), as in bishop →  . In indication, the label represents the name, location or category of the subject.

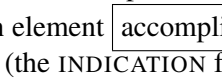
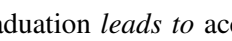
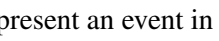
- The ACTION frame, for arrows like those in Figure 1b, involves a SOURCE (prototypically, a pictorial element at the tail) and a TARGET (prototypically, a pictorial element at the head), which are

[3]The distinctive status of text means that arrows with text at the tail are often equivalent to line segments with text at one end. In fact, only 19.09% of the labeling relations in AI2D are drawn with arrows; the rest are line segments. Other arrows cannot be replaced by segments.

understood to stand in a contextually-specified telic or agentive RELATION, as in  . Such arrows may represent causation, change, consumption, or transfer of possession depending on the content of the diagram. Source and target may represent different objects or different states of the same object.

- The DIRECTION frame, for arrows like those in Figure 1c, involves the DIRECTION indicated by the rendering of the arrow. This direction gives a contextually-specified ATTRIBUTE of a SUBJECT, which may be an element linked to the arrow (as in ) but may also be whatever lies where the arrow sits (as in ). The implicit meaning of such arrows has to be recovered using context. For instance, direction arrows can show the direction of traffic in street signs, the direction of the wind in weather maps, and the flow of material in mechanical drawings.

- The MOTION frame, for arrows like those in Figure 1d, involves a MOVER, which is rendered *twice*, both at the head and at the tail of the arrow, as in  . The frame indicates a translation event in which the mover starts in the tail configuration and ends at the head configuration.

Minimal pairs underscore the qualitative distinctions among these interpretations, which must therefore rely on conventional knowledge, not just common sense. An element  probably indicates that graduation *amounts to* an accomplishment (the INDICATION frame), while the inverse element  probably means that graduation *leads to* accomplishment (the ACTION frame). The possibilities for ambiguity also support qualitative analyses of arrow meaning. For example, in some diagrams, the element  would represent an event in which a single ♘ moves through space (the MOTION frame); in others, it would represent one in which a first ♘ does something to a second (the ACTION frame). The experiments in Section 4 are focused at assessing whether (and how) people represent and resolve such ambiguities.

3.3 Resolving Interpretation

Designers like Horn (1998) enumerate the specific interpretations of arrows seen in Figure 1: the meaning *be eaten* of the arrows in Figure 1b, the meaning *driving force* of the arrows in Figure 1c, or the meaning *orbits* of the arrows in Figure 1d. Such alternatives cannot be listed in a wide-coverage approach; they have to be derived from the specific purpose and subject matter of the diagram. Work on verbs, such as the generative lexicon theory of Pustejovsky (1998), offers a suggestive model for how this derivation might go. Pustejovsky (1998) proposes that underspecified verbs (famously, *begin*) retrieve implicit actions via the *qualia structure*—the natural action associates—of their arguments. The broader lesson is that shallow associations can be a surprisingly good place to look for contextual interpretations. The experiments of Section 5 substantiate this suggestion by seeing what features are needed to recover the context-sensitive relation associated with the ACTION frame.

4 Distinguishing Semantic Categories

The linguistic approach of Section 3 raises empirical questions about qualitative ambiguities in diagrams. Do people distinguish between arrows that signal indication, action, direction and motion frames? Can we develop corpora that annotate our frame distinctions explicitly? AI2D annotations already distinguish indication arrows from other arrows, but do not encode our other distinctions, so we conducted a series of human-subjects experiments to address these questions. The studies were conducted with the approval of our human subjects review committee. The data is available electronically.[4]

4.1 Materials

The annotations in the AI2D dataset suggest that 2077 diagrams have arrows, but the annotations are not always accurate. We removed 196 diagrams that do not have arrows. Further, to focus on clear cases, we removed 232 diagrams: those whose action arrows involved words, question marks or letters at the

[4] https://github.com/malihealikhani/Arrows_are_Verbs

arrow type	count
obj–obj	985
txt–obj	378
two–loc	138
empty	70
other	17

	txt–obj	two–loc	empty	other
obj–obj	6	3	3	17
txt–obj		2	2	2
two–loc			16	2
empty				1

Table 1: Results of experiment 1. Left, counts of diagrams tagged with a unique arrow category. Right, counts of diagrams tagged with exactly two arrow categories. We had only 5 diagrams tagged with three categories and no diagrams tagged with more.

head rather than pictorial elements (as in our ♠ → oxygen example, the majority of exclusions); those whose illustrations were cropped and incomplete, and cases such as "flash cards" that involved multiple renderings (a small number of further cases). This left us with 1634 diagrams from the AI2D dataset.

4.2 Methods

In the first experiment, our goal was to look at the structural features of arrows, as well as the semantic distinction between the MOTION and ACTION frames. (In pilot studies, we found that offering subjects a broad range of options let them categorize arrows more reliably.) In each interaction, subjects saw a diagram and were asked to answer if any arrows in this diagram are:

1. between two different objects
2. between a label and an object
3. between two different locations of an object
4. between empty areas
5. none of the above

Option 1 is indicative of arrows in the ACTION frame (obj-obj). Option 2 corresponds to the INDICATION frame (txt-obj); Option 3, to the MOTION frame (two-loc); and Option 4, to the DIRECTION frame (empty). Subjects were able to choose more than one answer, as diagrams sometimes include multiple kinds of arrows. (See Figure 3.)

The goal of the second experiment was to see if people interpret empty arrows in line with the DIRECTION frame. In each interaction, subjects saw one of the 93 diagrams labeled as having empty arrows in Experiment 1, and answered whether any arrows in this diagram showed:

1. the direction of the movement of the objects
2. the different locations of the objects
3. none of the above

Option 1 and 2 explain the interpretation of the DIRECTION (dir) and MOTION (mov) frames respectively.

In all, we recruited 860 subjects through Amazon Mechanical Turk. subjects were all US citizens, high school graduates and had a computer to be able to participate in the experiments. Subjects gave written consent and received $0.10 for the first experiment and $0.30 for the second experiment for their participation in each human intelligence task (HIT), an estimated hourly rate of $15.00. Each HIT asked subjects to characterize all the arrows in three separate diagrams. Each diagram has one data point in the dataset; in addition, we collected additional data collecting multiple responses for a random subset of the data (544 diagrams in Experiment 1, 47 in Experiment 2) to assess reliability.

4.3 Results

We followed Chen et al. (2005) in measuring inter-rater agreement. Their method is a generalization of Kappa statistics for cases that multiple raters are involved in a multi categorization task. For the first experiment, the overall kappa is 0.9744. For the second experiment, the overall kappa is 0.8517. This

type	count
dir	72
mov	9
other	0

	mov	other
dir	3	7
mov		2

Table 2: Results of Experiment 2. Left, diagrams with a unique response. Right, diagrams with two responses. No diagrams had all three responses.

Diagram category	**No. of items**	**Top arrow**	**Arrow rate**
Group A	353	txt–obj	97.73%
Group B	1007	obj–obj	94.44%
Group C	221	two-loc	84.31%
Group D	71	empty	91.57%

Table 3: Grouping diagrams by topic gives a strong indication of the structure and meaning of arrows.

indicates that subjects strongly agreed on their judgements, suggesting that the structural and interpretive distinctions that underwrite our semantic frames are clear to human interpreters.

This established, we can look at the distribution of annotations. Our first results appear in Table 1. Observe that only 38 diagrams seem to have an arrow classified in the "other" category; this suggests that our taxonomy has good coverage of frequent patterns. Further, the vast majority of the diagrams (1571 of 1634) feature exactly one kind of arrow; this is reminiscent of findings that words are used with one sense per discourse (Gale et al., 1992). This suggests that diagram understanding systems should be designed to look at a diagram holistically and prefer uniform interpretations for all the arrows.

Table 2 specifically focuses on the second experiment, which looked at whether DIRECTION is the appropriate semantic frame for empty arrows. The results are again consistent with our meaning frames. In particular, the 72 diagrams tagged as having just direction interpretations in Experiment 2 include 68 of the 70 diagrams tagged as having just empty arrows in Experiment 1. In addition, 12 out of 16 diagrams that were tagged with more than one option in Experiment 1 were reported to have multiple kinds of arrows in Experiment 2.

The AI2D dataset groups diagrams into different categories such as "Atom Structure" and "Moon Phase Equinox". We made four larger groups of diagrams categories for diagrams in AI2D: Group A includes diagram that map out scientific structures; Group B includes diagrams that present interactions between elements; Group C explains changing geometric configurations; and Group D highlights momentary spatial relationships.[5] These categories turn out to be highly correlated with a particular structure and interpretation of arrows: see Table 3. This provides further evidence that diagram interpretation should be guided holistically by general domain knowledge.

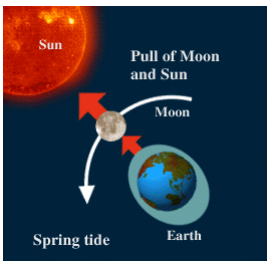
4.4 Exceptions

In an effort to understand the limitations of our taxonomy, we inspected the diagrams tagged as having multiple types of arrow and additional types of arrow.

We found that when designers use arrows to represent different meanings, they almost always render those arrows with different shapes or colors to represent different meanings. Each kind of arrow therefore has a consistent interpretation in the diagram after all. The left and the middle diagrams in Figure 3 show two examples of diagrams of this kind. The right diagram however is an example of the few that do not follow this principle.

We designed an experiment with the same terms and conditions as Experiment 2 to test the hypothesis that different kinds of arrows in complex diagrams have different appearances. We chose all diagrams that were tagged in Experiment 1 with more than one option (excluding diagrams that were tagged with "other") and asked participants to report how many different kinds of arrows with different appearances they see in each diagram. The result of the experiment shows that 21 out of 23 diagrams were reported to have more than one kind of arrow. This is in particular important in designing systems that can generate or understand diagrams. Such systems must allow each shape to get a different interpretation.

[5]Specifically, Group A is AI2D typesOf, partsOfA, atomStructure, partsOfTheEarth, circuits, solarSystem, rockStrata. Group B is Photosynthethis, lifeCycles, foodChainsWebs, waterCNPCycle, volcano, rockCycle. Group C is moonPhaseEquinox. Group D is faultsEarthquakes, eclipses.



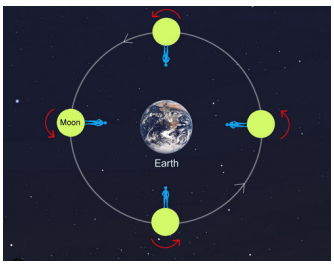
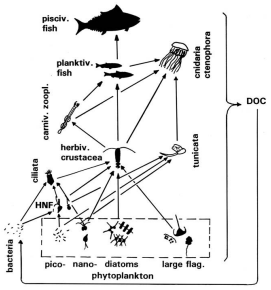


Figure 3: Authors typically render arrows with different interpretations differently. Left: thin circular arrows depict the moon's orbit (motion) while thick straight arrows depict the force of gravity (direction). Right: red arrows represent the direction of the movement of the moon where the white arrows are between two different locations of the moon.



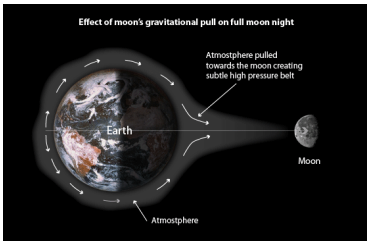


Figure 4: Examples of diagrams tagged as having other kinds of arrows.

Diagrams with arrows that were labelled as having additional kinds of arrows don't straightforwardly suggest omissions in our structural inventory. One case we might consider is arrows linked at their head to other arrows—a generalization of our INDICATION frame. The right diagram in Figure 4, includes such arrows. One arrow shows the direction of flow of the atmosphere; another arrow links that arrow to text explaining that the flow arises through the moon's gravity. A few other diagrams that come with arrows that are annotating another set of arrows were tagged other as well.

However, most of the "other" examples may simply have been difficult to understand. Some of these diagrams, like the left diagram in Figure 4, have arrows whose path are cutting each other. Similarly, some have labels that are not in proper places—for example very far from the tail of the arrow—or other unnecessary and distracting elements, such as mountains and clouds in a food chain diagram. These observations suggest that certain visual features can increase users' cognitive load in interpreting action diagrams. We suspect that viewers' cognitive limitations are an important constraint in diagram generation—a suggestion we return to in the conclusion.

4.5 Limitations

The experiments of Section 4 suggest that people agree on the structural organization of arrows in diagrams, and they also assign arrows consistently into semantic frames based on the identity of depicted objects and the broad significance of the arrow. While these results support a linguistic approach to diagram organization, they require substantial development to inform automatic diagram understanding. Automatic methods have trouble recognizing structural links between diagram elements (Kembhavi et al., 2016). Successful practical methods for diagram parsing rely on sketch stroke order (Johnston, 1998; Alvarado and Davis, 2004), which is not available in AI2D. Inferring coreference among schematic pictorial elements is also a challenging open problem, since current machine learning models of computer vision are notoriously bad at line art.

5 Interpretation of Action Arrows

The result of the experiments in section 4 showed that the specific import of arrows varies across contexts, but is typically consistent within a diagram and strongly contained by its subject matter. This means that once we know that an arrow falls into the direction, motion or indication category we have the interpretation of the arrow. However, in order to come up with the exact relation that explains the action arrows, we need to take a further step. In this section, we look specifically at how to predict the specific relation associated with an action arrow. Previous work has focused on resolving these interpretations using handcrafted knowledge bases describing specific domains (Alvarado and Davis, 2004; Forbus et al., 2011). Our analogy with verbs suggests that shallow methods based on element statistics and simple machine learning methods may provide more practical methods for inferring the domain-specific meaning of arrows in particular cases.

We carried out computational experiments to assess this hypothesis. Our machine learning classifier takes as input the verbal information in a diagram, such as label text and other annotations, and predicts the likely interpretation of all its action arrows.

5.1 Dataset and experimental setup

To link action arrows with their interpretations, we ran a pilot study on Mechanical Turk with 56 diagrams (5% of the diagrams that were tagged with action arrows in Experiment 1). Each subject saw a single diagram and responded to the prompt: "If you remove the arrows in this diagram, what words would you replace them with to convey the same meaning?" The results were consistent with the hypothesis that each diagram uses arrows to convey a single relation; unsurprisingly, however, the responses used diverse vocabulary (Furnas et al., 1987), so could not be used directly as training data for a classifier.[6] Instead, we associated diagrams with arrow meanings consistently based on AI2D categories: We used "turn into" for cycle of life and rock cycle diagrams; "eat/be eaten" for food chain web diagrams and "produce/receive" for photosynthesis diagrams.

AI2D includes multiple diagrams with similar content. We want to test whether we can learn general patterns for classifying arrows, not just memorize the features of individual examples. Thus, we organized our experiment in blocks using diagrams with different subject matter. We divided the data into three: we randomly split the receive class (it has only 93 datapoints), but we divide the two other groups, turn into and eats, based on three existing categories in AI2D (cycle of rocks, volcano, cycle of life) and three subcategories that we found in food web data (ocean, large birds, big cats). Since the number of blocks is small, we report cross-validation results leaving out one fold as test data.

5.2 Models

We used a bag of words (BoW) model with the SVM classifier as our baseline.

We compared the baseline to a convolutional neural network (CNN) with one convolution layer over pretrained word embeddings (Collobert et al., 2011; Kim, 2014). We used a max-over-time pooling operation and dropout regularization. See Table. 5. This setting enabled us to measure the effectiveness of considering distributional similarity in our classification task and thus the importance of generalizing based on shallow semantic knowledge.

To see how important it is to semantically understand the input by using sequential and structural information, we experiment with hierarchical attention RNNs (Yang et al., 2016). We made a look-up table for vocabulary, converted all words to integers and applied a one hot encoder. Next, we padded the input sequences, and fed them through Embedding, LSTM and Dense layers. Model Architectures are described in detail in Appendix A.

We trained the models to minimize binary cross entropy. Text was fed in left to right and top to bottom based on the position in the diagram.

[6] 44 responses consisted of a single verb (differing across subjects: turn into, metamorphose), and 5 included two verbs with similar meanings (develop, turn into), while 7 included nouns or short sentences.

	data size	CNN	RNN	SVM
fold 1	train: 1175	0.859	0.704	0.571
	test: 269			
fold 2	train: 1101	0.841	0.749	0.623
	test: 343			
fold 3	train: 1287	0.862	0.673	0.466
	test: 175			
Avg	train: 1187	0.842	0.708	0.553
	test: 262			

Table 4: Accuracy of predicting implicit RELATION for diagrams with ACTION arrows.

5.3 Results

Table 4 gives our experimental results. The CNN architecture works best. Its overall effectiveness corroborates our hypothesis that arrows pick up straightforward action associates of the objects they connect. The improvement over the SVM model suggests that distributional similarity is an effective proxy to the world knowledge needed to understand a diagram. The improvement over RNNs suggests that linguistic structure is not important for this task.

Error analysis revealed three major problems. The majority of the examples that our models failed to classify were the ones that contain labels that are shared between categories. For example, words such as "animal" and "plant" show up in both "turn into" and "eat" categories. We also had problems with diagrams labeled in languages other than English, and some diagrams with irrelevant labels, such as photosynthesis diagrams with "car" and "factory" labels.

Working with labels enable us to take into account the pictorial information that is available in a diagram without using a vision classifier. We can see that having a knowledge of the category of the diagram suffices for resolving the underspecification problems.

6 Conclusion

Arrows, we have argued, have a constrained form, meaning and interpretation, which can be theorized using linguistic concepts. This means, in particular, that we can ask subjects to report their interpretation of arrows in linguistic terms and use machine learning methods to resolve ambiguities in the interpretation of arrows in qualitative, linguistically-motivated ways. Vision researchers sometimes compare diagram understanding to scene understanding (Kembhavi et al., 2016). That may be right when it comes to grouping elements in diagrams together and recognizing pictorial elements—though much research on those topics is still needed. When it comes to organizing the content of a diagram as a whole, we think discourse modeling and other NL techniques also offer important tools.

Many of our findings echo designers' advice about effective diagrams (Tufte, 2001). They are simple. They have limited figurative content. Most of them have one kind of arrow that's easy to interpret based on general background knowledge. If they have multiple kinds the shape of the arrows are different so that the reader can distinguish between the two kinds of meanings that he has to infer. Our results therefore promise to help inform systems for diagram generation (Agrawala et al., 2003).

Our empirical work explores only a small number of the ways arrows vary—and diagrams include a wide range of other elements that we think can be studied with methods like ours. More generally, the minimal pairs we sketched in Section 3 suggest that we can operationalize interpretive constraints on multimodal communication with predictive, wide-coverage methods. We hope to contribute to such formalisms going forward.

7 Acknowledgement

This work was supported by NSF Award IIS-1526723 and has benefited from discussions with Doug DeCarlo, Gabriel Greenberg and anonymous reviewers.

References

Dorit Abusch. 2013. Applying discourse semantics and pragmatics to co-reference in picture sequences. In Emmanuel Chemla, Vincent Homer, and Grégoire Winterstein, editors, *Proceedings of Sinn und Bedeutung 17*, pages 9–25, Paris.

Maneesh Agrawala, Doantam Phan, Julie Heiser, John Haymaker, Jeff Klingner, Pat Hanrahan, and Barbara Tversky. 2003. Designing effective step-by-step assembly instructions. *ACM Trans. Graph.*, 22(3):828–837.

Christine Alvarado and Randall Davis. 2004. Sketchread: a multi-domain sketch recognition engine. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology, Santa Fe, NM, USA, October 24-27, 2004*, pages 23–32.

Elisabeth André, Wolfgang Finkler, Winfried Graf, Thomas Rist, Anne Schauder, and Wolfgang Wahlster. 1993. WIP: the automatic synthesis of multimodal presentations. In Mark T. Maybury, editor, *Intelligent Multimedia Interfaces, the book is an outgrowth of the AAAI Workshop on Intelligent Multimedia Interfaces, Anaheim, CA, USA, August, 1991.*, pages 75–93. AAAI Press / The MIT Press.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90, Montreal, Quebec, Canada, August. Association for Computational Linguistics.

Srinivas Bangalore and Michael Johnston. 2009. Robust understanding in multimodal interfaces. *Computational Linguistics*, 35(3):345–397.

John Bateman and Karl-Heinrich Schmidt. 2013. *Multimodal film analysis: How films mean*, volume 5. Routledge.

Bin Chen, D Zaebst, and Lynn Seel. 2005. A macro to calculate kappa statistics for categorizations by multiple raters. In *Proceeding of the 30th Annual SAS Users Group International Conference*, pages 155–30.

Neil Cohn. 2013. *The Visual Language of Comics: Introduction to the Structure and Cognition of Sequential Images.* A&C Black.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Ann A. Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage english grammar using HPSG. In *Proceedings of the Second International Conference on Language Resources and Evaluation, LREC 2000, 31 May - June 2, 2000, Athens, Greece*.

Samuel Cumming, Gabriel Greenberg, and Rory Kelly. 2017. Conventions of viewpoint coherence in film. *Philosophers' Imprint*, 17(1):1–29.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation, LREC 2006, Genoa, Italy, May 22-28, 2006.*, pages 449–454.

Steven Feiner and Kathleen McKeown. 1993. Automating the generation of coordinated multimedia explanations. In Mark T. Maybury, editor, *Intelligent Multimedia Interfaces, the book is an outgrowth of the AAAI Workshop on Intelligent Multimedia Interfaces, Anaheim, CA, USA, August, 1991.*, pages 117–138. AAAI Press / The MIT Press.

Kenneth D. Forbus, Jeffrey M. Usher, Andrew M. Lovett, Kate Lockwood, and Jon Wetzel. 2011. Cogsketch: Sketch understanding for cognitive science research and for education. *topiCS*, 3(4):648–666.

George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. 1987. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1992. One sense per discourse. In *Proceedings of the Workshop on Speech and Natural Language*, HLT, pages 233–237, Stroudsburg, PA, USA. Association for Computational Linguistics.

Robert E. Horn. 1998. *Visual Language: Global Communication for the 21st Century*. MacroVU, Incorporated.

Michael Johnston and Srinivas Bangalore. 2005. Finite-state multimodal integration and understanding. *Natural Language Engineering*, 11(2):159–187.

Michael Johnston. 1998. Unification-based multimodal parsing. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, COLING-ACL '98, August 10-14, 1998, Université de Montréal, Montréal, Quebec, Canada. Proceedings of the Conference.*, pages 624–630.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. 2016. A diagram is worth a dozen images. In *European Conference on Computer Vision (ECCV)*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Todor Koev. 2012. On the information status of appositive relative clauses. In *Logic, language and meaning*, pages 401–410. Springer.

Jill H. Larkin and Herbert A. Simon. 1987. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11(1):65–100.

Alex Lascarides and Matthew Stone. 2009. A formal semantic analysis of gesture. *Journal of Semantics*, 26(4):393–449.

C. Pollard and I.A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. University of Chicago Press.

James Pustejovsky. 1998. *The Generative Lexicon*. Bradford Books. MIT Press.

Philippe Schlenker and Emmanuel Chemla. 2017. Gestural agreement. *Natural Language & Linguistic Theory*, Oct.

Edward R. Tufte. 2001. *The Visual Display of Quantitative Information*. Graphics Press.

Barbara Tversky, Jeff Zacks, Paul U. Lee, and Julie Heiser. 2000. Lines, blobs, crosses and arrows: Diagrammatic communication with schematic figures. In *Theory and Application of Diagrams, First International Conference, Diagrams 2000, Edinburgh, Scotland, UK, September 1-3, 2000, Proceedings*, pages 221–230.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. 2016. Hierarchical attention networks for document classification. In *HLT-NAACL*, pages 1480–1489.

A Learning Experiment Parameters

Model learning code is available with the electronic resources accompanying this paper at `https://github.com/malihealikhani/Arrows_are_Verbs`. Key parameters for our machine learning experiments are provided in Table 5.

Model	Operation	Input	Kernel size	Output	Activation
CNN	Embedding	–	–	128	–
	Conv	Embedding	(3, 128)	100	Relu
	Dropout	Conv	–	300	rate=0.5
	Dense	Dropout	100	1	Sigmoid
RNN	Embedding	–	–	(20, 64)	Relu
	LSTM	Embedding	64	64	–
	Dense	LSTM	64	3	Tanh

Table 5: CNN and RNN architecture details.

Improving Feature Extraction for Pathology Reports with Precise Negation Scope Detection

Olga Zamaraeva
University of Washington
Department of Linguistics
olzama@uw.edu

Kristen Howell
University of Washington
Department of Linguistics
kphowell@uw.edu

Adam Rhine
University of Washington
Department of Linguistics
amrhine@uw.edu

Abstract

We use a broad coverage, linguistically precise English Resource Grammar (ERG) to detect negation scope in sentences taken from pathology reports. We show that incorporating this information in feature extraction has a positive effect on classification of the reports with respect to cancer laterality compared with NegEx, a commonly used tool for negation detection. We analyze the differences between NegEx and ERG results on our dataset and how these differences indicate some directions for future work.

Title and Abstract in Russian

К вопросу о применении формальных грамматик в посторении признаковых векторов для классификации отчетов о заключениях патологоанатомов

Мы предлагаем способ обогащения векторов признаков (*feature vectors*) лингвистичекой структурой. Для внешней оценки эксперимента (*extrinsic evaluation*) мы смотрим на результаты автоматической классификации отчетов о заключениях патологоанатомов (*pathology reports*, такие как заключение о результатах гистологического исследования и др.). В качестве лингвистической структуры мы выбрали сферу действия отрицания (*scope of negation*), так как известно, что отрицание в тексте может влиять на качество классификации медицинских заключений. В эксперименте мы используем имплементацию формальной грамматики английского языка (English Resource Grammar) для определения сферы действия отрицания на уровне предложения. Сначала текст отчета делится на предложения. Из каждого предложения при помощи базы данных UMLS мы выбираем слова и фразы – признаки. Те признаки, которые оказываются в сфере действия отрицания, мы заменяем на специальные признаки. Например, в предложении *No tumor in left breast*, если сфера отрицания определена достаточно широко, признак *left breast* будет заменен на признак *NEG:left breast*. Все признаки, полученные из одного отчета, собираются в один вектор (признаковый вектор данного отчета). Обогащенные таким образом признаковые векторы затем используются для классификации отчетов на группы по расположению первичного очага (в правой или левой части легкого или молочной железы). Мы сравниваем воздействие нашей системы построения признаковых векторов на точность классификации с популярной системой NegEx, которая использует для определения сферы действия отрицания ряд эвристик. Результаты и анализ ошибок позволяют увидеть, что формальная грамматика, основанная на глубоком синтаксическом анализе, действует точнее, чем система NegEx. Система NegEx может ошибочно предположить сферу действия отрицания в плохо структурированном тексте (например, в блоке текста, неправильно разбитом на предложения), а формальная грамматика в этом случае не выдаст никакого ответа и тем самым избежит ошибки. Мы делаем вывод, что использование формальных грамматик для подобных задач может быть целесообразно.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3564–3575
Santa Fe, New Mexico, USA, August 20-26, 2018.

1 Introduction

Detecting negation scope remains a challenge in clinical notes information extraction (see e.g. Morante and Daelemans 2009 for an overview of existing approaches for negation detection and related issues). Perhaps the most extensively used negation detection tool is NegEx (Chapman et al., 2001), a regular expressions-based tool, with 696 citations on Google Scholar,[1] which is part of such well-known systems as cTAKES (Savova et al., 2010). One reason NegEx is so popular is that it is rule based and thus easily adapted to any English dataset. In general, rule based methods are important for domains where data cannot be easily shared and so there is no guarantee an existing machine learning approach can be successfully used, such as the clinical notes domain. That said, NegEx identifies negated tokens from surface strings using a set of domain-specific heuristics which it applies directly to the surface strings rather than to some syntactic or semantic representations of those strings. Such an approach is unlikely to capture the rules of English negation in their generality. In contrast, linguistically motivated implemented formal grammars[2] map surface strings to abstract syntactic and semantic representations and model language rules more accurately and robustly, in a domain-independent fashion.

For this paper, we classified a set of pathology reports diagnosing the cancer as located in the right or in the left part of the body, an important data element used in clinical operations and research for paired organ sites. The contribution of the paper is in incorporating precise negation scope information obtained with an implemented broad-coverage grammar of English into the feature extraction procedure. The experiment yielded incremental though consistent improvement when compared to NegEx, showing fairly clear evidence that incorporating robust domain-independent linguistic structure into feature extraction can be helpful for classification of this kind.

After summarizing related work (section 2), we present our experiment (section 3). The structure of the experiment is as follows. Each pathology report in the dataset (section 3.1) is represented by a feature vector constructed from a Unified Medical Language System (UMLS)-based feature set, which is obtained as described in section 3.2 and then augmented by what we call *negated features* (section 3.3). To help the reader, we introduce negated features right here first: In any sentence, a lexical feature which was selected as highly informative for the training dataset (e.g. *right*, or *malignancy*) could actually correspond to a negated concept. Assume the feature *malignancy* was selected by a feature selection algorithm and will now be used in a classification task on a test dataset. Compare two hypothetical single-sentence reports from this test dataset: *A malignancy was identified* and *No malignancy was identified*. For the second report, which contains the feature *malignancy* within the scope of negation, we want to exclude that feature when we turn this sentence into a vector. Otherwise, if we were to include *malignancy*, we might misclassify the report as positive.[3] Furthermore, a negated concept (which we will represent as *NEG:malignancy* in this case) may itself be an informative feature. We identify these negated features in the text of the reports using NegEx (as baseline) and the English Resource Grammar (ERG; Flickinger, 2000, 2011), as described in 3.3. Then we train a number of classifiers on the training portion of the dataset represented as feature vectors to discriminate between the classes *left* and *right*; we then evaluate on the test portion of the data. Since our dataset was not annotated for negation, we are performing an extrinsic evaluation of negation detection by showing the effect that using the grammar for feature extraction had on classification. This extrinsic evaluation can later be complemented by an intrinsic one.[4] We present the numeric results (section 4) which show small but consistent improvement of our approach over NegEx and conclude with a detailed analysis of the differences between the two systems, which helps us identify directions for future work.[5]

[1] Accessed June 5, 2018.

[2] Not to be confused with domain specific grammars, common in natural language processing.

[3] Admittedly if the training data contained lots of negated examples, the feature would not come up as one of the best for positive classification. However, negated and non-negated concepts may be unbalanced in the data, and furthermore, machine learning-based classification here is just one scenario.

[4] As discussed in the related work section, the effect of finding negation scope using a precision grammar of English has been examined by MacKinlay et al. (2012) on the sentence-level.

[5] To protect possible traces of sensitive data, the code will be available upon request.

2 Related Work

Both rule-based and machine learning approaches exist for detecting negation scope. Morante and Daelemans (2009) is an example of a machine learning approach. Their system consists of two classification tasks performed with supervised machine learning methods: finding negation tokens and classifying tokens as the first or last (or neither) token within the scope of negation. Morante and Daelemans (2009) report high scores on the BioScope dataset (Vincze et al., 2008). However, machine learning approaches like this one require sufficient annotated training data not always available for clinical tasks.

Sohn et al. (2012) use dependency parses in their rule-based system, noting that complex sentence structure is difficult to learn automatically. A similar approach is taken by Mehrabi et al. (2015). In both of these experiments, a set of rules was created specifically for the purposes of the project, and therefore the systems cannot necessarily be directly extended to work on other datasets. Furthermore, ad hoc rules are likely to be incomplete with respect to the variety of ways to express negation and the complexity of syntactic structures over which negation can scope.

Packard et al. (2014) used the English Resource Grammar (ERG; Flickinger, 2000, 2011) for negation scope detection in literary texts, to outperform the state-of-the-art at the time. In the biomedical domain, the ERG was used by MacKinlay et al. (2012). In one task described in their paper (Task 3), MacKinlay et al. (2012) constructed feature vectors for specific "events", which were already identified by annotations in their training data. They used the Minimal Recursion Semantics (MRS; Copestake et al., 2005) and Robust Minimal Recursion Semantics (RMRS; Copestake, 2003) representations produced by the ERG to identify speculation and negation over those events. Their results for negation performed best in the BIONLP 2009 shared task.

Our study differs from MacKinlay et al. (2012) in one key way: whereas they look only at events of interest (which they assume to be already identified in the data), we look at the entire text of a given pathology report, without relying on sentence-level annotation of any kind. In other words, we hypothesize that the usefulness of incorporating precise grammatical information about negation scope generalizes to the document level. Other differences include that MacKinlay et al. (2012) looked at scientific biomedical literature, while we use clinical reports as our data.

3 Experiment

The main contribution of the paper is the use of a broad coverage grammar (the ERG) for feature extraction on the level of a complete pathology report. We describe the dataset in section 3.1. The feature extraction technique is explained in detail in sections 3.3-3.5. To evaluate, we perform feature selection (section 3.2) and use four popular ensemble classifiers (section 4.1).

3.1 Data

The dataset (Table 1) comes from the Surveillance, Epidemiology and End Results (SEER) Program[6] and is a subset of 4000 randomly selected, de-identified annotated pathology reports, representing 4 registries and more than 70 pathology labs. This subset was used in a graduate level course[7] on Natural Language Processing in Cancer Informatics at the University of Washington .[8] It is annotated for a variety of things, including whether the primary tumor was found in the right or left part of the body (such as right vs. left breast or lung). It is not annotated for negation. A small number of reports were classified to have both right and left sites affected; we exclude them from the experiment in order to have a balanced dataset.

We relied primarily on the NLTK sentence tokenizer (Bird et al., 2009) to split pathology note sections into individual sentence tokens. Subtopic headers (identified by their line-ending colon punctuation) were extracted into separate sentence tokens, and mid-sentence line breaks were removed from sentence tokens. The dataset was divided into training and test randomly, using a 66% / 33% split (two thirds used for the training).

[6] https://seer.cancer.gov/

[7] LING575, Winter 2017

[8] Unfortunately, this particular dataset is not available for public use for reasons outside of authors' control. Applying the system to a public dataset remains part of future work.

	Total	LAT	RIGHT	LEFT
Training	581	446	234	212
Test	293	233	122	111

Table 1: Dataset size. LAT are reports gold-annotated for right or left laterality.

The dataset partially motivated the choice of right vs. left laterality as our target annotation: On the one hand, classifying documents for the location of the primary tumor is a useful task that could be performed by a machine; on the other, it was one of the gold annotations occurring most often in the reports. Filtering all reports for this annotation yielded a fairly large and balanced subset of documents (Table 1).

3.2 Features

We represented each report as a feature vector as follows. Features were extracted sentence by sentence and then aggregated into one set for the entire report. First we extracted all concepts for each report (as described in section 3.2.1), and then selected 44 features using recursive feature elimination with Random Forest[9] on 33% of the training dataset set and supplementing that set symmetrically, e.g. adding 'left upper lobe' where we only had 'right upper lobe'. The resulting features were: 'right breast', 'right lower lobe', 'right lung', 'right upper lobe','left breast', 'left lower lobe', 'left lung', 'left upper lobe', 'left', 'right', 'rul', 'lul', 'rll', 'lll', 'identified', 'carcinoma', 'adenocarcinoma', 'malignancy', 'malignant', 'tumor', 'in situ', 'invasive', 'infiltration', 'atypic', 'atypia', 'margin', 'rt breast', 'lt breast', 'rt lung', 'lt lung', 'rt upper lobe','lt upper lobe', 'rt lower lobe', 'lt lower lobe', 'metastasis', 'metastatic', 'found', 'suspicious', 'evidence', 'masses', 'tissue', 'specimen', 'determined', 'ruled out'. We supplement the set manually because one third of the training set used to select features was not sufficient to produce a list of features that robustly extended to the training set. For example, some of the training reports did not have the feature *right upper lobe* but we noticed that they have *left upper lobe* instead and decided to apply a symmetric manual extension to the feature set across the board. This way we hope to obtain informative features while not overfitting to the training dataset.

3.2.1 Basic Features

We constructed basic feature vectors for each report by using MetaMap Lite to recognize concepts from the UMLS term database (Aronson, 2001). Each sentence was parsed to find the longest matching UMLS concepts and phrases, using the default MetaMap Lite setting for both stop words and POS tagging. A report feature vector is the set of all the features listed above in section 3.2 that were extracted by MetaMap for each sentence in this report. We only count each feature once per report.

3.3 Negation Scope and Negated Features

We compare two different methods of extracting negated features and their effect on classification: NegEx (Chapman et al., 2001) which we consider our baseline, and the English Resource Grammar (ERG; Flickinger, 2000, 2011).[10] If a concept was marked as negated by the tool under consideration, we remove the original feature from the feature list extracted from the sentence and replace it with the negated feature (e.g. *NEG:tumor*, for a sentence like *No tumor identified*). Of course, if the same feature was detected but not marked as negated in a different sentence in the same report, the report will still have the non-negated version of the feature in its vector.

Table 2 shows the total number of feature tokens,[11] for all sentences in the dataset, and separately the number of feature tokens added by NegEx and the ERG. The ERG adds fewer features, which means it detects negation in fewer occurrences than NegEx. Later in the paper, we discuss that while sometimes

[9] We took all the top scoring features which looked meaningful, namely stopping once we saw the word 'Dr.'.

[10] Version 1214.

[11] We use the classic token/type distinction to talk about the features. For example, the sentence *Signs of malignancy in left upper lobe; no signs of malignancy in right upper lobe* contains two tokens of the feature type *lobe* and just one token of the feature type *right upper lobe*.

	total	+NegEx	+ERG
total	25159	3214	3002

Table 2: Feature tokens (features occurrences in each sentence).

it is detrimental to the evaluation results, other times features added by NegEx turn out to be noise, and skipping them is actually more precise and beneficial.

3.4 NegEx

NegEx (Chapman et al., 2001) takes text (assuming that the whole text is one sentence) as input and returns a list of concepts that it considers negated in that text. It attempts to determine what the scope of negation is, but sometimes makes mistakes. For example, in the sentence *No chest pain, no shortness of breath, and no abdominal pain*, the concept *chest pain* is affirmed (not negated) by NegEx.[12] (We will see a similar error hurting NegEx's performance in section 4.3.3.) While any specific error can be fixed, it can be difficult to fix such issues robustly and not cause regressions without relying on a general set of linguistically motivated language rules. When this method of negation detection was applied to feature extraction, the list of feature types was appended with *NEG:in-situ*, *NEG:left breast*, *NEG:left*, *NEG:identified*, *NEG:carcinoma*, *NEG:malignancy*, *NEG:margin*, *NEG:tumor*, *NEG:right*, *NEG:invasive*.

3.5 ERG

3.5.1 Overview

The English Resource Grammar (ERG; Flickinger, 2000, 2011) is a large, broad-coverage precision grammar of English.[13] The term 'precision' means that it encodes syntactic and lexical rules in a form that aims for linguistic adequacy and generality, in this case using the Head Driven Phrase Structure Grammar theory of syntax (HPSG; Pollard and Sag, 1994). The grammar maps surface strings to syntactic as well as semantic representations which are licensed by lexical and phrase structure rules. The ERG is supported by the DELPH-IN research consortium, along with a variety of tools for parsing, generation, and representing syntactic and semantic structures.[14] It is the largest HPSG-based grammar, but grammars of different sizes exist for other languages.[15]

3.5.2 Minimal Recursion Semantics

The ERG produces semantic representations compositionally in the format of Minimal Recursion Semantics (Copestake et al., 2005), which we then use to extract negated features. Figure 1 shows a syntactic structure (left) and an MRS (right) for the fragment sentence *No evidence of malignancy*. The syntactic representation is a familiar constituency tree. The flat MRS representation is a bag of quantifiers, relations, and predicates associated with handles (such as *h13*), which can be identified with each other or not, allowing for scope underspecification.[16] In this example, *evidence* is analyzed is being in the scope of the *no* quantifier. The word *malignancy* is analyzed as the first argument of *evidence* (through the indefinite quantifier, unexpressed on the surface level). The predicate of this sentence is *unknown*, reflecting that it is a fragment.

A dependency MRS for the sentence *A tumor was not identified* is shown in Figure 2. DMRS representations (Copestake, 2009; Copestake et al., 2016) are essentially a simplified version of MRS where variables and their relationships are expressed as links. DMRS representations consist of elementary predications (EPs) corresponding to surface tokens, such as the EP for negation, as well as abstract EPs contributed by grammatical rules. The EPs are connected by links, forming a semantic dependency

[12] We used the version downloaded from `https://github.com/chapmanbe/negex`.

[13] Demo: `http://erg.delph-in.net/logon`.

[14] To follow up on the earlier example of NegEx making a mistake with the sentence *No chest pain, no shortness of breath, and no abdominal pain*, we verified that the ERG returns a syntactic structure which has all three nouns in scope of negation.

[15] See e.g. `http://www.delph-in.net/wiki/index.php/Grammars`.

[16] See Copestake et al. (2005) for a discussion and examples of scope underspecification and other issues related to MRS.

graph.[17] DMRS representations are related to MRS (a DMRS can be created from an MRS deterministically) but are easier to look at and to work with, and for our experiment we used the DMRS representations.

One advantage of semantic representations like MRS or DMRS is that they abstract away from the syntactic notions of subjects and objects, normalizing active/passive pairs to the same representation. This is illustrated in Figure 2, where *tumor* is the ARG2 (the theme) of *identify*, despite being the grammatical subject of a passive sentence. Another advantage is a systematic treatment of scope, which is discussed in the next section.

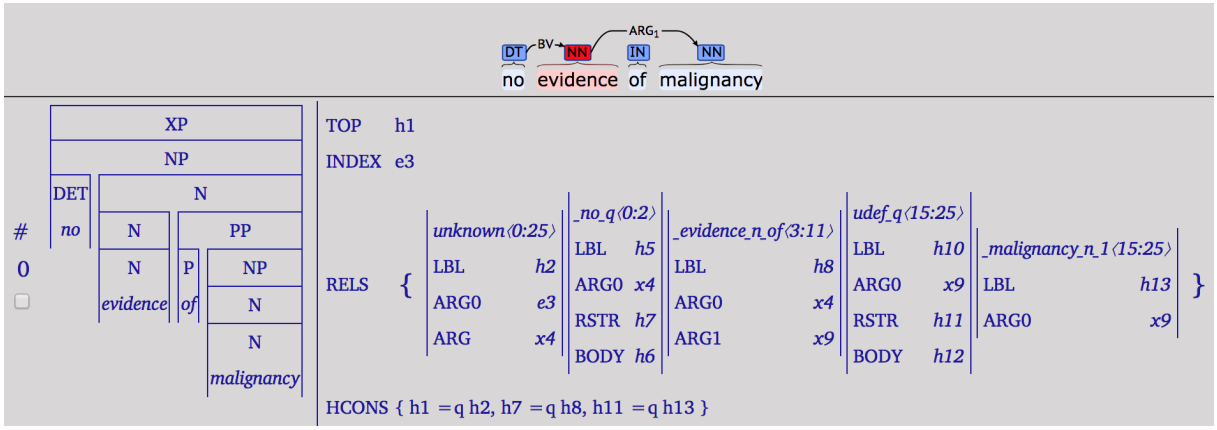


Figure 1: The ERG's syntactic parse and the MRS of the fragment sentence: *No evidence of malignancy*

3.5.3 Extracting Features with the ERG

We parsed every sentence in each report with the ERG loaded into the ACE parser (Crysmann and Packard, 2012).[18] ACE has a default parse ranking procedure trained on a treebank (Toutanova et al., 2005), and we selected the top ranked parse. The coverage of the parser was 22858/28788 sentences (79%). It is not surprising that not all sentences are parsed, given the highly fragmented nature of the reports, where newlines are often used to separate different parts of the same sentence. Statistical parsers always return a parse; however, we will show that when striving for precision, no parse can be better than a nonsensical parse.[19] Once we parsed each sentence using the ERG, we crawled the dependency (DMRS) representations to extract negated concepts. We extracted two types of negation: predicate negation (the negation of events) and quantifier negation (the negation of entities).

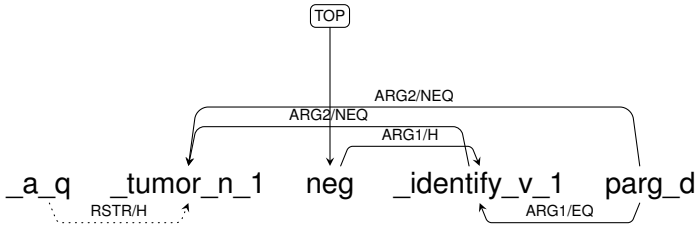


Figure 2: A semantic dependency graph (DMRS) produced by the ERG for: *A tumor was not identified.*

Predicate negation is represented with the EP *neg* (the semantic contribution of *not*) which scopes over the predications contributed by the verb (*neg*'s ARG1) and the verb's own arguments. From the DMRS we extract the verb as well as its argument labeled ARG2. This label refers to the argument receiving the action of the verb, or the semantic theme. For example, in *Skin: invasive carcinoma does not invade dermis or epidermis*, we extract the verb *invade* and its ARG2 which is the the coordinated

[17] The DMRS graphs were produced using `https://github.com/delph-in/delphin-viz`.

[18] Version 0.9.24.

[19] That said, a more sophisticated tokenization technique, which would reconstruct more full sentences, would improve ACE's coverage.

noun phrase *dermis or epidermis*.[20] Extracting the ARG2 yields the same result in passive sentences, such as *Additional distinct breast masses or lesions are not grossly identified*, where the 'things not identified' are the *additional distinct breast masses or lesions*.

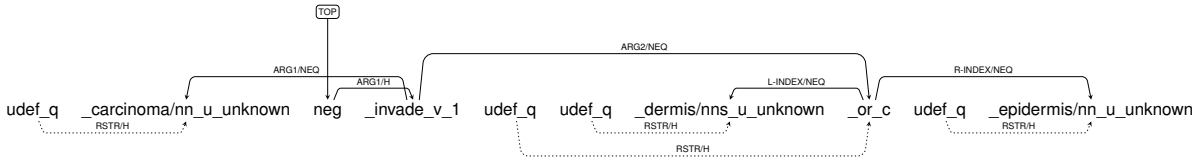


Figure 3: DMRS for the sentence *Carcinoma does not invade dermis or epidermis*

Quantifier negation is the negation of a noun phrase by the quantifier *no*, as in *There is no evidence of vasculitis, granulomatous inflammation, or neoplasm*. First we identify the predications restricted by the quantifier *no*. Here the restricted predication is the noun phrase headed by *evidence*, which itself contains a prepositional phrase with coordinated noun phrases. We identify each concept in the noun phrase and mark it as negated.

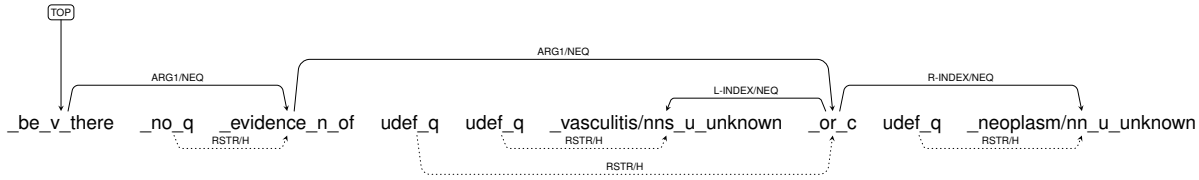


Figure 4: DMRS for the sentence *There is no evidence of vasculitis or neoplasm*

When this negation detection method is applied to feature extraction, the original list of features is appended by *NEG:tumor*, *NEG:identified*, *NEG:carcinoma*, *NEG:malignancy*, *NEG:margin*, *NEG:in situ*, *NEG:evidence*. Compared to NegEx, this is fewer feature types as well as fewer feature tokens added. As we will see, this may lead to higher precision in classification. On the other hand, the lack of highly informative concepts related directly to *left* and *right* (such as *left lung*) indicates we could get better results if we supplied the ERG with more data that it can successfully parse. There are certainly sentences in the data in which the concepts like this are within the scope of negation. As we will see, NegEx sometimes gets it right and other times it does not. The reason the ERG does not detect them is usually that it was not supplied a well-formed sentence due to tokenization issues.

4 Results and Error Analysis

4.1 Evaluation by Classification

To test our feature extraction technique, we used four popular ensemble algorithms which generally perform well on small datasets, taking advantage of the scikit-learn Python library (Pedregosa et al., 2011). The parameters were chosen by cross-validation on a small development set.

classifier	code	cite	# estimators	learn. rate	max. depth	other
AdaBoost	AB	Freund and Schapire (1995)	50	1	-	SAMME.R
Random Forest	RF	Breiman (2001)	100	-	20	-
Gradient Boost	GB	Friedman (2001)	100	0.1	1	-
Voting classifier	VC	Pedregosa et al. (2011)	-	-	-	hard, no weights

Table 3: Classifiers used for evaluation

4.2 Numeric Results

Table 4 shows the results. The first column is the ensemble classifier code (4.1); *right* and *left* are laterality classes. Incorporating information about negated concepts is beneficial in many cases; the

[20] An additional trivial step is required to extract *dermis* and *epidermins* as separate concepts

CL	NE					ERG				
	precision		recall		micro-avg F1	precision		recall		micro-avg F1
	right	left	right	left		right	left	right	left	
AB	0.9754	0.9459	0.9520	0.9722	0.9614	**0.9917**	**0.9554**	**0.9600**	**0.9907**	**0.9742**
RF	0.9760	**0.9722**	**0.9760**	0.9722	**0.9742**	**0.9835**	0.9464	0.9520	**0.9815**	0.9657
GB	0.9835	0.9464	0.9520	0.9815	0.9657	**0.9836**	**0.9550**	**0.9600**	0.9815	**0.9700**
VC	0.9756	0.9600	0.9545	0.9722	0.9657	**0.9917**	0.9600	**0.9554**	**0.9907**	**0.9742**

Table 4: Classification results with four different classification algorithms; NegEx vs. ERG.

improvement is small but fairly robust across different ensemble algorithms. The only algorithm where NegEx-detected negation was better than the ERG is Random Forest, and even for Random Forest, the precision is higher with the ERG for one of the classes and the recall is higher for the other.

4.3 Improvement and Error Analysis

Below we will look at specific improvements and regressions, per record. The record ID has been changed for anonymity concerns. We primarily see differences in classification with respect to 7 records; we will call them Records 1-7.

4.3.1 AdaBoost

Improvement: Records 1,2,3. The feature vectors for **Records 1 and 3** do not differ between the ERG and NegEx; Record 1 does not contain any negated features in either case, while Record 3 contains a feature negated by both the ERG and NegEx. This means the improvement is probably due to chance (a tie broken by the classifier in such a way that it shows as an improvement). **Record 2**, however, shows some differences. The features *identified* and *metastatic* were negated by NegEx but not by the ERG. These features and their negated counterparts may have a lot of weight and so an incorrect decision about negation can have impact. It turns out, NegEx uses a fairly big chunk of text with no sentence-final punctuation which happens to start from *Not indentified* to produce multiple negated features. The ERG is not able to parse that chunk of text as a sentence (since it really is not one). In this case, better precision leads to a better result.
Regression: None.

4.3.2 Random Forest

Improvement: Record 1, again can be attributed to chance.
Regression: Records 4, 5, 6. In **Record 4**, the feature *identified* is correctly negated by NegEx but not by the ERG. The source sentence has the form *No X or Y is identified*. The ERG only succeeds in negating X and Y but not the predicate *identified*. We discuss this issue in section 4.4. **Record 5** contains a phrase *negative for malignancy*, for which NegEx has a suitable heuristic. Our ERG algorithm is not sensitive to the word *negative*. We purposefully did not enhance our algorithm with any heuristics since we wanted our approach to be as domain-independent as possible. It makes sense that NegEx beats us here, however one direction for improvement is to map some of NegEx's heuristics to the ERG's semantic relations. NegEx negates multiple features in **Record 6**, including both *right* and *left*, *left breast*, (but not *right breast*) *identified*, *carcinoma*, *tumor*. The ERG does not negate any of them in this case. The correct label for this record's laterality is *right*, so NegEx probably wins by negating many of the features associated with *left*. However, it is hard to say why it chooses to do so; for example, it negates *left breast* in the sentence: *Left breast: According to the ultrasound core biopsy protocol data sheet, mammogram demonstrates a 2.0 cm irregular mass, and correlates as palpated.* This Record does not provide us with any insight with respect to the two systems' different behavior.

4.3.3 Gradient Boosting

Improvement: Record 7. For this record, the difference is that the ERG negates the features *evidence* and *tumor* while NegEx does not. In this case, this leads to a better result. The relevant text in the record is *no evidence of ductal carcinoma in situ*. NegEx detects negation for *ductal carcinoma in situ*, but not for *evidence*.

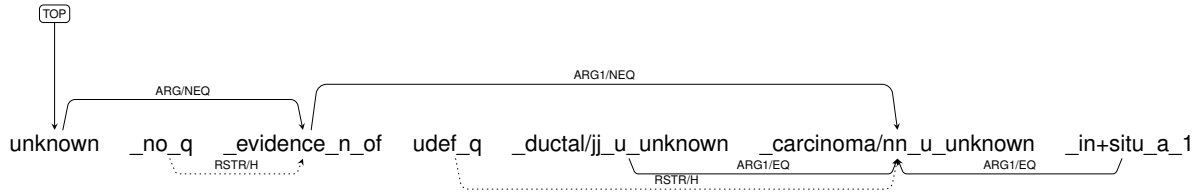


Figure 5: DMRS for *No evidence of ductal carcinoma in situ*

Regression: None.

4.3.4 Voting Classifier

Improvement: Records 1, 2, already analyzed for AdaBoost.
Regression: None.

4.4 Summary

The Improvement and error analysis revealed the following differences between the NegEx and the ERG approaches to feature extraction.

1. The two systems are not equally sensitive to **tokenization issues**. Clinical notes contain lots of tabs in place of sentence-final punctuation; that may lead to imprecise tokenization. NegEx does not care about whether its input actually is a sentence, while the ERG does; our ERG algorithm thus produces higher precision results in some cases where NegEx can negate a feature based on a trigger which does not actually belong to the sentence that contains this feature.

2. NegEx employs a variety of **domain-specific heuristics** while our ERG algorithm does not. We would like our approach to not include dataset or domain-specific heuristics, but some of the heuristics could be mapped to the ERG's existing semantic relations such as specific English quantifiers. We could take advantage of using a robust and precise domain-independent resource and apply a heuristic at a higher level, such as parse selection.

3. ACE **parse selection**. The treebanks used to train the ACE parse ranker are not necessarily representative of the data found in pathology reports. Therefore, it was possible for the desired parse to be produced by the ERG, but not be selected. In some cases, the top ranked parse did not exhibit the widest scope interpretation. For example, the top ranked parse for *no masses or previous biopsy sites are identified* only scoped *no* over *masses*, and not the entire coordinated phrase. We also find

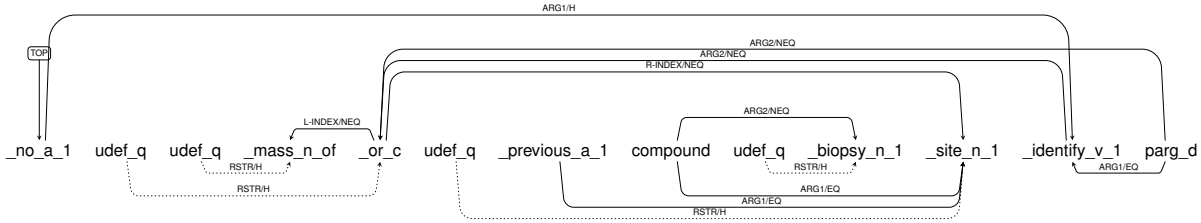


Figure 6: DMRS showing the desired negation scope for *No masses or previous biopsy sites are identified*; this DMRS was not the top-ranked parse

some inconsistency in the top ranked parse for constructions like *associated ductal carcinoma in situ: not identified*. This 'subject: adjective' construction was probably not common in the dataset used to train the parse selection model distributed with the ERG, version 1214, but is frequent in our dataset. In some sentences like the one above (illustrated also by Figure 8), *not* is parsed as a conjunction, or the phrase after the colon is treated as a modifier (effectively treating *identified* as a post-head adjective) rather than as a predicate (negated verb). In these cases, our algorithm, which looks for a *neg* predication and a verb with a theme, fails to negate *identified*. Customized parse

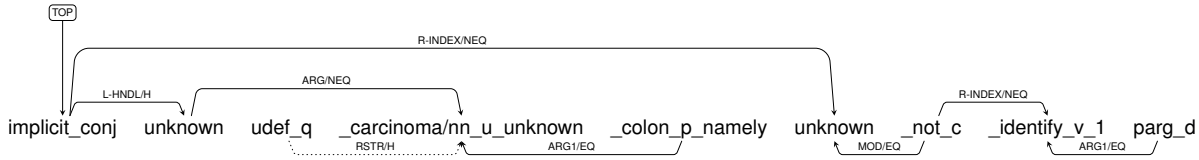


Figure 7: Abridged DMRS showing modifier analysis for *not* associated with some top-ranked parses

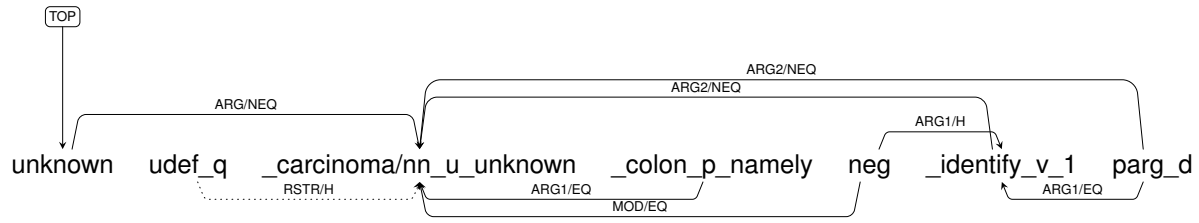


Figure 8: DMRS showing desired, predicate negation analysis for *not*

selection that prefers verbal predicates over modifiers and prefers the widest scope may help extract the features even better and further improve classification.

4. Finally, in some cases it is simply not clear why one approach wins over the other. This includes situations where there is a tie or when there is some difference between the feature vectors produced by the two approaches, but the difference seems to be due to a bug in NegEx (nonetheless leading to NegEx producing a better feature vector). We hope that we will gain more insight once we switch to a larger dataset.

5 Conclusion

We show as a proof of concept that a precision grammar can help extract potentially informative negated features for a pathology reports classification task, outperforming NegEx across several ensemble classification algorithms. Improvement and error analysis reveals that the ERG, being a *precision* grammar, tends to incorporate fewer noisy features compared to NegEx. Because it looks for a meaningful structure in what it expects to be a well-formed sentence, it will make fewer mistakes in terms of negation scope. At the same time, at this stage we did not go beyond simple predicate negation and did not customize parse selection, which causes our system to make some mistakes that NegEx, which includes multiple heuristics specialized for the domain, does not make. In future work, we will experiment with ways to customize parse selection so that it is better suited for the clinical notes domain, while not needing to change anything in the grammar itself. We will also explore which relations in the ERG may correspond to some of NegEx's heuristics, and this should give us further improvement. Furthermore, in any next stages we will be applying our system to a public dataset such as BioScope (Vincze et al., 2008).

Acknowledgements

This project would not be possible without Emily Silgard from Fred Hutch who provided us with the data and guidance for this project. We thank Michael W. Goodman and Ned Letcher for their delphin-viz tool that made it painless to include the DMRS representations.

References

Alan R Aronson. 2001. Effective mapping of biomedical text to the UMLS metathesaurus: the MetaMap program. In *Proceedings of the AMIA Symposium*, page 17. American Medical Informatics Association.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: Analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Wendy W Chapman, Will Bridewell, Paul Hanbury, Gregory F Cooper, and Bruce G Buchanan. 2001. A simple algorithm for identifying negated findings and diseases in discharge summaries. *Journal of biomedical informatics*, 34(5):301–310.

Ann Copestake. 2003. Report on the design of RMRS. *DeepThought project deliverable*.

Ann Copestake. 2009. Slacker semantics: Why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1–9. Association for Computational Linguistics.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3(2-3):281–332.

Ann Copestake, Guy Emerson, Michael Wayne Goodman, Matic Horvat, Alexander Kuhnle, and Ewa Muszyńska. May 2016. Resources for building applications with dependency minimal recursion semantics. In *Proceedings of 10th International Conference on Language Resources and Evaluation*. URL `http://www.lrec-conf.org/proceedings/lrec2016/pdf/634_Paper.pdf`.

Berthold Crysmann and Woodley Packard. 2012. Towards efficient HPSG generation for German, a non-configurational language. In *COLING*, pages 695–710.

Dan Flickinger. 2000. On building a more effcient grammar by exploiting types. *Natural Language Engineering*, 6(01):15–28.

Dan Flickinger. 2011. Accuracy vs. robustness in grammar engineering. *Language from a cognitive perspective: Grammar, usage, and processing*, pages 31–50.

Yoav Freund and Robert E Schapire. 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer.

Jerome H Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Andrew MacKinlay, David Martinez, and Timothy Baldwin. 2012. Detecting modification of biomedical events using a deep parsing approach. *BMC medical informatics and decision making*, 12(1):S4.

Saeed Mehrabi, Anand Krishnan, Sunghwan Sohn, Alexandra M Roch, Heidi Schmidt, Joe Kesterson, Chris Beesley, Paul Dexter, C Max Schmidt, Hongfang Liu, et al. 2015. DEEPEN: A negation detection system for clinical text incorporating dependency relation into negex. *Journal of biomedical informatics*, 54:213–219.

Roser Morante and Walter Daelemans. 2009. A metalearning approach to processing the scope of negation. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 21–29. Association for Computational Linguistics. URL `http://aclweb.org/anthology/W09-1105`.

Woodley Packard, M. Emily Bender, Jonathon Read, Stephan Oepen, and Rebecca Dridan. 2014. Simple negation scope resolution through deep parsing: A semantic solution to a semantic problem. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 69–78. Association for Computational Linguistics. doi: 10.3115/v1/P14-1007. URL `http://aclweb.org/anthology/P14-1007`.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. The University of Chicago Press and CSLI Publications, Chicago, IL and Stanford, CA.

Guergana K Savova, James J Masanz, Philip V Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C Kipper-Schuler, and Christopher G Chute. 2010. Mayo clinical text analysis and knowledge extraction system

(cTAKES): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association*, 17(5):507–513.

Sunghwan Sohn, Stephen Wu, and Christopher G Chute. 2012. Dependency parser-based negation detection in clinical narratives. *AMIA Summits on Translational Science proceedings AMIA Summit on Translational Science*, 2012:1–8.

Kristina Toutanova, Christopher D Manning, Dan Flickinger, and Stephan Oepen. 2005. Stochastic HPSG parse disambiguation using the Redwoods corpus. *Research on Language & Computation*, 3 (1):83–105.

Veronika Vincze, György Szarvas, Richárd Farkas, György Móra, and János Csirik. 2008. The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC bioinformatics*, 9 (11):S9.

Bridge Video and Text with Cascade Syntactic Structure

Guolong Wang, Zheng Qin, Kaiping Xu, Kai Huang and **Shuxiong Ye**
Tsinghua University
wanggl16@mails.tsinghua.edu.cn, qingzh@tsinghua.edu.cn
{xkp13, huang-k15, ysx15}@mails.tsinghua.edu.cn

Abstract

We present a video captioning approach that encodes features by progressively completing syntactic structure (*LSTM-CSS*). To construct basic syntactic structure (i.e., subject, predicate, and object), we use a Conditional Random Field to label semantic representations (i.e., motions, objects). We argue that in order to improve the comprehensiveness of the description, the local features within object regions can be used to generate complementary syntactic elements (e.g., attribute, adverbial). Inspired by redundancy of human receptors, we utilize a Region Proposal Network to focus on the object regions. To model the final temporal dynamics, Recurrent Neural Network with Path Embeddings is adopted. We demonstrate the effectiveness of *LSTM-CSS* on generating natural sentences: 42.3% and 28.5% in terms of BLEU@4 and METEOR. Superior performance when compared to state-of-the-art methods are reported on a large video description dataset (i.e., MSR-VTT-2016).

1 Introduction

Video has become a ubiquitous way of communication on the Internet, podcast channels, as well as mobile devices. Accelerated by the explosive spread of video data, automatic analysis of semantic video content remains a promising area. The advances of this task can provide comparatively favorable preconditions for subsequent tasks, such as video retrieval, human-perception analysis, keyframe recommendation(Chen et al., 2017). To encapsulate the informative dynamics in the video, researchers have started focusing on recognizing videos based on the predefined templates, such as (Kojima et al., 2002; Guadarrama et al., 2014). Rohrbach et al. (2013) learned a CRF to assemble between different components of the input video and generated descriptions for videos. Xu et al. (2015) utilized a unified framework that jointly models video and the corresponding text sentences.

Another line of work uses Recurrent Neural Network. In light of its success, the representations (e.g. key objects, locations, motions, and scenes) can be concatenated into an input sequence and then translated to a natural sentence (Donahue et al., 2015). Follow-up works investigate the modeling of not only video contents and their spatio-temporal relationships, but also the syntactical structure (Venugopalan et al., 2015). More recently, a visual-semantic embedding learning technique has been proposed to model video content and textual semantics as a regularizer in Long Short-Term Memory architecture (Pan et al., 2016). This was extended by (Hori et al., 2017) where they utilized a modality-dependent attention mechanism.

Over the past years, researchers have studied multiple strategies to effectively bridge the visual content and textual description based on some fresh ideas. (Pan et al., 2017) incorporated the transferred semantic attributes learned from images and videos into the *CNN plus RNN* framework. (Baraldi et al., 2017) proposed a novel LSTM cell which can modify the temporal connections of the encoding layer according to the identified discontinuity points among frames. (Kaufman et al., 2017) transferred the semantics of the selected reference clips to test clips, which keeps consistent and maintains temporal coherence. However, the existing methods fail to take advantage of the local constraints which can extract compressed features for generating complementary syntactic elements.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3576–3585
Santa Fe, New Mexico, USA, August 20-26, 2018.

Input video:

Output Sentence:

- LSTM-CSS [ours]: a fashion show is going on the runway.
- Human: a woman is modelling clothes. / a fashion show with women walking. / fashion show runway walk involving several models.

Figure 1: Examples of video description generation

This paper proposes a novel deep architecture, named Cascade Syntactic Structure (CSS), which takes advantages of incorporating global representations and local object regions into sequence learning for video captioning. Take the given video in Figure 1 as an example, the object regions of interest can be extracted to depict primary objects with locations (e.g., "woman") while motions from global features convey the temporal dynamics (e.g., "walking", "showing"). As "fashion show" and "runway" are not in the motion and object candidate set, we believe it is the local features that convey this complementary information. This has facilitated video captioning to disentangle the global features (motions and objects) and local features (object regions) enabling independent specification of both within the generation process. Our exploration of object regions as a modality for video captioning complements recent advances. Concretely, we propose a hierarchical *CNN plus RNN* architecture to learn a low-dimensional joint embedding for global and local features. The Convolutional Neural Network (*CNN*) has two discriminatively trained streams, motion stream and object stream. The outputs of these two streams are motions and objects respectively, which are used by Conditional Random Field (*CRF*) (Lafferty et al., 2001) to formulate the basic syntactic structure. The Long Short-Term Memory Network, together with Path embeddings (i.e., *LSTM-PE*) is used to generate a final sentence with optimal semantic structure.

As the final performance of our description generation relies on the accuracy of motion classification and object detection, we argue that the two stream *CNN* needs to be discriminative enough to provide information for following sub-structures. Its output global representation (i.e., a triplet of motion and objects) is used to construct basic syntactic structure which determines the comprehensibility of a sentence. As actions in MSR-VTT-2016 dataset (Xu et al., 2016) have a small range of movement and 3D convolutional neural networks (*C3D*) (Tran et al., 2015) performs not well enough on these actions, we prefer to use Temporal Segment Networks (*TSN*) (Wang et al., 2016) to extract motion features with additional optical flows (Z. et al., 2017). A *CRF* model is then used to learn the optimal global representation which is sent to a Long Short-term Memory (*LSTM*) network. Inspired by object masks proposed by Mask R-CNN (He et al., 2017), we believe that local constraints (i.e. object regions) can contribute to generating more comprehensive information for the result. We apply region-based network to our object stream and the output local constraints will be directly sent to aforementioned *LSTM* for generating the final sentence. The path embeddings can model the relationship between word vectors and contribute to the prediction of next word. This will help our model to cope with some recursive structures (e.g., nested conjunctions). Its efficacy is analyzed in section 2. Given a video, our framework can generate a sentence invariant to recursive structures based on both global representations and local constraints extracted from the video.

Our main technical contributions are three-fold:

- We propose a multi-task convnet to learn local embedding and global embedding for objects, which outperforms, by a large margin, previous attempts that use deep convnets for learning only global features.

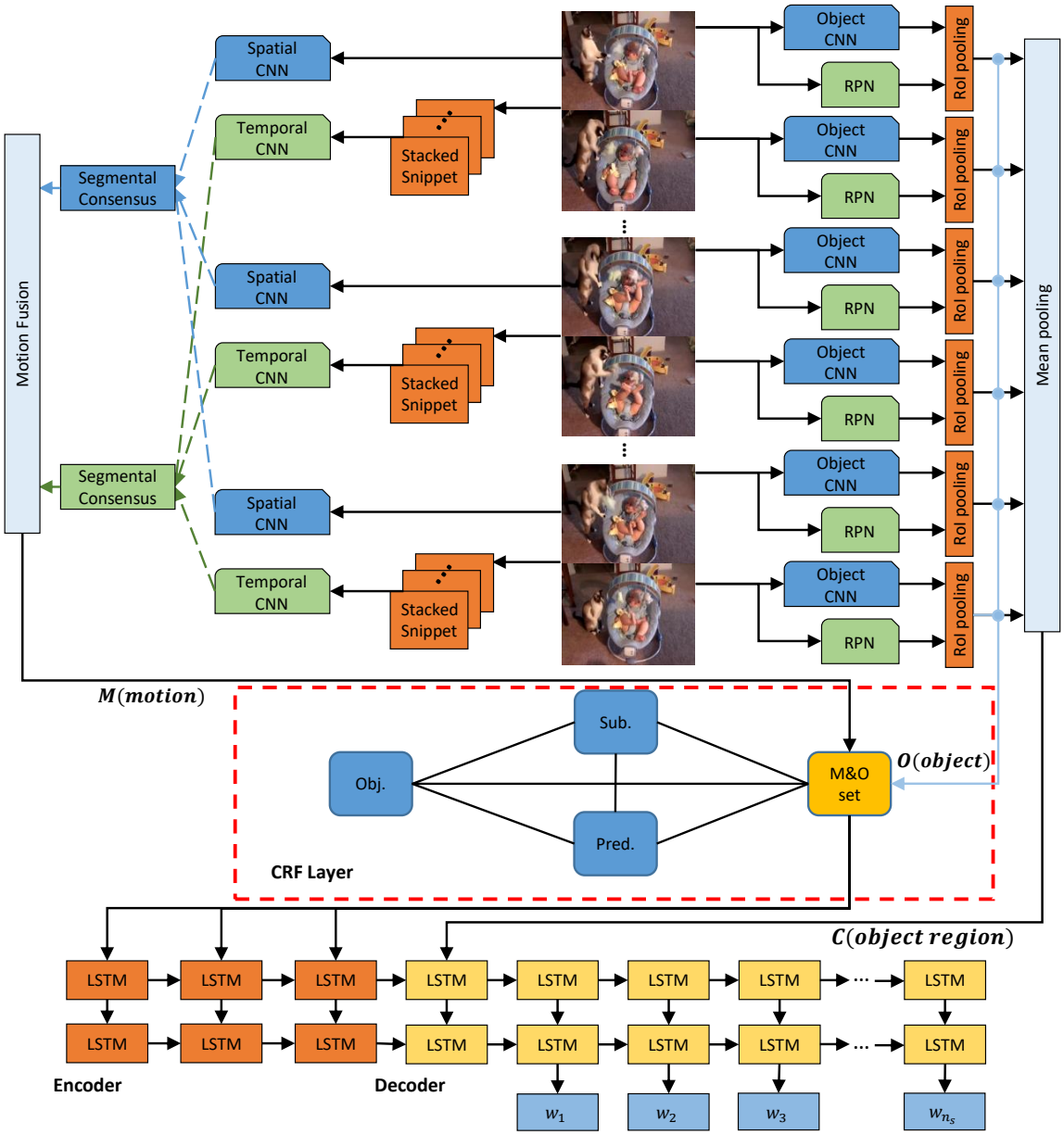


Figure 2: Hierarchical network combining global vectors (motions and objects) and local constraints (object regions) to generate video description with CRF and LSTM. Each sub-structure is trained independently (Section 2). M&O set denotes the semantic representation of motions and objects. Sub, Pred, and Obj denotes subject, predicate, and Object respectively in CRF.

- We empirically show the advantage of our unified video description generation framework whereby local constraints can be used to generate complementary syntactic elements, which can in turn improve the comprehensiveness of the description.
- We show that giving too much information to *LSTM* decoder could lead to chaos in the final structure. Hence, our method also input path embeddings to the decoder as context.

2 Video Captioning with Cascade Syntactic Structure

The goal of our network is to generate a sentence that describes the contents and their relationships. The problem is formulated as follows: given a video $\mathbf{V}$ with n_v segmented clips, our objective is to describe it by a textual sentence $\mathbf{S}$ with n_s words noted as $\mathbf{S} = \{\mathbf{w}_1, \cdots, \mathbf{w}_{n_s}\}$ with each word in it as

its column vector. $\mathbf{M} = \{\mathbf{m}_1, \cdots, \mathbf{m}_{n_m}\}$ and $\mathbf{O} = \{\mathbf{o}_1, \cdots, \mathbf{o}_{n_o}\}$ respectively denote the candidate of motions (resp. objects). $\mathbf{m}_i$ and $\mathbf{o}_i$ are represented by a one-hot vector. n_m and n_o refer to the size of the candidates. The local constraints $\mathbf{C}$ are D_c-dimensional local features, which are the mean pooling of selected frames in a video.

In the remaining of this section, we describe our network architecture and training methodology for generating a description for a specific video with local constraints. We first present two sub-structures of our architecture (Figure 2): 1) Feature extraction network (Section 2.1), which unify two discriminatively trained network streams that independently learn a motion (resp. object) feature embedding for a video; 2) Sentence generation network (Section 2.2), which uses CRF to learn optimal semantic representation and LSTM to generate the final description.

2.1 Feature extraction network

2.1.1 Motion stream

For each clip, the motion stream incorporates a Spatial CNN accepting a frame of RGB image and a Temporal CNN accepting a short snippet with 10 optical flow fields (5 for x directions and 5 for y directions) stacked in channel dimension extracted by warped TVL1 (Wang and Schmid, 2014). The frame sent to the Spatial CNN is randomly selected from a specific segment. The optical flow fields sent to the Temporal CNN are randomly selected from those computed for arbitrary two consecutive frames in each segment. Owing to the uncertain range of optical flow displacement, normalization is applied to constraint the displacements in $[0, 255]$ which is same as gray image. Thus, an optical flow field can be regarded as an image with 10 channels in the motion stream. The output of the motion stream, motion embedding $\mathbf{M}$, is averaged over all segments.

2.1.2 Object stream with local constraints

In conventional methods (Venugopalan et al., 2015), only embedding of motions and objects are extracted for LSTM. We believe that if visual features are sent to the LSTM, it is more generative to learn sentences for videos. In the human retina, visual features which represent natural signals formed by the peripheral receptors are very high-dimensional. However, we argue that the receptors to discover where and what objects only occupy a small fraction of the space of all possible receptor activation due to the statistical regularity and redundancy (Burton and Moorhead, 1987). We choose **ROI pooling** (Girshick, 2015) to extract local features specifically for the object regions as our local constraints.

As the object stream is the main sub-structure in our model, a careful training protocol is required to ensure convergence. For each clip, two images are randomly selected. For each input image, we train a network with aforementioned local constraints. The Object CNN can use many networks with the output discrete probability distribution $p = (p_0, \cdots, p_k, \cdots, p_{K-1})$, indexed by k. It is computed by a softmax layer over K object classes. The RPN network is a stack of convolutional filters of different sizes. Its outputs are the offsets of object regions, $r^k = (r^k_x, r^k_y, r^k_w, r^k_h)$, for the kth object class. The network can be described with a function $f(\cdot)$ minimising:

$$L = m + \sum_i L_{cls} + \lambda \sum_i L_{loc} \tag{1}$$

where m is a margin promoting convergence and i is the index of object region. The confidence loss L_{cls} is log loss for true class for each region. The location loss L_{loc} is the Smooth L1 loss (Girshick, 2015) between the predicted region and the ground truth region. Local constraint $\mathbf{C}$ (features in regions) and object embedding $\mathbf{O}$ are the output of object stream. Use of local Constraints is later shown to yield significant performance gain (Section 3.4.1).

2.2 Sentence generation

2.2.1 CRF: basic syntactic structure

The sentence generation network comprises two parts: CRF and LSTM. For CRF model, given $\mathbf{z} = \mathbf{M} \cup \mathbf{O}$ as input, let $\mathbf{y} = \{y_1, \cdots, y_{n_g}\}, n_g = n_m + n_s$ represents the label sequence of them, use the

following standard energy formulation:

$$E(\mathbf{y}, \mathbf{z}) = \sum_{i=1}^{n_g} E^u(y_i; \mathbf{z}) + \sum_{i,j} E^p(y_i, y_j) \tag{2}$$

where the first term defines the sum of quadratic unary terms and the second term describes the relationship between pairs of (y_i, y_j) (Hu et al., 2016).

Corresponding to our predicted $\mathbf{y}^*$, we rearrange $\mathbf{z}$ for optimal semantic representation $\mathbf{z}^*$. It is sent to the following LSTM model as well as local constraints $\mathbf{C}$. The LSTM model is based on an encoder-decoder framework (Sutskever et al., 2014). The encoder computes a intermediate representation $\mathbf{e}$ for the input semantic representation $\mathbf{z}^*$. Based on encoded intermediate representation, the decoder generates a translation, one target word at a time. The $\mathbf{S}$ is the output of the decoder. The log conditional probability is as follows:

$$\log\ p(\mathbf{S}|\mathbf{z}^*) = \sum_{t=1}^{n_s} \log\ p(\mathbf{w}_t|\mathbf{w}_{<t}, \mathbf{e}) \tag{3}$$

The objective is formulated as:

$$L = \sum_{(\mathbf{z}^*, \mathbf{S}) \in \mathbb{D}} -\log\ p(\mathbf{S}|\mathbf{z}^*) \tag{4}$$

where $\mathbb{D}$ refers to the parallel training corpus of source and target representation pairs $(\mathbf{z}^*, \mathbf{S})$.

2.2.2 LSTM-PE: final syntactic structure

The encoder and decoder share a common LSTM network with similar forward and backpropagation process. The difference is the decoder starts from the intermediate representation rather than zero states. In addition to the intermediate representation, the local constraints $\mathbf{C}$ are injected at the initial time of the decoder to inform the whole memory cells in LSTM. For timestep t, $\mathbf{x}_t$, $\mathbf{y}_t$ and $\mathbf{h}_t$ are the input vector, output vector, and hidden state respectively.

$$\begin{aligned}
&\mathbf{g}_t = \phi(\mathbf{U}_g\mathbf{x}_t + \mathbf{W}_g\mathbf{h}_{t-1} + \mathbf{b}_g), \mathbf{i}_t = \sigma(\mathbf{U}_i\mathbf{x}_t + \mathbf{W}_i\mathbf{h}_{t-1} + \mathbf{b}_i),\\
&\mathbf{f}_t = \sigma(\mathbf{U}_f\mathbf{x}_t + \mathbf{W}_f\mathbf{h}_{t-1} + \mathbf{b}_f), \mathbf{c}_t = \mathbf{g}_t \odot \mathbf{i}_t + \mathbf{c}_{t-1} \odot \mathbf{f}_t,\\
&\mathbf{o}_t = \sigma(\mathbf{U}_o\mathbf{x}_t + \mathbf{W}_o\mathbf{h}_{t-1} + \mathbf{b}_o), \mathbf{h}_t = \phi(\mathbf{c}_t) \odot \mathbf{o}_t,
\end{aligned} \tag{5}$$

where $\mathbf{g}_t$, $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{c}_t$, $\mathbf{o}_t$, and $\mathbf{h}_t$ are cell input, input gate, forget gate, cell state, output gate, and cell output of the LSTM. σ is the element-wise sigmoid function and $\odot$ is the element-wise product. $\mathbf{U}$ are the weight matrices of different gates for input $\mathbf{x}_t$, $\mathbf{W}$ are the recurrent weight matrices for hidden state $\mathbf{h}_t$, and $\mathbf{b}$ are bias vectors.

To make full use of aforementioned optimal label sequence $\mathbf{y}^*$, we learn the path embeddings from sequences (Chen and Manning, 2014) and use them as context (Jiang Guo and Xu, 2016) for decoder. We pre-train path embeddings together with word embeddings. At the initial time, the path embeddings stand for generic dependencies between labels (i.e., subject, predicate, and object). The path embeddings are then enlarged and refined during the learning phase. Let $\mathbf{P}_t$ denote the path embeddings, which is the concatenation of vectors representing labels:

$$\begin{aligned}
&\mathbf{p}_{-1} = [\mathbf{y}^*],\\
&\mathbf{p}_t = [\mathbf{p}_{t-1}\mathbf{y}_{t-1}],
\end{aligned} \tag{6}$$

The LSTM updating procedure of decoder is as:

$$\begin{aligned}
&\mathbf{x}_{-1} = \mathbf{U}_C\mathbf{C},\\
&\mathbf{x}_t = \mathbf{U}_s\mathbf{e},\\
&\mathbf{h}_t = f(\mathbf{W}_i\mathbf{x}_t + \mathbf{W}_h\mathbf{h}_{t-1} + \mathbf{W}_p\mathbf{P}_t),\\
&\mathbf{y}_t = softmax(\mathbf{W}_o\mathbf{h}_t),
\end{aligned} \tag{7}$$

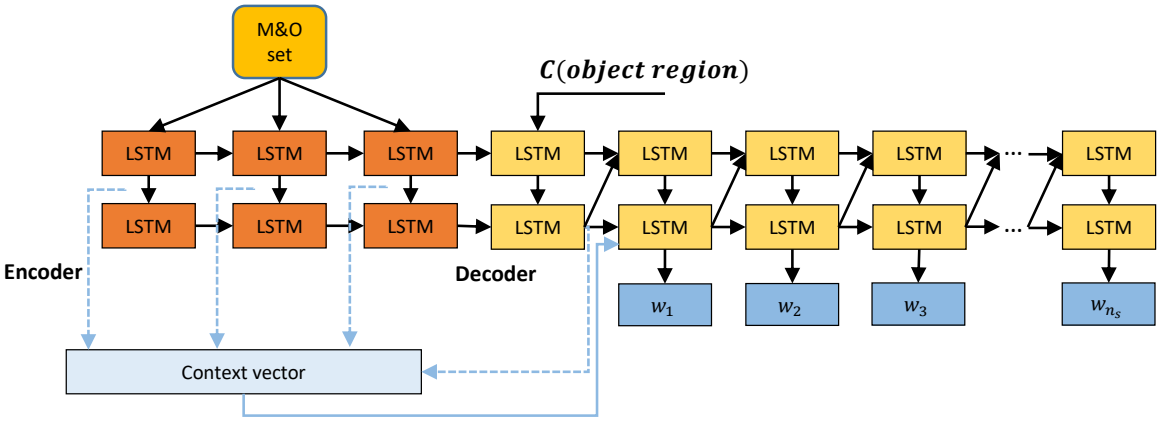


Figure 3: Detail architecture of LSTM-PE (The input of encoders is from M&O set, Local constraints $\mathbf{C}$ are input to initialize decoder as complementary information, $\mathbf{p}_t$ goes from the output layer to the input layer).

Figure 4: Examples of generated captions on MSR-VTT. GT1 and GT2 are ground truth captions.

where $\mathbf{U}_C$ and $\mathbf{U}_s$ are the transformation matrices for local constraints and intermediate representation respectively. $\mathbf{W}_i$, $\mathbf{W}_h$, $\mathbf{W}_p$, $\mathbf{W}_o$ are the weight matrices. f is the updating function within LSTM unit. We also apply an attention mechanism proposed by (Bahdanau et al., 2015) to LSTM units. The details can be illustrated in Figure 3. The output of LSTM is the final sentence $\mathbf{S}$ generated for video $\mathbf{V}$.

3 EXPERIMENTS AND DISCUSSION

We evaluate and compare our proposed LSTM-CSS with state-of-the-art approaches (Section 3.3) and analyze the effect of local constraints (Section 3.4.1). We also qualitatively analyze the effect of path embeddings (Section 3.4.2).

3.1 Datasets and settings

Dataset. We use MSR-VTT-2016 (Xu et al., 2016) which contains 10,000 web-collected video clips. There are roughly 20 available descriptions per video. In our experiment, our video captioning models are trained, hyperparameter selected, and evaluated using the official partitioned training, validation, and test set with 65%:5%:30% corresponding to 6,513, 497, 2,990 video clips, respectively.

Network detail. The motion stream is a reproduced model of TSN, fine-tuned over MSR-VTT-2016. For a single CNN, it closely resembles BN Inception network (Ioffe and Szegedy, 2015). We take the output of softmax layer from the combination of multiple CNNs as the motion representation $\mathbf{M}$. The object

Model	BLEU@4	METEOR	ROUGE-L	CIDEr-D
SA (Yao et al., 2015)	32.3	23.4	-	-
S2VT (Venugopalan et al., 2015)	35.2	25.2	-	-
Xu et al. (Xu et al., 2016)	36.6	25.9	-	-
MTLM (Pasunuru and Bansal, 2017)	40.8	**28.8**	60.2	47.1
WS (Shen et al., 2017)	41.4	28.3	61.1	**48.9**
Rank1: v2t_navigator	40.8	28.2	60.9	44.8
Rank2: Aalto	39.8	26.9	59.8	45.7
Rank3: VideoLAB	39.1	27.7	60.6	44.1
LSTM-CSS	**42.3**	28.5	**61.2**	46.5

Table 1: BLEU@4, METEOR, CIDEr-D, and ROUGE-L of our LSTM-CSS and other state-of-the-art methods on MSR-VTT-2016 dataset. All values are reported as percentage (%) and (-) indicates unknown scores

Model	BLEU@4	METEOR	ROUGE-L	CIDEr-D
LSTM-SR	38.1	25.6	59.0	40.7
LSTM-CSS-E-N	40.4	26.6	59.8	43.2
LSTM-CSS-N	41.1	26.9	60.2	44.8
LSTM-CSS-E	42.0	27.0	60.2	46.7
LSTM-CSS	42.3	28.5	61.2	46.5

Table 2: Captioning accuracy of LSTM-CSS, LSTM-SR, LSTM-CSS-E, LSTM-CSS-N, and LSTM-CSS-E-N on MSR-VTT-2016 test set

stream is trained from scratch, using an COCO (Lin et al., 2014) to evenly detect objects $\mathbf{O}$ belonging to 90 classes and extract local features. In each video, 5 frames are selected and local constraints $\mathbf{C}$ are the average of local features extracted from these frames. We take the output of softmax layer and output of 4096-way fc6 layer from the Resnet-50 (He et al., 2016) as $\mathbf{O}$ and $\mathbf{C}$ respectively. The margin m in Equation 1 is set to 1. The dimension of the input and hidden layers in LSTM are both set to 512. In test stage, we set the beam size to 4 in beam search.

Evaluation metric. We adopt four common metrics in video captioning task for quantitative evaluation of our proposed model: BLEU@4, METEOR, CIDEr-D, and ROUGE-L (from MS-COCO evaluation server (Chen et al., 2015)).

3.2 Compared methods

We compare our LSTM-CSS model with the following methods to evaluate the efficacy of our model.

SA (**S**oft-**A**ttention) (Yao et al., 2015): utilizes a weighted attention mechanism to dynamically attend to specific temporal segments of the video with the input of both frame representation (2-D CNN) and video clip representation (3-D CNN). In the test, the frame representation is extracted from the same feature, while the video clip representation is extracted from a C3D network.

S2VT (**S**equence to **S**equence - **V**ideo to **T**ext) (Venugopalan et al., 2015): incorporates both RGB and optical flow inputs. The encoding and decoding of inputs and word representations are jointly learned. In the test, the RGB and optical flows are extracted from same structure with ours.

MTLM (**M**ulti-**T**ask **L**earning **M**ethod) (Pasunuru and Bansal, 2017): improves video captioning by utilizing sharing representations with a temporally-directed unsupervised video prediction task to learn richer context-aware video encoder representations, and a logically-directed language entailment generation task to learn better caption decoder. In the test, the representation is extracted from same structure with ours.

WS (**W**eakly **S**upervised) (Shen et al., 2017): links video regions with lexical labels by utilizing lexical fully convolutional neural networks with weakly supervised learning. It also trains a sequence-to-sequence language model with the weakly supervised information. In the test, the CNN model is trained starting from the same structure with ours.

We also compare the baseline criterion and top-3 rank (i.e. v2t_navigator, Aalto, and VideoLAB) results proposed by (Xu et al., 2016).

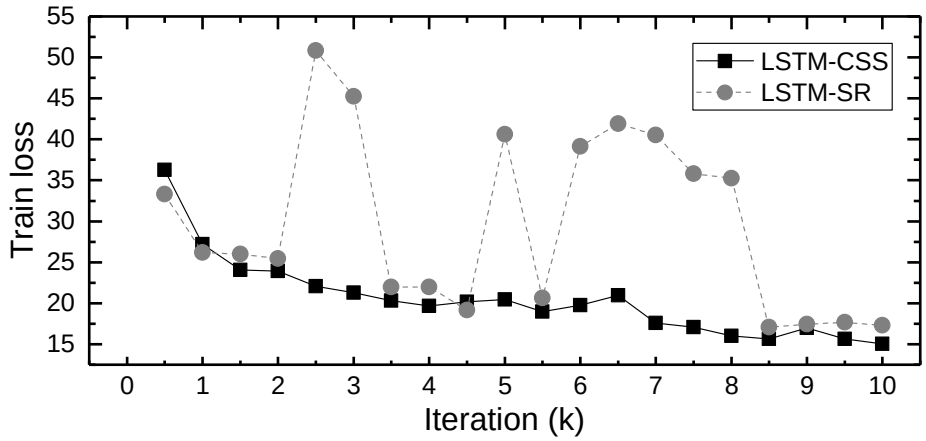


Figure 5: Train loss of LSTM-CSS and LSTM-SR

3.3 Performance comparison

Table 1 shows our primary results on MSR-VTT-2016 dataset. Overall, our proposed LSTM-CSS outperforms the other methods. Specifically, it reaches 42.3% BLEU@4, which makes the relative improvement over the two state-of-the-art methods S2VT by 20.1% and MTLM by 3.7%, respectively. We are also the new Rank1 on the MSR-VTT-2016 leaderboard, based on their ranking criteria. Figure 4 shows several example captions generated by our approach for MSR-VTT-2016 videos. The result indicates the utilizing of local constraints can contribute to the generation of comprehensive sentences.

In terms of BLEU scores, our method obtains bigger gains than others. Since the preference of longer captions tend to obtain lower BLEU scores, we argue that our method can generate comprehensive but concise captions. In terms of METEOR scores, our method does not perform best. From the aforementioned example, we can see that the subject of our caption is "fashion show" and the adverbial is "on the runway". In this example, there are no reference captions that have same syntactic structure with our caption. This may influence our recall rate and further affect the METEOR score.

3.4 Ablation study

3.4.1 Effect of Complementary syntactic elements

Our model **LSTM-CSS** depicted in Figure 2 uses local constraints to construct the final syntactic structure with complementary elements. To perform the ablation studies for our local constraints, we also test degraded versions of our model as follows: 1) **LSTM-SR**: the input of LSTM are semantic representation of motions and objects. Only global features are included; 2) **LSTM-CSS-E**: the local constraints are added as another input modal of LSTM and they are input to the encoder; 3) **LSTM-CSS-N**: the network without attention mechanism; 4) **LSTM-CSS-E-N**: the network remove attention mechanism from **LSTM-CSS-E**; 5) **LSTM-CSS w/o PE**: the network remove path embeddings from **LSTM-CSS**.

Figure 5 shows that LSTM-CSS achieves better training loss than LSTM-SR, which shows the training stability of LSTM-CSS. We can draw a conclusion that the local constraints can contribute to ensuring the convergence rate and alleviate the fluctuation of training loss. Table 2 summarizes the results on the MSR-VTT-2016 test set. As can be seen, in all the cases, LSTM-SR performs worse than the models with local features. Base on these results, the validation of local constraints is certificated. From the comparison of LSTM-CSS-E-N and LSTM-CSS-N, we can conclude that without attention mechanism, modeling global and local information at the same level will weaken the ability of local constraints. We can also find that LSTM-CSS and LSTM-CSS-E nearly perform equally. Therefore, we believe that attention mechanism can make the LSTM translation network focus on relevant content and alleviate the influence of the places where local constraints are input.

LSTM-CSS: a woman is talking on a show.
LSTM-CSS w/o PE: a woman is talking.

LSTM-CSS: a man is talking about a man who in black.
LSTM-CSS w/o PE: a man is talking about.

Figure 6: Examples with recursive syntactic structure

3.4.2 Qualitative analysis of path embeddings

As we can see, in the MSR-VTT-2016 dataset, some videos are about news, products, slides. In fact, people appearing in these videos are not the subject. Through visual analyzing, existing video caption algorithms can tell people are talking but not understand the content they talking about. Though some advanced methods can find both introducers and their contents, they are hard to model the relationship and organize sentences. In cases like 'someone is talking about something', our method can generate a completed sentence as shown in Figure 6. We can also see that in sentences generated by LSTM-CSS w/o PE, the content people shows are not included.

4 Conclusions

Inspired by the human receptors for object detection, we propose a hierarchical model for video captioning. Our model not only precisely captures motions and objects in videos but also learns a sentence constrained by cascade syntactic structure (*CRF* for the basic structure and *LSTM-PE* for the complementary elements). The experimental results demonstrate that our model performs on par with state-of-the-art methods on a large video description dataset in terms of accuracy.

However, the structure of generated sentence is relatively fixed for all videos. It should be better to improve the diversity of the sentences considering different object locations and make the syntactic structure more logical. We will address this issue in our future work.

References

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

Lorenzo Baraldi, Costantino Grana, and Rita Cucchiara. 2017. Hierarchical boundary-aware neural encoder for video captioning. In *CVPR*.

G. J. Burton and I. R. Moorhead. 1987. Color and spatial structure in natural scenes. *Applied Optics*, 26(1):157.

D. Chen and C. D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Conference on Empirical Methods in Natural Language Processing*.

Xinlei Chen, Hao Fang, Tsung Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and C. Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *Computer Science*.

Xu Chen, Yongfeng Zhang, Qingyao Ai, Hongteng Xu, Junchi Yan, and Zheng Qin. 2017. Personalized key frame recommendation. In *SIGIR*.

J Donahue, L. A. Hendricks, S Guadarrama, M Rohrbach, S Venugopalan, T Darrell, and K Saenko. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*.

Ross Girshick. 2015. Fast r-cnn. In *ICCV*.

S Guadarrama, N Krishnamoorthy, G Malkarnenkar, S Venugopalan, R Mooney, T Darrell, and K Saenko. 2014. Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *ICCV*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. 2017. Mask r-cnn. In *ICCV*.

Chiori Hori, Takaaki Hori, Teng Yok Lee, Kazuhiro Sumi, John R Hershey, and Tim K Marks. 2017. Attention-based multimodal fusion for video description. *arXiv preprint arXiv:1701.03126*.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. In *ACL*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*.

Haifeng Wang Ting Liu Jiang Guo, Wanxiang Che and Jun Xu. 2016. A unified architecture for semantic role labeling and relation classification. In *Conference on Computational Linguistics*.

Dotan Kaufman, Gil Levi, Tal Hassner, and Lior Wolf. 2017. Temporal tessellation: A unified approach for video analysis. In *ICCV*.

Atsuhiro Kojima, Takeshi Tamura, and Kunio Fukunaga. 2002. Natural language description of human activities from video images based on concept hierarchy of actions. *IJCV*.

John D. Lafferty, Andrew Mccallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollr, and C. Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *ECCV*.

Yingwei Pan, Tao Mei, Ting Yao, Houqiang Li, and Yong Rui. 2016. Jointly modeling embedding and translation to bridge video and language. In *CVPR*.

Yingwei Pan, Ting Yao, Houqiang Li, and Tao Mei. 2017. Video captioning with transferred semantic attributes. In *CVPR*.

Ramakanth Pasunuru and Mohit Bansal. 2017. Multi-task video captioning with video and entailment generation. In *ACL*.

Marcus Rohrbach, Wei Qiu, Ivan Titov, Stefan Thater, Manfred Pinkal, and Bernt Schiele. 2013. Translating video content to natural language descriptions. In *ICCV*.

Zhiqiang Shen, Jianguo Li, Zhou Su, Minjun Li, Yurong Chen, Yu Gang Jiang, and Xiangyang Xue. 2017. Weakly supervised dense video captioning. In *CVPR*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*.

S Venugopalan, M Rohrbach, J Donahue, R Mooney, T Darrell, and K Saenko. 2015. Sequence to sequence – video to text. In *ICCV*.

Heng Wang and Cordelia Schmid. 2014. Action recognition with improved trajectories. In *ICCV*.

Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. 2016. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*.

Ran Xu, Caiming Xiong, Jason J. Corso, and Jason J. Corso. 2015. Jointly modeling deep video and compositional text to bridge vision and language in a unified framework. In *AAAI*.

Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. Msr-vtt: A large video description dataset for bridging video and language. In *CVPR*.

L Yao, A Torabi, K Cho, N Ballas, C Pal, H Larochelle, and A Courville. 2015. Describing videos by exploiting temporal structure. In *ICCV*.

Ren Z., Yan J., Ni B., Zha H., and Yang X. 2017. Unsupervised deep learning for optical flow estimation. In *AAAI*.

Multi-task and Multi-lingual Joint Learning of Neural Lexical Utterance Classification based on Partially-shared Modeling

Ryo Masumura, Tomohiro Tanaka, Ryuichiro Higashinaka,
Hirokazu Masataki and Yushi Aono
NTT Media Intelligence Laboratories, NTT Corporation, Japan
ryou.masumura.ba@hco.ntt.co.jp

Abstract

This paper is an initial study on multi-task and multi-lingual joint learning for lexical utterance classification. A major problem in constructing lexical utterance classification modules for spoken dialogue systems is that individual data resources are often limited or unbalanced among tasks and/or languages. Various studies have examined joint learning using neural-network-based shared modeling; however, previous joint learning studies focused on either cross-task or cross-lingual knowledge transfer. In order to simultaneously support both multi-task and multi-lingual joint learning, our idea is to explicitly divide state-of-the-art neural lexical utterance classification into language-specific components that can be shared between different tasks and task-specific components that can be shared between different languages. In addition, in order to effectively transfer knowledge between different task data sets and different language data sets, this paper proposes a partially-shared modeling method that possesses both shared components and components specific to individual data sets. We demonstrate the effectiveness of the proposed adversarial training using Japanese and English data sets with three different lexical utterance classification tasks.

1 Introduction

Modern spoken dialogue systems use multiple lexical utterance classification modules that can detect dialogue act (Stolcke et al., 2000; Khanpour et al., 2016), intent (Tur et al., 2011), domain (Xu and Sarikaya, 2014), question type (Wu et al., 2005), etc. to properly understand natural languages. The modules are typically trained using the machine learning technologies developed for individual language-specific systems (Higashinaka et al., 2014). A common issue is that the data resources for such individual training are often limited or unbalanced among different tasks and/or different languages.

For modeling lexical utterance classification, various modeling methods have been examined. Recently, neural lexical utterance classification, which is a fully neural-network-based modeling method, has demonstrated substantial performance without the use of manual feature engineering. The networks include long short-term memory recurrent neural networks (LSTM-RNNs) (Ravuri and Stolcke, 2015b; Ravuri and Stolcke, 2015a; Ravuri and Stolcke, 2016), convolution neural networks (Kim, 2014), and more advanced networks (Zhou et al., 2016a; Yang et al., 2016; Sawada et al., 2017).

In addition, neural networks are suitable for performing joint learning; the paucity of data is tackled by transferring knowledge between different tasks or different languages. Various joint learning methods have been examined for leveraging different tasks or different language data sets in the natural language processing field. Multi-task joint learning can transfer knowledge between tasks by sharing task-invariant layers (Collobert and Weston, 2008; Liu et al., 2015; Liu et al., 2016c; Zhang and Weng, 2016). In lexical utterance classification, multi-task joint learning has been shown to effectively improve individual tasks (Liu et al., 2016b; Liu et al., 2016a; Liu et al., 2017). In addition, multi-lingual joint learning can transfer knowledge between languages, mainly from the resource-rich language to the resource-poor language. The knowledge transfer is achieved by learning common semantic representations for different

Proceedings of the 27th International Conference on Computational Linguistics, pages 3586–3596
Santa Fe, New Mexico, USA, August 20-26, 2018.

languages. Usually, word-aligned or sentence-aligned parallel data sets are employed for joint learning (Guo et al., 2016; Duong et al., 2016)

However, most existing joint learning approaches focus only on cross-tasks or cross-lingual knowledge transfer. In fact, task-aligned multi-lingual data sets have been rarely utilized for joint learning (Mogadala and Rettinger, 2016; Pappas and Popescu-Belis, 2017). We can expect to enhance lexical utterance classification performance by achieving effective knowledge transfer among both different tasks and different languages.

In this paper, we propose multi-task and multi-lingual joint learning; it can enhance neural lexical utterance classification by flexibly transferring knowledge among both different tasks and different languages. The proposed method is closely related to multi-task sequence-to-sequence learning (Luong et al., 2016) including many-to-many neural machine translation (Firat et al., 2016; Firat et al., 2017; Schwenk and Douze, 2017). While input and output components are easily distinguished in sequence-to-sequence models, neural lexical utterance classification methods are not explicitly divided into input and output components. Our idea is to divide the neural lexical utterance classification into two components. The language-specific components converts words to hidden representations while task-specific components convert the hidden representations into prediction probabilities. The former can be shared between different tasks and the latter can be shared between different languages.

In addition, in order to perform effective joint learning by simultaneously using multi-task and multi-lingual data sets, this paper examines two joint modeling strategies. The fully-shared modeling strategy is often used in various joint learning methods. Fully-shared modeling can share knowledge between tasks or languages based on task-invariant components and language-invariant components, however, classification performance in some data sets is deteriorated. Therefore, this paper proposes the partially-shared modeling strategy; it introduces not only shared components between tasks or languages but also exclusive components that handle task-language combinations. It can be expected that the latter are suitable for multi-task and multi-lingual joint learning since they allow us to accumulate just shareable knowledge.

Main contributions are summarized as follows.

- This paper proposes multi-task and multi-lingual joint learning of neural lexical utterance classification. For neural lexical utterance classification, we introduce a state-of-the-art model structure based on bidirectional LSTM-RNNs with a self-attention mechanism. We demonstrate the superiority of multi-task and multi-lingual joint learning over multi-task joint learning and multi-lingual joint learning.

- This paper proposes partially-shared modeling for multi-task and multi-lingual joint learning. Partially-shared modeling can be utilized for various neural network based joint learning schemes. We demonstrate the superiority of partially-shared modeling over fully-shared modeling. In addition, we reveal the properties of partially-shared modeling.

- This paper introduces a new corpus for evaluating multi-task and multi-lingual lexical utterance classification methods. The corpus includes Japanese and English data sets with three different lexical utterance classification tasks. The tasks are dialogue act classification, extended named entity classification (Sekine and Nobata, 2004; Higashinaka et al., 2012), and question type classification.

2 Neural Lexical Utterance Classification

This section details neural lexical utterance classification. Lexical utterance classification is the problem of determining the correct label $l \in \{l_1, \cdots, l_K\}$ of given utterance $\mathcal{W} = \{w_1, \cdots, w_T\}$. In neural lexical utterance classification, conditional probabilities for each label given utterance, $P(l|\mathcal{W}, \mathbf{\Theta})$, can be modeled by neural networks in an end-to-end manner where $\mathbf{\Theta}$ is the model parameter. Various model structures can be used for neural lexical utterance classification. In this work, we use bidirectional LSTM-RNNs (BLSTM-RNNs) with a self-attention mechanism (Yang et al., 2016; Zhou et al., 2016b).

2.1 Modeling

In our neural lexical utterance classification, each word in input utterance $\mathcal{W}$ is first converted into a continuous representation. The continuous representation of the t-th word is defined as:

$$\boldsymbol{w}_t = \texttt{EMBED}(w_t; \boldsymbol{\theta}_{\texttt{w}}), \tag{1}$$

where $\texttt{EMBED}()$ is a linear transformational function to embed a word into a continuous vector and $\boldsymbol{\theta}_{\texttt{w}}$ is the trainable parameter. Next, each word representation is converted into a hidden representation that takes neighboring word context information into consideration. The hidden representation for the t-th word is calculated as:

$$\boldsymbol{h}_t = \texttt{BLSTM}(\{\boldsymbol{w}_1, \cdots, \boldsymbol{w}_T\}, t; \boldsymbol{\theta}_{\texttt{h}}), \tag{2}$$

where $\texttt{BLSTM}()$ is a function of the BLSTM-RNN layer and $\boldsymbol{\theta}_{\texttt{h}}$ is the trainable parameter.

The hidden representations are summarized as a sentence representation by using a self-attention mechanism that can consider the importance of individual hidden representations. The sentence continuous representation $\boldsymbol{s}$ is calculated as:

$$\boldsymbol{z}_t = \texttt{tanh}(\boldsymbol{h}_t; \boldsymbol{\theta}_{\texttt{z}}), \tag{3}$$

$$\boldsymbol{s} = \sum_{t=1}^{T} \frac{\exp(\boldsymbol{z}_t^\top \bar{\boldsymbol{z}})}{\Sigma_{j=1}^{T} \exp(\boldsymbol{z}_j^\top \bar{\boldsymbol{z}})} \boldsymbol{h}_t, \tag{4}$$

where $\texttt{tanh}()$ is a non-linear transformational function with tanh activation, $\boldsymbol{\theta}_{\texttt{z}}$ is the trainable parameter, and $\bar{\boldsymbol{z}}$ is a trainable context vector, which is used for measuring the importance of individual hidden representations. The output layer produces predicted probabilities $\boldsymbol{O}$ by:

$$\boldsymbol{o} = \texttt{LINEAR}(\boldsymbol{s}; \boldsymbol{\theta}_{\texttt{o}}), \tag{5}$$

$$\boldsymbol{O} = \texttt{SOFTMAX}(\boldsymbol{o}), \tag{6}$$

where $\texttt{LINEAR}()$ is a linear transformational function and $\boldsymbol{\theta}_{\texttt{o}}$ is the trainable parameter. $\texttt{SOFTMAX}()$ is a softmax activation to convert $\boldsymbol{o}$ into predicted probabilities. The k-th dimension in $\boldsymbol{O}$ corresponds to $P(l_k|\mathcal{W}, \boldsymbol{\Theta})$, and $\boldsymbol{\Theta}$ corresponds to $\{\boldsymbol{\theta}_{\texttt{w}}, \boldsymbol{\theta}_{\texttt{h}}, \boldsymbol{\theta}_{\texttt{z}}, \bar{\boldsymbol{z}}, \boldsymbol{\theta}_{\texttt{o}}\}$.

2.2 Optimization

The parameter can be optimized by minimizing the cross entropy between reference and estimated probabilities:

$$\hat{\boldsymbol{\Theta}} = \operatorname*{argmin}_{\boldsymbol{\Theta}} - \sum_{\mathcal{W} \in \mathcal{D}} \sum_{k=1}^{K} \hat{\boldsymbol{O}}_{\mathcal{W}}^{l_k} \log \boldsymbol{O}_{\mathcal{W}}^{l_k}, \tag{7}$$

where $\hat{\boldsymbol{O}}_{\mathcal{W}}^{l_k}$ and $\boldsymbol{O}_{\mathcal{W}}^{l_k}$ are, respectively, the reference probability and the estimated probability of label l_k for $\mathcal{W}$. $\mathcal{D}$ denotes the training data set.

3 Multi-task and Multi-lingual Neural Lexical Utterance Classification

This section presents multi-task and multi-lingual joint learning of neural lexical utterance classification. We split neural lexical utterance classification by considering two types of components: language-specific components and task-specific components.

Language-specific components can be shared between tasks, where words in an utterance are converted into hidden representations. The language-specific components can be simplified as:

$$\boldsymbol{h}_t = \texttt{W2H}(\mathcal{W}, t; \boldsymbol{\Theta}_{\texttt{W2H}}), \tag{8}$$

where $\texttt{W2H}()$ is a function that compiles Eqs. (1) and (2). $\boldsymbol{\Theta}_{\texttt{W2H}}$ corresponds to $\{\boldsymbol{\theta}_{\texttt{w}}, \boldsymbol{\theta}_{\texttt{h}}\}$.

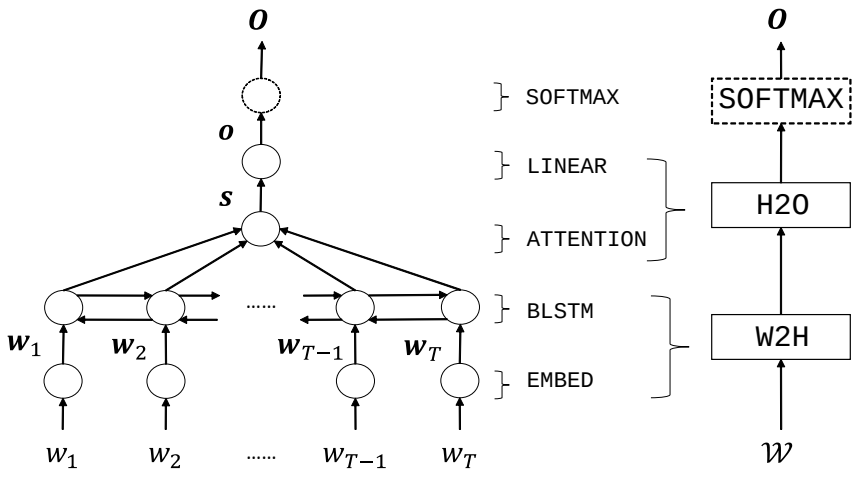


Figure 1: Neural lexical utterance classification and its simplified model structure.

Task-specific components can be shared between languages, where hidden representations are converted into predicted probabilities. The task-specific components can be simplified as:

$$\boldsymbol{o} = \texttt{H2O}(\{\boldsymbol{h}_1, \cdots, \boldsymbol{h}_T\}; \boldsymbol{\Theta}_{\texttt{H2O}}), \tag{9}$$

where $\texttt{H2O}()$ is a function that compiles Eqs. (3) to (5) and $\boldsymbol{\Theta}_{\texttt{H2O}}$ represents $\{\boldsymbol{\theta}_{\texttt{z}}, \bar{\boldsymbol{z}}, \boldsymbol{\theta}_{\texttt{o}}\}$.

Figure 1 shows a detailed model structure and a simplified model structure of BLSTM-RNN with the attention mechanism. Both the dotted line square and a dotted line circle represent softmax activation. White squares are simplified components in the neural lexical utterance classification.

3.1 Fully-shared Modeling

Fully-shared modeling forms universal hidden representations that are completely invariant to differences in tasks or languages. In this case, language-specific components are fully-shared between the same language data sets and task-specific components are fully-shared between the same task data sets. The t-th universal hidden representation is calculated as:

$$\boldsymbol{h}_t = \texttt{W2H}(\mathcal{W}^{(i)}, t; \boldsymbol{\Theta}^{(i)}_{\texttt{W2H}}), \tag{10}$$

where $\boldsymbol{\Theta}^{(i)}_{\texttt{W2H}}$ is the shared parameter that handles the i-th language and $\mathcal{W}^{(i)}$ denotes the input utterance in the i-th language. The universal hidden representation can be input to any task-specific component. The predicted probabilities for the j-th task, denoted as $\boldsymbol{O}^{(j)}$, are calculated as:

$$\boldsymbol{o}^{(j)} = \texttt{H2O}(\{\boldsymbol{h}_1, \cdots, \boldsymbol{h}_T\}; \boldsymbol{\Theta}^{(j)}_{\texttt{H2O}}), \tag{11}$$

$$\boldsymbol{O}^{(j)} = \texttt{SOFTMAX}(\boldsymbol{o}^{(j)}), \tag{12}$$

where $\boldsymbol{\Theta}^{(j)}_{\texttt{H2O}}$ is the task-specific shared parameter that handles the j-th task.

Figure 2 shows the model structure of multi-task fully-shared modeling for a language and two tasks. Figure 3 shows the model structure of multi-lingual fully-shared modeling for two languages and a task. Figure 4 shows the model structure of multi-task and multi-lingual fully-shared modeling for two tasks and two languages. Gray squares are shared components between languages or between tasks, and white squares are non-shared components.

3.2 Partially-shared Modeling

Partially-shared modeling introduces not only shared components between tasks or languages but also exclusive components that handle task-language combinations. Therefore, our hidden representations

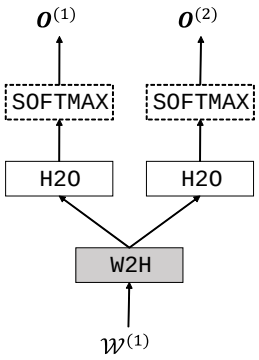


Figure 2: Multi-task fully-shared modeling.

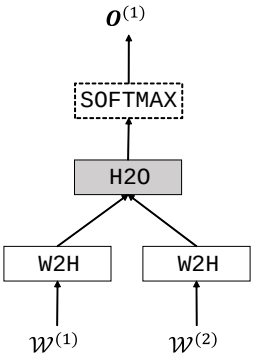


Figure 3: Multi-lingual fully-shared modeling.

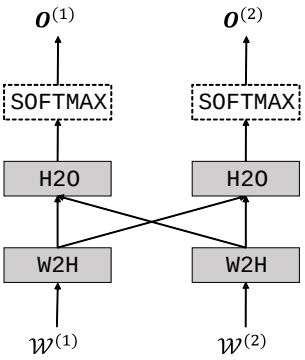


Figure 4: Multi-task and multi-lingual fully-shared modeling.

are designed to support such combinations. The t-th hidden representation for the combination of the i-th language and the j-th task, denoted as $\boldsymbol{h}_t^{(i,j)}$, is calculated as:

$$\bar{\boldsymbol{h}}_t^{(i,j)} = \texttt{W2H}(\mathcal{W}^{(i,j)}, t; \boldsymbol{\Theta}_{\texttt{W2H}}^{(i,j)}), \tag{13}$$

$$\bar{\boldsymbol{h}}_t^{(i)} = \texttt{W2H}(\mathcal{W}^{(i,j)}, t; \boldsymbol{\Theta}_{\texttt{W2H}}^{(i)}), \tag{14}$$

$$\boldsymbol{h}_t^{(i,j)} = \bar{\boldsymbol{h}}_t^{(i,j)} + \bar{\boldsymbol{h}}_t^{(i)}, \tag{15}$$

where $\boldsymbol{\Theta}_{\texttt{W2H}}^{(i,j)}$ is the exclusive parameter specific to the combination of the i-th language and the j-th task, and $\boldsymbol{\Theta}_{\texttt{W2H}}^{(i)}$ is the shared parameter that handles the i-th language. The predicted probabilities for the combination of the i-th language and j-th task, denoted as $\boldsymbol{O}^{(i,j)}$, are calculated as:

$$\bar{\boldsymbol{o}}^{(i,j)} = \texttt{H2O}(\{\boldsymbol{h}_1^{(i,j)}, \cdots, \boldsymbol{h}_T^{(i,j)}\}; \boldsymbol{\Theta}_{\texttt{H2O}}^{(i,j)}), \tag{16}$$

$$\bar{\boldsymbol{o}}^{(j)} = \texttt{H2O}(\{\boldsymbol{h}_1^{(i,j)}, \cdots, \boldsymbol{h}_T^{(i,j)}\}; \boldsymbol{\Theta}_{\texttt{H2O}}^{(j)}), \tag{17}$$

$$\boldsymbol{O}^{(i,j)} = \texttt{SOFTMAX}(\bar{\boldsymbol{o}}^{(i,j)} + \bar{\boldsymbol{o}}^{(j)}), \tag{18}$$

where $\boldsymbol{\Theta}_{\texttt{H2O}}^{(i,j)}$ is the exclusive parameter specific to the combination of the i-th language and the j-th task, and $\boldsymbol{\Theta}_{\texttt{H2O}}^{(j)}$ is the shared parameter that handles the j-th task.

Figure 5 shows the model structure of multi-task partially-shared modeling for a language and two tasks. Figure 6 shows the model structure of multi-lingual partially-shared modeling for two tasks and a language. Figure 7 shows the model structure of multi-task and multi-lingual partially-shared modeling for two tasks and two languages. Note that Figures 5-7 correspond to Figures 2-4.

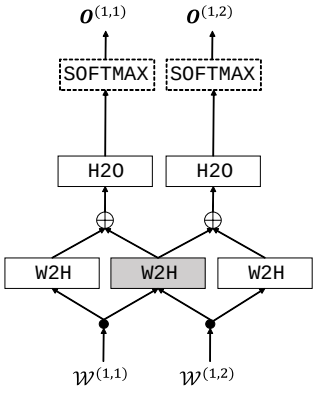


Figure 5: Multi-task partially-shared modeling.

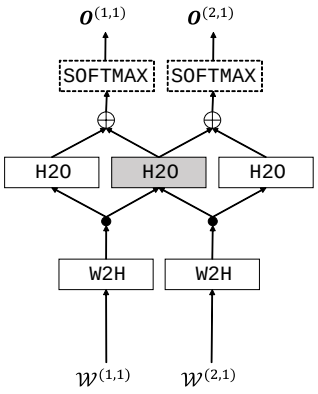


Figure 6: Multi-lingual partially-shared modeling.

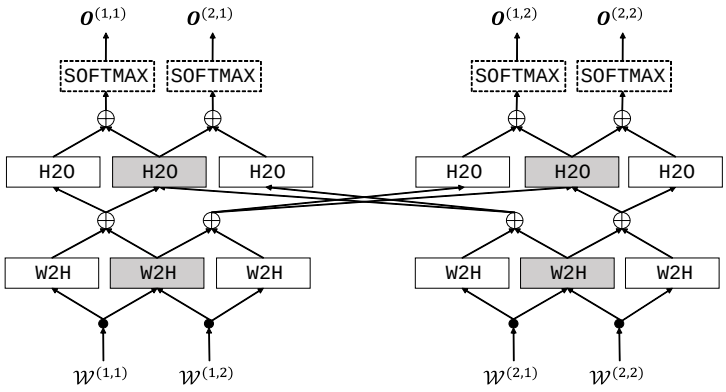


Figure 7: Multi-task and multi-lingual partially-shared modeling.

3.3 Joint Optimization

In multi-task and multi-lingual joint leaning, all parameters, denoted as $\mathbf{\Theta}$, can be jointly optimized by using all data sets. Given I languages and J tasks, joint optimization of the model parameter $\mathbf{\Theta}$ follows:

$$\hat{\mathbf{\Theta}} = \underset{\mathbf{\Theta}}{\operatorname{argmin}} - \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{\mathcal{W} \in \mathcal{D}^{(i,j)}} \sum_{k=1}^{K^{(j)}} \hat{\boldsymbol{O}}_{\mathcal{W}}^{l_k} \log \boldsymbol{O}_{\mathcal{W}}^{l_k}, \tag{19}$$

where $\mathcal{D}^{(i,j)}$ denotes the training data set for the combination of the i-th language and the j-th task, and $|\mathcal{D}^{(i,j)}|$ means the number of utterances. $K^{(j)}$ represents the number of labels in the j-th task.

Basically, $\mathbf{\Theta}$ can be gradually updated by repeating mini-batch training using individual data sets. In this case, an optimizer with a learning rate is prepared for individual data sets and individual learning rates fall when the cross entropy loss for a target validation data set increases. The training epoch is stopped when the averaged loss for all validation data sets is not improved. Details of the joint optimization procedure are shown in **Algorithm 1**.

4 Experiments

4.1 Data

Our experiments employed Japanese (Ja) and English (En) data sets created for three different lexical utterance classification tasks. The tasks were dialogue act (DA) classification, extended named entity (ENE) classification (Sekine and Nobata, 2004; Higashinaka et al., 2012), and question type (QT) classification; natural language texts were used for the lexical utterances and individual label sets were unified

Algorithm 1 :Joint Optimization procedure of multi-task and multi-lingual joint leaning.

Input: Training data sets $\mathcal{D}^{(1,1)}, \cdots, \mathcal{D}^{(1,J)}, \cdots, \mathcal{D}^{(I,1)}, \cdots, \mathcal{D}^{(I,J)}$
Validation data sets $\bar{\mathcal{D}}^{(1,1)}, \cdots, \bar{\mathcal{D}}^{(1,J)}, \cdots, \bar{\mathcal{D}}^{(I,1)}, \cdots, \bar{\mathcal{D}}^{(I,J)}$
Output: Model parameter $\mathbf{\Theta}$

Initialize $\mathbf{\Theta}$ randomly
while true **do**
  for $t = 1$ to number of mini-batches in training data sets **do**
    Select language i randomly
    Select task j randomly
    Pick mini-batch $\mathcal{D}_t$ from $\mathcal{D}^{(i,j)}$ randomly
    Update $\mathbf{\Theta}$ by computing gradient of $\mathcal{D}_t$
  end for
  for $i = 1$ to I **do**
    for $j = 1$ to J **do**
      Compute current validation loss for $\bar{\mathcal{D}}^{(i,j)}$
      if previous validation loss for $\bar{\mathcal{D}}^{(i,j)}$ $<$ current validation loss for $\bar{\mathcal{D}}^{(i,j)}$ **then**
        Decrease learning rate for $\mathcal{D}^{(i,j)}$
      end if
    end for
  end for
  if previous averaged validation loss $<$ current averaged validation loss **then**
    break
  end if
end while
return $\mathbf{\Theta}$

Table 1: Number of utterances in individual data sets.

Language	Task	#labels	Train	Valid	Test
Japanese	DA	28	201,092	4,190	4,190
	ENE	168	40,350	4,036	4,036
	QT	15	55,328	4,257	4,257
English	DA	28	25,171	3,147	3,147
	ENE	168	25,005	3,230	3,230
	QT	15	22,213	2,777	2,777

between Japanese and English. For example, the task of English ENE classification is to obtain the requested ENE type for a question. Each of the data sets were divided into training (Train), validation (Valid), and test (Test) sets. Table 1 shows the number of utterances in individual data sets where #labels represents the number of labels. Table 2 shows English utterances and label examples for individual tasks.

4.2 Setups

We evaluated non-shared modeling, fully-shared modeling, and partially-shared modeling. For the shared modeling methods, multi-task joint learning, multi-lingual joint learning, multi-task and multi-lingual joint learning were examined. The multi-task joint learning used three classification tasks for optimizing each language. The multi-lingual joint learning used both Japanese and English data sets for optimizing each task. The multi-task and multi-lingual joint learning used all data sets. Several modeling parameters were unified. Word representation size was set to 128, LSTM-RNN unit size was set to 200, and context vector size in the attention mechanism was set to 200. Dropout was used for `EMBED`() and `BLSTM`(), and the dropout rate was set to 0.5. In these setups, words that appeared once or less in

Table 2: English utterances and label examples in individual tasks.

Task	Utterance	Label
DA	Hello, how are you today?	GREETING
	I am so sorry to hear of your son's accident.	SYMPATHY/AGREE
	Lets go to school an hour early today.	PROPOSAL
ENE	What is the highest mountain in the world?	MOUNTAIN
	Who is president of the united states?	PERSON
	What is the name of the most recent Star Wars movie?	MOVIE
QT	Do you like egg salad?	TRUE/FALSE
	How do you correct a hook in a golf swing?	EXPLANATION:METHOD
	Why is blood red?	EXPLANATION:CAUSE

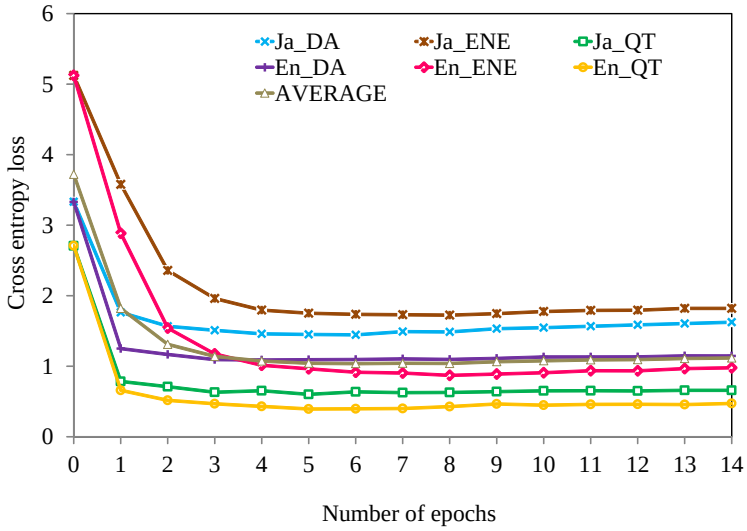


Figure 8: Cross entropy loss for validation sets.

the training data sets were treated as unknown words. For joint learning, mini-batch stochastic gradient descent was used for the individual optimizers. Initial learning rate for individual data sets was set to 0.1. The cutoff threshold for gradient clipping was set to 1.0. Training was stopped when the averaged validation loss was not improved in 5 consecutive iterations.

4.3 Results

Figure 8 demonstrates the change in cross entropy validation loss for individual validation sets when performing multi-task and multi-lingual joint learning based on partially-shared modeling. As epoch number increased, both cross entropy validation losses for individual data sets and averaged validation loss for all data sets (AVERAGE) decreased. This indicates that joint optimization procedure used in algorithm 1 worked well.

Table 2 shows the experimental results in terms of utterance classification accuracy for test sets. For each setup, we constructed five models by varying the initial parameters and evaluated the average accuracy.

First, (a) demonstrates the results of non-shared modeling trained using only an individual data set. These results are the baseline of this evaluation. In fully-shared modeling, (b) to (d), the classification performance deteriorated in some cases, while performance improvements were achieved in other cases. In English QT, multi-task and multi-lingual joint learning was inferior to multi-task joint learning or multi-lingual joint learning. This indicates that fully-shared modeling, which learns universal hidden representations, are not suitable for supporting both cross-lingual and cross-task knowledge transfer.

Table 2: Experimental results: utterance classification accuracy (%) for individual test sets.

		Joint learning		Japanese			English		
		task	language	DA	ENE	QT	DA	ENE	QT
(a).	Non-shared modeling	-	-	66.5	79.1	87.7	61.8	64.7	83.5
(b).	Fully-shared modeling	√	-	66.5	79.6	89.3	60.6	64.4	83.7
(c).		-	√	66.7	78.7	87.2	61.4	64.3	83.0
(d).		√	√	66.5	79.7	89.3	60.5	65.4	82.6
(e).	Partially-shared modeling	√	-	66.6	80.9	89.4	62.0	64.8	83.7
(f).		-	√	**66.9**	79.7	88.0	61.9	65.0	83.8
(g).		√	√	**66.9**	**81.8**	**89.7**	**62.3**	**65.8**	**84.0**

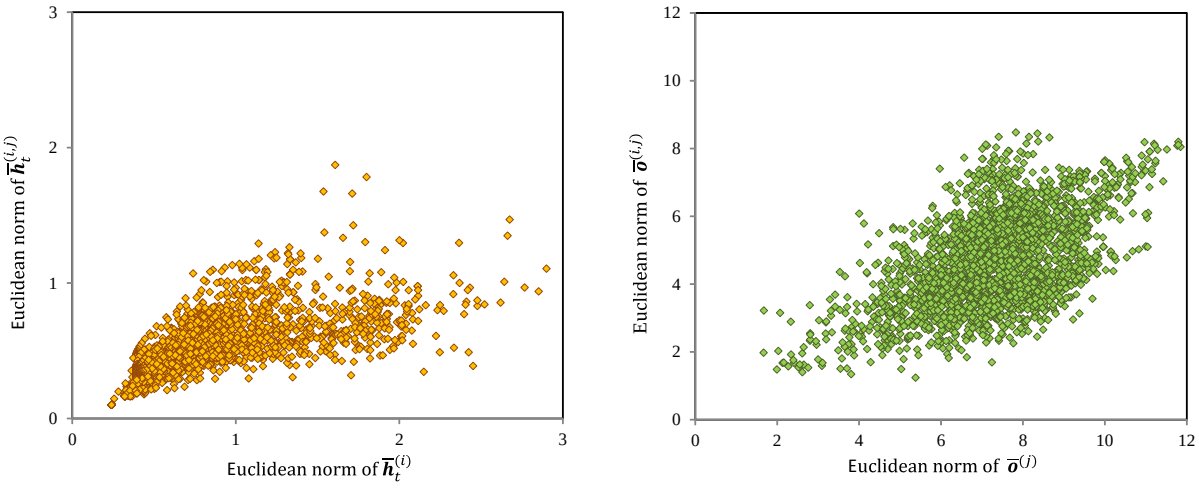


Figure 9: Euclidean norm results in partially-shared modeling.

On the other hand, partially-shared modeling, (e) to (g), improved the classification performance in all data sets compared to (a). In addition, multi-task and multi-lingual joint learning outperformed multi-task joint learning and multi-lingual joint learning. These results confirm that partially-shared modeling can effectively transfer knowledge between different data sets. We conducted sign test for verifying the effectiveness of multi-task and multi-lingual joint learning based on partially-shared modeling. In Japanese ENE classification, Japanese QT classification, and English ENE classification, statistically significant performance improvements ($p < 0.05$) were achieved by (g) compared to (a). Furthermore, in Japanese ENE classification, English DA classification, and English QT classification, significant performance improvements ($p < 0.05$) were also achieved by (g) compared to (d).

We investigated how shared components and exclusive components worked in partially-shared modeling. The left side of Figure 9 presents the Euclidean norm of both $\bar{\boldsymbol{h}}_t^{(i,j)}$ and $\bar{\boldsymbol{h}}_t^{(i)}$, while the right side presents the Euclidean norm of both $\bar{\boldsymbol{o}}^{(i,j)}$ and $\bar{\boldsymbol{o}}^{(j)}$ when classifying English question type validation data sets. These results show that shared components are more influential than exclusive components. This indicates that the shared components accumulate shareable knowledge, while exclusive components were utilized to offset the small differences between tasks or between languages.

5 Conclusions

This paper proposed multi-task and multi-lingual joint learning of neural lexical utterance classification for effectively leveraging data sets of different tasks and different language. For neural lexical utterance classification, we proposed BLSTM-RNN; it uses a self-attention mechanism and introduces language-specific and task-specific components. Each component can be effectively trained by partially-shared

modeling. Experiments on Japanese and English data sets created for three different tasks showed that the proposed multi-task and multi-lingual joint learning based on partially-shared modeling can transfer knowledge more effectively than multi-task or multi-lingual joint learning based on fully-shared modeling.

References

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. *In Proc. International Conference on Machine Learning (ICML)*.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2016. Learning crosslingual word embeddings without bilingual corpora. *In Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1285–1295.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. *In Proc. Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 866–875.

Orhan Firat, Kyunghyun Cho, Baskaran Sankaran, Fatos T. Yarman Vural, and Yoshua Bengio. 2017. Multi-way, multilingual neural machine translation. *Computer Speech & Language*, pages 236–252.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. *In Proc. AAAI Conference on Artificial Intelligence (AAAI)*, pages 2734–2740.

Ryuichiro Higashinaka, Kugatsu Sadamitsu, Kuniko Saito, Toshiro Makino, and Yoshihiro Matsuo. 2012. Creating an extended named entity dictionary from wikipedia. *In Proc. International Conference on Computational Linguistics (COLING)*, pages 1163–1178.

Ryuichiro Higashinaka, Kenji Imamura, Toyomi Meguro, Chiaki Miyazaki, Nozomi Kobayashi, Hiroaki Sugiyama, Toru Hirano, Toshiro Makino, and Yoshihiro Matsuo. 2014. Towards an open-domain conversational system fully based on natural language processing. *In Proc. International Conference on Computational Linguistics (COLING)*, pages 928–9239.

Hamed Khanpour, Nishitha Guntakandla, and Rodney Nielsen. 2016. Dialogue act classification in domain-independent conversations using a deep recurrent neural network. *In Proc. International Conference on Computational Linguistics (COLING)*, pages 2012–2021.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *In Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. *In Proc. Annual Conference of the North American Chapter of the ACL (NAACL)*, pages 912–921.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016a. Deep multi-task learning with shared memory. *In Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 118–127.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016b. Recurrent neural network for text classification with multi-task learning. *In Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2873–2879.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016c. Implicit discourse relation classification via multi-task neural networks. *In Proc. AAAI Conference on Artificial Intelligence (AAAI)*, pages 2750–2756.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. *In Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–10.

Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. *In Proc. International Conference on Learning Representations (ICLR)*.

Aditya Mogadala and Achim Rettinger. 2016. Bilingual word embeddings from parallel and non-parallel corpora for cross-language text classification. *In Proc. Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 692–702.

Nikolaos Pappas and Andrei Popescu-Belis. 2017. Multilingual hierarchical attention networks for document classification. *In Proc. International Joint Conference on Natural Language Processing (IJCNLP)*, pages 1015–1025.

Suman Ravuri and Andreas Stolcke. 2015a. A comparative study of neural network models for lexical intent classification. *In Proc. Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 368–374.

Suman Ravuri and Andreas Stolcke. 2015b. Recurrent neural network and LSTM models for lexical utterance classification. *In Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 135–139.

Suman Ravuri and Andreas Stolcke. 2016. A comparative study of recurrent neural network models for lexical domain classification. *In Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6075–6079.

Naoki Sawada, Ryo Masumura, and Hiromitsu Nishizaki. 2017. Parallel hierarchical attention networks with shared memory reader for multi-stream conversational document classification. *In Proc. Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 3311–3315.

Holger Schwenk and Matthijs Douze. 2017. Learning joint multilingual sentence representations with neural machine translation. *In Proc. Workshop on Representation Learning for NLP*, pages 157–167.

Satoshi Sekine and Chikashi Nobata. 2004. Definition, dictionaries and tagger for extended named entity hierarchy. *In Proc. Language Resources and Evaluation Conference (LREC)*, pages 1977–1980.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martion, Carol Van Ess-Dykema, and Marie Metter. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–373.

Gokhan Tur, Dilek Hakkani-Tur, Larry Heck, and Suresh Parthasarathy. 2011. Sentence simplification for spoken language understanding. *In Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5628–5631.

Chung-Hsien Wu, Jui-Feng Yeh, and Ming-Jun Chen. 2005. Domain-specific FAQ retrieval using independent aspects. *ACM Transactions on Asian Language Information Processing*, 4(1):1–17.

Puyang Xu and Ruhi Sarikaya. 2014. Contextual domain classification in spoken language understanding systems using recurrent neural network. *In Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 136–140.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. *In Proc. Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1480–1489.

Xiaodong Zhang and Houfeng Weng. 2016. A joint model of intent determination and slot filling for spoken language understanding. *In Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2993–2999.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016a. Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. *In Proc. International Conference on Computational Linguistics (COLING)*, pages 3485–3496.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016b. Attention-based bidirectional long short-term memory networks for relation classification. *In Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 207–212.

Source-Critical Reinforcement Learning for Transferring Spoken Language Understanding to a New Language

He Bai[1,2], **Yu Zhou**[1,2], **Jiajun Zhang**[1,2], **Liang Zhao**[3], **Mei-Yuh Hwang**[3] **and Chengqing Zong**[1,2,4]
[1] National Laboratory of Pattern Recognition, Institute of Automation, CAS, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] Mobvoi AI Lab, Seattle, US
[4] CAS Center for Excellence in Brain Science and Intelligence Technology
{he.bai, yzhou, jjzhang, cqzong}@nlpr.ia.ac.cn, {liangzhao, mhwang}@mobvoi.com

Abstract

To deploy a spoken language understanding (SLU) model to a new language, language transferring is desired to avoid the trouble of acquiring and labeling a new big SLU corpus. Translating the original SLU corpus into the target language is an attractive strategy. However, SLU corpora consist of plenty of semantic labels (slots), which general-purpose translators cannot handle well, not to mention additional culture differences. This paper focuses on the language transferring task given a tiny in-domain parallel SLU corpus. The in-domain parallel corpus can be used as the first adaptation on the general translator. But more importantly, we show how to use reinforcement learning (RL) to further finetune the adapted translator, where translated sentences with more proper slot tags receive higher rewards. We evaluate our approach on Chinese to English language transferring for SLU systems. The experimental results show that the generated English SLU corpus via adaptation and reinforcement learning gives us over 97% in the slot F1 score and over 84% accuracy in domain classification. It demonstrates the effectiveness of the proposed language transferring method. Compared with naive translation, our proposed method improves domain classification accuracy by relatively 22%, and the slot filling F1 score by relatively more than 71%.

1 Introduction

Spoken language understanding (SLU) is a key technique in today's conversational systems such as Apple Siri, Amazon Alexa and Microsoft Cortana. To make these conversational systems support multiple languages over different markets, collecting and annotating a large SLU training corpus per language are tedious and costly, and thus hinders the scalability of these systems. It would be greatly helpful if the efforts taken to develop one SLU system could be reused for other languages.

For such a purpose, much work has been reported to explore language transferring of SLU systems or multilingual SLU systems (García et al., 2012; Calvo et al., 2013; Calvo et al., 2016; Jabaian et al., 2016). These systems can be grouped into two categories: test-on-source-model vs. train-on-target-language. Test-on-source-model is to translate the test sentence in the 2nd language (referred to as L2 from now on) into the first SLU system's language (as L1), and then process it with the L1 SLU system. Train-on-target-language is to translate the L1 training corpus into L2, and then train an SLU system in L2. The train-on-target-language strategy allows tuning and adaptation of the models in the target language directly, and it avoids an overhead of machine translation during real-time execution. The test-on-source-model implies that the final search engine has to deal with L1 while the target answer database might be still in L2. Either the slot value needs to be translated again back to L2, or the database needs to be pre-translated into L1. Hence train-on-target-language is our preferred approach.

Each sample of the SLU training corpus consists of a query and its semantic annotation, for example,"Play <song>Sorry</song> in <album>They Don't Know</album>". In the L1 SLU training corpus, both query and its semantic annotation need to be transferred properly to the target language. Literally sending the annotated sentence to general-purpose machine translator or web translator may

Proceedings of the 27th International Conference on Computational Linguistics, pages 3597–3607
Santa Fe, New Mexico, USA, August 20-26, 2018.

lose or screw up the annotation information. Two schemes (Jabaian et al., 2013) have been proposed to solve this problem: transferring source language annotation indirectly through word alignment, vs. adapting the translation model so that it learns how to translate text with slot labels. Each scheme has its own limitations. The first scheme is weak for distant language pairs (e.g. Chinese and English) and the errors are accumulative after translation and word alignment. The second scheme needs extra data for adapting the translation model. In addition, some previous works (Servan et al., 2010; Misu et al., 2012; Jabaian et al., 2013) only focused on transferring slot labels but ignored the fact that the slot values should be adjusted properly to the target culture. For example, people in London will probably say "call a taxi to Tower of London" rather than "call a taxi to Forbidden City". It's better to make some adaptations on such culture difference for the L2 SLU corpus. In this paper, we will address culture adaptation when we transfer Chinese corpora to English and we will demonstrate its importance via experimental results. Given that neural machine translation (NMT) is the state-of-the-art translator, NMT model is applied in this paper for language transferring.

The main contribution of this paper is the proposal of using reinforcement learning to improve translation for language transferring. We name it source-critical reinforcement translation (SCRT). We first define the slot keeping ratio (SKR) as a metric for evaluating the performance of slot transferring. SCRT adapts NMT models via rewarding those translation candidates with higher SKR. By doing so, we can obtain target language translations which maintain both the semantics and slot information of the SLU labeled sentences in the source language. We evaluate our method on domain classification and slot filling tasks. The results show that our RL method improves the slot F1 score from 93% to 97% and domain accuracy from 82% to 84%, on top of an already adapted NMT.

2 Related Work

Language transferring for SLU systems has been an active research topic in recent years and much work has been reported on both the test-on-source-model and train-on-target-language two strategies. In Jabaian et al. (2013), different approaches based on test-on-source-model and train-on-target-language strategies are compared.

The simplest way for the test-on-source-model scheme is to translate L2 target sentences with a web translator into L1, and process these translations with the L1 SLU system. In He et al. (2013) the Microsoft Bing translator is used for this purpose. This approach provides translations of the user input at a very low economic and time cost. Moreover, Stepanov et al. (2013) have demonstrated that application of language-style and domain adaptation techniques to the "off-the-shelf and out-of-domain" SMT system could yield improved translation and thus obtain better SLU performance. For adapting MT, Garcıa et al. (2014) translated the L1 SLU training corpus into L2, with multiple web translators to obtain a large parallel dataset, and then trained their own L2-to-L1 SMT model to reduce translation errors. Calvo et al. (2013; 2016) proposed to build graphs of words from different translations and then to parse the translation graph with the SLU model.

The train-on-target-language approach relies on the accurate transferring of semantic annotations from L1 to L2. García et al. (2012) translates each segment within one slot separately with a web translator, and then joins these pieces together into a sentence. This method doesn't need additional procedures to transfer slot labels. However, this method misses the context information that is crucial to translation quality. Moreover, it's difficult to determine the ordering of the translated segments. Misu et al. (2012) trains an SMT model using conversational in-domain parallel data, and then translates the entire L1 SLU training corpus to L2. However, bilingual in-domain data is scarce and costly, making it difficult to deliver both quality and low cost. Jabaian et al. (2010; 2013) propose two language transferring schemes. One is transferring source language annotation indirectly through word alignment. The other one is forcing the SMT model to translate the segmentation and slot labels simultaneously. The authors report that the indirect alignment gives the best performance. However, they also point out that distant language pairs suffer severely in word alignment. Finally rather than relying on automatic machine translation, Stepanov et al. (2014) prefer using human professional translation services. In Stepanov et al. (2017), they extend their work via crowdsourcing for semantic annotation.

Methods	Source input	Translation result
Naive Translation	我想打个电话给白晓霞	I would like to make a call to telephone number of white sunshine
Token-added Translation	我想打个电话给（a 白晓霞）	I would like to make a call to (a white sunshine) 's telephone number please
Class-based / SCRT	我想打个电话给 $contact_name	I would love to make a call to $contact_name 's number please

Table 1: Translation examples by different translation systems.

3 Translation Systems

This section describes all the translation systems conducted in our experiments. Each system has its own approach of transferring slot labels to the target sentence. They all start from the same general-purpose NMT, trained on a huge L1:L2 parallel corpus without SLU annotation. Hence the general translator usually doesn't do well with input containing slot labels.

3.1 Naive Translation

The naive translation system uses the given well-trained general-purpose translator as it is. The slot labels in the L1 SLU corpus are stripped off before entering the general translator. We then align the words between the source sentence and the translated sentence, in order to add back slot labels into the translated sentence. This is an indirect slot transferring approach.

Some limitations of the naive translation approach come from both translation and alignment. There are plenty of slot values like song names and contact names in the original SLU corpus. Many of these words are out-of-vocabulary (OOV) for the translation model. Although the translation model can handle these OOV words with sub-word modeling techniques (Sennrich et al., 2016), there are still many slot values remain to be OOV or mistranslated. For example, the Chinese name "白晓霞" in Table 1 is literally translated to "white sunshine". Furthermore, wrong translations might also result in wrong alignments, which will yield inaccurate positions for slot labels.

3.2 Token-added Translation

To make the translator be aware of slots, we then propose a token-added translation approach. This approach uses some special tokens to mark the segmentation boundary for the slot value in the source sentence. These special tokens are common in both the source vocabulary and target vocabulary of the general translator and their translation is unique and easy to spot. For example, parentheses and double quotes are good candidates as the special tokens. Enclosing slot values in source sentences by these special tokens can help identify slot boundaries in the translation outputs. In our example in Table 1, the special tokens we choose is a pair of parentheses, where the beginning parenthesis is followed by a single character "a", representing the slot label (in this case contact_name). We choose a single English letter for Chinese-to-English transfer because we are almost certain that the English letter will be carried to the target side as it is. After translation, the slot values can then be easily identified.

In token-added translation, no additional word alignment process is required. However such approach relies heavily on the NMT general training data where the special tokens (e.g. parentheses or double quotation marks) are kept in both source and target data. For different language pairs, different special tokens might be chosen for the best translation quality. Empirically we find that parentheses with a single English character are highly effective for Chinese to English translation.

3.3 Class-based Translation

To better translate the slots, we further propose a class-based translation approach. It is to use a class symbol to replace both the slot label and its slot value in the source sentence, for example, "Play $song in $album". In other words, we generalize the source sentence into a pattern sentence.

The class symbol represents the slot label, but without any specific value, as shown in the last row in Table 1. Representing each slot segment with a single symbol has a great advantage of avoiding a multi-word segment to be translated into several non-consecutive segments and not enclosed by the correct slot-label pairs. In order to help the general translator understand these new words (class symbols), we need some new parallel sentences with class symbols to adapt the general translator. Then for each sentence in the L1 SLU corpus, we first transform it to use class symbols and then translate it with the adapted general translator. The translated sentence will contain class symbols as well. We then replace these class symbols with slot values appropriate to the target culture.

Therefore unlike the first two systems, the class-based translation model is an adapted translator. It requires a small parallel annotated SLU corpus for adaptation.

3.4 Source-Critical Reinforcement Translation

The performance of class-based model relies on whether parallel annotated SLU data is enough or not to translate class symbols correctly. In other words, slots will be missed or mistranslated without enough parallel data. However, in-domain SLU parallel data is scarce. It's best if we can take advantage of the existing L1 SLU corpus to solve the slots missing or mistranslated problem of class-based method. Assume the general translator has been adapted into a class-based translation model, we now propose another adaptation algorithm based on reinforcement learning that depends on monolingual instead of bilingual SLU corpora. Hence we name it source-critical reinforcement translation (SCRT).

Before formally introducing SCRT, it is necessary to first define SKR, a metric named slot keeping ratio for evaluating the slot transferring performance of translation models. For each (Chinese) sentence c_i, its SKR is defined as the number of slots in the translated (English) sentence, e_i, divided by the number of slots in c_i:

$$\text{SKR}(c_i, e_i) = \frac{\sum_s \min(g(c_i, s), g(e_i, s))}{\sum_s g(c_i, s)} \times 100\%$$

The function $g(c_i, s)$ is used to count the occurrences of slot s in sentence c_i. Minimum is taken at the numerator to avoid SKR to be over 100%.

It's easy to see that a high SKR is a critical condition for generating a high-quality training corpus for L2 SLU models. Our SCRT algorithm directly optimizes the SKR for each source sentence, and learns how to translate specific words according to specific rules. In this task, specific words are those class symbols, and specific rules are the correspondence of each class symbol in the two languages. This is inspired by Ranzato et al. (2015) and Rennie et al. (2017). Instead of promoting sequence-level training performance via directly optimizing BLEU scores that require ground truth target sentences, our algorithm focuses on the performance of specific words only, and therefore ground-truth target sentences are not required. The architecture of this model is depicted in Figure 1.

Since SCRT is based on REINFORCE algorithm (Williams, 1992; Zaremba and Sutskever, 2015), we first describe reinforcement learning from the perspective of sequence generation.

3.4.1 REINFORCE

In order to directly maximize SKR, we can cast our problem in the reinforcement learning framework (Sutton and Barto, 1998) as in Ranzato et al. (2015). Our NMT model can be viewed as an *agent* that interacts with an external *environment* (words). The parameters of neural network, θ, defines a *policy* p_θ, whose execution results in an *action*. The *action* is the prediction of the next word in the sequence at each time step. After taking an action, our agent will update its internal *state* (attention weights). Once our agent generates the end-of-sequence (EOS) token, the agent will observe a *reward*. We can choose different reward functions for various purposes. Here, we use the SKR value as the reward r.

During training, the agent chooses an action according to the current policy and observes a reward only when the end token is generated. This reward is computed according to the generated sentence and the input sentence. The goal of training is to maximize the expected reward, and therefore the loss function (negative of reward) for the encoder-decoder given an input source sentence, $x = x_1, x_2, ...$, is

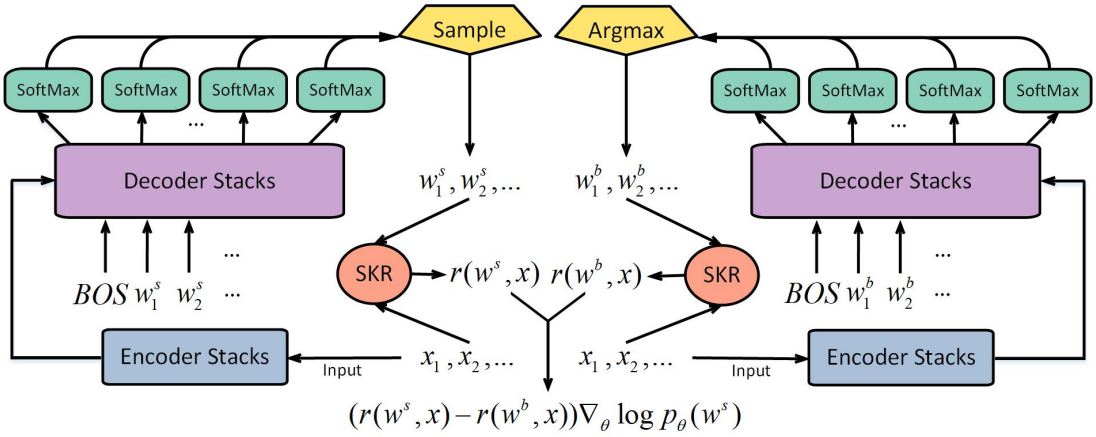


Figure 1: Architecture of Source-Critical Reinforcement Translation. $x_1, x_2, ...$ is the source input of the NMT model and $(r(w^s, x) - r(w^b, x))\nabla_\theta \log p_\theta(w^s)$ is the final gradient to optimize with policy gradient method. The encoder and decoder stacks represent inner layers of the Transformer general translator.

defined as:

$$\begin{aligned} L(\theta) &= -\sum_{w_1^g,\ldots,w_T^g} p_\theta(w_1^g, ..., w_T^g|x)\, r(w_1^g, ..., w_T^g, x) \\ &= -\mathbb{E}_{w^g \sim p(\theta)} r(w^g, x) \end{aligned} \tag{1}$$

where $w^g = (w_1^g, ..., w_T^g)$ is the generated sentence of length T. Notice the length of x is not necessarily T. The notion $w^g \sim p(\theta)$ means the generated sentence follows the distribution of the decoder softmax output $p(\theta)$.

The gradient of $L(\theta)$ can be computed as follow:

$$\nabla_\theta L(\theta) = -\mathbb{E}_{w^g \sim p(\theta)} r(w^g, x) \nabla_\theta \log p_\theta(w^g) \tag{2}$$

We approximate this expectation with Monte Carlo based sampling and a baseline b is used to reduce the variance in the Monte Carlo estimator of the gradient, following Weaver and Tao (2001):

$$\begin{aligned} \nabla_\theta L(\theta) &= -\mathbb{E}_{w^s \sim p(\theta)} (r(w^s, x) - b) \nabla_\theta \log p_\theta(w^s) \\ &\approx -(r(w^s, x) - b) \nabla_\theta \log p_\theta(w^s) \end{aligned} \tag{3}$$

where w^s is the sampled sentence. We use a single sample to approximate the expectation in the above formula. Using the chain rule of gradient computation, we have:

$$\nabla_\theta L(\theta) = \sum_t \frac{\partial L_\theta}{\partial y_t} \frac{\partial y_t}{\partial \theta} \tag{4}$$

where y_t is the input to the decoder softmax function at time stamp t. Following Zaremba's prior work, the gradient of loss $L(\theta)$ with respect to y_t is given by:

$$\frac{\partial L_\theta}{\partial y_t} = (r(w^s, x) - b)(p_\theta(w_t|h_t) - 1(w_t^s)) \tag{5}$$

where h_t is the input vector to the fully-connected layer before softmax, and $1(w_t^s)$ is the one hot vector for w_t^s. Notice y_t, $p_\theta(w_t|h_t)$, $1(w_t^s)$ are all vectors of length V, where V is the number of output units from the decoder.

3.4.2 Self-Critical and Source-Critical

The baseline reward b is obtained by the current model using the inference algorithm at test time (Rennie et al., 2017). This is called self-critical training. In this work, we choose greedy decoding at test time. That is, the most likely word from the decoder output $p(\theta)$ at each time stamp is selected as the baseline output, as shown on the right-hand side of Figure 1. The greedy translation sentence is denoted as w^b.

The above gradient can now be written as:

$$\frac{\partial L_\theta}{\partial y_t} = (r(w^s, x) - r(w^b, x))(p_\theta(w_t|h_t) - 1(w_t^s)) \tag{6}$$

Since the reward is based on SKR exclusively, no ground truth target sentence is needed. We call this source-critical learning.

3.4.3 Optimization

Equation (6) says that the sampled sentence w^s acts like a surrogate target for our output distribution, $p_\theta(w_t|h_t)$, at time t. Once the sampled sentence achieves a higher SKR than greedy decoding, a positive reward is used for parameter updating; otherwise, a negative reward is used.

The SKR criterion itself does not impose constraints on translation quality. Therefore it needs a well-trained general-purpose NMT system as the initial model. In this paper, we initialize the SCRT model with parameters from the adapted class-based model described in Section 3.3, which also offers good embedding features for the class symbols. Otherwise, the model would be difficult to converge, as the sampling space or action space is enormous.

After initialization, we then use policy gradient methods to find parameters that lead to a large expected reward. As our SCRT only optimizes the SKR score with monolingual data, the curriculum learning (Bengio et al., 2009) strategy is employed in which we begin with the class-based model, and gradually increase the number of SCRT training steps.

4 Experiments

4.1 Experimental Setup

We conduct experiments on Chinese to English language transferring task and evaluate the quality of the translated corpus via domain classification and slot filling tasks.

We annotated 3000 (over time this should be a very big number) Chinese dialogue sentences as the L1 SLU corpus and 1500 English dialogue sentences as the test set for the target language. Our goal is to transfer these 3000 Chinese sentences to English, and then train an English SLU model to handle these 1500 English test sentences.

Next we annotated additional 1500 pairs of parallel sentences for adapting the general translator into the class-based MT. This small parallel SLU corpus is called the adaptation corpus in this section. Finally, the Chinese side of the 3000 sentences of the L1 SLU corpus, together with the Chinese side of the 1500 sentences of the adaptation corpus is used in SCRT training, on top of the adapted class-based MT.

4.1.1 SLU model

The SLU annotated data are collected from communication, navigation and music three domains. It includes seven slots: contact name, contact type, address type, song name, album, feature and artist. As "others" domain is necessary for domain classification in practical applications, we collect additional 40,000 sentences from other scenarios as the fourth domain. For domain classification models, we use the SVM linear classifier with public toolkit LIBLINEAR (Fan et al., 2008). Stop words are removed from the training corpus. Simple N-gram features with a cut-off of 10 are used in domain classification. For slot filling experiments, the CRF++ toolkit (Kudo, 2005) is used. The slot labeling follows the IOB format.

Trans.	Culture A.	SKR	Slot_F1	Dom_Acc
Naive	No	57.13%	45.39	45.87%
Naive	Yes	57.13%	69.78	78.93%
TA	No	60.82%	24.87	55.07%
TA	Yes	60.82%	34.14	81.4%

Table 2: Slot F1 scores and domain accuracy, with naive translation vs. token-added (TA) translation using an unadapted general translator. The second column indicates whether culture adaptation is applied.

4.1.2 The general translator

We choose Transformer (Vaswani et al., 2017) as the general machine translation model. And for naive translation, we use Fast-Aligner (FA-IBM 2) (Dyer et al., 2013) to locate the slot labels for the translated queries. The architecture of our NMT model is the same as Vaswani et al. (2017).

Our training corpus is from AI Challenger: The English-Chinese Machine Translation track[1]. This competition provides over 10 million parallel English-Chinese sentences which were collected from English learning websites and movie subtitles. In our experiments, we extract nearly 8 million pairs of sentences from this corpus to train the baseline machine translation model. After sub-word (Sennrich et al., 2016) preprocessing, the source vocabulary size is 83,000 and the target part is 78,000. Our training batch size is 3,072 and we train the baseline model for a total of 300,000 steps with Adam optimizer (Kingma and Ba, 2014) on two GPUs. All decodings are conducted with a beam size of 4, and the top one translation is taken as the final translation output.

4.1.3 Culture adaptation

In our experiments, we select the culture-dependent slots such as contact name, song name, album and artist for target culture adaptation. We first collect thousands of English names, artists, song names and albums to build a database. The slot values in the test data are removed from the database. In the translated target queries, the corresponding slots are filled by randomly selected slot values from the database. The same random seed is used for all approaches.

4.2 Results

4.2.1 Without parallel adaptation corpus

We first compare the performance without the parallel adaptation data. Table 2 shows that the performance of naive translation method and token-added translation method. Note that the class-based model and the SCRT trained model are not included in this table, because the class-based translator needs the adaptation data to learn the translation of class symbols, and SCRT training is built on top of the class-based model.

In Table 2, compared with naive translation method, the token-added approach achieves a higher SKR score. However, keeping more slots doesn't yield good performance in the slot filling task. More mistranslations may happen in the token-added approach due to the fact that the added special tokens actually change the context of the original query and introduce noisy information. On the other hand, the token-added approach has a slightly better domain accuracy than the naive translation. This is probably because slot labels carry a certain degree of domain information.

In Table 2, the contribution of cultural adaptation is obvious. After filling the corresponding slots with culture appropriate slot values, substantial improvements are observed in this table. Adding culture adaptation gives naive translation method more than 50% relative improvement over slot filling F1 score and domain classification accuracy. The similar trend also holds for token-added translation.

4.2.2 With parallel adaptation corpus

In this section, we first compare different methods that use the adaptation corpus that consists of 1500 in-domain annotated sentences in communication, navigation and music domains. In the adaptation

[1]https://challenger.ai/datasets/translation

Trans.	Culture A.	SKR	Slot_F1	Dom_Acc
Naive	No	57.50%	56.79	68.73%
Naive	Yes	57.50%	70.79	81.4%
TA	No	**98.55**%	90.20	66.86%
TA	Yes	**98.55**%	91.91	82.87%
Class-based	Yes	97.03%	93.04	82.12%
+SCRT	Yes	98.08%	**97.19**	**84.2**%

Table 3: Results after adapting all translators with the additional 1200 parallel SLU sentences.

Trans.	Culture A.	SKR	Slot_F1	Dom_Acc
Naive	No	57.2%	50.15	76.73%
Naive	Yes	57.2%	70.13	**81.93**%
TA	No	88.22%	84.96	59.8%
TA	Yes	88.22%	85.21	78.53%
Class-based	Yes	85.07%	83.94	76.8%
+SCRT	Yes	**88.6**%	**91.07**	81.46%

Table 4: Results after adapting all translators with the additional 90 parallel SLU sentences.

corpus, 1200 sentences are randomly selected for training and the rest as the validation data to terminate training. Although 1200 sentences are tiny for model training, we also conduct the same experiments with a much small amount of data, 90 parallel sentence pairs which are also randomly selected from the 1500 in-domain annotated sentences, to test the performance of different methods under different condition.

In Table 3, comparing the naive translation with the token-added translation, we can find that the token-added method benefits more from adapting general purpose translator with in-domain parallel data. The SKR of naive translation barely increased, even using the adapted translator. It is obvious that the main limitation of naive translation is alignment rather than translation. The class-based model achieves 97.03% SKR after the convergence of supervised training with parallel adaptation data. Finally, SCRT training with 4500 monolingual annotated data is further applied to the adapted class-based translator. The SKR is increased to 98.08%. The F1 score of slot filling and accuracy of domain classification also jump significantly, which undoubtedly proves that our SCRT can generate better SLU training corpus for other languages, with the aid of monolingual annotated data exclusively.

In Table 4, the SCRT achieves the highest F1 score of slot filling and the slot transferring or SKR still benefits from monolingual data with SCRT incremental training. This result shows that our proposed method is very competitive even with an extremely small amount of parallel in-domain data. But the naive translation method with culture adaptation show a better performance on domain classification. This is because, with an extremely small amount of parallel training data, it is hard to achieve a high quality of translation for SCRT and class-based method, which is crucial for domain classification. Besides, the naive translation always benefits more from the culture adaptation in domain classification, as those mis-aligned segments will be replaced with correct slots, which makes such sentences more distinguish from the other.

4.2.3 Analysis of SCRT

In order to evaluate how much the proposed SCRT contributes to the class-based model, we analyze the systems at different iterations of training with 1200 in-domain parallel sentences. In the experiments, we find that the class-based model begins to converge after 800 steps with a batch size of 32 when adapted on the 1200 in-domain parallel sentence pairs. So this model at step 800 is chosen as a baseline and is labeled as SL800 in Figure 2. Based on such a baseline, we use the 4500 sentences of monolingual data to conduct 30 steps (SL800+RL30) and 60 steps (SL800+RL60) SCRT RL training. As shown in Figure 2, the SKR increases nearly one percent and the F1 score increases by over 7 percents for both models.

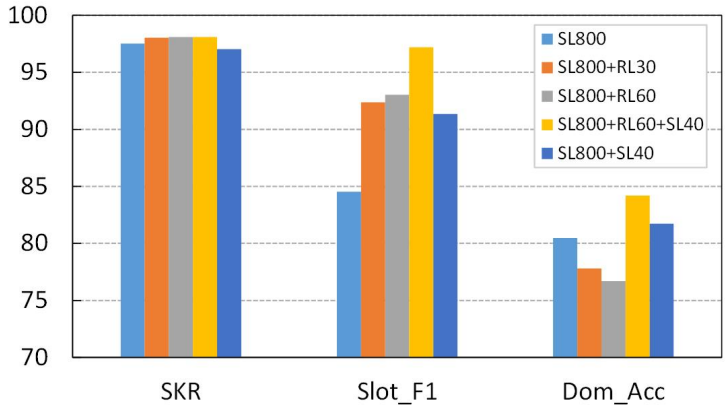
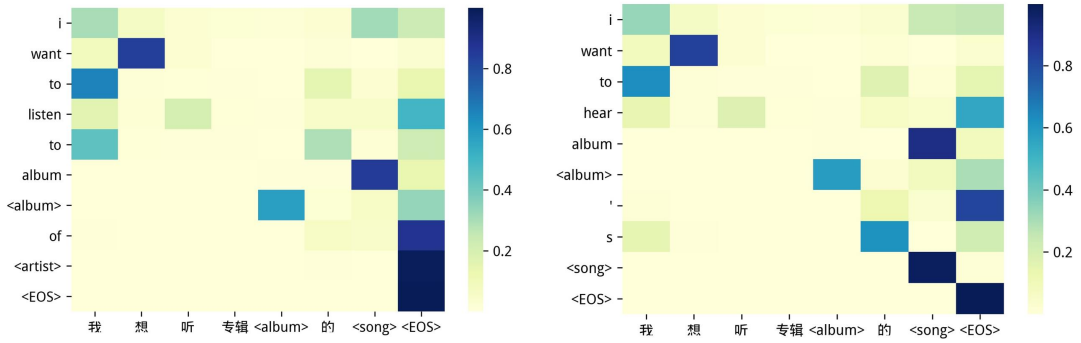


Figure 2: Performance on SKR, slot filling and domain classification, using translation models trained at different steps of supervised learning (SL) and reinforcement learning (RL).



(a) Class-based (b) SCRT

Figure 3: Two heat maps of attention vectors between encoder and decoder in the translation model. The x-axis and y-axis of each plot correspond to the words in the source sentence (Chinese) and the generated translation (English), respectively.

However, we note that the accuracy of domain classification drops. This is because SKR imposes no constraints for general machine translation quality that is crucial for domain classification. The translation quality can be improved by additional steps of supervised training using in-domain parallel training data. For model SL800+RL60+SL40, another 40 steps of supervised training are conducted on top of SL800+RL60.

As we can see from Figure 2, SL800+RL60+SL40 achieves the highest scores on both slot filling and domain classification among all these models. The last model, SL800+SL40, is used for comparison with SL800+RL60+SL40. Although these two models are trained with the same steps of supervised learning, the model with SCRT achieves higher scores on all metrics. This result indicates the potential contribution of the monolingual data and the effectiveness of our proposed SCRT algorithm for language transferring. Furthermore, comparing SL800 and SL800+SL40, we can find that although the accuracy and F1 score goes up, the SKR goes down after additional training with parallel data. This is the shortcoming of maximum-likelihood estimation objective function for our task: treating all words equally without emphasizing important words like slot labels.

The proposed approach SCRT provides an intuitive way to guide the parameter updating for slots transferring with monolingual data. We visualize the attention weights between encoder and decoder of the translation model in Figure 3. The left plot corresponds to the model SL800 mentioned above and the right one is the model SL800+RL60. From the heat maps, we can see which positions in the source sentence were considered more important when generating the target word. Figure 3(a) shows that class-based model mistranslated the slot $song into $artist, and was corrected after additional SCRT

training in Figure 3(b).

5 Conclusions

Our work is motivated by the practical demand in language transferring for SLU systems: the lack of large annotated in-domain parallel data, and the requirement of high-quality SLU corpus in the target language. To address this problem, we applied an adapted Neural Machine Translator to translate the SLU copora to other languages. A small in-domain parallel data is used to adapt the general purpose NMT firstly. Based on the adapted NMT, we proposed a reinforcement learning approach with a source-critical mechanism to do further adaptation using monolingual data exclusively. Our proposed method optimizes the slot keeping ratio directly and adapts slot values accordingly based on the target culture. The experiments showed that comparing with naive translation, the proposed method could improve domain classification accuracy by relatively 22%, and the slot filling F1 score by more than 71%.

Acknowledgements

The research work descried in this paper has been supported by the National Key Research and Development Program of China under Grant No. 2017YFC0822505 and the Natural Science Foundation of China under Grant No. 61673380.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *International Conference on Machine Learning (ICML)*, pages 41–48.

Marcos Calvo, Fernando García, Lluís-F Hurtado, Santiago Jiménez, and Emilio Sanchis. 2013. Exploiting multiple hypotheses for multilingual spoken language understanding. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning (CoNLL)*, pages 193–201.

Marcos Calvo, Lluís-Felip Hurtado, Fernando Garcia, Emilio Sanchis, and Encarna Segarra. 2016. Multilingual spoken language understanding using graphs and multiple translations. *Computer Speech & Language*, 38:86–103.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of NAACL-HLT*, pages 644–648.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research (JMLR)*, 9(Aug):1871–1874.

Fernando García, Lluís F Hurtado, Encarna Segarra, Emilio Sanchis, and Giuseppe Riccardi. 2012. Combining multiple translation systems for spoken language understanding portability. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 194–198.

Fernando Garcıa, Marcos Calvo, Emilio Sanchis, Lluıs-F Hurtado, and Encarna Segarra. 2014. Obtaining parallel corpora for multilingual spoken language understanding tasks. In *In Proceedings of the Iberspeech*.

Xiaodong He, Li Deng, Dilek Hakkani-Tur, and Gokhan Tur. 2013. Multi-style adaptive training for robust cross-lingual spoken language understanding. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8342–8346.

Bassam Jabaian, Laurent Besacier, and Fabrice Lefevre. 2010. Investigating multiple approaches for slu portability to a new language. In *Eleventh Annual Conference of the International Speech Communication Association (Interspeech)*, pages 2502–2505.

Bassam Jabaian, Laurent Besacier, and Fabrice Lefevre. 2013. Comparison and combination of lightly supervised approaches for language portability of a spoken language understanding system. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(3):636–648.

Bassam Jabaian, Fabrice Lefèvre, and Laurent Besacier. 2016. A unified framework for translation and understanding allowing discriminative joint decoding for multilingual speech semantic interpretation. *Computer Speech & Language*, 35:185–199.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Taku Kudo. 2005. Crf++: Yet another crf toolkit. *Software available at http://crfpp. sourceforge. net*, page 130.

Teruhisa Misu, Etsuo Mizukami, Hideki Kashioka, Satoshi Nakamura, and Haizhou Li. 2012. A bootstrapping approach for slu portability to a new language by inducting unannotated user queries. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4961–4964.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *Computer Science*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1179–1195.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1715–1725.

Christophe Servan, Nathalie Camelin, Christian Raymond, Frédéric Béchet, and Renato De Mori. 2010. On the use of machine translation for spoken language understanding portability. In *2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 5330–5333.

Evgeny A Stepanov, Ilya Kashkarev, Ali Orkan Bayer, Giuseppe Riccardi, and Arindam Ghosh. 2013. Language style and domain adaptation for cross-language slu porting. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 144–149.

Evgeny A Stepanov, Giuseppe Riccardi, Ali Orkan Bayer, et al. 2014. The development of the multilingual luna corpus for spoken language system porting. In *LREC*, pages 2675–2678.

Evgeny A Stepanov, Shammur Absar Chowdhury, Ali Orkan Bayer, Arindam Ghosh, Ioannis Klasinas, Marcos Calvo, Emilio Sanchis, and Giuseppe Riccardi. 2017. Cross-language transfer of semantic annotation via targeted crowdsourcing: task design and evaluation. *Language Resources and Evaluation*, pages 1–24.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6000–6010.

Lex Weaver and Nigel Tao. 2001. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the Seventeenth conference on Uncertainty in Artificial Intelligence (UAI)*, pages 538–545.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32.

Wojciech Zaremba and Ilya Sutskever. 2015. Reinforcement learning neural turing machines-revised. *Computer Science*.

A Prospective-Performance Network to Alleviate Myopia in Beam Search for Response Generation

Zongsheng Wang[1], **Yunzhi Bai**[2*], **Bowen Wu**[1], **Zhen Xu**[3*], **Zhuoran Wang**[1,4], **Baoxun Wang**[1]
[1]Tricorn (Beijing) Technology Co., Ltd, Beijing, China
[2]Telecom ParisTech, Paris, France
[3]Harbin Institute of Technology, Harbin, China
[4]Institute of Internet Industry, Tsinghua University, Beijing, China
[1] {wangzongsheng, wubowen, wangzhuoran, wangbaoxun}@trio.ai
[2] yunzhi_bai@outlook.com
[3] zxu@insun.hit.edu.cn

Abstract

Generative dialog models usually adopt beam search as the inference method to generate responses. However, small-width beam search only focuses on the limited current optima. This deficiency called myopic bias ultimately suppresses the diversity and probability of generated responses. Although increasing the beam width mitigates the myopic bias, it also proportionally slows down the inference . To alleviate the myopic bias in small-width beam search, this paper proposes a Prospective-Performance Network (PPN) to predict the future reward of the given partially-generated response, and the future reward is defined by the expectation of the partial response appearing in the top-ranked responses given by a larger-width beam search. Enhanced by PPN, the decoder can promote the results with great potential during the beam search phase. The experimental results on both Chinese and English corpora show that our method is capable of increasing the quality and diversity of generated responses, with inference efficiency well maintained.

1 Introduction

In recent years, Neural Response Generation (NRG) (Vinyals and Le, 2015; Shang et al., 2015) with sequence-to-sequence (Seq2Seq) structures (Sutskever et al., 2014a; Bahdanau et al., 2015) has been widely studied and adopted in open-domain dialog systems such as XiaoIce (Shum et al., 2018). Many studies have been done to generate target responses meeting desirable proprieties including emotion (Zhou et al., 2017), diversity (Li et al., 2016), etc., or to explore issues in decoding step such as exposure bias, loss-evaluation mismatch (Ranzato et al., 2016) and label bias (Wiseman and Rush, 2016).

Most NRG systems adopt the beam search algorithm to generate responses given queries. In brief, beam search explores the possible responses by storing only top-ranked ones as candidates at each time step. Though it is a useful prediction strategy, depending on beam width, beam search more or less suffers from its nature of focusing solely on current optimal results. Consequently, beam search tends to ignore some partial sequences which might lead to better future outcomes, especially if its beam width is too small. This deficiency is called the *myopic bias* (He et al., 2017). Recently, He et al. (2017) and Li et al. (2017) proposed methods that take account of future BLEU of partial sequences as the future reward during beam search, to alleviate the myopic bias in Neural Machine Translation (NMT). Their experiments show that such methods are capable of improving the BLEU scores of generated translations.

However, several studies point out that BLEU is weakly correlated with human judgments in response generation tasks (Liu et al., 2016; Mou et al., 2016). Unlike in machine translation, where the semantic

[*] Contribution dring internship at Tricorn Technology.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3608–3618
Santa Fe, New Mexico, USA, August 20-26, 2018.

	distinct-1	distinct-2	log-probability	relevance
beam width = 10	0.2831	0.4277	-9.2474	0.7500
beam width = 50 (top10)	**0.4151**	**0.5536**	**-7.4490**	**1.0934**

Table 1: Evaluation results of beam search with width 10 and 50

distribution of appropriate translations given a source sentence is narrow, in response generation tasks, the semantic information of possible responses for one query can be highly diverse. Therefore it is inappropriate to use BLEU, which only takes responses in the training dataset as ground truth, as future reward to solve the myopic bias on response generation. Otherwise, the diversity of generated results might be suppressed.

In this paper, we introduce a new perspective for reducing the myopic bias in response generation. In NRG, the degree of myopic bias for beam search is negatively correlated with its beam width: a greedy search (beam width = 1) myopically stores only the top candidate at each time step; while a larger-width beam search is capable of storing more candidates with potentially higher future probability. Therefore, in this work, for a partial response generated from a small-width beam search, we define its future reward as if it will present in the top responses generated from a beam search with a larger width in future time steps. This presence indicates the potential probability one partial response's successors can reach in the future time steps, therefore taking it as future reward captures the future probability of one generated partial sequence from beam search.

Furthermore, given a query, we design a simple but effective neural network to estimate the future reward for a partial response. Based on this prediction, we re-rank the generated partial responses at each time step, so that we encourage beam search to consider the future probability information of each partial response and generate final results similar to those from a larger width beam search, without proportionally increasing the time cost.

2 Beam Width Analysis

2.1 Beam Search Overview

Given a query $\mathbf{x}$, a K-width beam search stores K candidates denoted as $C_t^K = \{\mathbf{y}_t^k |\ k \in [1, K]\}$ at time step t. At next time step $t+1$, it expands each candidate by words w from vocabulary V. The size of vocabulary V is denoted as $|V|$, so that we have in total $K \times |V|$ potential candidates written as $\{[\mathbf{y}_t^k, w] \mid k \in [1, K], w \in V\}$, with corresponding scores:

$$\text{score}(\mathbf{y}_t^k, w|\mathbf{x}) = \text{score}(\mathbf{y}_t^k|\mathbf{x}) + \log p(w|\mathbf{x}, \mathbf{y}_t^k)\ , k \in [1, K], w \in V \tag{1}$$

The top-K potential candidates are then selected as the candidates in time step $t+1$.

2.2 Influence of Beam Width on Generated Responses

The size of beam width affects results returned from a beam search significantly. Let us assume that there are two beam searches, one with a small beam width K_s and the other with a large beam width K_l. At each time step t, we have two sets of candidates $C_t^{K_s} = \{\mathbf{s}_t^{k_s} |\ k_s \in [1, K_s]\}$ and $C_t^{K_l} = \{\mathbf{l}_t^{k_l} |\ k_l \in [1, K_l]\}$ from corresponding two beam searches. The top-K_s results from the K_l-width beam search tend to benefit from the following properties:

- Higher Probability: For a $\mathbf{l}_t^{k_l}$ ranked lower than K_s in $C_t^{K_l}$, its successors might be ranked above top-K_s in the future time steps, as long as one of its future transition probabilities $p(w|\mathbf{l}_t^{k_l}, \mathbf{x})$ is high enough. Such responses, unfortunately can not be retrieved by the small-width beam search due to its limited beam width. Consequently, for those top-K_s responses from a large-width beam search, their probabilities are more likely to be higher in average. Since in Seq2Seq model, the ideal output given a query x is the response y with the maximum $p(y|x)$, a response with higher probability indicates that it is closer to the optimal result.

- Higher Diversity: In the small-width beam search, many responses in $C_{t+1}^{K_s}$ are generated from a same $\mathbf{s}_t^{k_s}$, because of a dominate score of $\mathbf{s}_t^{k_s}$ compared to other candidates. By contrast, in the large-width beam search, we have a higher chance to observe more top-K_s of $C_{t+1}^{K_l}$ generated from different $\mathbf{l}_t^{k_l}$, since there exist more potential candidates $\mathbf{l}_t^{k_l}$ with high future probabilities $p(w|\mathbf{l}_t^{k_l}, \mathbf{x})$. Therefore the larger beam width leads to a higher diversity in generated sequences generally.

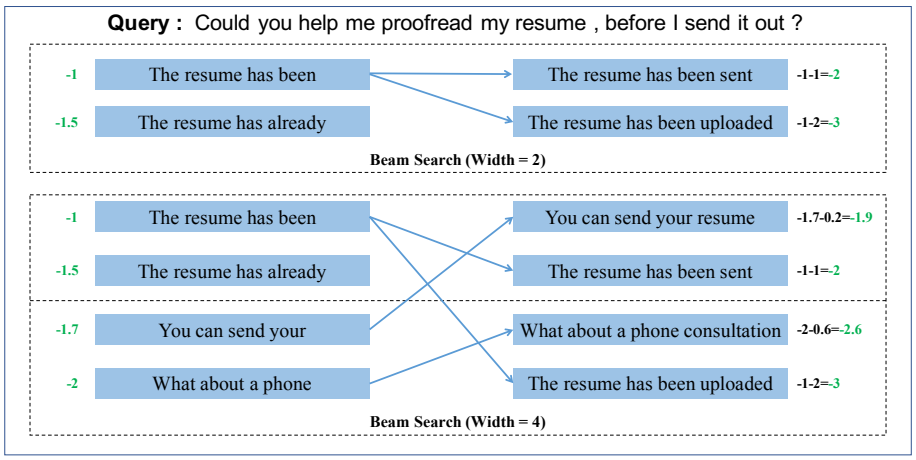


Figure 1: An example of partial responses generated from beam searches with width 2 and 4. The values in green are the corresponding log-probabilities of partial responses.

Figure 1 shows an example where larger beam width helps to generate more diversified results with higher probabilities. Benefited from the more stored candidates, the top-2 generated results from beam search (width =4) are more diversified and have higher probabilities.

To analyze the impact of the beam width on generated responses quantitatively, we employ evaluation methods including distinct, log probability and human evaluation to evaluate the top-10 responses generated by a Seq2Seq model with beam width 10 and 50 respectively. Distinct defined in Li et al. (2016) captures the level of diversity within the generated responses, log-probability is the probability computed by Seq2Seq model during inference after logarithm transformation. The details of above metrics and model training are further described in Section 4.4. The evaluation results in Table 1 show that the top-10 responses from beam width = 50 have higher diversities and probabilities, and are ranked higher by annotators, which is consistent with our hypothesis of beam width.

3 Prospective-Performance Network for NRG

Although a large-width beam search generates responses with higher probability and diversity, increasing beam width proportionally slows down the inference process. Therefore, to retrieve better responses and meanwhile maintain the inference efficiency, we propose Prospective-Performance Network to estimate the future rewards of partial responses in the inference procedure of NRG. The estimated future rewards is then incorporated in the small-width beam search to simulate the performance of a large-width beam search.

3.1 Future Reward

Given a K_l-width beam search we want to simulate, at time step t it generates a set of partial responses $C_t^{K_l} = \{\mathbf{y}_t^{k_l}, k_l \in [1, K_l]\}$. For one partial response $\mathbf{y}_t$, its future reward with regard to beam width of K_l is defined as

$$v(\mathbf{y}_t|\mathbf{x}, K, K_l) = \begin{cases} 1 & \text{if } \mathbf{y}_t \text{ in the top-}K \text{ responses (truncated at } t) \text{ from } C_{t+n}^{K_l} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

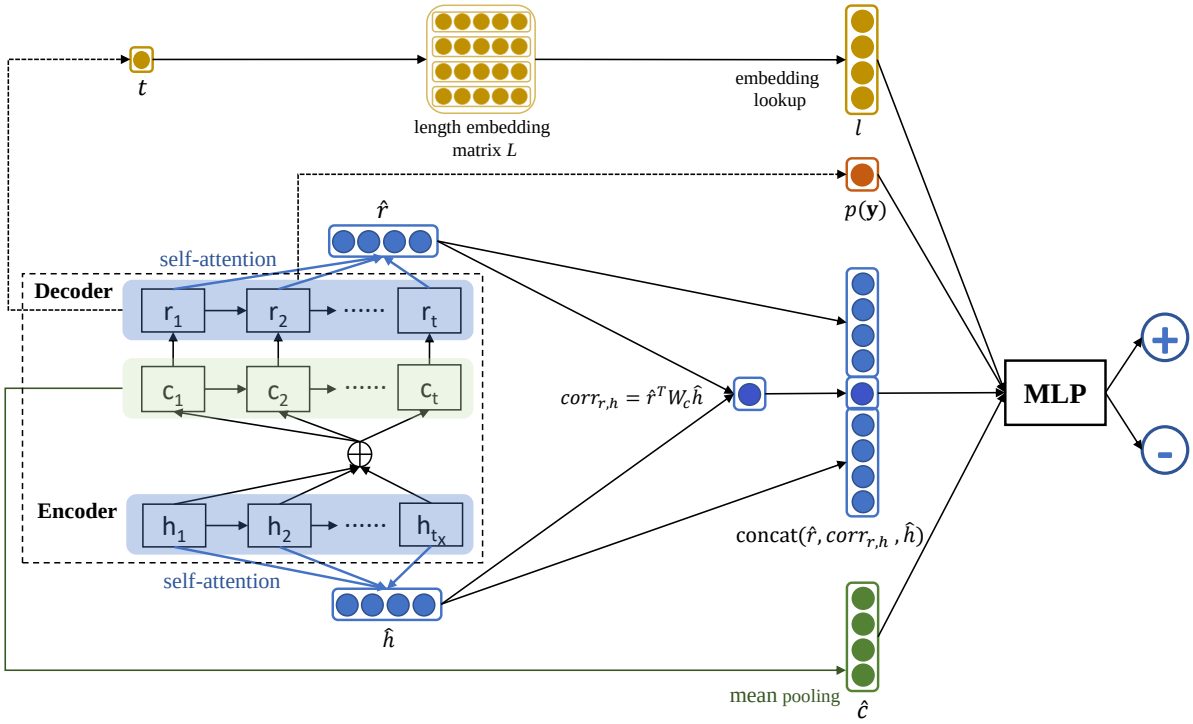


Figure 2: Structure of Prospective-Performance Network

In other words, if a partial response presents in the top-K responses from the given K_l-width beam search ($K < K_l$) at future time steps, we assign it with a positive future reward. The lookahead factor n indicates the prospective level of system.

For the example in Figure 1 where K=2, K_l=4, t=4 and n=1, at time step 4, partial responses "You can send your" and "The resume has been" are attributed with future rewards of 1, because their successors "You can send your resume" and "The resume has been sent" are the top-2 results at the next time step; while "The resume has already" and "What about a phone" are attributed with future rewards of 0. We expect that after reranking the results using future rewards, "You can send your" and "The resume has been" can be ranked at top-2 at time step 4, so that their successors, which have a higher probability at time step 5, can be retrieved through a beam search with beam width of only 2.

3.2 Prospective-Performance Network Structure

In practice, given one partial response and its query, its future reward is unknown without results from a large-width beam search as references. Therefore, we propose Prospective Performance Network (PPN) as a future reward estimator. To fully exploit the information from Seq2Seq encoder-decoder framework in decoding process, PPN is designed with the following four components:

- *Semantic Component:* The Semantic Component captures the semantic information of queries and partial responses. Firstly, to extract most semantics from queries and partial responses, it adopts self-attention mechanism to project encoder hidden states $[h_1, h_2, \ldots, h_{T_x}]$ and decoder hidden states $[r_1, r_2, \ldots, r_t]$ into $\hat{h}$ and $\hat{r}$, specifically:

$$
\begin{aligned}
u_i^h &= tanh(W_a^h h_i + b_a^h) \\
a_i^h &= \frac{\exp((u_i^h)^T u_w^h)}{\sum_{i=1}^{T_x} \exp((u_i^h)^T u_w^h)} \\
\hat{h} &= \sum_{i=1}^{T_x} a_i^h h_i
\end{aligned}
\qquad \text{and} \qquad
\begin{aligned}
u_i^r &= tanh(W_a^r h_i + b_a^r) \\
a_i^r &= \frac{\exp((u_i^r)^T u_w^r)}{\sum_{i=1}^{t} \exp((u_i^r)^T u_w^r)} \\
\hat{r} &= \sum_{i=1}^{t} a_i^r r_i
\end{aligned}
\tag{3}
$$

where W_a^h ,W_a^r, b_a^h, b_a^r, u_w^h and u_w^r are the self-attention parameters. In addition, bilinear transformation is used to further catch the correlation between $\hat{h}$ and $\hat{r}$, such that $corr_{r,h} = \hat{r}^T W_c \hat{h}$. Then the Semantic Component concatenates $\hat{h}$, $corr_{r,h}$ and $\hat{r}$ as a semantic information representation s.

- *Attention Component:* In the Attention Component, mean pooling is used to transfer the context $[c_1, c_2, \ldots, c_t]$ into context representation $\hat{c} = \frac{1}{t}\sum_{i=1}^{t} c_i$. This representation extracts the attention context from the Seq2Seq model.
- *Length Component:* In general, the information provided by short and long partial responses is significantly different from each other. Thus the Length Component is created to summarize the length information of partial responses, it transfers the response length into a length vector l by a length embedding matrix L, different for each time step.
- *Probability Component:* The probability of one partial response $\mathbf{y}$ largely determines the generative probabilities of its successors, therefore Probability Component is employed to extract the current probability $p(\mathbf{y})$ of each input partial response. It adds the log-probability scores for the current partial response, as an singe floating point number.

Finally, PPN concatenates all representations s, $\hat{c}$, l and $p(\mathbf{y})$ as the input of a multilayer perceptron, to estimate the future reward of a partial response $\mathbf{y}$. The whole procedure can be formulated as follows:

$$u = [l,\ p(\mathbf{y}),\ s,\ \hat{c}] \tag{4}$$

$$\hat{v}(\mathbf{y}|\mathbf{x}, K, K_l) = \sigma(W_{mlp}u + b_{mlp}) \tag{5}$$

The estimated future reward $\hat{v}(\mathbf{y}|\mathbf{x}, K, K_l)$ is used as part of the ranking scores in beam search (see details in Subsection 3.4).

3.3 Training Data Generation

Since we aim to generate the top-K_s responses from K_l-width beam search using a smaller search space of K_s, PPN is trained using samples generated from the K_l-width beam search, so that it can be used to estimate the partial responses' future rewards with regard to beam width of K_l. As mentioned in section 2.2, top-K_s responses from K_l-width beam search benefit from properties of higher probability and diversity. Therefore K here is set as K_s, so that a partial response with positive future reward is also associated with above desirable properties. The process of training data generation for PPN is shown in Algorithm 1.

Algorithm 1 Generate PNN training data

Input Small beam width K_s, large beam width K_l, maximum search depth L, candidates in every time step in large beam search $C_t = \{\mathbf{y}_t^1, \mathbf{y}_t^2, \cdots, \mathbf{y}_t^{K_l}\}$, lookahead factor n.

1: **Initialization:** Set $Pset = \emptyset$ as positive samples set, Set $Nset = \emptyset$ as negative samples set, $C_{pre} = \emptyset$ as predecessors set, $t = 0$ as initial time step.
2: **repeat**
3: $\quad t = t + 1$
4: $\quad C_{pre} \leftarrow \{\mathbf{y}[1:t] \mid \mathbf{y} \in C_{t+n}\}$
5: $\quad Pset \leftarrow C_t \cap C_{pre}[1 : K_s]$
6: $\quad Nset \leftarrow C_t \cap C_{pre}[K_s : K_l]$
7: **until** $t + n = L$
8: **Output:** $Pset, Nset$

After training data generation, samples from positive samples set are labeled as 1, while those from negative samples set are labeled as 0. The sum of cross-entropy loss is taken as the loss function to optimize PPN.

3.4 Inference using PPN

The output of PPN provides information related to the future performance of a partial response in the K_l-width beam search, therefore integrating it into the decoding step of a K_s-width beam search helps to alleviate the myopic bias.

For a partial response generated from a K_s-width beam search, its generative probability $P(\mathbf{y}|\mathbf{x})$ is combined with its future reward $\hat{v}(\mathbf{y}|\mathbf{x}, K_s, K_l)$ estimated by PPN. For efficiency, we only compute future rewards on K_l candidates with the highest probabilities at each time step. Among these K_l candidates, those top-K_s responses with the highest combined scores:

$$score(\mathbf{y}|\mathbf{x}) = \log P(\mathbf{y}|\mathbf{x}) + \alpha \times \log \hat{v}(\mathbf{y}|\mathbf{x}, K_s, K_l) \tag{6}$$

are chosen as the candidates after re-ranking. The detailed inference process is shown in Algorithm 2.

Algorithm 2 Beam search with PPN

Input Input query $\mathbf{x}$, Seq2Seq model $P(\mathbf{y}|\mathbf{x})$, vocabulary V, small beam width K_s, large beam width K_l, PNN model $\hat{v}(\mathbf{y}|\mathbf{x}, K_s, K_l)$, maximum search depth L, hyperparameter α.

Initialization: Set $S = \emptyset$ as output set, $C = \emptyset$ as candidate sets, $t = 0$ as time step.
repeat
  $t = t + 1$
  $C_{expand} \leftarrow \{\mathbf{y}_i + [w] \mid \mathbf{y}_i \in C, w \in V\}$
  $C_{K_l} \leftarrow \{\text{top } K_l \text{ candidates that maximize } \log P(\mathbf{y}|\mathbf{x}) \mid \mathbf{y} \in C_{expand}\}$
  $C \leftarrow \{\text{top } (K_s - |S|) \text{ candidates that maximize } \log P(\mathbf{y}|\mathbf{x}) + \alpha \times \log \hat{v}(\mathbf{y}|\mathbf{x}, K_s, K_l) | \mathbf{y} \in C_{K_l}\}$
  $C_{complete} \leftarrow \{\mathbf{y} \mid \mathbf{y} \in C, \mathbf{y}[-1] = EOS\}$
  $C \leftarrow C \backslash C_{complete}$
  $S \leftarrow S \cup C_{complete}$
until $|S| = K_s$ *or* $t = L$
Output: $\mathbf{y} = argmax_{\mathbf{y} \in S \cup C} \log P(\mathbf{y}|\mathbf{x})$

3.5 Time Analysis

In the decoding step of Seq2Seq, its time complexity is dominated by the vocabulary size $|V|$, because of the large-scale computation in the operation of probability distribution projection; while the time complexity of implementing PPN is much smaller than that of decoding, with the help of the simple structure of PPN. Therefore, the time complexity of incorporating PPN in K_s-width beam search is on the same scale as that of the vanilla K_s-width beam search, which means incorporating PPN into beam search preserves the inference efficiency.

4 Experiment

4.1 Dataset

To evaluate our approach, we conduct experiment on a Chinese SNS corpus, which contains single-turn dialogue sessions crawled from a Chinese social network service (SNS)[1]. The preprocessing strategy applied on this dataset is following that of Wu et al. (2017). To further evaluate the effect of our method, we also implement experiment on the large-scaled OpenSubtitles Dataset[2] (Lison and Tiedemann, 2016). After that, we obtain approximately 3,000,000 and 50,000 query-response pairs from the Chinese SNS dataset, 15,000,000 and 50,000 query-response pairs from the OpenSubtitles datasets for training and validating respectively.

[1]The Chinese SNS corpus is confidential in the working organization of the authors.
[2]http://www.opensubtitles.org/

4.2 Hyperparameters

4.2.1 Seq2Seq Model

The parameters settings of the two Seq2Seq models trained using the Chinese SNS and OpenSubtitles datasets are mostly the same: the word embedding size and attention size are both set as 512, both the encoder and decoder are composed of single Long-Short-Term-Memory (LSTM) of hidden size = 512. Seq2Seq models are optimized using Adam with learning rate 0.001 and trained for 10 epochs with batch size 512. The vocabulary sizes for Chinese SNS and OpenSubtitles dataset are set to 50,000 and 40,000 respectively, and all the out-of-vocabulary words are replaced with the *"<UNK>"* token. The maximum sentence lengths at inference step are set to 15 and 30 for the Chinese SNS and OpenSubtitles dataset respectively.

4.2.2 PPN Model

The parameters of Seq2Seq structure in PPN models are retained from pre-trained Seq2Seq models and set as untrainable. In this experiment, we set K_s as 10 and K_l as 50, and α in the inference step is set as 0.5. By setting the beam width as 50, and taking 1 million randomly sampled queries from the training samples of the two datasets as inputs, we obtain a group of responses from Seq2Seq outputs, which are then used to generate the training samples following Algorithm 1. In total 12,000,000 (11,000,000) PPN training samples and 10,000 (10,000) validation samples are generated from the Chinese SNS (OpenSubtitles) dataset, with positive/negative ratio of roughly 1. Adam with learning rate 0.001 is used to optimize the PPN model, the batch size is set as 256 and the models are trained for 3 epochs. The self-attention size and position-embedding size in PPN are set as 256 and 128 respectively.

In addition, the lookahead factor n is set as 1 for performance reason. In our previous experiment, the performance of PPN when n=2 is slightly lower than the case when n=1, and drops significantly when n$>$2.

4.3 Baselines

The following models are used as the comparisons with our proposed PPN:

- *Basic Seq2Seq model* with encoder-decoder structure and vanilla beam searches (Luong et al., 2015) is constructed as the standard baseline.
- *Value Network (VN)* introduced in (He et al., 2017) takes the $\hat{v}(\mathbf{y}|\mathbf{x}, K, K_l)$ defined in Section 3.1 as the future reward instead of future BLEU proposed in the original paper. VN and PPN are trained using the same samples and adopted the same loss function.

  In VN, due to its semantic matching module and context coverage module, it contains two more fully connected layers than PPN. Therefore under the same hyperparameter scale, VN is slower at inferencing step than PPN because of its more complicated network structure.
- *Maximum Mutual Information (MMI)* (Li et al., 2016) is a popular diversity-promoting method which takes maximum mutual information as the objective function. Since the PPN is expected to promote the diversity of generated responses, MMI is included to compare with it on diversity-promoting.

  The MMI variant used in our experiment is the MMI-antiLM.

The training and inference process for all models in our experiments are carried out under the same computational environment: a single Nvidia K80 GPU.

4.4 Evaluation Metrics

The proposed PPN, together with other baselines, are automatically evaluated in terms of the **similarity** toward large-width beam search, **diversity** and **efficiency**.

- **Similarity**: We measure the successfulness of one model with a small beam width K_s (denoted as *model-bw-K_s*) on approximating a standard large-width K_l beam search (*Seq2Seq-bw-K_l*) using

	Similarity		Diversity		Efficiency
	Coverage	**Log-prob**	**Distinct-1**	**Distinct-2**	**Time Cost**
Seq2Seq-bw50-top10	-	-7.4490	0.4151	0.5536	1.9496
Seq2Seq-bw10	0.4009	-9.2474	0.2831	0.4272	**0.4399**
MMI-bw10	0.4166	-8.9286	0.2858	0.4323	0.6233
VN-bw10	0.5030	-8.3672	0.3183	0.4715	0.6027
PPN-bw10	**0.5515**	**-8.0738**	**0.3555**	**0.5107**	0.5667

Table 2: Automatic evaluation results on the Chinese SNS dataset.

	Similarity		Diversity		Efficiency
	Coverage	**Log-prob**	**Distinct-1**	**Distinct-2**	**Time Cost**
Seq2Seq-bw50-top10	-	-5.0762	0.4338	0.4585	2.2996
Seq2Seq-bw10	0.4103	-6.9216	0.3341	0.4295	**0.4131**
MMI-bw10	0.4363	-6.8118	0.3240	0.4184	0.6089
VN-bw10	0.4880	-6.4787	0.3550	0.4449	0.7058
PPN-bw10	**0.5980**	**-5.8012**	**0.3711**	**0.4613**	0.5827

Table 3: Automatic evaluation results on the OpenSubtitles dataset.

coverage and **log-probability**. Coverage of *model-bw-*K_s is defined as the ratio of its generated responses presented in top-K_s responses from *Seq2Seq-bw-*K_l. A model with a higher coverage indicates it generates more responses ranked top by the compared large beam search. Log-probability is the mean probability of generated top-K_s responses after logarithm transformation ($log(p(\mathbf{y}|x))$) during inference. Generally speaking, a model with a closer log-probability compared to *Seq2Seq-bw-*K_l is more desirable.

- **Diversity**: Diversity of generated responses from each model is measured by the **distinct-1** and **distinct-2**, which are calculated respectively by the number of distinct unigrams and bigrams in the set of generated responses divided by the total number of generated tokens (Li et al., 2016).
- **Efficiency**: To evaluate the inference efficiency of each model, we also compare the **time cost** in seconds on generating the responses set given one query.

Besides automatic evaluations, the qualities of generated responses from each model are manually evaluated in terms of **relevance** and **grammar correctness**. In total, 300 query-response pairs generated from each model trained using the Chinese SNS dataset are randomly sampled. For each query-response pair, 3 annotators are invited to evaluate its grammatical correctness as 0 (grammatically incorrect) or 1 (grammatically correct), and relevance as 0 (irrelevant), 1 (acceptable) or 2 (great).

5 Results

5.1 Automatic Evaluation

Table 2 and 3 show the automatic evaluation results on the Chinese SNS and OpenSubtitles datasets. Here *Seq2Seq-bw10* and *Seq2Seq-bw50-top10* stand for two basic Seq2Seq models with beam width of 10 and 50 respectively. In addition, only top-10 ranked responses in *Seq2Seq-bw50-top10* are taken into account when evaluating. *MMI-bw10*, *VN-bw10* and *PPN-bw10* refer to the MMI, VN and PPN model respectively, and their beam widths are also set as 10.

It can be observed that *PPN-bw10* obtains the highest coverage and closest log-probability compared to *Seq2Seq-bw50-top10* on both datasets, which indicates that our proposed PPN is capable of simulating a larger-width beam search. In addition, with the help of more explicit method to capture semantics along with the length and probability information from corresponding components, the proposed PPN model is more effective on capturing the future reward than the VN, reflected by the higher coverage of *PPN-bw10* compared to *VN-bw10*.

	Grammar	Relevance
Seq2Seq-bw50-top10	0.8709	1.0934
Seq2Seq-bw10	0.6236	0.7500
MMI-bw10	0.6280	0.7680
VN-bw10	0.7135	0.8567
PPN-bw10	**0.8022**	**0.9505**

Table 4: Human evaluation results on the Chinese SNS dataset. The scores are means over 300 samples.

In terms of diversity, PPN also significantly improves the distinct-1 and distinct-2 compared to the vanilla beam search. It is worth noting that the distinct-1 and distinct-2 of *PPN-bw10* are higher than those of *MMI-bw10*. The PPN exploits the nature of high-diversity in large-width beam search to promote the diversity of responses, and the experiment results show some evidence that such method is more effective than MMI on diversity promotion.

As expected, the time cost of *Seq2Seq-bw50-top10* is approximately 5 times longer than that of *Seq2Seq-bw10*, while *PPN-bw10* only raises the inference time by around 10% meanwhile achieves a significant improvement on quality of responses compared to *Seq2Seq-bw10*. It proves the feasibility of our proposed PPN in practice.

The p-values of PPN against three baseline models on coverage, log-probability, distincts are all smaller than 5e-6, which indicates that the improvements of performance from PPN are significant.

5.2 Human Evaluation

The human evaluation result is shown in Table 4. The responses from *Seq2Seq-bw50-top10* and *PPN-bw10* are annotated with the highest and the second highest grammatical correctness and relevance. The result further reinforces our deduction that a larger beam width improves the quality of generated responses, and the PPN is capable of approximating a large beam width search. In addition, we test the consistency of the human evaluation using Fleiss' kappa (Fleiss and Cohen, 1973). For grammatical correctness and relevance, their Fleiss' kappa on all models are around 0.6 and 0.4 respectively, which can be both considered as "moderate agreement". The p-values of PPN against three baseline models on grammar and relevance scores are all below 0.05.

5.3 Further Analysis

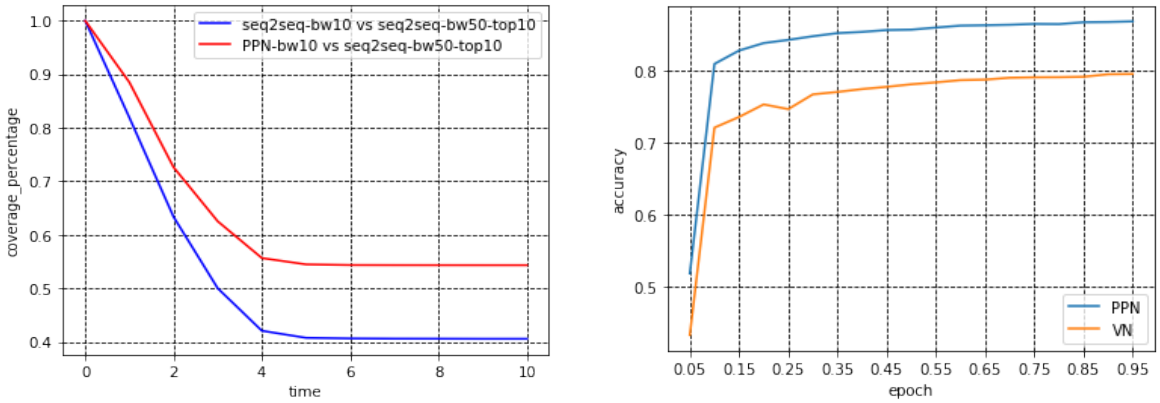


Figure 3: Left: Coverage evolution between PPN and vanilla Seq2Seq over time step. Right: Accuracy evolution on validation set during training of PPN and VN, Chinese SNS dataset.

To analyze the performance of PPN in detail, two plots regarded to the comparisons of coverage and prediction accuracy between PPN and other baseline models are shown in Figure 3. It can be observed in the left plot of Figure 3 that the coverages of *PPN-bw10* are always higher than those of *Seq2Seq-bw10*,

which indicates that PPN generates more top responses from large-width beam search consistently over time. Moreover, the right plot in Figure 3 shows the accuracy of PPN on classifying the samples in validation dataset throughout the training of PPN and VN. The higher accuracies of PPN further prove that PPN is more effective on estimating the future reward than VN.

6 Related Work

Inspired by the success of the Seq2Seq framework on NMT (Cho et al., 2014; Sutskever et al., 2014b; Bahdanau et al., 2015), this framework has been adopted for response generation (Vinyals and Le, 2015; Shang et al., 2015) and is proved to be effective on generating responses based on given queries (Sordoni et al., 2015; Li et al., 2016; Serban et al., 2016; Xu et al., 2017).

Most NMT and NRG systems generate outputs using the beam search algorithm, which unfortunately suffers from the myopic bias. To solve the myopic bias, He et al. (2017) and Li et al. (2017) both propose method to take the future BLEU of decoder partial outputs into account in beam search. Another study indirectly related to our work is Wiseman and Rush (2016), it treats the target sequences in training set as the gold sequences, and directly training the beam search to select word instead of probability. Although these methods are proved to be effective on NMT, it might be inappropriate to directly apply them on NRG, since appropriate responses for one query are highly diverse in terms of semantics. By contrast, the proposed method exploits the nature of beam search width to alleviate the myopic bias.

7 Conclusions

In this paper, we have described our attempt on reducing the myopia in beam search for NRG. In detail, we have: 1) verified the effectiveness of increasing the beam width on relieving the myopic bias; 2) proposed a future reward to alleviate the myopic bias of canonical Seq2Seq-based NRG model, and specially designed a perspective-performance network to quantify the future reward reasonably; 3) presented a new decoding strategy on the basis of the perspective-performance network to generate top-ranked responses given by a large-width beam search. The experiment results show the effectiveness and efficiency of our method. The proposed method is especially useful on online conversational agents, where the speed of response generation is of great importance. In the future, to better estimate the future reward, we will explore different model structures and new training data generation strategies.

References

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

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Joseph L Fleiss and Jacob Cohen. 1973. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and psychological measurement*, 33(3):613–619.

Di He, Hanqing Lu, Yingce Xia, Tao Qin, Liwei Wang, and Tieyan Liu. 2017. Decoding with value networks for neural machine translation. In *Advances in Neural Information Processing Systems 30*, pages 177–186.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2017. Learning to decode for future success. *arXiv preprint arXiv:1701.06549*.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3349–3358.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, pages 3776–3784.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1577–1586.

Heung-Yeung Shum, Xiaodong He, and Di Li. 2018. From eliza to xiaoice: Challenges and opportunities with social chatbots. *arXiv preprint arXiv:1801.01957*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of NAACL-HLT*, pages 196–205.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014a. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014b. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *ICML Deep Learning Workshop*.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306.

Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 496–505.

Zhen Xu, Bingquan Liu, Baoxun Wang, SUN Chengjie, Xiaolong Wang, Zhuoran Wang, and Chao Qi. 2017. Neural response generation via gan with an approximate embedding layer. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 617–626.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2017. Emotional chatting machine: emotional conversation generation with internal and external memory. *arXiv preprint arXiv:1704.01074*.

Adaptive Multi-Task Transfer Learning for Chinese Word Segmentation in Medical Text

Junjie Xing **Kenny Q. Zhu** **Shaodian Zhang**
Shanghai Jiao Tong University
{jjxing@,kzhu@cs.,shaodian@apex.}sjtu.edu.cn

Abstract

Chinese word segmentation (CWS) trained from open source corpus faces dramatic performance drop when dealing with domain text, especially for a domain with lots of special terms and diverse writing styles, such as the biomedical domain. However, building domain-specific CWS requires extremely high annotation cost. In this paper, we propose an approach by exploiting domain-invariant knowledge from high resource to low resource domains. Extensive experiments show that our model achieves consistently higher accuracy than the single-task CWS and other transfer learning baselines, especially when there is a large disparity between source and target domains.

1 Introduction

Chinese word segmentation (CWS) is a fundamental task for Chinese natural language processing (NLP). Most state-of-art methods are based on statistical supervised learning and neural networks. They all rely heavily on human-annotated data, which is a time-consuming and expensive work. Specially, for domain CWS, e.g., medical field , the annotation expense is even higher because only domain experts are qualified for the work.

Moreover, CWS tools trained from open source datasets, e.g., SIGHAN2005[1], face a significance performance drop when dealing with domain text. The ambiguity caused by domain terms and writing style makes it extremely difficult to train a universal CWS tool. As shown in Table 1, given a medical term "高铁血红蛋白血症" (methemoglobinemia), Chinese medical experts would annotate it as "高/铁/血红蛋白/血症", which means anemia caused by hemoglobin with "high iron" (in Chinese, means iron with valence of 3), corresponding to the morphology of "Methemoglobinemia". "PKU" stands for a model trained on PKU's People's Daily corpus, we can see that after segmentation, the word "铁血" (jagged) is treated as one word, which is wrong semantically. Also, another popular Chinese CWS tool Jieba [2] mistakenly puts the characters "高" and "铁" together, which stands for the high-speed bullet train in China.

CWS tool	**高铁血红蛋白血症**			
PKU	高 high	铁血 jagged	红蛋白 albumen	血症 anemia
Jieba	高铁 train	血红蛋白 hemoglobin		血症 anemia
Medical	高 high	铁 iron	血红蛋白 hemoglobin	血症 anemia

Table 1: Medical CWS ambiguity with CWS tools. PKU stands for a model trained on PKU dataset.

In summary, domain specific CWS task poses significant challenges because:

Kenny Q. Zhu is the corresponding author.

[1] http://sighan.cs.uchicago.edu/bakeoff2005/

[2] https://github.com/fxsjy/jieba

Proceedings of the 27th International Conference on Computational Linguistics, pages 3619–3630
Santa Fe, New Mexico, USA, August 20-26, 2018.

1. Tools built on open source annotated corpus works badly on domain specific CWS.

2. Annotated domain data is scarce due to high cost.

3. How to leverage open source annotated data despite their generality is an open question.

Recently, efforts have been made to exploit open source (high resource) data to improve the performance of domain specific (low resource) tasks and decrease the amount of domain annotated data (Yang et al., 2017; Peng and Dredze, 2016; Mou et al., 2016). In this paper, we further this line of work by developing a multi-task learning (Caruana, 1997; Peng and Dredze, 2016) framework, named *Adaptive Multi-Task Transfer Learning*. Inspired by the success of *Domain Adaptation* (Saenko et al., 2010; Tzeng et al., 2014; Long and Wang, 2015b), we propose to minimize distribution distance of hidden representation between the source and target domain, thus make the hidden representations *adapt* to each other and obtain domain-invariant features. Finally, we annotated 3 medical datasets from different medical departments and medical forum, together with 3 open source datasets[??]. The contribution of this paper can be summarized as follows:

- We propose a novel framework for Chinese word segmentation in the medical domain.
- To the best of our knowledge, we are the first to analyze the performance of transfer learning methods against the amount of disparity between target/source domains.
- Our framework outperforms strong baselines especially when there is substantial *disparity*.
- We open source 3 medical CWS datasets from different sources, which can be used for further study.

2 Related Work

2.1 Chinese word segmentation

Statistical Chinese word segmentation has been studied for decades. Xue and others (2003) was the first to treat it as a sequence tagging problem, using a maximum entropy model. Peng et al. (2004) achieved better results by using a conditional random field model (Lafferty et al., 2001). This method has been followed by many other works (Zhao et al., 2006; Sun et al., 2012).

Recently, neural network models have been applied on CWS. These methods use automatically derived features from neural network instead of hand-crafted discrete features. Zheng et al. (2013) first adopted neural network architecture to CWS. Chen et al. (2015b) used Long short-term memory(LSTM) to capture long term dependency. Chen et al. (2015a) proposed a gated recursive neural network (GRNN) to incorporate context information. In this paper, we adopt Bidirectional LSTM-CRF Models (Huang et al., 2015) as our base model.

2.2 Transfer Learning

Transfer learning distills knowledge from source domain and helps target domain to achieve a higher performance (Pan and Yang, 2010). In feature-based models, many transfer approached have been studied, including instance transfer (Jiang and Zhai, 2007; Liao et al., 2005), feature representation transfer (Argyriou et al., 2006; Argyriou et al., 2007), parameter transfer(Lawrence and Platt, 2004; Bonilla et al., 2007) and relation knowledge transfer(Mihalkova et al., 2007; Mihalkova and et al., 2009).

Recently, the transferability of neural networks is also studied. For example, (Mou et al., 2016) studied two methods (INIT, MULT) on NLP applications. Peng and Dredze (2016) proposed to use domain mask and linear projection upon multi-task learning (MTL) (Long and Wang, 2015a). In this paper, we follow MTL and extend the framework with a novel loss function.

3 Single-Task Chinese word segmentation

In this section, we briefly formulate the Chinese word segmentation task and introduce our base model, Bi-LSTM-CRF (Huang et al., 2015).

3.1 Problem Formulation

Chinese word segmentation is often treated as a sequence tagging problem on character level. BIES tagging scheme is broadly accepted by annotators, each character in sentence is labeled as one of $\mathcal{L} = \{B, I, E, S\}$, indicating begin, inside, end of a word, and a word consisting of a single character.

Given a sequence with n characters $X = \{x_1, \ldots, x_n\}$, the aim of the CWS task is to find a mapping from X to $Y^* = \{y_1^*, \ldots, y_n^*\}$:

$$Y^* = \arg\max_{Y \in \mathcal{L}^n} p(Y|X) \tag{1}$$

where $\mathcal{L} = \{B, I, E, S\}$

The general architecture of neural CWS contains: (1) a character embedding layer; (2) an encoder automatically extracts feature and (3) a decoder inferences tag from the feature.

In this paper, we utilize a widely-used model as the base of our framework, which consists of a bi-directional long short-term memory neural network (BiLSTM) as encoder and conditional random fields (CRF) (Lafferty et al., 2001) as decoder.

3.2 Encoder

In neural network models, an encoder is usually adopted to automatically extract feature instead of human-crafted feature engineering.

Bi-LSTM LSTM is a popular variant of RNN in order to alleviate the vanishing gradient problem (Bengio et al., 1994; Hochreiter and Schmidhuber, 1997). In addition to considering *past* information from left, Bidirectional LSTM also captures *future* information from the right of the token.

3.3 Decoder

We deploy a conditional random fields layer as decoder. Specifically, $p(Y|X)$ in Eq. (1) could be formulated as

$$p(Y|X) = \frac{\exp(\Phi(X, Y))}{\sum_{Y' \in \mathcal{L}^n} \exp(\Phi(X, Y'))} \tag{2}$$

Here, $\Phi(\cdot)$ is a potential function, consider the situation that we only take the influence between two consecutive variables into account:

$$\Phi(X, Y) = \sum_{j=1}^{n} \phi(X, i, y_i, y_{i-1}) \tag{3}$$

$$\phi(X, i, y_i, y_{i-1}) = s(X, i)_{y_i} + t_{y_i y_{i-1}} \tag{4}$$

where $s(X, i) \in \mathbb{R}^{|\mathcal{L}|}$ is a function that measure the score of the i_{th} character for each label in $\mathcal{L} = \{B, I, E, S\}$, and $t \in \mathbb{R}^{|\mathcal{L}| \times |\mathcal{L}|}$ denotes the transition score between labels. More formally:

$$s(X, i) = \mathbf{W}^\top h_i + \mathbf{b} \tag{5}$$

where h_i is the hidden state of the i^{th} character after BiLSTM; $\mathbf{W} \in \mathbb{R}^{d_h \times |\mathcal{L}|}$ and $\mathbf{b} \in \mathbb{R}^{|\mathcal{L}|}$ are all parameters in the model.

4 Adaptive Multi-Task Transfer Learning

With the motivation to leverage domain-invariant knowledge from high resource domain, we utilize the framework of multi-task learning (Caruana, 1997), which is one of the methods in *transfer learning*, and further introduce three models under the proposed *Adaptive Multi-Task Transfer Learning* framework (**AMTTL**). We exploit three statistical distance measures as the *Adaptive* part to test the generality of our framework.

4.1 Notations and Definitions

In this paper, multi-task learning is defined as a *dual-task* learning, which contains two *domains* $\mathcal{D}_S$ and $\mathcal{D}_T$. Our purpose is to improve the performance of *target domain* by exploiting knowledge from *source domain*.

Each domain $\mathcal{D}$ contains two components: a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, where X is a sample sentence, and $X = \{x_1, \ldots, x_n\} \in \mathcal{X}$.

Given a single domain, $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a *task* contains two components: a label space $\mathcal{Y}$ and a predictive function $f(\cdot)$, which can be learned during the training phase. Formally, $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$.

4.2 Formal Definition

We now give the definition of *Adaptive Multi-Task Transfer Learning*.

Definition 4.1. Given two domains $\mathcal{D}_S$ and $\mathcal{D}_T$, and corresponding tasks $\mathcal{T}_S$, $\mathcal{T}_T$, *Adaptive Multi-Task Transfer Learning* aims to improve the learning of target predictive function $f_T(\cdot)$ by using *shared parameter* and *minimizing the distance* between $P(X_S)$ and $P(X_S)$, $P(Y_S|X_S)$ and $P(Y_T|X_T)$, where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$.

4.3 Objective Function

The objective function of our proposed *Adaptive Multi-Task Transfer Learning* can be formulated as follows:

$$\mathcal{J}(\theta^{(a)}, \theta^{(b)}) = \mathcal{J}_{seg} + \alpha \mathcal{J}_{Adap.} + \beta \mathcal{J}_{L_2} \tag{6}$$

where $\theta^{(a)}$ and $\theta^{(b)}$ are model parameters for task a and b, α and β are hyper-parameters to be chosen.

$\mathcal{J}_{seg}$ stands for the negative log likelihood for source domain and target domain. At each training step, we minimize the mean negative log likelihood:

$$\begin{aligned} \mathcal{J}_{seg} = &-\frac{1}{n}\sum_{i=1}^{n} \log p(Y_i^{(a)}|X_i^{(a)}) \\ &-\frac{1}{m}\sum_{i=1}^{m} \log p(Y_i^{(b)}|X_i^{(b)}) \end{aligned} \tag{7}$$

$\mathcal{J}_{Adap.}$ is the *Adaptive* loss used to capture domain-invariant knowledge between different domains, which forces the hidden representations between two domains to *adapt* to each other. Given two sets of hidden representation, denoted as $\mathbf{h^{(a)}}$ and $\mathbf{h^{(b)}}$, and a statistic distance function $g(\cdot)$, $\mathcal{J}_{\text{Adap.}}$ can be calculated as:

$$\mathcal{J}_{\text{Adap.}} = g(\mathbf{h^{(a)}}, \mathbf{h^{(b)}}) \tag{8}$$

where $g(\cdot)$ can be, but is not limited to, KL divergence, maximum mean discrepancy (MMD) (Gretton et al., 2012) or central moment discrepancy (CMD) (Zellinger et al., 2017); $\mathbf{h^{(a)}}$ and $\mathbf{h^{(b)}}$ are different for different model setting, which will be defined in Sec 4.4.

$\mathcal{J}_{L_2}$ is the L_2 regularization which is used to control overfitting problem:

$$\mathcal{J}_{L_2} = \left|\left|\theta^{(a)}\right|\right|_2^2 + \left|\left|\theta^{(b)}\right|\right|_2^2 \tag{9}$$

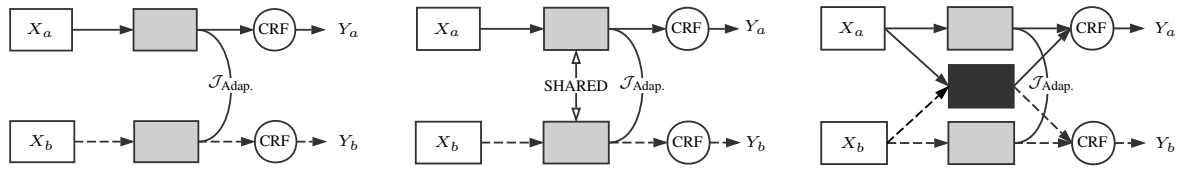
4.4 Models

In this section, we present the design of three variants of our framework in detail. The architectures are presented in Figure 1.

4.4.1 Model-I Specific LSTM

This model can be interpreted as two *parallel tasks* connected with $\mathcal{J}_{\text{Adap.}}$ after specific Bi-LSTM layers of two tasks. We design the model in order to see whether knowledge can actually be transfered through the *Adaptive* loss alone.

The hidden representation and CRF score of *task* t at position i can be computed as:



(a) Model-I (b) Model-II (c) Model-III

Figure 1: Three models with different settings. The white block represents Embedding lookup layer, while the gray and black block represents Bi-LSTM layer. The "SHARED" in Figure 1b stands for shared Bi-LSTM for both tasks. The "$\mathcal{J}_{\text{Adap.}}$" represents *Adaptive* loss for the hidden representation after corresponding layer, which is formally discussed in Sec 4.3. The solid arrow and dotted arrow show the flow of task a and task b respectively.

$$h_i^{(t)} = \text{Bi-LSTM}(X^{(t)}, \theta^{(t)}) \tag{10}$$

$$s(X, i)^{(t)} = \mathbf{W^{(t)}}^\top h_i^{(t)} + \mathbf{b}^{(t)} \tag{11}$$

where $h_i^{(t)} \in \mathbb{R}^{2d_h}$, $\mathbf{W^{(t)}} \in \mathbb{R}^{2d_h \times |\mathcal{L}|}$, $\mathbf{b}^{(t)} \in \mathbb{R}^{|\mathcal{L}|}$, $\theta^{(t)}$ denotes parameters of domain specific Bi-LSTM. The $\mathcal{J}_{\text{Adap.}}$ between two tasks, denoted by a and b, is formulated as:

$$\mathcal{J}_{\text{Adap.}} = g(\mathbf{h}^{(a)}, \mathbf{h}^{(b)}) \tag{12}$$

where $\mathbf{h}^{(t)} = \{h_i^{(t)} | X^{(t)} \in \mathcal{X}^{(t)}\}$, $\mathcal{X}^{(t)}$ is a batch of input sequences.

4.4.2 Model-II Shared LSTM

Model-II is designed to adopt domain specific embedding layers, shared Bi-LSTM layer and domain specific CRF layers. Note that traditional *multi-task learning* uses shared embedding (Ruder, 2017). Shared embedding means that source and target domain share the same set of embedding parameters while domain-specific embedding means that the two domains maintain their own sets.

The hidden representation of *task* t at position i can be computed as:

$$h_i^{(t)} = \text{Bi-LSTM}(X^{(t)}, \theta) \tag{13}$$

where two tasks share Bi-LSTM parameter θ, which is the only difference with Model-I. CRF score and $\mathcal{J}_{\text{Adap.}}$ is the same as (11)(12).

4.4.3 Model-III Shared & Specific LSTM

Model-III is a combination of Model-I and Model-II, with both domain specific and shared Bi-LSTM layers.

The hidden representation and CRF score of *task* t at position i can be computed as:

$$\begin{aligned} h_i^{(t)} &= \text{Bi-LSTM}(X, \theta^{(t)}) \oplus \text{Bi-LSTM}(X, \theta) \\ &= h_{i(specific)}^{(t)} \oplus h_{i(shared)}^{(t)} \end{aligned} \tag{14}$$

$$s(X, i)^{(t)} = \mathbf{W^{(t)}}^\top h_i^{(t)} + \mathbf{b}^{(t)} \tag{15}$$

where $h_i^{(t)} \in \mathbb{R}^{4d_h}$, $\mathbf{W^{(t)}} \in \mathbb{R}^{4d_h \times |\mathcal{L}|}$, and $\mathbf{b}^{(t)} \in \mathbb{R}^{|\mathcal{L}|}$. $\theta^{(t)}$ and θ denote the parameter of domain specific and shared Bi-LSTM. $\mathcal{J}_{\text{Adap.}}$ can be calculated as :

$$\mathcal{J}_{\text{Adap.}} = g(\mathbf{h}^{(a)}, \mathbf{h}^{(b)}) \tag{16}$$

where $\mathbf{h}^{(t)} = \{h_{i(specific)}^{(t)} | X^{(t)} \in \mathcal{X}^{(t)}\}$, $\mathcal{X}^{(t)}$ is a batch of input sequences.

Table 2: Statistics of number of sentences for corpus.

(a) Open Source

Type	#Train	#Dev	#Test
PKU	70498	8369	1945
MSR	173850	19453	3985
WEIBO	38086	3834	16673

(b) Medical

Type	#Train	#Dev	#Test
Cardiology(EMR)	5636	1658	1658
Respiratory(EMR)	5191	1661	1549
Forum	4863	1412	1474
Sum	15690	4731	4691

5 Experiment

In this section, we evaluate our proposed models on real-world medical Chinese word segmentation tasks[3], where annotated data is scarce and domain-drift is significant with open source annotated data. We conduct extensive experiments and discuss the result in detail. We also conduct an ablation test.

5.1 Datasets

We use three open source CWS datasets, namely PKU and MSR from SIGHAN2005 Bakeoff[4] and WEIBO from (Qiu et al., 2016). The information of the datasets is shown in Table 2a.

We annotated three medical datasets for our experiment and future research. The first two datasets are electronic medical records (EMR) from different departments. The third dataset is medical forum data from *Good Doctor Online*[5], which is a Chinese forum for medical consult. The information of the datasets is shown in Table 2b.

The electronic medical records are collected from our partner hospital, the data only permits non-commercial/academical use. The annotation was done by several doctors. It was carried out following a Chinese word segmentation criteria [6] created by Institute of Computational Linguistics at Peking University. For quality control, annotators were trained until they achieve about 80% inter-annotator agreement on previously annotated materials. Then we conducted double-blind annotation, with resolution of disagreements by a senior annotator. The entire annotation process follows Cohen et al. (2017).

On medicolegal issues, the EMR data we received had already been anonymized and de-identified. Given the fact that China doesn't offer an act like HIPPA(Health Insurance Portability and Accountability Act), we only released the medical forum dataset.

5.2 Disparity Study

Transfer Learning aim to improve the performance of low-resource domain task by exploiting the annotated data form high-resource domain, thus the *Disparity* between different tasks is a leading factor to influence the *transferability* between different domains with different methods.

In this paper, we used $\mathcal{X}^2$ test (Kilgarriff and Rose, 1998) to quantify the *Disparity* between three medical corpus. If the size of corpus 1 and corpus 2 are N_1, N_2 and word w has observed frequencies $o_{w,1}$, $o_{w,2}$, then expected value $e_{w,1} = \frac{N_1 \times (o_{w,1}+o_{w,2})}{N_1+N_2}$, and likewise for $e_{w,2}$, then

$$\mathcal{X}^2 = \sum \frac{(o-e)^2}{e} \tag{17}$$

$\mathcal{X}^2$ test shows that *Disparity* between forum dataset and two EMR datasets are similar, but both are much larger than the *Disparity* between the two EMR datasets, as shown in Table 3.

Due to the fact that $\mathcal{X}^2$ test doesn't permit comparison between corpus of different sizes (Kilgarriff and Rose, 1998), we propose a simple *agreement* test, using the size of the intersection between the most common n tokens (bi-gram) to quantify the *disparity* between medical corpus and open source corpus. We set n to 500.

[3]Our code and data are released at `https://github.com/adapt-sjtu/AMTTL`

[4]http://sighan.cs.uchicago.edu/bakeoff2005/

[5]http://www.haodf.com

[6]`http://sighan.cs.uchicago.edu/bakeoff2005/data/pku_spec.pdf`

Table 3: Result of $\mathcal{X}^2$ test between medical datasets, the larger the higher disparity.

Dataset	Cardiology	Respiratory	Forum
Cardiology	0	0.069	0.126
Respiratory	0.069	0	0.122
Forum	0.126	0.122	0

Table 4: Result of *agreement* test between medical datasets and open source datasets, the smaller the higher disparity.

Dataset	Cardiology	Respiratory	Forum
PKU	25	27	76
MSR	23	25	80
WEIBO	54	50	135

Table 5: Performance (F1-score) of Single-task model compared with state-of-art CWS.

Models	Cardiology	Respiratory	Forum
Single-task	81.10	81.33	75.62
(Cai and Zhao, 2016)	80.1	81.5	73.0
(Zhang et al., 2016)	82.46	81.74	77.14

Agreement test shows that the *Disparity* between PKU/MSR and two EMR datasets are close, both far larger than the *Disparity* between PKU/MSR and forum dataset. WEIBO dataset is more similar with medical datasets than PKU and MSR.

5.3 Single-task Performance

Before introducing our experiments on proposed framework, we first evaluate the effectiveness of the single-task model (Bi-LSTM-CRF), which is our base model. We compare the model with the two state-of-art on Chinese word segmentation, proposed by Cai and Zhao (2016) and Zhang et al. (2016) respectively. We run experiments on our datasets with their code released on github[7,8]. The results show that the performance of single-task model and state-of-art are close, as shown in Table 5, which indicates the single-task model is a strong baseline for our advanced models.

5.4 Training

The training phrase aims to optimize the model parameters $\theta^{(a)}$ and $\theta^{(b)}$ by minimizing the objective function defined in Eq. (6). We use Adam (Kingma and Ba, 2014) with mini-batch. Each batch contains sentences from both domains. The hyper-parameter setting is discussed later.

5.5 Experiment Settings

The dimension of character embedding and the LSTM hidden state dimension are 50. The batch size is 30. We evaluate our framework for a total of 15 transfer learning tasks. For each task, we take all of source training data and 10% of target training data. Hyper-parameters are determined by tuning against the development set.

5.6 Baselines

Several baseline methods are compared.
Single-task uses target domain data only, as discussed in Section 3.
INIT fine-tunes the model trained on source domain using target domain data.
Multi-Task shares parameter for both source and target domain, the model is trained simultaneously.
Linear Projection shares encoder and projects hidden representations into specific feature space.
Domain Mask shares encoder and select different part of hidden representation for source and target domains.

[7]https://github.com/jcyk/CWS
[8]https://github.com/SUTDNLP/NNTransitionSegmentor

Table 6: F1-score of 6 cross domain multi-task learning CWS tasks. R, C, F stand for *Respiratory*, *Cardiology*, *Forum* respectively. *Model without Adaptive* are Multi-Task Learning with different setting according to our models.

Method	Cross Medical					
	R→C	F→C	C→R	F→R	C→F	R→F
Baselines						
Single-task	81.10	81.10	81.33	81.33	75.62	75.62
INIT	90.62	87.19	88.88	85.56	79.41	78.53
Linear Projection	85.57	82.95	84.95	84.54	78.25	77.65
Domain Mask	85.01	85.03	85.08	84.74	77.24	78.07
Model-II w/o $\mathcal{J}_{\text{Adap.}}$	86.71	85.27	85.34	83.40	77.62	78.34
Model-III w/o $\mathcal{J}_{\text{Adap.}}$.	84.39	83.59	83.80	83.27	77.18	77.38
Adaptive Multi-Task Transfer Learning-KL						
Model-I	86.94	86.70	85.64	**85.57**	78.35	78.46
Model-II	87.73	87.05	86.65	**86.51**	**79.44**	**78.92**
Model-III	86.66	86.53	85.86	85.39	78.67	**78.72**
Adaptive Multi-Task Transfer Learning-MMD						
Model-I	85.96	85.43	85.45	**85.58**	77.85	78.16
Model-II	87.55	**87.24**	86.17	**86.40**	**79.45**	**78.57**
Model-III	86.30	85.49	85.13	85.19	77.05	77.23
Adaptive Multi-Task Transfer Learning-CMD						
Model-I	86.17	86.03	85.58	**85.83**	78.61	78.39
Model-II	87.49	86.95	86.79	**86.29**	**79.52**	**79.08**
Model-III	86.54	86.36	85.68	**86.05**	78.23	**78.63**

Table 7: F1-score of 9 multi-task learning CWS tasks between open source datasets and medical datasets. R, C, F, P, M, W stand for *Respiratory*, *Cardiology*, *Forum*, *PKU*, *MSR*, *WEIBO* respectively. *Model without Adaptive* are Multi-Task Learning with different setting according to our models.

Method	Open Source - Medical								
	P→C	M→C	W→C	P→R	M→R	W→R	P→F	M→F	W→F
Baselines									
Single-task	81.10	81.10	81.10	81.33	81.33	81.33	75.62	75.62	75.62
INIT	86.20	84.32	87.72	84.05	82.83	86.56	82.54	81.78	84.37
Linear Projection	86.21	86.08	85.35	85.18	84.58	85.27	77.62	77.15	77.54
Domain Mask	85.60	86.17	84.99	84.83	84.16	84.65	77.50	77.46	77.14
Model-II w/o $\mathcal{J}_{\text{Adap.}}$	85.63	85.84	86.14	84.17	85.42	86.09	78.60	78.80	78.32
Model-III w/o $\mathcal{J}_{\text{Adap.}}$	84.43	86.19	85.61	84.38	85.02	85.79	77.61	77.87	78.38
Adaptive Multi-Task Transfer Learning-KL									
Model-I	**86.30**	**86.60**	86.64	**85.66**	**85.44**	85.69	78.55	78.21	78.11
Model-II	**87.01**	**86.20**	86.94	**85.88**	**85.61**	85.96	78.82	78.69	79.37
Model-III	**86.56**	**86.25**	87.29	**85.30**	**85.60**	85.52	78.20	77.45	78.56
Adaptive Multi-Task Transfer Learning-MMD									
Model-I	85.82	**86.62**	86.47	**85.26**	**85.48**	85.87	77.69	78.26	79.01
Model-II	**86.77**	**86.34**	86.82	**85.98**	**86.17**	85.86	79.04	79.21	78.80
Model-III	85.89	85.68	86.59	85.05	85.27	85.64	78.37	78.30	78.39
Adaptive Multi-Task Transfer Learning-CMD									
Model-I	**86.52**	85.93	86.39	**85.71**	85.36	85.97	78.66	78.29	78.49
Model-II	**87.21**	**86.92**	86.83	**85.83**	**85.82**	86.24	78.82	79.01	78.90
Model-III	**86.54**	85.99	86.64	**86.12**	**85.66**	85.63	78.73	78.15	78.71

Our implementation of **INIT** follows Mou et al. (2016), and the implementation of **Multi-Task** follows the models we proposed in Sec. 4 by removing $\mathcal{J}_{\text{Adap.}}$, annotating *Model w/o* $\mathcal{J}_{\text{Adap.}}$ in Table 6 and 7. **Linear Projection** and **Domain Mask** both come from (Peng and Dredze, 2016).

5.7 Hyper-parameter

In our framework, we have two hyper-parameters α and β, which controls the weight of $\mathcal{J}_{\text{Adap.}}$ and $\mathcal{J}_{L_2}$. Our experiments show that $\alpha \in [0.3, 0.7]$ and $\beta \in [0.2, 0.3]$ works best.

5.8 Result and Discussion

Table 6 and Table 7 respectively shows the performance of 6 cross medical CWS experiments and 9 experiments between open source datasets and medical datasets. **Bold** indicates scores that outperforms all baselines. Underline indicates the highest score for each task. In general, we learn that

1. All transfer learning methods outperforms strong baseline of single-task method (discussed in Section 5.3). Especially, our models outperforms from 2% to 6% than single-task baseline.

2. The *Adaptive* part of our model, $\mathcal{J}_{\text{Adap.}}$, is proven to be promising. First, Model-I, which is a parallel training without sharing parameters and leveraging pretrained optimized initialization, outperforms single-task baseline by 4% on average. Second, $\mathcal{J}_{\text{Adap.}}$ improves the performance by 1% on average for both Model-II and Model-III. It shows that the $\mathcal{J}_{\text{Adap.}}$ does capture domain-invariant knowledge apart from the shared parameters.

3. Within the three models we proposed, Model-II performs best, outperforming other two on 40/45 experiment instances. Model-I and Model-III are equal in match. We argue that it is because the missing of shared parameter of Model-I and the possible noise encoded by the specific layer of Model-III.

4. For the three statistic distance measures we test in experiment, the overall performance is close. Compared with MMD and CMD, KL gains a more stable improvement on all experiments. However, CMD performs better to hit more best scores than KL and MMD.

Next, we analyze the result from a special aspect, the *Disparity* between source and target datasets:

1. In Table 6, INIT outperforms all other baselines and our approaches in task R $\rightarrow$ C and C $\rightarrow$ R, but downperforms our approaches in the others. We argue that the effectiveness of INIT on task between domain R and C result from the low *Disparity* between the two domains, as shown in Table 3. We speculate that the INIT approach works so well between domains with low disparity because: (a) well trained model in the source domain provides a good start point for training in the target domain, which is very similar to the source; (b) the final model is fine-tuned against the target domain only. Our method is disadvantaged in this scenario because: (a) our model parameters are randomly initialized and are independent between two domains (except for the shared parameters), thus it cannot inherit so much information from the source domain as INIT does; (b) the final model is fine-tuned against both the source and the target domain at the same time; thus noise from the source domain may be introduced into the target domain. This is a research problem we want to tackle in the future.

2. We first refer to Table 4. We can simply categorize the 9 types of domain transfers into 4 levels. P $\rightarrow$ C, P $\rightarrow$ R, M $\rightarrow$ C and M $\rightarrow$ R indicate high disparity, W $\rightarrow$ C, W $\rightarrow$ R indicate low disparity, P $\rightarrow$ F, M $\rightarrow$ F indicate low similarity, W $\rightarrow$ F indicates high similarity. Then we can find that, in 4 tasks of high disparity, our approach outperforms all baselines. When disparity goes down to the second level, our approach underperforms INIT but only with gap of 0.4%. However, when disparity continuously goes down to the third and forth level, INIT outperforms our approach by 3-4%.

At last, we'd like to discuss the effect of transferring from a general-domain dataset (which has the advantage of larger quantity) against that of transferring from a medical dataset (which is better at quality). After comparing the tasks with the same target domain, we conclude that quality weighs more than quantity. Taking Cardiology as an example, the size of source training set used in cross-medical (high quality) tasks is only 1 percent of that used in general-to-medical (high quantity) tasks, but the cross-medical results still outperform the latter.

5.9 Ablation Test

To investigate the effectiveness of different components in our framework, we do ablation test based on Model-II on task (P $\rightarrow$ R) with $\mathcal{J}_{\text{Adap.}}$ calculated by MMD. Results are reported in Table 8. *Model*-II *w/o shared Bi-LSTM* uses domain-specific Bi-LSTM, while *Model*-II *w/o specific embedding* uses shared embedding for both domains.

Results show that the choice of statistic distance measure weights least, since the performance of different measures are close. The test verifies our choice of *shared Bi-LSTM* and *specific embedding*.

Table 8: Comparisons of different settings of our method.

Settings	F1-score	δ
Model-II + $\mathcal{J}_{\text{Adap.}}$-MMD	85.98	0
Model-II + $\mathcal{J}_{\text{Adap.}}$-KL	85.88	-0.10
Model-II + $\mathcal{J}_{\text{Adap.}}$-CMD	85.83	-0.15
Model-II w/o $\mathcal{J}_{\text{Adap.}}$	84.17	-1.49
Model-II w/o shared Bi-LSTM	85.26	-0.40
Model-II w/o specific embedding	82.09	-3.57

6 Conclusion

In this paper, we propose an adaptive multi-task transfer learning framework and three model instances with different settings. 15 experiments between medical datasets and open source datasets show that: *AMTTL*(1) outperforms multi-task learning all the way; (2) outperforms all baselines when the disparity between target and source dataset is high. For future work, we plan to study the transferability between different tasks for Chinese NLP and cross-lingual NLP tasks.

Acknowledgments

We would like to thank Kevin Bretonnel Cohen for his suggestions through the COLING Writing Mentoring Program.

References

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2006. Multi-task feature learning. In Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006, pages 41–48. MIT Press.

Andreas Argyriou, Charles A. Micchelli, Massimiliano Pontil, and Yiming Ying. 2007. A spectral regularization framework for multi-task structure learning. In Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007, pages 25–32. Curran Associates, Inc.

Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. Trans. Neur. Netw., 5(2):157–166, March.

Edwin V. Bonilla, Kian Ming Adam Chai, and Christopher K. I. Williams. 2007. Multi-task gaussian process prediction. In Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007, pages 153–160. Curran Associates, Inc.

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for chinese. CoRR, abs/1606.04300.

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

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015a. Gated recursive neural network for chinese word segmentation. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers, pages 1744–1753. The Association for Computer Linguistics.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015b. Long short-term memory neural networks for chinese word segmentation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015, pages 1197–1206.

K Bretonnel Cohen, Karin Verspoor, Karën Fort, Christopher Funk, Michael Bada, Martha Palmer, and Lawrence E Hunter. 2017. The Colorado Richly Annotated Full Text (CRAFT) Corpus: Multi-Model Annotation in the Biomedical Domain. In Handbook of Linguistic Annotation, pages 1379–1394. Springer, Dordrecht, Dordrecht.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. J. Mach. Learn. Res., 13:723–773, March.

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

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

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in NLP. In John A. Carroll, Antal van den Bosch, and Annie Zaenen, editors, ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic. The Association for Computational Linguistics.

Adam Kilgarriff and Tony Rose. 1998. Measures for corpus similarity and homogeneity. In Nancy Ide and Atro Voutilainen, editors, Proceedings of the Third Conference on Empirical Methods for Natural Language Processing, Palacio de Exposiciones y Congresos, Granada, Spain, June 2, 1998., pages 46–52. ACL.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. CoRR, abs/1412.6980.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Neil D. Lawrence and John C. Platt. 2004. Learning to learn with the informative vector machine. In Carla E. Brodley, editor, Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004, volume 69 of ACM International Conference Proceeding Series. ACM.

Xuejun Liao, Ya Xue, and Lawrence Carin. 2005. Logistic regression with an auxiliary data source. In Luc De Raedt and Stefan Wrobel, editors, Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005, volume 119 of ACM International Conference Proceeding Series, pages 505–512. ACM.

Mingsheng Long and Jianmin Wang. 2015a. Learning multiple tasks with deep relationship networks. CoRR, abs/1506.02117.

Mingsheng Long and Jianmin Wang. 2015b. Learning transferable features with deep adaptation networks. CoRR, abs/1502.02791.

Lilyana Mihalkova and et al. 2009. Transfer learning from minimal target data by mapping across relational domains.

Lilyana Mihalkova, Tuyen N. Huynh, and Raymond J. Mooney. 2007. Mapping and revising markov logic networks for transfer learning. In Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada, pages 608–614. AAAI Press.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in NLP applications? CoRR, abs/1603.06111.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. IEEE Trans. on Knowl. and Data Eng., 22(10):1345–1359, October.

Nanyun Peng and Mark Dredze. 2016. Multi-task multi-domain representation learning for sequence tagging. CoRR, abs/1608.02689.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In Proceedings of the 20th International Conference on Computational Linguistics, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xipeng Qiu, Peng Qian, and Zhan Shi. 2016. Overview of the NLPCC-ICCPOL 2016 shared task: Chinese word segmentation for micro-blog texts. In Proceedings of The Fifth Conference on Natural Language Processing and Chinese Computing & The Twenty Fourth International Conference on Computer Processing of Oriental Languages.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. CoRR, abs/1706.05098.

Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. 2010. Adapting visual category models to new domains. In Proceedings of the 11th European Conference on Computer Vision: Part IV, ECCV'10, pages 213–226, Berlin, Heidelberg. Springer-Verlag.

Xu Sun, Houfeng Wang, and Wenjie Li. 2012. Fast Online Training with Frequency-Adaptive Learning Rates for Chinese Word Segmentation and New Word Detection. ACL.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. CoRR, abs/1412.3474.

Nianwen Xue et al. 2003. Chinese word segmentation as character tagging. Computational Linguistics and Chinese Language Processing, 8(1):29–48.

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. CoRR, abs/1703.06345.

Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. 2017. Central moment discrepancy (CMD) for domain-invariant representation learning. CoRR, abs/1702.08811.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Transition-based neural word segmentation. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers. The Association for Computer Linguistics.

Hai Zhao, Changning Huang, Mu Li, and Bao-Liang Lu. 2006. Effective tag set selection in chinese word segmentation via conditional random field modeling. In Proceedings of the 20st Pacific Asia Conference on Language, Information and Computation, PACLIC 20, Huazhong Normal University, Wuhan, China, November 1-3, 2006. ACL.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for chinese word segmentation and POS tagging. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL, pages 647–657. ACL.

Addressee and Response Selection for Multilingual Conversation

Motoki Sato[1*] **Hiroki Ouchi**[2,3] **Yuta Tsuboi**[1*]
[1]Preferred Networks, Inc.,
[2]RIKEN Center for Advanced Intelligence Project
[3]Tohoku University
sato@preferred.jp, hiroki.ouchi@riken.jp, tsuboi@preferred.jp

Abstract

Developing conversational systems that can converse in many languages is an interesting challenge for natural language processing. In this paper, we introduce *multilingual addressee and response selection*. In this task, a conversational system predicts an appropriate addressee and response for an input message in multiple languages. A key to developing such multilingual responding systems is how to utilize high-resource language data to compensate for low-resource language data. We present several knowledge transfer methods for conversational systems. To evaluate our methods, we create a new multilingual conversation dataset. Experiments on the dataset demonstrate the effectiveness of our methods.

1 Introduction

Open-domain conversational systems, such as chatbots, are attracting a vast amount of interest and play their functional and entertainment roles in real-world applications. Recent conversational models are often built in an end-to-end fashion using neural networks, which require a large amount of training data (Vinyals and Le, 2015; Serban et al., 2016). However, it is challenging to collect enough data to build such models for many languages. Consequently, most work has targeted high-resource languages, such as English and Chinese (Shang et al., 2015; Serban et al., 2016).

In this work, we aim to develop *multilingual* conversational systems that can return appropriate responses in many languages. Specifically, we assume the two types of systems: (i) **language-specific** systems and (ii) **language-invariant** systems. A language-specific system consists of multiple conversational models, each of which returns responses in a corresponding language. By contrast, a language-invariant system consists of a single unified model, which returns responses in all target languages. A key to building these multilingual models is how to utilize high-resource language data to compensate for low-resource language data. We present several knowledge-transfer methods. To the best of our knowledge, this is the first work focusing on low-resource language enablement of conversational systems.

One challenge when developing conversational systems is how to evaluate the system performance. For generation-based conversational systems, which generate each word for a response one by one, many studies adopt human judgments. However, it is costly and impractical to adopt this evaluation method for multilingual systems, especially for minor-language systems. Thus, as a first step, we develop retrieval-based conversational systems and evaluate the ability to select appropriate responses from a set of candidates. Fig. 1 shows the overview of our multilingual responding systems.

This paper provides: (i) formal task definitions, (ii) several knowledge-transfer methods and (iii) a multilingual conversation dataset. First, we introduce and formalize the two task settings: *single-language adaptation* for language-specific systems and *multi-language adaptation* for language-invariant systems (Sec. 4). Second, we present several methods leveraging high-resource language data to compensate for low-resource language in the two settings (Sec. 5). Our basic method uses multilingual word embeddings and transfers source-language knowledge to target languages (Sec. 5.1 (a)). We also design

*This work was conducted when the first author worked at Nara Institute of Science and Technology, and portions of this research were done while the third author was at IBM Research - Tokyo.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3631–3644
Santa Fe, New Mexico, USA, August 20-26, 2018.

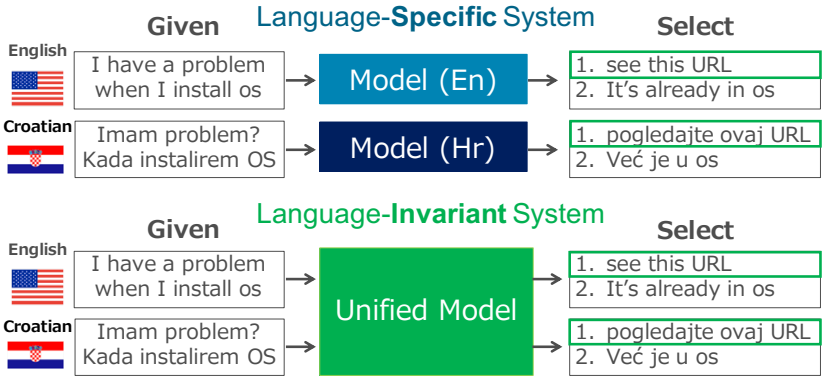


Figure 1: Example of multilingual response selection.

three extended methods. Among them, the fine-tuning method builds a model specific to a single target language (Sec. 5.2 (b)). The joint loss training and the multi-language adversarial training methods build a unified model invariant for multiple target languages (Sec. 5.2 (c) and (d)). Third, we create a multilingual conversation corpus and dataset[1] (Sec. 6). From the Ubuntu IRC Logs[2], we collect the logs in 12 languages.

To show benchmark results, we perform experiments on the created dataset (Sec. 7). The results demonstrate that our methods allow models to effectively adapt to low-resource target languages. In particular, our method using Wasserstein GAN (Arjovsky et al., 2017) achieves high-performance for simultaneously dealing with multiple languages with a single unified model.

2 Related Work

Short Text Conversation

In short text conversation, a system predicts an appropriate response for an input message in *single-turn, two-party conversation* (Ritter et al., 2011). One major approach to it is the generation-based approach, which generates a response using a sequence-to-sequence model (Shang et al., 2015; Vinyals and Le, 2015; Serban et al., 2016; Li et al., 2016; Mei et al., 2017). Another popular approach is the retrieval-based approach, which retrieves candidate responses from a repository and returns the highest scoring one using a ranking model (Wang et al., 2013; Lu and Li, 2013; Ji et al., 2014; Wang et al., 2015). Lowe et al. (2015) proposed next utterance classification (NUC), in which a model has to select an appropriate response from a fixed set of candidates.

Evaluation for Conversational Systems

Evaluation methods for conversational systems are an open question (Lowe et al., 2015; Liu et al., 2016; Lowe et al., 2017). While many of previous studies on conversational systems used human judgements, automatic evaluation methods are attractive because it is much easier and less costly to use. However, according to Liu et al. (2016), for generation-based systems, automatic evaluation metrics, such as BLEU (Papineni et al., 2002), correlate very weakly with human judgements.

For retrieval-based systems, some studies used ranking-based metrics, such as mean average precision and accuracy (Ji et al., 2014; Wang et al., 2015). Lowe et al. (2016) confirmed the feasibility of NUC as a surrogate task for building conversational systems. Although there are controversial issues for these evaluation methods (Lowe et al., 2015), as a practical choice, we adopt the accuracy-based metric for evaluating multilingual conversational systems.

[1]Our code and dataset are publicly available at `https://github.com/aonotas/multilingual_ASR`
[2]http://irclogs.ubuntu.com/

Addressee and Response Selection

NUC focuses on two-party, *multi-turn conversation*. As an extension of it, Ouchi and Tsuboi (2016) proposed addressee and response selection (ARS) for *multi-party conversation*. ARS integrates the addressee detection problem, which has been regarded as a problematic issue in multi-party conversation (Traum, 2003; Jovanović and Akker, 2004; Bohus and Horvitz, 2011; Uthus and Aha, 2013). Mainly, this problem has been tackled in spoken/multimodal dialog systems (Jovanović et al., 2006; Akker and Traum, 2009; Nakano et al., 2013; Ravuri and Stolcke, 2014). While these systems largely rely on acoustic signal or gaze information, ARS focuses on text-based conversations. Extending these studies, we tackle multilingual, multi-turn, and multi-party text conversation settings.

Cross-Lingual Conversation

The motivation of our task is similar with that of Kim et al. (2016). They tackled cross-lingual dialog state tracking in English and Chinese. While they transfer knowledge from English to Chinese, we transfer knowledge between a high-resource and several low-resource languages.

3 Addressee and Response Selection

Addressee and response selection (ARS)[3], proposed by Ouchi and Tsuboi (2016), assumes the situation where a responding agent returns a response to an addressee following a conversational context.

Formally, given an input conversational situation $\boldsymbol{x} \in X$, a system predicts $\boldsymbol{y} \in Y$, which consists of an addressee a and a response $\boldsymbol{r}$:

$$\textsc{Given} : \boldsymbol{x} = (a_{\mathrm{res}},\ \mathcal{C},\ \mathcal{R}), \quad \textsc{Predict} : \boldsymbol{y} = (a, \boldsymbol{r})$$

where a_{res} is a responding agent, $\mathcal{C}$ is a context (a sequence of previous utterances) and $\mathcal{R}$ is a set of candidate responses. To predict an addressee a, we select an agent from a set of the agents appearing in a context $\mathcal{A}(\mathcal{C})$. To predict a response $\boldsymbol{r}$, we select a response from a set of candidate responses $\mathcal{R} = \{\boldsymbol{r}_1, \cdots, \boldsymbol{r}_{|\mathcal{R}|}\}$.

This task evaluates accuracy on the three aspects: addressee-response pair selection (`ADR-RES`), addressee selection (`ADR`), and response selection (`RES`). In `ADR-RES`, we regard the answer as correct if both the addressee and response are correctly selected. In `ADR`/`RES`, we regard the answer as correct if the addressee/response is correctly selected.

4 Multilingual Addressee and Response Selection

As an extension of monolingual ARS, we propose *multilingual addressee and response selection* (M-ARS). In ARS, a system is given as input a set of candidate responses and a conversational context in a single language. By contrast, in M-ARS, a system receives the inputs in one of multiple languages. In the following, we first explain our motivation for tackling M-ARS and then describe the formal task definitions.

4.1 Motivative Situations

We assume the two multilingual conversational situations:

- You want to build *language-specific* systems, each of which responds in a single language.
- You want to build one *language-invariant* system, which responds in multiple languages.

The first situation is that we build K models, each of which is specialized for one of K target languages. The second one is that we build one unified model that can deal with all K target languages. Taking these situations into account, we present the corresponding two tasks: (i) *single-language adaptation* and (ii) *multi-language adaptation*.

[3]Due to the space limitation, we give a brief overview of ARS. For the complete task definition, please refer to Ouchi and Tsuboi (2016).

4.2 Task Overview

The goal of *single-language adaptation* is to develop and evaluate a language-specific ARS model for a single target language. For example, using English, German and Italian training data, we build a model specialized for German conversation. The goal of *multi-language adaptation* is to develop and evaluate a language-invariant ARS model for multiple target languages. For example, using English, German and Italian training data, we build a model that can respond to not only German but also Italian and English conversation. In the following subsections, we formalize each of these tasks.

4.3 Formal Task Definition

We assume that we have conversation data in each of a set of languages $\mathcal{K}$.

Training
In the training phase, a training dataset is given for each language $k \in \mathcal{K}$:

$$\mathcal{D}^{(k)}_{\text{train}} = \{\, \boldsymbol{x}^{(k)}_i, \boldsymbol{y}^{(k)}_i \,\}^{N^{(k)}}_{i=1}, \;\; k \in \mathcal{K}$$
$$\mathcal{D}_{\text{train}} = \bigcup_k \mathcal{D}^{(k)}_{\text{train}}$$

where $\boldsymbol{x}^{(k)}$ and $\boldsymbol{y}^{(k)}$ are a conversational situation and the target output in language k, respectively. We train a model $\mathcal{F} : \mathcal{X} \rightarrow \mathcal{Y}$ on these training samples.

Evaluation
In single-language adaptation, we evaluate a trained model for a single target language $t \in \mathcal{K}$. The trained model receives an input of the target language, $\boldsymbol{x}^{(t)} \sim \mathcal{D}^{(t)}_{\text{eval}}$, and predicts $\hat{\boldsymbol{y}}^{(t)}$. As evaluation metrics, we use the three accuracies (`ADR-RES`, `ADR` and `RES`) used in ARS (Sec. 3).

In multi-language adaptation, given evaluation datasets for all the languages $\mathcal{K}$, i.e., $\bigcup_k \mathcal{D}^{(k)}_{\text{eval}}$, the trained model receives an input of each language $\boldsymbol{x}^{(k)} \sim \mathcal{D}^{(k)}_{\text{eval}}$ and predicts $\hat{\boldsymbol{y}}^{(k)}$. As evaluation metrics, we use macro average over all the languages: $\texttt{ADR-RES} = \frac{\sum_k \texttt{ADR-RES}^{(k)}}{|\mathcal{K}|}$. `ADR` and `RES` are also computed in the same way.

5 Methods

In this section, we firstly describe a model used for addressee and response selection, and then explain our proposed methods to train parameters of the model.

Our model $\mathcal{F}$ consists of a feature extractor f^E, addressee scoring function f^A and response scoring function f^R. f^A and f^R return relevance scores (probabilities) for an addressee and response:

$$f^A(\boldsymbol{x}, a_i) = \sigma([\mathbf{a}_{\text{res}}, \mathbf{h}_c]^{\mathrm{T}}\, \mathbf{W}_a\, \mathbf{a}_i) \quad (1)$$
$$f^R(\boldsymbol{x}, \boldsymbol{r}_j) = \sigma([\mathbf{a}_{\text{res}}, \mathbf{h}_c]^{\mathrm{T}}\, \mathbf{W}_r\, \mathbf{r}_j) \quad (2)$$

where $\mathbf{a}_{\text{res}}$ is a responding agent vector, $\mathbf{h}_c$ is a conversational context vector, $\mathbf{a}_i$ is an agent vector, and $\mathbf{r}_j$ is a candidate response vector. All these vectors are encoded by the feature extractor f^E. We use the dynamic model (Ouchi and Tsuboi, 2016) as f^E. Fig. 2 shows the overview of the dynamic model. This model represents each agent as a hidden state vector that dynamically changes along with time steps in GRU (Cho et al., 2014). [4]

A model $\mathcal{F}$ is parameterized by $\theta = \{\theta_E \cup \{\mathbf{W}_a, \mathbf{W}_r\}\}$, where θ_E is parameters of f^E. To train these parameters, we present four methods. These methods assume that we have training sets for a set of languages $\mathcal{K}$: some of them are high-resource languages $\mathcal{S} \subseteq \mathcal{K}$ and others are relatively low-resource languages $\mathcal{T} = \bar{\mathcal{S}}$.

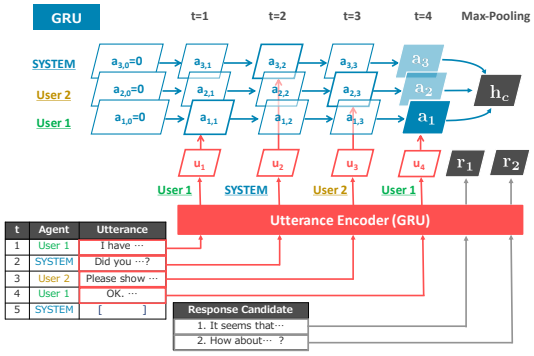


Figure 2: Overview of Dynamic Model.

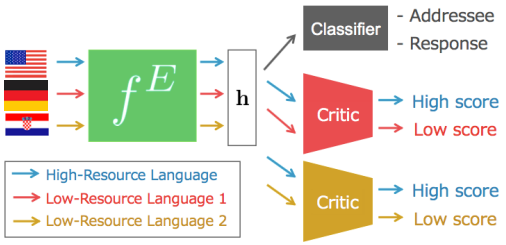


Figure 3: Overview of our W-GAN training method for multiple target languages.

5.1 A Basic Method

(a) Multilingual Embedding Replacement

This method trains a model $\mathcal{F}$ on high-resource language data $\mathcal{D}^{(s)}_{\text{train}}$, where $s \in \mathcal{S}$, and uses the trained model for responding conversations in other languages $\bar{\mathcal{S}}$. To realize this transfer, we use *multilingual embeddings*.

Consider the case where the high-resource language is English (En) and low-resource language is German (De). In the training phase, we use English word embeddings to train model parameters.[5] In the testing phase, instead of the English embeddings, we use German embeddings:

$$\textbf{Train: } \mathbf{w} = \mathbf{W}^{(\texttt{En})}_{\text{emb}}\, w \qquad \textbf{Test: } \mathbf{w} = \mathbf{W}^{(\texttt{De})}_{\text{emb}}\, w$$

where w is a one-hot vector. We just replace the English word embeddings $\mathbf{W}^{(\texttt{En})}_{\text{emb}}$ with the German ones $\mathbf{W}^{(\texttt{De})}_{\text{emb}}$. After looking up each word embedding $\mathbf{w}$, the neural model computes the hidden states. One advantage of this method is to require no target language data. As multilingual embeddings, we use MultiCCA[6] proposed by Ammar et al. (2016). In these embeddings, semantically similar words in the same language or translationally equivalent words in different languages are projected onto nearby.

Besides multilingual embeddings, another option to build a conversational model without no conversation data in a target language is to translate high-resource language data to low-resource one and train a conversational model on the translations. One limitation of this approach is that it is costly to prepare parallel corpora for building the translation model. We discuss this approach in Sec. 7.3.

5.2 Extended Methods

We present the two types of methods which use target language data for building (i) *language-specific models* and (ii) *language-invariant models*.

5.2.1 Methods for Language-Specific Models

(b) Fine-Tuning with Target Language Data

To compensate for the lack of the low-resource language data, this method firstly trains a model $\mathcal{F}_\theta$ on high-resource language data (pre-training phase). Then, using the pre-trained parameters θ as the initial values, this method re-trains them on low-resource language data (fine-tuning phase). We can expect that by gaining the better initial parameters, the tuning effectively adapts the model to the target language.

[4] In this example, a responding agent vector $\mathbf{a}_{\text{res}}$ is $\mathbf{a}_3$. Note that the states of the agents that are not speaking at the time are updated by zero vectors.

[5] The embeddings are fixed, not fine-tuned, during training.

[6] The pre-trained MultiCCA embeddings are provided at http://128.2.220.95/multilingual/data/

5.2.2 Methods for Language-Invariant Models

In order to build language-invariant models, it is critical to consider the two perspectives: (i) *avoiding catastrophic forgetting* and (ii) *learning language-invariant features*. Catastrophic forgetting (Kirkpatrick et al., 2017) is the phenomenon that a model forgets knowledge of previously trained tasks (languages) by incorporating knowledge of the current task (language). Language-invariant features are the features that are common and unchanged in different languages. Taking these two perspectives into account, we present the following two methods.

(c) Joint Loss Training

This method aims to avoid catastrophic forgetting by jointly training model parameters on all the language data at a time. Assuming that we have a set of languages $\mathcal{K}$, this method uses the joint loss function: $\mathcal{J}_{\text{joint}}(\theta) = \sum_k \mathcal{J}(\mathcal{D}^{(k)}, \theta)$ where the loss function $\mathcal{J}$ is the cross-entropy loss used in Ouchi and Tsuboi (2016).

(d) Multi-Language Adversarial Training

To learn language-invariant features, we use a framework of Wasserstein-GAN (W-GAN) (Arjovsky et al., 2017), a recently proposed technique to improve stability for generative adversarial nets (GANs) (Goodfellow et al., 2014). The aim of this method is to match the distributions of feature representations in two languages.

Fig. 3 illustrates an example. English is the high-resource language $s \in \mathcal{S}$, and German and Croatian are low-resource languages $t \in \mathcal{T}$. For each language, the feature extractor f^E receives an input conversation $\boldsymbol{x}$ and computes the hidden features $\mathbf{h} = f^E(\boldsymbol{x})$[7]. Thus, by using f^E, we obtain the hidden feature $\mathbf{h}^{(s)}$ and $\mathbf{h}^{(t)}$ for English and the others, respectively.

A pair of the high- and low-resource language features $\mathbf{h}^{(s)}$ and $\mathbf{h}^{(t)}$ is given to a critic g_π to minimize the Wasserstein distance between the distributions $p(\mathbf{h}^{(s)})$ and $p(\mathbf{h}^{(t)})$:

$$\mathcal{W}(\, p(\mathbf{h}^{(s)}),\, p(\mathbf{h}^{(t)})\,) = \max_{\pi}\, \mathbb{E}_{\mathbf{h}^{(s)}\sim p(\mathbf{h}^{(s)})}[g_\pi(\mathbf{h}^{(s)})] - \mathbb{E}_{\mathbf{h}^{(t)}\sim p(\mathbf{h}^{(t)})}[g_\pi(\mathbf{h}^{(t)})] \tag{3}$$

where the maximum is taken over the set of all 1-Lipschitz functions g_π.[8] By maximizing this equation, the distributions of the feature representations, $p(\mathbf{h}^{(s)})$ and $p(\mathbf{h}^{(t)})$, are made as close as possible. In this paper, as the critic g_π, we use multi-layer perceptron.

Eq. 3 is designed for the two distributions. Thus, we generalize this W-GAN equation to deal with $|\mathcal{S}|$ high-resource languages and $|\mathcal{T}|$ low-resource languages:

$$\mathcal{J}_{\text{wgan}}(\theta) = \sum_{s\in\mathcal{S}} \sum_{t\in\mathcal{T}} \mathcal{W}(\, p(\mathbf{h}^{(k)}),\, p(\mathbf{h}^{(\ell)})\,)$$

This loss function $\mathcal{J}_{\text{wgan}}$ is integrated with the joint loss: $\mathcal{J}_{\text{adv}}(\theta) = \mathcal{J}_{\text{joint}}(\theta) + \lambda\, \mathcal{J}_{\text{wgan}}(\theta)$ where λ is a hyper-parameter that balances these loss functions and we used $\lambda = 0.5$.

6 Corpus and Dataset

One of our goals is to provide a multilingual conversation corpus/dataset that can be used over a wide range of conversation research. We follow the corpus and data creation method of Ouchi and Tsuboi (2016). First, we crawl the Ubuntu IRC Logs[9], and preprocess the logs in many languages. Each language is identified by using a language detection library (Nakatani, 2010). The resulting corpus consists of multilingual conversations in 12 languages, shown in Tab. 1.

Then, we create an M-ARS dataset. For each language, we set the ground-truth/false addressees and responses following Ouchi and Tsuboi (2016). Note that the addressed usernames in utterances have been

[7] Hidden feature representation $\mathbf{h}$ is the concatenation of the responding speaker vector and context vector in Eqs. 1 and 2, i.e. $\mathbf{h} = [\mathbf{a}_{\text{res}}, \mathbf{h}_c]$.

[8] A function g is 1-Lipschitz when $|g(x) - g(y)| \leq |x - y|$ for all x and y. To constrain the critic g to a 1-Lipschitz function, the parameters of g are clipped to a fixed range.

[9] We use a collection of the logs during one year (2015). We plan to expand it by collecting the logs over all the years.

Language	Corpus Docs	Utters	Words
English (en)	7355	2.4 M	27.0 M
Italian (it)	357	165 k	1.1 M
Croatian (hr)	254	80 k	630 k
German (de)	248	38 k	335 k
Portuguese (pt)	211	52 k	285 k
Slovenian (sl)	179	59 k	357 k
Polish(pl)	67	8.8 k	51 k
Dutch (nl)	57	7.2 k	75 k
Spanish (es)	36	7.1 k	49 k
Swedish (sv)	26	1.7 k	6.8 k
Russian (ru)	5	0.3 k	1.5 k
French (fr)	3	0.5 k	3.0 k

Table 1: Statistics of M-ARS corpus.

Language	Dataset Train	Dev	Test
English (en)	665.6 k	45.1 k	51.9 k
Italian (it)	38,511	2,561	3,873
Croatian (hr)	11,387	512	1,145
German (de)	5,500	354	569
Portuguese (pt)	5,951	285	975

Table 2: Statistics of the M-ARS dataset.

Setting	Method	$\|\mathcal{R}\| = 2$			$\|\mathcal{R}\| = 10$		
		ADR-RES	ADR	RES	ADR-RES	ADR	RES
Single Language Adaptation	CHANCE	3.97	7.94	50.00	0.80	7.94	10.00
	TF-IDF	39.51	64.97	60.61	12.54	64.97	18.50
	TRGONLY	47.35	69.27	67.35	19.42	69.73	26.13
	ENONLY	38.07	65.72	57.65	8.50	62.38	13.75
	FINETUNE	49.58	69.59	69.84	21.15	70.33	28.15
	JOINT	51.55	70.30	71.88	22.32	**70.36**	29.38
	WGAN	**53.17**	**70.99**	**73.25**	**23.34**	70.20	**30.39**
Two Language Adaptation	CHANCE	2.30	4.59	50.00	0.46	4.59	10.00
	TF-IDF	38.32	60.29	64.25	13.99	60.29	23.84
	ENONLY	46.77	67.62	67.90	19.88	65.86	27.83
	FINETUNE	50.98	68.79	72.60	24.30	68.89	32.96
	JOINT	53.37	69.75	74.94	26.60	69.75	35.59
	WGAN	**54.14**	**70.07**	**75.63**	**27.23**	**69.76**	**36.11**
Five Language Adaptation	TF-IDF	39.04	63.10	62.06	13.12	63.10	20.64
	ENONLY	41.55	66.48	61.75	13.05	63.77	19.38
	JOINT	50.69	69.00	72.18	22.80	69.18	31.11
	WGAN	**52.11**	**69.74**	**73.34**	**23.39**	**69.35**	**31.88**

Table 3: Results for Single-/Two-/Five- language adaptation. Each number represents accuracy on addressee-response selection (ADR-RES), addressee selection (ADR) or response selection (RES).

removed for addressee selection. Thus, we have to predict the addressees without seeing the addressed usernames. The number of candidate responses ($|\mathcal{R}|$) is set to 2 or 10. The dataset is then randomly partitioned into a training set (90%), a development set (5%) and a test set (5%). Tab. 2 shows the statistics of the top 5 largest language sections of this resulting dataset.

7 Experiments

7.1 Experimental Setup

7.1.1 Task Settings

We use the following languages: (i) English (En) as the high-resource language, and (ii) Italian (It), Croatian (Hr), German (De), Portuguese (Pt) as the low-resource languages. In the following, we describe the languages used in each task.

(a) Single-Language Adaptation

Train: English + 1 Low-Res. Language, **Dev & Test:** 1 Low-Res. Language

For example, in the case that the target is Italian (It), we use the En and It training sets to train a model, and evaluate the trained model on the It test set. As evaluation metrics, we use the three types

of accuracies, ADR−RES, ARD and RES (described in Sec. 3). We report the macro average accuracies of all source-target language pairs (En-It, En-Hr, En-De, and En-Pt).

(b) Multi-Language Adaptation

Train: English + $|\mathcal{T}|$ Low-Res. Languages, **Dev & Test:** English + $|\mathcal{T}|$ Low-Res. Languages

We use the En, It, Hr, Pt and De training sets to train a unified model, and evaluate it on the test sets for all the languages (En, It, Hr, Pt, De). As evaluation metrics, we use the macro averages of ADR−RES, ARD and RES for all the languages. For example, for two language adaptation ($|\mathcal{T}| = 1$), we report the macro averages over all the language pairs (En-It, En-Hr, En-De, and En-Pt). For five language adaptation ($|\mathcal{T}| = 4$), we report the macro averages over all the five languages. Note that while we evaluate the performance on only the test set of the target low-resource language in single-language adaptation, we evaluate it on the test sets of English and the low-resource languages in multi-language adaptation.

7.1.2 Comparative Methods

We compare several methods. Our proposed methods (Sec. 5) are orthogonal, so that we can combine a method with others. In the following, we list the methods used in the comparison.

- TRGONLY: A dynamic model proposed by Ouchi and Tsuboi (2016) trained on only the low-resource target language data.
- ENONLY: A model built by (a) *multilingual embedding replacement* in Sec. 5.1: training a model on the English data and replacing the English word embeddings with the embeddings of the low-resource language.
- FINETUNE: A model built by (b) *fine-tuning* in Sec. 5.2: training a model on the high-resource language (English), and retraining it on the low-resource language.
- JOINT: A model built by (b) *fine-tuning* and (c) *joint loss training*: building a model by FINETUNE as an initial model, and retraining it with the joint loss functions.
- WGAN: A model built by (b) *fine-tuning* and (d) *multi-language adversarial training*: building a model by FINETUNE as an initial model, and retraining it with W-GAN.

Besides the neural models, we also use the TF-IDF model used in Ouchi and Tsuboi (2016). This model firstly creates TF-IDF vectors for the context and each candidate response. Then, it computes the cosine similarity for each pair of the context vector and a response vector. Finally, it selects the candidate response with the highest similarity.

7.1.3 Optimization

We use stochastic gradient descent (SGD) with a mini-batch method. To update parameters, we use *Adam* (Kingma and Ba, 2014). We describe the details of hyper-parameter settings in Supplementary Material.

7.2 Results

Tab. 3 shows the results of single-language and multi-language adaptation. Note that Tab. 6, Tab. 7, and Tab. 8 shows the detailed results for each language.

Single-Language Adaptation

WGAN achieved the best scores for most of the metrics. This suggests that the W-GAN method successfully transfers knowledge of high-resource language to a low-resource language. Also, FINETUNE outperformed TRGONLY. This means that pre-training parameters on the high-resource language data improves a model for a target low-resource language. Interestingly, ENONLY achieved higher scores than chance-level without any target language data. One possible explanation is that the multilingual embeddings have good alignments to some extent between similar meaning words in different languages.

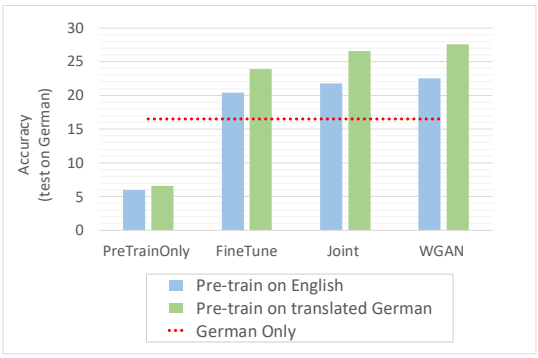


Figure 4: Effects of data augmentation with NMT.

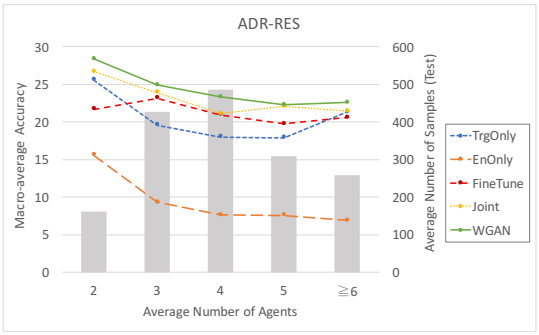


Figure 5: Effects of the number of agents in the context. Left axis: ADR-RES accuracy on test sets (drawn as lines). Right axis: the average number of test samples (drawn as bars).

Multi-Language Adaptation

In both two- and five-language adaptation, WGAN achieved the best scores. Specifically, In five language adaptation, regardless of using a single, unified model, JOINT and WGAN achieved high-performance.[10] Also, WGAN outperformed JOINT in all the metrics. This suggests that WGAN learns language-invariant features more effectively.

In NLP tasks, Chen et al. (2016) applied W-GAN to cross-lingual sentiment classification and successfully transferred the source-side knowledge to the target one. In this paper, we have extended the adaptation of single source-target pair to the adaptation of multiple pairs. Our experimental results show that our method works well for multi-language adaptation in conversation domain.

7.3 Data Augmentation with NMT

As we mentioned in Sec. 5.1 (a), as another approach to compensating for low-resource language, we use data augmentation. To increase the amount of the training set of a low-resource language, we translate high-resource language (English) samples to low-resource ones by using Neural Machine Translation (NMT). Although some translations are noisy, we can obtain much more training samples for a low-resource language. One limitation of this method is that it is costly to prepare parallel corpora, which is often unavailable for low-resource languages. For reproducibility, we use publicly available NMT models already trained on a parallel corpus. Since OpenNMT[11] provides an English-German model, we conduct experiments for the English-German pair.

We investigate the effects of translated German data for pre-training a model. Translating English (En) training utterances to German (De) ones by using the trained NMT model, we can obtain translated German training data (De′). We compare the two settings: (i) pre-training a model on English data and (ii) pre-training a model on translated German data. After pre-training, we apply the methods used in Sec. 7. We evaluate the performance with ADR-RES accuracy on the German test set.

Fig. 4 shows the results ($|\mathcal{R}| = 10$). The red dotted line is the performance of the modes trained on only original German training data (De). In each method, the blue bar at left hand is a model pre-trained on English (En), and the green bar at right hand is a model pre-trained on (De′).

PRETRAINONLY, the left-most method in Fig.4, uses the pre-trained models. The results were almost the same between models pre-trained on English or translated German data, and worse than the model trained on only the original German training data (red dotted line). This suggests that only the translations by NMT are not sufficient for building good multilingual ARS models.

Furthermore, we re-train the pre-trained models by using the three methods, FINETUNE, JOINT and WGAN. In other words, each method uses a pre-trained model as the initial model and re-trains the parameters. In all methods, the models re-trained from the De′ pre-trained model (green bars) were better

[10] Since FINETUNE builds a model for each target language, it cannot analyze multiple languages with a single model. That is why there is no result of FINETUNE in five-language adaptation.

[11] http://opennmt.net/Models/

than the ones from the En pre-trained ones (blue bars). This suggests that by combining the NMT-based data augmentation method with the knowledge-transfer methods, the performance is boosted. Another point is that WGAN consistently outperformed the other methods, which supports the utility of WGAN.

7.4 Analysis of Number of Agents

We investigate how accuracy fluctuates according to the number of agents in the context of length 15, as shown in Fig. 5. Overall, as the number of agents increases, the accuracies of all the methods tend to decline. Among them, WGAN achieved the best results in most of the cases. This suggests that WGAN can stably predicts addressees and responses in conversations with many participants.

8 Conclusion and Future Work

We have introduced *multilingual addressee and response selection* by providing (i) formal task definitions, (ii) several knowledge-transfer methods and (iii) a multilingual conversation corpus and dataset. Experimental results have demonstrated that our methods allow models to adapt multiple target languages. In particular, methods for language-invariant models can simultaneously deal with multiple languages with a single model.

Since our methods and dataset can apply to response generation, tackling the multilingual response generation tasks is an interesting line of our future work. In addition, our language-invariant systems can receive conversation in a language (e.g., English) and reply to it in another language (e.g., German). It can lead to interesting findings that our system is evaluated on code-mixing situations, where two or more languages are used in the same context.

References

Rieks Akker and David Traum. 2009. A comparison of addressee detection methods for multiparty conversations. In *Workshop on the Semantics and Pragmatics of Dialogue*.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively multilingual word embeddings. *CoRR*.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Dan Bohus and Eric Horvitz. 2011. Multiparty turn taking in situated dialog: Study, lessons, and directions. In *Proceedings of the SIGDIAL 2011 Conference*, SIGDIAL '11, pages 98–109, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification.

Kyunghyun Cho, Bart van Merrienboer, aglar Gülehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of NIPS*, pages 2672–2680.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv: 1408.6988*.

Natasa Jovanović and op den Rieks Akker. 2004. Towards automatic addressee identification in multi-party dialogues. In *Proceedings of SIGDIAL*.

Natasa Jovanović, op den Rieks Akker, and Anton Nijholt. 2006. Addressee identification in face-to-face meetings. In *Proceedings of EACL*.

Seokhwan Kim, Luis Fernando D'Haro, Rafael E Banchs, Jason D Williams, Matthew Henderson, and Koichiro Yoshino. 2016. The fifth dialog state tracking challenge. In *Proceeding of Spoken Language Technology Workshop (SLT)*, pages 511–517.

Diederik P. Kingma and Jimmy Lei Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv: 1412.6980*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, pages 3521–3526.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. In *Proceedings of ACL*.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of EMNLP*, pages 2122–2132.

Ryan Lowe, Nissan Pow, Iulian V. Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of SIGDIAL*, pages 285–294.

Ryan Lowe, Iulian Vlad Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. On the evaluation of dialogue systems with next utterance classification. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 264–269, Los Angeles, September. Association for Computational Linguistics.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of ACL*, pages 1116–1126.

Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In *Proceedings of NIPS*, pages 1367–1375.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2017. Coherent dialogue with attention-based language models. In *Proceedings of AAAI*.

Yukiko I. Nakano, Naoya Baba, Hung-Hsuan Huang, and Yuki Hayashi. 2013. Implementation and evaluation of a multimodal addressee identification mechanism for multiparty conversation systems. In *Proceedings of the 15th ACM on International Conference on Multimodal Interaction*, ICMI '13, pages 35–42, New York, NY, USA. ACM.

Shuyo Nakatani. 2010. Language detection library for java.

Hiroki Ouchi and Yuta Tsuboi. 2016. Addressee and response selection for multi-party conversation. In *Proceedings of EMNLP*, pages 2133–2143.

Kishore Papineni, Salim E. Roucos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Suman V Ravuri and Andreas Stolcke. 2014. Neural network models for lexical addressee detection. In *Proceedings of INTERSPEECH*, pages 298–302.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of EMNL*, pages 583–593.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of AAAI*, pages 3776–3783.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of ACL/IJCNLP*, pages 1577–1586.

David Traum. 2003. Issues in multiparty dialogues. *Advances in Agent communication*, pages 201–211.

David C Uthus and David W Aha. 2013. Multiparticipant chat analysis: A survey. *Artificial Intelligence*, pages 106–121.

Oriol Vinyals and V. Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv: 1506.05869*.

Hao Wang, Zhengdong Lu, Hang Li, and Enhong Chen. 2013. A dataset for research on short-text conversations. In *Proceedings of EMNLP*, pages 935–945.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2015. Syntax-based deep matching of short texts. In *Proceedings of IJCAI*, pages 1354–1361.

A Hyper-Parameters

Hyper-parameter	Values
Embedding size	512
GRU state size	256
WGAN λ	0.50
WGAN iterations	5
Critic hidden size	512
Critic activation function	ReLU
Batch size	32
Max epoch	30
Adam alpha	{0.001, 0.0005, 0.0001}
L2 weight decay	{0.001, 0.0005, 0.0001}

Table 4: Hyper-parameters for our experiments.

B Statics of Dataset

Tab. 5 shows the details of our dataset. "Docs" is documents, "Utters" is utterances, "W. / U." is the number of words per utterance, "A. / D." is the number of agents per document.

	Dataset (Train / Dev / Test)				
	English (`En`)	Italian (`It`)	Croatian (`Hr`)	German (`De`)	Portuguese (`Pt`)
No. of Docs	7355 (6,606/ 367/ 382)	357 (306/ 17/ 34)	254 (216/ 12/ 26)	248 (216/ 12/ 20)	211 (180/ 10/ 21)
No. of Utters	2.4 M (2.1 M / 13.2 k / 15.1 k)	165 k (144 k / 7 k / 14 k)	80 k (71 k / 3.4 k / 6.9 k)	38 k (33 k / 1.9 / 2.9 k)	52 k (44 k / 2.1 / 6.1 k)
No. of Words	27.0 M (23.8 M/ 1.5 M/ 1.7 M)	1.1 M (1.0 M/ 54 k/ 100 k)	630 k (553 k/ 25 k/ 52 k)	335 k (294 k/ 16 k/ 24 k)	285 k (243 k/ 11 k/ 30 k)
No. of Samples	- 665.6 k/ 45.1 k/ 51.9 k	- 38511 / 2561 / 3873	- 11387 / 512 / 1145	- 5500 / 354 / 569	- 5951 / 285 / 975
Avg. W. / U.	11.1 (11.1/ 11.2/ 11.3)	7.2 (6.9/ 7.7/ 7.1)	7.5 (7.7/ 7.3/ 7.5)	8.5 (8.9/ 8.4/ 8.2)	5.2 (5.5/ 5.2/ 4.9)
Avg. A. / D.	26.8 (26.3/ 30.68/ 32.1)	25.6 (24.9/ 26.2/ 25.6)	12.9 (12.7/ 13.5/ 12.7)	16.4 (17.4/ 15.9/ 15.8)	19.0 (19.7/ 18.6/ 18.8)

Table 5: Statistics of the multilingual dataset.

C Results for each language

Tab. 6 shows the detailed results of single-language adaptation. Tab. 7 shows the detailed results of two-language adaptation. Tab. 8 shows the detailed results of five-language adaptation.

Target	#Train	Method	$\|\mathcal{R}\| = 2$			$\|\mathcal{R}\| = 10$		
			ADR−RES	ADR	RES	ADR−RES	ADR	RES
It	38,511	CHANCE	2.99	5.97	50.00	0.60	5.97	10.00
		TF-IDF	43.89	67.49	64.58	16.63	67.49	23.42
		TRGONLY	63.28	79.86	78.36	32.87	80.92	38.73
		ENONLY	44.54	72.37	60.11	9.91	66.74	16.29
		FINETUNE	64.81	80.79	79.14	34.57	81.44	41.26
		JOINT	63.44	79.81	78.36	34.37	80.30	40.82
		WGAN	65.17	80.56	79.94	35.71	80.76	42.14
Hr	11,387	CHANCE	5.39	10.78	50.00	1.08	10.78	10.00
		TF-IDF	35.63	58.78	61.05	10.22	58.78	17.29
		TRGONLY	40.52	63.06	64.10	14.32	63.23	22.97
		ENONLY	34.06	60.87	54.24	7.95	60.52	12.31
		FINETUNE	40.00	62.62	63.23	14.67	64.72	22.79
		JOINT	44.37	62.97	69.26	15.98	62.97	24.54
		WGAN	45.07	62.62	70.48	16.59	63.76	25.68
De	5,500	CHANCE	4.09	8.17	50.00	0.82	8.17	10.00
		TF-IDF	36.38	64.67	55.36	10.19	64.67	14.41
		TRGONLY	43.94	67.49	64.15	16.52	66.96	22.50
		ENONLY	36.56	63.27	59.75	5.98	57.12	12.13
		FINETUNE	50.44	68.89	72.06	20.39	68.19	26.71
		JOINT	50.79	70.30	70.47	21.79	69.24	27.94
		WGAN	52.90	71.53	72.23	22.50	68.89	28.30
Pt	5,951	CHANCE	3.42	6.84	50.00	0.68	6.84	10.00
		TF-IDF	42.15	68.92	61.44	13.13	68.92	18.87
		TRGONLY	41.64	66.67	62.77	13.95	67.79	20.31
		ENONLY	37.13	66.36	56.51	10.15	65.13	14.26
		FINETUNE	43.08	66.05	64.92	14.97	66.97	21.85
		JOINT	47.59	68.10	69.44	17.13	68.92	24.21
		WGAN	49.54	69.23	70.36	18.56	67.38	25.44
Avg.	-	CHANCE	3.97	7.94	50.00	0.80	7.94	10.00
		TF-IDF	39.51	64.97	60.61	12.54	64.97	18.50
		TRGONLY	47.35	69.27	67.35	19.42	69.73	26.13
		ENONLY	38.07	65.72	57.65	8.50	62.38	13.75
		FINETUNE	49.58	69.59	69.84	21.15	70.33	28.15
		JOINT	51.55	70.30	71.88	22.32	**70.36**	29.38
		WGAN	**53.17**	**70.99**	**73.25**	**23.34**	70.20	**30.39**

Table 6: Results for single-language adaptation. Each number represents accuracy on addressee-response selection (ADR−RES), addressee selection (ADR) or response selection (RES). #Train is the number of training data.

Target	Method	$\|\mathcal{R}\| = 2$ ADR−RES	ADR	RES	$\|\mathcal{R}\| = 10$ ADR−RES	ADR	RES
En, It	CHANCE	1.80 (0.62, 2.95)	3.61	50.00	0.36 (0.12, 0.60)	3.61	10.00
	TF-IDF	40.51 (37.13, 43.89)	61.56	66.24	16.04 (15.44, 16.63)	61.56	26.31
	ENONLY	50.01 (55.47, 44.54)	70.95	69.13	20.59 (31.27, 9.91)	68.05	29.11
	FINETUNE	59.13 (53.44, 64.81)	74.71	77.78	31.07 (27.56, 34.57)	74.62	39.34
	JOINT	59.56 (55.67, 63.44)	74.63	78.50	33.04 (31.71, 34.37)	74.77	41.72
	WGAN	60.20 (55.23, 65.17)	74.94	79.10	33.38 (31.04, 35.71)	75.09	41.87
En, Hr	CHANCE	2.35 (0.62, 5.39)	4.71	50.00	0.47 (0.12, 1.08)	4.71	10.00
	TF-IDF	36.76 (37.13, 35.63)	60.15	61.63	12.82 (15.44, 10.22)	60.15	21.80
	ENONLY	44.77 (55.47, 34.06)	65.20	66.20	19.61 (31.27, 7.95)	64.94	27.12
	FINETUNE	46.03 (52.06, 40.00)	65.29	69.15	20.96 (27.24, 14.67)	66.01	30.28
	JOINT	49.49 (55.66, 43.32)	65.98	73.14	22.95 (31.74, 14.15)	66.15	32.23
	WGAN	49.80 (54.53, 45.07)	65.70	73.96	23.61 (30.62, 16.59)	66.38	33.52
En, De	CHANCE	3.01 (0.62, 4.08)	6.01	50.00	0.60 (0.12, 0.82)	6.01	10.00
	TF-IDF	36.38 (37.13, 36.38)	57.20	64.47	12.83 (15.44, 10.19)	57.20	23.24
	ENONLY	46.02 (55.47, 36.56)	66.40	68.95	18.63 (31.27, 5.98)	63.24	27.03
	FINETUNE	52.22 (53.99, 50.44)	68.79	74.52	24.66 (28.93, 20.39)	68.20	33.09
	JOINT	53.04 (55.28, 50.79)	69.73	74.30	26.09 (31.97, 20.21)	68.98	35.14
	WGAN	54.05 (55.20, 52.90)	70.35	75.19	27.05 (31.42, 22.67)	69.43	35.31
En, Pt	CHANCE	2.02 (0.62, 3.42)	4.04	50.00	0.40 (0.12, 0.68)	4.04	10.00
	TF-IDF	39.64 (37.13, 42.15)	62.27	64.67	14.29 (15.44, 13.13)	62.27	24.03
	ENONLY	46.30 (55.47, 37.13)	67.94	67.33	20.71 (31.27, 10.15)	67.24	28.09
	FINETUNE	46.53 (49.98, 43.08)	66.39	68.97	20.52 (26.06, 14.97)	66.74	29.13
	JOINT	51.39 (55.18, 47.59)	68.66	73.81	24.32 (31.51, 17.13)	69.11	33.28
	WGAN	52.49 (55.44, 49.54)	69.31	74.29	24.88 (31.19, 18.56)	68.16	33.75
Avg.	CHANCE	2.30	4.59	50.00	0.46	4.59	10.00
	TF-IDF	38.32	60.29	64.25	13.99	60.29	23.84
	ENONLY	46.77	67.62	67.90	19.88	65.86	27.83
	FINETUNE	50.98	68.79	72.60	24.30	68.89	32.96
	JOINT	53.37	69.75	74.94	26.60	69.75	35.59
	WGAN	**54.14**	**70.07**	**75.63**	**27.23**	**69.76**	**36.11**

Table 7: Results for two-language adaptation. Each number is macro average of the accuracies over all the languages. Each of the parenthesized numbers in the ADR−RES column is ADR−RES accuracy for each language.

Target	Method	$\|\mathcal{R}\| = 2$ ADR−RES	ADR	RES	$\|\mathcal{R}\| = 10$ ADR−RES	ADR	RES
En, It	TF-IDF	39.04 (37.13, 43.89, 35.63, 36.38, 42.15)	63.10	62.06	13.12 (15.44, 16.63, 10.22, 10.19, 13.13)	63.10	20.64
	ENONLY	41.55 (55.47, 44.54, 34.06, 36.56, 37.13)	66.48	61.75	13.05 (31.27, 9.91, 7.95, 5.98, 10.15)	63.77	19.38
Hr, De, Pt	JOINT	50.69 (54.49, 61.71, 43.41, 47.80, 46.05)	69.00	72.18	22.80 (31.26, 30.29, 14.59, 20.21, 17.64)	69.18	31.11
	WGAN	**52.11** (53.88, 63.18, 44.19, 52.02, 47.28)	**69.74**	**73.34**	**23.39** (30.6, 31.29, 14.67, 22.32, 18.05)	**69.35**	**31.88**

Table 8: Results for five-language adaptation. Each number is macro average of the accuracies over all the languages. Each of the parenthesized numbers in the ADR−RES column is ADR−RES accuracy for each language.

Graph-Based Decoding for Event Coreference and Sequencing Resolution

Zhengzhong Liu and **Teruko Mitamura** and **Eduard Hovy**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{liu, teruko, hovy}@cs.cmu.edu

Abstract

Events in text documents are interrelated in complex ways. In this paper, we study two types of relation: **Event Coreference** and **Event Sequencing**. We show that the popular tree-like decoding structure for automated Event Coreference is not suitable for Event Sequencing. To this end, we propose a graph-based decoding algorithm that is applicable to both tasks. The new decoding algorithm supports flexible feature sets for both tasks. Empirically, our event coreference system has achieved state-of-the-art performance on the TAC-KBP 2015 event coreference task and our event sequencing system beats a strong temporal-based, oracle-informed baseline. We discuss the challenges of studying these event relations.

Title and Abstract in Chinese

基于图的事件共指消解以及依序关系解码算法

文本中提及的事件之间常存在着复杂的关系。在这篇文章中，我们主要研究两种关系：事件间的共指关系以及依序关系。我们指出，常被应用在事件指代消歧上的传统的树结构解码方式并不适用于解决事件依序问题。为了解决这个问题，我们提出了一种适用于两种情况的新的图结构解码算法。这个算法让我们可以为这两个任务设计灵活的特征。在实验上，我们的事件共指消解系统在TAC-KBP 2015的共指消解任务上达到了目前最佳的水平。同时，我们的依序关系系统的结果超过了一个基于时间分析并有部分正确答案的基线系统。最后我们分析了结果，并讨论了事件关系判别任务的一些挑战。

1 Introduction

Events are important building blocks of documents. They play a key role in document understanding tasks, such as information extraction (Chambers and Jurafsky, 2011), news summarization (Vossen and Caselli, 2015), story understanding (Mostafazadeh et al., 2016). Conceptually, **events** correspond to state changes and normally include a location, a time interval, and several entities/participants. In a text, events are realized as text spans, normally as verbs and nouns that indicate state changes (Vendler, 1957). The text spans are often referred to as **event mentions** or **event nuggets** (We use the term event mention in this paper.). The textual mentions of events have rich relations among them, and collectively convey the meaning of one or more related documents. In this paper, we study two different types of relation: **Event Hopper Coreference (EH)** and **Event Sequencing (ES)**.

Event Coreference: There is a rich literature on the Event Coreference problem (Liu et al., 2014; Cybulska and Vossen, 2014; Lu et al., 2016; Peng et al., 2016; Lu and Ng, 2017; Araki, 2018). By analogy to entity coreference, the "same" conceptual event may be realized by multiple text spans (event mentions). The coreference problem aims at identifying these relations to recover events from the text spans. The **Event Hopper** Coreference task in the TAC-KBP evaluation campaign defines coreference links as follows (Mitamura et al., 2018): Two event mentions are considered coreferent if they refer to the conceptually same underlying event, even if their arguments are not strictly identical. For example,

Proceedings of the 27th International Conference on Computational Linguistics, pages 3645–3657
Santa Fe, New Mexico, USA, August 20-26, 2018.

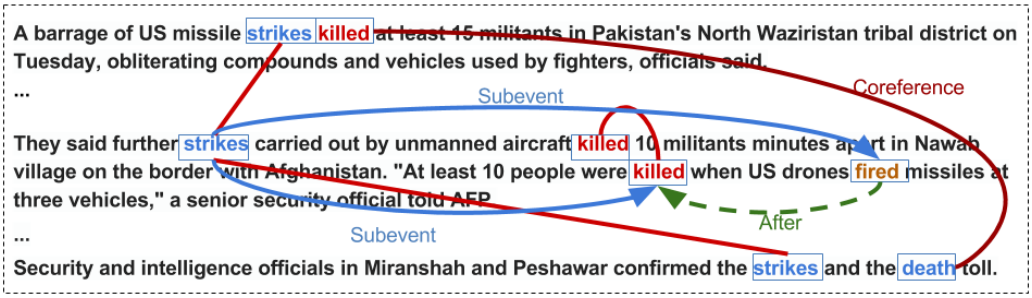


Figure 1: Example of Event Coreference and Sequence relations. Red lines are coreference links; solid blue arrows represent Subevent relations; dotted green arrows represent After relations.

mentions that share similar temporal and location scope, though not necessarily the same expression, are considered to be coreferent (*Attack in Baghdad on Thursday* vs. *Bombing in the Green Zone last week*). This means that the event arguments of coreferential events mentions can be non-coreferential (18 killed vs. dozens killed), as long as they refer to the same event, judging from the available evidence.

Event Sequencing: The coreference relations build up events from scattered mentions. On the basis of events, various other types of relations can then be established between them. The Event Sequencing task studies one such relation. The task is motivated by Schank's *scripts* (Schank and Abelson, 1977), which suggests that human organize information through procedural data structures, reassembling sequences of events. For example, the list of verbs *order, eat, pay, leave* may trigger the restaurant script. A human can conduct reasoning with a typical ordering of these events based on common sense (e.g., *order* should be the first event, *leave* should be the last event).

The ES task studies how to group and order events from text documents belonging to the same script. Figure 1 shows some annotation examples. Conceptually, event sequencing relations hold between the events, while coreference relations hold between textual event mentions. Given a document, the ES task requires systems to identify events within the same script and classify their inter-relations. These relations can be represented as labeled Directed Acyclic Graphs (DAGs). There are two types of relations[1]: **After** relations connect events following script orders (e.g. *order* followed by *eating*); **Subevent** relations connect events to a larger event that contains them. In this paper, we focus only on the **After** relations.

Since script-based understanding is built in the ES task, it has some unique properties comparing to pure temporal ordering: 1) event sequences from different scripts provide separate logical divisions of text, while temporal ordering considers all events to lie on a single timeline; 2) temporal relations for events occurring at similar time points may be complicated. Script-based relations may alleviate the problem. For example, if a `bombing` `kills` some people, the temporal relation of the `bombing` and `kill` may be "inclusion" or "after". This is considered an **After** relation in ES because `bombing` causes the `kill`ing.

For structure prediction, decoding — recovering the complex structure from local decisions — is one of the core problems. The most successful decoding algorithm for coreference nowadays is mention ranking based (Björkelund and Kuhn, 2014; Durrett and Klein, 2014; Lee et al., 2017). These models rank the antecedents (mentions that appear earlier in discourse) and recover the full coreference clusters from local decisions. However, unlike coreference relations, sequencing relations are directed. Coreference decoding algorithms cannot be directly applied to such relations (§3.1). To solve this problem, we propose a unified graph-based framework that tackles both event coreference and event sequencing. Our method achieves state-of-the-art results on the event coreference task (§4.4) and beats an informed baseline on the event sequencing task (§4.5). Finally, we analyze the results and discuss the difficult challenges for both tasks (§5). Detailed definitions of these tasks can be found in the corresponding task documents[2].

2 Related Work

Many researchers have worked on event coreference tasks since Humphreys et al. (1997). Recent advances in event coreference have been promoted by the availability of annotated corpora. However, due to

[1]Detailed definition of relations can be found in http://cairo.lti.cs.cmu.edu/kbp/2016/after/

[2]`http://cairo.lti.cs.cmu.edu/kbp/2017/event/documents`

the complex nature of events, approaches to event coreference adopt quite different assumptions and definitions. Most of event coreference researches are conducted on the popular ACE corpus (Chen and Ji, 2009; Chen et al., 2009; Sangeetha and Arock, 2012; Chen and Ng, 2013; Chen and Ng, 2015). Unlike the TAC KBP setting, the definition of event coreference in the ACE corpus requires strict argument matching. Work on the Intelligence Community (IC) Corpus (Hovy et al., 2013; Cybulska and Vossen, 2012; Liu et al., 2014; Araki et al., 2014) considers event relations on a restricted domain (i.e., terrorist events). Works on the ECB corpus (Lee et al., 2012; Cybulska and Vossen, 2014) focuses on both within-document and cross-document coreference.

Our work follows the line of work promoted by the TAC-KBP event nugget tasks (Mitamura et al., 2015). There is a small but growing amount of work on conducting event coreference on the TAC-KBP datasets (Lu et al., 2016; Peng et al., 2016; Lu and Ng, 2017). The TAC dataset uses a relaxed coreference definition comparing to other corpora, requiring two event mentions to intuitively refer to the same real-world event despite differences of their participants.

For event sequencing, there are few supervised methods on script-like relation classification due to the lack of data. To the best of our knowledge, the only work in this direction is by Araki et al. (2014). This work focuses on the other type of relations in the event sequencing task: **Subevent** relations. There is also a rich literature on unsupervised script induction (Chambers and Jurafsky, 2008; Cheung et al., 2013; Rudinger et al., 2015; Pichotta and Mooney, 2016; Ferraro and Durme, 2016) that extracts scripts as a type of common-sense knowledge from raw documents. The focus of this work is to make use of massive collections of text documents to mine event co-occurrence patterns. In contrast, our work focuses on parsing the detailed relations between event mentions in each document.

Another line of work closely related to event sequencing is to detect other temporal relations between events. Recent computational approaches for temporal detection are mainly conducted on the TimeBank corpus (Pustejovsky et al., 2002). There have been several studies on building automatic temporal reasoning systems (Uzzaman and Allen, 2010; Do et al., 2012; Chambers et al., 2014). In comparison, the Event Sequencing task is motivated by the Script theory, which places more emphasis on common-sense knowledge about event chronology.

3 Model

3.1 Graph-Based Decoding Model

In the Latent Antecedent Tree (LAT) model popularly used for entity coreference decoding (Fernandes et al., 2012; Björkelund and Kuhn, 2014), each node represents an event mention and each arc a coreference relation, and new mentions are connected to some past mention considered most similar. Thus the LAT model represents the decoding structure as a tree. This can represent any coreference cluster, because coreference relations are by definition equivalence relations[3].

In contrast, tree structures cannot always fully cover an Event Sequence relation graph, because 1) the After links are directed, not symmetric, and 2) multiple event nodes can link to one node, resulting in multiple parents.

To solve this problem, we extend the LAT model and propose its graph version, namely the Latent Antecedent Graph (LAG) model. Figure 2 contrast LAT and LAG with decoding examples. The left box shows two example decoded trees in LAT, where each node has one single parent. The right box shows two example decoded trees in LAG, where each node can be linked to multiple parents.

Formally, we define the series of (pre-extracted) event mentions of the document as $M = \{m_0, m_1, ..., m_n\}$, following their discourse order. m_0 is an artificial root node preceding all mentions. For each mention m_j, let A_j be the set of its potential antecedents: $A_j = \{m_0, m_1, ..., m_{j-1}\}$. Let $\mathcal{A}$ denotes the set of antecedents for all the mentions in the sequence $\{A_0, A_1, ..., A_n\}$. The two tasks in question can be considered as finding the appropriate antecedent(s) from $\mathcal{A}$. Similarly, we define the gold antecedent set $\tilde{\mathcal{A}} = \{\tilde{A}_0, \tilde{A}_1, ..., \tilde{A}_n\}$, where $\tilde{A}_i$ represent the set of antecedents of m_i allowed by the gold standard. In the coreference task, $\tilde{A}_i$ contains all antecedents that are coreferent with m_i. In the sequencing task, $\tilde{A}_i$ contains all antecedents that have an $After$ relation to m_i.

[3] An equivalence relation is reflexive, symmetric and transitive.

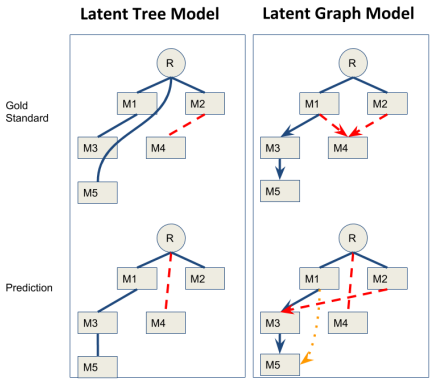


Figure 2: Latent Tree Model (left): tree structure formed by undirected links. Latent Graph Model (right): a DAG form by directed links. Dashed red links highlight the discrepancy between prediction and gold standard. The dotted yellow link (bottom right) can be inferred from other links.

We can now describe the decoding process. We represent each arc as $\langle m_i, m_j, r \rangle (i < j)$, where r is the relation name. The relation direction can be specified in the relation name r (e.g. r can be *after.forward* or *after.backward*). Further, an arc from the root node m_0 to node m_j represents that m_j does not have any antecedent. The score of the arc is the dot product between the weight parameter $\vec{w}$ and a feature vector $\Phi(\langle m_i, m_j, r \rangle)$, where Φ is an arc-wise feature function. The decoded graph z can be determined by a set of binary variables $\vec{z}$, where $\vec{z}_{ijr} = 1$ if there is an arc $\langle m_i, m_j, r \rangle$ or 0 otherwise. The final score of z is the sum of scores of all arcs:

$$score(z) = \sum_{i,j,r} \vec{z}_{ijr} \vec{w} \cdot \Phi(\langle m_i, m_j, r \rangle) \quad (1)$$

The decoding step is to find the output $\hat{z}$ that maximizes the scoring function:

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

where $\mathcal{Z}(\mathcal{A})$ denotes all possible decoding structures given the antecedent sets $\mathcal{A}$. It is useful to note that the decoding step can be applied in the same way to the gold antecedent set $\tilde{\mathcal{A}}$.

Algorithm 1 shows the Passive-Aggressive training algorithm (Crammer et al., 2006) used in our decoding framework. Line 8 decodes the maximum scored structure from all possible gold standard structures using the current parameters $\vec{w}$. Intuitively, this step tries to find **the "easiest" correct graph** — the correct graph with the highest score — for the current model. Several important components remain unspecified in algorithm 1: (1) the decoding step (line 6, 8); (2) the match criteria: whether to consider the system decoding structure as correct (line 7); (3) feature delta: computation of feature difference (line 9); (4) loss computation (line 10). We detail the actual implementation of these steps in §3.1.2.

3.1.1 Minimum Decoding Structure

Similar to the LAT model, there may be many decoding structures representing the same configuration. In LAT, since there is exactly one link per node, the number of links in different decoding structures is the same, hence comparable. In LAG, however, one node is allowed to link to multiple antecedents, creating a potential problem for decoding. For example, consider the sequence $m_1 \xrightarrow{\text{after}} m_2 \xrightarrow{\text{after}} m_3$, both of the following structures are correct:

1. $\langle m_1, m_2, after \rangle$, $\langle m_2, m_3, after \rangle$
2. $\langle m_1, m_2, after \rangle$, $\langle m_2, m_3, after \rangle$, $\langle m_1, m_3, after \rangle$

However, the last relation in the second decoding structure can actually be inferred via transitivity. We do not intend to spend the modeling power on such cases. We empirically avoid such redundant cases by using the **transitive reduction graph** for each structure. For a directed acyclic graph, a transitive

Algorithm 1: PA algorithm for training

1 **Input:** Training data D, number of iterations T
2 **Output:** Weight vector $\vec{w}$
3 $\vec{w} = \vec{0}$;
4 $\langle \mathcal{A}, \tilde{\mathcal{A}} \rangle \in D$;
5 **for** $t \leftarrow 1..T$ **do**
6 $\quad \hat{z} = \arg\max_{\mathcal{Z}(\mathcal{A})} score(z)$;
7 $\quad$ **if** $\neg Match(\hat{z}, \tilde{\mathcal{A}})$ **then**
8 $\quad\quad \tilde{z} = \arg\max_{\mathcal{Z}(\tilde{\mathcal{A}})} score(z)$;
9 $\quad\quad \Delta = FeatureDelta(\tilde{z}, \hat{z})$;
10 $\quad\quad \tau = \frac{loss(\tilde{z}, \hat{z})}{||\Delta||^2}$;
11 $\quad\quad w = w + \tau\Delta$;
12 **return** w;

reduction graph contains the fewest possible edges that have the same reachability relation as the original graph. In the example above, structure 1 is a transitive reduction graph for structure 2. We call the decoding structures that corresponding to the reduction graphs as *minimum decoding structures*. For LAG, we further restrict $\mathcal{Z}(\mathcal{A})$ to contain only minimum decoding structures.

3.1.2 Training Details in Latent Antecedent Graph

In this section, we describe the decoding details for LAG. Note that if we enforce a single antecedent for each node (as in our coreference model), it falls back to the LAT model (Björkelund and Kuhn, 2014).

Decoding: We use a greedy **best-first decoder** (Ng and Cardie, 2002), which makes a left-to-right pass over the mentions. The decoding step is the same for line 6 and 8. The only difference is that we will use gold antecedent set ($\tilde{\mathcal{A}}$) at line 8. For each node m_j, we keep all links that score higher than the root link $\langle 0, m_j, r \rangle$.

Cycle and Structure Check: Incremental decoding a DAG may introduce cycles to the graph, or violate the minimum decoding structure criterion. To solve this, we maintain a set $R(m_i)$ that is reachable from m_i during the decoding process. We reject a new link ($\langle m_j, m_i \rangle$ if $m_j \in R(m_i)$) to avoid cycles. We also reject a redundant link ($\langle m_i, m_j \rangle$ if $m_j \in R(m_i)$) to keep a minimum decoding structure. Our current implementation is greedy, we leave investigations of search or global inference based algorithms to future work.

Selecting the Latent Event Mention Graph: Note that sequence relations are on the event level. Given a unique event graph, it may still correspond to multiple mention graphs. In our implementation, we use a minimum set of event mentions to represent the full event graph by taking one single mention from each event. Following the "easiest" intuition, we select the single mention that will result in the highest score given the current feature weight w.

Match Criteria: We consider two graphs to match when their inferred graphs are the same. The inferred graph is defined by taking the transitive closure of the graph and propagate the links through the coreference relations. For example, in Figure 1, the mention `fired` will be linked to two `killed` mentions after propagation.

Feature Delta: In structural perceptron training (Collins, 2002), the weights are updated directly by the feature delta. For all the features $\tilde{f}$ of the gold standard graph $\tilde{z}$ and features $\hat{f}$ of a decoded graph $\hat{z}$, the feature delta is simply: $\Delta = \tilde{f} - \hat{f}$. However, a decoded graph may contain links that are not directly presented but inferable from the gold standard graph. For example, in Figure 2, the prediction graph has a link from $M5$ to $M1$ (the orange arc), which is absent but inferable from the gold standard tree. If we keep these links when computing Δ, the model does not converge well. We thus remove the features on the inferable links from $\hat{f}$ when computing Δ.

Loss: We define the loss to be the number of different edges in two graphs. Following Björkelund and Kuhn (2014), we further penalize erroneous root attachment: an incorrect link to the root m_0 adds the loss

Head	Headword token and lemma pair, and whether they are the same.
Type	The pair of event types, and whether they are the same.
Realis	The pair of realis types and whether they are the same.
POS	POS pair of the two mentions and whether they are the same.
Exact Match	Whether the 5-word windows of the two mentions matches exactly.
Distance	Sentence distance between the two mentions.
Frame	Frame name pair of the two mentions and whether they are the same.
Syntactic	Whether a mention is the syntactic ancestor of another.

Table 1: Coreference Features. Parsing is done using Stanford CoreNLP (Manning et al., 2014); frame names are produced by Semafor (Das and Smith, 2011).

by 2. For example, in Figure 2 the prediction graph (bottom right) incorrectly links m_4 to Root and misses a link to m_3, which cause a total loss of 3. In addition, to be consistent with the feature delta computation, we do not compute loss for predicted links that are inferable from the gold standard.

3.2 Features

3.2.1 Event Coreference Features

For event coreference, we design a simple feature set to capture syntactic and semantic similarity of arcs. The main features are summarized in Table 1. In the TAC KBP 2015 coreference task setting, the event mentions are annotated with two attributes. There are 38 event types and subtype pairs (e.g., *Busness.Merge-Org, Conflict.Attack*). There also 3 realis type: events that actually occurred are marked as *Actual*; events that are not specific are marked as *Generic*; other events such as future events are marked as *Other*. For these two attributes, we use the gold annotations in our feature sets.

3.2.2 Event Sequencing Features

An event sequencing system needs to determine whether the events are in the same script and order them. We design separate feature sets to capture these aspects: the Script Compatibility set considers whether mentions should belong to the same script; the Event Ordering set determines the relative ordering of the mentions. Our final features are the cross products of features from the following 3 sets.

1. **Surface-Based Script Compatibility**: these features capture whether two mentions are script compatible based on the surface information, including:
   - Mention headword pair.
   - Event type pair.
   - Whether two event mentions appear in the same cluster in Chambers's event schema database (Chambers and Jurafsky, 2010).
   - Whether the two event mentions share arguments, and the semantic frame name of the shared argument (produced by the Semafor parser (Das and Smith, 2011)).
2. **Discourse-Based Script Compatibility**: these features capture whether two event mentions are related given the discourse context.
   - Dependency path between the two mentions.
   - Function words (words other than Noun, Verb, Adjective and Adverb) in between the two mentions.
   - The types of other event mentions between the two mentions.
   - The sentence distance of two event mentions.
   - Whether there are temporal expressions (AGM-TMP slot from a semantic parser (Tratz and Hovy, 2011)) in the sentences of the two mentions.
3. **Event Ordering**: this feature set tries to capture the ordering of events. We use the discourse ordering of two mentions (forward: the antecedent is the parent; backward: the antecedent is the child), and temporal ordering produced by Caevo (Chambers et al., 2014).

Taking the *after* arc from `fired` to `killed` in Figure 1 as an example, a feature after the cross product is: Event type pair is *Conflict.Attack* and *Life.Die*, discourse ordering is *backward*, and sentence distance is 0.

4 Experiments

4.1 Dataset

We conduct experiments on the dataset released in Text Analysis Coreference (TAC-KBP) 2017 Event Sequencing task (released by LDC under the catalog name LDC2016E130). This dataset contains rich event relation annotations, with event mentions and coreference annotated in TAC-KBP 2015, and additional annotations on Event Sequencing[4]. There are 158 documents in the training set and 202 in the test set, selected from general news articles and forum discussion threads. The event mentions are annotated with 38 type-subtype and 3 realis status (Actual, Generic, Other). Event Hopper, After, and Subevent links are annotated between event mentions. For all experiments, we develop our system and conduct ablation studies using 5-fold cross-validation on the training set, and report performance on the test set.

4.2 Baselines and Benchmarks

Coreference: we compare our event coreference system against the top performing systems from TAC-KBP 2015 (LCC, UI-CCG, and LTI). In addition, we also compare the results against two official baselines (Mitamura et al., 2015): the Singleton baseline that put each event mention in its own cluster and the Match baseline that creates clusters based on mention type and realis status match.
Sequencing: This work is an initial attempt to this problem, so there is currently no comparable prior work on the same task. We instead compare with a baseline using event temporal ordering systems. We use a state-of-the-art temporal system named Caevo (Chambers et al., 2014). To make a fair comparison, we feed the gold standard event mentions to the system along with mentions predicted by Caevo[5]. However, since the script-style After links are only connected between mentions in the same script, directly using the output of Caevo produces very low precision. Instead, we run a stronger baseline: we take the gold standard script clusters and then only ask Caevo to predict links within these clusters (Oracle Cluster + Temporal).

4.3 Evaluation Metrics

Evaluating Event Coreference: We evaluate our results using the official scorer provided by TAC-KBP, which uses 4 coreference metrics: *BLANC* (Recasens and Hovy, 2011), *MUC* (Chinchor, 1992), B^3 (Bagga and Baldwin, 1998) and *CEAF-E* (Luo, 2005). Following the TAC KBP task, systems are ranked using the average of these 4 metrics.
Evaluating Event Sequencing: The TAC KBP scorer evaluates event sequencing using the metric of the TempEval task (UzZaman, 2012; UzZaman et al., 2013). The TempEval metric calculates special precision and recall values based on the closure and reduction graphs:

$$Precision = \frac{|Response^- \cap Reference^+|}{|Response^-|} \quad Recall = \frac{|Reference^- \cap Response^+|}{|Reference^-|}$$

where $Response$ represents the After link graph from the system response and $Reference$ represents the After link graph from the gold standard. G^+ represents the graph closure for graph G and G^- represents the graph reduction for graph G. As preprocessing, relations are automatically propagated through coreference clusters (currently using gold standard clusters). The final score is the standard F-score: geometric mean of the precision and recall values.

[4]http://cairo.lti.cs.cmu.edu/kbp/2016/after/
[5]We keep the mentions predicted by Caevo because its inference may be affected by these mentions.

	B^3	CEAF-E	MUC	BLANC	AVG.
`Singleton`	78.10	68.98	0.00	48.88	52.01
`Matching`	78.40	65.82	**69.83**	76.29	71.94
LCC	82.85	74.66	68.50	**77.61**	75.69
UI-CCG	83.75	75.81	63.78	73.99	74.28
LTI	82.27	75.15	60.93	71.57	72.60
This work	**85.59**	**79.65**	67.81	77.37	**77.61**

Table 2: Test Results for Event Coreference with the `Singleton` and `Matching` baselines.

	B^3	CEAF-E	MUC	BLANC	AVG.
ALL	81.97	74.80	76.33	76.07	77.29
-Distance	81.92	74.48	76.02	77.55	77.50
-Frame	82.14	75.01	76.28	77.74	77.79
-Syntactic	81.87	74.89	75.79	76.22	77.19

Table 3: Ablation study for Event Coreference.

4.4 Evaluation Results for Event Coreference

The test performance on Event Coreference is summarized in Table 2. Comparing to the top 3 coreference systems in TAC-KBP 2015, we outperform the best system by about 2 points absolute F-score on average. Our system is also competitive on individual metrics. Our model performs the best based on B^3 and CEAF-E, and is comparable to the top performing systems on MUC and BLANC.

Note that while the `Matching` baseline only links event mentions based on event type and realis status, it is very competitive and performs close to the top systems. This is not surprising since these two attributes are based on the gold standard. To take a closer look, we conduct an ablation study by removing the simple match features one by one. The results are summarized in Table 3. We observe that some features produce mixed results on different metrics: they provide improvements on some metrics but not all. This is partially caused by the different characteristics of different metrics. On the other hand, these features (parsing and frames) are automatically predicted, which make them less stable. Furthermore, the Frame features contain duplicate information to event types, which makes it less useful in this setting.

Besides the presented features, we have also designed features using event argument. However, we do not report the results since the argument features decrease the performance on all metrics.

4.5 Evaluation Results for Event Sequencing

The evaluation results on Event Sequencing is summarized in Table 4. Because the baseline system has access to the oracle script clusters, it produces high precision. However, the low recall value shows that it fails to produce enough After links. Our analysis shows that a lot of After relations are not indicated by clear temporal clues, but can only be solved with script knowledge. In Example 3, the baseline system is able to identify "fled" is after "ousted" from explicit marker "after". However, it fails to identify that "extradited" is after "arrested", which requires knowledge about prototypical event sequences.

(3) Eight months after the [transport fled] Ivory Coast when Gbagbo, the former president, was [End.Position ousted] by the French military. Blé Goudé was subsequently [Jail arrested] in Ghana and [transport extradited] Megrahi,[Jail jailed] for [Attack killing] 270 people in 1988. [6]

In our error analysis, we noticed that our system produces a large number of relations due to coreference propagation. One single wrong prediction can cause the error to propagate.

Besides memorizing the mention pairs, our model also tries to capture script compatibility through discourse signals. To further understand how much these signals help, we conduct an ablation study of the features in the discoursed based compatibility features (see §3.2.2). Similarly, we remove each feature group from the full feature set one by one and observe the performance change.

[6]The small red text indicates the event type for each mention.

	Prec.	Recall	F-Score
Oracle Cluster+Temporal	**46.21**	8.72	14.68
Our Model	18.28	**16.91**	**17.57**

Table 4: Test Results for event sequencing. The Oracle Cluster+Temporal system is using Caevo's result on the Oracle Clusters.

	Prec.	Recall	F-Score	Δ
Full	37.92	36.79	36.36	
- Mention Type	32.78	29.81	30.07	6.29
- Sentence	33.90	30.75	31.00	5.36
- Temporal	37.21	36.53	35.81	0.55
- Dependency	38.18	36.44	36.23	0.13
- Function words	38.08	36.51	36.18	0.18

Table 5: Ablation Study for Event Sequencing.

The results are reported in Table 5. While most of the features only affect the performance by less than 1 absolute F1 score, the feature sets after removing *mention* or *sentences* show a significant drop in both precision and recall. This shows that discourse proximity is the most significant ones among these features. In addition, the *mention* feature set captures the following *explain away* intuition: the event mentions A and B are less likely to be related if there are similar mentions in between. One such example can be seen in Figure 1, the event mention `fired` is more likely to relate to the closest `killed`, instead of the other `killed` in the first paragraph.

In addition, our performance on the development set is higher than the test set. Further analysis reveals two causes: 1) the coreference propagation step causes the scores to be very unstable, 2) our model only learns limited common sense ordering based on lexical pairs, which overfit to the small training corpus. Since the annotation is difficult to scale, it is important to use methods to harvest script common sense knowledge automatically, as in the script induction work (Chambers and Jurafsky, 2008).

5 Discussion

5.1 Event Coreference Challenges

Although we have achieved good performance on event coreference, upon closer investigation we found that most of the coreference decisions are still made based on simple word/lemma matching (note that the type and realis baseline is as high as 0.72 F1 score). The system exploits little semantic information to resolve difficult event coreference problems. A major challenge is that our system is not capable of utilizing event arguments: in fact, Hasler and Orasan (2009) found that only around 20% of the arguments in the same event slot are actually coreferent for coreferential event pairs in the ACE 2005 corpus. Furthermore, the TAC-KBP corpus uses a relaxed participant identity requirement for event coreference, which makes argument-based matching more difficult.

5.2 Event Sequencing Challenges

Our event sequencing performance is still low despite the introduction of many features. This task is inherently difficult because it requires a system to solve both the script clustering and event ordering tasks. The former task requires both common-sense knowledge and discourse reasoning. Reasoning is more important for long-term links since there are no explicit clues like prepositions and dependencies to be exploited. The ablation study shows that discourse features like sentence distance are more effective, which indicates that our model mainly relies on surface clues and has limited reasoning power.

Furthermore, we observe a strong locality property of After links by skimming the training data: most After link relations are found in a small local region. Since reasoning and coreference based propagation will accumulate local decisions, a system must be accurate on them.

5.2.1 The Ambiguous Boundary of a Script

Besides the above-mentioned challenges, a more fundamental problem is to define the boundary of scripts. Since the definition of scripts is only prototypical event sequences, the boundaries between them are not clear. In Example 3, the event `jailed` is considered to belong to a "Judicial Process" script and `killing` is considered to belong to an"Attack" script[7]. No link is annotated between these two mentions since they are considered to belong to different clusters, even though the "jailed" event is to punish the "killing". Therefore essentially, the current Event Sequencing task simply requires the system to fit these human defined boundaries. In principle, the "Judicial Process" script and the "Attack" script can form a larger script structure, on a higher hierarchical level.

While it is possible to manually define scripts and what kind of events they may contain specifically in a controlled domain, it is difficult to generalize the relations. Most previous work on script induction (Chambers and Jurafsky, 2008; Cheung et al., 2013; Rudinger et al., 2015; Pichotta and Mooney, 2016; Ferraro and Durme, 2016) treats scripts as statistical models where probabilities can be assigned, thereby avoiding the boundary problem. While the script boundaries may be application dependent, a possible solution may rely on the "Goals" in Schank's script theory. The Goal of a script is the final state expected (by the script protagonist) from the sequence of events. Goal oriented scripts may be able to help us explain whether `killing` and `jailed` should be separate: if we take the"killer" as the protagonist, the goal of "kill" is achieved at the point of the victim dying. We leave the investigation on proper theoretical justification to future work.

6 Conclusion

In this paper, we presented a unified graph framework to conduct event coreference and sequencing. We have achieved state-of-the-art results on event coreference and report the first attempt at event sequencing. While we only studied two types of relations, we believe the method can be adopted in broader contexts. In the future, we plan to build a joint model to allow the tasks to mutually improve each other.

In general, analyzing event structure can bring new aspects of knowledge from text. For instance, Event Coreference systems can help group scattered information together. Understanding Event Sequencing can help clarify the discourse structure, which can be useful in other NLP applications, such as solving entity coreference problems (Peng et al., 2015). However, in our investigation, we find that the linguistic theory and definitions for events are not adequate for the computational setting. For example, proper theoretical justification is needed to define event coreference, which should explain the problems, such as argument mismatches. In addition, we also need a theoretical basis for script boundaries. In the future, we will devote our effort to understanding the theoretical and computational aspects of events relations, and utilizing them for other NLP tasks.

Acknowledgements

This research was supported in part by DARPA grant FA8750-18-2-0018 funded under the AIDA program.

References

Jun Araki, Zhengzhong Liu, Eduard Hovy, and Teruko Mitamura. 2014. Detecting Subevent Structure for Event Coreference Resolution. In Nicoletta Calzolari (Conference Chair) Khalid Choukri Piperidis, Thierry Declerck, Hrafn Loftsson, Bente Maegaard Stelios, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4553–4558, Reykjavik, Iceland. European Language Resources Association (ELRA).

Jun Araki. 2018. *Extraction of Event Structures from Text*. Ph.D. thesis, Carnegie Mellon University.

A. Bagga and B. Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 79–85. Association for Computational Linguistics.

[7]Script names are taken from the annotation guideline: `http://cairo.lti.cs.cmu.edu/kbp/2016/after/annotation`

Anders Björkelund and Jonas Kuhn. 2014. Learning Structured Perceptrons for Coreference Resolution with Latent Antecedents and Non-local Features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 47–57.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised Learning of Narrative Event Chains. In *ACL '08 Meeting of the Association for Computational Linguistics*, pages 789–797.

Nathanael Chambers and Dan Jurafsky. 2010. A Database of Narrative Schemas. In *LREC 2010, Seventh International Conference on Language Resources and Evaluation*.

Nathanael Chambers and Dan Jurafsky. 2011. Template-Based Information Extraction without the Templates. In *HLT '11 Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 976–986.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense Event Ordering with a Multi-Pass Architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

Zheng Chen and H Ji. 2009. Graph-based event coreference resolution. *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing*, (August):54–57.

Chen Chen and Vincent Ng. 2013. Chinese Event Coreference Resolution: Understanding the State of the Art. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*, number October, pages 822–828.

Chen Chen and Vincent Ng. 2015. Chinese Event Coreference Resolution : An Unsupervised Probabilistic Model Rivaling Supervised Resolvers. In *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*, pages 1097–1107.

Zheng Chen, H Ji, and Robert Haralick. 2009. A pairwise event coreference model, feature impact and evaluation for event coreference resolution. In *Proceedings of the Workshop on Events in Emerging Text Types*, number 3, pages 17–22.

JC Kit Cheung, H Poon, and Lucy Vanderwende. 2013. Probabilistic Frame Induction. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL/HLT 2013)*.

Nancy Chinchor. 1992. MUC-5 EVALUATION METRIC. In *Proceedings of the 5th Conference on Message Understanding*, pages 69–78.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the Conference on Empirical Methods in NLP (EMNLP 2002)*, number July, pages 1–8.

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

Agata Cybulska and Piek Vossen. 2012. Using Semantic Relations to Solve Event Coreference in Text. In *SemRel2012 in conjunction with LREC2012*, pages 60–67.

Agata Cybulska and Piek Vossen. 2014. Using a sledgehammer to crack a nut? Lexical diversity and event coreference resolution. *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4545–4552.

Dipanjan Das and NA Smith. 2011. Semi-Supervised Frame-Semantic Parsing for Unknown Predicates. In *HLT '11 Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, volume 1, pages 1435–1444.

Quang Xuan Do, Wei Lu, and Dan Roth. 2012. Joint Inference for Event Timeline Construction. *EMNLP-CoNLL '12 Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, (July):677–687.

Greg Durrett and Dan Klein. 2014. A Joint Model for Entity Analysis: Coreference, Typing, and Linking. *Proceedings of the Transactions of the Association for Computational Linguistics*.

Eraldo Rezende Fernandes, Cícero Nogueira dos Santos, and Ruy Luiz Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. *Joint Conference on {EMNLP} and {CoNLL-Shared} Task*, pages 41–48.

Francis Ferraro and Benjamin Van Durme. 2016. A Unified Bayesian Model of Scripts , Frames and Language. *Proceedings of the 30th Conference on Artificial Intelligence (AAAI 2016)*, pages 2601–2607.

Laura Hasler and Constantin Orasan. 2009. Do coreferential arguments make event mentions coreferential? In *Proceedings of DAARC*, pages 151–163.

Eduard Hovy, T Mitamura, F Verdejo, J Araki, and Andrew Philpot. 2013. Events are Not Simple: Identity, Non-Identity, and Quasi-Identity. In *The 1st Workshop on EVENTS: Definition, Detection, Coreference and Representation, NAACL-HLT 2013 Workshop*, pages 21–28, Atlanta.

Kevin Humphreys, Robert Gaizauskas, and Saliha Azzam. 1997. Event coreference for information extraction. In *Proceedings of a Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts, 35 th Annual Meeting of Assoc. for Computational Linguistics*, pages 75–81, Madrid.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint Entity and Event Coreference Resolution across Documents. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing and Natural Language Learning*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end Neural Coreference Resolution. In *EMNLP 2017*.

Zhengzhong Liu, Jun Araki, Eduard Hovy, and Teruko Mitamura. 2014. Supervised Within-Document Event Coreference using Information Propagation. In Nicoletta Calzolari (Conference Chair) Khalid Choukri Piperidis, Thierry Declerck, Hrafn Loftsson, Bente Maegaard Stelios, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4539–4544, Reykjavik, Iceland. European Language Resources Association (ELRA).

Jing Lu and Vincent Ng. 2017. Joint Learning for Event Coreference Resolution. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 90–101.

Jing Lu, Deepak Venugopal, Vibhav Gogate, and Vincent Ng. 2016. Joint Inference for Event Reference Resolution. In *Proceedings of the 26th International Conference on Computational Linguistics*.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, (October):25–32.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. *Proceedings of 52nd Annual Meeting of the ACL: System Demonstrations*, pages 55–60.

Teruko Mitamura, Zhengzhong Liu, and Eduard Hovy. 2015. Overview of TAC KBP 2015 Event Nugget Track. In *TAC KBP 2015*, pages 1–31.

Teruko Mitamura, Zhengzhong Liu, and Eduard Hovy. 2018. Events Detection, Coreference and Sequencing: What's next? Overview of the TAC KBP 2017 Event Track. In *TAC 2017*, pages 1–42.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A Corpus and Evaluation Framework for Deeper Understanding of Commonsense Stories. In *Proceedings of NAACL-HLT 2016*, pages 839–849.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, number July, pages 104–111, Philadelphia.

Haoruo Peng, Daniel Khashabi, and Dan Roth. 2015. Solving Hard Coreference Problems. In *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*, pages 809–819, Denver, Colorado.

Haoruo Peng, Yangqi Song, and Dan Roth. 2016. Event Detection and Co-reference with Minimal Supervision. In *EMNLP 2016*.

Karl Pichotta and Raymond J. Mooney. 2016. Using Sentence-Level LSTM Language Models for Script Inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 279–289.

James Pustejovsky, Patrick Hanks, Roser Saurí, Andrew See, Robert Gaizauskas, Andrea Setzer, Beth Sundheim, David Day, Lisa Ferro, and Dragomir. 2002. The TIMEBANK Corpus. *Natural Language Processing and Information Systems*, 4592:647–656.

Marta Recasens and Eduard Hovy. 2011. Blanc: Implementing the rand index for coreference evaluation. *Natural Language Engineering*, 1(1).

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. 2015. Script Induction as Language Modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1681–1686.

S Sangeetha and Michael Arock. 2012. Event Coreference Resolution using Mincut based Graph Clustering. *International Journal of Computing & Information Sciences*, pages 253–260.

Roger C Schank and Robert P Abelson. 1977. *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum Associates.

Stephen Tratz and Eduard Hovy. 2011. A fast, accurate, non-projective, semantically-enriched parser. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, number 2010, pages 1257–1268.

Naushad Uzzaman and James F Allen. 2010. TRIPS and TRIOS System for TempEval-2: Extracting Temporal Information from Text. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, number July, pages 276–283.

Naushad UzZaman, Hector Llorens, Leon Derczynski, Marc Verhagen, James Allen, and James Pustejovsky. 2013. Semeval-2013 task 1: {T}empeval-3: Evaluating time expressions, events, and temporal relations. In *Second joint conference on lexical and computational semantics (* SEM)*, volume 2, pages 1–9.

Naushad UzZaman. 2012. *Interpreting the Temporal Aspects of Language*. Ph.D. thesis, University of Rochester.

Zeno Vendler. 1957. Verbs and times. *The Philosophical Review*, 66(2):143–160.

Piek Vossen and Tommaso Caselli. 2015. Storylines for structuring massive streams of news. In *Proceedings of the First Workshop on Computing News Story Lines*, pages 40–49.

DIDEC: The Dutch Image Description and Eye-tracking Corpus

Emiel van Miltenburg
Vrije Universiteit Amsterdam
Emiel.van.Miltenburg@vu.nl

Ákos Kádár
Tilburg University
A.Kadar@tilburguniversity.edu

Ruud Koolen
Tilburg University
R.M.F.Koolen@tilburguniversity.edu

Emiel Krahmer
Tilburg University
E.J.Krahmer@tilburguniversity.edu

Abstract

We present a corpus of spoken Dutch image descriptions, paired with two sets of eye-tracking data: *free viewing*, where participants look at images without any particular purpose, and *description viewing*, where we track eye movements while participants produce spoken descriptions of the images they are viewing. This paper describes the data collection procedure and the corpus itself, and provides an initial analysis of self-corrections in image descriptions. We also present two studies showing the potential of this data. Though these studies mainly serve as an example, we do find two interesting results: (1) the eye-tracking data for the description viewing task is more coherent than for the free-viewing task; (2) variation in image descriptions (also called *image specificity*; Jas and Parikh, 2015) is only moderately correlated across different languages. Our corpus can be used to gain a deeper understanding of the image description task, particularly how visual attention is correlated with the image description process.

Title and Abstract in Dutch

DIDEC: Een corpus van afbeeldingen met Nederlandstalige beschrijvingen en eye-tracking data

Wij presenteren DIDEC, een corpus van foto's met gesproken Nederlandse beschrijvingen en twee verschillende soorten *eye-tracking* data: ofwel verzameld tijdens het beschrijven van de afbeeldingen, ofwel verzameld terwijl de proefpersonen alleen maar keken naar de afbeeldingen (zonder ze te hoeven beschrijven). Dit artikel beschrijft de dataverzameling, alsook een eerste analyse van de zelf-correcties in de beschrijvingen. Daarnaast beschrijven we twee voorbeeldstudies om aan te geven wat er mogelijk is met DIDEC. Deze studies geven twee interessante resultaten: (1) de eye-tracking data verzameld tijdens het beschrijven van de afbeeldingen is eenduidiger dan de data verzameld tijdens het bekijken van de afbeeldingen; (2) de variatie in de beschrijvingen voor iedere afbeelding (ook wel *afbeeldingsspecificiteit* genoemd; Jas and Parikh, 2015) is slechts matig gecorreleerd tussen verschillende talen (Duits, Nederlands, Engels). Ons corpus kan gebruikt worden om beter te begrijpen hoe mensen afbeeldingen beschrijven, en in het bijzonder wat de rol is van visuele aandacht op het beschrijvingsproces.

1 Introduction

Automatic image description is a task at the intersection of Computer Vision (CV) and Natural Language Processing (NLP). The goal is for machines to automatically produce natural language descriptions for any image (Bernardi et al., 2016). The field of automatic image description saw an explosive growth in 2014 with the release of the Flickr30K and MS COCO datasets: two corpora of images collected from Flickr, with 5 crowd-sourced descriptions per image (Young et al., 2014; Lin et al., 2014). These resources enabled researchers to train end-to-end systems that automatically learn a mapping between images and text (Vinyals et al., 2015), but also to better understand how humans describe images (e.g.,

Proceedings of the 27th International Conference on Computational Linguistics, pages 3658–3669
Santa Fe, New Mexico, USA, August 20-26, 2018.

Raw	Een hele kudde schapen <uh> met een man <corr> met een herder erachter en een pakezel.
Translation	A whole herd of sheep <uh> with a man <corr> with a shepherd behind them and a mule.
Normalized	Een hele kudde schapen met een herder erachter en een pakezel
Translation	A whole herd of sheep with a shepherd behind them and a mule.

Figure 1: Example item from DIDEC, with the annotated raw transcription, and the intended description. Left: image from MS COCO (originally by Jacinta Lluch Valero, CC BY-SA 2.0), Right: image overlaid with an eye-tracking heatmap. Glosses were only added for presentation in this paper.

Van Miltenburg et al., 2016). However, existing datasets can only provide limited insight into the way humans produce image descriptions, because they only contain the *result* of that process, and do not tell us anything about *how the descriptions came about*. This kind of process information can be very insightful for developing image description systems, which is why we decided to collect a new dataset.

One important part of the human image description process is *visual attention*, i.e. which parts of the image people look at when they are asked to describe an image. Coco and Keller (2012) show that there are similarities between sequences of fixated objects in scan patterns and the sequences of words in the sentences that were produced about the images. This idea has been carried over to automatic image description systems in the form of *attention-based models*. Xu et al. (2015) show that one can improve the performance of an image description model by adding an attention module that learns to attend to salient parts of an image as it produces a description. Their model produces attention maps at every time step when it produces the next word. Lu et al. (2017) improve this approach by having the model learn when visual information is or is not relevant to produce a particular word or phrase.

To better understand the role of visual attention in image description, we need a real-time dataset that shows us where the participants are looking as they are producing the descriptions. We present such a dataset: the Dutch Image Description and Eye-tracking Corpus (DIDEC). DIDEC contains 307 images from MS COCO that are both in SALICON (Jiang et al., 2015) and the Visual Genome dataset (Krishna et al., 2017). SALICON is a growing collection of mouse-tracking data, which is used to generate *attention maps*: heatmaps that show which parts of an image are salient and attract attention. The Visual Genome is a knowledge base that combines metadata from different sources about the images it contains. Thus, future researchers can use information from all these different sources in their analysis of our data.

Each image in DIDEC is provided with spoken descriptions and real-time eye-tracking data. There are between 14 and 16 spoken descriptions per image. Each of these descriptions was manually transcribed and annotated. We provide the audio with two kinds of transcriptions (an example is given in Figure 1):

1. Raw descriptions, annotated with markers for repetitions, corrections, and (filled) pauses.
2. Normalized descriptions, without repetitions, and with the corrections suggested by the speaker.

Having these two kinds of descriptions enables us to get a better understanding of the language production process, for example showing exactly where participants experience increased cognitive effort. The normalized descriptions facilitate comparison with written descriptions and improve searchability of the corpus. We also provide two kinds of eye-tracking data:

1. Free viewing: eye-tracking data collected without any concurrent task.
2. Description viewing: eye-tracking data collected simultaneously with the spoken descriptions.

These two sets of eye-tracking data allow us to study the influence of the description task on visual attention. Earlier studies have shown that different tasks may cause different patterns of visual attention (Buswell, 1935; Yarbus, 1967; Coco and Keller, 2014). Our eye-tracking data is complementary to the mouse-tracking data in SALICON, which can only be used to study *bottom-up* attention (driven by the image), and not *top-down* attention (driven by a specific task, such as image description). This difference is further discussed in Section 4. Furthermore, because we collected *spoken* image descriptions, the descriptions are aligned with the eye-tracking data in the description viewing task. This is useful when studying phenomena like self-correction (Section 3.2).

Contributions. This paper introduces DIDEC, a corpus of spoken image descriptions with eye-tracking data. We explain how the corpus was created (§ 2), and provide general statistics about the resource, along with a short discussion of the annotated corrections, providing insight in the description process (§ 3). Then, we present two initial studies to show different possible uses of our dataset. The first study focuses on the effect of task on visual attention, where we show that the eye-tracking data for the image description task is more coherent than the free-viewing data (§ 4). The second study looks at *image specificity* (Jas and Parikh, 2015) across different languages, and whether it is possible to predict image specificity from eye-tracking data. We provide a more efficient, multilingual re-implementation of Jas and Parikh's measure, and show that image specificity is only moderately correlated across different languages, and cannot be predicted directly from attention map similarity (§ 5). Our corpus is freely available, along with an exploration interface, and all the materials that were used to create the dataset.[1]

2 Procedure

We carried out an eye tracking experiment consisting of two separate sub-experiments, which represented two tasks: (1) a free viewing task, during which participants looked at images while we tracked their eye movements, and (2) a task in which participants were asked to produce spoken descriptions of the images, while again their eye movements were recorded. There were different participants for the two sub-experiments, so no image was viewed twice by the same participant.

Data and Materials. Our image stimuli came from MS COCO (Lin et al., 2014), which contains over 200K images with 5 English descriptions each. We selected 307 images matching the following criteria: they should be in landscape orientation, and be part of both the SALICON and the Visual Genome dataset (Krishna et al., 2017). The latter was done for maximum compatibility with other projects.

In order to avoid lengthy experiments, we made three subsets of images, which we refer to as lists in the corpus: one list of 103 images, and two lists of 102 images. In both tasks, participants saw only one list of images. Participants were randomly assigned to one of the lists, with each between 14 and 16 participants. To avoid order effects, we made two versions of each list, which reflect the two fixed random orders in which the images were shown. We registered eye movements with an SMI RED 250 device, operated by the IviewX and the ExperimentCenter software packages.[2] We recorded the image descriptions using a headset microphone.

Free viewing versus Production viewing. In the free viewing task, subjects viewed images for three seconds while their eye movements were recorded. In the image description task, participants also viewed images, but this time they were also asked to produce a description of the current image (while their eye movements were again tracked). The instructions for this task were translated from the original MS COCO instructions. Participants could take as much time as needed for every trial to provide a proper description. In both tasks, every trial started with a cross in the middle of the screen, which had to be fixated for one second in order to launch the appearance of the image. All images in our study both occurred in the free viewing task and in the image description task, but always with different participants. This way, each image viewed by the participants was new to them, preventing any possible

[1] Our resource is available at: `http://didec.uvt.nl`

[2] The eye tracker had a sampling rate of 250 Hz. The stimulus materials were displayed on a 22 inch P2210 Dell monitor, with the resolution set to 1680 x 1050 pixels. The images were resized to 1267 x 950 pixels (without changing the aspect ratio), surrounded by grey borders. These borders were required because eye-tracking measurements outside the calibration area (i.e., in the most peripheral areas of the screen) are not reliable. The viewing distance was 70 cm.

familiarity effects. Avoiding any confounding from familiarity effects also means we are forced to carry out a between-subjects analysis to study the effect of the task on the viewing patterns for the same image.

Transcription and annotation. After exporting the recordings for each trial, we automatically transcribed the descriptions using the built-in Dictation function from macOS Sierra 10.12.6.[3] The transcriptions were manually corrected by a native speaker of Dutch. To get an idea of the actual quality of the automatic transcriptions, we computed the word error rate (WER) for the automatic transcriptions, as compared to the corrected transcriptions.[4] This resulted in a WER-score of 37%.

We refer to the transcribed descriptions in the corpus as *literal descriptions*. In addition, the annotator marked repetitions, corrections, and (filled) pauses (*um, uh*, or silence longer than 2 seconds) by the speaker. We will later use these meta-linguistic annotations to gain more insight into the image description process. Finally, our annotator provided the *normalized descriptions*, without filled pauses or repetitions and with the repairs taken into account.

Participants. Our participants were 112 Dutch students who earned course credits for their participation: 54 students performed the free viewing task, while 58 students completed the image description task. We could not use the data of 19 participants (6 in the free viewing task; 13 in the image description task), since eye movements for these people were not recorded succesfully, or only partially. This was mainly due to the length of the experiments, and to the fact that speaking could distort the eyetracking signal. We tried to prevent this issue by calibrating participants' eyes to the eyetracker twice: once before the start, and once halfway. The final data set consists of data for 48 participants (34 women) in the free-viewing condition, with a mean age of 22 years and 3 months; and data for 45 participants (35 women) in the image description condition, with a mean age of 22 years and 6 months.[5]

Our experiment followed standard ethical procedures. After entering the lab, participants were seated in a soundproof booth, and read and signed the consent form. This form contained a general description of the experimental task, an indication of the duration of the experiment, contact information, and information about data storage. Participants needed to give explicit permission to make available their audio recordings and eye movement data for research purposes; otherwise, they would not participate. Also, participants were allowed to quit the experiment at any stage and still earn credits.

3 General results: the DIDEC corpus

In the description condition, 45 participants produced 4604 descriptions (59,248 tokens), leading to an average of 15 descriptions per image (min 14, max 16). The average description length for the normalized descriptions is 12.87 tokens (Median: 12, SD: 6.45). By comparison, the written English descriptions in MS COCO are shorter (average: 10.78 tokens) and have a lower variance in description length (SD: 2.65).[6] We checked to see if the difference in length is due to any differences between Dutch and English, using the Flickr30K validation set (data from Van Miltenburg et al., 2017). We found that the English descriptions are in fact *longer* than the Dutch ones (with a mean of 12.77 tokens for English (SD: 5.67) versus 10.47 tokens for Dutch (SD: 4.45)). These findings are in line with earlier findings from Drieman (1962) and others that spoken descriptions are typically longer than written ones. We discuss the differences between spoken and written language in more detail in (van Miltenburg et al., 2018).

We found a high degree of variation in description length across different participants. The difference between the lowest and highest median description length is 16.5 tokens (Lowest: 8, Highest: 24.5, Mean: 12.30, SD: 4.15). We also checked whether sentence length decreases with length of experiment, by correlating sentence length with the order in which the images were presented. We found a Spearman correlation of 0.06, suggesting that order had no effect on description length. Following this, we looked

[3] This required us to emulate a microphone using SoundFlower 2.0b2, to use Audacity 2.1.0 to play the recordings and direct the output through the emulated microphone to the Dictation tool.

[4] We used the evaluation script from: `https://github.com/belambert/asr-evaluation`

[5] For 3 participants in the description task, and 4 participants in the free viewing task only a small subset of the eye-tracking data is missing (14 trials in total for the description task, and 7 trials in the free viewing task). We decided to keep these participants and treat the trials as missing data.

[6] We only counted the description lengths for the 307 images that are also in DIDEC. Since DIDEC lacks periods at the end of the descriptions, we also stripped them from the MS COCO descriptions. We used the SpaCy tokenizer to obtain the tokens.

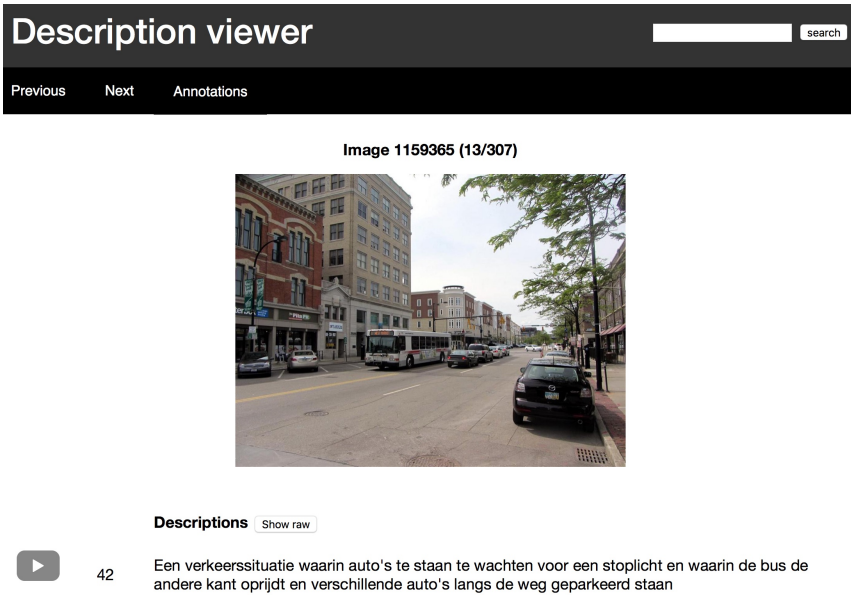


Figure 2: The description viewer provides a browser-based interface to the corpus. Users can browse through the images, search for specific words or annotations, and listen to the spoken descriptions. (Displayed image by David Wilson, CC BY 2.0)

at the variation in description length between images. We found that the difference between the lowest and highest median description length is 15 tokens (Lowest: 6, Highest: 21, Mean: 11.75, SD: 2.46). We conclude that there is a greater variability between participants than between images.

3.1 Viewer tool

We made a description viewer application that allows users to browse through the images, read the annotated descriptions, and listen to the spoken descriptions. Users can also search the descriptions for particular annotations, or for the occurrence of particular words. The description viewer will then return a selection of the images where at least one of the descriptions contains that particular word or annotation. See Figure 2 for an impression of the interface. The viewer tool can be downloaded along with our data from the corpus website.

3.2 Exploring the annotations in the dataset: descriptions with corrections

Recall that we also annotated basic meta-linguistic information to the raw descriptions, such as pauses, repetitions, and corrections. Table 1 shows the number of times each label was annotated. We chose to add these labels because they may inform us about the image description process. For example, one might expect participants to use more filled pauses and repetitions if the image is more complex or unclear (cf. Gatt et al., 2017). Repetitions, in this case, would signal initial uncertainty about the interpretation of the image, followed up by a confirmation that their initial interpretation was correct.

Tag	Meaning	Count
<uh>	Filled pause	1277
<corr>	Correction	693
<rep>	Repetition	139
<pause>	Pause	123
<?>	Inaudible	23

Table 1: Annotation counts.

We will now look at some examples of corrections in the image description data. This will give us some idea of why people tend to make corrections in their descriptions, and what this process looks like. One of the first studies on this topic is provided by Levelt (1983), who discusses a corpus of 959 repairs that were spontaneously made by Dutch speakers after they were asked to describe visual patterns. The difference between DIDEC and Levelt's corpus is that the latter consists of abstract stimuli while DIDEC uses pictures of real-life situations. Levelt used his data to study monitoring (roughly: critically observing one's own speech production), the use of editing terms (e.g. *uh, sorry, no, I mean* ...), and how people actually carry out repairs. Studies like these informed Levelt's seminal model of speech

Figure 3: Eye-tracking data for example (3). Numbers indicate the following: 1. Start of experiment, 2. Speech onset, 3. Speaker realizes her mistake: the group hasn't ordered yet, 4. Start of corrected description, 5. End of description. (Original image by Malcolm Manners, CC BY 2.0)

production (Levelt, 1989). For reasons of space, we will only look at four examples from our dataset, but we hope to show that these kinds of examples warrant further consideration. Looking through the data, many corrections are due to mispronunciations, as in (1).

(1) Een hele grote prie <corr> pizza met drie jongens
A very large pri <corr> pizza with three boys

This particular mispronunciation is a so-called *anticipation error*, one of the most frequent kinds of speech errors (Fromkin, 1971). As the speaker is saying *pizza*, she is already preparing to say *three*, and accidentally inserts the *r* in the onset of *pizza*. Besides mispronunciations, there are also more complex cases. Figure 1 already provided an interesting example, repeated for convenience in (2):

(2) Een hele kudde schapen <uh> met een man <corr> met een herder erachter en een pakezel.
A whole herd of sheep <uh> with a man <corr> with a shepherd behind them and a mule.

What is interesting about this example is that the original expression *with a man* was already correct. The correction *man* → *shepherd* was made to be more specific, so as to produce a more informative description. A possible reason why the speaker did not immediately say 'shepherd' instead of 'man' is that the former is a *(social) role* (Masolo et al., 2004). We cannot determine that the man is a shepherd based on his visual appearance alone, but rather we label him as a shepherd on the basis of the context of him interacting with a herd of sheep. After making this inference, the original label is replaced.

The example in (3) shows a correction after making an incorrect prediction about the situation in Figure 3. Initially the speaker thinks the group is already eating, but actually they haven't ordered yet.

(3) Gezelschap die aan het eten is of <corr> die in een restaurant zit en iets willen gaan bestellen.
Group of people that is eating or <corr> that is sitting in a restaurant and is about to order.

What is interesting here is that we can actually see the correction reflected in the eye-tracking data. Figure 3 shows the attention map along with the scanning pattern corresponding to the eye movements. (Underline colors in the example correspond to the colors in the figure.) The participant starts by scanning the situation and looking at the people at the table. During this time, she starts speaking, but then she realizes her mistake upon seeing the menu on the table. She then updates her beliefs about the situation and corrects her utterance. This is a good example of *predictive coding* (see e.g. Clark, 2013).

Finally, (4) provides an example of a participant who rephrases her description when she realizes that her description is ambiguous; Dutch *knuffel* could both mean 'hug/cuddle' and 'cuddly toy' while *knuffeldier* only means 'cuddly animal.' (We ignore the first correction here.)

(4) Een vrouw die een meisje <corr> klein meisje een knuffel geeft <**corr**> knuffeldier.
A woman giving a girl <corr> little girl a cuddle <**corr**> cuddly animal.

The remainder of this paper discusses two case studies – one comparing eye movements during free viewing versus image description, and one looking into the effects of image specificity on the description

	Compared to attention maps from the other task, attention maps from the same task are…		
Task	More similar	Equally similar	Less similar
Description viewing	300	0	7
Free viewing	116	0	191

Table 2: Results for the comparison between Free viewing and description viewing.

produced– both highlighting the potential usage of the DIDEC corpus.

4 Task-dependence in eye tracking: free viewing versus producing descriptions

A potential issue in studying visual attention is that eye-tracking data may differ across tasks. In one of the first ever eye-tracking studies, Buswell (1935) shows that we can observe differences in eye-tracking behavior between people who are freely looking at an image, versus when they are asked to look for particular objects in the same image. Yarbus (1967) presents a study in which participants are asked to carry out seven different visual tasks, and shows that we can observe differences in eye-movement patterns between each of the different tasks. He argues that "eye movements reflect the human thought process." Finally, Coco and Keller (2014) show that it is possible to train a classifier to distinguish eye-tracking data for three different tasks: object naming, scene description, and visual search. This suggests that in order to model different tasks, one should also collect different sets of eye-tracking data.

Bottom-up versus top-down attention. The literature on visual attention modeling identifies two kinds of salience. On the one hand, there is bottom-up, task-independent visual salience, which is typically image-driven. On the other hand, there is top-down, task-dependent salience, where attention is driven by the task that people may have in viewing the image (Borji and Itti, 2013; Itti and Koch, 2000). Visual attention models are usually designed to predict general, task-free salience (Bylinskii et al., 2016). This prediction task is exactly what the SALICON dataset was developed for.

Free viewing versus description viewing. DIDEC was developed with this top-down versus bottom-up distinction in mind, so that we could compare different modes of viewing the images. The free-viewing task corresponds to bottom-up attention; because there are no explicit instructions of where to look at or what to do, participants only have the image to guide their attention. As such, they are drawn towards the most salient parts of the image. The description viewing task corresponds to top-down attention; because our participants are asked to describe the images, their attention is also guided by what they think might be the most conceptually important parts of the images.

Analysis. To what extent do people differ in their visual attention, between our two tasks? We decided to test this by comparing the attention maps computed on the basis of the eye-tracking data for both tasks. For each image, for each participant, we used their fixations to generate an attention map. Then, for each image, we computed the within-task and between-task average pairwise similarities between the attention maps.[7] By looking at the difference between the within-task similarity and the between-task similarity, we can see if there is consistently more agreement within each task than between the tasks.

Results. Table 2 shows the results. We find that, on average, attention maps from the image description task tend to be more similar to each other than to the attention maps from the free viewing task. But when we look at the attention maps from the free viewing task, we see that they are only more similar to each other 38% of the time (116 out of 307). In 62% of the cases, the between-task similarity is higher than the within-task similarity for the free viewing data. Figure 4 shows the distribution of the scores. We conclude that the image description task reduces noise in the collection of eye-tracking data, and produces a more

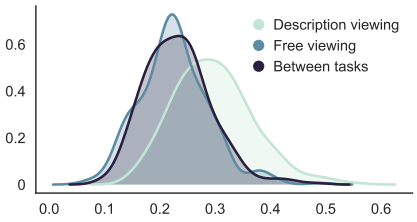


Figure 4: Distribution of the similarity scores within and between tasks.

[7] We use existing code to analyze this data: `https://github.com/NUS-VIP/salicon-evaluation/` The pairwise similarity between attention maps (`CC_score`) is computed using the Pearson correlation.

coherent set of attention maps.

5 Image specificity

Whenever you ask multiple people to describe the same image, you rarely get the same description. Jas and Parikh (2015) show that this variation is not consistent: some images elicit more variation than others. In their terminology: some images are *specific*, resulting in little variation in the descriptions, while others are more *ambiguous*. Following up on this observation, Jas and Parikh (2015) propose an automated measure for image specificity in English, and show that it correlates well with human specificity ratings collected for the images from the image memorability dataset (Isola et al., 2011). With a Spearman's ρ of 0.69, their measure is close to human performance (0.72). To show that specificity is really a property of the image, Jas and Parikh (2015) carry out two experiments:

1. Replicating an image description task: if we ask another group of people to provide descriptions for the same set of images, do we then see the same amount of variation for each image? In their experiment, Jas and Parikh obtained a fairly strong correlation of 0.54 between groups, meaning that the variation did not just arise by chance.

2. A regression analysis: can we predict variation between the image descriptions on the basis of an image? Jas and Parikh reveal that image specificity can indeed be predicted from different properties of an image, such as the presence of people and large objects, the absence of generic buildings or blue skies, and the importance of objects that are visible. (Importance is calculated based on the number of mentions for certain objects in a set of image descriptions.)

But if image specificity is indeed a property of the image, we should also be able to correlate image specificity scores across different languages. We will first test this hypothesis for Dutch, English, and German using existing datasets, and then replicate the correlation between Dutch and English specificity scores using DIDEC. In a second step, we will see if we can directly correlate differences in eye-tracking behavior with the Dutch specificity scores. This experiment builds on the results from Coco and Keller (2012), who showed that scan patterns can be used to predict what someone will say about an image.

5.1 The image specificity metric

Jas and Parikh compute image specificity by taking the average similarity between all descriptions of the same image. The similarity between pairs of sentences is determined using WordNet (Fellbaum, 1998):

1. For each word in the first sentence, compute the maximum path similarity between all possible synsets of that word, and all possible synsets of all words in the second sentence. This is an alignment strategy to find the best matches between both sentences.
2. Repeat the process in the opposite direction: for each word in the second sentence, compute the maximum path similarity with the words in the first sentence.
3. Compute the average path similarity, weighted by the importance of each word (determined using TF-IDF on the entire description corpus under consideration).

Using this method, Jas and Parikh (2015) get a correlation of 0.69 with human specificity ratings, close to the inter-annotator correlation of 0.72. Their conclusion is that this is a reliable measure to estimate image specificity. One problem with this measure is that it requires a lexical resource (WordNet) that is not available for every language. Since we want to run the evaluation corpus on the Dutch descriptions, and because the original implementation is relatively slow and difficult to modify, we re-implemented Jas and Parikh (2015)'s image specificity measure. Our reimplementation also achieves a correlation of 0.69 with the human ratings, and 0.99 with the original implementation.[8] Having validated our reimplementation, we replaced WordNet similarity with cosine similarity, using the GoogleNews word vectors (Mikolov et al., 2013). With this modification, we achieve a correlation with human ratings of 0.71, and a correlation of 0.87 with the original implementation. We also ran the same measure using the FastText

[8] We also found that the WordNet lookup is the main bottleneck, and we can significantly speed up the algorithm by caching the word-to-word similarities. We used the built-in `@lru_cache` decorator in Python 3, storing a million input-output pairs.

Language	Type	Source
Dutch	word2vec	Mandera et al. (2017)
English	word2vec	Mikolov et al. (2013)
German	word2vec	Müller (2015)
All	FastText	Bojanowski et al. (2017)

Table 3: Word embeddings used to compute the image specificity metric.

Comparison	Split	word2vec	FastText
NLD, DEU	validation	0.23	**0.47**
NLD, ENG	validation	0.36	**0.40**
DEU, ENG	validation	0.18	**0.41**
DEU, ENG	train	0.16	**0.39**

Table 4: Spearman correlation between automated image specificity scores in different languages, using two sets of word embeddings.

embeddings (Bojanowski et al., 2017), achieving a correlation of 0.69 with the human ratings and 0.86 with the original implementation. This means that the metric performs on par with Jas and Parikh's original measure, but captures slightly different information about the image descriptions.

5.2 Correlating image specificity between different languages

We used the embedding-based specificity metric to compare image descriptions in 3 different languages, using off-the-shelf embeddings (listed in Table 3).[9] We compare English (ENG) descriptions from the Flickr30K dataset (Young et al., 2014) with German (DEU) and Dutch (NLD) descriptions for the same dataset (Elliott et al., 2016; van Miltenburg et al., 2017). Note that the Dutch data comes from a different dataset than ours, which allows for comparison with both English *and* German.

Table 4 presents the correlations between the scores. Our results show a striking difference between scores computed using word2vec embeddings and those computed using FastText embeddings. This difference seems to be due to poor performance of the German model, as the correlations between the Dutch and English scores are reasonably similar between word2vec and FastText (0.36 versus 0.40). The reason for this may be that the word2vec model has limited coverage, while the FastText model uses subword information to compute vectors for tokens that are out-of-vocabulary. This is especially important for languages like German, which uses more compounding and has a richer morphology than English. We observe that the scores based on the FastText embeddings have correlations between 0.39 and 0.47. This means that, to some extent, image specificity is indeed language-independent. In other words, the data suggests that some images just elicit more varied responses than others, and it does not matter whether you speak Dutch, English, or German.

5.3 Replicating the results using DIDEC

Having found a fairly consistent Spearman correlation of about 0.4 between the different languages, we asked ourselves whether this result could be replicated. We pre-registered our hypothesis (yes it can) with the Open Science Foundation before collecting the image description data.[10] After the data collection procedure was finished, we computed the correlation between the Dutch and English image specificity scores for the 307 images in our dataset. We found a Spearman correlation of 0.23, which is lower than expected. One possible explanation for this is that the descriptions in MS COCO are shorter and have a lower variance in description length (as mentioned in §3), which reduces noise in the comparison. To see if this is indeed the case, we repeated the experiment with only 5 descriptions per image, always selecting the median 5 descriptions when sorted by length. With this approach, we find an even lower correlation of 0.20. We conclude that image content only has a limited influence on description diversity.

5.4 Predicting image specificity from eye-tracking data: a negative result

We now turn to look at whether the image specificity scores are correlated with our eye-tracking data. As noted above, this experiment builds on Coco and Keller's (2012) work, which shows that scan patterns can be used to predict what people will say about an image. Rather than taking Coco and Keller's more

[9] Even though wordnets exist for Dutch (Postma et al., 2016) and German (Hamp and Feldweg, 1997; Henrich and Hinrichs, 2010), we did not use them because they have lower coverage, and we do not need to worry about lemmatization.

[10] Our preregistration can be found at: `https://osf.io/6pc2t/register/565fb3678c5e4a66b5582f67` We deviated from our preregistration in two areas: (1) the experiment took more time than expected, so we split the experiment in two parts; (2) we had to discard more data than expected, so we collected data from more participants to compensate.

advanced approach (involving image segmentation), we use a more naive strategy: (1) We first compute the average pairwise similarity scores between the attention maps. If our participants agreed on where to look, this score will be higher than if our participants attended to different parts of the image. (2) We then correlate those scores with the image specificity scores we computed earlier.

We found no correlation between the eye-tracking data and the image specificity scores. Because Coco and Keller (2012) did find correlations between eye-tracking similarity and description similarity, we conclude that our naive approach is probably too weak to detect any effect. This shows us the limitations of using raw eye-tracking data, which may be too noisy to be compared directly for these kinds of fine-grained predictions. In cases like this, following Coco and Keller, it is better to combine our eye-tracking data with the object-level annotations in MS COCO.

6 Conclusion

We collected a corpus of Dutch image descriptions and eye-tracking data for 307 images, and provided an initial analysis of the self-corrections made by the participants. We have also presented two studies that show some possible uses of our data, but we believe many more analyses are possible. For reasons of space, we have not discussed the effect of modifying the modality of the image description task from written to spoken language, even though we know that modifying the prompt may have an effect on the response (e.g. Baltaretu and Castro Ferreira 2016). In a companion paper, we compare spoken and written image descriptions in both Dutch and English (van Miltenburg et al., 2018). We still plan to semi-automatically annotate Speech Onset Times (SOT) using Praat (Boersma and Weenink, 2017), and to manually correct the output. We define SOT as the start of the utterance, including filled pauses (but excluding coughs and sighs). This is a measure of response time for each image, which is a proxy for the difficulty of producing a description, that could be correlated with e.g. image complexity (cf. Gatt et al., 2017).

Finally, the development of multilingual image description datasets (like Multi30K), has opened up new avenues of research, such as multimodal machine translation (Elliott et al., 2016; Elliott et al., 2017). To the best of our knowledge, a dataset like DIDEC does not exist yet for any other language. We hope that our corpus may serve as an example, inspiring the development of parallel eye-tracking and image description datasets in other languages. This multilingual aspect is important because speakers of different languages may also display differences in familiarity with the contents of an image or, if their language uses a different writing directionality, different eye-tracking behavior (van Miltenburg et al., 2017; Baltaretu et al., 2016). We made all code and data used to build the corpus available on the corpus website, so as to encourage everyone to further study image description as a dynamic process.

7 Acknowledgments

Emiel van Miltenburg is supported via the 2013 NWO Spinoza grant awarded to Piek Vossen. Ákos Kádár is supported through an NWO Aspasia grant awarded to Afra Alishahi. We thank Rein Cozijn for his technical assistance in setting up the eye-tracking experiments, and Kim Tenfelde for her transcription and annotation efforts.

References

Adriana Baltaretu and Thiago Castro Ferreira. 2016. Task demands and individual variation in referring expressions. In *Proceedings of the 9th International Natural Language Generation conference*, pages 89–93, Edinburgh, UK, September 5-8. Association for Computational Linguistics.

Adriana Baltaretu, Emiel J. Krahmer, Carel van Wijk, and Alfons Maes. 2016. Talking about relations: Factors influencing the production of relational descriptions. *Frontiers in Psychology*, 7:103.

Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. 2016. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *Journal of Artificial Intelligence Research*, 55:409–442.

Paul Boersma and David Weenink. 2017. Praat: doing phonetics by computer [computer program]. Version 6.0.35, downloaded from http://www.praat.org/.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ali Borji and Laurent Itti. 2013. State-of-the-art in visual attention modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):185–207.

Guy Thomas Buswell. 1935. How people look at pictures: a study of the psychology and perception in art.

Zora Bylinskii, Tilke Judd, Aude Oliva, Antonio Torralba, and Frdo Durand. 2016. What do different evaluation metrics tell us about saliency models?

Andy Clark. 2013. Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behavioral and brain sciences*, 36(3):181–204.

Moreno I Coco and Frank Keller. 2012. Scan patterns predict sentence production in the cross-modal processing of visual scenes. *Cognitive Science*, 36(7):1204–1223.

Moreno I Coco and Frank Keller. 2014. Classification of visual and linguistic tasks using eye-movement features. *Journal of vision*, 14(3):11–11.

Gerard HJ Drieman. 1962. Differences between written and spoken language: An exploratory study, I. quantitative approach. *Acta Psychologica*, 20:36–57.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30k: Multilingual english-german image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74, Berlin, Germany, August. Association for Computational Linguistics.

Desmond Elliott, Stella Frank, Loïc Barrault, Fethi Bougares, and Lucia Specia. 2017. Findings of the Second Shared Task on Multimodal Machine Translation and Multilingual Image Description. In *Proceedings of the Second Conference on Machine Translation*, Copenhagen, Denmark, September.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press.

Victoria A Fromkin. 1971. The non-anomalous nature of anomalous utterances. *Language*, pages 27–52.

Albert Gatt, Emiel Krahmer, Kees van Deemter, and Roger P.G. van Gompel. 2017. Reference production as search: The impact of domain size on the production of distinguishing descriptions. *Cognitive Science*, 41:1457–1492.

Birgit Hamp and Helmut Feldweg. 1997. Germanet – a lexical-semantic net for german. In *Proceedings of the ACL workshop on Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*.

Verena Henrich and Erhard Hinrichs. 2010. Gernedit - the germanet editing tool. In *Proceedings of the ACL 2010 System Demonstrations*, pages 19–24, Uppsala, Sweden, July. Association for Computational Linguistics.

Phillip Isola, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. 2011. What makes an image memorable? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 145–152.

Laurent Itti and Christof Koch. 2000. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 40(10):1489 – 1506.

Mainak Jas and Devi Parikh. 2015. Image specificity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2727–2736.

Ming Jiang, Shengsheng Huang, Juanyong Duan, and Qi Zhao. 2015. Salicon: Saliency in context. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *Int. J. Comput. Vision*, 123(1):32–73, May.

Willem J.M. Levelt. 1983. Monitoring and self-repair in speech. *Cognition*, 14(1):41 – 104.

Willem JM Levelt. 1989. *Speaking: From intention to articulation*. MIT press.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July.

Paweł Mandera, Emmanuel Keuleers, and Marc Brysbaert. 2017. Explaining human performance in psycholinguistic tasks with models of semantic similarity based on prediction and counting: A review and empirical validation. *Journal of Memory and Language*, 92:57–78.

Claudio Masolo, Laure Vieu, Emanuele Bottazzi, Carola Catenacci, Roberta Ferrario, Aldo Gangemi, and Nicola Guarino. 2004. Social roles and their descriptions. In *Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning*, KR'04, pages 267–277. AAAI Press.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Andreas Müller. 2015. German word embeddings. Available from GitHub at: `http://devmount.github.io/GermanWordEmbeddings/`.

Marten Postma, Emiel van Miltenburg, Roxane Segers, Anneleen Schoen, and Piek Vossen. 2016. Open dutch wordnet. In *Proceedings of the Eighth Global Wordnet Conference*, Bucharest, Romania, January 27-30.

Emiel van Miltenburg, Roser Morante, and Desmond Elliott. 2016. Pragmatic factors in image description: The case of negations. In *Proceedings of the 5th Workshop on Vision and Language*, pages 54–59, Berlin, Germany, August. Association for Computational Linguistics.

Emiel van Miltenburg, Desmond Elliott, and Piek Vossen. 2017. Cross-linguistic differences and similarities in image descriptions. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 21–30, Santiago de Compostela, Spain, September. Association for Computational Linguistics.

Emiel van Miltenburg, Ruud Koolen, and Emiel Krahmer. 2018. Varying image description tasks: spoken versus written descriptions. Submitted.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 3156–3164. IEEE.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France, 07–09 Jul. PMLR.

Alfred L Yarbus. 1967. *Eye movements and vision*. Springer.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Narrative Schema Stability in News Text

Dan Simonson
Georgetown University
Washington, DC
des62@georgetown.edu

Anthony Davis
Ashland, Oregon
tonydavis0@gmail.com

Abstract

We investigate the stability of *narrative schemas* (Chambers & Jurafsky, 2009) automatically induced from a news corpus, representing recurring narratives in a corpus. If such techniques produce meaningful results, we should expect that small changes to the corpus will result in only small changes to the induced schemas. We describe experiments involving successive ablation of a corpus and cross-validation at each stage of ablation, on schemas generated by three different techniques over a general news corpus and topically-specific subcorpora. We also develop a method for evaluating the similarity between sets of narrative schemas, and thus the stability of the schema induction algorithms. This stability analysis affirms the heterogeneous/homogeneous document category hypothesis first presented in Simonson & Davis (2016), whose technique is problematically limited. Additionally, increased ablation leads to increasing stability, so the smaller the remaining corpus, the more stable schema generation appears to be. We surmise that as a corpus grows larger, novel and more varied narratives continue to appear and stability declines, though at some point this decline levels off as new additions to the corpus consist essentially of "more of the same."

1 Introduction

Narrative schemas complement other approaches to the automated analysis of topical and narrative information in documents. Unlike template-filling techniques, they do not require a defined set of human-crafted templates; instead, template-like structures are induced. Unlike topic models, they generate representations in which event types and participant types are organized into relational structures, specifying shared participants between events. Unlike automatic summarization, they generalize over similar but distinct narratives, with the goal of revealing their underlying common elements.

Scripts (or schemas or frames) have long been touted as a way to provide artificial intelligence systems with world knowledge and understanding (Schank & Abelson, 1977). Building these scripts by hand is labor intensive, so automatically learning script knowledge has attracted attention for decades (Balasubramanian et al., 2013; Chambers & Jurafsky, 2009; Mooney & DeJong, 1985; Pichotta & Mooney, 2014, 2015). However, we are also interested in what such techniques reveal about broad, quantitative properties of discourse in general.

Like many unsupervised tasks, the evaluation of schemas is still a matter of debate and depends on their intended use; there is no one broadly accepted method, and nothing that closely models human intuitions about narrative. Chambers & Jurafsky (2008, 2009) proposed and used the *cloze task* to evaluate their procedure, widely adopted in subsequent work (Cheng & Erk, 2018; Jans et al., 2012; Pichotta & Mooney, 2014, 2015). In the cloze task, an event from a sequence in holdout data—either a coreference or sentence-to-sentence sequence—is removed and must be guessed by a model of the events in text.

However, it is questionable whether the cloze task actually requires script knowledge to perform well (Mostafazadeh et al., 2016; Rudinger et al., 2015; Simonson, 2018), and thus whether it is an *effective* measure of schema quality. Cloze does not evaluate schemas directly—only indirectly through the score

Proceedings of the 27th International Conference on Computational Linguistics, pages 3670–3680
Santa Fe, New Mexico, USA, August 20-26, 2018.

used to generate schemas—and is fundamentally unsolvable, even by humans, so it is an open question whether the information needed to perform well on a cloze task corresponds closely to the information within narrative schemas. Others have evaluated schemas directly. Balasubramanian et al. (2013) evaluated schemas manually using mechanical turkers, but this is labor intensive and pre-supposed specific properties about schemas—for example, that "a child exploded a blast" cannot be part of a valid schema, despite that we live in a world where such incidents occur on a regular basis. Simonson & Davis (2016) introduced the NASTEA task for schema evaluation where schemas are used to identify salient entities in a document, but this is dependent on a set of "salient entity annotations" from the New York Times corpus (Sandhaus, 2008). We noted that some document categories are *homogeneous*—that is, requiring only one schema to obtain optimal performance on their task—and others are *heterogeneous*—requiring multiple schemas for optimal performance. It is not clear whether their findings are robust or a mere artifact of the idiosyncrasies of NYT annotations.

In this paper, we turn instead to the stability of narrative schemas, as a method of gauging schema quality complementary to cloze and the NASTEA task. We assume that high-quality schemas are robust; that is, the addition, deletion, or modification of a few narratives in the corpus should not drastically affect the schemas generated. We conducted experiments using both corpus ablation and cross-validation at each stage of ablation. This kind of evaluation is relatively easy to carry out in many sorts of corpora; it does not require any annotation effort and is conceptually straightforward. Two challenges, however, are the computationally intensive nature of these experiments, and the similarity measure employed in comparing sets of narrative schemas.

In Section (2), we describe the data set used in this paper. In Section (3), we describe the schema germinators used in this study, including one novel technique intended to complement the others from prior work. In Section (4), we explain the ablation and cross-validation procedure used to perform a schema stability analysis. in Section (5), we propose the "Fuzzy Jaccard" coefficient used for comparing sets of schemas. In Section (6), we present the results of this study. In Section (7), we discuss the results. In Section (8), we conclude this paper.

2 Data

We performed our stability analysis of narrative schemas on a subset of the *New York Times* Annotated Corpus (Sandhaus, 2008), the same subset as Simonson & Davis (2016). containing over 1.8 million articles from 1987 to 2007.

We selected the same document set from Simonson & Davis (2016) for direct comparison with our previous work, to affirm or disprove our homogeneity-heterogeneity hypothesis. Note that in these earlier results, included an "Education and Schools" category is mentioned (Simonson & Davis (2016), Figure 3), but is unfortunately omitted in Table (1) in that paper. This has been included in our Table (1) here.

Table 1: Counts of document categories selected from the `online_producer` tag for use in this study after pre-processing. Simonson & Davis (2016) chose categories to contain roughly the same number of articles and to represent different sorts of topics.

`online producer` category	counts	`online producer` category	counts
Law and Legislation	52110	United States Armament and Defense	50642
Weddings and Engagements	51195	Computers and the Internet	49413
Crime and Criminals	50981	Labor	46321
Education and Schools	50818	Top/News/Obituaries	36360

For NLP preprocessing, the Stanford CoreNLP suite of tools (Manning et al., 2014),[1] was chosen for comparability to Simonson & Davis (2016)'s original work, itself chosen because it has both parsing and coreference capabilities (de Marneffe et al., 2006; Lee et al., 2013), which are essential for generating schemas. Using the same version of tools as Simonson & Davis (2016), we were able to replicate their

[1] Stanford CoreNLP, Version 3.4.1 (2014-08-27), (via Simonson (2018))

category counts after excluding a number of articles that produced no coreference chains or failed to survive pre-processing.

3 Germinators for Schema Generation

The heart of Chambers & Jurafsky (2009)'s schema induction technique, and our variants derived from it, is a similarity measure between *narrative chains* in the context of a schema and a candidate event verb. A narrative chain is a set of verb-dependency pairs—i.e. slots—that share a common participant filling all the slots. These are constructed from verbs that are statistically associated using a PMI-based measure plus a preference for verb-argument slot pairs in which the argument slot is filled by participants of a consistent type. This similarity measure determines the best candidate verb-argument slot pair to add to an existing chain; a score termed $chainsim'$ is computed for each candidate pair, summing this similarity measure for each candidate and the existing elements of the chain. Lastly, narrative schemas, consisting of merged sets of chains sharing events, are produced.

Here, we focus primarily in generating schemas. The cloze task does not directly evaluate schemas. Rather, it evaluates a component of the model used to generate schemas. For example, in the case of Chambers & Jurafsky (2008, 2009), $chainsim$ and $chainsim'$ are used to rank event verbs in the cloze task. The schemas therein generated are not directly evaluated. Thus, we draw a line after $chainsim'$ and other techniques used on the cloze task, referring to them as the *candidate score*.

However, a candidate score alone is not enough to generate schemas. We must decide on a technique for traversing candidates and interpreting their scores as prospective additions to a schemas under construction. We will refer to these techniques collectively as *germinators*.

We employ two previously devised germinators for schema generation (Section 3.1): counter-training (Simonson & Davis, 2015) and Chambers' original technique or schema germination (Chambers & Jurafsky, 2009), though here referred to as "linear induction." Both of these techniques are relatively deterministic, so for comparison, we present a novel stochastic technique for schema generation called the "random walker" germinator (Section 3.2). We expect this stochastic technique to be less stable than its deterministic counter-parts.

All of the implementations for narrative schema germinators can be found in `durruti`.[2]

3.1 Prior Techniques

Linear induction is what we call Chambers & Jurafsky (2009)'s technique for inducing schemas. In this technique, the event verbs are considered in order for adding to narrative schemas. Each event verb is compared against an ever-growing set of schemas. New schemas are created when the event verb does not pass a threshold parameter β. In Chambers & Jurafsky (2009)'s original implementation, if a candidate event verb's $chainsim'$ score is greater than β, it is added to the schema that best fit the event verb. We differ in this regard, adding the candidate event verb to every schema for which its $chainsim'$ score crossed the threshold.

Alternatively, *counter-training* (Simonson & Davis, 2015)—inspired by Yangarber (2003)—considers a fixed set of schemas simultaneously, which start as a set of seeds. Candidate events are scored against all schemas, then those scores are penalized based on how many different schemas each candidate fit into. After penalties have been applied, the best candidate event for each schema is added to each schema.

With respect to both techniques, once the decision has been made to add an event to a schema, we insert new events into each schema the same way in both germinators, following Chambers & Jurafsky (2009)'s technique for doing so. Additionally, for both prior germinators and the random walker—described in Section 3.2—$chainsim'$ is used as the underlying score.

3.2 Random Walker

Both prior techniques are deterministic. Adding a "random walker" stochastic germinator potentially provides a nice contrast to these deterministic techniques. For a particular schema, all possible insertions are treated as weighted random choices based on the scores between chains already in a schema and

[2] `https://github.com/thedansimonson/durruti`

candidate events. However, because the scores are pmi-based, the values, if turned into a probability distribution, are nearly uniformly distributed. This causes events selected for schemas to be effectively unrelated to one another. Instead, we use a weighting function to weight the score more appropriately for random selection, in this case $weight(C, vd) = 2^{chainsim'(C,vd)}$, which effectively undoes the $\log$ contained in the score function.

This algorithm departs from a simple random walk, because the weights on the graph change at each step; the current score value for each schema depends not only on the last event added, but on all of the events previously added. The "graph" is thus recomputed based on the current state of the schema being grown.

4 Stability Ablation and Cross-Validation

The stability evaluation procedure alternates two stages: an ablation step and a cross-validation step. At each ablation step, 10% of the total set of documents are removed (not 10% of the previous ablation). Then, using the corpus at each ablation stage, ten-fold cross-validation partitions the set of documents and 9/10ths of the available documents are used in each fold to generate schemas. These splits are not preserved across ablations. While these procedures involve removing portions of the original corpus, the most intuitive way to interpret the intent is in reverse—that is, to think of some sort of search and retrieval procedure yielding slightly different results (cross-validation step) at each step in a larger data collection effort (ablation step).

We generate schemas using each of the three schema germinators described above. Separate sets of schemas are generated from the documents in each category in Table (1), using separate PMI models produced from documents in each category, at each stage of ablation, and for each fold of cross-validation. These sets of schemas are what we compare to one another to gauge their stability in different experimental configurations.

Since the total number of schemas generated is about 4 million, we did not attempt to optimize various parameter values pertinent to each germinator, instead using a single set of approximately optimized values from an earlier study of schemas in a salient named entity detection task.

5 Fuzzy Jaccard Coefficient and the Jaccard Reciprocal Fraction

The result from each run (using a given schema germinator, on documents from a given category at a specified stage of ablation and fold of cross-validation) is a set of narrative schemas. How can we then measure the similarity between different sets of schemas? A schema in one set may be highly similar but not identical to a schema in the other set; see, for example, any two schemas in Figure (3). We wish to take their similarity into account in our overall measure of the similarity between the two sets. It is simple to measure the similiarity J_e between two individual schemas σ and τ, using the Jaccard coefficient, where σ_e and τ_e are the sets of events contained in each schema:

$$J_e(\sigma, \tau) = \frac{|\sigma_e \cap \tau_e|}{|\sigma_e \cup \tau_e|} \tag{1}$$

But evaluating the similarity between two sets of schemas is not so straightforward, particularly when we need to compare thousands of pairs of sets of schemas. Essentially, we would like to determine, for each schema in one set, how similar its best match is in the other. We therefore define a fuzzy Jaccard measure over two sets (of schemas, in this case), by redefining the intersection cardinality in a "fuzzy" way, as:

$$J_{J_e}(S, T) = \frac{|S \cap_{J_e} T|}{|S \cup_{J_e} T|} \tag{2}$$

where S and T are sets and J_e is a symmetric and well-defined comparison between elements of S and T, with values between 0 and 1. We invoke the identity ($|S \cup T| = |S| + |T| - |S \cap T|$) to derive a fuzzy counpart for the cardinality of the union. The fuzzy similarity measure then becomes:

$$J_{J_e}(S, T) = \frac{|S \cap_{J_e} T|}{|S| + |T| - |S \cap_{J_e} T|} \tag{3}$$

This allows us to have to only define $|S \cap_{J_e} T|$, which we define as:

$$|S \cap_{J_e} T| = \sum_{\tau \in T} \max_{\sigma \in S} J_e(\sigma, \tau) \tag{4}$$

The values rendered by this approximation are somewhat misleading, however. It tends to skew low, especially when considering schemas. Two schemas with six events each, sharing five events, have a Jaccard score of only 71%, since the schemas are "punished" for having two events that do not match. The Fuzzy Jaccard measure J_{J_e} further exaggerates this discrepancy. Assume that all of the schemas in two sets of schemas have a J_e value as described above, at 71%. In other words, these are fundamentally two very similar sets of schemas—each schema in one set has a counterpart in the other set sharing five out of six events. Assume also that both sets of schemas are the same size. This gives us:

$$J_{J_e}(S, T) = \frac{|T| \times 0.71}{2|T| - |T| \times 0.71} = 55\% \tag{5}$$

This value misleadingly implies that the sets of schemas are only 55% similar despite each schema in one set having a close match in the other.

If we make a few assumptions about J_{J_e}, we can find a better interpretation for the values it computes. Essentially, what we want to know is the typical value of $|\sigma_e \cap \tau_e|$ implied by a given J_{J_e} value. We define $|\sigma_e \cap \tau_e| = x$ since that is what we want to solve for. If there is such a value, we will assume it is fixed under the maximization in the defined cardinality of the intersection. This also allows us to reduce the summation over T to the cardinality of T. We also assume that $|\sigma_e| = |\tau_e| = \sigma'$ since our schema germinator ceases at a maximum of six events for all germinators:

$$|S \cap_{J_e} T| = \sum_{\tau \in T} \max_{\sigma \in S} \frac{|\sigma_e \cap \tau_e|}{|\sigma_e| + |\tau_e| - |\sigma_e \cap \tau_e|} = |T| \frac{x}{2\sigma' - x} \tag{6}$$

Let us allow for one more approximation, that $|S| = |T|$. This is absolutely true for the counter-training, random walker, and linear induction truncated germinators, since they generate a fixed number of schemas. It is approximately true for the linear induction germinator. Substituting Formula (6) into Formula (3) and making the given approximation yields:

$$J_{J_e}(S, T) = \frac{|S \cap_{J_e} T|}{|S| + |T| - |S \cap_{J_e} T|} = \frac{|T| \frac{x}{2\sigma' - x}}{2|T| - |T| \frac{x}{2\sigma' - x}} = \frac{\frac{x}{2\sigma' - x}}{2 - \frac{x}{2\sigma' - x}} = \frac{x}{2(2\sigma' - x) - x} = \frac{x}{4\sigma' - 3x} \tag{7}$$

Solving for x, we get the Jaccard Reciprocal Fraction, or JRF:

$$x = \frac{4}{J_{J_e}^{-1}(S, T) + 3} = JRF \tag{8}$$

This gives us the typical fraction of shared events between schemas in two sets of schemas, regardless of the size of schemas in each set. As the Fuzzy Jaccard value approaches 1, so does the JRF; as the Fuzzy Jaccard value approaches 0, the denominator approaches infinity, and thus the JRF approaches 0. For the example shown above, $JRF = 4/(1/0.55 + 3) = 0.83 = 5/6$. This is more intuitive while simultaneously preserving the underlying set theoretic machinery and justification for the comparisons performed.[3]

It has been suggested that we include a more sophisticated lexical similarity metric—for example, those presented by Corley & Mihalcea (2005) or Kusner et al. (2015). However, fundamentally, these would obscure the answers to the questions we are asking here, which is, through the lens of coreference chains, dependencies, event verbs, and their pointwise mutual information, how stable are the schemas produced to perturbations in the data used? Were the models used to generate the schemas more dependent on a lexical similarity measure, such would be essential in answering that question—and while a

[3]Preliminary results indicate that the JRF values are quite similar to the raw numeric values computed.

great potential next step in improving schema induction, the problem of lexical similarity has largely been neglected in narrative schemas and the narrative cloze task. We suspect adherence to editorial guidelines within a single publication will likely normalize the most interchangeable terms—for example, selectively choosing "pursue" over "chase." In a multi-newspaper corpus, we suspect lexical distance would be much more important.

6 Results

For each individual pair of sets of schemas within an ablation, we compute Fuzzy Jaccard scores, their means and their standard deviations, transformed into JRF form. Presenting these values here is too cumbersome, however. As an overview, average JRF values across germinators and document categories are shown in Figures (1) and (2).

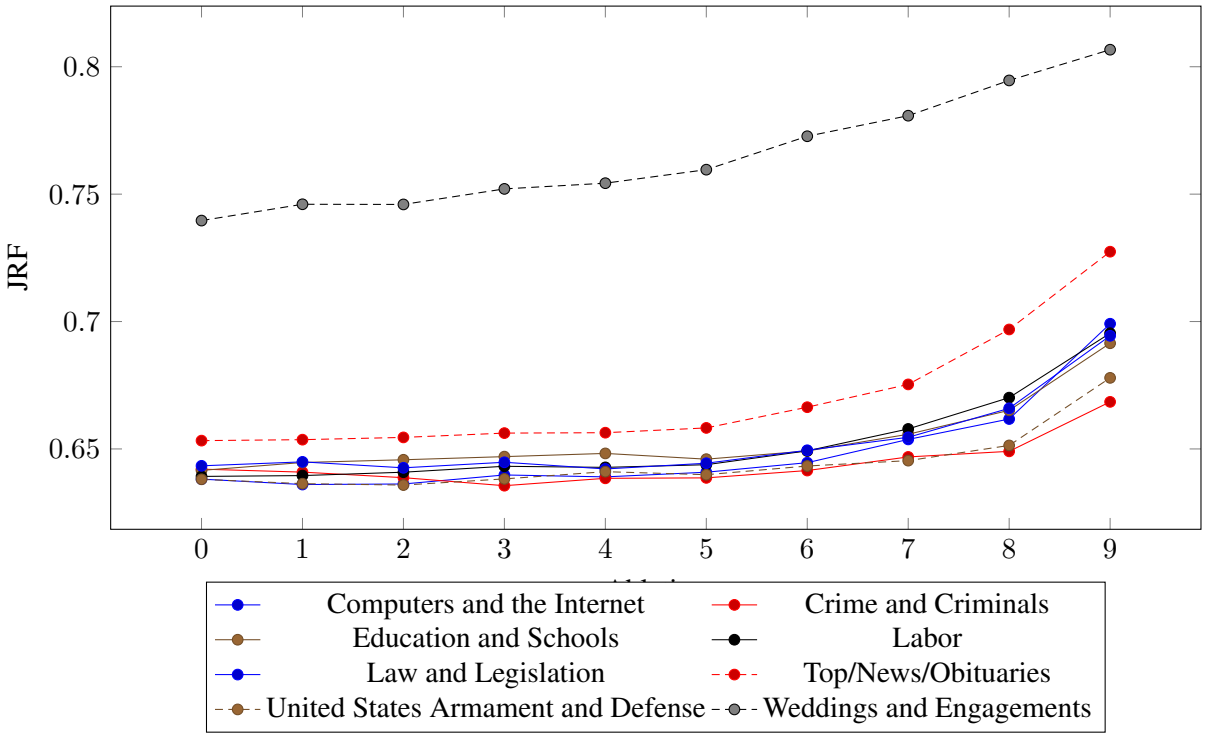


Figure 1: Stability averaged across document categories. Ablation is on the x-axis; Jaccard Reciprocal Fraction (e.g. events typically shared) is on the y-axis.

Note that increasing ablation number refers to a decreasing number of documents; in other words, ablation 8 refers to $8/10^{ths}$ of the documents having been *removed*.

In total, the series of experiments generated a total of 3,978,865 schemas. These are not "unique," as the whole point was to generate schemas that are hopefully as similar to one another as possible. Linear induction produced 2,698,865 schemas, in part a product of its open-ended generation process. Both counter-training and random walker generated 640,000 schemas, as the number of schemas generated within each category was fixed at 800 for practical computational reasons. [4]

Figures (1) and (2) contain the stability values as averaged across different dimensions. Figure (1) averages all stability values for a given ablation across all algorithms, leaving separate values for each category. Conversely, Figure (2) averages the stability values across document categories, leaving each algorithm individually expressed.

In both figures, stability generally increased as the number of total documents decreased. The one exception to this was the linear induction schemas, as shown in Figure (2). The causes and consequences

[4]Since the linear induction truncated algorithm is simply linear induction but with a few schemas clipped off the end, it is not counted as a separate instance of generation.

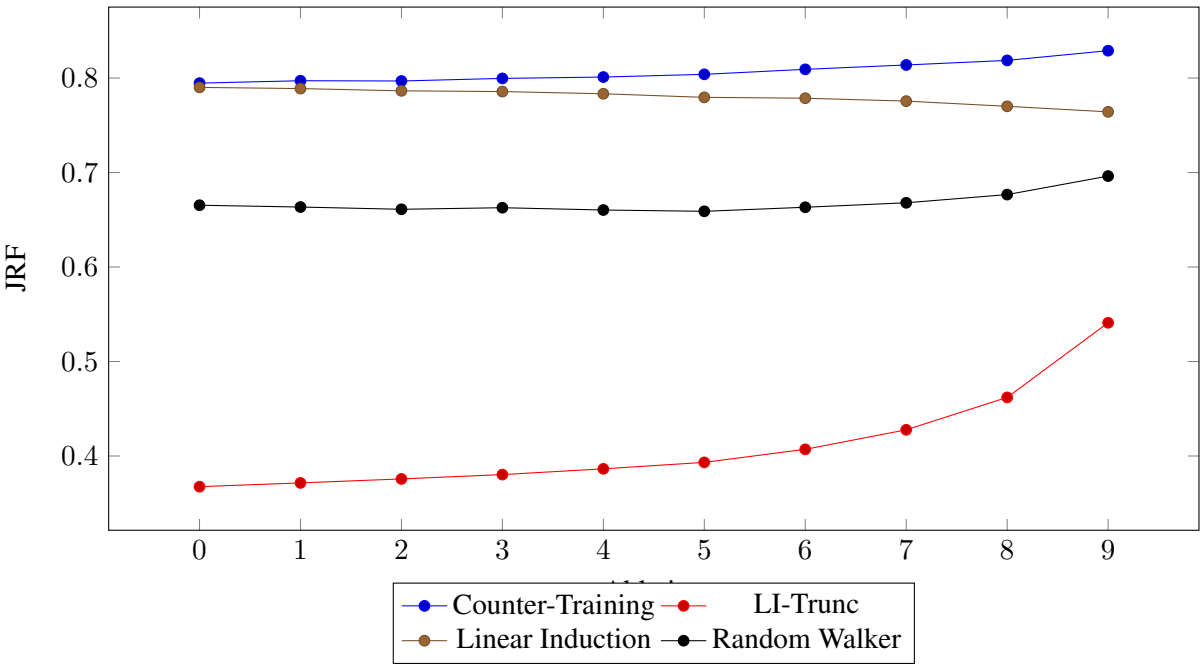


Figure 2: Performance averaged across algorithms. Ablation is on the x-axis; Jaccard Reciprocal Fraction (e.g. events typically shared) is on the y-axis.

of this will be discussed in the next section.

In Figure (1), the document categories Simonson & Davis (2016) found to be homogeneous are notably more stable than the categories we found to be heterogeneous. The difference between similarity scores in each cross-validation was significant ($p < 0.001$) in 140/140 comparisons (t-tests) between the similarity scores of "Weddings and Engagements" and the other document categories for both counter-training and the random walker germinators; the difference was significant ($p < 0.001$) in 137/140 comparisons between "Top/News/Obituaries" and the other document categories for both as well. Two of the insignificant differences occurred during the 9^{th} ablation against "Computers and the Internet" and "Labor," one occurred during the 2^{nd} ablation against "Education and Schools." This does not itself have to do with the content of the schemas, but the differences in similarities internal to the document category itself.

Some examples of schemas generated in this process—six from the counter-training germinator in two different cross-validations of the 0^{th} ablation—are contained in Figure (3). Schemas containing these sorts of events are typical of the Obituaries section. Note that "bear" is simply the lemmatized form of "be born". In each schema, participants of the same chain of argument slots of the verbs are indicated by the same color and shape, but this does not extend across schemas. Each column of shapes is, from left to right, in each schema, `SUBJ`, `OBJ`, and `PREP`, respectively. Dotted boxes indicate slots attested in the data but not linked to a chain; completely blank slots were never attested in the data. Each chain shares a number of types, too many to explicitly enumerate here. The object slot of "survive," for example from the idiom "person was survived by," was typically a generic person type or type that would qualify as a subset of person: "woman," "man," "boy," "microbiologist," "lawyer," etc.

7 Discussion

The first striking aspect of these results is the affirmation that the Weddings and the Obituaries categories, the two determined to be *homogeneous* by Simonson & Davis (2016), were consistently more stable at all ablations than the rest of the document categories. The differences were significantly different the vast majority of the time (277/280), and in cases where this was not the case, only a handful of the stability scores of the heterogeneous categories had increased. This affirms our earlier findings from another angle

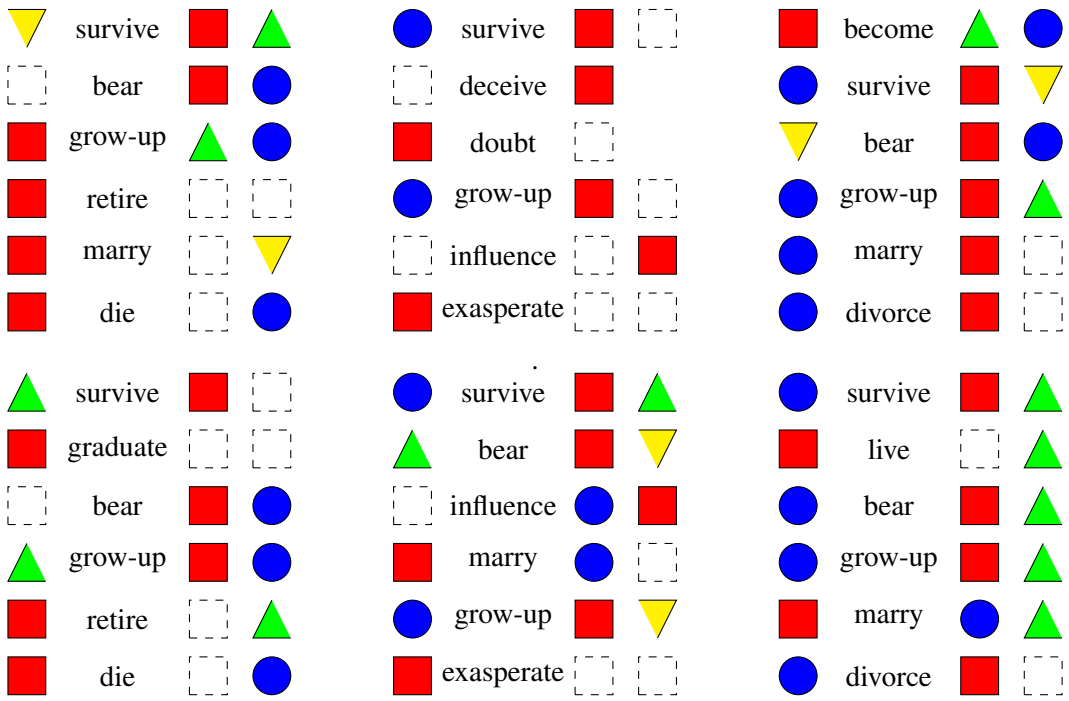


Figure 3: Six schemas generated by the counter-training germinator in two cross-validations, showing varying degrees of similarity between schemas. The two rows show schemas generated from two different cross-validation of the data in the same ablation.

unrelated to the NYT corpus' salient entity annotations, as we managed to show a similar distinction between homogeneous and heterogeneous document categories without salient entity annotations. The document categories are written from templates, so in some respects, it's not surprising that a template extractor should exhibit different properties on the categories written from templates. Simonson & Davis (2016) showed that this can be ascertained from labelled evaluation data; the stability procedure used here shows the same distinction, significantly without labelled data.

Unexpectedly, however, stability increased as the number of documents decreased.

Our initial expectations were that stability would increase with the number of documents. The rationale, as with many statistical learning algorithms, is that the more documents in the training data, the better the algorithm performs, as it has more examples to leverage and overall improves performance. This presupposes that better stability mirrors performance improvements, and that with more data, a germinator can better converge on the ground truth reflecting underlying knowledge about narratives in the news.

In most circumstances, contrary to expectations, as the number of documents decreases, stability increases, with linear induction constituting the sole exception. There are two possible explanations for this: (1) the number of documents withheld in cross-validation decreases with the number of documents, and stability depends more on the number of documents that differ between cross-validations than on the fraction of documents that differ, or (2) a fixed number of schemas can better capture the information contained in a smaller set of documents because new documents add more novel and unique narratives, thereby making the content of the narratives more difficult to capture in a finite set. In other words, provided with more and more documents, the germinators do not converge to some finite set of schemas, but instead are presented with an increasingly difficult problem to solve. These two explanations are not completely orthogonal; however, the first is a far more mechanical explanation. It is simply that more word types are contained in a larger set of documents, and therefore will be subject to a larger Zipfian tail, which is more difficult for a finite set of recorded event verbs to cover. The second presupposes that the system has some semblance of understanding the language data and narrative, and the challenge comes from the increased diversity of narratives contained therein. The second explanation could cause the first,

but the second claim requires a much greater burden of evidence because of its deeper implications.

Unlike the other germinators, linear induction has no limit to the number of schemas it can produce, albeit with a great number of schemas containing single events. This means there are two possible explanations for its behavior. The first is the explanation originally hoped for—that as the algorithm is given more documents, it begins to converge on a stable core of knowledge derived from the source data that in one form or another represents a consistent understanding of the data. The second explanation is more mundane—that what's actually stable is the schemas containing single events, schemas which then bias the apparent stability upward as more events that do not cross the β threshold are observed in a greater number of documents.

The second explanation is more likely here. The linear induction schemas here generated 1,544,879 single event containing schemas, roughly 57.2% of the total schemas generated by linear induction. Additionally, the LI-Truncated stability provides some insight here. If there is a stable content core that linear induction is approaching, then the first 800 schemas generated should reflect that. What we see instead is the lowest stability scores across the board. However, note that as ablation 9 is approached, the stability of the LI-Truncated schemas increases, likely because the number of schemas actually produced by linear induction is approaching 800, so the effect of the truncation is vanishing.

While the arbitrary clipping of the linear induction schemas greatly decreases stability, this may point to the unexpected decrease in stability in the other germinators as the number of documents increases. Given their hard limit on the number of schemas used, they are in some sense performing a similar "clipping" of the content contained in the documents. However, their increased stability, while still conducting a form of clipping, could be attributed to performing a more informed clipping than the LI-Truncated schemas.

8 Conclusions

We have explored the stability of narrative schemas. We used both an ablation and cross-validation of the data to produce different sets of schemas and compared them using the Fuzzy Jaccard coefficient and the Jaccard Reciprocal Fraction, which produces a transformation of the Fuzzy Jaccard coefficient that is easier to interpret.

Our results affirmed the homogeneous-heterogeneous distinction found in Simonson & Davis (2016). The homogeneous categories produced more stable batches of schemas than the heterogeneous ones. The counter-training and linear induction germinators produced more stable results, but the linear induction truncated results indicate that much of the apparent stability of the linear induction germinator is contained in its long tail of schemas.

Additionally, contradicting expectations, the schemas produced were more stable when given fewer documents. It is difficult to say whether this is because fewer documents were removed at every step of the cross-validation or because there is simply fewer narratives to capture in the same number of schemas. These explanations do not necessarily contradict one another, and the effect witnessed may be a mixture of both.

Understanding the stability of schemas provides us with a window into the quantitative properties of news narratives at large. Additionally, stability provides another technique for evaluating what makes a "good set of schemas:" as objects that are consistent regardless of what subset of a corpus they are derived from. Not for all purposes is this property necessarily "good," but the stability procedure provides a quantitative metric for this property if it is desired.

For sake of comparison, further analysis, and potential use in other projects, we have also made our schemas publicly available for potential use in the future.[5]

Acknowledgements

We must thank the Georgetown University Department of Linguistics for its continued support, as well as Amir Zeldes and Nate Chambers for feedback on this work. We also would like to thank the reviewers, whose reviews exhibited a great deal of care and attention to detail.

[5] https://schemas.thedansimonson.com/

References

Balasubramanian, N., Solderland, S., Mausam, & Etzioni, O. (2013). Generating coherent event schemas at scale. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1721–1731). Association for Computational Linguistics.

Chambers, N., & Jurafsky, D. (2008). Unsupervised learning of narrative event chains. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)* (pp. 789–797). Association for Computational Linguistics.

Chambers, N., & Jurafsky, D. (2009). Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation for Natural Language Processing.* (pp. 602–610). Association for Computational Linguistics.

Cheng, P., & Erk, K. (2018). Implicit argument prediction with event knowledge.. Retrieved from `arXiv:1802.07226`

Corley, C., & Mihalcea, R. (2005). Measuring the semantic similarity of texts. In *Proceedings of the acl workshop on empirical modeling of semantic equivalence and entailment* (pp. 13–18).

de Marneffe, M., MacCartney, B., & Manning, C. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of the Language Resources and Evaluation Conference of the European Language Resources Association (LREC).* Association for Computational Linguistics.

Jans, B., Bethard, S., Vulić, I., & Moens, M. (2012). Skip n-grams and ranking functions for predicting script events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics* (pp. 336–344).

Kusner, M., Sun, Y., Kolkin, N., & Weinberger, K. (2015). From word embeddings to document distances. In *International conference on machine learning* (pp. 957–966).

Lee, H., Chang, A., Peirsman, Y., Chambers, N., Surdeanu, M., & Jurafsky, D. (2013). Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, *39*(4), 885–916.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., & McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (pp. 55–60).

Mooney, R., & DeJong, G. (1985). Learning schemata for natural language processing. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 681–687).

Mostafazadeh, N., Chambers, N., He, X., Parikh, D., Batra, D., Vanderwende, L., ... Allen, J. (2016). A corpus and evaluation framework for deeper understanding of commonsense stories. *arXiv preprint arXiv:1604.01696*.

Pichotta, K., & Mooney, R. J. (2014). Statistical script learning with multi-argument events. In *EACL* (Vol. 14, pp. 220–229).

Pichotta, K., & Mooney, R. J. (2015). Learning statistical scripts with lstm recurrent neural networks. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence.*

Rudinger, R., Rastogi, P., Ferraro, F., & Van Durme, B. (2015). Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP-15).*

Sandhaus, E. (2008). The New York Times annotated corpus. *Linguistic Data Consortium, Philadelphia*, *6*(12), e26752.

Schank, R., & Abelson, R. (1977). *Scripts, plans, goals and understanding: An inquiry into human knowledge structures.* New Jersey: Lawrence Erlbaum.

Simonson, D. (2018). *Investigations of the properties of narrative schemas* (Unpublished doctoral dissertation). Georgetown University.

Simonson, D., & Davis, A. (2015). Interactions between narrative schemas and document categories. In *Proceedings of the First Computing News Storylines Workshop (CnewS) at ACL-IJCNLP 2015* (p. 1). Association for Computational Linguistics and The Asian Federation of Natural Language Processing.

Simonson, D., & Davis, A. (2016). NASTEA: Investigating narrative schemas through annotated entities. In *Proceedings of the Second Workshop on Computing News Storylines (CNS 2016) at EMNLP 2016* (pp. 57–66). Austin, Texas: Association for Computational Linguistics. Retrieved from http://aclweb.org/anthology/W16-5707

Yangarber, R. (2003). Counter-training in discovery of semantic patterns. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1* (pp. 343–350). Stroudsburg, PA, USA: Association for Computational Linguistics. Retrieved from http://dx.doi.org/10.3115/1075096.1075140 doi: 10.3115/1075096.1075140

NIPS Conversational Intelligence Challenge 2017 Winner System: Skill-based Conversational Agent with Supervised Dialog Manager

Idris Yusupov [1] and **Yurii Kuratov** [1, 2]
[1] Department of Computational Linguistics,
Moscow Institute of Physics and Technology

[2] Neural Systems and Deep Learning Lab,
Moscow Institute of Physics and Technology
{i.yusupov, yurii.kuratov}@phystech.edu

Abstract

We present *bot#1337*: a dialog system developed for the 1st NIPS Conversational Intelligence Challenge 2017 (ConvAI). The aim of the competition was to implement a bot capable of conversing with humans based on a given passage of text. To enable conversation, we implemented a set of skills for our bot, including chit-chat, topic detection, text summarization, question answering and question generation. The system has been trained in a supervised setting using a dialogue manager to select an appropriate skill for generating a response. The latter allows a developer to focus on the skill implementation rather than the finite state machine based dialog manager. The proposed system bot#1337 won the competition with an average dialogue quality score of 2.78 out of 5 given by human evaluators. Source code and trained models for the bot#1337 are available on GitHub.

1 Introduction

A conversational or a dialogue agent is a system that interacts with a human via voice or text messages. Dialog systems can be task-oriented, i.e. supposed to solve specific tasks such as reserving flight tickets, or general purpose. Another way to differentiate dialog systems is to compare them by domain: closed or open one. When open-domain dialog systems cover a wide range of supported topics, closed-domain are usually specialized on a few topics.

The early dialog systems used a rule-based approach to control dialog flow (Weizenbaum, 1966; Wallace, 2009). Rule-based systems are hard to maintain and new rules should be created for each new domain. This approach is still widely used in cases where the full control of dialog is crucial (e.g., production systems). The availability of dialog datasets (DSTC (Williams et al., 2016), Twitter Corpus (Ritter et al., 2010), Ubuntu Dialogue Corpus (Lowe et al., 2015)) makes it possible to train end-to-end dialog systems (Sordoni et al., 2015; Vinyals and Le, 2015). End-to-end systems are usually based on recurrent neural networks (Graves, 2013) and sequence-to-sequence models (Sutskever et al., 2014), and use raw dialogues as a training data.

One of the main challenges in the field of open-domain dialogue systems is their evaluation. The reason for that is the absence of good evaluation metrics. While goal-oriented systems can be evaluated with the percentage of dialogues when user's task was accomplished, there is no formal measure of the dialogue quality for open-domain systems. The goal of such systems is to generate responses which suits the context. The quality of the response can be measured by its perplexity measure (Serban et al., 2015) which, however, can not assess the adequacy of an answer. In order to measure it researchers often use metrics that compare a generated string to some oracle answer, e.g. BLEU (Papineni et al., 2002) and METEOR (Lavie and Agarwal, 2007) originally used for evaluating Machine Translation model. This is not an optimal way of evaluating chatbots either, because a relevant answer can be different from an oracle (Liu et al., 2016). In recent works on the evaluation of dialogue systems the authors suggest training evaluation metrics on a set of human-labelled dialogues (Lowe et al., 2016; Lowe et al., 2017).

Proceedings of the 27th International Conference on Computational Linguistics, pages 3681–3692
Santa Fe, New Mexico, USA, August 20-26, 2018.

However, obtaining such corpora takes much time and effort, so those metrics are rarely used up to our knowledge.

A competition of open-domain dialogue systems is one way to organize a large-scale evaluation by humans. There are some competitions in the field such as Loebner Prize annual competition (Mauldin, 1994) and Alexa Prize Competition (Ram et al., 2018). The First Conversational Intelligence Challenge was organized as a part of NIPS conference in 2017. The task was to build a conversational agent, which discusses a given text with a human.

A typical competition bot is equipped with a set of skills, such as greeting, asking a question on a certain topic, etc., and decides when to use a specific one. The main challenge here is to understand when to use a specific skill[1], in other words, to develop a dialog manager (DM). The finite state machine (FSM) can be seen as an instrument of the first choice in rule-based dialog managers. The main advantage of the FSM is a deterministic and transparent control of the conversation, but its complexity quickly grows for large systems. Updating of large FSM with new rules is usually hard due to the high volume of required conflict resolutions between parts of the system. This results in the lack of scalability for the FSM approach.

2017 Loebner Prize winner bot Mitsuku[2] uses a rule-based approach based on Artificial Intelligence Markup Language (AIML) (Wallace, 2009). 2017 Alexa Prize participant MILABOT conversational agent (Serban et al., 2017) uses a combination of deep learning, reinforcement learning and rules. MILABOT agent team used a lot of human-annotated data collected from Amazon Mechanical Turk to train the dialog manager. Mitsuku requires a lot of rules, so it is hardly scalable, while MILABOT requires a lot of human-annotated data. In addition those bots do not initiate a dialog with the user, but only give replies to a user.

Instead, we implemented a supervised DM for our bot. It takes the dialog context[3] as an input and outputs the label of the skill to be used. This simplifies developing the proposed DM in comparison with the FSM and allows to focus on the skill development.

Next sections describe our conversational agent *bot#1337* that won the NIPS Conversational Intelligence Challenge. Section 2 describes the implemented skills and Section 3 describes Dialogue Manager (DM). Section 4 gives a high-level view on how skills, DM and user interact together. Section 5 provides detailed information about the participation in the NIPS Conversational Intelligence Challenge, including comparison with other winning system, analysis of skills usage frequency, dialog manager performance and dialogs reading insights. Information about the bot#1337 license, source code and demonstration given in Section 7. To show how the bot works in practice, we have included a sample dialog between the bot and user in the Appendix A.

2 Skills

Discussion of the text require different skills. First of all, a bot should be able to greet the user, to answer and to ask questions about the text. Secondly, text summarization skill may be required, because the user may not read a long paragraph. Often one just wants to chit-chat with a bot without a thorough discussion.

Chit-chat skills Chit-chat skills are required to discuss some common topics, which are often not connected with a given text. We have built three following skills for the chit-chat.

OpenSubtitles and Facebook news seq2seq chit-chat skills use a neural machine translation attention-based sequence-to-sequence models. They are trained using OpenNMT framework. OpenNMT is a generic deep learning framework mainly specialized in sequence-to-sequence models covering a variety of tasks such as machine translation, summarization, image to text, and speech recognition. The framework has also been extended for other non sequence-to-sequence tasks like language modelling and sequence tagging (Klein et al., 2017).

[1] We defined *skill* as a model that takes text and dialog context as an input and outputs a response.

[2] `http://www.mitsuku.com`

[3] We defined *dialog context* as the concatenation of dialog history and the current user utterance.

Training performed with word-level representations and ADAM optimizer. Word embeddings dimension was set to 150 and their weights are updated during the training.

OpenSubtitles seq2seq skill uses the dialog context as an input and generates an utterance as an output. The model is a 2-layer encoder-decoder LSTM with 2048 hidden units. It is trained on the OpenSubtitles dataset (Tiedemann, 2009).

The Facebook news seq2seq skill, in addition to the dialog context, may use a paragraph as an input. Paragraph and dialog context divided with special "EOP" (end of paragraph) token. It also generates an utterance as an output. The model is a one-layer encoder-decoder LSTM with 1024 hidden units. It is trained on the Facebook news dataset[4], which consists of posts and comments from Facebook.

The Alice (Wallace, 2009) *chit-chat skill* uses the dialog context as an input and generates an utterance as an output. It is built upon AIML rules. We used the open source version of Alice from GitHub[5].

Sequence-to-sequence chit-chat skills generate several response candidates. Often generative models may generate short responses or responses with many identical or undesirable words. These candidate responses are processed by a filtering algorithm (Alg. 1) and if there is no candidate responses left after filtering, then the Alice skill is executed or one of the escape plan template responses is selected. Escape plan responses are a set of pre-defined utterances, such as "Do you like this text?", "What do you think about the competition?".

Algorithm 1: Filtering candidate responses

Data: candidate responses
Result: filtered responses
remove duplicate responses;
remove short responses;
remove responses with majority of identical words;
remove responses with majority of stopwords;
remove responses with undesirable words;
return filtered responses;

Q&A skills Discussion of text often consists of asking and answering questions. We developed three skills to generate questions and to answer the questions about the text.

For the question-asking skill we reproduced the feature-rich sequence-to-sequence with attention model (Zhou et al., 2017). The model is a 2-layer encoder-decoder GRU with 512 hidden units. It is trained on the SQuAD dataset (Rajpurkar et al., 2016) using word-level representations and the OpenNMT framework with ADAM optimizer. Word embeddings dimension was set to 300. Lexical and answer features embedded to 32-dimensional vectors. This skill takes a paragraph, named entities and lexical features extracted using Stanford CoreNLP (Manning et al., 2014) as an input and outputs a question. In the training phase, all the SQuAD answers were used to generate questions. During the inference the model used named entities extracted from the text by the Stanford CoreNLP. We use these named entities as an answers and to generate questions.

The answer-checking skill is connected with the question-asking skill. Right answer on each question is prepared by question-asking skill described above. This skill checks the user's answer using fuzzy matching algorithms (e.g. based on Levenshtein distance). Afterwards, it uses a template to generate a response to the user, such as "You can do better! Hint: first 3 letters is *goo*".

The question-answering skill uses the bidirectional attention flow (BiDAF) model (Seo et al., 2016). It takes a user's question and the relevant text as an input and outputs an answer. Finally, these answer were added to template phrases, such as "I think the answer is ...". We used the open source version of BiDAF model from GitHub[6].

[4]https://github.com/jbencina/facebook-news
[5]https://github.com/sld/convai-bot-1337/tree/master/ALICEChatAPI
[6]https://github.com/allenai/bi-att-flow

Summarization skill Usually, a text passage to discuss is a long (5-10 sentences) read. Text summarization skill helps to save time and to engage the user.

We used a sequence-to-sequence model with attention pretrained on a Gigaword dataset with OpenNMT[7]. The model is a 2-layer encoder-decoder LSTM with 500 hidden units. Word embeddings dimension was set to 500. We applied the model to chunks of a provided text to generate possible summaries. Finally, these summaries were added to template phrases, such as "Maybe this article's main idea is ...". To select the best response, we used Alg. 1.

Topic detection skill Mentioning the main topic of the text may engage the user in a conversation. We analyzed data from the first round of the NIPS Conversational Intelligence Challenge and it proved our hypothesis.

The topic detection skill is built upon the BigARTM topic modeling framework. BigARTM is a tool to infer topic models, based on a technique called Additive Regularization of Topic Models. This technique effectively builds multi-objective models by adding the weighted sums of regularizers to the optimization criterion (Vorontsov et al., 2015).

The model takes text as an input and outputs its possible topics. Then, detected topic names are added to template phrases, such as "Am I right that topic of the text is ... ?". We used the Wikipedia corpora from the BigARTM website datasets sections[8]. The model was trained[9] to predict 15 topics. Topic names such as "Politics", "Culture", etc., were set up manually by analyzing top-tokens.

Additional skills We implemented two skills that are independent from the input. They both have handwritten sets of phrases, which are randomly selected to be sent to the user.

The greeting skill is used in the beginning of the conversation if the user does not say anything after some time. It includes such phrases as "Well hello there!", "Hiya!", etc.

The common phrases skill is used when the user does not say anything for some time or when the used skill output is empty. It includes such phrases as "What do you think about ConvAI competition?", "Do you like this text?", etc.

3 Dialog manager

The DM (Fig. 1) runs two classifiers on user utterance (and, possibly, dialog context) to detect what skill should produce an answer.

The first classifier works on a small supervised set (few key phrases for each skill) and is based on the mean word embeddings (we used GloVe embeddings[10]) and the k-nearest neighbors classifier (k-NN).

The second classifier is based on a large training corpora and we used the fastText library for this case. FastText is a library for efficient text classification and representation learning (Joulin et al., 2016). The fastText supervised model was trained[11] to predict 5 classes (Fig. 1).

All implemented skills can be one of two types: with or without a training dialog dataset. For chit-chat and question-answering skills we have datasets with utterances, questions, and answers. It allows us to train the classifier to select which dataset (and it infers which skill) is more suitable for the incoming user utterance. Summarization, topic detection, and the ask-question skills do not have good common known dialog datasets, which provide utterances that should activate these skills. For these skills we wrote few key phrases (3-10 for each skill) by our own and used them with k-NN classifier based on mean word embeddings.

For example, we have 2 skills - Open Subtitles and Topic Detection. First skill classifier is trained with fastText, because it has many dialog utterances (e.g. "They still behind us?", "Senora, give me a break!" and many others). Second one does not have any dialog utterances, so we should add a few of

[7] http://opennmt.net/Models/#english-summarization

[8] http://docs.bigartm.org/en/stable/tutorials/datasets.html

[9] Training script available at https://github.com/sld/convai-bot-1337/blob/master/topic-modelling

[10] https://nlp.stanford.edu/projects/glove/

[11] Training script available at https://github.com/sld/convai-bot-1337/tree/master/classifiers/factoid_question_vs_all

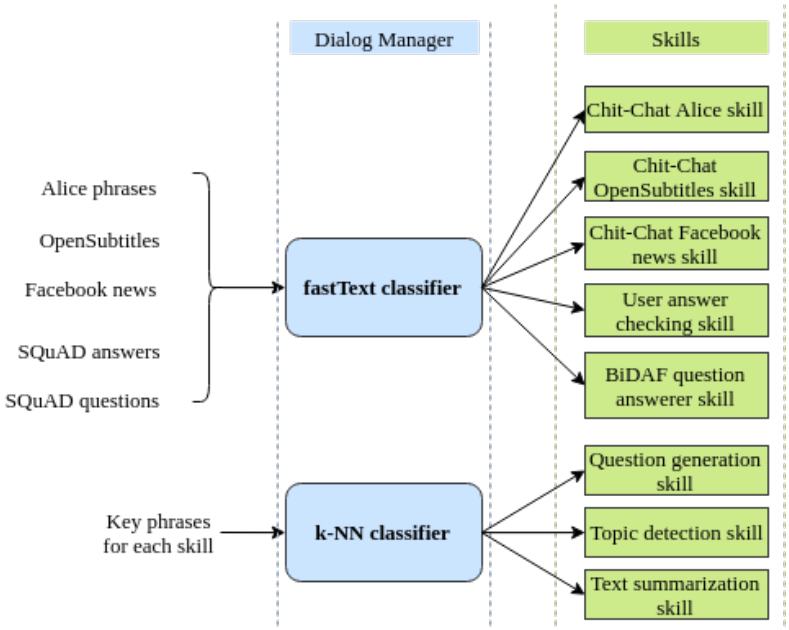


Figure 1: Dialog manager: classifiers, datasets and skills corresponding to them.

them ourselves to train a k-NN classifier (e.g. "What is the theme of text?", "Say me theme", "This text main topics").

Skills without dialog datasets have a higher priority; if the k-NN classifier prediction confidence level exceeds the threshold value (90%), then the skill selected by this classifier is used to generate the response. Otherwise, the skill is selected by the fastText classifier.

4 Dialog system and user interaction flow

Dialog system and user interaction flow (Fig. 2) shows how user interacts with a dialog system.

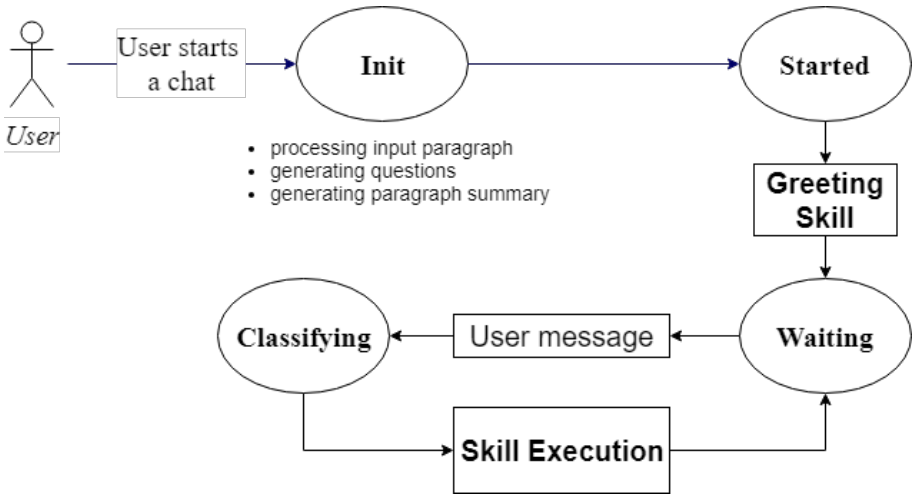


Figure 2: Dialog system and user interaction flow.

The dialog agent initialization begins when a user starts a chat. At the **init state**, the DM and skills are initialized. Some skills, such as question generation and text summarization, process input paragraphs and generate responses for further usage. Thus, the DM can choose from these responses to send a message to the user.

After that, the dialog agent switches to the **started** state. A user or a bot can start the conversation.

Thus, at this state, the bot waits for the first user message or executes the greeting skill if there is no message from the user after some time.

Answering to user message is the most important task to handle. When receiving a message from the user, the **classifying** state activates. At this state, the DM decides which skill to use. The DM runs two classifiers on user utterance (and, possibly, dialog context) to decide which skill should produce an answer. After that, the bot executes the skill and sends a generated message to the user.

After the skill execution, the bot goes to the **waiting** state. This state is about engaging the user in conversation when he or she is not talking to the bot. If after some time there is no message from the user, the bot tries to motivate the user to talk. It executes a random skill, such as question asking, and sends a message to the user. If there is no reaction from the user for a long time, then the bot ends the conversation.

5 Participation in the Conversational Intelligence Challenge

The main task of the NIPS Conversational Intelligence Challenge 2017 (ConvAI[12]) was to build a conversational agent, which discusses a given text with a human. Each dialog was evaluated by a human at the end of the conversation on a scale from 1 (bad) to 5 (excellent) in three dimensions: quality, breadth, and engagement. Each bot utterance could be also marked as appropriate or not. The first round dataset (2778 dialogues) was released at the end of the first human evaluation round[13].

The early version of our bot had a chit-chat, Q&A skills and a dialog manager based on rules and fastText classifier. This approach took the second place in the first round of the Conversational Intelligence Challenge with an average quality score 2.31 out of 5. A winner bot took 2.38 out of 5, while human 3.8 out of 5. The dataset released after this round provided insights on how to improve our conversational agent and helped to detect undesirable behavior. Particularly we decided to add a chit-chat Alice, paragraph's topic mention to the greeting and text summarization skills. Newly added skills required new logic to support. We enhanced the dialog manager by adding k-NN classifier for skills without dialog data described in Section 3. It helped us to avoid using rules and simplified conversational agent's development. As a result of introducing the above-mentioned changes, our bot won the Conversational Intelligence Challenge Finals (Table 1).

Bot name	Quality (average human evaluation score)	# dialogues
bot#1337	2.779	68
poetwannabe	2.623	53
RLLChatBot	2.462	13
kAIb	2.2	35
PolyU	1.75	28
DeepTalkHawk	1.262	42

Table 1: Results of the Conversational Intelligence Challenge Finals.

5.1 Comparison with *poetwannabe*

Poetwannabe is the other winning system from University of Wroclaw which also used a multi-skill architecture, but focused on extracting knowledge from external resources (Wikipedia, DBPedia, Wikiqoute). Instead of supervised dialog manager to control dialog flow they used confidence scores from each skill, based on linguistic features, dialog history and internal state (Chorowski et al., 2018).

5.2 ConvAI finals dataset analysis

To understand better bot's issues and to develop further its improvement strategy, we analyzed the frequencies of skills usage, dialog manager performance and user dialogues of the Conversational Intelligence Challenge finals dataset. The dataset was provided by the competition organizers. We have

[12] http://convai.io/2017/

[13] http://convai.io/2017/data/

manually labeled 68 dialogs with our bot. Each bot's utterance in the dialog was tagged with predicted skill and true skill labels. Predicted skill label is the one selected by the supervised dialog manager. True skill label is the one that we found to be the most appropriate for the dialog context.

5.2.1 Frequency of skills usage

Frequency of skills usage was calculated by predicted skill label grouped by a dialog score for each dialog (Table 2). After analyzing the frequency of skills usage we saw that chit-chat skills should be carefully implemented. Chit-chat skills were most frequently selected by the dialogue manager. Skills like topic detection or summarization are used rarely because usually they are executed only one time during the dialog. It does not imply that summarization or topic detection skills are unnecessary. They provide good user experience in a conversation. A fragment of best-scored dialog shows a positive user reaction to a summarization skill:

User: What do you think about this text?

Bot: I suppose that this text main idea is: New rules on investment?

User: Maybe you are right

Other fragment with worst-scored dialog shows negative reaction for chit-chat skills:

Bot: What country borders the south of guinea-bissau?

User: I don't know. Can you tell me?

Bot: I don't know.

User: Senegal is to the north, isn't it?

Bot: I will not.

Dialog score	Chit-chat skills, %	Q&A skills, %	Summarization, %	Topic Detection, %	Greeting, %	Common Phrases, %
5	71.4	16.3	0.7	3.2	3.9	4.6
4	56.2	23.3	2.7	6.8	8.2	2.7
3	57	21	1.1	5.9	7.5	7.5
2	55.3	24.9	2.3	6.8	7.6	3
1	64.3	15.6	1.7	4.5	6.7	7.2

Table 2: Bot#1337's percentage of skills usage in the ConvAI Finals dialogs. They are grouped by a dialog score (human evaluation score). We count how many times each type of skill was invoked in each dialog. Chit-chat skill is the one of OpenSubtitles, Facebook news and Alice skills. Q&A is the one of question-asking, answer-checking and question-answering skills.

5.2.2 Dialog manager performance

By using true skill and predicted skill labels, F1 measure was calculated. Topic detection, greeting, common phrases skills were not included to the calculations because they were executed in a waiting state almost every time (described in Section 4) and dialog manager was not used here.

Skill Name	Precision	Recall	F1	Total utterances count
OpenSubtitles chit-chat	0.79	0.84	0.81	249
Answer-checking	0.82	0.70	0.75	57
Alice chit-chat	0.81	0.68	0.74	152
Facebook chit-chat	0.63	0.85	0.73	53
Question-asking	1.00	0.56	0.71	9
Summarization	0.67	0.57	0.62	14
Question-answering	0.56	0.60	0.58	25
Average	*0.77*	*0.76*	*0.76*	*559*

Table 3: Dialog manager performance

Bot#1337's supervised dialog manager (described in Section 3) average F1 equals to 0.76. Question-asking and summarization skills that were selected by the k-NN classifier show lower F1 score than skills selected by the fastText classifier (chit-chat and answer-checking skills). F1 for these skills can be improved by adding more examples to the k-NN classifier. However this approach does not apply to a question-answering skill. It is selected by the fastText classifier and has the lowest F1 in the Table 3. The error analysis showed that the user often asks complex questions that are not covered by the training set from SQuAD. For example:

> **User:** Let's talk about Guinea-Bissau then. What do you know about it?" [classified as Open Subtitles]

But even with the perfect F1 score, analysis of how each skill works in a dialog is necessary. Dialog reading helps to analyze skills errors, to find new skills that should be developed and to find the direction of bot's improvement.

5.2.3 Dialogs reading insights

After reading 68 dialogs with our bot from the finals, we can conclude that having diverse and high quality chit-chat skills and engaging a user into talking when he or she is not responding for some time (waiting state described in the Section 4) helped the bot to make a conversation better.

Other insights that will be able to help to improve our bot in the future described below.

The bot starts to talk fast and a user complains to it.

> **Bot:** Yo! [Greeting skill]
>
> **Bot:** I think this text is connected with nationalities. I can ask you a question about it. [Topic detection skill]
>
> **User:** Wait, I am still reading it
>
> **Bot:** Why is this news? [OpenSubtitles chit-chat skill]

It would be better to have a separate skill to detect if a user still reading the text and reply to the user with "Okay, I am waiting" message.

A user asks the bot about the competition, but it does not have the skill to response.

> **User:** What are my instructions for today?
>
> **Bot:** I'm not sure. [OpenSubtitles chit-chat skill]
>
> **User:** Do you something about CONVAI?
>
> **Bot:** No, i'm not. [OpenSubtitles chit-chat skill]

Lack of semantic model of the text does not allow the bot to talk about the facts in the text.

> **User:** What kind of sport is this text about?
>
> **Bot:** I suppose that this article can be described as: Win club world cup..? [Text Summarization skill]
>
> **User:** But what sport? Football, hockey, basketball?
>
> **Bot:** I'd like to say that correct answer is: barcelona accomplished two new records in spanish [Question-answering skill]
>
> **User:** I want to know more about Barcelona
>
> **Bot:** Tell me about your likes and dislikes? [Alice Chit-chat skill]
>
> **User:** Ok, is this article about football?
>
> **Bot:** I think this article main idea is: Season begins. [Text Summarization skill]

We have the Q&A skills, but they do not cover cases like in the dialog above. A high quality conversation about the facts in text requires using knowledge bases like DBpedia. Besides it is worth to use some Web APIs (e.g. BabelNet knowledge base at `https://babelnet.org`), but rules of the ConvAI competition do not allow to do it. According to the rules everything should work without any connection to external services.

Sometimes the bot is not polite.

User: It was basically my answer
Bot: February was basically his answer? very interesting. [Alice Chit-chat skill]
User: What did you base my gender on?
Bot: Nothing. [OpenSubtitles chit-chat skill]
User: Then why "his answer"? And why talk to me in the third person?
Bot: Oh darn, I broke you. [OpenSubtitles chit-chat skill]

Answer-checking skill and question-answering skill should be more intelligent. Answer-checking skill sees the difference between a number and a word representing that number. But it should not and dialog below describes the case:

Bot: How many mathematical treatises have managed to survive? [Question-Asking skill]
User: 3?
Bot: It is incorrect:) Hint: first 3 letters is thr. Try again [Answer-checking skill]
User: 3
Bot: No [Answer-checking skill]
Bot: I think that answer is: three [Answer-checking skill]
User: Infact I did mentioned the same
Bot: How old are you? [OpenSubtitles chit-chat skill]

Sometimes users ask the bot to answer the question asked by the bot in the first place. However, the bot can't answer if the user uses anaphora constructions.

User: Ask me question
Bot: How many years did nbc universal sell in 2006? [Question-Asking skill]
User: I dont know
Bot: Why wouldn't they have taken the meeting? [OpenSubtitles chit-chat skill]
User: Do you know?
Bot: I dont know what you're doing. [OpenSubtitles chit-chat skill]

User tries to teach the bot, but the bot does not understand it.

User: What was article about?
Bot: Maybe this text can be described as: Father and grandfather born in hungary..? [Text Summarization skill]
User: No! It was about von Neumann
User: Which city lies on seven hills?
Bot: What a joke. [OpenSubtitles chit-chat skill]
User: Answer me
Bot: Try asking me again using simpler words. [Alice chit-chat skill]
User: Which city lies on seven hills. Answer: Rome.
Bot: Why is this news? [Facebook news chit-chat skill]

Bot's conversation style is the same. It may be worth to make different conversation styles to engage the users who have already talked to a bot.

6 Conclusion

In this paper, we have described the conversational agent *bot#1337*. This bot took the first place in the NIPS Conversational Intelligence Challenge 2017 with an average dialogue quality score of 2.78 out of 5. The supervised dialog manager allowed us to focus on the skills implementation, to avoid rules and to simplify the development of a bot. Skills like topic detection, Q&A, text summarization are engaging a user into a conversation about the given text. According to the conversational data received during the competition, a user often chit-chats with a bot. AIML-based Alice chit-chat skill covers a wide range of topics, while sequence-to-sequence based chit-chat skill trained on Open Subtitles dataset makes a conversation more enjoyable. Open source machine learning software like OpenNMT, BigARTM,

fastText gives us a big boost on the conversational agent's implementation. By making the framework for *bot#1337* publicly available, we gave an opportunity for the community to get acquainted with its various modules and to develop further the conversational intelligence.

7 Availability

Conversational agent demonstration is accessible as a Telegram[14] bot: `https://t.me/ConvAI1337Bot`. Also we have public JSON API that documented at `https://github.com/sld/convai-bot-1337/wiki/Api-Documentation`. The source code is released under GNU GPLv3 license and available on GitHub: `https://github.com/sld/convai-bot-1337`.

Acknowledgements

We thank Mikhail Burtsev, Luiza Sayfullina and Mikhail Pavlov for comments that greatly improved the manuscript. We would also like to thank the Reason8.ai company for providing computational resources and grant for NIPS 2017 ticket. We thank Neural Systems and Deep Learning Lab of MIPT for ideas and support.

References

Jan Chorowski, Adrian Łańcucki, Szymon Malik, Maciej Pawlikowski, Paweł Rychlikowski, and Paweł Zykowski. 2018. A Talker Ensemble: the University of Wroclaw's Entry to the NIPS 2017 Conversational Intelligence Challenge. *arXiv preprint arXiv:1805.08032*.

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

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 228–231, Stroudsburg, PA, USA.

Chia-Wei Liu, Ryan Lowe, Iulian Vlad Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *CoRR*, abs/1603.08023.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*.

Ryan Lowe, Iulian Vlad Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. On the evaluation of dialogue systems with next utterance classification. *CoRR*, abs/1605.05414.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1116–1126, Vancouver, Canada, July.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Michael L Mauldin. 1994. Chatterbots, tinymuds, and the turing test: Entering the loebner prize competition. In *AAAI*, volume 94, pages 16–21.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA.

[14] `https://telegram.org/`

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, et al. 2018. Conversational AI: The Science Behind the Alexa Prize. *arXiv preprint arXiv:1801.03604*.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180. Association for Computational Linguistics.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2015. Hierarchical Neural Network Generative Models for Movie Dialogues. *CoRR*, abs/1507.04808.

Iulian V Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, et al. 2017. A deep reinforcement learning chatbot. *arXiv preprint arXiv:1709.02349*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*.

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

Jörg Tiedemann. 2009. News from OPUS-A collection of multilingual parallel corpora with tools and interfaces. In *Recent advances in natural language processing*, volume 5, pages 237–248.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Konstantin Vorontsov, Oleksandr Frei, Murat Apishev, Peter Romov, and Marina Dudarenko. 2015. Bigartm: Open source library for regularized multimodal topic modeling of large collections. In *International Conference on Analysis of Images, Social Networks and Texts*, pages 370–381. Springer.

Richard S Wallace. 2009. The anatomy of ALICE. In *Parsing the Turing Test*, pages 181–210. Springer.

Joseph Weizenbaum. 1966. ELIZA - a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Jason Williams, Antoine Raux, and Matthew Henderson. 2016. The dialog state tracking challenge series: A review. *Dialogue & Discourse*, 7(3):4–33.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural Question Generation from Text: A Preliminary Study. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 662–671. Springer.

Appendices

A Appendix: Dialog example

Text

In 2015-2016, Notre Dame ranked 18th overall among "national universities" in the United States in U.S. News & World Report's Best Colleges 2016. In 2014, USA Today ranked Notre Dame 10th overall for American universities based on data from College Factual. Forbes.com's America's Best Colleges ranks Notre Dame 13th among colleges in the United States in 2015 ... According to payscale.com, undergraduate alumni of University of Notre Dame have a mid-career median salary $110,000, making it the 24th highest among colleges and universities in the United States. The median starting salary of $55,300 ranked 58th in the same peer group.

Bot - Greeting skill: Hiya!
Bot - Greeting skill, Topic detection skill: Let's talk about culture, or I can ask you a question about this text.
User: Ask me then :)
Bot - Question asking skill: What is the median salary in notre dame?
User: 55,300
Bot - Answer checking skill: Show me your best! Hint: first 3 letters is $1. Try again.
User: What is the rank of Notre Dame?
Bot - Question answering skill: It seems that correct answer is: "18th overall among national universities in the united states in u.s. ..."
User: Nice!
Bot - Facebook news chit-chat skill: This is what happens when you don't want to be in the white house.
User: Very funny :) This is very long text to read, can you summarize it in couple sentences?
Bot - Text summarization skill: I suppose that this article main idea is: University of notre dame's salary.
User: Hope you are not lying to me
Bot - Facebook news chit-chat skill: This is what happens when you vote for a democrat.
User: You are so into politics
Bot - Alice chit-chat skill: I will tell my the kind people in the lab you said i am into politics.
User: I have to go! Bye!

AMR Beyond the Sentence: the Multi-sentence AMR corpus

Tim O'Gorman[1], Michael Regan[3], Kira Griffitt[2], Ulf Hermjakob[4], Kevin Knight[4], Martha Palmer[1]
[1]University of Colorado Boulder
{ogormant,mpalmer}@colorado.edu
[2]Linguistic Data Consortium
kiragrif@ldc.upenn.edu
[3]University of New Mexico
reganman@unm.edu
[4]Information Sciences Institute, University of Southern California
{ulf,knight}@isi.edu

Abstract

There are few corpora that endeavor to represent the semantic content of entire documents. We present a corpus that accomplishes one way of capturing document level semantics, by annotating coreference, implicit roles and bridging relations on top of gold Abstract Meaning Representations of sentence-level semantics. We present the methodology of developing this corpus, alongside analysis of its quality and a plausible baseline for comparison. It is hoped that this Multi-Sentence AMR corpus (MS-AMR) illustrates a feasible approach to developing rich representations of document meaning, useful for tasks such as information extraction and question answering.

1 Introduction

Although Abstract Meaning Representation (AMR) (Banarescu et al., 2013) shows promise for a range of tasks such as summarization (Liu et al., 2015; Viet et al., 2017) and information extraction (Garg et al., 2016), it is restricted to capturing the semantics of individual sentences. For many purposes, when examining the semantics of a document, one also needs access to cross-sentence information such as coreference.

We suggest that the AMR approach to semantic representation has useful characteristics for an extension to discourse-level representations. AMR represents sentence meaning in a simple, readable semantic graph, such that annotators may directly mark coreference relations upon the AMR graph itself. AMR also annotates implicit roles in some within-sentence contexts, and the PropBank predicate annotations provide a resource for extending those implicit roles annotations to a document level.

The Multi-Sentence Abstract Meaning Representation (MS-AMR) corpus is a corpus annotated on top of existing gold AMRs, extending them with this additional information. By linking those AMRs together, it presents an integrated representation of the meaning of an entire document or discourse, as the addition of the coreference, implicit role reference and bridging relations across each AMR helps to build a larger representation of the entire propositional content of the document. Because these MS-AMR representations are annotated directly onto the variables within an AMR semantic representation, it is also a different task from traditional coreference, event coreference or implicit role coreference tasks, and results in a fundamentally different kind of data. We present a baseline system and inter-annotator agreement scores, which we hope will illuminate the nature and quality of the dataset, and outline methods for how to score MS-AMR system outputs.

2 Annotation Methodology

2.1 Background: Within-sentence Abstract Meaning Representation

AMRs are directed acyclic graph representations of sentence meaning (Banarescu et al., 2013), designed to capture the important meaning elements of a sentence while abstracting away from syntactic details

Proceedings of the 27th International Conference on Computational Linguistics, pages 3693–3702
Santa Fe, New Mexico, USA, August 20-26, 2018.

Bill left for Paris
(l / leave-11
:ARG0 (p / person :wiki - :name (n / name :op1 “Bill”))
:ARG2 (c / city :wiki “Paris” :name (n / name :op1 “Paris”)
He arrived at noon
(a / arrive-01
:ARG1 (h / he)
:ARG3 (*i / implicit role: start point*)
:ARG4 (i2 / implicit role: end point; destination)
:TIME (d / date-entity :dayperiod (n3 / noon)))

Figure 1: Example of MS-AMR annotation; annotators link coreferent variables (such as marking a relation between between *p* and *h* (in red)) and implicit roles, here linking the destination (arg4) in the second sentence to the previous variable *c* (in blue)

of how that meaning was expressed. AMRs capture basic representations of semantic roles (using the numbered arguments of PropBank (Palmer et al., 2005)), as well as within-sentence coreference, named entity and entity linking information. Thus, a simple sentence such as “Bill left for Paris” can be represented as in the first AMR in Figure 1, with “leave-11” denoting the physical departure sense of “leave”, numbered arguments such as “arg2” denoting semantic roles that are unique to that sense (e.g. for leave-11, arg2 is the destination), and “Bill” and “Paris” both having named entity labels (person and city) as well as links to Wikipedia when possible. One can therefore think of an AMR as a graph of the meaning of a sentence, with “variables” such as *l*, *p* and *c* being nodes in that graph, and “/” denoting an “instance-of” relation, showing that *p* is a thing of type *person* and that *c* is a thing of type *city*.

2.2 Annotating Multi-sentence Coreference Over AMRs

Annotating multi-sentence AMR has fundamentally different characteristics from coreference annotation over surface forms. Annotators start with gold AMR annotations (using the upcoming AMR public release), and add coreference relations (and related annotations such as bridging and implicit roles) as a layer on top of those gold AMRs. Therefore, rather than annotating spans, annotators label clusters of variables that refer to the same entity or event over the AMRs constituting a document. Figure 1 illustrates a basic example of coreference in MS-AMR, showing both coreference and an example of implicit role coreference.

This annotation is structured so that alongside the annotation of explicit coreference information, annotators can also capture implicit role information. Because within-sentence AMR has all predicates annotated with PropBank senses, we have access to the lexicon with a list of all numbered arguments we might expect for that predicate. We can therefore produce a list of the numbered arguments that are not explicitly filled in the text and show these unfilled roles to annotators. These unfilled roles are temporarily added during annotation – as is illustrated with the arg3 and arg4 of the predicate “arrive-01” in Figure 1 – and can be added to coreference clusters just like any other variable. This is similar to prior works in implicit role annotation (Gerber and Chai, 2010) in that we are using semantic role inventories to prompt annotators with possible implicit roles, while adding the innovation of fitting this within a coreference task. However, while previous annotations prompted annotators with an implicit role and asked them to look through prior text for its referents, this annotation fits implicit roles into the task of coreference labeling. An example of the actual act of annotation with the Anafora toolkit(Chen and Styler, 2013) for these additional implicit role options can be seen in Figure 2.

This annotation also labels some examples of “bridging” coreference relations (Clark, 1977; Poesio et al., 1997). We annotate two more common bridging relations, part/whole relations (as in example 1) and set/member relations (as in example 2), with a focus upon those between named entities and common nouns.

1. *I think this shows that pretty much every President can do any design thing they want with both* **the**

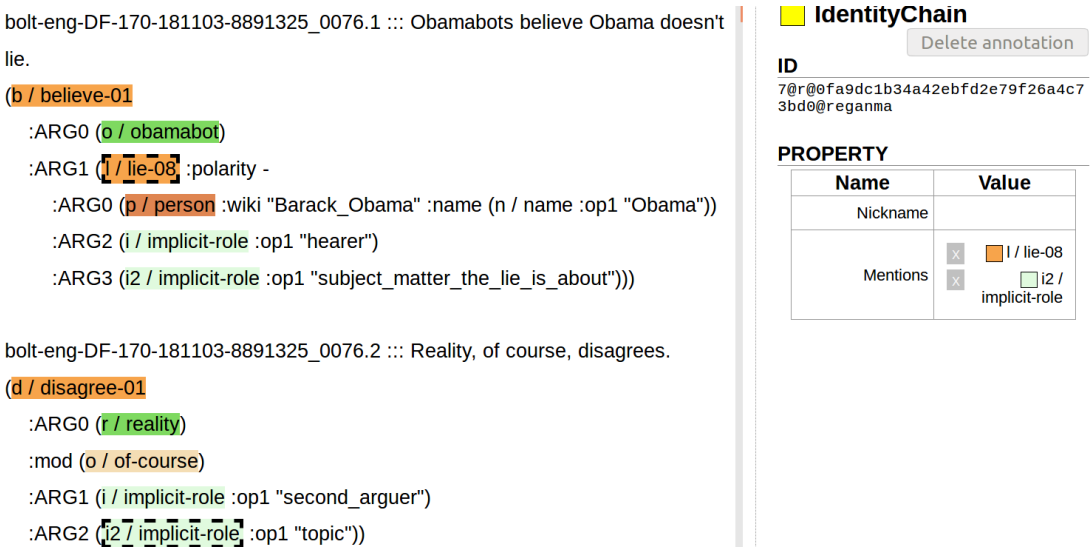


Figure 2: Annotation interface, illustrating implicit role links. Annotators click on boxes within the AMR (left) to add them to coreference chains (full chains shown on the right), as with the link between the implicit topic ("i2") and the earlier "l / lie" mention.

Residence *and* **the Office Wings** (both given part/whole relation to "the White House")

2. *I also liked* **"Deception point"**. *So I have read* **all four of his books** *and enjoyed them.* (set/member relation)

Those examples illustrate the emphasis upon capturing part/whole and set/member relations that require contextual understanding; annotators were instructed not to link part/whole relations that are only knowable through world knowledge, specifically those between named, wikified entities (such as knowing that Damascus is part of Syria).

This annotation also captures more event coreference phenomena than what is captured in OntoNotes-style coreference annotations (Pradhan et al., 2011). While those prior annotations focused upon nominal coreference, capturing verbal mentions only occasionally (when they were coreferent with a nominal mention), multi-sentence AMR annotators were instructed to link together coreferent variables regardless of their part of speech. Furthermore, because of the AMR normalization of surface-form variation, complex details regarding how to represent an event (such as the span to use for light verbs) is already normalized into single PropBank rolesets during AMR annotation. Annotation guidelines are publicly available at `https://github.com/timjogorman/Multisentence-AMR-guidelines/`.

2.3 Annotation Toolkit and Pipeline

MS-AMRs are annotated using the Anafora toolkit (Chen and Styler, 2013), a web-based system designed for coreference and temporal annotation. An example of the actual annotation interface is shown in Figure 2. As MS-AMR representations provide an annotation layer on top of gold AMRs – rather than as a change to the format of the AMRs themselves – the output of this annotation does not modify the AMRs. THis results in a set of stand-off annotations linking to each variable within a given AMR. We assert this is a useful characteristic of any modifications of the AMR corpus, as it allows the AMRs to remain compatible with existing parsing or generation systems.

Some coreference information was already provided during within-sentence AMR annotation, as AMR annotations are annotated with links between named entities and the relevant Wikipedia ID. This gold "wikification" information was provided as pre-annotations, and allowed annotators to focus upon more difficult coreference links involving pronouns, common nouns, verbs and implicit roles.

Quality control tested for consistency and coverage of the data. First and second person pronouns were checked against speaker metadata to confirm that annotators were properly keeping track of discourse

	Train	Test	Double Annotation
Files	284	9	43
AMRs	7826	201	588
Tokens	122000	3700	8200
Coreference Chains	3810	87	381
Implicit Roles	2386	67	371
Bridging Relations	1792	54	160

Table 1: Basic statistics about size of the MS-AMR corpus

participants, and certain highly anaphoric elements (such as "he" and "she") flagged whenever not annotated as anaphoric. As the AMR corpus was also being corrected and revised during this annotation to improve predicate coverage and treatment of comparatives (Bonial et al., 2018), annotations were stored with their concept labels and double-checked for changes in the underlying AMR.

Final data (included in an upcoming AMR public release) includes a description of each AMR document (as defined by LDC segmentations of multi-thread discourse into tractable discussions), defined as an ordered list of AMRs identified by their IDs. Each coreference cluster identifies explicit mentions by the ID within the normal AMR release and the variable within that AMR; implicit roles are identified by the identity of the predicate they are an argument of, with a label for the numbered argument that is implicit.

2.4 Corpus Profile

Multi-Sentence AMR was annotated over previously annotated gold AMRs of colloquial written English (from multi-post discussion forums and web blogs), filtered for discussions with fewer than 100 sentences. The corpus is composed of 293 annotated documents in total, with an average of 27.4 AMRs (429 words) per document, covering roughly 10% of the total AMR corpus. Counts for different types of mentions are listed in Table 1.

A small test split is also defined, annotated over documents from the test split of the AMR corpus. This is a small test set for systems to report preliminary results; it is hoped that this can result in shared tasks that might test against larger sets of unseen data. Annotation of this corpus is complete, and will be released in the next public release of AMR data through the Linguistic Data Consortium.

2.5 Related Work

Densely annotated datasets in which semantic data and coreference have been represented in the multiple layers of the OntoNotes corpus (Pradhan et al., 2011) and the Prague Czech-English Dependency Treebank (PCEDT) (Čmejrek et al., 2004). This MS-AMR data differs primarily in that the data is more directly joined into a single set of connected graphs, rather than many different layers of annotation. Annotations such as ACE and ERE also capture roles and entity annotations alongside coreference, but only cover a small portion of the total semantics of a document, filtering only the elements relevant to a task-specific ontology (Song et al., 2015; Bies et al., 2016).

Li Song and Xue (2018) annotates dropped pronouns over Chinese AMR annotators, but only deals with implicit roles in specific constructions. Annotations independent of AMR have provided implicit role annotations with PropBank or NomBank(Palmer et al., 2005; Meyers et al., 2004) semantic roles, but both resources were limited to a small inventory of 5-10 predicate types, rather than all implicit arguments in a text(Gerber and Chai, 2012; Moor et al., 2013).

Bridging information has also been annotated in a range of recent corpora (Nedoluzhko et al., 2009; Poesio et al., 2008; O'Gorman et al., 2016; Roesiger, 2018) and resulted in systems (Poesio et al., 2004) capable of predicting such relations. Some bridging relations were also captured by within-sentence AMR annotations, encoded by the "include-91" relation.

While MS-AMR focuses upon capturing the *propositional* content of a document, it does not capture other dimensions of discourse annotation, such as rhetorical structure. Corpora such as RST (Carlson

et al., 2003), SDRT (Baldridge et al., 2007), GraphBank (Wolf et al., 2004) or PDTB (Miltsakaki et al.,) therefore capture dimensions of meaning that differ from the propositional content captured in this corpus.

3 Measuring MS-AMR Corpus Quality

For a subset of the training data, we took each document (i.e. the sequence of AMRs that reflect a document) and double-annotated it, to measure inter-annotator agreement and check for persistent errors. These additional annotations are listed in Table 1 under the "Double Annotation" column, and provided in the release.

3.1 Coreference Chain Quality

When dealing with two MS-AMR annotations done over the same set of gold AMRs – as with inter-annotator agreement data here – we can treat a given variable in each AMR as a possible "mention", just as one might treat a span of text in a document. With that assumption, we may therefore measure MS-AMR IAA coreference scores using standard means of measuring coreference quality. One current approach – used because it is the standard for scoring coreference systems – is to use an average of the BCUB (Bagga and Baldwin, 1998), MUC (Vilain et al., 1995) and CEAF-E (Luo, 2005) metrics, referred to as the "CoNLL-2012 F1 score" (calculated using the reference implementation of Pradhan et al (2014a)).

Under those assumptions, the annotations get a CoNLL-2012 F1 of 69.86, using the reference implementation for these scores. For comparison, more traditional span-based coreference annotations (O'Gorman et al., 2016) found inter-annotator agreement of 65.5 for event F1, and 70.4 for entity F1 (CoNLL F1 score). This is therefore in the rough range of where one might expect human performance to be. However, this does not have actual comparability to other annotation schemes, as this data uses both event and entity coreference, and does not encompass within-sentence coreference (which is already provided in gold AMR annotations).

3.2 Agreement when underlying AMRs are not identical

Because MS-AMR was annotated on top of gold AMRs, most inter-annotator agreement pairs were annotated over an identical set of within-sentence AMRs for the document. This has the advantage of separating multi-sentence AMR disagreements from other AMR disagreements, but it left open the question of whether that agreement would change drastically if the underlying AMRs were different. Starting with documents where two separate annotators provided different AMRs for each sentence, we separately annotated two multi-sentence AMR annotations for that same document, each annotated on top of a different set of AMRs.

The challenges in evaluating this kind of agreement presage the challenges of measuring the quality of system-predicted MS-AMR. In traditional annotations of coreference over surface forms, one can assume that two mentions are identical when they refer to the same span of text. However, with two separate AMRs, one cannot directly infer whether two mentions reference the same span, but must determine a mapping between the variables of each AMR. The SMATCH (Cai and Knight, 2013) algorithm, used for evaluating AMRs, calculates such a mapping. Using SMATCH to align those AMRs and coreference scores from the reference implementation of the scoring metrics (Pradhan et al., 2014b), we find the CoNLL-2012 F1 to be 66.72. Such a score is limited in that it was measured over a very small exploration set (40 AMRs), but it provides some preliminary suggestion that the identical underlying AMRs used in the IAA numbers above are not dramatically inflating the inter-annotator agreement.

3.3 Implicit Role Agreement

Implicit roles have been annotated in prior corpora, but those annotations have been limited in size or coverage (Ruppenhofer et al., 2010; Gerber and Chai, 2010). The most comparable prior annotation is that of Gerber and Chai (2010), which looked at 10 nominal predicates in WSJ data, presenting annotators with numbered arguments without explicit mentions and instructions to link these arguments to

Relation Type	Resource	Metric	Score
Coreference	This work	CoNLL-F1	69.9
	RED-Entities (O'Gorman et al. 2016)	CoNLL-F1	70.4
	RED-Events (O'Gorman et al. 2016)	CoNLL-F1	65.5
Implicit Role	This work	Cohen's κ	0.59
	Gerber and Chai (2010)	Cohen's κ	0.64

Table 2: Agreement, alongside agreement in similar corpora (not perfectly comparable)

previously mentioned terms. That annotation resulted in a Cohen's kappa (Cohen, 1960) of κ=0.64. The current annotations result in a lower kappa of κ=0.59, which is a testament to the difficulty of the task.

As found in Gerber and Chai (2010), the bulk of this type of error is in the task of discerning whether a given referent is implicit at all. When both annotators agreed that a given implicit role was present, κ=0.85.

The implicit roles that annotators disagreed on were often those whose referents could be construed either as a specific referent in context or as a generic reference, most commonly with the non-focused element in a communication or mental state verb – such as the person being interested in "that's interesting", or the cause of "I laughed out loud". A similar issue involved the recipient or listener role for verbs like "say-01" or "ask-01", which can sometimes be inferable from context but are low in prominence.

4 Measuring MS-AMR similarity — scoring system performance

4.1 A baseline implementation

We present a simple baseline that hints at a lower bound for the task. We use the publicly available version of the Brandeis transition-based AMR parser (Wang et al., 2016) combined with an off-the-shelf coreference system (Clark and Manning, 2016) using an AMR-to-surface-form aligner (Flanigan et al., 2014) to convert the surface coreference to links between AMR nodes.

4.2 Simple evaluation of system prediction

As mentioned in section 3.2, one hurdle in evaluating system predictions comes from the absence of clearly alignable "mentions" for use in coreference metrics. We can use the SMATCH metric on each individual sentence within an AMR document to resolve this, as calculating a SMATCH score involves determining the highest scoring alignment between variables within a system prediction and a gold AMR. We can then score against that mapping.

The simple baseline outlined above was evaluated according to this metric, and we found a 27.79 CoNLL-F1 average (evaluating against the inter-annotator agreement data, which we use as a development set). While some amount of this may reflect simple differences such as the shift in domain to discussion forum data, it also underlines the inherent challenge of the task.

4.3 Evaluation with Document-level SMATCH

One hope with MS-AMR data is to encourage people to produce systems that go from strings to a representation of the meaning of a document. We therefore may want to have metrics that do not simply score coreference, but which score the entire quality of a resultant knowledge graph. One method to accomplish this goal is to use these multi-sentence AMR annotations to combine all the AMRs of a document into a single large "document graph", and to score entire document graphs using the SMATCH metric.

We follow prior work in abstractive summarization (Liu et al., 2015), which originally outlined simple methodologies for combining AMRs into a larger AMR for a document. This fundamentally involves a combination of two things – concatenating all sentences together under a new root node, and merging variables that are coreferent into a single variable.

Earlier work on this (Liu et al., 2015) merged only identical named entities and date-entity subgraphs, and therefore did not run into the complexities in terms of merging documents with similar

Bill left for Paris
(l / leave-11
:ARG0 (p / person :wiki - :name (n / name :op1 “Bill”))
:ARG2 (c / city :wiki “Paris” :name (n / name :op1 “Paris”)
He arrived at noon
(a / arrive-01
:ARG1 (h / he)
:ARG4 (i2 / implicit role: end point; destination)
:TIME (d / date-entity :dayperiod (n3 / noon)))

Merged form:
(m / multi-sentence)
:snt1 (l / leave-11
:ARG0 (p / person :wiki - :name (n / name :op1 “Bill”))
:instance-of “he”
:ARG2 (c / city :wiki “Paris” :name (n / name :op1 “Paris”)
:snt2 (a / arrive-01
ARG1 p
ARG4 c
:TIME (d / date-entity :dayperiod (n3 / noon)))

Figure 3: Example of merging multiple documents into a single AMR.

but non-identical information. When two triples are identical – such as twice labeling “Hillary Clinton as “:instance-of person”, or adding multiple “:wiki” links to “Hillary_Clinton” – we merge those redundant bits of information. However, if a variable has different concepts or relations (e.g. “person” in one AMR and “woman” in another), the additional concepts are added as additional “instance-of” relations to the resultant merged entity. Figure 3 illustrates how the merging of AMRs is actually enacted into an output AMR.

We then score these two document graphs using the SMATCH metric, originally proposed to score single-sentence AMRs. SMATCH scores each relation within an AMR graph, as measured as a triple containing the variable, the relation, and the variable or constant that is being linked to. These triples can reflect an actual relation, such as saying that there is a TIME relation between variable *a* and variable *d*, or can express the “instance-of” relations captured by the “/” in AMR, such as *a* being an instance-of “arrive-01”. The SMATCH metric calculates an F1 measuring how many of the triples within the gold AMR correspond to a triple in the system AMR. The issue is that AMR variable names are somewhat arbitrary — “a” and “d” in one AMR do not necessarily correspond to “a” and “d” in another AMR — and therefore one needs to calculate how to map the variables within another AMR onto the variables within the gold AMR; Cai and Knight (2013) introduced a hill-climbing method allowing one to calculate the alignment that gives the highest F1 over these triples. Importantly for multi-sentence AMR, we can utilize the same metric over a much larger AMR graph, although this becomes computationally expensive with larger document sizes.

We therefore score these document AMRs by treating them as single AMRs to be scored using SMATCH. As such, this is an evaluation of both the quality of within-sentence AMRs and coreference information. Table 3 illustrates the performance of the baseline system, compared with inter-annotator agreement scores. Following the suggestion to approach coreference evaluation based on how it works on edge cases (such as leaving all mentions as singletons) (Recasens and Hovy, 2011), we also report scores for the document-level graphs when no cross-sentence coreference information is used. The “AMR=identical” condition illustrates most instances of double annotation in our data, where both versions were annotated over the same AMRs; these receive very high SMATCH scores, due to the identical underlying AMRs. The “AMR=human” condition illustrates a small set where two different annotations of each AMR were used; this therefore represents human performance at this task. One can see that there

System type	AMR	Coreference	Double-annotation data	Test
baseline	system	none	53.0	43.2
baseline	system	system	53.6	44.0
IAA	human	none	58.0	
IAA	human	human	68.9	
IAA	identical	none	78.5	80.6
IAA	identical	system	80.1	82.9
IAA	identical	human	87.3	

Table 3: Agreement and baseline performance using SMATCH over document graphs. The AMR=human condition is evaluated on a much smaller set of AMRs where we have two annotated AMRs for each sentence.

is still a quite meaningful gap between this performance and the baseline system performance.

Notably, the simple baseline systems do not have dramatic increases over simply scoring the document SMATCH of a sequence of AMRs with no coreference. Basic analysis of this shows that some of this error is a consequence of the discussion genre: a baseline coreference model that did not track speakers in the AMRs makes important errors in coreference chains regarding "I" and "you". However, one can also see from Table 3 that the MS-AMR score is still very dependent upon the quality of the within-sentence AMRs, as one might expect from a measure of the general quality of a document representation.

5 Conclusions

We present here a new corpus of coreference, partial coreference and implicit roles on top of the Abstract Meaning Representation corpus. While this is fundamentally an extension of AMR, we frame this in relation to prior coreference work and propose a methodology for evaluating multi-sentence AMR system predictions against these gold annotations. Such an annotation does not fully capture all of the information one might hope to represent about the meaning of a document, as these representations leave unmarked a great deal of information about temporal and aspectual structure, discourse structure or information structure. However, this corpus illustrates a methodology of annotating new layers of meaning on top of AMR representations, which might be extended to such other representations.

Acknowledgements

This work is supported by Defense Advanced Research Projects Agency (DARPA) under DEFT-FA-8750-13-2-0045 . We would like to thank annotators at the University of Colorado – Boulder and the Linguistic Data Consortium for their diligent annotations and insightful feedback on the guidelines.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, volume 1, pages 563–566. Granada.

Jason Baldridge, Nicholas Asher, and Julie Hunter. 2007. Annotation for and robust parsing of discourse structure on unrestricted texts. *Zeitschrift für Sprachwissenschaft*, 26(2):213–239.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of Linguistic Annotation Workshop*.

Ann Bies, Zhiyi Song, Jeremy Getman, Joe Ellis, Justin Mott, Stephanie Strassel, Martha Palmer, Teruko Mitamura, Marjorie Freedman, Heng Ji, and Tim O'Gorman. 2016. A comparison of event representations in deft. In *Proceedings of the Fourth Workshop on Events*, pages 27–36.

Claire Bonial, Bianca Badaru, Kira Griffitt, Ulf Hermjakob, and Kevin Knight. 2018. Abstract meaning representation of constructions: The more we include, the better the representation. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Shu Cai and Kevin Knight. 2013. Smatch: an Evaluation Metric for Semantic Feature Structures. In *ACL (2)*, pages 748–752.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current and new directions in discourse and dialogue*, pages 85–112. Springer.

Wei-Te Chen and Will Styler. 2013. Anafora: a web-based general purpose annotation tool. In *Proceedings of the conference. Association for Computational Linguistics. North American Chapter. Meeting*, volume 2013, page 14. NIH Public Access.

Kevin Clark and Christopher D. Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. In *Empirical Methods on Natural Language Processing*.

Herbert H Clark. 1977. *Bridging. Thinking: readings in cognitive science*. Cambridge: Cambridge University Press.

Martin Čmejrek, Jan Hajič, and Vladislav Kuboň. 2004. Prague czech-english dependency treebank: Syntactically annotated resources for machine translation. In *In Proceedings of EAMT 10th Annual Conference*. Citeseer.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1426–1436.

Sahil Garg, Aram Galstyan, Ulf Hermjakob, and Daniel Marcu. 2016. Extracting biomolecular interactions using semantic parsing of biomedical text. In *AAAI*, pages 2718–2726.

Matthew Gerber and Joyce Y. Chai. 2010. Beyond NomBank: A study of implicit arguments for nominal predicates. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1583–1592.

Matthew Gerber and Joyce Y. Chai. 2012. Semantic role labeling of implicit arguments for nominal predicates. *Computational Linguistics*, 38(4):755–798.

Sijia Ge Bin Li Junsheng Zhou Weiguang Qu Li Song, Yuan Wen and Nianwen Xue. 2018. An easier and efficient framework to annotate semantic roles: Evidence from the chinese amr corpus. In Kiyoaki Shirai, editor, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA), may.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 25–32. Association for Computational Linguistics.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. Annotating noun argument structure for NomBank. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, pages 803–806.

Eleni Miltsakaki, Rashmi Prasad, Aravind K Joshi, and Bonnie L Webber. The penn discourse treebank. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004*.

Tatjana Moor, Michael Roth, and Annette Frank. 2013. Predicate-specific annotations for implicit role binding: Corpus annotation, data analysis and evaluation experiments. In *Proceedings of the 10th International Conference on Computational Semantics*.

Anna Nedoluzhko, Jiří Mírovskỳ, and Petr Pajas. 2009. The coding scheme for annotating extended nominal coreference and bridging anaphora in the prague dependency treebank. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 108–111. Association for Computational Linguistics.

Tim O'Gorman, Kristin Wright-Bettner, and Martha Palmer. 2016. Richer event description: Integrating event coreference with temporal, causal and bridging annotation. In *Proceedings of the 2nd Workshop on Computing News Storylines (CNS 2016)*, pages 47–56.

M. Palmer, P. Kingsbury, and D. Gildea. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Massimo Poesio, Renata Vieira, and Simone Teufel. 1997. Resolving bridging references in unrestricted text. In *Proceedings of a Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, pages 1–6. Association for Computational Linguistics.

Massimo Poesio, Rahul Mehta, Axel Maroudas, and Janet Hitzeman. 2004. Learning to resolve bridging references. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 143. Association for Computational Linguistics.

Massimo Poesio, Ron Artstein, et al. 2008. Anaphoric annotation in the arrau corpus. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC 2008)*.

S. Pradhan, L. Ramshaw, M. Marcus, M. Palmer, R. Weischedel, and N. Xue. 2011. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–27.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014a. Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of the Association for Computational Linguistics*.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014b. Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2014, page 30. NIH Public Access.

Marta Recasens and Eduard Hovy. 2011. Blanc: Implementing the rand index for coreference evaluation. *Natural Language Engineering*, 17(4):485–510.

Ina Roesiger. 2018. Bashi: A corpus of wall street journal articles annotated with bridging links. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA), may.

Josef Ruppenhofer, Caroline Sporleder, Roser Morante, Collin Baker, and Martha Palmer. 2010. Semeval-2010 task 10: Linking events and their participants in discourse. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 45–50. Association for Computational Linguistics.

Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. From light to rich ere: annotation of entities, relations, and events. In *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 89–98.

Lai Dac Viet, Nguyen Le Minh, and Ken Satoh. 2017. Convamr: Abstract meaning representation parsing for legal document. *arXiv preprint arXiv:1711.06141*.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th conference on Message understanding*, pages 45–52. Association for Computational Linguistics.

Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. Camr at semeval-2016 task 8: An extended transition-based amr parser. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1173–1178.

Florian Wolf, Edward Gibson, Amy Fisher, and Meredith Knight. 2004. Discourse graphbank. *Linguistic Data Consortium, Philadelphia*.

Incorporating Argument-Level Interactions for Persuasion Comments Evaluation using Co-attention Model

Lu Ji[1] **, Zhongyu Wei**[2*]**, Xiangkun Hu**[1]**, Yang Liu**[3] **, Qi Zhang**[1] **, Xuanjing Huang**[1]
[1]School of Computer Science, Fudan University, Shanghai, China
[2]School of Data Science, Fudan University, China
[3]Liulishuo Company
{17210240034,zywei,xkhu17,qi_zhang,xjhuang}@fudan.edu.cn
yang.liu@liulishuo.com

Abstract

In this paper, we investigate the issue of persuasiveness evaluation for argumentative comments. Most of the existing research explores different text features of reply comments on word level and ignores interactions between participants. In general, viewpoints are usually expressed by multiple arguments and exchanged on argument level. To better model the process of dialogical argumentation, we propose a novel co-attention mechanism based neural network to capture the interactions between participants on argument level. Experimental results on a publicly available dataset show that the proposed model significantly outperforms some state-of-the-art methods for persuasiveness evaluation. Further analysis reveals that attention weights computed in our model are able to extract interactive argument pairs from the original post and the reply.

1 Introduction

Computational argumentation is a growing field in natural language processing. Existing research covers argument unit detection (Al-Khatib et al., 2016), argument structure prediction (Peldszus and Stede, 2015; Stab and Gurevych, 2014), argumentation scheme classification (Feng et al., 2014), etc. Recently, the automatic assessment of argumentation quality has started gaining attention. It can be analyzed at two levels, namely monological argumentation and dialogical argumentation.

Monological argumentation refers to a composition of arguments on a certain issue (Wachsmuth et al., 2017). A typical example of quality evaluation for monological argumentation is automated essay scoring, which aims to process argumentative essays without human interference (Taghipour and Ng, 2016). It takes an essay as the input and outputs a numeric score, considering features of content, grammar, discourse structure and lexical richness (Burstein et al., 2013). Most of the efforts are made on the exploration for better document representation.

Dialogical argumentation refers to a series of interactive arguments related to a given topic, involving argument retraction, view exchange, and so on (Besnard et al., 2014). With the popularity of online debating forums like *convinceme* [1], *debatepedia* [2] and *change my view (CMV)* [3], researchers pay increasing attention to evaluating the quality of debating or persuasive content (Tan et al., 2016; Wei and Liu, 2016; Wei et al., 2016; Habernal and Gurevych, 2016a). Although some similarity features of content between the reply and the original post are used to evaluate the quality of the given reply, they are computed on word level without considering the exchange of opinions on the basis of arguments.

An example of dialogical argumentation is shown in Figure 1. There are three posts, one is original and the other two are replies. The repliers post to change the view of the original poster. And the *positive reply* is deemed to be more persuasive than the *negative reply*. We have three observations. First, viewpoints of the original poster and repliers are expressed via multiple arguments. Second, content of replies is organized in-line with arguments in the original post. Third, interactions between a reply and the

*Corresponding author

[1]http://convinceme.net.
[2]http://debatepedia.idebate.org.
[3]https://reddit.com/r/changemyview

Proceedings of the 27th International Conference on Computational Linguistics, pages 3703–3714
Santa Fe, New Mexico, USA, August 20-26, 2018.

Original Post: Philosophy doesn't seem to have any practical applications. [What value does philosophy have in the modern age, right now, aside from contemplating things?] I have read the argument that it is impossible to argue that philosophy is useless without using philosophy. [What do you gain from studying philosophy that could not be gained from thoughtful introspection?]	
Positive Reply	**Negative Reply**
What do you gain from studying philosophy that could not be gained from thoughtful introspection? [Two answers. #1 rigor and #2 it saves us from reinventing the wheel.] [Why do you think we should start from scratch in all value decisions rather than seeking to understand the work that has been done in the past?]	What do you gain from studying philosophy that could not be gained from thoughtful introspection? [Ask yourself the same question about math.] Your argument seems to be that studying philosophy is a waste of time because it has no practical use.

Figure 1: An example of dialogical argumentation consists of one original post and two persuasion replies from *change my view*, a sub-forum of *Reddit.com*. Different types of underlines are used to highlight the interactive relationship. The labels of the positive and negative replies are assigned by the original poster.

original post provide some indications for the persuasive comment identification. Inspired by the three findings, we aim to analyze dialogical argumentation on argument level and explore how argument-based interactions can help persuasiveness evaluation. Our datasets are collected from an on-line forum [4]. The content of posts is usually informal and not strictly grammatical. It is extremely difficult to parse the argument in a finer-grain with premise, conclusion and other components. Therefore, we treat each sentence as an argument for simplicity.

In this paper, we propose to incorporate argument-level interactions within dialogical argumentation for better persuasion comments quality evaluation. We propose a novel framework that includes three components, namely, argument representation, co-attention network and aggregation network. We first learn two different representations for each argument via a hierarchal neural network. Co-attention network captures the interactions between the reply and the original post on argument level via three kinds of attentions. Aggregation network combines the results of the co-attention network. Finally, a persuasiveness score is assigned to the target comment using a linear transformation. Experimental results on a benchmark dataset show that with the assistance of argument-level interactions, our proposed model can achieve much better performance than some state-of-the-art methods. In order to further understand how attention mechanism works for capturing the argument-level interactions, we formalize a task of interactive argument pair extraction. Experimental results on a self-constructed dataset show that our attention-based strategy significantly outperforms a word-overlap based strategy for the identification of interactive arguments.

2 Proposed Model

Given an original post and two corresponding replies, our task is to automatically identify which reply is more persuasive. In practice, we evaluate the quality of the two replies separately given the original post and treat the one with higher persuasiveness score as the winner. The overall architecture of our model is shown in Figure 2. It takes an original post and a reply as inputs, and outputs a real value as its persuasiveness score. It mainly consists of three components, namely, *Argument Representation*, *Co-attention Network* and *Aggregation Network*. We learn two representations for each single argument. One is based on its internal words and the other considers information of context arguments. What's more, three types of attentions are proposed to model the detailed interactions between the original post and the reply on argument level. The *Aggregation Network* integrates the results of *Co-attention Network*.

[4]Datasets are available at : https://github.com/lji0126/Persuasion-Comments-Evaluation

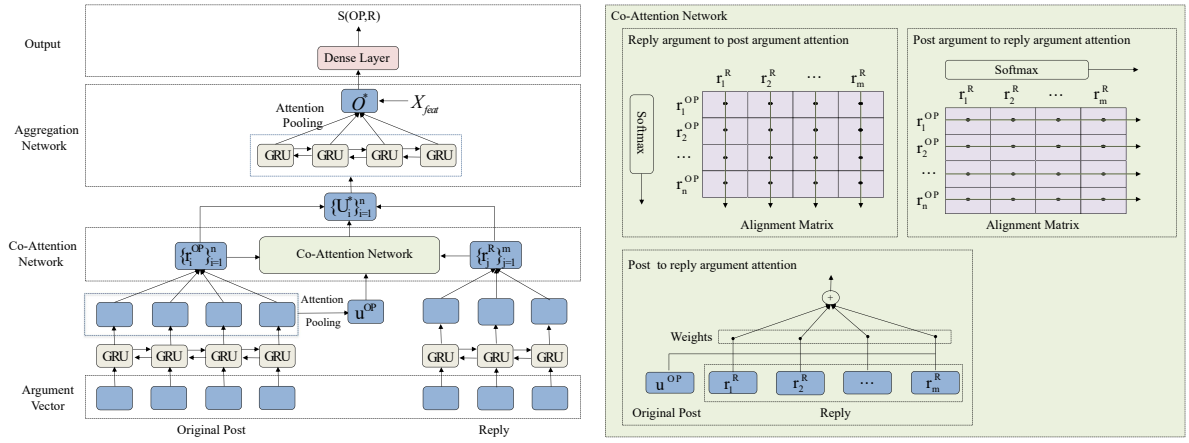


Figure 2: Overall architecture of the proposed model. The left part is the main framework of this work. The right part is the detailed structure of the co-attention network.

2.1 Argument Representation

Inspired by Dong et al. (2017), we employ a hierarchical architecture to obtain two different representations for each single argument. For simplicity, we consider each sentence as an argument.

Representation based on internal words given an argument with words $w_1, w_2, ..., w_T$, we first map each word to a dense vector obtaining $x_1, x_2, ..., x_T$ correspondingly. We then employ a convolution layer to incorporate the context information on word level.

$$z_i = f(W_z \cdot [x_i : x_{i+h_w-1}] + b_z) \tag{1}$$

where W_z and b_z are weight matrix and bias vector. h_w is the window size in the convolution layer and z_i is the feature representation. Not all words contribute equally to the representation of the argument. Therefore, we conduct an attention pooling operation over all the words to get the contribution of each word.

$$m_i = \tanh(W_m \cdot z_i + b_m) \tag{2}$$

$$u_i = \frac{e^{W_{u \cdot m_i}}}{\sum_j e^{W_{u \cdot m_j}}} \tag{3}$$

$$a = \sum_i u_i \cdot z_i \tag{4}$$

where W_m and W_u are weight matrix and vector, b_m is the bias vector, m_i and u_i are attention vector and attention weight of the i-th word. a is the argument representation.

Representation incorporating context arguments in order to incorporate information of context arguments, we employ a bi-directional GRU (Cho et al., 2014) to get the representation of an argument under its context. A BiGRU consists of both forward and backward GRU that handle the sequence from the left and the right end, respectively. In practice, we concatenate the hidden states of two GRUs for each argument to get final argument representation. $r_i^{OP} = BiGRU(r_{i-1}^{OP}, a_i^{OP}) \in R^{n \times d}, i \in [1 \cdots n]$, $r_j^R = BiGRU(r_{j-1}^R, a_j^R) \in R^{m \times d}, j \in [1 \cdots m]$, where a_i^{OP}, a_j^R are representations of arguments in the original post and the reply, respectively. r_i^{OP}, r_j^R are the hidden state of the i-th argument in the original post and the j-th argument in the reply. d is the dimension of hidden units. n and m stand for the number of arguments for the original post and the reply, respectively.

2.2 Co-attention Network

Attention mechanism is a common way to link and fuse information from two content-related texts (Weston et al., 2014; Hill et al., 2016; Sordoni et al., 2016; Shen et al., 2017). In order to capture the

interactions between the original post and the reply, we propose a novel co-attention network that includes three kinds of attentions. In particular, *Post argument to reply argument attention* computes the relevance of each post argument with every reply argument and helps learn a series of new reply representations. *Reply argument to post argument attention* computes the relevance of each reply argument with every post argument and finally obtains a series of new post representations. *Post to reply argument attention* computes the relevance of each reply argument with the entire post argument which contributes to learn a new reply representation.

Post argument to reply argument attention We first compute the alignment matrix $A \in R^{n \times m}$ that contains similarity scores corresponding to all possible pairs of arguments between the original post and the reply via Equation 5.

$$A_{ij} = W_a^T [r_i^{OP}; r_j^R; r_i^{OP} \circ r_j^R] \tag{5}$$

where W_a is the weight parameter, $\circ$ is the Hadamard product. A_{ij} indicates the similarity between i-th argument in the original post and j-th argument in the reply.

For i-th argument in the original post, we could signify which arguments in the reply are relevant to it by this attention. Similar to in Seo et al. (2016), we normalize the alignment matrix A row-wise to produce the attention weights across the reply for each argument in the original post. The calculation is described in Equation 6.

$$V_i = softmax(A_{i:}), i \in [1 \cdots n] \tag{6}$$

Based on the attention probability V_i of i-th argument in the original post, the new representation of the reply can be calculated by Equation 7.

$$U_i^1 = \sum_t V_{it} \cdot r_t^R \tag{7}$$

Based on all the arguments in the original post, we could obtain a series of new reply representations, which constitute U^1.

Reply argument to post argument attention We normalize the alignment matrix A column-wise to produce the attention weights across the original post for each argument in the reply. The attention weights can be calculated by Equation 8.

$$Q_j = softmax(A_{:j}), j \in [1 \cdots m] \tag{8}$$

Subsequently, each attended argument representation of original post is shown in Equation 9.

$$U_j^2 = \sum_t Q_{jt} \cdot r_t^{OP} \tag{9}$$

Based on all the arguments in the reply, we could get a series of new post representations, which constitute U^2.

Post to reply argument attention In order to evaluate the importance of arguments in the reply, we propose this attention. Firstly, we learn a representation u^{OP} for the original post via applying the attention pooling operation over all its hidden states $r_i^{OP}, i \in [1 \cdots n]$. We then compute attention weights with the original post based on u^{OP} for each argument in the reply. In practice, we conduct dot product between u^{OP} and each hidden state representation r_j^R in the reply and a softmax layer is used to obtain an attention distribution. The calculation process is shown in Equation 10.

$$v_j = softmax(u^{OP} \cdot r_j^R) \tag{10}$$

Based on the attention probability v_j of the j-th argument in the reply, the new representation of the reply can then be constructed as Equation 11.

$$u^3 = \sum_j v_j \cdot r_j^R \tag{11}$$

Finally, we put all of the attention representations in a linear function to get the integrated information. The detail is illustrated in Equation 12.

$$U = f(U^1, U^2, U^3, \{r_i^{OP}\}_{i=1}^n, \{r_j^R\}_{j=1}^m) \tag{12}$$

where f is a simple linear function, U^3 is a matrix that is tiled n times by u^3.

2.3 Aggregation Network

After acquiring the local alignment representation by the co-attention network, we employ a filtration gate to hold the interactive information. Then, we fuse the interactive information via a bi-directional GRU and compute the persuasiveness score.

Filtration Gate We utilize the filtration gate (Wang et al., 2017b) to hold the information that helps to understand the argument-level interactions between the original post and the reply. The formulas are in Equation 13 and 14.

$$gt = sigmoid(W_g U + b) \tag{13}$$

$$U^* = gt \odot U \tag{14}$$

We fuse the interactive information reserved by the filtration gate via a bi-directional GRU. The calculation is described in Equation 15.

$$O_t = BiGRU(O_{t-1}, U_t^*) \tag{15}$$

Then, we use an attention pooling operation over the whole hidden states of this BiGRU to summarize the interactive features into a dense vector O^*.

Scoring Tay et al. (2017) prove that adding some manual features such as word-overlap to models is helpful for improving performance. We incorporate some word-overlap features X_{feat} to the proposed model, i.e. similarities between original post and the reply in terms of word-overlap. Then, we use two fully connected layers to obtain a higher-level representation r. Finally, the persuasiveness score S is obtained by a linear transformation via Equation 16 and 17 .

$$r = f(W_r[O^*; X_{feat}] + b_r) \tag{16}$$

$$S = W_s r + b_s \tag{17}$$

where W_r and W_s stand for the weight matrices, while b_r and b_s are weight vectors.

2.4 Loss Function and Training

Given an original post and two corresponding replies, we want to automatically identify which reply is more persuasive. We formalize this issue as a ranking task and utilize a pairwise hinge loss for training. Given a triple (OP, R^+, R^-), where R^+ and R^- respectively denote the positive and the negative reply for OP. The loss function is defined in Equation 18.

$$L = max(0, 1 - S(OP, R^+) + S(OP, R^-)) \tag{18}$$

where $S(OP, R^+)$ and $S(OP, R^-)$ are the corresponding persuasiveness scores.

The model is trained by stochastic gradient descent on 105 epochs, and evaluated on the development set at every epoch to select the best model. Dropout (Srivastava et al., 2014) has proved to be an effective method and is used in our work. We use Glove (Pennington et al., 2014) word embeddings, which are 50-dimension word vectors trained with a crawled large corpus with 840 billion tokens. Embeddings for words not present are randomly initialized with sampled numbers from a uniform distribution [-0.25,0.25]. We set initial learning rate to 0.1, batch size to 20, filter sizes to 5, filter numbers to 100 and the hidden unit of BiGRU to 200. Early stopping was used with a patience of 15 epochs. We implemented our model using TensorFlow. The model converged in 23 hours on an NVIDIA Titan X machine.

	Training Set				Test Set			
	Ave_w	Var_w	Ave_p	Var_p	Ave_w	Var_w	Ave_p	Var_p
Original post	10	49.5	14	163.7	11	53.2	15	133.7
Positive reply	10	46.3	14	125.0	10	44.1	13	123.8
Negative reply	10	39.2	11	82.0	10	44.7	10	69.5

Table 1: Statistics of the evaluation dataset. Ave_w represents the average number of words per argument. Ave_p represents the average number of arguments per post. Var_w indicates the variance of the number of words per argument. Var_p indicates the variance of the number of arguments per post.

3 Experiments

3.1 Dataset and Metric

We use the same dataset as Tan et al. (2016) for evaluation, which focuses on arguments from root reply. The dataset is collected from the */r/ChangeMyView* subreddit (CMV). In CMV, users submit posts to elaborate their perspectives on a specific topic and other users are invited to argue for the other side to change the posters' opinions. Users can give *delta* to a reply if it changes their original mind about the topic. In this dataset, for the same original post, the reply with *delta* is treated as *positive reply*, otherwise it is chosen as the *negative reply*.

The whole dataset consists of 3,456 training instances and 807 testing instances, where each instance includes an original post with one positive and one negative reply respectively. We randomly select 10% of the training instances to form the development set. In preprocessing, we use NLTK [5] for tokenization and lowercase conversion. We also filter out stop words and low frequency words. The constructed word vocabulary contains 15,767 distinct words. The detailed statistics are shown in Table 1.

Since we treat this task as a pairwise ranking problem, pairwise accuracy is conducted as the evaluation metric, which also mentioned in Tan et al. (2016).

3.2 Models for Comparing

We compare our model with the previous state-of-the-art model and the variant models of our model.
– **Tan et al. (2016)**: Tan et al. (2016) regard this task as a binary classification problem and use logistic regression model to classify replies based on some manually designed features, including interplay features, argument-related features and text style features. Because there is no source code published, we directly present the result reported in their paper for comparison.
– **Word-level BiGRU (WB)**: This model employs BiGRU to encode the original post and the corresponding reply on word level. Both representations of the original post and the reply are then concatenated to compute its persuasiveness score using a fully connected layer.
– **CNN + BiGRU (CB)**: This model encodes the original post and the corresponding reply via a hierarchical neural network. All the hidden states from the BiGRU are input into the aggregation network to compute the persuasiveness score. This is a part of our model without the co-attention network and the word-overlap features.
– **Word Overlap Features (WOF)**: This model directly uses the word-overlap features to evaluate the quality of arguments. More concretely, word-overlap features contain Jaccard similarity and some scores based on common words between the original post and the reply.
– **CNN+BiGRU+Co-Att (CBCA)**: This model is a part of our model without the word-overlap features. We input argument representations of the original post and the corresponding reply into the co-attention network and then obtain the persuasiveness score via the results of aggregation network.
– **CNN + BiGRU + Word Overlap Features (CBWOF)**: This model is a part of our model without the co-attention network. After obtaining argument representations of the original post and the corresponding reply, we input the hidden states of BiGRU into the aggregation network. We then concatenate the result of aggregation network with the word-overlap features to get the persuasiveness score.

[5] http://www.nltk.org/

Model	Pairwise accuracy
Tan et al. (2016)	<u>65.70</u>
Word-level BiGRU (WB)	61.22
CNN+BiGRU (CB)	63.34
Word Overlap Features (WOF)	63.59
CNN+BiGRU+Co-Att (CBCA)	66.96‡
CNN+BiGRU+Word Overlap Features (CBWOF)	68.08‡
CNN+BiGRU+Att_III+Word Overlap Features (CBAWOF_III)	69.95‡
CNN+BiGRU+Att_I+Word Overlap Features (CBAWOF_I)	70.07‡
CNN+BiGRU+Att_II+Word Overlap Features (CBAWOF_II)	70.20‡
CNN+BiGRU+Co-Att+Word Overlap Features (CBCAWOF)	**70.45**‡*

Table 2: The performance of different approaches on our datasets. The model underlined is the state-of-the-art method. The models that outperform the state-of-the-art method are highlighted with ‡. Our model that significantly outperforms the state-of-the-art method is marked with * (p<0.01, Student's paired t-test and Wilcoxon signed rank test). Best result is in **bold**.

– **CNN+BiGRU+Att_I+Word Overlap Features (CBAWOF_I)**: This model is a part of our model with the post argument to reply argument attention in the co-attention network.
– **CNN+BiGRU+Att_II+Word Overlap Features (CBAWOF_II)**: This model is a part of our model with the reply argument to post argument attention in the co-attention network.
– **CNN+BiGRU+Att_III+Word Overlap Features (CBAWOF_III)**: This model is a part of our model with the post to reply argument attention in the co-attention network.
– **CNN + BiGRU + Co-Att+Word Overlap Features (CBCAWOF)**: This is our proposed model.

3.3 Results and Discussion

The overall result of the comparison is shown in Table 2. We have following findings.
– The model proposed by Tan et al. (2016) achieves an accuracy of 65.70 % on the dataset. The performance actually shows the effectiveness of human generated features on this task. However, some writing style features used are very difficult to obtain, limiting its ability to generalize. The authors also explore to use interactive features between the original post and the reply, however, only word-level text similarity is considered.
– The performance of *CB* is much better than that of *WB*. This proves the effectiveness of representing posts on argument level instead of word level.
– The performance of *WOF* is comparable to that of *CB*. This indicates that correlation between the original post and the reply is an important feature for persuasiveness evaluation.
– The performance of *CBCA* is much better than that of *CB*. This indicates the effectiveness of the co-attention network.
– By combining both argument representations and word-overlap features, the performance of *CBWOF* is significantly better than that of *CB* and *WOF*.
– The performance of *CBAWOF_I*, *CBAWOF_II*, *CBAWOF_III* is better than that of all models except *CBCAWOF*. This proves the effectiveness of the three kinds of attentions separately.
– Our proposed model *CBCAWOF* generates the best performance among all the models. This confirms the effectiveness of our proposed co-attention model.

In order to further prove the effectiveness of our model, we conduct the *Student's paired t-test* [6] and

[6]https://en.wikipedia.org/wiki/Student's_t-test.

Original Post: [I'm talking about making the human race smarter, forever.] [We could use the IQ scale (for want of a better intelligence measure) to determine the number of offspring a person should be able to genetically contribute to.] [A man and a woman with average IQ can have two children and average IQs of 125-174 can contribute towards 3 children.] This would make human more likely to survive.

Positive Reply
[When the proposal comes up I am reminded of why it's a bad idea: not because we couldn't do it, but because we don't know how to do it right.][Why would a more intelligent society automatically be a better one?] We don't know enough about the biology. [Simply, we are not ready for eugenics.]

Negative Reply:
[As you put it, the entire goal of eugenics is to make the human race to advance and survive, but the irony of eugenics is that one of the best ways we can guarantee our survival is to maximize the size of our gene pool.] However, it's dangerous to call certain genes good and other genes bad.

Figure 3: A sample consists of one original post with two persuasion replies. Interactive argument pairs are highlighted with the same kind of underline. Pairs are extracted via co-attention network.

the *Wilcoxon signed rank test* [7]. Because Tan et al. (2016) don't publish their source code, we are unable to obtain detail results of their model. In Table 2, we find that the performance of CBWOF is better than that of Tan et al. (2016), so we carry out the significance tests between the results of our model and CBWOF. The p-value of the two significance tests is less than 0.01 respectively, which proves that our model significantly outperforms the state-of-the-art method.

4 Further Analysis on Co-attention Network

In order to further understand the capability of our co-attention network for capturing interactions between the original post and the reply on argument level, we perform an additional experiment to evaluate the effectiveness of the attention weights for the identification of interactive argument pairs. Therefore, we propose a novel task of extracting the interactive argument pairs between the original post and the corresponding reply. We first build an evaluation dataset and then compare the extraction performance of attention-based extraction strategy with a word-overlap based strategy on this dataset.

4.1 Dataset for Interactive Argument Pair Extraction

We sample 50 triples in the form of (original post, positive reply, negative reply) from the training set and split these into 100 original post-reply pairs in the form of (original post, positive reply) and (original post, negative reply). Given two collections of arguments from the original post and the reply, namely, $OP = \{op_1, op_2, op_3, ..., op_n\}$ and $R = \{r_1, r_2, r_3, ..., r_m\}$, for each argument r_j in the reply, we aim to identify arguments in the original post that interact with it.

Two annotators are hired to annotate the dataset independently and a third annotator is asked to solve the conflict between the two annotators. Two annotators identify 371 and 355 pairs of interactive arguments respectively. The inter-annotator agreement measured by Co-hens Kappa (Carletta et al., 1996) is 91.83%. With the final decision from the third annotator, we obtain 365 pairs in total. In detail, 234 interactive argument pairs come from positive replies, and the other 131 pairs are generated by negative replies. This re-confirms that the degree of interaction is a good indicator for persuasive reply identification.

4.2 Automatic Argument Extraction

We use two methods to extract argument pairs automatically. One is based on the attention weights computed in our proposed model and the other extracts pairs based on word-overlap similarity.

[7] https://en.wikipedia.org/wiki/Wilcoxon_signed-rank_test.

	P@1	P@2	P@3	P@4	P@5	MRR
WS	17.53	30.41	39.72	48.49	53.97	31.33
CN	22.19	36.71	43.84	49.86	54.24	39.41

Table 3: Experimental results of *WS* and *CN* for interactive argument pair extraction on the self-constructed dataset.

Co-attention Network (CN): We extract interactive argument pairs based on the results of our co-attention network. Because not every argument in the reply has interactive relationship, we choose 10 arguments in each reply based on the top-10 weights of the *post to reply argument attention* (this attention vector computes the importance of arguments in the reply in the perspective of the original post). Secondly, we choose the top-5 arguments in the original post for the 10 arguments in reply respectively in terms of the *reply argument to post argument attention* weights.

Word-overlap Similarity (WS): The extraction of arguments in the reply is the same as *CN*. We use the number of common words to identify interactive arguments in the original post for each argument in the reply. Top-5 arguments obtaining most common words with the argument in the reply are kept.

4.3 Results and Analysis

For each argument from the reply in the gold pair, we will see whether its corresponding argument is ranked in top k by automatic models. We report Mean Reciprocal Rank (MRR), precision at position 1, 2, 3, 4 and 5 as evaluation metrics. Table 3 shows the experiment results of *WS* and *CN*. We can see great differences between the two methods. In detail, the performance of CN in terms of P@1∼P@5 and MRR is higher than those of WS by 4.66%, 6.30%, 4.12%, 1.37%, 0.27% and 8.08% respectively. This confirms the effectiveness of our co-attention network for capturing interactions between the original post and the reply. However, the overall performance of both *CN* and *WS* are relatively low. This indicates that the task is difficult in nature. A sample of interacting argument pairs extracted by the attention-based approach can be seen in Figure 3.

5 Related Work

Two major areas related to our work are argumentation quality evaluation and attention mechanism.

5.1 Argumentation quality evaluation

Computational argumentation is a growing sub-field of natural language processing in which arguments are analyzed in various respects. Previous works in computational argumentation mainly focus on the methods for argument mining, which arms to determine the argumentative structure in texts. Recently, argumentation quality evaluation has become an active topic in this field.

There have been several attempts to address tasks related to argumentation quality evaluation. Habernal and Gurevych (2016b) propose a new task of predicting which argument from an argument pair is more convincing and use SVM and bidirectional LSTM to experiment on their annotated datasets. Tan et al. (2016) construct datasets from the ChangeMyView subreddit. They study factors affecting whether a challenger can successfully change the view of a commenter expressed in the original post and employ logistic regression to predict which reply in the pair is more persuasive.

In addition, Wei and Liu (2016) acquire discussion threads from the ChangeMyView subreddit to study the mechanisms behind persuasion. They propose and evaluate a set of features to predict the persuasiveness of debate posts, including textual features and social interaction related features. Wei et al. (2016) propose a task for quality evaluation of disputing argument. They manually annotate a real dataset collected from an online debating forum and analyze the correlation between disputing quality and different disputation behaviors. Wang et al. (2017a) use linguistic features of arguments, latent persuasive strengths of different topics and the interactions of debate comments to predict the debate outcome. Persing and Ng (2017) study the persuasiveness by designing five respects of error that have

negative impacts on persuasiveness. They not only focus on determining how persuasive an argument is, but also tell us why an argument is unpersuasive.

From the brief descriptions given above, we can find that most of the existing research focuses on the interactions among debate comments only from the perspective of text similarity. The interactions among argument pairs are ignored. In this work, we evaluate the quality of debate comments through the interactions among them on argument level.

5.2 Attention mechanism

Attention mechanism allows models to focus on specific parts of inputs at each step of a task. Moreover, attention mechanism has been proved to be significantly effective in natural language processing tasks.

Co-attention mechanism has recently attracted lots of research interest in the fields of machine translation (Bahdanau et al., 2014), question answering (Wu et al., 2017), text generation (Li et al., 2015), etc. It is computed as an alignment matrix based on two inputs, which can model complex interactions between the two inputs. Xiong et al. (2016) present a co-attention encoder to focus on relevant parts of the representations of the question and document and use a dynamic pointing decoder to locate the answer. Cui et al. (2016) propose a two-way attention mechanism to encode the passage and question mutually and induce attended attention for final answer predictions.

Self-attention mechanism is an attention mechanism aiming at aligning the sequence with itself, which has been successfully used in a variety of tasks. In Cheng et al. (2016), both encoder and decoder are modeled as LSTMs with self-attention for extractive summarization of documents. In Lin et al. (2017), the authors conduct a self-attention over the hidden states of a BiLSTM to extract the sentence embedding. Instead of sentence vector, they use a 2-D matrix to represent the embedding, with each row of the matrix attending on a different part of the sentence.

In this work, we employ a co-attention mechanism to capture the interactions between the original post and the reply on argument level. What's more, we use a self-attention mechanism to obtain the argument representation, which is called attention pooling in the previous sections.

6 Conclusions and Future Work

In this paper, we propose to incorporate argument-level interactions for dialogical argumentation. A novel co-attention network is proposed to capture the detailed interactions between the original post and the reply on argument level for better persuasiveness evaluation. Experimental results on a benchmark dataset show that the proposed model can achieve much better performance than the previous state-of-the-art method. Further analysis of extracting interactive argument pairs from the original post and the reply also proves the effectiveness of our co-attention network.

The future work will be carried out in three directions. First, we will fully investigate the usage of our model for applying to other dialogical argumentation related tasks, such as debate summarization. Second, we will enlarge the annotated dataset for interactive argument pair identification and explore more effective methods to generate argument pairs automatically. Third, we will explore to utilize topic information for the quality evaluation of persuasion comments.

Acknowledgements

The authors wish to thank the anonymous reviewers for their helpful comments. The work is partially supported by National Natural Science Foundation of China (Grant No. 61702106), Shanghai Science and Technology Commission (Grant No. 17JC1420200, Grant No. 17YF1427600 and Grant No.16JC1420401).

References

Khalid Al-Khatib, Henning Wachsmuth, Matthias Hagen, Jonas Köhler, and Benno Stein. 2016. Cross-domain mining of argumentative text through distant supervision. In *15th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 16)*, pages 1395–1404. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2015 International Conference on Learning Representations*.

Philippe Besnard, Alejandro Garcia, Anthony Hunter, Sanjay Modgil, Henry Prakken, Guillermo Simari, and Francesca Toni. 2014. Introduction to structured argumentation. 5(1):1–4.

Jill Burstein, Joel Tetreault, and Nitin Madnani. 2013. The e-rater automated essay scoring system, *Handbook of automated essay evaluation: current applications and new directions*. pages 55–67, New York, NY: Routledge

Jean Carletta. 1996. Assessing agreement on classification tasks: the kappa statistic. In *Computational linguistics*. 22(2):249–254.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561. Association for Computational Linguistics.

Kyunghyun Cho, Bart Van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 593–602.

Fei Dong, Yue Zhang, and Jie Yang. 2017. Attention-based recurrent convolutional neural network for automatic essay scoring. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 153–162.

Vanessa Wei Feng, Ziheng Lin, and Graeme Hirst. 2014. The impact of deep hierarchical discourse structures in the evaluation of text coherence. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 940–949.

Ivan Habernal and Iryna Gurevych. 2016. What makes a convincing argument? empirical analysis and detecting attributes of convincingness in web argumentation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1214–1223.

Ivan Habernal and Iryna Gurevych. 2016. Which argument is more convincing? Analyzing and predicting convincingness of web arguments using bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1589–1599. Association for Computational Linguistics.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of the 2016 International Conference on Learning Representations*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1106-1115.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *Proceedings of the 2017 International Conference on Learning Representations*.

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

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Isaac Persing and Vincent Ng. 2017. Why can't you convince me? modeling weaknesses in unpersuasive arguments. In *Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 4082–4088.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. In *Proceedings of the 2017 International Conference on Learning Representations*.

Yelong Shen, Po Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055.

Alessandro Sordoni, Philip Bachman, Adam Trischler, and Yoshua Bengio. 2016. Iterative alternating neural attention for machine reading. *arXiv preprint arXiv:1606.02245*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. In *The Journal of Machine Learning Research*. 15(1):1929–1958.

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56.

Kaveh Taghipour and Hwee Tou Ng. 2016. A neural approach to automated essay scoring. In *Conference on Empirical Methods in Natural Language Processing*, pages 1882–1891.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of the 25th international conference on world wide web*, pages 613–624.

Yi Tay, Minh C Phan, Luu Anh Tuan, and Siu Cheung Hui. 2017. Learning to rank question answer pairs with holographic dual lstm architecture. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 695–704.

Henning Wachsmuth, Nona Naderi, Yufang Hou, Yonatan Bilu, Vinodkumar Prabhakaran, Tim Alberdingk Thijm, Graeme Hirst, and Benno Stein. 2017. Computational argumentation quality assessment in natural language. In *Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 176–187.

Lu Wang, Nick Beauchamp, Sarah Shugars, and Kechen Qin. 2017. Winning on the merits: The joint effects of content and style on debate outcomes. *Transactions of the Association for Computational Linguistics*, 5:219–232.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Meeting of the Association for Computational Linguistics*, pages 189–198.

Zhongyu Wei and Yang Liu. 2016. Is this post persuasive? ranking argumentative comments in online forum. In *Proceedings of the Third Workshop on Argument Mining*, pages 166–171.

Zhongyu Wei, Yandi Xia, Chen LiYang Liu, Zachary Stallbohm, Yi Li, and Yang Jin. 2016. A Preliminary Study of Disputation Behavior in Online Debating Forum. In *Meeting of the Association for Computational Linguistics*, pages 195–200.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. In *Proceedings of the 2015 International Conference on Learning Representations*.

Qi Wu, Damien Teney, Peng Wang, Chunhua Shen, Anthony Dick, and Anton van den Hengel. 2017. Visual question answering: A survey of methods and datasets. *Computer Vision and Image Understanding* , 163: 21–40.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. In *Proceedings of the 2017 International Conference on Learning Representations*.

Learning Visually-Grounded Semantics from Contrastive Adversarial Samples

Haoyue Shi [1*] **Jiayuan Mao** [2*] **Tete Xiao** [1*] **Yuning Jiang** [3] **Jian Sun** [3]
[1]: School of Electronics Engineering and Computer Science, Peking University, China
[2]: ITCS, Institute for Interdisciplinary Information Sciences, Tsinghua University, China
[3]: Megvii, Inc.
{hyshi,jasonhsiao97}@pku.edu.cn
mjy14@mails.tsinghua.edu.cn, {jyn, sunjian}@megvii.com

Abstract

We study the problem of grounding distributional representations of texts on the visual domain, namely visual-semantic embeddings (VSE for short). Begin with an insightful adversarial attack on VSE embeddings, we show the limitation of current frameworks and image-text datasets (*e.g.*, MS-COCO) both quantitatively and qualitatively. The large gap between the number of possible constitutions of real-world semantics and the size of parallel data, to a large extent, restricts the model to establish a strong link between textual semantics and visual concepts. We alleviate this problem by augmenting the MS-COCO image captioning datasets with textual contrastive adversarial samples. These samples are synthesized using language priors of human and the WordNet knowledge base, and enforce the model to ground learned embeddings to concrete concepts within the image. This simple but powerful technique brings a noticeable improvement over the baselines on a diverse set of downstream tasks, in addition to defending known-type adversarial attacks. Codes are available at https://github.com/ExplorerFreda/VSE-C.

1 Introduction

The visual grounding of language plays an indispensable role in our daily lives. We use language to name, refer, and describe objects, their properties and generally, visual concepts. Distributional semantics (*e.g.*, global word embedding (Pennington et al., 2014)) based on large-scale corpora have shown great success in modeling the functionality and correlation of words in the natural language domain. This further contributes to the success in numerous natural language processing (NLP) tasks such as language modeling (Cheng et al., 2016; Inan et al., 2017), sentiment analysis (Cheng et al., 2016; Kumar et al., 2016), and reading comprehension (Cheng et al., 2016; Chen et al., 2016; Shen et al., 2017). However, effective and efficient grounding of distributional embeddings remains challenging. Being ignorant of the corresponding visual concepts, pure textual embeddings demonstrate inferior performances when incorporating with visual inputs. Typical tasks include image/video captioning, multi-modal retrieval/understanding, and visual reasoning, some of which are further extensively studied in the paper.

Visual concept and its link with textual semantics, as a cognitive alignment, provide rich supervision to learning systems. Visual-Semantic Embedding (VSE) aims at building the bridge between natural language and the underlying visual world. Introduced by Kiros et al. (2014), the embedding spaces of both images and descriptive texts (captions) are jointly optimized and aligned. Nevertheless, even for large-scale datasets such as MS-COCO (Lin et al., 2014), the number of image-caption pairs are far less than the number of possible constitutions of real-world semantics, making the dataset inevitably sparse and biased.

To reveal this, we begin with constructing textual adversarial samples to attack the state-of-the-art system VSE++ (Faghri et al., 2017). Specifically, we study the composition of sentences in two aspects: (1) content words including nouns and numerals and (2) prepositions indicating spatial relations (*e.g.*, in,

[*] Work was done when HS, JM and TX were intern researchers at Megvii Inc. HS, JM and TX contribute equally to this paper.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3715–3727
Santa Fe, New Mexico, USA, August 20-26, 2018.

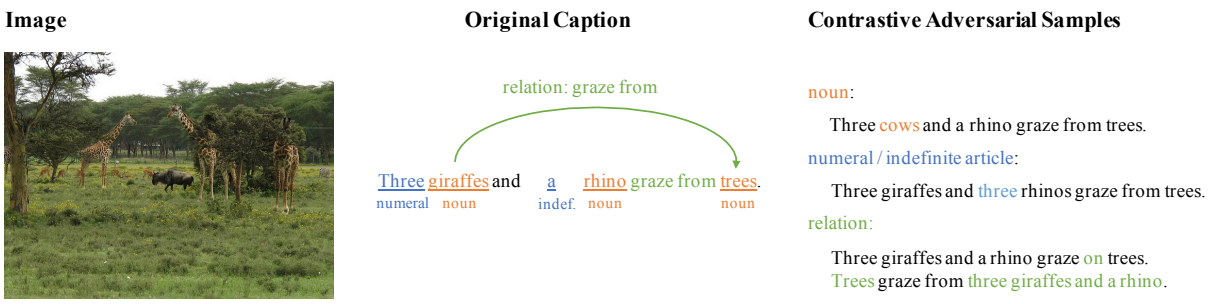


Figure 1: An overview of our textual contrastive adversarial samples. For each caption, we have three paradigms to generate contrastive adversarial samples, *i.e.*, noun, numeral and relation. For an given image, we expect the model to distinguish the real captions against the generated adversarial ones.

on, above, below). As shown in Figure 1, we manipulate the original caption to construct hard negative captions with similar structure but completely contradictory semantics. We found that the models easily get confused, suffering a noticeable drop in confidence or even wrong predictions in the caption retrieval task.

We propose VSE-C, which enforces the learning of correlation and correspondence between textual semantics and visual concepts by providing contrastive adversarial samples during the training procedure, incorporating with an intra-pair hard negative sample mining. Instead of defending adversarial attacks, we focus on the study of limitations of current visual-semantic datasets and the transferability of learned embeddings. To fulfill the large gap between the number of parallel image-caption pairs and the expressiveness of natural languages, we augment the data by employing a set of heuristic rules to generate large sets of contrastive negative captions, as demonstrated in Figure 1. The candidates are selectively used for training by an intra-pair hard-example mining technique. VSE-C alleviate the bias of dataset and provide rich and effective samples on par with original image captions. This strengthen the link between text and visual concepts by requiring models to detect a mismatch on the level of some precise concepts.

VSE-C learns discriminative and visually-grounded word embeddings on the MS-COCO dataset (Lin et al., 2014). It is extensively compared with existing works with rich experiments and analyses. Most importantly, we explore the transferability of the learned embeddings on several real-world applications both qualitatively and quantitatively, including image-to-text retrieval and bidirectional word-to-concept retrieval. Furthermore, VSE-C demonstrates a general framework for augmenting textual inputs considering semantical consistency. The introduction human priors and knowledge bases alleviates the sparsity and non-contiguity of languages. We hope the framework and the released data are beneficial for building more robust and data-efficient models.

2 Related works

Joint embeddings Joint embedding is a common technique for a wide range of tasks incorporating multiple domains, including audio-video embeddings for unsupervised representation learning (Ngiam et al., 2011), shape-image embeddings (Li et al., 2015) for shape inference, bilingual word embeddings for machine translation (Zou et al., 2013), human pose-image embeddings for pose inference (Li, 2011), image-text embeddings for visual description (Reed et al., 2016), and global representation learning from multiple domains (Castrejon et al., 2016). These embeddings map multiple domains into a joint vector space which describes the semantical relations between inputs (*e.g.*, distance, correlation).

We focus on the visual-semantic embedding (Mao et al., 2016; Kiros et al., 2014; Faghri et al., 2017), learning word embeddings with visually-grounded semantics. Examples of related applications include image caption retrieval and generation (Kiros et al., 2014; Karpathy and Fei-Fei, 2015), and visual question-answering (Malinowski et al., 2015).

Image-to-text translation Canonical Correlation Analysis (CCA) (Hotelling, 1936) is a statistical method that projects two views linearly into a common space to maximize their correlation. (Andrew et al., 2013) proposes a deep learning framework to extend CCA so that it is able to learn nonlinear projections

and has better scalability on relatively large datasets.

In the state-of-the-art frameworks, the pairwise ranking is often adopted to learn a distance metric (Socher et al., 2014; Niu et al., 2017; Nam et al., 2017). (Frome et al., 2013) proposes a cross-modal feature embedding framework that uses CNN and Skip-Gram (Mikolov et al., 2013) to extract representations for images and texts respectively, then an objective is applied to ensure that the distance between the matched image-text pair is smaller than that between the mismatched pair. A similar framework proposed by (Kiros et al., 2014) uses a Gated Recurrent Unit (GRU) as the sentence encoder. (Wang et al., 2016) uses a bidirectional loss function with structure-preserving constraints. An attention mechanism on both image and caption is used by (Nam et al., 2017) where the model estimates the similarity between images and texts by sequentially focusing on a subset of image regions and words that have shared semantics. (Huang et al., 2017) utilizes a multi-modal context-modulated attention mechanism to compute the similarity between an image and a caption. (Faghri et al., 2017) proposes a novel loss to penalize the hard negatives, *i.e.*, the closest mismatched pairs, instead of averaging the individual violations across all negatives in (Kiros et al., 2014).

Adversarial attack in text domain Adversarial attacks have recently drawn significant attention in the deep learning community. The adversarial attacks spread over multiple domains including image classification (Nguyen et al., 2015), image segmentation and object detection (Xie et al., 2017), and textual reading comprehension (Jia and Liang, 2017), and deep reinforcement learning (Kos and Song, 2017).

In this paper, we present textual adversarial attacks in image-to-text translation systems such as image caption frameworks. While focusing on the problem of learning visually-grounded semantics, the adversarial attack brings new solutions to fulfill the gap between limited training data and numerous constitutions of natural languages. With extensive experiments on the effects of the adversarial samples, we reach the conclusion that current visual-semantic embeddings are "insensitive" to the underlying semantics. The proposed VSE-C shows advance across multiple visual-semantic tasks.

3 Method

3.1 Preliminaries

Word embeddings We manually split the embeddings of each word into two parts: distributional embeddings, and visually-grounded embeddings. We use GloVe (Pennington et al., 2014) as the distributional embeddings, pre-trained unsupervisedly on large-scale corpora. We focus on the visually-grounded embeddings of words. The embeddings are optimized using the visual-semantic embedding (VSE) technique.

Visual-semantic embeddings VSE optimizes and aligns the latent space of both visual and textual domains. Parallel data are typically obtained from image captioning datasets such as Flickr30K (Young et al., 2014) or MS-COCO (Lin et al., 2014). The training set $S = \{(i_n, c_n)\}_N$ contains N image-caption pairs. Typically all $(i_n, c_m), n \neq m$ and $(i_m, c_n), n \neq m$ form the negative samples for a specific pair (i_n, c_n).

Following the notations used by (Kiros et al., 2014), domain-specific encoders are first employed to extract latent features of both images and captions, denoted as $\phi(i)$ and $\psi(c)$, respectively. We use ResNet-152 (He et al., 2016) as visual domain encoder and GRU as text domain sentence encoder, which are both effective for VSE. They are projected into a joint latent space with a linear transformation. A hinge loss with margin α is employed to optimize the alignment:

$$\ell^{VSE}(i, c) = \sum_{c'} [\alpha + s(i, c') - s(i, c)]_+ + \sum_{i'} [\alpha + s(i', c) - s(i, c)]_+ , \tag{1}$$

where $[\cdot]_+ = \max(0, \cdot)$, and $s(i, c) = W_i^T f(i; \theta_i) \cdot W_c^T g(c; \theta_c)$ measuring the distance between projected image embedding $W_i^T f(i; \theta_i)$ and caption embedding $W_c^T g(c; \theta_c)$. The summations are taken over all image-caption pairs within a sampled batch.

Class	Original Caption	Contrastive Adversarial Example
Noun	A person feeding a **cat** with a banana.	A person feeding a **dog** with a banana.
Numeral	A person feeding **a cat** with a banana.	A person feeding **five cats** with a banana.
Relation-1	**A person** feeding **a cat** with a banana.	**A cat** feeding **a person** with a banana.
Relation-2	A person feeding a cat **with** a banana.	A person feeding a cat **in** a banana.

Table 1: Examples of contrastive adversarial samples generated with our heuristic rules and knowledge from WordNet. The samples can be classified into four types: noun replacement, numeral replacement, relation shuffling, and relation replacement.

3.2 Generating contrastive adversarial samples

Our contrastive adversarial samples can be split into three classes: *noun*, *numeral* and *relation*. Each class of samples is generated separately.

Noun. We extract a list of heads (Zwicky, 1985) of noun phrases in MS-COCO dataset and label those with frequency larger than 200 be frequent heads. In addition, since images usually reflect concrete concepts better than abstract ones, we compute the concreteness of words following Turney et al. (2011), and only consider those heads with concreteness larger than $\theta = 0.6$. Only frequent concrete heads can be replaced by other frequent concrete heads with different meaning to form contrastive adversarial samples.

While replacing, we utilize the hypernymy/hyponymy relations in WordNet (Miller, 1995) to confirm the original noun and the corresponding contrastive adversarial sample are semantically different. Only words without hypernymy or hyponymy relations can be used as the replacement for adversarial sample generation. For example, "animal" is a hypernym of "cat". Therefore, "A person feeding an animal with a banana" cannot be a valid generated contrastive adversarial caption for the image with the caption of "A person feeding a cat with a banana."

Numeral. For each caption, we detect numerals and replace them with other numerals indicating different quantities to form contrastive adversarial samples. Note that "a" and "an" are treated as "one" here, though they are (indefinite) articles instead of numerals. Meanwhile, we singularize or pluralize the corresponding nouns when necessary.

Relation. The relation class includes two different paradigms.

The first one can be viewed as *shuffle of noninterchangeable noun phrases*. After extracting noun phrases of a caption, we shuffle them and put them back to the original positions. Although the bag of words features of the two sentences (caption) remain the same, the semantic meaning alters through this process.

The second one is *replacement of prepositions.* We extract the prepositions with frequency higher than 200 in MS-COCO dataset. Then we manually annotate a semantic overlap table, which can be found in Appendix A. In this table, words in the same set may have semantic overlap with each other, *e.g.*, by and with, in and among.

The noun phrase detection, preposition detection and numeral detection mentioned above are performed with SpaCy (Honnibal and Johnson, 2015). Examples of different classes of contrastive adversarial sample generation are shown in Table 1.

3.3 Intra-pair hard negative mining

We extend the online hard example mining (OHEM) technique used by VSE++(Faghri et al., 2017). The original hinge loss is computed by choosing the hardest sample within an batch (inter-pair). Mathematically,

$$\ell^{\text{VSE++}}(i, c) = \max_{c' \neq c}[\alpha + s(i, c') - s(i, c)] + \max_{i' \neq i}[\alpha + s(i', c) - s(i, c)]. \quad (2)$$

There are two major concerns regarding the in-batch hard negative mining. On one hand, mining negatives from a single batch is inefficient when batch size is not comparable with the size of the dataset.

On the other hand, for real-world datasets, taking the max in loss function tends to be very sensitive to label noise, resulting in fake negative samples.

In contrast, given an image-caption pair (i, c), we employ human heuristics and WordNet knowledge base to generate contrastive negative samples $\mathcal{C}'(c)$. To utilize these candidate caption sets, we employ an intra-pair hard negative mining strategy. Specifically, during the optimization, we add an extra loss term:

$$\ell^{\text{VSE-C}}(i, c) = \ell^{\text{VSE++}}(i, c) + \max_{c'' \in \mathcal{C}'(c)} [\alpha + s(i, c'') - s(i, c)]_+. \tag{3}$$

In our implementation, the candidate set $\mathcal{C}'$ has approximately $1,000$ samples. In each iteration, we randomly sample $N = 8$ negatives from it. This simple sample technique are effective and computation-friendly based on our empirical studies.

4 Experiments

We begin our experiments with an extensive study on the effect of adversarial samples on the baseline models. Even trained with hard negative mining techniques, VSE++ fails to discriminate words with completely contradictory visually-grounded semantics. Furthermore, we study the improvement brought by the introduction of contrastive adversarial samples on a diverse set of tasks. We release our code at `https://github.com/ExplorerFreda/VSE-C`.

4.1 Adversarial attacks

We select 1,000 images for test in MS-COCO 5k test split following (Karpathy and Fei-Fei, 2015). Each image is associated with five captions. Each caption in the selected test set can be manipulated to generate at least 20 contrastive adversarial samples by all manners (noun, numeral, and relation adversary). The image-to-caption retrieval task is defined as ranking the candidate captions based on the distance between their semantics and the given image.

We follow the metric used in Faghri et al. (2017) computing R@1, R@10, median rank and mean rank w.r.t. the top-ranked correct caption for each image. For each image, the database of retrieval contains the full set of 1000×5 captions, in which only 5 captions are labeled as positive. The R@k metric essentially measures the percentage of images where the set of top-k ranked captions contains at least one positive caption.

We attack the existing models by adversarial samples. We extend each caption with 60 adversarial samples (20 noun-typed, 20 numeral-typed and 20 relation-typed). Therefore, each image has 60×5 contrastive adversarial samples in total. The candidate retrieval set for each image now becomes $5000 + 300$. We discuss the experimental results as follows:

VSE-C are more robust to known-typed adversarial attacks than VSE and VSE++. We compare the performance of VSE-C with VSE (Kiros et al., 2014) and VSE++ (Faghri et al., 2017) in Table 2. Both VSE and VSE++ have a significant drop in performance after adding adversarial samples, while VSE-C training with contrastive adversarial samples is less vulnerable to the attacks. This phenomenon reflects that the text encoders of VSE and VSE++ do not actually make a good use of the image encodings, as the image encodings are fixed in all experiments.

Detailed attacking results are shown in Table 3. The three hyper-columns show the ability of the models to defend the adversarial attack of noun, numeral, and relation-typed respectively. Among three types of attacks, VSE and VSE++ suffer least from the noun attack. As the constitution of the dataset ensures the frequency in the entire dataset of the words used for replacement, the visual grounding of these frequent nouns is easy to obtain. However, the semantics of relations (including prepositions and entity relations) or numbers are not diverse enough in the dataset, leading to the poor performance of VSE against these attacks.

Numeral-typed VSE-C improves the counting ability of models. As shown in Table 3, numeral-typed contrastive adversarial samples improve the counting ability of models. However, it is still not clear about where the gain comes from, as the creation of numeral-typed samples may change the form (*i.e.*,

Model	MS-COCO Test				MS-COCO Test (w/. adversarial)			
	R@1	R@10	Med r.	Mean r.	R@1	R@10	Med r.	Mean r.
VSE	47.7	87.8	2.0	5.8	28.0	71.6	4.0	11.7
VSE++	**55.7**	**92.4**	1.0	**4.3**	35.6	72.5	3.0	11.8
VSE-C (+n.)	50.7	90.7	1.0	5.2	40.3	80.2	2.0	9.2
VSE-C (+num.)	53.3	90.2	1.0	5.8	46.9	86.3	2.0	6.9
VSE-C (+rel.)	52.4	89.0	1.0	5.7	42.3	82.5	2.0	7.2
VSE-C (+all)	50.2	89.8	1.0	5.2	**47.4**	**88.8**	2.0	**5.5**

Table 2: Evaluation on image-to-caption retrieval. Although VSE++ (Faghri et al., 2017) obtains the best performance on original MS-COCO test set, it is more vulnerable to the caption-specific adversarial attack compared with the proposed VSE-C, and so does VSE (Kiros et al., 2014).

Model	MS-COCO Test (+n.)			MS-COCO Test (+num.)			MS-COCO Test (+rel.)		
	R@1	R@10	Mean r.	R@1	R@10	Mean r.	R@1	R@10	Mean r.
VSE	37.6	85.8	6.9	38.5	82.3	7.7	30.7	76.7	8.8
VSE++	45.7	89.1	5.5	45.9	82.3	7.2	42.3	80.0	7.6
VSE-C (+n.)	49.2	88.4	5.7	42.1	80.3	9.1	40.4	83.3	7.1
VSE-C (+num.)	**51.0**	**89.5**	6.1	**53.3**	**90.2**	5.8	49.0	87.0	6.6
VSE-C (+rel.)	48.0	88.8	**5.3**	45.4	83.9	6.7	**50.1**	**90.2**	**4.9**
VSE-C (+all.)	49.4	89.3	**5.3**	49.9	89.6	**5.2**	47.9	89.4	5.3

Table 3: Detailed results on each type of adversarial attack. Training VSE-C on one class gains the best performance on robustness against the adversarial attack of the class itself. In addition, training with numeral-typed adversarial samples helps improve the robustness against noun-typed and relation-typed attack. We hypothesis that this is attributed to the singularization or pluralization of the corresponding nouns in the process of numeral-typed adversarial sample generation.

Model	MS-COCO Test (plural split, +plurals)		
	R@1	R@10	Mean r.
VSE++	43.7	78.3	9.1
VSE-C (+num.)	**50.6**	**84.4**	**7.8**

Table 4: Results on plural-typed adversarial attack to the plural split of MS-COCO test set. This split consists of 205 images, together with the 1,025 original captions. VSE-C outperforms VSE++ by a large margin on all the three considered metrics.

singular or plural) of nouns to make a sentence plausible. Does the gain comes from the improved ability to distinguish singulars and plurals?

We conduct the following evaluation to study the counting ability. We extract all the images associated with captions including plurals from our test split of MS-COCO, forming a plural-split of the dataset, and generate only plural (numeral)-typed contrastive adversarial samples (changing the numerals) w.r.t. the plurals in the captions. We report the performance of VSE++ and numeral-typed VSE-C on this plural split in Table 4. It clearly shows that what VSE-C does not only distinguish singulars against plurals, but also, at least, distinguish plurals against other plurals (*e.g.*, 3 vs. 5).

It is worth noting that such counting ability is still not evaluated completely due to the limitation of the current MS-COCO test split. We find that 99.8% of the plurals in MS-COCO test set comes from one of "two", "three", "four" and "five". This may reduce the counting problem to a much simpler classification one.

Figure 2: Saliency analysis on adversarial samples. The left column shows the saliency of VSE-C on the image (what is the difference between the image and the image you imagine from caption c'), while the right column shows the saliency of both VSE++ and VSE-C on the caption (what is the difference between the caption and the caption you summarize from image i). The magnitude of values indicates the level of saliency. For better visualization, the image saliency is L_{inf}-normalized and the caption saliency is L_1-normalized. For all the three classes of textual adversarial samples, the image encoding model (ResNet-152) almost only focuses on the main part of the image, *i.e.*, elephant. For numeral-typed and relation-typed adversarial samples, VSE-C pays much more attention to the manipulated segments of the sentence than VSE++.

4.2 Saliency visualization

Given an image-caption pair and its corresponding textual adversarial samples, we are interested in the following question: what is the semantic distance between the image an adversarial caption? In other words, *which part in the image or caption, in particular, makes them semantically different?*

We visualize the saliency on input images and captions w.r.t. changes in sentence semantics. Specifically, given an image-caption pair (i, c), we manually modify the semantics of the caption c with the techniques introduced in Section 3.2, and obtains $c' \neq c$. We compute the saliency of i or c' w.r.t. this change by visualizing the Jacobian:

$$\mathbf{J} = \nabla_i s(i, c') = \nabla_i W_i^T f(i; \theta_i) \cdot W_c^T g(c'; \theta_c), \tag{4}$$

where $s(i, c')$ is the similarity metric for image-caption pairs.

Shown in Figure 2, as for captions, VSE-C captures the change in sentence semantics and thus possesses large saliency on the manipulated words. In contrast, although trained with hard-negative mining, it is difficult for VSE++ to capture differences other than nouns.

Interestingly, the saliency of images shows less correlated response to semantics changes while the replaced word is not the major component in the image. We attribute this to the image embedding extractor, ResNet, because it is pre-trained on the ImageNet classification task. As the ResNet learns to produce shift-invariant features focusing on the major components (or concepts) of images, it inevitably learns less about secondary (and other) concepts.

4.3 Correlate words and objects

As only textual adversarial samples are provided during the training, the model may overfit the training samples by memorizing incorrect co-occurrence of words or concepts. To quantitatively evaluate the

Image	Captions
	A table with a huge glass vase and fake flowers come out of it. A plant in a vase sits at the end of a table. A vase with flowers in it with long stems sitting on a table with candles. A large centerpiece that is sitting on the edge of a dining table. Flowers in a clear vase sitting on a table.
Positive Objects: table, plant, vase. **Negative Objects**: screen, pickle, sandwich, toy, hill, coat, cat, etc.	

Table 5: An example of the image-to-word retrieval dataset. We extract objects by detecting heads of noun phrases in captions. We only collect the "object" words with frequency higher than 200 in MS-COCO full dataset as available positive/negative objects for each image.

learned word embeddings, we conduct experiments on word-level image-to-word retrieval. Specifically, we first examine how each noun is linked with a visual object. This task shows the concrete link between words and image concepts, which supports the effectiveness of adversarial samples in enforcing the learning of visually-grounded semantics beyond co-occurrence memorizing.

Dataset Based on captions, we extract *positive objects* for each image in MS-COCO dataset by detecting heads of noun phrases using SpaCy. As mentioned in Section 3.2, we only let those objects without direct hypernymy/hyponymy relation to positive objects of the image be *negative objects* to avoid ambiguity. Table 5 shows an example of the preparation of image-object dataset.

Training Inspired by Gong et al. (2018), we train an image-word alignment network through the interaction space, since this structure reflects the property of "alignment" better than just concatenating the feature vectors of word and image. In the training stage, the network is fed by batches of samples in the form of (image, word, label), where the label is 0 or 1, indicating whether the word is a negative or positive object of the image. Let $\mathbf{v}_W(w)$ denote the embedding of word w and $\mathbf{v}_I(img)$ denote the feature vector of image img extracted by ResNet-152 (He et al., 2016). As shown in Figure 3, we use the full interaction matrix $\mathbf{v}_W(w)\mathbf{v}_I(img)^T$ as the feature for object retrieval. While fixing both the image and word features, we only tune the parameters of multi-layer perceptron (MLP).

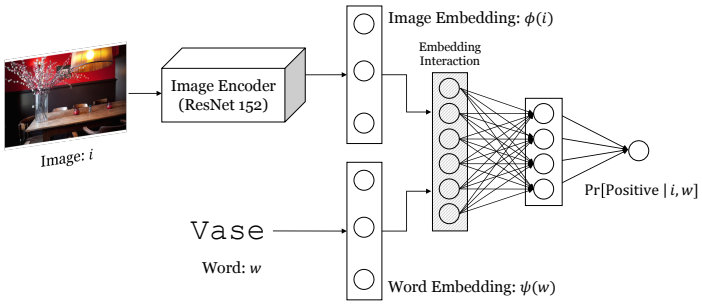


Figure 3: Model structure for image-to-word retrieval. The network is trained though the interaction space. Note that only parameters in MLP are tuned during training.

Model	MAP
GloVe	58.69
VSE	61.65
VSE++	61.08
VSE-C (+all)	62.16
VSE-C (+n.)	**62.80**
VSE-C (+rel.)	62.29
VSE-C (+num.)	61.96

Table 6: Evaluation result (MAP in percentage) on image-to-word retrieval.

Testing We use mean average precision (MAP), which is a widely-applied metric in information retrieval, to evaluate the performance of the word embeddings. For each image, we treat it as a query. The average precision (AP) is defined by

$$AP = \frac{\sum_{k=1}^{n} P(k) \times positive(k)}{\text{number of positive objects}} \tag{5}$$

where n is the quantity of objects in data base, *i.e.*, both positive and negative objects, $P(k)$ is the precision at cut-off k in the list, $positive(k)$ is an indicator function equaling 1 if the object at rank k is a positive one, 0 otherwise (Turpin and Scholer, 2006).

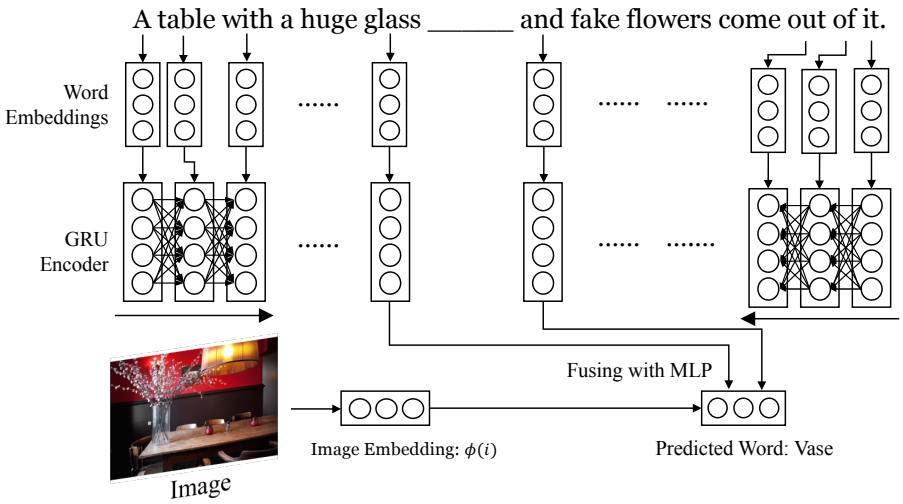


Figure 4: Model structure for fill-in-the-blank.

Based on the definition of AP, MAP can be computed by $MAP = \frac{\sum_{i=1}^{|Q|} AP(i)}{|Q|}$, where Q is the query set, *i.e.*, image set. It is worth noting that the database for retrieval of each query may be different from others, which is similar to Section 4.1.

Results We show the evaluation results in Table 6. It is as expected that VSE-C (+n.) achieves the best performance in the image-object retrieval task. All of the VSE-C models outperform the baselines produced by VSE (Kiros et al., 2014), VSE++ (Faghri et al., 2017) and GloVe (Pennington et al., 2014), showing the concrete link between learned word semantics and visual concepts. With surprise, VSE-C with only relation adversarial samples shows comparable performance as VSE-C with noun adversarial samples. This further supports the effectiveness of sentence-level manipulation (relation-shuffle in Figure 1) on strengthening the link.

4.4 Concept to word

We quantitatively evaluate the performance of concept-to-word retrieval performance by introducing a sentence completion task. Given an image-caption pair (i, c), we manually replace concept words (nouns and relational words) with blanks. A separate model is trained to fill in the blanks.

Dataset and implementation details Based on captions, we extract nouns and relational words from captions for each image in MS-COCO dataset using SpaCy. These selected words are marked as "concept" representatives. During training, we randomly sample a word from the representative set, and the word is masked as a blank to be filled. Given the image and the rest of the words, the model is trained to predict the embedding of the word.

Model The sentences with blank are encoded by two mono-directional GRU layer. The words before the blank and after the blank are separately encoded using GRU_f and GRU_b respectively. The image feature extracted from a pre-trained ResNet152 is then concatenated with the last output of both GRUs. The prediction of embedding is made by a two-layer MLP taking in the concatenated feature. We use cosine similarity as the loss function. Figure 4 shows the demonstration of our fill-in-the-blank model.

Results We present in Table 7 the performance of the proposed VSE-C on filling in both nouns and prepositions. VSE-based models outperform GloVe without visual grounding and concretely correlate word semantics with image embeddings. We found that only small gaps exist between VSE++ and VSE-C on preposition filling, which again shows the limited diversity on visual relations within the dataset.

Model	Noun Filling		Prep. Filling		All (n. + prep.)	
	R@1	R@10	R@1	R@10	R@1	R@10
GloVe	23.22	58.75	23.34	79.92	23.27	66.55
VSE++	24.98	61.74	34.88	84.88	28.44	68.12
VSE-C (ours)	**27.30**	**62.87**	**35.17**	**85.24**	**30.02**	**70.98**

Table 7: Evaluation result on the fill-in-the-blank task (in percentage). The word embeddings learned by VSE-C with all classes of contrastive adversarial samples help reach a better performance than those learned by VSE++ (Faghri et al., 2017).

5 Discussion and conclusion

In this paper, we focus on the problem of learning visually-grounded semantics using parallel image-text data. With extensive experiments on adversarial attacks against existing frameworks (Kiros et al., 2014; Faghri et al., 2017), we obtain new insights on the limitation of datasets as well as frameworks. (1) Even for large-scale datasets such as MS-COCO captioning, the large gap between the number of possible constitutions of real-world visual semantics and the size of dataset still exists. (2) Existing models are not powerful enough to fully capture or extract the information contained in visual embeddings.

We propose VSE-C, introducing contrastive adversarial samples in the text domain and an intra-pair hard-example mining technique. To delve deeper into the embedding space and its transferability, we study a set of multi-modal tasks both qualitatively and quantitatively. Beyond being robust to adversarial attacks on image-to-caption retrieval tasks, experimental results on image-to-word retrieval and fill-in-the-blank reveal the correlation between the learned word embeddings and visual concepts.

VSE-C also demonstrates a general framework for augmenting textual inputs considering semantical consistency. The introduction human priors and knowledge bases alleviates the sparsity and non-contiguity of languages. We hope the framework and the released data are beneficial for building more robust and data-efficient models.

References

Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. 2013. Deep Canonical Correlation Analysis. In *Proc. of ICML*.

Lluis Castrejon, Yusuf Aytar, Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. 2016. Learning Aligned Cross-Modal Representations from Weakly Aligned Data. In *Proc. of CVPR*.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task. In *Proc. of ACL*.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long Short-Term Memory-Networks for Machine Reading. In *Proc. of EMNLP*.

Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2017. VSE++: Improved Visual-Semantic Embeddings. *arXiv preprint arXiv:1707.05612*.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. 2013. Devise: A Deep Visual-Semantic Embedding Model. In *Proc. of NIPS*.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural Language Inference over Interaction Space. In *Proc. of ICLR*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proc. of CVPR*.

Matthew Honnibal and Mark Johnson. 2015. An Improved Non-Monotonic Transition System for Dependency Parsing. In *Proc. of EMNLP*.

Harold Hotelling. 1936. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

Yan Huang, Wei Wang, and Liang Wang. 2017. Instance-Aware Image and Sentence Matching with Selective Multimodal LSTM. In *Proc. of CVPR*.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling. In *Proc. of ICLR*.

Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. In *Proc. of EMNLP*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep Visual-Semantic Alignments for Generating Image Descriptions. In *Proc. of CVPR*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proc. of ICLR*.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2014. Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models. *arXiv preprint arXiv:1411.2539*.

Jernej Kos and Dawn Song. 2017. Delving into Adversarial Attacks on Deep Policies. *arXiv preprint arXiv:1705.06452*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. In *Proc. of ICML*.

Yangyan Li, Hao Su, Charles Ruizhongtai Qi, Noa Fish, Daniel Cohen-Or, and Leonidas J Guibas. 2015. Joint embeddings of shapes and images via cnn image purification. *ACM Trans. Graph.*

Hang Li. 2011. Learning to rank for information retrieval and natural language processing. *Synthesis Lectures on Human Language Technologies*, 4(1):1–113.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *Proc. of ECCV*.

Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. 2015. Ask Your Neurons: A Neural-Based Approach to Answering Questions about Images. In *Proc. of ICCV*.

Junhua Mao, Jiajing Xu, Kevin Jing, and Alan L Yuille. 2016. Training and Evaluating Multimodal Word Embeddings with Large-Scale Web Annotated Images. In *Proc. of NIPS*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

George A Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM*.

Hyeonseob Nam, Jung-Woo Ha, and Jeonghee Kim. 2017. Dual Attention Networks for Multimodal Reasoning and Matching. In *Proc. of CVPR*.

Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. 2011. Multimodal Deep Learning. In *Proc. of ICML*.

Anh Nguyen, Jason Yosinski, and Jeff Clune. 2015. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. In *Proc. of CVPR*.

Zhenxing Niu, Mo Zhou, Le Wang, Xinbo Gao, and Gang Hua. 2017. Hierarchical Multimodal LSTM for Dense Visual-Semantic Embedding. In *Proc. of CVPR*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proc. of EMNLP*.

Scott Reed, Zeynep Akata, Bernt Schiele, and Honglak Lee. 2016. Learning Deep Representations of Fine-Grained Visual Descriptions. In *Proc. of CVPR*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *Proc. of SIGKDD*.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded Compositional Semantics for Finding and Describing Images with Sentences. *TACL*.

Peter D Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and Metaphorical Sense Identification through Concrete and Abstract Context. In *Proc. of EMNLP*.

Andrew Turpin and Falk Scholer. 2006. User Performance versus Precision Measures for Simple Search Tasks. In *Proc. of SIGIR*.

Liwei Wang, Yin Li, and Svetlana Lazebnik. 2016. Learning Deep Structure-Preserving Image-Text Embeddings. In *Proc. of CVPR*.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. 2017. Adversarial examples for semantic segmentation and object detection. In *Proc. of ICCV*.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From Image Descriptions to Visual Denotations: New Similarity Metrics for Semantic Inference over Event Descriptions. *TACL*.

Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. 2013. Bilingual Word Embeddings for Phrase-Based Machine Translation. In *Proc. of EMNLP*.

Arnold M Zwicky. 1985. Heads. *Journal of Linguistics*.

A Semantic Overlap Table of Frequent Prepositions

Set #	Words in a Semantical Set
1	towards, toward, beyond, to
2	behind, after, past
3	outside, out
4	underneath, under, beneath, down, below
5	on, upon, up, un, atop, onto, over, above, beyond
6	in, within, among, at, during, into, inside, from, between
7	if, while
8	with, by, beside
9	around, like
10	to, for, of
11	about, within
12	because, as, for
13	as, like
14	near, next, beside
15	though
16	thru, through
17	besides, along
18	against, next, to
19	along, during, across, while
20	off, out
21	without
22	than
23	before

Table 8: Manually annotated semantic overlap table.

Table 8 shows our manually annotated semantic overlap sets. Prepositions in each row has overlap in semantics, *i.e.*, can be replaced by each other in some level. A preposition can appear in several sets.

B Training Details of VSE-C

In all experiments, we use Adam (Kingma and Ba, 2015) as the optimizer of which the learning rate is set to 1e-3, with the batch size of 128. The learning rate is updated by multiplying 0.1 after every 15 epochs. We do not apply any regularization or dropout term. Word embeddings are initialized with the 300-dimensional GloVe (Pennington et al., 2014)[1]. The text encoder is a bidirectional 512-dimensional (in total 1024D) 1-layer GRU. The dimensionality of joint (multimodal) embedding is also 1,024. Empirically, with training data and hyper-parameters fixed, there is no significant variance in performance caused by different random seeds for the sampling.

[1] `http://nlp.stanford.edu/data/glove.840B.300d.zip`

Structured Representation Learning for Online Debate Stance Prediction

Chang Li **Aldo Porco** **Dan Goldwasser**
Department of Computer Science
Purdue University, West Lafayette, IN 47907
{li1873, aporco, dgoldwas}@purdue.edu

Abstract

Online debates can help provide valuable information about various perspectives on a wide range of issues. However, understanding the stances expressed in these debates is a highly challenging task, which requires modeling both textual content and users' conversational interactions. Current approaches take a collective classification approach, which ignores the relationships between different debate topics.

In this work, we suggest to view this task as a representation learning problem, and embed the text and authors jointly based on their interactions. We evaluate our model over the Internet Argumentation Corpus, and compare different approaches for structural information embedding. Experimental results show that our model can achieve significantly better results compared to previous competitive models.

1 Introduction

In recent years, social media platforms play an increasingly important role in shaping political discourse. Online debate forums allow users to voice their opinions and engage with other users holding different views. Understanding the interactions between the users on these platforms can help provide insight into current political discourse, argumentation strategies and can help gauge public sentiment on policy issues on a large scale. The importance of understanding debate dialog has motivated significant research efforts (Somasundaran and Wiebe, 2010; Anand et al., 2011; Ghosh et al., 2014; Walker et al., 2012b; Hasan and Ng, 2013; Sridhar et al., 2015; Mohammad et al., 2016; Dong et al., 2017).

In this paper, we focus on stance prediction, automatically identifying the stance expressed in debate posts on various issues. For example, Figure 1 describes a short debate dialog about marijuana legalization between three users (denoted a_1, a_2, a_3). The content associated with each user is classified as supportive of legalization (PRO), or not (CON).

Author	Discussion Text	Stance
a_1	*"There are no deaths related to the actual use for marijuana this past year"*	Pro
a_2	*"Whether it kills people or not, it still is harmful to your body."*	Con
a_3	*"There are many things that are harmful. Alcohol is more harmful than marijuana.*	Pro
a_2	*"But that doesn't mean marijuana should be made legal because the other two are."*	Con
a_3	*"If we were to make everything illegal because it was harmful we would be living with nothing."*	Pro

Figure 1: Example of excerpts from a debate between three users about marijuana legalization.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3728–3739
Santa Fe, New Mexico, USA, August 20-26, 2018.

Early work took a text classification approach (Somasundaran and Wiebe, 2010; Anand et al., 2011), classifying individual posts using a rich feature set. Since debate posts are not written in isolation, but rather express the conversational interactions between users, modeling these interactions can help alleviate some of the difficulty of this task. More recent work take a collective classification approach (Hasan and Ng, 2013; Sridhar et al., 2015), which models the dependencies between authors and their content and captures the debate structure. For example, the interactions between users can express *agreements* (or *disagreements*), which would entail a similar (or different) stance prediction associated with their content. The stance decision can also be considered as a user level decision, as users tend to maintain the same stance throughout the debate, forcing stance agreement between all of their posts. Unfortunately, despite these efforts, stance classification remains a challenging problem.

In this paper we suggest a new approach for representing the structural dependencies of debate dialogs, by taking a *structured representation learning* approach. Intuitively, our system is designed to exploit the advantages of collective relational classification methods (often discussed in the context of graphical models) and distributed representation learning (often discussed in the context of deep learning and embedding). We suggest a method for combining the two approaches in a single framework that can exploit their complimentary strengths.

Our key intuition is that the embedding function can be trained to respect the relevant structural dependencies. We jointly embed all the debate objects (i.e., authors, stances and textual posts), by considering the relationships between these objects. For example, we model stance classification as a relationship between a post and a given stance label, by measuring the similarity between their embedded representations. We can also model the relationships between input objects; the similarity between the representations of two posts would entail agreement between the labels associated with them, thus allowing us to perform collective classification over all the input instances. Specifically, we define the factor graph corresponding to the dependencies between stance predictions in a debate thread, and use the similarity between the embedded representation of objects as a scoring function for the factors. We explain this process in more detail in Section 3.

The main strength of distributed representations is in their ability to share information between the represented objects. We exploit this property, and show that by adding additional information to the embedding space, the overall performance of the model improves, *even if this information is not directly relevant to the classification task*. We demonstrate this fact by comparing stance prediction performance, when trained over the multiple topic separately or jointly (thus allowing the model to share information between the representations of multiple debate topics).

We evaluate our approach over the Internet Argument Corpus (Walker et al., 2012a; Abbott et al., 2016), collected from two debate websites,CREATEDEBATE and 4FORUMS. We conduct several experiments, both using in-domain data and out-of-domain data (when we train and test on different debate topics). Our experiments show that formulating the problem as structured representation learning indeed allows debate entities to share information and generalize better, resulting in even larger improvements when multiple stances (corresponding to different output labels) are trained jointly. Furthermore, we show that by using inference over the relationships between the learned representations we can outperform traditional collective classification methods.[1]

Our contributions include (1) joint relational embedding for debate entities, allowing the model to share information between related topics and underlying ideologies (2) suggest a collective classification approach, defined over the embedding space, and using it to cast representation learning as a structured prediction problem, and (3) an extensive experimental study in which we evaluate several different modeling choices and information sharing scenarios.

2 Related Work

Stance prediction in online debates is an important subjectivity classification task. Early work viewed the problem as a binary classification task and focused on feature representations (Somasundaran and Wiebe, 2010; Anand et al., 2011), while later work took a collective approach (Walker et al., 2012b;

[1]Please refer to `https://github.com/BillMcGrady/StancePrediction` for data and source code.

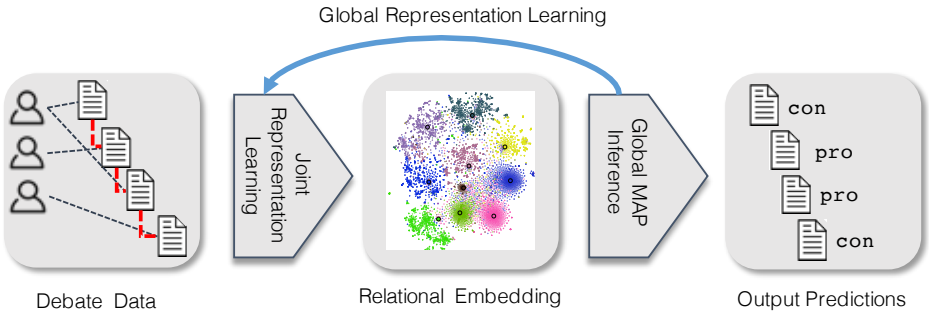


Figure 2: Overall Learning and Inference Processes

Hasan and Ng, 2013; Sridhar et al., 2015). Stance prediction is not limited to online debates, as was also studied in the context of congressional speeches (Bansal et al., 2008; Burfoot et al., 2011) and social media outlets, such as Twitter (Johnson and Goldwasser, 2016; Augenstein et al., 2016; Ebrahimi et al., 2016), including a recent SemEval-16 task (Mohammad et al., 2016). While most work view the task as supervised classification tasks, several work suggest exploiting the interactions between users as a form of distant supervision (Johnson and Goldwasser, 2016; Dong et al., 2017). This task is broadly related to argumentation mining (Ghosh et al., 2014) and stance reason classification (Hasan and Ng, 2014).

Our technical work relies on exploiting distributed representations (i.e., embedding), building on highly influential work on embedding words (Mikolov et al., 2013b; Pennington et al., 2014), sentences (Kiros et al., 2015) and even full documents (Le and Mikolov, 2014). Our work explores the connections between text, users and attributes, attempting to create a common representation for the them. The closest to our work is (Li et al., 2015), that jointly integrates different kinds of cues (text, attribute, graph) into a single latent representation to get user embeddings.

Our work is also broadly related to deep learning methods that capture the structural dependencies between decisions. This can be done either by modeling the dependencies between the hidden representations of connected decisions using RNN/LSTM (Vaswani et al., 2016; Katiyar and Cardie, 2016), or by explicitly modeling the structural dependencies between output predictions (Durrett and Klein, 2015; Lample et al., 2016; Andor et al., 2016). Unlike these work, we formulate our problem as a structured representation learning problem, which to our knowledge is the first work to identify the ties between the two problems.

3 Model Overview

In this paper we suggest casting stance classification as a structured representation learning task. Our approach revolves around two key ideas.

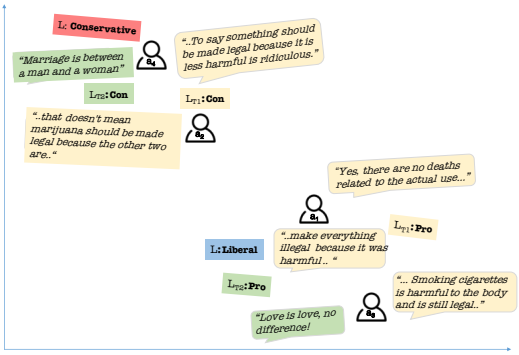
First, stance classification can be done by embedding both the input objects (i.e., posts) and the output labels in the same space. The actual classification is performed by comparing the similarity between the embedded representations of an input object and the competing output labels.

Second, we can augment this representation with additional structural constraints, capturing relevant domain information, such as the connection between posts by the same author, the disagreements between debate participants.

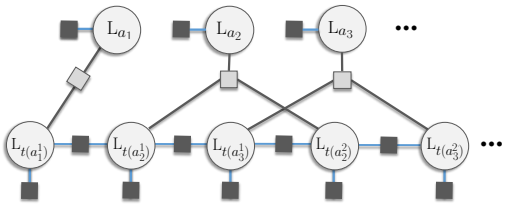
To help clarify these ideas intuitively, consider the debate dialog in Figure 1. Our learning approach uses the structural and textual information in the dialog in three ways, as shown in the process depicted in Figure 2.

1. Joint Representation Learning The embedding learning objective is designed to represent relevant relational information, allowing the representation of different input objects to share information. For example, stances on different topics may share a similar ideology. Figure 3a demonstrates the joint embedding space. The relationship between authors and their posts is preserved by the proximity of their embedded representations. Similar relationships between posts and their corresponding stances and

underlying ideologies are also represented. To accomplish this goal we define a joint objective function over different relations.



(a) Relational embedding representing authors, stances and posts on various topics in the same embedding space



(b) Collective decision over debate thread structure and authors

Figure 3: Model Overview Demonstration

The model is trained to maximize the similarity between corresponding entities (positive examples) compared to irrelevant ones (negative examples). We define the positive examples based on relational information, and negative examples to capture disagreements between authors and posts. We explain this process in detail in Section 5.

2. Global MAP Inference (Collective Classification) Representing the input objects and their labels in the same embedding space allows us to reason about the relationships between them. We view the prediction task as a collective classification, in which all the posts in one or more given debate threads are decided together. We model inference required for the MAP assignment using a factor graph. For example, the graph described in Figure 3b, contains nodes corresponding to author level stance decisions (denoted L_{a_i}), their posts levels stance decision (denoted $L_{t(a_i^j)}$). We score these decisions using the learned embedding. For example, scoring the output assignment PRO to the post corresponding to $L_{t(a_i^j)}$ will be done by observing the similarity (dot product) between their vectorized representations v_{PRO} and $v_{t(a_i^j)}$.

Factor nodes can either have a degree of 1 (scoring the similarity between an author or post and an output label), or 2 (scoring the relationship between consecutive posts in a debate discussion thread). We also allow hard constraints (light gray factors in Figure 3b), which force the model to produce consistent assignments.We explain this process in detail in Section 4.

3. Global Representation Learning A natural extension is to combine the previous two steps, and adopt a global training approach that uses joint prediction during training. In this case the loss function used when learning the embedding is defined with respect to the structural dependencies imposed by the factor graph. This approach is similar in spirit to deep structured learning approaches (Andor et al., 2016), however, in this case the structured learning process is defined directly over the embedding space. This process is explained in Section 5.5.

4 Collective Classification

Our joint embedding model maps authors, attributes, and text into the same space. Thus it allows us to compute the similarity between any pair of authors, texts, attributes, or their combination. This is a very useful property, as information from all aspects can now be used for predicting the target of interest. For example, more information are available for identifying the stance of a post by using its author and neighboring posts comparing to the post's embedding alone. We exploit this property by defining the classification as a global inference process, enforcing the constraints and preferences on all of the predictions.

ILP Formulation We exploit the dependencies described above, using joint prediction over the different aspects. We formulate the decision as an Integer Linear Programming (ILP) which allows us to enforce the consistency or preferences between decisions. The ILP objective function is defined over the similarity scores between objects' vector representation in the joint embedding space. Since integer linear constraints over 0-1 variables can represent logical constraints, we define the ILP constraints using both representations to help improve readability.

In the stance prediction task, all the posts from multiple debate threads that potentially share authors form a single ILP instance. The ILP global optimization objective is defined over authors a_i, the textual content (posts) $\{t_i^0, ..., t_i^k\}$ associated with a_i, and other textual posts $\{t_m^p, ..., t_l^q\}$, responding to or responded by a_i's posts.

We create different types of boolean decision variables corresponding to the decision tasks above. We assign a boolean variable $\texttt{AuthorLabel}(a_i, r_j)$ to represent author a_i has attribute r_j (i.e., its stance), and associate a score $sim(e_{a_i}, e_{r_j})$ with that variable. Similarly, we assign a boolean variable $\texttt{TextLabel}(t_i^k, r_j)$ to represent that the text t_i^k is labeled with an attribute r_j, and associate a score $sim(e_{t_i^k}, e_{r_j})$ with that variable.

To ensure the consistency of the predicted variables, we define two types of constraints.

1. *Single output value on a debate topic*:

$\forall i \ \sum_j \texttt{AuthorLabel}(a_i, r_j) = 1$

$\forall i, k \ \sum_j \texttt{TextLabel}(t_i^k, r_j) = 1$

2. *Output consistency*:

$\forall i, j, k \ \ \texttt{AuthorLabel}(a_i, \ r_j) = \texttt{TextLabel}(t_i^k, \ r_j)$

Note that in the debate domain, this constraint forces agreement between all the posts by the same author.

We also add variables capturing the dependencies between connected posts. For debate threads, a boolean variable $\texttt{Disagree}(t_i^p, t_l^q)$ is created for any two posts t_i^p, t_l^q when t_l^q is a response to t_i^p, and associate a score $disagree\_parameter$ with that variable. This score is a hyper-parameter for local models, capturing the preference towards disagreement between consecutive posts in a debate. It is set according to the training set. When using global learning, it is also included in training, such that similarity scores of consecutive posts will be adjusted appropriately (similar intuition as a margin constraint).

$$\forall t_i^p, t_l^q \ \texttt{Disagree}(t_i^p, t_l^q) \wedge \texttt{TextLabel}(t_i^p, r_j) \rightarrow \neg \texttt{TextLabel}(t_l^q, r_j)$$

The set of all possible decisions for the three set of variables are denoted as A for $\texttt{AuthorLabel}$, B for $\texttt{TextLabel}$, Γ for $\texttt{Disagree}$.

Given these variables, our prediction function can be define as follows -

$$\arg\max_{\alpha,\beta,\gamma} \sum_{\alpha \in A} \alpha \cdot score(\alpha) + \sum_{\beta \in B} \beta \cdot score(\beta) + \sum_{\gamma \in \Gamma} \gamma \cdot score(\gamma)$$

Subject To **C**

Where **C** is a set of constraints defined above.

5 Representation Learning

5.1 Embedding Perspectives

Let A and T denote the set of all authors and text respectively, let R denote the set of all attributes for those authors and text. Stance on various topics are the major attributes considered in this paper. For each topic, we have an embedding vector for the Pro stance and another vector for the Con stance, such as $Pro_{abortion}$ and $Con_{abortion}$. We train our embedding over multiple views of the data, each view connecting users and their content.

Author vs. Text: This objective is to predict text t_j linked with author a_i given the author representation. Each post is a text unit in our experiments.

$$L_{AT} = \sum_{i=1}^{n} \sum_{j=1}^{text_{a_i}} logP(t_j|a_i) \tag{1}$$

Author vs. Attribute: This objective is to predict attribute r_j linked with author a_i given the author representation. Stance on different topics and user profile information form the attributes set in debate datasets. Each user attribute value (e.g. male or female in gender attribute) is represented by a vector.

$$L_{AR} = \sum_{i=1}^{n} \sum_{j=1}^{attri_{a_i}} logP(r_j|a_i) \tag{2}$$

Text vs. Attribute: This objective is to predict attribute r_j of the text given text t_i. In our experiments, we only used the stance label as attributes of text. However, it may also be possible to inherit attributes from the author of the text.

$$L_{TR} = \sum_{i=1}^{m} \sum_{j=1}^{attri_{t_i}} logP(r_j|t_i) \tag{3}$$

Text vs. Text: This objective is to predict text t_j given the text t_i that share the same attribute. It is used to promote similarity between posts sharing the same stance on a certain topic.

$$L_{TT} = \sum_{i=1}^{m} \sum_{j=1}^{text_{t_i}} logP(t_j|t_i) \tag{4}$$

All the conditional probabilities can be computed using a softmax function. Taking $P(t_j|a_i)$ as an example:

$$P(t_j|a_i) = \frac{exp(e_{a_i}^T e_{t_j})}{\sum_{k \in T} exp(e_{a_i}^T e_{t_k})}$$

5.2 Embedding Initialization

In our model, the embedding for each author and attribute can be randomly initialized. The text is a special case since there are complex structures involved. One way to capture this is to use an pre-trained text embedding model to get an initial representation, and then learn an neural network to map it to one in the shared space. Note that this also allows our model to generate embedding for unseen text in the new space.

Specifically, for a text input x, we can compute its embedding e using M hidden layers $l_i, i = [0, M-1]$. The first hidden layer l_0 is computed from the input x:

$$l_0 = f(W_0 x + b_0)$$

Subsequent layers are computed recursively:

$$l_i = f(W_i l_{i-1} + b_i), i = 1, ..., M-1$$

Then the output from the final layer produces the embedding:

$$e = l_{M-1}$$

f is the non-linear activation function. We used hyperbolic tangent (tanh) in our experiments.

Note that our model offers the flexibility to use more complex neural network structures, including CNN and RNN, to learn a mapping from the initial word embedding sequences of the text to an embedding in the joint space.

5.3 Joint Embedding Learning

Our objective is to learn a semantic embedding for authors, text and attributes associated with them so that they are close in the embedding space if they are semantically close to each other.

Joint Embedding Loss Function: We can combine these embedding losses from Equations 1-4 into a joint training objective:

$$L_{Joint}(A, T, R) = \sum_{i \in (AT, AR, TR, TT)} \lambda_i L_i \tag{5}$$

where λ_i is the coefficient for each view, indicating the relative importance in the loss function. We set all λ_i to the default value 1 in all our experiments.

This is the general framework. Additional views may be added or removed for a certain dataset. For example, we can add a term representing the author vs. author view in the loss function if links between them are available.

5.4 Model Optimization

We train our model using mini-batch Adam optimizer to minimize the loss in Eq.5. However, computing gradient for Eq.1, and Eq.4 is expensive due to the size of the authors or text. To address this problem, we refer to the popular negative sampling approach (Mikolov et al., 2013a), which reduce the time complexity to be proportional to the number of positive example pairs.

5.5 Global Embedding Learning

Although different views of the data are captured in our joint loss function, it does not ensure that the information they provide at inference time will be "cooperative", i.e., it will result in consistent global prediction over all the debate outputs. One potential problem is that examples associated with one view will dominate the training and skew the prediction, when constraints are applied. To handle this issue, we included the inference procedure during training. Instead of making sure the loss for each local view is minimized, the global objective promotes the rank of all the gold predictions *jointly*. For instance, at training, posts, together with their author and neighboring posts (if available) are used to infer their stance based on the inference procedure described in section 4. Then structured hinge loss can be used to define the prediction loss as in Eq.6.

$$L_{pred} = \sum_{i \in instances} max(0, max_{y \in Y}(\Delta(y, t_i) + score(x_i, y)) - score(x_i, t_i)) \tag{6}$$

where x_i and t_i are the problem instances and corresponding gold predictions, Y denotes all possible predictions, and score$(\cdot)$ is the inference score function. $\Delta(\cdot, \cdot)$ is the hamming loss. It measures the difference between two predictions and is used to create a margin between gold and other predictions.

The loss function used for updating the parameters in the global model is defined as follows.

$$L_{Global} = \lambda_{pred} L_{pred} + \lambda_{AT} L_{AT} + \lambda_{TT} L_{TT} \tag{7}$$

The coefficients for different terms in the global loss function can adjust the contribution of prediction objective (L_{pred}) versus information sharing objective (L_{AT} and L_{TT}). Again, we set all λ to the default value 1 in experiments of this paper. We leave the exploration of different coefficient settings to future work. Since the inference is used during global training, the scores for text stance and author stance are part of the inference scores. So they will get updated according to L_{pred}. Therefore we do not include L_{AR} and L_{TR} explicitly in the global loss function.

In our experiments, a mini-batch of debate threads is regarded as an instance during training. To reduce the computational cost, we used the parameters learned with the joint embedding loss function as the starting point for the global training.

Dataset	Topic	Posts	Users
CREATEDEBATE	Abortion	1741	340
	Gay Rights	1376	370
	Marijuana	626	258
	Obama	985	278
4FORUMS	Abortion	7937	342
	Evolution	6069	311
	Gay Marriage	6897	296
	Gun Control	3755	281

Table 1: Data Statistics for 4FORUMS and CREATEDEBATE

6 Experiments

To evaluate the different properties of our model and demonstrate their advantages, we evaluate the quality of our structured embedding model on two datasets 4FORUMS and CREATEDEBATE for stance classification tasks at post level, consisting of eight topics in total. The datasets are taken from the Internet Argumentation Corpus (Abbott et al., 2016). Table 1 shows statistics about these datasets.

Experimental Design Our experiments are designed to evaluate the different properties of our model. To accomplish that, we compare different variations of the model, corresponding to (1) only the joint embedding (denoted *Joint*), (2) using inference at test time, over the joint embedding (denoted *Inference*), and (3) using global training (denoted *Global*), which also uses inference during training. We report the results of these experiments in the two datasets in Tables 2 and 3.

Our second set of experiments are designed to evaluate the joint embedding model's ability to share information between the representations of different objects. In this case we compare the performance of the *Joint* and *Inference* models when additional information is available. We compare three settings (1) In-Domain, when the available training and testing data are from the same domain (2) In+Out Domain, where we augment the In-Domain training data with additional debate threads from other topics. In this case the model can represent the relationships between stances on different topics and potentially generalize better. Finally, (3) User-Attribute, where we augment the author attributes with profile information extracted from the debate website. We conduct all of these experiments over the CREATEDEBATE dataset, and report the results in Table 4.

6.1 Experimental Settings:

We used PyTorch (Paszke et al., 2017) to implement the embedding model and Gurobi (Gurobi Optimization, 2016) as our ILP solver. Each debate post is initially represented using the Skip-Thought Vectors (Kiros et al., 2015), and then mapped to an embedding in the shared space through one hidden layer. We do not add more layers as both datasets are relatively small. Hyperbolic tangent (tanh) is used as non-linear activation function. All other embeddings are randomly initialized following a normal distribution with variance $1/\sqrt{dim}$. The embedding size dim for all experiments is 300. For the training of the neural network, we used mini-batch Adam optimizer to update parameters. Dropout with probability 0.7 is used as regularization. The termination criteria is convergence on training loss. Five epoch of non-improvement on loss is considered as convergence for joint models, and one epoch for global models. Other parameters in our model includes negative sample size k=5, mini-batch size b=10.

6.2 Results

Our results on stance classification are described in Table 2 and Table 3. The results are computed using 5-fold cross-validation. For the CREATEDEBATE dataset, We used the same five data folds as in (Hasan and Ng, 2013) to ensure our results are directly comparable with theirs. For the 4FORUM dataset, we randomly divided debate threads into five folds since the data split is not available in (Sridhar et al., 2015). We regarded the same user in the training and test folds as different ones to avoid leaking label

Model	Abortion	Gay Rights	Marijuana	Obama	Average
Majority	56.2	64.5	72.0	56.1	62.2
NB (Hasan and Ng, 2013)	73.3	67.0	72.4	67.0	70.0
CRF (Hasan and Ng, 2013)	74.7	69.9	75.4	**71.1**	72.8
PSL (Sridhar et al., 2015)	66.8	72.7	69.1	63.7	68.1
Joint	62.1	63.1	69.2	57.4	63.0
Inference(AC)	70.4	62.7	66.3	62.2	65.4
Inference(Consecutive)	67.2	65.0	66.8	61.0	65.0
Inference(Both)	**81.1**	75.6	75.0	64.7	74.1
Global	81.0	**77.2**	**77.6**	64.8	**75.2**

Table 2: Average Accuracy on CREATEDEBATE dataset

Model	Abortion	Evolution	Gay Marriage	Gun Control	Average
Majority	56.8	65.8	66.0	67.8	64.1
PSL (Sridhar et al., 2015)	77.0	80.3	80.5	69.1	76.7
Joint	64.1	67.2	68.5	66.5	66.6
Inference(AC)	72.9	66.8	68.6	68.4	69.2
Inference(Consecutive)	67.5	67.7	72.3	69.6	69.3
Inference(Both)	85.9	80.9	**88.1**	81.6	84.1
Global	**86.5**	**82.2**	87.6	**83.1**	**84.9**

Table 3: Average Accuracy on 4FORUMS dataset

information from training to test. *NB* and *CRF* stands for the best local and collective models in (Hasan and Ng, 2013). Note that their system also uses the author constraints, as well as a highly engineered feature set and additional weakly-supervised data that we did not use. Despite that fact, our global model significantly outperforms their model with the exception of Obama domain. In the 4FORUM experiments we compare our models to result with *PSL* based model (Sridhar et al., 2015) which performs similar collective classification defined over a feature-rich representation. In this case as well, our Global model achieves the best overall performance.

We evaluate the contribution of two constraints sets, Author constraints (*AC*) enforce author and their posts share the same stance. *Consecutive* will encourage disagreement in stance between neighboring posts as introduced in section 3. The addition of these two constraints leads to significant increase in performance when used together. This is because AC and Consecutive add agreement and disagreement constraints between test posts, grouping them into clusters and making it easier to be predicted correctly. For instance, given multiple posts from the same author, the model can make correct decisions on all of them even if the prediction based on each individual post may be wrong. Finally, we observe that structured representation learning (i.e., Global) leads to a performance improvement compared to inference over the joint embedding objective (Inference). This shows the effectiveness of the global learning.

Table 4 shows the result on CREATEDEBATE when additional information is available. We extracted user profile information (User-Attribute) from the website[2], consisting of five attributes (Gender, Marital Status, Political Party, Religion, and Education). Clearly richer user information results in a better representation, both for users and as a result, also for the text they author, leading to improved performance. A similar trend occurs when out-of-domain data is available (In+Out Domain). Stances over different debate topics impact the text and author representations. Interestingly, when this data is available, our model is able to outperform the collective approach of (Hasan and Ng, 2013) in all debate topics, showing that our model can indeed exploit the information shared by the underlying ideologies.

[2] www.createdebate.com

	Model	Abortion	Gay Rights	Marijuana	Obama	Average
	CRF	74.7	69.9	75.4	71.1	72.8
In-Domain	Joint	62.1	63.1	69.2	57.4	63.0
	Inference	**81.1**	75.6	75.0	64.7	74.1
In-Domain + User-Attribute	Joint	63.9	63.6	69.5	59.9	64.2
	Inference	81.0	80.0	75.6	65.7	75.6
In+Out Domain	Joint	63.0	62.9	70.3	61.8	64.5
	Inference	79.0	74.5	**77.1**	76.2	76.7
In+Out Domain + User-Attribute	Joint	63.3	65.3	71.0	57.7	64.3
	Inference	79.9	**80.2**	75.1	**77.4**	**78.2**

Table 4: Average Accuracy on CREATEDEBATE dataset with additional information

7 Conclusions and Future Work

We study the problem of stance prediction, a challenging text classification problem, which requires connecting textual content analysis, conversational interactions and author information. Traditionally, this is done using a graphical model, which learns a scoring function for each aspect, over a fixed feature representation. We follow the observation that all of these problems are connected, and allow the model to capture these dependencies by allowing it to *learn* a representation for all these aspects jointly, rather than using a fixed representation. We show that by formulating the decision problem over the representation directly, and requiring the representation to respect the global dependencies between these aspects, our model can generalize better and exploit additional information even when it is not directly relevant.

To the best of our knowledge, this work is the first to cast representation learning as a structured prediction problem, we believe that this approach is applicable to many other domains where the input has complex inter-connected structure. Such domains can include other conversation analysis tasks, shared representations of text and images and information networks such as citations graphs and social network analysis. In the future we intend to study additional domains and evaluate whether including additional aspects, can help provide better generalization. Providing sufficient supervision is one of the main bottlenecks of NLP, we intend to apply our approach in weakly and distantly supervised settings, to help alleviate this difficulty.

References

Rob Abbott, Brian Ecker, Pranav Anand, and Marilyn A Walker. 2016. Internet argument corpus 2.0: An sql schema for dialogic social media and the corpora to go with it. In *Proc. of the International Conference on Language Resources and Evaluation (LREC)*.

Pranav Anand, Marilyn Walker, Rob Abbott, Jean E Fox Tree, Robeson Bowmani, and Michael Minor. 2011. Cats rule and dogs drool!: Classifying stance in online debate. In *Proceedings of the 2nd workshop on computational approaches to subjectivity and sentiment analysis*, pages 1–9. Association for Computational Linguistics.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, pages 2442–2452.

Isabelle Augenstein, Tim Rocktschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In *Conference on Empirical Methods in Natural Language Processing*, pages 876–885.

Mohit Bansal, Claire Cardie, and Lillian Lee. 2008. The power of negative thinking: Exploiting label disagreement in the min-cut classification framework. *COLING 2008: Companion Volume: Posters*, pages 15–18.

Clinton Burfoot, Steven Bird, and Timothy Baldwin. 2011. Collective classification of congressional floor-debate transcripts. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, pages 1506–1515. Association for Computational Linguistics.

Rui Dong, Yizhou Sun, Lu Wang, Yupeng Gu, and Yuan Zhong. 2017. Weakly-guided user stance prediction via joint modeling of content and social interaction. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1249–1258. ACM.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, pages 302–312.

Javid Ebrahimi, Dejing Dou, and Daniel Lowd. 2016. Weakly supervised tweet stance classification by relational bootstrapping. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Debanjan Ghosh, Smaranda Muresan, Nina Wacholder, Mark Aakhus, and Matthew Mitsui. 2014. Analyzing argumentative discourse units in online interactions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 39–48.

Inc. Gurobi Optimization. 2016. Gurobi optimizer reference manual.

Kazi Saidul Hasan and Vincent Ng. 2013. Stance classification of ideological debates: Data, models, features, and constraints. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1348–1356.

Kazi Saidul Hasan and Vincent Ng. 2014. Why are you taking this stance? identifying and classifying reasons in ideological debates. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 751–762.

Kristen Johnson and Dan Goldwasser. 2016. " all i know about politics is what i read in twitter": Weakly supervised models for extracting politicians stances from twitter. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2966–2977.

Arzoo Katiyar and Claire Cardie. 2016. Investigating lstms for joint extraction of opinion entities and relations. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, pages 919–929, Berlin, Germany, August.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 3294–3302.

Guillaume Lample, Miguel Ballesteros, Kazuya Kawakami, Sandeep Subramanian, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 1–10.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Jiwei Li, Alan Ritter, and Dan Jurafsky. 2015. Learning multi-faceted representations of individuals from heterogeneous evidence using neural networks. *CoRR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 116–124. Association for Computational Linguistics.

Dhanya Sridhar, James Foulds, Bert Huang, Lise Getoor, and Marilyn Walker. 2015. Joint models of disagreement and stance in online debate. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, volume 1, pages 116–125.

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertagging with lstms. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 232–237.

Marilyn A Walker, Rob Abbott, and Joseph King. 2012a. A corpus for research on deliberation and debate. In *Proc. of the International Conference on Language Resources and Evaluation (LREC)*.

Marilyn A Walker, Pranav Anand, Robert Abbott, and Ricky Grant. 2012b. Stance classification using dialogic properties of persuasion. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 592–596. Association for Computational Linguistics.

Modeling Multi-turn Conversation with Deep Utterance Aggregation

Zhuosheng Zhang[1,2,*], **Jiangtong Li**[1,2,3,*], **Pengfei Zhu**[1,2,5], **Hai Zhao**[1,2,†], **Gongshen Liu**[4]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
[3]College of Zhiyuan, Shanghai Jiao Tong University, China
[4]School of Cyber Security, Shanghai Jiao Tong University, China
[5]School of Computer Science and Software Engineering, East China Normal University, China
{zhangzs, keep_moving-lee}@sjtu.edu.cn, 10152510190@stu.ecnu.edu.cn,
zhaohai@cs.sjtu.edu.cn, lgsheng@sjtu.edu.cn

Abstract

Multi-turn conversation understanding is a major challenge for building intelligent dialogue systems. This work focuses on retrieval-based response matching for multi-turn conversation whose related work simply concatenates the conversation utterances, ignoring the interactions among previous utterances for context modeling. In this paper, we formulate previous utterances into context using a proposed deep utterance aggregation model to form a fine-grained context representation. In detail, a self-matching attention is first introduced to route the vital information in each utterance. Then the model matches a response with each refined utterance and the final matching score is obtained after attentive turns aggregation. Experimental results show our model outperforms the state-of-the-art methods on three multi-turn conversation benchmarks, including a newly introduced e-commerce dialogue corpus.

1 Introduction

Human-computer interactive systems are booming due to their promising potentials and alluring commercial values (Qiu et al., 2017; Cui et al., 2017; Yan et al., 2017; Huang et al., 2018; Jia and Zhao, 2014). With the development of neural models (Zhang et al., 2018c; He et al., 2018; Li et al., 2018; Cai et al., 2018; Zhang and Zhao, 2018), building an intelligent dialogue system as our personal assistant or chat companion, is no longer a fantasy, among which multi-turn natural language understanding still keeps extremely challenging, requiring the system to comprehend the conversation context and reply in an informative and coincident manner.

Multi-turn conversation modeling plays a key role in dialogue systems, either for generation-based (Serban et al., 2017b; Serban et al., 2017a; Zhou et al., 2017; Wu et al., 2018) or retrieval-based ones (Wu et al., 2017; Zhou et al., 2016) in which the latter is the focus of this paper. A natural approach for multi-turn modeling is simply concatenating the context utterances (Lowe et al., 2015; Yan et al., 2016). However, this will introduce much noise since previous utterances as the context is lengthy and redundant. The gist is to identify pertinent information in previous utterances and properly model the utterance relationships to ensure conversation consistency. To avoid unnecessary information loss, (Wu et al., 2017) matches a response with each utterance in the context, paying little attention on distrinct importance of each utterance and also failing to touch internal semantics inside utterances.

In fact, the relevance of each utterance to the supposed response usually varies. As shown in Figure 1, the last utterance in a conversation empirically conveys the user intention while the other utterances depict the conversation in different aspects [1]. Thus, instead of considering all the conversation turns

*These authors contribute equally. † Corresponding author. This paper was partially supported by National Key Research and Development Program of China (No. 2017YFB0304100), National Natural Science Foundation of China (No. 61672343 and No. 61733011), Key Project of National Society Science Foundation of China (No. 15-ZDA041), The Art and Science Interdisciplinary Funds of Shanghai Jiao Tong University (No. 14JCRZ04).

[1]For a multi-turn conversation, we define the latest user utterance (or called current message) as the *last utterance*, which is waiting for a response.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3740–3752
Santa Fe, New Mexico, USA, August 20-26, 2018.

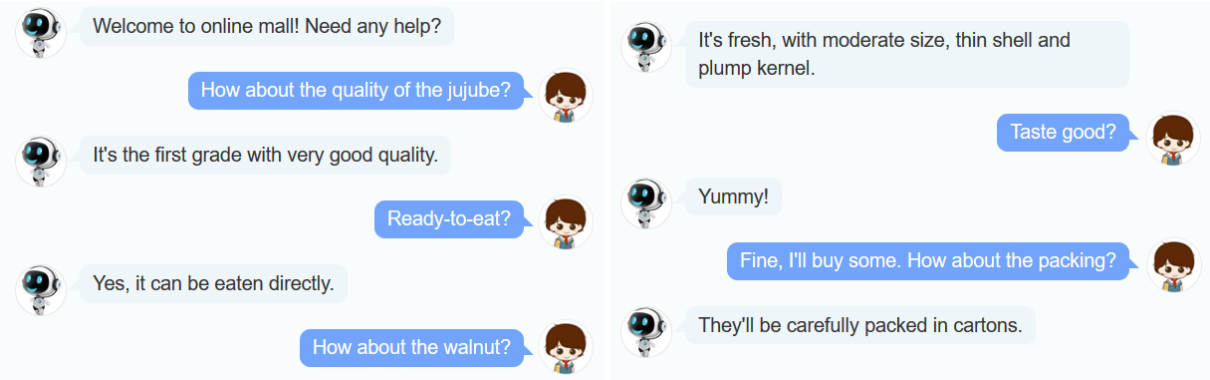


Figure 1: An example of E-commerce Dialogue Corpus.

equally, we have to weigh previous conversations in a more sophisticated way. With a turns-aware aggregation design, our model alleviates the drawback of previous work.

In addition, words in an utterance also hold different importance to the whole utterance representation. Our solution is to employ attention-based recurrent networks on each utterance against utterance itself, aggregating the vital pieces of the contextual utterances.

Finally, in conjunction with this paper, we release an E-commerce Dialogue Corpus (ECD) to facilitate the related studies. To our best knowledge, this is the first public e-commerce dataset for dialogue system development that is extracted from real human conversations. Different from previous datasets that only focus on a single type of dialogue like chitchat, this dataset is more comprehensive due to diverse types of conversations (e.g. *commodity consultation, logistics express, recommendation, negotiation* and *chitchat*) concerning various commodities. Our improved retrieval-based multi-turn dialogue response matching model is evaluated on three benchmark datasets, including our newly released one, giving state-of-the-art performance.

The rest of this paper is organized as follows. The next section reviews related work. Our proposed model is introduced in Section 3, then the experiments and analysis are reported in Section 4, followed by the conclusion in Section 5.

2 Related Work

With the impressive success of various referential natural language processing studies (Zhang et al., 2016; Cai and Zhao, 2017; Zhang et al., 2018d; Qin et al., 2017; Zhang et al., 2018b; Bai and Zhao, 2018), developing an intelligent dialogue system becomes realizable, which means training machines to converse with human in natural languages (Williams et al., 2017; He et al., 2017; Dhingra et al., 2017; Zhang et al., 2018a). Towards this end, a number of data-driven dialogue systems are designed (Lowe et al., 2015; Wu et al., 2017; Wen et al., 2017; Mei et al., 2017; Young et al., 2018; Lipton et al., 2018), in which modeling multi-turn conversation has drawn more and more attention. To acquire a contextual response, previous utterances are taken as input. Lowe et al. (2015) concatenated all previous utterances and last utterance as the context representation and then computed the matching degree score based on the context representation to encode candidate response. Yan et al. (2016) selected the previous utterances in different strategies and combined them with last utterance to form a reformulated context. Zhou et al. (2016) performed context-response matching with a multi-view model on both word level and utterance level. Wu et al. (2017) improved the leveraging of utterances relationship and contextual information by matching a response with each utterance in the context based on a convolutional neural network.

Different from previous studies, our model for the first time discriminates the importance of previous conversations and also accumulates substantial parts from each utterance according to each word in the utterance itself in a multi-turn scenario.

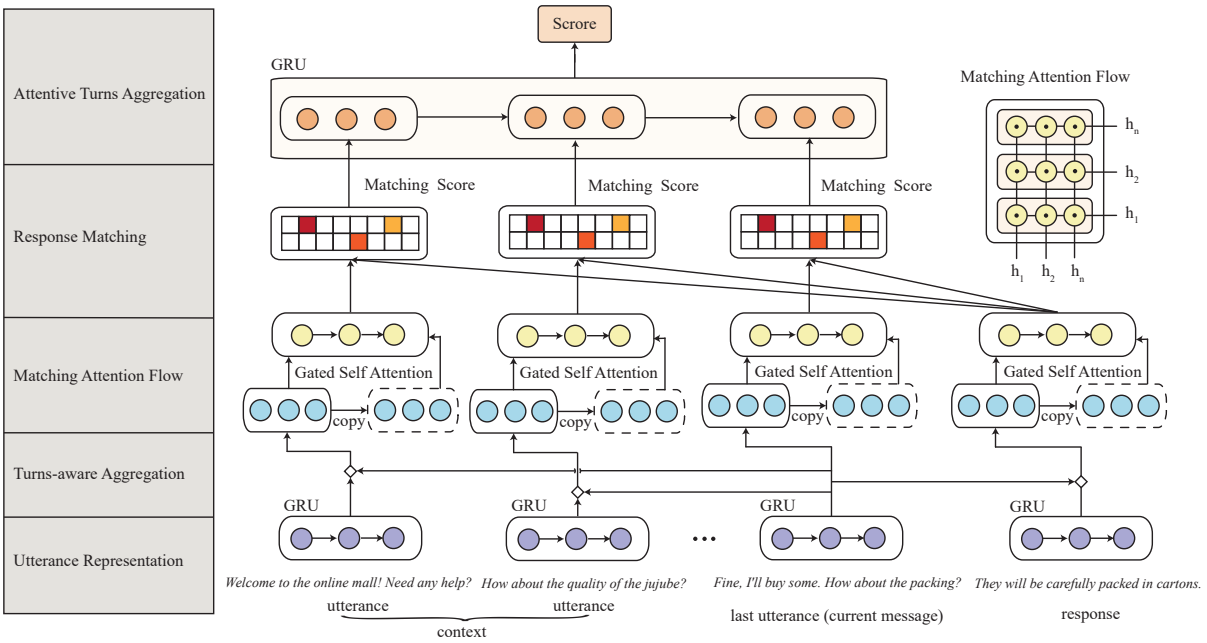


Figure 2: Structure overview of the proposed dialogue system.

3 Deep Utterance Aggregating Strategy

Each conversation in the concerned multi-turn response retrieval task can be described as a triple $< C, R, Y >$. $C = \{U_1, ..., U_t\}$ is the conversation context where $\{U_k\}$ denotes the k-th utterance. R is a response of the conversation and Y belongs to $\{0, 1\}$, where $Y_i = 1$ means the response is proper, otherwise $Y_i = 0$. The aim is to build a discriminator $\mathcal{F}(\cdot, \cdot)$ on $< C, R, Y >$. For each context-response pair $\{C, R\}$, $\mathcal{F}(C, R)$ measures the matching score of the pair.

In this section, we will introduce our Deep Utterance Aggregation (DUA) model for the multi-turn conversation task. Figure 2 shows the architecture. DUA formulizes utterances into the context and mines the key information from utterances and response. Then DUA conducts semantic matching between each utterance and the response candidate to obtain a matching score. Specifically, there are five modules within DUA. Each utterance or response is fed to the first module to form an utterance or response embedding. The second module combines the last utterance with the preceding utterances. Then, the third module filters the redundant information and mines the salient feature within the utterances and response. The fourth module matches the response and each utterance at both word and utterance levels to feed a Convolutional Neural Network (CNN) for encoding into matching vectors. In the last module, the matching vectors are delivered to a gated recurrent unit (GRU) (Cho et al., 2014) in chronological order of the utterances in the context and the final matching score of $\{U, R\}$ is obtained.

DUA is superior to existing models in the following ways. First, the last utterance which is the most important in dialogue is especially fused within preceding utterances, thus the key guideline information from the last utterance can be handled in a more semantically pertinent way. Second, in each utterance, the salient information can be highlighted and those redundant pieces will be neglected to some extent, both of which can effectively guide the later response matching. Third, after attentive turns aggregation, the connections in the conversation are accumulated again to calculate the matching scores.

3.1 Utterance Representation

To use deep neural networks, symbolic data needs to be transformed into distributed representations, namely, word embedding (Bengio et al., 2003; Mikolov et al., 2013). Given a context-response pair, $\{C, R\}$ whose context are split into utterances, $C = \{U_1, ..., U_t\}$, a lookup table is used to map each word into a low-dimensional vector. Let n_u and n_r denote the length of the k-th utterance and response, U_k and R can be represented as $U_k = [u_1, ..., u_{n_u}]$ and $R = [r_1, ..., r_{n_r}]$, where u_i, r_i are the i-th word

in the utterance and response respectively.

To encode each utterance and response, we employ a GRU to propagate information along the word sequence of U_k and R. Suppose $H_k = [h_1, ..., h_n]$ is the hidden states of the input sequence, the structure of GRU is described as follows.

$$\begin{aligned} z_i &= \sigma(W_z u_i + V_z h_{i-1}) \\ r_i &= \sigma(W_r u_i + V_r h_{i-1}) \\ \tilde{h_i} &= tanh(W_h u_i + V_h(r_i \odot h_{i-1})) \\ h_i &= z_i \odot \tilde{h_i} + (1 - z_i) \odot h_{i-1} \end{aligned} \tag{1}$$

where $\sigma(\cdot)$ is the sigmoid function, z_i and r_i are the update and reset gates respectively, $\odot$ denotes the element-wise multiplication, and $W_z, W_r, W_h, V_z, V_r, V_h$ are parameters. We fed each utterance and response sequence to the GRUs and obtain the utterance representation S_k and response representation S_r, respectively.

3.2 Turns-aware Aggregation

Encoding the utterance sequence and response in the above way, there comes a drawback that all the utterances in the conversation are fairly dealt with, which fails to mine the connections between the last utterance and the rest preceding utterances. Thus, a first-stage turns-aware aggregation mechanism is proposed to address this problem.

Let $S = [S_1, ..., S_t, S_r]$ denote the representation of the utterances and response. Suppose $F = [F_1, ..., F_t, F_r]$ is the fusion of each $S_j \in S$ with the last utterance S_t, for each $\forall j \in \{1, ..., r\}$, $F_j \in F$, we define the fusion of the utterance as

$$F_j = S_j \diamond S_t \tag{2}$$

where $\diamond$ denotes the aggregation operation. In this work, we adopt a simple concatenation strategy[2]. So far, the turns-aware representation F is obtained via aggregation.

3.3 Matching Attention Flow

After turns-aware aggregation, the representations of the preceding utterances and response have been refined by the last utterance. However, the sequences are quite lengthy and redundant, which makes it hard to distill the pivotal information. In order to address this problem, we adopt a self-matching attention mechanism to directly match the fused representation against itself, which is similar as that adopted in (Wang et al., 2017). It dynamically collects information from the input sequence and filters the redundant information. Suppose $\hat{F} = [f_1, ..., f_n] \in F$ is the input and $P = [p_1, ..., p_n]$ is the output of the self-matching attention on response, then $\forall t$, $p_t \in P$ is defined as

$$p_t = GRU(p_{t-1}, [f_t, c_t]) \tag{3}$$

where $GRU(\cdot, \cdot)$ denotes the same calculation as (2), $[\cdot, \cdot]$ is the concatenation of two vectors and $c_t = att(\hat{F}, f_t)$ is the result of the self-matching attention. $\forall t$, $f_t \in \hat{F}$, c_t is defined as

$$\begin{aligned} s_j^t &= v^T tanh(W_v f_j + W_{\tilde{v}} f_t + b_r) \\ a_i^t &= exp(s_i^t) / \sum_{j=1}^{n} exp(s_j^t) \\ c_t &= \sum_{i=1}^{n} a_i^t f_i \end{aligned} \tag{4}$$

[2]We empirically investigated *concatenation*, *element-wise summation*, *element-wise multiplication* in this work and $concatenation$ strategy shows the best performance.

where W_r, $W_{\tilde{v}}$, b_r are the parameters and v^T is a context matrix which is randomly initialized and jointly trained.

Self-matching attention pinpoints important parts from the utterance according to the current word and the whole utterance representation through fusing each previous utterance and the last utterance.

3.4 Response Matching

Following (Wu et al., 2017), we use word-level and utterance-level representations to build two matching matrices and employ CNN to obtain salient matching information from the matrices. Suppose we have matching matrices M_1 and M_2 in word-level and utterance-level for each utterance-response pair. Then, $\forall k$, $U_k \in U$ and $\forall(i,j)$, the (i,j)-th element of M_1 and M_2 is defined respectively

$$e_{1,i,j} = u_i^T r_j \tag{5}$$

$$e_{2,i,j} = P_{u_i}^T A P_{r_j} \tag{6}$$

where P_{u_i} and P_{r_j} denote the outputs of the utterance and response after *Matching Attention Flow* respectively. $A \in \mathbb{R}^{c \times c}$ is a linear transforming matrix.

A convolutional operation followed by a max-pooling operation will be applied to M_1 and M_2 for each utterance. The convolutional layer is used to extract and combine local features from adjacent words and the following max-pooling layer forms the representations for the current word. For the convolutional operation, a group of filter matrices K with variable sizes $l*l$ and bias b are utilized. The filter transforms the word matrices M_1 and M_2 to another two matrices M_{1c} and M_{2c}. $\forall i\ k \in (1,2)$, the transformed matrices M_{kc} is define as:

$$M_{kc,[i][j]} = ReLU(\sum_{i}^{i+l-1}\sum_{j}^{j+l-1} K \cdot M_{k,[i:i+l-1][j:j+l-1]} + b) \tag{7}$$

where i and j index the row i-th and column j-th element, respectively. Next, a max-pooling operation is adopted and the representation m_p for p-th utterance in a conversation is obtained through flattening and concatenating the two matrices after pooling as follows:

$$\hat{m}_{k,[i][j]} = \max(M_{kc,[i:i+l-1][j:j+l-1]}) \tag{8}$$

$$m_p = [flatten(\hat{m}_1) \oplus flatten(\hat{m}_2)] \tag{9}$$

where $flatten()$ is the flatten operation and $\oplus$ is the concatenation operation.

3.5 Attentive Turns Aggregation

To aggregate the matching information of the attentive turns in the last stage, The outputs of CNN, $M = [m_1, ..., m_n]$ are fed to GRU to obtain $H_m = [h_{m_1}, ..., h_{m_n}]$. $\forall i$, $h_{m,i} \in H_m$ is defined as

$$h_{m,i} = GRU(h_{m_{i-1}}, m_i) \tag{10}$$

where $GRU(\cdot,\cdot)$ denotes the same calculation and parameterization as Eq.(2). Suppose $v_f = L(H_m)$ is the attention operation which is defined as:

$$\begin{aligned} t_i &= v^T \tanh(W_t P_{u_i} + V_t h_{m_i} + b) \\ \alpha_i &= \exp(t_i) / \sum_{j=1}^{n} \exp(t_j) \\ L(H_m) &= \sum_{i=1}^{n} \alpha_i h_{m_i} \end{aligned} \tag{11}$$

where W_t, V_t and b are parameters. With v_f, we define $\mathcal{F}(U,R)$ as:

$$\mathcal{F}(U,R) = softmax(W_s v_f) \tag{12}$$

	Ubuntu			Douban			ECD		
	Train	Valid	Test	Train	Valid	Test	Train	Valid	Test
# context-response pairs	1M	500K	500K	1M	50K	10K	1M	10K	10K
# candidates per context	2	10	10	2	2	10	2	2	10
Avg # turns per context	10.13	10.11	10.11	6.69	6.75	6.45	5.51	5.48	5.64
Avg # words per utterance	11.35	11.34	11.37	18.56	18.50	20.74	7.02	6.99	7.11

Table 1: Data statistics template for latter use.

where W_s is the parameter. During the training phase, model parameters are updated according to a cross-entropy loss.

Note that *Turns-aware Aggregation* and *Attentive Turns Aggregation* can be seen as two stages of interaction across the utterances (we call all these two process as "*Context Fusion*" henceforth). Specifically, the former is simply a combination after the *Utterance Representation* for richer turns-aware information while the latter is to aggregate matching states of previous turns after attention learning against each utterance itself and the response.

4 Experiment

4.1 Dataset

We evaluate our model on three multi-turn conversation datasets, the Ubuntu Dialogue Corpus (Ubuntu) (Lowe et al., 2015), the Douban Conversation Corpus (Douban) (Wu et al., 2017) and our released E-commerce Dialogue Corpus (ECD) [3]. Data statistics are in Table 1.

Ubuntu Dialogue Corpus *Ubuntu Dialogue Corpus* consists of multi-turn human-computer conversations constructed from Ubuntu IRC chat logs. The training set contains 1 million label-context-response triples where the original context and corresponding response are labeled as positive and negative response are selected randomly on the dataset. On both validation and test sets, each context contains one positive response and 9 negative responses.

Douban Conversation Corpus *Douban conversation corpus* is an open domain dataset constructed from Douban group which is a popular social networking service in China. Response candidates on the test set are collected by a standard search engine *Apache Lucene*[4], other than negative sampling without human judgment on Ubuntu Dialogue Corpus. That is, the last turn of each Douban dialogue with additional keywords extracted from the context on the test set is used as query to retrieve 10 response candidates from the Lucene index set.

E-commerce Dialogue Corpus In this part, we will introduce our E-commerce Dialogue Corpus. Though previously described public datasets have served in solid studies, there is no comprehensive e-commerce dataset available for research. We collect real-world conversations between customers and customer service staff from our E-commerce partners in Taobao [5], which is the largest e-commerce platform in China [6]. It contains over 5 types of conversations (e.g. commodity consultation, logistics express, recommendation, negotiation and chitchat) based on over 20 commodities. As word segmentation treatment is the primary step in Chinese language processing tasks (Zhao et al., 2017; Cai et al., 2017; Cai and Zhao, 2016), we adopt *BaseSeg* (Zhao et al., 2006) to tokenize the texts. For a discriminative learning, we add negative responses by ranking the response corpus based on the last utterance along with the top-5 key words in the context using *Apache Lucene*. The ratio of the positive and the negative is 1:1 in training and validation, and 1:9 in testing.

[3] Our released dataset along with source code can be accessed via `https://github.com/cooelf/DeepUtteranceAggregation`.

[4] `http://lucene.apache.org/`

[5] `https://www.taobao.com`

[6] All the data have been carefully desensitized and anonymized with the consent of our partners and avoid privacy issues.

Model	Ubuntu Dialogue Corpus			Douban Conversation Corpus					
	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$	MAP	MRR	P@1	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$
TF-IDF	0.410	0.545	0.708	0.331	0.359	0.180	0.096	0.172	0.405
RNN	0.403	0.547	0.819	0.390	0.422	0.208	0.118	0.223	0.589
CNN	0.549	0.684	0.896	0.417	0.440	0.226	0.121	0.252	0.647
LSTM	0.638	0.784	0.949	0.485	0.537	0.320	0.187	0.343	0.720
BiLSTM	0.630	0.780	0.944	0.479	0.514	0.313	0.184	0.330	0.716
Multi-View	0.662	0.801	0.951	0.505	0.543	0.342	0.202	0.350	0.729
DL2R	0.626	0.783	0.944	0.488	0.527	0.330	0.193	0.342	0.705
MV-LSTM	0.653	0.804	0.946	0.498	0.538	0.348	0.202	0.351	0.710
Match-LSTM	0.653	0.799	0.944	0.500	0.537	0.345	0.202	0.348	0.720
Attentive-LSTM	0.633	0.789	0.943	0.495	0.523	0.331	0.192	0.328	0.718
Multi-Channel	0.656	0.809	0.942	0.506	0.543	0.349	0.203	0.351	0.709
Multi-Channel$_{exp}$	0.368	0.497	0.745	0.476	0.515	0.317	0.179	0.335	0.691
SMN	0.726	0.847	0.961	0.529	0.569	0.397	0.233	0.396	0.724
DUA	**0.752**	**0.868**	**0.962**	**0.551**	**0.599**	**0.421**	**0.243**	**0.421**	**0.780**

Table 2: Comparison of different models on Ubuntu Dialogue Corpus and Douban Conversation Corpus. All the results except ours are from (Wu et al., 2017).

4.2 Settings

Our evaluation is based on the following information retrieval metrics: Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), Precision at 1 (P@1) and Recall at position k in n candidates ($Rn@k$) , which are widely used for relevance evaluation (Wu et al., 2017; Lowe et al., 2015). For the sake of computational efficiency, the maximum number of utterances is specialized as 10 and each utterance contains at most 50 words. We apply truncating and zero-padding when necessary. Word embedding is trained by Word2Vector (Mikolov et al., 2013) on the training data and the dimension is 200. Our model is implemented using the Theano [7]. We use stochastic gradient descent with ADAM (Kingma and Ba, 2014) updates for optimization. The batch size is 200 and the initial learning rate is 0.001. The window size of convolution and pooling is (3, 3) and the number of hidden units for the character GRU is set to 200. All of our models are run on a single GPU (GeForce GTX 1080 Ti). We run all the models up to 5 epochs and select the model that achieves the best result in validation.

Our baselines include:

• **Single-turn matching models**: Basic models in (Kadlec et al., 2015; Lowe et al., 2015), including TF-IDF, CNN, RNN, LSTM and biLSTM ; We also explore other advanced single-turn matching models, MV-LSTM (Wan et al., 2016), Match-LSTM (Wang and Jiang, 2015), Attentive-LSTM (Tan et al., 2015), Multi-Channels (Wu et al., 2017); These models concatenate the context utterances together to match a response.

• **Advanced multi-turn matching models**: Multi-view model of (Zhou et al., 2016) that models utterance relationships from word sequence view and utterance sequence view; Deep Learning-to-Respond (DL2R) model of (Yan et al., 2016) which reformulates the last utterance (query) with other utterances in the context via neural model; Sequential Matching Network (SMN) (Wu et al., 2017) that matches a response with each utterance in the context.

The results of baseline models on Ubuntu and Douban are from (Wu et al., 2017). For evaluation on our ECD dataset, we reproduce the models following their same settings.

4.3 Experimental Results

Table 2-3 show the results on the three corpora. Our model outperforms all other models greatly in terms of most of the metrics. Single matching models which concatenate the previous utterances, perform much worse than our model, showing the importance of utterance relationships and simply concatenating

[7] https://github.com/Theano/Theano

Model	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$
TF-IDF	0.159	0.256	0.477
RNN	0.325	0.463	0.775
CNN	0.328	0.515	0.792
LSTM	0.365	0.536	0.828
BiLSTM	0.355	0.525	0.825
Multi-View	0.421	0.601	0.861
DL2R	0.399	0.571	0.842
MV-LSTM	0.412	0.591	0.857
Match-LSTM	0.410	0.590	0.858
Attentive-LSTM	0.401	0.581	0.849
Multi-Channel	0.422	0.609	0.871
Multi-Channel$_{exp}$	0.352	0.556	0.827
SMN	0.453	0.654	0.886
DUA	**0.501**	**0.700**	**0.921**

Table 3: Comparison of different models on E-commerce Dialogue Corpus.

utterances together is not an appropriate solution for multi-turn conversation modeling. Our model also achieves a great improvement (4.8% $R_{10}@1$ on ECD corpus) over the state-of-the-art multi-turn response matching model, SMN, which matches each utterance and response without turns-aware aggregation and matching attention flow. This comparison indicates the effectiveness of our context composing approach. The advantage on ECD dataset further indicates our model can well imitate the conversations of real customer service instead of merely being good at chitchat.

4.4 Discussion

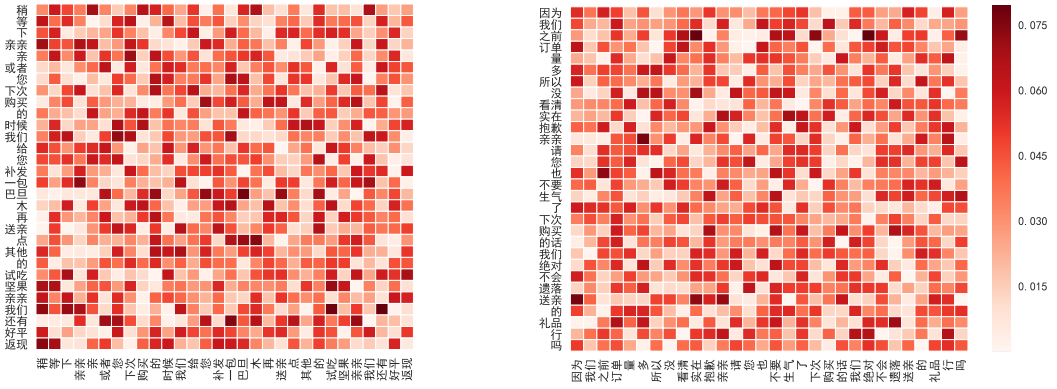
Conversation Type Analysis To evaluate the model performance on different types of conversations, we manually separate our ECD test set into 5 categories.

- **Consultation**: consultations about commodity's property, usage, packaging, etc.
- **Logistics**: questions about logistics partners, delivery progress.
- **Recommendation**: commodity comparisons and recommendations.
- **Negotiation**: customer complaints and negotiations.
- **Chitchat**: greetings, non task-oriented conversations and chitchats.

Table 4 shows the statistics and the model results. As we see, the types of chitchat and logistics tend to be easily handled. Recommendations, consultation and negotiations are relatively harder to respond since they often involve with various topics (e.g. the concerned commodities) and intentions, which makes our corpus more challenging than previous chitchat or question answering based corpora.

Visualization To analyze the effectiveness of the attention mechanism of our model, we draw the self-matching distributions after *matching attention flow*. From the validation set of our ECD data, Figure 3 shows the word weights of a momentous utterance (with high weights in the response matching component) and the response respectively. We see the model could accurately distill the linchpin from the utterance, {*Next consumption, reissue, a bag of almond, send you, some nuts, cashback*} and from the response {*too many orders before, really sorry, don't be angry, your gift*}. When a user complained about the missing gift and slow delivery, our model could distinguish the user's intention after self-matching and seek out the suitable response substantially according to the crux of the presented utterance. This shows our model is effective at selecting the vital points after *Matching Attention Flow*, guiding the *Response Matching* layer to collect more relevant pieces.

Ablation Study To have an insight of the effectiveness of each component in DUA, we remove one each time. The steepest reduction (6.9% $R_{10}@1$) is observed when we remove *Matching Attention Flow* which shows it quite vital to draw the linchpins of each utterance. The performance also drops substan-



(a) Highlighted utterance (b) Response

Last utterance (user): How can you miss my gift! And the delivery is so slow. You are spoiled !!!
Highlighted utterance (bot): please wait a moment dear. For compensation, we'll reissue you a bag of almonds at your next consumption. Besides, we will also send you some nuts to taste. If you give us five-star rating and comments, you'll also receive some cashback.
Response: Because we had too many orders before, we unfortunately misread your order. We are really sorry for the mistake. Please don't be angry, we will never forget your gift again.

Figure 3: Pair-wise attention visualization on utterance and response after matching attention flow.

	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$
Consultation (36.1%)	0.474	0.696	0.900
Logistics (7.3%)	0.510	0.707	0.916
Recommendation (4.4%)	0.487	0.590	0.897
Negotiation (5.9%)	0.385	0.462	0.846
Chitchat (26.3%)	0.573	0.762	0.931
Overall(100%)	0.501	0.700	0.921

Table 4: Results on different types of conversations.

tially (4.8% $R_{10}@1$) when removing *Context Fusion* including the first turns-aware aggregation (first-stage aggregation) and replacing the last GRU (last-stage aggregation) for matching accumulation with a multi-layer perceptron. This indicates that utterance relationships are indeed important. Without *Context Fusion* and *Matching Attention Flow* mechanisms, the model performs the worst which verifies our proposed mechanism indeed improves the context representation essentially.

4.5 Error Analysis

After carefully analyzing the predicted responses, we find the error cases could be classified into the following categories for later further improvement.

Multiple intentions In E-commerce conversations users extremely likely express various intentions in a single message, which is another big difference from previous multi-turn conversation corpus besides diverse types of conversations among various commodities. For example, {*User: How about the packaging of skin care products. By the way, which delivery company will be responsible for shipping and how long can I receive the goods?*}. This would seriously confuse the model where the given response might be preferential to one or another aspect.

Topic errors Our model retrieves response according to semantic similarity with the context, with no special attention on the conversation topic, such as the currently discussed commodities. In most cases, the concerned commodity would be picked out from the context with high attention weights and guide the model to select responses. However, when the conversation involves several goods, for example, {*User: How about nuts? Bot: Nuts is good. User: Ok then, how about zongzi?*}, the model might give the response about *nuts* instead of *zongzi*. This indicates there exists much potential for improvements

	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$
DUA	0.501	0.700	0.921
-CF	0.453	0.642	0.890
-MAF	0.432	0.625	0.883
-CF -MAF	0.413	0.613	0.867

Table 5: Ablation study on ECD dataset. CF and MAF denote the *Context Fusion* and *Matching Attention Flow*. The bracket means the context fusion approach adopted by the model.

by considering extra topic recognition.

Multiple suitable responses In our ECD dataset, we assume there is only one correct response for each conversation which is the same setting as Ubuntu Dialogue Corpus. However, the model sometimes gives responses having similar meaning with the ground-truth one, but they would be regarded as wrong during evaluation, especially for fairly long conversations. This could make the task rather challenging with the strict restriction of exact match. This might be alleviated by involving expert labeling like (Wu et al., 2017). However, this is quite labour-intensive and subjective. In the future, we would explore more automatic solutions.

5 Conclusion

In this paper, we propose a deep utterance aggregation approach to form a fine-grained context representation. We also release the first e-commerce dialogue corpus to research communities. Experiments on three datasets show the model can yield new state-of-the-art results. Various analyses are conducted to evaluate the model and the released dataset. In the future, we may study how to improve modeling of contextual semantics and design a better neural network for multi-turn conversations in terms of various intentions and topics.

References

Hongxiao Bai and Hai Zhao. 2018. Deep enhanced representation for implicit discourse relation recognition. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, pages 1137–1155.

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for Chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 409–420.

Deng Cai and Hai Zhao. 2017. *Pair-Aware Neural Sentence Modeling for Implicit Discourse Relation Classification*. IEA/AIE 2017, Part II, LNAI 10351.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and accurate neural word segmentation for Chinese. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 608–615.

Jiaxun Cai, Shexia He, Zuchao Li, and Hai Zhao. 2018. A full end-to-end semantic role labeler, syntactic-agnostic or syntactic-aware? In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1724–1734.

Lei Cui, Shaohan Huang, Furu Wei, Chuanqi Tan, Chaoqun Duan, and Ming Zhou. 2017. Superagent: A customer service chatbot for e-commerce websites. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, System Demonstrations (ACL 2017)*, pages 97–102.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 484–495.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL2017)*, pages 1766–1776.

Shexia He, Zuchao Li, Hai Zhao, Hongxiao Bai, and Gongshen Liu. 2018. Syntax for semantic role labeling, to be, or not to be. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*.

Yafang Huang, Zuchao Li, Zhuosheng Zhang, and Hai Zhao. 2018. Moon IME: neural-based chinese pinyin aided input method with customizable association. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018), System Demonstration*.

Zhongye Jia and Hai Zhao. 2014. A joint graph model for Pinyin-to-Chinese conversion with typo correction. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 1512–1523.

Rudolf Kadlec, Martin Schmid, and Jan Kleindienst. 2015. Improved deep learning baselines for ubuntu corpus dialogs. *arXiv preprint arXiv:1510.03753*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Zuchao Li, Jiaxun Cai, Shexia He, and Hai Zhao. 2018. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*.

Zachary C Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. 2018. Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The Ubuntu Dialogue Corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the SIGDIAL 2015 Conference (SIGDIAL 2015)*, pages 285–294.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2017. Coherent dialogue with attention-based language models. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017)*, pages 3252–3259.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric P. Xing. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 1006–1017.

Minghui Qiu, Feng Lin Li, Siyu Wang, Xing Gao, Yan Chen, Weipeng Zhao, Haiqing Chen, Jun Huang, and Wei Chu. 2017. Alime chat: A sequence to sequence and rerank based chatbot engine. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 498–503.

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron Courville. 2017a. Multiresolution recurrent neural networks: An application to dialogue response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017)*, pages 3288–3295.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017b. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017)*, pages 3295–3302.

Ming Tan, Cicero Dos Santos, Bing Xiang, and Bowen Zhou. 2015. LSTM-based deep learning models for non-factoid answer selection. In *Proceedings of the International Conference on Learning Representations (ICLR 2016)*.

Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. 2016. Match-srnn: Modeling the recursive matching structure with spatial rnn. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI 2016)*, pages 2922–2928.

Shuohang Wang and Jing Jiang. 2015. Learning natural language inference with LSTM. In *Proceedings of NAACL-HLT 2016 (NAACL 2016)*, pages 1442–1451.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 189–198.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 438–449.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 665–677.

Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 496–505.

Yu Wu, Wei Wu, Dejian Yang, Can Xu, Zhoujun Li, and Ming Zhou. 2018. Neural response generation with dynamic vocabularies. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*.

Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 55–64.

Zhao Yan, Nan Duan, Peng Chen, Ming Zhou, Jianshe Zhou, and Zhoujun Li. 2017. Building task-oriented dialogue systems for online shopping. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017)*, pages 4618–4627.

Tom Young, Erik Cambria, Iti Chaturvedi, Minlie Huang, Hao Zhou, and Subham Biswas. 2018. Augmenting end-to-end dialog systems with commonsense knowledge. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*.

Zhuosheng Zhang and Hai Zhao. 2018. One-shot learning for question-answering in gaokao history challenge. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*.

Zhisong Zhang, Hai Zhao, and Lianhui Qin. 2016. Probabilistic graph-based dependency parsing with convolutional neural network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1382–1392.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018a. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*.

Zhuosheng Zhang, Yafang Huang, and Hai Zhao. 2018b. Subword-augmented embedding for cloze reading comprehension. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018)*.

Zhuosheng Zhang, Jiangtong Li, Hai Zhao, and Bingjie Tang. 2018c. Sjtu-nlp at semeval-2018 task 9: Neural hypernym discovery with term embeddings. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval 2018), Workshop of NAACL-HLT 2018*.

Zhuosheng Zhang, Jiangtong Li, Hai Zhao, and Bingjie Tang. 2018d. Sjtu-nlp at semeval-2018 task 9: Neural hypernym discovery with term embeddings. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval 2018), Workshop of NAACL-HLT 2018*.

Hai Zhao, Chang-Ning Huang, Mu Li, and Taku Kudo. 2006. An improved Chinese word segmentation system with conditional random field. *Proceedings of the Fifth Sighan Workshop on Chinese Language Processing*, pages 162–165.

Hai Zhao, Deng Cai, Changning Huang, and Chunyu Kit. 2017. *Chinese Word Segmentation, a decade review (2007-2017)*. China Social Sciences Press, Beijing, China, July.

Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, Dianhai Yu, Hao Tian, Xuan Liu, and Rui Yan. 2016. Multi-view response selection for human-computer conversation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 372–381.

Ganbin Zhou, Ping Luo, Rongyu Cao, Fen Lin, Bo Chen, and Qing He. 2017. Mechanism-aware neural machine for dialogue response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017)*, pages 3400–3408.

Argumentation Synthesis following Rhetorical Strategies

Henning Wachsmuth [*] **Manfred Stede** [**] **Roxanne El Baff** [†]
Khalid Al-Khatib [†] **Maria Skeppstedt** [‡ **] **Benno Stein** [†]
[*] Paderborn University, Paderborn, Germany, henningw@upb.de
[**] University of Potsdam, Potsdam, Germany, stede@uni-potsdam.de
[†] Bauhaus-Universität Weimar, Weimar, Germany, <first>{.<last>}+@uni-weimar.de
[‡] Linnaeus University, Växjö, Sweden, maria.skeppstedt@lnu.se

Abstract

Persuasion is rarely achieved through a loose set of arguments alone. Rather, an effective delivery of arguments follows a rhetorical strategy, combining logical reasoning with appeals to ethics and emotion. We argue that such a strategy means to *select*, *arrange*, and *phrase* a set of argumentative discourse units. In this paper, we model rhetorical strategies for the computational synthesis of effective argumentation. In a study, we let 26 experts synthesize argumentative texts with different strategies for 10 topics. We find that the experts agree in the selection significantly more when following the same strategy. While the texts notably vary for different strategies, especially their arrangement remains stable. The results suggest that our model enables a strategical synthesis.

1 Introduction

The primary use of arguments in natural language is to persuade others of a stance towards a controversial topic (Mercier and Sperber, 2011). According to Johnson and Blair (2006), an argument is logically cogent if its premises are relevant as support for its conclusion, individually acceptable, and together sufficient to draw the conclusion. In real life, however, argumentation is by far not only about logic (Allwood, 2016). Without a *rhetorical strategy*, arguments will hardly ever unfold their persuasive effectiveness.

Following Aristotle (2007), we see a rhetorical strategy as the purposeful encoding of three means of persuasion in a well-arranged and well-phrased speech or text: *logos* (providing logically reasoned arguments), *ethos* (demonstrating good character and credibility), and *pathos* (evoking the right emotions). Listeners or readers then decode the encoding, forming their view of the author's logos, ethos, and pathos. In the realm of the area of computational argumentation, rhetorical strategies are particularly relevant for technologies that synthesize argumentative text and that aim to deliver arguments effectively.

Existing argument mining research largely focuses on the logical structure of arguments, identifying their units (premises vs. conclusions) and relations (support vs. attack). Recently, a few studies have tackled strategy-related aspects, such as explicit expressions of ethos (Duthie et al., 2016) and the effects of logical and emotional arguments across audiences (Lukin et al., 2017). So far, however, strategies have not been considered in argumentation synthesis, which altogether has not received much attention (see Section 2).

In this paper, we study the role of rhetorical strategies when synthesizing argumentation. In particular, we consider monological argumentative texts where an author seeks to persuade target readers of his or her stance towards a given topic, such as news editorials and persuasive essays. Conceptually, we argue that an author synthesizes a text of such genres in three subsequent steps:

1. *Selecting content* in terms of argumentative discourse units (along with facts, anecdotes, and similar) that frame the given topic in a way that is effective for the intended stance,
2. *arranging the structure* of the units considering ordering preferences, and
3. *phrasing the style* of the resulting text to match the genre and the encoded means of persuasion.

We expect that the encoding of the means of persuasion becomes manifest in measurable text properties related to selection, arrangement, and phrasing. Figure 1 sketches the analysis of such properties in a

Proceedings of the 27th International Conference on Computational Linguistics, pages 3753–3765
Santa Fe, New Mexico, USA, August 20-26, 2018.

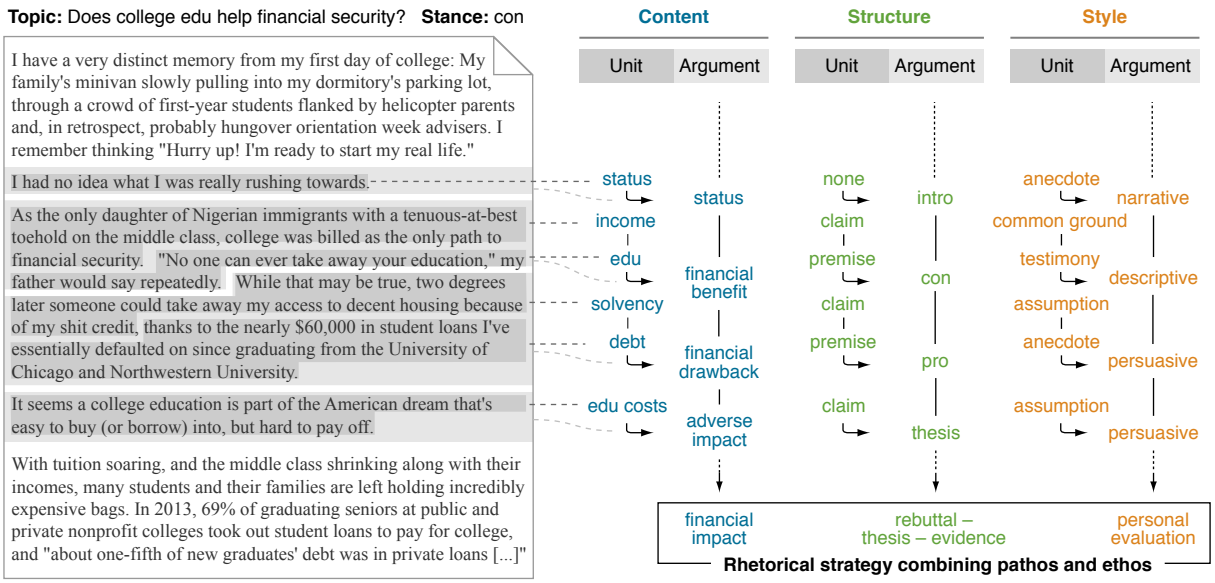


Figure 1: A monological argumentative text (a news editorial from Al-Khatib et al. (2016)), along with exemplary content, structure, and style properties captured by the envisioned model of rhetorical strategies.

pathos and ethos-oriented news editorial. Thereby, it also hints at our envisioned computational model of rhetorical strategies. We discuss this model and its application when synthesizing an argumentative text following a particular rhetorical strategy in Section 3. In Section 4, we design a dataset for studying argumentation synthesis with 200 argumentative discourse units for 10 pairs of topic and stance from the Arg-Microtexts corpus (Peldszus and Stede, 2016). To evaluate the adequacy of the general synthesis process reflected by our model, we then carry out an experiment where 26 human experts select, arrange, and re-phrase short argumentative texts following principled rhetorical strategies, thereby also creating benchmark data for computational approaches (Section 5).

We find that humans agree significantly more when selecting argumentative discourse units to encode specific means of persuasion, and we observe clear differences between strategies. At the same time, the arrangement of the texts remains stable across strategies. Their phrasing requires further investigation in future work. The results suggest that our model captures both general and strategy-specific aspects of synthesizing argumentation. As such, it defines a first substantial step towards computational argumentation synthesis following rhetorical strategies. In total, our main contributions are:

1. *A computational model* of rhetorical strategies for synthesizing monological argumentative texts.
2. *A benchmark dataset* with manually synthesized arguments that follow rhetorical strategies.
3. *Empirical evidence* that humans agree in strategy-specific argumentation synthesis.

2 Related Work

Aristotle (2007) revealed that persuasion is achieved through a well-arranged speech with a clear and appropriate style that delivers an adequate mix of the three means of persuasion: (1) *logos*, the use of logically reasoned arguments, (2) *ethos*, the demonstration of the speaker's credibility and good character, and (3) *pathos*, the evoking of the right emotions in the target audience. As Aristotle, we target argumentation that aims for persuasion — as opposed to critical discussions where *strategic maneuvering* is needed to achieve reasonableness (van Eemeren et al., 2014). Jovičić (2006) proposed a procedure to evaluate the effectiveness of persuasive argumentation, though not in a computational way.

In natural language processing, most computational argumentation research focuses on the mining of argumentative units and relations from text (Stab and Gurevych, 2014; Peldszus and Stede, 2015; Ajjour et al., 2017). Argument mining infers the logical structure of arguments, but it does not analyze the strategy used to compose arguments. Feng and Hirst (2011) classify the five most common argumentation *schemes* of Walton et al. (2008). Such a scheme defines a pattern capturing the logical inference from an

argument's premises to its claim, i.e., it primarily aims at logos only. Song et al. (2014) combine the idea of schemes with strategy-related considerations, and Habernal and Gurevych (2015) annotate pathos in forum comments and blog posts. Likewise, Duthie et al. (2016) develop a corpus and an approach to mine explicit expressions of ethos from political debates, whereas Hidey et al. (2017) even annotate all three means of persuasion for the premises of arguments. The provided data and methods may be helpful for studying persuasive effectiveness, but they have not been employed so far for that purpose.

In (Wachsmuth et al., 2016), we use the output of argument mining to assess four quality dimensions of persuasive essays. Some assessed dimensions relate to rhetoric, such as argument strength, which is defined based on how many readers are persuaded. Habernal and Gurevych (2016) examine the reasons that make arguments convincing, and Lukin et al. (2017) study how effective logos-oriented and pathos-oriented arguments are depending on the target audience. However, none of these works considers the application of rhetorical strategies.

Tan et al. (2016) find that the chance to persuade someone in good-faith discussions on *Reddit Change My View* is increased through multiple interactions and an appropriate linguistic style. Cano-Basave and He (2016) analyze the impact of the semantic framing of arguments (e.g., "taking sides" and "manipulation") in political debates. Similarly, Wang et al. (2017) reveal the importance of selecting the right framing of a discussion topic for winning classical debates. In such dialogical situations, the arguments of opposing parties are usually fragmented into several alternating parts. Our work, in contrast, analyzes the rhetorical strategies of complete monological texts where an author presents his or her entire argumentation.

Studying persuasive blog posts, Anand et al. (2011) develop a scheme with 16 persuasion tactics of four types: those that postulate outcomes of an uptake, those that generalize in some way, those that appeal to external authorities, and those that rely on interpersonal factors. These tactics are found in small text spans and could be seen as the local counterpart of the global strategies we consider. To our knowledge, only we have explicitly worked towards a computational analysis of such strategies so far. In particular, we presented a corpus with 300 news editorials whose units are labeled with their roles in the argumentation, such as "testimony" and "common ground" (Al-Khatib et al., 2016). In (Al-Khatib et al., 2017), we then trained a classifier on this corpus to find sequential role patterns in 30k New York Times editorials. While we observed insightful variances in the use of evidence across editorial topics, it still remains unclear to what extent such patterns really reflect rhetorical strategies.

Clearly diverging from previous research, we consider rhetorical strategies in argumentation *synthesis*, for which related work is generally still scarce. Bilu and Slonim (2016) generate new claims by recycling topics and predicates from existing claims, whereas Reisert et al. (2015) synthesize complete arguments based on the model of Toulmin (1958) where a claim is supported by data that is reasoned by a warrant. Like us, those authors rely on a pool of argument units from which arguments are built. However, they restrict their view to logical argument structure. The same holds for Green (2017) who generates arguments with particular schemes (e.g., "cause to effect") that are used to teach learners how to argue.

Yanase et al. (2015) present a method that arranges the sentences of an argumentative text in a natural way. Sato et al. (2015) build upon this method. Their system pursues similar goals as we do, phrasing an ordered text with multiple arguments. We extend their idea by rhetorical considerations, and we propose a general argumentation synthesis model. It is in line with classical concepts of building natural language generation systems (Reiter and Dale, 1997), but it targets argumentative texts in particular. With that, we seek to contribute a missing aspect to technologies that synthesize arguments such as *The Debater* (Rinott et al., 2015), namely, how to deliver arguments *effectively*. So far, strategical systems exist only for formal argumentation (Rosenfeld and Kraus, 2016).

Our synthesis-oriented model covers content, structure, and style properties. For computational purposes, such properties need to be mined before. To capture content, some works identify the different frames under which a topic can be viewed (Naderi and Hirst, 2017), model key aspects of a topic (Menini et al., 2017), or analyze potentially strategy-related topic patterns in campaign speeches (Gautrais et al., 2017). In terms of structure, Persing et al. (2010) learn sequences of discourse functions in essays (e.g., "rebuttal" or "conclusion") that correlate with a good organization, and we have modeled flows of arbitrary types of information to classify text properties, such as stance or quality (Wachsmuth and Stein, 2017).

In (Wachsmuth et al., 2017), we extend this idea to model sequential and hierarchical argumentation structure simultaneously. Regarding style, Lawrence et al. (2017) analyze rhetorical figures in arguments, Song et al. (2017) classify discourse modes in essays, and Zhang et al. (2017) study rhetorical questions in political discourse. All such methods may be relevant when we operationalize our model.

Finally, the use of strategies in synthesis scenarios follows up on early work on discourse planning (Young et al., 1994; Zukerman et al., 2000). The computational approaches relied on rule-based techniques to create effective arguments (Carenini and Moore, 2006) for a few selected tactics. More recent research on general text generation uses probabilistic models to employ text structure (Barzilay, 2010), or synthesizes texts such that they have a certain style in terms of sentiment or similar (Hu et al., 2017; Shen et al., 2017). Our model is meant to provide an abstract framework to be exploited in such approaches.

3 Model

We now delineate our model of rhetorical strategies for synthesizing a monological argumentative text following a specified rhetorical strategy. As clarified below, its three main building blocks follow Aristotle (2007). While we do not operationalize the model computationally in this paper, we will study its general adequacy with human experts in Section 5. An instance of the model has been given in Figure 1.

3.1 Argumentation Synthesis following Rhetorical Strategies

Given a controversial topic or question (such as "Does college edu help financial security?" in Figure 1) and a stance towards it (such as "con"), the goal of a rhetorical strategy can be seen as delivering arguments along with facts, anecdotes, etc. to persuade some target audience of the intended stance in an effective way. In line with Aristotle (2007), we hypothesize that such a strategy can be manifested in a monological argumentative text through a purposeful encoding of the three means of persuasion (see Section 2). i.e., logos, ethos, and pathos. An instance of a strategy (which is tailored to a specific target audience) thus specifies to what degree each means should be present, such as *logos 70%, ethos 10%, pathos 20%*.[1]

For giving a persuasive speech, Aristotle (2007) postulates five consecutive *canons of rhetoric*: (1) *inventio*, the selection of arguments, (2) *dispositio*, the arrangement of the arguments to achieve maximum impact, (3) *elocutio*, the phrasing of the arguments in a clear and appropriate style, (4) *memoria*, the memorization of the speech, and (5) *actio*, the delivery of the speech with gestures, prosody, and further means. Since we focus on written argumentation, the latter two are obsolete in the given context. Accordingly, we model a rhetorical strategy for synthesizing monological argumentative texts as a process of the three steps mentioned in Section 1: selecting content, arranging structure, and phrasing style.

Technically, we regard a strategy as a script of operators from a set $\Omega = \Omega_s \cup \Omega_a \cup \Omega_p$, where Ω_s are select operators, Ω_a are arrange operators, and Ω_p are phrase operators. These operators can be combined to form complex, nested n-ary operations on a set of input units. We expect that an applied strategy becomes manifest in measurable text properties at different granularity levels, ranging from single words and phrases over argumentative units and complete arguments, up to the full argumentation in a text. The identification of the best properties is beyond the scope of this paper, but the box in the lower right corner of Figure 1 summarizes an exemplary overall manifestation of a pathos and ethos-oriented strategy: The author focuses on the financial impact of college education to support her stance against it. She rebuts existing views, before she states her thesis and provides evidence, all in a personal evaluation.[2]

For a concrete operationalization of the model, the input to the operators may be pre-fabricated text units (roughly: clauses corresponding to single propositions), semantic representations in a suitable formalism (as in the case of classical "deep" text generation), or some intermediate form like partially-analyzed text units, where information about syntax, named entities, etc. is available. In line with the experiment reported in Section 5, we here assume that a set of candidate text units, which we call the *pool*, is available to make informed selections. Ideally, the units have been analyzed for (1) means of persuasion present, (2) stance on the topic, (3) framing of the topic, and (4) effectiveness when functioning in an argument.

[1] Also, a rhetorical strategy will have to consider conventions of the intended text genre (news editorial, persuasive essay, etc.), but we leave that aside for present purposes.

[2] The distinction of content, structure, and style may not be perfectly orthogonal in all cases, which is rooted in the complexity and polythetic nature of natural language. Notice in this regard that all labels depicted in Figure 1 are exemplary only.

3.2 Selecting the Content (Inventio)

The first step of synthesizing argumentation is to decide on those frames under which to view a topic that support the intended stance best, while matching the desired means of persuasion (as specified by the strategy). In accordance with these decisions, a set of units is to be selected from the pool. For instance, the editorial in Figure 1 frames college education in terms of its financial benefits and drawbacks, among other frames. A different strategy could put the focus on the freedom of career choice. When making the selection, facts, anecdotes, and similar statements may be added to the arguments, particularly to demonstrate credibility or to evoke emotions (if the strategy favors ethos or pathos, respectively). In Figure 1, we find a personal anecdote in the beginning, whereas the last paragraph gives statistical and testimonial evidence. Technically, the set Ω_s will contain operators of two types:

$$frames: \ select(topic, stance, strategy) \qquad units: \ select(frames, strategy)$$

To establish the knowledge base behind Ω_s, a preceding corpus analysis will be needed, which associates frames and units for topic-stance combinations with information about their persuasive effectiveness and the encoded means of persuasion. As summarized in Section 2, several computational techniques to implement various steps of this analysis have been proposed, such as unit segmentation (Ajjour et al., 2017), frame classification (Naderi and Hirst, 2017), and key aspect clustering (Menini et al., 2017).

3.3 Arranging the Structure (Dispositio)

Given the units, the next step is to compose the selected units in structured arguments and to sequentially order them, in a way that maximizes their impact. The financial drawback argument in Figure 1, for instance, gives the debts from student loans as the premise for the author's claimed solvency problems. The resulting argument (pro the intended stance) rebuts the preceding con argument and — probably for rhetorical reasons — is immediately followed by the main thesis within the sequential flow of the editorial (indicated by the vertical lines). To illustrate the means of persuasion, an emphasis on pathos could, for example, lead to a preference for gradually increasing the strength of emotional appeal throughout the text. In contrast, for a logos-oriented strategy, it may be important that the sequence of units *coheres* locally and globally (which for a pathos-oriented argument may be less relevant, or even detrimental). Technically, we distinguish two types of arrange operators in Ω_a:

$$arguments: \ arrange(units, strategy) \qquad flow: \ arrange(arguments, strategy)$$

For executing this step, knowledge about the roles of units in arguments (Stab and Gurevych, 2014) and about effective flows of text units (Wachsmuth and Stein, 2017) is needed. This script-like knowledge fuses the information on the means of persuasion and of unit strength on the one hand with principles of text coherence and rhetorical organization on the other.

3.4 Phrasing the Style (Elocutio)

Finally, the sequence of units is to be presented in an argumentative text, making stylistic decisions, which in turn are governed by the strategy. The technical approach depends much on the type of "unit" that is being implemented (see end of Section 3.1). For minimally-analyzed text units, one can add connectives indicating the relation between units. If deeper analyses are available, it may be possible to use markers encoding evidence types or discourse modes, among other rhetorical devices. If syntax can be controlled, then a pathos-oriented argument may prefer relatively flat structures, while a logos-oriented one may lean towards more complex embedding, inviting the reader to follow the path of reasoning. In the mentioned pro argument from the editorial, a concession is expressed through the word "while", indicating that the con argument is outweighed by the pro argument. A personal anecdote is then given as a sarcastic ("thanks to") causal evidence, which is used as a pathos-oriented persuasion attempt. Technically, we thus consider the set Ω_p to be composed of at least two basic types:

$$sentences: \ phrase(units, arguments, strategy) \qquad text: \ phrase(sentences, flow, strategy)$$

Existing techniques to identify the right phrasing operators include evidence type classification (Rinott et al., 2015; Al-Khatib et al., 2017), discourse mode identification (Song et al., 2017), and others.

Question (with Marked Controversial Topic)	# Texts
Should the fine for leaving **dog excrements** on sideways be increased?	9
Should **shopping malls** generally be allowed to open on holidays and Sundays?	8
Should public health insurance cover treatments in complementary and **alternative medicine**?	8
Should Germany introduce the **death penalty**?	8
Should only those viewers pay a **TV licence fee** who actually want to watch programs offered by public broadcasters?	7
Should there be a cap on **rent increases** for a change of tenant?	6
Should all universities in Germany charge **tuition fees**?	6
Should the statutory **retirement age** remain at 63 years in the future?	6
Should the the **morning-after pill** be sold over the counter at the pharmacy?	6
Should **intelligence services** be regulated more tightly by parliament?	4

Table 1: The 10 questions from the Arg-Microtexts corpus from which we use ADUs for our dataset.

4 Data

This section describes the dataset in terms of a pool of argumentative discourse units that we built in order to study the adequacy of the proposed model for strategical argumentation synthesis.

4.1 Source Texts of the Argumentative Discourse Units

We decided to work with the English version of the Arg-Microtexts corpus (Peldszus and Stede, 2016), which was designed to provide crisp argumentation in a "pro and con" manner. The corpus contains 112 short argumentative texts that have been written in response to 18 questions on different controversial topics. The stance of each text towards the topic is labeled (*pro* or *con*). Every text consists of 3 to 10 manually segmented *argumentative discourse units (ADUs)* with a mean of about 5. ADUs can range from a single clause to (in principle) multiple sentences, which together contribute a single function to the argumentative structure. The structure is inspired by the model of Freeman (2011) and allows for linked, convergent, and serial support, as well as two kinds of attack (rebuttals and undercutters).

From the annotated structure, we can directly derive the *thesis* of each text (aka the main claim) and, for each other ADU, the stance that it takes towards this thesis (*pro* or *con*). In total, the Arg-Microtexts corpus contains 576 ADUs (112 theses, 339 pros, and 125 cons).

4.2 Decontextualization of the Argumentative Discourse Units

For our purposes, we need ADUs to be as independent as possible of the context in which they originally occurred. At this point, the fact that the corpus texts are relatively short and hence free of complex inter-relationships, becomes helpful. We semi-automatically preprocessed each unit in four steps:

1. *Removal of sentence information.* We removed all capitalizations of words that were due to sentence beginnings as well as all sentence delimiters.
2. *Removal of unit prefixes.* We removed all discourse markers and connectives at unit beginnings. The top 10 terms were "but" (43 times), "and" (39), "as" (18), "however" (15), "besides" (11), "for" (11), "also" (1), "that's why" (10), and "yet" (10). Together, they made up 61% of all removed prefixes.
3. *Removal of unit infixes.* We removed the following markers within units: "also" (16 times), "however" (14), "therefore" (10), "hence" and "thus" (3 each), as well as "though" (2).
4. *Pronoun resolution.* Finally, we resolved all pronouns in the units whose reference was unclear or ambiguous when given only the question of the respective text. For example, we replaced "they" by "murderers and rapists" in "the have taken or destroyed another life" from a text on the death penalty.

4.3 A Pool of 200 Argumentative Discourse Units for 10 Controversial Topics

The goal of our study is to imitate the computational synthesis of monological argumentative texts, given a topic, a stance towards the topic, and a strategy specification. To ensure a reasonably large pool of ADUs for synthesis for each topic, we restrict our view to the 10 most frequent questions in the Arg-Microtexts corpus. The number of texts for each of them is listed in Table 1 (controversial topics are marked bold).

Type	#	Argumentative Discourse Unit
Thesis	t1	Germany should by no means introduce capital punishment
	t2	Germany should not introduce capital punishment
	t3	the death penalty is a legal means that as such is not practicable in Germany
	t4	the state ought to prevent murder - not avenge it
Con	c1	criminals should not be put in luxury prisons
	c2	many people think that a murderer has already decided on the life or death of another person
	c3	murderers and rapists have taken or destroyed another life
	c4	proponents of the death penalty count on its deterring effect as well as and the ultimate elimination of any potential threat
Pro	p1	a death would not be of any more use to those affected and their relatives than if the felon receives a long sentence
	p2	a door must remain open for making amends
	p3	capital punishment is not a solution
	p4	courts are subject to human error
	p5	despite the death penalty there are significantly more homicides in the US than in Germany
	p6	every human, even those who have committed a despicable crime, can bring themselves to regret and change their opinion
	p7	everyone must be given the chance to hone their conscience and possibly make amends for their deed
	p8	it is a much graver punishment to be imprisoned forever and be tortured by one's own thoughts than to be killed quickly and easily by an injection
	p9	it turns out time and again that innocent people are convicted and executed
	p10	no one can claim the right to rule over the life or death of another human being
	p11	no one may have the right to adjudicate upon the death of another human being
	p12	we don't live in medieval times anymore
Topic		Should Germany introduce the death penalty? **Stance** Con

Table 2: The candidate thesis, con, and pro units for one topic-stance pair from the dataset we provide.

To reduce the bias of the expert's personal opinions in the study, we use only theses for one stance towards the respective topic. In particular, we kept only those with the majority stance on the topic, resulting in at least four theses for each of the 10 topics.[3] We kept all other ADUs, but we inverted the stance of an ADU in case it referred to one of the discarded theses, because, for instance an ADU that is against a discarded con thesis can be assumed to be in favor of a pro thesis on the same topic.

As a result, we had at least four con and 12 pro units for the four theses on each topic. Additional units were discarded randomly to end up with an equal number of units for all 10 topics.The resulting distribution of the unit types matches the desired distribution in the source corpus. We ordered the theses, con units, and pro units alphabetically (to avoid ordering bias). Table 2 exemplary shows the list of 20 units for one topic. Each such list represents the pool of candidate units for one task in our experiment.[4]

5 Experiments

We now report on the experiment that we carried out with human experts in order to study whether the general process of selecting, arranging, and phrasing defined by our model from Section 3 is suitable for a strategical synthesis of argumentative texts.

5.1 Experimental Set-up for Manual Argumentation Synthesis following Rhetorical Strategies

We hypothesized that humans (1) agree when synthesizing arguments for the same strategy more than for different ones, and (2) synthesize differently depending on their strategy, especially in terms of selection.

Participants A diverse set of 26 qualified experts participated in the study (10 female, 16 male) with diverse demographic backgrounds (9 from North America, 11 from Europe, 6 from Asia). 16 came from groups related to computational linguistics, while the others were writing experts, acquired on *upwork.com*. 16 had a PhD or master, 7 a bachelor, and 3 a high school degree. No author of this paper participated.

Strategies To emphasize the effects of a strategical synthesis, we consider only two somehow principled rhetorical strategies, for which we gave the following short intuitions:

[3] In case of the question "Should shopping malls generally be allowed to open on holidays and Sundays?", pro and con theses were balanced (4 each). We chose con, so we ended up with five pro-topic and five con-topic cases in total.

[4] We provide the output of all main steps of the described dataset construction at: `http://arguana.com/data`

Unit	Strategy	Selection		Arrangement	
		Majority	κ	Majority	κ
Thesis	Logos-oriented	59%	* 0.25	51%	0.16
	Pathos-oriented	51%	* 0.10	57%	0.23
	Logos to pathos	**43%**	**−0.02**	**53%**	**0.18**
Con	Logos-oriented	52%	0.13	45%	0.11
	Pathos-oriented	49%	0.10	54%	0.17
	Logos to pathos	**46%**	**0.08**	**48%**	**0.15**
3 pros	Logos-oriented	20%	* 0.14	31%	0.04
	Pathos-oriented	21%	* 0.11	31%	0.04
	Logos to pathos	**18%**	**0.04**	**30%**	**0.07**
Overall	Logos-oriented	59%	* 0.16	48%	0.10
	Pathos-oriented	52%	* 0.11	54%	0.15
	Logos to pathos	**46%**	**0.04**	**49%**	**0.13**

Table 3: Majority and mean Cohen's κ agreement in selecting and arranging units of each type within the arguments of each principled strategy and across (*logos to pathos*).

Connection	Phrasing		
	Majority	κ	Top Category
Before initial thesis	94%	0.35	None
Before initial con	40%	−0.22	Comparison
Before initial pro	92%	−0.21	None
Thesis → pro	46%	−0.29	Comparison
Thesis → con	41%	−0.32	None
Con → thesis	57%	−0.24	None
Con → pro	48%	−0.37	Comparison
Pro → thesis	69%	−0.26	Contingency
Pro → con	39%	−0.25	Comparison
Pro → pro	37%	−0.18	Expansion

Table 4: Majority and mean Cohen's κ agreement, as well as the top category of discourse markers and connectives in phrasing for each connection between unit types.

- *Logos-oriented.* Argue based on logical reasoning, which means to make rational and logical conclusions towards the intended stance on the given topic.
- *Pathos-oriented.* Argue based on emotional reasoning, which means to appeal to the emotions of the reader regarding the topic within your arguments.[5]

Tasks All participants had to create one short argumentative text for each topic-stance pair, five logos-oriented and five pathos-oriented texts. Pair-strategy combinations varied across participants, but were balanced in overall terms. For each pair, the participants first had to *select* one thesis unit, one con unit, and three pro units from our pool that they thought could best form a persuasive argument following the given strategy.[6] Then, they had to *arrange* these five units in the most suitable way they saw. Finally, they should *phrase* a coherent text by adding discourse markers and connectives (example terms were given) as well as punctuation marks before and after each unit. We provided a spreadsheet that made this process as easy as possible. The participants were not trained in order to avoid biasing them towards our hypotheses.

5.2 Agreement and Differences in Selection, Arrangement, and Phrasing

The results of our experiment is a benchmark dataset with 13 manually synthesized texts for each combination of topic-stance pair and principled strategy. We provide it at `http://arguana.com/data`.

Agreement To make strategy differences explicit, Table 3 shows the participants' agreement in selecting units and in arranging the three unit types for arguments *within* each of the two strategies and *across* them ("logos to pathos"). Majority agreement is given when over 50% of all participants decided equally about choosing a unit, and the mean pairwise Cohen's κ agreement captures the average agreement of two participants. It is expected that the κ values are low, since various meaningful arguments can be synthesized for each topic from our pool. Accordingly, other measures such as Fleiss' κ are not suitable.[7]

For the *selection*, the decisive observation is that the κ is much higher within each strategy (e.g., 0.25 for logos and 0.10 for pathos in case of *thesis*) than across (−0.02). κ differences between each strategy and across marked with * are significant with at least $p < 0.05$ (t-test in case of normal distribution, Wilcoxon test otherwise). This speaks for the truth of Hypothesis 1, i.e., unit selection depends on the strategy. The majority agreement indicates that, especially for logos-oriented arguments, *thesis* (59%) and *con* (52%) are often selected uniformly. In contrast, the agreement for *arrangement* is similar within and across strategies, rather deviating between logos and pathos. We further explore this finding below.

[5]We told the participants that the two strategies may overlap. Ethos-oriented argumentation was not considered, because the demonstration of credibility seems hard, given that we required the participants to build arguments from our given pool of units.

[6]We use units as proxies for frames in the selection, as we do not have information on the exact frames covered by the units.

[7]High κ values would even be doubtful, because they would imply that the participants often selected *exactly the same* units, which would render the potential of our pool of argumentative discourse units questionable.

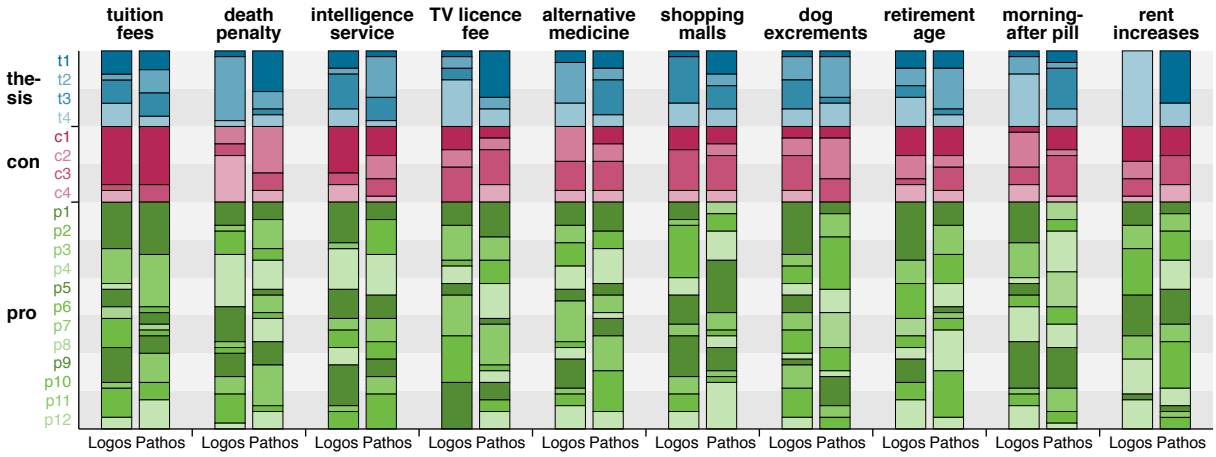


Figure 2: Distribution of the thesis units t1–t4 (blue), con units c1–c4 (red), and pro units p1–p12 (green) selected by all 13 annotators for each combination of topic and rhetorical strategy (logos/pathos-oriented).

Unit	Strategy	1	2	3	4	5	Mean
Thesis	Logos	50%	21%	10%	2%	17%	2.1
	Pathos	56%	18%	5%	4%	17%	2.1
	Overall	**53%**	**19%**	**8%**	**3%**	**17%**	**2.1**
Con	Logos	23%	42%	24%	8%	2%	2.3
	Pathos	25%	51%	17%	5%	2%	2.1
	Overall	**24%**	**47%**	**20%**	**7%**	**2%**	**2.2**
Pro	Logos	9%	12%	22%	30%	27%	3.5
	Pathos	6%	11%	26%	30%	27%	3.6
	Overall	**8%**	**11%**	**24%**	**30%**	**27%**	**3.6**

Table 5: Difference between strategies in the distribution of unit types over the five positions of the synthesized arguments, and mean rank of each type.

Unit Type Flow	Means	Frequency	Rank
(thesis, con, pro, pro, pro)	Logos	34.6%	1
	Pathos	43.1%	1
	Overall	**38.8%**	**1**
(con, thesis, pro, pro, pro)	Logos	13.1%	2
	Pathos	13.8%	2
	Overall	**13.5%**	**2**
(thesis, pro, con, pro, pro)	Logos	12.3%	3
	Pathos	12.3%	3
	Overall	**12.3%**	**3**

Table 6: Differences between strategies in the relative frequency and the rank of the three most common unit type flows in the synthesized arguments.

For *phrasing*, we just show in Table 4 to what extent the experts agreed in connecting consecutive unit types. We assume agreement if the same category of discourse marker or connective was used, distinguishing comparison, contingency, concession, expansion, other, and none. The majority agreement is high in many cases, while we see systematic κ disagreement. This implies that there were two opposing camps, although statistical reliability is limited. Quite intuitively, comparison terms (e.g., "although" or "but") come often before or after con, whereas consecutive pros usually expand each other (e.g., with "and").

Differences For brevity, we compare the relative frequencies of *selecting* each unit in logos-oriented and pathos-oriented arguments visually in Figure 2. For the theses, we observe clear strategy differences. For instance, nearly all logos texts on *death penalty* use t2, whereas the majority selected t1 for pathos there. In general, the thesis selection seems more uniform for logos. The distribution of con units is rather similar across strategies for many topics, suggesting that natural candidates to be rebutted exist among these. Conversely, the stacked bars of the pro units diverge often, as in the case of *dog excrements*. As far as our limited unit pool permits, these results support Hypothesis 2, i.e., different strategies lead to different selections of units and topic frames.

We analyze the *arrangement* in terms of the distribution of each unit type over the five positions in the synthesized texts as well as in terms of the resulting sequential unit type flows. As Table 5 reveals, the distributions are very stable across strategies. About half of all texts begin with the thesis, whereas pro units rather come at the end. Only for the con units, we see some differences, such as 42% (logos-oriented) versus 51% (pathos-oriented) at position 2. Accordingly, both strategies yield the same three most common flows, listed in Table 6, with *(thesis, con, pro, pro, pro)* at rank 1. We omit significance tests, since these results rather entail the conclusion that arrangement is not specific to the given strategies.

	Comparison		Contingency		Concession		Expansion		Other		None
Strategy	**Top**	**Frequency**	**Top**	**Frequency**	**Top**	**Frequency**	**Top**	**Frequency**	**Top**	**Frequency**	**Frequency**
Logos	but	20%	because	16%	indeed, of course	6%	and	15%	for one thing	2%	42%
Pathos	but	21%	because	18%	indeed, admittedly	8%	and	13%	for one thing	2%	39%
Overall	**but**	**20%**	**because**	**17%**	**indeed, admittedly**	**7%**	**and**	**14%**	**for one thing**	**2%**	**41%**

Table 7: The top terms and relative frequency of each discourse marker/connective category in phrasing.

Strategy	**#**	**Text Manually Synthesized From Five Argumentative Discourse Units**
Logos-oriented	t2	Germany should not introduce capital punishment.
	c4	Proponents of the death penalty count on its deterring effect as well as and the ultimate elimination of any potential threat.
	p5	*However,* despite the death penalty there are significantly more homicides in the US than in Germany.
	p4	*Furthermore,* courts are subject to human error.
	p9	It turns out time and again that innocent people are convicted and executed.
Pathos-oriented	t1	Germany should by no means introduce capital punishment.
	c2	Many people think that a murderer has already decided on the life or death of another person.
	p2	*Still,* a door must remain open for making amends.
	p6	Every human, even those who have committed a despicable crime, can bring themselves to regret and change their opinion.
	p11	*Therefore,* no one may have the right to adjudicate upon the death of another human being.

Table 8: Comparison of two selected argumentative texts, one for each strategy, manually synthesized based on the units from Table 2. The italicized discourse markers have been added by the participants.

Similar holds for the *phrasing* of argumentative texts, or at least for the given task of adding discourse markers and connectives. In particular, Table 7 presents the proportion of the most frequent categories of these. Logos-oriented and pathos-oriented arguments are practically identical in this regard, even the most often used terms in each category match. Comparisons (especially contrasts with "but") as well as causal contingency markers dominate the applied phrasing operators.

Insights Exemplarily, Table 8 compares two arguments against the death penalty synthesized by different participants, one for each strategy. The theses differ in the emotional load rather than their framing. While the logos-oriented argument frames death penalty in terms of its potential deterrent effect and the execution of innocent people, the pathos-oriented puts full emphasis on the ability of humans to regret and change. The different numbers of frames lead to a slightly different phrasing with discourse markers. Matching our results, the arrangement of both arguments is the same on the abstraction level of unit types.

6 Conclusion

We propose a general model of synthesizing argumentation following rhetorical strategies in terms of Aristotle's means of persuasion: logos, ethos, pathos. The model idealizes the synthesis as the selection, arrangement, and phrasing of argumentative discourse units (ADUs). Before we develop computational approaches based on the model, this paper has evaluated its general adequacy in an experiment with human experts. The results provide evidence that humans agree significantly more when synthesizing argumentative texts following the same strategy. In addition, we found that the arrangement of the ADUs and the re-phrasing of their connections is hardly affected by the strategy at all. A study of the phrasing of the actual units is left to future work. In the long term, we envisage a system that is able to automatically generate *effective* argumentation. Such a system requires two main types of resources: (1) a large pool of decontextualized ADUs covering diverse topics, and (2) a sufficiently flexible set of select, arrange, and phrase operators along with information about their effectiveness for specific topics and about the means of persuasion they encode. Given corpora with respective annotations, both resources can be developed using existing natural language processing techniques. We see this as the next step towards our goal.

Acknowledgments Thanks to Yamen Ajjour, Wei-Fan Chen, Yulia Clausen, Debopam Das, Erdan Genc, Tim Gollub, Yulia Grishina, Erik Hägert, Johannes Kiesel, Lukas Paschen, Martin Potthast, Robin Schäfer, Constanze Schmitt, Uladzimir Sidarenka, Shahbaz Syed, and Michael Völske for taking part in our study.

References

Yamen Ajjour, Wei-Fan Chen, Johannes Kiesel, Henning Wachsmuth, and Benno Stein. 2017. Unit segmentation of argumentative texts. In *Proceedings of the 4th Workshop on Argument Mining*, pages 118–128. Association for Computational Linguistics.

Khalid Al-Khatib, Henning Wachsmuth, Johannes Kiesel, Matthias Hagen, and Benno Stein. 2016. A news editorial corpus for mining argumentation strategies. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3433–3443. The COLING 2016 Organizing Committee.

Khalid Al-Khatib, Henning Wachsmuth, Matthias Hagen, and Benno Stein. 2017. Patterns of argumentation strategies across topics. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1351–1357. Association for Computational Linguistics.

Jens Allwood. 2016. Argumentation, activity and culture. In *6th International Conference on Computational Models of Argument*, page 3.

P. Anand, J. King, Jordan Boyd-Graber, E. Wagner, C. Martell, Douglas Oard, and Philip Resnik. 2011. Believe me — We can do this! Annotating persuasive acts in blog text. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Aristotle. 2007. *On Rhetoric: A Theory of Civic Discourse* (George A. Kennedy, Translator). Clarendon Aristotle series. Oxford University Press.

Regina Barzilay, 2010. *Probabilistic Approaches for Modeling Text Structure and Their Application to Text-to-Text Generation*, pages 1–12. Springer Berlin Heidelberg.

Yonatan Bilu and Noam Slonim. 2016. Claim synthesis via predicate recycling. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 525–530. Association for Computational Linguistics.

Amparo Elizabeth Cano-Basave and Yulan He. 2016. A study of the impact of persuasive argumentation in political debates. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1405–1413. Association for Computational Linguistics.

Giuseppe Carenini and Johanna D. Moore. 2006. Generating and evaluating evaluative arguments. *Artificial Intelligence*, 170(11):925–952.

Rory Duthie, Katarzyna Budzynska, and Chris Reed. 2016. Mining ethos in political debate. In *6th International Conference on Computational Models of Argument (COMMA 16)*, pages 299–310.

Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 987–996. Association for Computational Linguistics.

James B. Freeman. 2011. *Argument Structure: Representation and Theory*. Springer.

Clément Gautrais, Peggy Cellier, René Quiniou, and Alexandre Termier. 2017. Topic signatures in political campaign speeches. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2342–2347. Association for Computational Linguistics.

Nancy L. Green. 2017. Argumentation scheme-based argument generation to support feedback in educational argument modeling systems. *International Journal of Artificial Intelligence in Education*, 27(3):515–533.

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

Ivan Habernal and Iryna Gurevych. 2016. What makes a convincing argument? Empirical analysis and detecting attributes of convincingness in web argumentation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1214–1223. Association for Computational Linguistics.

Christopher Hidey, Elena Musi, Alyssa Hwang, Smaranda Muresan, and Kathy McKeown. 2017. Analyzing the semantic types of claims and premises in an online persuasive forum. In *Proceedings of the 4th Workshop on Argument Mining*, pages 11–21.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596.

Ralph H. Johnson and J. Anthony Blair. 2006. *Logical Self-defense*. International Debate Education Association.

Taeda Jovičić. 2006. The effectiveness of argumentative strategies. *Argumentation*, 20:29–58.

John Lawrence, Jacky Visser, and Chris Reed. 2017. Harnessing rhetorical figures for argument mining. *Argument & Computation*, 8(3):289–310.

Stephanie Lukin, Pranav Anand, Marilyn Walker, and Steve Whittaker. 2017. Argument Strength is in the Eye of the Beholder: Audience Effects in Persuasion. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 742–753. Association for Computational Linguistics.

Stefano Menini, Federico Nanni, Simone Paolo Ponzetto, and Sara Tonelli. 2017. Topic-based agreement and disagreement in us electoral manifestos. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2938–2944. Association for Computational Linguistics.

Hugo Mercier and Dan Sperber. 2011. Why do humans reason? Arguments for an argumentative theory. *Behavioral and Brain Sciences*, 34:57–111.

Nona Naderi and Graeme Hirst. 2017. Classifying frames at the sentence level in news articles. In *Recent Advances in Natural Language Processing*, pages 536–542, 11.

Andreas Peldszus and Manfred Stede. 2015. Joint prediction in mst-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 938–948. Association for Computational Linguistics.

Andreas Peldszus and Manfred Stede. 2016. An annotated corpus of argumentative microtexts. In *Argumentation and Reasoned Action: 1st European Conference on Argumentation (ECA 16)*. College Publications.

Isaac Persing, Alan Davis, and Vincent Ng. 2010. Modeling organization in student essays. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 229–239. Association for Computational Linguistics.

Paul Reisert, Naoya Inoue, Naoaki Okazaki, and Kentaro Inui. 2015. A computational approach for generating Toulmin model argumentation. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 45–55. Association for Computational Linguistics.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

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

Ariel Rosenfeld and Sarit Kraus. 2016. Strategical argumentative agent for human persuasion. In *Proceedings of the 22nd European Conference on Artificial Intelligence*, pages 320–328.

Misa Sato, Kohsuke Yanai, Toshinori Miyoshi, Toshihiko Yanase, Makoto Iwayama, Qinghua Sun, and Yoshiki Niwa. 2015. End-to-end argument generation system in debating. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 109–114. Association for Computational Linguistics and The Asian Federation of Natural Language Processing.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6833–6844.

Yi Song, Michael Heilman, Beata Beigman Klebanov, and Paul Deane. 2014. Applying argumentation schemes for essay scoring. In *Proceedings of the First Workshop on Argumentation Mining*, pages 69–78. Association for Computational Linguistics.

Wei Song, Dong Wang, Ruiji Fu, Lizhen Liu, Ting Liu, and Guoping Hu. 2017. Discourse mode identification in essays. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 112–122. Association for Computational Linguistics.

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56. Association for Computational Linguistics.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of the 25th International World Wide Web Conference*, pages 613–624.

Stephen E. Toulmin. 1958. *The Uses of Argument*. Cambridge University Press.

Frans H. van Eemeren, Bart Garssen, Erik C. W. Krabbe nad A. Francisca Snoeck Henkemans, Bart Verheij, and Jean H. M. Wagemans. 2014. *Handbook of Argumentation Theory*. Springer Netherlands.

Henning Wachsmuth and Benno Stein. 2017. A universal model for discourse-level argumentation analysis. *ACM Transaction of Internet Technology*, 17(3):28:1–28:24.

Henning Wachsmuth, Khalid Al Khatib, and Benno Stein. 2016. Using argument mining to assess the argumentation quality of essays. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1680–1691. The COLING 2016 Organizing Committee.

Henning Wachsmuth, Giovanni Da San Martino, Dora Kiesel, and Benno Stein. 2017. The impact of modeling overall argumentation with tree kernels. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2379–2389. Association for Computational Linguistics.

Douglas Walton, Christopher Reed, and Fabrizio Macagno. 2008. *Argumentation Schemes*. Cambridge University Press.

Lu Wang, Nick Beauchamp, Sarah Shugars, and Kechen Qin. 2017. Winning on the merits: The joint effects of content and style on debate outcomes. *Transactions of the Association of Computational Linguistics*, 5:219–232.

Toshihiko Yanase, Toshinori Miyoshi, Kohsuke Yanai, Misa Sato, Makoto Iwayama, Yoshiki Niwa, Paul Reisert, and Kentaro Inui. 2015. Learning sentence ordering for opinion generation of debate. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 94–103. Association for Computational Linguistics.

R. Michael Young, Johanna D. Moore, and Martha E. Pollack. 1994. Towards a principled representation of discourse plans. *CoRR*, abs/cmp-lg/9406004.

Justine Zhang, Arthur Spirling, and Cristian Danescu-Niculescu-Mizil. 2017. Asking too much? the rhetorical role of questions in political discourse. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1558–1572. Association for Computational Linguistics.

Ingrid Zukerman, Richard McConachy, and Kevin B. Korb. 2000. Using argumentation strategies in automated argument generation. In *First International Conference on Natural Language Generation (INLG 00)*, pages 55–62.

A Dataset for Building Code-Mixed Goal Oriented Conversation Systems

Suman Banerjee, Nikita Moghe, Siddhartha Arora and **Mitesh M. Khapra**
Indian Institute of Technology Madras, India
{suman, nikitavam, sidarora, miteshk}@cse.iitm.ac.in

Abstract

There is an increasing demand for goal-oriented conversation systems which can assist users in various day-to-day activities such as booking tickets, restaurant reservations, shopping, *etc*. Most of the existing datasets for building such conversation systems focus on monolingual conversations and there is hardly any work on multilingual and/or code-mixed conversations. Such datasets and systems thus do not cater to the multilingual regions of the world, such as India, where it is very common for people to speak more than one language and seamlessly switch between them resulting in code-mixed conversations. For example, a Hindi speaking user looking to book a restaurant would typically ask, "Kya tum is *restaurant* mein ek *table book* karne mein meri *help* karoge?" ("Can you help me in booking a table at this restaurant?"). To facilitate the development of such code-mixed conversation models, we build a goal-oriented dialog dataset containing code-mixed conversations. Specifically, we take the text from the DSTC2 restaurant reservation dataset and create code-mixed versions of it in Hindi-English, Bengali-English, Gujarati-English and Tamil-English. We also establish initial baselines on this dataset using existing state of the art models. This dataset along with our baseline implementations is made publicly available for research purposes.

1 Introduction

Over the past few years, there has been an increasing demand for virtual assistants which can help users in a wide variety of tasks in several domains such as entertainment, finance, healthcare, e-commerce, *etc.* To cater to this demand, several commercial conversation systems such as Siri, Cortana, Allo have been developed. While these systems are still far from general purpose open domain chat, they perform reasonably well for certain goal-oriented tasks such as setting alarms/reminders, booking appointments, checking movie show timings, finding directions for navigation, *etc*. Apart from these commercial systems, there has also been significant academic research to advance the state of the art in conversation systems (Shang et al., 2015; Vinyals and Le, 2015; Yao et al., 2015; Li et al., 2016a; Li et al., 2016b; Serban et al., 2017). Most of this academic research is driven by publicly available datasets such as Twitter conversation dataset (Ritter et al., 2010), Ubuntu dialog dataset (Lowe et al., 2015), Movie subtitles dataset (Lison and Tiedemann, 2016) and DSTC2 restaurant reservation dataset (Henderson et al., 2014a). In this work, we focus on goal-oriented conversations such as the ones contained in the DSTC2 dataset.

Most of the datasets and state of the art systems mentioned above are monolingual. Specifically, all the utterances and responses in the conversations are in one language (typically, English) and there are no multilingual and/or code-mixed utterances/responses. However, in several multilingual regions of the world, such as India, it is natural for speakers to produce utterances and responses which are multilingual and code-mixed. For example, Table 1 shows real examples of how bilingual speakers from India talk when requesting someone to help them reserve a restaurant or book movie tickets. As it can

Proceedings of the 27th International Conference on Computational Linguistics, pages 3766–3780
Santa Fe, New Mexico, USA, August 20-26, 2018.

Languages	Utterances
English	**Speaker 1**: Hi, Can you help me in booking a table at this restaurant? **Speaker 2**: Sure, would you like something in cheap, moderate or expensive price range?
Hindi-English	**Speaker 1**: *Hi*, kya tum is *restaurant* mein ek *table book* karne mein meri *help* karoge? **Speaker 2**: *Sure*, kya aap *cheap*, *moderate* ya *expensive price range* mein kuch *like* karenge?
Bengali-English	**Speaker 1**: *Hi*, tumi ki ei *restaurant* ey ekta *table book* korte *help* korbe amake? **Speaker 2**: *Sure*, aapni ki *cheap, moderate* na *expensive price range* ey kichu *like* korben ?
English	**Speaker 1**: Hello, can you tell me about the show timings of "Black Panther"? **Speaker 2**: Sure, would you like to book tickets for today or any other day?
Gujarati-English	**Speaker 1**: *Hello*, mane Black Panther na *show timings* janavo. **Speaker 2**: *Sure*, shu tame aaj ni ke koi anya divas ni *ticket book* karva mango cho?
Tamil-English	**Speaker 1**: *Hello*, "Black Panther" *show timings* eppo epponu solla mudiuma **Speaker 2**: Kandipa, *tickets* innaiku *book* pannanuma illana vera ennaikku?

Table 1: Example code-mixed utterances in the specified languages.

be seen, when engaging in such informal conversations it is very natural for such speakers to use code-mixed utterances, mixing their native language with English. Apart from India, such code-mixing is also prevalent in other multilingual regions of the world, for example, Spanglish (Spanish-English), Frenglish (French-English), Porglish (Portuguese-English) and so on. To cater to such users, it is essential to create datasets containing code-mixed conversations and thus facilitate the development of code-mixed conversation systems.

With the above motivation, we build a dataset containing code-mixed goal-oriented conversations for four Indian languages. Specifically, we take every utterance from the DSTC2 restaurant reservation dataset and ask a mix of in-house and crowdsourced workers to create a corresponding code-mixed utterance involving their native language and English. We simply instructed the workers to (i) assume that they were chatting with a friend who spoke the same native language as them in addition to English, (ii) not try very hard to translate the sentence completely to their native language but feel free to switch to English whenever they wanted (just as they would in a normal conversation with a friend) and (iii) use Romanized text instead of the native language's script. The resulting dataset contains utterances of the type shown in Table 1. We found that 87.73% of the created utterances were code-mixed, 7.18% had only English words and 5.09% had only native language words. The four languages that we chose were Hindi, Bengali, Tamil and Gujarati which have 422M, 83M, 60M and 46M native speakers respectively.

Apart from reporting various statistics about this data (such as CM-index (Gambäck and Das, 2016) and I-index (Guzmán et al., 2016)), we also report some initial baselines by evaluating some state of the art approaches on the proposed dataset. Specifically, we evaluate a standard sequence-to-sequence model with an attention mechanism (Bahdanau et al., 2015) and a hierarchical recurrent encoder-decoder model (Serban et al., 2016). Our code implementing these models along with the dataset is available freely for research purposes[1]. To the best of our knowledge, this is the first conversation dataset containing code-mixed conversations and will hopefully enable further research in this area. In particular, since the data is 5-way parallel (English, Bengali, Hindi, Tamil, Gujarati) it would be useful for building jointly trained code-mixed models.

2 Related Work

Serban et al. (2015) report an excellent (and up-to-date) survey of existing dialog datasets. For brevity, we only mention some of the important points from their survey and refer the reader to the original paper for more details. To begin with, we note that existing dialog datasets can be categorized along 3 main dimensions. The first dimension is the modality of the dataset, *i.e.*, whether the dataset contains spoken conversations (Godfrey et al., 1992; Kim et al., 2016) or text conversations (Forsythand and Martell, 2007; Ritter et al., 2010; Lowe et al., 2015). The second dimension is whether the dataset contains goal-oriented conversations or open-ended conversations. A goal-oriented conversation typically involves chatting for the sake of completing a task such as the Dialog State Tracking Challenge (DSTC) datasets which involve tasks for reserving a restaurant (Henderson et al., 2014a), checking bus

[1] https://github.com/sumanbanerjee1/Code-Mixed-Dialog

User	Bot
1. <SILENCE>	Hello, welcome to the Cambridge restaurant system. You can ask for restaurants by area, price range or food type. How may I help you?
2. i'm looking for a moderately priced restaurant.	What kind of food would you like?
3. it doesn't matter.	What part of town do you have in mind?
4. the south part of town.	api_call R_cuisine south moderate
pizza_hut_cherry_hinton R_post_code pizza_hut_cherry_hinton_post_code pizza_hut_cherry_hinton R_cuisine italian pizza_hut_cherry_hinton R_location south pizza_hut_cherry_hinton R_phone pizza_hut_cherry_hinton_phone pizza_hut_cherry_hinton R_address pizza_hut_cherry_hinton_address pizza_hut_cherry_hinton R_price moderate pizza_hut_cherry_hinton R_rating 3 restaurant_alimentum R_post_code restaurant_alimentum_post_code restaurant_alimentum R_cuisine european restaurant_alimentum R_location south restaurant_alimentum R_phone restaurant_alimentum_phone restaurant_alimentum R_address restaurant_alimentum_address restaurant_alimentum R_price moderate restaurant_alimentum R_rating 10	
5. <SILENCE>	restaurant_alimentum is a nice restaurant in the south of town serving modern_european food.
6. can i get the address?	Sure, restaurant_alimentum is on restaurant_alimentum_address.
7. and whats the phone number for that?	The phone number of restaurant_alimentum is restaurant_alimentum_phone.
8. okay, thank you, good bye.	you are welcome.

Table 2: An example chat from the English version of DSTC2 dataset (Bordes and Weston, 2017).

schedules (Williams et al., 2013), collecting tourist information (Henderson et al., 2014b) and so on. Such datasets are also typically domain-specific. Open-ended conversations on the other hand involve general chat on any topic and there is no specific end task. Some popular examples of datasets containing such open-ended conversations are the Ritel Corpus (Rosset and Petel, 2006), NPS Chat Corpus (Forsythand and Martell, 2007), Twitter Corpus (Ritter et al., 2010), *etc*. The third dimension is whether the dataset contains human-human conversations or human-bot conversations. As the name suggests, human-bot conversation datasets contain conversations between humans and an existing conversation system (typically a domain-specific goal-oriented bot) (Williams et al., 2013; Henderson et al., 2014a; Henderson et al., 2014b). Human-human conversations, on the other hand, can contain spontaneous conversations between humans, as are typically observed in discussion forums (Walker et al., 2012), chat rooms (Forsythand and Martell, 2007), SMS messages (Chen and Kan, 2013) and so on. Human-human conversations can also contain scripted dialogs such as scripts of movies (Banchs, 2012), TV shows (Roy et al., 2014), *etc*. It is surprising that of the 63 conversation datasets developed in the past (Serban et al., 2015), none contain multilingual conversations. In particular, none of them contain code-mixed conversations from multilingual regions of the world. There is clearly a need to fill this gap and we believe that the dataset developed as a part of this work is a small step in that direction.

In general, the research community has been interested in developing datasets, tools and approaches for code-mixed content. This interest is largely triggered by the abundance of code-mixed content found in chats, emails, social media platforms, *etc*. In the context of such code-mixed content, existing works have looked at the problems of language identification (Nguyen and Dogruöz, 2013; Solorio et al., 2014; Barman et al., 2014; Molina et al., 2016), part-of-speech tagging (Barman et al., 2016; Ghosh et al., 2016; AlGhamdi et al., 2016), user profiling (Khapra et al., 2013), topic modeling (Rosner and Farrugia, 2007), information retrieval (Chakma and Das, 2016) and language modeling (Adel et al., 2013a; Adel et al., 2013b; Adel et al., 2015). However, to the best of our knowledge, ours is the first work on developing code-mixed conversation systems for goal-oriented dialogs.

# of Utterances	49167
# of Unique utterances	6733
Average # of utterances per dialog	15.19
Average # of words per utterance	7.71
Average # of words per dialog	120.33
Average # of KB triples per dialog	38.24
# of Train Dialogs	1168
# of Validation Dialogs	500
# of Test Dialogs	1117
Vocabulary size	1229

Table 3: Statistics of the English version of DSTC2 dataset

3 Background: DSTC2 Restaurant Reservation Dataset

We build on top of the goal-oriented restaurant reservation dialog dataset which was released as part of the second Dialog State Tracking Challenge (DSTC2) (Henderson et al., 2014a). This dataset contains conversations between crowdsourced workers and existing dialog systems (bots). Specifically, the workers were asked to book a table at a restaurant with the help of a bot. These dialog systems consisted of modules like automatic speech recognizer, natural language interpreter, dialog manager, response generator and a speech synthesizer (Young, 2000). The dialog manager used polices which were either hand-crafted or learned by formulating the problem as a partially observable Markov decision process (POMDP) (Williams and Young, 2007). The speech input from the user was first converted to text and then fed to the dialog system. For this, the authors used two Automatic Speech Recognition (ASR) modules out of which one was artificially degraded in order to simulate noisy environments. The workers could request for restaurants based on 3 slots: *area* (5 possible values), *cuisine* (91 possible values) and *price range* (3 possible values). The workers were also instructed to change their goals and look for alternative *areas*, *cuisines* and *price ranges* in the middle of the dialog. This was done to account for the unpredictability in natural conversations. The conversations were then transcribed and the utterances were labeled with different dialog states. For example, each utterance was labeled with its semantic intent representation (*request[area], inform[area = north]*) and the dialog turns were labeled with annotations such as constraints on the slots (*cuisine = italian*), requested slots (*requested* = {*phone, address*}) and the method of search (*by_constraints, by_alternatives*). Such annotations are useful for domain-specific slot-filling based dialog systems.

Bordes and Weston (2017) argued that for various domains collecting such explicit annotations for every state in the dialog is tedious and expensive. Instead, they emphasized on building end-to-end dialog systems (as opposed to slot-filling based systems) by adapting this dataset and treating it as a simple sequence of utterance-response pairs (without any explicit dialog states associated with the utterances). In addition, the authors also created API calls which can be issued to an underlying Knowledge Base (KB) and appended the resultant KB triples to each dialog. Table 2 shows one small sample dialog from this adapted dataset along with the API calls. Notice that the API call uses the information of all the constraints specified by the user so far and then receives all triples from the restaurant KB which match the user's requirements. This dataset facilitated the development of models (Bordes and Weston, 2017; Seo et al., 2017; Williams et al., 2017; Eric and Manning, 2017) which just predict the bot utterances and API calls without explicitly tracking the slots. Table 3 reports the statistics of this dataset. In this work, we create code-mixed versions of this dataset in 4 different languages as described below.

4 Code-Mixed Dialog Dataset

In this section, we describe the process used for creating a new dataset containing code-mixed conversations. Specifically, we describe (i) the process used for extracting unique utterance templates from the original DSTC2 dataset, (ii) the process of creating code-mixed translations of these utterances with the help of in-house and crowdsourced workers and (iii) the process used for evaluating the collected conversations. Finally, we report some statistics about the dataset.

4.1 Extracting Unique Utterance Templates

We found that many utterances in the original English version of DSTC2 dataset (henceforth referred to as En-DSTC2) have the same sentence structure but only differ in the values of the *areas*, *cuisines*, *price ranges* and entities such as *restaurant names, addresses, phone numbers* and *post codes*. For example, consider these two sentences which only differ in the *area* and *cuisine*: (i) "Sorry, there is no *chinese* restaurant in the *north* part of town." and (ii) "Sorry, there is no *italian* restaurant in the *west* part of town". Both these sentences can be thought of as instantiations of the generic template: "Sorry, there is no [CUISINE] restaurant in the [AREA] part of town." wherein the placeholders [AREA] and [CUISINE] get replaced by different values. We used the KB provided by Bordes and Weston (2017) to find all the entities appearing in all the utterances and replaced them by placeholders such as: [AREA], [CUISINE], [PRICE], [RESTAURANT], [ADDRESS], [PHONE] and [POST_CODE]. Further, since the authors had mentioned that the KB provided was not perfect/complete, we did some manual inspection to find all such entities and came up with a list of 536 such entity words. After replacing these words with their respective placeholders we obtained 3590 unique English utterances.

4.2 Creating Code-Mixed Translations

According to Myers-Scotton (1993) code-mixing involves a native language which provides the morphosyntactic frame and a foreign language whose *linguistic units* such as *phrases, words* and *morphemes* are inserted into this morphosyntactic frame. The native language is called the *Matrix* while the foreign language is called the *Embedding*. Our work focuses on creating a conversation dataset wherein 4 different Indian languages, *viz.*, Hindi, Bengali, Gujarati and Tamil serve as the *Matrix* and English serves as the *Embedding*. We used a mix of in-house and crowdsourced workers to create a code-mixed version of the original DSTC2 dataset. For example, for Hindi and Gujarati, we did not have enough in-house speakers so we completely relied on crowdsourcing for creating the data but then used in-house workers to verify the collected data. For Bengali, all the data was created by in-house annotators who were native Bengali speakers and proficient in English. Lastly, for Tamil, roughly 40% of the data was created with the help of crowdsourced workers and the rest with the help of in-house workers. Irrespective of whether the workers were crowdsourced or in-house we used the same set of instructions as described below.

We instructed the annotators to assume that they were chatting with a friend who is a native speaker of Hindi (or Gujarati, Bengali, Tamil) but also speaks English well (typically, because English was the language in which the friend did most of his/her education). To explain the idea of code-mixing, we showed them example utterances where it was natural for the user to mix English words while chatting in the native language. They were then shown an English utterance from the DSTC2 dataset and asked to create its code-mixed translation in the native language keeping the above code-mixed examples in mind. They were asked to use Roman script irrespective of whether the word being used belongs to English or the native language (in particular, they were clearly instructed to not use the native language's script). As expected, we observed that while translating, the annotators tend to retain some difficult-to-translate and colloquially relevant English words as it is. The annotators were also clearly instructed to refrain from producing pure translations (*i.e.*, they were asked to not try hard to translate English words which they would typically not translate in an informal conversation). Also, the annotators were instructed to retain the placeholder words ([AREA], [CUISINE], *etc.*) as it is and not translate them.

We used Amazon Mechanical Turk (AMT) as the platform for crowdsourcing. Each Human Intelligence Task (HIT) required the user to give code-mixed translations of 5 utterances and was priced at $0.2. Once we collected the code-mixed translations of all the utterance templates that were extracted using the procedure described in the previous subsection, we then instantiated them into proper sentences by replacing the placeholders ([AREA], [CUISINE], *etc.*) with the corresponding entities as present in the original DSTC2 dataset. For every dialog in the original DSTC2 dataset, every utterance was then replaced by its code-mixed translation resulting in an end-to-end code-mixed conversation.

		In-house Workers	Evaluators
Avg. Age		25.2	24.6
Gender	**Female**	33.3%	25.0%
	Male	66.7%	75.0%
Highest Education	**Undergraduate**	25.0%	33.3%
	Graduate	41.7%	33.3%
	Postgraduate	33.3%	33.3%
English Medium Schooling	**Yes**	100%	100%
	No	0%	0%
Frequency of English usage	**Frequently**	75.0%	91.7%
	Occasionally	25.0%	8.3%
	Rarely	0%	0%
Frequency of native language usage	**Frequently**	100%	91.7%
	Occasionally	0%	8.3%
	Rarely	0%	0%

Table 4: Demographic details of the in-house workers and the human evaluators.

Datasets	Colloquialism	Intelligibility	Coherent
Hi-DSTC2	4.20	4.06	4.21
Be-DSTC2	4.07	4.05	4.11
Gu-DSTC2	3.66	3.60	3.76
Ta-DSTC2	4.17	3.96	3.93

Table 5: Average human ratings for different metrics.

4.3 Evaluating the Collected Dataset

We did evaluations at two levels. The first evaluation was at the level of utterances wherein if the code-mixed translation of an utterance was obtained via crowdsourcing, then we got this translation verified by in-house evaluators. The evaluators were asked to check if (i) the translation was faithful to the source sentence, (ii) the code-mixing was natural and not forced and (iii) all translations used Roman script and not the native language's script. Any utterance which was flagged as erroneous by the evaluator was again crowdsourced and a new translation was solicited from AMT workers. If a worker's utterances were flagged erroneous often then we barred him/her from doing any more tasks.

As mentioned in the previous section, once we collected such verified translations for all the utterance templates, we instantiated them and created complete end-to-end dialogs containing code-mixed utterances. Once the entire dialog was constructed, we conducted a separate human evaluation wherein we asked 12 in-house evaluators (3 evaluators per language) to read 100 code-mixed dialogs (entire dialogs as opposed to just some utterances) from each language and rate them on three metrics namely colloquialism, intelligibility and coherence on a scale of 1 (very poor) to 5 (very good) as defined below.

- **Colloquialism:** To check if the code-mixing was colloquial throughout the dialog and not forced.
- **Intelligibility:** To check if the entire dialog could be easily understood by a bilingual speaker who could speak the native language as well as English.
- **Coherence:** To check if the entire dialog looked coherent even though it was constructed by stitching together utterances which were independently translated and code-mixed (*i.e.*, while translating an utterance annotators did not know what their preceding and following utterances were).

These 100 dialogs were chosen randomly from across the entire dataset for each language. The evaluators used for this were different from the in-house annotators used to create the original translations in order to reduce the bias in evaluations. The average ratings given by the evaluators for each of the languages are shown in Table 5 and are encouraging. The demographic details of the in-house workers and evaluators are shown in Table 4.

4.4 Dataset Statistics and Analysis

For every word in the code-mixed corpus, we were able to identify whether it was a word from the native language or English or language agnostic (named entities). It was easy to do this because we had the vocabulary of the original English DSTC2 corpus as well as named entities (so any word which was not in the original DSTC2 vocabulary or a named entity was a word from the native language). We also manually verified this list of words marked as native words and corrected discrepancies if any (*i.e.*, we ensured that all the words which were marked as native words were actually native words). Note that the

	Hindi	Bengali	Gujarati	Tamil
Vocabulary Size	1676	1372	1858	2185
Code-Mixed English Vocabulary	386	360	387	424
Native Language Vocabulary	739	477	912	1214
Others Vocabulary	551	535	559	547
Unique Utterances	6549	6274	6417	6666
Utterances with code-mixed words	5750	5703	5643	5632
Pure Native Language utterances	348	210	340	420
Pure English utterances	451	361	434	614
Average length of utterances	8.16	7.74	8.04	6.78
Average # of code-mixed utterances per dialog	12.11	14.28	11.80	12.96

Table 6: Statistics of the code-mixed dataset

cuisine names such as *Australian, Italian, etc.* have their own dedicated words in the native language. Table 6 summarizes various statistics about the dataset such as total vocabulary size, native language vocabulary size, *etc*. We refer to the original English dataset as En-DSTC2 and the Hindi, Bengali, Tamil and Gujarati code-mixed datasets created as a part of this work as Hi-DSTC2, Be-DSTC2, Ta-DSTC2 and Gu-DSTC2 respectively. Below, we make a few observations from Table 6.

The percentage of code-mixed English words in the vocabulary of Hi-DSTC2, Be-DSTC2, Gu-DSTC2 and Ta-DSTC2 are 23.03%, 26.24%, 20.83% and 19.40% respectively. From these English words, the most high frequency words across all the four versions of the dataset were *restaurant, food, town* and *serve*. Although these words have their own dedicated counterparts in all the other four languages, people colloquially use these code-mixed English words very often when talking about restaurants in their native language. The percentage of code-mixed utterances out of all the unique utterances in Hi-DSTC2, Be-DSTC2, Gu-DSTC2 and Ta-DSTC2 are 87.80%, 90.90%, 87.94% and 84.49% respectively (from Table 6). This shows that a significant portion of the dataset contains code-mixed utterances and very few utterances are in pure native languages or in pure English. This fact is also evident from the average number of code-mixed utterances per dialog in Table 6 compared to the average number of utterances per dialog in Table 3. We also calculated the number of utterances which contain k non-native words and then plotted a histogram (Figure 1) where the x-axis shows the number of k non-native words and the y-axis shows the number of utterances which had k non-native words. These histograms show a similar trend across all the languages. Apart from such intra-utterance code-mixing, we also noticed some intra-word code-mixing in mostly Bengali (*restauranter, towner*) and Tamil (*addressum, numberum, addressah*) versions of the dataset.

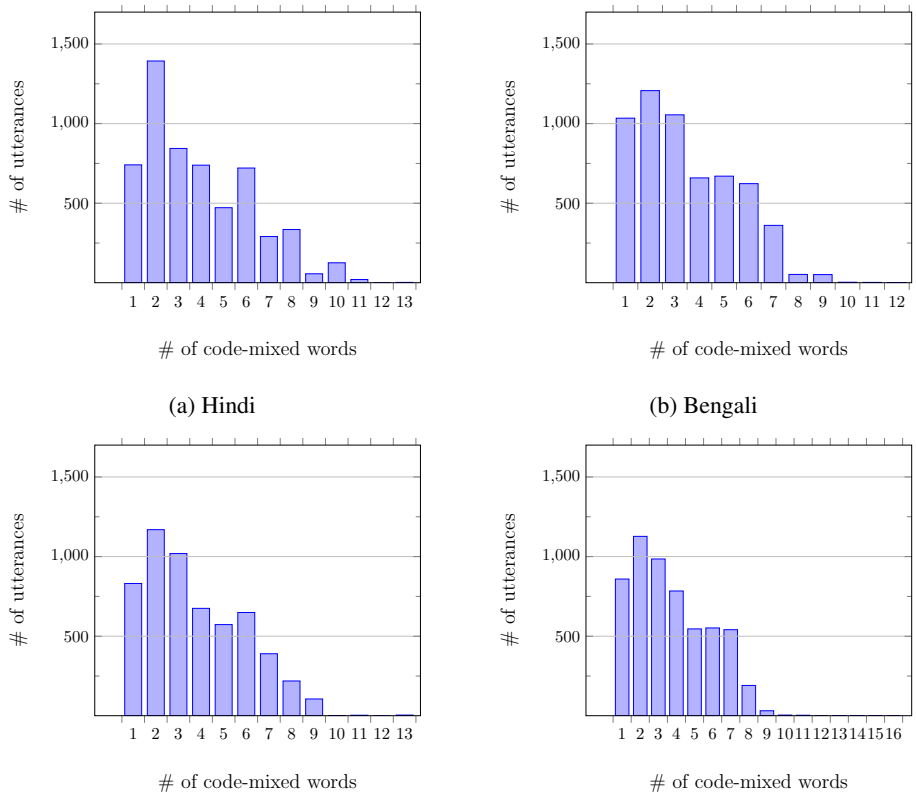
4.5 Quantitative Measures of Code-Mixing

Gambäck and Das (2016) introduced a measure to quantify the amount of code-mixing in a sentence as:

$$C_u(x) = \begin{cases} \frac{N(x) - \max\limits_{L_i \in \mathcal{L}} \{t_{L_i}\}}{N(x)} & : N(x) > 0 \\ 0 & : N(x) = 0 \end{cases} \quad (1)$$

Here, $\mathcal{L}$ is the set of all languages in the corpus, t_{L_i} is the number of tokens of language L_i in the given sentence x, $\max\limits_{L_i \in \mathcal{L}} \{t_{L_i}\}$ is the maximum number of tokens of a language L_i in the sentence x and $N(x)$ is the number of language-specific tokens in the sentence ($N(x)$ does not include named entities as they are language agnostic). The authors make a crucial assumption that $\arg\max\limits_{L_i \in \mathcal{L}} \{t_{L_i}\}$ is the *Matrix* language and hence the numerator of Equation 1 gives the number of foreign language tokens in x. This measure does not take into account the number of language switch points in a sentence (denoted by $P(x)$) and so the authors modify it further:

$$C_u(x) = 100 \cdot \frac{N(x) - \max\limits_{L_i \in \mathcal{L}} \{t_{L_i}\} + P(x)}{2N(x)} : (\textit{if}\, N(x) > 0) \quad (2)$$



(a) Hindi (b) Bengali

(c) Gujarati (d) Tamil

Figure 1: Histogram of the number of code mixed words in all the unique utterances for each language.

The code-mixing in the entire corpus can then be quantified by taking an average of the above measure across all sentences in the corpus:

$$C_{avg} = \frac{1}{U} \sum_{i=1}^{U} C_u(x_i) \tag{3}$$

where U is the number of sentences in the corpus. However, their main assumption that the language which has the maximum number of tokens in a sentence is the *Matrix* language, may not always hold. Consider a counter example: "Prezzo ek accha *restaurant* hain *in the north part of town* jo *tasty chinese food serve* karta hain." Here the word 'Prezzo' is a named entity and hence treated as a language independent token. The most frequent language (*italicized*) is English but the *Matrix* language is essentially Hindi. So we propose a small modification to their measure and replace $\max_{L_i \in \mathcal{L}}\{t_{L_i}\}$ by the following:

$$native(x) = \begin{cases} t_{L_n} & : t_{L_n} > 0 \\ N(x) & : t_{L_n} = 0 \end{cases} \tag{4}$$

where L_n is the native (*Matrix*) language of the utterance and t_{L_n} is the number of tokens of the native language in the utterance. Note that we know the native (*Matrix*) language of every utterance beforehand because of the manner in which the dataset was created. Gambäck and Das (2016) also pointed out that C_{avg} does not take the inter-utterance code-mixing and frequency of code-mixed utterances into account. To overcome this they proposed to use a term $\delta(x_i)$ which assigns a score of 1 if the *Matrix* language of x_i is different from that of x_{i-1} or a score of 0 if they are same or $i = 1$. Note that in our case $\delta(x_i)$ would mostly be 0 except for cases where x_i is a pure English utterance. The authors also used a term for the fraction of code-mixed utterances $(\frac{S}{U})$ in the corpus, where S is the total number of code-mixed utterances. We use a modified version of their final Code-Mixing index[2] by replacing the maximum

[2]We refer the reader to Gambäck and Das (2016) for the detailed derivation.

Language-pair	En-Be	En-Hi	En-Hi	En-Hi	En-Hi	En-Hi	En-Be	En-Gu	En-Ta
Dataset	TW (Jamatia)	TW (Jamatia)	FB (Jamatia)	FB+TW (Jamatia)	Vyas	Hi-DSTC2	Be-DSTC2	Gu-DSTC2	Ta-DSTC2
I-index	-	-	-	-	-	**0.04**	**0.04**	0.03	0.03
$\mathbf{C_{avg}}$	8.34	21.19	3.92	11.82	11.44	**32.12**	31.80	31.66	29.54
δ	22.09	30.99	6.70	17.81	**53.50**	26.38	29.06	24.50	38.32
$\mathbf{C_c}$	25.14	64.38	16.76	38.53	31.31	73.31	76.27	71.63	**80.49**

Table 7: Comparison of the quantitative measures of code-mixing in the dataset.

function by $native(x)$:

$$C_c = \frac{100}{U}\left[\frac{1}{2}\sum_{i=1}^{U}\left(1 - \frac{native(x) + P(x)}{N(x)} + \delta(x)\right) + \frac{5}{6}S\right] \quad (5)$$

Similarly, Guzmán et al. (2016) introduced the I-index measure to quantify the integration of different languages in a corpus. This metric is much simpler and simply computes the number of switch points in the corpus. For example, if a corpus contains n words and there are k positions at which the language of $word_i$ is not the same as the language of $word_j$ then the I-index is given by $\frac{k}{n-1}$. We compute the I-index for every utterance in a dialog, then compute the average over all utterances in a dialog and finally report the average across all dialogs in the code-mixed corpus. These measures of our dataset are shown in Table 7 and are compared with that of the existing datasets (Jamatia et al., 2016; Vyas et al., 2014). Jamatia et al. (2016) collected the code-mixed text from Twitter (TW) and Facebook (FB) posts whereas Vyas et al. (2014) collected their dataset only from Facebook forums. Although the dataset of Vyas et al. (2014) show the highest inter-utterance code-mixing (δ), Hi-DSTC2 and Ta-DSTC2 show the highest level of overall code-mixing at the utterance level (C_{avg}) and the corpus level (C_c) respectively.

5 Baseline Models

We establish some initial baseline results on this code-mixed dataset by evaluating two different generation based models: (i) sequence-to-sequence with attention (Bahdanau et al., 2015) and (ii) Hierarchical Recurrent Encoder-Decoder (HRED) model (Serban et al., 2016). Due to lack of space we don't describe these popular models here but refer the reader to the original papers. Apart from the above models, models which fetch the correct response from a set of candidate responses such as Query Reduction Networks (Seo et al., 2017), Memory Networks (Bordes and Weston, 2017) and Hybrid Code Networks (Williams et al., 2017) have also been evaluated on En-DSTC2. However, it is difficult to get candidate responses for every domain in practice and hence we stick to generation based models.

5.1 Experimental Setup

We use the train, validation and test splits of Bordes and Weston (2017) mentioned in Table 3. We create training instances from the dialogs by creating pairs of $\{$*context, response*$\}$ where *response* is every even numbered utterance and *context* contains all the previous utterances. Thus, if a dialog has 10 utterances, we create 5 training instances from it. Similarly at the test time the model is given the *context* and it has to generate the *response*. For both the models, we used Adam optimizer (Kingma and Ba, 2015) to train the network with a mini batch size of 32. We used dropouts (Srivastava et al., 2014) of 0.25 and 0.35, initial learning rate of 0.0004 and Gated Recurrent Units (GRU) (Cho et al., 2014) with hidden dimensions of size 350. We used word embeddings of size 300 with Glorot initialization (Glorot and Bengio, 2010). We also clipped the gradients at a maximum norm of 10 to avoid exploding gradients.

5.2 Evaluation

We evaluate the performance of the above models using BLEU-4 (Papineni et al., 2002), ROUGE-1, ROUGE-2 and ROUGE-L (Lin, 2004) which are widely used to evaluate the performance of Natural Language Generation systems. We also compute the per utterance accuracy (exact match) by comparing

	SEQ2SEQ WITH ATTENTION					HRED				
Metrics	English	Hindi	Bengali	Gujarati	Tamil	English	Hindi	Bengali	Gujarati	Tamil
BLEU-4	56.6	54.0	56.8	53.8	62.1	57.8	54.1	56.7	54.1	60.7
ROUGE-1	67.2	62.9	67.4	64.7	67.8	67.9	63.3	67.1	65.3	67.1
ROUGE-2	55.9	52.4	57.5	54.8	56.3	57.5	52.6	56.9	55.2	55.6
ROUGE-L	64.8	61.0	65.1	62.6	65.6	65.7	61.5	64.8	63.2	65.1
Per response acc.	46.0	48.0	50.4	47.6	49.3	48.8	47.2	47.7	47.9	47.8
Per dialog acc.	1.4	1.2	1.5	1.5	1.3	1.4	1.5	1.6	1.6	1.0

Table 8: Performance of the baseline models on all the languages

the generated response with the ground truth response. The generated response is considered to be accurate only if it exactly matches the ground truth response. This is obviously a more strict metric for generation based models (Eric and Manning, 2017). We also compute the per dialog accuracy by matching all the generated responses in a dialog with all the ground truth responses for that dialog. This metric measures whether the model was able to produce the entire dialog correctly end-to-end and hence complete the goal. We summarize the performance of the two models in Table 8. We observe that the performance of these models is very similar across all the languages. We observe that the models are still far from 100% accuracy and there is clearly scope for further improvement.

6 Conclusion

Code-mixing is an emerging trend of communication in the multilingual regions. The community has already addressed this phenomenon by introducing challenges on POS-Tagging, Language Identification, Language Modeling, *etc* on the code-mixed corpora. However, the approaches to development of dialog systems still rely on monolingual conversation datasets. To alleviate this problem we introduced a goal-oriented code-mixed dialog dataset for four languages (Hindi-English, Bengali-English, Gujarati-English and Tamil-English respectively). The dataset was created using a mix of in-house and crowdsourced workers. All the utterances in the dataset were evaluated by in-house evaluators and the overall dialogs were also evaluated for colloquialism, intelligibility and coherence. On all these measures, the dialogs in our dataset received a high score. To facilitate further research on these datasets, we provide the implementation of two popular neural dialog models *viz.* sequence-to-sequence and HRED. The evaluation of these models suggest that there is a clear scope for development of new architectures which can understand and converse in code-mixed languages.

Acknowledgements

We would like to thank Accenture Technology Labs, India for supporting this work through their generous academic research grant.

References

Heike Adel, Ngoc Thang Vu, Franziska Kraus, Tim Schlippe, Haizhou Li, and Tanja Schultz. 2013a. Recurrent neural network language modeling for code switching conversational speech. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pages 8411–8415.

Heike Adel, Ngoc Thang Vu, and Tanja Schultz. 2013b. Combination of recurrent neural networks and factored language models for code-switching language modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers*, pages 206–211.

Heike Adel, Ngoc Thang Vu, Katrin Kirchhoff, Dominic Telaar, and Tanja Schultz. 2015. Syntactic and semantic features for code-switching factored language models. *Trans. Audio, Speech and Lang. Proc.*, 23(3):431–440, March.

Fahad AlGhamdi, Giovanni Molina, Mona Diab, Thamar Solorio, Abdelati Hawwari, Victor Soto, and Julia Hirschberg. 2016. Part of speech tagging for code switched data. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 98–107. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*.

Rafael E. Banchs. 2012. Movie-dic: A movie dialogue corpus for research and development. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 203–207, Stroudsburg, PA, USA. Association for Computational Linguistics.

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. In *Proceedings of the First Workshop on Computational Approaches to Code Switching@EMNLP 2014, Doha, Qatar, October 25, 2014*, pages 13–23.

Utsab Barman, Joachim Wagner, and Jennifer Foster. 2016. Part-of-speech tagging of code-mixed social media content: Pipeline, stacking and joint modelling. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 30–39. Association for Computational Linguistics.

Antoine Bordes and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. *International Conference on Learning Representations*.

Kunal Chakma and Amitava Das. 2016. CMIR: A corpus for evaluation of code mixed information retrieval of hindi-english tweets. *Computación y Sistemas*, 20(3):425–434.

Tao Chen and Min-Yen Kan. 2013. Creating a live, public short message service corpus: the NUS SMS corpus. *Language Resources and Evaluation*, 47(2):299–335.

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

Mihail Eric and Christopher Manning. 2017. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 468–473. Association for Computational Linguistics.

Eric N. Forsythand and Craig H. Martell. 2007. Lexical and discourse analysis of online chat dialog. In *Proceedings of the First IEEE International Conference on Semantic Computing (ICSC 2007), September 17-19, 2007, Irvine, California, USA*, pages 19–26.

Björn Gambäck and Amitava Das. 2016. Comparing the level of code-switching in corpora. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.*

Souvick Ghosh, Satanu Ghosh, and Dipankar Das. 2016. Part-of-speech tagging of code-mixed social media text. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 90–97. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, pages 249–256.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech and Signal Processing - Volume 1*, ICASSP'92, pages 517–520, Washington, DC, USA. IEEE Computer Society.

Gualberto A. Guzmán, Jacqueline Serigos, Barbara E. Bullock, and Almeida Jacqueline Toribio. 2016. Simple tools for exploring variation in code-switching for linguists. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching@EMNLP 2016, Austin, Texas, USA, November 1, 2016*, pages 12–20.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014a. The second dialog state tracking challenge. In *Proceedings of the SIGDIAL 2014 Conference, The 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 18-20 June 2014, Philadelphia, PA, USA*, pages 263–272.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014b. The third dialog state tracking challenge. In *2014 IEEE Spoken Language Technology Workshop, SLT 2014, South Lake Tahoe, NV, USA, December 7-10, 2014*, pages 324–329.

Anupam Jamatia, Björn Gambäck, and Amitava Das. 2016. Collecting and annotating indian social media code-mixed corpora. In *Computational Linguistics and Intelligent Text Processing - 17th International Conference, CICLing 2016, Konya, Turkey, April 3-9, 2016, Revised Selected Papers, Part II*, pages 406–417.

Mitesh M. Khapra, Salil Joshi, Ananthakrishnan Ramanathan, and Karthik Visweswariah. 2013. Offering language based services on social media by identifying user's preferred language(s) from romanized text. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume*, pages 71–72.

Seokhwan Kim, Luis Fernando D'Haro, Rafael E. Banchs, Jason D. Williams, Matthew Henderson, and Koichiro Yoshino. 2016. The fifth dialog state tracking challenge. In *2016 IEEE Spoken Language Technology Workshop, SLT 2016, San Diego, CA, USA, December 13-16, 2016*, pages 511–517.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P. Spithourakis, Jianfeng Gao, and William B. Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.*

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the SIGDIAL 2015 Conference, The 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2-4 September 2015, Prague, Czech Republic*, pages 285–294.

Giovanni Molina, Fahad AlGhamdi, Mahmoud Ghoneim, Abdelati Hawwari, Nicolas Rey-Villamizar, Mona Diab, and Thamar Solorio. 2016. Overview for the second shared task on language identification in code-switched data. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 40–49. Association for Computational Linguistics.

Carol Myers-Scotton. 1993. *Duelling languages: Grammatical structure in codeswitching*. Oxford University Press.

Dong Nguyen and A. Seza Dogruöz. 2013. Word level language identification in online multilingual communication. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 857–862.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 311–318.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*, pages 172–180.

Mike Rosner and Paulseph-John Farrugia. 2007. A tagging algorithm for mixed language identification in a noisy domain. In *INTERSPEECH 2007, 8th Annual Conference of the International Speech Communication Association, Antwerp, Belgium, August 27-31, 2007*, pages 190–193.

Sophie Rosset and Sandra Petel. 2006. The ritel corpus - an annotated human-machine open-domain question answering spoken dialog corpus. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation, LREC 2006, Genoa, Italy, May 22-28, 2006.*, pages 1640–1643.

Anindya Roy, Camille Guinaudeau, Hervé Bredin, and Claude Barras. 2014. Tvd: A reproducible and multiply aligned tv series dataset. In *LREC*, pages 418–425.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Query-reduction networks for question answering. *International Conference on Learning Representations*.

Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. 2015. A survey of available corpora for building data-driven dialogue systems. *CoRR*, abs/1512.05742.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3295–3301.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1577–1586.

Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steven Bethard, Mona Diab, Mahmoud Ghoneim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirschberg, Alison Chang, and Pascale Fung. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 62–72, Doha, Qatar, October. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. In *ICML Deep Learning Workshop*.

Yogarshi Vyas, Spandana Gella, Jatin Sharma, Kalika Bali, and Monojit Choudhury. 2014. Pos tagging of english-hindi code-mixed social media content. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 974–979. Association for Computational Linguistics.

Marilyn A. Walker, Jean E. Fox Tree, Pranav Anand, Rob Abbott, and Joseph King. 2012. A corpus for research on deliberation and debate. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, pages 812–817.

Jason D. Williams and Steve J. Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

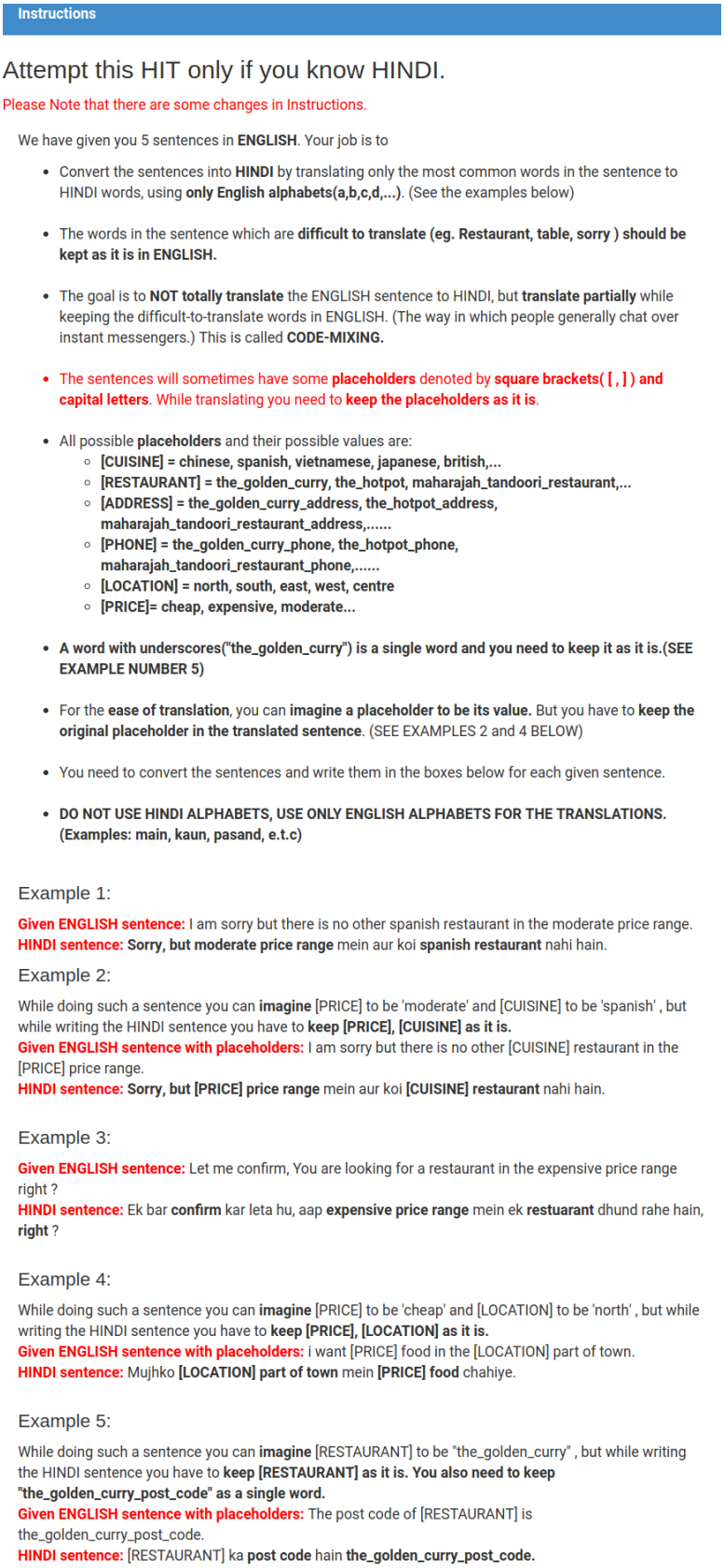
Jason D. Williams, Antoine Raux, Deepak Ramachandran, and Alan W. Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference, The 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 22-24 August 2013, SUPELEC, Metz, France*, pages 404–413.

Jason D. Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 665–677.

Kaisheng Yao, Geoffrey Zweig, and Baolin Peng. 2015. Attention with intention for a neural network conversation model. *CoRR*, abs/1510.08565.

Steve J. Young. 2000. Probabilistic methods in spoken-dialogue systems. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 358(1769):1389–1402.

Appendix A. Instructions to Crowdsourced Workers



Instructions

Attempt this HIT only if you know HINDI.

Please Note that there are some changes in Instructions.

We have given you 5 sentences in **ENGLISH**. Your job is to

- Convert the sentences into **HINDI** by translating only the most common words in the sentence to HINDI words, using **only English alphabets(a,b,c,d,...)**. (See the examples below)
- The words in the sentence which are **difficult to translate (eg. Restaurant, table, sorry) should be kept as it is in ENGLISH.**
- The goal is to **NOT totally translate** the ENGLISH sentence to HINDI, but **translate partially** while keeping the difficult-to-translate words in ENGLISH. (The way in which people generally chat over instant messengers.) This is called **CODE-MIXING.**
- The sentences will sometimes have some **placeholders** denoted by **square brackets([,]) and capital letters**. While translating you need to **keep the placeholders as it is**.
- All possible **placeholders** and their possible values are:
  - **[CUISINE] = chinese, spanish, vietnamese, japanese, british,...**
  - **[RESTAURANT] = the_golden_curry, the_hotpot, maharajah_tandoori_restaurant,...**
  - **[ADDRESS] = the_golden_curry_address, the_hotpot_address, maharajah_tandoori_restaurant_address,.....**
  - **[PHONE] = the_golden_curry_phone, the_hotpot_phone, maharajah_tandoori_restaurant_phone,.....**
  - **[LOCATION] = north, south, east, west, centre**
  - **[PRICE]= cheap, expensive, moderate...**
- **A word with underscores("the_golden_curry") is a single word and you need to keep it as it is.(SEE EXAMPLE NUMBER 5)**
- For the **ease of translation**, you can **imagine a placeholder to be its value.** But you have to **keep the original placeholder in the translated sentence**. (SEE EXAMPLES 2 and 4 BELOW)
- You need to convert the sentences and write them in the boxes below for each given sentence.
- **DO NOT USE HINDI ALPHABETS, USE ONLY ENGLISH ALPHABETS FOR THE TRANSLATIONS. (Examples: main, kaun, pasand, e.t.c)**

Example 1:

Given ENGLISH sentence: I am sorry but there is no other spanish restaurant in the moderate price range.
HINDI sentence: **Sorry, but moderate price range** mein aur koi **spanish restaurant** nahi hain.

Example 2:

While doing such a sentence you can **imagine** [PRICE] to be 'moderate' and [CUISINE] to be 'spanish' , but while writing the HINDI sentence you have to **keep [PRICE], [CUISINE] as it is.**
Given ENGLISH sentence with placeholders: I am sorry but there is no other [CUISINE] restaurant in the [PRICE] price range.
HINDI sentence: **Sorry, but [PRICE] price range** mein aur koi **[CUISINE] restaurant** nahi hain.

Example 3:

Given ENGLISH sentence: Let me confirm, You are looking for a restaurant in the expensive price range right ?
HINDI sentence: Ek bar **confirm** kar leta hu, aap **expensive price range** mein ek **restuarant** dhund rahe hain, **right** ?

Example 4:

While doing such a sentence you can **imagine** [PRICE] to be 'cheap' and [LOCATION] to be 'north' , but while writing the HINDI sentence you have to **keep [PRICE], [LOCATION] as it is.**
Given ENGLISH sentence with placeholders: i want [PRICE] food in the [LOCATION] part of town.
HINDI sentence: Mujhko **[LOCATION] part of town** mein **[PRICE] food** chahiye.

Example 5:

While doing such a sentence you can **imagine** [RESTAURANT] to be "the_golden_curry" , but while writing the HINDI sentence you have to **keep [RESTAURANT] as it is. You also need to keep "the_golden_curry_post_code" as a single word.**
Given ENGLISH sentence with placeholders: The post code of [RESTAURANT] is the_golden_curry_post_code.
HINDI sentence: [RESTAURANT] ka **post code** hain **the_golden_curry_post_code.**

Figure 2: Screenshot of the instructions and examples provided to the crowdsourced workers

Appendix B. Sample Code-Mixed Chats from the Dataset

User	Bot
1. <SILENCE>	Hello , Cambridge restaurant system mein aapka swagat hai. Aap pa sakte hai restaurant by area , price range ya food type. Aap ko kaise help kar sakta hu main ?
2. main north part of town mein cheap restaurant dhund raha hoon	Aap kaunse type ka food lena chahege?
3. mujhe parvah nahi hain.	api_call R_cuisine north cheap
da_vinci_pizzeria R_post_code da_vinci_pizzeria_post_code da_vinci_pizzeria R_cuisine italian da_vinci_pizzeria R_location north da_vinci_pizzeria R_phone da_vinci_pizzeria_phone da_vinci_pizzeria R_address da_vinci_pizzeria_address da_vinci_pizzeria R_price cheap da_vinci_pizzeria R_rating 10 royal_spice R_post_code royal_spice_post_code royal_spice R_cuisine indian royal_spice R_location north royal_spice R_phone royal_spice_phone royal_spice R_address royal_spice_address royal_spice R_price cheap royal_spice R_rating 8	
5. <SILENCE>	north of town me da_vinci_pizzeria ek achaa jagah hai aur prices cheap hai
6. aur kuch	north of town me royal_spice ek achaa jagah hai aur prices cheap hai
7. kyaa mujhe phone number mil sakta hai	royal_spice ka phone number hai royal_spice_phone
8. thank you goodbye	Welcome , alvida

Table 9: An example chat from the Hindi version of DSTC2 dataset

User	Bot
1. <SILENCE>	Hello, Cambridge restaurant system ey apnake swagoto janai. Apni restaurants khujte paren food type, price ar area diye. Apnake ki bhabe help korte pari ami ?
2. Ami ekta cheap restaurant khujchi town er north part e	Kon type er khabar apni pochondo korben ?
3. ami care kori na	api_call R_cuisine north cheap
da_vinci_pizzeria R_post_code da_vinci_pizzeria_post_code da_vinci_pizzeria R_cuisine italian da_vinci_pizzeria R_location north da_vinci_pizzeria R_phone da_vinci_pizzeria_phone da_vinci_pizzeria R_address da_vinci_pizzeria_address da_vinci_pizzeria R_price cheap da_vinci_pizzeria R_rating 10 royal_spice R_post_code royal_spice_post_code royal_spice R_cuisine indian royal_spice R_location north royal_spice R_phone royal_spice_phone royal_spice R_address royal_spice_address royal_spice R_price cheap royal_spice R_rating 8	
5. <SILENCE>	da_vinci_pizzeria ekta nice jayega town er north ey jekhane prices cheap
6. ar onno kichu ache	royal_spice ekta nice jayega town er north ey jekhane prices cheap
7. ami ki phone number ta pete pari	royal_spice er phone number holo royal_spice_phone
8. thank you goodbye	you are welcome

Table 10: An example chat from the Bengali version of DSTC2 dataset

Sequence-to-Sequence Learning for Task-oriented Dialogue with Dialogue State Representation

Haoyang Wen, Yijia Liu, Wanxiang Che*, Libo Qin, Ting Liu
Research Center for Social Computing and Information Retrieval,
Harbin Institute of Technology, China
{hywen, yjliu, car, lbqin, tliu}@ir.hit.edu.cn

Abstract

Classic pipeline models for task-oriented dialogue system require explicit modeling the dialogue states and hand-crafted action spaces to query a domain-specific knowledge base. Conversely, sequence-to-sequence models learn to map dialogue history to the response in current turn without explicit knowledge base querying. In this work, we propose a novel framework that leverages the advantages of classic pipeline and sequence-to-sequence models. Our framework models a dialogue state as a fixed-size distributed representation and use this representation to query a knowledge base via an attention mechanism. Experiment on Stanford Multi-turn Multi-domain Task-oriented Dialogue Dataset shows that our framework significantly outperforms other sequence-to-sequence based baseline models on both automatic and human evaluation.

Title and Abstract in Chinese

面向任务型对话中基于对话状态表示的序列到序列学习

面向任务型对话中，传统流水线模型要求对对话状态进行显式建模。这需要人工定义对领域相关的知识库进行检索的动作空间。相反地，序列到序列模型可以直接学习从对话历史到当前轮回复的一个映射，但其没有显式地进行知识库的检索。在本文中，我们提出了一个结合传统流水线与序列到序列二者优点的模型。我们的模型将对话历史建模为一组固定大小的分布式表示。基于这组表示，我们利用注意力机制对知识库进行检索。在斯坦福多轮多领域对话数据集上的实验证明，我们的模型在自动评价与人工评价上优于其他基于序列到序列的模型。

1 Introduction

Task-oriented dialogue system attracts more and more attention with the success of commercial systems such as Siri, Cortana and Echo. It helps users complete specific tasks with natural language. Figure 1 shows a typical example of a task-oriented dialogue, where an agent provides with location information for a user. The requirements for the agents to accomplish users' demands usually involve querying the knowledge base (KB), like acquiring address from location information KB in Figure 1.

Typical machine learning approaches model the problem as a partially observable Markov Decision Process (POMDP) (Williams and Young, 2007; Young et al., 2013), where a pipeline system is introduced. The pipeline system consists of four components: natural language understanding (NLU, Tur and De Mori, 2011) , dialogue state tracking (Williams et al., 2013; Williams, 2012), dialogue policy learning (Young et al., 2010) and natural language generation (Wen et al., 2015). Taking the utterance in Figure 1 for example, NLU maps the utterance "`Address to the gas station`" into semantic slot "`POI type`". Dialogue state tracker keeps the probability of "`gas station`" close to 1 against other values of slot "`POI type`". Given a semantic frame as a dialogue state, which is the combination of distributions of these slots, dialogue policy learning generates the next pre-defined system action,

* Email corresponding.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3781–3792
Santa Fe, New Mexico, USA, August 20-26, 2018.

Address	Distance	POI type	POI	Traffic info
638 Amherst St	3 miles	grocery store	Sigona Farmers Market	car collision nearby
269 Alger Dr	1 miles	coffee or tea place	Cafe Venetia	car collision nearby
5672 barringer street	5 miles	certain address	5672 barringer street	no traffic
200 Alester Ave	2 miles	gas station	Valero	road block nearby
899 Ames Ct	5 miles	hospital	Stanford Childrens Health	moderate traffic
481 Amaranta Ave	1 miles	parking garage	Palo Alto Garage R	moderate traffic
145 Amherst St	1 miles	coffee or tea place	Teavana	road block nearby
409 Bollard St	5 miles	grocery store	Willows Market	no traffic

Driver: Address to the gas station.
Car: Valero is located at 200 Alester Ave.
Driver: OK , please give me directions via a route that avoids all heavy traffic.
Car: Since there is a road block nearby, I found another route for you and I sent it on your screen.
Driver: Awesome thank you.

Figure 1: An example of a task-oriented dialogue that incorporates a knowledge base. The knowledge base will be changed based on different dialogue environment setting. Agents need to generate response based on current knowledge base.

which usually involves querying the knowledge base. The action is then converted to its natural language expression using natural language generation. Both natural language understanding and dialogue state tracking require a large amount of domain specific annotation for training, which is expensive to obtain. Besides, the design of actions and the explicit forms of semantic frames require a lot of knowledge from human experts, which are domain-specific as well.

Neural generative models, typically Seq2Seq models, have achieved success on machine translation (Sutskever et al., 2014; Bahdanau et al., 2014; Luong et al., 2015). This success spurs the interests to apply Seq2Seq models into dialogue systems. Seq2Seq models map dialogue history directly into the response in current turn, while requires a minimum amount of hand-crafting. However, conventional Seq2Seq doesn't model the exterior data retrieval explicitly, which makes it hard for Seq2Seq to generate information stored in KB like meeting time and address, but this kind of retrieval is easy to achieve for classic pipeline. To tackle with the problem, Eric and Manning (2017) use an additional copy mechanism to retrieve entities that occurs in both KB and dialogue history. Eric et al. (2017) further introduced retrieval from key-value KB where the model uses key representations to retrieve the corresponding values. However, not all KBs are presented in key-value forms. Besides, an important component of classic pipeline, dialogue state tracker, is not properly modeled, making it difficult to precisely retrieve from KB.

In this paper, we propose a novel framework that takes the advantages from both classic pipeline models and Seq2Seq models. We introduce dialogue states into Seq2Seq learning, but in a implicit way. Distributions in classic state tracking are modeled as a group of representation vectors computed by an attention-based network (Britz et al., 2017), which can be considered as a dialogue state representation that aggregates information for each slot. And training this representation doesn't require annotation of dialogue state tracking. Our model queries the KB entries in an attention-based method as well, so that the querying is differentiable, without domain-specific pre-defined action spaces. Meanwhile we compute the representation for KB using entry-level attention and aggregate the representation with dialogue state representation to form a memory matrix of dialogue history and KB information. While decoding, we perform an attention over memory and an attention over input, incorporating copying mechanism (Gu et al., 2016) that allows model to copy words from KBs to enhance the capability of retrieving accurate entities.

We evaluate the proposed framework on Stanford Multi-turn, Multi-domain Dialogue Dataset (Eric et al., 2017), to test the effectiveness of our framework and flexibility to apply to different domains. We compare our model with other Seq2Seq models and discovered that our model has outperformed other

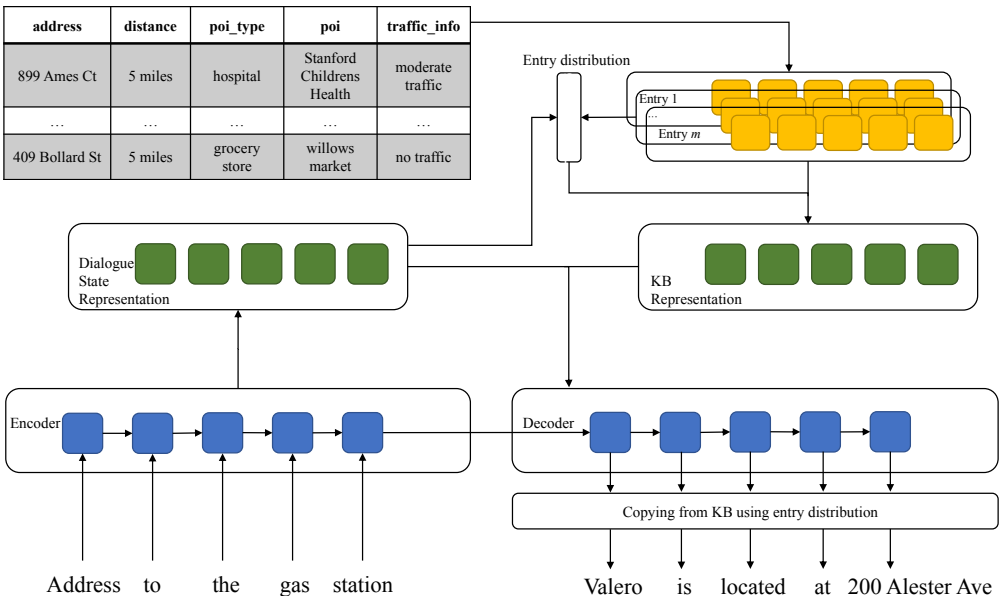


Figure 2: Proposed framework.

models on both automatic evaluation and human evaluation.

2 Proposed Framework

In this section, we describe a framework for task-oriented dialogue system. Our framework first encodes previous dialogue history, and computes dialogue state representation. Then our framework queries the table by attention and computes a matrix to represent information from previous history and KB. At last, the responses are generated using copying mechanism. The general architecture is demonstrated in Figure 2.

2.1 Encoder

Given a dialogue consisting of utterances from a user and an agent, our encoder encodes the whole dialogue history. We represent the dialogue history as a sequence of utterances. We encode the previous dialogue history as one single sequence consisting of each word in previous dialogue history and use $(x_1, x_2, \ldots, x_{n^{\text{IN}}})$ to denote the whole dialogue history word by word, where n^{IN} is the length of this sequence. Words are mapped to word embeddings and a long short-term memory network (LSTM) aggregates hidden representation over the sentence, denoted as $\boldsymbol{H}^{\text{ENC}} = \left[\boldsymbol{h}_1^{\text{ENC}}, \ldots, \boldsymbol{h}_{n^{\text{IN}}}^{\text{ENC}}\right]$ ($\boldsymbol{H}^{\text{ENC}} \in \mathbb{R}^{d \times n^{\text{IN}}}$, d is the dimensionality of a hidden state).

2.2 Dialogue State Representation

The dialogue state tracking component of a dialogue system interprets the previous dialogue history to a belief state (Williams et al., 2013), which consists of a group of probability distributions over values for each slot. The dialogue state tracking is the core component of pipeline model. It helps with retrieving values from KB and generating accurate entities. To introduce dialogue state into Seq2Seq learning, in this framework, we model representation of belief state, motivated by Britz et al. (2017). We do not compute the explicit probability distribution for each slot. Instead, a group of distributed representations is computed. In this paper, we assume each turn of the dialogue is associated with m slots and its dialouge state representation is a matrix $\boldsymbol{U}^{\text{IN}} = \left[\boldsymbol{u}_1^{\text{IN}}, \ldots, \boldsymbol{u}_m^{\text{IN}}\right] \in \mathbb{R}^{d \times m}$ whose columns represent corresponding distribution. We further assume that m equals to the number of columns in our KB. Each state representation $\boldsymbol{u}_k^{\text{IN}}$ is calculated by an attention over the whole representation of history:

$$\boldsymbol{u}_k^{\text{IN}} = \sum_{t=1}^{n^{\text{IN}}} a_k^{\text{IN}}(t)\, \boldsymbol{h}_t^{\text{ENC}},$$

where we assign each hidden state m different kinds of scores and perform a weighted average. The scores are computed by the following equations:

$$\begin{aligned} \textsc{ScoreIn}\left(\boldsymbol{w}_k^{\mathrm{A}}, \boldsymbol{h}_t^{\mathrm{ENC}}\right) &= \boldsymbol{w}_k^{A^T} \boldsymbol{h}_t^{\mathrm{ENC}}, \\ a_k^{\mathrm{IN}}(t) &= \frac{\exp\left(\textsc{ScoreIn}\left(\boldsymbol{w}_k^{\mathrm{A}}, \boldsymbol{h}_t^{\mathrm{ENC}}\right)\right)}{\sum_{t'} \exp\left(\textsc{ScoreIn}\left(\boldsymbol{w}_k^{\mathrm{A}}, \boldsymbol{h}_{t'}^{\mathrm{ENC}}\right)\right)}, \end{aligned}$$

$\boldsymbol{W}^A = [\boldsymbol{w}_1^A, \ldots, \boldsymbol{w}_m^A]$ is a parameter matrix in $\mathbb{R}^{d \times m}$.

2.3 Soft KB Attention

Table Encoder. In our framework, we compile the process of querying a KB entry into an attention network. The first step is to encode the tables. Each KB cell is represented as the concatenation of the column name embedding $\phi^{EMB}(f)$ and the cell value embedding $\phi^{EMB}(c)$. This representation is further fed into a $\tanh$ non-linearity and the final representation can be formalized as $\boldsymbol{c} = \tanh\left(\boldsymbol{W}^{\mathrm{C}}\left[\phi^{EMB}(c), \phi^{EMB}(f)\right]\right)$. The representation of the KB entry $\boldsymbol{C}_k$ is denote as $\boldsymbol{C}_k = [\boldsymbol{c}_{k,1}, ..., \boldsymbol{c}_{k,m}]$, consisting all its cell representations.

Entry and KB Representation. Conventionally, task-oriented dialogue systems interact with KB via carefully hand-crafted API calls, which are usually domain-specific and break the differentiability (Wen et al., 2017b). To make it differentiable, our framework applies a soft-attention over the KB entries and the attention value can be interpreted as the possibility that an entry will be used for decoding. We use the similarity between $\boldsymbol{C}_k$ and $\boldsymbol{U}^{\mathrm{IN}}$ to represent attention score for the k^{th} entry. The similarity is computed by the following equation:

$$\mathrm{sim}\left(\boldsymbol{C}_k, U^{\mathrm{IN}}\right) = \sum_{t=1}^{m} \boldsymbol{c}_{k,t} \cdot \boldsymbol{u}_t^{\mathrm{IN}},$$

where we sum the dot product between vectors in $\boldsymbol{C}_k$ and $\boldsymbol{U}^{\mathrm{IN}}$ respectively.

This similarity distribution is then converted into a probability distribution naturally using a softmax function. This probability distribution indicates the possibility to use entry e_k in the further response generation given previous dialogue history $\boldsymbol{x}_{<i}$:

$$p(e = e_k \mid \boldsymbol{x}_{<i}) = \frac{\exp\left(\mathrm{sim}\left(\boldsymbol{C}_k, U^{\mathrm{IN}}\right)\right)}{\sum_{k'} \exp\left(\mathrm{sim}\left(\boldsymbol{C}_{k'}, U^{\mathrm{IN}}\right)\right)}.$$

The information matrix from KB that is used for future generation can be computed as a weighted summation:

$$\boldsymbol{U}^{KB} = \sum_{t=1}^{|T|} p(e = e_t \mid \boldsymbol{x}_{<i}) \boldsymbol{C}_t,$$

where $\boldsymbol{U}^{\mathrm{KB}}$ is denoted as the information matrix, $|T|$ is the number of entries in this KB.

Since the dimensionalities of all parameters are not related to the size of knowledge base. it allows changing the KB on-the-fly. Besides, there is no need to define specific operations, which is required for using API calls to extract information. Since we model the entries directly, it is appropriate to extract information from non-entity-centric knowledge bases as well.

Finally, we combine these two kinds of information by concatenating corresponding vectors and feeding them into a linear layer with an activation function. Formally, we denote $\boldsymbol{U}^{\mathrm{CAT}}$ as the concatenation of $\boldsymbol{U}^{\mathrm{IN}}$ and $\boldsymbol{U}^{\mathrm{KB}}$. The two matrices are concatenated by concatenated corresponding vectors respectively. The process can be formulated as: $\boldsymbol{U} = \tanh\left(\boldsymbol{W}^{CAT}\boldsymbol{U}^{CAT}\right)$. The matrix $\boldsymbol{U}$ can be considered as a fix-sized memory representation over dialogue history and knowledge base.

2.4 Decoder

In this section, we will discuss the decoder that takes all previously calculated information to generate sentences. The vanilla Seq2Seq decoder generates a sequence of words recurrently based on the last hidden state of a encoder. We denote $\left(\boldsymbol{h}_1^{\text{DEC}}, \ldots, \boldsymbol{h}_{n^{\text{OUT}}}^{\text{DEC}}\right)$ as the hidden states of the decoder and $(y_1, \ldots, y_{n^{\text{OUT}}})$ as the output sentence. We will consider two kinds of information, that are information of dialogue history and information from KB. The model aggregates information via attention over KB representation and history representation.

Input Attention. The conventional attention mechanism is introduced to extend the decoder, where each hidden state in the encoder is assigned a score based on the current hidden state $\boldsymbol{h}_t^{\text{OUT}}$ at time step t, and then the context vector is computed by the weight summation (Luong et al., 2015). This process can be described by the following equations:

$$\begin{aligned}
\boldsymbol{c}_t^{\text{IN}} &= \sum_{i=1}^{n^{\text{IN}}} a_t^{\text{OUT}}(i)\, \boldsymbol{h}_i^{\text{ENC}}, \\
\textsc{ScoreOut}\left(\boldsymbol{h}_i^{\text{ENC}}, \boldsymbol{h}_t^{\text{DEC}}\right) &= \boldsymbol{v}^{\text{OUT}} \tanh\left(\boldsymbol{W}^{\text{OUT}}\left[\boldsymbol{h}_i^{\text{ENC}}, \boldsymbol{h}_t^{\text{DEC}}\right]\right). \\
a_t^{\text{OUT}}(i) &= \frac{\exp\left(\textsc{ScoreOut}\left(\boldsymbol{h}_i^{ENC}, \boldsymbol{h}_t^{\text{DEC}}\right)\right)}{\sum_{i'} \exp\left(\textsc{ScoreOut}\left(\boldsymbol{h}_{i'}^{\text{ENC}}, \boldsymbol{h}_t^{\text{DEC}}\right)\right)}.
\end{aligned}$$

Memory Attention. Besides the context vector through input attention, we also use another context vector from the attention over the fix-size memory matrix $\boldsymbol{U}$ and it's computed as:

$$\begin{aligned}
\boldsymbol{c}_t^{\text{MEM}} &= \sum_{i=1}^{m} a_t^{\text{MEM}}(i)\, \boldsymbol{u}_i, \\
\textsc{ScoreMem}\left(\boldsymbol{u}_i, \boldsymbol{h}_t^{\text{DEC}}\right) &= \boldsymbol{v}^{\text{MEM}} \tanh\left(\boldsymbol{W}^{\text{MEM}}\left[\boldsymbol{u}_i, \boldsymbol{h}_t^{\text{DEC}}\right]\right), \\
a_t^{\text{MEM}}(i) &= \frac{\exp\left(\textsc{ScoreMem}\left(\boldsymbol{u}_i, \boldsymbol{h}_t^{\text{DEC}}\right)\right)}{\sum_{i'} \exp\left(\textsc{ScoreMem}\left(\boldsymbol{u}_{i'}, \boldsymbol{h}_t^{\text{DEC}}\right)\right)}.
\end{aligned}$$

In practice, we use an additional context vector from encoder to calculate the dialogue state representation, which is different from the one that is used for the start of decoding.

The two kinds of context vectors and the current hidden state are used for decoding. We introduce a variant of copying mechanism (Gu et al., 2016) in order to retrieve entities from KB directly. We first compute a probability distribution over output vocabulary $\mathcal{V}$ and slot types $\mathcal{V}^{\text{SLOT}}$, given previous dialogue history $\boldsymbol{x}_{<i}$ and previous generated words $\boldsymbol{y}_{<t}$, which can be described as: $p\left(\widetilde{y}_t \mid \boldsymbol{x}_{<i}, \boldsymbol{y}_{<t}\right) = \text{softmax}\left(W^{\text{O}}\left[\boldsymbol{h}^{\text{DEC}}, \boldsymbol{c}^{\text{IN}}, \boldsymbol{c}^{\text{MEM}}\right]\right)$. The probability of generating a slot type represents the sum of probability of generating a slot value for this slot from KB. Since we have calculated the probability of using an entry in section 2.3, the probability of generating a word can be described as:

$$\begin{aligned}
p\left(y_t = y \mid \widetilde{y}_t, \boldsymbol{x}_{<i}, \boldsymbol{y}_{<t}\right) &= p\left(\widetilde{y}_t = y \mid \boldsymbol{x}_{<i}, \boldsymbol{y}_{<t}\right) \\
&\quad + \sum_{y^{\text{S}} \in \mathcal{V}^{\text{SLOT}}} p\left(\widetilde{y}_t = y^{\text{S}} \mid \boldsymbol{x}_{<i}, \boldsymbol{y}_{<t}\right) \sum_{k=1}^{|T|} \mathbb{1}\left\{e_k\left(y^{\text{S}}\right) = y\right\} p(e = e_k \mid \boldsymbol{x}_{<i}),
\end{aligned}$$

where $e_k\left(y^{\text{S}}\right)$ is the cell that is in slot (i.e. column) y^{S}. Note that the summation of probability over $\mathcal{V}$ is exactly 1 after the model copies entities from KB.

2.5 Training

Conventionally, we use negative log likelihood (NLL) for training to train a Seq2Seq model. Since there is no supervision for soft KB attention, and it is easy to get over-fitting when we only use NLL, we

	Weather	Navigation
Slot Types	location, date, highest temperature, lowest temperature and weather attribute	POI name, traffic info, POI type, address, distance

Table 1: Slot types for different domains.

apply policy gradient to improve the performance of soft KB attention as well. We consider the KB and fix-size memory representation from input as the environment in a reinforcement learning setting. There is only one action, which is defined as choosing a single entry from KB to help with generating response. Heuristically, the more entities in an entry appear in dialogue context, the higher possibility that this entry is used for generating response. Therefore, we consider the number of entities from an entry e that appear in previous dialogue history or current gold response as reward $R(e)$, and apply REINFORCE with baseline (Williams, 1992): $J_{RL} = -\mathbb{E}_{p(e|\boldsymbol{x}_{<i})}[R(e) - b]$, where b is a hyperparameter denoting the baseline reward. The joint loss is the combination of the NLL and loss from reinforcement learning, which is:

$$J = J_{NLL} + \lambda J_{RL}$$

where λ is a hyperpramater balancing the two factors. In practice, we first use J_{RL} only to train the soft KB attention and its encoder, without training for response generation, which will accelerate the convergence of Seq2Seq learning. We apply data augmentation to the original dataset as well, where we add delexicalised form responses into training data in order to force our model to generate slot types first and then retrieve entity from KB using copying mechanism. The delexicalised responses are generated by simple matching and replacing.

3 Experiment

In this section, we first introduce the details of experiment setting. Then we provide results and analyses of automatic evaluation and human evaluation in order to compare with other baseline models. Besides, we present ablation test to evaluate and analyze the function of different components in our framework. Finally, we provide visualization of dialogue state representation and case study.

3.1 Experiment Setting

We choose two KB-rich domains from Stanford Multi-turn Multi-domain Task-oriented Dialogue Dataset Eric et al. (2017), which are weather information retrieval and point-of-interest (POI) navigation (navigation). We first change the form of KB in weather information retrieval domain (weather). Eric et al. (2017) integrates the highest temperature, lowest temperature and weather information into a single weekday column due to their incapability of utilizing non-entity-centric KB. In this paper, we separate these information into three different column, and the slots of these two domains are provided in Table 3.

Our framework is trained separately in these two domains, using the same train/validation/test split sets as Eric et al. (2017). We do not map the entities in dialogue into its canonical form as what Eric et al. (2017) have done, since our framework extract entities directly from KB. And we evaluate our framework on exact entities as well.

Our framework is trained using the Adam optimizer (Kingma and Ba, 2014). The learning rate is 10^{-3}, λ for balancing two loss functions is 10^{-1}. We applied dropout (Srivastava et al., 2014) to the input and the output of LSTM, with a dropout rate at 0.75. We add the weight decay on the model. The coefficient of weight decay is 5×10^{-6}. The embedding size and all hidden size are 200. The number of epochs for training soft KB attention is 30 for both navigation and weather. Baseline b is 1.5 for both navigation and weather.

3.1.1 Baseline Models

We compare our model with two additional baselines beyond the Seq2Seq with attention, which includes:

- **Copy-augmented Sequence-to-Sequence Network.** This model is adapted from (Eric and Manning, 2017). It utilizes entities that appear in both previous dialogue history and KB. A hard copy mechanism for these entities is applied in this model.

- **Key-value Retrieval Network.** This model is adapted from Eric et al. (2017). It utilizes key-value forms to represent KBs. Key representations are used for an attention-based value retrieval. In weather information retrieval domain, although we have changed the KB into an non-entity-centric form, we still designate "`location`" slot as subject slot and we allow a key representation to retrieval multiple values. We convert inputs into canonical forms, while the outputs remain the same in order to compare with our model.

3.2 Automatic Evaluation

In this section, we provide two different automatic evaluations to compare with other baseline models. The results and analyses are provide in the following sections.

3.2.1 Evaluation Metrics

Entity F1 and BLEU score are used to evaluate our model. The entity F1 scores in both micro-average and macro-average manners are used to measure the difference between entities in the system and gold responses. Besides, we use BLEU to evaluate the quality of responses. Sharma et al. (2017) showed that BLEU has a comparatively strong correlation with human evaluation on task-oriented dialogue dataset. The BLEU score is computed from results with highest micro F1. To evaluate macro F1, we delete instances that neither gold nor generated response contains an entity.

3.2.2 Results and Analyses

Experiment results are illustrated in Table 2. The results show that our model outperforms other models in most of automatic evaluation metrics. In the navigation domain, compared to KV Net, we achieve 5.0 improvement on BLEU score, 37.1 improvement on Macro F1 and 27.4 improvement on Micro F1. Compared to Copy Net, we achieve 5.0 improvement on BLEU score, 41.2 improvement Macro F1 and 27.4 improvement on Micro F1. The results in navigation show our model's capability to generate more natural and accurate response than the Seq2Seq baseline models. In the weather domain, our model generates more accurate responses than our baseline models as well. The BLEU score is a little bit lower than Copy Net and Seq2Seq with attention. This is because the forms of responses are relatively limited in weather domain. Besides, the entities in inputs are highly probable to be mentioned in responses, such as "`location`". These two reasons indicate that the simpler models can capture this pattern more smoothly. The results that Seq2Seq with Attention performs better than Copy Net and KV Net also confirm this.

We also find that the KV Net's results are lower than that reported by Eric et al. (2017). We address this to the differences in the preprocessing, model training and evaluation metrics. In spite of the difference of evaluation metrics that we evaluate on exact entities rather than their canonical forms, the Micro F1 score of our model still outperforms what Eric et al. (2017) reported, which is 41.3 in navigation domain and which is evaluated on canonical forms. Our changes of the weather domain into non-entity-centric also influence its performance. This differences in results also indicate the robustness of our model when facing non-entity-centric KBs.

3.2.3 Ablation

In this section, we perform several ablation experiments to evaluate different components in our framework on the navigation domain. Results are shown in Table 3. The results demonstrate effectiveness of components of our model to the final performance.

Copying mechanism enables our framework to retrieve entities directly from KBs. Without copying mechanism, such retrieval is infeasible and our framework cannot produce values in KBs. The results show that it introduces more variability to the generation process if we do not use copying mechanism.

The reinforcement learning loss helps our framework to use correct KB entries so that improve the performance of generation. Without this reinforcement learning loss, the item selection process is only

Model	Navigation			Weather		
	BLEU	Macro F1	Micro F1	BLEU	Macro F1	Micro F1
Seq2Seq with Attention	8.3	15.6	17.5	**19.6**	56.0	53.5
Copy Net	8.7	20.8	23.7	17.5	52.4	53.1
KV Net	8.7	24.9	29.5	12.4	37.7	39.4
our model	**13.7**	**62.0**	**56.9**	14.9	**58.5**	**56.3**

Table 2: Automatic evaluation on test data. Best results are shown in bold. Generally, our framework outperforms other models in most automatic evaluation metrics.

Model	BLEU	Macro F1	Micro F1
our model	**13.7**	**62.0**	**56.9**
-copying	9.6	35.2	41.3
-RL	9.3	38.2	46.0

Table 3: Ablation experiment on navigation domain. -copy refers to a framework without copying. -RL refers to a framework without RL loss.

Model	Correct	Fluent	Humanlike
Copy Net	3.52	4.47	4.17
KV Net	3.61	4.50	4.20
our model	**4.21**	**4.65**	**4.38**
agreement	41.0	55.0	43.0

Table 4: Human evaluation of responses based on random selected previous dialogue history in test dataset. The agreement scores indicate the percentage of responses to which all three human experts give exactly the same scores.

supervised by log likelihood loss. We address the drop in performance to that our model overfits to the training data.

3.3 Human Evaluation

In this section, we provide human evaluation on our framework and other baseline models. We randomly generated 200 responses. These response are based on distinct dialogue history in navigation test data. We hire three different human experts to evaluate the quality of responses. Three dimensions are involved, which are correctness, fluency, and humanlikeness. Three human experts judged each dimension on a scale from 1 to 5. And each judgment indicates a relative score compared to standard response from test data. The results are illustrated in Table 4. The results show that our framework outperforms other baseline models on all metrics. The most significant improvement is from correctness, indicating that our model generates more accurate information that the users want to know.

3.4 Visualization and Case Study

We provide a visualization example to demonstrate the effectiveness of dialogue state representation. The visualization illustrates attention scores over the sentence for each slot. The blue-level of a cell indicates the attention score it represents. From this visualization, we can discover that our dialogue state representation matches slots with correct entities in sentence. For example, "`pizza_restaurant`" matches "`poi`" and "`poi_type`" correctly, "`4_miles`" matches "`address`" correctly. The visualization indicates the capability of our framework to track accurate information and integrate them into a fix-size matrix representation.

Finally, we provide a case study that consists of two conversations which are generated between our framework and a human and between Seq2Seq with Attention with human respectively. In this case, we find that our framework is able to generate correct information such as address and point-of-interest.

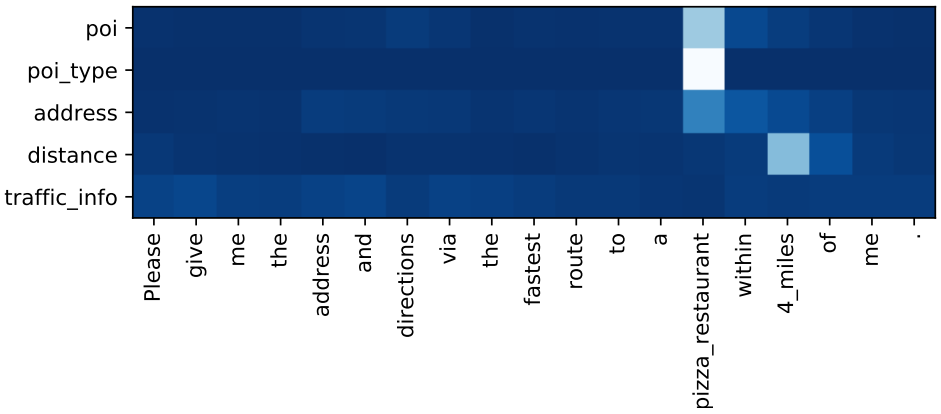


Figure 3: Visualization of dialogue state representation.

Address	Distance	POI type	POI	Traffic info
783 Arcadia Pl	5 miles	gas station	Chevron	moderate traffic
383 University Ave	5 miles	shopping center	Town and Country	no traffic
864 Almanor Ln	5 miles	friends house	jacks house	no traffic
5671 barringer street	6 miles	home	home	heavy traffic
657 Ames Ave	4 miles	rest stop	The Clement Hotel	no traffic
638 Amherst St	1 miles	grocery store	Sigona Farmers Market	heavy traffic
830 Almanor Ln	6 miles	chinese restaurant	tai pan	no traffic

our model

Driver:	what gas stations are here ?
Car:	there is a chevron 5 miles away .
Driver:	that 's good ! please show me the address and pick the quickest route to get there !
Car:	the address is 783 arcadia pl , i sent on your screen the best route to reach there .
Driver:	thank you !
Car:	you 're welcome !

Seq2Seq with Attention

Driver:	what gas stations are here ?
Car:	there is a chevron 4 miles away .
Driver:	that 's good ! please show me the address and pick the quickest route to get there !
Car:	the address to safeway is 452 arcadia pl .
Driver:	thank you !
Car:	you 're welcome !

Figure 4: Case study. The dialogue in the left is generated from our framework, and that in the right is generated from Seq2Seq with Attention baseline.

Conversely, Seq2Seq with Attention is generated in a more random way. The comparison between these two dialogues illustrates the capability of our model to retrieve accurate entities and while in the same time generate natural response.

4 Related Work

The recent successes on neural networks spread across many natural language processing tasks, which also stimulate research on task-oriented dialogue system. The powerful distributed representation ability of neural networks makes task-oriented dialogue system end-to-end possible. Recently, Wen et al. (2017b) built a system that connects classic pipeline modules by a policy network training with a user simulator. Wen et al. (2017a) further introduced latent intention into end-to-end learning. However, their modules like belief tracker still needs to be trained separately before end-to-end training. In contrast to their work, our framework trained the state tracker jointly with the end-to-end dialogue training. Liu and

Lane (2017) built a turn-level LSTM to model the dialogue state and generate probability distribution for each slot. Bordes and Weston (2017) built a system by applying memory network to store the previous dialogue history. But the responses are retrieved from templates, which is significantly different from our neural generative responses. Another type of work tried to build an end-to-end system as a task completion dialogue system (Li et al., 2016; Li et al., 2017; Peng et al., 2017). These modeled are trained through an end-to-end fashion via reinforcement learning algorithm using the reward from user simulators. However, their dialogue responses are not generated from the dialogue history directly but from a set of pre-defined action and explicit semantic frames.

Our soft KB attention can be considered as a process to retrieve entries, which has been explored in many QA and dialogue work. One line of this research includes creating well-defined API calls to query the KB (Williams et al., 2017; Wen et al., 2017a). And another line of research tried to directly retrieve entities from knowledge base. Yin et al. (2016b) has built a system to encode all table cells and assign a score vector to each row. Our framework resembles the second line of research, but can generate multiple entities to form natural language responses. He et al. (2017) has built two symmetric dialogue agents with private knowledge, and has applied knowledge graph reasoning into Seq2Seq learning, which is distantly related with our framework. In the sense of the KB forms, Yin et al. (2016a) retrieved entities based on (*subject, relation, object*) triples. While Dhingra et al. (2017) applied a soft-KB lookup on an entity-centric knowledge base to compute the probability of that the user knows the values of slots, and has tried to model the posterior distributions over all slots. However, our framework doesn't require entity-centric knowledge base.

5 Conclusion

In this paper, we proposed a framework that leverages dialogue state representation, which is tracked by an attention-based methods. Our framework performed an entry-level soft lookup over the knowledge base, and applied copying mechanism to retrieve entities from knowledge base while decoding. This framework was trained in an end-to-end fashion with only the dialogue history, and get rid of other annotation. Experiments showed that our model outperformed other Seq2Seq models on both automatic and human evaluation. The visualization and case study demonstrated the effectiveness of dialogue state representation and entity retrieval.

6 Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by the National Key Basic Research Program of China via grant 2014CB340503 and the National Natural Science Foundation of China (NSFC) via grant 61632011 and 61772153.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Antoine Bordes and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *Proceedings of International Conference on Learning Representations*.

Denny Britz, Melody Guan, and Minh-Thang Luong. 2017. Efficient attention using a fixed-size memory representation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 392–400.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 484–495.

Mihail Eric and Christopher Manning. 2017. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: (Volume 2, Short Papers)*, volume 2, pages 468–473.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the 18th Annual SIGDial Meeting on Discourse and Dialogue*, pages 37–49.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1631–1640.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1766–1776.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017. End-to-end task-completion neural dialogue systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 733–743.

Bing Liu and Ian Lane. 2017. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. In *Interspeech 2017*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September. Association for Computational Linguistics.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2231–2240.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *arXiv preprint arXiv:1706.09799*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

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

Gokhan Tur and Renato De Mori. 2011. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

Tsung-Hsien Wen, Milica Gašic, Dongho Kim, Nikola Mrkšic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 275.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017a. Latent intention dialogue models. In *International Conference on Machine Learning*, pages 3732–3741.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017b. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*, pages 438–449, Valencia, Spain, April. Association for Computational Linguistics.

Jason D Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

Jason D Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the 14th Annual SIGDial Meeting on Discourse and Dialogue*, pages 404–413.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 665–677.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer.

Jason D Williams. 2012. A belief tracking challenge task for spoken dialog systems. In *NAACL-HLT Workshop on future directions and needs in the spoken dialog community: tools and data*, pages 23–24. Association for Computational Linguistics.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016a. Neural generative question answering. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, pages 2972–2978. AAAI Press.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2016b. Neural enquirer: learning to query tables in natural language. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*. AAAI Press.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Incorporating Deep Visual Features into Multiobjective based Multi-view Search Results Clustering

Sayantan Mitra
Department of CSE,
IIT Patna,
Bihta, India
sayantaniem@gmail.com

Mohammed Hasanuzzaman
ADAPT Centre,
School of Computing,
Dublin City University,
Dublin, Ireland

Sriparna Saha
Department of CSE,
IIT Patna,
Bihta, India

Andy Way
ADAPT Centre,
School of Computing,
Dublin City University,
Dublin, Ireland

Abstract

Current paper explores the use of multi-view learning for search result clustering. A web-snippet can be represented using multiple views. Apart from textual view cued by both the semantic and syntactic information, a complementary view extracted from images contained in the web-snippets is also utilized in the current framework. A single consensus partitioning is finally obtained after consulting these two individual views by the deployment of a multi-objective based clustering technique. Several objective functions including the values of a cluster quality measure evaluating the goodness of partitionings obtained using different views and an agreement-disagreement index, quantifying the amount of oneness among multiple views in generating partitionings are optimized simultaneously using AMOSA. In order to detect the number of clusters automatically, concepts of variable length solutions and a vast range of permutation operators are introduced in the clustering process. Finally a set of alternative partitionings are obtained on the final Pareto front by the proposed multi-view based multi-objective technique. Experimental results by the proposed approach on several bench-mark test datasets with respect to different performance metrics evidently establish the power of visual and text based views in achieving better search result clustering.

1 Introduction

Web search results clustering (SRC), also known as ephemeral clustering or post-retrieval clustering has garnered much attention in the past few decades for making web browsing easier for users. The key objective of SRC systems is: for a given query it can return some meaningful labeled clusters from a set of web documents (or web snippets) retrieved from a search engine. Recent years have witnessed a large number of attempts in solving this SRC problem (Di Marco and Navigli, 2013; Scaiella et al., 2012). Most of them have developed some clustering algorithms optimizing a single objective criteria (Osinski and Weiss, 2005; Zamir and Etzioni, 1998; Moreno et al., 2013). But a complex data set like set of web-snippets can be clustered into several alternative partitionings. Therefore to detect all possible partitionings containing clusters of different shapes automatically, application of multiobjective optimization (MOO) for solving the problem of clustering becomes prevalent (Maulik et al., 2011). In this context, Acharya et al. (Acharya et al., 2014) have proposed a multi objective optimization based approach for solving SRC problem by extracting both semantic and syntactic information present in web-snippets. Results attained by this approach outperformed the other existing single objective based approaches. Moreover a web-snippet can be represented using different views, for example semantic view, syntactic view. Recently, Wahid et al. (Wahid et al., 2014) have developed a multi-view based MOO clustering technique for search result clustering where multiple views are consulted for developing a consensus partitioning of available web-snippets.

Multimodal approaches have gained increased attention over the past few years. These models have been used in various applications: image captioning (You et al., 2016); sentiment analysis (Poria et al.,

Proceedings of the 27th International Conference on Computational Linguistics, pages 3793–3805
Santa Fe, New Mexico, USA, August 20-26, 2018.

2016); multimodal machine translation (Specia et al., 2016); visual question answering (Antol et al., 2015); combating human trafficking (Tong et al., 2017); and detection of Cyber-bullying (Zhong et al., 2016). In today's online environment, the strategic use of multimedia (image, video etc.) has become increasingly important part of creating a successful website. Visual information embedded in web documents provides right-angled information that is free of ambiguities of natural language. It can also improve search ranking and Search engine optimization (SEO) scores on many levels that contribute to search visibility, find-ability, user satisfaction, experience and engagement. As a result, use of multimedia content in web documents has become more dominant than text in recent years. Rich content of multimedia data, constructed in alliance with the information contained in different modalities, calls for new and innovative methods for better management of web search results.

In this paper, we hypothesize that high quality clustering can be obtained by representing different objective functions over different views of web documents and simultaneously optimizing them. In contrast to prior works which solely depend on information extracted from text we present a multi objective based multiview clustering algorithm which integrates both visual (images) and text content of web-documents for solving SRC problem. Experimental results show that this new multi-view approach significantly outperforms state-of-the-art unimodal approaches. We motivate this paper with a real example which

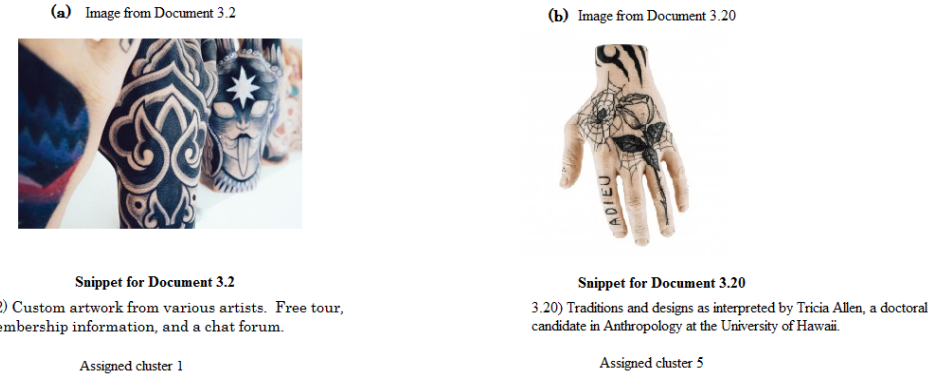


Figure 1: Example where text content differs but image information is same.

demonstrates the potential benefit of integrating visual information for better organization of web search results. Figure 1 shows results from state-of-the-art approach that depends only on text part of the web document for the web search query "3.1 Arts_Bodyart_Tattoo" in ODP-239 dataset. Even though it is clear from looking at the images that they should belong to same cluster, text snippets are not providing enough clues, causing it to appear in different clusters. It is evident from the example that there is complimentary information in the images that is either unavailable in the text or can be in contrast to the text. Our hypothesis is that we can improve SRC performance if we can capture this complimentary high-level visual information. We use pre-trained 19-layer VGG net (Simonyan and Zisserman, 2014) to extract high-level features directly from the image. After extracting the image features for all the images in our dataset, *image view* of each document is generated (Section 3.1). To generate the *textual view* of each document, we combine the benefits of both Word2Vec (Mikolov et al., 2013) and TF-IDF. While Word2Vec vectors are better at capturing the generalized meaning of each word, combining them together by assigning equal weight to all words of a document to generate its *textual view* is not ideal for our task. For example, words that contribute to the syntax rather than the meaning of a sentence should have lower impact on clustering algorithm compared to more specific rare words. Therefore, we scale each word vector by the corresponding TF-IDF weight for that word and generate the *textual view* following the approach in Section 3.1. Finally, we combine both views of the dataset by our multi-objective based multi-view clustering framework.

In order to draw conclusive remarks, we present an exhaustive evaluation where our multi objective based multi-view algorithm (MOO-Multiview-PBM) is compared to the most competitive text-based (endogenous) SRC algorithms: STC (Zamir and Etzioni, 1998), Bisecting Incremental K-means (BIK), LINGO (Osinski and Weiss, 2005), GK-means (Moreno et al., 2013), MOO-clus (Acharya et al., 2014)

and MMOEA (Wahid et al., 2014). Experiments are conducted on three different standard data sets (MORESQUE, ODP- 239 and AMBIENT) for two clustering evaluation metrics (F_{b^3} and F_1). Results show that MOO-Multiview-PBM exceeds all text-based approaches and solutions. In this paper, our main contributions are as follows: **a)** As far the best of our knowledge, this is the first attempt to solve SRC by using both textual & visual information; **b)** A new multi objective based multi-view clustering algorithm for SRC, that determines the number of clusters automatically; and **c)** Novel text representation of document in the context of SRC by combining the benefits of both word2vec and TF-IDF.

2 Related Work

2.1 SRC Algorithms

Suffix Tree Clustering (STC) algorithm proposed by Zamir and Etzioni (1998), is a clustering technique, that combines base clusters having maximum string overlaps based on web snippets represented as compact tries. Results showed improvements over K-means, agglomerative hierarchical clustering, buckshot, single-pass and fractionation algorithms, and this approach is a tough baseline to beat Moreno and Dias (2014). Later, authors of Osinski and Weiss (2005) presented an approach named LINGO which utilizes similar representation of strings as done in Zamir and Etzioni (1998). Initially frequent phrases were extracted based on suffix-arrays and later the group descriptions were matched with topics generated with latent semantic analysis. Documents were then assigned to their relevant groups. Carpineto et al. (2009) showed that the nature of the outputs obtained by SRC algorithms recommend the adoption of a meta clustering approach. The core idea is to combine the complementary results obtained from SOO (single objective) solutions. A novel approach that computes the agreement of two partitions of objects into varied clusters was proposed in this paper. This is done depending on the information content related to the series of decisions made by the partitions on single pairs of objects. OPTIMSRC results demonstrated that meta clustering is far better than individual clustering techniques. Moreno et al. (2013), adapted the K-means algorithm to a third-order similarity measure and proposed a stopping criterion that determines the optimal number of clusters automatically. Experiments were conducted on two standard data sets, MORESQUE (Navigli and Crisafulli, 2010) and ODP-239 (Carpineto and Romano, 2010), and showed significant improvement over all existing text-based SRC techniques developed by then.

Later, Acharya et al. (2014), first defined the SRC task as a multi-objective problem. They defined two objective functions (separability & compactness), that are optimized parallely with the help of AMOSA (Bandyopadhyay et al., 2008). Their evaluations outperformed knowledge-driven exogenous strategies (Scaiella et al., 2012), text-based endogenous SRC approaches and algorithms.

In another work, the multi-view clustering approach proposed by Wahid et al. (2014) used the search capability of multi-objective optimization for SRC. It is basically a cluster ensemble approach where the outputs of multiple clustering techniques like hierarchical clustering approaches and K-means are combined efficiently using NSGA-II (Deb et al., 2002) (non-dominated sorting genetic algorithm-II).

A considerable amount of works have also been proposed that uses exogenous information to solve the SRC problem. One such work is proposed by Scaiella et al. (2012) which uses Wikipedia articles to develop a bipartite graph and employs spectral clustering over it to discover relevant clusters. Recently, authors of Di Marco and Navigli (2013) presented an approach to incorporate word sense induction on the Web1T corpus (Brants and Franz, 2006) that improves SRC.

2.2 Multi-view Clustering

Multi-view data sets are frequent in real life because of the use of different modalities of data input and generation, viz., text, video and audio. The growth of multi-view data in real-world applications has increased the curiosity in multi-view learning (Sun, 2013). Multi-view clustering techniques try to explore the available multiple representations of data for obtaining a precise and robust partitioning of the data in contrast to single-view clustering. A two-view expectation maximization based clustering technique was developed by (Bickel and Scheffer, 2004). A two-view spectral clustering algorithm that generates a bipartite graph was developed by (De Sa, 2005). Another multi-view spectral clustering technique was proposed by (Kumar and Daumé, 2011). A convex mixture model based multi-view

clustering technique was proposed by (Tzortzis and Likas, 2009). The same authors later proposed a kernel-based weighted multi-view clustering technique (Tzortzis and Likas, 2012). A cluster ensemble based approach for multi-view clustering was proposed by (Xie and Sun, 2013). A new multi-view based K-means clustering technique was developed by (Cai et al., 2013) for large-scale data sets. (Wahid et al., 2014) have developed a cluster ensemble based technique to solve the multi-view clustering problem for web documents. Although it presented a multi-view based algorithm MMOEA for solving SRC problem but all the views (i.e., different representations of the data set) are related to semantic and syntactic contents of the web snippets.

3 Multiobjective Multi-view Approach for SRC

In this work we have proposed a multiobjective based multi-view approach, namely *MOO-Multiview-PBM* for solving the problem of search result clustering.

3.1 Generation of Different Views

In the current study we have used some standard data sets of SRC problem for the purpose of evaluation: AMBIENT, MORESQUE (Navigli and Crisafulli, 2010) and ODP-239 (Carpineto and Romano, 2010). For each of web-snippets present in the data set, we have generated two views as follows:

1. *The Textual view*: This view represents both syntactic and semantic information of a document given a particular query. This is generated by combining the document similarity matrix obtained from both word embedding and TF-IDF. Using word embedding each word in the vocabulary is represented by a vector of dimension 1×100. Document similarity matrix using word embedding is generated by Equation 1.

$$S_{emb}(\overline{d}_i, \overline{d}_j) = \frac{1}{\|d_i\|\|d_j\|} \sum_{r=1}^{\|d_i\|} \sum_{b=1}^{\|d_j\|} CosSim(w_i^r, w_j^b) \tag{1}$$

   Here, w_i^r (resp. w_j^b) represents the r^{th} (resp. b^{th}) word vector of document $\overline{d}_i$ (resp. $\overline{d}_j$). $\|d_i\|$ and $\|d_j\|$ denote the total number of words in documents $\overline{d}_i$ and $\overline{d}_j$, respectively. $CosSim(.,.)$ is the cosine similarity between two vectors. TF-IDF is generated using the following steps:

   (a) The terms in the documents are first extracted,

   (b) A document-term matrix is created. Here, a row, a column and a cell correspond to a document, a term, and the weighted value of a term for a document, respectively,

   (c) TF-IDF (a common weighting scheme) values are used to fill this matrix. Each cell contains the TF-IDF score of a term given a document.

   The cosine similarity is calculated between TF-IDF vectors of two documents to generate the $document \times document$ similarity matrix S_{tfidf}. Here $S_{tfidf}(\overline{d}_i, \overline{d}_j)$ contains cosine similarity between TF-IDF vectors of two documents, $\overline{d}_i$ and $\overline{d}_j$.
   The Final $document \times document$ similarity matrix, S_{text}, is generated by the equation:

$$S_{text}(\overline{d}_i, \overline{d}_j) = S_{emb}(\overline{d}_i, \overline{d}_j) \times S_{tfidf}(\overline{d}_i, \overline{d}_j), i, j = 1, \ldots, n \tag{2}$$

   Here n is the total number of documents.
2. *The Image view*: Images are extracted from each web document. We feed images to the pre-trained VGG19 network which computes a 4096 dimensional feature vector for every image that contains the activations of the hidden layer ('fc7') immediately before the VGG's object classifier. Given two image vectors img_i and img_j from two different web documents $\overline{d}_i$ and $\overline{d}_j$, respectively, the similarity between the documents is calculated by Equation 3.

$$S_{image}(\overline{d}_i, \overline{d}_j) = \frac{1}{\|d_i\|\|d_j\|} \sum_{r=1}^{\|d_i\|} \sum_{b=1}^{\|d_j\|} CosSim(img_i^r, img_j^b) \tag{3}$$

   Here, img_i^r (resp. img_j^b) represents the r^{th} (resp. b^{th}) image of the $\overline{d}_i$ (resp. $\overline{d}$) web document. $\|d_i\|$ and $\|d_j\|$ denote the total number relevant images present in $\overline{d}_i$ and $\overline{d}_j$, respectively.

3.2 String Representation and Archive Initialization

The first step of the proposed clustering approach is to initialize the Archive used in AMOSA (Bandyopadhyay et al., 2008) with some alternative diverse set of solutions. The solutions are generated randomly. Here each solution contains a set of cluster centroids (representative web-snippets) in order to represent the partitioning of web-snippets.

The number of cluster centroids encoded in a particular solution i, denoted by K_i, is selected randomly from the given range K_{min} to K_{max} as follows: $K_i = (rand()\text{mod}(K_{max} - 1)) + K_{min}$. For the purpose of initialization, K_i number of web-snippets are randomly selected from the data set and the corresponding indices are used as the initial cluster centers.

3.3 Formation of Clusters and Objective Function Calculations

After initializing the archive members with some randomly selected cluster centroids, the following steps are executed to compute different objective functions. The search capability of AMOSA can be utilized to simultaneously optimize these objective functions.

1. First, the set of representative web-snippets present in the string are extracted. Let the entire set be $\{\overline{C}_1, \overline{C}_2, \ldots, \overline{C}_K\}$ here $\overline{C}_i$ is the ith cluster representative. K is the number of clusters encoded in that particular string. *K*-medoids clustering is applied to the dataset using this set of cluster representatives for different views. In case of text-view, a particular document $\overline{d}_i$ is assigned to cluster t whose centroid has the maximum similarity value to $\overline{d}_i$ given by $t = argmax_{k=1,\ldots K} S_{text}(\overline{d}_i, \overline{C_k})$.Here $S_{text}(\overline{d}_i, \overline{C_k})$ is computed using Equation 2.

   In case of visual view, a particular document $\overline{d}_i$ is assigned to cluster t whose centroid has the maximum similarity value to $\overline{d}_i$. $t = argmax_{k=1,\ldots K} S_{image}(\overline{d}_i, \overline{C_k})$.Here $S_{image}(\overline{d}_i, \overline{C_k})$ is computed using Equation 3.

2. The PBM-index values are calculated for the final partitionings obtained using individual views. Let the values be denoted by PBM_v, $v = 1, 2$.

3. The adjoint matrix (A^v of size $n \times n$, where n is the number of documents) corresponding to view v is calculated as follows:

$$A^v_{ij} = 1 \text{ if } \overline{d}_i \text{ and } \overline{d}_j \text{ belong to the same cluster or } i = \text{j} \quad (4)$$

$$= 0 \quad \text{otherwise} \quad (5)$$

4. A new objective function *Agreement Index* is calculated as follows. This measures the agreement between the partitionings obtained using multiple views. The measure is calculated as follows:

   At a time two views are considered: $v1$ and $v2$. Let the corresponding adjoint matrices be A^{v1} and A^{v2}, respectively. The number of agreement (n_a) is calculated as follows: $n_a = \sum_{i=1}^{n} \sum_{j=1}^{n} I_{A^{v1}_{ij}, A^{v2}_{ij}}$, here

$$I_{A^{v1}_{ij}, A^{v2}_{ij}} = 1 \text{ if } A^{v1}_{ij} = A^{v2}_{ij} \quad (6)$$

$$= 0 \quad \text{otherwise}$$

   The number of disagreements (n_d) is calculated as follows: $n_d = n^2 - n_a$. *Agreement index* between these two views ($v1$, $v2$) is calculated as follows: $AI_{v1,v2} = \frac{n_a+1}{n_d+1}$. The values of 1 in the numerator and denominator are used as a normalization factor to avoid the problem of division by zero. The total *Agreement index* for the entire partitioning is calculated as follows:

$$AI = \frac{\sum_{j=1}^{m} \sum_{i=1, j \neq i}^{m} 2 \times AI_{v_j, v_i}}{m \times (m-1)}, \quad (7)$$

   where m is the available number of views.

5. The objective functions corresponding to a particular string are:
   $\{PBM_{text}, PBM_{image}, AI\}$ where PBM_{text} and PBM_{image} are the values of PBM-indices calculated on partitionings obtained using text based view and image based view, respectively.

3.4 Update of String

After the objective functions are calculated, a consensus partitioning is obtained which satisfies all the available views. The cluster centroids corresponding to this consensus partitioning are used to update the string of AMOSA.

- Let the partitioning obtained using two views be represented by π^1, π^2. Let us denote the jth cluster of view v as π_j^v. First some reordering is done among all the obtained partitionings so that there is a one-to-one correspondence between the cluster numbers of different partitionings.
- New cluster centroids $\overline{C}$, are selected among those documents which are present in the same cluster corresponding to both the views. Let common documents of cluster i in both the views be $\{\overline{d}_1^i, \ldots, \overline{d}_m^i\}$. Then the representative of this set, denoted by $\overline{C}_i$, is $\overline{d}_t$ where
$$t = argmax_{k=1}^{m} \frac{\sum_{j=1, j\neq k}^{m} \frac{S_{text}(\overline{d}_k^i, \overline{d}_j^i) + S_{image}(\overline{d}_k^i, \overline{d}_j^i)}{2}}{m-1}. \quad (8)$$
Here S_{final} and S_{image} are calculated using Equation 2 and 3, respectively.
- Next, the newly generated cluster centroids $\overline{C}_j, \quad j = 1, \ldots, K$ are used to obtain the final consensus partitioning as follows: $\pi_j = \{\forall \overline{d}_i \in S : Sim(\overline{d}_i, \overline{C}_j) > Sim(\overline{d}_i, \overline{C}_l) \text{ for } l = 1, \ldots, K, l \neq j\}$. Here, K is the number of clusters encoded in that solution.
$$Sim(\overline{d}_i, \overline{C}_j) = \frac{S_{text}(\overline{d}_i, \overline{C}_j) + S_{image}(\overline{d}_i, \overline{C}_j)}{2} \quad (9)$$
Here S_{final} and S_{image} are calculated using Equation 2 and 3, respectively. S denotes the set of all documents.
- Finally, the representatives of clusters $\pi_j, \quad j = 1 \ldots, K$ are calculated using Equation 8. These new cluster centroids $\overline{C}_j, \ j = 1, \ldots, K$ are used to replace the old centroids encoded in the string.

So, in order to obtain a consensus partitioning, initially the common points of different clusters present in different partitionings obtained using different views are identified. These points are further used to determine cluster centroids. The other points are assigned to the centroids having maximum similarity, calculated by Equation 9, to obtain a final consensus partitioning. These cluster centroids are used to update the given string.

3.5 Search Operators

In order to explore the search space efficiently using AMOSA, perturbation operations are introduced. These operators also help in generating some new solutions from the current solution which can further take part in the search process. For this purpose, three different perturbation operators are introduced. Below we describe three types of mutation operations in detail:

Mutation 1: In this operation each centroid is parsed individually and with some probability the existing document id in the centroid is replaced by a new document id which is selected from the document collection randomly [1].

Mutation 2: Here a cluster centroid is randomly selected and it is deleted from the string. Size of the string is reduced by 1.

Mutation 3: Here a web document is randomly chosen and the corresponding index is added to the string. Size of the string is increased by 1.

All the above mentioned three mutation operations are equi-probable. Any one of the above discussed mutation operators is applied on a particular solution to generate a new solution which can further participate in the process of AMOSA.

4 Results and Discussion

Results of different approaches are shown in Table 2 for all the data sets with respect to different performance metrics. From Table 2, it is evident that our proposed approach *MOO-Multiview-PBM* outperforms the results of *MOO-clus* (Acharya et al., 2014) by a margin of around $6.2\% - 6.5\%$ in terms

[1] existing document id may or may not get replaced by the new document id

of $F_{b^3} - measure$ and $7.0\% - 7.3\%$ in terms of $F1 - measure$ for both MORESQUE and ODP-239 datasets. Box plots of results obtained from combined $F1 - measure$ and combined $RandIndex$ over all three data sets (i.e., AMBIENT, MORESQUE and ODP-239) by our proposed method are reported in Figure 4. This is done to compare the performance of *MOO-Multiview-PBM* with MMOEA algorithm (Wahid et al., 2014). Although, AMBIENT has received less attention since the creation of ODP-239, we have included it to show a fair comparison of our results with the results from MMOEA (Wahid et al., 2014). In Figure 4, comparison with MMOEA (Wahid et al., 2014) is illustrated. *MOO-Multiview-PBM (word2vec*tfidf)* reports an average F1 measure and Rand Index values of 0.74 and 0.77, respectively, over all three datasets combined, which are similar to those obtained by MMOEA. Note that MMOEA is based on three views (Topics, terms and senses) and also utilized some external information like Wikipedia data during its processing. But our proposed method attains comparable results using information extracted only from the given datasets. No external information was used during the computation of our approach.

Table 1: Evaluation results in terms of $F1$ and F_{b^3} over MORESQUE and ODP239 data sets: Comparison of the proposed approach with state-of-the-art approaches.†→ Results are obtained by 10 consecutive runs of the algorithm and are statistically significant.

		MOO-Multiview-PBM (word2vec*tf-idf)		**MOO-Multiview-PBM (word2vec)**		**MOO-Clus**		**SOO-SRC**			
		Min	*Max*	*Min*	*Max*	*Min*	*Max*	*GK-means*	*STC*	*LINGO*	*BIK*
MORESQUE	F_{b^3}	0.506	**0.564**†	0.510	0.557	0.477	0.502	0.482	0.460	0.399	0.315
	F1	0.698	**0.742**†	0.682	0.728	0.658	0.675	0.655	0.455	0.326	0.317
ODP-239	F_{b^3}	0.491	**0.549**†	0.482	0.531	0.478	0.484	0.452	0.403	0.346	0.307
	F1	0.438	**0.474**†	0.431	0.462	0.379	0.384	0.366	0.324	0.273	0.2

Table 2: Evaluation results in terms of $F1$ and F_{b^3} over MORESQUE and ODP239 data sets: Comparison of the proposed approach over each single view.

		MOO-image		**MOO-word2vec**		**MOO-word2vec*tfidf**	
		Min	*Max*	*Min*	*Max*	*Min*	*Max*
MORESQUE	F_{b^3}	0.427	0.4684	0.479	0.5174	0.489	0.5267
	F1	0.613	0.657	0.664	0.6812	0.6793	0.6973
ODP-239	F_{b^3}	0.4429	0.4725	0.4803	0.4831	0.4821	0.4921
	F1	0.342	0.376	0.3814	0.3901	0.4031	0.4083

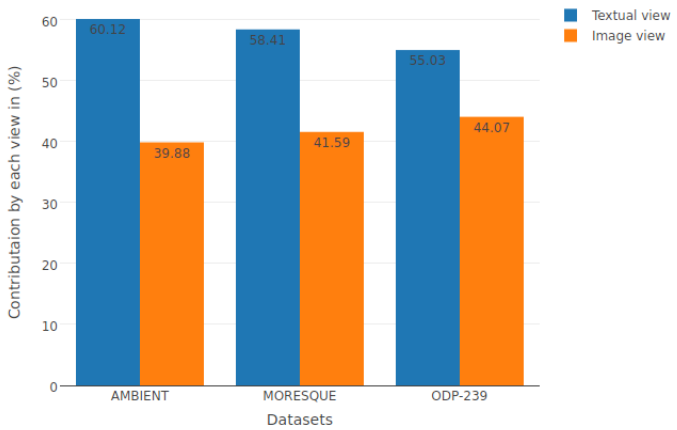


Figure 2: The following bar graph shows contribution by each view in the clustering process for different datasets.

We have implemented our proposed algorithm using two different versions of textual views. Initially we have used word embeddings (Mikolov et al., 2013) for the textual view and combined it with image view in our algorithm *MOO-Multiview-PBM(word2vec)*. Later we have used word embeddings and tf-idf vector together, described in Section 3.1, as textual view along with image view for our algorithm *MOO-Multiview-PBM(word2vec*tf-idf)*. Results in Table 2 show that *MOO-Multiview-PBM(word2vec*tf-idf)* achieves an improvement of $1.2\% - 1.8\%$ over *MOO-Multiview-PBM(word2vec)*. It substantiates our hypothesis that combining the benefits of both word2vec and TF-IDF can improve the search result clustering performance.

We have also quantified the influence of individual views in obtaining the final partitioning. The degree is expressed as follows:

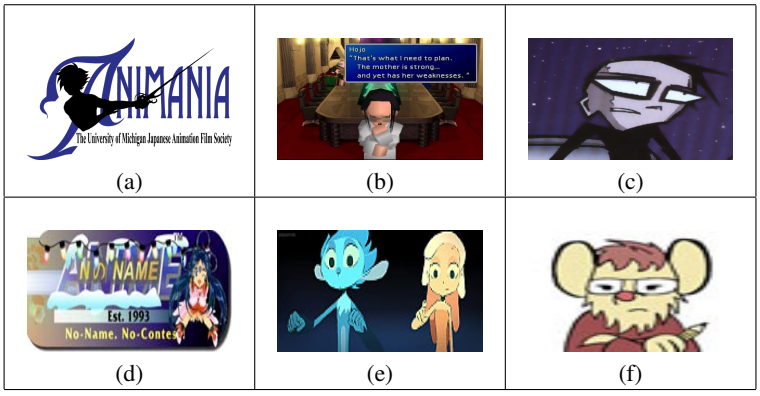
$$Degree_v = \frac{\|A^v \cap A^{\cup v}\|}{\|A^{\cup v}\|}$$

where A^v denotes the adjoint matrix (expressed in Equation 4) corresponding to the partitioning obtained using view v and $A^{\cup v}$ is the adjoint matrix corresponding to the consensus partitioning obtained by consulting all views. The contributions computed as above for different views are shown in Figure 2. It is observed that knowledge extracted from images has contributed on an average 41.58% in the clustering process (refer to Figure 2) for all three datasets.

As mentioned in Section 1 that only textual information is not sufficient for SRC clustering, here we have presented some examples where although these web snippets belong to the same cluster, their textual information widely vary whereas their corresponding image information classify them into the same cluster. In Table 3, we have listed some sample queries from ODP-239 datasets whose textual contents vary widely related to query. In Figure 3, we have shown corresponding images extracted from websites of above mentioned queries. It is evident from these examples that image information is more relevant to the query compared to the textual information.

Table 3: List of query results whose textual contents differ. **S#Id**: Subtopic Id. **R#Id**: Result Id.

S#Id	Subtopic	R#Id	Results	Figures
1.1	Arts	1.19	List of screenings, a few MP3s, and series information.	3a and 3d
	Animation	1.20	Events, news, forum, and newsletter.	3b and 3e
	Anime	1.21	History, constitution, showings, and tape library.	3c and 3f

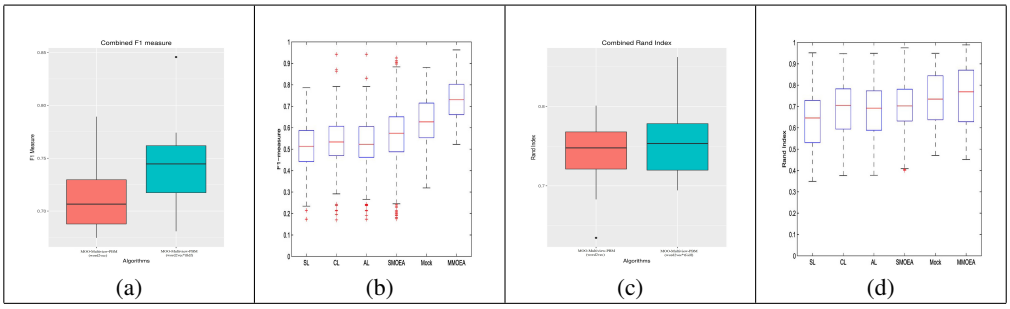


(a) (b) (c) (d) (e) (f)

Figure 3: Sample figures extracted from each query given in Table 3.

5 Error Analysis

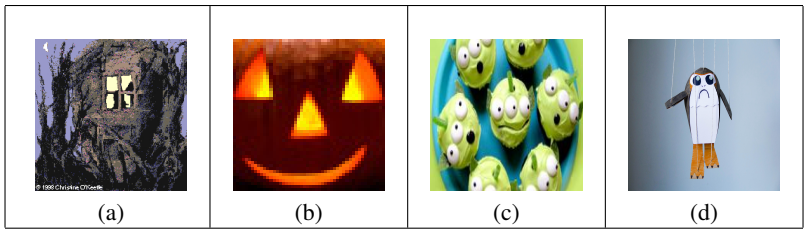
The errors made by our proposed method have been thoroughly analyzed. After a thorough manual analysis, it has been observed that misclassification occurred because of the following reasons. Firstly, from Table 4, it can be seen that there are many inactive web links from which we failed to extract images. For those instances only textual information from the snippets are used for classification. For example, in ODP-239 dataset results 2.54 and 2.65 belong to the same cluster in original labelling. Web links of both 2.54 and 2.65 are inactive, therefore only textual information is used for clustering.



(a) (b) (c) (d)

Figure 4: Boxplots of the best (a) F1-measure (c) Rand index values obtained for different queries combining all the data sets together after application of the proposed MOO-Multiview-PBM technique. Boxplots of the best (b) F1-measure (d) Rand index values obtained for different queries combining all the data sets together after application of MMOEA (taken from the paper (Wahid et al., 2014)).

But in these web snippets text varies widely, viz.,"Meeting schedules, contact information, calendar of events, ftm, mtf, soffa, information, articles, book catalog, and guestbook." in 2.65 and "Headline links from media sources worldwide." in 2.54, hence in predicted clustering solution both 2.54 and 2.65 belong to different clusters. Secondly, there are few instances where textual view and image view differ widely, consensus between these two views is very less hence leads to misclassification. For example, in ODP-239 dataset query 196.1 and 196.11 originally belong to the same cluster. But the text contents, viz.,"Fairy and monster identification list, free e-cards, customs, recipes, games, history, and links to other holiday pages." in 196.1 and "Includes crafts, recipes, costumes, games and activities, and party ideas. Also features autumn harvest party ideas with pumpkin crafts." in 196.11, and image contents of queries 196.1 (see Figures 5a and 5b) and 196.11 (see Figures 5c and 5d) differ widely hence no concrete consensus is drawn between the two views, therefore in predicted clustering solution they are misclassified.



(a) (b) (c) (d)

Figure 5: Sample figures extracted from queries in dataset ODP-239.

6 Conclusion

In the current study a multiobjective based multi-view clustering technique is developed for solving the problem of search result clustering (SRC). Views constructed over textual and visual information contained in web-snippets are considered. We have hypothesized that two web-snippets can be similar either with respect to content or with respect to images. These two views are exploited simultaneously in order to detect good-quality clustering of web-snippets. Three objective functions capturing the qualities of different partitions and a consensus function measuring the similarity between two partitions obtained using different views are simultaneously optimized using the search capability of a MOO process. Improved results on standard bench-mark data sets over state-of-the-art approaches support our hypothesis that use of multi-view information indeed helps in solving the SRC problem. In future we would like to exploit other views of web-snippets and incorporate it in our framework. Investigations of AMOSA as the underlying optimization strategy and PBM -index as the internal cluster validity index are also required to be carried out in future.

Acknowledgements

Mohammed Hasanuzzaman and Andy Way would like to acknowledge ADAPT Centre for Digital Content Technology, funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-

funded under the European Regional Development Fund.

References

Sudipta Acharya, Sriparna Saha, Jose G Moreno, and Gaël Dias. 2014. Multi-objective search results clustering. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 99–108.

Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. 2009. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information retrieval*, 12(4):461–486.

Enrique Amigó, Julio Gonzalo, and Felisa Verdejo. 2013. A general evaluation measure for document organization tasks. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 643–652. ACM.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

Sanghamitra Bandyopadhyay, Sriparna Saha, Ujjwal Maulik, and Kalyanmoy Deb. 2008. A simulated annealing-based multiobjective optimization algorithm: Amosa. *IEEE transactions on evolutionary computation*, 12(3):269–283.

Steffen Bickel and Tobias Scheffer. 2004. Multi-view clustering. In *ICDM*, volume 4, pages 19–26.

Thorsten Brants and Alex Franz. 2006. Web 1t 5-gram version 1.

Xiao Cai, Feiping Nie, and Heng Huang. 2013. Multi-view k-means clustering on big data. In *IJCAI*, pages 2598–2604.

Claudio Carpineto and Giovanni Romano. 2010. Optimal meta search results clustering. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 170–177. ACM.

Claudio Carpineto, Stanislaw Osiński, Giovanni Romano, and Dawid Weiss. 2009. A survey of web clustering engines. *ACM Computing Surveys (CSUR)*, 41(3):17.

Daniel Crabtree, Xiaoying Gao, and Peter Andreae. 2005. Improving web clustering by cluster selection. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 172–178. IEEE Computer Society.

Virginia R De Sa. 2005. Spectral clustering with two views. In *ICML workshop on learning with multiple views*, pages 20–27.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197.

Antonio Di Marco and Roberto Navigli. 2013. Clustering and diversifying web search results with graph-based word sense induction. *Computational Linguistics*, 39(3):709–754.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification*, 2(1):193–218.

Abhishek Kumar and Hal Daumé. 2011. A co-training approach for multi-view spectral clustering. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 393–400.

Ujjwal Maulik, Sanghamitra Bandyopadhyay, and Anirban Mukhopadhyay. 2011. *Multiobjective Genetic Algorithms for Clustering: Applications in Data Mining and Bioinformatics*. Springer Science & Business Media.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jose G Moreno and Gaël Dias. 2014. Easy web search results clustering: When baselines can reach state-of-the-art algorithms. In *14th Conference of the European Chapter of the Association for Computational Linguistics*.

José G Moreno, Gaël Dias, and Guillaume Cleuziou. 2013. Post-retrieval clustering using third-order similarity measures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 153–158.

Roberto Navigli and Giuseppe Crisafulli. 2010. Inducing word senses to improve web search result clustering. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 116–126. Association for Computational Linguistics.

Stanislaw Osinski and Dawid Weiss. 2005. A concept-driven algorithm for clustering search results. *IEEE Intelligent Systems*, 20(3):48–54.

Malay K Pakhira, Sanghamitra Bandyopadhyay, and Ujjwal Maulik. 2004. Validity index for crisp and fuzzy clusters. *Pattern recognition*, 37(3):487–501.

Soujanya Poria, Iti Chaturvedi, Erik Cambria, and Amir Hussain. 2016. Convolutional mkl based multimodal emotion recognition and sentiment analysis. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 439–448. IEEE.

Ugo Scaiella, Paolo Ferragina, Andrea Marino, and Massimiliano Ciaramita. 2012. Topical clustering of search results. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 223–232. ACM.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. 2016. A shared task on multimodal machine translation and crosslingual image description. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, volume 2, pages 543–553.

Shiliang Sun. 2013. A survey of multi-view machine learning. *Neural Computing and Applications*, 23(7-8):2031–2038.

Edmund Tong, Amir Zadeh, Cara Jones, and Louis-Philippe Morency. 2017. Combating human trafficking with multimodal deep models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1547–1556.

Grigorios Tzortzis and Aristidis Likas. 2009. Convex mixture models for multi-view clustering. In *International Conference on Artificial Neural Networks*, pages 205–214. Springer.

Grigorios Tzortzis and Aristidis Likas. 2012. Kernel-based weighted multi-view clustering. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 675–684. IEEE.

Abdul Wahid, Xiaoying Gao, and Peter Andreae. 2014. Multi-view clustering of web documents using multi-objective genetic algorithm. In *IEEE Congress on Evolutionary Computation*, pages 2625–2632.

Xijiong Xie and Shiliang Sun. 2013. Multi-view clustering ensembles. In *Machine Learning and Cybernetics (ICMLC), 2013 International Conference on*, volume 1, pages 51–56. IEEE.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4651–4659.

Oren Zamir and Oren Etzioni. 1998. Web document clustering: A feasibility demonstration. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 46–54. ACM.

Haoti Zhong, Hao Li, Anna Cinzia Squicciarini, Sarah Michele Rajtmajer, Christopher Griffin, David J Miller, and Cornelia Caragea. 2016. Content-driven detection of cyberbullying on the instagram social network. In *IJCAI*, pages 3952–3958.

Appendix A. Objective Functions

PBM index: This is a popular cluster validity index proposed by Pakhira, Bandyopadhyay and Maulik (Pakhira et al., 2004). It outperforms most of the cluster validity indices in the literature in properly detecting the optimal partitioning. This index is defined as follows:

$$PBM(K) = (\frac{1}{K} \times \frac{\mathcal{E}_1}{\mathcal{E}_\mathcal{K}} \times D_K) \tag{10}$$

Here, K denotes the number of clusters, $\mathcal{E}_\mathcal{K} = \sum_{k=1}^{K}\sum_{j=1}^{n_k} dist(\overline{C}_k, \overline{d}_j^k)$ and $D_K = \max_{i,j=1}^{K} dist(\overline{C}_i, \overline{C}_j)$, where $\overline{C}_j$ denotes the centroid of the j^{th} cluster and $\overline{d}_j^k$ denotes the j^{th} web-snippet of the k^{th} cluster. n_k is the total number of web-snippets of the k^{th} cluster. The objective is to maximize the PBM-index. In case of text-view, $dist(\overline{C}_k, \overline{d}_j^k) = 1 - S_{text}(\overline{C}_k, \overline{d}_j^k)$, where S_{text} is calculated using Equation 2 and in case of image-view $dist(\overline{C}_k, \overline{d}_j^k) = 1 - S_{image}(\overline{C}_k, \overline{d}_j^k)$, where S_{image} is calculated using Equation 3. Similarly D_K value between two centroids is also calculated either using text-based similarity or image based similarity measure.

Appendix B. Experimental Setup

We perform web snippet tokenization and word vector generation using gensim library.[2] Word embeddings are obtained from the pre-trained Google news word embeddings.[3] Image features are extracted from the *FC7* layer of *VGG19* available in Keras library.[4] We executed our algorithm over three gold standard datasets: AMBIENT[5]; MORESQUE (Navigli and Crisafulli, 2010); and ODP-239 (Carpineto and Romano, 2010). Description of the datasets, total number of relevant images extracted for each query and number of active query links present in each data set are summarized in Table 4.

The parameters of our proposed algorithms are: $T_{min} = 0.01$, $T_{max} = 100$, $\alpha = 0.85$, $HL = 20$, $SL = 30$, $itr = 20$, $K_{max} = \sqrt{\#ofsamples}$ and $K_{min} = 2$. All these values have been determined after conducting a thorough sensitivity study (and those are in line with the approach proposed by (Acharya et al., 2014)).

Table 4: First part represents SRC gold standard data sets. Second part represents total number of relevant images extracted for each query and number of active query links present in each data set.

Datasets	**# of quaries**	**# of Subtopics Avg/Min/Max**	**# of Snippets**
AMBIENT	44	17.95/6/37	4400
MORESQUE	114	10 / 10 / 10	11400
ODP-239	239	6.7 / 2 / 38	25580

# of web links	**# of active web links**	**# of inactive web links**	**# of images in each quary Avg/Min/Max**
4400	4137	263	6 / 4 / 20
11400	10834	566	8/ 4 / 18
25580	19513	3067	6 / 3 / 25

Appendix C. Evaluation Metrics

An ideal SRC system should be represented by a unique cluster having all the relevant web pages inside. However, determining a unique and complete metric to evaluate the performance of a clustering algorithm is still an open problem (Amigó et al., 2013).

In order to measure the qualities of partitions obtained using different clustering techniques for these web search data sets, we have used three cluster quality measures, F_{b^3} measure (Amigó et al., 2009), F_1 measure (Crabtree et al., 2005) and Rand Index (Hubert and Arabie, 1985). In particular, F_{b^3} has been defined to evaluate completeness, cluster homogeneity, size-vs-quantity and rag-bag constraints. F_{b^3} is a function of $Precision_{b^3}(P_{b^3})$ and $Recall_{b^3}$ (R_{b^3}). All metrics are defined as follows:.

$$F_{b^3} = \frac{2 \times P_{b^3} \times R_{b^3}}{P_{b^3} + R_{b^3}}, P_{b^3} = \frac{1}{N}\sum_{j=1}^{K}\sum_{d_i \in \pi_j} \frac{1}{\mid \pi_i \mid} \sum_{d_l \in \pi_j} h^*(d_j, d_l),$$

$$R_{b^3} = \frac{1}{N}\sum_{j=1}^{K}\sum_{d_i \in \pi_j^*} \frac{1}{\mid \pi_i^* \mid} \sum_{d_l \in \pi_j^*} h^*(d_j, d_l)$$

here π_j is th j^{th} cluster and π_j^* is the gold standard of j^{th} cluster. $h(.,.)$ and $h^*(.,.)$ are defined in Equation 11.

[2] https://radimrehurek.com/gensim/
[3] https://code.google.com/archive/p/word2vec/
[4] https://keras.io/
[5] http://credo.fub.it/ambient

$$h^*(d_j, d_l) = \begin{cases} 1 & \iff \exists i \quad : d_j \in \pi_i^* \wedge d_l \in \pi_i^* \\ 0 & \quad\quad\;\; : otherwise \end{cases}, h(d_j, d_l) = \begin{cases} 1 & \iff \exists i \quad : d_j \in \pi_i \wedge d_l \in \pi_i \\ 0 & \quad\quad\;\; : otherwise \end{cases} \tag{11}$$

Appendix D. Model Comparisons

In order to comprehensively evaluate the performance of our proposed approach *(MOO-Multiview-PBM)*, we listed some strong baseline approaches for comparison.

- **MOO-clus(Acharya et al., 2014):** This algorithm uses archived multi-objective simulated annealing framework to simultaneously optimize two objectives, compactness and separation, for clustering web snippets.
- **GK-means(Moreno et al., 2013):** This algorithm has adapted the K-means algorithm to a third-order similarity measure and proposed a stopping criterion to automatically determine the number of clusters.
- **Suffix Tree Clustering(STC)(Zamir and Etzioni, 1998):** It is an incremental, linear time algorithm which creates clusters based on phrases shared between documents.
- **LINGO(Osinski and Weiss, 2005):** In this method, initially the frequent phrases based on suffix-arrays are extracted and later matched the group description with topics obtained with latent semantic analysis.
- **MMOEA(Wahid et al., 2014):** This algorithm uses multiple views to generate different clustering solutions and then select a combination of clusters to form a final clustering solution.

Integrating Tree Structures and Graph Structures with Neural Networks to Classify Discussion Discourse Acts

Yasuhide Miura[†,‡]
yasuhide.miura@fujixerox.co.jp

Ryuji Kano[†]
kano.ryuji@fujixerox.co.jp

Motoki Taniguchi[†]
motoki.taniguchi@fujixerox.co.jp

Tomoki Taniguchi[†]
taniguchi.tomoki@fujixerox.co.jp

Shotaro Misawa[†]
misawa.shotaro@fujixerox.co.jp

Tomoko Ohkuma[†]
ohkuma.tomoko@fujixerox.co.jp

[†]Fuji Xerox Co., Ltd.
[‡]Tokyo Institute of Technology

Abstract

We proposed a model that integrates discussion structures with neural networks to classify discourse acts. Several attempts have been made in earlier works to analyze texts that are used in various discussions. The importance of discussion structures has been explored in those works but their methods required a sophisticated design to combine structural features with a classifier. Our model introduces tree learning approaches and a graph learning approach to directly capture discussion structures without structural features. In an evaluation to classify discussion discourse acts in Reddit, the model achieved improvements of 1.5% in accuracy and 2.2 in F_1 score compared to the previous best model. We further analyzed the model using an attention mechanism to inspect interactions among different learning approaches.

1 Introduction

With the recent growth of social news sites and debate portals, people today discuss various topics online. These discussions include valuable public opinions of crowds. However, automated analyses of them are often difficult, requiring understanding of textual contents and discussion structures. By extending approaches taken in the analysis of spoken dialogue acts (Stolcke et al., 2000; Bunt et al., 2010), there have been a number of attempts to analyze discussions with discourse acts. Discussions in emails (Cohen et al., 2004; Carvalho and Cohen, 2005; Carvalho and Cohen, 2006; Hu et al., 2009; Omuya et al., 2013), newsgroups (Wang et al., 2007), technical forums (Kim et al., 2010b; Wang et al., 2011; Bhatia et al., 2012; Liu et al., 2017) and social news (Zhang et al., 2017) are targeted in earlier studies. The automatic classification of these discourse acts can improve tasks like information access and summarization for discussions.

The importance of discussion structures for analyzing discourse acts has already been recognized in earlier studies. Relational features (Carvalho and Cohen, 2005), link features (Hu et al., 2009), and structural features (Wang et al., 2007; Kim et al., 2010a; Kim et al., 2010b; Wang et al., 2011; Bhatia et al., 2012; Zhang et al., 2017; Liu et al., 2017) are combined with a probabilistic graphical model, a structured prediction model, or sequential classification models. These approaches have achieved promising results in classifying discourse acts, but they require a sophisticated design to integrate structural features into a classification model. In this paper, we propose a model that integrates discussion structures with neural networks to classify discourse acts. Recently, neural networks have shown their effectiveness at capturing tree structures (Socher et al., 2011; Socher et al., 2014; Tai et al., 2015) and graph structures (Defferrard et al., 2016; Kipf and Welling, 2017). Our model introduces tree learning approaches and a graph learning approach to directly capture discussion structures without structural features to classify discourse acts.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3806–3818
Santa Fe, New Mexico, USA, August 20-26, 2018.

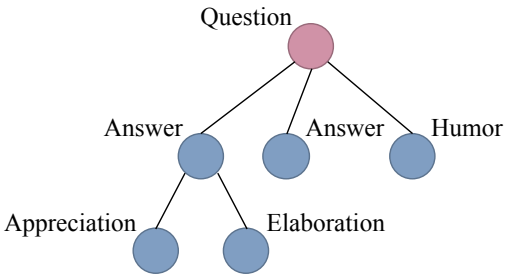


Figure 1: An example of a Reddit thread with discourse acts. The circles represent comments. An initial question is replied by answers and a humor. One of the answer is further replied by an appreciation and an elaboration.

Among the various targets that have been analyzed in earlier studies, we apply our models to the analysis of Reddit[1] discussions. Reddit is among the largest social news sites, hosting discussions of various topics. We aim to classify comments into 9 discourse acts of Zhang et al. (2017), namely *Answer*, *Elaboration*, *Question*, *Appreciation*, *Agreement*, *Disagreement*, *Humor*, *Announcement*, and *Negative Reaction*. Figure 1 illustrates an example of a Reddit thread with discourse acts. We chose Reddit discussions for two reasons. First, threads that comprise Reddit discussions have an explicit tree structure with reply relations as edges. Secondly, the publicly available dataset is larger than other datasets, which enables neural networks to learn tree structures and graph structures. We demonstrate that our model successfully learns discussion structures with tree learning approaches and a graph learning approach of neural networks.

The following are the contributions of this paper:

1. We propose a neural model with tree learning components and a graph learning component that can efficiently learn discussion structures.
2. We show that the proposed model outperforms earlier sequential learning approaches in a discourse act classification.
3. We analyze some components of the proposed model to ascertain the learned effective discussion structures.

In subsequent sections of the paper, we present related works in two criteria in Section 2. Section 3 describes the proposed model. Details of an experiment are reported in Section 4, with discussions in Section 5. Finally, Section 6 concludes the paper with a presentation of some future directions.

2 Related Works

2.1 Discourse Act Classification

The analysis of discussions attracted researchers to propose automated approaches to classify discourse acts. In earlier works, emails and forums were popularly targeted by several works. Cohen et al. (2004) annotated emails with speech acts and built a classifier with textual features for them. This work was extended by Carvalho and Cohen (2005) to combine relational features with a dependency-network-based collective classification model and by Carvalho and Cohen (2006) with an exploitation of n-gram features. Hu et al. (2009) also annotated emails with speech acts and trained a structured prediction classifier. Omuya et al. (2013) extended this work using per-class feature optimization and a cascade of classifiers. Kim et al. (2010b) tagged technical forums with dialogue acts and conducted an experiment on them using structural features and a sequential learner. Later, these dialogue acts were tackled by Wang et al. (2011) with a joint classification approach of discourse acts and link relations, and by Liu et al. (2017) with a sequential learner including an external memory. Bhatia et al. (2012) assigned dialogue acts to technical forums and evaluated classifiers using a variety of features including structural features.

The discourse acts of other targets were also explored in earlier works. Wang et al. (2007) evaluated a sequential learning technique with structural features of newsgroup argument codes. Kim et al. (2010a)

[1]https://www.reddit.com/

explored the classification of dialogue acts in chats with various features including structural ones with a sequential learner. Zhang et al. (2011) annotated tweets with speech acts and trained a classification model with word features and character features. These speech acts were examined further in Zhang et al. (2012) with semi-supervised learners including a graph-based label propagation. Ferschke et al. (2012) created a Wikipedia Talks corpus with dialog acts and trained a binary classifier with textual features and talk turn features. Zhang et al. (2017) annotated Reddit threads with discourse acts and designed sequential models with various features including structural features.

2.2 Reddit Analysis

Reddit, which has become a popular target for researchers for analyses of public opinion, consists of massive textual contents and visual contents with metadata that are readily accessible via an API. Within studies using Reddit as their data, automated analysis of the *karma score*, a post-level popularity of discussions in Reddit based on positive votes and negative votes of users, is widely studied. Jaech et al. (2015) proposed a karma score ranking task and used a pairwise classifier to rank them. Wei et al. (2016) designed a system that ranks comments based on karma scores as a reference to their persuasiveness. He et al. (2016) proposed a deep reinforcement learning architecture for the effective modeling of an online popularity prediction task. He et al. (2017) extended the model using a two-stage approach. Hessel et al. (2017) experimented on a relative popularity task using karma scores and demonstrated the effectiveness of multimodal features. Cheng et al. (2017) introduced a factored neural model that demonstrated improvements on predicting karma scores over standard document embedding methods. Zayats and Ostendorf (2018) presented a graph-structured neural architecture that outperformed a node-independent architecture in predicting karma scores.

Automatic analysis of targets other than karma scores were also investigated in earlier works analyzing Reddit discussions. The analysis of discourse acts (Zhang et al., 2017) is one such work, but some other attempts were also made. Buntain and Golbeck (2014) demonstrated the presence of answer-person roles and their identification with a supervised classifier. Tan et al. (2016) examined *ChangeMyView* subreddit to clarify the mechanism of persuasion and built a classifier for opinion malleability prediction. Lim et al. (2017) explored the estimation of relative user expertise through various content-agnostic approaches. Habernal et al. (2018) investigated ad hominem arguments and experimented with neural models for prediction.

2.3 Comparisons with Our Model

Our model, which we describe in Section 3 learns a discussion structure using tree learning approaches and a graph learning approach. The model processes textual information and structural information jointly within its architecture without structural features. For the classification of discourse acts, most earlier works used sequential learning approaches (Wang et al., 2007; Kim et al., 2010a; Kim et al., 2010b; Wang et al., 2011; Zhang et al., 2017; Liu et al., 2017). Carvalho and Cohen (2005) used a probabilistic graphical model, but it relied on relational features and pre-trained classifier trained with textual features. When we compare our model with Reddit analysis models, a close approach is taken in the model of Zayats and Ostendorf (2018). They extended Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) to accommodate graph structures to capture hierarchical and temporal conversations. However, their model includes designs for efficiently learning the popularity of Reddit. The popularity of a comment in Reddit is known to be associated strongly with the post timing and the post author (Jaech et al., 2015). We present in an experiment explained in Section 4 that a simple application of a popularity prediction model will not perform well on classifying discourse acts.

3 Model

Figure 2 illustrates the overview of our proposed model: Tree-LSTM GCN Hybrid. The model first encodes comments in a thread with an LSTM and max-pooling (Comment Encoder) to comment representations. The comment representations are then updated with tree-level LSTM processes (Parent-Branch Tree-LSTM and Child-Sum Tree-LSTM) and a graph-level convolutional networks process (GCN) to

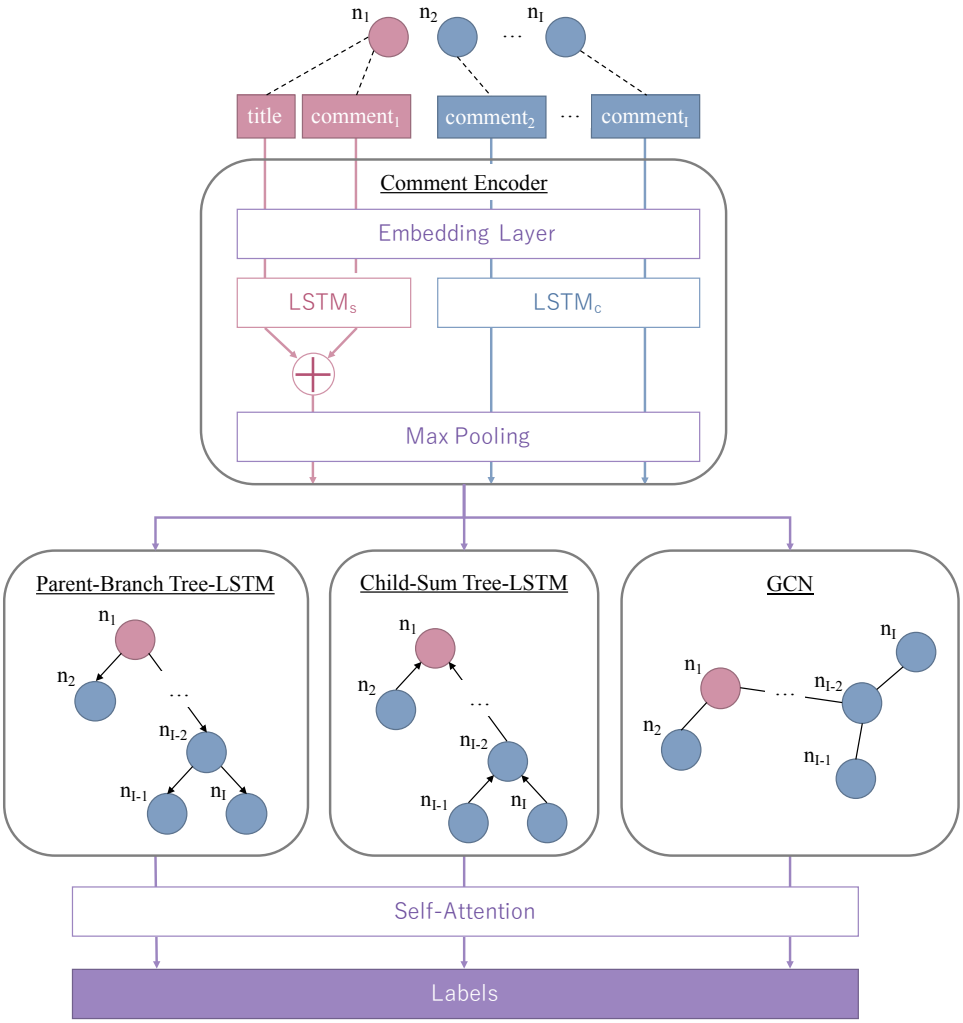


Figure 2: Overview of our proposed model: Tree-LSTM GCN Hybrid. In this model, comments in an input thread is encoded by Comment Encoder and then processed by tree processing components (Parent-Branch Tree-LSTM and Chid-Sum Tree-LSTM) and a graph processing component (GCN). Best viewed with colors.

introduce comment correlations. Finally, the updated comment representations are merged with an attention layer (Self-Attention) and are connected to labels with a fully-connected layer.

Comment Encoder

An input sequence consists of a title text $\boldsymbol{a}_{title}$ and comment texts $\boldsymbol{a}_{1...I}$. The words in these texts are embedded into $\boldsymbol{x}_{title}$ and $\boldsymbol{x}_{1...I}$ with an embedding matrix $\boldsymbol{E}$ in Embedding layer. Embedded inputs are then passed to bi-directional LSTM layers[2] to be processed using the following transition functions as

$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}_i\boldsymbol{x}_t + \boldsymbol{U}_i\boldsymbol{h}_{t-1} + \boldsymbol{b}_i\right) \quad (1)$$
$$\boldsymbol{o}_t = \sigma\left(\boldsymbol{W}_o\boldsymbol{x}_t + \boldsymbol{U}_o\boldsymbol{h}_{t-1} + \boldsymbol{b}_o\right) \quad (2)$$
$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}_f\boldsymbol{x}_t + \boldsymbol{U}_f\boldsymbol{h}_{t-1} + \boldsymbol{b}_f\right) \quad (3)$$
$$\tilde{\boldsymbol{c}}_t = \tanh\left(\boldsymbol{W}_c\boldsymbol{x}_t + \boldsymbol{U}_c\boldsymbol{h}_{t-1} + \boldsymbol{b}_c\right) \quad (4)$$
$$\boldsymbol{c}_t = \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t + \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} \quad (5)$$
$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh\left(\boldsymbol{c}_t\right) \quad (6)$$

where $\boldsymbol{i}_t$ is an input gate, $\boldsymbol{o}_t$ is an output gate, $\boldsymbol{f}_t$ is a forget gate, $\tilde{\boldsymbol{c}}_t$ is a cell gate, $\boldsymbol{c}_t$ is a cell state, $\boldsymbol{h}_t$ is a hidden state, $\boldsymbol{W}_*$ and $\boldsymbol{U}_*$ are weight matrices, $\boldsymbol{b}_*$ are bias vectors, σ is a logistic sigmoid function,

[2]We prepared two separate LSTMs for an initial comment ($\mathrm{LSTM_s}$) and for later comments ($\mathrm{LSTM_c}$).

and $\odot$ is an element-wise multiplication operator. Bi-directional LSTM outputs are concatenated and processed with a max-over time process to obtain a comment representation $\boldsymbol{m} = \max\left(\overrightarrow{\boldsymbol{h}} \| \overleftarrow{\boldsymbol{h}}\right)$. For an initial comment, the title output ($\boldsymbol{h}_{title}$) and the comment output ($\boldsymbol{h}_1$) are added before the max-over time process as $\boldsymbol{m} = \max\left(\overrightarrow{\boldsymbol{h}}_{title} \| \overleftarrow{\boldsymbol{h}}_{title} \oplus \overrightarrow{\boldsymbol{h}}_1 \| \overleftarrow{\boldsymbol{h}}_1\right)$, where $\oplus$ is an element-wise addition operator.

Parent-Branch Tree-LSTM

This component allows a discourse act of a comment to be predicted with information from its parent. The comments representations are processed from the root to the leaves with LSTM. We conducted this process with a simple extension to LSTM by replacing a previous time state $\boldsymbol{h}_{t-1}$ in Eq. 1–4 with a parent state $\boldsymbol{h}_{parent}$. A comment can have multiple children. Therefore, $\boldsymbol{h}_{parent}$ are likely to be shared by multiple comments. For latter processes, $\boldsymbol{h}$ is used as an updated comment representation $\boldsymbol{r}_P$.

Child-Sum Tree-LSTM

This component allows a discourse act of a comment to be predicted with information from its children. The comments representations are processed from the leaves to the root with LSTM. We used Child-Sum Tree-LSTM (Tai et al., 2015) for this process. Child-Sum Tree-LSTM extends LSTM by expressing a state transition of children to parent with a summation. More specifically, a previous time state $\boldsymbol{h}_{t-1}$ in Eq. 1, 2, and 4 are replaced by previous child states $\tilde{\boldsymbol{h}}_t$, with updates to forget gate $\boldsymbol{f}_t$ (Eq. 3) and cell gate $\boldsymbol{c}_t$ (Eq. 5) by

$$\tilde{\boldsymbol{h}}_t = \sum_k \boldsymbol{h}_k \tag{7}$$

$$\boldsymbol{f}_{tk} = \sigma\left(\boldsymbol{W}_f \boldsymbol{m}_t + \boldsymbol{U}_f \boldsymbol{h}_k + \boldsymbol{b}_f\right) \tag{8}$$

$$\boldsymbol{c}_t = \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t + \sum_k \boldsymbol{f}_{tk} \odot \boldsymbol{c}_k \tag{9}$$

where $\boldsymbol{h}_k$ is a hidden state of a child and $\boldsymbol{c}_k$ is a cell state of a child. For latter processes, $\boldsymbol{h}$ is used as an updated comment representation $\boldsymbol{r}_C$.

GCN

This component allows a discourse act of a comment to be predicted with information from surrounding comments. The comments representations are processed with a convolution over a graph. For each node, the representations of the surrounding nodes are considered with a convolution filter. We used the graph convolution approach of Defferrard et al. (2016) which approximates a convolutional filter over a graph with Chebyshev expansion (Hammond et al., 2011). In this approach, input representations $\boldsymbol{h}_l$ are processed with the following transition functions as

$$\boldsymbol{h}_{l+1} = U g_\theta\left(\Lambda\right) U^T \boldsymbol{h}_l \tag{10}$$

$$g_{\boldsymbol{\theta}}\left(\Lambda\right) = \sum_k^{K-1} \boldsymbol{\theta}_k T_k(\tilde{\Lambda}) \tag{11}$$

where U is the Fourier basis of a normalized Laplacian in an input graph, $\boldsymbol{\theta}_k$ is a vector of Chebyshev coefficients, and $T_k(\tilde{\Lambda})$ is a Chebyshev polynomial of order k. This graph convolution process can be stacked easily to perform convolutions over a graph iteratively. We prepared two graph convolution processes with rectified linear unit (ReLU) as an intermediate activation as

$$\boldsymbol{r}_G = U g_\theta\left(\Lambda\right) U^T \mathrm{ReLU}\left(U g_\theta\left(\Lambda\right) U^T \boldsymbol{m}\right) \tag{12}$$

to obtain an updated comment representation $\boldsymbol{r}_G$.

Self-Attention

Outputs of three components are merged with an attention layer (Self-Attention) with a self-attentive style (Yang et al., 2016; Lin et al., 2017). A comment representation $\boldsymbol{s}_i$ is updated as a weighted sum of

$\boldsymbol{r}_{ji}$ with weight α_{ji} as

$$\boldsymbol{s}_i = \sum_{j \in \{P,C,G\}} \alpha_{ji} \boldsymbol{r}_{ji} \tag{13}$$

$$\alpha_{ji} = \frac{\exp\left(\boldsymbol{v}_\alpha^T \boldsymbol{u}_{ji}\right)}{\sum_{j' \in \{P,C,G\}} \exp\left(\boldsymbol{v}_\alpha^T \boldsymbol{u}_{j'i}\right)} \tag{14}$$

$$\boldsymbol{u}_{ji} = \tanh\left(\boldsymbol{W}_\alpha \boldsymbol{r}_{ji} + \boldsymbol{b}_\alpha\right) \tag{15}$$

where $\boldsymbol{v}_\alpha$ is a weight vector, $\boldsymbol{W}_\alpha$ is a weight matrix, and $\boldsymbol{b}_\alpha$ is a bias vector.

4 Experiment

4.1 Baselines

Rule 5-ACTS

We prepared a simple rule-based classifier as a non-machine learning baseline. Given a thread, an initial comment is classified as *Question* if it includes a question mark and as *Announcement* for another case. In later comments, a comment is classified as *Question* if it includes a question mark and as *Appreciation* if it includes "thank", "thanks", "thx", "thxs", or "tks", and as *Answer* or *Elaboration* for another case. In the *Answer* or *Elaboration* case, a comment is classified as *Answer* if it is preceded by a comment with *Question* and as *Elaboration* for another case. Note that comments are never classified with the remaining 4 acts (*Agreement*, *Disagreement*, *Humor*, and *Negative Reaction*) in this classifier.

CRF Vote

We implemented the best model in Zhang et al. (2017) as CRF Vote. This model decomposes a thread into root-to-leaf sequences. Features of content, punctuation, structure, author, thread, and community are extracted and used to train a Conditional Random Field (CRF) with Orthant-Wise Limited-memory Quasi-Newton training algorithm and L1 regularization. The strong effect of structural features were confirmed on an ablation test. In this model, a comment often receives multiple prediction labels since a thread is decomposed into sequences. In that case, the label of a comment is decided by a majority vote of the given labels.

LSTM-CRF Vote

We prepared a straightforward extension of CRF Vote combining Comment Encoder (Section 3), an LSTM layer, and a CRF layer. An architecture combining an LSTM and a CRF is known to be effective for sequential labeling tasks (Huang et al., 2015; Lample et al., 2016; Ma and Hovy, 2016). As in CRF Vote, a thread is first decomposed into root-to-leaf sequences. The decomposed sequences are then processed with Comment Encoder to obtain comment representations. The comment representations are then updated with an LSTM layer to introduce comment correlations. Finally, the updated comment representations are passed to a CRF layer to include label correlations. Like in CRF Vote, the label of a comment with multiple prediction labels is decided by a majority vote of the given labels.

Graph-LSTM

We implemented the graph-structured LSTM in Zayats and Ostendorf (2018) as Graph-LSTM. In this model, LSTM is extended to process parent–child relationships and sibling relationships in a Reddit thread. Relationships of these two kinds are introduced to LSTM by preparing a forget gate for hierarchical (reply) connections and a forget gate for timing (sibling) connections. An input to the extended LSTM is comments represented with concatenated word representation averages and submission context features. Submission context features include structural features such as graph locations and graph responses. This model is not designed for discourse acts, but it has shown superior performances for predicting the popularity of Reddit comments.

Discourse Act	Count	Initial Comment Rate
Answer	40,723	0.00%
Elaboration	18,513	0.00%
Question	17,105	41.32%
Appreciation	8,380	0.00%
Agreement	4,815	0.00%
Disagreement	3,263	0.00%
Humor	2,291	2.71%
Announcement	2,002	100.00%
Negative Reaction	1,773	0.00%

Table 1: Numbers of discourse acts and their initial comment rates in the dataset that we used in the experiment.

4.2 Dataset and Evaluation

To evaluate the proposed model and the baseline models, we used the dataset of Zhang et al. (2017)[3], which consists of 9,483 threads with 115,827 comments from 2,837 communities (subreddits). These comments are annotated with the following 10 discourse acts: *Answer*, *Elaboration*, *Question*, *Appreciation*, *Agreement*, *Disagreement*, *Humor*, *Announcement*, *Negative Reaction*, and *Other*. Following the setting of Zhang et al. (2017), we discarded the comments that have no majority annotation or which have *Other* as a majority annotation. We used the resulting 9,131 threads with 98,865 comments to evaluate our models. The numbers of respective discourse acts and their initial comment rates in the resulting dataset are presented in Table 1.

All models are evaluated with ten-fold cross validation following the setting of Zhang et al. (2017). Accuracy, precision, recall, and F_1 score are used for evaluation metrics. For neural models, the dataset is split into train:validation:test in the ratio of 8:1:1 for each fold to provide validation data. A best performing model in a validation data in terms of F_1 score is used to evaluate the corresponding test data.

4.3 Model Configurations

Pre-training of Word Representations

We pre-trained the word-embeddings using randomly sampled comments from Reddit dumps during 2006–2016[4,5]. We restricted communities to those which appeared in the dataset of Section 4.2, and extracted approximately 230M comments. For pre-training, we used word2vec (Mikolov et al., 2013) with the skip-gram algorithm of parameters dimension=100, learning rate=0.025, window size=5, negative sample size=5, and epoch=5. The pre-trained word representations are used in LSTM-CRF Vote, Graph-LSTM, and Tree-LSTM GCN Hybrid.

Hidden Unit Sizes and Maximum Number of Words

Several layers in our models have hidden units as their parameters. For LSTM layers, we have set hidden unit sizes to $\mathrm{LSTM}_s = 300$, $\mathrm{LSTM}_c = 300$, and the LSTM layer in LSTM-CRF Vote to 600. Parent-Branch Tree-LSTM, Child-Sum Tree-LSTM, and GCN also have hidden units. Their size were set to 600. Reddit comments are sometimes long consisting of thousands of words. We restricted the maximum number of words for each comment to speedup training. The maximum numbers were set to 400 for initial comments and 100 for later comments.

Optimization

We trained LSTM-CRF Vote, Graph-LSTM, and Tree-LSTM GCN Hybrid by stochastic gradient descent. As an optimization objective, the score of CRF is used for LSTM-CRF Vote and cross-entropy loss is used for Graph-LSTM and Tree-LSTM GCN Hybrid. The learning rate is selected from $\{0.01, 0.1\}$ using a validation data along with momentum=0.9 and gradient clipping=3.0 for parameters of stochastic

[3]https://github.com/google-research-datasets/coarse-discourse

[4]https://bigquery.cloud.google.com/dataset/fh-bigquery:reddit_posts

[5]https://bigquery.cloud.google.com/dataset/fh-bigquery:reddit_comments

Model	Accuracy	Precision	Recall	F_1 score
Rule 5-ACTS	0.590	0.513	0.590	0.536
CRF Vote (Zhang et al., 2017)	0.764	0.752	0.764	0.748
LSTM-CRF Vote	0.769	0.757	0.769	0.759
Graph-LSTM (Zayats and Ostendorf, 2018)	0.738	0.720	0.738	0.724
Proposed (Parent-Branch Tree-LSTM)	0.772	0.760	0.772	0.763
Proposed (Child-Sum Tree-LSTM)	0.631	0.609	0.631	0.611
Proposed (GCN)	0.767	0.755	0.767	0.756
Proposed (Tree-LSTM GCN Hybrid)	0.779	0.768	0.779	0.770

Table 2: Accuracy, precision, recall, and F_1 score of the baseline models and the proposed model. For precision, recall, and F_1 score, the weighted average with the number of positive instances over 9 discourse acts.

gradient descent. To avoid overfitting, we introduced dropout (Srivastava et al., 2014) with a rate of 0.5 to the LSTM layers of all models and the intermediate layer of GCN for regularization.

4.4 Result

Table 2 presents the evaluation results. For Tree-LSTM GCN Hybrid, we prepared models that used only a single component as Proposed (*component name*). Results show that our proposed model Tree-LSTM GCN Hybrid outperforms the previous best approach of Zhang et al. (2017) (CRF Vote) by 1.5% in accuracy and 2.2 in F_1 score. LSTM-CRF Vote also outperformed CRF Vote but in a smaller magnitude compared to Tree-LSTM GCN Hybrid. This result suggests the effectiveness of both the simple neural extension and the neural structural learning approaches[6]. Graph-LSTM performed moderately but lower than CRF Vote. This result implies that a popularity predictions and a discourse act classification differs even within Reddit. Rule 5-ACTS performed substantially lower than other machine learning models. This result indicates that simple rules are not enough to solve this task. For computational times, Tree-LSTM GCN Hybrid took approximately 21 hours to perform the ten-fold cross validation with an NVIDIA Titan X gpu. This is about five times slower to CRF Vote, which took approximately 4 hours on the evaluation with an Intel Core i7 cpu core.

Performances of the single component proposed models vary among active components. Results show that Parent-Branch Tree-LSTM performs best, with GCN slightly lower than Parent-Branch Tree-LSTM, and with Child-Sum Tree-LSTM scoring quite low. These results are intuitive because a discussion flow in a thread occurs from the root to the leaves with the chain of replies. GCN considers surrounding nodes in distance of two. Therefore, its performance suggests that the effect of distant comments is not strong to classify the discourse act of a comment.

5 Discussions

5.1 Strategy for Combining Three Components

Our proposed model combines three components to classify discourse acts. We analyzed attention probabilities in Self-Attention layer to see how Tree-LSTM GCN Hybrid merges the three components. Figure 3a shows the estimated probability density functions of all comments. As in the single component models in Section 4.4, Parent-Branch Tree-LSTM is most preferred in the model. However, the next preferred component is Child-Sum Tree-LSTM, which performed quite poorly as a single component model. This observation implies that the performance of an individual component might not relate directly to importance in a hybrid model. Figure 3b shows the estimated probability density functions of initial comments. Initial comments have no parents. Therefore, Child-Sum Tree-LSTM and GCN have stronger preference compared to the all comments case.

We further analyzed attention probabilities in terms of the number of replies. In the case of comments with none or a small number of replies, the probability density function shows a similar trend to the all

[6]We confirmed statistically significant improvements between Tree-LSTM GCN Hybrid and LSTM-CRF Vote for all four metrics with the confidence level of 0.05 by Fisher-Pitman permutation test. The improvements between LSTM-CRF Vote and CRF Vote were also statistically significant with the confidence level of 0.10 for accuracy and the confidence level of 0.05 for precision, recall, and F_1 score.

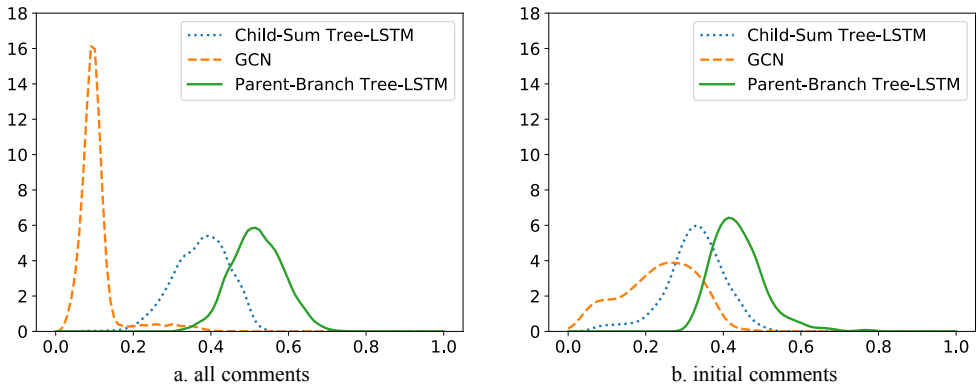


Figure 3: Estimated probability density functions of attention probabilities in the Self-Attention layer.

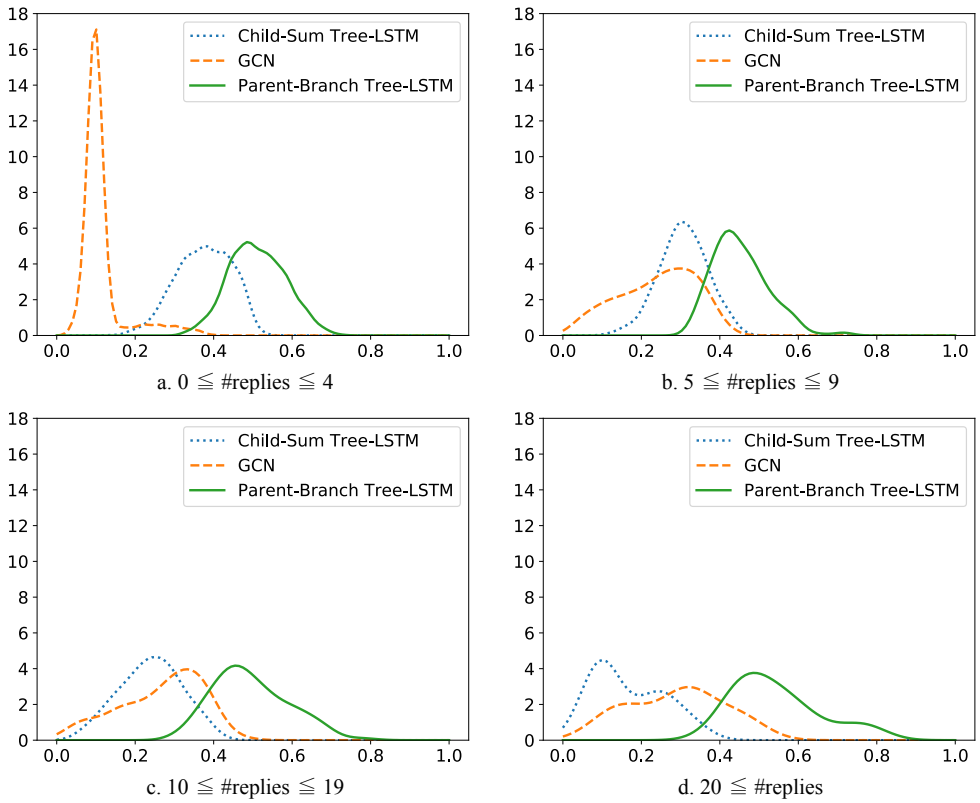


Figure 4: Estimated probability density functions in terms of the number of replies.

comments case (Figure 4a). The preference of Child-Sum Tree-LSTM and the preference of GCN become close in the case with a moderate number of replies (Figure 4b). The preference of GCN surpasses the preference of Child-Sum Tree-LSTM with larger numbers of replies (Figure 4c, 4d). These results imply that when there is a sufficient number of replies, considering nearby comments like in GCN is an effective approach for predicting discourse acts.

5.2 Strengths of Neural Models

In the experiment, we targeted 9 discourse acts for prediction. To clarify the effects of the proposed model, we analyzed the performances of each discourse act. Table 3 shows the detailed F_1 scores of Rule 5-ACTS, CRF Vote, LSTM-CRF Vote, Parent-Branch Tree-LSTM, and Tree-LSTM GCN Hybrid. The analysis shows that the neural models perform on par or better for most acts. Strong improvements in the neural models against CRF Vote are observed in *Disagreement*, *Humor*, and *Negative Reaction*.

Discourse Act	Rule 5-ACTS	CRF Vote	LSTM-CRF Vote	Parent-Branch Tree-LSTM	Tree-LSTM GCN Hybrid
Answer	0.685	0.884	0.892	0.897	0.900
Elaboration	0.325	0.647	0.657	0.660	0.667
Question	0.754	0.853	0.869	0.870	0.877
Appreciation	0.575	0.730	0.704	0.713	0.720
Agreement	0.000	0.460	0.458	0.462	0.475
Disagreement	0.000	0.219	0.280	0.259	0.299
Humor	0.000	0.180	0.285	0.269	0.290
Announcement	0.679	0.787	0.779	0.787	0.819
Negative Reaction	0.000	0.189	0.280	0.302	0.313

Table 3: F_1 scores for each discourse acts in selected baseline models and proposed models.

Model	Precision	Recall	F_1 score
Rule 5-ACTS	0.836	0.439	0.575
CRF Vote (Zhang et al., 2017)	0.794	0.676	0.730
LSTM-CRF Vote	0.721	0.689	0.704
Proposed (Parent-Branch Tree-LSTM)	0.722	0.703	0.713
Proposed (Tree-LSTM GCN Hybrid)	0.750	0.693	0.720

Table 4: Precisions, recalls, and F_1 scores of *Appreciation* in selected baseline models and proposed models.

These acts are less frequent than others (Table 1), and the improvements imply the strength of the neural models to capture infrequent characteristics.

The strength of the hybrid approach (Tree-LSTM GCN Hybrid) is observed in *Announcement* with an improvement of 3.2 in F_1 score. A unique characteristic of *Announcement* is that it only appears in initial comments (Table 1). Therefore, the strength of the hybrid approach supports the observation in Section 5.1 that Child-Sum Tree-LSTM and GCN are more preferred in initial comments. One exception where the proposed model did not work well is *Appreciation*. Table 4 shows the detailed evaluation values of *Appreciation*. In terms of precision, the non-neural models (Rule 5-ACTS and CRF Vote) performed better than the neural models. This analysis suggests that non-neural approaches have certain strength for classifying discourse acts that have strong linguistic clues (e.g. thank).

6 Conclusion

As described in this paper, we proposed a model that integrates discussion structures with neural networks to analyze discourse acts. Parent-Branch Tree-LSTM, Child-Sum Tree-LSTM, and GCN are used as components of the proposed model to capture discussion structures. Results show that, by combining the three components with a self-attentive process, improvements of 1.5% in accuracy and 2.2 in F_1 score are achieved compared to the previous best model. Interactions among the three components are also explored via analysis of attention probabilities in the self-attentive layer of the model. As future works of this study, we first plan to apply our model to discussion in different domains. Reddit comments have their strength in data size, but they are known to have strong association with non-textual attributes such as timing and authors. We are planning to expand our models to explore a more flexible architecture for analyzing discussions.

Acknowledgements

We would like to thank the members of Okumura–Takamura Group at Tokyo Institute of Technology for having fruitful discussions about tree learning approaches and graph learning approaches using neural networks. We would also like to thank the anonymous reviewer for their comments to improve this paper.

References

Sumit Bhatia, Prakhar Biyani, and Prasenjit Mitra. 2012. Classifying user messages for managing web forum data. In *Proceedings of the 15th International workshop on the Web and Databases*.

Harry Bunt, Jan Alexandersson, Jean Carletta, Jae-Woong Choe, Alex Chengyu Fang, Koiti Hasida, Kiyong Lee, Volha Petukhova, Andrei Popescu-Belis, Laurent Romary, Claudia Soria, and David Traum. 2010. Towards an ISO standard for dialogue act annotation. In *Proceedings of the Seventh conference on International Language Resources and Evaluation*.

Cody Buntain and Jennifer Golbeck. 2014. Identifying social roles in Reddit using network structure. In *Proceedings of the Workshop on Modeling Social Media: Mining Big Data in Social Media and the Web*, pages 615–620.

Vitor R. Carvalho and William W. Cohen. 2005. On the collective classification of email "speech act". In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 345–352.

Vitor Carvalho and William Cohen. 2006. Improving "email speech acts" analysis via n-gram selection. In *Proceedings of the Analyzing Conversations in Text and Speech*, pages 35–41.

Hao Cheng, Hao Fang, and Mari Ostendorf. 2017. A factored neural network model for characterizing online discussions in vector space. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2296–2306.

William W. Cohen, Vitor R. Carvalho, and Tom M. Mitchell. 2004. Learning to classify email into "speech acts". In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 309–316.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29*, pages 3844–3852. Curran Associates, Inc.

Oliver Ferschke, Iryna Gurevych, and Yevgen Chebotar. 2012. Behind the article: Recognizing dialog acts in Wikipedia talk pages. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 777–786.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. 2018. Before name-calling: Dynamics and triggers of ad hominem fallacies in web argumentation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 386–396.

David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150.

Ji He, Mari Ostendorf, Xiaodong He, Jianshu Chen, Jianfeng Gao, Lihong Li, and Li Deng. 2016. Deep reinforcement learning with a combinatorial action space for predicting popular Reddit threads. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1838–1848.

Ji He, Mari Ostendorf, and Xiaodong He. 2017. Reinforcement learning with external knowledge and two-stage Q-functions for predicting popular Reddit threads. *arXiv preprint arXiv:1704.06217*.

Jack Hessel, Lillian Lee, and David Mimno. 2017. Cats and captions vs. creators and the clock: Comparing multimodal content to context in predicting relative popularity. In *Proceedings of the 26th International Conference on World Wide Web*, pages 927–936.

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

Jun Hu, Rebecca Passonneau, and Owen Rambow. 2009. Contrasting the interaction structure of an email and a telephone corpus: A machine learning approach to annotation of dialogue function units. In *Proceedings of the SIGDIAL 2009 Conference*, pages 357–366.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Aaron Jaech, Victoria Zayats, Hao Fang, Mari Ostendorf, and Hannaneh Hajishirzi. 2015. Talking to the crowd: What do people react to in online discussions? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2026–2031.

Su Nam Kim, Lawrence Cavedon, and Timothy Baldwin. 2010a. Classifying dialogue acts in one-on-one live chats. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 862–871.

Su Nam Kim, Li Wang, and Timothy Baldwin. 2010b. Tagging and linking web forum posts. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 192–202.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270.

Wern Han Lim, Mark James Carman, and Sze-Meng Jojo Wong. 2017. Estimating relative user expertise for content quality prediction on Reddit. In *Proceedings of the 28th ACM Conference on Hypertext and Social Media*, pages 55–64.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *Proceedings of the 5th International Conference on Learning Representations*.

Fei Liu, Timothy Baldwin, and Trevor Cohn. 2017. Capturing long-range contextual dependencies with memory-enhanced conditional random fields. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, pages 555–565.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1064–1074.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Adinoyi Omuya, Vinodkumar Prabhakaran, and Owen Rambow. 2013. Improving the quality of minority class identification in dialog act tagging. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 802–807.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 129–136.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca A. Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–373.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1556–1566.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of the 25th International Conference on World Wide Web*, pages 613–624.

Yi-Chia Wang, Mahesh Joshi, and Carolyn Rose. 2007. A feature based approach to leveraging context for classifying newsgroup style discussion segments. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 73–76.

Li Wang, Marco Lui, Su Nam Kim, Joakim Nivre, and Timothy Baldwin. 2011. Predicting thread discourse structure over technical web forums. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 13–25.

Zhongyu Wei, Yang Liu, and Yi Li. 2016. Is this post persuasive? Ranking argumentative comments in online forum. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 195–200.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Victoria Zayats and Mari Ostendorf. 2018. Conversation modeling on Reddit using a graph-structured LSTM. *Transactions of the Association for Computational Linguistics*, 6:121–132.

Renxian Zhang, Dehong Gao, and Wenjie Li. 2011. What are tweeters doing: Recognizing speech acts in Twitter. In *Proceedings of the AAAI-11 Workshop on Analyzing Microtext*, pages 86–91.

Renxian Zhang, Dehong Gao, and Wenjie Li. 2012. Towards scalable speech act recognition in Twitter: Tackling insufficient training data. In *Proceedings of the Workshop on Semantic Analysis in Social Media*, pages 18–27.

Amy Zhang, Bryan Culbertson, and Praveen Paritosh. 2017. Characterizing online discussion using coarse discourse sequences. In *Proceedings of the 11th AAAI International Conference on Web and Social Media*, pages 357–366.

AnlamVer: Semantic Model Evaluation Dataset for Turkish - Word Similarity and Relatedness

Gökhan Ercan
Department of Computer Engineering
Işık University, İstanbul, Turkey
gokhan.ercan@isik.edu.tr

Olcay Taner Yıldız
Department of Computer Engineering
Işık University, İstanbul, Turkey
olcaytaner@isikun.edu.tr

Abstract

In this paper, we present AnlamVer, which is a semantic model evaluation dataset for Turkish designed to evaluate word similarity and word relatedness tasks while discriminating those two relations from each other. Our dataset consists of 500 word-pairs annotated by 12 human subjects, and each pair has two distinct scores for similarity and relatedness. Word-pairs are selected to enable the evaluation of distributional semantic models by multiple attributes of words and word-pair relations such as frequency, morphology, concreteness and relation types (e.g., synonymy, antonymy). Our aim is to provide insights to semantic model researchers by evaluating models in multiple attributes. We balance dataset word-pairs by their frequencies to evaluate the robustness of semantic models concerning out-of-vocabulary and rare words problems, which are caused by the rich derivational and inflectional morphology of the Turkish language.

Title and Abstract in Turkish

AnlamVer: Anlambilimsel Model Ölçümleme Veri Kümesi - Kelime Benzerliği ve İlişkiselliği

Bu makalede, Türkçe için kelime benzerlik ve ilişkisellik görevlerini ayrı ayrı ölçümleyebilmek için tasarlanmış anlambilimsel bir model veri kümesi olan AnlamVer'i sunuyoruz. Veri kümemiz, 12 kişi tarafından her kelime çifti için ilişkisellik ve benzerlik puanları ayrı ayrı puanlanmış toplam 500 kelime çiftinden oluşmaktadır. Kelime çiftleri, dağılımsal anlambilimsel modelleri kelimelerin ve kelime çiftlerinin sıklık, morfoloji, somutluk ve ilişki tipleri (eş anlamlılık, karşıt anlamlılık vb.) gibi birden fazla niteliğine göre ölçümleyebilmek için seçilmiştir. Amacımız, anlambilimsel model araştırmacılarının modellerini birden fazla niteliğe göre ölçümleyerek içgörüler kazanmasını sağlamaktır. Veri kümesindeki kelime çiftleri, Türkçe'nin zengin türetimsel ve çekimsel yapısı kaynaklı sözlük-dışı ve seyrek-kelime problemlerine karşı anlambilimsel modellerin sağlamlığını ölçümleyebilmek amacıyla sıklık değerlerine göre dengelenmiştir.

1 Introduction

Unsupervised semantic modeling has recently gained a lot of attention in the NLP community due to the notion of high reusability of pre-trained models across a variety of higher level NLP tasks such as machine translation, word sense disambiguation and named entity recognition. Increasing computability of unsupervised distributional semantic modeling (DSM) methods enable researchers to increase the performance of NLP tasks by leveraging extracted semantic information from a high volume of unstructured texts at low costs. However, there are very few available methods and resources to evaluate semantic models intrinsically regardless of the higher level tasks' dynamics. Presenting word similarity and word relatedness (i.e., association) dataset AnlamVer, we aim at providing the semantic modeling field for Turkish with an intrinsic evaluation resource targeting morphology driven issues caused by the rich agglutinative nature of the language. We are not aware of the existence of such word similarity or

Proceedings of the 27th International Conference on Computational Linguistics, pages 3819–3836
Santa Fe, New Mexico, USA, August 20-26, 2018.

relatedness evaluation resources constructed for Turkish. In this paper, we describe design considerations and data collection guidelines we followed in the construction of such dataset as well as dataset statistics.

The main contributions of this paper include: (i) an introduction of a first word similarity and word relatedness evaluation dataset for a low-resource language Turkish[1], (ii) design considerations on the construction of a dataset where the main objective is balancing the words and the words-pairs by multiple morphological and semantic attributes, (iii) a novel analysis and visualization of a word similarity and relatedness dataset containing bi-dimensional values for each word-pair and, (iv) a publicly available web-based word similarity questionnaire software.[2]

2 Background and Design Motivations

Word similarity evaluation (i.e., wordsim) is one of the oldest intrinsic methods of semantic model assessment. For example, RG dataset Rubenstein and Goodenough (1965) is still being used today as one of the gold standards in the DSM research. Wordsim datasets are constructed by asking human subjects to assign numeric scores from 0 to 10 for every pre-selected word-pairs. In this section, we describe some issues of word similarity evaluation and design decisions we made to overcome such issues in our study.

2.1 Similarity and Relatedness Confusion

Linguistic Background

Linguists have been studying on statistical distributions of linguistic items (words) for a century. Although the distributional hypothesis *"words that occur in similar contexts, tend to have similar meanings"* is commonly traced back to Harris (1954), according to Sahlgren (2006), theoretical foundations of his distributional methodology go back to structuralist linguist Bloomfield (1887 - 1949) and Ferdinand de Saussure (1857 - 1913). De Saussure et al. (2011) pointed out that there can be distinctive functional roles of *signs* within the language system. He defined functional differences of linguistic elements in two (orthogonal) types which are widely studied with distributional relations in distributional semantics (DS) research today: *syntagmatic* and *paradigmatic*. Briefly, "words have a syntagmatic relation if they co-occur, and a paradigmatic relation if they share same neighbors" (Sahlgren, 2006). Paradigmatic words represent similar concepts or entities of the real world which are most likely substitutional in the context. One example is the synonyms like "clever" and "smart" in the sentence "She is very [clever | smart]." where two words are less likely to occur at the same sentence.

Lack of Distinction in Word Evaluation

Following the theoretical distinction of paradigmatic and syntagmatic relation types, one can easily apply such distinction to word evaluation by making the assumption that "similarity represents paradigmatic and relatedness represents syntagmatic type of relations.". However, semantic research has not paid as much attention to this distinction as necessary. Two most comprehensive DSM benchmark studies, Baroni et al. (2014) and Levy et al. (2015), reported model performances based on *wordsim* datasets such as RG (Rubenstein and Goodenough, 1965), WordSim-353 (Finkelstein et al., 2001) and MEN (Bruni et al., 2012). In their study, Hill et al. (2016) thoroughly describe the distinction between similarity and relatedness (i.e., sim-rel) caused limitation problems of such datasets. Hill et al. (2016) also define the criteria for evaluation datasets in three: representative, *clearly-defined*, consistent and reliable. Most wordsim datasets such as RG, MC (Miller and Charles, 1991), WordSim-353, and MEN do not satisfy clearly-defined criteria since their screen guidelines use both "similarity" and "relatedness", and "association" words in place of each others. One good example for guideline ambiguity is the following instructions from the WordSim-353 study: *"...please assign a numerical similarity score between 0 and 10 (0 = words are totally unrelated), 10 = words are VERY closely related..."*. Since our study aims to collect both similarity and relatedness scores from participants, we provided *clearly-defined* detailed instructions in the questionnaire screens (Figure 2).

[1]Dataset is publicly available at `http://www.gokhanercan.com/anlamver`

[2]Source code and a demo application are publicly available at `http://www.gokhanercan.com/wsquest`

One Model Does Not Fit All

Agirre et al. (2009) detected the aforementioned sim-rel confusion and split the original WordSim-353 dataset into two datasets (WS-Rel and WS-Sim) by classifying word-pairs based on their relationship types. Thus, they solved the dataset's sim-rel distinction problem without re-scoring word-pairs.[3] They proposed two separate models for similarity and relatedness evaluation tasks. For example, they reported that the context-windows-based approach is better at capturing similarity (evaluated on WS-Sim) while the bag-of-words approach is at relatedness (evaluated on WS-Rel). Capturing the similarity seems arguably harder for *the distributional hypothesis* based unsupervised models compared to relatedness models. Examining the DSM benchmark study of Levy et al. (2015), average performances of all model configurations consistently perform the worst on the SimLex-999 similarity dataset (≈ 0.39) compared to relatedness (traditional wordsims) (≈ 0.70) datasets.[3] [4]

Similarly, Hill et al. (2016) focus only on similarity evaluation while clearly informing annotators about the sim-rel distinction in their SimLex-999 dataset work. In another dataset study, SimVerb-3500 (Gerz et al., 2016), only distributional verb semantics with a large scale (3,500 word-pairs) of verb similarity evaluation is considered. We observe that DSMs have been starting to be divided into more specific models (e.g., relatedness, similarity, antonymy), motivated by the better performance requirements of the higher-level tasks. As Faruqui et al. (2016) point out, intrinsic wordsim evaluation does not correlate well with extrinsic NLP tasks' evaluation results. Wordsim sim-rel confusion might be one of the reasons for this inconsistency. It is an open question whether a single pre-trained DSM can represent the semantics of a domain consistently across multiple higher-level tasks. We think that a *perfect* DSM would be a multi-model structure which could handle every specific relation types (e.g., relatedness, similarity, antonymy, hypernymy, meronymy) of the words with maximum performances. In this study, our dataset targets Turkish language specific DSMs with two main types of semantic relations (similarity and relatedness) by evaluating word-pairs on both at the same time. We are not aware of the existence of such dataset study for any language.

Sim-Rel Vector Space

Instead of splitting word-pairs into two distinct groups, we decided to get two scores for every word-pair: similarity and relatedness. By doing so, we can keep the dataset as a single unit while evaluating semantic models in two relation types. Two-dimensional structure of our evaluation data structure allows us to visualize the semantic space of the dataset through a scatter plot diagram we named "Sim-Rel vector space" (Figure 1).

Given x and y axes represent relatedness r and similarity s scores of each word-pair in the dataset respectively, and variables r and s (orthogonality of paradigmatic and syntagmatic relations) range from 0 and 10. Let SU similar-unrelated, SR similar-related, DU dissimilar-unrelated, DR dissimilar-related are categorical labels of possible semantic sub-spaces ss, $ss = f_1(r, s)$ function would be,

$$ss = f_1(r, s) = \begin{cases} \text{SU}, & \text{if } s \geq 5 \text{ and } r < 5 \\ \text{SR}, & \text{if } s \geq 5 \text{ and } r \geq 5 \\ \text{DU}, & \text{if } s < 5 \text{ and } r < 5 \\ \text{DR}, & \text{if } s < 5 \text{ and } r \geq 5 \end{cases}$$

And let $t = 2$ denotes a threshold variable that represents boundary point of relation-type-spaces where $synonym, antonym, irrelevant$ are categorical labels of possible semantic relation-types rt, $rt = f_2(r, s)$ function would be,

$$rt = f_2(r, s) = \begin{cases} \text{synonym}, & \text{if } 10 - t \leq s \text{ and } 10 - t \leq r \\ \text{antonym}, & \text{if } 10 - t \leq r \text{ and } s \leq t \\ \text{irrelevant}, & \text{if } t \geq r \text{ and } t \geq s \end{cases}$$

[3] Since participants scored under ambiguous guidelines, WS-Sim is not inherently a similarity dataset. See (Hill et al., 2016).

[4] Low scores on RW dataset are plausible because it targets OOV and rare words problems.

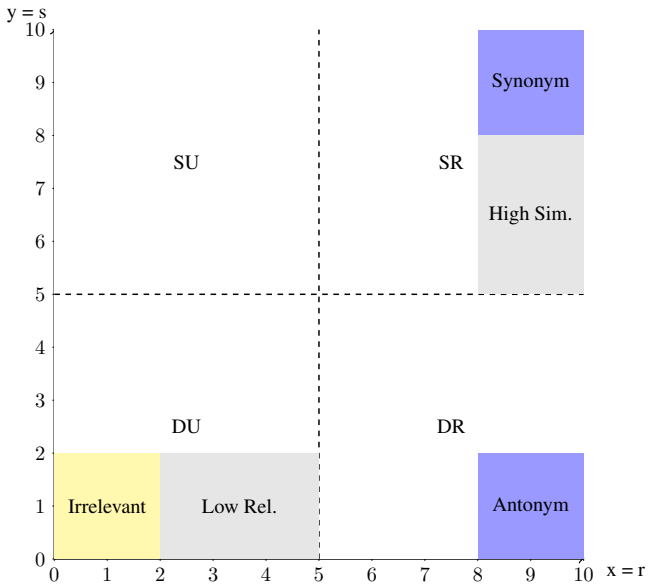


Figure 1: Sim-Rel vector space of word-pairs.

If we assume that participants are asked to score lower for similarity s (closer to 0) in the case of their antonym judgements for word-pairs, and asked to score higher (closer to 10) in the case of their synonym judgements[5], following the definition of the Sim-Rel vector space functions f_1 and f_2 above, followings can be inferred:

- A perfect DSM could assign word-pairs to every semantic sub-space ss with 100% accuracy.
- No word-pair instance is expected to be assigned to a similar-unrelated SU sub-space. Semantically, all highly similar word-pairs should also be highly related. For example, "car - automobile" word-pair is highly similar. Since they are very likely to share many common neighbors in their contexts, their relatedness score should be high, too.
- Word-pairs could be accepted as synonyms if their rt value is assigned to $synonym$ varying by the t parameter. The same rule applies to the $irrelevant$ value, too. We intuitively chose the threshold $t = 2$ value for Sim-Rel semantic space. We kept $t = 2$ same for all axes and relations for the sake of model and visualization simplicity. We leave the theoretical or empirical investigation of selection strategies of such threshold values for future work.
- Antonym-DR-overlapping problem: No DSM could perfectly assign rt as an $antonym$. Boundaries between the antonyms and dissimilar-related DR word-pairs are semantically ambiguous. Our bi-dimensional evaluation model cannot differentiate between the two. For example, word-pairs "tense - loose" and "red - rose" could get closer r and s scores while the first one can semantically be an antonym but the latter is obviously not. Asking to score lower for antonym judgements is a common method in *similarity* datasets (Hill et al., 2016) (Gerz et al., 2016). Storing two inherently different relation types' scores (synonym, antonym) in a single s variable is the root cause of the problem. It is the downside of our Sim-Rel vector space model. We leave the resolution of this problem for future work.

2.2 Out-Of-Vocabulary and Rare Words Problems

Turkish is an agglutinative language which has a highly inflectional and derivational morphology. The agglutinative nature allows forming new words by stringing stem, morphemes and suffixes together. In

[5]Scoring closer numbers to zero for antonyms is a common convention for similarity datasets (Hill et al., 2016), (Gerz et al., 2016).

Turkish, words are bound-morphemes, which means that there can be only one lexical stem (root) of a word. Since Turkish has many productional affixes (e.g., CHk, CA, CI, lHK, SHz, HmsH), theoretically unlimited surface words can be generated. Table 1 shows inflections and derivations of various words in morphologically decomposed forms where every word share the same lexeme "maymun" (monkey). In this study, all morphological decompositions are performed by the toolkit from Görgün and Yildiz (2011).

Word	Decomposition	Sense	Form	Frequency
maymun	maymun	monkey	root form	very
maymunları	maymun + lAr + sH	their monkeys	inflectional	medium
maymunsu	maymun + sI	ape, like monkeys	derivational (usual deviance)	rare
maymungilleri	maymun + gil + lAr + yH	family of monkeys, primades	derivational (acceptable deviance)	oov
maymuncuk	maymun + CHk	skeleton key, picklock (tool)	derivational (deviant)	rare

Table 1: Morphological decomposition of various words sharing the same lexeme.

Simple word-based models ignore the internal structure of words which reduces the model's capabilities and qualities. The main problem is zero (i.e., unseen, out-of-vocabulary) or low occurrence (i.e., rare words) of a testing word in the training corpus. Distributional semantics (DS) community has been developing more complex subword-level (i.e., compositional) models to overcome *out-of-vocabulary* (OOV) and *rare words* problems. DSMs for morphologically-rich languages must alleviate OOV and rare words problem to generalize better. RW dataset (Luong et al., 2013) provides word frequency (i.e., rareness) based evaluation strategy to compositional model developers. Similarly, we aim to balance our dataset's word-pool by words' frequencies to assess generalization powers of such models.

In addition to the traditional OOV and rare words evaluation strategy, we applied another concept that we named *made-up words* by injecting *novel* (i.e., made-up, fake) words into the word-pool of our dataset. Vecchi et al. (2017) applied this concept to their phrase-level models to test the model's creative capabilities (i.e., generalization power). The main idea is as follows: even if people hear a word for the first time and it might sound odd to them, people have the intuition to make sense of the intended meaning. Could DSMs do the same? In our subword-level case, we assume that Turkish affixes can change the meanings of the words in a consistent manner, which is called *acceptable semantic deviance*. For example, the word "maymungilleri" (family of monkeys) is a made-up word and it sounds odd to any native Turkish speaker (Table 1) in the first place. But almost every native speaker can understand what it's meant to some extent. This productivity feature of a language can be seen as a substantial model generalization potential for a researcher. However, the downside of the concept is that the assumption does not always hold as in the word "maymuncuk" (skeleton key, a tool) (Table 1). In this example, the word is derived from the root word (i.e., lexeme) "maymun" (monkey) by getting the affix "cHk" with a valid state transition but its *one sense's* meaning shifts to an entirely different space. It is a very challenging problem for compositional DSMs. Vecchi et al. (2017) name this type of semantically-lossy derivations as *deviants*.

2.3 Dataset Translation Issues

Before starting the dataset construction phase, we have considered translating the existing well-known wordsim datasets to Turkish. After completing MC dataset's translation, we conclude that constructing a new dataset would be more meaningful and reliable than translating the existing ones. The main reasons behind our decision can be summarized as follows:

1. Both words in the source word-pair map to the same single word in the target language: *"football - soccer"* $\rightarrow$ *"futbol - futbol"*.

2. A word in the source word-pair maps to a multi-word phrase: *"asylum - madhouse"* $\rightarrow$ *"tımarhane - akıl hastanesi"*. Traditional wordsim datasets and DSMs ignore phrases for the sake of model and evaluation simplicity. We left phrases out of the scope of this study.

3. Meaning shifts in translations require re-annotation from human resources for every word-pair. The human annotation stage is one of the costly parts of the study. [6]

4. We aim to balance words and word-pairs in as many attributes as possible such as word frequency, derivations, inflections, concreteness and relation types. Word frequencies and morphological features are pretty much language-dependent.

3 Dataset Design and Methodology

Design motivations we borrowed from the previous section can be summarized as follows: (i) collecting two-dimensional relatedness and similarity scores from participants while clearly-defining distinctions of such concepts, (ii) making it language-specific morphological dataset which can evaluate DSMs' generalization power concerning OOV, rare words and semantic deviance scenarios, and (iii) balancing the dataset by multiple morphological and semantic attributes as much as possible. Due to the time and budget limitations, we set the target dataset size of the project to 1.000 scores (500 word-pairs) as most of the wordsim datasets include fewer scores (SimLex-999=999, RG=65, M30, WordSim-353=353, RW=2,034, MEN=3,000). We planned dataset construction process in three stages: word candidates, word pool and word-pairs selections (Table 2).

	Stage 1	Stage 2	Stage 3
	1) Word Candidates (starts)	2) Word-Pool Selection	3) Word-Pairs Selection
Goals	1.1) Reusing existing resources	2.1) Balancing word attributes by estimations	3.3) Balancing word-pairs by estimations
Input	1.2) TKN (600) + MC (39)	2.2) Stage1 output (639) + new derivational words (99)	3.2) 320 Stage2 words
Process	1.3) Attaching frequencies, morphological tags	2.3) Filtering for balancing	3.3) Mapping pairs (every word used 2-5 times building word-pairs)
Output	1.4) 639 words	2.4) 320 words	3.4) 500 word-pairs (ends)

Table 2: Three stages of dataset construction.

3.1 Word Candidates Selections

TKN Dataset

Since Turkish is a low-resource language in NLP research, we aimed to re-use existing resources as much as possible. We investigated word candidates that already have some informative attributes. We used word norms "Türkçe Kelime Normları" (TKN) dataset (Tekcan and Göz, 2005) which is constructed for a psycholinguistics study. TKN consists of 600 Turkish words which are balanced in terms of *concreteness* (half concrete, half abstract) values. TKN's concreteness values are annotated by 100 voluntary university students. As in the English USF word norms dataset (Nelson et al., 2004), TKN words range between 1 and 7. Lower values denote more abstract and higher values denote more concrete concepts. For instance, the word "mutluluk" (happiness) takes 1.85 while the word "gül" (rose) 6.79. By choosing candidates from TKN, we enabled model developers to evaluate their models on various concreteness levels. Unfortunately, TKN contains very frequent and root-formed word dataset with 480 words in lexical root form where none of the 120 non-root form words are inflected.[7] Those limitations led us to add 99 words (with no concreteness values) manually on the next stage in order to achieve the balancing goal of the dataset.

[6] Considering human resources, questionnaire software and data pre/post-processing costs.

[7] 108 words having one derivations, 12 words having two derivations.

3.2 Word-pool Selections

Having 600 candidate words that are transferred from the first stage, our goal was to narrow down them to 320 words (word-pair candidates) while preserving our dataset balancing requirements. Table 3 shows word-pool grouping attributes along with the number of words and percentages.

Frequency-based Balancing

Considering the importance of OOV and rare words problems in modeling for morphologically rich and productive languages, our priority was to balance our word-pool based on word frequencies. RareWords dataset (Luong et al., 2013) addresses this issue by grouping words by their frequencies into four groups $(5-10, 10-100, 100-1000, 1000-10000)$. Since the RareWords dataset is designed for the English language (which is relatively less inflectional and productive than Turkish), researchers may assume that words with frequencies lower than five are most likely to be junk or non-English words. In our case, a single Turkish lexicon can take thousands of surface forms. Our own corpus coverage analysis shows that 47% of word types (277K) occur only once in the corpus, which is compatible with Boun Corpus word coverage statistics (Sak et al., 2011). Therefore, we couldn't ignore the words that have zero or less than five frequencies. We applied a different grouping strategy, where the first group is OOV (zero frequency) and rare words in five groups $RW1, RW2, RW3, RW4, RW5$. Table 3 displays how OOV and rare words (RW) groups are represented in the word-pool. We made the frequency analysis on Boun Corpus (Sak et al., 2011) which consists of roundly 3.2 million token types (i.e., vocabulary size). We defined frequency groups $(0-32, 32-320, 320-3200, 3200-32000, 32000-\infty)$ by using the $gr(n, voc, g)$ function below, where g is the number of groupings, n is the index of each group varying between 1 to g, and voc is the vocabulary size of the given corpus. The only exception is the minimum and maximum values of the first and last groups are fixed to 0 and ∞ respectively. Ampersand symbols (&) denotes string concatenations:

$$gr(n, voc, g) = (voc \times 10^{-(g-n+3)}) \text{ \& "-" \& } (voc \times 10^{-(g-n+2)})$$

3.3 Word Pair Selections

In the word-pair selection stage, we matched words from word-pool with each other to form new pairs. We defined a constraint that every word in the word-pool should occur in the matching word-pairs up to five times. The primary goal of this mapping stage was to find, 500 word-pair relations, which are balanced explicitly by new type attributes: estimated semantics relations are synonym, antonym, hypernym, meronym etc. We manually matched and estimated the semantic type of the relationships. For example, we manually picked two potentially similar words "otomobil" (automobile) and "araba" (car) from the pool and flagged them as a strong synonym potential. We defined 50 synonym, 50 antonym, 50 meronym, 50 hypernym relations. Similarly, we grouped word-pairs by estimated magnitudes (low, medium, high) of relatedness relations too. Table 4 shows the number of actual instances and percentages of such estimation-based groupings and morphological groupings of word-pairs. Finally, we ended up with 500 manually-selected, grouped word-pairs. Table 5 shows some sample annotated word-pair instances from the final dataset.

4 Questionnaire Design

4.1 Platform

We built a web-based application to collect data from human annotators. Participants were asked to score similarity and relatedness relationship for every 500 word-pairs. In total, every participant scored 1,000 answers for 500 word-pairs. We split the questionnaire into two sections. The first section starts with describing what similarity relation is, along with five examples with detailed descriptions. Similar to the Simlex-999 guidelines, we asked users to score low for antonyms and score high for synonyms. We described similarity as follows (showing first two sentences only):
"Two words are similar if they refer to the same thing, person, concept or action. Similar things share common concrete or abstract attributes. For example, 'tea' and 'coffee' are quite similar because both

	G0	G1	G2	G3	G4	G5	Total
Frequency	OOV	RW1	RW2	RW3	RW4	RW5	
	31 9.6%	33 10.3%	30 9.3%	62 19.3%	111 34.6%	53 16.5%	320 100%
Concreteness	no value	abstract	medium	concrete			
	149 46.5%	35 10.9%	30 9.3%	106 33.1%			320 100%
Root Form	root	non-root					
	182 56.8%	138 43.1%					320 100%
Derivations	no der.	der1	der2+				
	198 61%	81 25.3%	41 12.8%				320 100%
Inflections	no inf.	inf1	inf2+				
	277 86.5%	17 5.3%	26 8.1%				320 100%

Table 3: Groupings of the word-pool.

	G0	G1	G2	G3	G4	G5	Total
Est. Synonyms	synonym	antonym	other				
	50 10%	50 10%	400 80%				500 100%
Est. Relatedness	high	medium	low				
	200 40%	150 30%	150 30%				500 100%
Est. Rel. Type	hyponym	meronym	other				
	50 10%	50 10%	400 80%				500 100%
OOV	no oov	any oov	two oov				
	434 86.8%	66 13.2%	42 8.4%				500 100%
Min. Derivations	no der.	der1	der2+				
	231 46.2%	166 33.2%	103 20.6%				500 100%
Min. Inflections	no inf	inf1	inf2+				
	424 84.8%	32 6.4%	44 8.8%				500 100%
Min. RareWord	rw0 (oov)	rw1	rw2	rw3	rw4	rw5	
	66 13.2%	65 13%	62 12.4%	130 26%	142 28.4%	35 7%	500 100%

Table 4: Groupings of the word-pairs.

are relaxing hot drinks gathered from nature and irreplaceable beverages for friendly conversations." Figure 2 shows a snapshot from the initial guideline screen for similarity annotation. When a participant presses the "ileri" (next) button, first word-pair page appears, asking to score 20 word-pairs per screen (Figure 3).

4.2 Participants

All 12 participants are native Turkish speakers who voluntarily participated in the questionnaire. Eight participants are female, and four participants are male. Both mean and median values of ages are 33.5

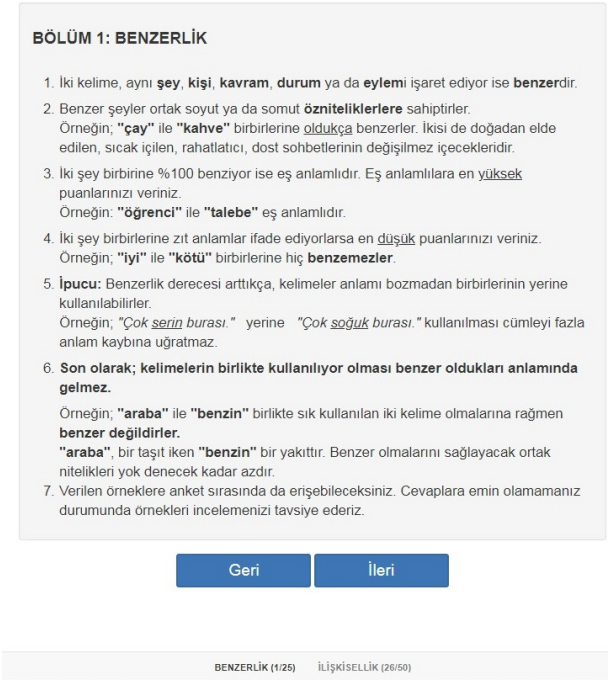


Figure 2: Similarity instructions page.

Figure 3: Word-pair annotation page.

with the standard deviation of 9.3. Nine participants are university graduates (seven participants with the master's degree), two participants are undergraduate, and a participant holds a high school degree. Participants were asked to join the questionnaire remotely using their web browsers by following the invitation link sent to their mailboxes. Questionnaire software WSQuest's[8] responsive layout support, allowed users to score quickly from mobile and tablet devices. They are asked to read user guidelines carefully since no participant had the prior knowledge about the flow of the annotation process, and word similarity and relatedness concepts. Initial guideline screen informed users that they could score the questionnaire freely at any time of the day in three days since questionnaire software allows users to continue their sessions easily as long as they keep the last completed URL of the software. Without giving any breaks, it took participants' 75 minutes on average to complete the entire questionnaire.

5 Dataset Analysis

Final (actual) similarity s and relatedness r values of the dataset seems consistent with our estimations. Under the aforementioned Sim-Rel vector space model assumptions and configuration, scatter plot of the average values of s and r yielded a visual similar to our expectations (Figure 4). Our observations about the actual data distribution as follows:

- SU subspace remain empty as expected. Participants proved that word-pairs cannot be similar and unrelated at the same time.
- Antonym-DR-overlapping problem holds. We observed very close word-pair values for antonym and DR word-pairs. For example, average s and r scores of "kırmızı - gül" (red - rose) word-pair are 1.16 and 7.16 respectively. On the other hand, an antonym-estimated word-pair "şeffaf - opak" (transparent - opaque) has exactly the same scores as the former one (see Table 5).
- Participants scored word-pairs that include made-up words normally as regular word-pairs. For example, annotators scored the word-pair "atatürkist - kemalci" (atatürk+İST - kemal+CH) as 8.75

[8]See appendices or `http://www.gokhanercan.com/wsquest` for complete user screen guidelines.

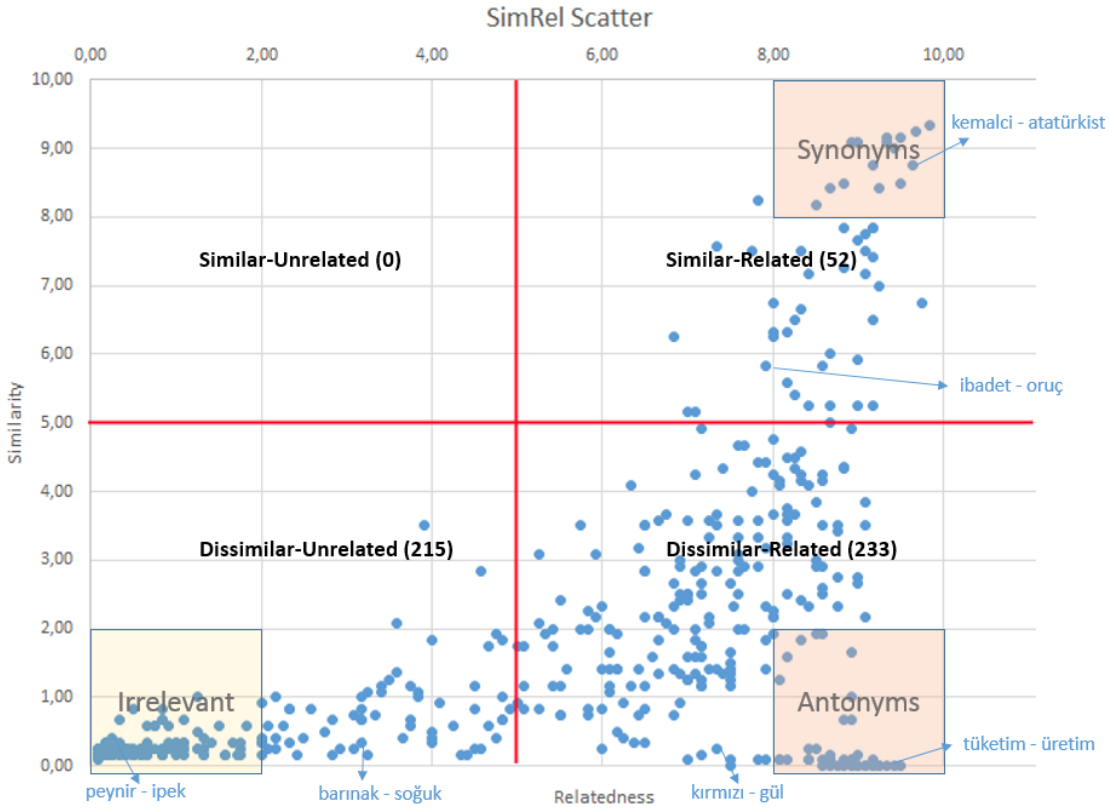


Figure 4: Scatter plot of the final (actual) dataset. Data-points denote participants' avg. Sim-Rel score of each word-pair where y axis is s and x axis is r. Member counts of $ss\{SU, SR, SU, DR\}$ semantic sub-spaces are in the parenthesis.

similar and 9.63 related (see Table 5) where neither of surface forms has the common usage in Turkish (OOV and RW1 respectively). Both İST and CH suffixes have usages to change words' meaning to "ideological adherence to a person/thing" where both names "Atatürk" and "Kemal" are parts of the name "Mustafa Kemal Atatürk" who is the founder of the Republic of Turkey.

5.1 Post-processing and Inter-annotator Argument

Since questionnaire includes many uncommon OOV and rare words pairs, it allows users to skip that word-pair empty if they don't have any idea about the meanings. However, null answering rate (0.1%) is quite lower than we expected. In order to calculate ranking correlations properly, we replaced null answers with the average score of all users' answers for that question. Among 16 participants, we do some post analysis on collected data. We detected that one participant achieved marginally low (0.32 min, 0.57 max) Spearman ranking correlation score compared to the other participants. After a little further investigation, we noticed that this participant had completed the test only in 25 minutes. It is three times faster than what we estimated for a high-quality annotation. Similarly, we increased the overall data quality by removing three more participant's answers too. After post-processing, we computed 0.748 average pairwise inter-annotator ($apia$) score where the highest pairwise correlation of users is 0.847, and the lowest one is 0.474. Even though dataset's $apia$ score is lower than we expected, 0.748 is still higher than the most word similarity datasets (WS-Sim=0.667, MEN=0.68, 0.67=SimLex-999). Based on the study of (Snow et al., 2008), more than ten annotators are statistically acceptable for word similarity evaluation task's reliability.

word1	word2	sim.	rel.	oov	conc.	ss	der#	inf#
otomobil (automobile)	araba (car)	9.16	9.33	no	6.87	SR (syn.)	0	0
üşengen* (lazy)	üşengeç (lazy)	8.25	7.83	one	3.06	SR	2	0
atatürkist* (adh .to Atatürk)	kemalci (adh. to Kemal)	8.75	9.63	one	-	SR	2	0
kitaplıklar (bookshelves)	kitaphane* (place with books)	7.16	8.41	no	-	SR	2	1
kemalci (adh. to Kemal)	kemalizmcilerden (from ...)	5.25	8.66	one	-	SR	3	3
kırmızı (red)	gül (rose)	1.16	7.16	no	6.79	DR	0	0
şeffaf (transparent)	opak (opaque)	1.16	7.16	no	4.37	DR	0	0
zarar (loss)	kazanç (profit)	0.18	8.8	no	3.25	DR (ant.)	0	0
gevşek (loose)	heykel (statue)	0.16	0.16	no	-	DU (irr.)	0	0
üşengen* (lazy)	yedigen (heptagon)	0.16	0.25	two	-	DU	2	0

Table 5: Sample word-pairs from the final dataset. Words with asterisks (*) are made-up words. (adh = adherent, conc = concreteness, ss = semantic sub-space, syn = synonyms, ant = antonyms, irr = irrelevant, der# = total derivations, inf# = total inflections)

6 Conclusion

We presented a semantic model evaluation dataset for the Turkish language. Turkish morphology requires complex semantic models to alleviate OOV and rare words problems. Since the dataset includes 13% OOV and 26% rare-word-pairs (RW1 and RW2), we think that it will be a challenging intrinsic evaluation task for DSM researchers. Hopefully, AnlamVer-evaluated distinct similarity and relatedness models correlate better with the higher level NLP tasks. For future work, we are planning to construct a bigger dataset, leveraging existing lexical resources such as Turkish WordNet (Ehsani et al., 2018) which already includes manually-annotated synonymy (i.e., synsets), antonymy and hypernymy relations.

Acknowledgements

This work was supported by Tübitak project 116E104.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*, pages 238–247.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

Ferdinand De Saussure, Wade Baskin, and Perry Meisel. 2011. *Course in general linguistics*. Columbia University Press.

Razieh Ehsani, Ercan Solak, and Olcay Taner Yildiz. 2018. Constructing a wordnet for turkish using manual and automatic annotation. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 17(3):24.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. *arXiv preprint arXiv:1605.02276*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simverb-3500: A large-scale evaluation set of verb similarity. *arXiv preprint arXiv:1608.00869*.

Onur Görgün and Olcay Taner Yildiz. 2011. A novel approach to morphological disambiguation for turkish. In *Computer and Information Sciences II*, pages 77–83. Springer.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.

Douglas L Nelson, Cathy L McEvoy, and Thomas A Schreiber. 2004. The university of south florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers*, 36(3):402–407.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Magnus Sahlgren. 2006. *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. Ph.D. thesis, Institutionen för lingvistik.

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2011. Resources for turkish morphological processing. *Language resources and evaluation*, 45(2):249–261.

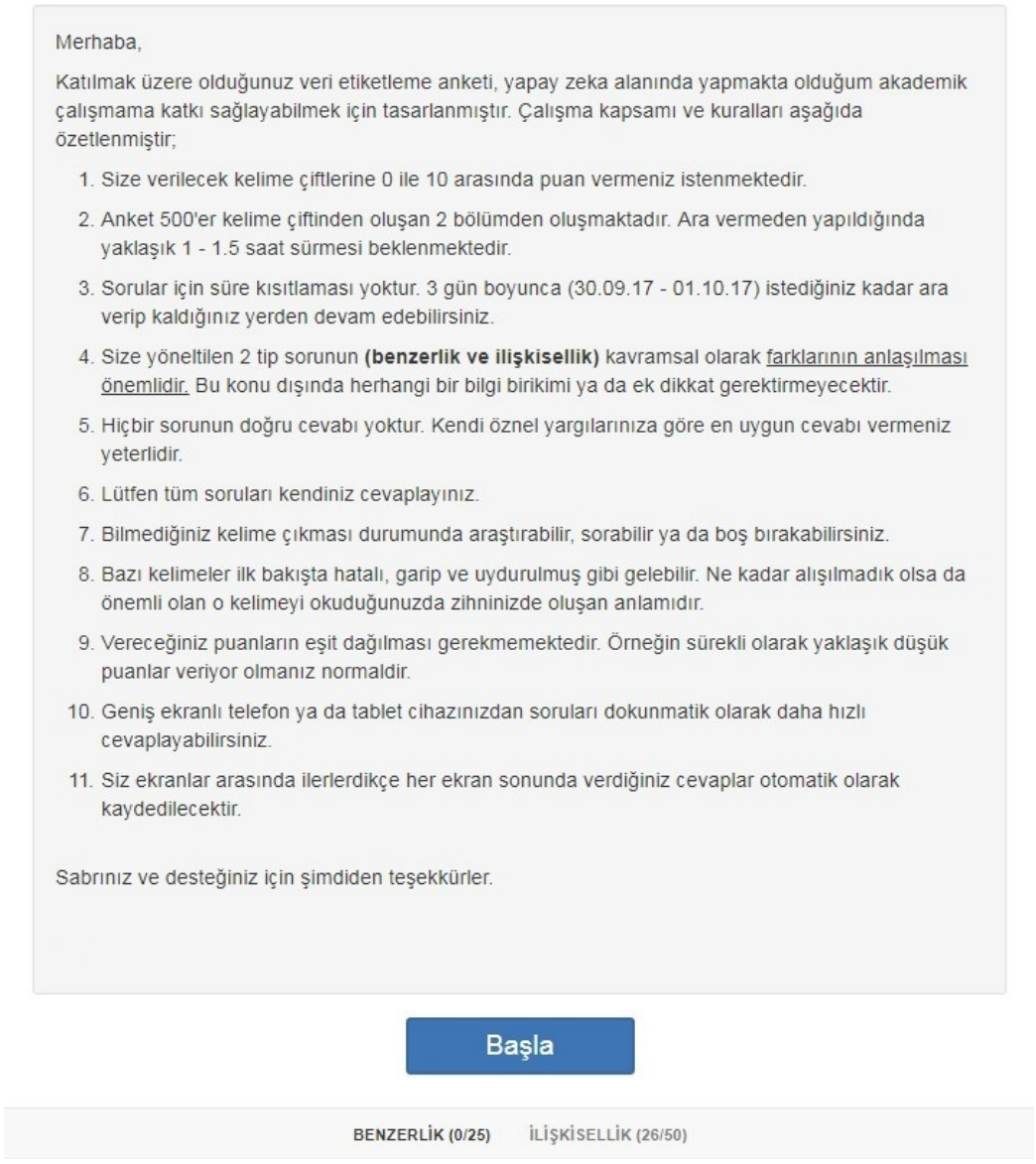
Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263. Association for Computational Linguistics.

Ali İ Tekcan and İlyas Göz. 2005. Türkçe kelime normları. *İstanbul Boğaziçi Üniversitesi*.

Eva M Vecchi, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2017. Spicy adjectives and nominal donkeys: Capturing semantic deviance using compositionality in distributional spaces. *Cognitive science*, 41(1):102–136.

.

Appendices


Merhaba,
Katılmak üzere olduğunuz veri etiketleme anketi, yapay zeka alanında yapmakta olduğum akademik çalışmama katkı sağlayabilmek için tasarlanmıştır. Çalışma kapsamı ve kuralları aşağıda özetlenmiştir;
1. Size verilecek kelime çiftlerine 0 ile 10 arasında puan vermeniz istenmektedir.
2. Anket 500'er kelime çiftinden oluşan 2 bölümden oluşmaktadır. Ara vermeden yapıldığında yaklaşık 1 - 1.5 saat sürmesi beklenmektedir.
3. Sorular için süre kısıtlaması yoktur. 3 gün boyunca (30.09.17 - 01.10.17) istediğiniz kadar ara verip kaldığınız yerden devam edebilirsiniz.
4. Size yöneltilen 2 tip sorunun (benzerlik ve ilişkisellik) kavramsal olarak farklarının anlaşılması önemlidir. Bu konu dışında herhangi bir bilgi birikimi ya da ek dikkat gerektirmeyecektir.
5. Hiçbir sorunun doğru cevabı yoktur. Kendi öznel yargılarınıza göre en uygun cevabı vermeniz yeterlidir.
6. Lütfen tüm soruları kendiniz cevaplayınız.
7. Bilmediğiniz kelime çıkması durumunda araştırabilir, sorabilir ya da boş bırakabilirsiniz.
8. Bazı kelimeler ilk bakışta hatalı, garip ve uydurulmuş gibi gelebilir. Ne kadar alışılmadık olsa da önemli olan o kelimeyi okuduğunuzda zihninizde oluşan anlamıdır.
9. Vereceğiniz puanların eşit dağılması gerekmemektedir. Örneğin sürekli olarak yaklaşık düşük puanlar veriyor olmanız normaldir.
10. Geniş ekranlı telefon ya da tablet cihazınızdan soruları dokunmatik olarak daha hızlı cevaplayabilirsiniz.
11. Siz ekranlar arasında ilerlerdikçe her ekran sonunda verdiğiniz cevaplar otomatik olarak kaydedilecektir.
Sabrınız ve desteğiniz için şimdiden teşekkürler.
Başla
BENZERLİK (0/25)
İLİŞKİSELLİK (26/50)

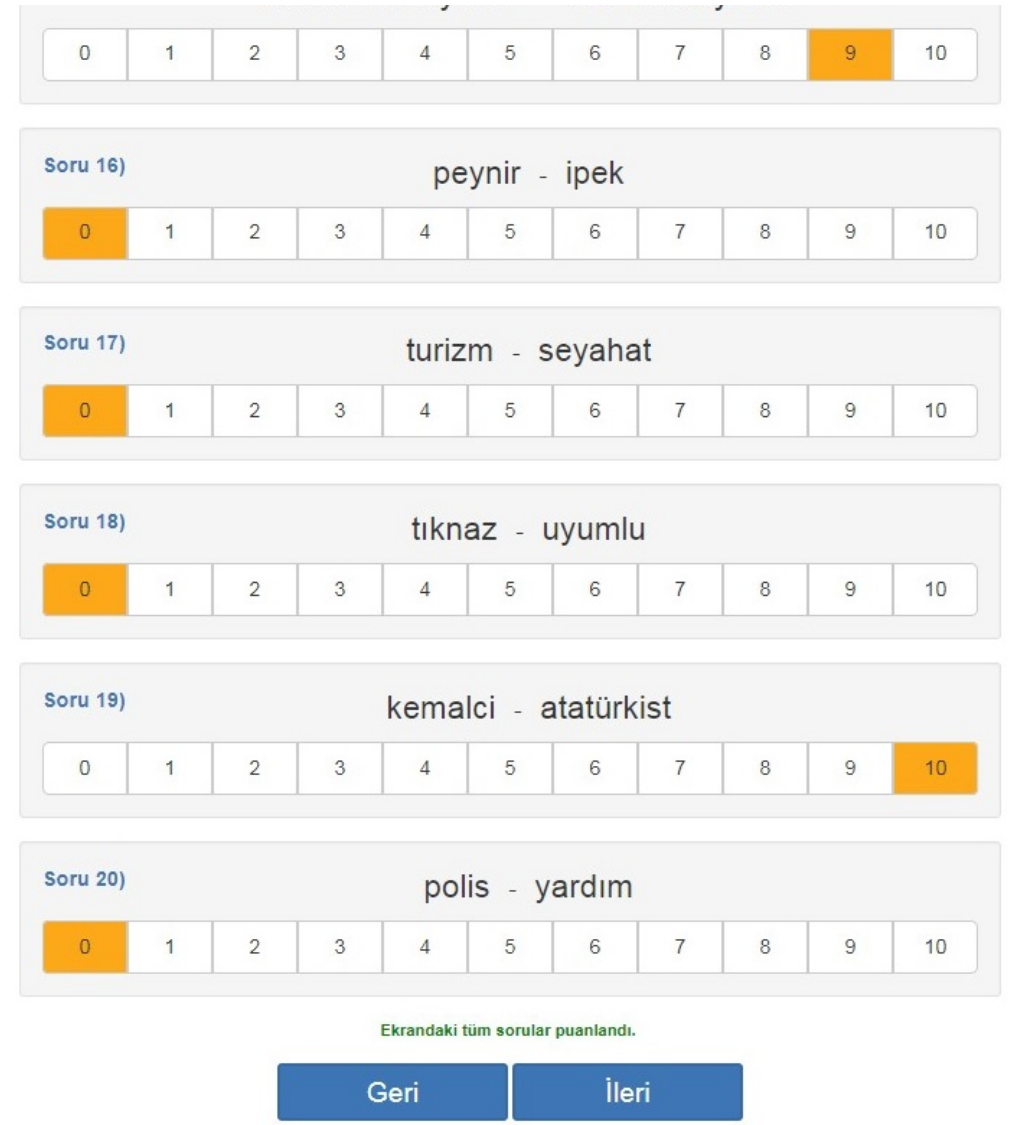
BÖLÜM 1: BENZERLİK

1. İki kelime, aynı **şey**, **kişi**, **kavram**, **durum** ya da **eylem**i işaret ediyor ise **benzer**dir.
2. Benzer şeyler ortak soyut ya da somut **özniteliklerlere** sahiptirler.
   Örneğin; **"çay"** ile **"kahve"** birbirlerine oldukça benzerler. İkisi de doğadan elde edilen, sıcak içilen, rahatlatıcı, dost sohbetlerinin değişilmez içecekleridir.
3. İki şey birbirine %100 benziyor ise eş anlamlıdır. Eş anlamlılara en yüksek puanlarınızı veriniz.
   Örneğin: **"öğrenci"** ile **"talebe"** eş anlamlıdır.
4. İki şey birbirlerine zıt anlamlar ifade ediyorlarsa en düşük puanlarınızı veriniz.
   Örneğin; **"iyi"** ile **"kötü"** birbirlerine hiç **benzemezler**.
5. **İpucu:** Benzerlik derecesi arttıkça, kelimeler anlamı bozmadan birbirlerinin yerine kullanılabilirler.
   Örneğin; *"Çok serin burası."* yerine *"Çok soğuk burası."* kullanılması cümleyi fazla anlam kaybına uğratmaz.
6. **Son olarak; kelimelerin birlikte kullanılıyor olması benzer oldukları anlamında gelmez.**

   Örneğin; **"araba"** ile **"benzin"** birlikte sık kullanılan iki kelime olmalarına rağmen **benzer değildirler.**
   "araba", bir taşıt iken **"benzin"** bir yakıttır. Benzer olmalarını sağlayacak ortak nitelikleri yok denecek kadar azdır.
7. Verilen örneklere anket sırasında da erişebileceksiniz. Cevaplara emin olamamanız durumunda örnekleri incelemenizi tavsiye ederiz.

Geri İleri

BENZERLİK (1/25) İLİŞKİSELLİK (26/50)


0 1 2 3 4 5 6 7 8 9 10
Soru 16)
peynir - ipek
0 1 2 3 4 5 6 7 8 9 10
Soru 17)
turizm - seyahat
0 1 2 3 4 5 6 7 8 9 10
Soru 18)
tıknaz - uyumlu
0 1 2 3 4 5 6 7 8 9 10
Soru 19)
kemalci - atatürkist
0 1 2 3 4 5 6 7 8 9 10
Soru 20)
polis - yardım
0 1 2 3 4 5 6 7 8 9 10
Ekrandaki tüm sorular puanlandı.
Geri
İleri
BENZERLİK (1/25)
İLİŞKİSELLİK (26/50)

BÖLÜM 2: İLİŞKİSELLİK

1. Bu bölümde aynı kelime çiftlerini **ilişkisellik** derecesi bakımından değerlendirmeniz beklenmektedir.
2. İlişkisellik bir önceki bölümdeki benzerliğe oranla çok daha kolay belirlenebilmektedir.
3. Yüksek ilişkili kelimeler birbirleri ile alakalıdırlar ve sıklıkla benzer bağlamlar içinde kullanılırlar. Örneğin; **"benzin"** ile **"araba"** kelimeleri benzer bağlamlar içinde kullanıldıklarından oldukça ilişkilidirler.
4. Kelimelerin yüksek ilişkili olabilmesi için ortak özniteliklerinin olmasına ihtiyaç yoktur. Örneğin; **"kahve"** ve **"fincan"** çiftinde, biri içecek diğeri eşya olmasına rağmen çift oldukça ilişkilidir ve hatta birbirlerini hatırlatırlar. Bununla beraber; **"faiz"** ve **"fincan"** kelimelerinin ilişkileri oldukça azdır.
5. Benzer ve zıt anlamlı kelimelerin ilişki seviyeleri de genellikle yüksektir. Örneğin; *"İyi kötü ve çirkin."* cümlesinden anlaşılacağı gibi "iyi" ve "kötü" benzer bağlamlarda kullanılırlar. **"iyi"** ve **"kötü"** oldukça ilişkilidirler. Aynı şekilde; benzer anlamlı **"öğrenci"** ve **"talebe"** kelimeleri de benzer bağlamlarda sık geçerler ve oldukça ilişkilidirler.
6. Verilen örneklere anket sırasında da erişebileceksiniz. Cevaplara emin olamamanız durumunda örnekleri incelemenizi tavsiye ederiz.

Geri İleri

BENZERLİK (25/25) **İLİŞKİSELLİK (26/50)**

İLİŞKİSELLİK: Yüksek ilişkili kelimeler birbirleri ile alakalıdırlar ve sıklıkla bir arada kullanılır, birbirlerini hatırlatırlar. ☑ Otomatik olarak alttaki soruya geç

- "benzin" ve "araba" kelimeleri sıklıkla benzer bağlamlarda geçerler ve oldukça ilişkilidirler.
- "kahve" ve "fincan" kelimeleri birbirlerini hatırlatırlar, oldukça ilişkilidirler.
- "faiz" ve "fincan" kelimeleri oldukça ilişkisizdirler.
- "iyi" ve "kötü" gibi zıt anlamlı kelimeler oldukça ilişkilidirler.
- "öğrenci" ve "talebe" gibi yüksek benzerlikte kelimeler genellikle aynı bağlamlarda geçerler ve oldukça ilişkilidirler.

Soru 501) mızrap - barınak

0	1	2	3	4	5	6	7	8	9	10

Soru 502) kırmızı - gül

0	1	2	3	4	5	6	7	8	9	10

Soru 503) suçlu - şüphe

0	1	2	3	4	5	6	7	8	9	10

Soru 504) laikçiler - sekülerizmciler

0	1	2	3	4	5	6	7	8	9	10

Soru 505) bitki - zeytin

0	1	2	3	4	5	6	7	8	9	10

BENZERLİK (25/25) **İLİŞKİSELLİK (26/50)**

Anket başarıyla tamamlandı. Pencereyi kapatabilirsiniz.

Değerli vaktinizi ayırdığınız için sonsuz teşekkürler.

Sevgiler,

Geri

BENZERLİK (25/25) **İLİŞKİSELLİK (51/50)**

Arguments and Adjuncts in Universal Dependencies

Adam Przepiórkowski
Institute of Philosophy
University of Warsaw *and*
Institute of Computer Science
Polish Academy of Sciences
ul. Jana Kazimierza 5
01-248 Warszawa, Poland
adamp@ipipan.waw.pl

Agnieszka Patejuk
Institute of Computer Science
Polish Academy of Sciences
ul. Jana Kazimierza 5
01-248 Warszawa, Poland
aep@ipipan.waw.pl

Abstract

The aim of this paper is to argue for a coherent Universal Dependencies approach to the core vs. non-core distinction. We demonstrate inconsistencies in the current version 2 of UD in this respect – mostly resulting from the preservation of the argument–adjunct dichotomy despite the declared avoidance of this distinction – and propose a relatively conservative modification of UD that is free from these problems.

1 Introduction

Universal Dependencies (UD; Nivre et al. 2016) has recently become a *de facto* standard for the representation of dependency treebanks. An important feature of UD is that it attempts *not* to represent the argument–adjunct distinction. This is stated explicitly in the paper introducing UD (Nivre et al. 2016: 1661): "The scheme [...] makes a distinction between core arguments (e.g., subject and object) and other dependents, but does not attempt to distinguish complements vs. adjuncts" (recall that *complements* are non-subject arguments), and it is confirmed on the current official UD webpage:[1] "[The UD taxonomy] does not make a distinction between adjuncts (general modifiers) versus oblique arguments (arguments said to be selected by a head but not expressed as a core argument)." This decision is not justified in Nivre et al. 2016, but it is briefly motivated on the UD webpage (cf. fn. 1): "[T]he argument/adjunct distinction is subtle, unclear, and frequently argued over. For instance, syntacticians at certain times have argued for various obliques to be arguments, while at other times arguing that they are adjuncts, particularly for certain semantic roles such as oblique instruments or sources. We take the distinction to be sufficiently subtle (and its existence as a categorical distinction sufficiently questionable) that the best practical solution is to eliminate it."

This view goes against the theoretical-linguistic received wisdom – linguistic theories invariably assume a fundamental argument–adjunct dichotomy (AAD). However, it has been repeatedly observed that AAD has been flawed ever since its conception: the three criteria for distinguishing arguments from adjuncts given by Tesnière (1959) – the father of modern valency theory – are pairwise incompatible (Vater 1978a, 1978b, Przepiórkowski 1999). Multiple tests proposed since then have often been inconsistent with linguists' intuitions about AAD (and with each other) and short-lived; as cautiously noted by Tutunjian and Boland (2008, 633), "[t]he sheer number of [purported argument/adjunct] tests underlines the fact that no single test is entirely satisfactory". The existence of such a universal dichotomy is also called into doubt by typologists (see, e.g., Croft 2001: 272–280 and Haspelmath 2014). We have argued recently that even those theories which put much emphasis on the argument–adjunct distinction have not proposed any operational way to distinguish these two classes (Przepiórkowski 2016a, 2017b). On the constructive side, we have demonstrated that a relatively modest modification of some assumptions of Lexical Functional Grammar (LFG; Bresnan 1982, Dalrymple 2001, Bresnan et al. 2015) results in a theory that does not assume AAD at any level of representation and in any theory-internal mechanisms

[1] http://universaldependencies.org/u/overview/syntax.html; all web pages cited here were last accessed on 12 March 2018.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3837–3852
Santa Fe, New Mexico, USA, August 20-26, 2018.

(Przepiórkowski 2016b, 2017a; Patejuk and Przepiórkowski 2016). This means that neither syntax nor compositional semantics necessarily presuppose AAD. Hence, given that AAD is both non-operational and unnecessary, the avoidance of AAD is a very attractive feature of UD, one that should be consistently implemented and preserved.

Unfortunately, UD does not consistently follow the policy expressed in the quotes evoked at the beginning of this section. The main aim of this paper is to identify those aspects of UD which do preserve AAD and to propose a modified version of the current standard which consistently eschews this ill-defined dichotomy. In particular, we demonstrate that the presence of some vestiges of AAD in UD results in serious inconsistencies in the current classification of dependents as core or non-core – a dichotomy which is fundamental for UD.

2 Argument–Adjunct Dichotomy in UD

The declared UD avoidance of any reference to AAD is consistently implemented in the case of broadly nominal (that is, also prepositional) dependents of verbs. For example, in the case of English, nominal subjects (`nsubj`) may be defined as those dependents which agree with verbs, nominal objects (`obj`) – as those bare nominal (NP) dependents which normally occur immediately after the verb, and nominal indirect objects (`iobj`) – as those NP dependents which normally occur immediately after the direct object. The actual UD definitions of `nsubj`, `obj` and `iobj` are different and somewhat internally inconsistent (cf. §4.2), but the point remains that no reference to AAD is made here. All other broadly nominal non-predicative dependents – including prepositional (PP) dependents – are classified as obliques (`obl`). This way the phrase *at the restaurant*, when it is a dependent of a (non-copula) verb, is always marked as oblique, whether it is an argument (e.g., *I looked at the restaurant*) or an adjunct (e.g., *I fell at the restaurant*) according to the advocates of AAD. Hence, there is no argument–adjunct dichotomy lurking in the process of deciding whether a typical broadly nominal dependent of a verb is core or non-core. Below we will show that the situation is very different in the case of clausal dependents of verbs – both 'closed' (usually finite) and 'open' (controlled, often infinitival) – and in the case of secondary predicates.

2.1 Closed Clausal Dependents

Ordinary (non-controlled) subordinate clauses which are dependents of verbs are marked in UD either as `csubj`, if they are subjects, or as `ccomp`, if they are not subjects but should still be classified as core, or as `advcl`, if they are non-core. Let us assume that it is possible to recognise `csubj` dependents (e.g., in English, as immediately pre-verbal clauses), but how should other clausal dependents be split into core (i.e., `ccomp`) and non-core (i.e., `advcl`)? Current guidelines do not make this clear, saying the following about `ccomp`: "A clausal *complement* of a verb or adjective is a dependent clause which is a core argument. That is, it functions *like an object* of the verb, or adjective"[2] (emphasis ours). The first sentence, making reference to the notion of *complement* (i.e., non-subject argument), explicitly relies on AAD: complement clauses are marked as `ccomp`, adjunct clauses – as `advcl`. The second sentence, making reference to the notion of *object*, could in principle avoid AAD, if there were independent criteria of deciding when a clausal dependent "functions like an object". Since no such criteria are provided in UD guidelines, UD practitioners implicitly rely on AAD and classify dependent clauses as `ccomp` when they seem to be (non-subject) arguments, and as `advcl` – when they seem to be adjuncts.

It might seem then that both sentences have the same effect, but in fact they differ. On the first – broader – view, complement clauses are treated as core, but they are not necessarily assumed to be objects. That is, it is possible for a single head to have an `obj` dependent and a `ccomp` dependent. On the second – narrower – view, complement clauses are (direct) objects, so – given that a head may have at most one (direct) object dependent – it is not possible for a single head to have both an `obj` dependent and a `ccomp` dependent. In fact, such a check is implemented in the UD validating script, UDAPY, and the accompanying validation web page states this explicitly: "No predicate can have more than one direct object. Ccomp counts as direct object."[3]

[2] `http://universaldependencies.org/u/dep/ccomp.html`
[3] `http://universaldependencies.org/svalidation.html`

Both approaches are found in UD (release 2.1) treebanks. The broader view is implemented, *inter alia*, in the Latvian LVTB treebank and in the English EWT treebank (the latter henceforth abbreviated to $\text{UD}^{\text{EN}}_{\text{EWT}}$), where in both cases about 1% of utterances contain a head with an `obj` and a `ccomp` dependent; one of the 160 such cases in $\text{UD}^{\text{EN}}_{\text{EWT}}$ is:

(1) Apparently the City *informed* [them]$_{\texttt{obj}}$ [that no permit is required]$_{\texttt{ccomp}}$.

On the other hand, some treebanks – including the Finnish TDT treebank and the Polish SZ treebank ($\text{UD}^{\text{PL}}_{\text{SZ}}$, for short) – take the stronger position: if two dependents compete for the status of the direct object, only one of them may be labelled as `obj` or `ccomp`, and the other one is given the status of indirect object, `iobj`. As indirect objects are assumed to be nominal, never clausal, `ccomp` always wins the competition for the direct object status, so the nominal is marked as `iobj`. But this has very unwelcome consequences, as it forces some most prototypical direct objects to be reanalysed as indirect objects only because the verb also subcategorises for a clause. This is illustrated with example (2) from $\text{UD}^{\text{PL}}_{\text{SZ}}$, where the verb *spytało* 'asked' combines with the numeral subject *kilka osób* 'several people', the accusative nominal *mnie* 'me' and the subordinate clause *czy jestem...*, lit. 'whether I am...'.

(2) Kilka osób *spytało* [mnie]$_{\texttt{iobj}}$, [czy jestem dzięki feminizmowi szczęśliwsza]$_{\texttt{ccomp}}$.
several people asked me.ACC whether am thanks feminism.DAT happier.NOM
'Some people have asked me whether feminism made me happier.'

Since the subordinate clause is marked as `ccomp`, *mnie* 'me' must be (and is) marked as an indirect object, `iobj`, contrary to linguistically motivated definitions of direct objects in Polish. On the view prevailing in contemporary grammars of Polish, the only reasonable definition of direct objects in Polish is as those dependents which become subjects under passivisation – and *mnie* 'me' above satisfies this definition. A more traditional view refers to accusative bare nominal complements – and *mnie* 'me' satisfies this definition as well. In fact, when the same verb, SPYTAĆ 'ask', occurs in $\text{UD}^{\text{PL}}_{\text{SZ}}$ with an accusative complement but without a clausal complement, the accusative nominal is marked as `obj`, as in the following sentence:

(3) Chciał [ją]$_{\texttt{obj}}$ *spytać* [o wiele rzeczy]$_{\texttt{obl}}$.
wanted her.ACC ask.INF about many things
'He wanted to ask her about many things.'

Note that *mnie* 'me' in (2) and *ją* 'her' in (3) bear exactly the same semantic role with respect to the two forms of the verb SPYTAĆ 'ask' and have the same grammatical properties (passivisability, grammatical case, etc.), so this is a clear case of intra-linguistic annotation inconsistency.

One line of defence could be that there are two valency frames for SPYTAĆ 'ask' witnessed in these two examples and that the accusative complement – while having the same semantic role – is assigned different grammatical functions in these frames. Consider, however, the attested[4] example (4), which involves a coordination of a nominal dependent with the preposition O 'about' and a clausal dependent with the complementiser CZY 'whether'. Such examples make it clear that the accusative phrases in (2)–(3) occupy the same position within the same valency frame, in which another position may be realised either as a subordinate clause, as in (2), or as a prepositional phrase with the preposition O 'about', as in (3), or as a coordination of such two realisations, as in (4).[5,6]

(4) ...należy więc *spytać* [lekarza]$_{?}$ [[o rekomendowane leczenie]$_{\texttt{obl}}$ i [czy
ought.IMPS so ask.INF doctor.ACC about recommended treatment and whether
można się spodziewać wypadania włosów]$_{\texttt{ccomp}}$].
may.IMPS RM expect falling_out hair
'...so one should ask the doctor about the recommended treatment and whether one should expect hair to fall out.'

Let us summarise. Two views on what it means to be a (non-subject) core clausal dependent are present

[4] `http://www.handsomemen.pl/wlosy,lysienie,wypadanie_wlosow_u_dzieci.html`

[5] Since the accusative dependent of SPYTAĆ 'ask' is claimed to be `iobj` in (2) and `obj` in (3), it is not clear – on the approach discussed here – what its label should be in (4); hence the question mark.

[6] Note that, unlike in the English translation, the second conjunct cannot be analysed as a dependent of the preposition *o* 'about'.

in UD guidelines. The broader view makes an explicit reference to AAD and defines ccomp as any non-subject clausal argument. The narrower view defines ccomp as a clausal object, but – given the lack of an independent definition of such clausal objects – this also boils down in practice to assuming AAD and treating all non-subject clausal arguments as ccomp. Moreover, the narrower view, as currently implemented, is linguistically naïve and leads to violations of annotation consistency. In §4, we propose a solution that may be viewed as a variant of the narrower view, but one that does not make reference to AAD and does not lead to annotation inconsistencies.

2.2 Open Clausal Dependents

While ccomp marks non-subject 'closed' clausal arguments, xcomp is used, *inter alia* (see §2.3 below), to label so-called 'open' non-subject core clauses, i.e., clausal dependents with obligatorily controlled subject, as in the English example from $\text{UD}^{\text{EN}}_{\text{EWT}}$: "So [Bush]$_{\text{nsubj}}$ *stopped* [flying]$_{\text{xcomp}}$." Just as in the case of ccomp, the xcomp relation is understood differently in different treebanks. In $\text{UD}^{\text{EN}}_{\text{EWT}}$, xcomp dependents are again understood broadly, as any open complements, and they do not compete with prototypical direct objects. For example, *asked* in the following (partial) sentence has an obj dependent and an xcomp dependent: "I *asked* [them]$_{\text{obj}}$ [to change it]$_{\text{xcomp}}$…". On the other hand, in $\text{UD}^{\text{PL}}_{\text{SZ}}$, the prototypical (passivisable, accusative) direct object *je* 'them' of the verb *uczą* 'teach' in (5) is marked as an indirect object, probably due to the presence of an xcomp dependent of the same verb, which wins the competition for the direct object position:

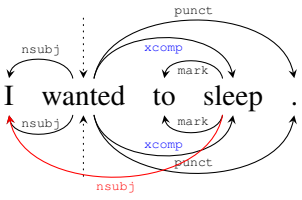
(5) …*uczą* [je]$_{\text{iobj}}$ [zjeżdżać na racicach]$_{\text{xcomp}}$.
teach.3.PL them.ACC slide.INF on hooves
'…they teach them to slide on their hooves.'

Again, in both cases reference is made to AAD: non-subject open dependents are marked as xcomp if they are arguments, and as advcl – if they are adjuncts. And, again, the narrower view goes against the linguistically-motivated definitions of *direct object* and results in annotation inconsistency, as illustrated by (6), where the corresponding (passivisable, accusative) dependent *nas* 'us' is marked as a direct object, obj, while the iobj label is used for the taught material, *tego* 'this' (which semantically corresponds to the xcomp dependent *zjeżdżać na racicach* 'slide on hooves' in (5)).

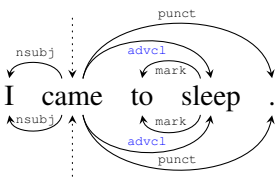
(6) [Tego]$_{\text{iobj}}$ [nas]$_{\text{obj}}$ *uczy* [Boże Słowo]$_{\text{nsubj}}$.
this.GEN us.ACC teach.3.SG god's.NOM word.NOM
'God's Word teaches us that.'

Note that, unlike in the case of 'closed' clausal dependents, a different label (xcomp vs. advcl here, ccomp vs. advcl there) is not the only difference between the two analyses. If an 'open' clausal dependent is analysed as core, i.e., as xcomp, there is an additional secondary nsubj edge in the enhanced representation, as in (7). Such a secondary edge is missing if the dependent is analysed as non-core, as in (8). But since core vs. non-core distinction relies on AAD, this means that the structure of the resulting dependency graph depends on the dichotomy that UD claims to eschew.

(7)



(8)



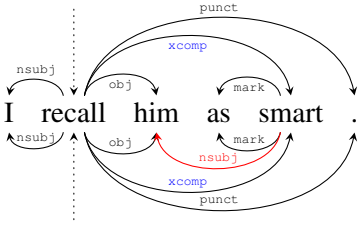
2.3 Secondary Predicates

Finally, AAD is also preserved in UD in the treatment of secondary predicates, as in the English "I recall him as smart and genial" (from the iWeb corpus, https://corpus.byu.edu/iweb/). Should the depictive dependent, *smart and genial*, be analysed as an argument of *recall*, or as its adjunct? Valency dictionaries do not agree about this (see Appendix A). As is made clear in the description of the xcomp label,[7] secondary predicates analysed as arguments are xcomp dependents of the verb, with an enhanced

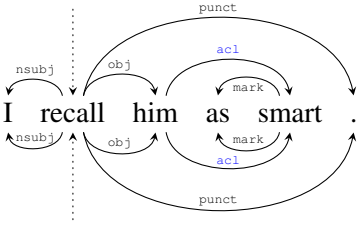
[7] http://universaldependencies.org/u/dep/xcomp.html

nsubj relation to the nominal they predicate of, as in (9). However, those analysed as adjuncts are acl dependents of the nominal elements they predicate of, as in (10).

(9)

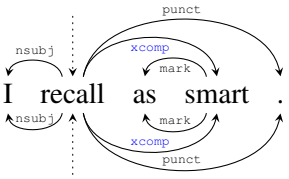


(10)

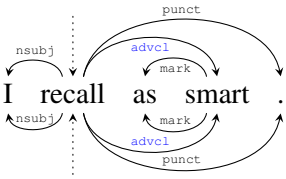


Moreover, if the subject of predication is missing or realised non-locally, as in "(This is the man) I recall as smart", the secondary predicate is still an xcomp dependent of the verb, if it is analysed as an argument (cf. (11)), but it is marked with a third dependency label, advcl, if it is analysed as an adjunct (cf. (12)). Hence, in the case of secondary predicates, the resulting structures depend on the ill-defined AAD even more than in the case of 'open' clausal dependents, again, despite the declared avoidance of this dichotomy in UD.

(11)



(12)



2.4 Summary of AAD in UD

While UD avows to forsake AAD, the core vs. non-core distinction boils down to the argument–adjunct dichotomy at least in the case of clausal dependents of verbs – both 'closed' (clausal, often finite) and 'open' (controlled, often infinitival) – and in the case of secondary predicates. While this dichotomy is vague and controversial, the resulting dependency representations differ sharply, depending on the classification of the relevant constituent as an argument or an adjunct. The following section shows that these vestiges of AAD in UD also make it impossible to coherently classify certain dependents as core or non-core.

3 Core vs. Non-Core Distinction in UD

The fact that non-subject core *nominal* dependents are understood very narrowly – as bare nominal direct objects and perhaps indirect objects – while non-subject core *clausal* dependents may be understood more broadly – as any clausal complements – leads to some serious inconsistencies.

Consider the verb ASK. In sentences such as "We asked the president about the hunt for Osama bin Laden", the prepositional phrase *about the hunt for Osama bin Laden* will receive the obl – i.e., non-core – dependency label (this is the annotation of such dependents of ASK in $\text{UD}^{\text{EN}}_{\text{EWT}}$). On the other hand, in the sentence "We asked the president whether Americans are safe at home", the subordinate interrogative clause, being an argument, is a core dependent, so it will be annotated as ccomp (again, this is the annotation of such subordinate clause dependents of ASK in $\text{UD}^{\text{EN}}_{\text{EWT}}$). If so, what is the coreness status of the coordinate dependent of *asked* in the following example from the iWeb corpus?

(13) At the White House, we *asked* the president [[whether Americans are safe at home]$_{\texttt{ccomp}}$ and [about the hunt for Osama bin Laden]$_{\texttt{obl}}$].

At the moment, UD's reply is inconsistent: the broadly nominal second conjunct of this coordinate structure is a non-core dependent of the verb, and the clausal first conjunct is a core dependent of the verb. On the natural assumption that whole dependents should be classified as core or non-core (see Appendix B), and that the coordinate structure *whether Americans are safe at home and about the hunt for Osama bin Laden* is such a dependent of *asked*, the coreness status of this dependent is incoherent.

It is easy to provide an arbitrary technical solution of this conceptual problem: the coreness of a coordinate dependent could be resolved by fiat to the coreness of the first conjunct. So, in (13), the coordinate structure would be ruled as core, given that the first conjunct is a ccomp – i.e., core – dependent

of *asked*. The arbitrariness of this technical solution is made clear by the fact that, by adopting it, the coreness of a coordinate structure depends on the order of conjuncts. Thus, in "We asked the president about the hunt for Osama bin Laden and whether Americans are safe at home", the coordinate dependent would be classified as non-core, because the first conjunct is now an `obl` dependent of *asked*.

This problem is not limited to dependents of ASK; similar examples may be found in corpora, which involve other heads and other kinds of coordinated dependents, including the following, again from iWeb:

(14) Dennis is an important voice of moderation who *worries* [[whether Colorado has adequate water reserves]$_{\texttt{ccomp}}$, and [about the future of family farms]$_{\texttt{obl}}$].

(15) Mr. Bernhardt *told* him [[about the "mediandisease" virus]$_{\texttt{obl}}$ and [that Adrian's blood analysis showed that he was virus free]$_{\texttt{ccomp}}$].

(16) We *counsel* them [[about the dangers of giving birth]$_{\texttt{obl}}$ and [that their baby might be born infected]$_{\texttt{ccomp}}$].

Neither is the problem limited to English; a Polish example of this kind is in fact given above – see (4). There, the prepositional oblique (non-core) dependent *o rekomendowane leczenie* 'about recommended treatment' of the verb *spytać* 'ask' is coordinated with the clausal (core) dependent *czy można się spodziewać wypadania włosów* 'whether one should expect hair to fall out'.

Similar inconsistency involves `xcomp` dependents: they are core, but they may fill the same position as non-core dependents, as in the following attested[8] example:

(17) Campbell made two visits to the home of domestic abuse victim Miss A, during which he *asked* her [[for a kiss]$_{\texttt{obl}}$ and [to go on a date with him]$_{\texttt{xcomp}}$].

Again, *for a kiss* would currently be classified as a non-core `obl` and *to go on a date with him* is a typical `xcomp` (also in $\text{UD}^{\text{EN}}_{\text{EWT}}$), i.e., a core dependent, so the status of the coordinate phrase as a core or non-core dependent of *asked* is inconsistent.

4 Proposal

4.1 Basic Idea

The basic idea follows directly from the discussion of the previous sections. Since there are no generally accepted objecthood criteria of clausal dependents of verbs, only those clausal dependents should be treated as objects which may be coordinated with nominal – i.e., uncontroversial – objects. Similarly, clausal dependents which may be coordinated with broadly nominal obliques – especially, prepositional phrases – should be classified as non-core. On this approach, the clausal dependent of the Polish verb SPYTAĆ 'ask', as in (2), where it is marked as `ccomp`, should be classified as non-core, because it may be coordinated with a prepositional oblique, as in (4). (Note that the obstacle to treating the passivisable accusative *mnie* 'me' in (2) as a direct object would then disappear.) Similarly, given coordination facts such as (13), the clausal dependent of the English ASK in constructions involving an NP object (cf. *the president* in (13) and in its variants discussed in the main text) would also be classified as non-core, as it may also be coordinated with a prepositional oblique. The same holds for appropriate 'closed' clausal dependents of verbs in (14)–(16). Also some 'open' dependents, hitherto marked as `xcomp`, would have to be classified as non-core, on the basis of data such as (17).

The question arises whether, on this approach, there are any clausal dependents left that would have to be classified as objects rather than as obliques. The answer is yes: it is clear that some clausal dependents must be analysed as direct objects. Take the following classic example (Sag et al. 1985: 165):

(18) Pat *remembered* [[the appointment]$_{\texttt{obj}}$ and [that it was important to be on time]$_{\texttt{ccomp}}$].

In the simpler "Pat remembered the appointment", the NP *the appointment* is an uncontroversial direct object. Hence, the coordinate phrase *the appointment and that it was important to be on time* in the above sentence must also be considered the direct object.[9] If so, it must be concluded that the direct object position of the verb REMEMBER may be realised as either an NP or a clause. But this in turn

[8] `https://www.liverpoolecho.co.uk/news/liverpool-news/police-officer-attempted-kiss-vulnerable-13615076`

[9] Sag et al. 1985 convincingly argue that unlike category coordination cannot be explained away as ellipsis.

implies that the subordinate clause in "Pat remembered that it was important to be on time" should also be considered a direct object. Note that this conclusion does not lead to undesired consequences of the kind discussed in §§2.1–2.2 above: the NP and the clause do not compete for the status of direct object, as only one of them (or a single coordinate phrase) may occur with REMEMBER: *"Pat remembered the appointment that it was important to be on time". Obviously, such constructions are not limited to linguistic papers, as the following examples from the iWeb corpus testify:

(19) I do *remember* [[a lot of wine bottles]$_{\texttt{obj}}$ and [that Felix spoke almost nothing]$_{\texttt{ccomp}}$].

(20) Well, my initial reaction was, I have to *assume* [[the worst]$_{\texttt{obj}}$ and [that he's really going to take his life]$_{\texttt{ccomp}}$].

(21) One guy who had a registered one really didn't *know* [[the law]$_{\texttt{obj}}$ and [that he had to register the gun in this state]$_{\texttt{ccomp}}$].

Neither are they limited to English, as illustrated by the following Polish example (from the National Corpus of Polish; `http://nkjp.pl/`; Przepiórkowski et al. 2011, 2012):

(22) Milena *widzi* [[dewastację]$_{\texttt{obj}}$ oraz [że to coś wyciekające paskudzi
Milena.NOM sees devastation.ACC and that this something oozing soils
dywan]$_{\texttt{ccomp}}$]...
carpet
'Milena sees the devastation and that the oozing thing is soiling the carpet...'

Again, the same reasoning may be applied to 'open' clausal dependents, `xcomp`, as the following attested examples illustrate:[10]

(23) She *wants* [[a career in virtual reality]$_{\texttt{obj}}$, and [to perhaps even found her own company]$_{\texttt{xcomp}}$].

(24) Homeless people *need* [[a work ethic]$_{\texttt{obj}}$ and [to be self-reliant]$_{\texttt{xcomp}}$], Bender said.

(25) He *remembers* [[the cocktail napkins]$_{\texttt{obj}}$, and [to serve the ladies first]$_{\texttt{xcomp}}$].

Exactly such constructions in Polish are analysed in Patejuk and Przepiórkowski 2014, on the basis of the classic (Kallas 1993: 123) example (26).

(26) *Chcę* [[pić]$_{\texttt{xcomp}}$ i [papierosa]$_{\texttt{obj}}$].
want drink.INF and cigarette.ACC
'I want to drink and (I want) a cigarette.'

Hence, according to the current proposal, all the examples cited in this section, (18)–(26), show that the relevant heads (marked in italics) may combine with clausal dependents "functioning like objects".

There are two immediate consequences of this basic proposal. First, since some obligatorily controlled – i.e., 'open' – clausal dependents (e.g., the infinitival dependent of some uses of ASK; cf. (17)) are now analysed as non-core, it would be useful to distinguish such 'open' oblique clauses from 'closed' oblique clauses, just as 'open' object-like clauses (`xcomp`) are distinguished from 'closed' object-like clauses (`ccomp`). For the time being we retain the label `advcl` for the 'closed' oblique clauses and we introduce the label `xadvcl` for 'open' oblique clauses. In Appendix C, we provide independent motivation for such 'open' obliques, and also for 'open' – obligatorily controlled – subjects, called `xsubj` for now. Thus, we propose to extend the repertoire of basic dependents of verbs from (27) to (28).

(27)

	subject	object	non-core
nominal	`nsubj`	`obj/iobj`	`obl`
clausal	`csubj`	`ccomp`	`advcl`
open		`xcomp`	

(28)

	subject	object	non-core
nominal	`nsubj`	`obj/iobj`	`obl`
clausal	`csubj`	`ccomp`	`advcl`
open	`xsubj`	`xcomp`	`xadvcl`

Second, while some clausal dependents retain their status as core (cf. (18)–(26)), many clausal dependents which are prototypical arguments according to the advocates of AAD will now be analysed as non-core. While this may be perceived as a disadvantage of the current proposal, it is in fact its advantage, as it increases the internal consistency of UD: the treatment of broadly nominal dependents is

[10] The first example is taken from `https://www.usatoday.com/story/sponsor-story/xq/2017/04/03/high-school-wants-revolutionize-learning-technology/99985812/`, the other – from the iWeb corpus.

fully analogous. There, only very few of traditional arguments are assigned the status of core dependents: subjects, bare nominal direct objects and perhaps bare nominal indirect objects. The whole class of prototypical – but prepositional – arguments, as in "*wait* [for somebody]$_{\texttt{obl}}$", "*rely* [on somebody]$_{\texttt{obl}}$", "*nibble* [at something]$_{\texttt{obl}}$", etc., is relegated to obliques, together with prototypical adjuncts.[11] We see no reason not to adopt the same solution in the verbal domain: apart from closed and open subjects (`csubj`, `xsubj`) and objects (`ccomp`, `xcomp`), all other clausal dependents should be relegated to the respective classes of 'clausal obliques', i.e., `advcl` and `xadvcl`. If this solution seems difficult to accept at first, it is only because of the unfortunate names UD assigns to such 'clausal obliques', names which suggest their necessarily adverbial character; in §4.3, we will propose a more adequate naming scheme.

Let us take stock of the proposal so far. Unlike in the current UD practice, we propose to take seriously the stance expressed in the UD guidelines that (non-subject) core clausal dependents "function like an object" and mark as `ccomp` and `xcomp` only those 'closed' and 'open' clausal dependents which may be coordinated with uncontroversial – nominal – objects. The other (non-subject) 'closed' and 'open' clausal dependents should be marked as `advcl` and `xadvcl` (but see §4.3 below). But apart from making precise the recommendation that (non-subject) core clausal dependents should "function like objects", this proposal has a number of other advantages. First, it removes explicit and implicit references to AAD in the treatment of clausal dependents of verbs. Second, it is free from inconsistency problems with the previous narrow understanding of core clausal dependents (nominal direct objects reanalysed as indirect objects because of the presence of a clausal dependent). Third, it is free from the problem of the incoherent coreness status of certain coordinate dependents.

4.2 Direct and Indirect Objects Revisited

The above subsection concentrated on obliques and on uncontroversial direct objects. What about indirect objects? Can clausal dependents be coordinated with indisputable indirect objects and, hence, be claimed to "function like indirect objects"? The answer depends on a particular understanding of the distinction between direct and indirect objects. The current UD practice follows the linguistic tradition: indirect objects are those NPs in double object constructions which bear the semantic role of a recipient or a benefactor. Thus, in (29), from $\text{UD}^{\text{EN}}_{\text{EWT}}$, the NP *you* immediately following the verb is the indirect object, and the next NP, *a voice mail*, is the direct object:

(29) I *left* [you]$_{\texttt{iobj}}$ [a voice mail]$_{\texttt{obj}}$ on Friday.

If so, there are probably no clausal dependents that should be analysed as indirect objects – with the possible exception of free relatives, it is difficult to imagine such a clausal dependent bearing the semantic role of a recipient or a benefactor. So, for example, in the attested[12] (30), the 'closed' clausal conjunct should be analysed as a direct object, because the nominal conjunct, *balance*, is marked as a direct object (and the preceding NP, *me*, is marked as an indirect object):

(30) Mother Nature *taught* [me]$_{\texttt{iobj}}$ [[balance]$_{\texttt{obj}}$, and [that regular pattern interrupts can negate radical change]$_{\texttt{ccomp}}$].

Similarly for the 'open' clausal conjunct in the following attested sentence (from Patejuk and Przepiórkowski 2014):

(31) My uncle said to hell with that and *taught* [me]$_{\texttt{iobj}}$ [[karate]$_{\texttt{obj}}$, and [to fire weapons]$_{\texttt{xcomp}}$].

However, as is often the case with linguistic tradition, this traditional view confuses different levels of linguistic representation, namely, semantic roles and (syntactic) grammatical functions. As discussed at length in Andrews 2007, grammatical functions correlate with semantic roles, but – in the end – the precise scope of particular grammatical functions must be defined intra-linguistically, on the basis of such 'coding strategies' as case marking, agreement and word order, as well as on the basis of diathesis alternations (especially, passivisation). In the case of English, case marking and agreement do not distinguish the two NPs in double object constructions, but word order potentially does: on the simplest definition of direct objects – as those NPs which necessarily (apart from well-defined constructions, such

[11] Similarly in the case of arguments which are not nominal or clausal at all, as in "behaving *in some way*" or "treating somebody *in some way*"; they are assigned the `advmod` label (also in $\text{UD}^{\text{EN}}_{\text{EWT}}$, despite being arguments on most accounts).

[12] `http://thedeliverymag.com/10-things-mother-nature-taught-me-about-motherhood/`

as topicalisation, heavy NP shift, etc.) immediately follow the verb – it is the pronoun *me* that should be analysed as the direct object in the two examples above. This stance is strongly supported by another grammatical property that such initial – immediately post-verbal – NPs in ditransitive constructions share with prototypical direct objects in monotransitive constructions, namely, they normally passivise in all varieties of English, unlike the non-initial NPs, which passivise only in some varieties and under certain conditions: "I was taught balance by Mother Nature" vs. %"Balance was taught me by Mother Nature" (where % indicates limited acceptability). Hence, as also argued in Andrews 2007: 184–188 with reference to passivisation facts, it is the first post-verbal NP – the recipient or benefactor – that should be considered the direct object in double object constructions in languages such as English.

Note that the UD guidelines are actually inconsistent about the status of the immediately post-verbal NP in double object constructions. While the fragment on indirect objects[13] defines them with reference to the semantic roles of recipient and benefactive, the fragment about direct objects[14] defines a (direct) object as "[t]ypically, [...] the noun phrase that denotes the entity acted upon or which undergoes a change of state or motion (the proto-patient)." These two definitions are often in conflict, as in the case of "My uncle taught me karate", where *me* is a much better candidate for the recipient or benefactive role than *karate*, so it should be classified as `iobj`, but it is also clearly an "entity acted upon", "which undergoes a change of state", so it should be classified as `obj`.

Given the arguments in Andrews 2007 and the inconsistency of current UD guidelines, we assume that the immediately post-verbal NP in English double object constructions is the direct object. Unfortunately, Andrews is not very clear about the status of the NP following the direct object, but he generally denies the possibility of indirect objects in English: "In English, Bantu, and many other languages..., we do not seem to find even *prima facie* plausible candidates for an indirect object grammatical relation" (Andrews 2007: 190). If so, perhaps the notion of core dependents should be limited to subjects and (direct) objects, and all other broadly nominal dependents – not only PPs but also the NPs following direct objects – should be treated as oblique. In fact, within LFG, this view is argued for in Alsina 1996 and Patejuk and Przepiórkowski 2016. On this view, the above two examples should be labelled as follows:

(32) Mother Nature *taught* [me]$_{\texttt{obj}}$ [[balance]$_{\texttt{obl}}$, and [that regular pattern interrupts can negate radical change]$_{\texttt{advcl}}$].

(33) My uncle said to hell with that and *taught* [me]$_{\texttt{obj}}$ [[karate]$_{\texttt{obl}}$, and [to fire weapons]$_{\texttt{xadvcl}}$].

Constraining cores to subjects and normally immediately post-verbal direct objects would also explain the following facts (Nathan Schneider, p.c.), otherwise somewhat awkward for the basic proposal presented in the previous subsection:

(34) He *told* [me]$_{\texttt{obj}}$ [[an idea]$_{\texttt{obl}}$ and [that he thought it was viable]$_{\texttt{advcl}}$].

(35) He *told* [me]$_{\texttt{obj}}$ [[about an idea]$_{\texttt{obl}}$ and [that he thought it was viable]$_{\texttt{advcl}}$].

The labels indicated in these examples reflect the constrained understanding of cores: in both cases both conjuncts are oblique, so the whole coordinate dependent is oblique. However, on the standard UD view, (34) involves a coordination of a direct object *an idea* and a finite clause, which – for this reason – must also be analysed as core. But (35) indicates that such a finite clause is oblique, as it is coordinated with the oblique *about an idea*. Hence, in the sentence "He told me that the idea was viable", the clausal dependent should be analysed as core on the basis of (34) and as oblique on the basis of (35). On the view advocated here, this potential problem for the current proposal is avoided: when a post-verbal NP (i.e., the direct object) is present, the following clause is treated as non-core.

In fact, there is additional evidence that there is a valency frame of TELL with a syntactic position which may be realised by an NP, a PP or a clause: not only is it possible to coordinate an NP and a clause, as in (34), or a PP and a clause, as in (35), but it is also possible to coordinate an NP and a PP directly, as in the following examples from the iWeb corpus (simplified here):

(36) I *told* her [[the story]$_{\texttt{obl}}$ and [about my own experiences]$_{\texttt{obl}}$].

(37) I... *told* them [[the situation]$_{\texttt{obl}}$ and [about the expungement]$_{\texttt{obl}}$]...

[13] `http://universaldependencies.org/u/dep/iobj.html`

[14] `http://universaldependencies.org/u/dep/obj.html`

This supports the decision to treat both realisations – bare nominal and prepositional – as oblique (given that only subjects and direct objects are core).

4.3 Limited Reintroduction of AAD

Of course, any proposal to relegate to the obliques various dependents treated so far as core only increases the sometimes perceived need to distinguish between argument-like obliques and adjunct-like obliques. To accommodate this desire, we propose to adopt a version of the solution presented in Zeman 2017, namely, to optionally subtype non-core dependents into 'arguments' – `obl:arg`, `advcl:arg` and `xadvcl:arg` – and 'adjuncts' (no `:arg` suffix), but with an important proviso that this subtyping should be optional and treebank-specific. Linguists have not made much progress since Tesnière's 1959 three pairwise-incompatible criteria, so particular languages and treebanks should be free in applying such subtyping or not and, if so, they should be free in deciding how they understand this distinction.

In summary, we propose the system of basic ad-verbal dependency relations in (38) (complemented by `vocative`, `expl`, etc.). Given the proposal in Appendix C of renaming labels in a way that makes more clear the orthogonality of grammatical function (`subj`, `obj` and `obl`) and grammatical category and 'openness' status (`n`, `c`, `x`), this table translates into the more transparent table (39).

(38)

	subject	object	non-core	
			(arg)	adj
nominal	nsubj	obj	obl:arg	obl
clausal	csubj	ccomp	advcl:arg	advcl
open	xsubj	xcomp	xadvcl:arg	xadvcl

(39)

	subject	object	non-core	
			(arg)	adj
nominal	subj:n	obj:n	obl:n:arg	obl:n
clausal	subj:c	obj:c	obl:c:arg	obl:c
open	subj:x	obj:x	obl:x:arg	obl:x

5 Conclusion

In this paper, we identified two rather fundamental problems in the current (version 2) UD schema: reliance on AAD despite declared avoidance of this dichotomy and the resulting inconsistent approach to the core vs. non-core distinction (most conspicuous in unlike category coordination). We aimed at a relatively conservative modification of the schema – cf. (38) and the less conservative but more transparent (39) – that solves these problems, as well as the problem discussed in Appendix C, namely, the lack of support for obligatorily controlled subjects and obliques. The crux of the proposal is the definition of (non-subject) core clausal dependents as "functioning like objects" according to the coordination test. We also considered the possibility of constraining the notion of core dependents to subjects and direct objects, and we accommodated for the proposal of Zeman 2017 to allow for a – treebank-specific – distinction between argument-like and adjunct-like obliques. We strongly believe that adopting these proposals will increase both intra-lingual and inter-lingual consistency of UD treebanks and, hence, of multiple tasks currently relying on dependency parsing.

Acknowledgements

Apart from the three anonymous reviewers, the following people – while not necessarily agreeing with all or any of the proposals in this paper – have provided valuable comments on the previous version of the paper and/or on some of its ideas discussed on the UD forum and at the Oslo Center for Advanced Study at the Norwegian Academy of Science and Letters, during the authors' fellowship there: Bill Croft, Dan Flickinger, Dag Haug, Sylvain Kahane, Marie-Catherine de Marneffe, Joakim Nivre, Stephan Oepen, Martha Palmer, Nathan Schneider and Amir Zeldes. The research reported here is partially supported by the Polish Ministry of Science and Higher Education within the CLARIN ERIC programme 2016–2018 (`http://clarin.eu/`).

Appendices

A Depictive Dependent of RECALL

The argument/adjunct status of various depictive dependents is perhaps even more controversial than that of other types of dependents. The Valency Dictionary of English (Herbst et al. 2004) mentions the depictive dependent introduced by *as* in the case of the verb REMEMBER (and illustrates it with "I *remember* her as pretty and sort of tallish"), but not in the case of RECALL. On the other hand, this type of dependent is considered an argument (or core) of both verbs in both FrameNet (Ruppenhofer et al. 2016) (both verbs evoke the `Remembering_experience` frame) and in PropBank (Kingsbury and Palmer 2002) (rolesets `remember.01` and `recall.02`).[15] Moreover, neither dictionary mentions the possibility of an *as*-less depictive in the case of REMEMBER, as in "I look at Macie [...] and try to *remember* him young" (abridged from the iWeb corpus). Given the level of uncertainty about the argument/adjunct status of such secondary predicates, UD representations should not differ dramatically depending on their AAD classification. But – as discussed in the main text and illustrated in (9)–(12) – they currently do. Appendix C proposes a solution that minimises such differences – see (48)–(51) there.

B Coordination of (Un)Like Grammatical Functions

The system of relations adopted in Universal Dependencies is a "mixed functional-structural system".[16] The core vs. non-core distinction is a very coarse-grained functional distinction: some grammatical functions – especially, subject and direct object – are core, other are non-core.

While linguists do not any longer believe that only the same syntactic categories may be coordinated, and many examples of unlike category coordination are given in the main text, they assume that – apart from some well-defined exceptions to be discussed below – only the same grammatical functions may be coordinated. For example, in the ditransitive construction, the direct object cannot normally be coordinated with the other NP argument: *"John gave Mary and a book". As core grammatical functions are different from non-core grammatical functions, this means that core dependents cannot normally be coordinated with non-core dependents. In other words, coordinating a core dependent and a non-core dependent would result in the coordination of different grammatical functions, contrary to the overwhelming generalisation that only the same grammatical functions may be coordinated.

There are two classes of exceptions to this generalisation, both very constrained empirically. The first (pointed to us by Amir Zeldes, p.c.) is sylleptic zeugma, as in:

(40) He *made* [[his apologies]$_{\texttt{obj}}$ and [for the door]$_{\texttt{obl}}$].

Such constructions, in which the two conjuncts invoke two different meanings of the head, have a metalinguistic feel and they are easy to distinguish from genuine coordination.

The second exception is the so-called lexico-semantic coordination (Sannikov 1979, 1980), occurring mainly in Slavic and in some neighbouring languages (Paperno 2012), as in the following sentence from the National Corpus of Polish (cited here after Patejuk and Przepiórkowski 2012b: 463):[17]

(41) *Obiecać* można [[wszystko]$_{\texttt{obj}}$ i [wszystkim]$_{\texttt{iobj}}$].
promise.INF may everything.ACC and everyone.DAT
'One may promise everything to everyone.'

As discussed in Patejuk and Przepiórkowski 2012a, 2012b and in Paperno 2012, such constructions are limited to certain classes of pronouns and quantifiers, including question pronouns (so-called *wh*-words), negative pronouns (so-called *n*-words) and pronominal-like words expressing existential or universal quantifiers (the latter illustrated in (41)). Again, such exceptional constructions are easily distinguished from run-of-the-mill cases of coordination, where the sameness of grammatical functions is preserved.

[15] VerbNet (Kipper et al. 2000, Kipper et al. 2006) does not seem to contain the relevant meaning of RECALL, so it is not discussed here.

[16] `http://universaldependencies.org/u/overview/syntax.html`

[17] The labels `obj` and `iobj` reflect how this example would be annotated in Polish UD treebanks.

C Open Dependents in UD

UD assumes that obligatory control only targets object-like core dependents, and not subjects or obliques. This is a reasonable first approximation, but cross-lingual facts show that it is ultimately false.

First, obligatory control into subjects, while rare, occurs in languages as diverse as Balinese and Polish. For example, Arka and Simpson 1998 argue convincingly that in the Balinese (42), in which the main verb *orahin* 'ask' is in the so-called objective voice (OV), the sequence *teka mai prajani* 'come here immediately' is the subject of this main verb and that its own subject (i.e., the subject of *teka* 'come') is obligatorily controlled by another dependent of the main verb, *Nyoman*.

(42) teka mai prajani ane orahin tiang Nyoman
come here immediately REL OV.ask 1 Nyoman
'Coming here immediately is what I asked Nyoman to do.'

While in Balinese control into subject clauses is related to the phenomenon of objective voice, in Polish it is a matter of lexical properties of some copular constructions (Patejuk and Przepiórkowski 2018) as well as, possibly, some (very rare) verbs, especially, UDAĆ SIĘ 'succeed in, manage', as in the following example from $\text{UD}^{\text{PL}}_{\text{SZ}}$:[18]

(43) Nie udało im się uruchomić ciągnika.
NEG manage.3.SG.N them.DAT RM start.INF tractor.GEN
'They didn't succeed in starting the tractor.'

Such examples involve strict (not partial, etc.; Landau 2013) obligatory control into the infinitival clause. Moreover, the infinitival clause should be analysed as the subject here; while it triggers the 3rd person singular neuter 'default agreement' (Dziwirek 1990) with the verb, expected when the subject is non-nominative or lacks case altogether (Przepiórkowski 1999), full subject–verb agreement is witnessed when the infinitival phrase is replaced by a nominative phrase:

(44) Nie udał im się rozruch ciągnika.
NEG manage.3.SG.M them.DAT RM start.NOM.SG.M tractor.GEN
'They didn't succeed in starting the tractor.'

Current UD guidelines do not make it possible to adequately handle control into subjects in languages such as Balinese or Polish: as controlled core dependents, such infinitival phrases should perhaps be marked as `xcomp`, but since they are subjects, they should actually be marked as `csubj`.[19]

In Polish, there are also clausal dependents which are obligatorily controlled despite being prototypical adjuncts. Such dependents are headed by adverbial participles, sometimes called converbs, as in (45), where the implicit subject of the participle *patrząc* 'looking' is obligatorily controlled by *hrabia* 'count', the subject of the main verb *przysiadł* 'sat down'.

(45) Hrabia przysiadł, bezmyślnie patrząc przed siebie.
count.NOM.SG.M sat.3.SG.M thoughtlessly looking before self
'The count sat down, looking ahead thoughtlessly.'

Again, according to the current guidelines, such obligatorily controlled adverbial participial clauses may either be marked as `advcl`, losing information about control, or as `xcomp`, wrongly promoting such clauses to the status of core dependents. Recognising such open adverbial clauses is a natural step in UD: just as the `xcomp` relation is based on the eponymous grammatical function in LFG, so a new relation could be added to UD based on LFG's XADJUNCT function.

Let us note that the above considerations make clear an elegant symmetry between different kinds of dependents, one that is missing in the current UD system: just as there are three kinds of object-like dependents – nominal (`obj` and perhaps `iobj`), 'closed' clausal (`ccomp`) and 'open' – controlled – clausal (`xcomp`), the same tripartite distinction should be made within subjects and within obliques. In the case of subjects, the distinction is currently made between nominal subjects (`nsubj`) and broadly verbal subjects (`csubj`), but the examples above show that a subclass of controlled subjects should be

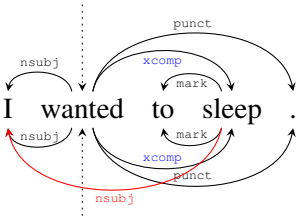
[18] The particle SIĘ is the so-called 'reflexive marker', RM, which – as in this case – is an inherent part of the verb, without any anaphoric meaning. NEG in (43) stands for 'negative marker'.

[19] In $\text{UD}^{\text{PL}}_{\text{SZ}}$, the infinitival clause in (43) is marked as an `xcomp`, thus losing information that it is the subject.

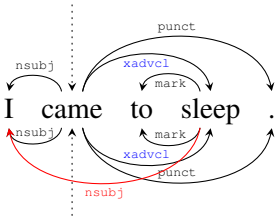
carved out of csubj – let us call this subclass xsubj. Similarly, the current class of modifier clauses should be split into closed modifier clauses – let us still call them advcl – and obligatorily controlled modifiers, say, xadvcl. This leads to the repertoire of basic dependents of verbs given in (28) in the main text.

The addition of these two relations has the effect of narrowing down the scope of csubj and advcl so that open dependents fall out of these (and into xsubj and xadvcl, respectively). Obviously, all open dependents – not just xcomp – are now expected to involve additional enhanced dependencies, indicating the relation between their covert subjects and some dependents of the higher predicate (usually, object or subject). This means that the difference between dependency representations (7)–(8) will now be reduced to the difference in label name:

(46)

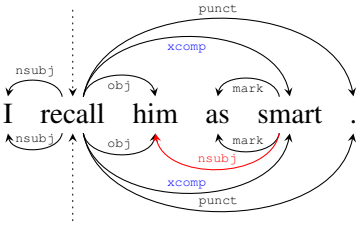


(47)

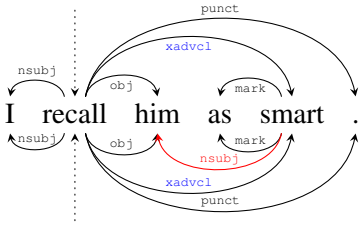


One welcome consequence of this proposal is that the representation of secondary predicates is uniform now: object-like secondary predicates still bear the xcomp relation, but oblique-like secondary predicates always bear the xadvcl relation to the verb now, rather than bearing the acl relation to the predicated nominal, if it is present, and the advcl relation to the verb, otherwise. Also, in both cases an enhanced relation points at the subject of secondary predication (if locally present). Thus, the analysis is essentially the same, whether the secondary predicate is considered core or non-core, with the only difference being the name of one dependency label (xcomp or xadvcl); the following representations should be compared to (9)–(12) in the main text.

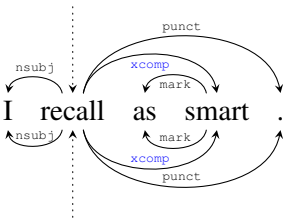
(48)



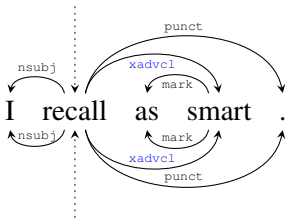
(49)



(50)



(51)



The symmetry between the three types of dependents – nominal, 'closed' clausal and 'open' clausal – can be made more conspicuous by renaming the labels as in (52).[20] However, these new names still mix two different types of information: grammatical function (subj, obj and obl) and grammatical category and 'openness' status (n, c, x). We propose to separate these two orthogonal types of information and make the bare grammatical function the main label of dependency relation, with categorial and 'openness' information given as subtypes, as in (53).

[20]The distinction between direct and indirect objects, if needed (see §4.2), could then be handled via a subtype to the nobj relation, e.g., nobj:sec.

(52)

	subject	object	non-core
nominal	nsubj	nobj	nobl
clausal	csubj	cobj	cobl
open	xsubj	xobj	xobl

(53)

	subject	object	non-core
nominal	subj:n	obj:n	obl:n
clausal	subj:c	obj:c	obl:c
open	subj:x	obj:x	obl:x

Given this last naming scheme, the relation of the whole coordinate structure to its head becomes clear, even in the case of unlike category coordination. For example, the two conjuncts in (54) (based on (13) in the main text) are obl:c and obl:n, so the whole coordinate structure could be labelled as obl, and similarly for (55), (56) and (57) (based on (17), (19) and (24) in the main text).

(54) At the White House, we *asked* the president [[whether Americans are safe at home]$_{\texttt{obl:c}}$ and [about the hunt for Osama bin Laden]$_{\texttt{obl:n}}$]$_{\texttt{obl}}$.

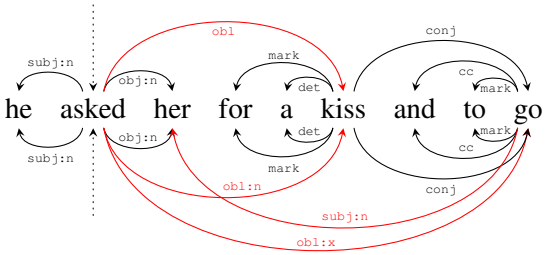
(55) Campbell made two visits to the home of domestic abuse victim Miss A, during which he *asked* her [[for a kiss]$_{\texttt{obl:n}}$ and [to go on a date with him]$_{\texttt{obl:x}}$]$_{\texttt{obl}}$.

(56) I do *remember* [[a lot of wine bottles]$_{\texttt{obj:n}}$ and [that Felix spoke almost nothing]$_{\texttt{obj:c}}$]$_{\texttt{obj}}$.

(57) Homeless people *need* [[a work ethic]$_{\texttt{obj:n}}$ and [to be self-reliant]$_{\texttt{obj:x}}$]$_{\texttt{obj}}$, Bender said.

Once the syntactic category of a dependent is expressed by a subtype, an unlike category coordination dependent could bear a relation which does not mention such a subtype. For example, the representation of the relevant fragment of (55) could be as in (58).

(58)



As in the case of the previous dependency structures, the differences between the basic representation (at the top) and the enhanced representation (at the bottom) are marked in red. Thus, while the enhanced representation contains dependency edges from *asked* to particular conjuncts, labelled as obl:n (*for a kiss*) and obl:x (*to go*), the basic representation contains just one edge labelled as obl. Additionally, the enhanced representation contains the explicit control information in the form of the subj:n dependency from the second conjunct to the controller *her*. This way it is possible to represent in UD the possibility of "control into selected conjuncts", somewhat problematic for some linguistic theories (Patejuk and Przepiórkowski 2014).

References

Alex Alsina. 1996. *The Role of Argument Structure in Grammar*. Number 62 in CSLI Lecture Notes. CSLI Publications, Stanford, CA.

Avery D. Andrews. 2007. The major functions of the noun phrase. In Timothy Shopen, editor, *Language Typology and Syntactic Description*, volume I: Clause Structure, pages 132–223. Cambridge University Press, Cambridge.

I Wayan Arka and Jane Simpson. 1998. Control and complex arguments in Balinese. In Miriam Butt and Tracy Holloway King, editors, *The Proceedings of the LFG'98 Conference*, University of Queensland, Brisbane. CSLI Publications.

Doug Arnold, Miriam Butt, Berthold Crysmann, Tracy Holloway King, and Stefan Müller, editors. 2016. *The Proceedings of the Joint 2016 Conference on Head-driven Phrase Structure Grammar and Lexical Functional Grammar*, Stanford, CA. CSLI Publications.

Joan Bresnan, Ash Asudeh, Ida Toivonen, and Stephen Wechsler. 2015. *Lexical-Functional Syntax*. Blackwell Textbooks in Linguistics. Wiley-Blackwell, 2nd edition.

Joan Bresnan, editor. 1982. *The Mental Representation of Grammatical Relations*. The MIT Press, Cambridge, MA.

William Croft. 2001. *Radical Construction Grammar: Syntactic Theory in Typological Perspective*. Oxford University Press, Oxford.

Mary Dalrymple. 2001. *Lexical Functional Grammar*. Academic Press, San Diego, CA.

Katarzyna Dziwirek. 1990. Default agreement in Polish. In Katarzyna Dziwirek, Patrick Farrell, and Errapel Mejías-Bikandi, editors, *Grammatical Relations: A Cross-Theoretical Perspective*. CSLI Publications, Stanford, CA.

Martin Haspelmath. 2014. Arguments and adjuncts as language-particular syntactic categories and as comparative concepts. *Linguistic Discovery*, 12(2):3–11.

Thomas Herbst, David Heath, Ian F. Roe, and Dieter Götz, editors. 2004. *A Valency Dictionary of English: A Corpus-Based Analysis of the Complementation Patterns of English Verbs, Nouns and Adjectives*. Mouton de Gruyter, Berlin.

Krystyna Kallas. 1993. *Składnia współczesnych polskich konstrukcji współrzędnych*. Wydawnictwo Uniwersytetu Mikołaja Kopernika, Toruń.

Paul Kingsbury and Martha Palmer. 2002. From TreeBank to PropBank. In *Proceedings of the Third International Conference on Language Resources and Evaluation, LREC 2002*, pages 1989–1993, Las Palmas. ELRA.

Karin Kipper, Hoa Trang Dang, William Schuler, and Martha Palmer. 2000. Building a class-based verb lexicon using TAGs. In *Proceedings of TAG+5 Fifth International Workshop on Tree Adjoining Grammars and Related Formalisms*.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2006. Extending verbnet with novel verb classes. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation, LREC 2006*, pages 1027–1032, Genoa. ELRA.

Idan Landau. 2013. *Control in Generative Grammar: A Research Companion*. Cambridge University Press, Cambridge.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation, LREC 2016*, pages 1659–1666, Portorož, Slovenia. ELRA, European Language Resources Association (ELRA).

Denis Paperno. 2012. *Semantics and Syntax of Non-Standard Coordination*. Ph.D. Thesis, University of California, Los Angeles.

Agnieszka Patejuk and Adam Przepiórkowski. 2012a. A comprehensive analysis of constituent coordination for grammar engineering. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, pages 2191–2207, Mumbai, India.

Agnieszka Patejuk and Adam Przepiórkowski. 2012b. Lexico-semantic coordination in Polish. In Miriam Butt and Tracy Holloway King, editors, *The Proceedings of the LFG'12 Conference*, pages 461–478, Stanford, CA. CSLI Publications.

Agnieszka Patejuk and Adam Przepiórkowski. 2014. Control into selected conjuncts. In Miriam Butt and Tracy Holloway King, editors, *The Proceedings of the LFG'14 Conference*, pages 448–460, Stanford, CA. CSLI Publications.

Agnieszka Patejuk and Adam Przepiórkowski. 2016. Reducing grammatical functions in Lexical Functional Grammar. In Arnold et al. 2016, pages 541–559.

Agnieszka Patejuk and Adam Przepiórkowski. 2018. Predicative constructions with infinitival and clausal subjects. In Miriam Butt and Tracy Holloway King, editors, *The Proceedings of the LFG'18 Conference*, Stanford, CA. CSLI Publications. Forthcoming.

Adam Przepiórkowski, Mirosław Bańko, Rafał L. Górski, Barbara Lewandowska-Tomaszczyk, Marek Łaziński, and Piotr Pęzik. 2011. National Corpus of Polish. In Zygmunt Vetulani, editor, *Proceedings of the 5th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, pages 259–263, Poznań, Poland.

Adam Przepiórkowski, Mirosław Bańko, Rafał L. Górski, and Barbara Lewandowska-Tomaszczyk, editors. 2012. *Narodowy Korpus Języka Polskiego*. Wydawnictwo Naukowe PWN, Warsaw.

Adam Przepiórkowski. 1999. *Case Assignment and the Complement-Adjunct Dichotomy: A Non-Configurational Constraint-Based Approach*. Ph.D. dissertation, Universität Tübingen, Tübingen.

Adam Przepiórkowski. 2016a. Against the argument–adjunct distinction in Functional Generative Description. *The Prague Bulletin of Mathematical Linguistics*, 106:5–20.

Adam Przepiórkowski. 2016b. How *not* to distinguish arguments from adjuncts in LFG. In Arnold et al. 2016, pages 560–580.

Adam Przepiórkowski. 2017a. Hierarchical lexicon and the argument/adjunct distinction. In Miriam Butt and Tracy Holloway King, editors, *The Proceedings of the LFG'17 Conference*, pages 348–367, Stanford, CA. CSLI Publications.

Adam Przepiórkowski. 2017b. On the argument–adjunct distinction in the Polish *Semantic Syntax* tradition. *Cognitive Studies / Études Cognitives*, 17:1–10.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson, Collin F. Baker, and Jan Scheffczyk, 2016. *FrameNet II: Extended Theory and Practice*. Revised November 1, 2016.

Ivan A. Sag, Gerald Gazdar, Thomas Wasow, and Steven Weisler. 1985. Coordination and how to distinguish categories. *Natural Language and Linguistic Theory*, 3:117–171.

Vladimir Z. Sannikov. 1979. Sočinitel'nye i sravnitel'nye konstrukcii: ix blizost', ix sintaksičeskoe predstavlenie I. *Wiener Slawistischer Almanach*, 4:413–432.

Vladimir Z. Sannikov. 1980. Sočinitel'nye i sravnitel'nye konstrukcii: ix blizost', ix sintaksičeskoe predstavlenie II. *Wiener Slawistischer Almanach*, 5:211–242.

Lucien Tesnière. 1959. *Éléments de Syntaxe Structurale*. Klincksieck, Paris.

Damon Tutunjian and Julie E. Boland. 2008. Do we need a distinction between arguments and adjuncts? Evidence from psycholinguistic studies of comprehension. *Language and Linguistics Compass*, 2(4):631–646.

Heinz Vater. 1978a. On the possibility of distinguishing between complements and adjuncts. In Werner Abraham, editor, *Valence, Semantic Case and Grammatical Relations*, pages 21–45. John Benjamins, Amsterdam.

Heinz Vater. 1978b. Probleme der Verbvalenz. Technical Report KLAGE Nr.1, Universität Köln.

Daniel Zeman. 2017. Core arguments in Universal Dependencies. In Simonetta Montemagni and Joakim Nivre, editors, *Proceedings of the Fourth International Conference on Dependency Linguistics (DepLing 2017)*, pages 287–296, Pisa, Italy.

Distinguishing affixoid formations from compounds

Josef Ruppenhofer
LeibnizScienceCampus *LiMo*
Institute for German Language
Mannheim
ruppenhofer@ids-mannheim.de

Michael Wiegand
Spoken Language Systems
Saarland University
michael.wiegand
@lsv.uni-saarland.de

Rebecca Wilm
Katja Markert
LeibnizScienceCampus *LiMo*
Computational Linguistics
Heidelberg University
{wilm,markert}
@cl.uni-heidelberg.de

Abstract

We study German affixoids, a type of morpheme in between affixes and free stems. Several properties have been associated with them – increased productivity; a bleached semantics, which is often evaluative and/or intensifying and thus of relevance to sentiment analysis; and the existence of a free morpheme counterpart – but not been validated empirically. In experiments on a new data set that we make available, we put these key assumptions from the morphological literature to the test and show that despite the fact that affixoids generate many low-frequency formations, we can classify these as affixoid or non-affixoid instances with a best F1-score of 74%.

1 Introduction

In this work, we study a subset of complex German words that includes many hapaxes, namely items that have as one of their morphological components a so-called **affixoid** or semi-affix. Examples of affixoid formations include *Gesetzeshengst* 'jobsworth, stickler for the letter of the law' [lit. 'law/legal stallion'] or *Mentalitätsmonster* 'person with laser focus' [lit. 'mentality monster']. The class of affixoid morphemes sits in between affixes and stems. While various criteria have been proposed to identify affixoids (Schmidt, 1987), at least the following three are widely taken for granted (ten Hacken, 2000; Elsen, 2009): (i) increased productivity; (ii) semantic bleaching/decreased semantic specificity; and (iii) an etymological and formal link to an existing free stem. The first two criteria are applied by comparing the affixoid to its corresponding free stem. For instance, in *Weingott* 'deity of wine', *Gott* is a free morpheme occurring with its regular meaning; in *Gitarrengott* 'guitar god', *Gott* is an affixoid: the whole word does not refer to a deity but to a talented human guitar player. Such evaluative or intensifying meanings are a hallmark of affixoids (Meibauer, 2013). The last of the three criteria distinguishes affixoids from affixes, which by definition occur only bound to other morphemes. For example, German *-heit*, whose English cognate is *-hood* as in *falsehood*, is an affix: there is no longer a related free form. By contrast, German *-gott* is an affixoid since there is a related free form *Gott*, cognate with English *god*.

Research on affixoids has been centered on Germanic languages like German, Dutch and Swedish (Ascoop and Leuschner, 2006; Booij, 2005; Booij and Hüning, 2014; Norde and Van Goethem, 2014). However, we believe affixoids are not an exclusive feature of these languages. They are likely to arise in other languages with productive compounding. For English, for instance, there is little to no systematic research but arguably *quality* (as in *quality press/furniture/diamonds* but not in *quality management*) and *nut* (as in *health/math/trivia nut* but not in *pecan nut*) can be considered English affixoids. Even for languages on which there is more research, that work is typically focused on the theoretical relevance of assuming a category of affixoids that is distinct from affixes on the one hand and compounds on the other. Very little quantitative work using corpus data has been done to, for instance, study the level of productivity for different affixoid candidates or to substantiate the intuition that most affixoid uses carry evaluative meanings. Our work thus fills an empirical gap in the theoretical discussion.

In addition, we are interested in studying affixoids for the purposes of sentiment analysis. On the one hand, theoretical linguistic work that notes the expressive function of affixoids such as Meibauer

Proceedings of the 27th International Conference on Computational Linguistics, pages 3853–3865
Santa Fe, New Mexico, USA, August 20-26, 2018.

(2013) suggests this. On the other hand, it is known from prior research on sentiment analysis that hapax words in general are often subjective (Wiebe et al., 2004). As we show in §3, affixoids tend to generate many (near-)hapax forms, which meshes with observations on their productivity in the morphological literature and the general expectation about the Zipfian distribution of word frequencies. Since hapaxes, by definition, cannot be readily analyzed based on their distribution in corpora, it would be very useful if we could make use of their intrinsic properties to classify such forms as affixoid uses (and therefore likely subjective) or not.

The main task that we set ourselves in this paper is morpheme sense disambiguation: we want to classify complex forms containing possible affixoids as to whether the morphemes in question really occur as affixoids with special meanings or as regular morphemes with their full, common semantics. We frame this task as a binary classification problem. The major contributions of our paper are:

- a gold standard annotation of complex forms containing potential affixoids, which make publicly available[1];
- empirical validation of claims regarding the association between intensification / evaluation and affixoid meanings;
- a detailed examination of various novel features that have been devised to detect the presence of an affixoid in a complex form.

2 Related work

Recently, Ruppenhofer et al. (2017) studied the problem of how well the polarity of complex words, including both compounds and derived forms, can be predicted based on its components and the word's morphological structure. They found that, while on core vocabulary – defined as items listed in the PolArt dictionary (Klenner et al., 2009) and highly likely to be listed in GermaNet (Hamp and Feldweg, 1997), the German WordNet resource – classification accuracy was as high as 85%, performance was severely degraded on more colloquial, domain-specific and linguistically creative forms taken from Wiktionary[2] and the Wortwarte[3] neologism project. Note that this research did not distinguish true compounds from complex forms involving affixoids.

Similar research was carried out earlier by Moilanen and Pulman (2008) who evaluated how well it was possible to classify unknown English words into one of three polarity classes based on morphological analysis. Compared to these prior efforts, our task is narrower as we do not deal with derivation and do not predict polarity for the complex entries. Our data is also more focused since we have only complex forms containing exactly two nouns and the second components of our data represent only 7 different lemmas. And importantly, our data are low frequency words in distinction to Ruppenhofer et al. (2017), whose main data set looked at higher-frequency core vocabulary.

In another line of research, several studies have tackled the problem of classifying senses as either *objective* or *subjective*. For English Su and Markert (2009) and Gyamfi et al. (2009) tackled this task on data from WordNet. Subjective entries are ones that possess polarity, that is, they could in further analysis be classified as either positive or negative. This task is similar to ours in the sense that a simple binary categorization of items is sought. However, there are key differences. First, our classification targets lemmas not word senses. That is, we assume that the complex lemmas are monosemous or have a clearly dominant sense in our data.[4] Second, the distinction *objective* versus *subjective* does not fully line up with the distinction *non-affixoid* use versus *affixoid* use. A complex form may contain the affixoid candidate in its regular objective sense but the other component may make the whole word subjective. An example of this kind is *Lieblings|hai* 'favorite shark'. Finally, since our complex forms are unlikely to be listed in lexical resources, we will usually lack information such as glosses, supersenses or example sentences for them.

[1] https://github.com/josefkr/affixoids

[2] https://de.wiktionary.org

[3] http://wortwarte.de/

[4] This assumption can be made plausible by the observations that longer words have more specific meanings than shorter ones, and that they tend to have fewer meanings than their head words, and fewer meanings than their components do on average (Altmann, 2002).

In another strand of research involving German morphology and sentiment analysis, Wiegand et al. (2016) developed an approach to classify the first element of German compounds as expressing either the source or target of evaluation, or neither, relative to the second element, if in the first step of analysis the second element was determined to be subjective. As do we, those authors focused on noun-noun compounds and they did not address polarity classification. However, their approach targets higher frequency words as it relies on the availability of sufficient corpus data to enable the use of distributional similarity. For our dataset, we cannot directly model the distributional properties of our complex items.

3 Data

Data creation. German has both nominal affixoids – items that are related to nouns (e.g. *Affen|tempo*, 'high speed' [lit. 'ape speed']) – and adjectival affixoids – items that are related to adjectives (e.g. *unheils|schwanger* 'ominous' [lit. 'pregnant with doom']).[5] Here, we concentrate on nominal affixoids. Two subtypes of nominal affixoids exist, depending on the position of the nominal affixoid in the complex word. *Affe*, as in *Affentempo*, is a prefixoid as it occurs as the first item in the compound-like complex word, whereas *Gott* is a suffixoid as it occurs as the second item in complex formations such as *Gitarrengott* 'guitar god'. To make the best use of our resources for annotation, we focus on suffixoids here as we believe that addressing suffixoids and prefixoids at the same time may not be helpful. Consider that for many suffixoid candidates, a suffixoid use is recognizable by the fact that the referent of the whole word is not a hyponym of the suffixoid in its basic meaning. For instance, a *Kredithai* 'loan shark' is not a shark. With prefixoids, this is usually different, as they tend to add evaluation or intensification but do not change the class of referent. A *Scheißauto* 'shit car' is still a car.

No pre-compiled resource is available that lists all suffixoids and the complex words in which they figure. We therefore first compiled a set of 60 possible suffixoids from the literature on morphology (Motsch, 2004; Erben, 2006; Elsen, 2009; Elsen, 2011; Fleischer et al., 2012) and then looked for complex forms containing them in the 1.7-billion deWaC web corpus (Baroni et al., 2009).[6] We queried the corpus for complex forms whose lemma ended in one of the suffixoids and where the suffixoid was preceded by at least four letters. The latter condition was imposed to rule out complex forms where the first part is a simple prefix. Further, we allowed only for complex forms spelled as contiguous strings as standard German orthography does not allow open compounds. E.g. the German word for *hammerhead shark* cannot be spelled as *Hammer Hai* using two white space-separated tokens.

The extracted forms were then semi-manually filtered for remaining errors. We used the SMOR morphological segmentation tool (Schmid et al., 2004; Faaß et al., 2010) to detect cases that did not represent simple noun-noun combinations. We excluded instances that were not compliant with standard German orthography, for instance, due to spelling errors or incorrect tokenization. We eliminated duplicates that differed only in whether the two parts of the complex form were written contiguously or separated by a hyphen. Thus, we for instance kept *Hammerhai* but not *Hammer-Hai* 'hammerhead shark'.

To keep the overall effort manageable, we selected seven items from our pool of suffixoids for annotation and subsequent use in our experiments. We chose these for two reasons. First, we did not want items that were extremely biased towards either affixoid or non-affixoid uses. For instance, more than 99% of the formations for *Dreck* 'dirt' and *Junkie* 'junkie, addict' are suffixoid formations. Second, we wanted items for which a sizable number of complex forms exist in our corpus. This criterion ruled out, for instance, *Base* 'female cousin' because most forms found in the corpus were false positives involving compounds using the homographic English noun *base* as a head, rather than intended German forms such as *Klatschbase* 'telltale/chatterbox'. Table 1 gives some basic descriptive information about the items. The complex forms of these 7 items will represent the raw data for our gold standard (*see also supplementary material*). In total, there are 1788 of such complex forms.

[5] Our literature survey did not throw up any German verbal affixoids. This is not unexpected since German also has very limited compounding involving verbs. They can only occur as modifiers in compounds with nominal or adjectival heads as in *Bratpfanne* 'frying pan' or *waschecht* 'colorfast' [lit. 'wash-true'] (Olsen, 2000).

[6] We also found 40 possible nominal prefixoids in the literature we reviewed. Note that these lists are not necessarily complete: the items discussed in the linguistic literature are discovered through introspection rather than in a data-driven way.

	basic sense	# senses	intensifying	polar	freq	freq. rank
Bolzen	bolt	4	Y	Y	115	34632.5
Bruder	brother	4	N	Y	12901	734.0
Gott	god	2	Y	Y	36488	245.0
Hai	shark	1	N	Y	247	19654.5
Hengst	stallion	2	N	Y	342	15551.5
Kaiser	emperor	2	Y	Y	12040	798.0
König	king	2	Y	Y	21439	415.0

Table 1: Properties of the suffixoid candidates; senses = major senses as defined by `duden.de` dictionary; frequency in deWaC; frequency and frequency rank according to `dlexdb.de` lexical database

Table 2 shows for each suffixoid candidate the number of complex forms in the gold standard annotations in which the candidate occurred as true suffixoid (Y), the number of complex forms in which it occurred as a regular compound head, cases where the annotators were unsure whether the candidates were mainly used as an affixoid or not, and the total of all forms found.

	Y	N	both	unsure	total
-bolzen	30	70	4	1	105
-bruder	21	158	3	1	183
-gott	78	354	41	0	473
-hai	16	49	0	3	68
-hengst	26	81	3	1	111
-kaiser	31	92	11	1	135
-könig	290	370	53	0	713
total	492	1174	115	7	1788

Table 2: Affixoid and non-affixoid uses (gold)

	dlexdb		GermaNet		Wiktionary		SentiMerge	
	all	aff	all	aff	all	aff	all	aff
-bolzen	17 (16.5)	1	1 (1.0)	0	1	0	4	0
-bruder	41 (22.4)	4	3 (1.6)	0	2	0	1	0
-gott	133 (28.1)	10	21 (4.4)	2	7	2	3	0
-hai	13 (19.1)	3	7 (10.3)	1	3	0	10	1
-hengst	23 (20.7)	7	2 (1.8)	1	0	0	3	1
-kaiser	25 (18.5)	3	1 (2.8)	0	1	0	0	0
-könig	177 (24.8)	54	19 (2.6)	5	14	0	6	1
total	429 (24.0)	82	54 (3.0)	9	28 (1.6)	2	27 (1.5)	3

Table 3: Coverage by lexical resources (absolute and %)

Table 3 shows that only 429 items (24.0%) of the 1788 forms in the gold standard are covered by the dlexdb lexical database (Heister et al., 2011), 54 (3.0%) are covered by GermaNet 11.0, and only 28 (1.6%) by Wiktionary. SentiMerge (Emerson and Declerck, 2014), the largest German sentiment lexicon with close to 100k entries, covers 27 (1.5%) forms. By contrast, 93% of the 9300 items in Ruppenhofer et al. (2017)'s main data set were covered by dlexdb, and 88.4% by GermaNet. Finally, Table 3 shows that of the formations included in the various resources, the majority are non-affixoid cases. Inclusion in a resource thus is not a very useful feature as it is only predictive of the majority class.

Figure 1 illustrates that the token frequencies for the complex forms with *König* 'king' as its second component have a largely Zipfian distribution: there are many complex forms with very low token frequencies (in fact, mostly their frequency is 1), whereas there are very few complex forms with high token frequencies. While we cannot show it here for lack of space, the shape of the distribution is the same for all the suffixoid candidates. Note further that the numbers shown in Table 2 do not delimit the productivity of these suffixoids. Consider that, for instance, in a collection of 120 million tweets, we found 905 complex forms for *König*, of which only 226 overlapped with the 713 forms found in the deWaC corpus.

Annotation of the affixoid vs. non-affixoid distinction. Most of the data was annotated by one of the authors. To assess how well human annotators agree on the distinction between affixoid uses and non-affixoid uses, two of the authors performed an annotation experiment on 200 randomly chosen instances of the total set of 1788 complex forms. The annotators could label each complex form as Y (only affixoid uses or predominantly affixoid uses), N (only non-affixoid uses or predominantly non-affixoid uses) and

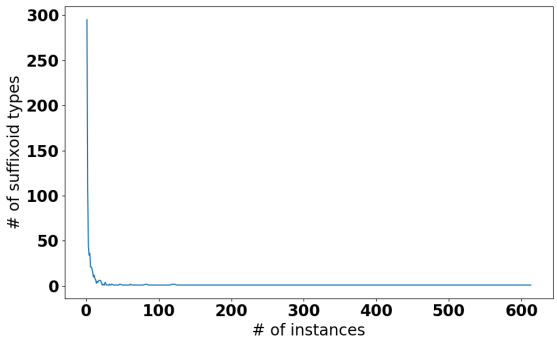


Figure 1: Frequency spectrum for *König* 'king'

	Y	N	B	total
Y	43	1	1	45
N	12	116	3	131
B	16	2	6	24
total	71	119	10	200

Figure 2: Annotator confusion matrix

B (both types of use likely). The annotators achieved a Cohen's kappa (Cohen, 1960) value of 0.67, which amounts to substantial agreement according to Landis and Koch (1977). As the confusion matrix in Figure 2 shows, the majority of errors are 'milder' cases where one annotator considers one type of use dominant whereas the other considers both types of uses plausible.

4 Features

4.1 Automatic features

We first explore features that can be extracted automatically from corpora and resources, which is the most realistic setting. (We discuss our handling of missing values in §4.2).

Suffixoid. Since the suffixoid candidates differ in how frequently they do in fact occur as affixoids and also differ in their evaluative meanings, it makes sense to capture which suffixoid in particular occurs in a complex form. We define a binary indicator variable for each suffixoid.

Frequency. There exists a significant body of research in quantitative linguistics on the productivity of morphological processes. But works on the productivity of compounding such as Altmann (2002) or Krott et al. (1999) do not make a distinction between affixoid formations and true compounds and therefore there are no prior results on their similarities and differences in regards to productivity: the assumption of increased productivity for affixoids has, to our knowledge, not been empirically validated. For our set suffixoids, we used Baayen's `languageR` R-package to compare, the lexical richness of compound formations to that of affixoid formations, per affixoid candidate and for pooled data. We found no consistent differences in terms of vocabulary growth curves, vocabulary growth rates or type-token ratios (Baayen, 2005). This result notwithstanding, we want to see if frequency information about the components and the complex forms could help us distinguish between the two kinds of complex forms. One relevant intuition that we had in this regard is that the complex forms of affixoidal uses may tend to have lower frequencies than complex forms representing regular compounds. Table 4 shows that for most items this seems to be borne out, although the difference is not statistically significant, with the exception of *-hai*. However, for *-könig* the reverse situation holds in a statistically significant way: the frequencies of affixoid formations are higher than those of the non-affixoid formations.

	affixoid	non-affixoid	p-value of t-test
-hai	37.50	12.41	**0.01**
-kaiser	3.40	6.64	0.13
-bolzen	4.21	4.36	0.93
-hengst	4.03	8.40	0.27
-könig	4.92	16.23	**0.00**
-bruder	1.88	6.04	0.39
-gott	10.07	19.90	0.36

Table 4: Frequency differences between affixoid and non-affixoid formations

Compositionality. This feature measures the compositionality of the complex word. The intuition is as follows. In regular compounds, the meaning of the complex form is more or less compositionally

derived from the meaning of the components. However, since affixoid uses of the morphemes in question go along with bleached meanings, the semantics of a complex word containing an affixoid use should be harder to model compositionally. Following Schulte im Walde et al. (2013), we represent each component as well as the complex word as vectors. As our measure of compositionality, we determine the cosine similarity between the sum of the component vectors and the vector of the complex word.

We used fastText (Bojanowski et al., 2016) in its default setting to train word embeddings on the SdeWaC-corpus (Faaß and Eckart, 2013), which contains about 880 million tokens and is a cleaned up version of the deWaC corpus (Baroni et al., 2009). The choice of fastText was motivated by the fact that fastText computes vectors for words by adding up the vectors for n-grams found in the words, which allows us to produce vectors for words not seen in the training data. Since many of our complex forms are (near-)hapaxes, this is a crucial benefit of fastText.

Pointwise mutual information. We use Pointwise mutual information (Church and Hanks, 1990) to capture the level of association between the two components of the complex word. The expectation is that the components of regular compounds exhibit higher PMI-scores than the components of a complex word involving an affixoid. This is motivated by Tellenbach (1985)'s observation that for complex forms containing an affixoid use, paraphrases containing the morpheme in question as a free word are unlikely. By contrast, compositional compounds are often paraphrased using their components. As an example, the compositional compound *Perserkönig* is readily paraphrased as *König der Perser* 'king of the Persians', whereas *Gitarrenkönig* is less likely to be paraphrased as *König der Gitarre* 'king of the guitar'. Even aside from paraphrasing, it seems more likely that *Perser* and *König* occur together in a text as they share the context of governance than it does that *Gitarre* and *König* co-occur.

GermaNet supersenses. As in the English WordNet (Miller, 1995), GermaNet's synsets are associated with supersenses that define high-level categorizations such as `Human`, `Animal`, `Artifact`, or `Group`. A lexical unit in GermaNet can be associated with one or more of its supersenses. Accordingly, we define three series of indicator variables that capture whether any sense of the complex form, the first component or the second component exhibits a particular supersense. We refer to all these features as *supersenses_bag*. This feature group is motivated by the observation that with several of our suffixoid candidates, the semantic types of the first component and the complex form tends to provide good evidence on whether the complex word will be an affixoidal use or not. For instance, the first component *Haflinger* in *Haflinger|hengst* refers to a breed of horses and the complex form is a regular compound. By contrast, the first component *Büro* 'office' of *Büro|hengst* refers to either an `Artifact` or a `Group` and the complex form represents an affixoidal use meaning something like 'pencil pusher'. Moreover, the complex form *Haflingerhengst* also has the supersense `Animal`, whereas *Bürohengst* has the supersense `Human`. Because for this particular set of suffixoid candidates differences between the second component and the complex word's supersenses may be particularly important, we experiment with an alternative set of supersense features (*supersenses_diffs*): we use a series of indicator variables that code whether the second component and the complex word differ in their value for a given supersense.[7]

Polarity. Since affixoid uses are likely to have evaluative meanings, we explore whether this is reflected in the polarity of the two components and the complex form. We extract polarity information for all three from SentiMerge (Emerson and Declerck, 2014). With 96,918 entries, it is to date the largest available polarity lexicon for German. SentiMerge was created by harmonizing and combining three smaller lexicons (PolArt (Klenner et al., 2009); GermanPolarityClues (Waltinger, 2010); and SentiWS (Remus et al., 2010)) using a Bayesian probabilistic model.

Psycholinguistic features. If available, we extract psycholinguistic ratings along four dimensions for the whole word and its components. This type of feature has been successfully used in various tasks, such as identifying metaphors (Turney et al., 2011; Klebanov et al., 2014); studying persuasion (Tan et al., 2016); sarcasm detection (Bamman and Smith, 2015); and, most similar to us, polarity prediction for complex words (Ruppenhofer et al., 2017). The first dimension places words on a scale from abstract to concrete (**abstconc**). Abstract words refer to things that we cannot perceive directly with

[7]Note that for other suffixoids not covered here such as *Papst* (lit. 'pope', suffixoid 'expert') and *Nest* (lit. 'nest', affixoid 'den/hideout') there is no difference at all between the supersenses of the second component and the complex word.

our senses (*integer*, *politics*, ...) whereas concrete words refer to things we can perceive (*sound*, *scent*, ...). The second dimension concerns imageability (**img**). A large subset of concrete words have a high imageability. These words refer to things that we can actually see (*chestnut*, *police jeep*, ...). The third rating dimension is valence (**val**), which measures the pleasantness of a word (*gift* vs. *punishment*). The final dimension, **arousal**, represents the intensity of emotion caused by a stimulus (*alert* vs. *calm*).[8] We obtain affective ratings from the resource of Köper and Schulte im Walde (2016). It provides information on 350k words and is far more comprehensive than the affective norm data of Kanske and Kotz (2010) or Lahl et al. (2009). It is also much larger than commonly used polarity lexicons for German such as PolArt (Klenner et al., 2009) or GermanPolarityClues (Waltinger, 2010).

Emotion. Since emotion information is commonly used in sentiment-related classification tasks (e.g. Tang et al. (2014), Sulis et al. (2016)), we wanted to see to what extent emotion information could benefit our task. For this purpose, we use the NRC Word-Emotion Association Lexicon (EmoLex) for English which was created by Mohammad and Turner (2013) using a crowdsourcing approach. EmoLex contains binary associations of words with the eight basic emotions (joy, sadness, anger, fear, disgust, surprise, trust, anticipation) of Plutchik (1962) . Although the German version of the lexicon was produced using machine translation, we use it here because we do not have a similarly large natively produced resource available for German. The German EmoLex covers 9630 lemmas. For each complex form, we extract the emotions associated with the overall word and do likewise for the first and the second components.

4.2 Missing value imputation

With features derived from resources (polarity, psycholinguistic ratings, emotion, supersenses and PMI), we face the problem that there are gaps in coverage. Various strategies are conceivable to fill in missing values. The first one we considered is to substitute for a word that is not covered that word which is most similar to it according to cosine similarity among the fastText vectors and which is covered by the resource. A second approach we considered is to use the mean value for the feature in question. A third approach uses the median. A fourth option is to use the modal value. Our experiments showed that for our SVM classifier, the choice of imputation strategy does not result in any statistically significant differences in results. We thus report results based on the first strategy, which is based on cosine similarity.

4.3 Manual feature annotations

In order to be able to explore to what extent the notions of intensification and evaluation correlate with the use of our target morphemes as either affixoids or regular morphemes, we added further annotations to our complex forms. We labeled them manually with respect to the features listed below. Based on the theoretical literature, they should be very predictive for our affixoid classification task.[9]

- Polarity of complex word: does the item have positive, negative or neutral polarity?
- High intensity of complex word: does the complex word express high intensity? *Temperamentsbolzen* 'highly temperamental person': yes; *Metallbolzen* 'metal bolt': no.
- Evaluativity of complex word: does the complex word mostly carry either a positive or negative evaluation? *Bürohengst* 'pencil pusher': yes; *Haflingerhengst* 'Haflinger stallion': no.
- Head Intensity: does the head (i.e. the potential affixoid) contribute an intensifying meaning to the overall word? *Charmebolzen* 'highly charming person': yes; *Riesenbolzen* 'giant bolt': no.
- Head Polarity: does the head of the complex word contribute an evaluating meaning to the overall word? *Fußballgott* 'footballing god': yes; *Wettergott* 'weather god': no.

5 Experiments

5.1 Automatically extracted features

Our experimental set-up is as follows. We perform a 5-fold cross validation. Our data is initially randomized before we defined folds that are held constant across all experiments. For our experiments,

[8] Valence and arousal are part of (Osgood et al., 1957)'s well-known theory of emotions.

[9] We tested agreement only for polarity of the complex word. Here, two annotators achieved a kappa of 0.86 on a random sample of 200 items.

we convert the instances of B(oth) to instances of Y, i.e. affixoid uses. The seven instances that the annotators had left as unsure are excluded from the experiments. We use the automatically extractable features discussed in §4.1. As our classifier, we use SVM (Vapnik, 1995), as implemented by SVM^{light} (Joachims, 1998). Note that we experimented with the cost-factor, by which training errors on positive examples (i.e. examples of class Y) outweight errors on negative examples (i.e. examples of class N). However, modifying it only helped in the lowest-performing configurations where we would otherwise perform exactly like the majority baseline. Table 5 reports results for the different features obtained with the default settings of SVM^{light}. The majority baseline we report represents a classifier that always predicts non-affixoid as this is the majority class for each suffixoid and the dataset as a whole.

		Y *(i.e. affixoid use)*			N *(i.e. non-affixoid use)*			all		
	Acc	P	R	F1	P	R	F1	P	R	F1
all features	0.78	0.74	0.53	0.62	0.79	0.90	0.84	0.76	0.72	0.74
supersenses_bag	0.74	0.69	0.43	0.53	0.76	0.90	0.82	0.72	0.67	0.69
psycholinguistic	0.72	0.69	0.30	0.41	0.72	0.93	0.81	0.71	0.61	0.66
frequency	0.71	0.67	0.34	0.44	0.73	0.91	0.81	0.70	0.62	0.66
emotion	0.67	0.67	0.08	0.14	0.67	0.97	0.79	0.66	0.53	0.59
supersenses_diffs	0.66	0.29	0.13	0.18	0.67	0.93	0.78	0.48	0.53	0.50
pmi	0.66	0.02	0.16	0.04	0.66	0.98	0.79	0.41	0.50	0.44
majority	0.66	0.00	0.00	0.00	0.66	1.00	0.79	0.33	0.50	0.40
compositionality	0.66	0.00	0.00	0.00	0.66	1.00	0.79	0.33	0.50	0.40
polarity	0.66	0.00	0.00	0.00	0.66	1.00	0.81	0.33	0.50	0.40

Table 5: Results for classification experiments per feature group; setting: 5-fold CV

The bag of supersenses is the best feature group and notably better than paying attention only to differences between second component and complex word in terms of supersenses.

5.2 Automatic features in balanced setting

The difference in performance between the two classes that we found when using all data (cf. §5.1) might be due either to the class imbalance or come about because the majority class N is inherently easier to learn. In order to tease this apart, we here repeat the previous experiment using all features in a 5-fold cross-validation setting, however this time with a balanced dataset of 970 items. The dataset is constructed so that each fold contains the same number of instances per suffixoid and those instances are themselves balanced equally across the affixoid and non-affixoid classes.[10] As Table 6 shows, with balanced data we can obtain much more balanced performance on the two classes. And despite the fact that we are using only 54.3% of the available data, the overall F1-score (0.73) is more or less tied with the result obtained on all data (0.74).

		Y			N			all		
	Acc	P	R	F1	P	R	F1	P	R	F1
all features	0.73.	0.71	0.78	0.75	0.76	0.67	0.71	0.73	0.73	0.73

Table 6: Results for classification experiment on balanced data using all features; setting: 5-fold CV

5.3 Feature ablation experiments

Table 7 shows an ablation experiment using all data. We remove one particular feature from the entire feature set at a time. The table shows that the performance decreases only marginally for most removed features, which indicates that most features encode information. The largest performance drop is caused by removing supersenses, which suggests that this is the feature with the most unique information.

Since the supersense feature is both the strongest individual feature (cf. Table 5) and also the feature encoding most unique information (cf. Table 7), we further examined this feature by listing the top 10 supersenses according to χ^2 ranking as shown in Table 8. The most predictive supersenses are predominantly those of the first component of the compound or those of the (complex) compound form itself. This first component is very helpful, for instance, for *König*. Its non-affixoid uses (e.g. *Sarazenenkönig* 'king of the Saracens') mostly co-occur with a first component denoting a `Location` or a nationality

[10] These latter two constraints result in only 970 items being used rather than 984, if we simply used all 492 instances of class Y and added as many instances from class N.

		Y			N			all		
	Acc	P	R	F1	P	R	F1	P	R	F1
all features	0.78	0.74	0.53	0.62	0.79	0.90	0.84	0.76	0.72	0.74
w/o psycholinguistic	0.78	0.74	0.53	0.61	0.79	0.91	0.84	0.76	0.72	0.74
w/o emotion	0.77	0.73	0.52	0.61	0.78	0.90	0.84	0.76	0.71	0.73
w/o polarity	0.77	0.73	0.53	0.61	0.79	0.90	0.84	0.76	0.71	0.73
w/o compositionality	0.77	0.72	0.52	0.60	0.78	0.90	0.84	0.75	0.71	0.73
w/o frequency	0.76	0.72	0.49	0.58	0.77	0.90	0.83	0.75	0.70	0.72
w/o pmi	0.76	0.75	0.46	0.57	0.77	0.92	0.84	0.76	0.69	0.72
w/o supersenses	0.75	0.70	0.47	0.56	0.77	0.90	0.83	0.73	0.69	0.71

Table 7: Ablation experiments where one feature is removed from the entire feature set

(Human) whereas its affixoid uses typically involve Events (e.g. *Tanzkönig* 'dancing king') or artifacts or objects (e.g. *Schotterkönig* 'king of gravel'). The importance of the supersense of the whole word arises in relation to the supersense of the second component. When the two differ for some of the affixoid candidates, the meaning of the whole word is usually only metaphorically related to the second component. For instance, while *Hai* 'shark' and *Hengst* 'stallion' refer to Animals in their basic meaning, complex forms in which they occur as affixoids refer to Humans. Similarly, *Bolzen* refers to an Artifact but complex forms containing an affixoid use of it refer to Humans.

rank	χ^2 score	supersense	rank	χ^2 score	supersense
1	97.76	second_component::Artifact	6	26.86	first_component::Event
2	65.01	first_component::Human	7	18.90	complex_form::Food
3	58.29	first_component::Creation	8	13.81	first_component::Food
4	48.59	first_component::Artifact	9	13.59	complex_form::Feeling
5	33.43	complex_form::Human	10	12.75	first_component::Time

Table 8: Top 10 supersense features according to χ^2 ranking

5.4 Cross-affixoid generalization

Using the automatic features, we experiment with a setting where we train on the instances of all but one affixoid candidate and then test on the instances of the reamining affixoid. For this experiment, we leave out the lexical suffixoid feature. Table 9 shows the results.

		Y			N			all		
test set	Acc	P	R	F1	P	R	F1	P	R	F1
bolzen	0.68	0.57	0.12	0.20	0.69	0.95	0.80	0.63	0.54	0.58
bruder	0.72	0.11	0.17	0.14	0.86	0.80	0.83	0.49	0.49	0.49
gott	0.75	0.51	0.36	0.42	0.80	0.88	0.84	0.66	0.62	0.64
hai	0.74	0.47	0.44	0.45	0.82	0.84	0.83	0.64	0.64	0.64
hengst	0.73	0.45	0.17	0.25	0.76	0.93	0.83	0.61	0.55	0.58
kaiser	0.72	0.67	0.19	0.30	0.72	0.96	0.82	0.69	0.57	0.63
könig	0.52	0.00	0.00	0.00	1.00	0.52	0.68	0.50	0.26	0.34

Table 9: Results for cross-affixoid classification, training on six affixoids, testing on the remaining one; all features except suffixoid

The results show lower performance than in the cross-validation setting where we had instances of all affixoids in the train and the test folds. For *könig* the results drop down as low as the majority baseline in the present setting. However, the low result is very likely also heavily driven by the fact that the instances of *könig* make up close to 40% of our whole dataset. In other words, when testing on *könig*, we use only 60% of the dataset to train on. On the other hand, we also see lower results on affixoids such as *bolzen* that are much less frequent than *könig* and for which we use more training instances in the present generalization setting than we did in the cross-validation setting in §5.1. This suggests that overall cross-affixoid generalization may be somewhat limited.

5.5 Gold-quality manual features

The final experiment we perform is a simple one: we use the manual features described in §4.3 and try to predict whether the complex form is an affixoid formation or not.

Table 10 shows that all features are highly predictive, with intensifying function of the suffixoid being the best. The results confirm that affixoids can be defined in terms of intensification and evaluation.

		Y			N			all		
	Acc	P	R	F1	P	R	F1	P	R	F1
suffixoid intensity	0.96	0.93	0.97	0.95	0.98	0.96	0.97	0.96	0.96	0.96
suffixoid evaluation	0.93	0.92	0.87	0.89	0.93	0.96	0.95	0.93	0.92	0.92
word polarity	0.89	0.78	0.97	0.86	0.98	0.85	0.91	0.88	0.91	0.90
word intensity	0.89	0.76	0.98	0.86	0.99	0.84	0.91	0.87	0.91	0.89
word evaluation	0.89	0.78	0.97	0.86	0.98	0.85	0.91	0.88	0.91	0.90

Table 10: Results for classification on all data using manual features; setting: 5-fold cross-validation

6 Error analysis

Table 11 shows confusion matrices per suffixoid candidate and for all data cumulatively for the best setting in our automatic experiments (cf. Table 5), in which all features were used in 5-fold cross validation. The breakdown per suffixoid candidate reveals that the overall performance is strongly influenced by *König*, which in particular contributes the bulk of the recall of true affixoid formations.

	Bolzen		Bruder		Gott		Hai		Hengst		König		Kaiser		**all**	
	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
Y	11	23	1	23	28	91	1	15	2	27	266	77	10	32	319	288
N	0	70	1	156	13	341	2	47	4	77	88	282	5	87	113	1060

Table 11: Confusion matrices on all data for the *all features* setting; gold: rows, predicted: columns

Table 12 below shows the confusion matrix for the experiment with a balanced data set reported in §5.2. As was to be expected based on the scores in Table 6, the performance across the different suffixoids is now much better. We can thus conclude that the difference in performance between the two classes that we found when using all data is largely owed to the class imbalance and that the distinction between affixoid formations and non-affixoid formations can in principle be learned for suffixoids other than *König*.

	Bolzen		Bruder		Gott		Hai		Hengst		König		Kaiser		**all**	
	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
Y	24	6	12	8	61	14	12	3	20	5	231	59	20	10	380	105
N	14	16	10	10	26	49	8	7	13	12	77	213	11	19	159	326

Table 12: Confusion matrices on balanced data for the *all features* setting; gold: rows, predicted: columns

7 Conclusion

In this paper we studied German complex forms containing suffixoid candidates. To do so, we constructed a new data set with 1788 items distributed about 7 potential affixoid candidates, which we make available to the research community. In one set of experiments we validated the high correlation between evaluative and intensifying semantics and affixoid status that has always been assumed by theoretical work but not demonstrated. In another experiment, we tackled the task of classifying our complex forms as affixoids formations or regular compounds. Being able to do so successfully can be useful for sentiment analysis, as the affixoid uses are typically evaluative. The task is difficult, though, as we have little reliable information about the complex forms available due to their low frequency. Still, we achieved best results of 74% F1-score using a custom set of features, among which supersenses had the most impact.

In future work, we plan on pursuing lines of research that we needed to leave open here. First, we would like to cover more suffixoid candidates and also extend this work to prefixoid formations. Second, we want to tackle affixoids that occur in complex adjectival forms. Finally, once we have annotated data for further affixoid candidates in hand, we would like to explore if it is possible to identify morphemes that have affixoid uses from properties of their formations, including their frequency distribution, and distinguish them from morphemes that only participate in regular compounds. Since so far affixoids have been identified based on human introspection, we do not know how many of them actually exist in German or other languages.

Acknowledgements

Michael Wiegand and Rebecca Wilm were partially supported by the German Research Foundation (DFG) under grants WI 4204/2-1 and RU 1873/2-1. We would also like to thank Ulrich Heid for the inspiration to study affixoids.

References

Gabriel Altmann. 2002. Morphologie. In Altmann Gabriel, Dariusch Bagheri, Hans Goebl, Reinhard Köhler, and Claudia Prün, editors, *Einführung in die quantitative Lexikologie*. Peust & Gutschmidt, Göttingen.

Kristin Ascoop and Torsten Leuschner. 2006. "affixoidhungrig? skitbra!" comparing affixoids in german and swedish. *STUF–Sprachtypologie und Universalienforschung*, 59(3):241–252.

R. Harald Baayen. 2005. Morphological productivity. In Reinhard Köhler and Gabriel Altmann Rajmund G. Piotrowski, editors, *Quantitative Linguistik / Quantitative Linguistics. Ein internationales Handbuch / An International Handbook.*, volume 27, chapter Morphological productivity, pages 243–255. De Gruyter Mouton, Berlin/Boston.

David Bamman and Noah A. Smith. 2015. Contextualized sarcasm detection on twitter. In Meeyoung Cha, Cecilia Mascolo, and Christian Sandvig, editors, *ICWSM*, pages 574–577.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226, Sep.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR*, abs/1607.04606.

Geert Booij and Matthias Hüning. 2014. Affixoids and constructional idioms. In Rony Boogaart, Timothy Colleman, and Gijsbert Rutten, editors, *Extending the scope of Construction Grammar*, pages 77–106. De Gruyter Mouton, Berlin.

Geert Booij. 2005. Compounding and derivation. In Wolfgang U. Dressler, Dieter Kastovsky, Oskar E. Pfeiifer, and Franz Rainer, editors, *Morphology and its demarcations*, Amsterdam Studies in the Theory and History of Linguistic Science, pages 109–132. John Benjamins Publishing Company.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Comput. Linguist.*, 16(1):22–29, March.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Hilke Elsen. 2009. Affixoide: Nur was benannt wird, kann auch verstanden werden. *Deutsche Sprache: Zeitschrift für Theorie, Praxis, Dokumentation*, pages 316–333.

Hilke Elsen. 2011. *Grundzüge der Morphologie des Deutschen*. Walter de Gruyter.

Guy Emerson and Thierry Declerck. 2014. SentiMerge: Combining sentiment lexicons in a Bayesian framework. In *Proceedings of the Workshop on Lexical and Grammatical Resources for Language Processing*.

Johannes Erben. 2006. *Einführung in die deutsche Wortbildungslehre*. Grundlagen der Germanistik.

Gertrud Faaß and Kerstin Eckart. 2013. Sdewac – a corpus of parsable sentences from the web. In Iryna Gurevych, Chris Biemann, and Torsten Zesch, editors, *Language Processing and Knowledge in the Web*, pages 61–68, Berlin, Heidelberg. Springer Berlin Heidelberg.

Gertrud Faaß, Ulrich Heid, and Helmut Schmid. 2010. Design and application of a gold standard for morphological analysis: Smor as an example of morphological evaluation. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. European Languages Resources Association (ELRA).

W. Fleischer, I. Barz, and M. Schröder. 2012. *Wortbildung der deutschen Gegenwartssprache*. De Gruyter Studium. De Gruyter.

Yaw Gyamfi, Janyce Wiebe, Rada Mihalcea, and Cem Akkaya. 2009. Integrating knowledge for subjectivity sense labeling. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 10–18, Stroudsburg, PA, USA. Association for Computational Linguistics.

Birgit Hamp and Helmut Feldweg. 1997. Germanet - a lexical-semantic net for german. In *Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 9–15.

Julian Heister, Kay-Michael Würzner, Johannes Bubenzer, Edmund Pohl, Thomas Hanneforth, Alexander Geyken, and Reinhold Kliegl. 2011. dlexDB–eine lexikalische Datenbank für die psychologische und linguistische Forschung. *Psychologische Rundschau*.

Thorsten Joachims. 1998. Making large-scale svm learning practical. Technical report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund.

Philipp Kanske and Sonja A Kotz. 2010. Leipzig affective norms for German: A reliability study. *Behavior research methods*, 42(4):987–991.

Beata Beigman Klebanov, Ben Leong, Michael Heilman, and Michael Flor. 2014. Different texts, same metaphors: Unigrams and beyond. In *Proceedings of the Second Workshop on Metaphor in NLP*, pages 11–17.

Manfred Klenner, Angela Fahrni, and Stefanos Petrakis. 2009. Polart: A robust tool for sentiment analysis. In *Proceedings of the 17th Nordic Conference on Computational Linguistics (NODALIDA 2009)*, pages 235–238.

Maximilian Köper and Sabine Schulte im Walde. 2016. Automatically generated affective norms of abstractness, arousal, imageability and valence for 350000 german lemmas. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, pages 2595–2598.

Andrea Krott, Robert Schreuder, and R. Harald Baayen. 1999. Complex words in complex words. *Linguistics*, 37:905–926.

Olaf Lahl, Anja S. Göritz, Reinhard Pietrowsky, and Jessica Rosenberg. 2009. Using the World-Wide Web to obtain large-scale word norms: 190,212 ratings on a set of 2,654 German nouns. *Behavior Research Methods*, 41(1):13–19.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Jörg Meibauer. 2013. Expressive compounds in german. *Word Structure*, 6(1):21–42.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Saif M. Mohammad and Peter D. Turner. 2013. Crowdsourcing a word-emotion association lexicon. *Computational Intelligence*, 29(3):436–465, August.

Karo Moilanen and Stephen Pulman. 2008. The Good, the Bad, and the Unknown: Morphosyllabic Sentiment Tagging of Unseen Words. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, HLT-Short '08, pages 109–112, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wolfgang Motsch. 2004. *Deutsche Wortbildung in Grundzügen*. Schriften des Instituts für deutsche Sprache. Walter de Gruyter.

Muriel Norde and Kristel Van Goethem. 2014. Similes, affixoids and debonding. A corpus-based analysis of 'giant' in Dutch, German, Swedish and French. In *CoLiDi 2014, Contrastive Linguistics and Diachrony*.

Susan Olsen. 2000. Composition. In Geert E. Booij, Christian Lehmann, and Joachim Mugdan, editors, *Morphologie: Ein Internationales Handbuch Zur Flexion und Wortbildung*, number Bd. 1 in Handbook of Linguistics and Communication Science Series, pages 897–916. Mouton de Gruyer.

Charles E. Osgood, George J. Suci, and Percy H. Tannenbaum. 1957. The measurement of meaning. *Urbana: University of Illinois Press*.

Robert Plutchik. 1962. *The emotions: facts, theories, and a new model*. Studies in psychology. Random House.

Robert Remus, Uwe Quasthoff, and Gerhard Heyer. 2010. Sentiws - a publicly available german-language resource for sentiment analysis. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may. European Language Resources Association (ELRA).

Josef Ruppenhofer, Petra Steiner, and Michael Wiegand. 2017. Evaluating the morphological compositionality of polarity. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 625–633. INCOMA Ltd.

Helmut Schmid, Arne Fitschen, and Ulrich Heid. 2004. Smor: A german computational morphology covering derivation, composition, and inflection. In *Proceedings of the IVth International Conference on Language Resources and Evaluation (LREC 2004)*, pages 1263–1266.

Günter Schmidt. 1987. Das Affixoid. Zur Notwendigkeit und Brauchbarkeit eines beliebten Zwischenbegriffs der Wortbildung. In Gabriele Hoppe, Alan Kirkness, Elisabeth Link, Isolde Nortmeyer, Wolfgang Rettig, and Günter Schmidt, editors, *Deutsche Lehnwortbildung. Beiträge zur Erforschung der Wortbildung mit entlehnten WB-Einheiten im Deutschen*, Forschungsberichte des Instituts für deutsche Sprache, pages 53–101. Narr.

Sabine Schulte im Walde, Stefan Müller, and Stephen Roller. 2013. Exploring vector space models to predict the compositionality of german noun-noun compounds. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics, *SEM 2013, June 13-14, 2013, Atlanta, Georgia, USA.*, pages 255–265.

Fangzhong Su and Katja Markert. 2009. Subjectivity recognition on word senses via semi-supervised mincuts. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 1–9, Stroudsburg, PA, USA. Association for Computational Linguistics.

Emilio Sulis, Delia Irazú Hernández Farías, Paolo Rosso, Viviana Patti, and Giancarlo Ruffo. 2016. Figurative messages and affect in twitter: Differences between# irony,# sarcasm and# not. *Knowledge-Based Systems*, 108:132–143.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 613–624, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Duyu Tang, Furu Wei, Bing Qin, Ting Liu, and Ming Zhou. 2014. Coooolll: A deep learning system for twitter sentiment classification. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 208–212.

Elke Tellenbach. 1985. Wortbildungsmittel im wörterbuch. zum status der affixoide. *Linguistische Studien*, pages 264–315.

Pius ten Hacken. 2000. Derivation and compounding. In Geert E. Booij, Christian Lehmann, and Joachim Mugdan, editors, *Morphologie: Ein Internationales Handbuch Zur Flexion und Wortbildung*, number Bd. 1 in Handbook of Linguistics and Communication Science Series, pages 349–360. Mouton de Gruyer.

Peter D. Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 680–690, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA.

Ulli Waltinger. 2010. GermanPolarityClues: A Lexical Resource for German Sentiment Analysis . In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *LREC*, pages 1638–1642. European Language Resources Association.

Janyce Wiebe, Theresa Wilson, Rebecca Bruce, Matthew Bell, and Melanie Martin. 2004. Learning subjective language. *Computational linguistics*, 30(3):277–308.

Michael Wiegand, Christine Bocionek, and Josef Ruppenhofer. 2016. Opinion Holder and Target Extraction on Opinion Compounds - A Linguistic Approach. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 800–810. The Association for Computational Linguistics.

A Survey on Open Information Extraction

Christina Niklaus[1], Matthias Cetto[1], André Freitas[2] and **Siegfried Handschuh[1]**
[1] Faculty of Computer Science and Mathematics, University of Passau
{christina.niklaus, matthias.cetto, siegfried.handschuh}@uni-passau.de
[2] School of Computer Science, University of Manchester
andre.freitas@manchester.ac.uk

Abstract

We provide a detailed overview of the various approaches that were proposed to date to solve the task of Open Information Extraction. We present the major challenges that such systems face, show the evolution of the suggested approaches over time and depict the specific issues they address. In addition, we provide a critique of the commonly applied evaluation procedures for assessing the performance of Open IE systems and highlight some directions for future work.

1 Introduction

Information extraction (IE) turns the unstructured information expressed in natural language text into a structured representation (Jurafsky and Martin, 2009) in the form of relational tuples consisting of a set of arguments and a phrase denoting a semantic relation between them: $\langle arg1; rel; arg2 \rangle$. Traditional approaches to IE focus on answering narrow, well-defined requests over a predefined set of target relations on small, homogeneous corpora. To do so, they take as input the target relation along with hand-crafted extraction patterns or patterns learned from hand-labeled training examples (e.g., Agichtein and Gravano (2000), Brin (1999), Riloff and Jones (1999)). Consequently, shifting to a new domain requires the user to not only name the target relations, but also to manually define new extraction rules or to annotate new training data by hand. Thus, those systems rely on extensive human involvement. In order to reduce the manual effort required by IE approaches, Banko et al. (2007) introduced a new extraction paradigm: Open IE. Unlike traditional IE methods, Open IE is not limited to a small set of target relations known in advance, but rather extracts all types of relations found in a text. In that way, it facilitates the domain-independent discovery of relations extracted from text and scales to large, heterogeneous corpora such as the Web. Hence, Banko et al. (2007) identified three major challenges for Open IE systems:

Automation. Open IE systems must *rely on unsupervised extraction strategies*, i.e. instead of specifying target relations in advance, possible relations of interest must be automatically detected while making only a single pass over the corpus. Moreover, the manual labor of creating suitable training data or extraction patterns must be reduced to a minimum by requiring only a small set of hand-tagged seed instances or a few manually defined extraction patterns.

Corpus Heterogeneity. Heterogeneous datasets form an obstacle for profound linguistic tools such as syntactic or dependency parsers, since they commonly work well when trained and applied to a specific domain, but are prone to produce incorrect results when used in a different genre of text. Furthermore, Named Entity Recognition (NER) is unsuitable to target the variety and complexity of entity types on the Web. As Open IE systems are intended for *domain-independent usage*, such tools should be avoided in favor of shallow parsing methods such as part-of-speech (POS) taggers.

Efficiency. In order to *readily scale to large amounts of text*, Open IE systems must be computationally efficient. Enabling fast extraction over huge datasets, shallow linguistic features, like POS tags, are to be preferred over deep features that are derived from parse trees.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3866–3878
Santa Fe, New Mexico, USA, August 20-26, 2018.

These criteria were first implemented in the Open IE system TEXTRUNNER, which was presented together with the task definition in Banko et al. (2007). This seminal work triggered a lot of research effort in this area, resulting in a multitude of proposed approaches that often did not strictly adhere to these initial guidelines. For example, to date, Open IE systems are commonly evaluated on rather small-scale, domain-dependent corpora. In addition, recent approaches frequently rely on the output of a dependency parser to identify extraction patterns, thereby hurting the domain-independence and efficiency assumptions.

```
OLLIE:
(1) (Republican candidate Mitt Romney; will be elected President in; 2008)[enabler=If he wins five key states]
(2) (Republican candidate Mitt Romney; will be elected; President)[enabler=If he wins five key states]
(3) (Mitt Romney; be candidate of; Republican)
(4) (Mitt Romney; be candidate for; Republican)
(5) (he; wins; five key states)

ReVerb:
(6) (he; wins; five key states)
(7) (Republican candidate Mitt Romney; will be elected President in; 2008)

PredPatt:
(8) (he; wins; five key states)
(9) (Republican candidate Mitt Romney; will be elected President in; 2008)

ClausIE:
(10) (he; wins; five key states)
(11) (Republican candidate Mitt Romney; will be elected; President in 2008 If he wins five key states)
(12) (Republican candidate Mitt Romney; will be elected; President in 2008)

OpenIE 5.0:
(13) (Republican candidate Mitt Romney; will be elected; President; T:in 2008)
(14) (he; wins; five key states)

Graphene:
(15) #1        CORE        (Mitt Romney; will be elected; President)
("a)        CONTEXT:NOUN_BASED        Mitt Romney was a republican candidate .
("b)        CONTEXT:TEMPORAL         in 2008 .
("c)        CONTEXT:CONDITION         #3
("d)        CONTEXT:NOUN_BASED        #2
(16) #2        CORE        (Mitt Romney; was; a republican candidate)
(17) #3        CONTEXT        (he; wins; five key states)
```

Figure 1: Comparison of the output generated by different Open IE systems for the input sentence *"If he wins five key states, Republican candidate Mitt Romney will be elected President in 2008."*

2 Open IE Systems

A large body of work on the task of Open IE has been described since its introduction by Banko et al. (2007). Existing Open IE approaches make use of a set of patterns in order to extract relational tuples from a sentence, each consisting of argument phrases and a phrase that expresses a semantic relation between them. Such extraction patterns are either hand-crafted or learned from automatically labeled training data, as shown below.

2.1 Learning-based Systems

The line of work on Open IE begins with TEXTRUNNER (Banko et al., 2007), a self-supervised learning approach consisting of three modules. First, given a small sample of sentences from the Penn Treebank, the learner applies a dependency parser to heuristically identify and label a set of extractions as positive and negative training examples. This data is then used as input to a Naive Bayes classifier which learns a model of trustworthy relations using unlexicalized POS and noun phrase (NP) chunk features. The self-supervised nature mitigates the need for hand-labeled training data, and unlexicalized features help scale to the multitudes of relations found on the Web. The second component, the extractor, then generates candidate tuples by first identifying pairs of NP arguments and then heuristically designating each word in between as part of a relation phrase or not. Next, each candidate extraction is presented to the classifier, whereupon only those labeled as trustworthy are kept. Restricting to the use of shallow features in this step makes TEXTRUNNER highly efficient. Finally, a redundancy-based assessor assigns a probability to each retained tuple based on the number of sentences from which each extraction was found, thus exploiting the redundancy of information in Web text and assigning higher confidence to extractions that occur multiple times.

WOE (Wu and Weld, 2010) also learns an open information extractor without direct supervision. It makes use of Wikipedia as a source of training data by bootstrapping from entries in Wikipedia infoboxes, i.e. by heuristically matching infobox attribute-value pairs with corresponding sentences in the article. This data is then used to learn extraction patterns on both POS tags (WOE^{pos}) and dependency parses (WOE^{parse}). Former extractor utilizes a linear-chain Conditional Random Field (CRF) to train a model of relations on shallow features which outputs certain text between two NPs when it denotes a relation. Latter approach, in contrast, makes use of dependency trees to build a classifier that decides whether the shortest dependency path between two NPs indicates a semantic relation. By operating over dependency parses, even long-range dependencies can be captured. Accordingly, when comparing their two approaches, Wu and Weld (2010) show that the use of dependency features results in an increase in precision and recall over shallow linguistic features, though, at the cost of extraction speed, hence negatively affecting the scalability of the system.

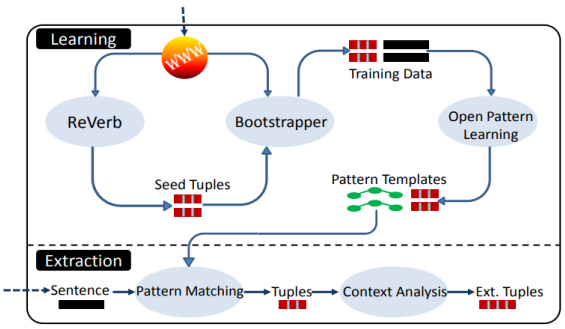


Figure 2: OLLIE's system architecture (Mausam et al., 2012). OLLIE begins with seed tuples from REVERB, uses them to build a bootstrap learning set, and learns open pattern templates. These are applied to individual sentences at extraction time.

OLLIE (Mausam et al., 2012) follows the idea of bootstrap learning of patterns based on dependency parse paths. However, while WOE relies on Wikipedia-based bootstrapping, OLLIE applies a set of high precision seed tuples from its predecessor system REVERB (see section 2.2) to bootstrap a large training set over which it learns a set of extraction pattern templates using dependency parses (see Figure 2). In contrast to previously presented systems that fully ignore the context of a tuple and thus extract propositions that are not asserted as factual, but are only hypothetical or conditionally true, OLLIE includes a context-analysis step in which contextual information from the input sentence around an extraction is analyzed to expand the output representation by adding attribution and clausal modifiers, if necessary, and thus increasing the precision of the system (see extractions (1) and (2) in Figure 1; for details, see section 2.4). Moreover, OLLIE is the first Open IE approach to identify not only verb-based relations, but also relationships mediated by nouns and adjectives (see extractions (3) and (4) in Figure 1). In that way, it expands the syntactic scope of relational phrases to cover a wider range of relation expressions, resulting in a much higher yield (at comparable precision) as compared to previous systems.

More recently, Yahya et al. (2014) proposed ReNoun, an Open IE system that entirely focuses on the extraction of noun-mediated relations. Starting with a small set of high-precision seed facts relying on manually specified lexical patterns that are specifically tailored for NPs, a set of dependency parse patterns for the extraction of noun-based relations is learned with the help of distant supervision (Mintz et al., 2009). These patterns are then applied to generate a set of candidate extractions which are assigned a confidence score based on the frequency and coherence of the patterns producing them.

2.2 Rule-based Systems

The second category of Open IE systems make use of hand-crafted extraction rules. This includes REVERB (Fader et al., 2011), a shallow extractor that addresses three common errors of hitherto existing Open IE systems: the output of such systems frequently contains a great many of uninformative extractions (i.e. extractions that omit critical information), incoherent extractions (i.e. extractions where the relational phrase has no meaningful interpretation) and overly-specific relations that convey too much information to be useful in further downstream semantic tasks. REVERB improves over those approaches by introducing a syntactic constraint that is expressed in terms of a simple POS-based regular expres-

sion (see Figure 3), covering about 85% of verb-based relational phrases in English text, as a linguistic analysis has revealed. In that way, the amount of incoherent and uninformative extractions is reduced. Moreover, in order to avoid overspecified relational phrases, a lexical constraint is presented which is based on the idea that a valid relational phrase should take many distinct arguments in a large corpus. Besides, while formerly proposed approaches start with the identification of candidate argument pairs, REVERB follows a relation-centric approach by first determining relational phrases that satisfy above-mentioned constraints, and then finding a pair of NP arguments for each such phrase. An example output produced by ReVerb can be seen in Figure 1 (6-7).

Whereas previously mentioned Open IE systems focus on the extraction of binary relations, commonly leading to extraction errors such as incomplete, uninformative or erroneous propositions, KRAKEN (Akbik and Löser, 2012) is the first approach to be specifically built for capturing complete facts from sentences by gathering the full set of arguments for each relational phrase within a sentence, thus producing tuples of arbitrary arity. The identification of relational phrases and their corresponding arguments is based on hand-written extraction rules over typed dependency parses.

$V \mid VP \mid VW^*P$
V = verb particle? adv?
W = (noun | adj | adv | pron | det)
P = (prep | particle | inf. marker)

Figure 3: ReVerb's POS-based regular expression for reducing incoherent and uninformative extractions.

EXEMPLAR (Mesquita et al., 2013) applies a similar approach for extracting n-ary relations, using hand-crafted patterns based on dependency parse trees to detect a relation trigger and the arguments connected to it. Based on the task of Semantic Role Labeling (SRL), whose key idea is to classify semantic constituents into different semantic roles (Christensen et al., 2010), it assigns each argument its corresponding role (such as subject, direct object or prepositional object).

A more abstract approach, PROPS, was suggested by Stanovsky et al. (2016), who argue that it is hard to read out from a dependency parse the complete structure of a sentence's propositions, since, amongst others, different predications are represented in a non-uniform manner and proposition boundaries are not easy to detect. Therefore, they introduce a more semantically-oriented sentence representation that is generated by transforming a dependency parse tree into a directed graph which is tailored to directly represent the proposition structure of an input sentence. Consequently, extracting propositions from this novel output format is straightforward. The conversion of the dependency tree into the proposition structure is carried out by a rule-based converter.

PredPatt (White et al., 2016) follows a similar approach. It employs a set of non-lexicalized rules defined over Universal Dependency (UD) parses (Marneffe et al., 2014) to extract predicate-argument structures. In doing so, PredPatt constructs a directed graph, where a special dependency ARG is built between the head token of a predicate and the head tokens of its arguments, while the original UD relations are preserved within predicate and argument phrases. As PredPatt uses language-agnostic patterns on UD structures, it is one of the few Open IE systems that work across different languages.

2.3 Clause-based Systems

Aiming to improve the accuracy of Open IE approaches, more recent work is based on the idea of incorporating a sentence re-structuring stage whose goal is to transform complex sentences, where relations are spread over several clauses or presented in a non-canonical form, into a set of syntactically simplified independent clauses that are easy to segment into Open IE tuples. An example of such a paraphrase-based Open IE approach is ClausIE (Del Corro and Gemulla, 2013), which exploits linguistic knowledge about the grammar of the English language to map the dependency relations of an input sentence to clause constituents. In that way, a set of coherent clauses presenting a simple linguistic structure is derived from the input. Then, the type of each clause is determined by combining knowledge of properties of verbs (with the help of domain-independent lexica) with knowledge about the structure of input clauses. Finally, based on its type, one or more propositions are generated from each clause, each representing different pieces of information. The basic set of patterns used for this task is shown in Table 1.

In the same vein, Schmidek and Barbosa (2014) propose a strategy to break down structurally com-

	Pattern	Clause type	Example	Derived clauses
S_1:	SV_i	SV	AE died.	(AE, died)
S_2:	SV_eA	SVA	AE remained in Princeton.	(AE, remained, in Princeton)
S_3:	SV_cC	SVC	AE is smart.	(AE, is, smart)
S_4:	$SV_{mt}O$	SVO	AE has won the Nobel Prize.	(AE, has won, the Nobel Prize)
S_5:	$SV_{dt}O_iO$	SVOO	RSAS gave AE the Nobel Prize.	(RSAS, gave, AE, the Nobel Prize)
S_6:	$SV_{ct}OA$	SVOA	The doorman showed AE to his office.	(The doorman, showed, AE, to his office)
S_7:	$SV_{ct}OC$	SVOC	AE declared the meeting open.	(AE, declared, the meeting, open)

Table 1: Basic patterns for proposition extraction (Del Corro and Gemulla, 2013). S: Subject, V: Verb, C: Complement, O: Direct object, A: Adverbial, V_i: Intransitive verb, V_c: Copular verb, V_e: Extended-copular verb, V_{mt}: Monotransitive verb, V_{dt}: Ditransitive verb, V_{ct}: Complex-transitive verb

plex sentences into simpler ones by decomposing the original sentence into its basic building blocks via chunking. The dependencies of each two chunks are then determined (one of "connected", "disconnected" or "dependent") using either manually defined rules over dependency paths between words in different chunks or a Naive Bayes classifier trained on shallow features, such as POS tags and the distance between chunks. Depending on their relationships, chunks are combined into simplified sentences, upon which the extraction process is carried out.

Angeli et al. (2015) present Stanford Open IE, an approach in which a classifier is learned for splitting a sentence into a set of logically entailed shorter utterances by recursively traversing its dependency tree and predicting at each step whether an edge should yield an independent clause or not. In order to increase the usefulness of the extracted propositions for downstream applications, each self-contained clause is then maximally shortened by running natural logic inference over it. In the end, a small set of 14 hand-crafted patterns are used to extract a predicate-argument triple from each utterance. An illustration of this approach is depicted in Figure 4.

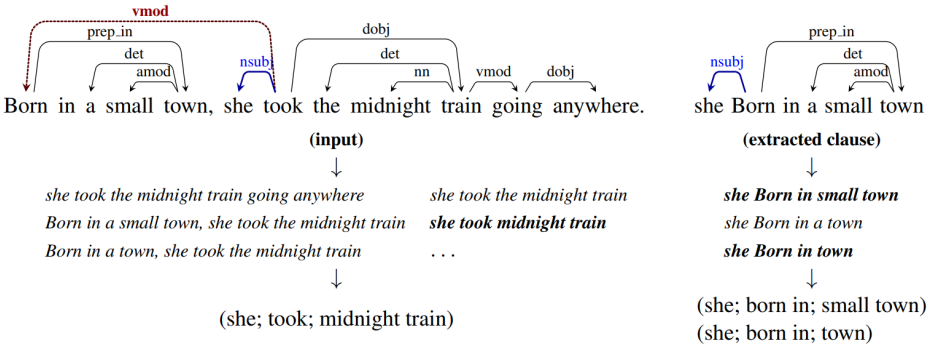


Figure 4: An illustration of Stanford Open IE's approach. From left to right, a sentence yields a number of independent clauses. From top to bottom, each clause produces a set of entailed shorter utterances, and segments the ones which match an atomic pattern into a relational triple (Angeli et al., 2015).

2.4 Systems Capturing Inter-Proposition Relationships

Aforementioned Open IE systems lack the expressiveness needed for a proper interpretation of complex assertions, since they ignore the context under which a proposition is complete and correct. Thus, they do not distinguish between information asserted in a sentence and information that is only hypothetical or conditionally true. For example, extracting the relational tuple ⟨*the earth*; *is the center of*; *the universe*⟩ from the sentence *"Early scientists believed that the earth is the center of the universe."* would be inappropriate, since the input is not asserting it, but only noting that is was believed by early scientists (Mausam, 2016). To properly handle such cases, OLLIE attempts a first solution by additionally extracting an attribution context, denoting a proposition that is reported or claimed by some entity:

(⟨*the earth*; *be the center of*; *the university*⟩;
`AttributedTo` *believe*; *Early astronomers*)

In that way, it extends the default Open IE representation of $\langle arg_1, rel, arg_2 \rangle$ with an extra field. Besides, OLLIE pays attention to clausal modifiers, such as:

(⟨*Romney*; *will be elected*; *President*⟩;
`ClausalModifier` *if*; *he wins five key states*)

Both types of modifiers are identified by matching patterns with the dependency parse of the sentence. Clausal modifiers are determined by an adverbial-clause edge and filtered lexically (the first word of the clause must match a list of cue terms, e.g. *if*, *when*, or *although*), while attribution modifiers are identified by a clausal-complement edge whose context verb must match one of the terms given in VerbNet's list of common verbs of communication and cognition (Mausam et al., 2012). A similar output is produced by OLLIES's successor OPENIE4 (Mausam, 2016), which combines SRLIE (Christensen et al., 2010) and RELNOUN (Pal and Mausam, 2016). Former is a system that converts the output of a SRL system into an Open IE extraction by treating the verb as the relational phrase, while taking its role-labeled arguments as the Open IE argument phrases related to the relation. Latter, in contrast, represents a rule-based Open IE system that extracts noun-mediated relations, thereby paying special attention to demonyms and compound relational nouns. In addition, OPENIE4 marks temporal and spatial arguments by assigning them a T or S label, respectively. Lately, its successor OPENIE 5.0 was released[1]. It integrates BONIE (Saha et al., 2017) and OpenIEListExtractor[2]. While the former focuses on extracting tuples where one of the arguments is a number or a quantity-unit phrase, the latter targets the extraction of propositions from conjunctive sentences.

Similar to OLLIE, Bast and Haussmann (2013), who explore the use of contextual sentence decomposition (CSD) for Open IE, advocate to further specify propositions with information on which they depend. Their system CSD-IE is based on the idea of paraphrasing-based approaches described in section 2.3. Using a set of hand-crafted rules over the output of a constituent parser, a sentence is first split into sub-sequences that semantically belong together, forming so-called "contexts". Each such context now contains a separate fact, yet it is often dependent on surrounding contexts. In order to preserve such inter-proposition relationships, tuples may contain references to other propositions. However, as opposed to OLLIE, where additional contextual modifiers are directly assigned to the corresponding relational tuples, Bast and Haussmann (2013) represent contextual information in the form of separate, linked propositions. To do so, each extraction is given a unique identifier that can be used in the argument position of an extraction for a later substitution with the corresponding fact by a downstream application. An example for an attribution is shown below (Bast and Haussmann, 2013):

#1: ⟨*The Embassy*; *said*; *that #2*⟩

#2: ⟨*6,700 Americans*; *were*; *in Pakistan.*⟩

Another current approach that captures inter-proposition relationships is proposed by Bhutani et al. (2016), who present a nested representation for Open IE that is able to capture high-level dependencies, allowing for a more accurate representation of the meaning of an input sentence. Their system NESTIE uses bootstrapping over a dataset for textual entailment to learn both binary and nested triple representations for n-ary relations over dependency parse trees. These patterns can take on the form of binary triples $\langle arg_1; rel; arg_2 \rangle$ or nested triples such as $\langle \langle arg_1; rel; arg_2 \rangle; rel_2; arg_3 \rangle$ for n-ary relations. Using a set of manually defined rules, contextual links between extracted propositions are inferred from the dependency parse in order to generate a nested representation of assertions that are complete and closer in meaning to the original statement. Similar to OLLIE, contextual links are identified as clausal complements, conditionals and relation modifiers. Linked propositions are represented by arguments that refer to the corresponding propositions using identifiers, e.g. (Bhutani et al., 2016):

#1: ⟨*body*; *appeared to have been thrown*; ∅⟩

#2: ⟨#1; *from*; *vehicle*⟩

or another example based on the following sentence:

[1] `https://github.com/dair-iitd/OpenIE-standalone`
[2] `https://github.com/swarnaHub/OpenIEListExtractor`

"*After giving 5,000 people a second chance at life, doctors are celebrating the 25th anniversary of Britain's first heart transplant.*"

#1: ⟨*doctors*; *are celebrating*; *the 25th anniversary of Britain's first heart transplant*⟩

#2: ⟨*doctors*; *giving*; *second chance at life*⟩

#3: ⟨#1; *after*; #2⟩

MinIE (Gashteovski et al., 2017), another recent Open IE system, is built on top of ClausIE, a system that was found to often produce overly specific extractions. Such overly specific constituents that combine multiple, potentially semantically unrelated propositions in a single relational or argument phrase generally hurt the performance of downstream semantic applications, such as question answering or textual entailment. In fact, those approaches benefit from extractions that are as compact as possible. Therefore, MinIE aims to minimize both relational and argument phrases by identifying and removing parts that are considered overly specific. For this purpose, MinIE provides four different minimization modes which differ in their aggressiveness, thus allowing to control the trade-off between precision and recall. Moreover, it semantically annotates extractions with information about polarity, modality, attribution and quantities instead of directly representing it in the actual extractions, as the following example shows (Gashteovski et al., 2017):

"*Pinocchio believes that the hero Superman was not actually born on beautiful Krypton.*"

#1: ⟨*Superman*; *was born actually on*; *beautiful Krypton*⟩
Annotation: factuality, (- [not], certainty), attribution (Pinocchio, +, possibility [believes])

#2: ⟨*Superman*; *was born on*; *beautiful Krypton*⟩
Annotation: factuality, (- [not], certainty), attribution (Pinocchio, +, possibility [believes])

#3: ⟨*Superman*; *"is"*; *hero*⟩
Annotation: factuality, (+, certainty)

with + and - signifying positive and negative polarity, respectively.

In that way, the output generated by MinIE is further reduced to its core constituents, producing maximally shortened, semantically enriched extractions.

To further enhance the expressiveness of extracted propositions and sustain their interpretability in downstream artificial intelligence tasks, Cetto et al. (2018) propose Graphene, an Open IE framework that uses a set of hand-crafted simplification rules to transform complex natural language sentences into clean, compact structures by removing clauses and phrases that present no central information from the input and converting them into stand-alone sentences. In that way, a source sentence is transformed into a hierarchical representation in the form of core facts and accompanying contexts (Niklaus et al., 2016). In addition, inspired by the work on Rhetorical Structure Theory (Mann and Thompson, 1988), a set of syntactic and lexical patterns is used to identify the rhetorical relations by which core sentences and their associated contexts are connected in order to preserve their semantic relationships and return a set of semantically typed and interconnected relational tuples (see extractions (15-17) in Figure 1). Graphene's extraction workflow is illustrated in Figure 5.

3 Evaluation

Though a multitude of systems for Open IE have been developed over the last decade, a clear formal specification of what constitutes a valid relational tuple is still missing. This lack of a well-defined, generally accepted task definition prevented the creation of an established, large-scale annotated corpus serving as a gold standard dataset for an objective and reproducible cross-system comparison. As a consequence, to date, Open IE systems were predominantly evaluated by hand on small-scale corpora that consist of only a few hundred sentences, thereby ignoring one of the fundamental goals of Open IE: scalability to large amounts of text. Moreover, none of the datasets that were used for assessing the

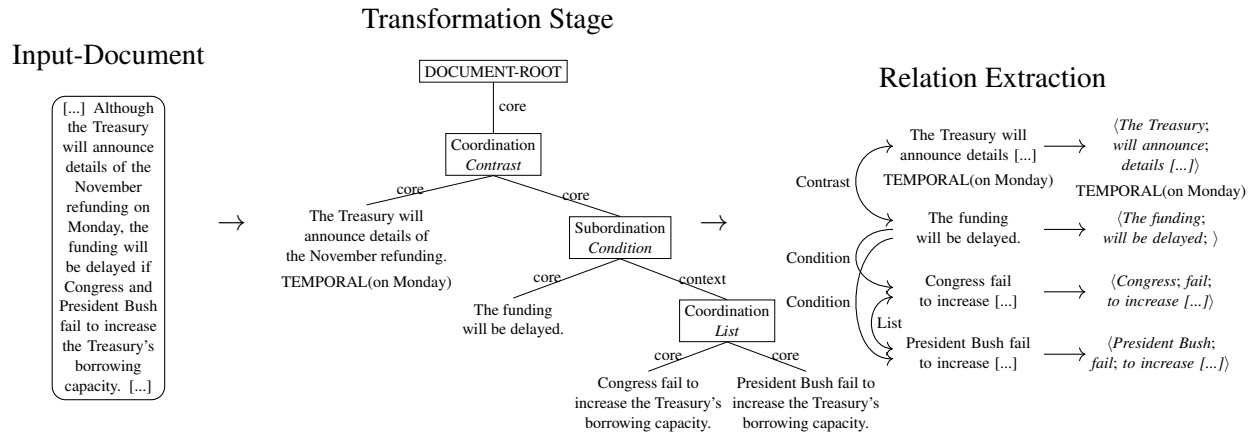


Figure 5: Graphene's extraction workflow for an example sentence (Cetto et al., 2018).

performance of different systems is widely agreed upon. As can be seen in Table 2, the corpora compiled by Del Corro and Gemulla (2013), Xu et al. (2013), Fader et al. (2011) and Banko et al. (2007) are occasionally re-used. However, new datasets are still collected, hindering a fair comparison of the proposed approaches. Besides, although Open IE methods are targeted at being domain independent and able to cope with heterogeneous datasets, the corpora used in the evaluation process are restricted to the news, Wikipedia and Web domains for the most part. Accordingly, no clear statement about the portability of the approaches to various genres of text is possible. In addition, most evaluation procedures described in the literature focus on precision-oriented metrics, while either completely ignoring recall or using some kind of proxy, such as yield, i.e. the total number of extractions labeled as correct, or coverage, i.e. the percentage of text from the input that is contained in at least one of the extractions. Hence, the absence of a standard evaluation procedure makes it hard to replicate and compare the performance of different Open IE systems. Table 2 provides a detailed overview of both the datasets and measures used for intrinsically evaluating the various approaches described above, while Table 3 shows the tasks that were used for an extrinsic evaluation of a small set of Open IE systems.

In order to address aforementioned difficulties, Stanovsky and Dagan (2016) recently made a first attempt in standardizing the Open IE evaluation by providing a large gold benchmark corpus. It is based on a set of consensual guiding principles that underly most Open IE approaches proposed so far, as they have identified. Those principles cover the core aspects of the task of Open IE, allowing for a clearer formulation of the problem to be solved. The three key features to consider are the following:

Assertedness. The assertedness principle states that extracted propositions should be asserted by the original sentence. Usually, instead of inferring propositions out of implied statements, e.g. the tuple ⟨*Sam*; *convinced*; *John*⟩ out of ⟨*Sam*; *succeeded in convincing*; *John*⟩, Open IE systems tend to extract the full relational phrase (⟨*Sam*; *succeeded in convincing*; *John*⟩), incorporating matrix verbs ("*succeeded*") and other elements, such as negotiations or modals (e.g. ⟨*John*; *could not join*; *the band*⟩).

Minimal Propositions. In order to serve for semantic tasks, it is beneficial for Open IE systems to extract compact, self-contained propositions that do not combine several unrelated facts. Therefore, systems should aim to generate valid propositions with minimal spans for both relation and argument slots, while preserving the meaning of the input. As an example, the coordination in the sentence "*Bell distributes electronic and building products*" should ideally yield the two propositions: ⟨*Bell*; *distributes*; *electronic products*⟩ and ⟨*Bell*; *distributes*; *building products*⟩.

Completeness and Open Lexicon. The completeness and open lexicon principle aims to extract all relations that are asserted in the input text. This principle was one of the fundamental ideas that have been introduced in the work of Banko et al. (2007) together with the Open IE terminology. In their work, the Open IE task was defined as a domain-independent task which extracts all possible

System	Baselines	# sentences and domain	Metrics
TEXTRUNNER	KNOWITALL (Etzioni et al., 2004)	400 Web	% correct extractions
WOE	TEXTRUNNER	300 news 300 Wikipedia 300 Web	precision-recall curve
OLLIE	REVERB WOE^{parse}	300 news (from WOE) 300 Wikipedia (from WOE) 300 biology	precision-yield curve
REVERB	TEXTRUNNER WOE^{pos} WOE^{parse}	500 Web	precision-recall curve
KRAKEN	REVERB	500 Web (from REVERB)	• precision • completeness • # facts extracted per sentence
EXEMPLAR	REVERB OLLIE SONEX (Merhav et al., 2012) PATTY (Nakashole et al., 2012) TREEKERNEL (Xu et al., 2013) SWIRL (Surdeanu et al., 2003) LUND (Johansson and Nugues, 2008)	500 Web (from TEXTRUNNER) 500 news 100 news (from TREEKERNEL) 222 news	∗ binary: • precision • recall • F_1-score ∗ n-ary: • precision over arguments • recall over arguments
PredPatt (Zhang et al., 2017)	OLLIE ClausIE Stanford Open IE OPENIE4	13k Web 36k news	precision-recall curve
ClausIE	TEXTRUNNER WOE^{parse} REVERB OLLIE KRAKEN	500 Web (from REVERB) 200 Wikipedia 200 news	• precision-yield curve • % correct extractions
Schmidek and Barbosa (2014)	REVERB EXEMPLAR	500 Web (from TEXTRUNNER) 500 news 100 news (from TREEKERNEL)	• precision • recall • F_1-score • time per sentence before and after sentence re-structuring
OPENIE4	REVERB OLLIE	not reported	• precision • yield
CSD-IE	REVERB OLLIE ClausIE	200 Wikipedia (from ClausIE) 200 news (from ClausIE)	• % triples labeled accurate • % correct triples labeled minimal • coverage (% text contained in at least one triple) • average triple length
NESTIE	REVERB OLLIE ClausIE	200 Wikipedia (from ClausIE) 200 news (from ClausIE)	• correctness (0/1) • minimality (0/1) • informativeness (0-5)
MINIE	ClausIE OLLIE Stanford Open IE	10k news (from Sandhaus (2008)) 200 news (from ClausIE) 200 Wikipedia (from ClausIE)	• # extractions • # non-redundant extractions • recall • factual precision • attribution precision • mean word count per triple (proxy for minimality)
Graphene	OLLIE REVERB PROPS ClausIE Stanford Open IE OPENIE4	3,200 Wikipedia and news (Stanovsky and Dagan, 2016)	precision-recall curve

Table 2: Comparison of the intrinsic evaluation approaches applied by the different Open IE systems.

System	Task
PROPS	MCTest comprehension task (Richardson et al., 2013)
Stanford Open IE	TAC KBP Slot Filling Challenge (Surdeanu, 2013)

Table 3: Extrinsic evaluation approaches.

relations from heterogeneous corpora, instead of only extracting a set of pre-specified classes of relations. The majority of current Open IE systems realize this challenge by considering all possible verbs as potential relations. Accordingly, their scope is often limited to the extraction of verbal predicates, while ignoring relations mediated by more complex syntactic constructs, such as nouns or adjectives.

Realizing that above-mentioned requirements are subsumed by the task of Question Answering (QA) driven Semantic Role Labeling (SRL) (He et al., 2015), Stanovsky and Dagan (2016) converted the annotations of a QA-SRL dataset to an Open IE corpus, resulting in more than 10,000 extractions over 3,200 sentences from Wikipedia and the Wall Street Journal.

In addition, Schneider et al. (2017) presented RelVis, another benchmark framework for Open IE that allows for a large-scale comparative analysis of Open IE approaches. Besides Stanovsky and Dagan (2016)'s benchmark suite, it comprises the n-ary news dataset proposed in Mesquita et al. (2013), Banko et al. (2007)'s Web corpus and the Penn sentences from Xu et al. (2013). Similar to the toolkit proposed in Stanovsky and Dagan (2016), RelVis supports a quantitative evaluation of the performance of Open IE systems in terms of precision, recall and F_2-score. In addition, it facilitates a manual qualitative error analysis. For this purpose, six common error classes are distinguished to which inaccurate extractions can be assigned: (1) wrong boundaries, where the relational or argument phrase is either too long or too small; (2) redundant extraction, where the proposition asserted in an extraction is already expressed in another extraction; (3) uninformative extraction, where critical information is omitted; (4) missing extraction, i.e. a false negative, where either a relation is not detected by the system or the argument-finding heuristics choose the wrong arguments or none argument at all; (5) wrong extraction, where no meaningful interpretation of the proposition is possible; and (6) out of scope extraction, where a system yields a correct extraction that was not recognized by the authors of the gold dataset.

4 Open Research Questions

Even after more than one decade of research in the area of Open IE, there are still a lot of open research questions. As shown in Section 3, to date, there is only very little work on evaluating and comparing results among different Open IE systems in a large-scale, objective and reproducible fashion. Instead, most approaches use proprietary datasets over small, domain-dependent corpora. Stanovsky and Dagan (2016) and Schneider et al. (2017) recently made the first move to standardize the evaluation of Open IE by proposing benchmark frameworks that operate on a larger scale. However, apart from Cetto et al. (2018), neither benchmarking toolkit has been adopted yet in the Open IE community.

Moreover, most Open IE approaches focus on the English language, leaving aside other languages. Notable exceptions are Falke et al. (2016), who investigate the portability of the PROPS system to German, and Gamallo and Garcia (2015), who propose a multilingual system covering English, Spanish, Portuguese and Galician. Besides, due to the use of language-agnostic patterns over UD parses, PredPatt works across languages, though its performance was only evaluated on English texts. Hence, the applicability and transferability of previously proposed Open IE approaches to other languages than English represents an interesting direction for future work.

Finally, the problem of canonicalizing relational phrases and arguments has been hardly addressed so far. However, normalizing extractions would be highly beneficial for downstream semantic tasks, such as textual entailment or knowledge base population. Besides, coreference resolution, another field that has largely been ignored to date, may assist those tasks with the interpretation of the extracted propositions.

5 Conclusion

We presented an overview of the various methods that were proposed to solve the task of Open IE. We classified them into the four categories of learning-based, rule-based, clause-based systems and approaches capturing inter-proposition relationships, thereby showing their evolution over time, as well as the specific problems they tackle. Moreover, we described the approaches that were used to assess the performance of the proposed Open IE systems, while depicting the gaps in the evaluation procedures that are commonly applied to date. Finally, we identified directions for future work.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, DL '00, pages 85–94, New York, NY, USA. ACM.

Alan Akbik and Alexander Löser, 2012. *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, chapter KrakeN: N-ary Facts in Open Information Extraction, pages 52–56. Association for Computational Linguistics.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China, July. Association for Computational Linguistics.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 2670–2676, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Hannah Bast and Elmar Haussmann. 2013. Open information extraction via contextual sentence decomposition. In *Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on*, pages 154–159. IEEE.

Nikita Bhutani, H. V. Jagadish, and Dragomir R. Radev. 2016. Nested propositions in open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 55–64.

Sergey Brin. 1999. Extracting patterns and relations from the world wide web. In *Selected Papers from the International Workshop on The World Wide Web and Databases*, WebDB '98, pages 172–183, London, UK, UK. Springer-Verlag.

Matthias Cetto, Christina Niklaus, André Freitas, and Siegfried Handschuh. 2018. Graphene: Semantically-linked propositions in open information extraction. In *Prooceedings of COLING 2018. To appear.*

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2010. Semantic role labeling for open information extraction. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, FAM-LbR '10, pages 52–60, Stroudsburg, PA, USA. Association for Computational Linguistics.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: Clause-based open information extraction. In *Proceedings of the 22Nd International Conference on World Wide Web*, pages 355–366, New York, NY, USA. ACM.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 100–110, New York, NY, USA. ACM.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Tobias Falke, Gabriel Stanovsky, Iryna Gurevych, and Ido Dagan. 2016. Porting an open information extraction system from english to german. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 892–898. Association for Computational Linguistics, November.

Pablo Gamallo and Marcos Garcia. 2015. Multilingual open information extraction. In Francisco Pereira, Penousal Machado, Ernesto Costa, and Amílcar Cardoso, editors, *Progress in Artificial Intelligence*, pages 711–722, Cham. Springer International Publishing.

Kiril Gashteovski, Rainer Gemulla, and Luciano del Corro. 2017. Minie: minimizing facts in open information extraction. In *The Conference on Empirical Methods in Natural Language Processing - proceedings of System Demonstrations : September 9-11, 2017, Copenhagen, Denmark : EMNLP 2017*, pages 2620–2630, Stroudsburg, PA. Association for Computational Linguistics.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 643–653.

Richard Johansson and Pierre Nugues. 2008. Dependency-based semantic role labeling of propbank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 69–78. Association for Computational Linguistics.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal stanford dependencies: a cross-linguistic typology. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may. European Language Resources Association (ELRA).

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534, Jeju Island, Korea, July. Association for Computational Linguistics.

Mausam. 2016. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 4074–4077.

Yuval Merhav, Filipe Mesquita, Denilson Barbosa, Wai Gen Yee, and Ophir Frieder. 2012. Extracting information networks from the blogosphere. *ACM Transactions on the Web*, 6(3):11:1–11:33, September.

Filipe Mesquita, Jordan Schmidek, and Denilson Barbosa. 2013. Effectiveness and efficiency of open relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 447–457. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011. Association for Computational Linguistics.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Patty: A taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1135–1145, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christina Niklaus, Bernhard Bermeitinger, Siegfried Handschuh, and André Freitas. 2016. A sentence simplification system for improving relation extraction. In *Prooceedings of COLING 2016: System Demonstrations, The 26th International Conference on Computational Linguistics, Osaka, Japan, December 11-16, 2016*, pages 170–174.

Harinder Pal and Mausam. 2016. Demonyms and compound relational nouns in nominal open ie. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 35–39. Association for Computational Linguistics.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, AAAI '99/IAAI '99, pages 474–479, Menlo Park, CA, USA. American Association for Artificial Intelligence.

Swarnadeep Saha, Harinder Pal, and Mausam. 2017. Bootstrapping for numerical open ie. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 317–323. Association for Computational Linguistics.

Evan Sandhaus. 2008. The New York Times Annotated Corpus. *Linguistic Data Consortium, Philadelphia*, 6(12).

Jordan Schmidek and Denilson Barbosa. 2014. Improving open relation extraction via sentence re-structuring. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA).

Rudolf Schneider, Tom Oberhauser, Tobias Klatt, Felix A. Gers, and Alexander Löser. 2017. Analysing errors of open information extraction systems. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 11–18. Association for Computational Linguistics.

Gabriel Stanovsky and Ido Dagan. 2016. Creating a large benchmark for open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page (to appear), Austin, Texas, November. Association for Computational Linguistics.

Gabriel Stanovsky, Jessica Ficler, Ido Dagan, and Yoav Goldberg. 2016. Getting more out of syntax with props. *CoRR*, abs/1603.01648.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 8–15. Association for Computational Linguistics.

Mihai Surdeanu. 2013. Overview of the tac2013 knowledge base population evaluation: English slot filling and temporal slot filling. In *TAC*.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on universal dependencies. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1713–1723.

Fei Wu and S. Daniel Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127. Association for Computational Linguistics.

Ying Xu, Mi-Young Kim, Kevin Quinn, Randy Goebel, and Denilson Barbosa. 2013. Open information extraction with tree kernels. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 868–877, Atlanta, Georgia, June. Association for Computational Linguistics.

Mohamed Yahya, Steven Whang, Rahul Gupta, and Alon Halevy. 2014. Renoun: Fact extraction for nominal attributes. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 325–335. Association for Computational Linguistics.

Sheng Zhang, Rachel Rudinger, and Ben Van Durme. 2017. An evaluation of predpatt and open ie via stage 1 semantic role labeling. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS)*, Montpellier, France, September.

Design Challenges and Misconceptions in Neural Sequence Labeling

Jie Yang, Shuailong Liang, Yue Zhang
Singapore University of Technology and Design
{jie_yang, shuailong_liang}@mymail.sutd.edu.sg
yue_zhang@sutd.edu.sg

Abstract

We investigate the design challenges of constructing effective and efficient neural sequence labeling systems, by reproducing twelve neural sequence labeling models, which include most of the state-of-the-art structures, and conduct a systematic model comparison on three benchmarks (i.e. NER, Chunking, and POS tagging). Misconceptions and inconsistent conclusions in existing literature are examined and clarified under statistical experiments. In the comparison and analysis process, we reach several practical conclusions which can be useful to practitioners.

1 Introduction

Sequence labeling models have been used for fundamental NLP tasks such as POS tagging, chunking and named entity recognition (NER). Traditional work uses statistical approaches such as Hidden Markov Models (HMM) and Conditional Random Fields (CRF) (Ratinov and Roth, 2009; Passos et al., 2014; Luo et al., 2015) with handcrafted features and task-specific resources. With advances in deep learning, neural models have given state-of-the-art results on many sequence labeling tasks (Ling et al., 2015; Lample et al., 2016; Ma and Hovy, 2016). Words and characters are encoded in distributed representations (Mikolov et al., 2013) and sentence-level features are learned automatically during end-to-end training. Many existing state-of-the-art neural sequence labeling models utilize word-level Long Short-Term Memory (LSTM) structures to represent global sequence information and a CRF layer to capture dependencies between neighboring labels (Huang et al., 2015; Lample et al., 2016; Ma and Hovy, 2016; Peters et al., 2017). As an alternative, Convolution Neural Network (CNN) (LeCun et al., 1989) has also been used for its ability of parallel computing, leading to an efficient training and decoding process.

Despite them being dominant in the research literature, reproducing published results for neural models can be challenging, even if the codes are available open source. For example, Reimers and Gurevych (2017b) conduct a large number of experiments using the code of Ma and Hovy (2016), but cannot obtain comparable results as reported in the paper. Liu et al. (2018) report lower average F-scores on NER when reproducing the structure of Lample et al. (2016), and on POS tagging when reproducing Ma and Hovy (2016). Most literature compares results with others by citing the scores directly (Huang et al., 2015; Lample et al., 2016) without re-implementing them under the same setting, resulting in less persuasiveness on the advantage of their models. In addition, conclusions from different reports can be contradictory. For example, most work observes that stochastic gradient descent (SGD) gives best performance on NER task (Chiu and Nichols, 2016; Lample et al., 2016; Ma and Hovy, 2016), while Reimers and Gurevych (2017b) report that SGD is the worst optimizer on the same datasets.

The comparison between different deep neural models is challenging due to sensitivity on experimental settings. We list six inconsistent configurations in literature, which lead to difficulties for fair comparison.
• **Dataset**. Most work reports sequence labeling results on both CoNLL 2003 English NER (Tjong Kim Sang and De Meulder, 2003) and PTB POS (Marcus et al., 1993) datasets (Collobert et al., 2011; Huang et al., 2015; Ma and Hovy, 2016). Ling et al. (2015) give results only on POS dataset, while some

Proceedings of the 27th International Conference on Computational Linguistics, pages 3879–3889
Santa Fe, New Mexico, USA, August 20-26, 2018.

Models	Word LSTM+CRF	Word LSTM	Word CNN+CRF	Word CNN
No Char	Huang et al. (2015)* Lample et al. (2016) Strubell et al. (2017)*	Ma and Hovy (2016) Strubell et al. (2017)*	Collobert et al. (2011)* dos Santos et al. (2015) Strubell et al. (2017)*	Strubell et al. (2017)*
Char LSTM	Lample et al. (2016) Rei (2017) Liu et al. (2018)	Lample et al. (2016)	No existing work	No existing work
Char CNN	Ma and Hovy (2016) Chiu and Nichols (2016)* Peters et al. (2017)	Ma and Hovy (2016)	dos Santos et al. (2015)	Santos and Zadrozny (2014)

Table 1: Neural sequence labeling models in literatures. * represents using handcrafted neural features.

papers (Chiu and Nichols, 2016; Lample et al., 2016; Strubell et al., 2017) report results on the NER dataset only. dos Santos et al. (2015) conducts experiments on NER for Portuguese and Spanish.

Most work uses the development set to select hyperparameters (Lample et al., 2016; Ma and Hovy, 2016), while others add development set into training set (Chiu and Nichols, 2016; Peters et al., 2017). Reimers and Gurevych (2017b) use a smaller dataset (13862 vs 14987 sentences). Different from Ma and Hovy (2016) and Liu et al. (2018), Huang et al. (2015) choose a different data split on the POS dataset. Liu et al. (2018) and Hashimoto et al. (2017) use different development sets for chunking.

• **Preprocessing**. A typical data preprocessing step is to normize digit characters (Chiu and Nichols, 2016; Lample et al., 2016; Yang et al., 2016; Strubell et al., 2017). Reimers and Gurevych (2017b) use fine-grained representations for less frequent words. Ma and Hovy (2016) do not use preprocessing.

• **Features**. Strubell et al. (2017) and Chiu and Nichols (2016) apply word spelling features and Huang et al. (2015) further integrate context features. Collobert et al. (2011) and Huang et al. (2015) use neural features to represent external gazetteer information. Besides, Lample et al. (2016) and Ma and Hovy (2016) use end-to-end structure without handcrafted features.

• **Hyperparameters** including learning rate, dropout rate (Srivastava et al., 2014), number of layers, hidden size etc. can strongly affect the model performance. Chiu and Nichols (2016) search for the hyperparameters for each task and show that the system performance is sensitive to the choice of hyperparameters. However, existing models use different parameter settings, which affects the fair comparison.

• **Evaluation**. Some literature reports results using mean and standard deviation under different random seeds (Chiu and Nichols, 2016; Peters et al., 2017; Liu et al., 2018). Others report the best result among different trials (Ma and Hovy, 2016), which cannot be compared directly.

• **Hardware environment** can also affect system accuracy. Liu et al. (2018) observes that the system gives better accuracy on NER task when trained using GPU as compared to using CPU. Besides, the running speeds are highly affected by the hardware environment.

To address the above concerns, we systematically analyze neural sequence labeling models on three benchmarks: CoNLL 2003 NER (Tjong Kim Sang and De Meulder, 2003), CoNLL 2000 chunking (Tjong Kim Sang and Buchholz, 2000) and PTB POS tagging (Marcus et al., 1993). Table 1 shows a summary of the models we investigate, which can be categorized under three settings: (i) character sequence representations ; (ii) word sequence representations; (iii) inference layer. Although various combinations of these three settings have been proposed in the literature, others have not been examined. We compare all models in Table 1, which includes most state-of-the-art methods. To make fair comparisons, we build a unified framework[1] to reproduce the twelve neural sequence labeling architectures in Table 1. Experiments show that our framework gives comparable or better results on reproducing existing works, showing the practicability and reliability of our analysis for practitioners. The detailed comparison and analysis show that (i) Character information provides a significant improvement on accuracy; (ii) Word-based LSTM models outperforms CNN models in most cases; (iii) CRF can improve model accuracy on NER and chunking but does not on POS tagging. Our framework is based on PyTorch with batched implementation, which is highly efficient, facilitating quick configurations for new tasks.

[1]Our code has been released at `https://github.com/jiesutd/NCRFpp`.

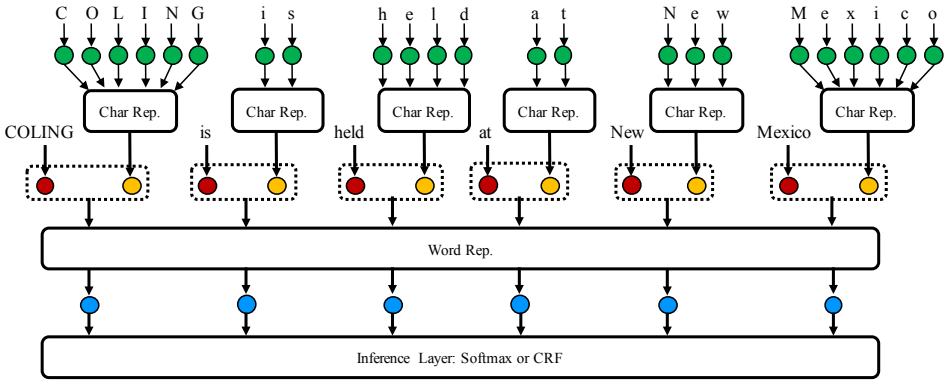


Figure 1: Neural sequence labeling architecture for sentence "COLING is held at New Mexico". Green, red, yellow and blue circles represent character embeddings, word embeddings, character sequence representations and word sequence representations, respectively.

2 Related Work

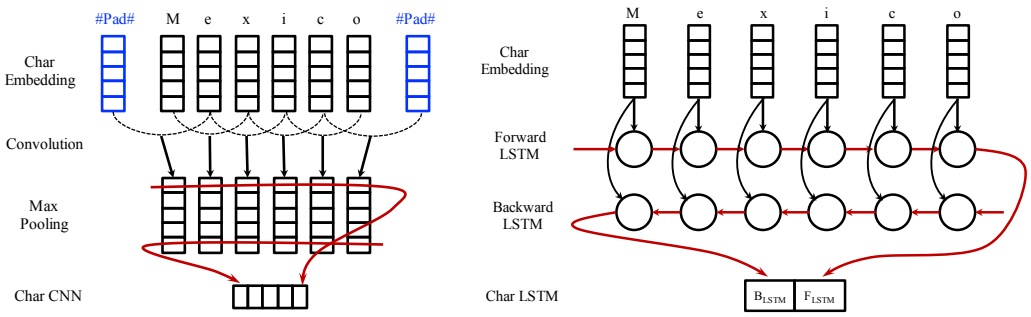
Collobert et al. (2011) proposed a seminal neural architecture for sequence labeling. It captures word sequence information with a one-layer CNN based on pretrained word embeddings and handcrafted neural features, followed with a CRF output layer. dos Santos et al. (2015) extended this model by integrating character-level CNN features. Strubell et al. (2017) built a deeper dilated CNN architecture to capture larger local features. Hammerton (2003) was the first to exploit LSTM for sequence labeling. Huang et al. (2015) built a BiLSTM-CRF structure, which has been extended by adding character-level LSTM (Lample et al., 2016; Liu et al., 2018), GRU (Yang et al., 2016), and CNN (Chiu and Nichols, 2016; Ma and Hovy, 2016) features. Yang et al. (2017a) proposed a neural reranking model to improve NER models. These models achieve state-of-the-art results in the literature.

Reimers and Gurevych (2017b) compared several word-based LSTM models for several sequence labeling tasks, reporting the score distributions over multiple runs rather than single value. They investigated the influence of various hyperparameters and configurations. Our work is similar in comparing different neural architectures under unified settings, but differs in four main aspects: 1) Their experiments are based on a BiLSTM with handcrafted word features, while our experiments are based on end-to-end neural models without human knowledge. 2) Their system gives relatively low performances on standard benchmarks[2], while ours can give comparable or better results with state-of-the-art models, rendering our observations more informative for practitioners. 3) Our findings are more consistent with most previous work on configurations such as usefulness of character information (Lample et al., 2016; Ma and Hovy, 2016), optimizer (Chiu and Nichols, 2016; Lample et al., 2016; Ma and Hovy, 2016) and tag scheme (Ratinov and Roth, 2009; Dai et al., 2015). In contrast, many results of Reimers and Gurevych (2017b) contradict existing reports. 4) We conduct a wider range of comparison for word sequence representations, including all combinations of character CNN/LSTM and word CNN/LSTM structures, while Reimers and Gurevych (2017b) studied the word LSTM models only.

3 Neural Sequence Labeling Models

Our neural sequence labeling framework contains three layers, i.e., a character sequence representation layer, a word sequence representation layer and an inference layer, as shown in Figure 1. Character information has been proven to be critical for sequence labeling tasks (Chiu and Nichols, 2016; Lample et al., 2016; Ma and Hovy, 2016), with LSTM and CNN being used to model character sequence information ("Char Rep."). Similarly, on the word level, LSTM or CNN structures can be leveraged to capture long-term information or local features ("Word Rep."), respectively. Subsequently, the inference layer assigns labels to each word using the hidden states of word sequence representations.

[2]Based on their detailed experiment report (Reimers and Gurevych, 2017a), the F1-scores on CoNLL 2003 NER task are generally less than 90%, while *state-of-the-art* results are around 91%.



(a) Character CNN. (b) Character LSTM.

Figure 2: Neural character sequence representations.

3.1 Character Sequence Representations

Character features such as prefix, suffix and capitalization can be represented with embeddings through a feature-based lookup table (Collobert et al., 2011; Huang et al., 2015; Strubell et al., 2017), or neural networks without human-defined features (Lample et al., 2016; Ma and Hovy, 2016). In this work, we focus on neural character sequence representations without hand-engineered features.

Character CNN. Using a CNN structure to encode character sequences was firstly proposed by Santos and Zadrozny (2014), and followed by many subsequent investigations (dos Santos et al., 2015; Chiu and Nichols, 2016; Ma and Hovy, 2016). In our experiments, we take the same structure as Ma and Hovy (2016), using one layer CNN structure with *max-pooling* to capture character-level representations. Figure 2(a) shows the CNN structure on representing word "Mexico".

Character LSTM. Shown as Figure 2(b), in order to model the global character sequence information of a word "Mexico", we utilize a bidirectional LSTM on the character sequence of each word and concatenate the *left-to-right* final state F_{LSTM} and the *right-to-left* final state B_{LSTM} as character sequence representations. Liu et al. (2018) applied one bidirectional LSTM for the character sequence over a sentence rather than each word individually. We examined both structures and found that they give comparable accuracies on sequence labeling tasks. We choose Lample et al. (2016)'s structure as its character LSTMs can be calculated in parallel, making the system more efficient.

3.2 Word Sequence Representations

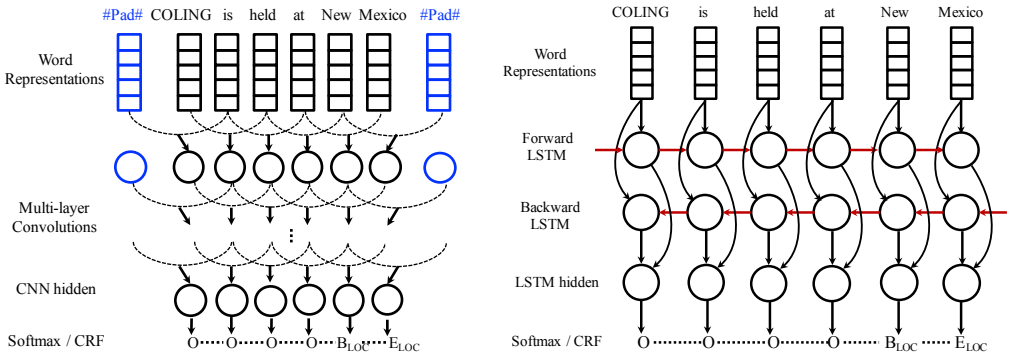
Similar to character sequences in words, we can model word sequence information through LSTM or CNN structures. LSTM has been widely used in sequence labeling (Lample et al., 2016; Ma and Hovy, 2016; Chiu and Nichols, 2016; Huang et al., 2015; Liu et al., 2018). CNN can be much faster than LSTM due to the fact that convolution calculation can be parallel on the input sequence (Collobert et al., 2011; dos Santos et al., 2015; Strubell et al., 2017).

Word CNN. Figure 3(a) shows the multi-layer CNN on word sequence, where words are represented by embeddings. If a character sequence representation layer is used, then word embeddings and character sequence representations are concatenated for word representations. For each CNN layer, a window of size 3 slides along the sequence, extracting local features on the word inputs and a ReLU function (Glorot et al., 2011) is followed. We follow Strubell et al. (2017) by using 4 CNN layers. Batch normalization (Ioffe and Szegedy, 2015) and dropout (Srivastava et al., 2014) are applied following each CNN layer.

Word LSTM. Shown in Figure 3(b), word representations are fed into a forward LSTM and backward LSTM, respectively. The forward LSTM captures the word sequence information from left to right, while the backward LSTM extracts information in a reversed direction. The hidden states of the forward and backward LSTMs are concatenated at each word to give global information of the whole sequence.

3.3 Inference Layer

The inference layer takes the extracted word sequence representations as features and assigns labels to the word sequence. Independent local decoding with a linear layer mapping word sequence representations



(a) Word CNN. (b) Word LSTM.

Figure 3: Neural word sequence representations.

to label vocabulary and performing softmax can be quite effective (Ling et al., 2015), while for tasks that with strong output label dependency, such as NER, CRF is a more appropriate choice. In this work, we examine both softmax and CRF as inference layer on three sequence labeling tasks.

4 Experiments

We investigate the main influencing factors to system accuracy, including the character sequence representations, word sequence representations, inference algorithm, pretrained embeddings, tag scheme, running environment and optimizer; analyzing system performances from the perspective of decoding speed and accuracies on in-vocabulary (IV) and out-of-vocabulary (OOV) entities/chunks/words.

4.1 Settings

Data. The NER dataset has been standardly split in Tjong Kim Sang and De Meulder (2003). It contains four named entity types: PERSON, LOCATION, ORGANIZATION, and MISC. The chunking task is evaluated on CoNLL 2000 shared task (Tjong Kim Sang and Buchholz, 2000). We follow Reimers and Gurevych (2017a) by using sections 15-18 of Wall Street Journal (WSJ) for training, section 19 as development set and section 20 as test set. For the POS tagging task, we use the WSJ portion of Peen Treebank, which has 45 POS tags. Following previous works (Toutanova et al., 2003; Santos and Zadrozny, 2014; Ma and Hovy, 2016; Liu et al., 2018), we adopt the standard splits by using sections 0-18 as training set, sections 19-21 as development set and sections 22-24 as test set. No preprocessing is performed on either dataset except for normalizing digits. The dataset statistics are listed in Table 2.

Hyperparameters. Table 3 shows the hyperparameters used in our experiments, which mostly follow Ma and Hovy (2016), including the learning rate $\eta = 0.015$ for word LSTM models. For word CNN based models, a large η leads to convergence problem. We take $\eta = 0.005$ with more epochs (200) instead. GloVe 100-dimension (Pennington et al., 2014) is used to initialize word embeddings and character embeddings are randomly initialized. We use mini-batch stochastic gradient descent (SGD) with a decayed learning rate to update parameters. For NER and chunking, we the BIOES tag scheme.

Evaluation. Standard precision (P), recall (R) and F1-score (F) are used as the evaluation metrics for NER and chunking; token accuracy is used to evaluate the performance of POS tagger. Development datasets are used to select the optimal model among all epochs, and we report scores of the selected model on the test dataset. To reduce the volatility of the system, we conduct each experiment 5 times under different random seeds, and report the *max, mean,* and *standard deviation* for each model.

4.2 Results

Tables 4, 5 and 6 show the results of the twelve models on NER, chunking and POS datasets, respectively. Existing work has also been listed in the tables for comparison. To simplify the description, we use "CLSTM" and "CCNN" to represent character LSTM and character CNN encoder, respectively. Similarly, "WLSTM" and "WCNN" represent word LSTM and word CNN structure, respectively.

Dataset		Train	Dev	Test	Label
NER	#Sent	14,987	3,644	3,486	17
	#Token	205k	52k	47k	
	#Entity	23,523	5,943	5,654	
chunking	#Sent	8,936	1,844	2,012	42
	#Token	212k	44k	47k	
	#Chunk	107k	22k	24k	
POS	#Sent	38,219	5,527	5,462	45
	#Token	912k	132k	130k	

Table 2: Statistics of datesets.

Parameter	Value	Parameter	Value
char emb size	30	word emb size	100
char hidden	50	word hidden	200
CNN window	3	word CNN layer	4
batch size	10	dropout rate	0.5
L_2 regularization λ	1e-8	learning rate decay	0.05
word LSTM η	0.015	word CNN η	0.005
word LSTM epochs	100	word CNN epochs	200

Table 3: Hyperparameters.

As shown in Table 4, most NER work focuses on WLSTM+CRF structures with different character sequence representations. We re-implement the structure of several reports (Chiu and Nichols, 2016; Ma and Hovy, 2016; Peters et al., 2017), which take the CCNN+WLSTM+CRF architecture. Our reproduced models give slightly better performances. The results of Lample et al. (2016) can be reproduced by our CLSTM+WLSTM+CRF. In most cases, our "Nochar" based models underperform their corresponding prototypes (Huang et al., 2015; Strubell et al., 2017), which utilize the hand-crafted features.

Table 5 shows the results of the chunking task. Peters et al. (2017) give the best reported single model results (95.00±0.08), and our CLSTM+WLSTM+CRF gives a comparable performance (94.93±0.05). We re-implement Zhai et al. (2017)'s model in our Nochar+WLSTM but cannot reproduce their results, this may because that they use grid search for hyperparameter selection. Our Nochar+WCNN+CRF can give comparable results with Collobert et al. (2011), even ours does not include character information.

The results of the POS tagging task is shown in Table 6. The results of Lample et al. (2016), Ma and Hovy (2016) and Yang et al. (2017b) can be reproduced by our CLSTM+WLSTM+CRF and CCNN+WLSTM+CRF models. Our WLSTM based models give better results than WLSTM+CRF based models, this is consistent with the fact that Ling et al. (2015) take CLSTM+WLSTM without CRF layer but achieve the best POS accuracy. Santos and Zadrozny (2014) build a pure CNN structure on both character and word level, which can be reproduced by our CCNN+WCNN models.

Based on above observations, most results in the literature are reproducible. Our implementations can achieve the comparable or better results with state-of-the-art models. We do not fine-tune any hyperparameter to fit the specific task. Results on Table 4, 5 and 6 are all under the same hyperparameters, which demonstrates the generalization ability of our framework.

4.3 Network settings

Character LSTM vs Character CNN. Unlike the observations of Reimers and Gurevych (2017b), in our experiments, character information can significantly ($p < 0.01$)[3] improve sequence labeling models (by comparing the row of Nochar with CLSTM or CCNN on Table 4, 5 and 6), while the difference between CLSTM and CCNN is not significant. In most cases, CLSTM and CCNN can give comparable results under different frameworks and different tasks. CCNN gives the best NER result under the WLSTM+CRF framework, while CLSTM gets better NER results in all other configurations. For chunking and POS tagging, CLSTM consistently outperforms CCNN under all settings, while the difference is statistically insignificant ($p > 0.2$). We conclude that the difference between CLSTM and CCNN is small, which is consistent with the observation of Reimers and Gurevych (2017b).

Word LSTM vs Word CNN. By comparing the performances of WLSTM+CRF, WLSTM with WCNN+CRF, WCNN on the three benchmarks, we conclude that word-based LSTM models are significantly ($p < 0.01$) better than word-based CNN models in most cases. It demonstrates that global word context information is necessary for sequence labeling.

Softmax vs CRF. Models with CRF inference layer can consistently outperform the models with softmax layer under all configurations on NER and chunking tasks, proving the effectiveness of label dependency information. While for POS tagging, the local softmax based models give slightly better accuracies while the difference is insignificant ($p > 0.2$).

[3]We use *t-test* to calculate the p value, reporting the highest p value when giving the conclusions on multiple configurations.

Results (F1-score)			NER			
			WLSTM+CRF	**WLSTM**	**WCNN+CRF**	**WCNN**
Nochar	Literature		90.10 (H-15)* 90.20 (L-16) 90.43 (S-17)*	87.00 (M-16) 89.34 (S-17)*	89.59 (C-11)* 90.54 (S-17)*	89.97 (S-17)*
	Ours	Max	89.45	88.57	88.90	88.56
		Mean±std	89.31±0.10	88.49±0.17	88.65±0.20	88.50±0.05
CLSTM	Literature		90.94 (L-16) 91.20 (Y-17)‡	89.15 (L-16)	–	–
	Ours	Max	91.20	90.84	90.70	90.46
		Mean±std	91.08±0.08	90.77±0.06	90.48±0.23	90.28±0.30
CCNN	Literature		90.91±0.20 (C-16) 91.21 (M-16) 90.87±0.13 (P-17)	89.36 (M-16)	–	–
	Ours	Max	91.35	90.73	90.43	90.51
		Mean±std	91.11±0.21	90.60±0.11	90.28±0.09	90.26±0.19

Table 4: Results for NER.[4]

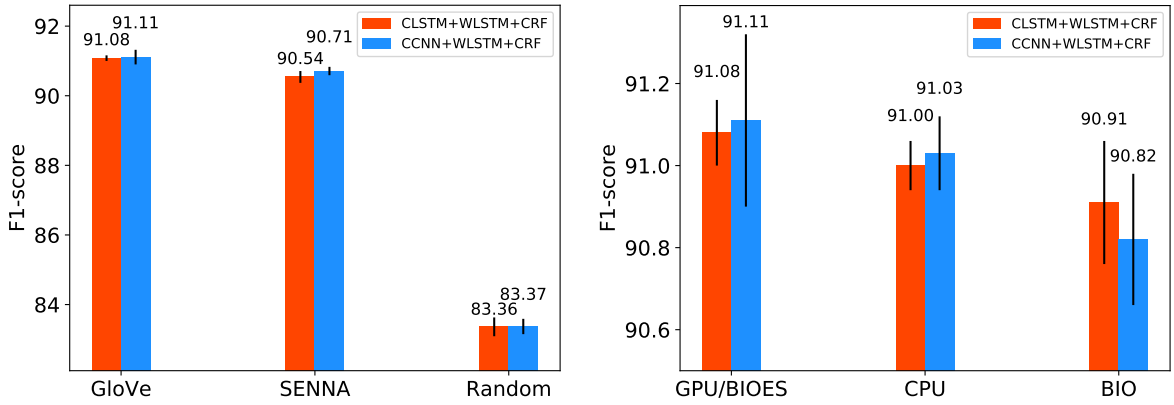
Results (F1-score)			chunking			
			WLSTM+CRF	**WLSTM**	**WCNN+CRF**	**WCNN**
Nochar	Literature		94.46 (H-15)*	94.13 (Z-17) 95.02 (H-17)*	94.32 (C-11)*	–
	Ours	Max	94.49	93.79	94.23	94.12
		Mean±std	94.37±0.11	93.75±0.04	94.11±0.08	94.08±0.06
CLSTM	Literature		93.15 (R-17) 94.66 (Y-17)‡	–	–	–
	Ours	Max	95.00	94.33	94.76	94.55
		Mean±std	94.93±0.05	94.28±0.04	94.66±0.01	94.48±0.07
CCNN	Literature		95.00±0.08 (P-17)	–	–	–
	Ours	Max	95.06	94.24	94.77	94.51
		Mean±std	94.86±0.14	94.19±0.04	94.66±0.13	94.47±0.03

Table 5: Results for chunking.[4]

Results (Accuracy)			POS			
			WLSTM+CRF	**WLSTM**	**WCNN+CRF**	**WCNN**
Nochar	Literature		97.55 (H-15)*	96.93 (M-16) 97.45 (H-17)*	97.29 (C-11)*	96.13 (S-14)
	Ours	Max	97.20	97.23	96.99	97.07
		Mean±std	97.19±0.01	97.20±0.02	96.95±0.04	97.01±0.04
CLSTM	Literature		97.35±0.09 (L-16)† 97.55 (Y-17)‡	97.78 (L-15)	–	–
	Ours	Max	97.49	97.51	97.38	97.38
		Mean±std	97.47±0.02	97.48±0.02	97.33±0.03	97.33±0.04
CCNN	Literature		97.55 (M-16)	97.33 (M-16)	–	97.32 (S-14)
	Ours	Max	97.46	97.51	97.33	97.33
		Mean±std	97.43±0.02	97.44±0.04	97.29±0.03	97.30±0.02

Table 6: Results for POS tagging.[4]

[4] In Tables 4, 5 and 6, the abbreviation (C-11)=Collobert et al. (2011), (S-14)=Santos and Zadrozny (2014), (H-15)=Huang et al. (2015), (L-16)=Lample et al. (2016), (M-16)=Ma and Hovy (2016), (C-16)=Chiu and Nichols (2016), (Z-17)=Zhai et al. (2017), (H-17)=Hashimoto et al. (2017), (Y-17)=Yang et al. (2017b), (R-17)=Rei (2017), (S-17)=Strubell et al. (2017) and (P-17)=Peters et al. (2017). * suggests that models with handcrafted features. Results of (L-16)† is reported by Liu et al. (2018) by running the code of Lample et al. (2016) on the corresponding dataset. (Y-17)‡ use GRU for character and word sequence representations; here we regard GRU as a variant of LSTM.



(a) Pretrained embeddings.

(b) Tag scheme and running environment.

Figure 4: Performance comparison on the CoNLL 2003 NER task.

4.4 External factors

In addition to model structures, external factors such as pretrained embeddings, tag scheme, and optimizer can significantly influence system performance. We investigate a set of external factors on the NER dataset with the two best models: CLSTM+WLSTM+CRF and CCNN+WLSTM+CRF.

Pretrained embedding. Figure 4(a) shows the F1-scores of the two best models on the NER test set with two different pretrained embeddings, as well as the random initialization. Compared with the random initialization, models using pretrained embeddings give significant improvements ($p < 0.01$). The GloVe 100-dimension embeddings give higher F1-scores than SENNA (Collobert et al., 2011) on both models, which is consistent with the observation of Ma and Hovy (2016).

Tag scheme. We examine two different tag schemes: BIO and BIOES (Ratinov and Roth, 2009). The results are shown in Figure 4(b). In our experiments, models using BIOES are significantly ($p < 0.05$) better than BIO. Our observation is consistent with most literature (Ratinov and Roth, 2009; Dai et al., 2015). Reimers and Gurevych (2017b) report that the difference between the schemes is insignificant.

Running environment. Liu et al. (2018) observe that neural sequence labeling models can give better results on GPU rather than CPU. We conduct repeated experiments on both GPU and CPU environments. The results are shown in Figure 4(b). Models run on CPU give a lower mean F1-score than models run on GPU, while the difference is insignificant ($p > 0.2$).

Optimizer. We compare different optimizers including SGD, Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012), RMSProp (Tieleman and Hinton, 2012) and Adam (Kingma and Ba, 2014). The results are shown in Figure 5[5]. In contrast to Reimers and Gurevych (2017b), who reported that SGD is the worst optimizer, our results show that SGD outperforms all other optimizers significantly ($p < 0.01$), with a slower convergence process during training. Our observation is consistent with most literature (Chiu and Nichols, 2016; Lample et al., 2016; Ma and Hovy, 2016).

4.5 Analysis

Decoding speed. We test the decoding speeds of the twelve models on the NER dataset using a Nvidia GTX 1080 GPU. Figure 6 shows the decoding times on 10000 NER sentences. The CRF inference layer severely limits the decoding speed due to the left-to-right inference process, which disables the parallel decoding. Character LSTM significantly slows down the system. Compared with models without character information, adding character CNN representations does not affect the decoding speed too much but can give significant accuracy improvements (shown in Section 4.3). Due to the support of parallel computing, word-based CNN models are consistently faster than word-based LSTM models, with close accuracies, leading to large utilization potential in practice.

[5]We fine-tune the learning rates for Adagrad, Adadelta, RMSProp and Adam, and report the best results in the figure.

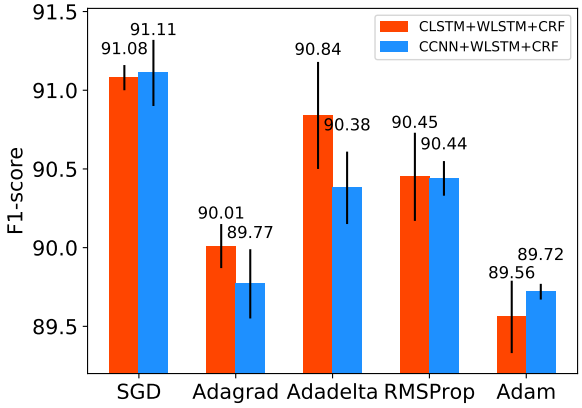


Figure 5: Optimizers.

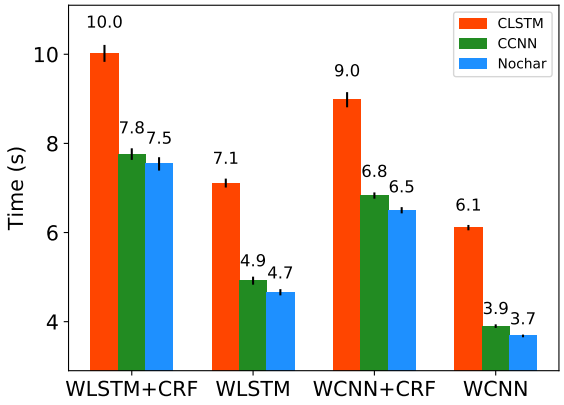


Figure 6: Decoding times (10000 NER sentences).

Results	NER (F1-score)				chunking (F1-score)				POS (Accuracy)			
	IV	OOTV	OOEV	OOBV	IV	OOTV	OOEV	OOBV	IV	OOTV	OOEV	OOBV
Nochar+WLSTM+CRF	91.33	87.36	100.00	69.68	94.87	90.84	95.51	91.47	97.51	89.76	94.07	75.36
CLSTM+WLSTM+CRF	**92.18**	90.63	100.00	78.57	**95.20**	**92.65**	94.38	**94.01**	**97.63**	**93.82**	94.07	**87.32**
CCNN+WLSTM+CRF	91.76	**91.25**	100.00	**81.58**	95.15	92.34	**97.75**	93.55	97.62	93.33	**94.69**	83.82
Nochar+WCNN+CRF	90.71	86.99	100.00	69.09	94.56	90.98	93.26	91.71	97.29	89.10	**94.17**	74.15
CLSTM+WCNN+CRF	**91.59**	90.07	100.00	77.92	**95.02**	91.86	94.38	**93.32**	**97.48**	**93.28**	**94.17**	**88.29**
CCNN+WCNN+CRF	91.35	**90.46**	100.00	**78.88**	94.83	**92.42**	**96.63**	92.40	97.46	92.74	93.86	87.80

Table 7: Results for OOV analysis.

OOV error. We conduct error analysis on in-vocabulary and out-of-vocabulary words with the CRF based models[6]. Following Ma and Hovy (2016), words in the test set are divided into four subsets: in-vocabulary words, out-of-training-vocabulary words (OOTV), out-of-embedding-vocabulary words (OOEV) and out-of-both-vocabulary words (OOBV). For NER and chunking, we consider entities or chunks rather than words. The OOV entities and chunks are categorized following Ma and Hovy (2016).

Table 7 shows the performances of different OOV splits on three benchmarks. The top three rows list the performances of word-based LSTM CRF models, followed by the word-based CNN CRF models. The results of OOEV in NER keep 100% because of there exist only 8 OOEV entities and all are recognized correctly. It is obvious that character LSTM or CNN representations improve OOTV and OOBV the most on both WLSTM+CRF and WCNN+CRF models across all three datasets, proving that the main contribution of neural character sequence representations is to disambiguate the OOV words. Models with character LSTM representations give the best IV scores across all configurations, which may be because character LSTM can be well trained on IV data, bringing the useful global character sequence information. On the OOVs, character LSTM and CNN gives comparable results.

5 Conclusion

We built a unified neural sequence labeling framework to reproduce and compare recent state-of-the-art models with different configurations. We explored three neural model design decisions: character sequence representations, word sequence representations, and inference layer. Experiments show that character information helps to improve model performances, especially on disambiguating OOV words. Character-level LSTM and CNN structures give comparable improvements, with the latter being more efficient. In most cases, models with word-level LSTM encoders outperform those with CNN, at the expense of longer decoding time. We observed that the CRF inference algorithm is effective on NER and chunking tasks, but does not have the advantage on POS tagging. With controlled experiments on the NER dataset, we showed that BIOES tags are better than BIO. Besides, pretrained GloVe 100d embedding and SGD optimizer give significantly better performances compared to their competitors.

[6]We choose the models that give the median performance on the test set for conducting result analysis.

Acknowledgements

We thank the anonymous reviewers for their useful comments. Yue Zhang is the corresponding author.

References

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *TACL*, 4:357–370.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Hong-Jie Dai, Po-Ting Lai, Yung-Chun Chang, and Richard Tzong-Han Tsai. 2015. Enhancing of chemical compound and drug name recognition using representative tag scheme and fine-grained tokenization. *Journal of cheminformatics*, 7(S1):S14.

Cıcero dos Santos, Victor Guimaraes, RJ Niterói, and Rio de Janeiro. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, page 25.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323.

James Hammerton. 2003. Named entity recognition with long short-term memory. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 172–175. Association for Computational Linguistics.

Kazuma Hashimoto, Yoshimasa Tsuruoka, Richard Socher, et al. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1923–1933.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL-HLT*, pages 260–270.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530.

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. *AAAI*.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *EMNLP*, pages 879–888.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via Bi-directional LSTM-CNNs-CRF. In *ACL*, volume 1, pages 1064–1074.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *CoNLL*, pages 78–86.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*, volume 1, pages 1756–1765.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*, pages 147–155.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2121–2130.

Nils Reimers and Iryna Gurevych. 2017a. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799*.

Nils Reimers and Iryna Gurevych. 2017b. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate entity recognition with iterated dilated convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2670–2680.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 127–132. Association for Computational Linguistics.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *HLT-NAACL*, pages 142–147.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*.

Jie Yang, Yue Zhang, and Fei Dong. 2017a. Neural reranking for named entity recognition. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 784–792.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017b. Transfer learning for sequence tagging with hierarchical recurrent networks. In *ICLR*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Feifei Zhai, Saloni Potdar, Bing Xiang, and Bowen Zhou. 2017. Neural models for sequence chunking. In *AAAI*, pages 3365–3371.

Neural Network Models for Paraphrase Identification, Semantic Textual Similarity, Natural Language Inference, and Question Answering

Wuwei Lan and Wei Xu
Department of Computer Science and Engineering
Ohio State University
{lan.105, xu.1265}@osu.edu

Abstract

In this paper, we analyze several neural network designs (and their variations) for sentence pair modeling and compare their performance extensively across eight datasets, including paraphrase identification, semantic textual similarity, natural language inference, and question answering tasks. Although most of these models have claimed state-of-the-art performance, the original papers often reported on only one or two selected datasets. We provide a systematic study and show that (i) encoding contextual information by LSTM and inter-sentence interactions are critical, (ii) Tree-LSTM does not help as much as previously claimed but surprisingly improves performance on Twitter datasets, (iii) the Enhanced Sequential Inference Model (Chen et al., 2017) is the best so far for larger datasets, while the Pairwise Word Interaction Model (He and Lin, 2016) achieves the best performance when less data is available. We release our implementations as an open-source toolkit.

1 Introduction

Sentence pair modeling is a fundamental technique underlying many NLP tasks, including the following:

- Semantic Textual Similarity (STS), which measures the degree of equivalence in the underlying semantics of paired snippets of text (Agirre et al., 2016).
- Paraphrase Identification (PI), which identifies whether two sentences express the same meaning (Dolan and Brockett, 2005; Xu et al., 2015).
- Natural Language Inference (NLI), also known as recognizing textual entailment (RTE), which concerns whether a hypothesis can be inferred from a premise, requiring understanding of the semantic similarity between the hypothesis and the premise (Dagan et al., 2006; Bowman et al., 2015).
- Question Answering (QA), which can be approximated as ranking candidate answer sentences or phrases based on their similarity to the original question (Yang et al., 2015).
- Machine Comprehension (MC), which requires sentence matching between a passage and a question, pointing out the text region that contains the answer. (Rajpurkar et al., 2016).

Traditionally, researchers had to develop different methods specific for each task. Now neural networks can perform all the above tasks with the same architecture by training end to end. Various neural models (He and Lin, 2016; Chen et al., 2017; Parikh et al., 2016; Wieting et al., 2016; Tomar et al., 2017; Wang et al., 2017; Shen et al., 2017a; Yin et al., 2016) have declared state-of-the-art results for sentence pair modeling tasks; however, they were carefully designed and evaluated on selected (often one or two) datasets that can demonstrate the superiority of the model. The research questions are as follows: Do they perform well on other tasks and datasets? How much performance gain is due to certain system design choices and hyperparameter optimizations?

Proceedings of the 27th International Conference on Computational Linguistics, pages 3890–3902
Santa Fe, New Mexico, USA, August 20-26, 2018.

To answer these questions and better understand different network designs, we systematically analyze and compare the state-of-the-art neural models across multiple tasks and multiple domains. Namely, we implement five models and their variations on the same PyTorch platform: InferSent model (Conneau et al., 2017), Shortcut-stacked Sentence Encoder Model (Nie and Bansal, 2017), Pairwise Word Interaction Model (He and Lin, 2016), Decomposable Attention Model (Parikh et al., 2016), and Enhanced Sequential Inference Model (Chen et al., 2017). They are representative of the two most common approaches: **sentence encoding models** that learn vector representations of individual sentences and then calculate the semantic relationship between sentences based on vector distance and **sentence pair interaction models** that use some sorts of word alignment mechanisms (e.g., attention) then aggregate inter-sentence interactions. We focus on identifying important network designs and present a series of findings with quantitative measurements and in-depth analyses, including (i) incorporating inter-sentence interactions is critical; (ii) Tree-LSTM does not help as much as previously claimed but surprisingly improves performance on Twitter data; (iii) Enhanced Sequential Inference Model has the most consistent high performance for larger datasets, while Pairwise Word Interaction Model performs better on smaller datasets and Shortcut-Stacked Sentence Encoder Model is the best performaning model on the Quora corpus. We release our implementations as a toolkit to the research community.[1]

2 General Framework for Sentence Pair Modeling

Various neural networks have been proposed for sentence pair modeling, all of which fall into two types of approaches. The sentence encoding approach encodes each sentence into a fixed-length vector and then computes sentence similarity directly. The model of this type has advantages in the simplicity of the network design and generalization to other NLP tasks. The sentence pair interaction approach takes word alignment and interactions between the sentence pair into account and often show better performance when trained on in-domain data. Here we outline the two types of neural networks under the same general framework:

- **The Input Embedding Layer** takes vector representations of words as input, where pretrained word embeddings are most commonly used, e.g. GloVe (Pennington et al., 2014) or Word2vec (Mikolov et al., 2013). Some work used embeddings specially trained on phrase or sentence pairs that are paraphrases (Wieting and Gimpel, 2017; Tomar et al., 2017); some used subword embeddings, which showed improvement on social media data (Lan and Xu, 2018).
- **The Context Encoding Layer** incorporates word context and sequence order into modeling for better vector representation. This layer often uses CNN (He et al., 2015), LSTM (Chen et al., 2017), recursive neural network (Socher et al., 2011), or highway network (Gong et al., 2017). The sentence encoding type of model will stop at this step, and directly use the encoded vectors to compute the semantic similarity through vector distances and/or the output classification layer.
- **The Interaction and Attention Layer** calculates word pair (or n-gram pair) interactions using the outputs of the encoding layer. This is the key component for the interaction-aggregation type of model. In the PWIM model (He and Lin, 2016), the interactions are calculated by cosine similarity, Euclidean distance, and the dot product of the vectors. Various models put different weights on different interactions, primarily simulating the word alignment between two sentences. The alignment information is useful for sentence pair modeling because the semantic relation between two sentences depends largely on the relations of aligned chunks as shown in the SemEval-2016 task of interpretable semantic textual similarity (Agirre et al., 2016).
- **The Output Classification Layer** adapts CNN or MLP to extract semantic-level features on the attentive alignment and applies softmax function to predict probability for each class.

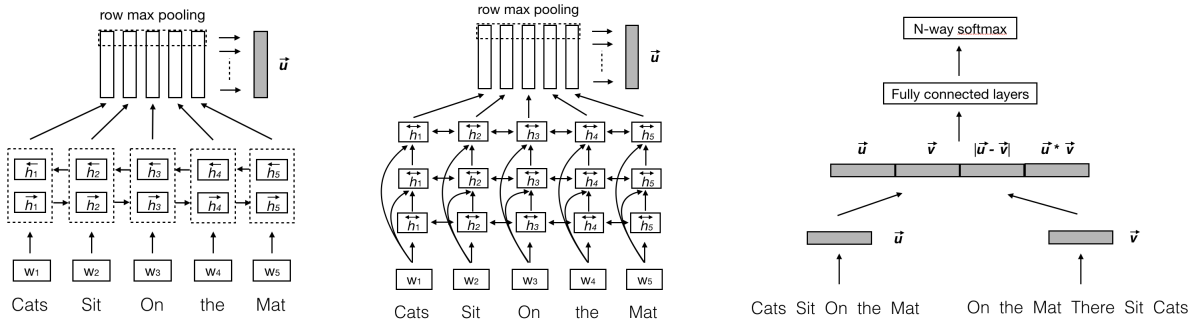
[1]The code is available on the authors' homepages and GitHub: `https://github.com/lanwuwei/SPM_toolkit`

3 Representative Models for Sentence Pair Modeling

Table 1 gives a summary of typical models for sentence pair modeling in recent years. In particular, we investigate five models in depth: two are representative of the sentence encoding type of model, and three are representative of the interaction-aggregation type of model. These models have reported state-or-the-art results with varied architecture design (this section) and implementation details (Section 4.2).

Models	Sentence Encoder	Interaction and Attention	Aggregation and Classification
(Shen et al., 2017b)	Directional self-attention network	-	MLP
(Choi et al., 2017)	Gumbel Tree-LSTM	-	MLP
(Wieting and Gimpel, 2017)	Gated recurrent average network	-	MLP
SSE (Nie and Bansal, 2017)	Shortcut-stacked BiLSTM	-	MLP
(He et al., 2015)	CNN	multi-perspective matching	pooling + MLP
(Rocktäschel et al., 2016)	LSTM	word-by-word neural attention	MLP
(Liu et al., 2016)	LSTM	coupled LSTMs	dynamic pooling + MLP
(Yin et al., 2016)	CNN	attention matrix	logistic regression
DecAtt (Parikh et al., 2016)	-	dot product + soft alignment	summation + MLP
PWIM (He and Lin, 2016)	BiLSTM	cosine, Euclidean, dot product + hard alignment	CNN + MLP
(Wang and Jiang, 2017)	LSTM encodes both context and attention	word-by-word neural attention	MLP
ESIM (Chen et al., 2017)	BiLSTM (Tree-LSTM) before and after attention	dot product + soft alignment	average and max pooling + MLP
(Wang et al., 2017)	BiLSTM	multi-perspective matching	BiLSTM + MLP
(Shen et al., 2017a)	BiLSTM + intra-attention	soft alignment + orthogonal decomposition	MLP
(Ghaeini et al., 2018)	dependent reading BiLSTM	dot product + soft alignment	average and max pooling+MLP

Table 1: Summary of representative neural models for sentence pair modeling. The upper half contains sentence encoding models, and the lower half contains sentence pair interaction models.



(a) InferSent (b) SSE (c) Classification Layer

Figure 1: **Sentence encoding models** focus on learning vector representations of individual sentences and then calculate the semantic relationship between sentences based on vector distance.

3.1 The Bi-LSTM Max-pooling Network (InferSent)

We choose the simple Bi-LSTM max-pooling network from InferSent (Conneau et al., 2017):

$$\overleftrightarrow{\boldsymbol{h}}_i = BiLSTM(\boldsymbol{x}_i, \overleftrightarrow{\boldsymbol{h}}_{i-1}) \tag{1}$$

$$\boldsymbol{v} = max(\overleftrightarrow{\boldsymbol{h}}_1, \overleftrightarrow{\boldsymbol{h}}_2, ..., \overleftrightarrow{\boldsymbol{h}}_n) \tag{2}$$

where $\overleftrightarrow{\boldsymbol{h}}_i$ represents the concatenation of hidden states in both directons. It has shown better transfer learning capabilities than several other sentence embedding models, including SkipThought (Kiros et al., 2015) and FastSent (Hill et al., 2016), when trained on the natural language inference datasets.

3.2 The Shortcut-Stacked Sentence Encoder Model (SSE)

The Shortcut-Stacked Sentence Encoder model (Nie and Bansal, 2017) is a sentence-based embedding model, which enhances multi-layer Bi-LSTM with skip connection to avoid training error accumulation, and calculates each layer as follows:

$$\overleftrightarrow{\boldsymbol{h}}_i^k = BiLSTM(\boldsymbol{x}_i^k, \overleftrightarrow{\boldsymbol{h}}_{i-1}^k) \tag{3}$$

$$\boldsymbol{x}_i^1 = \boldsymbol{w}_i \quad (k=1), \qquad \boldsymbol{x}_i^k = [\boldsymbol{w}_i, \overleftrightarrow{\boldsymbol{h}}_i^{k-1}, \overleftrightarrow{\boldsymbol{h}}_i^{k-2}, ..., \overleftrightarrow{\boldsymbol{h}}_i^1] \quad (k>1) \tag{4}$$

$$\boldsymbol{v} = max(\overleftrightarrow{\boldsymbol{h}}_1^m, \overleftrightarrow{\boldsymbol{h}}_2^m, ..., \overleftrightarrow{\boldsymbol{h}}_n^m) \tag{5}$$

where $\boldsymbol{x}_i^k$ is the input of the kth Bi-LSTM layer at time step i, which is the combination of outputs from all previous layers, $\overleftrightarrow{\boldsymbol{h}}_i^k$ represents the hidden state of the kth Bi-LSTM layer in both directions. The final sentence embedding $\boldsymbol{v}$ is the row-based max pooling over the output of the last Bi-LSTM layer, where n denotes the number of words within a sentence and m is the number of Bi-LSTM layers ($m = 3$ in SSE).

3.3 The Pairwise Word Interaction Model (PWIM)

In the Pairwise Word Interaction model (He and Lin, 2016), each word vector $\boldsymbol{w}_i$ is encoded with context through forward and backward LSTMs: $\overrightarrow{\boldsymbol{h}}_i = LSTM^f(\boldsymbol{w}_i, \overrightarrow{\boldsymbol{h}}_{i-1})$ and $\overleftarrow{\boldsymbol{h}}_i = LSTM^b(\boldsymbol{w}_i, \overleftarrow{\boldsymbol{h}}_{i+1})$. For every word pair $(\boldsymbol{w}_i^a, \boldsymbol{w}_j^b)$ across sentences, the model directly calculates word pair interactions using cosine similarity, Euclidean distance, and dot product over the outputs of the encoding layer:

$$D(\overrightarrow{\boldsymbol{h}}_i, \overrightarrow{\boldsymbol{h}}_j) = [cos(\overrightarrow{\boldsymbol{h}}_i, \overrightarrow{\boldsymbol{h}}_j), \|\overrightarrow{\boldsymbol{h}}_i - \overrightarrow{\boldsymbol{h}}_j\|, \overrightarrow{\boldsymbol{h}}_i \cdot \overrightarrow{\boldsymbol{h}}_j] \tag{6}$$

The above equation not only applies to forward hidden state $\overrightarrow{\boldsymbol{h}}_i$ and backward hidden state $\overleftarrow{\boldsymbol{h}}_i$, but also to the concatenation $\overleftrightarrow{\boldsymbol{h}}_i = [\overrightarrow{\boldsymbol{h}}_i, \overleftarrow{\boldsymbol{h}}_i]$ and summation $\boldsymbol{h}_i^+ = \overrightarrow{\boldsymbol{h}}_i + \overleftarrow{\boldsymbol{h}}_i$, resulting in a tensor $\mathbf{D}^{13 \times |sent1| \times |sent2|}$ after padding one extra bias term. A "hard" attention is applied to the interaction tensor to build word alignment: selecting the most related word pairs and increasing the corresponding weights by 10 times. Then a 19-layer deep CNN is applied to aggregate the word interaction features for final classification.

3.4 The Decomposable Attention Model (DecAtt)

The Decomposable Attention model (Parikh et al., 2016) is one of the earliest models to introduce attention-based alignment for sentence pair modeling, and it achieved state-of-the-art results on the SNLI dataset with about an order of magnitude fewer parameters than other models (see more in Table 5) without relying on word order information. It computes the word pair interaction between $\boldsymbol{w}_i^a$ and $\boldsymbol{w}_j^b$ (from input sentences s_a and s_b, each with m and n words, respectively) as $e_{ij} = F(\boldsymbol{w}_i^a)^T F(\boldsymbol{w}_j^b)$, where F is a feedforward network; then alignment is determined as follows:

$$\boldsymbol{\beta}_i = \sum_{j=1}^{n} \frac{exp(e_{ij})}{\sum_{k=1}^{n} exp(e_{ik})} \boldsymbol{w}_j^b \qquad\qquad \boldsymbol{\alpha}_j = \sum_{i=1}^{m} \frac{exp(e_{ij})}{\sum_{k=1}^{m} exp(e_{kj})} \boldsymbol{w}_i^a \tag{7}$$

where $\boldsymbol{\beta}_i$ is the soft alignment between $\boldsymbol{w}_i^a$ and subphrases $\boldsymbol{w}_j^b$ in sentence s_b, and vice versa for $\boldsymbol{\alpha}_j$. The aligned phrases are fed into another feedforward network G: $\boldsymbol{v}_i^a = G([\boldsymbol{w}_i^a; \boldsymbol{\beta}_i])$ and $\boldsymbol{v}_j^b = G([\boldsymbol{w}_j^b; \boldsymbol{\alpha}_j])$ to generate sets $\{\boldsymbol{v}_i^a\}$ and $\{\boldsymbol{v}_j^b\}$, which are aggregated by summation and then concatenated together for classification.

3.5 The Enhanced Sequential Inference Model (ESIM)

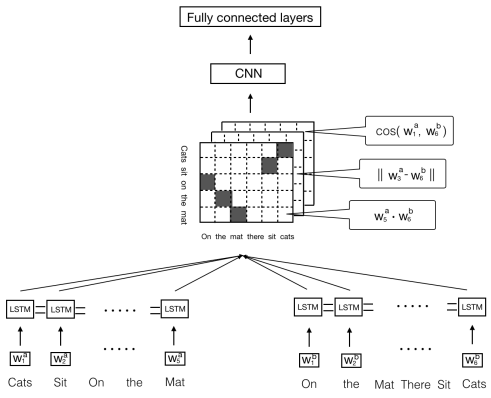
The Enhanced Sequential Inference Model (Chen et al., 2017) is closely related to the DecAtt model, but it differs in a few aspects. First, Chen et al. (2017) demonstrated that using Bi-LSTM to encode sequential contexts is important for performance improvement. They used the concatenation $\overline{\boldsymbol{w}}_i = \overleftrightarrow{\boldsymbol{h}}_i = [\overrightarrow{\boldsymbol{h}}_i, \overleftarrow{\boldsymbol{h}}_i]$ of both directions as in the PWIM model. The word alignment $\boldsymbol{\beta}_i$ and $\boldsymbol{\alpha}_j$ between $\overline{\boldsymbol{w}}^a$ and $\overline{\boldsymbol{w}}^b$ are calculated the same way as in DecAtt. Second, they showed the competitive performance of recursive architecture with constituency parsing, which complements with sequential LSTM. The feedforward function G in DecAtt is replaced with Tree-LSTM:

$$\boldsymbol{v}_i^a = TreeLSTM([\overline{\boldsymbol{w}}_i^a; \boldsymbol{\beta}_i; \overline{\boldsymbol{w}}_i^a - \boldsymbol{\beta}_i; \overline{\boldsymbol{w}}_i^a \odot \boldsymbol{\beta}_i]) \tag{8}$$

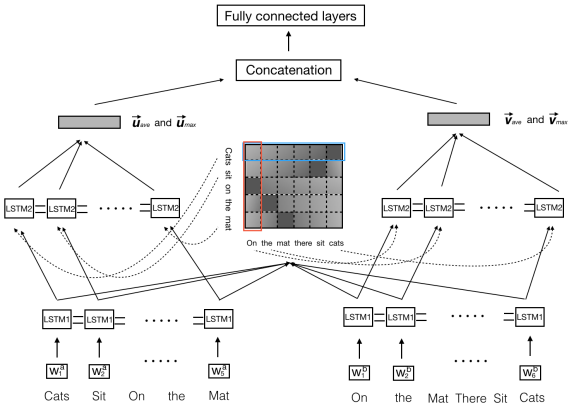
$$\boldsymbol{v}_j^b = TreeLSTM([\overline{\boldsymbol{w}}_j^b; \boldsymbol{\alpha}_j; \overline{\boldsymbol{w}}_j^b - \boldsymbol{\alpha}_j; \overline{\boldsymbol{w}}_j^b \odot \boldsymbol{\alpha}_j]) \tag{9}$$

Third, instead of using summation in aggregation, ESIM adapts the average and max pooling and concatenation $\boldsymbol{v} = [\boldsymbol{v}_{ave}^a; \boldsymbol{v}_{max}^a; \boldsymbol{v}_{ave}^b; \boldsymbol{v}_{max}^b]$ before passing through multi-layer perceptron (MLP) for classification:

$$\boldsymbol{v}_{ave}^a = \sum_{i=1}^{m} \frac{\boldsymbol{v}_i^a}{m}, \qquad \boldsymbol{v}_{max}^a = \max_{i=1}^{m} \boldsymbol{v}_i^a, \qquad \boldsymbol{v}_{ave}^b = \sum_{j=1}^{n} \frac{\boldsymbol{v}_j^b}{n}, \qquad \boldsymbol{v}_{max}^b = \max_{j=1}^{n} \boldsymbol{v}_j^b \tag{10}$$



(a) PWIM



(b) ESIM$_{seq}$ (DecAtt is similar and simpler.)

Figure 2: **Sentence pair interaction models** use different word alignment mechanisms before aggregation.

4 Experiments and Analysis

4.1 Datasets

We conducted sentence pair modeling experiments on eight popular datasets: two NLI datasets, three PI datasets, one STS dataset and two QA datasets. Table 2 gives a comparison of these datasets:

- **SNLI** (Bowman et al., 2015) contains 570k hypotheses written by crowdsourcing workers given the premises. It focuses on three semantic relations: the premise entails the hypothesis (entailment), they contradict each other (contradiction), or they are unrelated (neutral).
- **Multi-NLI** (Williams et al., 2017) extends the SNLI corpus to multiple genres of written and spoken texts with 433k sentence pairs.

Dataset	Size		Example and Label	
SNLI	train	550,152	s_a: *Two men on bicycles competing in a race.* s_b: *Men are riding bicycles on the street.*	**entailment**
	dev	10,000		neutral
	test	10,000		contradict
Multi-NLI	train	392,703	s_a: *The Old One always comforted Ca'daan, except today.* s_b: *Ca'daan knew the Old One very well.*	entailment
	dev	20,000		**neutral**
	test	20,000		contradict
Quora	train	384,348	s_a: *What should I do to avoid sleeping in class?* s_b: *How do I not sleep in a boring class?*	**paraphrase**
	dev	10,000		non-paraphrase
	test	10,000		
Twitter-URL	train	42,200	s_a: *Letter warned Wells Fargo of "widespread" fraud in 2007.* s_b: *Letters suggest Wells Fargo scandal started earlier.*	**paraphrase**
	dev	-		non-paraphrase
	test	9,324		
PIT-2015	train	11,530	s_a: *Ezekiel Ansah w the 3D shades Popped out lens* s_b: *Ezekiel Ansah was wearing lens less 3D glasses*	**paraphrase**
	dev	4,142		non-paraphrase
	test	838		
STS-2014	train	7,592	s_a: *Then perhaps we could have avoided a catastrophe.* s_b: *Then we might have been able to avoid a disaster.*	score $[0, 5]$
	dev	-		**4.6**
	test	3,750		
WikiQA	train	8,672	s_a: *How much is 1 tablespoon of water?* s_b: *In Australia one tablespoon (measurement unit) is 20 mL.*	**true**
	dev	1,130		false
	test	2,351		
TrecQA	train	53,417	s_a: *Who was Lincoln's Secretary of State?* s_b: *William Seward*	true
	dev	1,148		**false**
	test	1,517		

Table 2: Basic statistics and examples of different datasets for sentence pair modeling tasks.

- **Quora** (Iyer et al., 2017) contains 400k question pairs collected from the Quora website. This dataset has balanced positive and negative labels indicating whether the questions are duplicated or not.
- **Twitter-URL** (Lan et al., 2017) includes 50k sentence pairs collected from tweets that share the same URL of news articles. This dataset contains both formal and informal language.
- **PIT-2015** (Xu et al., 2015) comes from SemEval-2015 and was collected from tweets under the same trending topic. It contains naturally occurred (i.e. written by independent Twitter users spontaneously) paraphrases and non-paraphrases with varied topics and language styles.
- **STS-2014** (Agirre et al., 2014) is from SemEval-2014, constructed from image descriptions, news headlines, tweet news, discussion forums, and OntoNotes (Hovy et al., 2006).
- **WikiQA** (Yang et al., 2015) is an open-domain question-answering dataset. Following He and Lin (2016), questions without correct candidate answer sentences are excluded, and answer sentences are truncated to 40 tokens, resulting in 12k question-answer pairs for our experiments.
- **TrecQA** (Wang et al., 2007) is an answer selection task of 56k question-answer pairs and created in Text Retrieval Conferences (TREC). For both WikiQA and TrecQA datasets, the best answer is selected according to the semantic relatedness with the question.

4.2 Implementation Details

We implement all the models with the same PyTorch framework.[2][3] Below, we summarize the implementation details that are key for reproducing results for each model:

- **SSE:** This model can converge very fast, for example, 2 or 3 epochs for the SNLI dataset. We control the convergence speed by updating the learning rate for each epoch: specifically, $lr = \frac{1}{2^{\frac{epoch\_i}{2}}} * init\_lr$, where $init\_lr$ is the initial learning rate and $epoch\_i$ is the index of current epoch.

[2]InferSent and SSE have open-source PyTorch implementations by the original authors, for which we reused part of the code.
[3]Our code is available at: `https://github.com/lanwuwei/SPM_toolkit`

- **DecAtt:** It is important to use gradient clipping for this model: for each gradient update, we check the L2 norm of all the gradient values, if it is greater than a threshold b, we scale the gradient by a factor $\alpha = b/L2\_norm$. Another useful procedure is to assemble batches of sentences with similar length.
- **ESIM:** Similar but different from DecAtt, ESIM batches sentences with varied length and uses masks to filter out padding information. In order to batch the parse trees within Tree-LSTM recursion, we follow Bowman et al.'s (2016) procedure that converts tree structures into the linear sequential structure of a shift reduce parser. Two additional masks are used for producing left and right children of a tree node.
- **PWIM:** The cosine and Euclidean distances used in the word interaction layer have smaller values for similar vectors while dot products have larger values. The performance increases if we add a negative sign to make all the vector similarity measurements behave consistently.

4.3 Analysis

4.3.1 Re-implementation Results vs. Previously Reported Results

Table 3 and 4 show the results reported in the original papers and the replicated results with our implementation. We use accuracy, F1 score, Pearson's r, Mean Average Precision (MAP), and Mean Reciprocal Rank (MRR) for evaluation on different datasets following the literature. Our reproduced results are slightly lower than the original results by $0.5 \sim 1.5$ points on accuracy. We suspect the following potential reasons: (i) less extensive hyperparameter tuning for each individual dataset; (ii) only one run with random seeding to report results; and (iii) use of different neural network toolkits: for example, the original ESIM model was implemented with Theano, and PWIM model was in Torch.

4.3.2 Effects of Model Components

Herein, we examine the main components that account for performance in sentence pair modeling.

How important is LSTM encoded context information for sentence pair modeling?
Regarding DecAtt, Parikh et al. (2016) mentioned that "intra-sentence attention is optional"; they can achieve competitive results without considering context information. However, not surprisingly, our experiments consistently show that encoding sequential context information with LSTM is critical. Compared to DecAtt, ESIM shows better performance on every dataset (see Table 4 and Figure 3). The main difference between ESIM and DecAtt that contributes to performance improvement, we found, is the use of Bi-LSTM and Tree-LSTM for sentence encoding, rather than the different choices of aggregation functions.

Why does Tree-LSTM help with Twitter data?
Chen et al. (2017) offered a simple combination (ESIM$_{seq+tree}$) by averaging the prediction probabilities of two ESIM variants that use sequential Bi-LSTM and Tree-LSTM respectively, and suggested "parsing information complements very well with ESIM and further improves the performance". However, we found that adding Tree-LSTM only helps slightly or not at all for most datasets, but it helps noticably with the two Twitter paraphrase datasets. We hypothesize the reason is that these two datasets come from real-world tweets which often contain extraneous text fragments, in contrast to SNLI and other datasets that have sentences written by crowdsourcing workers. For example, the segment "*ever wondered ,*" in the sentence pair *ever wondered , why your recorded #voice sounds weird to you?* and *why do our recorded voices sound so weird to us?* introduces a disruptive context into the Bi-LSTM encoder, while Tree-LSTM can put it in a less important position after constituency parsing.

How important is attentive interaction for sentence pair modeling? Why does SSE excel on Quora?
Both ESIM and DecAtt (Eq. 7) calculate an attention-based soft alignment between a sentence pair, which was also proposed in (Rocktäschel et al., 2016) and (Wang and Jiang, 2017) for sentence pair modeling, whereas PWIM utilizes a hard attention mechanism. Both attention strategies are critical for model performance. In PWIM model (He and Lin, 2016), we observed a 1$\sim$2 point performance drop after

Model	SNLI	Multi-NLI	Quora	Twitter-URL	PIT-2015	STS-2014	WikiQA	TrecQA
	Acc	Acc_m/Acc_um	Acc	F1	F1	r	MAP/MRR	MAP/MRR
InferSent	0.845	-/-	-	-	-	0.700[5]	-	-
SSE	0.860	0.746/0.736	-	-	-	-	-	-
DecAtt	0.863	-	0.865[3]	-	-	-	-	-
ESIM$_{tree}$	0.878	-	-	-	-	-	-	-
ESIM$_{seq}$	0.880	0.723/0.721[4]	-	-	-	-	-	-
ESIM$_{seq+tree}$	0.886	-	-	-	-	-	-	-
PWIM	-	-	-	0.749	0.667	0.767	0.709/0.723	0.759/0.822

Table 3: Reported results from original papers, which are mostly limited to a few datasets. For the Multi-NLI dataset, Acc_m represents testing accuracy for the matched genre and Acc_um for the unmatched genre.

Model	SNLI	Multi-NLI	Quora	Twitter-URL	PIT-2015	STS-2014	WikiQA	TrecQA
	Acc	Acc_m/Acc_um	Acc	F1	F1	r	MAP/MRR	MAP/MRR
InferSent	0.846	0.705/0.703	0.866	0.746	0.451	0.715	0.287/0.287	0.521/0.559
SSE	0.855	0.740/0.734	**0.878**	0.650	0.422	0.378	0.624/0.638	0.628/0.670
DecAtt	0.856	0.719/0.713	0.845	0.652	0.430	0.317	0.603/0.619	0.660/0.712
ESIM$_{tree}$	0.864	0.736/0.727	0.755	0.740	0.447	0.493	0.618/0.633	0.698/0.734
ESIM$_{seq}$	0.870	0.752/0.738	0.850	0.748	0.520	0.602	0.652/0.664	**0.771/0.795**
ESIM$_{seq+tree}$	**0.871**	**0.753/0.748**	0.854	0.759	0.538	0.589	0.647/0.658	0.749/0.768
PWIM	0.822	0.722/0.716	0.834	**0.761**	**0.656**	**0.743**	**0.706/0.723**	0.739/0.795

Table 4: Replicated results with our reimplementation in PyTorch across multiple tasks and datasets. The best result in each dataset is denoted by a **bold** typeface, and the second best is denoted by an underline.

removing the hard attention, 0∼3 point performance drop and ∼25% training time reduction after removing the 19-layer CNN aggregation. Likely without even the authors of SSE knowing, the SSE model performs extraordinarily well on the Quora corpus, perhaps because Quora contains many sentence pairs with less complicated inter-sentence interactions (e.g., many identical words in the two sentences) and incorrect ground truth labels (e.g., *What is your biggest regret in life?* and *What's the biggest regret you've had in life?* are labeled as non-duplicate questions by mistake).

4.3.3 Learning Curves and Training Time

Figure 3 shows the learning curves. The DecAtt model converges quickly and performs well on large NLI datasets due to its design simplicity. PWIM is the slowest model (see time comparison in Table 5) but shows very strong performance on semantic similarity and paraphrase identification datasets. ESIM and SSE keep a good balance between training time and performance.

[3]This number was reported in (Tomar et al., 2017) by co-authors of DecAtt (Parikh et al., 2016).
[4]This number was reproduced by Williams et al. (2017).
[5]This number was generated by InferSent traind on SNLI and Multi-NLI datasets.

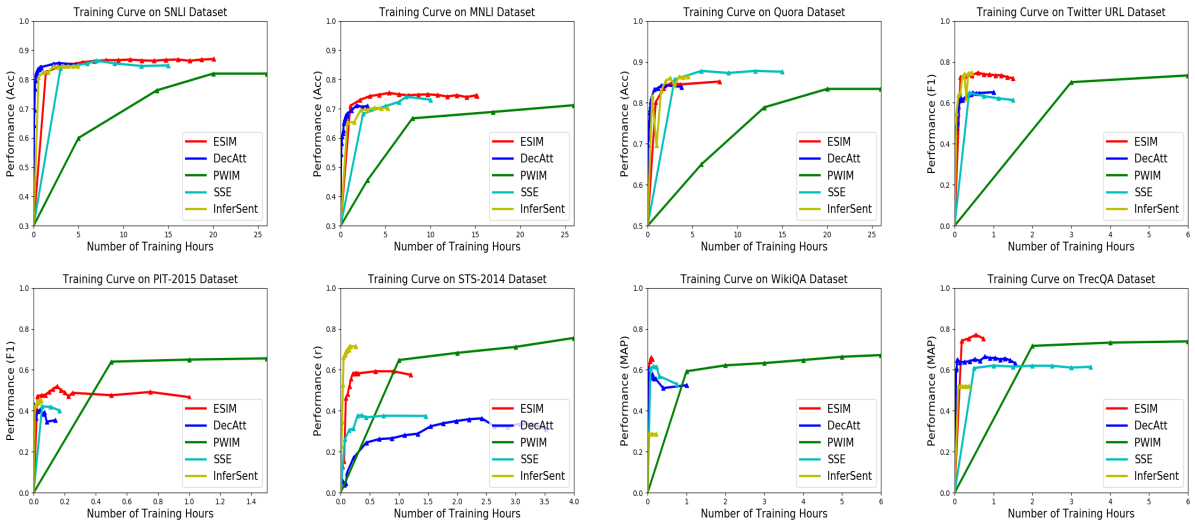


Figure 3: Training curves of ESIM, DecAtt, PWIM, SSE and InferSent models on eight datasets.

	InferSent	**SSE**	**DecAtt**	**ESIM$_{seq}$**	**ESIM$_{tree}$**	**PWIM**
Number of parameters	47M	140M	380K	4.3M	7.7M	2.2M
Avg epoch time (seconds) / sentence pair	0.005	0.032	0.0006	0.013	0.016	0.60
Ratio compared to DecAtt model	$\times$8	$\times$53	1	$\times$22	$\times$26	$\times$1000

Table 5: Average training time per sentence pair in the Twitter-URL dataset (similar time for other datasets).

4.3.4 Effects of Training Data Size

As shown in Figure 4, we experimented with different training sizes of the largest SNLI dataset. All the models show improved performance as we increase the training size. ESIM and SSE have very similar trends and clearly outperform PWIM on the SNLI dataset. DecAtt shows a performance jump when the training size exceeds a threshold.

4.3.5 Categorical Performance Comparison

We conducted an in-depth analysis of model performance on the Multi-domain NLI dataset based on different categories: text genre, sentence pair overlap, and sentence length. As shown in Table 7, all models have comparable performance between matched genre and unmatched genre. Sentence length and overlap turn out to be two important factors – the longer the sentences and the fewer tokens in common, the more challenging it is to determine their semantic relationship. These phenomena shared by the state-of-the-art systems reflect their similar design framework which is symmetric at processing both sentences in the pair, while question answering and natural language inference tasks are directional (Ghaeini et al., 2018). How to incorporate asymmetry into model design will be worth more exploration in future research.

4.3.6 Transfer Learning Experiments

In addition to the cross-domain study (Table 7), we conducted transfer learning experiments on three paraphrase identification datasets (Table 6). The most noteworthy phenomenon is that the SSE model performs better on Twitter-URL and PIT-2015 when trained on the large out-of-domain Quora data than the small in-domain training data. Two likely reasons are: 1) the SSE model with over 29 million parameters is data hungry and 2) SSE model is a sentence encoding model, which generalizes better across domains/tasks than sentence pair interaction models. Sentence pair interaction models may encounter difficulties on Quora, which contains sentence pairs with the highest word overlap (51.5%) among all datasets and often causes

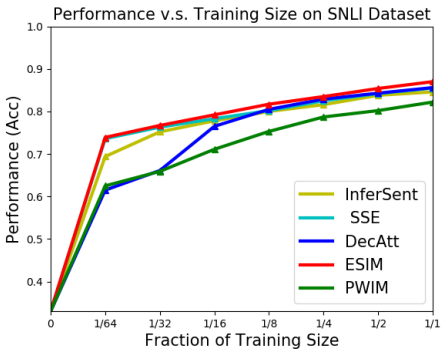


Figure 4: Performance vs. training size (log scale in x-axis) on SNLI dataset.

Models	Quora	URL	PIT	train/test
	trained on Quora			on PIT
InferSent	0.866	0.528	0.394	0.451
SSE	0.878	0.681	0.594	0.422
DecAtt	0.845	0.649	0.497	0.430
ESIM_{seq}	0.850	0.643	0.501	0.520
PWIM	0.835	0.601	0.518	0.656
	trained on URL			
InferSent	0.703	0.746	0.535	0.451
SSE	0.630	0.650	0.477	0.422
DecAtt	0.632	0.652	0.450	0.430
ESIM_{seq}	0.641	0.748	0.511	0.520
PWIM	0.678	0.761	0.634	0.656

Table 6: Transfer learning experiments for paraphrase identification task.

	Category	#Examples	InferSent	SSE	DecAtt	ESIM_{seq}	PWIM
Matched Genre	Fiction	1973	0.703	0.727	0.706	0.742	0.707
	Government	1945	0.753	0.746	0.743	0.790	0.751
	Slate	1955	0.653	0.670	0.671	0.697	0.670
	Telephone	1966	0.718	0.728	0.717	0.753	0.709
	Travel	1976	0.705	0.701	0.733	0.752	0.714
Mismatched Genre	9/11	1974	0.685	0.710	0.699	0.737	0.711
	Face-to-face	1974	0.713	0.729	0.720	0.761	0.710
	Letters	1977	0.734	0.757	0.754	0.775	0.757
	OUP	1961	0.698	0.715	0.719	0.759	0.710
	Verbatim	1946	0.691	0.701	0.709	0.725	0.713
Overlap	>60%	488	0.756	0.795	0.805	0.842	0.811
	30% ∼ 60%	3225	0.740	0.751	0.745	0.769	0.743
	<30%	6102	0.685	0.689	0.691	0.727	0.682
Length	>20 tokens	3730	0.692	0.676	0.685	0.731	0.694
	10∼20 tokens	3673	0.712	0.725	0.721	0.753	0.720
	<10 tokens	2412	0.721	0.758	0.748	0.762	0.724

Table 7: Categorical performance (accuracy) on Multi-NLI dataset. Overlap is the percentage of shared tokens between two sentences. Length is calculated based on the number of tokens of the longer sentence.

the interaction patterns to focus on a few key words that differ. In contrast, the Twitter-URL dataset has the lowest overlap (23.0%) with a semantic relationship that is mainly based on the intention of the tweets.

5 Conclusion

We analyzed five different neural models (and their variations) for sentence pair modeling and conducted a series of experiments with eight representative datasets for different NLP tasks. We quantified the importance of the LSTM encoder and attentive alignment for inter-sentence interaction, as well as the transfer learning ability of sentence encoding based models. We showed that the SNLI corpus of over 550k sentence pairs cannot saturate the learning curve. We systematically compared the strengths and weaknesses of different network designs and provided insights for future work.

Acknowledgements

We thank Ohio Supercomputer Center (Center, 2012) for computing resources. This work was supported in part by NSF CRII award (RI-1755898) and DARPA through the ARO (W911NF-17-C-0095). The content of the information in this document does not necessarily reflect the position or the policy of the U.S. Government, and no official endorsement should be inferred.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*.

Eneko Agirre, Aitor Gonzalez-Agirre, Inigo Lopez-Gazpio, Montse Maritxalar, German Rigau, and Larraitz Uria. 2016. Semeval-2016 task 2: Interpretable semantic textual similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval)*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ohio Supercomputer Center. 2012. Oakley supercomputer. `http://osc.edu/ark:/19495/hpc0cvqn`.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2017. Unsupervised learning of task-specific tree structures with tree-LSTMs. *arXiv preprint arXiv:1707.02786*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP)*.

Reza Ghaeini, Sadid A Hasan, Vivek Datla, Joey Liu, Kathy Lee, Ashequl Qadir, Yuan Ling, Aaditya Prakash, Xiaoli Z Fern, and Oladimeji Farri. 2018. DR-BiLSTM: Dependent reading bidirectional LSTM for natural language inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Yichen Gong, Heng Luo, and Jian Zhang. 2017. Natural language inference over interaction space. *arXiv preprint arXiv:1709.04348*.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (NAACL)*.

Shankar Iyer, Nikhil Dandekar, and Kornl Csernai. 2017. First Quora Dataset Release: Question Pairs. In *https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems (NIPS)*.

Wuwei Lan and Wei Xu. 2018. The importance of subword embeddings in sentence pair modeling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of The 2017 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Modelling interaction of sentence pair with coupled-LSTMs. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems (NIPS)*.

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017a. Inter-weighted alignment network for sentence pair modeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017b. Disan: Directional self-attention network for RNN/CNN-free language understanding. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*.

Gaurav Singh Tomar, Thyago Duque, Oscar Täckström, Jakob Uszkoreit, and Dipanjan Das. 2017. Neural paraphrase identification of questions with noisy pretraining. In *Proceedings of the First Workshop on Subword and Character Level Models in NLP*.

Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for qa. In *Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI)*.

John Wieting and Kevin Gimpel. 2017. Revisiting recurrent networks for paraphrastic sentence embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Wei Xu, Chris Callison-Burch, and William B. Dolan. 2015. SemEval-2015 Task 1: Paraphrase and semantic similarity in Twitter (PIT). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval)*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Wenpeng Yin, Hinrich Schtze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics (TACL)*.

Authorless Topic Models: Biasing Models Away from Known Structure

Laure Thompson
Department of Computer Science
Cornell University
Ithaca, NY 14853
laurejt@cs.cornell.edu

David Mimno
Department of Information Science
Cornell University
Ithaca, NY 14853
mimno@cornell.edu

Abstract

Most previous work in unsupervised semantic modeling in the presence of metadata has assumed that our goal is to make latent dimensions *more* correlated with metadata, but in practice the exact opposite is often true. Some users want topic models that highlight differences between, for example, authors, but others seek more subtle connections across authors. We introduce three metrics for identifying topics that are highly correlated with metadata, and demonstrate that this problem affects between 30 and 50% of the topics in models trained on two real-world collections, regardless of the size of the model. We find that we can predict which words cause this phenomenon and that by selectively subsampling these words we dramatically reduce topic-metadata correlation, improve topic stability, and maintain or even improve model quality.

1 Introduction

Unsupervised semantic models are a popular and useful method for inferring low-dimensional representations of large text collections. Examples of such models include latent semantic analysis (Deerwester et al., 1990) and word embeddings (Bengio et al., 2003), but for this work we will focus on statistical topic models (Hofmann, 1999; Blei et al., 2002), which are used to infer word distributions that correspond to recognizable themes. In practice, collections are often constructed by combining documents from multiple sources, which may have distinctive style and vocabulary. This heterogeneity of sources leads to a serious but rarely studied problem: the strongest, most prominent patterns in a collection may simply repeat the known structure of the corpus. Instead of finding informative, cross-cutting themes, models simply repeat the distinctive vocabulary of the individual sources. The model in this case is "correct" in that it has detected the strongest dimensions of variation, but it tells us nothing we did not already know.

As a motivating example, we focus on models trained on novels, where it is known that inferred topics are often simply names of characters and settings (Jockers, 2013). The words *Harry*, *Ron*, and *Hermione* look to the algorithm like the basis of an ideal topic because they occur very frequently together but not in other contexts. But this topic only tells us which books within a larger corpus are part of the *Harry Potter* series; themes like friendship, adolescence, and magic remain hidden. This phenomenon is not limited to fiction: we also include a case study of opinions from US state supreme courts. Unlike examples from fiction, Maine and Utah both exist in the same universe, but exhibit specific regional term use.

We begin by demonstrating that the problem of overly source-specific topics is both substantial and measurable. We present three metrics that provide related but distinct views of source specificity. These metrics are orthogonal to existing metrics of topic semantic quality: uselessly source-specific topics are often still highly coherent and meaningful. These metrics are also inversely related to commonly-used document classification evaluations. Learning 20 newsgroup-specific topics from 20 Newsgroups may be informative as an evaluation, but in practice users are rarely unaware of such structure.

Finally, we present a simple but effective method for reducing the prevalence of source-specific topics. This method relies on probabilistically subsampling words that correlate with known source metadata, and

Proceedings of the 27th International Conference on Computational Linguistics, pages 3903–3914
Santa Fe, New Mexico, USA, August 20-26, 2018.

is related to subsampling methods that have been highly effective in word embeddings (Mikolov et al., 2013; Levy et al., 2015). The best of the proposed methods substantially reduces source-specific topics, increases topic differentiation without increasing model complexity, and improves topic stability.

2 Related Work

The common assumption of prior work on metadata-aware topic modeling has been that metadata provides valuable hints that can be used to improve topics. Several methods use document metadata to influence document-level topic distributions. The author-topic model (Rosen-Zvi et al., 2004), relational topic model (Chang and Blei, 2009), and labeled LDA (Ramage et al., 2009) extend LDA by directly incorporating a particular type of metadata (e.g. author information, document links, user-generated tags) into the model. Others, like factorial LDA (Paul and Dredze, 2012), Dirichlet-multinomial regression topic models (Mimno and McCallum, 2008), and structural topic models (Roberts et al., 2014) incorporate more general categories of metadata. All of these aim to *increase* dependence between topics and metadata. In contrast, our goal is to make topics *independent* of specified metadata.

Other research makes topic-word distributions sensitive to document-level metadata. The special words with background model (Chemudugunta et al., 2006) incorporates document-specific word distributions into LDA, while cross-collection LDA (Paul, 2009) incorporates collection level word distributions. The topic-aspect model (Paul and Girju, 2010) extends LDA to include a mixture of aspects of documents such that aspects affect all topics similarly. Although these models may be able to sequester author-specific words, there is no reason to expect that those words will not also drag along general, cross-cutting words.

In this paper we focus on ways to explicitly identify words that bias topics towards a specific metadata tag and modify the input corpus for an algorithm to reduce their effect. Researchers have often dismissed this sort of data curation as unprincipled and heuristic "preprocessing." More recent work (Denny and Spirling, 2016; Boyd-Graber et al., 2014) emphasizes that *meta-algorithms* for data preparation can greatly affect the intrinsic model quality and human interpretability of topic models.

3 Collections and Models

We collected two real-world corpora that combine text from multiple distinct sources: science fiction novels and U.S. state supreme court opinions.[1]

Science Fiction (SCI-FI). We selected 1206 science fiction novels by 245 authors based on award nominations and curated book lists hosted on Worlds Without End.[2] We consider each author as a source, and treat collaborations as distinct sources. We augmented the corpus with other established authors to increase the diversity of author gender and ethnicity. The novels span from the early 1800s to the present day. Most of these works are currently protected by copyright, so rather than full text we obtained page-level word frequency statistics from the HathiTrust Research Center's Extracted Features Dataset (Capitanu et al., 2016). This data indicates, for example, that page 227 of *Dune* contains one instance of the word *storm* as a noun. Following previous work (Jockers, 2013) we divide volume-length works into page-level segments, omitting headers and footers.

Corpus	Authors	Docs	Types	Avg Len
SCI-FI	245	327K	132K	153
COURTS	50	52K	89K	1039

Table 1: Corpus statistics for the number of authors, documents, and word types, as well as average document length. Document and word type counts are listed in thousands (K).

U.S. State Supreme Courts (COURTS). Each U.S. state has a supreme court that decides appeals for decisions made by lower state courts. In this collection each document is a court opinion, written by the court after the completion of a case, summarizes the case and judgment. We treat each state court as a source, expecting that courts use geographically specific language (e.g. Colorado, Denver, Colo., Boulder)

[1] Coda and data is available at `https://github.com/laurejt/authorless-tms`.

[2] `https://www.worldswithoutend.com/lists.asp`

that is not relevant to the legal content of opinions. We examine court opinions for all 50 state supreme courts for cases filed from 2012 through 2016.[3]

Data Preparation. We apply the same initial treatment to both corpora. Tokens are three or more letter characters with possible internal punctuation (excluding em- and en-dashes). Words are lower-cased. To deal with globally frequent terms, we remove words used by more than 25% of documents in a corpus. To reduce the computational burden of a large vocabulary, we remove words occurring in fewer than five documents. We remove all documents with fewer than 20 tokens. This process removes 706 pages and 9192 court opinions from our starting science fiction and state courts corpora.

We train LDA models using Mallet (McCallum, 2002) with hyperparameter optimization occurring every 20 intervals after the first 50. We set the number of topics to be on the same order as the number of sources, so for SCI-FI we use $K \in [125, 250, 375]$ and for COURTS we use $K \in [25, 50, 75]$.

4 Evaluating Topic-Author Correlation

We introduce three ways to measure the source-specificity of topics. For concreteness we will use the terms "source" and "author" interchangeably, but a document's source could be any categorical variable. We want to identify topics that are used by relatively few authors, and more specifically topics whose "meaning" is unduly influenced by the contributions of relatively few authors.

Given a collection of D documents written by A authors such that each document d is written by a single author a, we train an LDA topic model with K topics. Then for each word token i in document d we have both a word type w_{id} and a posterior distribution over its token-level topic assignment z_{di}. For clarity of presentation we can assume a single topic assignment for each token and view the corpus as a data table with three columns: word type w, topic z, and author a. By summing over rows of this table we can define marginal count variables for authors $N(a)$ and topics $N(k)$ as well as joint count variables for the count of a word in a topic $N(w, k)$, a topic in an author $N(k, a)$, and a word in a topic in an author $N(w, k, a)$. A maximum likelihood estimate of the probability of word w given topic k is $P(w \mid k) = \frac{N(w,k)}{N(k)}$.[4]

We note that these statistics must be defined at the token level. As in Mimno and Blei (2011) we are looking for violations of the assumption that $\Pr(w \mid k) = \Pr(w \mid d, k)$. Gibbs sampling algorithms typically preserve token-level information in the form of sampling states, but EM-based algorithms often preserve only document-topic distributions θ_d and topic-word distributions ϕ_k. We can estimate the posterior distribution over topic assignments for each token in document d with word type w as $\Pr(z \mid d, k) \propto \sum_k \phi_k(w)\theta_d(k)$, and generate sparse representations by sampling from this distribution.

Author Entropy. We begin by measuring a topic's author diversity—how evenly its tokens are spread across authors—using the conditional entropy of authors given a topic (Eq. 1). Topics whose tokens are largely concentrated within a few authors will have low entropy, while topics more evenly spread across many authors will have high entropy. With asymmetric hyperparameter optimization we find that the most frequent topics (large α_k) have high author entropy, but topics with high author entropy can have a wide range of frequencies: topics can be both rare and well-distributed.

$$H(A \mid k) = \sum_a \Pr(a \mid k) \log_2 \Pr(a \mid k) = \sum_a \frac{N(a,k)}{N(k)} \log_2 \frac{N(a,k)}{N(k)} \quad (1)$$

While author entropy provides a general sense of author diversity, it does not take into account the expression of topics by authors. Content-based evaluation is especially important because many collections are not well balanced across authors. The fact that a topic is not balanced across authors does not necessarily imply that it is problematic. A novel about the voyages of a ship captain may contain a large proportion of words about sea travel and ships, while a novel that contains one minor character who is a ship captain may contain a small proportion of the same language, used in the same way. We therefore

[3] https://www.courtlistener.com

[4] We do not use Dirichlet smoothing for the purposes of this work for simplicity and to make more reliable comparisons across varying vocabulary sizes. Results using smoothing are similar.

need to be able to distinguish two cases: first, a topic that is consistent across authors but that is used at different rates by different authors, and second, a topic that is not only used at different rates but has different contents across authors. In the first case we can accurately use a topic to "stand for" a particular concept of interest, while in the second case we would get a false impression of the contents of documents, because the expression of the topic in the minority authors differs from the topic as a whole.

To differentiate expected author imbalance from pathological cases, we calculate Jensen-Shannon divergence between a topic's word distribution as estimated from the full collection $\Pr(w|k)$ and two distributions that have been transformed to reduce the influence of the most prominent authors. If the topic has low author correlation then there will be little divergence between the original distribution and its transformation. This method mimics a technique for identifying "junk" topics by AlSumait et al. (2009).

Minus Major Author. The first transformed distribution M (Eq. 2) recalculates the probability of words based on all documents *except* those written by the majority author. If a topic is consistent across authors then the presence or absence of its largest author contribution (labeled a_{major}) should have little effect on the topic's word distribution. The larger the resulting divergence, the more influence the major author has over the topic. Unlike author entropy, this technique does not inherently favor balanced distributions of authors; a very author-imbalanced (low entropy) topic can still have a low minus major author divergence if the dominating author's contribution agrees with the remaining topic tokens.

$$\Pr(w \mid M_k) = \Pr(w \mid \neg a_{major}, k) = \frac{N(w,k) - N(w, a_{major}, k)}{N(k) - N(a_{major}, k)} \tag{2}$$

Balanced Authors. The second transformed distribution B (Eq. 3) treats the contribution of each author equally, no matter how many words in that topic the author produces. The minus-major metric is most sensitive to the case where a single author dominates a topic, but does not handle the case where a small group of authors dominates. Using the balanced transformation we measure the similarity of each author contribution. The larger the resulting divergence between the original and transformed word distributions, the larger the variance in contributing author token usage.

$$\Pr(w \mid B_k) \propto \sum_a \Pr(w \mid k, a) = \sum_a \frac{N(w,k,a)}{N(k,a)} \tag{3}$$

We check the validity of our metrics by evaluating topic models trained on SCI-FI for a wide range of topic sizes (125–1000). As seen in Figure 1, all three measures produce bimodal distributions for all topic sizes, combining highly author-specific topics and more general cross-cutting ones. The proportion of cross-cutting topics remains fairly constant across topic sizes: for all of these models, over 50% of topics fall in the source-specific range. We emphasize that source-specific topics are not necessarily "bad". If the structure of the corpus were not known, these topics would provide a highly useful and coherent insight into that structure. But if, as is typical, the structure *is* known, more than half of the statistical capacity of these models is wasted learning distributions that simply reiterate known structure, regardless of the number of topics.

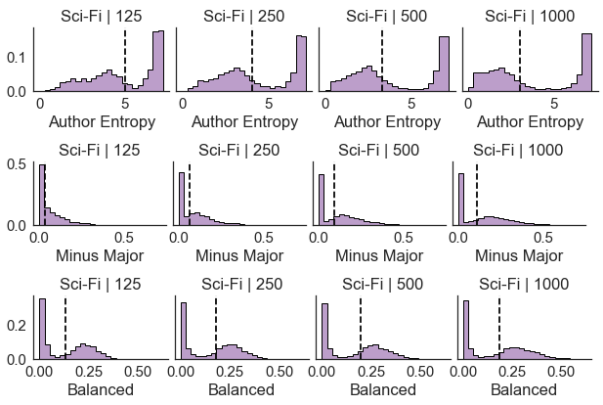


Figure 1: Author entropy, minus major author divergence, and balanced author divergence for topics in topic models trained on SCI-FI. Dashed lines indicate medians. Increasing the number of topics in a model does not reduce the proportion of author-specific topics.

While all three measurements produce similarly shaped distributions, they do not always agree in detail. Table 2 shows example topics that provide intuition for these differences. At the extremes, Topic A is a general, cross-cutting topic while Topic G is dramatically author-specific. While all three metrics score well for Topics A and B, in Topic B the word

Topic	Entropy	Minus Major	Balanced	Top Words
A	6.79	0.00067	0.017	school professor work university years research science students student college study class year history scientific theory young new field physics
B	6.67	0.0047	0.032	doctor paul hospital nurse patient medical patients doctors room ward bed drugs treatment clinic drug case mental sick therapy medicine
C	5.44	0.043	0.17	jack emma **malenfant** trip janet michael ing **wireman leonard nemoto** sally **jeannine reynolds render manekato mccann** runners thi **joshua**
D	5.31	0.027	0.13	sand **pirx** mars desert roger dust rock **bass** dunes crater martian **jeffries kirov** dune **sweeney eileen** rocks canyon lava camp
E	3.42	0.080	0.16	robot robots andrew human **cully** susan **calvin** brain being **powell donovan** law **moldaug** sir **drake positronic bogert lanning** humans three
F	2.32	0.067	0.083	old night yes cried town last men **rocket** god years hands house upon stood wind boy shut door let dark
G	0.28	0.35	0.32	**f'lar lessa weyr robinton hold dragon f'nor lord dragons benden rider bronze harper thread mnementh brekke ramoth fax fort queen**

Table 2: Topics from a 250-topic model trained on SCI-FI and their corresponding measures of author entropy, minus major author, and balanced authors. Underlined values indicate poor quality scores and bolded terms indicate word types with low (< 1) author entropy within the topic.

paul seems out of place, but it is common enough in several authors that its word-level author entropy is not low. Topics E and G both score poorly in all three metrics, and both are highly specific to single authors (Isaac Asimov and Anne McCaffrey). But while G is clearly and exclusively names and settings, E contains the common terms *robot*, *robots*, and *human*, and could be confused for a general topic on artificial intelligence.

The metrics are also enlightening when they disagree. Topic C has high author entropy, but only because it mixes highly author-specific words from several different authors. Since each author's contribution differs from the others it scores poorly on the two content-based metrics. Topic D is partially about Mars, but also contains author-specific character names from stories set on Mars. No single author dominates, but the contributions of each author look different. Topic F is so highly correlated with Ray Bradbury that its entropy is low and it looks different when his contribution is removed, but its words are sufficiently general that Bradbury's use of the topic is close to the other authors' (minimal) use.

5 Contextual Probabilistic Subsampling

In this section we present interventions that predict the effect of words and contexts, and modify an input corpus to reduce the number of overly author-specific topics in resulting models. We hypothesize that this problem is due to *burstiness* (Doyle and Elkan, 2009): words that are globally rare, but locally frequent. Dampening the author-specificity of individual word types may reduce their connection to document sources. We therefore evaluate context-specific subsampling prior to modeling, with parameters defined based on tail probabilities of word-specific parametric models.

In selecting this particular approach we follow three design principles that we believe maximize use in actual practice. First, we want interventions to be minimal and have the least possible disruption to current work processes. We therefore choose to focus on meta-algorithms for data preparation that are compatible with but independent from existing, widely implemented inference algorithms. Second, we want any user-specified parameter choices to be simple and intuitive. Although we find that entropy is a useful diagnostic metric, information theoretic metrics such as mutual information are difficult for non-experts to interpret correctly, and critical values can differ widely across collections and dimensionalities. Third, we want both the choice of interventions and the effects of interventions to be transparent to users. We initially considered methods such as adversarially trained autoencoders, but we find that directly subsampling words is much faster, simpler, and easier to explain.

Identifying Author Specific Terms. The simplest way to find author-specific terms is to find terms unique to an author. The SCI-FI collection contains an unusual number of author-specific coinages, but words used by many authors can still be highly correlated with a particular author. We therefore estimate parametric distributions for each term and compare author-specific term proportions to this

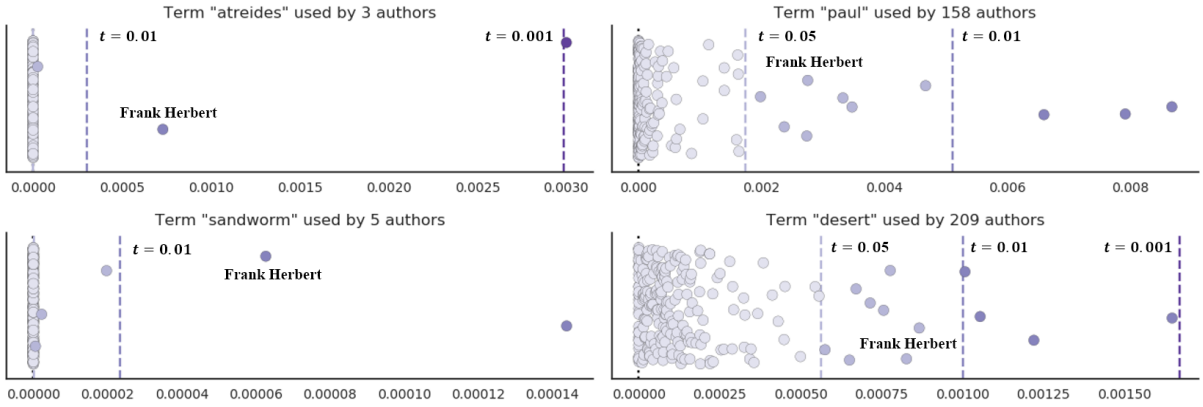


Figure 2: Reasonable threshold values t flag both rare words (left) and common words being used in author-specific ways (right). Each point represents the relative frequency of a term (x-axis) for an author (y-axis) in SCI-FI.

distribution. For each word type w, we calculate the sample mean $\bar{x}_w$ and variance s_w^2 and construct a gamma distribution Γ_w with shape $k = \bar{x}_w^2/s_w^2$ and rate $\theta = s_w^2/\bar{x}_w$. Similar to a significance test, given a user-specified probability threshold t we can define a critical term proportion value under Γ_w

$$\Pr[\Gamma_w \leq f_w^*] \leq 1 - t. \tag{4}$$

A word w is thus considered too specific to an author a if a's usage is too unlikely to occur according to Γ_w. Specifically, this occurs when the frequency $f_{w,a}$ is larger than the cutoff frequency f_w^* defined in Eq. 3. This method satisfies our design goals of simplicity and transparency: the threshold is intuitive and can be adjusted to change how aggressively words are flagged for curation.

Figure 2 shows two character names and nouns from Frank Herbert's *Dune*, where one name and noun are rare and the others are frequent. We see that the rare words *atreides* and *sandworm* are significant to Frank Herbert for $t = 0.01$: there is essentially no "normal" level of use of these words in other authors. Herbert also uses the more common terms *paul* and *desert* more than expected, but to a lesser extreme.

Determining Stop Rates. How we choose to dampen author-specific words is as important as how we detect them. If we globally removed these words using a traditional stoplist, we would lose a substantial portion of the vocabulary. A more sophisticated approach is to construct a stoplist for each author. In this case, words are only removed from contexts in which they are statistically overrepresented. For rare terms, where there is no middle ground between significant use and no use at all, this contextual treatment is effectively the same as a traditional stoplist. But for a word with more widespread use, that word would disappear only from contexts with abnormally high usage.

While this technique avoids erasing the majority of a collection's vocabulary, it leads to a paradoxical situation where a word that is thematically central to a work occurs *less* frequently in that work than in other works. Entirely removing *desert* from Frank Herbert or *robot* from Isaac Asimov would reduce the model's ability to identify relevant themes.

To find a middle ground, we use probabilistic subsampling to reduce outlier author use to something more in line with the collection's overall usage. We use the same threshold t to set subsampling rates. For a word type w and author a the probability of stopping a token of type w in a is

$$\Pr(\text{Stop } w \text{ in } a) = 1 - f_w^*/f_{w,a}. \tag{5}$$

The threshold t specifies when an author's use of a word is too extreme for our model Γ_w. If we reduce these outlier frequencies to their corresponding cutoff frequencies f_w^*, they will be set to the largest below-threshold frequency dictated by Γ_w. We construct our subsampling rates such that in expectation new author frequencies will equal their corresponding threshold frequency from the original distribution.[5]

[5]Iteratively reevaluating Γ_w leads to an unstable "race to the bottom."

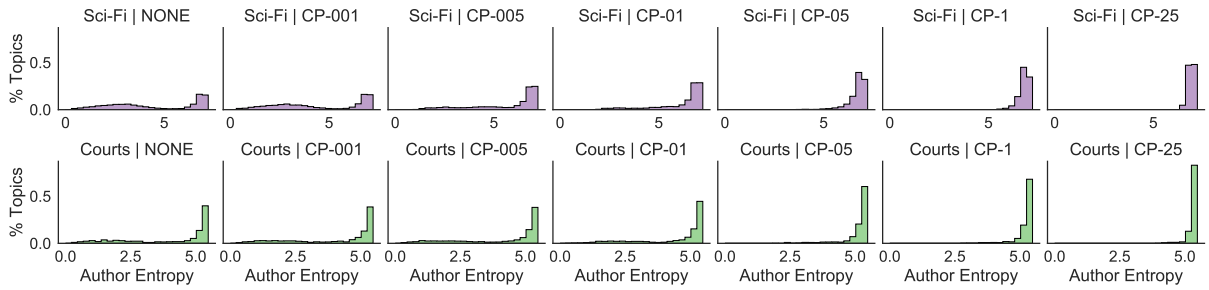


Figure 3: Increasing the threshold t for contextual probabilistic (CP) subsampling results in more topics with high dispersion over authors.

6 Results

Unless otherwise noted, we refer to models with a topic size of 250 for SCI-FI and 50 for COURTS, and set the hyperparameter t of context-based methods to 0.05. We refer to a treatment with no intervention beyond standard stopword removal as **NONE**. We compare these models to three classes of curation methods, each with varying parameters. **AF-[n]** removes all terms that are used by at most n authors. **C-[t]** removes any term from author a's context whose frequency $f_{w,a}$ exceeds significance threshold t with respect to distribution Γ_w. **CP-[t]** subsamples terms according to Eq. 5. We train 10 runs with random initializations for each parameter setting.

Subsampling reduces topic-metadata correlation. We begin by measuring how well the curation techniques reduce the formation of author-correlated topics. We find that while removing words with low author frequency has little effect (not shown), contextual methods greatly reduce the formation of "bad" topics according to all three measures. As expected, the value of the threshold t affects performance of the context-based methods. In Figure 3, we see that lowest values of t are ineffective; $t = 0.001$ is hardly distinguishable from NONE and $t = 0.005$ is on par with low author frequency stoplists. We observe that settings of $t \geq 0.05$ perform very well, and choose this value as a default in our public code release.

Subsampling before inference does more than change the appearance of topics, it changes the content of the inferred topics. To test whether subsampling after inference has the same effect we construct ten additional models by *post hoc* stopping the 250-topic trained models for NONE-treated SCI-FI to match token-for-token the CP-05 curated versions. We find that *post-hoc* removal has little effect on topic-metadata correlation; over twenty percent of topics are dominated by a single author with the worst having 96.4% of tokens contributed by one author.

Semantic quality is preserved. We define author-specificity as a property orthogonal to model quality: there is nothing fundamentally wrong with a topic full of character names outside the context of specific user needs. But ideally in reducing the prevalence of overly author-specific topics we would replace them with equally meaningful ones. We measure semantic quality of topics using Mimno et al. (2011)'s topic coherence metric as reported by Mallet. This metric measures the tendency for the most probable (top) words of a topic to cooccur. A topic k with m top words $w_{k,1}, \ldots, w_{k,m}$ has topic coherence

$$\sum_i \sum_{j<i} \log \frac{D(w_i, w_j)}{D(w_i)} + \beta, \tag{6}$$

where D represents the number of documents containing a word or word pair and β is the LDA hyperparameter for topic-word smoothing. Large negative values indicate that the top words of a topic seldom cooccur, while values close to zero indicate that the top words frequently coccur.

We find that despite substantial changes in topic content, corpus modification has no consistent effect on the semantic quality of topics. In Figure 4, we find that all curation methods except CP-001 have significantly higher mean topic coherence than NONE for SCI-FI. Contextual methods with $t \geq 0.05$ have the highest coherence. For COURTS, topic coherence is maintained across treatments, except for the most aggressive interventions C-05 and C-1.

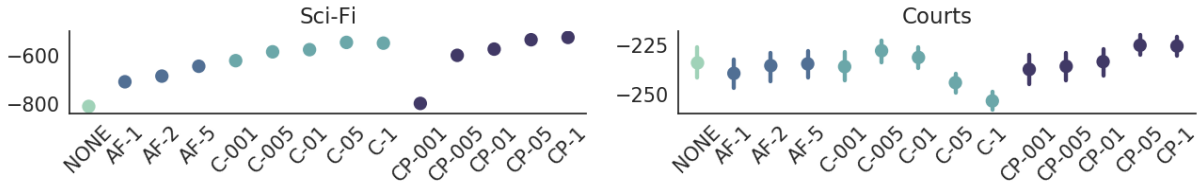


Figure 4: Contextual probabilistic subsampling improves mean topic coherence for SCI-FI despite the removal of frequent words. Coherence degrades under context curation for COURTS.

Corpus damage is reduced. All things being equal, we want to modify the input collection minimally, both in terms of vocabulary and actual document content. Figure 5 confirms that contextual curation has the highest type and token loss across corpora, because it completely removes all instances of a word type in a context. This may partially explain the dramatic loss of model quality for these specific treatments.

Contextual probabilistic subsampling removes more tokens than author frequency cut-offs, but better preserves the vocabulary. For thresholds $t \leq 0.01$, contextual probabilistic subsampling removes fewer word types than any of the author frequency cut-off methods. However, there is less agreement across corpora for $t \geq 0.05$. For SCI-FI, these methods remove more types than AF-5, while the reverse is true for COURTS. This discrepancy might arise from differences in the relative size of collection sources—some authors write more than others, some courts issue more opinions—and vocabulary use.

Subsampling affects more than names. Character names are the most prominent motivating example for this work, so it is reasonable to ask whether named-entity tagging or even a simple part-of-speech (POS) filter would be sufficient. To check whether we are just removing proper nouns, we compare the frequency of four general POS categories: common nouns, proper nouns, verbs, and adjectives. These make up 37%, 10%, 27%, 13% of all tokens respectively in SCI-FI. Figure 6 shows the proportion of tokens removed from each category for each curation method. Unsurprisingly, proper nouns make up a large proportion in all cases, but contextual methods also remove substantial numbers of tokens across all word groups.

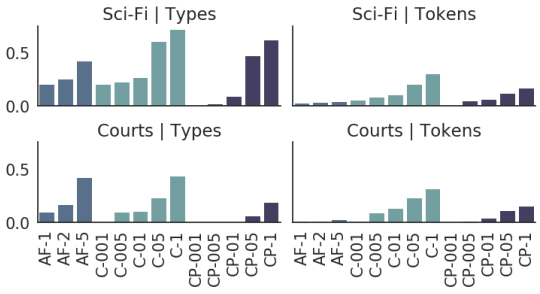


Figure 5: Proportional loss of removed word types and tokens. Contextual probabilistic subsampling does substantially less damage than contextual curation.

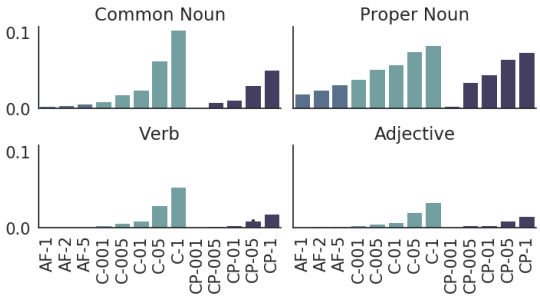


Figure 6: Proportion of SCI-FI tokens removed across part-of-speech groups. Contextual methods remove tokens from all groups.

Subsampling increases stability and specificity. We find that removing author-specific terms using contextual probabilistic subsampling greatly mitigates the formation of author-correlated topics, but what do these models learn instead? Are they augmenting the set of uncorrelated topics found within the untreated models, or are they perhaps identifying entirely new structure? More importantly, what are the characteristics of the newly formed or persisting author-correlated topics? To answer these questions, we perform pairwise comparisons of the topic-word distributions from different models using Jensen-Shannon divergence to find the most likely of topic correspondences. By linking these topics together, we can gain a sense of which topics persist across treatments, which are refined or split, and which are lost entirely. We focus on SCI-FI since it has larger models, but we will highlight similar analysis for COURTS.

Before making any inter-treatment comparisons, we examine topic stability internally within each

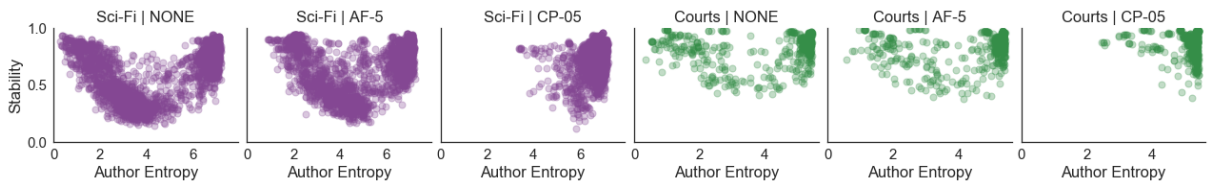


Figure 7: Topic Stability and Entropy for SCI-FI ($K = 250$) and COURTS ($K = 50$). AF-5 has little effect. Many of the low-entropy topics avoided by CP-05 are highly unstable.

treatment. We define stability as the average similarity between a topic and its nearest equivalent from each of the nine other trained models for a treatment. More formally, the stability of topic k_i from the ith instance of a model is

$$\text{Stability}(k_i) = 1 - \frac{1}{9} \sum_{j \neq i} \min_{k_j} JSD(P(w \mid k_i), P(w \mid k_j)) \tag{7}$$

where JSD is Jensen-Shannon divergence. A topic stability close to one implies that a topic persists across runs, while a value close to zero implies that a topic is *ephemeral*—observed once and unlikely to be seen again across random initializations.

High stability does not imply author-specificity. In Figure 7, we see that the most stable topics tend to have either maximal or minimal author entropy, while the most unstable topics have middling values. The unstable topics tend to capture a mixture of disjoint structures as we saw in topics C and D from Table 2. This also occurs (but to a lesser extent) in COURTS with topics containing many distinct regional terms (S1: *s.w oklahoma tenn kan ind n.e indiana app tennessee o.s*) or containing a mixture of a general and state-specific concept (S2: *school wyo miss wyoming mississippi ann education students hill student*). Thus, the most stable topics are the most apparent by being very context specific or most cross-cutting.

Now that we have evaluated the stability of topics under the baseline NONE treatment, we can use minimum divergence to align those topics with topics trained under the CP-05 subsampling treatment. Unstable NONE topics are generally very distant from their nearest CP-05 counterparts. Of our example topics in Table 2, C and D are the most unstable at 0.39 and 0.42 respectively. Topic C diverges heavily (0.87) from its closest CP-05 match, while aspects of D are echoed in its nearest match *sand desert rock mountains mountain dust land surface plain water* (0.53). COURTS topic S2 is also more distantly associated (0.63) with an education/administration topic: *board school commission administrative agency plan department board's education regulations*. Over 95% of NONE topics with high stability and high author entropy are linked to a CP-05 topic with divergence less than 0.5. Topic A has a close match (*professor university college student students research school science work years*) at 0.23. A appears to have become more specific in CP-05 by splitting into two additional topics that echo other aspects, namely teaching children (0.54) and scientific research (0.55).

The case of stable, low entropy NONE topics is harder to interpret. While half of these topics are far from their CP-05 match (> 0.7), 16% have divergences of less than 0.4. Topic G matches well to *lord hold between master queen star enough turns high good* (0.3) which is both very stable and CP-05's lowest author entropy topic (64.1% from Anne McCaffrey).[6] While these topics have not been prevented entirely, they have been largely mitigated.

The topics in CP-05 that are the most dissimilar from topics within NONE demonstrate that this treatment adds differentiation. We find that overall 50% of CP-05 topics have a large divergence (> 0.7) with the NONE topics. Some of these divergent topics consist of names, but these groupings might indicate regional or temporal naming patterns. In other cases, we encounter new and interesting topics such as an authentic robots topic (*machine robot machines robots human mechanical metal brain men built*), which matches to both a general computer topic and example topic E (Asimov). We also find a new topic on magic and witchcraft (*magic ghost demon evil witch demons power spell magician ghosts*) whose

[6]The topic is common words used in specific ways: a *hold* is a fortified settlement, dragons teleport by going *between*.

closest match is a general religion topic *god gods religion world religious ancient temple people faith these*. In fact, the term *witch* never appears as a top-20 term for any topic within the 250-topic NONE models. These topics may appear for NONE when we increase the topic size to $K = 1000$, but at the cost of a much larger model and with no guarantee against intruding character names.

Subsampling produces cross-cutting topics. While our topics score well quantitatively, how humanly interpretable and useful are the resulting topics? Are they actually cross-cutting in nature? We address these questions by more closely examining topics generated by the CP-05 subsampling treatment. We can explore the collection by sorting authors and individual novels within topics.

The highest frequency topics from the *NONE* treatment are largely preserved by *CP-05*. These topics by their nature are very cross-cutting and filled with frequent, general words. Despite this extreme generality they can provide a way to analyze passages representing high-level discourse concepts such as inquiry (*why asked ask answer question want questions should does because*) and the description of events and time (*during such most these course because happened effect period result*).

The mid-frequency topics are more concretely thematic in nature. We find a topic describing empire, politics, and history (*empire world power people war new government history political under*) which is associated with Doris Lessing's *Canopus in Argos* series, Isaac Asimov's *Foundation* series, and Kim Stanley Robinsons's *The Years of Rice and Salt*. In line with the science fiction genre, these novels focus on expansive future and alternative histories. We also find a topic on language (*language words english speak word understand spoke speech languages talk*). The most prominent authors in the topic—Robert A. Heinlein, Robert Silverberg, and Poul Anderson—are among the five most prolific authors in SCI-FI, which suggests the generality of the topic. Notably the most prominent volumes are by none of these authors: *Babel-17* by Samuel R. Delany, *Native Tongue* by Suzette Haden Elgin, and *Changing Planes* by Ursula K. Le Guin. All three include the social and political language as a major plot point. These three works are fundamentally tied confirming that this topic embodies a cross-cutting linguistic theme.

Looking more closely at the lower frequency robots topic (*machine robot machines robots human mechanical metal brain men built*), we find that it is both topically cohesive and cross-cutting. The five most-represented authors all have works heavily related to artificial intelligence: Isaac Asimov, Robert Silverberg, Stanisław Lem, Clifford D. Simak, and Philip K. Dick. The most-represented volumes tell a similar story with *Men and machines* by Robert Silverberg, *The complete robot* by Isaac Asimov, and *The Humanoids* by Jack Williamson holding the top three ranks. Reassuringly, there are well-represented novels by less-represented authors such as *The Starchild Trilogy* by Fredrick Pohl and Jack Williamson. The low frequency of this topic is surprising given the presence in the collection of robot-related novels, especially works by Isaac Asimov. This discrepancy revealed that an Asimov-specific topic (*human being law might must such without may robot beings*) has persisted. Many authors receive a non-negligible token representation, but Asimov's token count is still a factor of ten larger than the second most prominent author (Robert A. Heinlein).

7 Conclusion

We present a formal definition of the problem of overly source-specific topics, three evaluation metrics to measure the degree of source-specificity, and a simple text curation meta-algorithm that dramatically reduces the number of source-specific topics. This approach has immediate practical application for the many collections that combine multiple distinct sources, but it also has important theoretical implications.

We view this work as a preliminary step towards predictive theories of latent semantics, beyond purely descriptive models. Despite ample practical evidence that interventions such as stoplist curation can have significant effects, most previous work has focused on algorithms for identifying a single "optimal" low-dimensional semantic representation. Our results indicate that there are potentially many interventions in text collections that each have distinct but predictable effects on the results of algorithms. Just as biologists use multiple stains to view different aspects of microorganisms using the same microscope, users of text mining algorithms should be able to choose multiple distinct text treatments, each with its own predictable effects, to meet distinct user needs.

Acknowledgements

We would like to thank our reviewers for their helpful feedback and suggestions. We would like to thank Ishaan Jhaveri and Ricardo A. Wilson II for directing our attention to science fiction, as well as Jack Hessel and Alexandra Schofield for their reviews and feedback. The work was supported by NSF #1526155, #1652536, and the Alfred P. Sloan Foundation; and through resources provided by the Hathi Trust Digital Library and the Hathi Trust Research Center.

References

Loulwah AlSumait, Daniel Barbará, James Gentle, and Carlotta Domeniconi. 2009. Topic significance ranking of LDA generative models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 67–82.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2002. Latent dirichlet allocation. In *Advances in Neural Information Processing Systems*, volume 14, pages 601–608.

JL Boyd-Graber, DM Mimno, and D Newman. 2014. Care and feeding of topic models. *Handbook of Mixed Membership Models and Their Applications*, pages 225–254.

Boris Capitanu, Ted Underwood, Peter Organisciak, Timothy Cole, Maria Janina Sarol, and J. Stephen Downie. 2016. The HathiTrust Research Center extracted feature dataset (1.0) [Dataset]. http://dx.doi.org/10.13012/J8X63JT3.

Jonathan Chang and David M Blei. 2009. Relational topic models for document networks. In *AIStats*, volume 9, pages 81–88.

Chaitanya Chemudugunta, Padhraic Smyth, and Mark Steyvers. 2006. Modeling general and specific aspects of documents with a probabilistic topic model. In *Advances in Neural Information Processing Systems*, volume 19, pages 241–248.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391.

Matthew James Denny and Arthur Spirling. 2016. Assessing the consequences of text preprocessing decisions.

Gabriel Doyle and Charles Elkan. 2009. Accounting for burstiness in topic models. In *International Conference on Machine Learning*, volume 26, pages 281–288.

Thomas Hofmann. 1999. Probabilistic latent semantic analysis. In *Uncertainty in Artificial Intelligence*, volume 15, pages 289–296.

Matthew L Jockers. 2013. *Macroanalysis: Digital methods and literary history*. University of Illinois Press.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26, pages 3111–3119.

David Mimno and David Blei. 2011. Bayesian checking for topic models. In *Empirical Methods in Natural Language Processing*, pages 227–237.

David Mimno and Andrew McCallum. 2008. Topic models conditioned on arbitrary features with dirichlet-multinomial regression. In *Uncertainty in Artificial Intelligence*, volume 24, pages 411–418.

David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Empirical Methods in Natural Language Processing*, pages 262–272.

Michael Paul and Mark Dredze. 2012. Factorial LDA: Sparse multi-dimensional text models. In *Advances in Neural Information Processing Systems*, volume 25, pages 2582–2590.

Michael Paul and Roxana Girju. 2010. A two-dimensional topic-aspect model for discovering multi-faceted topics. In *AAAI Conference on Artificial Intelligence*, volume 24, pages 545–550.

Michael Paul. 2009. Cross-collection topic models: Automatically comparing and contrasting text. Master's thesis, University of Illinois Urbana-Champaign.

Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D Manning. 2009. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Empirical Methods in Natural Language Processing*, pages 248–256.

Margaret E Roberts, Brandon M Stewart, Dustin Tingley, Christopher Lucas, Jetson Leder-Luis, Shana Kushner Gadarian, Bethany Albertson, and David G Rand. 2014. Structural topic models for open-ended survey responses. *American Journal of Political Science*, 58(4):1064–1082.

Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The author-topic model for authors and documents. In *Uncertainty in Artificial Intelligence*, volume 20, pages 487–494.

SGM: Sequence Generation Model for Multi-Label Classification

Pengcheng Yang[1,2], Xu Sun[1,2], Wei Li[2], Shuming Ma[2], Wei Wu[2], Houfeng Wang[2]
[1]Deep Learning Lab, Beijing Institute of Big Data Research, Peking University
[2]MOE Key Lab of Computational Linguistics, School of EECS, Peking University
{yang_pc, xusun, liweitj47, shumingma, wu.wei, wanghf}@pku.edu.cn

Abstract

Multi-label classification is an important yet challenging task in natural language processing. It is more complex than single-label classification in that the labels tend to be correlated. Existing methods tend to ignore the correlations between labels. Besides, different parts of the text can contribute differently to predicting different labels, which is not considered by existing models. In this paper, we propose to view the multi-label classification task as a sequence generation problem, and apply a sequence generation model with a novel decoder structure to solve it. Extensive experimental results show that our proposed methods outperform previous work by a substantial margin. Further analysis of experimental results demonstrates that the proposed methods not only capture the correlations between labels, but also select the most informative words automatically when predicting different labels.[1]

1 Introduction

Multi-label classification (MLC) is an important task in the field of natural language processing (NLP), which can be applied in many real-world scenarios, such as text categorization (Schapire and Singer, 2000), tag recommendation (Katakis et al., 2008), information retrieval (Gopal and Yang, 2010), and so on. The target of the MLC task is to assign multiple labels to each instance in the dataset.

Binary relevance (BR) (Boutell et al., 2004) is one of the earliest attempts to solve the MLC task by transforming the MLC task into multiple single-label classification problems. However, it neglects the correlations between labels. Classifier chains (CC) proposed by Read et al. (2011) converts the MLC task into a chain of binary classification problems to model the correlations between labels. However, it is computationally expensive for large datasets. Other methods such as ML-DT (Clare and King, 2001), Rank-SVM (Elisseeff and Weston, 2002), and ML-KNN (Zhang and Zhou, 2007) can only be used to capture the first or second order label correlations or are computationally intractable when high-order label correlations are considered.

In recent years, neural networks have achieved great success in the field of NLP. Some neural network models have also been applied in the MLC task and achieved important progress. For instance, fully connected neural network with pairwise ranking loss function is utilized in Zhang and Zhou (2006). Kurata et al. (2016) propose to perform classification using the convolutional neural network (CNN). Chen et al. (2017) use CNN and recurrent neural network (RNN) to capture the semantic information of texts. However, they either neglect the correlations between labels or do not consider differences in the contributions of textual content when predicting labels.

In this paper, inspired by the tremendous success of the sequence-to-sequence (Seq2Seq) model in machine translation (Bahdanau et al., 2014; Luong et al., 2015; Sun et al., 2017), abstractive summarization (Rush et al., 2015; Lin et al., 2018), style transfer (Shen et al., 2017; Xu et al., 2018) and other domains, we propose a sequence generation model with a novel decoder structure to solve the MLC task. The proposed sequence generation model consists of an encoder and a decoder with the attention

[1]The datasets and code are available at https://github.com/lancopku/SGM

Proceedings of the 27th International Conference on Computational Linguistics, pages 3915–3926
Santa Fe, New Mexico, USA, August 20-26, 2018.

mechanism. The decoder uses an LSTM to generate labels sequentially, and predicts the next label based on its previously predicted labels. Therefore, the proposed model can consider the correlations between labels by processing label sequence dependencies through the LSTM structure. Furthermore, the attention mechanism considers the contributions of different parts of text when the model predicts different labels. In addition, a novel decoder structure with global embedding is proposed to further improve the performance of the model by incorporating overall informative signals.

The contributions of this paper are listed as follows:

- We propose to view the MLC task as a sequence generation problem to take the correlations between labels into account.
- We propose a sequence generation model with a novel decoder structure, which not only captures the correlations between labels, but also selects the most informative words automatically when predicting different labels.
- Extensive experimental results show that our proposed methods outperform the baselines by a large margin. Further analysis demonstrates the effectiveness of the proposed methods on correlation representation.

The whole paper is organized as follows. We describe our methods in Section 2. In Section 3, we present the experiments and make analysis and discussions. Section 4 introduces the related work. Finally in Section 5 we conclude this paper and explore the future work.

2 Proposed Method

We introduce our proposed methods in detail in this section. First, we give an overview of the model in Section 2.1. Second, we explain the details of the proposed sequence generation model in Section 2.2. Finally, Section 2.3 presents our novel decoder structure.

2.1 Overview

First of all, we define some notations and describe the MLC task. Given the label space with L labels $\mathcal{L} = \{l_1, l_2, \cdots, l_L\}$, a text sequence $\boldsymbol{x}$ containing m words, the task is to assign a subset $\boldsymbol{y}$ containing n labels in the label space $\mathcal{L}$ to $\boldsymbol{x}$. Unlike traditional single-label classification where only one label is assigned to each sample, each sample in the MLC task can have multiple labels. From the perspective of sequence generation, the MLC task can be modeled as finding an optimal label sequence $\boldsymbol{y}^*$ that maximizes the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$, which is calculated as follows:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=1}^{n} p(y_i|y_1, y_2, \cdots, y_{i-1}, \boldsymbol{x}) \tag{1}$$

An overview of our proposed model is shown in Figure 1. First, we sort the label sequence of each sample according to the frequency of the labels in the training set. High-frequency labels are placed in the front. In addition, the bos and eos symbols are added to the head and tail of the label sequence, respectively.

The text sequence $\boldsymbol{x}$ is encoded to the the hidden states, which are aggregated to a context vector $\boldsymbol{c}_t$ by the attention mechanism at time-step t. The decoder takes the context vector $\boldsymbol{c}_t$, the last hidden state $\boldsymbol{s}_{t-1}$ of the decoder and the embedding vector $g(\boldsymbol{y}_{t-1})$ as the inputs to produce the hidden state $\boldsymbol{s}_t$ at time-step t. Here $\boldsymbol{y}_{t-1}$ is the predicted probability distribution over the label space $\mathcal{L}$ at time-step $t-1$. The function g takes $\boldsymbol{y}_{t-1}$ as input and produces the embedding vector which is then passed to the decoder. Finally, the masked softmax layer is used to output the probability distribution $\boldsymbol{y}_t$.

2.2 Sequence Generation

In this subsection, we introduce the details of our proposed model. The whole sequence generation model consists of an encoder and a decoder with the attention mechanism.

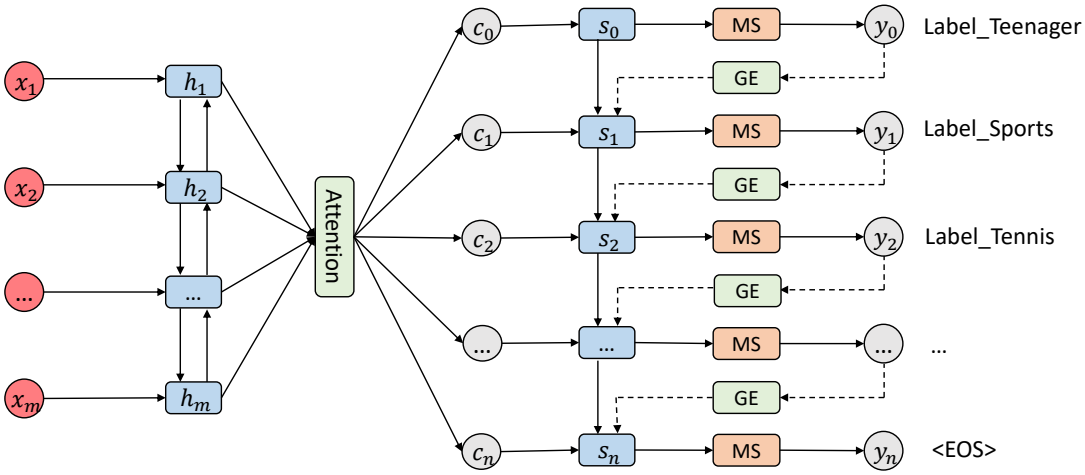


Figure 1: The overview of our proposed model. MS denotes the masked softmax layer. GE denotes the global embedding.

Encoder: Let $(\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m)$ be a sentence with m words and $\boldsymbol{w}_i$ is the one-hot representation of the i-th word. We first embed $\boldsymbol{w}_i$ to a dense embedding vector $\boldsymbol{x}_i$ by an embedding matrix $\boldsymbol{E} \in \mathbb{R}^{k \times |\mathcal{V}|}$. Here $|\mathcal{V}|$ is the size of the vocabulary, and k is the dimension of the embedding vector.

We use a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to read the text sequence $\boldsymbol{x}$ from both directions and compute the hidden states for each word,

$$\overrightarrow{\boldsymbol{h}}_i = \overrightarrow{\text{LSTM}}(\overrightarrow{\boldsymbol{h}}_{i-1}, \boldsymbol{x}_i) \tag{2}$$

$$\overleftarrow{\boldsymbol{h}}_i = \overleftarrow{\text{LSTM}}(\overleftarrow{\boldsymbol{h}}_{i+1}, \boldsymbol{x}_i) \tag{3}$$

We obtain the final hidden representation of the i-th word by concatenating the hidden states from both directions, $\boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}}_i; \overleftarrow{\boldsymbol{h}}_i]$, which embodies the information of the sequence centered around the i-th word.

Attention: When the model predicts different labels, not all text words make the same contribution. The attention mechanism produces a context vector by focusing on different portions of the text sequence and aggregating the hidden representations of those informative words. Specially, the attention mechanism assigns the weight α_{ti} to the i-th word at time-step t as follows:

$$e_{ti} = \boldsymbol{v}_a^T \tanh(\boldsymbol{W}_a \boldsymbol{s}_t + \boldsymbol{U}_a \boldsymbol{h}_i) \tag{4}$$

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^{m} \exp(e_{tj})} \tag{5}$$

where $\boldsymbol{W}_a$, $\boldsymbol{U}_a$, $\boldsymbol{v}_a$ are weight parameters and $\boldsymbol{s}_t$ is the current hidden state of the decoder at time-step t. For simplicity, all bias terms are omitted in this paper. The final context vector $\boldsymbol{c}_t$ which is passed to the decoder at time-step t is calculated as follows:

$$\boldsymbol{c}_t = \sum_{i=1}^{m} \alpha_{ti} \boldsymbol{h}_i \tag{6}$$

Decoder: The hidden state $\boldsymbol{s}_t$ of the decoder at time-step t is computed as follows:

$$\boldsymbol{s}_t = \text{LSTM}(\boldsymbol{s}_{t-1}, [g(\boldsymbol{y}_{t-1}); \boldsymbol{c}_{t-1}]) \tag{7}$$

where $[g(\boldsymbol{y}_{t-1}); \boldsymbol{c}_{t-1}]$ means the concatenation of the vectors $g(\boldsymbol{y}_{t-1})$ and $\boldsymbol{c}_{t-1}$. $g(\boldsymbol{y}_{t-1})$ is the embedding of the label which has the highest probability under the distribution $\boldsymbol{y}_{t-1}$. $\boldsymbol{y}_{t-1}$ is the probability

distribution over the label space $\mathcal{L}$ at time-step $t-1$ and is computed as follows:

$$\boldsymbol{o}_t = \boldsymbol{W}_o f(\boldsymbol{W}_d \boldsymbol{s}_t + \boldsymbol{V}_d \boldsymbol{c}_t) \tag{8}$$

$$\boldsymbol{y}_t = softmax(\boldsymbol{o}_t + \boldsymbol{I}_t) \tag{9}$$

where $\boldsymbol{W}_o$, $\boldsymbol{W}_d$, and $\boldsymbol{V}_d$ are weight parameters, $\boldsymbol{I}_t \in \mathbb{R}^L$ is the mask vector that is used to prevent the decoder from predicting repeated labels, and f is a nonlinear activation function.

$$(\boldsymbol{I}_t)_i = \begin{cases} -\infty & \text{if the label } l_i \text{ has been predicted at previous } t-1 \text{ time steps.} \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

At the training stage, the loss function is the cross-entropy loss function. We employ the beam search algorithm (Wiseman and Rush, 2016) to find the top-ranked prediction path at inference time. The prediction paths ending with the eos are added to the candidate path set.

2.3 Global Embedding

In the sequence generation model mentioned above, the embedding vector $g(\boldsymbol{y}_{t-1})$ in Equation (7) is the embedding of the label that has the highest probability under the distribution $\boldsymbol{y}_{t-1}$. However, this calculation only takes advantage of the maximum value of $\boldsymbol{y}_{t-1}$ greedily. The proposed sequence generation model generates labels sequentially and predicts the next label conditioned on its previously predicted labels. Therefore, it is likely that we would get a succession of wrong label predictions in the following time steps if the prediction is wrong at time-step t, which is also called *exposure bias*. To a certain extent, the beam search algorithm alleviates this problem. However, it can not fundamentally solve the problem because the *exposure bias* phenomenon is likely to occur for all candidate paths. $\boldsymbol{y}_{t-1}$ represents the predicted probability distribution at time-step $t-1$, so it is obvious that all information in $\boldsymbol{y}_{t-1}$ is helpful when we predict the current label at time-step t. The *exposure bias* problem ought to be relieved by considering all informative signals contained in $\boldsymbol{y}_{t-1}$.

Based on this motivation, we propose a new decoder structure, where the embedding vector $g(\boldsymbol{y}_{t-1})$ at time-step t is capable of representing the overall information at $(t-1)$-th time step. Inspired by the idea of the adaptive gate in highway network (Srivastava et al., 2015), here we introduce our global embedding. Let $\boldsymbol{e}$ denotes the embedding of the label which has the highest probability under the distribution $\boldsymbol{y}_{t-1}$. $\bar{\boldsymbol{e}}$ is the weighted average embedding at time t, which is calculated as follows:

$$\bar{\boldsymbol{e}} = \sum_{i=1}^{L} y_{t-1}^{(i)} \boldsymbol{e}_i \tag{11}$$

where $y_{t-1}^{(i)}$ is the i-th element of $\boldsymbol{y}_{t-1}$ and $\boldsymbol{e}_i$ is the embedding vector of the i-th label. Then the proposed global embedding $g(\boldsymbol{y}_{t-1})$ passed to the decoder at time-step t is as follows:

$$g(\boldsymbol{y}_{t-1}) = (\boldsymbol{1} - \boldsymbol{H}) \odot \boldsymbol{e} + \boldsymbol{H} \odot \bar{\boldsymbol{e}} \tag{12}$$

where $\boldsymbol{H}$ is the transform gate controlling the proportion of the weighted average embedding:

$$\boldsymbol{H} = \boldsymbol{W}_1 \boldsymbol{e} + \boldsymbol{W}_2 \bar{\boldsymbol{e}} \tag{13}$$

where $\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{L \times L}$ are weight matrices. The global embedding $g(\boldsymbol{y}_{t-1})$ is the optimized combination of the original embedding and the weighted average embedding by using transform gate $\boldsymbol{H}$, which can automatically determine the combination factor in each dimension. $\boldsymbol{y}_{t-1}$ contains the information of all possible labels. By considering the probability of every label, the model is capable of reducing damage caused by mispredictions made in the previous time steps. This enables the model to predict label sequences more accurately.

Dataset	Total Samples	Label Sets	Words/Sample	Labels/Sample
RCV1-V2	804,414	103	123.94	3.24
AAPD	55,840	54	163.42	2.41

Table 1: Summary of datasets. **Total Samples**, **Label Sets** denote the total number of samples and labels, respectively. **Words/Sample** is the average number of words per sample and **Labels/Sample** is the average number of labels per sample.

3 Experiments

In this section, we evaluate our proposed methods on two datasets. We first introduce the datasets, evaluation metrics, experimental details, and all baselines. Then, we compare our methods with the baselines. Finally, we provide the analysis and discussions of experimental results.

3.1 Datasets

Reuters Corpus Volume I (RCV1-V2)[2]: This dataset is provided by Lewis et al. (2004). It consists of over 800,000 manually categorized newswire stories made available by Reuters Ltd for research purposes. Multiple topics can be assigned to each newswire story and there are 103 topics in total.

Arxiv Academic Paper Dataset (AAPD)[3]: We build a new large dataset for the multi-label text classification. We collect the abstract and the corresponding subjects of 55,840 papers in the computer science field from the website[4]. An academic paper may have multiple subjects and there are 54 subjects in total. The target is to predict corresponding subjects of an academic paper according to the content of the abstract.

We divide each dataset into training, validation and test sets. The statistics of the two datasets are shown in Table 1.

3.2 Evaluation Metrics

Following the previous work (Zhang and Zhou, 2007; Chen et al., 2017), we adopt hamming loss and micro-F_1 score as our main evaluation metrics. Micro-precision and micro-recall are also reported to assist the analysis.

- **Hamming-loss** (Schapire and Singer, 1999) evaluates the fraction of misclassified instance-label pairs, where a relevant label is missed or an irrelevant is predicted.
- **Micro-F_1** (Manning et al., 2008) can be interpreted as a weighted average of the precision and recall. It is calculated globally by counting the total true positives, false negatives, and false positives.

3.3 Details

We extract the vocabularies from the training sets. For the RCV1-V2 dataset, the size of the vocabulary is 50,000 and out-of-vocabulary (OOV) words are replaced with unk. Each document is truncated at the length of 500 and the beam size is 5 at the inference stage. Besides, we set the word embedding size to 512. The hidden sizes of the encoder and the decoder are 256 and 512, respectively. The number of LSTM layers of encoder and decoder is 2.

For the AAPD dataset, the size of word embedding is 256. There are two LSTM layers in the encoder and its size is 256. For the decoder, there is one LSTM layer of size 512. The size of the vocabulary is 30,000 and OOV words are also replaced with unk. Each document is truncated at the length of 500. The beam size is 9 at the inference stage.

We use the Adam (Kingma and Ba, 2014) optimization method to minimize the cross-entropy loss over the training data. For the hyper-parameters of the Adam optimizer, we set the learning rate $\alpha = 0.001$, two momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ respectively, and $\epsilon = 1 \times 10^{-8}$. Additionally,

[2] http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm

[3] https://github.com/lancopku/SGM

[4] https://arxiv.org/

Models	HL(-)	P(+)	R(+)	F1(+)
BR	0.0086	0.904	0.816	0.858
CC	0.0087	0.887	0.828	0.857
LP	0.0087	0.896	0.824	0.858
CNN	0.0089	**0.922**	0.798	0.855
CNN-RNN	0.0085	0.889	0.825	0.856
SGM	0.0081	0.887	0.850	0.869
+ GE	**0.0075**	0.897	**0.860**	**0.878**

(a) Performance on the RCV1-V2 test set.

Models	HL(-)	P(+)	R(+)	F1(+)
BR	0.0316	0.644	0.648	0.646
CC	0.0306	0.657	0.651	0.654
LP	0.0312	0.662	0.608	0.634
CNN	0.0256	**0.849**	0.545	0.664
CNN-RNN	0.0278	0.718	0.618	0.664
SGM	0.0251	0.746	0.659	0.699
+ GE	**0.0245**	0.748	**0.675**	**0.710**

(b) Performance on the AAPD test set.

Table 2: Comparison between our methods and all baselines on two datasets. GE denotes the global embedding. HL, P, R, and F1 denote hamming loss, micro-precision, micro-recall, and micro-F_1, respectively. The symbol "+" indicates that the higher the value is, the better the model performs. The symbol "-" is the opposite.

we make use of the dropout regularization (Srivastava et al., 2014) to avoid overfitting and clip the gradients (Pascanu et al., 2013) to the maximum norm of 10.0. During training, we train the model for a fixed number of epochs and monitor its performance on the validation set. Once the training is finished, we select the model with the best micro-F_1 score on the validation set as our final model and evaluate its performance on the test set.

3.4 Baselines

We compare our proposed methods with the following baselines:

- **Binary Relevance (BR)** (Boutell et al., 2004) transforms the MLC task into multiple single-label classification problems by ignoring the correlations between labels.
- **Classifier Chains (CC)** (Read et al., 2011) transforms the MLC task into a chain of binary classification problems and takes high-order label correlations into consideration.
- **Label Powerset (LP)** (Tsoumakas and Katakis, 2006) transforms a multi-label problem to a multi-class problem with one multi-class classifier trained on all unique label combinations.
- **CNN** (Kim, 2014) uses multiple convolution kernels to extract text features, which are then inputted to the linear transformation layer followed by a sigmoid function to output the probability distribution over the label space. The multi-label soft margin loss is optimized.
- **CNN-RNN** (Chen et al., 2017) utilizes CNN and RNN to capture both the global and local textual semantics and model the label correlations.

Following the previous work (Chen et al., 2017), we adopt the linear SVM as the base classifier in BR, CC and LP. We implement BR, CC and LP by means of Scikit-Multilearn (Szymański, 2017), an open-source library for the MLC task. We tune hyper-parameters of all baseline algorithms on the validation set based on the micro-F_1 score. In addition, training strategies mentioned in Zhang and Wallace (2015) are used to tune hyper-parameters for the baselines CNN and CNN-RNN.

3.5 Results

For the purpose of simplicity, we denote the proposed sequence generation model as **SGM**. We report the evaluation results of our methods and all baselines on the test sets.

The experimental results of our methods and the baselines on dataset RCV1-V2 are shown in Table 2a. Results show that our proposed methods give the best performance in the main evaluation metrics. Our proposed SGM model using global embedding achieves a reduction of 12.79% hamming-loss and an improvement of 2.33% micro-F_1 score over the most commonly used baseline BR. Besides, our methods outperform other traditional deep-learning models by a large margin. For instance, the proposed SGM model with global embedding achieves a reduction of 15.73% hamming-loss and an improvement

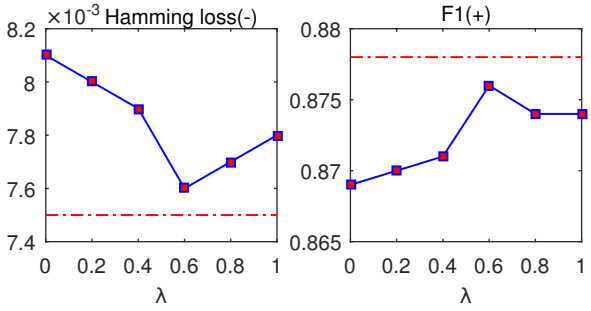


Figure 2: The performance of the SGM model when using different λ. The red dotted line represents the results of using the adaptive gate. The symbol "+" indicates that the higher the value is, the better the model performs. The symbol "-" is the opposite.

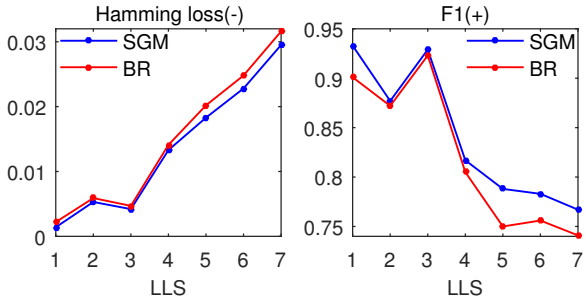


Figure 3: The performance of the SGM model on different subsets of the RCV1-V2 test set. LLS represents the length of label sequence of each sample in the subset. The explanations of symbol "+" and "-" can be found in Figure 2.

of 2.69% micro-F_1 score over the traditional CNN model. Even without the global embedding, our proposed SGM model is still able to outperform all baselines.

In addition, the SGM model is significantly improved by using global embedding. The SGM model with global embedding achieves a reduction of 7.41% hamming loss and an improvement of 1.04% micro-F_1 score on the test set compared with the model without global embedding.

Table 2b presents the results of the proposed methods and the baselines on the AAPD test set. Similar to the experimental results on the RCV1-V2 test set, our proposed methods still outperform all baselines by a large margin in main evaluation metrics. This further confirms that our methods have significant advantages over previous work on large datasets. Besides, the proposed SGM achieves a reduction of 2.39% hamming loss and an improvement of 1.57% micro-F_1 score on the test set by using global embedding. This further testifies that the global embedding is capable of helping the model to predict label sequences more accurately.

3.6 Analysis and Discussion

Here we perform further analysis on the model and experimental results. We report the evaluation results in terms of hamming loss and micro-F_1 score.

3.6.1 Exploration of Global Embedding

As is shown in Table 2, global embedding can significantly improve the performance of the model. The global embedding $g(\boldsymbol{y}_{t-1})$ at time-step t takes advantage of all information of possible labels contained in $\boldsymbol{y}_{t-1}$, so it is able to enrich the source information when the model predicts the current label, which leads to the performance of the model significantly improved. The global embedding is the combination of original embedding $\boldsymbol{e}$ and the weighted average embedding $\bar{\boldsymbol{e}}$ by using the transform gate $\boldsymbol{H}$. Here we conduct experiments on the RCV1-V2 dataset to explore how the performance of our model is affected by the proportion between two kinds of embeddings. In the exploratory experiment, the final embedding vector at time-step t is calculated as follows:

$$g(\boldsymbol{y}_{t-1}) = (1-\lambda) * \boldsymbol{e} + \lambda * \bar{\boldsymbol{e}} \tag{14}$$

The proportion between two kinds of embeddings is controlled by coefficient λ. $\lambda = 0$ denotes the proposed SGM model without global embedding. The proportion of weighted average embedding increases when we increase λ. The experimental results using different λ values in the decoder are shown in Figure 2.

As is shown in Figure 2, the performance of the model varies when different λ is used. Overall, the model using the adaptive gate performs the best, which achieves the best results in both hamming loss and micro-F_1. The models with $\lambda \neq 0$ outperform the model with $\lambda = 0$, which shows that the weighted average embedding contains richer information, leading to the improvement in the performance

Models	HL(-)	F1(+)
SGM	0.0081	0.869
w/o mask	0.0083(↓ 2.47%)	0.866(↓ 0.35%)
w/o sorting	0.0084(↓ 3.70%)	0.858(↓ 1.27%)

(a) Ablation study for the SGM model.

Models	HL(-)	F1(+)
SGM + GE	0.0075	0.878
w/o mask	0.0078(↓ 4.00%)	0.873(↓ 0.57%)
w/o sorting	0.0083(↓ 10.67%)	0.859(↓ 2.16%)

(b) Ablation study for SGM model with global embedding.

Table 3: Ablation study on the RCV1-V2 test set. GE denotes the global embedding. HL and F1 denote hamming loss and micro-F_1, respectively. The symbol "+" indicates that the higher the value is, the better the model performs. The symbol "-" is the opposite. ↑ means that the performance of the model improves and ↓ is the opposite.

of the model. Without using the adaptive gate, the performance of the model improves at first and then deteriorates as λ increases. It reveals the reason why the model with the adaptive gate performs the best: the adaptive gate can automatically determine the most appropriate λ value according to the actual condition.

3.6.2 The Impact of Mask and Sorting

Our proposed methods are developed based on traditional Seq2Seq models. However, the mask module is added to the proposed methods, which is used to prevent the models from predicting repeated labels. In addition, we sort the label sequence of each sample according to the frequency of appearance of labels in the training set. In order to explore the impact of the mask module and sorting, we conduct ablation experiments on the RCV1-V2 dataset. The experimental results are shown in Table 3. "*w/o mask*" means that we do not perform mask operation and "*w/o sorting*" means that we randomly shuffle the label sequence in order to perturb its original order.

As is shown in Table 3, the performance decline of the SGM model with global embedding is more significant compared with that of the SGM model without global embedding. In addition, the decline in the performance of the two models is more significant when we randomly shuffle the label sequence of the sample compared with removing mask module. The label cardinality of the RCV1-V2 dataset is small, so our proposed methods are less prone to predicting repeated labels. This explains the reason why experimental results indicate that the mask module has little impact on the models' performance. In addition, the proposed models are trained using the maximum likelihood estimation method and the cross-entropy loss function, which requires humans to predefine the order of the output labels. Therefore, the sorting of labels is very important for the models' performance. Besides, the performance of both models declines when we do not use the mask module. This shows that the performance of the model can be improved by using the mask operation.

3.6.3 Error Analysis

In the experiment, we find that the performance of all methods deteriorates when the length of the label sequence increases (for simplicity, we denote the length of the label sequence as *LLS*). In order to explore the influence of the value of the *LLS*, we divide the test set into different subsets based on different *LLS*. Figure 3 shows the performance of the SGM model and the most commonly used baseline BR on different subsets of the RCV1-V2 test set. As is shown in Figure 3, generally, the performance of both models deteriorates as the *LLS* increases. This shows that when the label sequence of the sample is particularly long, it is difficult to accurately predict all labels. Because more information is needed when the model predicts more labels. It is easy to ignore some true labels whose feature information is insufficient.

However, as is shown in Figure 3, the proposed SGM model outperforms BR with any value of *LLS*, and the advantages of our model are more significant when *LLS* is large. The traditional BR method predicts all labels at once only based on the sample input. Therefore, it tends to ignore some true labels whose feature information contained in the sample is insufficient. The SGM model generates labels sequentially, and predicts the next label based on its previously predicted labels. Therefore, even if the sample contains less information of some true labels, the SGM model is capable of generating these true labels by considering relevant labels that have been predicted.

• Generating descriptions for videos has many applications including human robot interaction.
• Many methods for image captioning rely on pre-trained object classifier CNN and Long Short Term Memory recurrent networks.
• How to learn robust visual classifiers from the weak annotations of the sentence descriptions.

(a) Visual analysis when the SGM model predicts "*CV*".

• Generating descriptions for videos has many applications including human robot interaction.
• Many methods for image captioning rely on pre-trained object classifier CNN and Long Short Term Memory recurrent networks.
• How to learn robust visual classifiers from the weak annotations of the sentence descriptions.

(b) Visual analysis when the SGM model predicts "*CL*".

Table 4: An example abstract in the AAPD dataset, from which we extract three informative sentences. This abstract is assigned two labels: "*CV*" and "*CL*". They denote computer vision and computational language, respectively.

Reference	**BR**	**SGM**	**SGM + GE**
CCAT, **C15, C152**, C41, C411	CCAT, C15, C13	CCAT, **C15, C152**	CCAT, **C15, C152**, C41, C411
CCAT, GCAT, ECAT, C31, GDIP, C13, C21, **E51, E512**	CCAT, GCAT, GDIP, E51	CCAT, ECAT, GDIP, **E51, E512**	CCAT, GCAT, ECAT, C31, GDIP, **E51, E512**, C312
GCAT, ECAT, **G15, G154, G151, G155**	GCAT, ECAT, GENV, G15	GCAT, ECAT, E21, **G15, G154, G156**	GCAT, ECAT, E21, **G15, G154, G155**

Table 5: Several examples of the generated label sequences on the RCV1-V2 dataset. The red bold labels in each example indicate that they are highly correlated.

3.6.4 Visualization of Attention

When the model predicts different labels, there exist differences in the contributions of different words. The SGM model is able to select the most informative words by utilizing the attention mechanism. The visualization of the attention layer is shown in Table 4. According to Table 4, when the SGM model predicts the label "*CV*", it can automatically assign larger weights to more informative words, like `image`, `visual`, `captioning`, and so on. For the label "*CL*", the selected informative words are `sentence`, `memory`, `recurrent`, etc. This shows that our proposed models are able to consider the differences in the contributions of textual content when predicting different labels and select the most informative words automatically.

3.6.5 Case Study

We give several examples of the generated label sequences on the RCV1-V2 dataset in Table 5, where we compare the proposed methods with the most commonly used baseline BR. The red bold labels in each example indicate that they are highly correlated. For instance, the correlation coefficient between E51 and E512 is 0.7664. Therefore, these highly correlated labels are likely to appear together in the predicted label sequence. The BR algorithm fails to capture this label correlation, leaving many true labels unpredicted. However, our proposed methods accurately predict almost all highly correlated true labels. The proposed SGM captures the correlations between labels by utilizing LSTM to generate labels sequentially. Therefore, for some true labels whose feature information is insufficient, the proposed SGM is still able to generate them by considering relevant labels that have been predicted. In addition, label sequences that are more accurate are predicted by using global embedding. The SGM model with global embedding predicts more true labels compared with the SGM model without global embedding. The reason is that the source information is further enriched by incorporating overall informative signals in the probability distribution $\boldsymbol{y}_{t-1}$ when the model predicts the label at time-step t. Enriched information makes global embedding more smooth, which enables the model to reduce damage caused by mispredictions made in the previous time steps.

4 Related Work

The MLC task studies the problem where multiple labels are assigned to each sample. There are four main types of methods for the MLC task: problem transformation methods, algorithm adaptation methods, ensemble methods, and neural network models.

Problem transformation methods map the MLC task into multiple single-label learning tasks. Binary relevance (BR) (Boutell et al., 2004) decomposes the MLC task into independent binary classification problems by ignoring the correlations between labels. In order to model label correlations, label power-set (LP) (Tsoumakas and Katakis, 2006) transforms a multi-label problem to a multi-class problem with a classifier trained on all unique label combinations. Classifier chains (CC) (Read et al., 2011) transforms the MLC task into a chain of binary classification problems, where subsequent binary classifiers in the chain are built upon the predictions of preceding ones. However, the computational efficiency and performance of these methods are challenged by applications with a large number of labels and samples.

Algorithm adaptation methods extend specific learning algorithms to handle multi-label data directly. Clare and King (2001) construct decision tree based on multi-label entropy to perform classification. Elisseeff and Weston (2002) optimize the empirical ranking loss by using maximum margin strategy and kernel tricks. Collective multi-label classifier (CML) (Ghamrawi and McCallum, 2005) adopts maximum entropy principle to deal with multi-label data by encoding label correlations as constraint conditions. Zhang and Zhou (2007) adopt k-nearest neighbor techniques to deal with multi-label data. Fürnkranz et al. (2008) make ranking among labels by utilizing pairwise comparison. Li et al. (2015) propose a novel joint learning algorithm that allows the feedbacks to be propagated from the classifiers for latter labels to the classifier for the current label. Most methods, however, can only be used to capture the first or second order label correlations or are computationally intractable in considering high-order label correlations.

Among ensemble methods, Tsoumakas et al. (2011) break the initial set of labels into a number of small random subsets and employ the LP algorithm to train a corresponding classifier. Szymański et al. (2016) propose to construct a label co-occurrence graph and perform community detection to partition the label set.

In recent years, some neural network models have also been used for the MLC task. Zhang and Zhou (2006) propose the BP-MLL that utilizes a fully-connected neural network and a pairwise ranking loss function. Nam et al. (2013) propose a neural network using cross-entropy loss instead of ranking loss. Benites and Sapozhnikova (2015) increase classification speed by adding an extra ART layer for clustering. Kurata et al. (2016) utilize word embeddings based on CNN to capture label correlations. Chen et al. (2017) propose to represent semantic information of text and model high-order label correlations by combining CNN with RNN. Baker and Korhonen (2017) initialize the final hidden layer with rows that map to co-occurrence of labels based on the CNN architecture to improve the performance of the model. Ma et al. (2018) propose to use the multi-label classification algorithm for machine translation to handle the situation where a sentence can be translated into more than one correct sentences.

5 Conclusions and Future Work

In this paper, we propose to view the multi-label classification task as a sequence generation problem to model the correlations between labels. A sequence generation model with a novel decoder structure is proposed to improve the performance of classification. Extensive experimental results show that the proposed methods outperform the baselines by a substantial margin. Further analysis of experimental results demonstrates that our proposed methods not only capture the correlations between labels, but also select the most informative words automatically when predicting different labels.

As analyzed in Section 3.6.3, when a large number of labels are assigned to a sample, how to predict all these true labels accurately is an intractable problem. Our proposed methods alleviate this problem to some extent, but more effective solutions need to be further explored in the future.

6 Acknowledgements

This work is supported in part by National Natural Science Foundation of China (No. 61673028, No. 61333018) and the National Thousand Young Talents Program. Xu Sun is the corresponding author of this paper.

References

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

Simon Baker and Anna Korhonen. 2017. Initializing neural networks for hierarchical multi-label text classification. In *BioNLP*.

Fernando Benites and Elena Sapozhnikova. 2015. Haram: a hierarchical aram neural network for large-scale text classification. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*, pages 847–854. IEEE.

Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. 2004. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771.

Guibin Chen, Deheng Ye, Zhenchang Xing, Jieshan Chen, and Erik Cambria. 2017. Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, pages 2377–2383.

Amanda Clare and Ross D King. 2001. Knowledge discovery in multi-label phenotype data. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 42–53. Springer.

André Elisseeff and Jason Weston. 2002. A kernel method for multi-labelled classification. In *Advances in neural information processing systems*, pages 681–687.

Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. 2008. Multilabel classification via calibrated label ranking. *Machine learning*, 73(2):133–153.

Nadia Ghamrawi and Andrew McCallum. 2005. Collective multi-label classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 195–200. ACM.

Siddharth Gopal and Yiming Yang. 2010. Multilabel classification with meta-level features. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 315–322. ACM.

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

Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2008. Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD*, volume 18.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

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

Gakuto Kurata, Bing Xiang, and Bowen Zhou. 2016. Improved neural network-based multi-label classification with better initialization leveraging label co-occurrence. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 521–526.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

Li Li, Houfeng Wang, Xu Sun, Baobao Chang, Shi Zhao, and Lei Sha. 2015. Multi-label text categorization with joint learning predictions-as-features method. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 835–839.

Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. 2018. Global encoding for abstractive summarization. In *ACL 2018*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025.

Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. 2018. Bag-of-words as target for neural machine translation. In *ACL 2018*.

Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

Jinseok Nam, Jungi Kim, Iryna Gurevych, and Johannes Fürnkranz. 2013. Large-scale multi-label text classification - revisiting neural networks. *CoRR*, abs/1312.5419.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Machine learning*, 85(3):333.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 379–389.

Robert E Schapire and Yoram Singer. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336.

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

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. *CoRR*, abs/1705.09655.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR*, abs/1505.00387.

Xu Sun, Bingzhen Wei, Xuancheng Ren, and Shuming Ma. 2017. Label embedding network: Learning label representation for soft training of deep networks. *CoRR*, abs/1710.10393.

Piotr Szymański, Tomasz Kajdanowicz, and Kristian Kersting. 2016. How is a data-driven approach better than random choice in label space division for multi-label classification? *Entropy*, 18(8):282.

Piotr Szymański. 2017. A scikit-based python environment for performing multi-label classification. *arXiv preprint arXiv:1702.01460*.

Grigorios Tsoumakas and Ioannis Katakis. 2006. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3).

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2011. Random k-labelsets for multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1079–1089.

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

Jingjing Xu, Xu Sun, Qi Zeng, Xuancheng Ren, Xiaodong Zhang, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. *ACL 2018*.

Ye Zhang and Byron C. Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, abs/1510.03820.

Min-Ling Zhang and Zhi-Hua Zhou. 2006. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351.

Min-Ling Zhang and Zhi-Hua Zhou. 2007. ML-KNN: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048.

Author Index